International Journal of Innovation and Applied Studies
ISSN 2028-9324 Vol. 35 No. 2 Jan. 2022, pp. 385-400

http://www.ijias.issr-journals.org/

Configuration d'une Raspberry Pi comme Serveur et Co-simulation avec un Client Socket pour la Téléopérabilité Optimale d'un Processus Dynamique Asservi par Régulation Quadratique Linéaire avec Poursuite

[Configuration of a Raspberry Pi as a Server and Co-Simulation with a Socket Client for Optimal Teleoperability of a Dynamic Servo Process by Linear Quadratic Control with Tracking]

Manga Betene Ignace Fabrice, Paune Felix, Mbihi Djoumessi Markov, and Nneme Nneme Leandre

Research Laboratory of Computer Science Engineering and Automation, Enset, University of Douala, Cameroon

ABSTRACT: In this paper, we present a system for controlling the angular velocities of the motors of a 2WD mobile robot using an optimal Linear Quadratic Regulator with Tracking (LQRT), thanks to a Co-simulation between two Raspberry Pi modules and the MATLAB R2018a software. Indeed, we have a system made up of a certain number of elements, notably a web interface for communication with users, a Raspberry Pi 4 Model B module that we have configured as a server and a Raspberry Pi 3 Model B module that plays the role of a socket client who's physical GPIOs are represented in an identical logical manner on Simulink in order to facilitate interaction with the process modelled in MATLAB R2018a. This system has been realized thanks to a combination of various software technologies such as the python flash framework for the development of the web application, the HTML and CSS programming languages for the client side of our user application, the library written in C language SQLite for the relational database engine accessible by the SQL language, the JavaScript library Socket. IO library for real-time bidirectional communication between clients and servers, the Python Threading library to facilitate the execution of parallel processes and the python RPI library to control the GPIO ports of the Raspberry Pi 3. The speed control simulation results on a 2WD mobile robot in both normal and Co-simulation modes show almost identical performance indices.

KEYWORDS: Co-simulation, Raspberry Pi, GPIO, 2WD mobile robot, LQRT control.

RESUME: Dans cet article, nous présentons un système d'asservissement des vitesses angulaires des moteurs d'un robot mobile 2WD effectué par un correcteur optimal Régulateur Quadratique Linéaire avec Poursuite (RQLP), grâce une Co-simulation entre deux modules Raspberry Pi et le logiciel MATLAB R2018a. En effet, nous disposons d'un système constitué d'un certains nombres d'éléments notamment une interface web pour la communication avec les utilisateurs, un module Raspberry Pi 4 Model B que nous avons configuré comme serveur et un module Raspberry pi 3 Model B qui joue le rôle de client socket dont les GPIO physiques sont représentées de manière logique identique sur Simulink afin de favoriser l'interaction avec le processus modélisé sous MATLAB R2018a. Ce système a été réalisé grâce à une combinaisons de diverses technologies logicielles à l'instar du Framework python flash pour le développement de l'application web, des langages de programmation HTML et CSS pour le coté client de notre application utilisateur, la bibliothèque écrite en langage C SQLite pour le moteur de base de données relationnelle accessible par le langage SQL, la bibliothèque JavaScript Socket.IO pour la communication bidirectionnelle en temps réel entre les clients et les serveurs, la bibliothèque Python Threading pour faciliter l'exécution des processus en parallèles et la bibliothèque python RPI pour contrôler les ports entrées/sorties GPIO de la Raspberry Pi 3. Les résultats de simulation de l'asservissement en vitesses effectués sur un robot mobile 2WD en mode normal comme en Co-simulation nous révèlent des indices de performance pratiquement identique.

MOTS-CLEFS: Co-simulation, Raspberry Pi, GPIO, Robot mobile 2WD, Correcteur RQLP.

Corresponding Author: Manga Betene Ignace Fabrice

1 INTRODUCTION

La téléopération signifie "effectuer un travail à distance", bien que par "travail", nous entendions presque tout. Ce que nous entendons par "distance" est également vague: il peut s'agir d'une distance physique où l'opérateur est séparé du processus par une grande distance, mais elle peut également faire référence à un changement d'échelle par exemple, un chirurgien peut utiliser un micromanipulateur pour effectuer une opération à un niveau microscopique. En industrie La Téléopération comprend une technologie robotique dans laquelle un opérateur humain (maître) contrôle un processus distant (esclave). Le système est formé de deux parties, le module de commande et le télémanipulateur [1]. Le processus qui est l'élément essentiel de commande utilise en fait différentes ressources pour transformer des données d'entrée en données de sorties [2]. En effet, au sens de l'automatique, un processus est défini comme un système possédant une entrée, une sortie et dont la représentation du modèle mathématique est soit statique, soit dynamique [2]. La commande de ces processus se fait généralement dans des salles aménagées appelées laboratoires [3], certains de ces laboratoires nécessitent la présence humaine pour la commande et d'autres peuvent s'en passer [4]. Ces laboratoires ont la possibilité en fonction de leur configuration serveur de permettre la commande d'un ou plusieurs processus [4]. La demande se faisant grande, seuls les laboratoires ayant une configuration multiserveur (proposant la commande de plusieurs processus) sont les plus prisés, or ils présentent certaines limites, notamment l'encombrement et le coût élevé des équipements. Il se pose donc un problème, celui de pouvoir facilité la commande des processus téléopérables sur des plates-formes serveurs en utilisant le moins d'espace possible et un matériel au moindre coût qui donne entière satisfaction. Dans le cadre de nos travaux nous avons opté pour l'utilisation d'un module Raspberry pi pour proposer une optimisation de la commande des processus téléopérables sur les plates-formes serveurs. Ce choix est consolidé par les nombreux avantages que peut nous offrir l'utilisation de ce composant comparer à d'autres comme le kit Arduino ou le module de commande ESP32. En effet, la Raspberry Pi est un nano ordinateur mono-carte qui contient un processeur ARM, un GPU, une RAM, et d'autres ports comme HDMI, RJ45 (pas dans tous les modèles), des pins où on peut enficher d'autres composants comme une caméra, des détecteurs. La Raspberry Pi utilise le model SOC ou *System on chip* c'est-à-dire tous les composant précédents se trouvent sur une seule carte. Ce nano ordinateur supporte le système d'exploitation Debian basé sur linux compatible même avec Windows OS [5]. Les recherches effectuées nous montrent que diverses utilisations ont déjà été faites de la Raspberry Pi, ce module a été utilisé pour configurer et mettre en place un serveur Web avec une adresse IP et une redirection de port, qui permettrait l'accès à partir d'une autre source connectée à un réseau [6], il a été utiliser pour l'Implémentation d'un Server Cloud pour le stockage de données en temps réel [7], il a été utiliser dans la mise sur pied d'un système de surveillance de l'environnement [8], des travaux ont portés sur la Configuration du module Raspberry Pi pour recevoir et décoder les signaux FM (Frequency Modulation) et les envoyer à un autre ordinateur sur un réseau local [9], La mise sur pied d'un système définissant un petit intranet qui peut être utilisé par n'importe quelle organisation pour communiquer en lui-même sans frais a également fait appel à l'utilisation d'un module Raspberry [10], certains systèmes intelligents pour le contrôle de la température et l'humidité dans des salles serveurs font également usage de module Raspberry [11]. Des modules Raspberry sont même utilisés dans la mise sur pied des laboratoires distant [12]. Il existe bien d'autres utilisations de ce composant qui ont consolidé notre choix de le configurer comme serveur pour l'optimisation de la commande des processus. Cependant plusieurs processus sont utilisés dans la littérature [13], [14], pour des besoins de téléopérabilité à savoir: les robots manipulateurs, les robots mobiles. Cependant la commande de ces systèmes dynamiques est généralement en boucle ouverte ou en boucle fermée, et l'utilisation de cette dernière typologie de commande est pour des besoins d'amélioration de performances statiques et dynamiques des dits systèmes. En outre l'évaluation de ces indices de performances s'obtient par synthèse de contrôleurs qui peuvent être de type classique, optimaux, robuste. Le processus utilisé pour des besoins de téléopérabilité est un robot mobile 2WD dont le modèle mathématique intègre les modèles dynamiques et cinématiques. Un contrôleur optimal RQLP (Régulateur Quadratique Linéaire avec Poursuite) est synthétisé afin d'asservir les vitesses angulaires des moteurs associées au robot. Les consignes de vitesses linéaire et angulaire sont données par l'intermédiaire d'un client http et envoyé à MATLAB/ Simulink à travers un client Socket pour simulation, ce qui conduit à une interaction ***hardware/software*** et en d'autres termes une Co-simulation. Dans cet article le travail est organisé comme suit: La section 2 porte sur les outils et méthodes utilisés, la section 3 porte sur les résultats de simulation et discussion, enfin la section 4 porte sur la conclusion de notre article.

