PolyPublie
Polytechnique Montréal

POLYTECHNIQUE MONTRÉAL
UNIVERSITÉ D'INGÉNIERIE

Titre: Title:	Commande par supervision de systèmes mécatroniques via internet
Auteur: Author:	Mathieu Lozeau
Date:	2009
Type:	Mémoire ou thèse / Dissertation or Thesis
Référence: Citation:	Lozeau, M. (2009). Commande par supervision de systèmes mécatroniques via internet [Master's thesis, École Polytechnique de Montréal]. PolyPublie. https://publications.polymtl.ca/174/

Document en libre accès dans PolyPublie
Open Access document in PolyPublie

URL de PolyPublie: PolyPublie URL:	https://publications.polymtl.ca/174/
Directeurs de recherche: Advisors:	El-Kébir Boukas
Programme: Program:	Génie mécanique

Ce fichier a été téléchargé à partir de PolyPublie, le dépôt institutionnel de Polytechnique Montréal
This file has been downloaded from PolyPublie, the institutional repository of Polytechnique Montréal

https://publications.polymtl.ca

UNIVERSITÉ DE MONTRÉAL

COMMANDE PAR SUPERVISION DE SYSTÈMES MÉCATRONIQUES VIA INTERNET

MATHIEU LOZEAU
DÉPARTEMENT DE GÉNIE MÉCANIQUE
ÉCOLE POLYTECHNIQUE DE MONTRÉAL

MÉMOIRE PRÉSENTÉ EN VUE DE L'OBTENTION
DU DIPLÔME DE MAÎTRISE ÈS SCIENCES APPLIQUÉES
SEPTEMBRE 2009

© Mathieu Lozeau, 2009.

UNIVERSITÉ DE MONTRÉAL

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

Ce mémoire intitulé:

COMMANDE PAR SUPERVISION DE SYSTÈMES MÉCATRONIQUES VIA INTERNET

présenté par: LOZEAU Mathieu
en vue de l'obtention du diplôme de: Maîtrise ès sciences appliquées
a été dûment accepté par le jury d'examen constitué de:

M. BARON Luc, Ph.D., président
M. BOUKAS El-Kébir, Ph.D., membre et directeur de recherche
M. SAYDY Lahcen, Ph.D., membre

À ma mère.

REMERCIEMENTS

Premièrement, j'aimerais remercier le professeur El-Kébir Boukas, mon directeur de recherche, avec qui j'ai eu la chance de travailler ces dernières années. Je le remercie tout particulièrement pour sa patience, son enthousiasme, l'intérêt qu'il a démontré tout au long de mon projet, ainsi que pour la compréhension et la persévérance dont il a toujours fait preuve à mon égard.

Je remercie les techniciens Jean-Marie Béland et Roch Brassard ainsi que le coordonnateur des ressources techniques au département de génie mécanique, M. François Morin, pour leur travail de qualité durant la fabrication du robot équilibriste conçu au cours de ce projet de recherche, le technicien Nour Aimene qui s'est toujours montré disponible pour me venir en aide lors de l'élaboration de montages électroniques, le technicien Jacques Girardin du département de génie électrique pour les composantes électroniques qu'il a été en mesure de donner au laboratoire, l'analyste Omar Toukal et la technicienne Tanya Alleyne du service informatique pour leur support technique en ce qui concerne le réseau informatique de l'école et les ordinateurs du laboratoire et le professeur Romano De Santis pour m'avoir donné l'opportunité de présenter mon travail de recherche dans le cadre de son cours sur la commande des systèmes non-linéaires.

Je remercie aussi mes collègues Cédric Akowanou et Hassan Bensalah au laboratoire de mécatronique pour leur soutient et pour m'avoir permis de partager mon intérêt marqué pour la mécatronique en discutant avec eux et pour leurs questions qui me poussent à en connaître toujours davantage, les étudiants Roneol Kamdem et Japhet Honvo du club *SmartBird* avec qui il me fait toujours plaisir de discuter et de partager mes idées sur la commande des systèmes en temps réel pour leur projet de commande automatique d'un avion autonome miniature, et particulièrement Vincent Bouverat qui a participé à la conception mécanique du robot équilibriste lors de son projet de fin d'études.

Je dois également remercier mes parents pour leur soutient tout au long de ce projet et pour leur aide particulière avec la correction du document, mon ami Ahmad Haidar pour

son amitié, son écoute et son soutient constant et mon ami Justin Knight pour son amitié de longue durée et ses appels téléphoniques surprises au laboratoire qui sont toujours très motivants.

RÉSUMÉ

Le projet de recherche présenté dans ce document traite de la commande par supervision de systèmes mécatroniques via Internet. Son objectif est de proposer, de concevoir et d'expérimenter une architecture d'application web basée sur des standards ouverts permettant ce type de commande en étudiant un exemple concret. L'approche proposée est basée sur une application web dynamique faisant appel à la technologie *Java 2.0 Enterprise Edition* (J2EE), une page web dynamique utilisant des Javascript et le protocole de communication à haut niveau *Asynchronous JavaScript and XML* (AJAX). De plus, afin de pouvoir commander à distance des montages mécatroniques, une architecture de communication a été mise en place. Celle-ci inclut des communications inter-puces réalisées par le protocole *Serial Peripheral Interface* (SPI), des communications sans fil réalisées par une solution propriétaire faisant appel à la technique de modulation *Gaussian Frequency-Shift Keying* (GFSK), des communications entre un système embarqué et un ordinateur PC réalisées par le *Universal Serial Bus* (USB), et des communications entre un ordinateur PC agissant en tant que serveur web et d'autres ordinateurs agissant comme clients réalisées par *Transmission Control Protocol / Internet Protocol* (TCP/IP). Afin d'expérimenter l'architecture proposée, un système mécatronique consistant en un pendule inversé mobile à deux roues a été conçu et un problème de poursuite de trajectoire générée à distance a été considéré. La conception mécanique de ce robot est constituée d'un corps avec une tige et de deux roues, chacune étant entraînée par un moteur à courant continu muni d'une boîte de réduction. Le corps et les roues ont été conçus en utilisant un logiciel de CAO et ont été fabriqués et assemblés à l'École Polytechnique de Montréal. Le système électronique du robot est composé d'un microcontrôleur, de deux encodeurs optiques utilisés pour la mesure des positions des roues, d'un accéléromètre à deux axes et d'un gyroscope utilisés pour mesurer l'angle d'inclinaison du robot, de deux puces de ponts en H utilisées pour la commande des moteurs à courant continu, d'un module d'affichage à cristaux liquides (*Liquid Crystal Display - LCD*), d'une puce

de communication sans fil pour la communication avec un ordinateur, d'un ensemble piles et de deux régulateurs de tension pour l'alimentation en puissance.

Le robot équilibriste conçu présente plusieurs défis dont le fait que celui-ci est un système instable et requiert donc un correcteur pour le stabiliser en le maintenant en équilibre. Afin de pouvoir concevoir ce contrôleur, la dynamique du robot a d'abord été analysée et un modèle mathématique de celle-ci a été développé sous la forme de modèles d'états linéaires. Des algorithmes de filtrage permettant d'obtenir de bons estimés des états malgré la présence de bruits dans les signaux des capteurs ont également été mis en place. Des lois de commande permettant le contrôle de l'inclinaison, du déplacement linéaire et de l'angle d'orientation du robot on été synthétisées en faisant appel à la technique de placement de pôles dans un premier temps puis à la technique H_∞ visant à minimiser la sensibilité du système aux perturbations externes dans un deuxième temps. Finalement, afin d'assurer que la trajectoire réellement empruntée par le robot suive de près la trajectoire voulue, un contrôleur de poursuite de trajectoire a été mis en place. Les algorithmes de filtrage et de commande on été implémentés numériquement dans un logiciel temps réel exécuté par le microcontrôleur du robot.

Une interface contenue dans une page web permettant à un opérateur de générer une trajectoire à suivre par le robot a été mise en place. Elle inclut une simulation visuelle réalisée par un Javascript dans la page web dans laquelle un modèle virtuel du robot permet d'anticiper le comportement réel du robot, et où il est possible de visualiser l'évolution de la trajectoire voulue ainsi que celle réellement empruntée par le robot.

Des expérimentations ont été effectuées afin d'évaluer la validité du modèle mathématique, la performance des contrôleurs du robot et le fonctionnement du système de commande à distance. Ces expérimentations démontrent la validité du modèle développé, la capacité du contrôleur conçu par la méthode H_∞ à rejeter des perturbations externes, la capacité de la simulation visuelle à anticiper le comportement du robot malgré la présence de délais de communication sur le réseau et le bon fonctionnement de l'algorithme de poursuite de trajectoire.

ABSTRACT

The research project presented in this document deals with the supervisory control of mechatronic systems via the Internet. Its objective is to propose, design and test a web application architecture based on open standards allowing this type of control by studying a concrete example. The proposed approach is based on a dynamic web application using *Java 2.0 Enterprise Edition* (J2EE), a dynamic web page using Javascript and the high-level communication protocol *Asynchronous JavaScript and XML* (AJAX). In addition, in order to be able to control mechatronic systems remotely, a communication architecture has been implemented. It includes inter-chip communications performed by the *Serial Peripheral Interface* (SPI), wireless communications conducted by a proprietary solution using the *Gaussian Frequency-Shift Keying* (GFSK) modulation method, communications between an electronic board and a PC realized by *Universal Serial Bus* (USB), and communications between a PC acting as a web server and other computers acting as clients with *Transmission Control Protocol / Internet Protocol* (TCP / IP).

To experiment with the proposed architecture, a mechatronic system consisting of a two-wheeled mobile inverted pendulum robot was developed and a remotely generated trajectory tracking problem has been considered. The mechanical design of this robot consists of a body with a shaft and two wheels, each driven by a DC motor through a gear box. The body and wheels were designed using CAD software and were manufactured and assembled at École Polytechnique de Montréal. The electronic system of the robot consists of a microcontroller, two optical encoders used for measuring the positions of the wheels, a two-axis accelerometer and a gyroscope used to measure the tilt angle of the robot, two H-bridge chips used for controlling the DC motors, a liquid crystal display module, a chip for wireless communication with a computer, a set of batteries and two voltage regulators used as a power supply.

The balancing robot designed presents several challenges including the fact that it is an unstable system and therefore requires a controller to stabilize it and maintain its ba-

lance. In order to design this controller, the dynamics of the robot has been analyzed and a mathematical model was developed in the form of linear state-space models. Filtering algorithms to obtain good estimates of the states despite the presence of noise in the sensors' signals were also implemented. Control laws to control the tilt angle, the linear displacement and the heading angle of the robot were synthesized using the pole placement technique in the first place and then using the H_∞ method to minimize the system's sensitivity to external disturbances. Finally, to ensure that the path actually taken by the robot follows closely the desired trajectory, a path-tracking controller has been designed. The algorithms for filtering and control have been implemented numerically in a real-time software executed by the microcontroller of the robot.

A user interface in a web page to allow an operator to generate a trajectory that the robot must follow has been created. It includes a visual simulation carried out by a JavaScript in the web page in which a virtual model of the robot anticipates the behavior of the real robot, and where it is possible to see the evolution of the desired trajectory as well as the actual path taken by the robot.

Experiments were conducted to assess the validity of the mathematical model, the performance of the controllers and the operation of the remote control system. These experiments demonstrate the validity of the model developed, the ability of the controller designed by the H_∞ method to reject external disturbances, the ability of the visual simulation to anticipate the behavior of the robot despite the presence of communication delays on the network and the proper functioning of the path-tracking algorithm.

TABLE DES MATIÈRES

DÉDICACE iii

REMERCIEMENTS iv

RÉSUMÉ vi

ABSTRACT viii

TABLE DES MATIÈRES x

LISTE DES TABLEAUX xv

LISTE DES FIGURES xvi

LISTE DES NOTATIONS ET DES SYMBOLES xviii

INTRODUCTION 1

CHAPITRE 1 DÉFINITION DU PROBLÈME 8

1.1 Robot équilibriste 11

1.1.1 Revue de la littérature 12

1.1.2 Conception 16

1.1.3 Modélisation Mathématique 16

1.1.4 Commande en temps réel 16

1.2 Commande via Internet 17

1.2.1 Architecture de communication 18

1.2.2 Application web 18

CHAPITRE 2 CONCEPTION DU ROBOT ÉQUILIBRISTE 19

2.1 Mécanique 19

2.1.1 Actionneurs 21
2.2 Électronique 21
2.2.1 Alimentation 22
2.2.2 Capteurs 24
2.2.2.1 Gyroscope 25
2.2.2.2 Accéléromètre 26
2.2.2.3 Encodeurs Optiques 28
2.2.3 Afficheur à cristaux liquides (LCD) 29
2.2.4 Interface de puissance 30
2.2.5 Communication sans fil 30
2.2.6 Microcontrôleur 31
2.2.6.1 Interruption externe 32
2.2.6.2 Interruption sur changements (CN) 33
2.2.6.3 Temporisateurs 35
2.2.6.4 Convertisseur analogique numérique (ADC) 35
2.2.6.5 Modulation de largeur d'impulsion (PWM) 36
2.2.6.6 Interface de périphériques séquentielle (SPI) 39

CHAPITRE 3 MODÉLISATION MATHÉMATIQUE DU ROBOT 42
3.1 Inclinaison et déplacement linéaire 44
3.1.1 Dynamique d'un moteur 45
3.1.2 Couple appliqué à une roue 46
3.1.3 Dynamique d'inclinaison 48
3.1.4 Dynamique des roues et du déplacement linéaire 49
3.1.5 Modèle d'état de la dynamique linéaire 51
3.2 Angle de direction 52
3.2.1 Modèle d'état de la dynamique de direction 58
3.3 Perturbations externes 59

3.3.1 Dynamique d'inclinaison et de déplacement linéaire 60
3.3.2 Dynamique de direction 62

CHAPITRE 4 COMMANDE EN TEMPS RÉEL 63
4.1 Filtrage 64
4.1.1 Angle d'inclinaison 65
4.1.1.1 Estimé du gyroscope 66
4.1.1.2 Estimé de l'accéléromètre 66
4.1.1.3 Filtres complémentaires 67
4.1.2 Filtres pour positions des roues 72
4.2 Correcteur 73
4.2.1 Références et intégrateurs 73
4.2.2 Structure du correcteur 77
4.2.3 Placement de pôles 78
4.2.3.1 Dynamique linéaire 79
4.2.3.2 Dynamique de direction 80
4.2.4 $\mathbf{H}_\infty$ 81
4.2.5 Poursuite de trajectoire 86
4.3 Implémentation numérique 93
4.3.1 Discrétisation 93
4.3.1.1 Fréquence d'échantillonnage 93
4.3.1.2 Filtre du gyroscope 95
4.3.1.3 Filtre de l'accéléromètre 96
4.3.1.4 Variables de poursuite de trajectoire 97
4.3.2 Système tempsréel 97
4.3.2.1 Tâches et priorités 98
4.3.2.2 Ressources partagées 100

CHAPITRE 5 COMMANDE VIA INTERNET 102
5.1 Architecture de communication 102
5.1.1 Noeuds 102
5.1.1.1 Robot 103
5.1.1.2 Gestionnaire de communication 103
5.1.1.3 Serveur web 104
5.1.1.4 Client PC 104
5.1.2 Protocoles 105
5.1.2.1 SPI 105
5.1.2.2 RF 105
5.1.2.3 USB 106
5.1.2.4 TCP/IP et HTTP 106
5.2 Application Web 106
5.2.1 Page Web 108
5.2.1.1 Interface graphique 108
5.2.1.2 Caméra web 109
5.2.1.3 Génération de trajectoire et simulation visuelle . . . 112
5.2.1.4 Javascript 118
5.2.1.5 Ajax 122
5.2.2 Serveur web J2EE 123
5.2.2.1 Page opérateur 126
5.2.2.2 Service Ajax 128
5.2.2.3 Interface de communication et Java Bean 129

CHAPITRE 6 EXPÉRIMENTATIONS 130
6.1 Validation du modèle 130
6.2 Comparaison des correcteurs 131
6.2.1 Réponse temporelle 131

6.2.2 Sensibilité aux perturbations 133
6.3 Commande par supervision 134

CONCLUSION 141

RÉFÉRENCES 144

LISTE DES TABLEAUX

2.1	Composantes du système	23
2.2	Table de vérité d'un pont en H	30
3.1	Paramètres du système	43
4.1	Variables de poursuite de trajectoire	86
5.1	Aperçu du code HTML de la page web	110

LISTE DES FIGURES

2.1	Structure mécanique du robot équilibriste	20
2.2	Architecture du système numérique	22
2.3	Circuit d'alimentation	24
2.4	Gyroscope **IDG-300**	25
2.5	Accéléromètre **LIS3LV02DQ**	27
2.6	Signaux d'un encodeur optique	28
2.7	Module d'afficheur à cristaux liquides **S01602 D/C**	29
2.8	Signal PWM	38
2.9	Connections sur un bus SPI	40
3.1	Inclinaison et déplacement linéaire du robot	45
3.2	Diagramme de corps libre de la dynamique d'inclinaison	48
3.3	Diagramme de corps libre d'une roue	49
3.4	Déplacements des roues lors d'un changement de direction	53
3.5	Diagramme de corps libre de la dynamique d'angle de direction	56
4.1	Traitement de signaux et estimation des états	65
4.2	Accéléromètre utilisé en tant qu'inclinomètre	66
4.3	Diagramme de Bode du filtre passe-bas $G_a(s)$	69
4.4	Diagramme de Bode du filtre passe-haut $G_g(s)$	70
4.5	Diagrammes de Bode des filtres complémentaires $G_a(s)$ et $G_g(s)$	71
4.6	Structure globale du correcteur par retour d'état	78
4.7	Emplacements des pôles en boucle fermée par placement de pôles	82
4.8	Emplacements des pôles en boucle fermée par $\mathbf{H}_\infty$	86
4.9	Poursuite de trajectoire	87
5.1	Architecture de communication	103
5.2	Page web	111
5.3	Interaction entre le client PC et le serveur WEB	127

5.4 Hiérarchie d'objets de l'application web 127
6.1 Simulation v.s. temps réel 136
6.2 Comparaison des deux contrôleurs 137
6.3 Rejet de perturbation placement de pôles 138
6.4 Rejet de perturbation H_∞ 139
6.5 Poursuite de trajectoire via Internet 140

LISTE DES NOTATIONS ET DES SYMBOLES

$\delta(t)$: angle de direction

η : efficacité de la boîte de réduction

$\theta(t)$: angle d'une roue

$\theta_i(t)$: angle de l'arbre moteur

$\theta_l(t)$: angle de la roue gauche

$\theta_o(t)$: angle de l'arbre de la boîte de réduction

$\theta_r(t)$: angle de la roue droite

$\omega_x(t)$: perturbation externe linéaire

$\omega_h(t)$: perturbation externe de direction

$\psi(t)$: angle d'inclinaison

ζ : facteur d'amortissement

C_F : constante de friction

d : distance entre l'arbre du moteur et le centre de gravité

$F(t)$: force appliqué au sol par une roue

$F_l(t)$: force appliqué au sol par la roue gauche

$F_r(t)$: force appliqué au sol par la roue droite

J_b : moment d'inertie de la moitié du corps du robot

J_d : moment d'inertie du robot autour de l'axe vertical

J_w : moment d'inertie de l'une des roues

K_e : constante de force électromotrice des moteurs

K_t : constante de couple des moteurs

M : masse de la moitié robot, y compris une roue

m_b : masse de la moitié du corps du robot

m_w : masse de l'une des roues

$T(t)$: couple livré à une roue

$T_i(t)$: couple livré à la boîte de réduction par un moteur à courant continu

$T_l(t)$: couple livré à la roue gauche

$T_r(t)$: couple livré à la roue droite

r_a : résistance d'armature des moteurs

r_g : rapport de la boîte de réduction

r_w : rayon des roues

S : distance entre les roues

$u_h(t)$: portion de la tension contrôlant la direction

$u_l(t)$: tension envoyée au moteur gauche

$u_r(t)$: tension envoyée au moteur droit

$u_x(t)$: tension moyenne envoyée aux moteurs

$x(t)$: position du robot

$x_l(t)$: position de la roue gauche

$x_r(t)$: position de la roue droite

LISTE DES ACRONYMES

ADC	Analog to Digital Converter
AJAX	Asynchronous JavaScript and XML
CGI	Common Gateway Interface
CN	Change Notification
DLL	Dynamic Link Library
EJB	Eterprise Java Beans
FIFO	First In First Out
GFSK	Gaussian Frequency-Shift Keying
HTML	Hypertext Markup Langage

HTTP	Hypertext Transfer Protocol
J2EE	Java 2.0 Enterprise Edition
JPEG	Joint Photographic Experts Group
LCD	Liquid Crystal Display
LMI	Linear Matrix Inequality
LQR	Linear Quadratic Regulator
MEMS	Micro-Electro-Mechanical Systems
MISO	Master In / Slave Out
MOSI	Master Out / Slave In
NCS	Networked Control System
PWM	Pulse Width Modulation
RTOS	Real Time Operating System
SPI	Serial Peripheral Interface
SSL	Secure Socket Layer
SVG	Scalable Vector Graphics
TCP	Transfert Control Protocol
TCP/IP	Transmission Control Protocol / Internet Protocol
T-WIP	Two-Wheeled Inverted Pendulum
UDP	User Datagram Protocol
USART	Universal Synchronous & Asynchronous Receiver Transmitter
USB	Universal Serial Bus
UWA	University of Western Australia
W3C	World Wide Web Consortium
XML	Extensible Markup Language

INTRODUCTION

Les systèmes de commande sont présents partout autour de nous. Ceux-ci sont utilisés aussi bien dans les produits commerciaux que dans les applications industrielles et militaires. Traditionnellement, ces systèmes étaient analogiques et leurs différentes composantes telles que des capteurs, actionneurs et circuits de commande, étaient inter-reliées par des connexions physiques sur lesquelles des signaux analogiques étaient transmis. De nos jours, avec l'évolution des ordinateurs et des technologies de l'information, la plupart des systèmes de commande sont conçus en tant que systèmes numériques à base de microprocesseur. Ceci permet donc à leurs différentes composantes d'être reliées par des connexions logiques sur un réseau de communication. On parle alors de système de commande par réseau (*Networked Control System* - NCS).

Le domaine de la commande par réseau est large. Celui-ci comprend principalement la commande distribuée et la commande par supervision ou téléopération. La commande distribuée adresse les systèmes dans lesquels des noeuds de capteurs, d'actionneurs et de contrôleurs sont géographiquement distribués et en liaison à travers un réseau de communication. Dans ce domaine, un des enjeux significatifs est de pouvoir prendre en considération les effets que le réseau de communication peut avoir sur la performance et la stabilité du système commandé. Ces effets incluent les délais variables de transmission et la perte de paquets d'information lors de ces transmissions. Par ailleurs, la commande par supervision traite des systèmes dans lesquels la commande en boucle fermée est effectuée par un contrôleur local et pour lesquels un opérateur est en mesure d'interagir avec le système à distance en lui envoyant des consignes et en surveillant son évolution par l'intermédiaire d'un réseau de communication. Dans ces systèmes, les effets dûs aux délais de transmission et aux pertes de données sont moindres mais non négligeables. Pour une revue complète des systèmes de commande par réseau, le lecteur est référé à (Gupta & Chow, 2008).

Le projet de recherche présenté dans ce document se situe dans le cadre des systèmes de commande par supervision. Ce type de système est généralement utilisé dans des applications où l'opération d'un équipement ou la réalisation d'une tâche demande une expertise spécialisée. On pense par exemple à des applications médicales comme des systèmes de chirurgie à distance, ou encore des applications où la tâche à réaliser doit être effectuée dans un environnement dangereux pour un ouvrier comme l'exploration minière. On retrouve également des applications industrielles dans lesquelles des machines de production ne peuvent être complètement automatisées puisqu'elles doivent remplir des fonctionnalités diverses et un opérateur spécialisé doit intervenir régulièrement pour assurer un déroulement efficace de l'opération. La commande par supervision via Internet est également utilisée dans le cadre de l'enseignement de la mécatronique et des systèmes de commande. C'est dans ce contexte que le projet présenté ici a vu le jour. L'idée initiale du projet était de concevoir un système permettant la commande de plusieurs montages mécatroniques à distance à travers le réseau Internet. Ce système pourrait par la suite être utilisé pour mettre en place un laboratoire virtuel dans le cadre des cours de mécatronique.

Un projet de référence dans le domaine de la robotique par téléopération via Internet est celui de *University of Western Australia* (UWA). Le robot télé-opéré de l'UWA a constitué un jalon historique pour l'Internet. Il a été le premier robot industriel mis à disposition pour un usage général sur Internet en 1994. Ce robot a été initialement développé dans le cadre d'une thèse de doctorat par le Dr. Kenneth Taylor (Taylor, K., 1999) et a été l'objet d'un autre doctorat par le Dr. Barney Dalton (Dalton, B., 2001). Le robot de l'UWA est toujours en ligne aujourd'hui, bien que le robot original ait été remplacé en 1996 et qu'il ne soit plus disponible au grand public sans autorisation. Le robot actuel de l'UWA est un modèle de série IRB1400 ABB. Le robot est commandé par un contrôleur ABB S4 relié à un serveur Linux et qui, à son tour communique avec un second serveur exécutant le logiciel RobComm d'ABB et une application Labview de National

Instruments qui a été écrite sur mesure pour cette tâche. La structure de contrôle de ce robot a subi de nombreux changements depuis sa conception initiale. Originalement, il était contrôlé via des pages web statiques avec un *Common Gateway Interface* (CGI). Les travaux de Dalton ont par la suite permis la mise en place d'une interface de réalité virtuelle basée sur Java. La solution finale de l'architecture de contrôle de ce robot est une application client LabVIEW qui intègre un vidéo en temps réel.

À titre d'exemple de laboratoire virtuel, (Lassó & Urbancsek, 2001) présentent un système robotique composé d'un microrobot muni d'un microscope capable de se déplacer dans son environnement alors qu'une caméra surveille l'environnement global et qu'une autre permet d'obtenir l'image du microscope. Ce travail utilise une approche basée sur les appliquettes Java (*Java Applets*) et propose une combinaison des protocoles *Transfert Control Protocol* (TCP) et *User Datagram Protocol* (UDP) pour les différents aspects de son application. On y propose également une technique d'encodage d'images à multiples résolutions afin de réduire le débit requis pour les transmettre.

Comme exemple de système de commande par supervision via Internet utilisé dans le cadre de l'enseignement de la mécatronique, on peut citer (Lindsay, Good & Halgamuge, 2000) où on présente un système permettant la commande d'un robot industriel CRS F3 à partir d'une appliquette Java intégrée dans une page web. Ce système a vu le jour à l'Université de Melbourne dans le cadre de la mise à jour ses pratiques d'enseignement de la mécatronique suite aux avancements des technologies de l'information et de l'Internet. L'approche utilisée est comparable à celle d'UWA et consiste en une interface de commande via Internet reliée à un système de commande local propre au robot industriel utilisé. Une interface utilisateur implémentée dans une appliquette Java permet d'envoyer des commandes dans un langage à haut niveau, de récupérer les messages du système en réponse à ces commandes et de visualiser l'état du système à la fois par l'intermédiaire de valeurs numériques et également à l'aide d'une image provenant d'une caméra. Les commandes entrées par l'utilisateur sont validées localement par l'appli-

quette avant d'être envoyées au serveur exécutant une application Java. Cette dernière est ensuite responsable d'interpréter les commandes et de communiquer avec le système de commande du robot via une connexion série.

Un autre exemple de système développé dans le cadre de l'enseignement de la mécatronique est présenté par (Roesh, Roth & Nicolescu, 2005). Les auteurs présentent un système de commande d'un pendule démontrant le principe de la commande de structures mécaniques flexibles. Le contrôleur temps réel du système est implémenté localement sur un PC à l'aide des logiciels Matlab, Simulink et WinCon alors qu'une interface de programmation composée d'un ensemble de *Dynamic Link Library* (DLL) permet l'intégration du logiciel Matlab et d'une application serveur en Java. L'interface utilisateur est implémentée en tant qu'appliquette Java exécutée dans une page web et communique avec l'application serveur via Internet.

Dans l'article de (Han, Kim, S., Kim, Y.J. & Kim, J.H., 2001), les auteurs proposent un système permettant la poursuite de trajectoire d'un robot mobile commandé par Internet. Leur approche inclut la considération des délais variables lors de la transmission de données via Internet et ils proposent une méthode pour gérer ce phénomène. Leurs travaux considèrent un modèle virtuel du système dans l'application client et le vrai système commandé. Leur approche tente de minimiser la différence entre l'état du système virtuel et celui du système réel puis l'écart de temps entre ces deux systèmes. Pour réduire l'écart entre l'état du système virtuel et celui du système réel, l'état du modèle virtuel du robot est estimé par un observateur basé sur un modèle de la cinématique de leur robot utilisant la position réelle du robot. Afin de réduire l'écart de temps, ils utilisent un type de filtre par lequel les signaux de consigne sont reconstruits avant d'être envoyés au système. Ce filtre achemine simplement les échantillons en respectant l'ordre d'arrivée et en introduisant un délai minimal égal à la période d'échantillonnage utilisée par le système de commande local entre chaque échantillon. Par contre, les auteurs n'abordent pas le problème d'inversion de paquets inhérent au problème de délais variables.

Un exemple d'application industrielle dans le domaine de la fabrication assistée par ordinateur est présenté par (Wang, Xi & Zhang, 2005). Les chercheurs ont utilisé une solution basée sur les appliquettes Java dans laquelle un modèle tridimensionnel est réalisé à l'aide de la librairie *Java 3D* afin de permettre de commander à distance un manipulateur robotique. Leur approche inclut un modèle virtuel pour la manipulation du système à distance et un autre pour la supervision. Il est alors possible de comparer l'état voulu du système à son état actuel.

Le but principal de ce projet de recherche est de proposer et d'expérimenter une architecture d'application web permettant la commande à distance par supervision de systèmes mécatroniques via Internet basée sur des standards ouverts. L'approche proposée est basée sur une application web dynamique faisant appel à la technologie *Java 2.0 Enterprise Edition* (J2EE), une page web dynamique utilisant des Javascript et le protocole de communication à haut niveau *Asynchronous JavaScript and XML* (AJAX). J2EE est un standard bien établi dans l'industrie pour les applications web dynamiques. Puisqu'il s'agit d'une solution accessible au grand public, en plus de réduire les coûts puisqu'elle est gratuite, celle-ci est très performante puisqu'elle a su profiter d'une grande communauté de développeurs qui ont contribué à son perfectionnement. Aussi, puisqu'il s'agit d'un standard ouvert, une application qui est basée sur ce standard pourra facilement être intégrée aux architectures déjà en place en entreprise. Le fait que cette technologie est largement utilisée facilite également le développement continuel d'une telle application ainsi que son opération et son support. Alors que les applications basées sur des logiciels commerciaux tel que Matlab et Labview demandent un investissement considérable dans de telles technologies et représentent une charge de traitement accrue pour le serveur, l'approche proposée ici est basée sur une architecture web optimisée pour répondre aux requêtes d'utilisateurs de façon efficace. En outre, Javascript est un langage spécialement conçu pour être exécuté par les logiciels de navigation ; il est donc intéressant de considérer la possibilité d'implémenter l'interface utilisateur dans une page

web avec ce langage plutôt que de développer une appliquette Java qui introduirait une charge de traitement supplémentaire et une source additionnelle de problèmes potentiels.

Dans ce projet de recherche, le système mécatronique mis en place afin d'expérimenter l'application web permettant la commande par supervision développée consiste en un robot équilibriste mobile à deux roues. Comme dans (Han, Kim, S., Kim, Y.J. & Kim, J.H., 2001), la possibilité d'interagir directement avec le robot en générant une trajectoire qu'il doit suivre en temps réel est considérée. Le système utilisé consiste en un système instable afin d'étudier la possibilité de commander à distance des systèmes relativement complexes malgré les effets dûs aux délais variables, à l'inversion de paquets et à la perte de paquets sur le réseau. L'approche proposée inclut l'envoi de la consigne la plus récente par le serveur web au robot afin d'éliminer les paquets inversés et l'élaboration d'un contrôleur local robuste par la technique H_∞, visant à minimiser la sensibilité du système aux perturbations externes que représentent les effets de délais variables et de perte de paquets.

La solution proposée pour permettre la supervision du système commandé à distance est basée sur une simulation visuelle réalisée par un Javascript dans la page web dans laquelle un modèle virtuel permet d'anticiper le comportement du robot, et où il est possible de visualiser l'évolution de la trajectoire voulue ainsi que celle réellement empruntée par le robot. L'évolution du robot virtuel est obtenue par une simulation discrète basée sur le modèle de la dynamique en boucle fermé du robot. De cette façon, il est possible d'anticiper le comportement du robot avant de recevoir l'information sur son état actuel en présence de délais, et de comparer la trajectoire voulue et la trajectoire réelle du robot. L'algorithme de commande locale du robot inclut un contrôleur de poursuite de trajectoire afin d'assurer que la trajectoire réellement empruntée par le robot suive de près la trajectoire voulue, et que la position finale du robot se rapproche le plus possible de la position désirée.

Le document est organisé de la manière suivante. Dans le chapitre 1, le problème est défini et les différentes parties du projet sont introduites. Le chapitre 2 présente la conception mécanique et électronique du robot équilibriste utilisé pour expérimenter la commande par supervision via Internet. Le chapitre 3 traite de la modélisation mathématique de ce système mécatronique. Le chapitre 4 porte sur la commande en temps réel du robot incluant la conception d'algorithmes de commande et leur implémentation sur un système à base d'un microcontrôleur. Le chapitre 5 traite de l'architecture de communication mise en place afin de permettre la commande à distance à travers Internet et de l'application web développée permettant à des utilisateurs de commander et de surveiller le système mécatronique en temps réel à partir d'une page web de façon intuitive. Et, finalement, le chapitre 6 discute des expérimentations effectuées avec le système conçu.

CHAPITRE 1

DÉFINITION DU PROBLÈME

Le problème considéré dans ce projet de recherche consiste en la commande par supervision d'un robot équilibriste en temps réel à travers Internet. Ceci présente plusieurs défis considérables. Ceux-ci entrent dans deux grandes catégories : la commande des systèmes dynamiques en temp réel et la commande par Internet.

Dans le cadre de la commande par supervision, afin de pouvoir commander un système à distance, il est d'abord nécessaire de concevoir un système de commande local. Puisque ce système est en interaction constante avec son environnement, celui-ci doit être en mesure de répondre à des événements à l'intérieur de délais raisonnables afin de pouvoir accomplir la tâche avec succès. On parle alors d'un système temps réel, et il est donc nécessaire d'avoir recours aux techniques de programmation propres à ce type de système. De plus, puisque le montage mécatronique consistant en un pendule inversé mobile est un système instable. il est nécessaire de concevoir un contrôleur capable de le stabiliser. Ceci représente un défi supplémentaire du point de vue de la théorie de la commande et des techniques d'analyse et de conception appropriées doivent être utilisées. Enfin, lors de la poursuite de trajectoire par un robot mobile, malgré l'utilisation d'un contrôleur pour la direction du robot, il existe souvent une erreur entre la position voulue et la position réelle du robot à cause d'incertitudes et de perturbations externes. Pour cette raison, il est nécessaire de développer un algorithme pour minimiser cette erreur et ainsi assurer que le robot suive la trajectoire voulue avec le plus de précision possible.

