

Année 2022/2023

# **INGÉNIEUR POLYTECH LILLE SYSTÈMES EMBARQUÉS – SYSTÈMES AUTONOMES**

## **Rapport Projet Ingénieur S9-SE5**

### **OMNIBLUE** **Navigation d'un robot mobile manipulateur pour l'impression 3D de matériaux**

**Présenté par : Haytham RABI & Ayman Moummadi**  
**Sous la supervision de :**  
**M Lakhal et M. Dherbomez,**



**Adresse :** Université de Lille, Sciences et  
technologies, Bâtiment Esprit, 59655 Villeneuve-  
d'Ascq



## Sommaire

|  |           |
|--|-----------|
| <b>Introduction .....</b>  | <b>5</b>  |
| <b>Liste des figures et tableaux.....</b>  | <b>6</b>  |
| <b>I. Présentation du projet .....</b>   | <b>9</b>  |
| 1. Contexte  |           |
| 2. Cahier de charges   |           |
| 3. Matériel à disposition.....   | 10        |
| <b>II. Gestion de projet .....</b>   | <b>11</b> |
| 1. Encadrement et suivi du projet  |           |
| 2. Organisation du travail   |           |
| 2.1. Diagramme de Gantt prévisionnel   |           |
| 2.2. Diagramme de Gantt réel.....  | 12        |
| <b>III. Travaux réalisés.....</b>  | <b>13</b> |
| <b>A- Base Mobile Holonome</b>   |           |
| 1. Modélisation cinématique, dynamique et simulation du<br>contrôle de la base mobile à roues mecanum          |           |
| 1.1. Modèle cinématique direct et inverse de la base mobile  |           |
| 1.2. Modélisation dynamique et simulation du contrôle de la<br>base mobile.....                                | 15        |
| 1.2.1. Modélisation dynamique  |           |
| 1.2.2. Simulation du contrôle d'une trajectoire de la base...  | 16        |
| 2. Programmation de l'algorithme du contrôle de la<br>base/Validation.....                                     | 21        |
| 2.1. Programmation   |           |
| 2.2. Résultats et validation.....  | 24        |
| <b>B- Bras Manipulateur .....</b>  | <b>34</b> |
| 1. Présentation du bras manipulateur Niryo NED et type de<br>programmation                                     |           |
| 1.1. Présentation du bras manipulateur Niryo NED   |           |
| 1.2. Type de programmation   |           |
| 2. Modélisation géométrique directe du bras manipulateur et<br>étude de l'enveloppe du travail.....            | 35        |
| 2.1. Modèle géométrique direct du bras manipulateur  |           |
| 2.2. Etude de l'enveloppe du travail.....  | 39        |
| 3. Implémentation du modèle géométrique sous python, tests<br>et validation des résultats avec Optitrack ..... | 40        |

|  |           |
|--|-----------|
| 3.1. Implémentation du modèle géométrique sous Python  |           |
| 3.2. Tests et validation avec Optitrack.....           | 40        |
| <b>C- Contrôle en simultané .....</b>                  | <b>43</b> |
| <b>Conclusion.....</b>                                 | <b>44</b> |
| <b>Les contraintes.....</b>                            | <b>45</b> |
| <b>Les apports de notre Projet Ingénieur.....</b>      | <b>46</b> |
| <b>Critique et futur du projet.....</b>                | <b>47</b> |
| <b>Annexes.....</b>                                    | <b>48</b> |
| <b>Référence bibliographique et webographique.....</b> | <b>50</b> |

## Introduction

L'objectif principal de notre projet de fin d'étude réside dans l'application des connaissances acquises dans notre cursus à des cas réels soit à l'échelle industrielle soit en recherche appliquée au sein de laboratoires de recherches spécialisés. Pour se faire, nous avons été amenés à réaliser un projet Ingénieur dans le domaine des Systèmes Embarqués. A ce niveau, nous étions obligés d'appréhender les réalisations avancées dans ce domaine en commençant par une étude bibliographique.

Notre projet de fin d'étude s'inscrit dans la continuité des travaux de recherches qui se déroulent au sein du laboratoire lillois CRISAL. Ce laboratoire est spécialisé dans l'automatique, l'informatique, la robotique et l'électronique. Le projet CIRMAP fait partie des recherches menées par le laboratoire groupant des travaux successifs de plusieurs intervenants travaillant sur le Robot MATRICE.

Le sujet de notre projet est l'étude séparée du contrôle de trajectoires de la base mobile et d'un bras robot type Niryo puis de proposer une commande simultanée de l'ensemble.