2 OUTILS ET METHODES

2.1 DISPOSITIF PROPOSE

Le schéma de la figure 1 illustre la structure architecturale du système. Il est constitué des utilisateurs à qui une interface de commande est proposée, un module Raspberry pi 4 model B configurer en serveur, une base de données, un Modem, une

Raspberry Pi 3model B qui joue le rôle de client Socket qui interagit le processus modélisé avec le logiciel MATLAB R2018a installé dans un ordinateur:

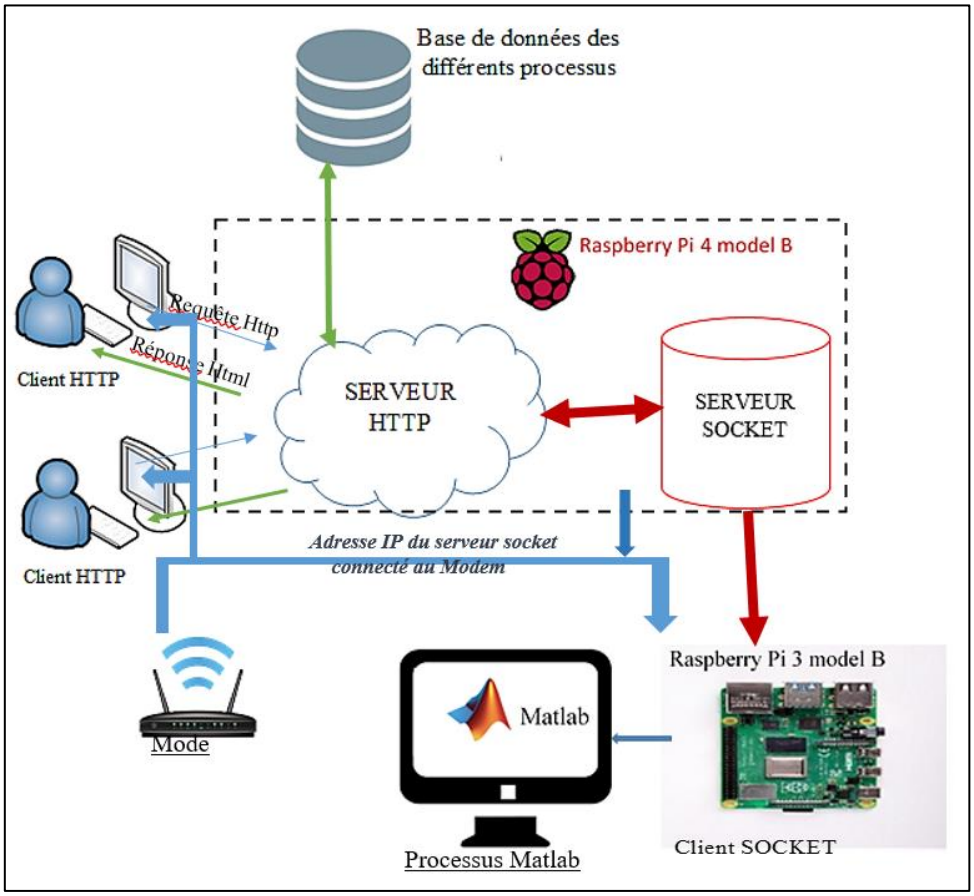

Fig. 1. Schéma bloc du système

2.2 MODULES ÉLECTRONIQUES UTILISES

La Raspberry Pi est un nano ordinateur mono-carte qui contient un processeur ARM, un GPU, une RAM, et d'autres ports comme HDMI, RJ45 (pas dans tous les modèles), des pins où on accroche des autres composants comme une caméra, des détecteurs. La Raspberry Pi utilise le model SOC ou *System on chip* c'est-à-dire tous les composant précédents se trouvent sur une seule carte. Ce nano ordinateur supporte le système d'exploitation Debian basé sur linux compatible même avec Windows OS [15]. Il existe plusieurs modèles de Raspberry Pi sur le marché, les modèles B des Raspberry Pi 4 et 3 que nous avons utilisés dans notre travail sont illustrés dans les tableaux 1 et 2 ainsi que leurs caractéristiques techniques.

Tableau 1. Caractéristiques techniques de la Raspberry Pi 4 modele B [15]

Caractéristiques	Désignations	
Carte mère	Raspberry Pi 4	
Processeur	Broadcom BCM2711, quad-core Cortex-A72 (ARM v8) 64-bit SoC @ 1.5GHz	
RAM	8 Go LPDDR4	
GPU	Video Core VI prenant en charge OpenGL ES 3.0, décodage HEVC 4K à 60 i/s	
Connexion sans fil	Bluetooth 5.0, Wi-Fi 802.11b/g/n/ac	
Connexion filaire	Gigabit Ethernet (RJ45)	
Poids	40 g	
Port caméra	CSI pour connecter la caméra Raspberry Pi	
Port d'affichage	DSI pour connecter l'écran tactile Raspberry Pi	
Audio	AV 3.5 mm	
Ports	2 x USB 3.0 / 2 x USB 2.0 / 1 x USB-C (alimentation seulement) / 1 x GPIO 40 pin / 1 x port quadripôle Audio/Vidéo composite / 2 x micro-HDMI	
Alimentation	5V DC via un connecteur USB-C (minimum 3A), 5V DC via un en-tête GPIO (minimum 3A), compatible Power over Ethernet (PoE) (nécessite un HAT pour PoE)	
Dimensions	85,60 mm × 56,5 mm × 11 mm	

Tableau 2. Caractéristiques techniques de la Raspberry Pi 3 modele B [16]

Caractéristiques	Désignations	
Cadencement	1,2 GHz	
Puce (SoC)	Broadcom BCM2837	
Processeur	ARM Cortex-A53 64 bits quatre cœurs	
Processeur graphique	Broadcom VideoCore IV double coeur (OpenGL ES 2.0, H.264 Full HD à 30 ips)	
Mémoire (SDRAM)	1GB LPDDR2	
Nombre de ports USB 2.0	4	
Port extension	GPIO 40 pin	
Sorties vidéos	HDMI et RCA, plus 1 connecteur de caméra CSI	
Sorties audio	Stéréo Jack 3,5mm ou HDMI	
Sauvegarde des données	Carte MicroSD	
Connexion réseau	10/100 Ethernet, WiFi 802.11n et Bluetooth 4.1 (BLE - Low Energy)	
Périphériques	17 × GPIO	
Alimentation	5v 2.5A via micro-USB	
Dimensions	85,60 mm × 53,98 mm × 17 mm	
Poids	45 g	

2.3 FONCTIONNEMENT

L'utilisateur une fois connecter au système après avoir entré son login et son mot de passe, choisis de commander le processus. Il entre des données de commandes qui sont transmises au serveur http sous forme de requêtes http. Le serveur http transmet ces données au serveur socket qui les achemine vers le client socket. Le client socket dirige les données vers le processus afin qu'il réalise la commande souhaitée par l'utilisateur. Une fois la commande exécutée l'information de réponse

empreinte le chemin inverse afin d'être envoyée à l'utilisateur. Il faut noter que chaque fois qu'un utilisateur effectue une commande, les dates de départ et de réponse ainsi que l'historique de la commande sont consignées dans la base de données. Cependant pour que cette commande soit possible, il faudrait au préalable que la connexion entre le client socket et le serveur socket existe. Au début pendant le démarrage des serveurs http et socket, la connexion s'établie par un jeu d'échange sous forme de question réponse. Pendant l'initialisation les informations circulent dans le réseau une fois la connexion en local établie. L'identificateur du serveur Socket est celui-là qui permet de faire la différence entre lui et le client socket. La figure 4 présente d'une part l'initialisation de la communication et d'autre part l'échange de données entre le client socket et le serveur socket.