Dans le but de réduire les coûts du montage conçu, le système de commande est basé sur des composantes électroniques accessibles tel qu'un microcontrôleur et des capteurs basés sur les technologies des microsystèmes électromécaniques (*Micro-Electro-*

Mechanical Systems - MEMS). Une attention est donc requise pour la sélection et l'intégration de ces composantes. Afin de prendre en considération le fait que les capteurs utilisés offrent des performances limitées, que leurs signaux contiennent du bruit et que le système de commande requiert une bonne précision sur la mesure de l'inclinaison du robot afin de pouvoir le stabiliser, une technique de fusion des signaux de plusieurs capteurs redondant est requise. Un algorithme particulier de traitement de signaux est donc nécessaire. Aussi, toujours dans l'optique de réduction des coûts, et dans le but de démontrer les principes de programmation des systèmes temp réel, le logiciel embarqué du système de commande est entièrement développé à l'aide du language C au lieu de dépendre d'un outil de génération de code tel que Matlab ou Labview.

Puisque le but du projet est de commander un système à distance à travers Internet, les problèmes reliés aux communications par Internet doivent être pris en considération. Ces problèmes incluent principalement une bande passante limitée ainsi que des délais de communication. Ces délais sont variables, ce qui implique la possibilité d'inversion et de perte de paquets.

Traditionnellement, tel que représenté par le robot de l'UWA, les systèmes de commande par supervision via Internet étaient conçus en tant que pages web émettant des requêtes à un serveur qui devaient alors être traitées par un CGI consistant en un programme exécuté par ce serveur. Cette approche demande un haut taux d'échange d'information entre le client et le serveur puisqu'à chaque requête le serveur doit émettre une réponse sous la forme d'une nouvelle page web et ceci entraîne donc des délais considérables dû aux traitements du serveur et aux échanges de données. Pour cette raison, il est difficile de commander un système en temps réel de cette façon. Il est donc préférable d'utiliser une topologie de système plus adaptée au problème de commande à distance permettant un échange de messages plus courts directement entre l'application client et l'application serveur. De plus, il est nécessaire de mettre en place un mécanisme efficace permettant l'échange d'information entre l'application serveur et le système robotique.

Aussi, on remarque que la plupart des systèmes de commande par supervision via Internet utilisés dans les laboratoires universitaires sont basés sur des logiciels commerciaux tels que Matlab, Labview et WinCon. Il serait intéressant d'étudier la possibilité de mettre en place un système basé entièrement sur des technologies ouvertes afin de réduire les coûts, favoriser la flexibilité et l'extensibilité du système ainsi que faciliter le développement, l'entretien et le support d'un tel système. Ceci demande donc un effort particulier afin d'adapter ces technologies au problème de la commande par supervision par Internet.

Tel que présenté par (Lassó & Urbancsek, 2001), des délais considérables sont introduits lors de transferts d'images en provenance d'une caméra web étant donné la quantité d'information relativement grande requise par celles-ci. Pour cette raison, il n'est pas souhaitable qu'un opérateur dépende uniquement de l'image provenant d'une caméra afin d'effectuer la supervision du système à distance. Il est donc souhaitable de mettre en place une solution alternative permettant à un opérateur de surveiller l'évolution du système à distance. Idéalement, il est souhaitable d'être en mesure d'anticiper le comportement du système avant même de recevoir l'information portant sur son état actuel, puisqu'il existe toujours des délais de communication lors de l'acheminement de ces données.

Afin de faire face à ces nombreux défis, ce projet de recherche est décomposé en plusieurs parties. Dans un premier temps, le montage mécatronique doit être réalisé. Cette première étape elle-même comprend plusieurs aspects tels que la conception mécanique, la conception électronique, l'analyse et la modélisation mathématique, la synthèse de lois de commande, ainsi que la conception du logiciel du système de commande. Ensuite, afin de pouvoir éventuellement commander à distance le montage mécatronique réalisé, une architecture de communication doit être mise en place. Cette architecture de communication comprend plusieurs couches de communication incluant des communications inter-puces réalisées par le protocole *Serial Peripheral Interface* (SPI) permettant

à plusieurs composantes électroniques intégrées sur puce de communiquer entre elles, des communications sans fil entre plusieurs systèmes électroniques embarqués réalisées par une solution propriétaire faisant appel à la technique de modulation *Gaussian Frequency-Shift Keying* (GFSK) et opérant dans la bande internationale ISM à $2.4\,GHz$, des communications entre un système embarqué et un ordinateur PC réalisées par le *Universal Serial Bus* (USB), et finalement des communications entre un ordinateur PC agissant en tant que serveur web et d'autres ordinateurs agissant comme clients réalisées par le *Transmission Control Protocol / Internet Protocol* (TCP/IP). Une fois, cette architecture de communication multi-couches mise en place, une application web permettant de commander et de surveiller le montage mécatronique doit être conçue. Cette application comporte deux composantes principales : l'application serveur basée sur les technologies J2EE et la page web générée par cette dernière. Cette page est de type Web 2.0 avec des éléments interactifs agissant comme interface de commande et de surveillance intuitive et conviviale. Elle utilise également le protocole de communication de haut niveau AJAX permettant des échanges de messages directement entre la page web et le serveur. Ces différentes parties du projet sont introduites dans les sections qui suivent.

1.1 Robot équilibriste

Un système mécatronique consistant en un pendule inversé mobile a été réalisé dans le but d'expérimenter le système de commande par supervision à travers Internet. Ce système présente plusieurs défis dont le fait que celui-ci est un système instable et requiert donc un correcteur pour le stabiliser en le maintenant en équilibre.

1.1.1 Revue de la littérature

Le sujet des robots équilibristes a attiré de nombreux chercheurs et amateurs en raison des défis qu'il représente. Dans la littérature, on peut trouver un certain nombre de projets intéressants qui méritent d'être mentionnés. Dans (Anderson D., 2007), l'auteur présente un robot nommé *nbot*, donne quelques informations générales sur la fusion des signaux des capteurs pour évaluer l'angle d'inclinaison et donne la structure de base d'un correcteur à retour d'état permettant de garder le robot en équilibre. Mais il ne donne pas de modèle mathématique de la dynamique et ne propose pas de technique de calcul des gains du correcteur. Dans (Cardi, Ener & Wagner, 2005), une équipe d'étudiants diplômés de *Woodruff School of Mechanical Engineering* de *Geogria Institute of Technology* présente un robot équilibriste nommé *The Teetering Buffalo*. Ils développent un modèle d'état linéaire simple mais efficace pour la dynamique de l'inclinaison et du déplacement linéaire et expérimentent différentes structures de correcteurs et observateurs par simulations. Toutefois, bien qu'ils offrent des vidéos de leur robot en action, leur contribution reste limitée car ils n'ont pas discuté des aspects pratiques de leur projet et ne présentent pas de résultats expérimentaux détaillés. Dans (Grasser, D'Arrigo, Colombi & Rufer, 2002), une équipe de chercheurs de l'École Polytechnique Fédérale de Lausane présente leurs travaux sur la conception d'un robot nommé *Joe*. Ils développent un modèle mathématique de la dynamique et introduisent l'idée de dissocier la dynamique de l'inclinaison et de la position linéaire de la dynamique de direction. Ils évoquent les problèmes liés à l'utilisation de boîtes de réduction et proposent l'utilisation d'un filtre de premier ordre pour y remédier et une technique de placement de pôles pour calculer les gains d'un correcteur par retour d'état. Toutefois, ils omettent de montrer les détails de l'élaboration de leur modèle, ils ignorent la dynamique des moteurs et utilisent un gyroscope directement pour mesurer l'angle d'inclinaison, ignorant son effet de dérive. Dans (Nawawi & Ahmad, 2008), les auteurs utilisent le modèle développé dans (Grasser, D'Arrigo, Colombi & Rufer, 2002) afin de contrôler leur propre robot pendule inversé à

deux roues *Two-Wheeled Inverted Pendulum* (T-WIP) en utilisant un correcteur à retour d'état pour lequel les gains ont été calculés avec la technique de placement de pôles. Leur contribution vient du fait qu'ils ont mis en place l'algorithme dans le logiciel Matlab et sont en mesure de contrôler leur robot en temps réel. Cela nécessite une configuration relativement coûteuse et offre des avantages seulement pendant la phase de développement du produit. Dans (Ooi R., 2003), l'auteur développe un modèle d'état pour la dynamique d'inclinaison et de déplacement linéaire, propose un filtre de Kalman pour estimer l'angle d'inclinaison et sa dérivée par rapport au temps, basé sur un sous-système composé exclusivement de capteurs, et propose un correcteur par retour d'état pour l'inclinaison et le déplacement linéaire et un correcteur PID pour la direction. Il propose également deux méthodes pour le calcul des gains du correcteur de l'inclinaison et du déplacement linéaire : Le *Linear Quadratic Regulator* (LQR) et le placement de pôles et montre des simulations et des résultats expérimentaux qu'il a obtenu. Malheureusement, la technique utilisée pour obtenir le modèle mathématique de la dynamique de l'inclinaison et du déplacement linéaire donne de nombreux termes non-linéaires et certains détails sont perdus après la linéarisation. En outre, l'approche du filtre de Kalman utilisé ne parvient pas à prendre avantage de la dynamique connue du système et exige le réglage manuel par essais et erreurs pour obtenir des résultats acceptables. Dans (Kadir H., 2005), il discrétise le modèle développé dans (Ooi R., 2003) dans le but de concevoir un correcteur discret. Il propose les approches du LQR et du placement de pôles et étudie la performance du système par simulation. Toutefois, sa contribution demeure restreinte car il ne montre pas de résultats expérimentaux. Dans (Abdollah M., 2006), le chercheur utilise le modèle développé dans (Ooi R., 2003), ajoute la dynamique de direction et propose un algorithme de commande non-linéaire par mode de glissement qu'il étudie par le biais de simulations. Toutefois, il ne donne pas de résultats expérimentaux et son algorithme est très intensif en calculs et serait difficile à implémenter numériquement. Dans (Kim, Y., Kim, S.H. & Kwak, 2005), une équipe de chercheurs du Département de génie mécanique de *Korean Advanced Institute of Science and Technology* ont mis au

point un modèle mathématique qui tente de cerner avec précision la dynamique d'un robot pendule inversé à deux roues en utilisant la méthode de (Kane & Levinson, 1985). Ils ont également étudié l'opération d'un tel robot sur une surface inclinée et lors de virages. Ils proposent ensuite un correcteur par retour d'état pour lequel les gains sont calculés en utilisant la technique de LQR et montrent certains résultats expérimentaux. Bien que leur tentative de saisir la dynamique exacte est intéressante d'un point de vue théorique, le modèle obtenu est hautement non-linéaire et de nombreux détails sont perdus au cours de la linéarisation. De plus, malgré le fait qu'ils ont mentionné l'utilisation d'un inclinomètre en plus d'un gyroscope, ils n'expliquent pas comment ils combinent leurs signaux et semblent ensuite indiquer qu'ils utilisent exclusivement le gyroscope afin de mesurer l'angle d'inclinaison et sa dérivée par rapport au temps. Dans (Ha & Yuta, 1996), l'équipe de chercheurs a développé un modèle mathématique en utilisant l'équation du mouvement de Lagrange en considérant l'angle d'inclinaison ainsi que sa dérivée par rapport au temps et la vitesse angulaire des roues comme étant les états du système et le couple appliqué aux roues comme étant l'entrée du système. Ils proposent un correcteur capable de maintenir le robot en équilibre tout en allant à une vitesse constante définie par un signal de référence dans lequel les valeurs en régime permanent des états et du signal d'entrée sont d'abord calculées et le signal d'entrée est ensuite corrigé en utilisant un retour d'erreurs des états. Ils augmentent également le système en ajoutant un intégrateur de l'erreur de vitesse des roues, ajoutant ainsi la position angulaire des roues comme un autre état, et font valoir que cet intégrateur est capable de remédier à l'effet de dérive du gyroscope et contribue à la robustesse paramétrique du système. Les gains de leur correcteur sont déterminés selon la technique de LQR. Pour le contrôle de la direction, ils ne font que calculer la vitesse désirée de chaque roue en fonction des signaux de référence pour la vitesse linéaire et la vitesse de changement de direction et envoient celles-ci aux correcteurs des roues tel que défini précédemment. Ils ont ensuite introduit un correcteur de poursuite de trajectoire dans lequel ils corrigent la référence de la vitesse de changement de direction en utilisant la distance perpendiculaire de la

trajectoire désirée à la position réelle du robot et l'erreur dans d'angle de direction. Enfin, ils montrent certains résultats expérimentaux. Bien que la solution qu'ils proposent est très complète, leur approche a plusieurs inconvénients. L'hypothèse selon laquelle le robot va en ligne droite a été émise lors de l'élaboration de la dynamique et la conception du correcteur qui est ensuite utilisé pour le contrôle de la direction, la structure du correcteur part de l'hypothèse que la référence des vitesses des roues demeure constante au fil du temps mais en pratique, ce n'est pas le cas, l'effet de dérive du gyroscope et le jeu des boîtes de réduction sont supposés être traités par l'utilisation d'un intégrateur sur l'erreur en vitesse des roues, et, en raison du fait qu'ils ont ignoré la dynamique des moteurs dans leur modèle, ils ont besoin d'utiliser des contrôleurs de moteurs séparés.

Toutes les références trouvées dans la littérature mentionnées ci-dessus utilisent un certain type de commande par retour d'état afin de stabiliser le système. Tous les chercheurs sauf (Ooi R., 2003) et (Kadir H., 2005) ont fait l'hypothèse que les encodeurs optiques captent les positions des roues par rapport au sol et peuvent donc être utilisés pour mesurer le déplacement linéaire du robot. Ceci n'est pas précisément le cas puisque l'inclinaison du robot provoque aussi le mouvement des moteurs et donc des encodeurs également. De plus, de nombreux modèles ne tiennent pas compte des boîtes de réduction, qui sont souvent nécessaires pour accroître le couple fourni par les moteurs à courant continu. La plupart des modèles existants ignorent la dynamique des moteurs et considère les couples générés par les moteurs comme étant les entrées du système, ce qui implique la nécessité d'utiliser des contrôleurs de moteurs additionnels. Ils prennent également l'hypothèse irréaliste et contradictoire que le robot va en ligne droite lors de l'élaboration du modèle de la dynamique d'angle de direction.

1.1.2 Conception

Le robot équilibriste conçu est composé d'un microcontrôleur dsPIC30F4011, de deux moteurs à courant continu, de deux puces de pont en H, d'un module d'affichage à cristaux liquides (*Liquid Crystal Display* - LCD), de batteries, de deux régulateurs de tension, d'un gyroscope, d'un accéléromètre, d'encodeurs optiques et d'une puce de communication sans fil.

1.1.3 Modélisation Mathématique

Dans le cadre de ce projet, un modèle complet de la dynamique d'un robot équilibriste a été développé. Ce modèle tente de capturer les relations exactes entre les signaux émis par les capteurs et les états du système, comprenant l'effet des boîtes de réduction, prend en compte la dynamique des moteurs, et évite d'avoir à prendre des hypothèses irréalistes en ce qui concerne le déplacement du robot lors de l'établissement des modèles dynamiques découplés de la position linéaire et de la direction. De plus, le modèle développé tente de représenter la dynamique du système dans un modèle d'état linéaire relativement simple mais efficace. Ceci est accompli en prenant l'hypothèse que certains phénomènes tels que l'effet des forces de réaction entre les roues et le corps du robot et la force centrifuge sont négligeables.

1.1.4 Commande en temps réel

Des lois de commande ont été implémentées numériquement dans un programme exécuté par le microcontrôleur. Ceux-ci consistent en des retours d'états avec actions intégrales sur la position linéaire et l'angle de direction du robot permettant de contrôler l'inclinaison et la vitesse linéaire du robot ainsi que sa direction, tout en assurant une er-

reur nulle en régime permanent en présence d'incertitudes et de perturbations. Les gains de ces correcteurs ont étés déterminés en utilisant la méthode de placement de pôles dans un premier temps puis en faisant appel à la théorie H_∞ afin de minimiser la sensibilité du système aux perturbations.

En plus des correcteurs, ce système requiert certains filtres afin de traiter les signaux des capteurs. Ceci est nécessaire afin de gérer le fait que les capteurs utilisés ne sont pas idéaux et qu'il existe certaines incertitudes et du bruit dans leurs signaux. À ce titre, un ensemble de filtres complémentaires a été utilisé pour permettre l'estimation de l'angle d'inclinaison en fusionnant les signaux d'un gyroscope et d'un accéléromètre. Des filtres passe-bas de type Butterworth on également été utilisés sur les mesures du déplacement linéaire et de l'angle de direction afin d'éliminer les bruits causés entre autre par les boîtes de réduction.

1.2 Commande via Internet

Afin de permettre éventuellement la commande à distance du système réalisé à partir d'une page web, plusieurs couches de communications sont nécessaires. De plus, afin de permettre des communications entre le serveur web et le robot, une autre carte électronique, surnommée gestionnaire de communication, a été conçue. Celle-ci communique par USB au PC agissant comme serveur web et par radio-fréquence avec le robot. Par la suite, une application web générant une page web agissant comme interface utilisateur a été développée.

1.2.1 Architecture de communication

L'architecture de communication inclut des communications *Serial Peripheral Interface* (SPI) entre des composantes électroniques sur les cartes électroniques du robot équilibriste et du gestionnaire de communication, des communications sans fil par radiofréquence entre le robot et le gestionnaire de communication, des communications USB entre le gestionnaire de communication et le PC agissant comme serveur web, puis des communications TCP/IP entre ce serveur web et le PC utilisé par un opérateur sur Internet.

1.2.2 Application web

Une page web permettant la commande par supervision du robot équilibriste via Internet a été réalisée. Cette page contient plusieurs éléments permettant la commande à distance du robot équilibriste ainsi que sa supervision. Ces éléments incluent une interface graphique, une simulation visuelle et une image provenant d'une caméra web. L'interface graphique permet à un opérateur de gérer la connexion au système et de générer une trajectoire voulue pour le robot. La simulation permet de visualiser l'évolution de la trajectoire générée, de la trajectoire anticipée et de celle réellement empruntée par le robot tandis que l'image provenant d'une caméra web sert de confirmation visuelle de la position actuelle du robot.

Cette page web est générée par une application web exécutée sur un PC agissant comme serveur web. Cette application est développée de toute pièce dans le cadre de ce projet et est basée sur la technologie J2EE. Les délais de communication entre la page web et le serveur sont minimisés en faisant appel à la technique AJAX permettant des communications HTTP directement entre ceux-ci. De plus, le temps de réponse du serveur a été minimisé en portant une attention particulière sur l'architecture de cette application.

CHAPITRE 2

CONCEPTION DU ROBOT ÉQUILIBRISTE

Dans le cadre de ce projet, un système mécatronique consistant en un robot équilibriste a été utilisé en tant que banc d'essais pour la commande par supervision via Internet. Ce montage a été réalisé suite à une proposition du professeur E.K. Boukas de concevoir et de tester un tel système au laboratoire de mécatronique. La conception a été réalisée avec les objectifs de simplicité et de réduction des coûts. Dans cette optique, plusieurs des composantes tels que les moteurs, les boîtes à réduction, les encodeurs optiques, les batteries, les puces de pont en H et le module d'affichage ont été récupérés parmi des composantes disponibles au laboratoire.

Dans un premier temps, la conception mécanique a été réalisée par Vincent Bouverat dans le cadre d'un projet de fin d'étude. Puis, la conception de la partie électronique du système, sa modélisation mathématique, le développement des algorithmes de commande et la programmation du système ont été réalisés par l'auteur et constituent une partie intégrale du projet de recherche présenté dans ce document. Les informations se rattachant aux divers aspects de la conception de ce système sont présentées dans ce chapitre.

2.1 Mécanique

Le design proposé pour le robot équilibriste à deux roues est simple, léger, et peu coûteux. La conception mécanique est constituée d'un corps avec une tige et de deux roues, chacune entraînée par un moteur à courant continu de $24\ V$ muni d'une boîte de réduction ayant un rapport de $1 : 6$. Le corps et les roues ont été conçus en utilisant un

logiciel de CAO et ont été fabriqués et assemblés à l'École Polytechnique de Montréal. La conception des roues comprend une série de trous circulaires afin de minimiser leur poids. Les paramètres physiques du corps et des roues ont été estimés en utilisant la géométrie des pièces avec le logiciel de CAO et les paramètres des moteurs, boîtes de réduction et des encodeurs optiques ont été obtenus à partir de leurs fiches techniques respectives. La structure mécanique du robot est montré à la FIG. 2.1.

Ce type de robot représente un système instable et présente certains défis qui seront abordés dans le reste du document. Dans un premier temps, afin de concevoir un correcteur capable de stabiliser le système, un modèle mathématique est nécessaire. Cet aspect sera examiné au chapitre 3.

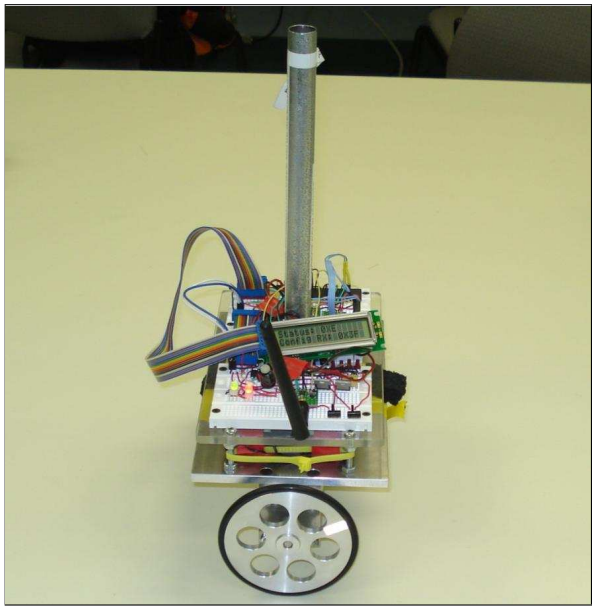

FIG. 2.1 Structure mécanique du robot équilibriste

2.1.1 Actionneurs

Les actionneurs utilisés pour le robot sont des moteurs à courant continu avec des boîtes de réduction. Il s'agit du modèle **F-2140-937** du fabriquant Maxon Motors. Ces moteurs ont un diamètre extérieur de $40\ mm$, une longueur de $86.2\ mm$, une puissance de $6\ W$ et une tension d'alimentation nominale de $24\ V$. Ils peuvent générer un couple constant allant jusqu'à $13.5\ mNm$, ont besoin d'un courant de $578\ mA$ pour démarrer et d'un courant maximal de $244\ mA$ de façon continue lors de leur opération. Les moteurs sont commandés par des signaux de modulation en largeur d'impulsions générés par le microcontrôleur à travers un circuit de pont en H. Ces aspects seront abordés aux sections 2.2.6.5 et 2.2.4 respectivement.

La boîte de réduction utilisée est le modèle **GS-38-110451** du même fabriquant. Il s'agit de têtes motrices à engrenages cylindriques ayant un facteur de réduction de $1 : 6$, un rendement de $81\ \%$ et pouvant supporter un couple maximal de $100\ mNm$. Pour plus de détails concernant les moteurs et boîtes de réduction, le lecteur est référé aux fiches techniques (Maxon DC Motors, 2005), (Maxon Gear, 2005).

2.2 Électronique

Le système électronique conçu pour la commande du robot équilibriste se compose d'un système embarqué utilisant un microcontrôleur en tant qu'unité de traitement principal. Deux encodeurs optiques avec une résolution de 100 impulsions par révolution sont utilisés pour mesurer les positions des roues. Un accéléromètre à deux axes et un gyroscope sont utilisés pour mesurer l'angle d'inclinaison du corps du robot et son taux de variation par rapport au temps. Deux puces de ponts en H sont utilisées pour la commande des moteurs à courant continu. Un module d'affichage à cristaux liquides est utilisé comme interface visuelle et une puce de communication sans fil a été utilisée pour permettre

la communication avec un ordinateur. Afin de garantir l'autonomie du robot, un ensemble de 20 piles de $1,2\ V$ montées en série est utilisé, ce qui donne un total de $24\ V$ permettant d'alimenter les moteurs. La partie numérique du système est alimentée par l'intermédiaire de deux régulateurs de tension. L'architecture du système numérique est montrée à la FIG. 2.2 et les principales composantes de ce système sont énumérées dans le tableau 2.1.

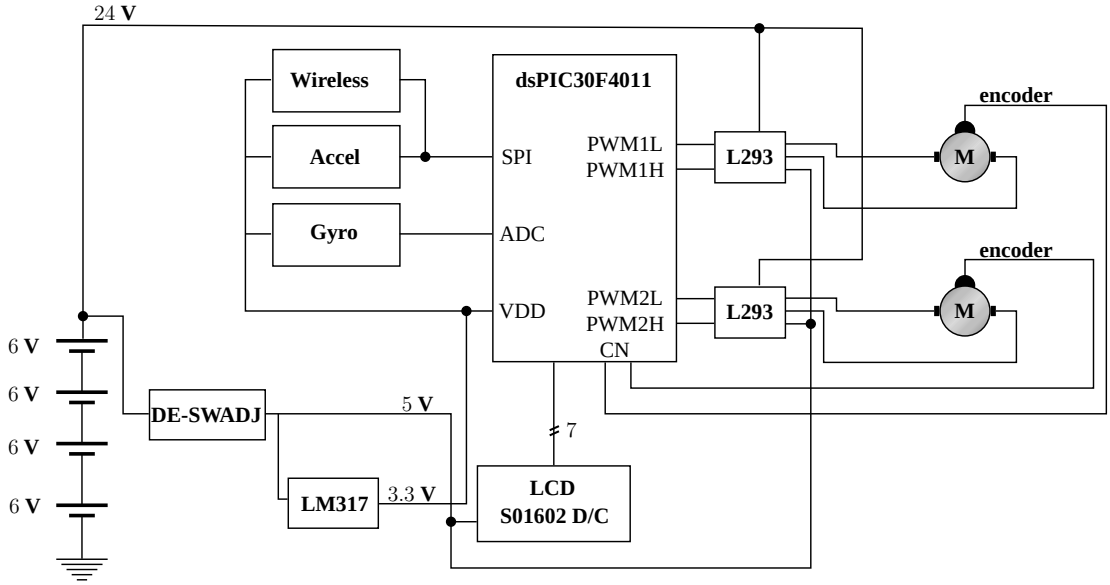

FIG. 2.2 Architecture du système numérique

2.2.1 Alimentation

L'alimentation des différentes composantes du système est réalisée par un circuit contenant un ensemble de batteries, des capacités, des résistances et des régulateurs de tension. Le schéma de ce circuit est présenté à la FIG. 2.3. Le niveau de tension de $24\ V$ requis par les moteurs est obtenu à partir de 4 blocs de batterie de $6\ V$. Chacun de ces blocs de batteries est lui-même composé d'un ensemble de 5 batteries de $1.2\ V$. La capacité de $1000\ \mu F$ est utilisée afin de stabiliser le niveau de tension lorsque les moteurs requièrent un courant considérable. Le régulateur de tension de modèle **DE-SWADJ** sert

Numéro	Description	Fonction
dsPIC30F4011	Microcontrôleur	Unité de calcul
LIS3LV02DQ	Accéléromètre	Estimation de l'inclinaison
IDG300	Gyroscope	Mesure de vitesse d'inclinaison
S01602 D/C	Afficheur à cristaux liquides	Affichage et débogage
DE-SWADJ	Régulateur de tension par découpage	Alimentation
LM317	Régulateur de tension variable	Alimentation
L293	Puce de ponts en H	Commande de Moteur
F2140	Moteur à courant continu	Actionneurs
GS38	Réducteur	Transmission
Enc22	Encodeur optique	Mesure rotation moteur
nRF24L01	Émetteur-récepteur 2.4GHz	Communication

TAB. 2.1 Composantes du système

à abaisser le niveau de tension à $5\ V$ afin d'alimenter le module d'affichage, les puces de ponts en H et les encodeurs optiques. Il s'agit d'un régulateur de tension par découpage permettant d'obtenir une sortie de tension variable. Cette composante a été sélectionnée puisque son principe de fonctionnement par découpage permet d'obtenir une meilleure efficacité énergétique en comparaison avec un régulateur de tension linéaire traditionnel. Ce régulateur de tension est alimenté par l'ensemble des piles afin que celles-ci se déchargent de façon uniforme. Un deuxième régulateur de tension de modèle **LM317** a pour but d'abaisser le niveau de tension à $3.4\ V$ afin d'alimenter le microcontrôleur, le gyroscope, l'accéléromètre et la puce de communication sans fil. Cette composante est un régulateur de tension linéaire variable. Celle-ci est utilisée puisqu'elle offre une bonne isolation contre les bruits afin d'éviter les interférences qui pourraient affecter le bon fonctionnement de la composante de communication sans fil. La résistance de $220\ \Omega$ ainsi que la résistance variable de $2\ K\Omega$ servent à ajuster le niveau de tension obtenu à la sortie de ce régulateur de tension. La capacité de $1\ \mu F$ sert à améliorer le fonctionnement de ce régulateur de tension en régime transitoire.

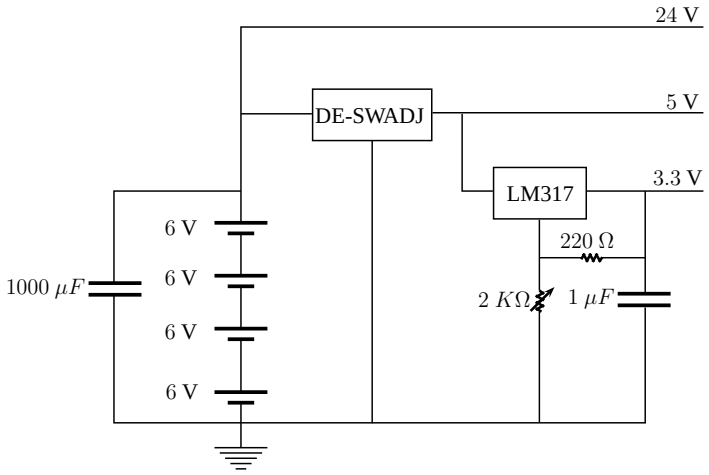

FIG. 2.3 Circuit d'alimentation

2.2.2 Capteurs

Afin de pouvoir stabiliser et commander le système en boucle fermée, on doit pouvoir mesurer certaines informations sur l'état actuel du système. Plus spécifiquement, on doit connaître l'angle d'inclinaison du robot, sa position linéaire, son angle de direction ainsi que les dérivées par rapport aux temps de ceux-ci. Afin de déterminer l'angle d'inclinaison du robot, un gyroscope ainsi qu'un accéléromètre ont été utilisés. Les signaux de ces capteurs comportent différents types de bruit et il est donc difficile d'obtenir une mesure fiable de l'angle d'inclinaison à partir d'un seul de ceux-ci. Par contre, il est possible d'obtenir un bon estimé en les combinant. Quant à la position linéaire et l'angle de direction du robot, ceux-ci sont déterminés à partir des positions angulaires des deux roues obtenues à partir des encodeurs optiques installés sur les arbres des moteurs. Ces différents capteurs sont présentés dans les sections qui suivent.

2.2.2.1 Gyroscope

Un gyroscope est un capteur qui produit un signal proportionnel à sa vitesse de rotation. Le gyroscope utilisé sur le robot équilibriste consiste en un **IDG-300** du fabriquant *InvenSense*. Cette composante est sous la forme d'un circuit imprimé QFN ayant des dimensions de 6 x 6 x 1.5 mm et elle est représentée à la FIG. 2.4. Il s'agit d'un gyroscope analogique permettant de mesurer des vitesses de rotation autour de deux axes perpendiculaires. Dans le cas de cette application, un seul de ces canaux est utilisé, celui correspondant à l'axe parallèle à celui passant par les arbres des moteurs autour duquel le robot s'incline. Le signal analogique de ce canal est récupéré à l'aide du module de conversion analogique numérique (A/D) du microcontrôleur présenté en détail à la section 2.2.6.4. Le gain reliant la sortie analogique à la vitesse de rotation est de $2\ mVs/^o$. Cette composante a également une troisième sortie analogique consistant en un signal de référence ayant une tension constante de $1.23\ V$. Ce signal de référence sera utilisé afin de calibrer les valeurs de conversion analogique numérique du signal indiquant la vitesse de rotation.

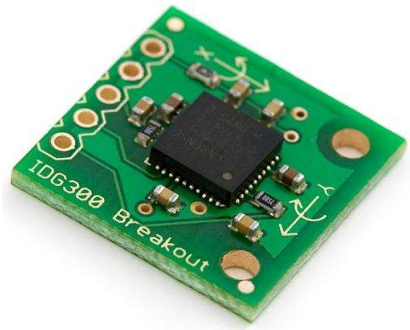

FIG. 2.4 Gyroscope **IDG-300**

L'une des faiblesses de ce type de capteur est qu'il existe une erreur ayant une moyenne non-nulle dans le signal qu'il produit dû à une sensibilité à la température ambiante. Puisque ce signal est proportionnel à la vitesse de rotation et qu'on cherche à estimer la position angulaire, ce signal doit être intégré et, à cause de cette erreur non-nulle, l'es-

timé dévie de la valeur réelle de position angulaire. Ce phénomène est connu sous le nom de dérive (*drift*). À cause de ceci, il est difficile d'estimer l'inclinaison du robot à partir du gyroscope uniquement. C'est pourquoi on aura aussi recours à un accéléromètre afin d'obtenir un autre estimé de l'inclinaison du robot. L'accéléromètre utilisé est présenté à la section 2.2.2.2. Il sera éventuellement possible d'obtenir un bon estimé de l'angle d'inclinaison en combinant les estimés provenant du gyroscope et de l'accéléromètre en utilisant une technique connue sous le nom de filtres complémentaires. Cette technique est présentée en détail à la section 4.1.1.3.

2.2.2.2 Accéléromètre

Comme mentionné à la section précédente, il est nécessaire de combiner l'estimé de l'angle d'inclinaison du robot obtenu à partir du signal du gyroscope à un estimé obtenu par un autre moyen pour avoir un estimé fiable. Un type de capteur couramment utilisé dans ce contexte est un inclinomètre. Par contre, à cause de son principe de fonctionnement basé sur un pendule, ce type de capteur a une dynamique relativement lente et donc une bande passante étroite limitant la performance du système d'estimation. De façon alternative, un accéléromètre détenant une bande passante largement supérieure à celle d'un inclinomètre peut être utilisé afin d'obtenir une mesure de l'angle d'inclinaison. Cette dernière alternative a été choisie dans la conception du système de mesure du robot équilibriste en raison de la meilleure performance offerte par un tel capteur et étant donné que ce type de capteur est également moins coûteux.

À la base, un accéléromètre est un capteur produisant un signal proportionnel à son accélération. Par contre, celui-ci ne détecte pas uniquement les accélérations mais aussi la force de la gravité. Par conséquent, il est possible d'obtenir un estimé de l'angle d'inclinaison du robot en utilisant un calcul de trigonométrie inverse sur les forces détectées dans les directions des axes perpendiculaires à l'axe autour duquel le robot s'incline.