## Liste des Tableaux :

**Tableau 1** : Liste du matériel à disposition pour le projet omnibloc.

**Tableau 2** : Diagramme de Gantt théorique

**Tableau 3** : Diagramme de Gantt réel (tâches réalisées en binôme en bleue, tâches réalisées par Haytham en vert et par Ayman en jaune).

**Tableau 4** : Mouvements possibles de la base avec  $R=50$  cm,  $2L=42.5$  cm,  $2W=33.5$  cm

## Liste des figures :

**Figure 1.1** : Modélisation cinématique de la base mobile

**Figure 1.2** : Mouvements possibles de la base mobile du robot à roues Mecanum

**Figure 1.3** : Schéma global proposé du contrôle de trajectoire de la base mobile à roues mecanum

**Figure 1.4** : Code Matlab/Simulink pour la simulation du contrôle de la base mobile

**Figure 1.4.b** : Codes Simulink : Bloc contrôleurs trajectoires

**Figure 1.4.c** : Codes Simulink : Modèle dynamique avec contrôle vitesses roues

**Figure 1.5** : Trajectoire effectuée par la base pour se déplacer du A à O sans écrêtage de vitesse

**Figure 1.6** : Trajectoire effectuée par la base pour se déplacer du A à O avec écrêtage de vitesse

**Figure 1.7** : Evolution des coordonnées de la base mobile en fonction du temps sans écrêtage de vitesses

**Figure 1.8** : Evolution des coordonnées de la base mobile en fonction du temps avec écrêtage de vitesses

**Figure 1.9** : Evolution des vitesses de la base mobile en fonction du temps avec écrêtage de vitesses (vitesse maxi 0.5m/s)

**Figure 1.10** : Evolution des vitesses de la base mobile en fonction du temps sans écrêtage de vitesses (vitesse maxi 0.5m/s)

**Figure 1.11** : Commande en trajectoire circulaire de la base : centre (0,1) et rayon=1

**Figure 2.1** : Structure d'un message Twist

**Figure 2.2** : Fonctionnement du bas niveau du robot avec ROS2

**Figure 2.3** : Structure d'un message Odometry

**Figure 2.4** : Fonctionnement en boucle fermée, mesure reçue de l'odométrie

**Figure 2.5** : Structure d'un message RigidBodyarray

**Figure 2.6** : Schéma blocs du control de position la base mobile

**Figure 2.7** : (rqt\_graph) Architecture ROS2 de la base mobile

**Figure 2.8** : Base posée sur sol : commande en vitesse (consigne  $V_x = 0.4$  m/s ) pour un déplacement longitudinal de la base selon X.

**Figure 2.9** : Base posée sur sol : Trajectoire X suite à une commande en vitesse (consigne  $V_x = 0.4 \text{ m/s}$  ) pour un déplacement longitudinal de la base selon X.

**Figure 2.10** : Base posée sur sol : commande en vitesse (consigne  $V_x = 0.3 \text{ m/s}$  ) pour un déplacement à droite en arrière de  $45^\circ$  .

**Figure 2.11** : Base soulevée du sol : commande en vitesse (consigne  $V_x = 0.3 \text{ m/s}$  ) pour un déplacement à droite en arrière de  $45^\circ$  .

**Figure 2.12** : Base posée sur sol : trajectoires X et Y suite à une commande en vitesse (consigne  $V_x = 0.3 \text{ m/s}$  ) pour un déplacement à droite en arrière de  $45^\circ$  .

**Figure 2.13** : Base soulevée du sol : trajectoires X et Y suite à une commande en vitesse pour un déplacement à droite en arrière de  $45^\circ$  .

**Figure 2.14** : Contrôle de position : déplacement de A(1.514m,1.488m) vers O(0m,0m).

**Figure 2.15** : Contrôle de position : Evolution de x et y en fonction du temps pour un déplacement de A(1.514m,1.488m) vers O(0m,0m)

**Figure 2.16** : Contrôle de position : Evolution de  $\phi = q_3$  en fonction du temps pour un déplacement de A(1.514m,1.488m) vers O(0m,0m)

**Figure 2.17**: Contrôle de position : déplacement de A(0m,0m) vers B(1m,0m) vers C(1m,2m) vers D(0m,2m) vers O(0m,0m)

**Figure 2.18**: Contrôle de position Evolution de x,y en fonction du temps lors d'un déplacement de A(0m,0m) vers B(1m,0m) vers C(1m,2m) vers D(0m,2m) vers O(0m,0m)