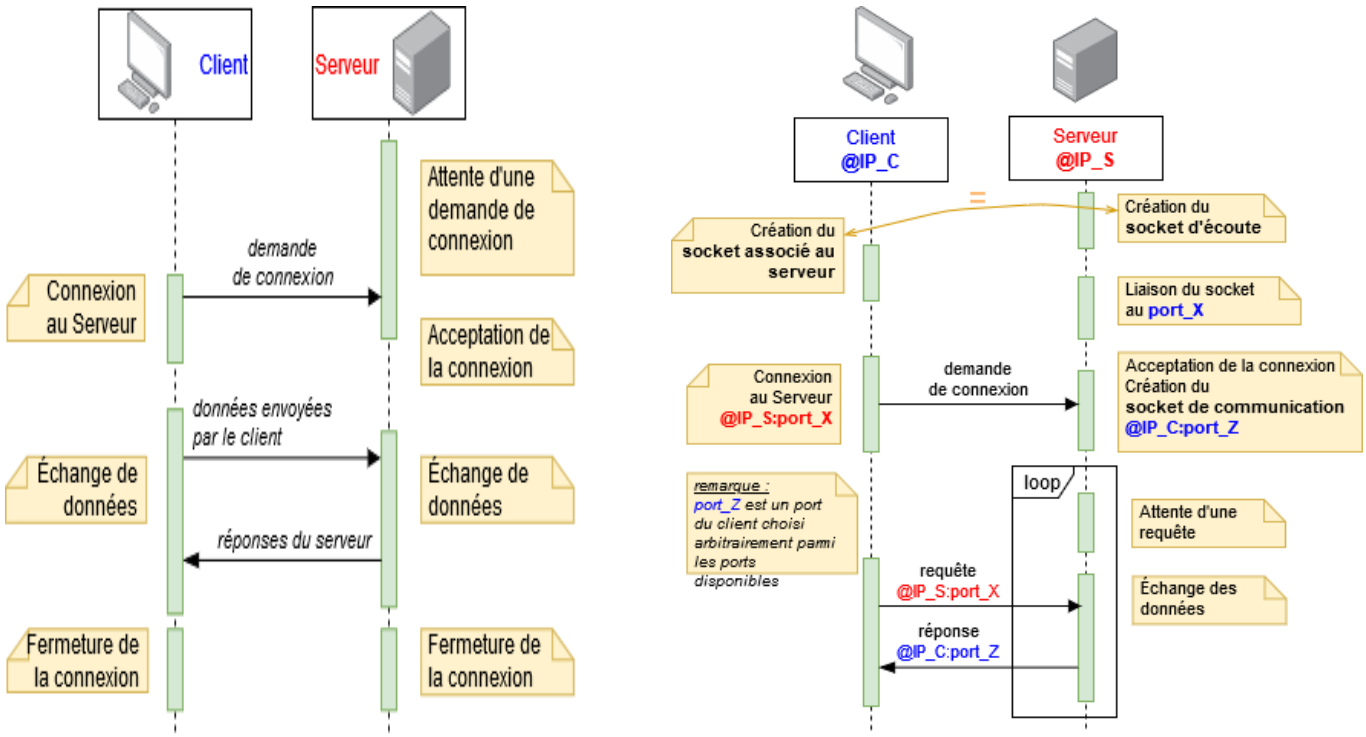

Fig. 2. Illustration de la connexion et de l'échange de données entre le client socket et le serveur socket

2.4 CHOIX TECHNOLOGIQUES DE LA CONFIGURATION SOCKET

Le tableau 3 présente les avantages qu'offre l'utilisation des caractéristiques que nous avons choisis pour notre système:

Tableau 3. Avantages des configurations et de la base de données

Configurations	Avantages
La configuration Serveur Socket-Serveur http	• Les protocoles (http et socket) sont des protocoles de grandes distances (à la portée d'internet) utilisant l'adresse IP pour communiquer; • En local les serveurs peuvent être regrouper dans le même matériel (Raspberry Pi); • En réseau internet il sera possible d'utiliser un hébergeur comme serveur Web et un Raspberry Pi présent dans le local comme Serveur socket; • Le nombre de client Socket ne dépend que du nombre d'hôte possible à distribuer en réseau et est donné par la relation: $N_{client\_socket} = N_{hote\_possible} - N_{serveurs}$ • En réseau internet, le nombre de serveurs socket est infinie.
La configuration Serveur Socket-Client Socket	• Cette disposition nous permet de gérer la configuration en logicielle passant le client http et le client socket; • Nous pouvons modifier le nombre processus à chaud sans endommager le système grâce à l'assignation dynamique des identifiants (ID) des clients socket dans le serveur socket; • La portée de la commande dépend de la portée du réseau WIFI et peut être extensive en utilisant les répéteurs.
L'utilisation d'une base de données	• Permet d'assurer la traçabilité des informations; • Permet aux clients d'effectuer des traitements statistiques concernant l'historique des différentes commandes effectuer.

Le système ayant été présenter en détails ainsi que son fonctionnement, dans la suite nous présenterons la modélisation, les tests ainsi que les résultats obtenus lors de l'asservissement d'un robot mobile 2WD en utilisant notre système de cosimulation.

2.5 MODELISATION DE LA DYNAMIQUE MOTEUR ET CONCEPTION DU CONTROLEUR RQLP DU ROBOT

Le processus utilisé comme plateforme matérielle à piloter est celui de Markov MBIHI dans [17]. Ce processus est un robot mobile 2WD, dont la fonction de transfert obtenue par modélisation expérimentale et à partir de la boîte à outils MATLAB, ***SystemIdentification***, est illustrée par l'équation 1.

$$G(s) = \frac{U(s)}{\Omega(s)} = \frac{41.83}{s^2 + 8.699s + 12.91} \quad (1)$$

Avec :

$-\ U(s)$ *la tension d'entrée de commande*;

$-\ \Omega(s)$ *la vitesse angulaire d'un moteur*;

Par ailleurs, pour synthétiser un correcteur à rétroaction d'état, il faudrait, transformer la fonction de transfert de l'équation 1, sous forme d'espace d'état. Cependant, par utilisation de la macro-commande « ***ss*** » de MATLAB, nous pouvons rapidement effectuer cette transformation sous forme commandable nous permettant d'appliquer ultérieurement une loi de commande sur le robot mobile 2WD à piloter. La formulation mathématique de la dynamique du moteur sous forme d'espace d'état est illustrée à l'équation 2.

$$\begin{cases} \dot{x} = Ax + Bu \\ y = Cx + Du \end{cases} \quad (2)$$

$$Avec \quad A = \begin{bmatrix} -8.699 & -3.228 \\ 4 & 0 \end{bmatrix}, \quad B = \begin{bmatrix} 4 \\ 0 \end{bmatrix}, \quad C = \begin{bmatrix} 0 & 2.614 \end{bmatrix} \quad et \quad D = 0$$

Où $A, B, C\ et\ D$ étant respectivement les matrices d'état, de commande, d'observation et d'action directe du système. Par ailleurs, x, $u\ et\ y$ représentent respectivement un vecteur d'état du moteur, la tension de commande d'entrée du moteur, la grandeur physique de sortie du moteur à observer.

Pour s'assurer de la poursuite de la grandeur de sortie, nous ajoutons un nouvel état intégral conformément aux recommandations de Markov MBIHI dans [17]; on obtient le nouveau système d'équation 3 obtenu dans [17].

$$\begin{cases} \dot{x} = Ax + Bu \\ \dot{x}_i = y - y_{ref} \\ y = Cx \end{cases} \rightarrow \begin{cases} \begin{bmatrix} \dot{x} \\ \dot{x}_i \end{bmatrix} = \underline{A}\begin{bmatrix} x \\ x_i \end{bmatrix} + \underline{B}\,u - \begin{bmatrix} 0 \\ 0 \\ yref \end{bmatrix} \\ y = \underline{C}\begin{bmatrix} x \\ x_i \end{bmatrix} \end{cases} \tag{3}$$

$$Avec \quad \underline{A} = \begin{bmatrix} -8.699 & -3.228 & 0 \\ 4 & 0 & 0 \\ 0 & 2.614 & 0 \end{bmatrix}, \quad \underline{B} = \begin{bmatrix} 4 \\ 0 \\ 0 \end{bmatrix}, \quad \underline{C} = \begin{bmatrix} 0 & 2.614 & 0 \end{bmatrix}$$

$\underline{A}, \underline{B}\ et\ \underline{C}$ étant respectivement les matrices d'état, de commande, d'observation et d'action directe du système. Par ailleurs $x,\ x_i, u\ et\ y$ représentent respectivement un vecteur d'état scalaire du moteur, un état intégral non mesurable, la tension de commande d'entrée du moteur, la grandeur physique de sortie du moteur à observer.

Afin, de déterminer les gains de Kalman, il est nécessaire de spécifier les valeurs des matrices $Q\ et\ R$, étant respectivement les matrices de pénalisation des états du système et de la commande en tension à appliquer au système, généralement choisies par essai-erreur. Ce gain de Kalman à déterminer sous horizon infini et illustré à l'équation 4 obtenu dans [3], doit minimiser la fonction coût illustrée par l'équation 5.

$$K = R^{-1}B^T P \tag{4}$$

$$Avec \quad P\ la\ matrice\ symétrique\ sous\ optimale\ définie\ positive$$

$$J = \int_0^\infty x^T(t)Qx(t) + u^T(t)Ru(t) \tag{5}$$

Par utilisation de la macro-commande MATLAB ***LQR***, le gain de Kalman se détermine rapidement en spécifiant les matrices $\underline{A}, \underline{B}, Q\ et\ R$. Le schéma bloc MATLAB de la commande optimale RQLP (Régulateur Quadratique Linéaire avec Poursuite) est illustré à la figure 3.