L'accéléromètre utilisé sur le robot équilibriste consiste en un **LIS3LV02DQ** du fabriquant *STMicroelectronics*. Cette composante se présente sous la forme d'un circuit imprimé QFPN-28 ayant des dimensions de 7 x 7 x 1.8 mm et est démontrée à la FIG. 2.5. Il s'agit d'un accéléromètre à interface sérielle numérique I2C/SPI permettant de mesurer des accélérations dans trois directions perpendiculaires. Dans le cas de cette application, on n'utilise que deux des canaux, étant ceux correspondant aux axes x et z tel que représentés à la FIG. 3.1. Ce sont dans ces directions que le vecteur de force de la gravité agit et il est donc possible d'obtenir un estimé de l'angle d'inclinaison à l'aide d'un calcul de trigonométrie inverse tel que décrit à la section 4.1.1.2.

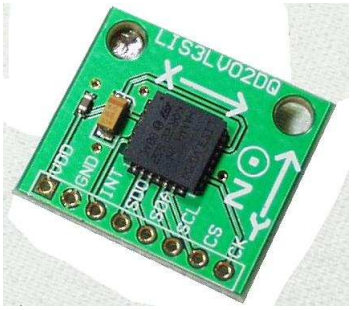

FIG. 2.5 Accéléromètre **LIS3LV02DQ**

L'une des faiblesses de ce type de capteur dans le contexte d'utilisation actuel est le fait que celui-ci ne détecte pas uniquement la force de la gravité mais aussi celles dûes aux vibrations et aux mouvements. Donc, cet estimé n'est exact que lorsque le robot ne bouge pas. De façon générale, celui-ci contient également du bruit dû aux vibrations. Pour cette raison, il est difficile d'obtenir un bon estimé de l'angle d'inclinaison à partir des signaux de l'accéléromètre uniquement. C'est pourquoi il est nécessaire de combiner l'estimé de l'accéléromètre à celui du gyroscope à l'aide de filtres complémentaires, tel que décrit à la section 4.1.1.3.

2.2.2.3 Encodeurs Optiques

Un encodeur optique est un capteur permettant de déduire la position actuelle de l'arbre d'un moteur. Celui-ci est composé de deux sources lumineuses, de deux capteurs optiques correspondants et d'une roue ayant des fentes laissant passer la lumière. À sa sortie, l'encodeur produit des signaux sur deux canaux, le canal A et le canal B, correspondant aux deux paires de source lumineuse/capteur. Ces paires de source lumineuse/capteur sont espacées de façon à ce que les signaux produits aient une différence de phase de 90^o, ce qui permet de déduire le sens de rotation du moteur. Une impulsion est produite dans l'un des canaux à chaque fois qu'une fente laisse passer de la lumière d'une source lumineuse à son capteur correspondant. La précision de ce type de capteur est en termes d'impulsions par révolution, ou, autrement dit, par le nombre de fentes de sa roue. Un exemple de signaux produits par un encodeur optique est présenté à la FIG. 2.6. On peut voir que lorsque le moteur tourne dans le sens positif, lors d'un front montant du signal du canal A, le signal du canal B est à l'état bas. Tandis que lorsque le moteur tourne dans le sens négatif c'est l'inverse qui se produit - lors d'un front montant du signal du canal A, le signal du canal B est à l'état haut. C'est cette propriété qui est utilisée afin de déterminer le sens de rotation d'un moteur.

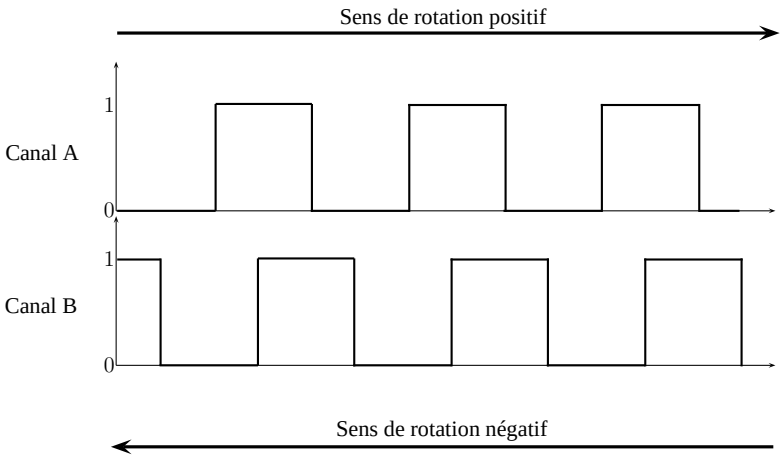

FIG. 2.6 Signaux d'un encodeur optique

Deux méthodes peuvent être utilisées afin de décoder les signaux d'un encodeur optique. La première considère seulement les fronts montants du signal de l'un des deux canaux. Ceci nous procure la résolution nominale de l'encodeur. La deuxième méthode considère les fronts montants et descendants des signaux des deux canaux ce qui a pour effet de quadrupler la résolution. Cette deuxième méthode a été utilisée dans le programme exécuté par le microcontrôleur du robot en utilisant les interruptions sur changements qui seront expliquées en détail à la section 2.2.6.2.

Les encodeurs optiques utilisés sur le robot équilibriste sont des **Enc 22** de la compagnie *Maxon*. Ils ont une résolution de 100 impulsions par révolution du moteur. Ceci donne une résolution de 600 impulsions par révolution d'une roue avec la première méthode de décodage, et 2400 impulsions par révolution d'une roue avec la deuxième méthode.

2.2.3 Afficheur à cristaux liquides (LCD)

Un afficheur à cristaux liquides (*Liquid Crystal Display* - LCD) a été utilisé sur le robot afin d'afficher la valeur de certaines variables pour des fins de débogage lors du développement de l'application. Le module LCD utilisé dans ce projet a le numéro de pièce **S01602 D/C** et provient du fabriquant *Lumex*. Il s'agit d'un module basé sur la technologie *Reflective Twisted Neumatic*. Il offre une interface parallèle afin de le configurer et de lui communiquer des commandes et des données à afficher.

FIG. 2.7 Module d'afficheur à cristaux liquides **S01602 D/C**

2.2.4 Interface de puissance

Puisque le microcontrôleur ne peut générer des signaux de puissance capables d'alimenter les moteurs directement, un circuit d'interface de puissance est requis. Le type d'interface choisi consiste en une composante de circuit intégré de format PDIP de 16 pattes comportant deux ponts en H portant le numéro de pièce **L293**. Chaque pont en H consiste en deux entrées et deux sorties et est capable de commander un moteur dans les deux sens de rotation. Les différents modes de fonctionnement d'un pont en H sont présentés dans le tableau suivant :

Entrée A	**Entrée B**	**Mode**
0	0	Moteur arrête ou freine
0	1	Moteur tourne dans le sens horaire
1	0	Moteur tourne dans le sens anti-horaire
1	1	Moteur arrête ou freine

TAB. 2.2 Table de vérité d'un pont en H

Puisque les moteurs ne fonctionnent pas à plein régime en tout temps, ceux-ci seront commandés par des signaux de modulation en largeur d'impulsion envoyés aux entrées des ponts en H de la composante d'interface de puissance. Le module du microcontrôleur permettant de générer de tels signaux sera présenté en détail à la section 2.2.6.5.

2.2.5 Communication sans fil

Le système contient une composante de communication sans fil permettant d'effectuer des transferts de données à distance avec une carte électronique nommée gestionnaire de communication agissant comme interface de communication avec un ordinateur. La composante choisie a le numéro de pièce **nRF24L01** et il s'agit d'une puce de format QFN20 avec un émetteur/récepteur embarqué à $2,4\ GHz$ et un engin de base de pro-

tocole, conçue pour des applications sans fil à ultra basse puissance. Cette composante opère dans la bande de fréquence ISM ($2.4 - 2.4835\ GHz$). Elle est configurée et gérée par le biais du bus SPI par lequel ses registres de configuration et d'opération sont accessibles.

L'engin de protocole intégré est basé sur des paquets de communication et prend en charge plusieurs modes de fonctionnement pour assurer une opération autonome du protocole. Des tampons de type premier arrivé premier sorti (*First In First Out* - FIFO) assurent un bon flux de données entre la partie radio et la partie numérique de la puce. Le protocole offre des caractéristiques telles que la confirmation de réception (*Acknowledgement*) avec des données encapsulées et la relance automatique lors de pertes de paquets, ce qui permet de réaliser des communications dans les deux sens de manière efficace.

La radio utilise une modulation de type GFSK. Plusieurs paramètres comme le canal de fréquence, la puissance de sortie et de débit de données sont entièrement configurables. Le débit soutenu est configurable jusqu'à 2Mbps. Ce haut débit combiné avec deux modes d'économie d'énergie rend cette composante très convenable pour la conception de très faible puissance.

2.2.6 Microcontrôleur

Le microcontrôleur utilisé dans le système embarqué du robot équilibriste a le numéro de pièce dsPIC30F4011 et provient du fabricant Microchip. Ce microcontrôleur a été choisi parce qu'il est puissant et peu coûteux. Il offre la possibilité de générer plusieurs signaux par modulation de largeur d'impulsion, ce qui permet de contrôler les deux moteurs indépendamment. Son fabricant offre un compilateur C gratuit, le compilateur C30, ce qui permet de créer et de maintenir la partie logicielle du système de manière pratique

et efficace. Ce circuit imprimé comporte plusieurs fonctionnalités utilisées dans l'implémentation du système de commande du robot équilibriste. Celles-ci sont brièvement présentées dans les sections qui suivent. Pour de plus amples détails sur ces modules du microcontrôleur, le lecteur est référé au manuel de référence *dsPIC30F Family Reference Manual* (Microchip, 2006).

2.2.6.1 Interruption externe

Le concept des interruptions n'est pas unique au modèle de microcontrôleur choisi pour cette application mais bien une particularité de fonctionnement de tous les microprocesseurs. Une interruption est le mécanisme permettant à un microprocesseur de réagir à des événements externes ou ponctuels. Lorsqu'un événement particulier survient, un bit indicateur (*flag*) est activé et le microprocesseur interrompt son exécution du programme afin d'exécuter la routine d'interruption responsable de gérer l'événement en question, puis continue à exécuter le programme principal qu'il avait interrompu. Le programmeur est responsable de remettre le bit indicateur de l'interruption à zéro à la fin de la routine d'interruption afin d'indiquer que celle-ci a été gérée.

Dans la famille des microcontrôleurs dsPIC30F, un niveau de priorité est attribué à chaque type d'interruption activée en assignant des valeurs à des bits de registres de configuration. Il existe 8 niveaux de priorité et le microcontrôleur s'assure de gérer les interruptions en respectant leurs niveaux de priorité. Aussi, il s'agit de priorités dynamiques puisqu'il est possible de changer ces niveaux de priorité durant l'exécution du programme. Cette fonctionnalité est utilisée afin de gérer les ressources partagées et éviter les conflits tel qu'expliqué à la section 4.3.2.2.

Un des types d'interruptions des plus simples est l'interruption externe. Ce type d'interruption réagit à un front montant ou à un front descendant survenant sur l'une des

entrées numériques identifiée par INTx du microcontrôleur. Afin d'avoir recourt à cette fonctionnalité, il suffit d'activer ce type d'interruption, de lui assigner une priorité et de sélectionner le type de front (montant ou descendant) la déclenchant. Voici la portion de code étant responsable d'initialiser l'interruption externe sur l'entrée INT0 :

```
IEC0bits.INT0IE = 1;                      // Activation interruption INT0
IPC0bits.INT0IP = COMMUNICATION_PRIORITY; // Priorité interruption externe
INTCON2bits.INT0EP = 1;                   // Interruption INT0 front descendant
```

Cette fonctionnalité a été utilisée dans le système du robot équilibriste afin de détecter un signal provenant de la composante de communication sans fil indiquant que celle-ci a reçu de nouvelles données prêtes à être récupérées. Plus particulièrement, les fronts descendants sont détectés sur l'entrée INT0 puis une routine d'interruption se charge d'entrer en communication avec la composante de communication sans fil à travers le bus SPI pour recevoir les nouvelles données, les décoder et réagir en conséquence.

Étant donné la nature temps réel de l'application de contrôle du robot, et donc son haut niveau d'interaction avec le monde extérieur, plusieurs autres types d'interruption sont utilisés dans celle-ci. Ces différents types d'interruptions seront introduits dans les sections qui suivent portant sur les différentes fonctionnalités du microcontrôleur utilisées dans le système de commande.

2.2.6.2 Interruption sur changements (CN)

Le microncontrôleur choisi offre la possibilité de générer une interruption lors de tout changement (*Change Notification* - CN) se produisant sur certaines de ses entrées numériques. Les entrées offrant cette fonctionnalité sont identifiées par CNx. Pour utiliser ce type d'interruption, il est d'abord nécessaire de sélectionner les entrées numériques qui produiront de telles interruptions lors de changements, d'initialiser le bit indicateur

de l’interruption à zéro, d’attribuer un niveau de priorité à ce type d’interruption puis d’activer le type d’interruption en question. La portion de code suivante effectue l’initialisation des interruptions sur changements :

```
CNEN1bits.CN0IE=1;  // Permet l’intéruption sur le port CN0
CNEN1bits.CN1IE=1;  // Permet l’intéruption sur le port CN1
CNEN1bits.CN2IE=0;  // Permet l’intéruption sur le port CN2
CNEN1bits.CN3IE=0;  // Permet l’intéruption sur le port CN3
CNEN1bits.CN4IE=1;  // Permet l’intéruption sur le port CN4
CNEN1bits.CN5IE=1;  // Permet l’intéruption sur le port CN5
CNEN1bits.CN6IE=0;  // Permet l’intéruption sur le port CN6
CNEN1bits.CN7IE=0;  // Permet l’intéruption sur le port CN7
CNEN2bits.CN17IE=0; // Permet l’intéruption sur le port CN17
CNEN2bits.CN18IE=0; // Permet l’intéruption sur le port CN18

IFS0bits.CNIF = 0;                // Mise à zéro du drapeau
IPC3bits.CNIP = ENCODER_PRIORITY; // Priorité interruptions CN
IEC0bits.CNIE = 1;                // Activation interruptions CN
```

Cette fonctionnalité a été utilisée afin de décoder les signaux provenant des encodeurs optiques. Les signaux des canaux des encodeurs optiques sont envoyés au microcontrôleur par les entrées CN0, CN1, CN4 et CN5 . De cette façon, il est possible de détecter les fronts montants et descendants des signaux des encodeurs et de mettre à jour les variables représentant les positions angulaires des roues dans la routine d’interruption.

Traditionnellement, les signaux des encodeurs optiques sont décodés à l’aide de composantes spécialisées contenant des compteurs et les valeurs de ces compteurs sont par la suite récupérées par le microcontrôleur. La solution proposée ici basée sur les interruptions CN réduit les coûts du système en éliminant ces composantes additionnelles et réduit le nombre de communications nécessaires qui auraient demandé soit plus d’entrées numériques soit plus de communications sur le bus SPI, qui auraient pu générer des conflits.

2.2.6.3 Temporisateurs

Les temporisateurs (*Timers*) sont des compteurs qui peuvent générer une interruption lorsqu'ils atteignent une valeur déterminée contenue dans un registre de configuration appelé le registre de période. Il est aussi possible de régler la fréquence à laquelle ces compteurs incrémentent leurs valeur avec un registre de configuration de prédiviseur de fréquence. Les temporisateurs sont utilisés dans l'application de commande du robot afin de cadencer les tâches de contrôle et de rafraîchissement du LCD. Pour utiliser ce type d'interruption, il est d'abord nécessaire de démarrer le temporisateur, de configurer son comportement quant à l'accumulation et son fonctionnement au repos (*sleep*), d'attribuer une valeur au prédiviseur, de sélectionner la source d'horloge du temporisateur, d'assigner une valeur au registre de période, d'initialiser le bit indicateur (*flag*) à zéro, d'attribuer une priorité à l'interruption puis d'activer ce type d'interruption. À titre d'exemple, la portion de code qui suit est responsable d'initialiser l'interruption du temporisateur 1 utilisée afin de cadencer les opérations se rattachant à l'algorithme de contrôle et de filtrage du robot :

```
T1CONbits.TON=1;   // Demarre le timer 1
T1CONbits.TGATE=0; // Accumulation désactivée
T1CONbits.TSIDL=0; // Continue durant repos
T1CONbits.TCKPS=1; // Prédiviseur 1:8
T1CONbits.TCS=0;   // Source d'horloge FOSC/4

PR1 = 14739;       // Période donnant fréquence de 250Hz

IFS0bits.T1IF = 0;                   // Mise à zéro du drapeau
IPC0bits.T1IP = CONTROLLER_PRIORITY; // Priorite 5 pour timer 1
IEC0bits.T1IE=1;                     // Activation interruption TMR1
```

2.2.6.4 Convertisseur analogique numérique (ADC)

Le microcontrôleur utilisé a un module de convertisseur analogique/numérique (*Analog to Digital Converter* - ADC) à 9 entrées capable d'offrir une précision de 10 bits.

Ce convertisseur est basé sur la technique d'approximation successive et est capable d'échantillonner jusqu'à quatre canaux à la fois. Dans l'application du robot, ce module est utilisé afin d'échantillonner deux signaux provenant du gyroscope. Plusieurs configurations sont nécessaires afin de choisir les canaux à échantillonner, si les conversions sont réalisées sur demande ou de façon continue, les niveaux de références positives et négatives ainsi que la fréquence et le temps d'acquisition. Ces configurations sont effectuées en attribuant des valeurs spécifiques à plusieurs registres. Des fonctions de la librairie du compilateur C30 ont été utilisées afin de faciliter la procédure. La portion de code suivante montre l'utilisation de ces fonctions afin de configurer le module de conversion analogique numérique du microcontrôleur :

```
// Sélection des cannaux
SetChanADC10(ADC_CH0_POS_SAMPLEA_AN1 & ADC_CH0_NEG_SAMPLEA_NVREF &
             ADC_CHX_POS_SAMPLEA_AN3AN4AN5 & ADC_CHX_NEG_SAMPLEA_NVREF);

// Désactivation de l'interruption
ConfigIntADC10(ADC_INT_DISABLE);

// Configuration du module en mode continue
OpenADC10(ADC_MODULE_ON & ADC_IDLE_CONTINUE & ADC_FORMAT_INTG &
          ADC_CLK_AUTO & ADC_AUTO_SAMPLING_ON & ADC_SAMPLE_SIMULTANEOUS,
          ADC_VREF_AVDD_AVSS & ADC_SCAN_OFF & ADC_CONVERT_CH_0ABC &
          ADC_SAMPLES_PER_INT_16 & ADC_ALT_BUF_OFF & ADC_ALT_INPUT_OFF,
          ADC_SAMPLE_TIME_31 & ADC_CONV_CLK_INTERNAL_RC  & ADC_CONV_CLK_32Tcy,
  ENABLE_AN1_ANA & ENABLE_AN4_ANA & ENABLE_AN5_ANA, SCAN_NONE);
```

Dans l'application du robot équilibriste, le module ADC du microcontrôleur a été configuré de façon à ce que deux canaux du gyroscope soit automatiquement échantillonnés de manière périodique. Les résultats des conversions les plus récentes se trouvent dans des registres prévus à cet effet.

2.2.6.5 Modulation de largeur d'impulsion (PWM)

Un signal de modulation de largeur d'impulsion (*Pulse Width Modulation* - PWM) est un signal à onde carrée pour lequel on contrôle la durée durant laquelle le signal reste à

l'état haut. Cette portion de temps est nommée le cycle actif (*Duty Cycle*). La fréquence de ce signal doit être significativement plus élevée que la fréquence d'échantillonnage utilisée pour la commande numérique du système afin que le niveau de tension puisse être perçu comme une moyenne durant une période d'échantillonnage. De plus, lorsque ce type de signal est utilisé pour contrôler un moteur, il est souhaitable d'utiliser une fréquence au delà de la plage de fréquences audibles par l'oreille humaine afin d'éviter de produire des sifflements.

Le microcontrôleur choisi pour cette application détient un module matériel capable de générer jusqu'à quatre signaux PWM. Ce module peut être opéré selon différents modes incluant le mode complémentaire et le mode indépendant. Avec le mode complémentaire, les quatre canaux sont opérés par paires de signaux complémentaires, ce qui signifie qu'un signal est à l'état opposé d'un autre en tout temps. Tandis qu'avec le mode indépendant, chacun des canaux peut être opéré de façon indépendante. Bien que les deux modes auraient pu être utilisés pour faire la commande des moteurs dans les deux sens, le mode indépendant a été utilisé puisque celui-ci offre un bit supplémentaire de précision pour une même fréquence. Avec ce mode, pour chacun des moteurs, un canal est mis à l'état bas selon le sens de rotation voulu, voir TAB. 2.2, tandis qu'un signal PWM est appliqué à l'autre canal afin de contrôler le niveau de tension appliqué au moteur.

Le module de PWM du microcontrôleur est basé sur un temporisateur PTMR. Un registre nommé PTPER ainsi qu'un prédiviseur permettent de fixer la période T du signal. Lorsque le compteur atteint la valeur de ce registre, il revient à zéro et le signal PWM est remis à l'état haut. Un autre registre nommé PDCx sert à déterminer le cycle actif du signal. Lorsque le compteur atteint la valeur de ce registre, le signal PWM est mis à l'état bas. La FIG. 2.8 montre l'évolution de la valeur du temporisateur PTMR ainsi que du signal PWM associé.

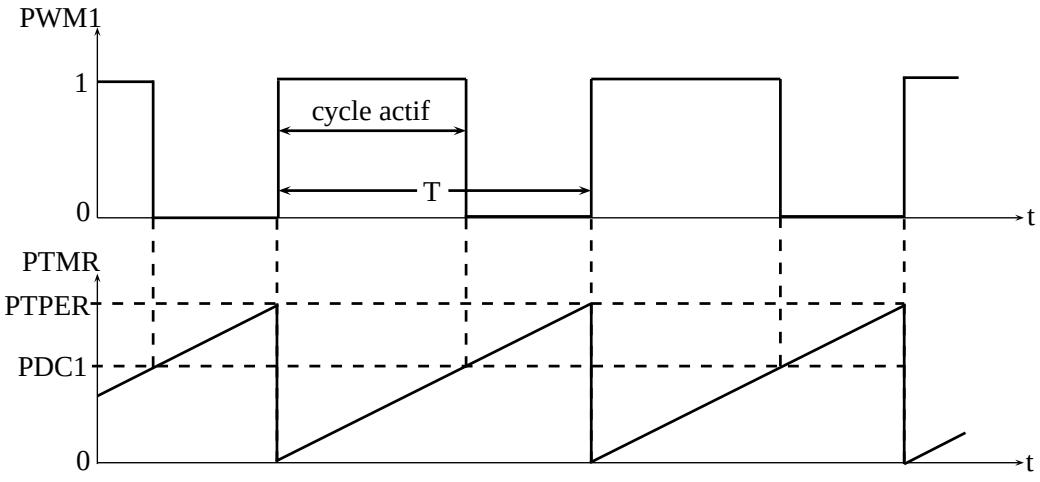

FIG. 2.8 Signal PWM

Dû au principe de fonctionnement du module de PWM, il existe un compromis à faire entre la fréquence du signal et la précision du cycle actif. En effet, plus la fréquence est basse, plus la valeur du registre PTPER est grande et donc plus on a de précision sur la valeur de PDCx. À l'inverse, si la fréquence est très élevée, la valeur du registre PTPER est basse et on a donc moins de précision sur la valeur de PDCx. Dans le cas de l'application du robot équilibriste, une valeur de PTPER a été choisie afin d'atteindre une fréquence d'environ $30\ KHz$, soit au-dessus de la plage de fréquences audibles se terminant autour de $20\ KHz$, tout en permettant aux valeurs du registre PDCx d'être codées sur 11 bits, soit entre 0 et 2047. Il est aussi à noter qu'en pratique, pour la commande des systèmes macroscopiques, il est souhaitable d'utiliser une précision d'au moins 10 bits pour les valeurs d'entrées et de sorties d'un système de commande (Franklin & Powell, 1997). Ce critère est donc également respecté par la configuration adoptée.

Le module PWM du microcontrôleur offre une multitude d'options pouvant être configurées par l'intermédiaire de registres de configurations. Pour une description détaillée sur les possibilités qu'offre ce module, le lecteur est référé au manuel de référence *dsPIC30F Family Reference Manual* (Microchip, 2006). Afin de faciliter la configuration du module PWM, des fonctions prévues à cet effet du compilateur C30 on été utilisées. Voici la

portion de code responsable de configurer le module PWM faisant appel à ces fonctions :

```
// Configuration PWM
OpenMCPWM(0x3FF, 0x0, PWM_EN & PWM_IDLE_CON & PWM_OP_SCALE1 & PWM_IPCLK_SCALE1 &
          PWM_MOD_FREE, PWM_MOD1_COMP & PWM_MOD2_COMP & PWM_PDIS3H & PWM_PEN2H &
          PWM_PEN1H & PWM_PDIS3L & PWM_PEN2L & PWM_PEN1L, PWM_SEVOPS1 &
          PWM_OSYNC_PWM & PWM_UEN);

// Configuration temps morts PWM
SetMCPWMDeadTimeGeneration(PWM_DTB0 & PWM_DTBPS1 & PWM_DTA10 & PWM_DTAPS1);

// Configuration fautes externes
SetMCPWMFaultA(PWM_FLTA1_DIS & PWM_FLTA2_DIS & PWM_FLTA3_DIS);
```

Une fois le module de PWM configuré, il suffit d'assigner une valeur au registre PDCx afin de faire varier le cycle actif d'un signal. Dans l'application du robot, les sorties PWM1 et PWM2 ont été utilisées afin de contrôler les deux moteurs du robot. Voici la portion de code mettant à jour les valeurs des registres appropriés :

```
PDC1 = ControlData.DutyCycleR; // Cycle actif moteur droit
PDC2 = ControlData.DutyCycleL; // Cycle actif moteur gauche
```

2.2.6.6 Interface de périphériques séquentielle (SPI)

L'interface de périphériques séquentiel (*Serial Peripheral Interface* - SPI) est un protocole de communications numériques permettant à plusieurs composantes électroniques de communiquer entre elles sur un bus. Il s'agit d'un protocole permettant des communications synchrones (dans les deux sens en même temps) ayant une topologie maître/esclave dans laquelle il existe un seul maître par système, celui-ci étant responsable de sélectionner l'esclave avec lequel communiquer et de générer un signal d'horloge pour cadencer les échanges de données.

Les différentes connections entre un maître et un esclave sur un bus SPI sont présentées à la FIG. 2.9. Le signal MOSI (*Master Out / Slave In*) est responsable d'acheminer les

données en provenance du maître vers l'esclave, le signal MISO (*Master In / Slave Out*) est responsable d'acheminer les données en provenance de l'esclave vers le maître, le signal SCLK constitue le signal d'horloge généré par le maître et $\overline{SS}$ (*Slave Select*) est le signal utilisé pour sélectionner l'esclave avec lequel on veut communiquer (actif à l'état bas).

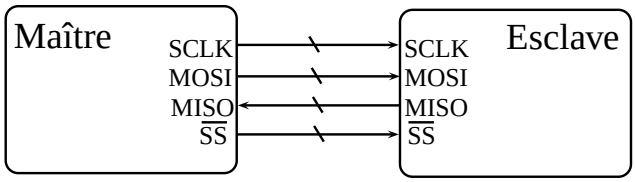

FIG. 2.9 Connections sur un bus SPI

Le microcontrôleur dsPIC30F4011 détient un module matériel implémentant ce protocole. Cette fonctionnalité est utilisée afin de communiquer avec l'accéléromètre et la composante de communication sans fil. Ce protocole est flexible et il existe une multitude de configurations possibles quant au nombre de bits par trame, la polarité du signal d'horloge (actif à l'état haut ou actif à l'état bas) et la synchronisation des échanges de données. Ces configurations sont réalisées au niveau des valeurs de plusieurs registres et des fonctions de la librairie du compilateur C30 facilitant la procédure ont été utilisées. Voici la portion de code utilisant ces fonctions :

```
ConfigIntSPI1(SPI_INT_DIS & SPI_INT_PRI_0);

OpenSPI1(FRAME_ENABLE_OFF & FRAME_SYNC_OUTPUT & ENABLE_SDO_PIN & SPI_MODE16_OFF &
         SPI_SMP_ON & SPI_CKE_ON & SLAVE_ENABLE_OFF & CLK_POL_ACTIVE_HIGH &
         MASTER_ENABLE_ON & SEC_PRESCAL_1_1 & PRI_PRESCAL_4_1, SPI_ENABLE &
         SPI_IDLE_CON & SPI_RX_OVFLOW_CLR);
```

Une fois le module configuré, des fonctions de la librairie du compilateur C30 permettent d'envoyer et de recevoir des données sur le bus SPI ainsi que de déterminer si le module a reçu une nouvelle donnée. Voici la séquence d'opérations utilisée afin d'initier un échange de données dans lequel une donnée est envoyée et une autre est reçue :

```
WriteSPI1(register_name); // Envoie de donnée
while(!DataRdySPI1());    // Attente de réception
in_byte = ReadSPI1();     // Réception de donnée
```

CHAPITRE 3

MODÉLISATION MATHÉMATIQUE DU ROBOT

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

Les valeurs des paramètres du moteur et de la boîte de réduction ont été obtenues à partir de leurs fiches techniques tandis que les valeurs des moments d'inertie du robot ont été estimés à partir du modèle de conception mécanique assisté par ordinateur dans le logiciel Catia et les poids des différentes composantes ont été obtenus en les pesant avec une balance électronique. Toutes les valeurs des paramètres sont présentées au TAB. 3.1.

Pour le développement du modèle mathématique représentant la dynamique du robot, certaines hypothèses doivent être faites. De façon générale, on considère que le robot se maintient autour de la position verticale, que ses roues restent en contact avec le sol en tout temps et qu'il se déplace à basse vitesse. Les forces de réactions entre le corps du robot et les roues ainsi que la force centrifuge due au mouvement d'inclinaison du robot sont également négligées. On considère que les moteurs appliquent des couples aux roues et sur le corps du robot simultanément et que les roues appliquent à leur tour des

TAB. 3.1 Paramètres du système

Symbole	Description	Valeur	Unité
r_w	rayon des roues	4.575	cm
M	masse de la moitié robot, y compris une roue	1.076	kg
m_b	masse de la moitié du corps du robot	1.054	kg
m_w	masse de l'une des roues	0.022	kg
J_b	moment d'inertie de la moitié du corps du robot	43.47	$kg \cdot cm^2$
J_d	moment d'inertie du robot autour de l'axe Z	108.21	$kg \cdot cm^2$
J_w	moment d'inertie de l'une des roues	0.04269	$kg \cdot cm^2$
d	distance entre l'arbre du moteur et le centre de gravité	6.19252	cm
S	distance entre les roues	22.9	cm
K_t	constante de couple des moteurs	0.0552	$\frac{N \cdot m}{A}$
K_e	constante de force électromotrice des moteurs	0.0552	$\frac{V \cdot s}{rad}$
r_a	résistance d'armature des moteurs	41.5	Ω
r_g	rapport de la boîte de réduction	6	N/A
η	efficacité de la boîte de réduction	80	$\%$
C_F	constante de friction	0.001	$\frac{N \cdot m \cdot s}{rad}$

forces sur le sol provoquant une accélération linéaire du robot ainsi qu'une accélération angulaire autour de l'axe vertical faisant varier l'angle d'orientation du robot. Ces hypothèses sont prises afin d'obtenir un modèle linéaire, bien qu'approximatif, relativement simple représentant bien la dynamique du système autour de la position d'équilibre. Le contrôleur conçu à la section 4.2 devra être suffisamment robuste afin d'assurer la stabilité du système malgré le fait que le modèle n'est qu'une approximation de la dynamique véritable du système.

Si les effets des forces de réactions entre le corps du robot et les roues ainsi que la force centrifuge due au mouvement d'inclinaison du robot étaient prises en considération lors du développement du modèle mathématique représentant la dynamique du robot, on se retrouverait avec un modèle plus complexe et contenant plus de termes non-linéaires devant être linéarisés. Cette approche a été utilisée par (Ooi R., 2003), (Kadir H., 2005) et (Abdollah M., 2006).