**Figure 2.19**: Contrôle de position Evolution de  $\phi = q_3$  en fonction du temps lors d'un déplacement de A(0m,0m) vers B(1m,0m) vers C(1m,2m) vers D(0m,2m) vers O(0m,0m)

**Figure 2.20**: Contrôle trajectoire circulaire de rayon 1m et du centre (0m,1m)

**Figure 2.21**: Contrôle trajectoire circulaire de rayon 1m et du centre (0m,1m) : Evolution de x,y en fonction du temps

**Figure 2.22**: Contrôle trajectoire circulaire de rayon 1m et du centre (0m,1m) : Evolution de  $\phi = q_3$  en fonction du temps

**Figure 1.1.B** : Logo de l'entreprise Niryo

**Figure 1.2.B** : Robot Niryo NED

**Figure 2.1.B** : Repères de coordonnées pour les 6 articulations et le repère universel

**Figure 2.2.B** : Enveloppe du travail du robot Niryo NED

**Figure 2.3.B** : Dimensions entre chaque articulation

**Figure 3.1.B** : Exemple de messages Modbus implémenté sous python

**Figure 3.2.B** : Extrait des adresses Modbus ainsi que leur description

**Figure 3.3.B** : Trajectoire désirée

**Figure 3.4.B** : Enveloppe du travail suivant  $q_3$

**Figure 3.5.B** : Trajectoire désirée

**Figure 3.6.B** : Enveloppe du travail suivant  $q_2$

**Figure 3.7.B** : Trajectoire désirée

**Figure 3.4.B** : Enveloppe du travail suivant  $q_1$

**Figure 4.C** : Organigramme global d'une éventuelle commande simultanée de la base et du bras robot



## **I. Présentation du projet**

### **1. Contexte**

De nos jours, la fabrication additive ou impression 3D dans la construction devient de plus en plus accessible en termes de : coûts, variété de matériaux imprimables et techniques d'impression technologiques. La taille des objets imprimés est donc limitée par la taille du bras manipulateur de la machine, ce qui rend l'impression de structures à grande échelle peu pratique et coûteuse.

La plateforme PRETIL de CRISAL a mené une activité autour de l'impression 3D pour le bâtiment et la construction. Elle s'est concrétisée par le projet MATRICE qui a permis de concevoir et fabriquer un robot manipulateur sur base mobile pour l'impression 3D.

Le bras manipulateur du KUKA possède un champ d'action autour de sa base, limitée par ses dimensions et ses degrés de liberté. Le couplage du bras KUKA et du chariot a pour but d'élargir l'envergure du bras et par conséquent sa zone de travail.

A terme dans le projet CIRMAP, la méthode de navigation est normalement couplée avec une commande d'écoulement du dépôt de matière (béton, céramique...) pour permettre des tests d'impressions au sol.

Dans le cadre de notre Projet ingénieur, l'objectif sera de reproduire un tracé (mesurable grâce au système de capteurs Optitrack). Celui-ci devra être réitérable et donc le robot devra être capable de faire plusieurs fois le même tracé.

### **2. Cahier des charges**

L'objectif principal de notre thème du PFE qui nous a été proposé dans un premier temps est de mettre au point une application de navigation autonome du robot MATRICE pour imprimer en 3D des matériaux avec deux principales fonctionnalités :

- Gestion du bras manipulateur Kuka pour le positionnement précis de la buse d'extrusion ;
- Gestion de la base mobile holonome pour le déplacement du robot afin d'atteindre des zones non accessibles dans le rayon d'action du bras.

Mais après la rédaction de cahier de charges et de spécifications, le sujet a été modifié à cause d'une panne de la baie du robot. Nos tuteurs nous ont proposé le 27/09/2022 de travailler sur un prototype de Matrice dans le même contexte, il s'agit d'une nouvelle base mobile plus petite avec un bras manipulateur Niryo. Il a fallu donc commencer la partie programmation à zéro, contrairement au premier thème où il y'avait déjà un existant sur ROS, notamment l'instrumentation et les algorithmes de contrôles étaient déjà implémentés et validés.

Ainsi le cahier des charges devient le suivant :

- 1- Modélisation cinématique de la base à roues mecanum pour comprendre les mouvements possibles et implémenter ensuite le contrôle en position de cette base.
- 2- Modélisation géométrique du bras robot Niryo et implémenter ensuite le contrôle en position

- 3- Coupler les commandes de la base mobile et du bras du robot pour exécuter une tâche en simultané.