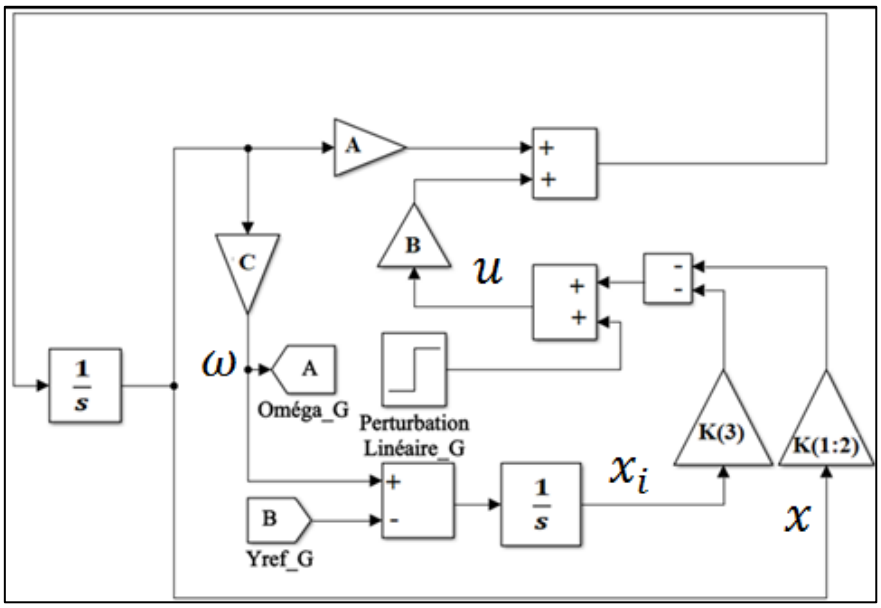

Fig. 3. Schéma bloc de la commande RQLP du moteur gauche du robot mobile

2.6 MODELISATION CINEMATIQUE DU ROBOT MOBILE

La figure 4 représente la géométrie du robot mobile sur le plan à partir duquel nous pouvons déterminer le système d'équation 6.

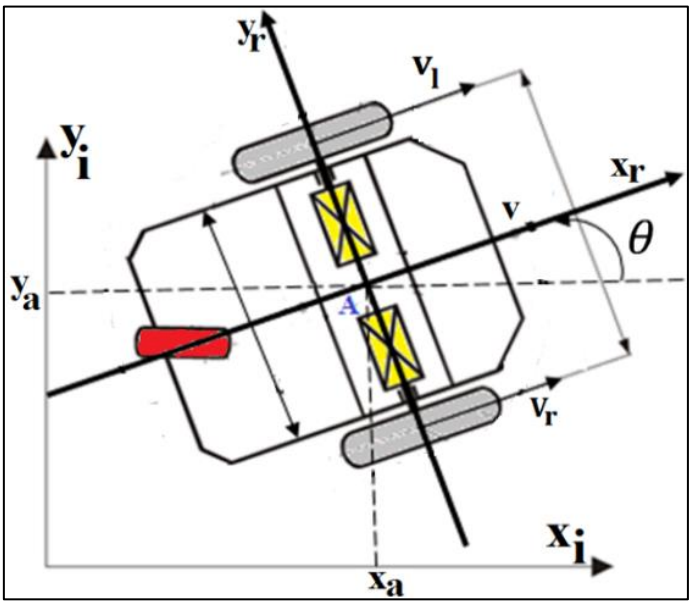

Fig. 4. Digramme cinématique du robot mobile 2WD [17]

$$\begin{cases} \dot{x}_a = v \cdot cos\theta \\ \dot{y}_a = v \cdot \sin\theta \end{cases} \quad (6)$$

Nous pouvons également ressortir les vitesses linéaire et angulaire du robot mobile illustrées par les équations 6 et 7 obtenues dans [17].

$$\begin{cases} v = \dfrac{v_l + v_r}{2} = R\dfrac{\omega_r + \omega_l}{2} & (a) \\ \omega = \dot{\theta} = R\dfrac{\omega_r - \omega_l}{2L} & (b) \end{cases} \quad (7)$$

Avec $v - La$ vitesse linéaire;

$R - Le\ rayon\ des\ roues$;

$\omega - La$ vitesse angulaire;

$L - Distance\ entre\ une\ des\ roues\ et\ le\ centre\ de\ masse$;

$\omega_r, \omega_l - Vitesse$ angulaire droite *et* gauche;

Le système d'équation 7, matérialisant la cinématique directe du robot mobile, nous pouvons également déterminer la cinématique inverse du robot par le système d'équation 8 obtenu dans [17].

$$\begin{cases} \omega_r = \dfrac{v + L\omega}{R} & (a) \\ \omega_l = \dfrac{v - L\omega}{R} & (b) \end{cases} \quad (8)$$

2.7 PRINCIPE D'ACQUISITION DES DONNEES NUMERIQUES ET SCHEMA BLOC GLOBAL DU ROBOT

Une des difficultés rencontrées dans la Co-simulation est la transmission des données de l'utilisateur à Matlab en passant par nos Raspberry. En effet, l'utilisateur envoi des données décimales notamment, la vitesse linéaire et la vitesse angulaire de rotation du robot. Pourtant la Raspberry ne communique qu'en données binaires. Pour pallier à cela, nous avons au niveau du client Socket écrit un code python dont l'algorithme est illustré ci-dessous, dont le but est de transformer les données décimales entrées par l'utilisateur, en données binaires utilisables par les ***GPIO*** (***G***eneral ***P***urpose ***I***nput/***O***utput) de la Raspberry Pi 3 Model B.