3.1 Inclinaison et déplacement linéaire

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

– la masse de la moitié du robot M consiste en la somme de la moitié de la masse du corps m_b et de la masse d'une roue m_w :

$$M = m_b + m_w$$

– la constante de couple K_t et la constante de la force électromotrice K_e sont équivalentes dans le système d'unités international

– la position linéaire du robot $x(t)$ peut être obtenue à partir du déplacement angulaire d'une roue $\theta(t)$ et le rayon d'une roue r_w de cette façon :

$$x(t) = r_w\theta(t) \tag{3.1}$$

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

$$\theta_i(t) = r_g\theta_o(t) \tag{3.2}$$

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

$$\theta_o(t) = \theta(t) + \psi(t) \tag{3.3}$$

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

$$F(t) = M\ddot{x}(t) \tag{3.4}$$

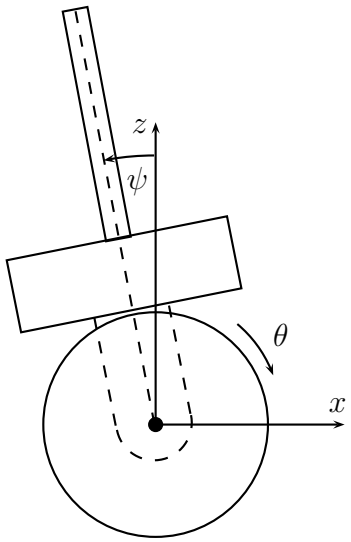

FIG. 3.1 Inclinaison et déplacement linéaire du robot

3.1.1 Dynamique d'un moteur

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

$$T_i(t) = K_t i(t). \tag{3.5}$$

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

$$i(t) = \frac{u_x(t)}{r_a} - \frac{K_e \dot{\theta}_i(t)}{r_a}. \tag{3.6}$$

On peut maintenant établir la relation entre le couple généré par le moteur et la tension appliquée à son entrée en combinant les équations (3.5) et (3.6) de cette façon :

$$T_i(t) = K_t \left[\frac{u_x(t)}{r_a} - \frac{K_e \dot{\theta}_i(t)}{r_a} \right] \tag{3.7}$$

3.1.2 Couple appliqué à une roue

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

$$T(t) = \eta r_g T_i(t)$$

En remplaçant $T_i(t)$ par la relation trouvée en (3.7), on obtient la relation suivante :

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

En utilisant la relation entre $\theta_i(t)$ et $\theta_o(t)$ donnée par l'équation (3.2), on peut exprimer l'équation précédente en fonction de $\theta_o(t)$ de cette façon :

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

Puis, en considérant la dérivée de l'équation (3.3) par rapport au temps, on obtient :

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

Finalement, à partir de l'équation (3.1), on obtient la relation donnant le couple appliqué à une roue $T(t)$ en fonction de la tension à l'entré d'un moteur $u_x(t)$, de la vitesse de déplacement linéaire du robot $\dot{x}(t)$ et de la dérivée par rapport au temps de son angle d'inclinaison $\dot{\psi}(t)$:

$$T(t) = \frac{\eta r_g K}{r_a}u_x(t) - \frac{\eta r_g^2 K^2}{r_a r_w}\dot{x}(t) - \frac{\eta r_g^2 K^2}{r_a}\dot{\psi}(t) \tag{3.8}$$

3.1.3 Dynamique d'inclinaison

En se rérérant à la FIG. 3.2, et en notant que le couple appliqué à une roue est également appliqué à la moitié du corps du robot, on peut faire le bilan des couples agissant sur la moité du corps du robot comme suit :

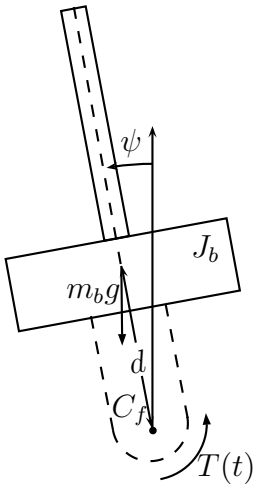

FIG. 3.2 Diagramme de corps libre de la dynamique d'inclinaison

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

En remplaçant $T(t)$ par la relation trouvée en (3.8), on obtient la relation suivante :

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

ce qui donne à son tour :

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

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

$$\ddot{\psi}(t) = \frac{\eta r_g K}{r_a J_b} u_x(t) + \frac{m_b g d}{J_b} \psi(t) - \left[\frac{\eta r_g^2 K^2 + C_f r_a}{r_a J_b}\right] \dot{\psi}(t) - \left[\frac{\eta r_g^2 K^2 + C_f r_a}{r_a r_w J_b}\right] \dot{x}(t) \tag{3.9}$$

3.1.4 Dynamique des roues et du déplacement linéaire

En se référant à la FIG. 3.3, on effectue le bilan des couples appliqués à l'une des roues comme suit :

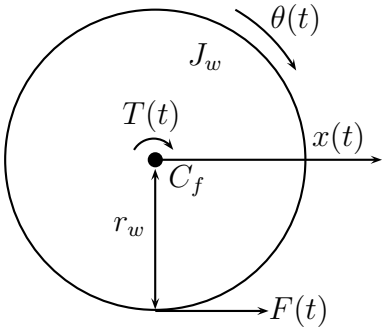

FIG. 3.3 Diagramme de corps libre d'une roue

$$J_w\ddot{\theta}(t) = T(t) - F(t)r_w - C_f\dot{\theta}(t) - C_f\dot{\psi}(t) \tag{3.10}$$

La force appliquée au sol par une roue $F(t)$, peut être remplacée par une relation équivalente selon l'équation (3.4) :

$$J_w\ddot{\theta}(t) = T(t) - r_wM\ddot{x}(t) - C_f\dot{\theta}(t) - C_f\dot{\psi}(t)$$

En remplaçant $T(t)$ par la relation trouvée en (3.8), on obtient la relation suivante :

$$J_w\ddot{\theta}(t) = \frac{\eta r_g K}{r_a}u_x(t) - \frac{\eta r_g^2 K^2}{r_a r_w}\dot{x}(t) - \frac{\eta r_g^2 K^2}{r_a}\dot{\psi}(t) - r_wM\ddot{x}(t) - \frac{C_f}{r_w}\dot{x}(t) - C_f\dot{\psi}(t)$$

Cette équation peut s'écrire sous la forme suivante :

$$\begin{aligned}\left[\frac{J_w}{r_w} + Mr_w\right]\ddot{x}(t) = \frac{\eta r_g K}{r_a}u_x(t) - \left[\frac{\eta r_g^2 K^2 + C_f r_a}{r_a}\right]\dot{\psi}(t) \\ - \left[\frac{\eta r_g^2 K^2 + C_f r_a}{r_a r_w}\right]\dot{x}(t)\end{aligned}$$

Ce qui donne à son tour :

$$\ddot{x}(t) = \left[\frac{\eta r_w r_g K}{r_a(J_w + Mr_w^2)}\right] u_x(t) - \left[\frac{\eta r_w r_g^2 K^2 + C_f r_w r_a}{r_a(J_w + Mr_w^2)}\right] \dot{\psi}(t) - \left[\frac{\eta r_g^2 K^2 + C_f r_a}{r_a(J_w + Mr_w^2)}\right] \dot{x}(t) \tag{3.11}$$

3.1.5 Modèle d'état de la dynamique linéaire

En définissant le vecteur d'état $\mathbf{x}(t) = \begin{bmatrix} \psi(t) & \dot{\psi}(t) & x(t) & \dot{x}(t) \end{bmatrix}^\top$ et le vecteur de sortie $\mathbf{y}(t) = \begin{bmatrix} \psi(t) & x(t) \end{bmatrix}^\top$, à partir des équations (3.9) et (3.11), on obtient le modèle d'état suivant :

$$\begin{cases} \dot{\mathbf{x}}(t) = A\mathbf{x}(t) + Bu_x(t) \\ \mathbf{y}(t) = C\mathbf{x}(t) \end{cases} \tag{3.12}$$

où

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 \\ \frac{m_b g d}{J_b} & -\frac{\eta r_g^2 K^2 + C_f r_a}{r_a J_b} & 0 & -\frac{\eta r_g^2 K^2 + C_f r_a}{r_a r_w J_b} \\ 0 & 0 & 0 & 1 \\ 0 & -\frac{\eta r_w r_g^2 K^2 + C_f r_w r_a}{r_a(J_w + Mr_w^2)} & 0 & \frac{\eta r_g^2 K^2 + C_f r_a}{r_a(J_w + Mr_w^2)} \end{bmatrix}$$

$$B = \begin{bmatrix} 0 \\ \frac{\eta r_g K}{r_a J_b} \\ 0 \\ \frac{\eta r_w r_g K}{r_a(J_w + Mr_w^2)} \end{bmatrix}, C = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

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

nominales suivantes :

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 147.2931 & -0.4864 & 0 & -10.6325 \\ 0 & 0 & 0 & 1 \\ 0 & -0.0429 & 0 & -0.9371 \end{bmatrix}$$

$$B = \begin{bmatrix} 0 \\ 1.4687 \\ 0 \\ 0.1295 \end{bmatrix}, C = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

En réalité, les valeurs des paramètres ne peuvent être connus avec exactitude. Aussi, celles-ci peuvent varier dans le temps à cause de certains phénomènes tel que l'usure des composantes et la décharge des batteries. De plus, le modèle a été établi sous certaines hypothèses, en négligeant certains phénomènes et a été approximé linéairement à un point de fonctionnement. Pour toutes ces raisons, le modèle contient des incertitudes et le contrôleur conçu à la section 4.2 devra être en mesure de stabiliser le système malgré leur présence.

3.2 Angle de direction

Le modèle représentant la dynamique de l'angle de direction du robot est établi en prenant en considération le fait que des couples égaux mais opposés doivent être appliqués par les deux moteurs de manière à engendrer un mouvement de rotation pure au robot sans affecter son inclinaison et sa position linéaire. Par conséquent, des tensions égales mais opposées doivent être appliquées aux deux moteurs et l'amplitude de ces tensions

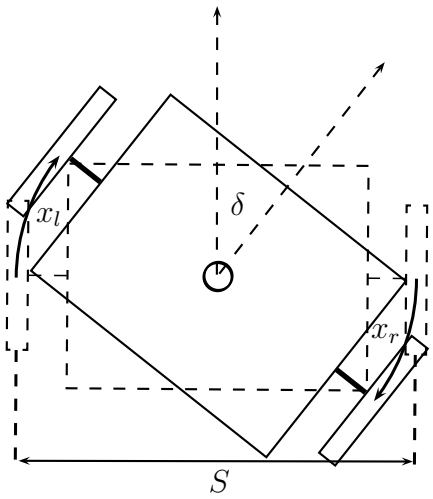

FIG. 3.4 Déplacements des roues lors d'un changement de direction

$u_h(t)$ est considérée comme étant l'entrée de ce sous-système. Ici, l'hypothèse que le robot se trouve dans le voisinage de la position verticale et que par conséquent son moment d'inertie autour de l'axe vertical peut être considéré comme une constante, est prise en considération. En se référant aux variables et aux paramètres définis dans la liste des notations et des symboles, on débute par observer le déplacement des roues produit par un changement d'angle de direction. En se référant à la FIG. 3.4, on peut voir que les distances parcourues par les deux roues $x_r(t)$ et $x_l(t)$ lorsque le robot tourne sur lui-même sont reliées à l'angle de direction $\delta(t)$ de la façon suivante :

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

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

$$\delta(t) = \left[\frac{x_l(t) - x_r(t)}{S}\right] \tag{3.13}$$

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

$$x_r(t) = r_w\theta_r(t) \tag{3.14}$$

$$x_l(t) = r_w\theta_l(t) \tag{3.15}$$

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

$$u_l(t) = -u_r(t) = u_h(t)$$

La différence entre les tensions appliquées aux moteurs peut être exprimée de la façon suivante :

$$u_l(t) - u_r(t) = 2u_h(t) \tag{3.16}$$

À partir de l'équation (3.10), on peut exprimer la force $F(t)$ appliquée par une roue sur le sol par la relation suivante :

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

Ensuite, à l'aide de l'équation (3.8), on peut exprimer cette dernière équation comme suit :

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

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

$$\begin{aligned} F_l(t) = {} & \frac{\eta r_g K}{r_a r_w}u_l(t) - \left[\frac{\eta r_g^2 K^2 + C_f r_a}{r_a r_w}\right]\dot{\psi}(t) \\ & - \left[\frac{\eta r_g^2 K^2 + C_f r_a}{r_a r_w^2}\right]\dot{x}_l(t) - \frac{J_w}{r_w}\ddot{\theta}_l(t) \end{aligned} \tag{3.17}$$

$$\begin{aligned} F_r(t) = {} & \frac{\eta r_g K}{r_a r_w}u_r(t) - \left[\frac{\eta r_g^2 K^2 + C_f r_a}{r_a r_w}\right]\dot{\psi}(t) \\ & - \left[\frac{\eta r_g^2 K^2 + C_f r_a}{r_a r_w^2}\right]\dot{x}_r(t) - \frac{J_w}{r_w}\ddot{\theta}_r(t) \end{aligned} \tag{3.18}$$

En se référant à la FIG. 3.5, on effectue le bilan des couples agissant sur le robot autour de l'axe vertical :

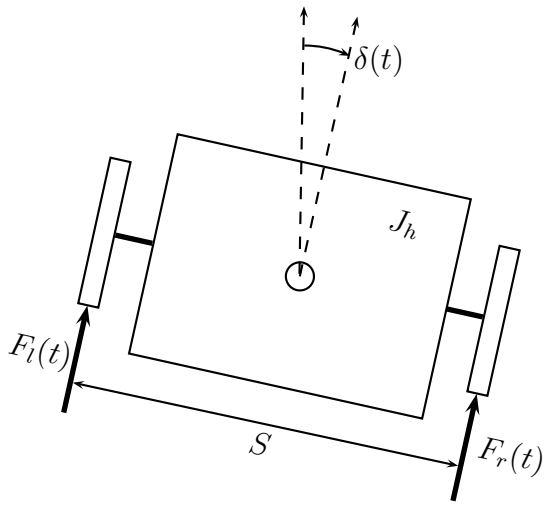

FIG. 3.5 Diagramme de corps libre de la dynamique d'angle de direction

$$J_d\ddot{\delta}(t) = [F_l(t) - F_r(t)]\,\frac{S}{2}$$

En remplaçant $F_l(t)$ et $F_r(t)$ par les expressions trouvées en (3.17) et (3.18), on établit la relation suivante :

$$\begin{aligned} J_d\ddot{\delta}(t) = {} & \frac{\eta r_g K S}{2 r_a r_w}\,[u_l(t) - u_r(t)] + \left[\frac{\eta S r_g^2 K^2 + S C_f r_a}{2 r_a r_w^2}\right][\dot{x}_r(t) - \dot{x}_l(t)] \\ & + \frac{J_w S}{2 r_w}\left[\ddot{\theta}_r(t) - \ddot{\theta}_l(t)\right] \end{aligned}$$

À partir de l'équation (3.16), on peut exprimer cette dernière équation en fonction de l'entrée $u_h(t)$ de la manière suivante :

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

En considérant la dérivée par rapport au temps de l'équation (3.13), on peut exprimer l'équation précédente en terme de la vitesse de changement de direction $\dot{\delta}(t)$ comme suit :

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

Les accélérations angulaires des roues $\ddot{\theta}_r(t)$ et $\ddot{\theta}_l(t)$ peuvent premièrement être exprimées en fonction des accélérations des roues en considérant la dérivée seconde par rapport au temps des équations (3.14) et (3.15) de cette façon :

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

Puis, ces accélérations des roues $\ddot{x}_r(t)$ et $\ddot{x}_l(t)$ peuvent être exprimées à leur tour en terme de l'accélération du changement de direction $\ddot{\delta}(t)$ en considérant la dérivée seconde par rapport au temps de l'équation (3.13) de la façon suivante :

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

ce qui donne :

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

Finalement, on obtient :

$$\ddot{\delta}(t) = \left[\frac{2\eta r_w r_g K S}{r_a(2 J_d r_w^2 + J_w S^2)}\right] u_h(t) - \left[\frac{\eta S^2 r_g^2 K^2 + S^2 C_f r_a}{r_a(2 J_d r_w^2 + J_w S^2)}\right]\dot{\delta}(t). \tag{3.19}$$

3.2.1 Modèle d'état de la dynamique de direction

En définissant le vecteur d'état $\mathbf{x}_h(t) = \begin{bmatrix} \delta(t) & \dot{\delta}(t) \end{bmatrix}^\top$ et le vecteur de sortie $\mathbf{y}_h(t) = \delta(t)$, à partir de l'équation (3.19), on obtient le modèle d'état suivant :

$$\begin{cases} \dot{\mathbf{x}}_h(t) = A_h \mathbf{x}_h(t) + B_h u_h(t) \\ \mathbf{y}_h(t) = C_h \mathbf{x}_h(t) \end{cases} \tag{3.20}$$

où

$$A_h = \begin{bmatrix} 0 & 1 \\ 0 & -\frac{\eta S^2 r_g^2 K^2 + S^2 C_f r_a}{r_a(2J_d r_w^2 + J_w S^2)} \end{bmatrix}$$
$$B_h = \begin{bmatrix} 0 \\ \frac{2\eta r_w r_g K S}{r_a(2J_d r_w^2 + J_w S^2)} \end{bmatrix}, C_h = \begin{bmatrix} 1 & 0 \end{bmatrix}$$

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

$$A_h = \begin{bmatrix} 0 & 1 \\ 0 & -2.4360 \end{bmatrix}$$
$$B_h = \begin{bmatrix} 0 \\ -2.9388 \end{bmatrix}, C_h = \begin{bmatrix} 1 & 0 \end{bmatrix}$$

La dynamique du robot équilibriste est alors décrite par (3.12) et (3.20). Ces modèles seront utilisés pour la conception d'algorithmes de commande par retour d'état en utilisant la méthode de placement de pôle.

3.3 Perturbations externes

Avant de passer à la conception des algorithmes de commande et de traitement de signaux, des perturbations externes doivent être considérées et être incluses dans les modèles mathématiques de la dynamique du robot afin d'avoir recours à la théorie H_∞ qui vise à minimiser leur impact sur le comportement dynamique du robot.

Le type de perturbation considérée sont des tensions additionnelles appliquées aux moteurs. On définit $\omega_x(t)$ comme étant une tension additionnelle appliquée aux deux moteurs affectant la dynamique linéaire du robot et $\omega_h(t)$ comme étant un écart de tension additionnel appliqué aux deux moteurs affectant la dynamique de direction. Ceux-ci produisent à leur tour des couples additionnels appliqués aux roues et au corps du robot par les moteurs venant perturber le système. Lorsque le robot est commandé à distance et que le contrôleur fait en sorte que le robot suit une trajectoire envoyée par un opérateur, des perturbations au niveau de la consigne en provenance de cet opérateur, tels que des délais causés par les transferts de données via Internet, sont éventuellement perçues comme des perturbations au niveau de la commande calculée par les contrôleurs et correspondent à des perturbations au niveau des tensions appliquées aux moteurs. De plus, cette façon de faire est appropriée puisqu'il s'agit d'un type de perturbation pouvant être produite de manière précise et réplicable lors de tests expérimentaux visant à valider l'efficacité et la performance du contrôleur.

Une fois que les perturbations externes ont été définies, il est maintenant nécessaire d'évaluer leur impact sur la dynamique du système.

3.3.1 Dynamique d'inclinaison et de déplacement linéaire

À partir de l'équation (3.9) et en considérant que la tension appliquée à un moteur affectant la dynamique linéaire du robot consiste maintenant en la somme de la tension provenant du contrôleur $u_x(t)$ et la perturbation externe $\omega_x(t)$, il est possible d'établir la relation suivante :

$$\ddot{\psi}(t) = \frac{\eta r_g K}{r_a J_b}\left[u_x(t) + \omega_x(t)\right] + \frac{m_b g d}{J_b}\psi(t) - \left[\frac{\eta r_g^2 K^2 + C_f r_a}{r_a J_b}\right]\dot{\psi}(t) - \left[\frac{\eta r_g^2 K^2 + C_f r_a}{r_a r_w J_b}\right]\dot{x}(t) \tag{3.21}$$

De façon similaire, pour la dynamique de déplacement linéaire, à partir de l'équation (3.11), on obtient :

$$\ddot{x}(t) = \left[\frac{\eta r_w r_g K}{r_a(J_w + M r_w^2)}\right]\left[u_x(t) + \omega_x(t)\right] - \left[\frac{\eta r_w r_g^2 K^2 + C_f r_w r_a}{r_a(J_w + M r_w^2)}\right]\dot{\psi}(t) - \left[\frac{\eta r_g^2 K^2 + C_f r_a}{r_a(J_w + M r_w^2)}\right]\dot{x}(t) \tag{3.22}$$

À partir des définitions de la section 3.1.5 et des équations (3.21) et (3.22), on peut représenter la dynamique d'inclinaison et de déplacement du robot soumis à une perturbation externe par le modèle d'état suivant :

$$\begin{cases} \dot{\mathbf{x}}(t) = A\mathbf{x}(t) + Bu_x(t) + B_\omega \omega_x(t) \\ \mathbf{y}(t) = C\mathbf{x}(t) \end{cases} \tag{3.23}$$

où les matrices A, B et C restent inchangées et $B_\omega = B$.

3.3.2 Dynamique de direction

Pour la dynamique de direction, à partir de l'équation (3.19) et en considérant que l'écart de la tension appliquée à un moteur par rapport à la moyenne consiste maintenant en la somme de la portion calculée par le contrôleur $u_h(t)$ et la perturbation externe $\omega_h(t)$, on obtient la relation suivante :

$$\ddot{\delta}(t) = \frac{2\eta r_w r_g K S}{r_a(2J_d r_w^2 + J_w S^2)} \left[u_h(t) + \omega_h(t)\right] - \left[\frac{\eta S^2 r_g^2 K^2 + S^2 C_f r_a}{r_a(2J_d r_w^2 + J_w S^2)}\right] \dot{\delta}(t) \quad (3.24)$$

À partir des définitions de la section 3.2.1 et de l'équation (3.24), on peut représenter la dynamique de direction du robot soumis à une perturbation externe par le modèle d'état suivant :

$$\begin{cases} \dot{\mathbf{x}}_h(t) = A_h \mathbf{x}_h(t) + B_h u_h(t) + B_{\omega h} \omega_h(t) \\ \mathbf{y}_h(t) = C_h \mathbf{x}_h(t) \end{cases} \quad (3.25)$$

où les matrices A_h, B_h et C_h restent inchangées et $B_{\omega h} = B_h$.

Les modèles d'états représentant la dynamique du robot lorsque soumis à des perturbations externes seront utilisés lors de la conception d'un contrôleur par retour d'état faisant appel à la théorie H_∞. Mais d'abord, puisqu'on ne dispose pas d'un accès direct aux variables d'états, des algorithmes sont nécessaires afin d'obtenir un bon estimé du vecteur d'état à partir des signaux disponibles provenant des capteurs. De plus, la plupart de ces signaux sont bruités et des filtres appropriés sont donc nécessaires. Ces aspects seront examinés dans le prochain chapitre.

CHAPITRE 4

COMMANDE EN TEMPS RÉEL

Puisque le système sous étude est un système instable en boucle ouverte, il est nécessaire de concevoir un contrôleur afin de le stabiliser et éventuellement pouvoir le commander à distance. Les algorithmes de commande développés reposent sur la technique de commande par retour d'état. Cette technique permet de stabiliser le système tout en lui assignant une certaine performance pouvant être caractérisée, entre autres, en termes de réponse temporelle et de sensibilité aux perturbations externes. Plusieurs techniques existent pour déterminer les gains appropriés d'un contrôleur par retour d'état. Dans le cadre de ce projet, deux méthodes ont été utilisées ; la méthode par placement de pôles reposant sur l'assignation d'une performance temporelle et la méthode H_∞ visant à minimiser la sensibilité du système à des perturbations externes. Aussi, la technique de commande par retour d'état ne garantit pas une erreur nulle en régime permanent. Pour remédier à cette situation, des actions intégrales sur les erreurs en position linéaire et en angle de direction sont ajoutées afin d'annuler ces erreurs en régime permanent.

Le type de contrôleur choisi requiert d'avoir accès aux valeurs des états du système tels que définis dans les modèles mathématiques développés au chapitre 3. Malheureusement, on ne dispose pas de capteurs capables de mesurer ces états directement. Il est donc nécessaire de développer des algorithmes permettant d'obtenir des estimés fiables de ces états à partir des signaux des capteurs, et ce malgré la présence de bruits dans ces signaux.

Une fois que les contrôleurs par retour d'état sont mis en place pour permettre de commander le robot en déplacement linéaire et en angle de direction tout en le stabilisant en maintenant son angle d'inclinaison au voisinage de la verticale, il est souhaitable d'ajou-

ter un contrôleur permettant à celui-ci de suivre une trajectoire envoyée par un opérateur à travers une architecture de communication. Des algorithmes permettant d'annuler l'écart entre la position réelle du robot et la position désirée de sa trajectoire seront donc également développés.

Finalement, une fois les différents algorithmes de contrôle et de traitement de signaux développés analytiquement dans le domaine continu, ceux-ci doivent être implémentés de façon discrète dans le microcontrôleur. Tous ces aspects seront présentés en détail dans ce chapitre.

4.1 Filtrage

En raison de la nature du système et de la sélection des capteurs, on n'a pas directement accès aux variables d'états. Seul les signaux $\psi_a(t)$, $\dot{\psi}_g(t)$, $\theta_{ir}(t)$ et $\theta_{il}(t)$ sont disponibles directement à partir de l'accéléromètre, du gyroscope et des encodeurs optiques respectivement. Les variables d'états doivent donc être déterminées à partir de ces signaux de sortie. Il est également important de noter que les capteurs utilisés pour mesurer l'angle d'inclinaison ont quelques inconvénients et il faut donc avoir recourt à des techniques de traitement de signaux afin d'obtenir un estimé fiable. Aussi, en raison de l'usage des boîtes de réduction, du bruit est introduit dans les mesures de la position linéaire et de l'angle de direction obtenues à partir des signaux des encodeurs optiques. Il est donc souhaitable de traiter ces signaux également avant de les utiliser pour la commande du robot en boucle fermée.

Dans le cadre de ce projet, la technique des filtres complémentaires a été choisie afin d'obtenir un estimé de l'inclinaison du robot. Tandis que des filtres passe-bas ont été utilisés afin de traiter les signaux provenant des encodeurs optiques. Le différents algorithmes utilisés afin d'obtenir de bons estimés des états sont présentés à la FIG. 4.1 et les

divers filtres conçus sont présentés en détail dans les sections qui suivent.

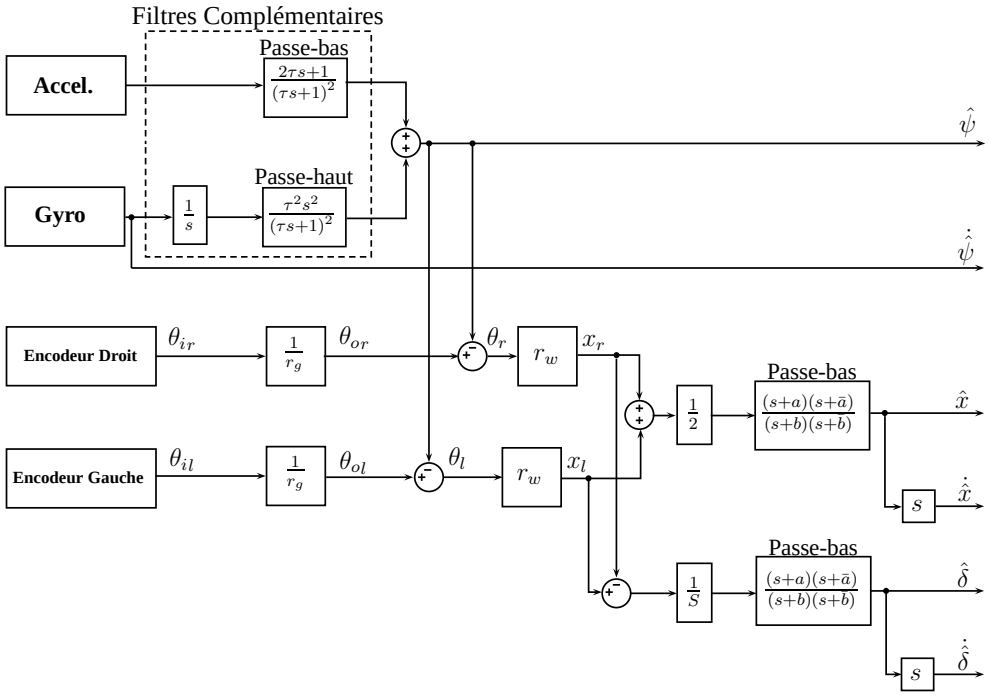

FIG. 4.1 Traitement de signaux et estimation des états

4.1.1 Angle d'inclinaison

Des estimés de l'angle d'inclinaison peuvent être obtenus de deux façons différentes à partir du gyroscope et de l'accéléromètre installés sur la plate-forme située au-dessus des moteurs du robot. Premièrement, puisque le gyroscope produit un signal proportionnel à la vitesse de rotation, comme mentionné à la section 2.2.2.1, il est possible d'intégrer ce signal afin d'obtenir un estimé de l'angle d'inclinaison. Deuxièmement, il est également possible d'obtenir un estimé à l'aide de la trigonométrie inverse à partir des signaux de l'accéléromètre afin de déterminer dans quelle direction la gravité agit sur celui-ci (voir FIG. 4.2).

4.1.1.1 Estimé du gyroscope

Pour le gyroscope, on a :

$$\psi_g = \int \dot{\psi}_g \, dt \tag{4.1}$$

où $\dot{\psi}_g$ est le signal à la sortie du gyroscope.

Malheureusement, il y a une erreur dans le signal $\dot{\psi}_g$ du gyroscope. La moyenne de cette erreur est non-nulle en raison de la sensibilité à la température ambiante et, en conséquence, l'estimé ψ_g obtenu à l'aide du gyroscope (4.1) a une erreur qui grandit avec le temps. Cet effet est connu sous le nom de dérive. Cette dérive peut être considérée comme un bruit à basse fréquence relativement aux bruits présents dans le signal de l'accéléromètre. Un filtre passe-haut est donc nécessaire pour se débarrasser de celui-ci. La méthode utilisée afin de déterminer la fréquence de coupure appropriée pour ce filtre sera présentée à la fin de cette section.

4.1.1.2 Estimé de l'accéléromètre

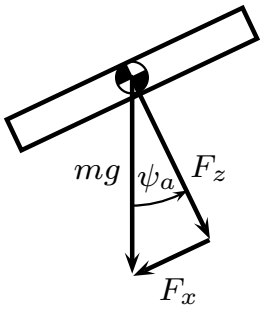

FIG. 4.2 Accéléromètre utilisé en tant qu'inclinomètre

À partir de la FIG. 4.2, on peut voir que :

$$\tan \psi_a = \frac{F_x}{F_z} \text{ ce qui donne } \psi_a = \tan^{-1} \frac{F_x}{F_z} \tag{4.2}$$

Malheureusement, les signaux F_x et F_z de l'accéléromètre contiennent du bruit dû aux vibrations qui sont détectés par l'accéléromètre. De plus, l'accéléromètre ne détecte pas seulement l'effet de la gravité, mais aussi les forces d'inertie qui sont présentes lorsque le robot est en mouvement. Par conséquent, l'estimé ψ_a obtenu à partir de l'accéléromètre (4.2) n'est valable que lorsque le robot est statique. Tous les bruits mentionnés ci-dessus peuvent être considérés comme des bruits à haute fréquence relativement au bruit présent dans le signal du gyroscope. Un filtre passe-bas est donc nécessaire afin de les éliminer.

4.1.1.3 Filtres complémentaires

Le concept de filtres complémentaires, tel qu'expliqué par (Baerveldt & Klang, 1997), consiste à sélectionner un ensemble de filtres passe-haut et passe-bas appliqués aux signaux des estimés obtenus à partir du gyroscope et de l'accéléromètre, respectivement, nous permettant d'obtenir un bon estimé de l'angle d'inclinaison en additionnant leurs sorties. Afin d'obtenir cet effet désiré, ces filtres doivent avoir la même fréquence de coupure et la somme de leurs fonctions de transfert doit être égale à 1, ce qui signifie une bande passante théoriquement infinie :

$$G_a(s) + G_g(s) = 1 \tag{4.3}$$

où $G_a(s)$ est la fonction de transfert du filtre passe-bas utilisé sur l'estimé obtenu à partir de l'accéléromètre et $G_g(s)$ est la fonction de transfert du filtre passe-haut utilisé sur l'estimé obtenu à partir du gyroscope.

En utilisant un ensemble de filtres ayant les formes suivantes :

$$G_a(s) = \frac{\bar{\psi}_a(s)}{\psi_a(s)} = \frac{2\tau s + 1}{(\tau s + 1)^2}$$
$$G_g(s) = \frac{\bar{\psi}_g(s)}{\psi_g(s)} = \frac{\tau^2 s^2}{(\tau s + 1)^2}$$

où τ est la constante de temps des deux filtres reliée à leur fréquence de coupure, $\bar{\psi}_a(t)$ et $\bar{\psi}_g(t)$ sont les estimés de l'angle d'inclinaison obtenus respectivement à partir de l'accéléromètre et du gyroscope une fois filtré. Il est clair que, peut importe la valeur de τ, les filtres sélectionnés satisfont le critère de largeur de bande passante infinie (4.3). En effet, on a :

$$G_a(s) + G_g(s) = \frac{2\tau s + 1}{(\tau s + 1)^2} + \frac{\tau^2 s^2}{(\tau s + 1)^2} = \frac{\tau^2 s^2 + 2\tau s + 1}{\tau^2 s^2 + 2\tau s + 1} = 1$$

Les diagrammes de Bode de ces filtres sont démontrés aux FIG. 4.3 et 4.4.

Graphiquement, le diagramme de Bode de la somme de ces filtres montre que le gain est unitaire pour toutes fréquences et la phase est égale à zéro pour toutes fréquences. En conséquence, on obtient un très bon estimé de l'angle d'inclinaison $\hat{\psi}(t)$ en additionnant les sorties des deux filtres $\bar{\psi}_a(t)$ et $\bar{\psi}_g(t)$. Les diagramme de Bode de chacun des filtres ainsi que celui de leur somme sont montrés sur le même graphique à la FIG. 4.5.

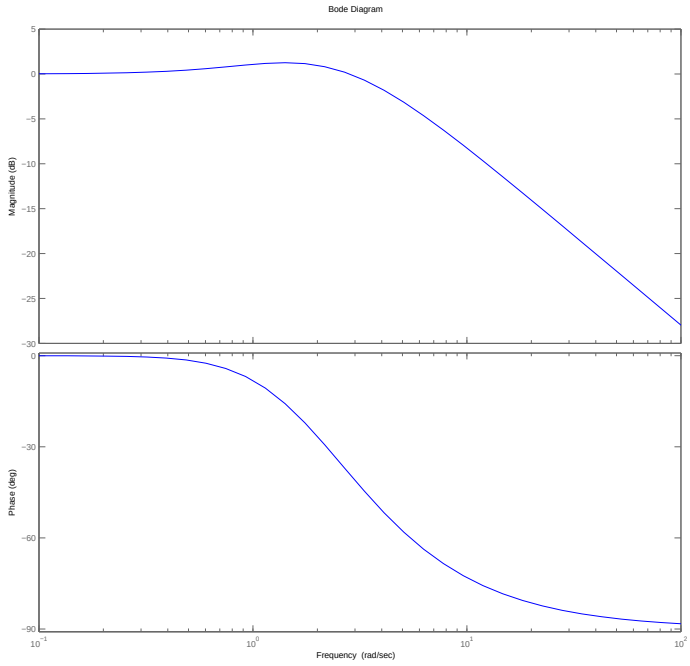

FIG. 4.3 Diagramme de Bode du filtre passe-bas $G_a(s)$

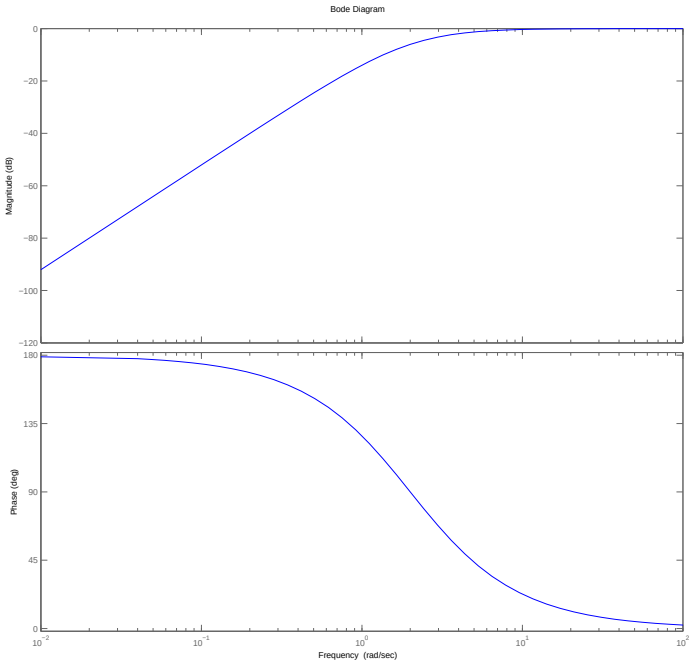

FIG. 4.4 Diagramme de Bode du filtre passe-haut $G_g(s)$

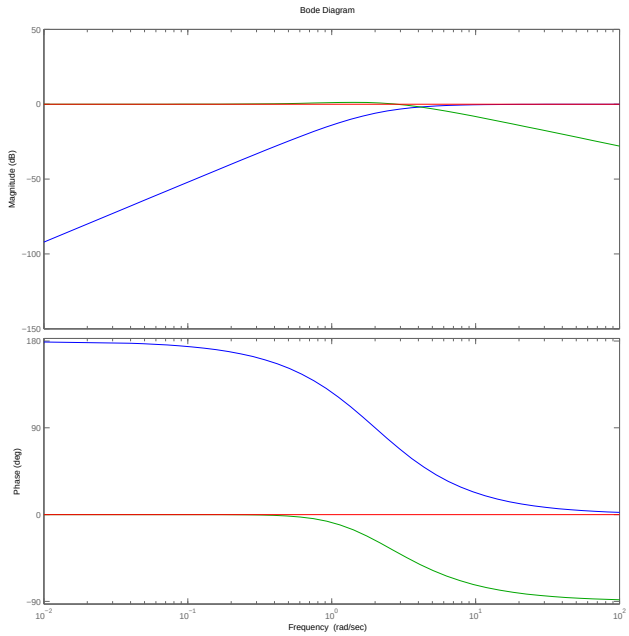

FIG. 4.5 Diagrammes de Bode des filtres complémentaires $G_a(s)$ et $G_g(s)$

Puisque l'estimé obtenu à partir de l'accéléromètre est valide uniquement lorsque le robot n'est pas en mouvement, il est préférable d'utiliser l'estimé du gyroscope autant que possible. Par conséquent, il est souhaitable d'utiliser une fréquence de coupure aussi basse que possible mais suffisamment élevée pour éliminer l'effet de dérive. En tant que telle, la constante de temps τ a été déterminée expérimentalement, en commençant avec une valeur élevée et en l'abaissant jusqu'à ce que le filtre passe-haut du gyroscope élimine effectivement la dérive. La valeur de τ déterminée de cette façon et utilisée dans l'implémentation est de $0,5\ s$.