### 3. Matériel et outils à disposition

Le matériel à disposition pour réaliser le projet est listé dans le tableau le suivant :

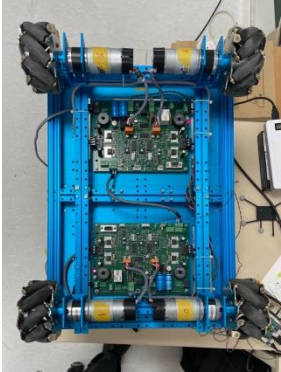
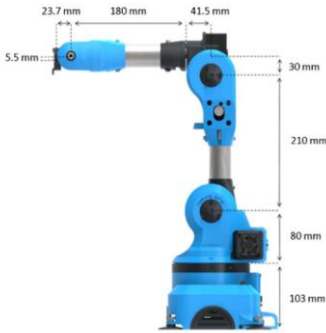

|  |   |
|--|---|
| <p>- La base mobile à roues mecanum pilotable par bus CAN, équipée d'une Raspberry pi (Ubuntu server 20), un microcontrôleur DsPic33EP256MC502 avec des encodeurs pour le contrôle de vitesses des roues motrices de la base.</p> <p>Dimensions :</p> <p>Longueur : 42.5 cm</p> <p>Largeur : 33.5 cm</p> <p>Rayon Roue : 5 cm</p>  |    |
| <p>- Le bras manipulateur Niyro Ned pilotable par l'interface TCP/IP, Modbus :</p> <p>Dimensions : (voir photo)</p> <p>Portée : 44 cm</p> <p>Poids : 3.3 Kg</p>  |   |
| <p>- Capteur de mouvement Optitrack</p> <p>Précision : <math>\pm 0,2\text{mm}</math> (calibrage excellent)</p> <p>Portée : <math>\sim 30\text{m}</math> (selon fiche technique)</p> <p>Avantages : - Haute précision - Traitement fluide des données pour un temps de réponse rapide en boucle fermée.</p> <p>Inconvénients : Très encombrant car nécessite une structure pour supporter les caméras – limite la zone de travail</p> |  |

Tableau 1 : Liste du matériel à disposition pour le projet omniblue.

Pour la partie programmation, nous avons utilisé ROS2 (Robot operating system version 2), python, C++ ainsi que les commandes Linux.

## II. Gestion de projet

### 1. Encadrement et suivi du projet

Pendant ce projet, nous étions encadrés par nos tuteurs **M. Lakhal** et **M. Dherbomez**, on réalisait des points réguliers avec eux (1 fois chaque semaine de projet) dans la mesure du possible pour pouvoir nous débloquent sur nos difficultés et suivre l'avancée de notre projet. Néanmoins, une troisième personne nous a aussi aidés pour avancer, c'est l'électronicien de l'école Centrale **M. Sanz Lopez**.

Nous avons mis à disposition de nos tuteurs un document où on rédige un petit CR journalier (extrait en Annexe) et un dépôt git (Annexe) pour tenir au courant nos tuteurs de nos avancées et nos difficultés éventuellement.

### 2. Organisation du travail

#### 2.1. Diagramme de Gantt prévisionnel

Pour pouvoir nous organiser au mieux, nous avons fait un GANTT qui met des “deadlines” pour chaque partie du projet. Ci-dessous le GANTT prévisionnel que nous avons réalisé au début de ce semestre.