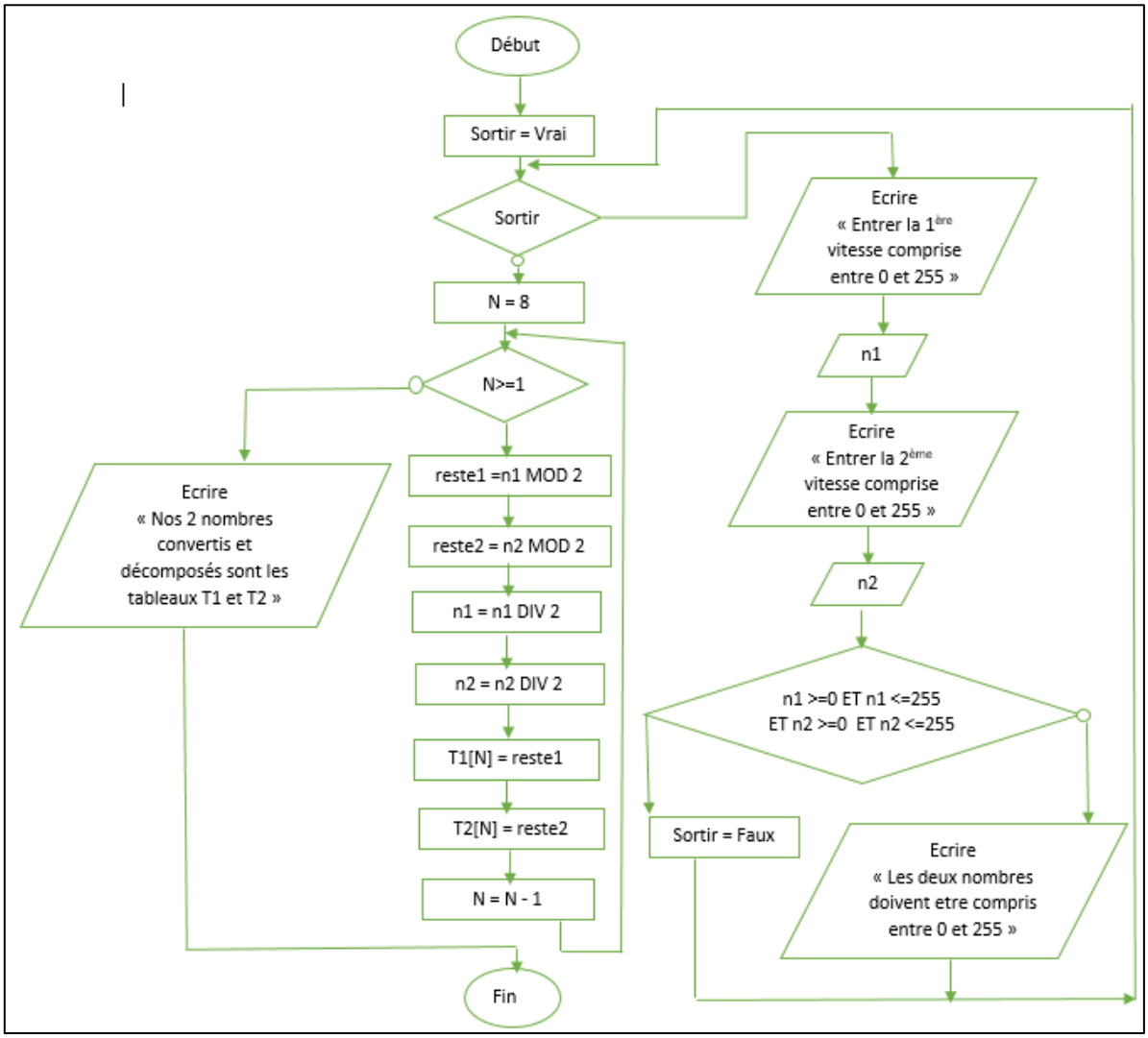

Fig. 5. Organigramme de conversion Décimale-Binaire

Une fois les données binaires transmises, au niveau de Simulink elles sont récupérées par un multiplexeur qui les confine dans une matrice. La matrice obtenue transite par la fonction ***b2d*** de Matlab qui fera la conversion binaire en décimal afin que ces données deviennent les consignes du robot 2WD comme illustré à la figure 6.

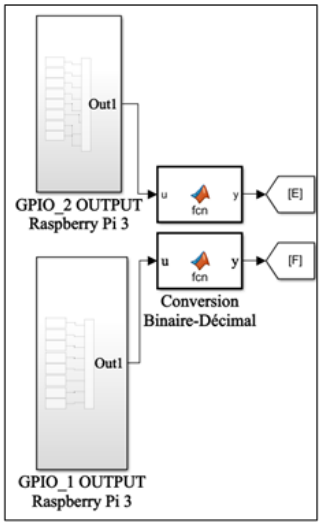

Fig. 6. Schéma bloc Simulink d'acquisition et de conversion de données numériques

Le principe d'acquisition et de conversion de données étant modélisé, ainsi que le modèle cinématique directe et inverse illustré par les équations 7 et 8, puis la dynamique des moteurs sous forme d'espace d'état intégrant notre régulateur RQLP aussi modélisé sous forme de schéma bloc Simulink, nous pouvons avoir le modèle global de notre processus à commander comme illustré à la figure 7.

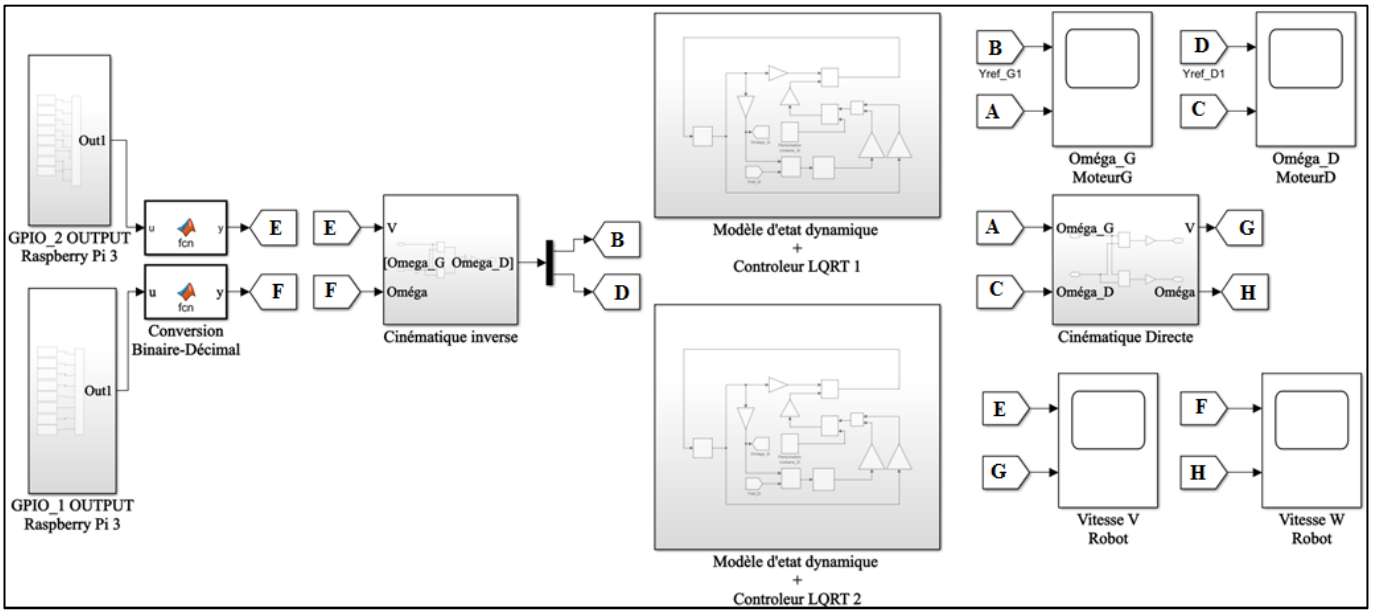

Fig. 7. Schéma bloc globale du robot mobile 2WD

3 RESULTATS ET DISCUSSIONS

3.1 BANC D'ESSAI D'EXPERIMENTATION

L'image du banc d'essai d'expérimentation est présentée dans la figure 8. On y retrouve les éléments:

a) Raspberry Pi 4 modèle B, b) Raspberry Pi 3 modèle B, c) Un modem, D) Un ordinateur portable, E) Un moniteur.

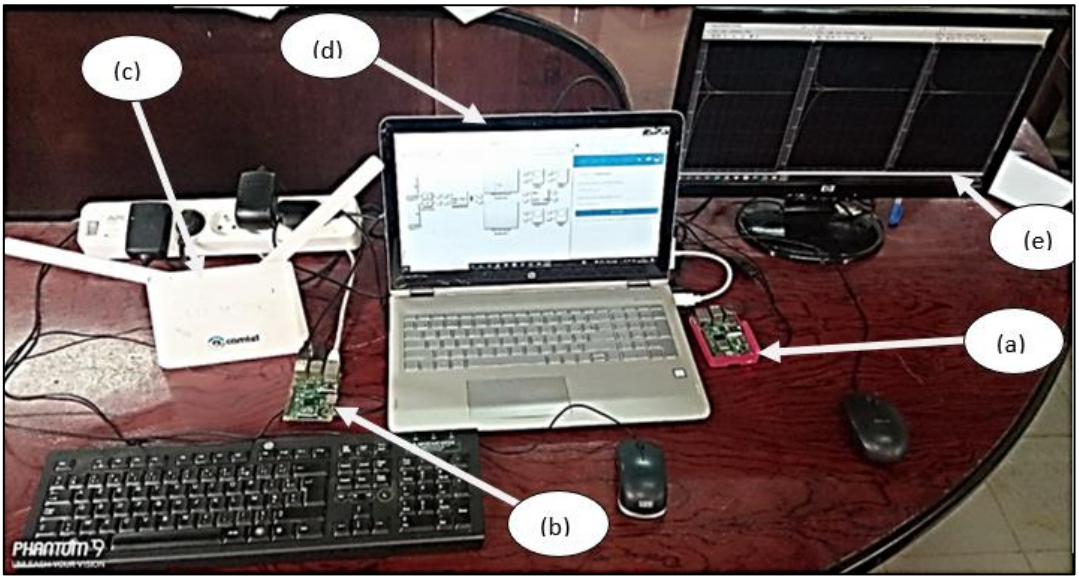

Fig. 8. Image du banc d'essais expérimental

3.2 INTERFACES LOGICIEL POUR LA COMMANDE DU ROBOT MOBILE 2WD

Pour faciliter l'utilisation de notre système, nous avons programmé une interface utilisateur à partir de laquelle un utilisateur désirant utiliser notre système pour commander le robot mobile 2WD pourra se connecter et envoyer des valeurs d'entrées de commande qui sont les vitesses angulaires et linéaires du robot. La figure 9 illustre nos interfaces.