Donc, numériquement, les fonctions de transfert des filtres utilisés sont :

$$G_a(s) = \frac{s+1}{0.25s^2 + s + 1}$$

$$G_g(s) = \frac{0.25s^2}{0.25s^2 + s + 1}$$

4.1.2 Filtres pour positions des roues

En raison de l'utilisation des boîtes de réduction, lorsqu'un moteur change de direction les dents d'engrenage perdent le contact et l'encodeur optique détecte des vitesses élevées qui ne reflètent pas la dynamique réelle du robot. Il est donc nécessaire de recourir à des filtres passe-bas en vue d'éliminer des mesures de la position linéaire $x(t)$ et de l'angle de direction $\delta(t)$ ce bruit de haute fréquence provenant de la position des roues $x_r(t)$ et $x_l(t)$ obtenus à partir des encodeurs optiques, en particulier avant de prendre des dérivées par rapport au temps.

De nombreux types de filtres passe-bas existent, mais des filtres Butterworth de second ordre ont été choisis, car ils offrent une réponse presque parfaitement plate sur la bande passante. La forme générale du filtre Butterworth de second ordre est composée de deux pôles complexes et deux zéros complexes :

$$G_x(s) = \frac{(s+a)(s+\bar{a})}{(s+b)(s+\bar{b})}$$

La fréquence de coupure de ces filtres a été choisie en tenant compte de la dynamique en boucle ouverte du système. Par conséquent, elle a été choisie suffisamment élevée afin de conserver toutes les informations significatives dans le signal tout en étant suffisamment faible pour éliminer le bruit introduit par les boîtes de réduction. À ce titre, elle a été sélectionnée comme étant 4 fois plus élevée que la fréquence associée au pôle le plus rapide de la dynamique en boucle ouverte, obtenue à partir des valeurs propres des matrices A et A_h des modèles d'états de l'angle d'inclinaison et du déplacement linéaire

et de l'angle de direction, respectivement.

4.2 Correcteur

Afin de stabiliser le système et de lui faire suivre une trajectoire désirée, on doit concevoir des lois de commande appropriées. Dans cette section, un correcteur par retour d'état pour l'inclinaison et le déplacement linéaire, un correcteur par retour d'état pour l'angle de direction et un correcteur proportionnel et dérivé (PD) pour la poursuite de trajectoire sont conçus. De plus, plusieurs techniques permettant de calculer la valeur des gains appropriés de ces correcteurs sont proposés.

4.2.1 Références et intégrateurs

Afin d'assurer une erreur nulle en régime permanent pour la position linéaire et l'angle de direction lors d'une poursuite de trajectoire, des intégrateurs sont ajoutés. Pour ce faire, chacun des modèles d'états développés est augmenté en ajoutant un état supplémentaire consistant en l'intégrale de l'erreur en position en utilisant la procédure décrite par (Boukas, 1995).

– Pour la dynamique de la position linéaire, on définit une référence en vitesse $V_r(t)$ indiquant la vitesse voulue du robot. À partir de cette référence en vitesse, il est également possible de définir la position de référence $x_r(t)$ comme suit :

$$x_r(t) = \int V_r(t)\, dt$$

Ensuite, selon la référence en position $x_r(t)$, on définit l'erreur $e_x(t)$ et un état additionnel $\tilde{x}(t)$ comme suit :

$$
\begin{aligned}
e_x(t) &= x_r(t) - x(t) \\
\tilde{x}(t) &= \int e_x(t)\, dt \\
\dot{\tilde{x}}(t) &= e_x(t)
\end{aligned}
$$

La dynamique du système augmenté devient alors :

$$
\begin{cases}
\dot{\tilde{\mathbf{x}}}(t) = \tilde{A}\tilde{\mathbf{x}}(t) + \tilde{B}u_x(t) + \tilde{B}_\omega\omega_x(t) + \tilde{E}x_r(t) \\
\tilde{\mathbf{y}}(t) = \mathbf{y}(t) = \tilde{C}\tilde{\mathbf{x}}(t)
\end{cases}
\tag{4.4}
$$

où

$$
\begin{aligned}
\tilde{\mathbf{x}}(t) &= \begin{bmatrix} \mathbf{x}^\top(t) & \tilde{x}(t) \end{bmatrix}^\top, \\
\tilde{A} &= \begin{bmatrix} A & 0 \\ -C_{xi} & 0 \end{bmatrix}, \\
\tilde{B} &= \begin{bmatrix} B^\top & 0 \end{bmatrix}^\top, \\
\tilde{B}_\omega &= \tilde{B}, \\
\tilde{C} &= \begin{bmatrix} C & 0 \end{bmatrix}, \\
\tilde{E} &= \begin{bmatrix} 0 & 0 & 0 & 0 & 1 \end{bmatrix}^\top, \\
C_{xi} &= \begin{bmatrix} 0 & 0 & 1 & 0 \end{bmatrix}.
\end{aligned}
$$

On peut facilement voir que le système est instable, mais contrôlable et observable. Par conséquent, le système peut être stabilisé par un correcteur à retour d'état. La loi de commande utilisée pour effectuer le suivi d'une vitesse et position linéaires utilise

l'action intégrale et l'action directe de la référence et prend donc la forme suivante :

$$u_x(t) = K\left(\mathbf{r}(t) - \mathbf{x}(t)\right) + K_{xi}\tilde{x}(t) \tag{4.5}$$

où

$$\mathbf{r}(t) = \begin{bmatrix} 0 & 0 & x_r(t) & V_r(t) \end{bmatrix}^\top,$$
$$K = \begin{bmatrix} K_\psi & K_{\dot{\psi}} & K_x & K_v \end{bmatrix}$$

On peut exprimer cette loi de commande de la façon suivante :

$$u_x(t) = K\mathbf{r}(t) - \tilde{K}\tilde{\mathbf{x}}(t)$$
$$\text{où } \tilde{K} = \begin{bmatrix} K & -K_{xi} \end{bmatrix}$$

– Pour la dynamique de l'angle de direction, de manière semblable, on débute par introduire la référence en vitesse de rotation $\omega_r(t)$ indiquant la vitesse de changement de direction voulue du robot. Puis, la référence en angle de direction $\delta_r(t)$ peut être définie de cette façon :

$$\delta_r(t) = \int \omega_r(t)\, dt$$

Selon la référence en angle de direction $\delta_r(t)$, on définit l'erreur $e_h(t)$ et le nouvel état $\tilde{\delta}(t)$ comme suit :

$$
\begin{aligned}
e_h(t) &= \delta_r(t) - \delta(t) \\
\tilde{\delta}(t) &= \int e_h(t)\, dt \\
\dot{\tilde{\delta}}(t) &= e_h(t)
\end{aligned}
$$

La dynamique augmentée du système devient donc :

$$
\begin{cases}
\dot{\tilde{\mathbf{x}}}_h(t) = \tilde{A}_h \tilde{\mathbf{x}}_h(t) + \tilde{B}_h u_h(t) + \tilde{B}_{\omega h} \omega_h(t) + \tilde{E}_h \delta_r(t) \\
\tilde{\mathbf{y}}_h(t) = \mathbf{y}_h(t) = \tilde{C}_h \tilde{\mathbf{x}}_h
\end{cases}
\tag{4.6}
$$

où

$$
\begin{aligned}
\tilde{\mathbf{x}}_h(t) &= \begin{bmatrix} \mathbf{x}_h^\top(t) & \tilde{\delta}(t) \end{bmatrix}^\top, \\
\tilde{A}_h &= \begin{bmatrix} A_h & 0 \\ -C_{\delta i} & 0 \end{bmatrix}, \\
\tilde{B}_h &= \begin{bmatrix} B_h^\top & 0 \end{bmatrix}^\top, \\
\tilde{B}_{\omega h} &= \tilde{B}_h, \\
\tilde{C}_h &= \begin{bmatrix} C_h & 0 \end{bmatrix}, \\
\tilde{E}_h &= \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^\top, \\
C_{\delta i} &= \begin{bmatrix} 1 & 0 \end{bmatrix}.
\end{aligned}
$$

L'algorithme de commande utilisé pour le suivi de la vitesse et position angulaire de la direction utilise l'action intégrale et l'action directe de la référence et adopte alors la forme suivante :

$$u_h(t) = K_h \left(\mathbf{r}_h(t) - \mathbf{x}_h(t)\right) + K_{\delta i}\tilde{\delta}(t) \tag{4.7}$$

où

$$\mathbf{r}_h(t) = \begin{bmatrix} \delta_r(t) & \omega_r(t) \end{bmatrix}^\top,$$
$$K_h = \begin{bmatrix} K_\delta & K_\omega \end{bmatrix}$$

On peut exprimer cette loi de commande de la façon suivante :

$$u_h(t) = K\mathbf{r}(t) - \tilde{K}_h\tilde{\mathbf{x}}_h(t)$$
$$\text{où } \tilde{K}_h = \begin{bmatrix} K_h & -K_{\delta i} \end{bmatrix}$$

4.2.2 Structure du correcteur

Les tensions d'entrée $u_r(t)$ et $u_l(t)$ à envoyer aux deux moteurs à courant continu sont calculées en combinant les résultats obtenus à partir des correcteurs de la dynamique linéaire et de la dynamique de l'angle de direction de la façon suivante :

$$u_r(t) = u_x(t) - u_h(t)$$
$$u_l(t) = u_x(t) + u_h(t)$$

de cette façon, on a :

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

tel que défini précédemment.

La structure complète du correcteur est présentée à la FIG. 4.6.

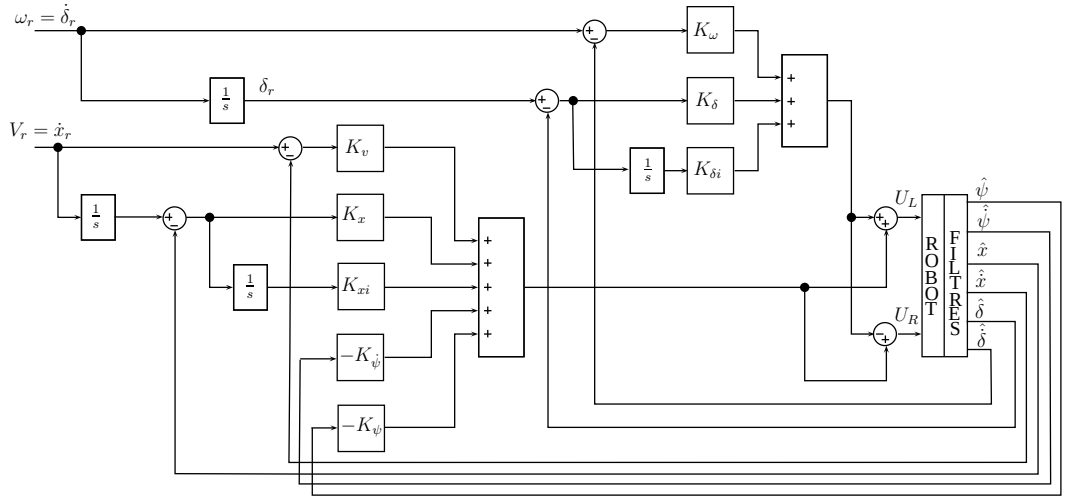

FIG. 4.6 Structure globale du correcteur par retour d'état

4.2.3 Placement de pôles

Maintenant qu'on a établi les lois de commandes permettant d'assurer la stabilité du système et de permettre la poursuite de référence, il est maintenant nécessaire de déterminer la valeur des différents gains du correcteur.

L'une des techniques élémentaires utilisée afin de calculer la valeur des gains $\tilde{K}$ et $\tilde{K}_h$ est la technique par placement de pôles.

4.2.3.1 Dynamique linéaire

Le modèle d'état augmenté de la dynamique de l'angle d'inclinaison et du déplacement linéaire (4.4) est de l'ordre 5. Donc, afin de déterminer le gain $\tilde{K}$ approprié, il est nécessaire de choisir les emplacements désirés de 5 pôles. En choisissant les critères de performance suivants :

– un système stable
– un temps de réponse à 5% de $2.0s$
– un dépassement de l'ordre de 5%.

Il est possible de choisir l'emplacement voulu de deux pôles dominants selon les formules suivantes :

$$D = e^{-\pi\zeta\sqrt{1-\zeta^2}} \approx 0.05 \rightarrow \zeta = \frac{\sqrt{2}}{2}$$
$$t_{s5\%} = \frac{3}{\zeta\omega_n} = 2\ s \rightarrow \omega_n = \frac{3}{t_{s5\%}\zeta} = 2.1213\ rad/s$$
$$p_{1,2} = -\zeta\omega_n \pm j\omega_n\sqrt{1-\zeta^2} = -1.5 \pm 1.5j$$

Les trois autres pôles doivent être dominés par ces deux pôles afin que la dynamique du système soit approximée par ces derniers. Deux de ceux-ci sont donc sélectionnés comme étant une paire de pôles complexes ayant un facteur d'amortissement ζ presque égal à 1 afin de réduire au minimum toute oscillation indésirable et étant 6 fois plus rapide que les pôles dominants. Tandis que le dernier pôle est choisi comme étant un pôle réel dont la valeur est égal à la partie réelle des deux pôles complexes dominés :

$$p_{3,4} = -9 \pm 0.0001j$$
$$p_5 = -9$$

En égalisant les pôles en boucle fermée voulus à l'équation caractéristique de la façon suivante :

$$\left| sI - \tilde{A} + \tilde{B}\tilde{K} \right| = (s - p_1)(s - p_2)(s - p_3)(s - p_4)(s - p_5)$$

et en utilisant une technique de résolution appropriée tel que Ackerman, on obtient les gains suivants :

$$\tilde{K} = \begin{bmatrix} 339.1161 & 27.4991 & -172.0486 & -91.2456 & 172.0486 \end{bmatrix}$$

4.2.3.2 Dynamique de direction

Le modèle d'état représentant la dynamique augmentée de l'angle de direction est d'ordre 3. On doit donc choisir l'emplacement de 3 pôles afin de déterminer le gain $\tilde{K}_h$ approprié. En raison de l'hypothèse selon laquelle la dynamique d'angle de direction soit découplée de la dynamique de déplacement linéaire, il est souhaitable de la rendre suffisamment plus rapide que la dynamique de l'angle d'inclinaison et du déplacement linéaire. Par conséquent, les pôles sont sélectionnés comme étant une paire de pôles dominants complexes avec un facteur d'amortissement presque égal à 1 et étant 6 fois plus rapide

que les pôles dominants de la dynamique de déplacement linéaire et un pôle réel 4 fois plus rapide que ces pôles complexes dominants :

$$p_{h_{1,2}} = -9 \pm 0.0001j$$
$$p_{h_3} = -36$$

En posant l'égalité entre les pôles voulus en boucle fermée et l'équation caractéristique de la façon suivante :

$$\left| sI - \tilde{A}_h + \tilde{B}_h \tilde{K}_h \right| = (s - p_{h_1})(s - p_{h_2})(s - p_{h_3})$$

il est possible d'utiliser une technique de résolution appropriée tel que Ackerman pour obtenir les gains suivants :

$$\tilde{K}_h = \begin{bmatrix} 248.0618 & 17.5460 & -992.2472 \end{bmatrix}$$

Les pôles de la dynamique en boucle fermée sont montrés à la FIG. 4.7.

4.2.4 $\mathbf{H}_\infty$

Une autre technique plus avancée permettant de calculer la valeur des gains $\tilde{K}$ et $\tilde{K}_h$ est la technique basée sur la théorie H_∞. Le principe de cette théorie est de minimiser la

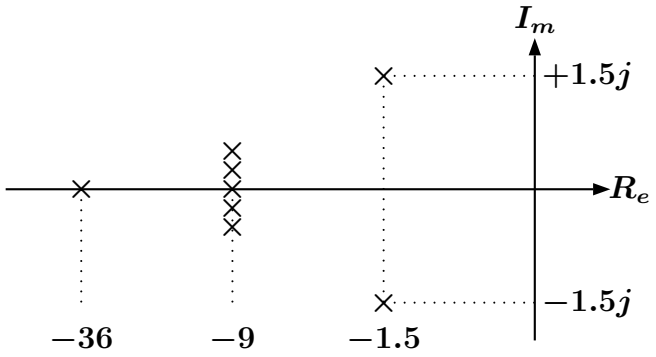

FIG. 4.7 Emplacements des pôles en boucle fermée par placement de pôles

sensibilité du système aux perturbations externes. De façon générale, la dynamique d'un système soumis à des perturbations externes peut être représentée par un modèle d'état ayant cette forme :

$$\begin{cases} \dot{\mathbf{x}}(t) = A\mathbf{x}(t) + Bu(t) + B_\omega\omega(t), \mathbf{x}(0) = x_0, \\ \mathbf{y}(t) = C\mathbf{x}(t) \end{cases} \tag{4.8}$$

où $\mathbf{x}(t) \in \mathbb{R}^n$ est le vecteur d'état, $u(t) \in \mathbb{R}^m$ est le vecteur de commande, $\mathbf{y}(t) \in \mathbb{R}^p$ est le vecteur de sorties contrôlées, $\omega(t) \in \mathbb{R}^l$ est le vecteur de perturbations externes du système, et A, B, B_ω, C sont des matrices réelles ayant les dimensions appropriées.

Lors de la conception de ce contrôleur, la dynamique du robot sera représentée par les modèles d'états (4.4) et (4.6) incluant les intégrateurs et les perturbations.

Afin d'avoir recours à la théorie de H_∞, les perturbations $\omega_x(t)$ et $\omega_h(t)$ doivent appartenir à $\mathscr{L}_2[0, \infty)$, ce qui veut dire que les inégalités suivantes sont satisfaites :

$$\int_0^\infty \omega_x^\top(t)\omega_x(t)\, dt < \infty$$

$$\int_0^\infty \omega_h^\top(t)\omega_h(t)\, dt < \infty$$

Ceci implique que les perturbations ont une énergie finie, ce qui en pratique est toujours le cas.

Mathématiquement, on est intéressés à la conception d'un contrôleur qui garantit l'inégalité suivante pour tout $\omega \in \mathscr{L}_2[0,\infty)$:

$$\|\mathbf{y}(t)\|_2 < \gamma \left[\|\omega(t)\|_2^2 + M(x_0)\right]^{\frac{1}{2}},$$

où $\gamma > 0$ est un niveau de rejet de perturbation à atteindre, x_0 sont les conditions initiales du vecteur d'état et $M(x_0)$ est une constante qui dépend de ces conditions initiales.

Le but du contrôle H_∞ est de trouver un contrôleur par retour d'état qui minimise la norme H_∞ de la matrice de transfert du système en boucle fermée entre la sortie contrôlée $\mathbf{y}(t)$ et les perturbations externes $\omega(t)$ appartenant à $\mathscr{L}_2[0,T]$. Autrement dit, mathématiquement :

$$\|G_{yw}\|_\infty = \sup_{\|\omega(t)\|_{2,[0,T]} \neq 0} \frac{\|\mathbf{y}(t)\|_{2,[0,T]}}{\|\omega(t)\|_{2,[0,T]}}, \tag{4.9}$$

où $\|G_{y\omega}\|$ est la matrice de transfert entre la sortie contrôlée $\mathbf{y}(t)$ et les perturbations externes $\omega(t)$.

Définition 4.2.1 *Soit γ une constante positive. Le système (4.8) est stable avec un niveau de rejet de perturbation γ s'il exite une constante $M(x_0)$ avec $M(0) = 0$, de façon à ce*

que l'inégalité suivante soit satisfaite :

$$\|\mathbf{y}\|_2 \triangleq \left[\int_0^\infty \mathbf{y}^\top(t)\mathbf{y}(t)dt\right]^{1/2} \leq \gamma \left[\|\omega\|_2^2 + M(x_0)\right]^{\frac{1}{2}}$$

Définition 4.2.2 *Le système (4.8) est considéré comme étant quadratiquement stable s'il existe un ensemble de matrices symétriques et positive-définies $P > 0$ satisfaisant l'inégalité suivante :*

$$A^\top P + PA < 0.$$

Selon la définition précédente, il est évident que le système (4.8) est quadratiquement stable lorsque $\omega(t) \equiv 0$, la dynamique étant libre de toute perturbation à son entrée.

Théorème 4.2.1 *(Boukas, E.K., 2005) γ étant une constante positive donnée, s'il existe une matrice symétrique et positive-définie $P > 0$ de façon à ce que l'inégalité matricielle linéaire (Linear Matrix Inequality - LMI) soit satisfaite :*

$$\begin{bmatrix} J_0 & PB_\omega \\ B_\omega^\top P & -\gamma^2\mathbb{I} \end{bmatrix} < 0, \tag{4.10}$$

où $J_0 = A^\top P + PA + C^\top C$, alors le système (4.8) avec $u(t) \equiv 0$ est stable et satisfait :

$$\|\mathbf{y}\|_2 \leq \left[\gamma^2\|\omega\|_2^2 + x_0^\top P x_0\right]^{\frac{1}{2}}, \tag{4.11}$$

ce qui implique que le système avec $u(t) = 0$ pour tout $t \geq 0$ est stable avec un niveau de rejet de perturbation γ.

Pour la conception du contrôleur, le théorème suivant peut être utilisé :

Théorème 4.2.2 *(Boukas, E.K., 2005)* γ *et* α *étant des constantes positives, s'il existe une matrice symétrique et positive-définie* $X > 0$ *et une matrice* Y *de façon à ce que les LMIs suivantes soient satisfaites :*

$$\begin{bmatrix} J & B_\omega & XC^\top \\ B_\omega^\top & -\gamma^2\mathbb{I} & 0 \\ CX & 0 & -\mathbb{I} \end{bmatrix} < 0, \tag{4.12}$$

$$J + 2\alpha X > 0 \tag{4.13}$$

avec $J = XA^\top + AX + Y^\top B^\top + BY$ *; alors le système (4.8) sous le contrôleur* $u(t) = -K\mathbf{x}(t)$ *avec* $K = -YX^{-1}$ *est stable avec les pôles à la droite de la limite imposée* $-\alpha$ *et, de plus, le système en boucle fermée satisfait le niveau de rejet de perturbation* γ*.*

Remarque 4.2.1 *La LMI (4.13) a été introduite afin de prévenir la saturation des moteurs. Ceci est réalisé en imposant une limite supérieure sur la localisation des pôles en boucle fermée (Chilali M., 1996).*

En appliquant le théorème 4.2.2 aux systèmes (4.4) et (4.6), en utilisant Matlab avec l'outil Sedumi, les gains suivant sont obtenus :

$$\tilde{K} = \begin{bmatrix} 437.4456 & 19.1238 & -110.7585 & -81.6985 & 47.5679 \end{bmatrix}$$
$$\tilde{K}_h = \begin{bmatrix} 27.4094 & 10.2282 & -17.9687 \end{bmatrix}$$

L'emplacement des pôles en boucle fermée en utilisant ces gains de contrôleur est montré à la FIG. 4.8

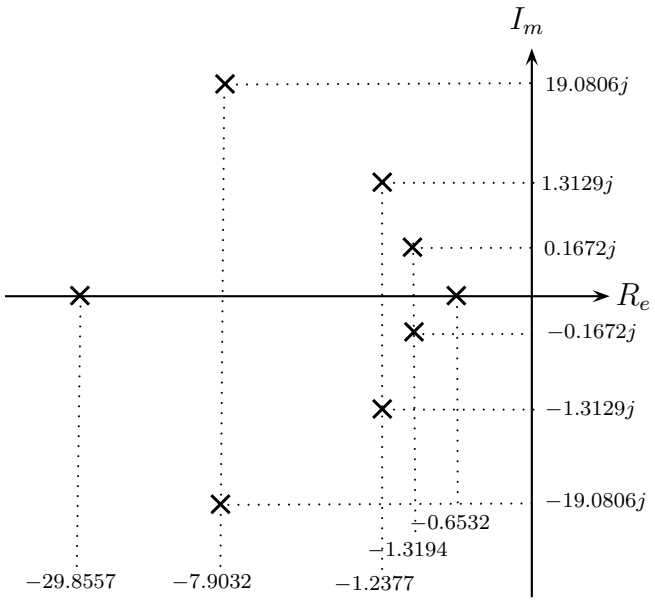

FIG. 4.8 Emplacements des pôles en boucle fermée par $\mathbf{H}_\infty$

4.2.5 Poursuite de trajectoire

Maintenant que des contrôleurs permettant de contrôler le déplacement linéaire et l'angle de direction du robot tout en le maintenant en équilibre ont été conçus, il est possible de concevoir un contrôleur qui assurera que le robot suive une trajectoire dans le plan. Afin de faire suivre au robot une trajectoire désirée, on doit introduire de nouvelles variables définissant la trajectoire voulue et la position réelle du robot sur une surface plane :

$x_r(t)$	: Position x désirée
$y_r(t)$	: Position y désirée
$x_p(t)$	: Position x réelle
$y_p(t)$	: Position y réelle

TAB. 4.1 Variables de poursuite de trajectoire

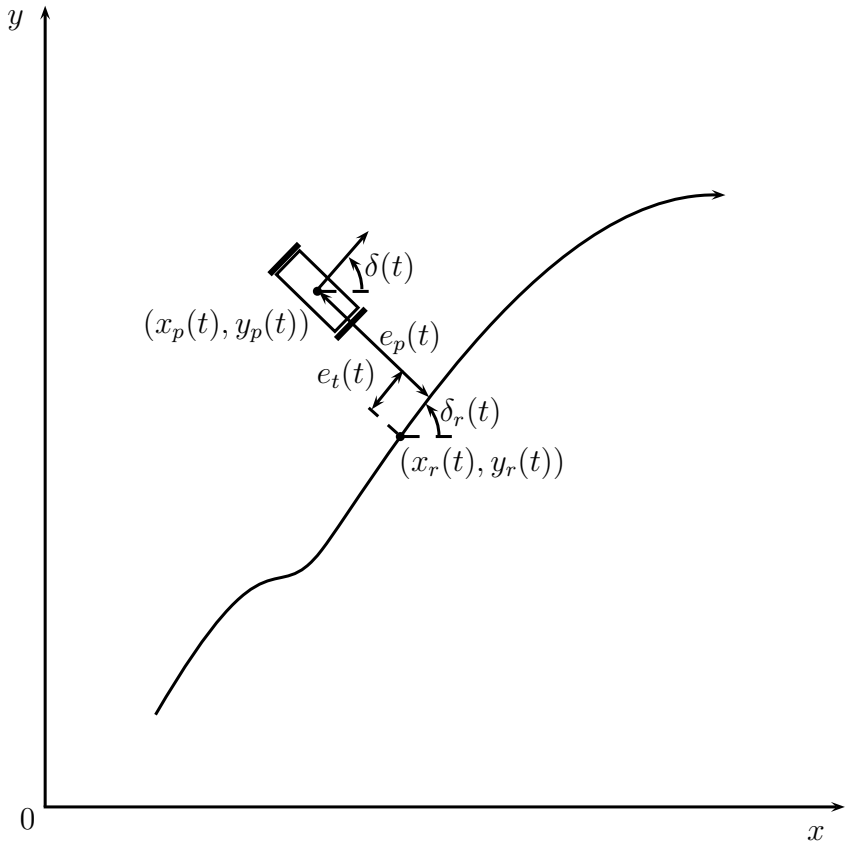

FIG. 4.9 Poursuite de trajectoire

La trajectoire souhaitée étant décrite en termes des références ω_r et V_r et la trajectoire réelle du robot étant décrite par les états $\dot{\delta}$ et $\dot{x}$, ces nouvelles variables de position peuvent être définies comme suit :

$$\dot{x}_r(t) = V_r(t)\cos\left(\delta_r(t)\right) \text{ ce qui donne } x_r(t) = \int_0^t V_r(\tau)\cos\left(\delta_r(\tau)\right)\,d\tau \quad (4.14)$$

$$\dot{y}_r(t) = V_r(t)\sin\left(\delta_r(t)\right) \text{ ce qui donne } y_r(t) = \int_0^t V_r(\tau)\sin\left(\delta_r(\tau)\right)\,d\tau \quad (4.15)$$

$$\dot{x}_p(t) = \dot{x}(t)\cos\left(\delta(t)\right) \text{ ce qui donne } x_p(t) = \int_0^t \dot{x}(\tau)\cos\left(\delta(\tau)\right)\,d\tau \quad (4.16)$$

$$\dot{y}_p(t) = \dot{x}(t)\sin\left(\delta(t)\right) \text{ ce qui donne } y_p(t) = \int_0^t \dot{x}(\tau)\sin\left(\delta(\tau)\right)\,d\tau \quad (4.17)$$

On peut alors définir l'erreur de trajectoire perpendiculaire $e_p(t)$ comme étant la distance perpendiculaire entre la trajectoire désirée et la position réelle du robot. Elle est donnée par la relation suivante :

$$e_p(t) = \cos\left(\delta_r(t)\right)\left(y_p(t) - y_r(t)\right) - \sin\left(\delta_r(t)\right)\left(x_p(t) - x_r(t)\right) \quad (4.18)$$

On peut maintenant calculer la dérivée de l'erreur de trajectoire $e_p(t)$:

$$\begin{aligned}\dot{e}_p(t) = {} & \cos\left(\delta_r(t)\right)\left(\dot{y}_p(t) - \dot{y}_r(t)\right) - \sin\left(\delta_r(t)\right)\omega_r(t)\left(y_p(t) - y_r(t)\right) - \\ & \sin\left(\delta_r(t)\right)\left(\dot{x}_p(t) - \dot{x}_r(t)\right) - \cos\left(\delta_r(t)\right)\omega_r(t)\left(x_p(t) - x_r(t)\right) \\ = {} & \cos\left(\delta_r(t)\right)\left(\dot{y}_p(t) - \dot{y}_r(t)\right) - \sin\left(\delta_r(t)\right)\left(\dot{x}_p(t) - \dot{x}_r(t)\right) - \\ & \omega_r(t)\left[\sin\left(\delta_r(t)\right)\left(y_p(t) - y_r(t)\right) + \cos\left(\delta_r(t)\right)\left(x_p(t) - x_r(t)\right)\right]\end{aligned}$$

mais, puisque par définition, le vecteur $[(x_p(t) - x_r(t)), (y_p(t) - y_r(t))]$ partant de la trajectoire voulue et allant à la position réelle du robot est perpendiculaire au vecteur $[\cos \delta_r(t), \sin \delta_r(t)]$ pointant dans la direction voulue de la trajectoire,
$\sin(\delta_r(t))(y_p(t) - y_r(t)) + \cos(\delta_r(t))(x_p(t) - x_r(t)) = 0$ et l'équation précédente se réduit donc à :

$$\dot{e}_p(t) = \cos(\delta_r(t))(\dot{y}_p(t) - \dot{y}_r(t)) - \sin(\delta_r(t))(\dot{x}_p(t) - \dot{x}_r(t))$$

et en remplaçant les dérivées des positions réelles et voulues du robot $\dot{x}_p(t)$, $\dot{y}_p(t)$, $\dot{x}_r(t)$ et $\dot{y}_r(t)$ par leurs expressions en termes de la vitesse de déplacement linéaire $\dot{x}(t)$, la référence en vitesse linéaire $V_r(t)$, l'angle de direction $\delta(t)$ et la référence en direction $\delta_r(t)$ exprimées par les équations (4.14) à (4.17), on obtient :

$$\begin{aligned}\dot{e}_p(t) = &\cos(\delta_r(t))(\dot{x}(t)\sin(\delta(t)) - V_r(t)\sin(\delta_r(t))) \\ &- \sin(\delta_r(t))(\dot{x}(t)\cos(\delta(t)) - V_r(t)\cos(\delta_r(t)))\,.\end{aligned}$$

En prenant l'hypothèse que le robot suit la référence en vitesse linéaire avec suffisamment de précision, on pose $\dot{x}(t) \approx V_r(t)$ et l'équation précédente se réduit donc à :

$$\begin{aligned}\dot{e}_p(t) = &\cos(\delta_r(t))(\dot{x}(t)\sin(\delta(t)) - \dot{x}(t)\sin(\delta_r(t))) \\ &- \sin(\delta_r(t))(\dot{x}(t)\cos(\delta(t)) - \dot{x}(t)\cos(\delta_r(t)))\end{aligned}$$

$$\begin{aligned}
\dot{e}_p(t) &= \dot{x}(t)\left[\cos\left(\delta_r(t)\right)\left(\sin\left(\delta(t)\right)-\sin\left(\delta_r(t)\right)\right)\right. \\
&\quad \left.-\sin\left(\delta_r(t)\right)\left(\cos\left(\delta(t)\right)-\cos\left(\delta_r(t)\right)\right)\right] \\
&= \dot{x}(t)\left[\sin\left(\delta(t)\right)\cos\left(\delta_r(t)\right)-\cos\left(\delta(t)\right)\sin\left(\delta_r(t)\right)\right. \\
&\quad \left.+\sin\left(\delta_r(t)\right)\cos\left(\delta_r(t)\right)-\cos\left(\delta_r(t)\right)\sin\left(\delta_r(t)\right)\right] \\
&= \dot{x}(t)\left[\sin\left(\delta(t)\right)\cos\left(\delta_r(t)\right)-\cos\left(\delta(t)\right)\sin\left(\delta_r(t)\right)\right] \\
&= \dot{x}(t)\sin\left(\delta(t)-\delta_r(t)\right).
\end{aligned}$$

En dérivant l'expression tout en supposant une vitesse linéaire constante $\dot{x}(t) = V_r$, on obtient :

$$\ddot{e}_p(t) = \dot{x}(t)\cos\left(\delta(t)-\delta_r(t)\right)\left(\dot{\delta}(t)-\omega_r(t)\right).$$

La loi de commande responsable de corriger la référence en vitesse angulaire de direction de façon à annuler l'erreur de trajectoire $e_p(t)$ est donnée par la relation suivante :

$$\tilde{\omega}_r(t) = \omega_r(t) - K_{e_p}e_p(t) - K_{\dot{e}_p}\dot{e}_p(t) \tag{4.19}$$

En supposant que le robot poursuit cette référence corrigée de vitesse angulaire de direction $\tilde{\omega}_r(t)$ avec suffisamment de précision de façon à ce que $\dot{\delta}(t) \approx \tilde{\omega}_r(t)$, on obtient :

$$\begin{aligned}\dot{\delta}(t) - \omega_r(t) &= \tilde{\omega}_r(t) - \omega_r(t) \\ &= -K_{e_p} e_p(t) - K_{\dot{e}_p} \dot{e}_p(t).\end{aligned}$$

Et de façon semblable, en supposant que le robot suit la référence en angle de direction avec suffisamment de précision de manière à ce que $\delta(t) \approx \delta_r(t)$ et donc $\cos\left(\delta(t) - \delta_r(t)\right) = 1$, on peut écrire l'équation précédente comme suit :

$$\ddot{e}_p(t) = -K_{e_p} V_r e_p(t) - K_{\dot{e}_p} V_r \dot{e}_p(t).$$

On peut maintenant utiliser cette dernière équation pour calculer les gains K_{e_p} et $K_{\dot{e}_p}$ du correcteur de poursuite de trajectoire en prenant la transformé de Laplace et en utilisant la technique de placement de pôle de cette façon :

$$\begin{aligned}e_p(s)s^2 + K_{\dot{e}_p} V_r e_p(s)s + K_{e_p} V_r e_p(s) &= e_p(s)\left(s^2 + K_{\dot{e}_p} V_r s + K_{e_p} V_r\right) \\ s^2 + K_{\dot{e}_p} V_r s + K_{e_p} V_r &= s^2 + 2\zeta\omega_n s + \omega_n^2.\end{aligned}$$

Par identification, on obtient :

$$K_{e_p} = \frac{\omega_n^2}{V_r} \tag{4.20}$$

$$K_{\dot{e}_p} = \frac{2\zeta\omega_n}{V_r}. \tag{4.21}$$

Il est nécessaire de choisir des pôles plus lents que ceux de la dynamique d'angle de direction en boucle fermée puisque le contrôle de poursuite de trajectoire dépend de celle-ci. Pour l'implémentation, des pôles complexes caractérisés par une partie réelle étant 0.8 fois celle des pôles dictant la dynamique d'angle de direction en boucle fermée et un taux d'amortissement $\zeta = 0.5$ ont été choisis. Ceci nous donne les gains de poursuite de trajectoire suivants :

$$K_{e_p} = 22.2816 \tag{4.22}$$

$$K_{\dot{e}_p} = 10.5550 \tag{4.23}$$

On peut également définir l'erreur de trajectoire tangent $e_t(t)$ comme étant la distance tangente entre la trajectoire désirée et la position réelle du robot comme suit :

$$e_t(t) = \sin\left(\delta_r(t)\right)\left(y_p(t) - y_r(t)\right) - \cos\left(\delta_r(t)\right)\left(x_p(t) - x_r(t)\right) \tag{4.24}$$

En utilisant cette erreur dans le calcul de la loi de commande (4.5) au lieu de la différence entre le déplacement voulu $x_r(t)$ et le déplacement réel $x(t)$ ainsi que dans le calcul de l'intégrale de l'erreur en position $\tilde{x}(t)$, il est possible de rapprocher le robot de sa position voulue.