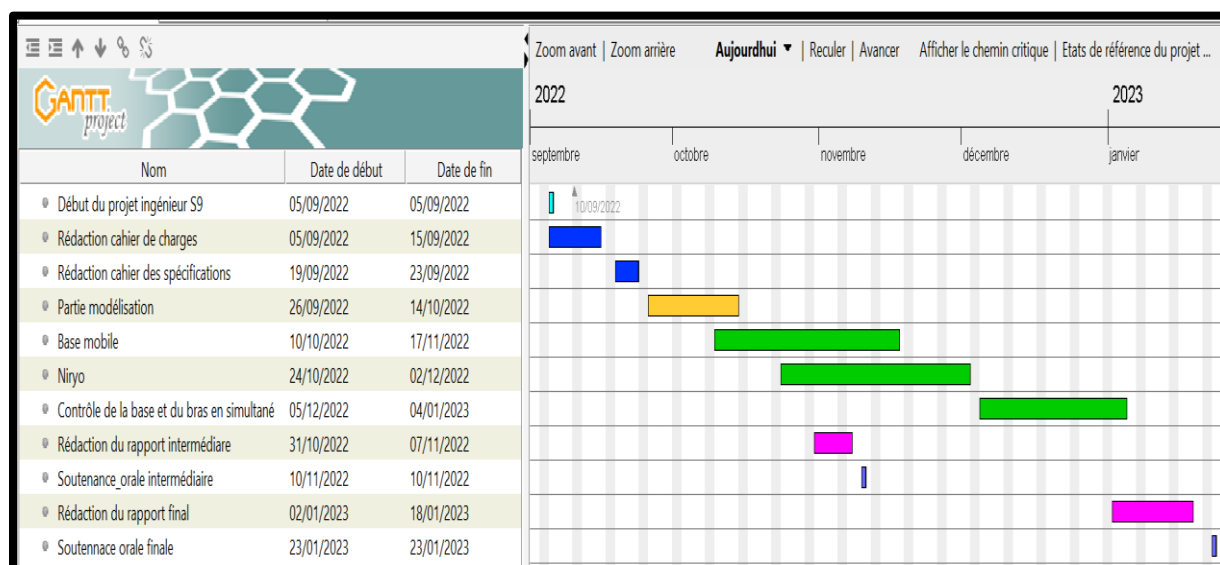


Tableau 2 : Diagramme de Gantt théorique

## 2.2. Diagramme de Gantt réel

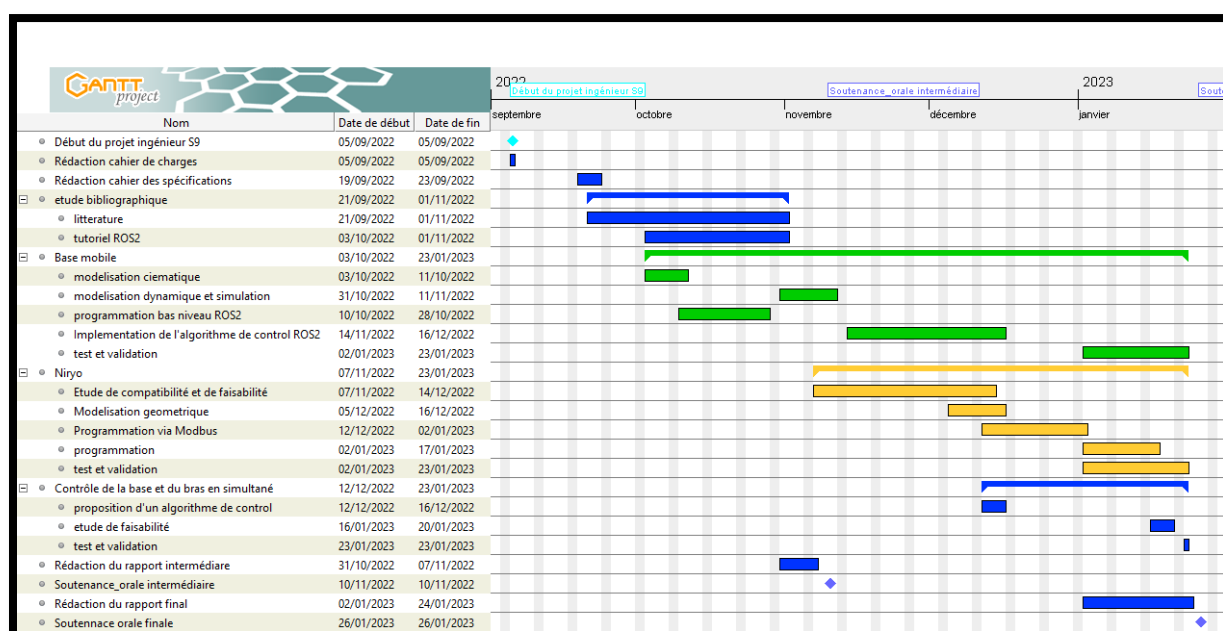


Tableau 3 : Diagramme de Gantt réel (tâches réalisées en binôme en bleue, tâches réalisées par Haytham en vert et par Ayman en jaune).

Nous constatons aisément que notre Gantt initial ne correspond quasiment pas au Gantt final car nous ne pouvions pas à l'époque anticiper précisément en quoi allaient se découper les grandes tâches prévues à la base et leur difficulté. Nous avons dû également faire face aux imprévus et aux différents problèmes techniques (panne base mobile, panne bras Niryo, problème réseau et wifi...).