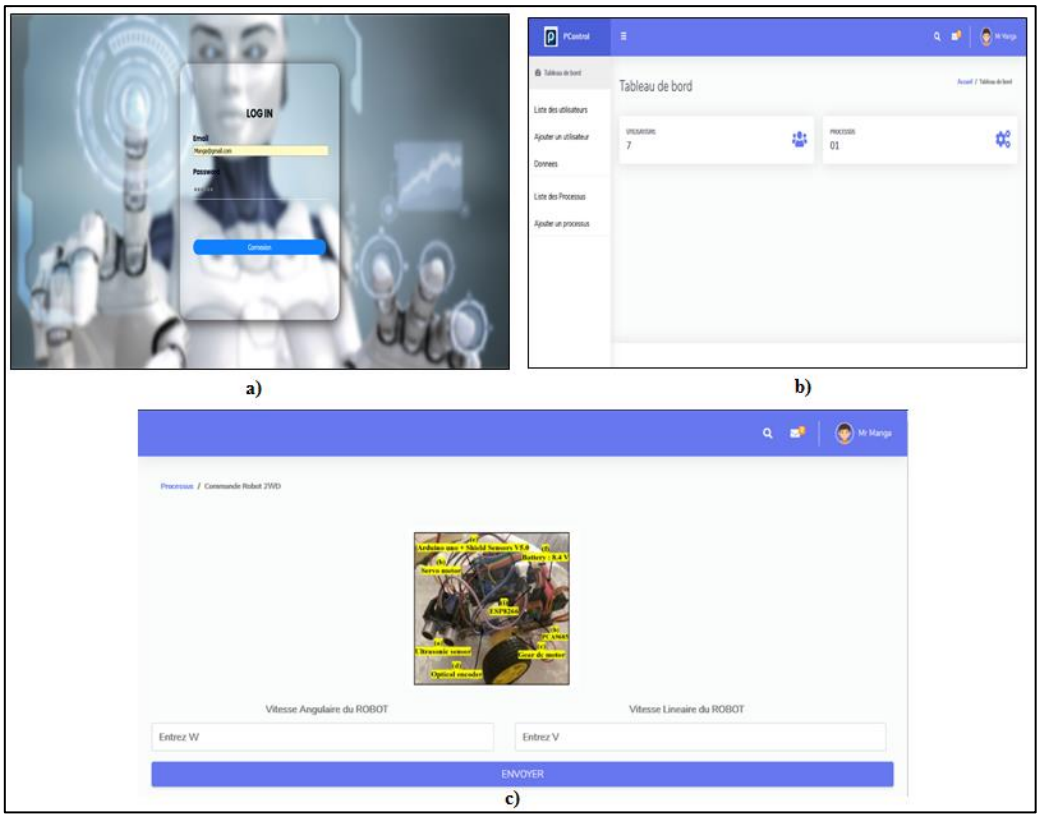

Fig. 9. Interfaces de l'application utilisateur pour la commande du robot mobile 2WD

La figure 9.a illustre l'interface de connexion au système, la figure 9.b illustre le tableau de bord et la figure 9.c représente l'interface utilisateur om ce dernier peut envoyer les consignes de vitesses linéaires et angulaires.

3.3 SIMULATION DE LA VITESSE ANGULAIRE D'UN MOTEUR DU ROBOT MOBILE SANS PERTURBATION

Les gains du régulateurs RQLP utilisés pour la simulation sont ceux de [17] et illustrés à l'équation 9.

$$\begin{cases} Q = \begin{bmatrix} 50.8884 & 0 & 0 \\ 0 & 113.7679 & 0 \\ 0 & 0 & 75.8359 \end{bmatrix} \\ R = 43.1824 \\ K = \begin{bmatrix} 1.0247 & 2.1641 & 1.3252 \end{bmatrix} \\ P = \begin{bmatrix} 11.0617 & 23.3626 & 14.3064 \\ 23.3626 & 74.3239 & 45.7719 \\ 14.3064 & 45.7719 & 65.0430 \end{bmatrix} \end{cases} \quad (9)$$

Ces matrices de pondération Q et R, sont obtenus par optimisation en utilisant l'algorithme d'optimisation des particules d'essaim explicité dans [17]. La simulation avec ses paramètres nous révèle que les vitesses obtenues en simulation et en Co-simulation, comme illustrées à la figure 10, sont pratiquement identiques au regard des indices de performances illustrées dans le tableau 4.

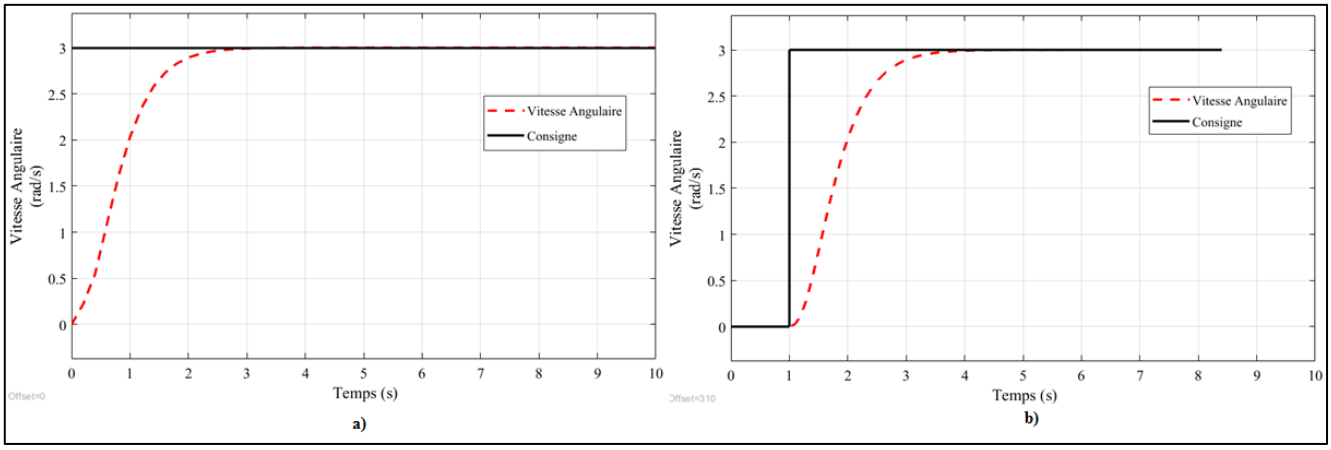

Fig. 10. Vitesses angulaires d'un moteur sans perturbation

L'analyse des performances en régime transitoire, est illustrée dans le tableau 4.

Tableau 4. Indices de performances en régimes transitoires-Vitesse angulaire du moteur

	Temps de réponse à 5% (s)	Dépassement (%)	Temps de montée (s)
a	2.352	0	3.757
b	2.330	0	3.780

3.4 SIMULATION DE LA VITESSE ANGULAIRE ET LINEAIRE DU ROBOT MOBILE SANS PERTURBATION

La figure 11 illustrée, représente les différentes courbes de vitesses linéaires simulées du robot mobile 2WD. Selon les indices de performances illustrés au tableau 5, nous constatons que ces indices sont pratiquement les mêmes en simulation normale et en Co-simulation.

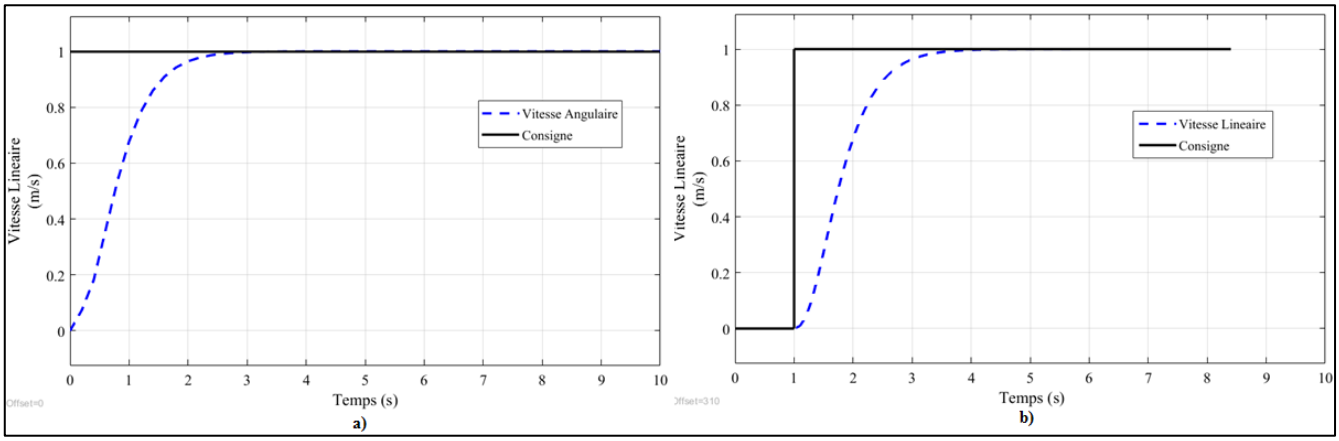

Fig. 11. Vitesses Linéaires du robot mobile Sans perturbation

Tableau 5. Indices de performances en régimes transitoires-Vitesse linéaire du robot

	Temps de réponse à 5% (s)	Dépassement (%)	Temps de montée (s)
a	1.875	0	3.757
b	1.864	0	3.780

Nous constatons également que pour la valeur de consigne de 2 rad/s envoyée depuis le client http, les vitesses angulaires sont toutes asservies et illustrées à la figure 12. Ces vitesses sont obtenues par passage du bloc cinématique directe dont le modèle mathématique est illustré à l'équation 7. Conformément aux résultats précédents, les vitesses angulaires du robot ont des performances semblables au regard des indices illustrées au tableau 6.