4.3 Implémentation numérique

Une fois que les différents algorithmes ont été développés de façon analytique, ceux-ci doivent être implémentés dans un programme exécuté par le microcontrôleur. Ceci implique différents aspects dont :

– La sélection d'une période d'échantillonnage
– La discrétisation des fonctions de transfert continues et
– Le développement d'une application de type temps réel

4.3.1 Discrétisation

Afin d'implémenter les différents algorithmes de commande développés dans les sections précédentes numériquement dans un programme opérant sur le microcontrôleur, il est nécessaire dans un premier temps de discrétiser les fonctions de transfert des différents filtres.

4.3.1.1 Fréquence d'échantillonnage

Avant de discrétiser les fonctions de transfert des différents filtres, il faut d'abord choisir une fréquence d'échantillonnage appropriée. La fréquence d'échantillonnage doit être au moins deux fois supérieure à la bande passante du système selon le critère de Shannon. Par contre, il ne faut pas qu'elle soit trop élevée afin de permettre aux calculs d'être effectués par le microcontrôleur à chaque période d'échantillonnage et de limiter la sensibilité aux bruits.

La bande passante du système est obtenue à partir des pôles du système en boucle fermée. Lorsque le robot est stabilisé avec le contrôleur par retour d'état obtenu avec la technique

H_∞, nous pouvons voir que le pôle représentant la plus haute fréquence du système est situé à environ -30. Donc, afin de respecter le critère de Shannon, nous devons utiliser une fréquence d'échantillonnage d'au moins $60\ rad/s$ soit environ $10\ Hz$, ce qui est très facilement réalisable avec le microcontrôleur choisi. Cependant, il est souhaitable d'utiliser la fréquence la plus élevée possible afin de pouvoir utiliser le contrôleur conçu dans l'espace continu. Selon (Franklin & Powell, 1997), il est souhaitable d'utiliser une fréquence d'échantillonnage étant au moins 30 fois plus rapide que la bande passante du système en boucle fermée. La fréquence d'échantillonnage a donc été choisie par essais et erreurs. Les valeurs suivantes de fréquences d'échantillonnage ont été testées, puisque celles-ci sont facilement réalisables en utilisant une interruption de temporisateur du microcontrôleur à la fréquence de fonctionnement de celui-ci :

- $50\ Hz$
- $125\ Hz$
- $250\ Hz$
- $400\ Hz$
- $500\ Hz$
- $800\ Hz$
- $1\ KHz$
- $1.6\ KHz$
- $2\ KHz$

La fréquence d'échantillonnage retenue est $T_e = 250\ Hz$. Ceci représente un fréquence étant plus de 50 fois supérieure à la fréquence associée au pôle le plus rapide de la dynamique en boucle fermée du robot.

4.3.1.2 Filtre du gyroscope

Dans le cas du filtre passe-haut appliqué à l'estimé de l'angle d'inclinaison obtenu à partir du gyroscope, celui-ci est premièrement combiné à l'intégrale qui est utilisée pour obtenir cet estimé. Ainsi, on obtient :

$$G_{gi}(s) = \frac{\bar{\psi}_g(s)}{\dot{\psi}_g(s)} = \frac{1}{s}\frac{0.25s^2}{0.25s^2 + s + 1} = \frac{0.25s}{0.25s^2 + s + 1} \tag{4.25}$$

La discrétisation de la relation (4.25) avec la méthode trapézoïdale donne la relation suivante :

$$s = \frac{2(z-1)}{T_e(z+1)}$$
$$G_{gi}(z) = \frac{\bar{\psi}_g(z)}{\dot{\psi}_g(z)} = \frac{0.001984z^2 - 0.001984}{z^2 - 1.984z + 0.9841}$$

Celle-ci est ensuite implémentée dans le microcontrôleur sous la forme d'équation aux différences suivante :

$$\bar{\psi}_g(k) = 0.001984(\dot{\psi}_g(k) - \dot{\psi}_g(k-2)) + 1.984\bar{\psi}_g(k-1) - 0.9841\bar{\psi}_g(k-2)$$

4.3.1.3 Filtre de l'accéléromètre

Dans le cas du filtre passe-bas appliqué à l'estimé de l'angle d'inclinaison obtenu à partir de l'accéléromètre, la fréquence d'échantillonnage utilisée est moins élevée que celle utilisée pour le reste du système puisque nous ne voulons conserver que les basses fréquences. Ceci a pour but de réduire le nombre de calculs à être effectués par le microcontrôleur. La fréquence d'échantillonnage pour ce filtre est choisie comme étant une fraction de celle utilisée pour le reste du système et est fixée à $T_{ea} = 50\ Hz$. Cette fréquence est obtenue dans le programme du microcontrôleur à l'aide d'un compteur agissant en tant que diviseur de fréquence. La fonction de transfert de ce filtre est ensuite discrétisée en utilisant la méthode trapézoïdale, ce qui donne la fonction de transfert discrète suivante :

$$s = \frac{2(z-1)}{T_{ea}(z+1)}$$
$$G_a(z) = \frac{0.03883z^2 + 0.0007689z - 0.03806}{z^2 - 1.922z + 0.9231}$$

Celle-ci est ensuite implémentée dans le microcontrôleur sous la forme d'équation aux différences suivante :

$$\bar{\psi}_a(k) = 0.03883\psi_a(k) + 0.0007689\psi_a(k-1) - 0.03806\psi_a(k-2)$$
$$+1.922\bar{\psi}_a(k-1) - 0.9231\bar{\psi}_a(k-2)$$

4.3.1.4 Variables de poursuite de trajectoire

La discrétisation des équations permettant de calculer la valeur des variables de poursuite de trajectoire prend différentes formes si la vitesse angulaire de direction est présentement nulle ou non. Pour la trajectoire voulue, on a :

$$\left.\begin{aligned} x_r(k+1) &= x_r(k) + T_s V_r(k)\cos\left(\delta_r(k)\right) \\ y_r(k+1) &= y_r(k) + T_s V_r(k)\sin\left(\delta_r(k)\right) \end{aligned}\right\} \text{ si } \omega_r(t) = 0 \quad (4.26)$$

$$\left.\begin{aligned} x_r(k+1) &= x_r(k) + \tfrac{V_r(k+1)}{\omega_r(k+1)}\left(\sin\left(\delta_r(k+1)\right) - \sin\left(\delta_r(k)\right)\right) \\ y_r(k+1) &= y_r(k) - \tfrac{V_r(k+1)}{\omega_r(k+1)}\left(\cos\left(\delta_r(k+1)\right) - \cos\left(\delta_r(k)\right)\right) \end{aligned}\right\} \text{ si } \omega_r(t) \neq 0 \quad (4.27)$$

Et pour la position réelle du robot, on a :

$$\left.\begin{aligned} x_p(k+1) &= x_p(k) + T_s\dot{x}(k)\cos\left(\delta(k)\right) \\ y_p(k+1) &= y_p(k) + T_s\dot{x}(k)\sin\left(\delta(k)\right) \end{aligned}\right\} \text{ si } \dot{\delta}(t) = 0 \quad (4.28)$$

$$\left.\begin{aligned} x_p(k+1) &= x_p(k) + \tfrac{\dot{x}(k+1)}{\dot{\delta}(k+1)}\left(\sin\left(\delta(k+1)\right) - \sin\left(\delta(k)\right)\right) \\ y_p(k+1) &= y_p(k) - \tfrac{\dot{x}(k+1)}{\dot{\delta}(k+1)}\left(\cos\left(\delta(k+1)\right) - \cos\left(\delta(k)\right)\right) \end{aligned}\right\} \text{ si } \dot{\delta}(t) \neq 0 \quad (4.29)$$

4.3.2 Système tempsréel

Afin que le programme du microcontrôleur soit apte à contrôler le robot en temps réel, celui-ci doit être en mesure de répondre à différents événements tel que le début d'une période d'échantillonnage, l'arrivée de nouvelles données par le système de communication sans fil et l'arrivée d'une impulsion de l'un des encodeurs optiques et ceci à

l'intérieur de délais acceptables. Afin de remplir ce critère, l'application de commande a été développée sous forme d'une application temps réel comportant plusieurs tâches ayant différentes priorités. De plus, l'application doit être en mesure de gérer l'accès aux ressources partagées par plusieurs tâches. Ces aspects sont discutés dans les sections qui suivent.

4.3.2.1 Tâches et priorités

Les tâches de l'application du système sont réalisées sous forme de routines d'interruption ayant différents niveaux de priorité. Cette stratégie permet de réaliser un système temps réel relativement simple mais efficace dans lequel une tâche plus prioritaire a la capacité d'interrompre une autre (préemption) sans avoir recours à un système d'exploitation temps réel (*Real Time Operating System* - RTOS) qui représenterait une charge de traitement supplémentaire pour le micocontrôleur. Les différentes tâches du système et leur priorités sont les suivantes :

– **Impression de données sur le LCD - Priorité : 1**

  Cette tâche est responsable d'imprimer certaines données sur l'écran LCD du robot. Ceci permet de faire du débogage lors du développement de l'application et d'afficher des données importantes telles que l'angle d'inclinaison sur robot lors de son fonctionnement. Cette tâche est la moins prioritaire du système puisque celle-ci doit s'exécuter à une fréquence relativement basse. De plus, celle-ci n'est pas cruciale au bon fonctionnement du système. Cette tâche est déclenchée par l'interruption du temporisateur 2, étant configuré pour produire une interruption à une fréquence d'environ $7\ Hz$, ce qui est largement suffisant comme fréquence de rafraîchissement du LCD.

– **Contrôle et filtres - Priorité : 5**

  Cette tâche est responsable d'effectuer tous les calculs relatifs au contrôle du robot et

au traitement des signaux des capteurs. Celle-ci est plus prioritaire que la tâche d'impression de données mais moins prioritaire que toutes les autres tâches du système. Ceci est dû au fait que cette tâche est la plus longue à s'exécuter puisqu'elle accomplit la fonctionnalité principale du système et qu'il est donc souhaitable d'être en mesure de l'interrompre afin d'effectuer d'autres tâches plus courtes à exécuter. Cette tâche est déclenchée par l'interruption du temporisateur 1 étant configuré pour produire une interruption à une fréquence d'exactement $250\ Hz$ puisque cette fréquence a été utilisée comme fréquence d'échantillonnage lors de la discrétisation des fonctions de transfert des différents filtres.

– **Communication - Priorité : 6**

  Cette tâche est responsable de récupérer les données reçues par la puce de communication sans fil portant sur la trajectoire désirée envoyée par l'opérateur et d'envoyer les données portant sur la position actuelle du robot. Celle-ci est plus prioritaire que la tâche de contrôle et de filtre afin que les données les plus récentes puissent être utilisées lors des calculs du contrôleur. Cette tâche est déclenchée par une interruption externe qui détecte une impulsion provenant de la puce de communication sans fil indiquant que celle-ci a de nouvelles données prêtes à être traitées. Après avoir récupéré les données de la puce de communication sans fil et avoir mis à jour les variables de trajectoire, un message contenant la position actuelle du robot est envoyé à la puce de communication sans fil afin que celui-ci soit acheminé lors de l'envoi de la confirmation suite à la réception du prochain message.

– **Décodage des encodeurs optiques - Priorité : 7**

  Cette tâche est responsable de la mise à jour des compteurs indiquant la positon des roues lorsqu'une impulsion d'un encodeur optique est produite. Cette tâche est la plus prioritaire du système puisque les encodeurs optiques peuvent produire des impulsions à des fréquences relativement hautes puisque ceux-ci ont une résolution de 2400 impulsions par révolution. Lorsque le robot se déplace à sa vitesse maximale d'envi-

ron $2\ m/s$, ceci représente une fréquence d'environ $16.7\ KHz$. De plus, cette tâche est la plus simple du système puisqu'il s'agit seulement de déterminer quel encodeur a généré une impulsion et dans quel sens le moteur associé tourne, puis incrémenter ou décrémenter un compteur. Elle peut donc interrompre toute autre tâche afin d'être traitée.

4.3.2.2 Ressources partagées

Un autre aspect de l'application temps réel du système qui doit être pris en compte est celui du partage de ressources. En effet, il est nécessaire de s'assurer que les tâches partageant certaines ressources communes soient bien coordonnées afin d'assurer une bonne exécution du programme. Dans le cas de cette application, les ressources partagées sont le bus SPI et les variables de références reçues par le système de communication sans fil.

– **Bus SPI**

  Le bus SPI est utilisé par la tâche de contrôle et de filtre afin d'interagir avec l'accéléromètre et par la tâche de communication afin de d'interagir avec la puce **nRF24L01**. Lorsque la tâche de contrôle et de filtre utilise le bus SPI pour communiquer avec l'accéléromètre, celle-ci étant moins prioritaire, il faut s'assurer qu'elle ne puisse être interrompue par la tâche de communication sans fil puisque ceci causerait une erreur fatale dû à un conflit sur le bus. Cette situation est évitée en changeant temporairement la priorité de la tâche de communication sans fil au niveau de priorité 4, soit juste en-dessous du niveau de priorité de la tâche de contrôle et de filtre présentement en exécution, et ceci seulement durant la portion critique de code qui utilise le bus SPI.

– **Variables de référence**

  Les variables de référence sont utilisées par la tâche de contrôle et de filtre lors du

calcul de la loi de commande et par la tâche de communication sans fil afin de mettre celles-ci à jour suite à la réception de nouvelles données. Comme dans le cas du bus SPI, lorsque la tâche de contrôle et de filtre utilise ces variables, il faut s'assurer que celle-ci ne puisse être interrompue par la tâche de communication sans fil. Ceci pourrait causer une corruption des données dû au fait que les variables seraient mises à jour pendant qu'on tente de les lire. Encore une fois, ceci est réalisé en changeant temporairement la priorité de la tâche de communication sans fil durant la portion critique de code utilisant ces variables.

CHAPITRE 5

COMMANDE VIA INTERNET

Un système de commande par supervision via Internet permettant à un opérateur de commander le robot équilibriste et de surveiller son évolution à partir d'une page web sécurisée a été développé. Ce système est composé de plusieurs éléments dont une architecture de communication multi-couches et une application web J2EE générant des pages web interactives de type web 2.0 basées sur le protocole de haut niveau AJAX. Ces différents éléments constituent le sujet de ce chapitre.

5.1 Architecture de communication

Afin de pouvoir commander le système mécatronique présenté aux chapitres précédents à distance à travers Internet, une architecture de communication comportant plusieurs noeuds et différents protocoles a été mise en place. Une vue d'ensemble de cette architecture est présentée à la FIG. 5.1.

5.1.1 Noeuds

Lors de la commande par supervision via Internet, l'information numérique circulant entre le robot et l'opérateur passe par plusieurs noeuds de communication. Ces différents noeuds sont présentés en détail dans les sections qui suivent.

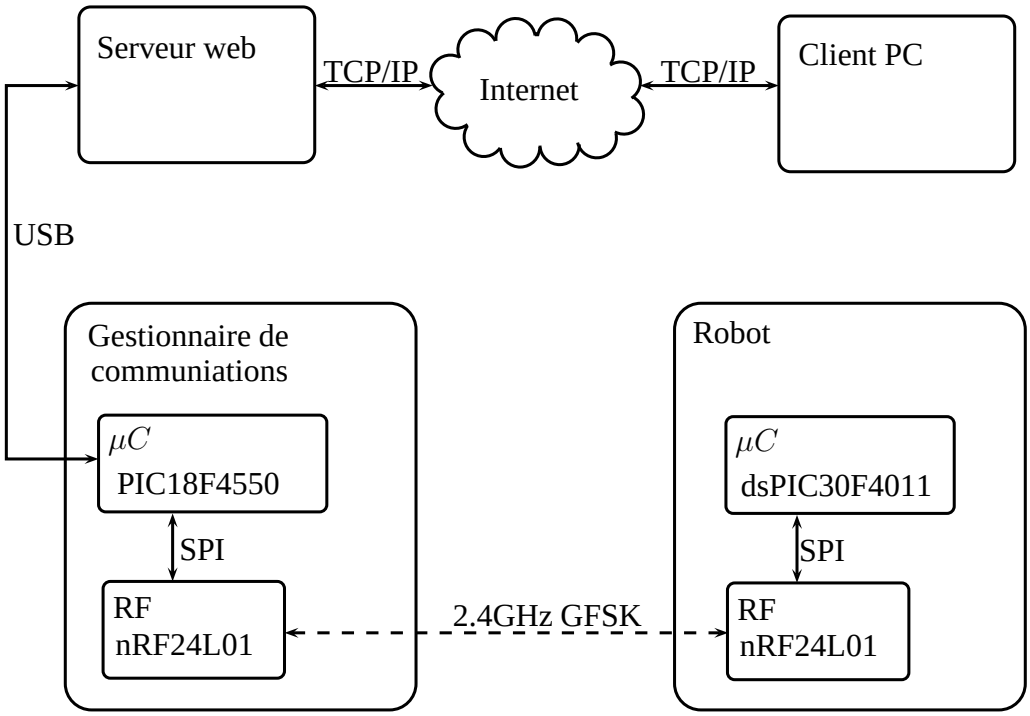

FIG. 5.1 Architecture de communication

5.1.1.1 Robot

Bien entendu, le premier noeud est le robot équilibriste. Sa conception et son fonctionnement on été présentés en détail dans les chapitres précédents. Du point de vue de l'architecture de communication, celui-ci est en lien avec un gestionnaire de communication par l'intermédiaire d'une liaison sans fil assurée par la composante de communication sans fil **nRF24L01**.

5.1.1.2 Gestionnaire de communication

Le gestionnaire de communication agit en tant qu'interface de communication entre le robot et le serveur web. Il s'agit d'une carte électronique basée sur un microcontrôleur de modèle PIC18F4550 du fabriquant *Microchip* comportant une composante de com-

munication sans fil **nRF24L01**. Ce microcontrôleur a été choisi parce qu'il contient un module de communication USB. Le protocole de communication USB est utilisé pour assurer un lien entre cette carte et le serveur web tandis que les communications avec le robot sont réalisées par les composantes de communication sans fil.

La logique du programme développé pour le microcontrôleur de cette carte est très simple. Lorsqu'il reçoit des données en provenance du serveur web via USB, celui-ci les envoie au robot par le lien de communication sans fil. Dans le sens inverse, lorsqu'il reçoit des données du robot, il les envoie au serveur web par USB.

5.1.1.3 Serveur web

Le serveur web est un ordinateur responsable d'exécuter l'application générant les pages web étant diffusées sur Internet et à partir desquelles un opérateur a accès à une interface graphique permettant de contrôler et de surveiller le robot équilibriste. L'application web responsable d'implémenter l'interface utilisateur sous forme de pages web est basée sur la technologie *Java 2.0 Enterprise Edition* (J2EE). Les détails de cette application seront présentés à la section 5.2.

5.1.1.4 Client PC

Le client PC est l'ordinateur utilisé par l'opérateur. Il s'agit de n'importe quel ordinateur capable d'exécuter un logiciel de navigation tel que Firefox de Mozilla ou Internet Explorer de Microsoft. C'est à partir d'un tel fureteur standard que l'opérateur a accès à l'interface graphique permettant la commande par supervision du robot équilibriste.

5.1.2 Protocoles

Les liens de communication entre les différents noeuds de l'architecture de communication sont réalisés à l'aide de plusieurs protocoles. Ceux-ci sont présentés dans les sections qui suivent.

5.1.2.1 SPI

Le protocole SPI a été introduit à la section 2.2.6.6. Du point de vue de l'architecture de communication, ce protocole a été utilisé pour les communications entre les microcontrôleurs et les puces de communication sans fil sur les cartes électroniques du robot et du gestionnaire de communication. Donc, les données portant sur l'état actuel du robot à transmettre à partir du robot vers le serveur web sont d'abord envoyées par le microcontrôleur dsPIC30F4011 dans la mémoire de la puce de communication sans fil de la carte électronique du robot à travers le bus SPI. Les données reçues par cette puce de communication sans fil en provenance de l'utilisateur sont également transmises au microcontrôleur dsPIC30F4011 sur le bus SPI. Ces données sont également transmises sur le bus SPI de la carte électronique du gestionnaire de communication entre la puce de communication sans fil et le microcontrôleur PIC18F4550.

5.1.2.2 RF

Les communications entre le robot et la carte électronique agissant en tant que gestionnaire de communication sont réalisées par un protocole de communication sans fil implémenté par des puces électroniques **nRF24L01** du fabriquant *Nordic Semiconductor*. Ces puces sont présentes sur les cartes électroniques du robot et du gestionnaire de communication. Cette composante électronique a déjà été présentée à la section 2.2.5.

5.1.2.3 USB

Les transmissions de données effectuées entre le gestionnaire de communication et le serveur web sont faites en utilisant un *Universal Serial Bus* (USB). Celles-ci sont implémentées dans module du microcontrôleur PIC18F4550 présent sur la carte électronique du gestionnaire de communication. Par ailleurs, sur le serveur web, un pilote permettant de créer un port de communication série virtuel a été utilisé. De cette façon, il est possible d'échanger des données à travers cette connexion en utilisant une interface de communication *Universal Synchronous & Asynchronous Receiver Transmitter* (USART) comme s'il s'agissait d'une connexion RS-232 traditionnelle. Ceci facilite le développement de l'application responsable d'implémenter les communications USB sur le serveur puisque le protocole USART est largement supporté.

5.1.2.4 TCP/IP et HTTP

Les données échangées entre le serveur web et le client PC sont acheminées à l'aide du protocole TCP/IP. Ce protocole est ensuite utilisé par un protocole de plus haut niveau permettant au fureteur d'envoyer des requêtes au serveur et à celui-ci de répondre par l'envoi de documents. Ce protocole est intitulé le *Hypertext Transfer Protocol* (HTTP). Celui-ci est directement implémenté par l'application web sur le serveur et par l'application fureteur sur le client PC.

5.2 Application Web

Afin de mettre en place une application permettant la commande à distance de systèmes mécatroniques à travers Internet, il est nécessaire de concevoir une application serveur responsable de gérer différents aspects. Ceux-ci incluent les communications, la sécurité,

l'interface graphique, l'affichage de séquences d'images provenant d'une caméra web et l'affichage d'éléments graphiques.

Plusieurs stratégies sont disponibles pour l'architecture d'une telle application. Une approche classique basée sur les technologies client/serveur aurait pu être utilisée et une application aurait pu être développée à l'aide d'un langage de programmation tels que C, C++ ou Java. Par contre, dans le cadre de ce projet, une approche basée sur les technologies web a été utilisée. Cette approche permet de développer et de maintenir l'application de manière centralisée. De cette manière, l'utilisateur n'a qu'à utiliser un logiciel de fureteur standard afin d'accéder à la version la plus récente de l'application sans avoir à se soucier de télécharger, d'installer et de configurer une application spécialisée sur le client PC. De plus, les technologies web offrent des mécanismes standards permettant de gérer plusieurs aspects présents dans ce type d'application tels que les connexions et les communications, la sécurité et le contrôle de l'accès, l'interaction avec un utilisateur à travers une interface graphique, etc. L'approche basée sur les technologies web offre des avantages autant à l'utilisateur qu'au développeur et au gestionnaire d'une telle application. C'est pourquoi il est intéressant d'étudier la possibilité de développer une application permettant la commande à distance par supervision de systèmes mécatroniques avec cette architecture.

Une telle application, basée sur les technologies web, peut être divisée en deux grandes parties. Premièrement, il y a la ou les page(s) web contenant les éléments graphiques avec lesquels l'utilisateur interagit. Et deuxièmement, il y a l'application serveur responsable de générer ces pages web ainsi que de gérer les connexions, les communications et l'état global de l'application. Dans un premier temps, les différents aspects reliés à la page web seront présentés. Puis, les aspects reliés à l'architecture de l'application web seront expliqués.

5.2.1 Page Web

Le robot équilibriste est commandé à distance par un opérateur à partir d'une page web générée par une application web exécutée sur un serveur. Cette page web contient plusieurs éléments permettant la commande et la supervision du robot à distance. Celle-ci est accessible à partir de n'importe quel ordinateur exécutant une application de fureteur de type Mozilla tel que Firefox. Elle peut aussi être accédée par un fureteur Microsoft Internet Explorer avec le plugiciel (Adobe SVG Plug-in) permettant de supporter les objets SVG. Les éléments principaux de la page web sont une interface graphique permettant à un opérateur d'interagir avec le système, une image provenant d'une caméra web pour effectuer la supervision du système et une simulation graphique permettant de visualiser, d'anticiper et de comparer le comportement voulu et réel du robot. Cette page web est réalisée à l'aide du langage de définition *Hypertext Markup Langage* (HTML). Les différentes composantes de cette page sont mises en place à l'aide de différents objets du langage HTML. Ces différents éléments seront présentés dans les sections qui suivent. Un aperçu de cette page web est illustré à la FIG. 5.2 et l'organisation générale du code HTML utilisé est présentée dans le tableau 5.1.

5.2.1.1 Interface graphique

L'interface graphique est un ensemble d'éléments graphiques qui permettent à un opérateur d'interagir avec le système à distance. Les différentes fonctions accessibles à l'utilisateur de l'application incluent la gestion de la connexion et la génération d'une trajectoire voulue pour le robot. Une interface graphique est incluse dans la page web afin de permettre chacune de ces fonctionnalités.

La gestion de la connexion se réalise par l'intermédiaire d'une interface permettant de se connecter au système et de se déconnecter de celui-ci à l'aide de boutons. Les ports

de communication séries du serveur sont affichés dans une liste et le port approprié est sélectionné automatiquement. Cette interface graphique est située dans le coin inférieur gauche de la page web.

La trajectoire voulue du robot est générée à partir de boutons permettant de le faire avancer, reculer, tourner vers la droite et tourner vers la gauche. Ces boutons se situent au milieu de la page web. De façon alternative, l'utilisateur peut utiliser les touches du clavier afin de générer la trajectoire voulue. Il est possible d'utiliser les flèches du clavier ou les touches 8, 2, 6 et 4 du pavé numérique. La trajectoire voulue est tracée dans la simulation visuelle en temps réel.

Tous ces boutons sont réalisés à l'aide d'objets HTML `<button>` et sont positionnés à l'aide d'objets HTML `<table>`. Tandis que les ports de communication séries sont présentés dans un objet HTML de type `<select>` implémentant un menu sous forme de liste.

La gestion dynamique de ces différents objets, incluant l'interaction avec l'utilisateur est réalisée à l'aide du langage Javascript. Cet aspect sera expliqué à la section 5.2.1.4.

5.2.1.2 Caméra web

La supervision du système commandé à distance à travers Internet s'effectue à l'aide d'une image provenant d'une caméra web connectée au serveur web observant l'évolution du robot. Cette image se situe juste au-dessus des boutons permettant de générer une trajectoire voulue pour le robot dans la page web. Plusieurs stratégies sont disponibles afin d'afficher une image provenant d'une caméra web dans une page web. Celles-ci incluent la diffusion d'une vidéo. Par contre, cette façon de faire introduit un délai supplémentaire dû à la conversion et l'encodage du flux de données vidéo. Une stratégie plus simple consistant à afficher une succession d'images comprimées de format *Joint*

```
<html>
<head>
<meta>  <!-- Configuration de la page -->
<title>Remote Robot Control</title>
</head>
<body>
<script language="javascript" type="text/javascript">
.
.       <!-- Comportements dynamiques -->
.
</script>
<table> <!-- Positionnement des éléments graphiques de la page-->
...
<!-- Caméra web -->
<img src="http://132.207.40.107:1201/pic/" name="webcam">
...
<!-- Interface utilisateur -->
<center>
<button type="button" name="Forward" onmousedown="ForwardDown()"
onmouseup="ForwardUp()"><img src="up.gif"></button>
</center>
...
<!-- Liste de ports séries -->
<select>
<option id=COM1 value=COM1>
COM1
</option>
<option id=COM2 value=COM2>
COM2
</option>
...
</select>
.
.
.
</table>
</body>
</html>
```

TAB. 5.1 Aperçu du code HTML de la page web

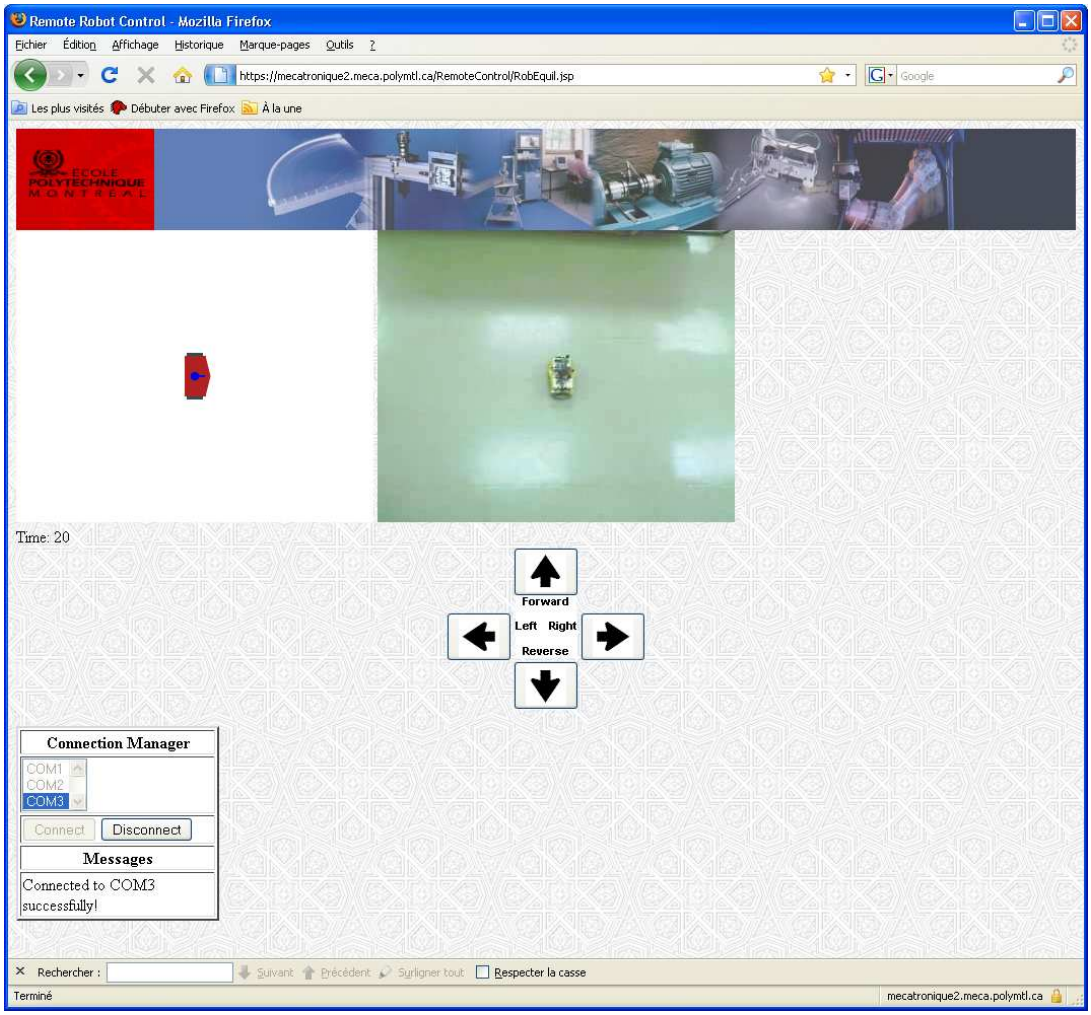

FIG. 5.2 Page web

Photographic Experts Group (JPEG) a été utilisée dans l'application conçue afin de réduire au maximum tout délai supplémentaire. Ceci a été réalisé en faisant appel à un logiciel permettant de diffuser l'image provenant d'une caméra web en format JPEG sur le protocole HTTP (Where's James, 2002). L'image est alors disponible à l'adresse URL du serveur web sur un port donné et peut alors être incluse dans la page web.