Depuis le début du projet, la répartition du travail s'est faite naturellement. **Haytham** s'occupait de la partie base mobile et **Ayman** du bras Niryo, et donc notre but final c'est de rassembler les deux parties pour travailler en binôme sur le contrôle en simultané.

Dans la partie ci-dessous, nous montrons les résultats qu'on a pu avoir lors de ce projet ainsi que l'étude de la base mobile et du bras manipulateur.

### III. Travaux réalisés

#### A- Base Mobile Holonome

#### 1. Modélisation cinématique, dynamique et simulation du contrôle de la base mobile à roues mecanum du robot

##### 1.1. Modèle cinématique direct et inverse de la base mobile [4], [6]

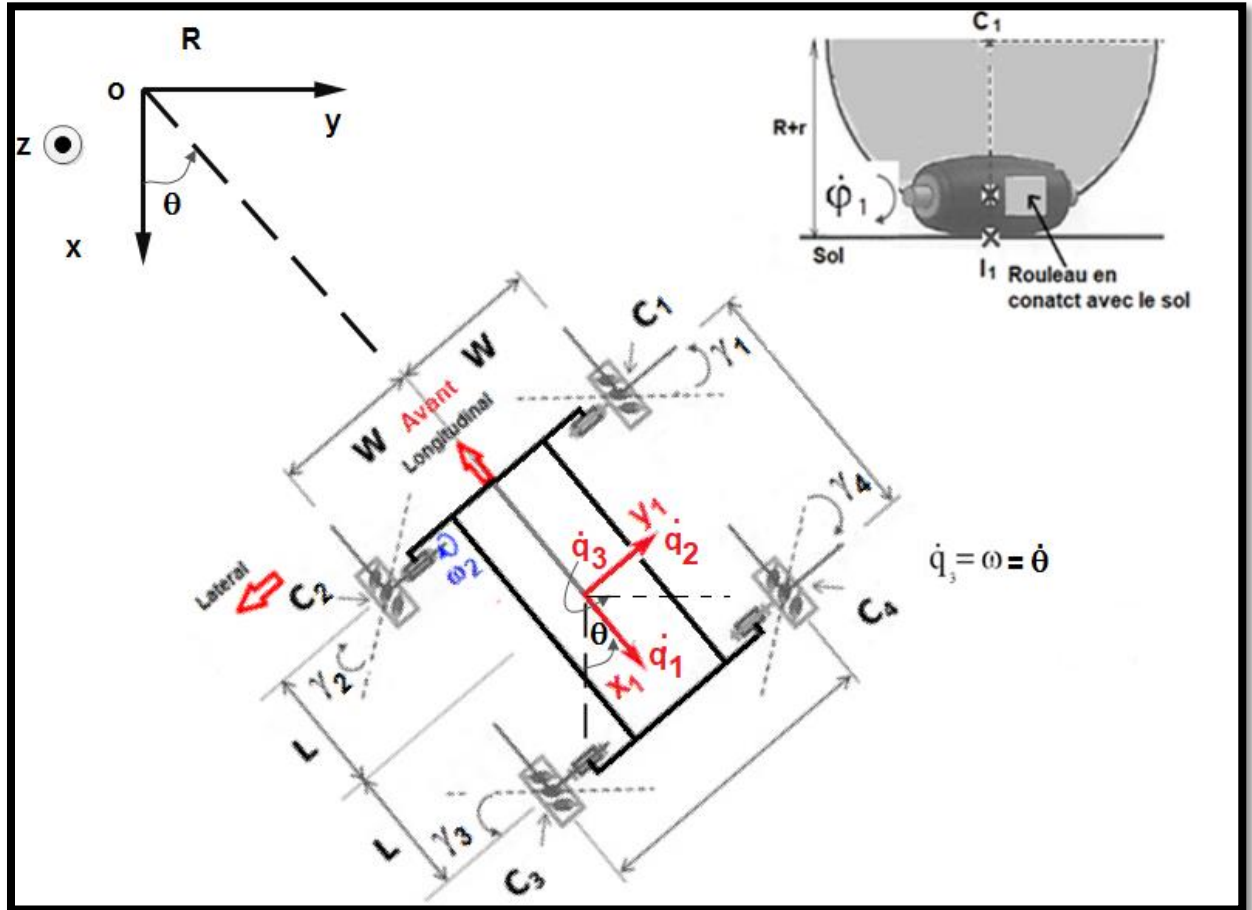


Figure 1.1 : Modélisation cinématique de la base mobile

On pose :

$$[v] = \begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \\ \dot{q}_3 \\ 0 \end{bmatrix} ; [\omega] = \begin{bmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \\ \omega_4 \end{bmatrix}$$