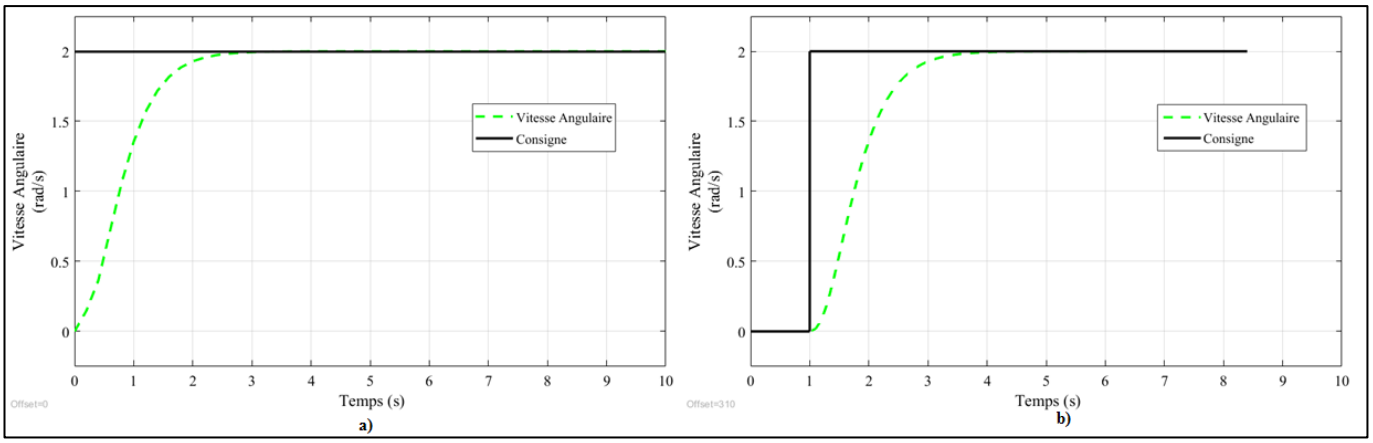

Fig. 12. Vitesses Angulaires du robot mobile Sans Perturbation

Tableau 6. Indices de performances en régimes transitoires-Vitesse angulaire du robot

	Temps de réponse à 5% (s)	Dépassement (%)	Temps de montée (s)
a	2.177	0	3.757
b	2.159	0	3.780

3.5 Simulation De La Vitesse Angulaire D'un Moteur Du Robot Mobile Avec Perturbation

Afin d'évaluer la robustesse de notre régulateur optimal, une perturbation linéaire de 1 Volt est ajoutée à partir de la sixième seconde. Cette perturbation est rapidement stabilisée, comme illustrée à la figure 12, pour retrouver la consigne de 3

rad/s envoyée depuis de notre client http. Nous constatons que la figure 13.a, étant celle obtenue dans [17], et la figure 13.b, étant celle obtenue par Co-simulation sont pratiquement semblables et ne révèlent aucune dégradation en régime transitoire comme au régime permanent. Le régulateur ainsi utilisé dans [17], reste efficace performant tant en simulation mode ***normal*** sur Matlab, comme en Co-simulation en mode ***external***.

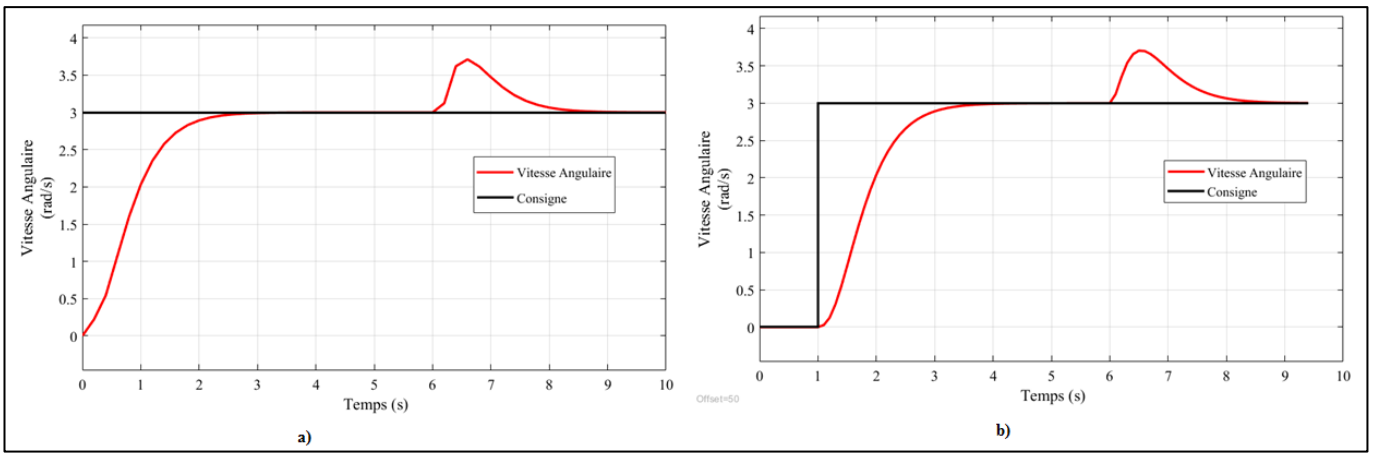

Fig. 13. Vitesses Angulaires d'un moteur Avec perturbation linéaire 1 Volt

3.6 SIMULATION DES VITESSES LINEAIRE ET ANGULAIRE DU ROBOT MOBILE AVEC PERTURBATION

La même analyse est effectuée pour évaluer le comportement de la vitesse angulaire du robot mobile. Il en ressort que lorsque l'on perturbe le fonctionnement du moteur du robot, la vitesse angulaire se retrouve perturbée et ce qui perturbe globalement la vitesse angulaire du véhicule, comme illustré à la figure 14, tant en simulation mode ***normal*** sur Matlab illustré à la figure 14.a, comme en Co-simulation en mode ***external*** à la figure 14.b.

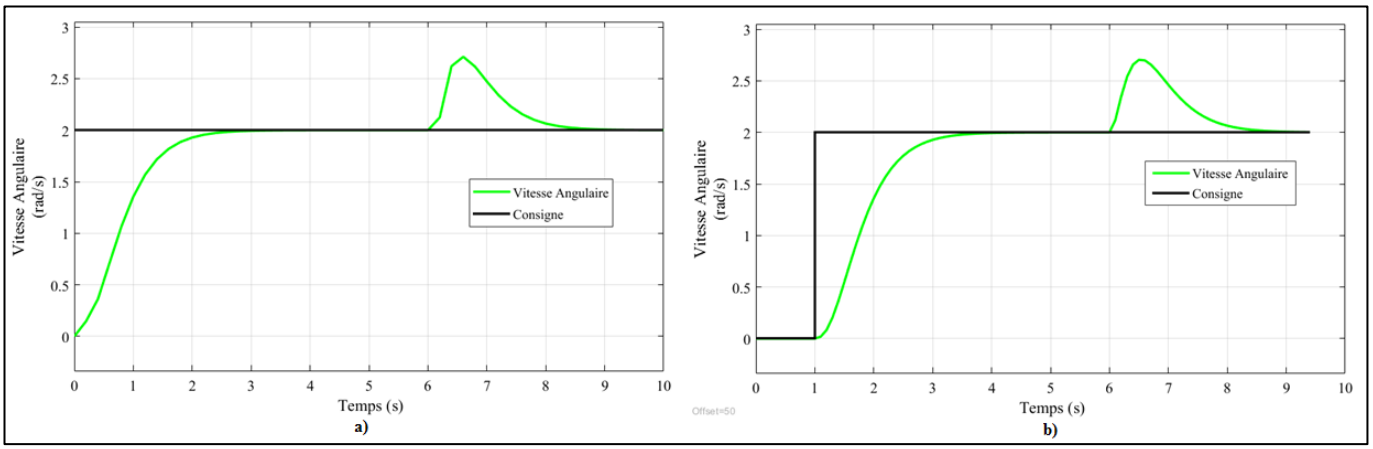

Fig. 14. Vitesses Angulaires du robot mobile avec perturbation linéaire de 1 Volt

Cependant, nous constatons, que les vitesses linéaires du robot mobile, illustrée à la figure 15, ne sont pratiquement pas influencées par le flux de perturbation linéaire.