Cette image est affichée dans la page web à l'aide de l'objet HTML `<img>`. Le changement périodique de cette image est effectué à l'aide du langage Javascript tel qu'expliqué à la section 5.2.1.4.

5.2.1.3 Génération de trajectoire et simulation visuelle

À cause des délais de communication par Internet, l'opérateur ne peut se fier à l'information provenant du robot ou de la caméra web afin de conduire le robot en temps réel. Une simulation virtuelle incluant la trajectoire de référence générée, une prédiction du comportement du robot par simulation et la trajectoire réellement empruntée par le robot est prévue à cet effet. Cette simulation permet de visualiser la trajectoire de référence générée par l'opérateur, de prédire le comportement du robot selon son modèle dynamique et de comparer la position estimée à la position réelle telle qu'envoyée par le robot. Cette simulation est réalisée par des objets de type *Scalable Vector Graphics* (SVG). Il s'agit d'un standard proposé par le *World Wide Web Consortium* (W3C) permettant de créer des objets visuels interactifs et dynamiques dans une page web. De manière spécifique, dans cette simulation, le robot virtuel est représenté par un ensemble de rectangles et de cercles de différentes couleurs. La position désirée et la position réelle du robot sont représentées par des points de différentes couleurs, tandis que les trajectoires voulues, anticipées et réelles du robot sont représentées par des lignes connectant un ensemble de points. Il est possible de créer tous ces éléments SVG ainsi que de changer les propriétés de ceux-ci telles que la position et l'orientation des formes géométriques ainsi que l'en-

semble de points d'une courbe à partir d'un Javascript tel que décrit à la section 5.2.1.4. Cette simulation visuelle est située dans le coin supérieur gauche de la page web.

Afin de mettre en place une simulation en temps réel du comportement du robot, il est d'abord nécessaire d'établir le modèle mathématique de la dynamique du robot stabilisé à l'aide du contrôleur développé. À partir du modèle d'état (3.12) et du contrôleur (4.5), il est possible d'établir le modèle d'état de la dynamique du système en boucle fermée de la façon suivante :

$$\begin{aligned}
\dot{\mathbf{x}}(t) &= A\mathbf{x}(t) + B\left[K\left(\mathbf{r}(t) - \mathbf{x}(t)\right) + K_{xi}\tilde{x}(t)\right] \\
&= \left(A - BK\right)\mathbf{x}(t) + BK\mathbf{r}(t) + BK_{xi}\tilde{x}(t) \\
&= \left(A - BK\right)\mathbf{x}(t) + B\begin{bmatrix} K & K_{xi} \end{bmatrix}\begin{bmatrix} \mathbf{r}(t) & \tilde{x}(t) \end{bmatrix}^{\top} \\
&= A_{cl}\mathbf{x}(t) + B_{cl}u_{cl}
\end{aligned} \tag{5.1}$$

où

$$\begin{aligned}
A_{cl} &= A - BK \\
B_{cl} &= B\begin{bmatrix} K & K_{xi} \end{bmatrix} \\
u_{cl} &= \begin{bmatrix} \mathbf{r}(t) & \tilde{x}(t) \end{bmatrix}^{\top}
\end{aligned}$$

Il est à noter que l'intégrale de l'erreur en position $\tilde{x}(t)$ est considérée ici comme étant une entrée du système. Ceci implique que cette intégration doit être faite implicitement. Cette façon de faire est appropriée puisque cette intégrale est bornée étant donné que le contrôleur assure que l'erreur en régime permanent de la position linéaire est nulle.

Donc, de cette façon, il est assuré que toutes les variables restent bornées lors de l'implémentation numérique de la simulation.

De la même manière, à partir du modèle dynamique (3.20) et de la loi de commande (4.7), il est possible d'établir le modèle de la dynamique de direction du système en boucle fermée de la façon suivante :

$$\begin{aligned}
\dot{\mathbf{x}}_h(t) &= A_h\mathbf{x}_h(t) + B_h\left[K_h\left(\mathbf{r}_h(t) - \mathbf{x}_h(t)\right) + K_{\delta i}\tilde{\delta}(t)\right] \\
&= \left(A_h - B_hK_h\right)\mathbf{x}_h(t) + B_hK_h\mathbf{r}_h(t) + B_hK_{\delta i}\tilde{\delta}(t) \\
&= \left(A_h - B_hK_h\right)\mathbf{x}_h(t) + B_h\begin{bmatrix} K_h & K_{\delta i}\end{bmatrix}\begin{bmatrix} \mathbf{r}_h(t) & \tilde{\delta}(t)\end{bmatrix}^\top \\
&= A_{hcl}\mathbf{x}_h(t) + B_{hcl}u_{hcl}
\end{aligned} \tag{5.2}$$

où

$$\begin{aligned}
A_{hcl} &= A_h - B_hK_h \\
B_{hcl} &= B_h\begin{bmatrix} K_h & K_{\delta i}\end{bmatrix} \\
u_{hcl} &= \begin{bmatrix} \mathbf{r}_h(t) & \tilde{\delta}(t)\end{bmatrix}^\top
\end{aligned}$$

Afin de réaliser cette simulation de manière discrète dans la page web, il est d'abord nécessaire de choisir une période d'échantillonnage. La période d'échantillonnage choisie est de $T_s = 0.04\ s$ correspondant à une fréquence de $50\ Hz$. Il est à noter que cette fréquence est choisie afin d'obtenir une animation fluide au yeux de l'opérateur et que ceci n'a pas d'effet sur la stabilité du système puisque les algorithmes de commande sont effectués en temps réel sur le microcontrôleur du robot à une fréquence beaucoup

plus élevée. Maintenant, il est possible de discrétiser les modèles d'états continus (5.1) et (5.2) à l'aide de la commande C2D de Matlab afin d'obtenir les matrices A_{cld}, B_{cld}, A_{hcld} et B_{hcld} des modèles d'états discrets suivants :

$$\mathbf{x}(k+1) = A_{cld}\mathbf{x}(k) + B_{cld}\begin{bmatrix} \mathbf{r}(k) & \tilde{x}(k) \end{bmatrix}^{\top} \tag{5.3}$$

$$\mathbf{x}_h(k+1) = A_{hcld}\mathbf{x}_h(k) + B_{hcld}\begin{bmatrix} \mathbf{r}_h(k) & \tilde{\delta}(k) \end{bmatrix}^{\top} \tag{5.4}$$

Les calculs responsables de déterminer l'état du système à partir de ces modèles d'états discrets à l'intérieur d'un Javascript sont réalisés à l'aide de la librairie *Sylvester* (Coglan, J., 2007) qui permet la manipulation de matrices.

L'information sur l'évolution de la trajectoire désirée du robot est contenue dans 4 variables booléennes : `isMoving` indique si l'opérateur désire que le robot se déplace de façon linéaire, `isForward` indique le sens de ce déplacement linéaire, `isRotating` indique si l'opérateur désire que le robot change son angle de direction et `isClocwise` indique le sens de rotation. l'ensemble de ces variables est appelé l'état de commande du robot. Les paramètres suivants sont introduits afin de définir la façon dont la trajectoire voulue du robot est générée suivant cet état de commande :

Paramètre	**Description**	**Valeur**
`MaxSpeed`	Vitesse linéaire maximale	$0.5\ m/s$
`Accel`	Accélération linéaire constante	$0.5\ m/s^2$
`Decel`	Décélération linéaire constante	$0.5\ m/s^2$
`MaxRotateSpeed`	Vitesse maximale de changement de direction	$\pi/2\ rad/s$
`RotateAccel`	Accélération de changement de direction	$\pi\ rad/s^2$
`RotateDecel`	Décélération de changement de direction	$2\pi\ rad/s^2$

La vitesse linéaire voulue du robot $V_r(t)$ évolue de la façon suivante :

– Commence à zéro
– Augmente de façon linéaire avec une accélération constante `Accel` pendant que son état de commande indique un déplacement vers l'avant
– Diminue de façon linéaire avec une accélération constante `Accel` pendant que son état de commande indique un déplacement vers l'arrière
– Est maintenue à vitesse constante $\pm$`MaxSpeed` lorsque celle-ci est atteinte et aussi longtemps que son état de commande indique un déplacement linéaire
– Revient à zéro avec une décélération constante `Decel` lorsque son état de commande indique un déplacement linéaire nul
– Est Maintenue à zéro aussi longtemps que son état de commande indique un déplacement linéaire nul

Dans le Javascript responsable de la mise à jour de la référence, les changements de vitesse sont réalisés en intégrant une accélération de façon discrète. À titre d'exemple, voici la manière dont la vitesse est mise à jour lors d'une accélération positive en réponse à la commande d'un opérateur :

$$V_r(k+1) = \begin{cases} V_r(k) + T_s \cdot \texttt{Accel} & \text{si } V_r(k+1) \leq \texttt{MaxSpeed} \\ \texttt{MaxSpeed} & \text{autrement} \end{cases} \tag{5.5}$$

La vitesse de changement de direction voulue du robot $\omega_r(k+1)$ est déterminée de la même façon en utilisant les paramètres `MaxRotateSpeed`, `RotateAccel` et `RotateDecel`.

Afin de simuler le comportement du robot sous l'action du contrôleur de poursuite de trajectoire, les étapes suivantes sont requises à chaque période d'échantillonnage :

- Calculer les prochains estimés de la positon $x(k+1)$ et la vitesse $\dot{x}(k+1)$ à partir du modèle d'état discret (5.3)
- Calculer les prochains estimés de l'angle de direction $\delta(k+1)$ et de sa dérivée $\dot{\delta}(k+1)$ à partir du modèle d'état discret (5.4)
- Déterminer la vitesse voulue $V_r(k+1)$ selon l'état de commande du robot tel que démontré par l'équation (5.5)
- Déterminer la vitesse de changement de direction voulue $\omega_r(k+1)$ selon l'état de commande du robot de manière semblable
- Calculer le déplacement linéaire voulu $x_r(k+1)$ en intégrant la vitesse voulue $V_r(k)$ afin de déterminer le vecteur de référence $\mathbf{r}(k+1)$ de l'équation (5.3)
- Intégrer $\tilde{\omega}_r(k)$ de manière discrète pour obtenir l'angle de direction désiré $\delta_r(k+1)$ et ainsi déterminer le vecteur de référence $\mathbf{r}_h(k+1)$ de l'équation (5.4)
- Obtenir l'intégrale de l'erreur en position linéaire $\tilde{x}(k+1)$ en intégrant l'erreur de trajectoire tangente $e_t(k)$ de façon discrète
- Obtenir l'intégrale de l'erreur en angle de direction $\tilde{\delta}(k+1)$ en intégrant la différence entre l'angle de direction estimée $\delta(k)$ et celle voulue $\delta_r(k)$ de façon discrète
- Calculer la position voulue $(x_r(k+1), y_r(k+1))$ à partir des équations (4.26) ou (4.27)
- Calculer la position réelle estimée $(x_p(k+1), y_p(k+1))$ à partir des équations (4.28) ou (4.29)
- Déterminer l'erreur de trajectoire perpendiculaire $e_p(k+1)$ à partir de l'équation (4.18) ainsi que sa dérivée par rapport au temps $\dot{e}_p(k+1)$ en utilisant une approximation du premier ordre
- Déterminer l'erreur de trajectoire tangente $e_t(k+1)$ à partir de l'équation (4.24)
- Obtenir $\tilde{\omega}_r(k+1)$ à partir de la loi de commande (4.19)

5.2.1.4 Javascript

Les aspects statiques de la page sont définis à l'aide du langage de spécification HTML. Par contre, le comportement dynamique de la page est réalisé à l'aide du langage Javascript. Celui-ci est un langage à orientation objet basé sur le langage Java qui permet de modifier les éléments d'une page web de façon dynamique et ceci localement sans nécessiter d'interactions supplémentaires avec le serveur. Cet aspect a été utilisé afin de modifier les paramètres des objets visuels de la simulation de façon à réaliser une animation en temps réel et pour mettre à jour l'image provenant de la caméra web.

De plus, ce langage offre un mécanisme permettant de réagir à différents événements générés par des éléments tels que des temporisateurs et des interfaces utilisateur. Ceci permet donc de réaliser des tâches périodiques et de réagir aux actions d'un opérateur. Les tâches périodiques de la page web incluent la mise à jour de la simulation visuelle, la mise à jour de l'image provenant de la caméra web et l'envoi de donnée au robot. L'objet HTML `<script>` permet d'inclure ces éléments applicatifs directement dans la page web générée par le serveur.

La tâche principale du Javascript de la page web consiste en fait en une tâche périodique responsable de mettre à jour la simulation visuelle. Cette tâche est accomplie par une fonction Javascript nommée `Update()`. Afin que cette fonction soit exécutée de façon périodique, un temporisateur est créé par la ligne de code suivante :

```
t=setTimeout(Update, Ts*1000);
```

où `Update` est la fonction à appeler lorsque le temporisateur expire et `Ts*1000` indique le temps d'expiration de ce temporisateur en ms.

La première chose que cette fonction doit réaliser est de calculer l'évolution de l'état

du robot en temps réel. Ceci est accompli en utilisant l'algorithme décrit précédemment à l'aide de la librairie *Sylvester*. À titre d'exemple, voici la déclaration de l'objet `Ad` de classe `Matrix` contenant la matrice A_{hcld} à l'aide de la fonction constructeur `Matrix.create` de cette classe :

```
var Ad = Matrix.create([[0.97448147920169315, 0.03209188872469262],
                        [-1.18360737678380020, 0.62405529325287767]]);
```

La ligne de code suivante montre la manière dont une équation aux différences telle que (5.3) est réalisée à l'aide des fonctions `add()` et `multiply()` de la classe `Matrix` :

```
X = A.multiply(X).add(B.multiply(Xr));
```

Une fois que l'évolution de l'état du robot est déterminée, les éléments graphiques de la simulation visuelle doivent être mis à jour. Cet aspect est également pris en charge par la fonction `Update()` de la page web. À titre d'exemple voici la portion de code responsable de créer et de positionner l'objet SVG `svgRobotBody` consistant en un rectangle rouge représentant le corps du robot :

```
 var svgRobotBody = document.createElementNS(svgNS, "rect");
     svgRobotBody.setAttributeNS(null, "x", StartPosX);
     svgRobotBody.setAttributeNS(null, "y", StartPosY);
     svgRobotBody.setAttributeNS(null, "width",  RobotLength);
     svgRobotBody.setAttributeNS(null, "height", RobotWidth);
     svgRobotBody.setAttributeNS(null, "fill", "FireBrick");
     svgRobotBody.setAttributeNS(null, "transform", "rotate(" +
         PosAngle + " " + RotateCenterX + " " + RotateCenterY + ")");
     svgRoot.appendChild(svgRobotBody);
```

Par la suite, la position et l'orientation des objets SVG sont mises à jour périodiquement. Par exemple, voici la portion de code responsable de mettre à jour les paramètres de l'objet `svgRobotBody` :

```
    svgRobotBody.setAttributeNS(null, "x", ScreenX);
    svgRobotBody.setAttributeNS(null, "y", ScreenY);
    svgRobotBody.setAttributeNS(null, "transform", "rotate(" +
        PosAngle + " " + RotateCenterX + " " + RotateCenterY + ")");
```

Les boutons de l'interface graphique sont créés à l'aide d'objets HTML. Par exemple, voici la portion de code qui créé le bouton permettant à l'opérateur de faire avancer le robot vers l'avant :

```
<button type="button" name="Forward" onmousedown="ForwardDown()"
onmouseup="ForwardUp()"><img src="up.gif"></button>
```

Les fonctions `ForwardDown()` et `ForwardUp()` sont attachées aux événements `onmousedown` et `onmouseup` générés lorsque l'opérateur appuie sur le bouton à l'aide de la souris puis le relâche, respectivement. Ces fonctions sont responsables de mettre à jour l'état de commande du robot. De plus, une image consistant en une flèche pointant vers le haut est incluse à l'intérieur de ce bouton.

Voici la fonction `ForwardDown()` mettant à jour l'état de commande du robot décrivant l'état actuel de l'évolution de la position de référence :

```
function ForwardDown(){
  isMoving = true;
  isForward = true;
}
```

Tel que mentionné précédemment, l'opérateur peur également utiliser le clavier pour faire évoluer la position désirée du robot. Ceci est accompli en créant une fonction responsable de réagir à l'événement `document.onkeydown` survenant lors de l'appui d'une touche du clavier. Cette fonction appelle ensuite la fonction appropriée selon la touche appuyée. Voici la portion de code où cette fonction est créée et attachée à l'événement :

```
function keyDownHandler(e)
{
switch(e.keyCode){
```

```
 case 38:
 case 104: ForwardDown();
           break;
 case 40:
 case 98:  ReverseDown();
           break;
 case 37:
 case 100: LeftDown();
           break;
 case 39:
 case 102: RightDown();
           break;
}

}

document.onkeydown = keyDownHandler;
```

Un Javascript est également responsable de mettre à jour l'image provenant de la caméra web. L'image initiale est d'abord incluse dans la page web dans un objet HTML de type `<img>`, puis un temporisateur est créé afin d'appeler la fonction responsable de mettre à jour l'image :

```
<img src="http://132.207.40.107:1201/pic/" name="webcam">
setTimeout('reloadImage()',500);
```

Par la suite, le changement de l'image est réalisé de façon périodique par la fonction nommée `reloadImage()`. Cette fonction change le URL de l'objet en incluant le nombre de ms écoulées depuis le 1er janvier 1970 de façon à forcer la page à télécharger l'image la plus récente du serveur. Elle fixe également le format de l'image puis crée un temporisateur afin que la fonction soit appelée périodiquement. Voici la déclaration de cette fonction :

```
function reloadImage() {
  var now = new Date();
  if (document.images) {
    document.images.webcam.src = 'http://132.207.40.107:1201/pic/' + now.getTime() + '.jpg';
    document.images.webcam.width = WebCamImageWidth;
    document.images.webcam.height = WebCamImageHeight;
  }
  setTimeout('reloadImage()',500);
}
```

5.2.1.5 Ajax

En plus de mettre à jour les aspects graphiques de la page et de gérer les événements dynamiques, le Javascript est aussi responsable d'établir la communication permettant l'échange de données entre le robot et la page web.

À la base, une application web est conçue pour envoyer des pages HTML en réponse à des requêtes en provenance de fureteurs sur le protocole HTTP. Par ailleurs, dans le contexte de la commande à distance, il est souhaitable d'établir un lien de communication direct entre la page web servant d'interface à l'opérateur et le serveur web étant en liaison avec le robot à travers une interface de communication. Ce type de communication est réalisé à l'aide du protocole à haut niveau AJAX. Ce protocole permet d'établir une liaison HTTP directement entre un Javascript d'une page web et le serveur par l'intermédiaire d'un objet Javascript spécialement conçu à cet effet. De cette façon, une requête HTTP est émise par le Javascript de la page web au serveur et celui-ci envoie sa réponse sous la forme d'un message XML au lieu d'une page HTML. Ensuite, l'information contenue dans ce message est interprétée par le Javascript de la page.

L'objet Javascript utilisé pour les communications AJAX est de différentes classes selon le type de fureteur utilisé. À titre d'exemple, un objet de classe `XMLHttpRequest` est utilisé dans les fureteurs Mozilla tels que Opéra, Safari et Firefox. Ce type d'objet permet d'émettre une requête au serveur à l'aide de sa fonction `open`, d'être signalé lors de la réception de la réponse du serveur à l'aide de son événement `onreadystatechange` et de récupérer cette réponse à travers son élément `responseText`. Dans la page web conçue pour la commande à distance du robot, une fonction Javascript appelée `ajaxFunction()` est créée afin de permettre des communications AJAX. Cette fonction crée un objet `XMLHttpRequest`, attache la fonction `Process()` responsable de réagir à l'événement `onreadystatechange` et envoie une requête au serveur contenant les données portant sur la position voulue du robot et une estampille temporelle.

Lorsque le serveur reçoit ce type de requête, celui-ci se charge d'envoyer la position désirée la plus récente au robot et répond avec la position actuelle la plus récente du robot. Ces données sont encodées dans des chaînes de caractères selon des formats spécifiques.

Par la suite, lors du changement de l'état de l'objet `XMLHttpRequest`, la fonction `Process()` est automatiquement appelée grâce au mécanisme d'événements et celle-ci se charge de récupérer la positon actuelle du robot reçue du serveur et de mettre à jour la trajectoire réelle du robot dans la simulation visuelle.

Les données relatives à la position voulue envoyées aux robot à partir de la page web sont contenues dans une chaîne de caractères ayant le format suivant :

```
X<Xref>Y<Yref>D<Deltaref>
```

où `<Xref>` désigne $x_r(t)$, `<Yref>` désigne $y_r(t)$ et `<Deltaref>` désigne $\delta_r(t)$.

Les données relatives à la position réelle du robot envoyées par celui-ci à la page web sont encodées dans une chaîne de caractères selon le format suivant :

```
X<Xpos>|<Ypos>|<Deltapos>
```

où `<Xpox>` indique $x_p(t)$, `<Ypos>` indique $y_p(t)$ et `<Deltapos>` indique $\delta(t)$. Ces différents formats ont été choisis en fonction des mécanismes de traitement de chaînes de caractères disponibles dans les différents environnements utilisés.

5.2.2 Serveur web J2EE

La tâche principale d'une application de serveur web est de générer des pages web en réponse à des requêtes en provenance de fureteurs. Ces pages sont encodées dans le

langage HTML et envoyées par le protocole HTTP. L'application de serveur web doit également gérer la sécurité en contrôlant l'accès aux différentes ressources et en encodant l'information sensible.

Une application web rudimentaire peut être basée sur des pages HTML statiques stockées en tant que fichiers sur le disque dur du serveur et étant accessibles à l'aide d'adresses *Universal Ressource Locator* (URL). Par contre, les applications web modernes sont dynamiques et leurs pages sont donc générées par des programmes. De cette façon, chaque URL n'adresse pas un fichier statique mais bien un service capable de produire une réponse de façon dynamique. Dans le cas de l'application présentée ici, deux types de services existent : un pour produire la page web servant d'interface graphique à l'opérateur, et un pour permettre les communications AJAX entre la page web et le robot.

Plusieurs architectures de serveur web basées sur différents langages de programmation sont disponibles. Notamment, PHP est couramment utilisé pour les applications du domaine public. Microsoft offre également sa propre solution nommée .NET. La solution choisie pour le développement de l'application présentée est la technologie J2EE puisque celle-ci est couramment utilisée pour le développement d'applications commerciales et industrielles. En effet, la plupart des grands fournisseurs d'applications d'entreprise, tel que SAP, Oracle et IBM privilégient cette technologie en raison de son extensibilité. C'est d'ailleurs dans l'optique de permettre l'intégration de modules permettant la commande à distance par Internet aux applications existantes d'une entreprise que cette technologie a été sélectionnée dans le cadre de ce projet. L'application web J2EE a été développée à l'aide de l'environnement de développement intégré Eclipse. Cet environnement est devenu le standard pour tout développement utilisant le langage Java. Il s'agit d'un logiciel du domaine public créé par IBM afin de promouvoir l'utilisation de standards ouverts basés sur le langage Java. Pour plus d'informations sur cet outil, le lecteur est référé à (Dai, N., 2007).

Les applications J2EE sont structurées selon un standard bien défini. Les services sont implémentés dans un type d'objet appelé *Servlet*. Ceux-ci sont responsables de répondre à une requête par la génération d'un document transmis sur le protocole HTTP. En général, puisqu'un serveur web peut contenir une multitude d'applications avec de nombreux services, ces services ne sont pas gardés en mémoire indéfiniment. Le serveur identifie plutôt le service demandé lors de la réception d'une requête et crée une instance de ce service afin de remplir la requête puis le retire de la mémoire afin de permettre à d'autres services d'être exécutés. Pour cette raison, les données qui doivent demeurer persistantes afin de permettre le bon fonctionnement d'une application ne sont pas contenues directement dans ces objets. Des objets spéciaux appelés *Enterprise Java Beans* (EJB) sont prévus à cet effet. Une application web J2EE est donc principalement composée de différents *Servlets* et EJBs.

L'application web J2EE est composée de plusieurs éléments. Java étant un langage de programmation à orientation objet, il est possible de définir l'application comme étant une hiérarchie d'objets. Au niveau supérieur, il s'agit de l'environnement J2EE lui-même, composé de librairies de fonctions et de la machine virtuelle interprétant et exécutant le code binaire obtenu lors de la compilation des fichiers sources. Ensuite vient le serveur d'application web J2EE aussi appelé *Servlet Container* puisque celui-ci est responsable d'exécuter les services d'une application et de fournir l'environnement nécessaire à leur exécution. Cette composante est aussi responsable de garder en mémoire les EJBs contenant les données d'une application. Ensuite, chaque *Servlet Container* peut contenir plusieurs applications. Chacune des applications est en fait un regroupement de services et de EJBs, une adresse URL permettant d'accéder à ces services, et une région de mémoire appelée contexte réservée afin de contenir les différents objets de l'application. Finalement viennent les différents services et EJBs des applications contenues sur le serveur web. Pour une étude plus approfondie des technologies J2EE, le lecteur est référé à (Mukhar, K., 2006).

L'application web permettant la commande à distance par Internet développée dans le cadre de ce projet a été créée en tant qu'application web JE22 appelée `RemoteControl` et contenant deux *Servlets* et un EJB. La version de Java utilisée est *Java(TM) 2 Runtime Environment, Standard Edition (build 1.5.0_09-b01)* et le *Servlet Container* utilisé est la solution du domaine public *Apache Tomcat V5.5.23*. Pour une étude approfondie de cette application serveur, le lecteur est référé à (Chopra, V. 2007). Les *Servlets* de l'application sont créés en tant que *Java Server Pages* (JSP). Cette façon de faire est une méthode simplifiant le processus de développement de services web responsables de générer des documents à être transmis par HTTP. JSP est en fait une sorte de langage de programmation où il est possible de combiner des éléments statiques du documents tels que des expressions HTML définissant l'organisation visuelle d'une page web et des Javascripts responsables d'implémenter le comportement dynamique de la page à des expressions Java définissant les actions dynamiques du serveur lors de la génération de ce document. L'application a été sécurisée à l'aide du standard *Secure Socket Layer* (SSL) mettant en place le protocole HTTPS, une version sécurisée du protocole HTTP. Ceci a deux effets sur l'application. Premièrement, une authentification des usagers à l'aide d'un nom d'utilisateur et d'un mot de passe est nécessaire afin d'avoir accès aux services de l'application. Deuxièmement, toutes les transmissions entre le serveur et le fureteur sont encodées à partir d'un certificat numérique émis par un tiers. L'interaction entre le client PC et le serveur web est illustrée à la FIG. 5.3 et la hiérarchie d'objets de l'application web est présentée à la FIG. 5.4. Les différents éléments de cette hiérarchie sont présentés dans les sections qui suivent.

5.2.2.1 Page opérateur

La page web `RobEquil.jsp` agissant en tant qu'interface permettant la commande par supervision du robot équilibriste est générée par un service de l'application web. Le contenu de cette page web est presque entièrement statique puisqu'il s'agit essentielle-

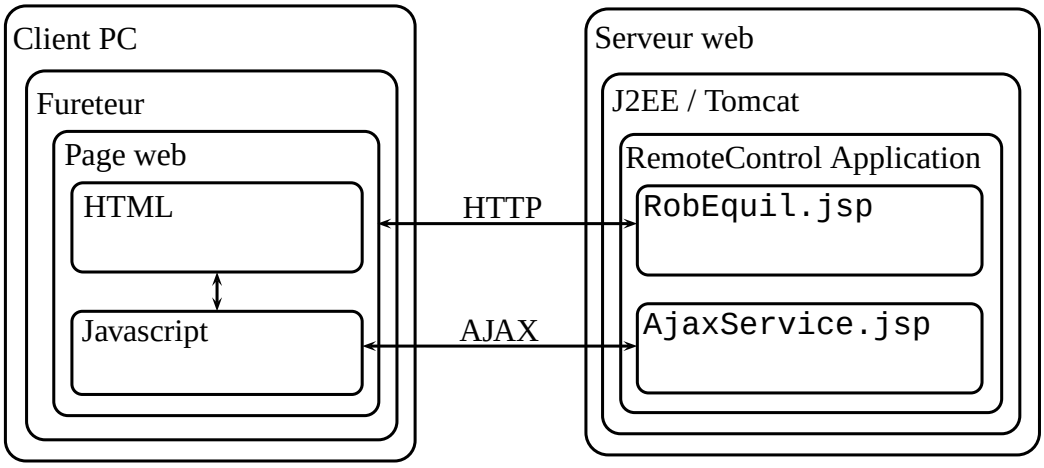

FIG. 5.3 Interaction entre le client PC et le serveur WEB

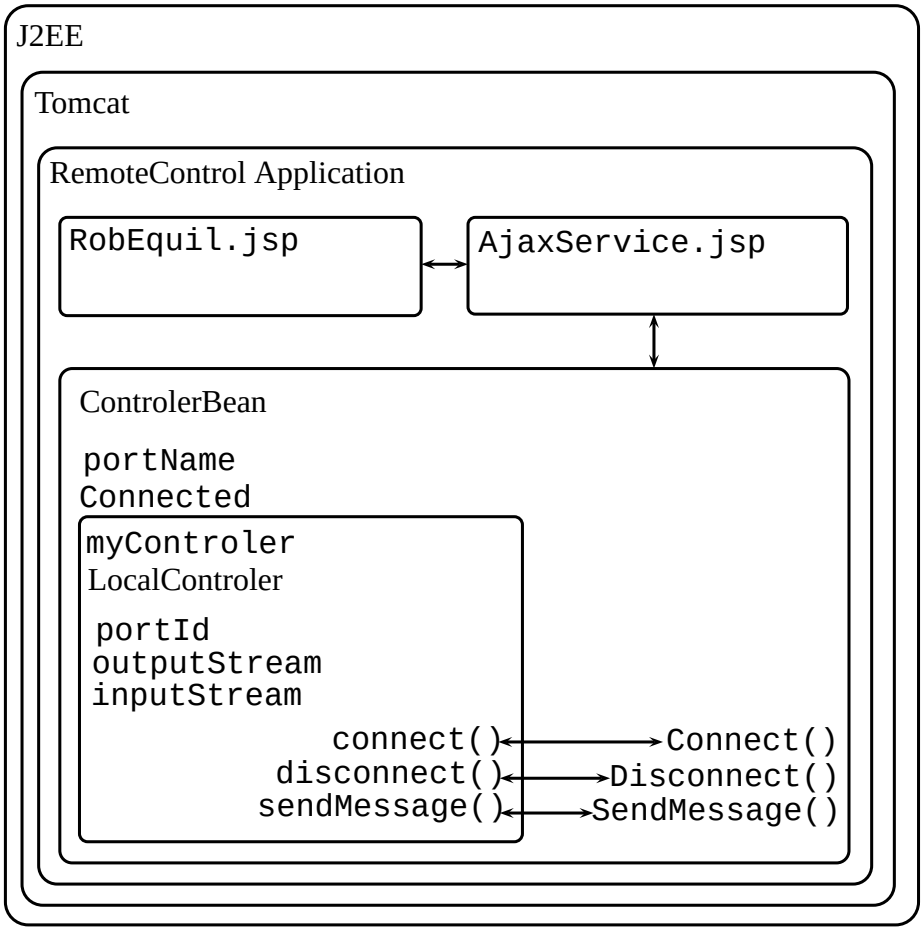

FIG. 5.4 Hiérarchie d'objets de l'application web

ment de l'organisation visuelle de la page comme l'interface graphique, la simulation visuelle et l'image provenant de la caméra web. La seule action dynamique accomplie par le serveur lors de sa génération est l'énumération des ports de communication et l'identification du port permettant de se connecter au gestionnaire de communication responsable d'assurer les communications entre le serveur web et le robot. Tous les comportements dynamiques de la page web sont implémentés par des Javascripts tel que discuté à la section 5.2.1.4.

5.2.2.2 Service Ajax

Le service `AjaxService.jsp` est responsable d'exposer la fonctionnalité de communiquer avec le robot à travers le gestionnaire de communication à la page web servant d'interface graphique. Celui-ci ne contient aucun élément statique. Ce service reçoit les requêtes AJAX en provenance du Javasctipt de la page web et implémente les différentes actions telles que la connexion au système et l'échange de messages. Un paramètre nommé `action` indique le type d'action demandé. Celui-ci peut prendre les valeurs de "`connect`", afin de se connecter au port de communication série assurant la liaison avec le gestionnaire de communication, "`disconnect`", afin de rompre cette communication, et "`sendmessage`", afin d'échanger des données avec le robot. Lors d'un échange de données avec le robot, un second paramètre nommé `message` contient la chaîne de caractères à acheminer au robot. Lorsque ce service est utilisé, celui-ci répond toujours par un message sous la forme d'une chaîne de caractères envoyée à la page web. Dans le cas des actions reliées à la connexion au système, il s'agit d'un message de confirmation ou d'erreur. Tandis que dans le cas de l'action d'envoi de messages, il s'agit de la chaîne de caractères contenant la position actuelle du robot.

5.2.2.3 Interface de communication et Java Bean

Les services exposés par la page `AjaxService.jsp` ne sont pas directement implémentés par celle-ci. Ceci est en raison du fait que les services d'une application J2EE ne restent pas en mémoire de façon durable. Les fonctionnalités permettant de communiquer avec le robot doivent rester en mémoire de façon persistante pour plusieurs raisons. Premièrement, une fois que la communication est établie avec le gestionnaire de communication à travers le port série virtuel permettant les communications USB avec celui-ci à l'aide du protocole USART, les paramètres de cette connexion doivent être conservés. Aussi, il est souhaitable de pouvoir réagir rapidement aux requêtes de communication avec le robot. Il est également nécessaire de garder actif le processus responsable de recevoir les données en provenance du robot par USB. Donc, ces différentes fonctionnalités sont implémentées dans une classe appelée `LocalControler`. Celle-ci fait appel à la librairie *Java Communication Api* (Sun, 1998) facilitant les communications séries à l'aide du protocole USART. Cette classe implémente les interfaces `Runnable`, afin de pouvoir s'exécuter en tant que processus parallèle, et `SerialPortEventListener`, afin de pouvoir réagir aux événements de réception de données en provenance du robot. Une instance de cette classe est alors incluse dans un EJB de l'application afin de la garder dans le contexte de l'application web .

Le EJB `ControlerBean` est responsable d'assurer la persistance des données et des objets permettant le bon fonctionnement de l'application. Ceux-ci incluent le nom du port de communication séries assurant la liaison avec le gestionnaire de communication `portName`, l'état actuel de la connexion `Connected` et un objet de la classe `LocalControler` appelé `myControler`. Les fonctionnalités de cet objet sont à leur tour exposées par le EJB à l'aide de fonctions afin que celles-ci puissent être appelées à partir du JSP.

CHAPITRE 6

EXPÉRIMENTATIONS

Afin d'évaluer la validité du modèle mathématique développé, la performance des algorithmes de commande conçus, et le fonctionnement du système de commande par supervision via Internet développé, plusieurs expérimentations on été effectuées. Celles-ci sont présentées dans les sections qui suivent.