On montre que le modèle cinématique direct :

$$[v] = A[\omega]$$

$$\text{Avec } A = \begin{bmatrix} \frac{R}{4} & \frac{R}{4} & \frac{R}{4} & \frac{R}{4} \\ \frac{R}{4} & -\frac{R}{4} & +\frac{R}{4} & -\frac{R}{4} \\ -\frac{R}{4(L+W)} & \frac{R}{4(L+W)} & \frac{R}{4(L+W)} & -\frac{R}{4(L+W)} \\ 1 & +1 & -1 & -1 \end{bmatrix}$$

Le déterminant de A vaut :  $\frac{R^3}{4(L+W)} \neq 0$  donc le système est bien contrôlable. L'équation de commande des vitesses est alors :

$$[\omega] = A^{-1}[v]$$

$$\text{Avec : } A^{-1} = \begin{bmatrix} \frac{1}{R} & \frac{1}{R} & -\frac{L+W}{R} & \frac{1}{4} \\ \frac{1}{R} & -\frac{1}{R} & \frac{L+W}{R} & \frac{1}{4} \\ \frac{1}{R} & \frac{1}{R} & +\frac{L+W}{R} & -\frac{1}{4} \\ \frac{1}{R} & -\frac{1}{R} & -\frac{L+W}{R} & -\frac{1}{4} \end{bmatrix}$$

| Vitesse                          | $\dot{q}_1$<br>(m/s) | $\dot{q}_2$<br>(m/s) | $\dot{q}_3$<br>(m/s) | $\omega_1$<br>(rd/s) | $\omega_2$<br>(rd/s) | $\omega_3$<br>(rd/s) | $\omega_4$<br>(rd/s) |
|----------------------------------|----------------------|----------------------|----------------------|----------------------|----------------------|----------------------|----------------------|
| <b>Déplacement</b>               |                      |                      |                      |                      |                      |                      |                      |
| <b>Arrière suivant x</b>         | 0.2                  | 0                    | 0                    | 4                    | 4                    | 4                    | 4                    |
| <b>Avant suivant x</b>           | -0.2                 | 0                    | 0                    | -4                   | -4                   | -4                   | -4                   |
| <b>A droite suivant y</b>        | 0                    | 0.2                  | 0                    | 4                    | -4                   | 4                    | -4                   |
| <b>A gauche suivant y</b>        | 0                    | -0.2                 | 0                    | -4                   | 4                    | -4                   | 4                    |
| <b>A droite à 45° en arrière</b> | 0.2                  | 0.2                  | 0                    | 8                    | 0                    | 8                    | 0                    |
| <b>A droite à 45° en avant</b>   | -0.2                 | 0.2                  | 0                    | 0                    | -8                   | 0                    | -8                   |
| <b>A gauche à 45° en arrière</b> | 0.2                  | -0.2                 | 0                    | 0                    | 8                    | 0                    | 8                    |
| <b>A gauche à 45° en avant</b>   | -0.2                 | -0.2                 | 0                    | -8                   | 0                    | -8                   | 0                    |
| <b>Rotation pure</b>             | 0                    | 0                    | 1                    | -7.6                 | +7.6                 | 7.6                  | -7.6                 |

Tableau 4 : Mouvements possibles de la base avec  $R=50$  cm,  $2L=42.5$  cm,  $2W=33.5$  cm