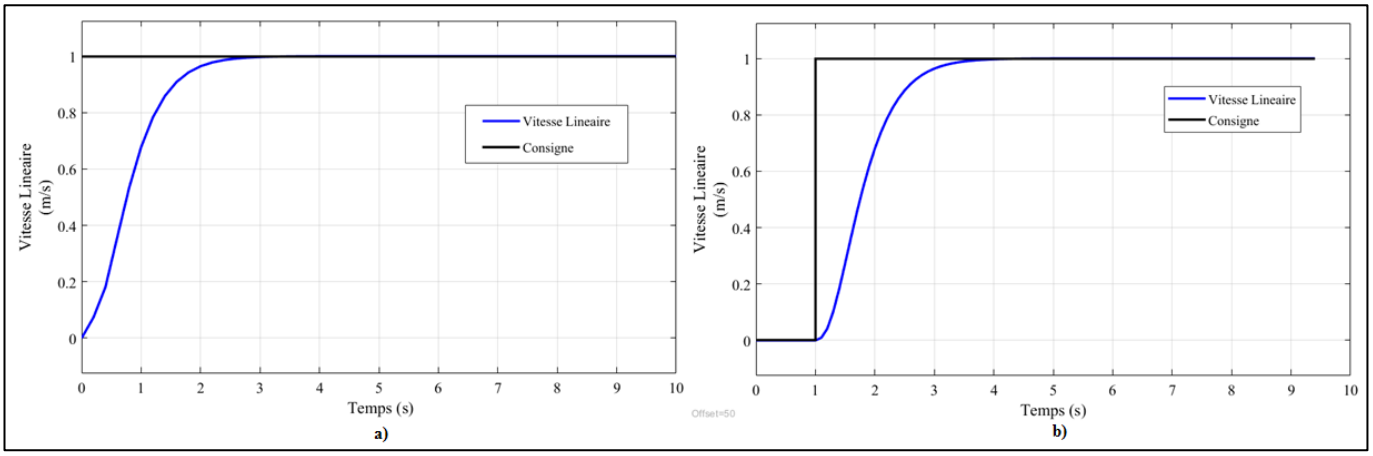

Fig. 15. Vitesses Linéaires du robot mobile Avec perturbation linéaire de 1 Volt

4 CONCLUSION

Cet article est la première Configuration expérimentale d'une Co-simulation matérielle et logicielle. Elle met en exergue un module Raspberry Pi 4 model B configurée comme serveur, un module Raspberry Pi 3 model B configurée comme client socket et le logiciel Matlab R2018a utilisée pour la mise en œuvre de stratégie de commande optimale RQLP appliquée à notre robot mobile 2WD. Les résultats obtenus de notre Co-simulation comparés à ceux d'une simulation normale et directe sur le même logiciel Matlab, nous montre que l'environnement réseau de communication ne modifie pas de façon sensible les performances de la boucle de simulation de commande de notre robot mobile 2WD. Il s'agit également dans cet article d'une nouvelle approche optimale et modulable de l'utilisation de la technologie socket, pour améliorer les performances de temps de rafraichissement lors d'une commande de processus. Cependant, il s'agit d'une téléopérabilité effectuée dans un réseau LAN. Il sera par conséquent question dans un travail futur d'une téléopérabilité dans un réseau WAN afin d'évaluer les performances de temps de rafraichissement qui seront certainement influencés par des contraintes de connectivité internet.

REMERCIEMENTS

Manga Betene Ignace Fabrice a apporté sa modeste contribution dans la mise sur pied de la configuration des Raspberry en Serveur, en client socket. Il a mis sur pied Une plateforme web d'interaction pour l'envoi des consignes de vitesse linéaire et angulaire du robot mobile. Il a apporté une nouvelle approche de Co-simulation matérielle et logicielle pour un concept de téléopérabilité effectivement fonctionnel dans un réseau WAN.

Paune Felix a contribué à la définition des objectifs du travail, de la démarche méthodologique à adopter pour la finalisation du manuscrit.

Mbihi Djoumessi Markov a contribué à la mise sur pied d'un contrôleur optimal linéaire quadratique avec poursuite, à la modélisation mathématique du processus dynamique a commandé et à la détermination des meilleurs paramètres de pondération des états et de la commande pour l'aboutissement des meilleurs performances.

Nneme Nneme Leandre a contribué à la recherche scientifique de bout à bout tant sur le plan d'une supervision générale, sur la partie évaluation ainsi que l'apport de son expertise sur tout l'ensemble de cet article, son expertise rigoureuse et à fond de l'ensemble du manuscrit.

REFERENCES

[1] S. Lichiardopol, "A Survey on Teleoperation," Eindhoven, 2007.
[2] P. Borne and J. P. Richard, Modélisation et identification des processus, TECHNIP. 1992.
[3] J. Mbihi, Techniques avancées et technologie de commande et régulation assistée par ordinateur, ISTE. London, 2018.
[4] F. Paune and J. Mbihi, "A Novel Web-Based Laboratory for Remote Control of Power Lighting Processes," WSEAS TRANSACTIONS on ADVANCES in ENGINEERING EDUCATION, vol. 13, pp. 7–19, 2016.
[5] E. Upton and H. Gareth, Raspberry Pi, PEARSON. 2013.
[6] M. ZiyaUlHaq, "Embedded Web server for Industrial Applications using Raspberry-Pi," International Journal of Engineering Science and Computing, vol. 7, no. 4, pp. 6371–6374, 2017.
[7] S. E. Princy and K. G. J. Nigel, "Implementation of Cloud Server for Real Time Data Storage using Raspberry Pi," in Online International Confernece on Green Engineering and Technologies, 2015, pp. 0–3.
[8] G. Jadhav, K. Jadhav, and K. Nadlamani, "Environment Monitoring System using Raspberry-Pi," International Research Journal of Engineering and Technology (IRJET), vol. 3, no. 4, pp. 1168–1172, 2016.
[9] R. Pi, R. Danymol, and T. Ajitha, "Real-Time Communication System Design using," in International Conference on Advanced Computing and Communication Systems, 2013, pp. 19–21.
[10] P. V Murkute and V. M. Deshmukh, "Implementing the VOIP Communication Principles using Raspberry Pi as Server," International Journal of Computer Applications, vol. 124, no. 4, pp. 34–38, 2015.
[11] R. Pi, W. Notifications, D. E. Kurniawan, M. Iqbal, J. Friadi, and R. I. Borman, "Smart Monitoring Temperature and Humidity of the Room Server Using Raspberry Pi and Whatsapp Notifications," in Journal of Physics: Conference Series, 2019, pp. 1–8, doi: 10.1088/1742-6596/1351/1/012006.
[12] A. Fernández-Pacheco, S. Martin, and M. Castro, "Implementation of an arduino remote laboratory with raspberry pi," IEEE Global Engineering Education Conference, EDUCON, vol. April-2019, pp. 1415–1418, 2019, doi: 10.1109/EDUCON.2019.8725030.
[13] A. Mustafa, "Sensor Noise Reduction with RHC and LQR for System with Backlash Nonlinearity," International Journal of Innovation and Applied Studies, vol. 5, no. 1, pp. 62–71, 2014.
[14] A. H. Ibrahim, L. Udeji, and H. K. Wye, "Design and Implementation of Robot Based Mobile Application for Humidity and Temperature Measurement," International Journal of Innovation and Applied Studies, vol. 11, no. 3, pp. 592–597, 2015.
[15] E. P. T. Kerja, Raspberry Pi 4 User Guide. 2019.
[16] M. B. Kalpana, M. T. Student, and R. R. Dist, "Online Monitoring Of Water Quality Using Raspberry Pi3 Model B," INTERNATIONAL JOURNAL OF INNOVATIVE TECHNOLOGY AND RESEARCH, vol. 4, no. 6, pp. 4790–4795, 2016.
[17] M. D. Mbihi, B. L. Moffo, and L. N. Nneme, "Design and Virtual Simulation of an Optimal PID / LQRT-PSO ControlSystem for 2WD Mobile Robots," algerian journal of signals and systems (ajss), vol. 6, no. 2, pp. 98–111, 2021.