6.1 Validation du modèle

Dans un premier temps, il est important d'évaluer la validité du modèle mathématique (3.12) décrivant la dynamique linéaire du système établi au chapitre 3. Pour ce faire, le contrôleur $\tilde{K}$ obtenu par la technique H_∞ à la section 4.2.4 a d'abord été implémenté dans le microcontrôleur dsPIC30F4011 du robot, et une acquisition de données a été effectuée en utilisant la liaison sans fil à travers le gestionnaire de communication afin d'obtenir l'évolution réelle des états du robot. D'autre part, une simulation utilisant le modèle d'état a été effectuée à l'aide des logiciels Matlab et Simulink. L'expérimentation effectuée représente la poursuite d'une référence en vitesse ayant une condition initiale nulle, augmentant avec une accélération constante de $0.5\ m/s^2$ jusqu'à une vitesse maximale de $1\ m/s$ et redescendant à zéro avec une décélération constante de $-0.5\ m/s^2$. Les résultats obtenus en simulation sont comparés à ceux obtenus en temps réel à la FIG. 6.1. Le graphique du haut compare l'évolution de la position x du robot en temps réel, l'évolution de la position x_s en simulation et l'évolution de la référence en position x_r. Le graphique du milieu compare l'évolution de la dérivée de la position $\dot{x}$ du robot en temps réel, l'évolution de la dérivée de la position $\dot{x}_s$ du robot en simulation et l'évolu-

tion de la référence en vitesse V_r. Tandis que le graphique du bas compare l'évolution de l'angle d'inclinaison ψ du robot en temps réel et l'évolution de l'angle d'inclinaison ψ_s du robot en simulation.

On peut voir que les résultats obtenus en simulation sont comparables à ceux obtenus en temps réel. Il est donc possible de conclure que le modèle établi représente bien la dynamique du système.

6.2 Comparaison des correcteurs

Un fois que le modèle développé est validé, il est également nécessaire d'évaluer la performance des contrôleurs synthétisés en utilisant les différentes techniques considérées. Cette évaluation sera effectuée en observant la réponse temporelle du système ainsi que sa capacité à rejeter une perturbation.

6.2.1 Réponse temporelle

Afin d'évaluer la performance temporelle des deux contrôleurs obtenus à l'aide de la technique de placement de pôles tel que décrit à la section 4.2.3 et de la technique H_∞ décrit à la section 4.2.4, ceux-ci ont été implémentés en temps réel dans le microcontrôleur du robot et le système a été soumis à la même expérimentation que celle utilisée pour la validation du modèle tel que décrit précédemment. Les comportements dynamiques du robot obtenus lors de l'utilisation de ces deux différent contrôleurs sont comparés à la FIG. 6.2. Le graphique du haut compare l'évolution de la position x_p du robot sous l'action du contrôleur par placement de pôles, l'évolution de la position x_h du robot sous l'action du contrôleur H_∞ et l'évolution de la référence en position x_r. Le graphique du milieu compare l'évolution de la dérivée de la position $\dot{x}_p$ du robot sous l'action du

contrôleur par placement de pôles, l'évolution de la dérivée de la position $\dot{x}_h$ du robot sous l'action du contrôleur H_∞ et l'évolution de la référence en vitesse V_r. Tandis que le graphique du bas compare l'évolution de l'angle d'inclinaison ψ_p du robot sous l'action du contrôleur par placement de pôles et l'évolution de l'angle d'inclinaison ψ_h du robot sous l'action du contrôleur H_∞.

En étudiant cette figure, il est possible de remarquer les effets des différentes approches utilisées pour calculer les gains du contrôleur. Dans un premier temps, le contrôleur par placement de pôles offre une bonne réponse temporelle. En effet, il peut être observé que la position du robot suit la référence en position avec beaucoup de précision et n'a qu'un léger dépassement lorsque la vitesse de référence diminue. Ce comportement peut être attribué au fait que le contrôleur a été synthétisé en visant une certaine réponse temporelle représentée par un emplacement de pôles. Par contre, ce contrôleur induit des oscillations au robot comme il est possible de voir dans l'évolution de la vitesse et de l'angle d'inclinaison. Ces oscillations peuvent être causées par des perturbations externes non anticipées et des incertitudes du modèle représentant des perturbations internes du système. Puisque le contrôleur par placement de pôles n'a pas été conçu en vue de minimiser la sensibilité du système aux perturbations, celui-ci n'est pas apte a absorber les effets des perturbations. D'une autre part, le contrôleur H_∞ offre une réponse ayant beaucoup moins d'oscillations, ou du moins des oscillations ayant une amplitude beaucoup plus faible. Cette caractéristique est obtenue aux dépens de la rapidité du système. En effet, il peut être remarqué que le système sous l'action du contrôleur H_∞ ne suit pas la référence avec autant de précision et qu'il a un plus grand dépassement lorsque la vitesse de référence diminue.

6.2.2 Sensibilité aux perturbations

Afin d'évaluer la capacité des contrôleurs à rejeter une perturbation externe, une expérimentation spécialement conçue à cet effet a été effectuée. La perturbation ω_x considérée dans le modèle d'état utilisé pour la synthèse d'un contrôleur à l'aide de la technique H_∞ consiste en une tension supplémentaire étant appliquée aux moteurs. Une telle perturbation a été appliquée au robot en temps réel alors que celui-ci était stabilisé par les différents contrôleurs et l'effet que cette perturbation a sur l'erreur en position linéaire est évalué en observant le comportement dynamique du robot. La perturbation est définie comme étant une impulsion de $24\ V$ d'amplitude débutant au temps $t = 4\ s$ et ayant une durée de $0.5\ s$. Cette perturbation est réalisée dans le microcontrôleur du robot en ajoutant $24\ V$ aux commandes envoyées aux moteurs durant la durée de cette perturbation.

Cette expérimentation a été effectuée en temps réel alors que le robot était stabilisé par le contrôleur obtenu à l'aide de la technique de placement de pôles. Les résultats obtenus sont présentés à la FIG. 6.3. Le premier graphique à partir du haut montre l'évolution de l'angle d'inclinaison ψ et de la position x du robot, le deuxième graphique montre l'évolution de la dérivée de l'angle d'inclinaison $\dot{\psi}$ et de la dérivée de la position $\dot{x}$, le troisième graphique montre l'évolution de la commande u_x calculée par le contrôleur afin de stabiliser le robot, et le dernier graphique montre la perturbation ω_x appliquée aux moteurs.

En étudiant ces résultats, il est possible de remarquer que le contrôleur est apte à maintenir le robot en équilibre malgré la présence de la perturbation. Par contre, cette perturbation a un effet considérable sur l'erreur en position linéaire du robot.

Cette expérimentation a été répétée alors que le robot était stabilisé par le contrôleur obtenu à l'aide de la technique H_∞. Les résultats obtenus sont présentés à la FIG. 6.4.

On peut voir que l'effet de la perturbation sur l'amplitude de l'erreur en position est diminué par rapport au contrôleur par placement de pôles. Il est donc possible de conclure que le contrôleur H_∞ est beaucoup plus apte à rejeter la perturbation externe.

6.3 Commande par supervision

Maintenant que la performance du contrôleur a été évaluée localement, il est nécessaire d'étudier la capacité du système à commander le robot à distance via le réseau Internet. Pour ce faire, la page développée a été accédé à distance et utilisée afin de générer une trajectoire désirée pour le robot et l'image finale de la simulation visuelle est utilisée pour comparer l'évolution de cette trajectoire, de la trajectoire anticipée par simulation, et de la trajectoire réellement empruntée par le robot. Dans cette image, tel que décrit à la section 5.2.1.3, la ligne bleue représente la trajectoire voulue générée par l'opérateur, la ligne rouge représente la trajectoire anticipée par simulation et la ligne noire représente la trajectoire empruntée par le robot telle que mesurée par celui-ci. On considère une trajectoire composée d'une combinaison de déplacements purement linaires et de déplacements purement angulaires. Le résultat obtenu est présenté à la FIG. 6.5.

Il est possible de remarquer que la trajectoire réelle du robot dévie quelque peu de la trajectoire voulue suite à une diminution de vitesse ou à un arrêt complet. Ceci est dû à l'inertie du robot et au fait que sa dynamique en boucle fermée est relativement lente puisqu'il a été jugé plus important de favoriser la robustesse du système face aux perturbations plutôt que sa rapidité. Aussi, la position réelle du robot s'éloigne de la position désirée lorsque celui-ci change de direction. Ceci est dû au fait que le robot ne tourne pas parfaitement sur lui-même lorsque l'opérateur demande un changement de direction. Par contre, il est possible d'observer que lorsque le robot dépasse la position désirée suite à un ralentissement, la trajectoire obtenue par simulation anticipe de manière efficace la trajectoire réelle du robot. Aussi, suite à un écart entre la position désirée et la position

réelle causé par un ralentissement ou un changement de direction, la trajectoire réelle du robot converge vers la trajectoire désirée lors d'un déplacement linéaire. Donc, il est possible de conclure que, pour ce type de trajectoire, la simulation conçue s'avère un moyen efficace de prédire le comportement du robot et que l'algorithme de poursuite de trajectoire est apte à assurer que la trajectoire empruntée par le robot suit de près la trajectoire voulue.

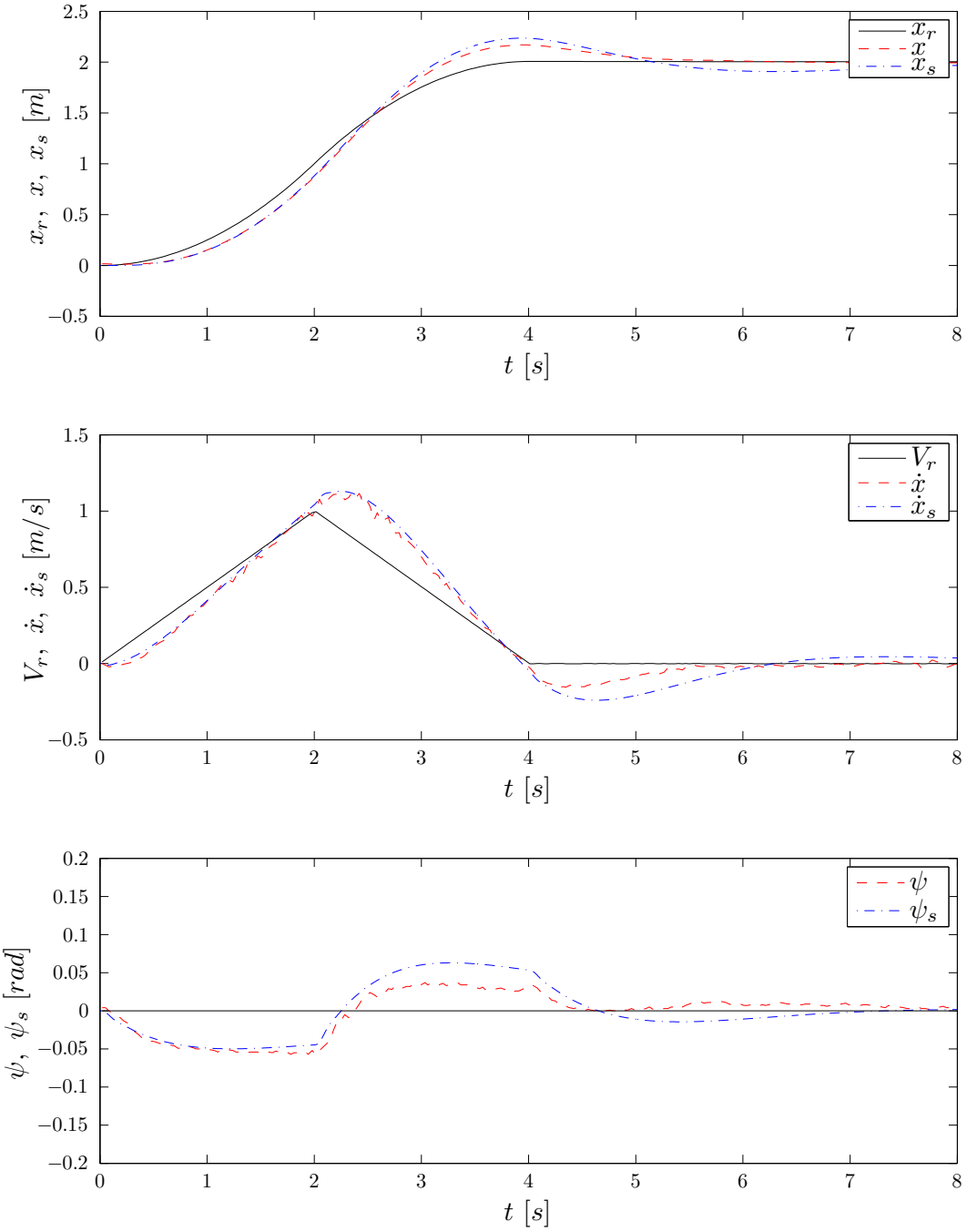

FIG. 6.1 Simulation v.s. temps réel

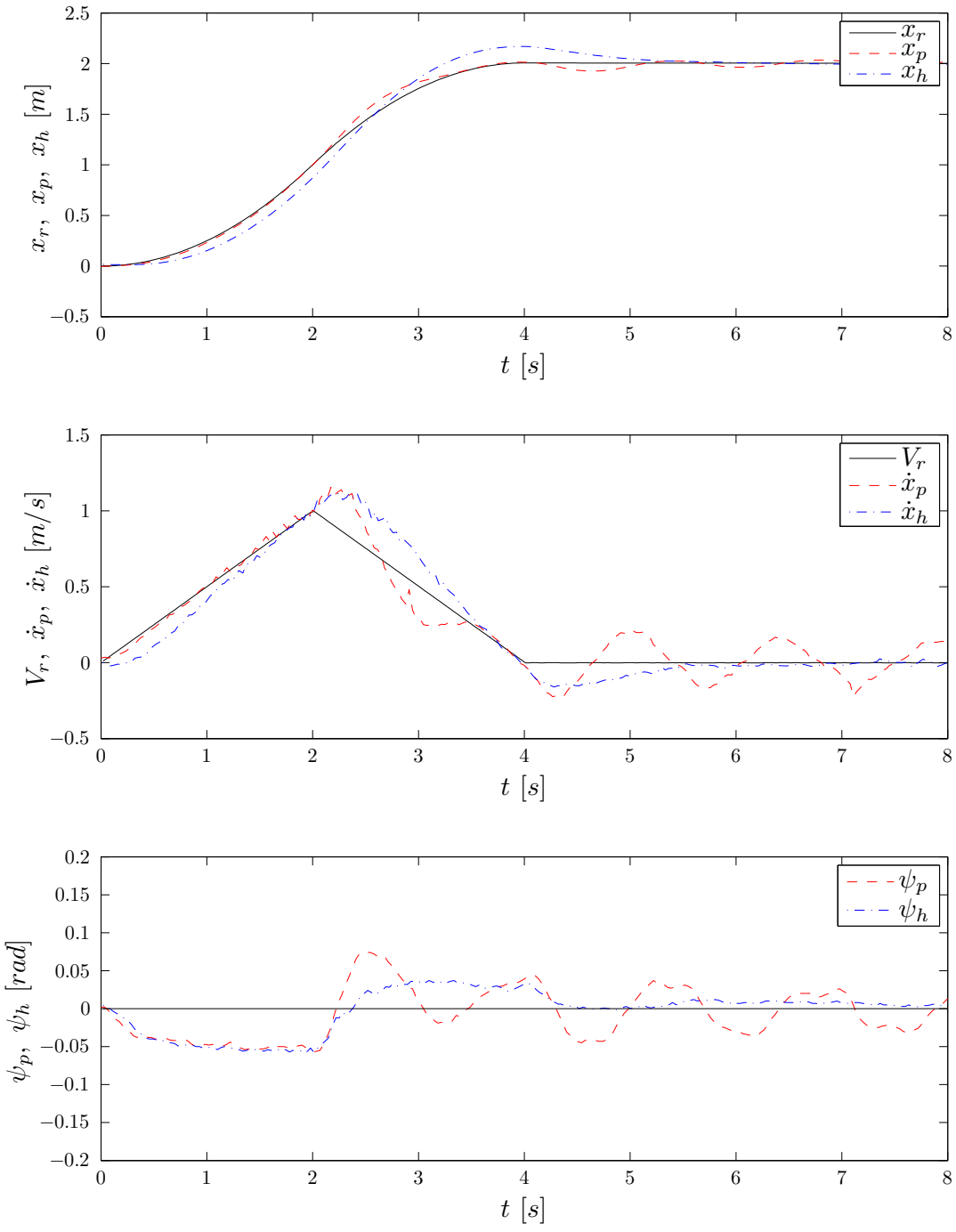

FIG. 6.2 Comparaison des deux contrôleurs

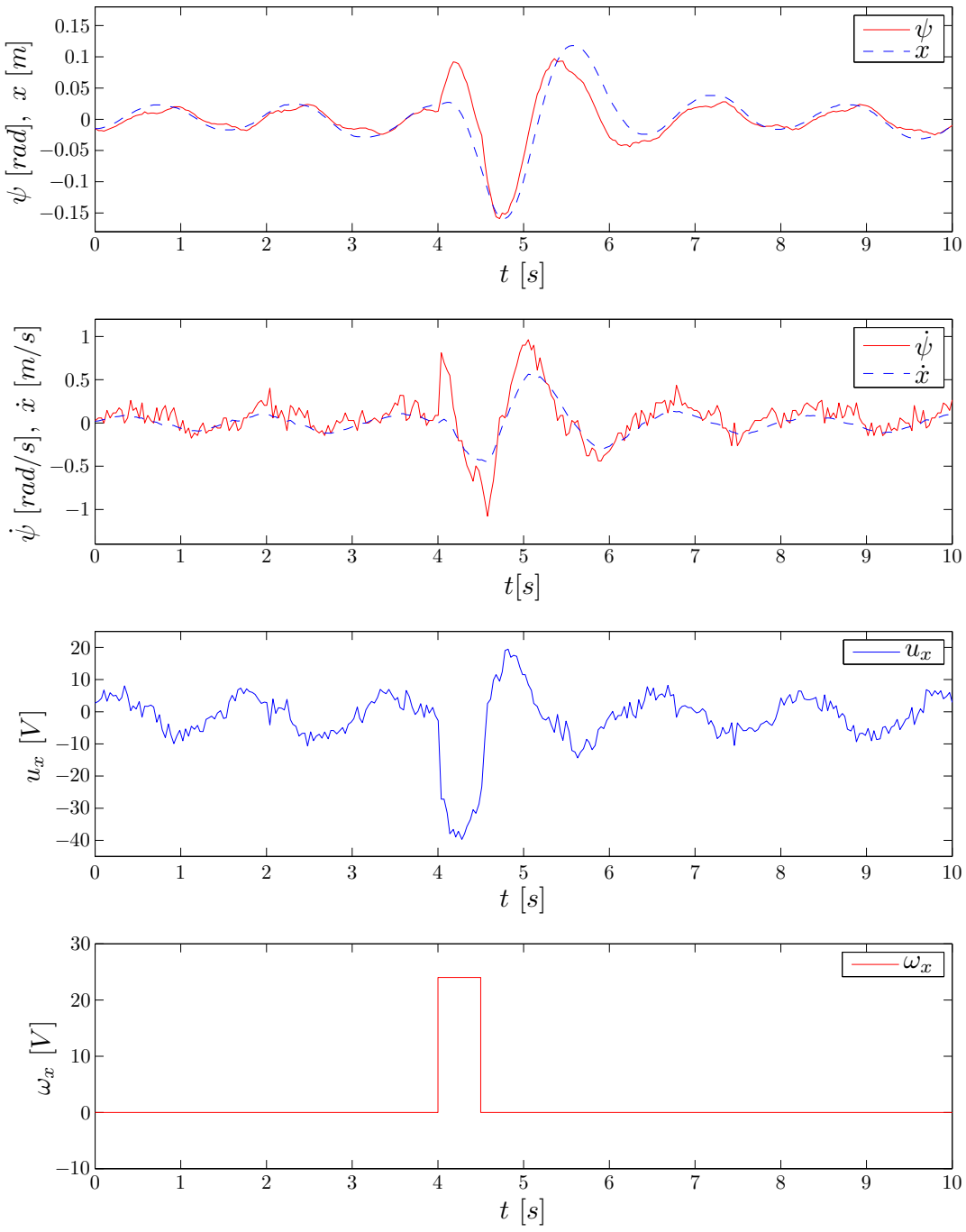

FIG. 6.3 Rejet de perturbation placement de pôles

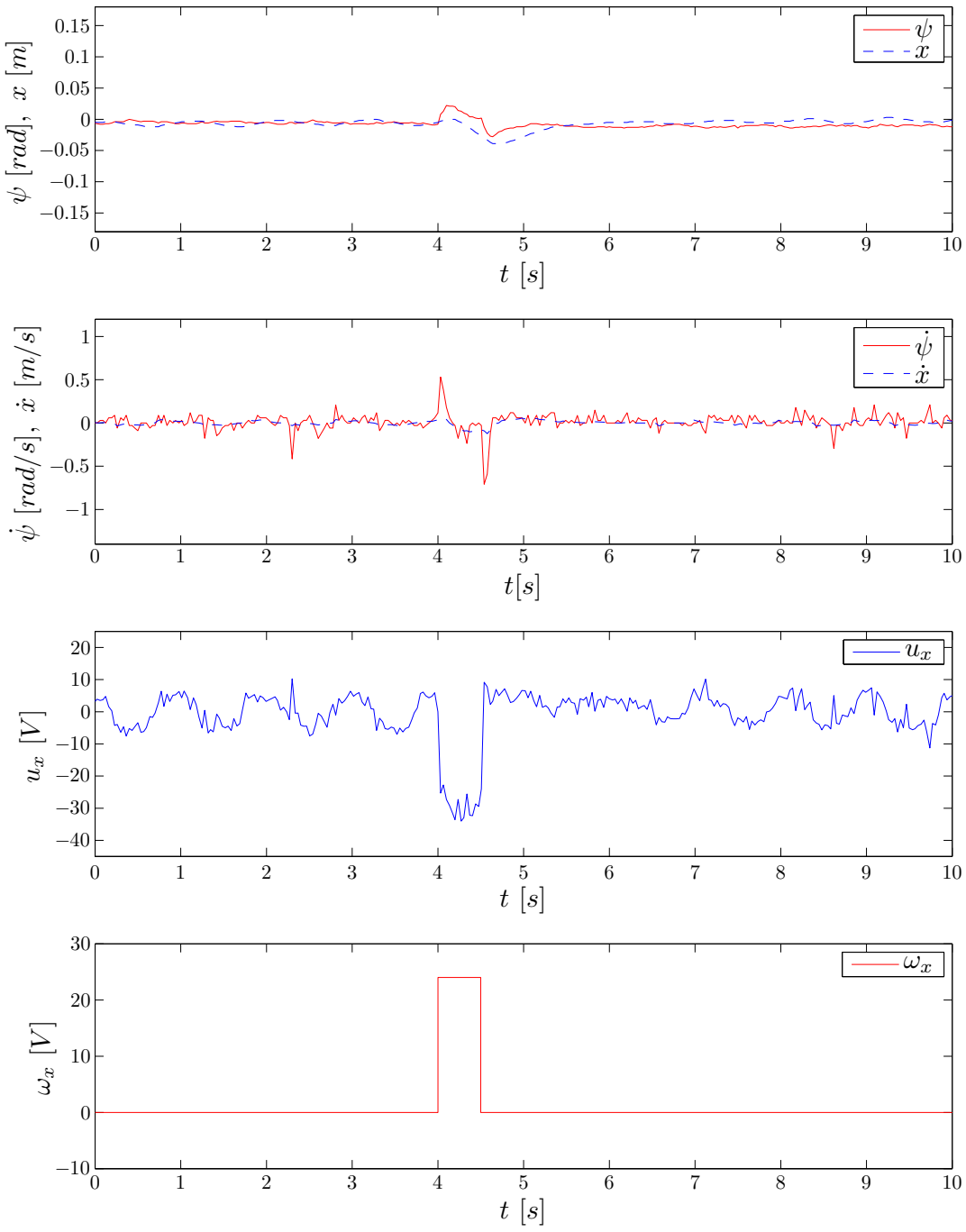

FIG. 6.4 Rejet de perturbation H_∞

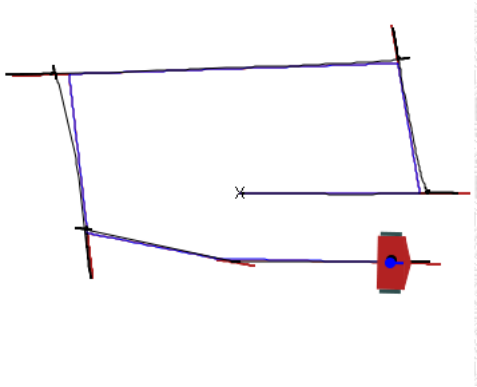

FIG. 6.5 Poursuite de trajectoire via Internet

CONCLUSION

Le but de ce projet de recherche était de proposer et d'expérimenter une architecture d'application web basée sur des standards ouverts permettant la commande par supervision de systèmes mécatroniques à travers Internet. Suite aux travaux effectués, un système mécatronique relativement complexe consistant en un robot équilibriste a été conçu afin de servir de banc d'essai pour un tel système. Ceci inclut la conception du système électronique à base d'un microcontrôleur et de capteurs abordables ainsi que la conception d'algorithmes de commande et le développement du logiciel embarqué du système de commande. Une architecture de communication comportant plusieurs couches a également été mise en place et une application web basée sur des technologies ouvertes permettant la commande à distance de ce système a été développée. Les diverses expérimentations réalisées démontrent le bon fonctionnement du système dans son ensemble.

Les contributions principales de ce projet de recherche se situent au niveau de différents éléments pouvant être réutilisés dans le cadre des activités de recherche et d'enseignement de la mécatronique et de la théorie de la commande à l'École Polytechnique de Montréal. Le robot conçu peut servir de système de démonstration dans les cours reliés au domaine de la commande. Ce système a d'ailleurs déjà été le sujet de plusieurs présentations données par l'auteur dans le cadre des cours MEC3300 (Analyse et commande des systèmes dynamiques), MEC4360 (Mécatronique II) et ELE6204 (Commande des systèmes non linéaires) afin de démontrer l'application de la théorie de la commande à un problème réel. Le système peut également servir de banc d'essai afin de tester différents algorithmes de commande et ainsi évaluer leurs performances. Ainsi, il peut être réutilisé dans le cadre d'autres projets de recherche et de fin d'étude. Aussi, l'architecture de commande par supervision par Internet basée sur la technologie J2EE développée dans le cadre de ce projet peut servir de point de départ pour des projets futurs.

De manière générale, le succès du projet ainsi que les connaissances acquises par l'auteur au cours de celui-ci dans les domaines de la conception de montages électroniques, de la programmation de microcontrôleurs, des systèmes temps réel et de la théorie de la commande ont été utiles pour l'ensemble de la communauté étudiante oeuvrant au laboratoire de mécatronique. Plus particulièrement, l'auteur a eu l'opportunité d'encadrer des étudiants réalisant des projets de fin d'étude et des stagiaires en visite au laboratoire.

Plusieurs améliorations pourraient être apportées au système. Du point de vue de l'électronique, le système de puissance s'avère être une faiblesse puisque les batteries n'offrent qu'une faible autonomie. Il serait donc intéressant de considérer la conception d'un nouveau circuit de puissance permettant une meilleur gestion de la puissance et de mettre en place un indicateur d'autonomie et un ajustement des gains des correcteurs en fonction de la puissance disponible.

Du point de vue des algorithmes de traitement des signaux et de commande, il est possible d'envisager d'autres techniques de traitement des signaux des capteurs afin d'éliminer leurs bruits et d'obtenir un estimé fiable des états du robot. Ceci peut être accompli par un observateur d'états et le calcul du gain de cet observateur pourrait être réalisé par la technique de placement de pôles, par la technique d'estimation optimale de Kalman, ou par la technique de H_∞ donnant un observateur robuste. Il est également possible de tester différents algorithmes de commande tel que le *sliding mode*, la commande floue et la commande par réseau neuronaux. Il est aussi possible de tester différents algorithmes de poursuite de trajectoire tels que la poursuite pure, la poursuite vectorielle ou d'autre algorithmes plus complexes.

D'autres extensions possibles à partir des contributions de ce projet incluent l'élaboration d'un laboratoire virtuel permettant à des étudiants de réaliser des manipulations à distance dans le cadre de travaux pratiques. Ceci sera grandement facilité par l'aspect modulaire de la solution développée. En effet, un système permettant de commander à

distance plusieurs montages mécatroniques pourrait être réalisé. Une application web centralisée serait responsable de gérer la sécurité, de contrôler l'accès aux différents montages et de générer les pages web servant d'interface utilisateur. Alors que des caméras web permettant de visualiser l'évolution des montages et des services AJAX permettant la communication avec ceux-ci à travers une connexion USB se retrouveraient sur différents PCs se situant à proximité des différents montages.

RÉFÉRENCES

Abdollah, M. (2006). *Proportional Integral Sliding Mode Control of a two-wheeled balancing robot*. Faculty of Electrical Engineering, Universiti Teknologi Malaysia

Adobe. Scalable Vector Graphics Plug-In. Version 3.03. [Logiciel]. Tiré de : http ://www.adobe.com/svg/viewer/install/main.html

Anderson, D. (2007). *nBot, a two wheel balancing robot*. http ://www.geology.smu.edu/ dpa-www/robo/nbot/

Boukas, E. K. (1995). *Systèmes Asservis*. Éditions de Polytechnique, Montréal, Canada.

Boukas, E. K. (2005). *Stochastic Switching Systems : Analysis and Design*. Birkhäuser

Baerveldt, A.J. & Klang, R. (1997). *A Low-cost and Low-weight Attitude Estimation System for an Autonomous Helicopter*. Center for Computer Systems Architecture, Halmstad University, Halmstad, Sweden

Cardi, A., Enes, A. & Wagner, M. (2005). *Control of an RC Segway*. Woodruff School of Mechanical Engineering, Georgia Institute of Technology. Tiré de : http ://www.prism.gatech.edu/ gth856d/buffalo/index.htm

Chilali, M. & Gahinet, P. (1996). H_∞ *Design with Pole Placement Constraints : An LMI Approach*. IEEE Transactions on Automatic Control, Vol. 41, No. 3, March 1996

Chopra, V. (2007). *Professional Apache Tomcat 5*

Coglan, J. (2007). *Sylverster*. Version 0.1.3. [Logiciel]. Tiré de : http ://sylvester.jcoglan.com/

Dai, N. (2007). *Eclipse Web tools platform : developing Java Web applications* . Addison-Wesley, Upper Saddle River, NJ

Dalton, B. (2001). *Techniques for Web Telerobotics*. (Ph.D., University of Western Australia, Perth, Western Australia, Australia). Tiré de : http ://telerobot.mech.uwa.edu.au/Information/Dalton thesis.pdf

Franklin, G. F., Powell, J.D. & Workman, M. (1997). Digigatal Control of Dynamic Systems. Addison Wesley, Menlo Park, CA

Grasser, F., D'Arrigo, A., Colombi, S. & Rufer, A.C. (2002). *JOE : A Mobile, Inverted Pendulum*. IEEE Transactions on Industrial Electronics. Tiré de : http ://www.geology.smu.edu/ dpa-www/robo/nbot/grasser_darrigo_colombi_rufer_mic_01.pdf

Gupta, R.A. & Chow, M.Y. (2008). *Overview of Networked Control Systems*. North Carolina State University, Raleigh, NC 27695, USA

Ha, Y. & Yuta, S. (1996). *Trajectory Tracking Control for Navigation of Self-contained Mobile Inverse Pendulum*. Institute of Information Sciences and Electronics, University of Tsukuba, Tsukuba , Japan. Tiré de : http ://ieeexplore.ieee.org/xpls/abs_all.jsp ?arnumber=407604

Han, K.H., Kim, S., Kim, Y.J. & Kim, J.H. (2001). *Internet Control Architecture for Internet-Based Personal Robot*. Kluwer Academic Publishers, Netherlands

Kane, T. R. & Levinson, D. A. (1985). *Dynamics, Theory and Applications*. McGraw-Hill

Kadir, H.B. (2005). *Modelling and control of a balancing robot using digital state space approach*. Faculty of Electrical Engineering, Universiti Teknologi Malaysia

Kim, Y., Kim, S.H. & Kwak, Y.K. (2005). *Dynamic Analysis of a Nonholonomic Two-Wheeled Inverted Pendulum Robot*. Journal of Intelligent and Robotic Systems

Lassó, A. & Urbancsek, T. (2001). *Communication architectures for web-based telerobotic systems*. Budapest University of Technology and Economics, Budapest, Hongrie

Lindsay, E.D., Good, M.C. & Halgamuge, S.K. (2000). *Web Based Telerobotics and Hardware-in-the-Loop Simulations as Enhancements to Mechatronics Teaching*. Department of Mechanical & Manufacturing Engineering. The University of Melbourne, Victoria, Australia. Tiré de : http ://mech-eng.curtin.edu.au/lindsay/publications/m2vip_HerveyBay_2000.pdf

Maxon. (2005). *Fiche technique des moteurs*.

Maxon. (2005). *Fiche techique des réducteurs*.

Microchip. (2006). *dsPIC30F Family Reference Manual*. Tiré de : http ://www.microchip.com/stellent/idcplg ?IdcService=SS_GET_PAGE&nodeId=2574

Mukhar, K. (2006). *Beginning Java EE 5 : from novice to professional* . Apress, Berkeley, California

Nawawi, S.W., Ahmad, M. N. & Osman, J.H. (2008). *Real-Time Control of a Two-Wheeled Inverted Pendulum Mobile Robot*. Proceedings of world academy of science, engineering and technology volume 29

Ooi, R.C. (2003). *Balancing a Two-Wheeled Autonomous Robot*. University of Western Australia

Roesh, O., Roth, H. & Nicolescu, S.I. (2005). *Remote Control of Mechatronic Systems over Communication Networks*. *Proceedings of the IEEE International Conference on Mechatronics & Automation, Niagara Falls, Canada* (Vol. 3 pp. 1648-1653). Tiré de : http ://ieeexplore.ieee.org/xpls/abs_all.jsp ?arnumber=1626802

Sun. (1998). *Java Communications API*. Version 3.0. [Logiciel]. Tiré de : http ://java.sun.com/products/javacomm/

Taylor, K. (1999). *Web Telerobotics : Reducing Complexity In Robotics*. (Ph.D., University of Western Australia, Perth, Western Australia, Australia). Tiré de : http ://www.urremote.com/index.php ?title=WebTelerobotic_Introduction

Université of Western Australia. (2004). *The UWA Telerobot*. http ://telerobot.mech.uwa.edu.au/Telerobot/index.html

Wang, L., Xi, F. & Zhang, D. (2005). *A parallel robotic attachment and its remote manipulation*. Robotics and Computer-Integrated Manufacturing, Elsevier

Where's James. (2002). *WheresJames Webcam Publisher*. Version 2.0.0015. [Logiciel]. Tiré de : http ://www.wheresjames.com/index.php ?page=wjwp2