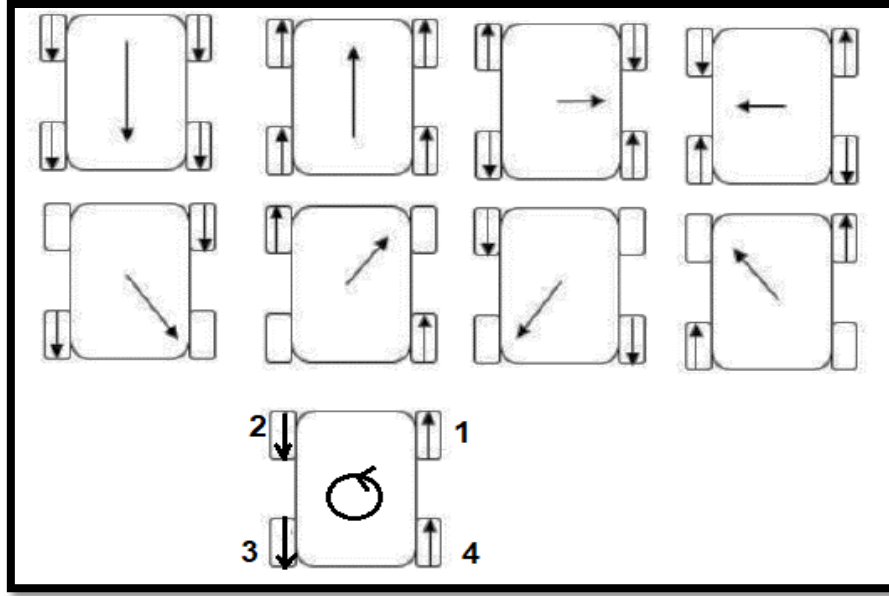


Figure 1.2 : Mouvements possibles de la base mobile du robot à roues Mecanum

## 1.2. Modélisation dynamique et simulation du contrôle de la base mobile [2], [3], [5], [7]

### 1.2.1. Modélisation dynamique

L'équation de LAGRANGE avec multiplicateurs est utilisée ici pour introduire un modèle dynamique pour la base à roues Mecanum. L'équation de LAGRANGE avec multiplicateurs s'écrit :

$$\frac{d}{dt} \left( \frac{\partial T}{\partial \dot{q}_j} \right) - \frac{\partial T}{\partial q_j} = Q_j$$

Le formalisme lagrangien a été utilisé afin d'obtenir les équations dynamiques du mouvement du WMR. Les équations de Lagrange du deuxième type pour l'objet holonome décrit peuvent être notées dans ce qui suit

$$\begin{aligned} \frac{d}{dt} \left( \frac{\partial L}{\partial \dot{q}_j} \right) - \frac{\partial L}{\partial q_j} &= Q_j \\ \begin{cases} L = T - P \\ P = \text{cte} \end{cases} &\Rightarrow \frac{\partial L}{\partial \dot{q}_j} = \frac{\partial T}{\partial \dot{q}_j} \end{aligned}$$

T= Energie cinétique de la base +les roues ; P = Energie potentielle de la base + les roues.

$Q_j$  = couple moteur roue j – couple frottement roue j ,  $q_j$  : coordonnée généralisée j  $j=1,2,\dots,s$ (nombre des degrés du liberté du système=4 dans notre cas).

Si on néglige les frottements, on obtient le système d'équations différentielle suivant :

$$\mathbf{M} \begin{bmatrix} \dot{\omega}_1 \\ \dot{\omega}_2 \\ \dot{\omega}_3 \\ \dot{\omega}_4 \end{bmatrix} = \begin{bmatrix} \tau_1 \\ \tau_2 \\ \tau_3 \\ \tau_4 \end{bmatrix} \quad \mathbf{M} = \begin{bmatrix} A+B+C & -B & B & A-B \\ -B & A+B+C & A-B & B \\ B & A-B & A+B+C & -B \\ A-B & B & -B & A+B+C \end{bmatrix}$$

A, B et C s'expriment en fonction des moments d'inertie de la base, des roues et du rayon R d'une roue et r du rouleau.

$$A = \frac{m_{br}(R+r)^2}{8} ; B = \frac{I_{br}(R+r)^2}{16(W+L)} \quad C = I_k$$

$$m_{br} = m_b + 4m_k \quad I_{br} = I_{bz} + 4I_{kz}$$

$$I_{bz} = \frac{1}{12}(W^2 + L^2) ; I_k \approx \frac{m_k}{2}(R^2)$$

Dans notre cas nous avons adoptés les valeurs approximatives suivantes :

$$m_b = 6.5\text{kg} ; m_k = 0.325\text{kg} ; I_{bz} = 0.1586\text{kg/m}^2 ;$$

$$R = 5\text{ cm} ; r = R/10 = 0.5\text{cm} ;$$

$$I_k = 4.0625 \cdot 10^{-4} \text{ kg/m}^2 ;$$

### 1.2.2. Simulation du contrôle d'une trajectoire de la base

Pour exprimer les vitesses et la position de la base dans le repère fixe  $R_0$ , on utilise la matrice du passage :

$$\mathbf{P} = \begin{bmatrix} q_1 \\ q_2 \\ q_3 \\ 1 \end{bmatrix}_{R_0} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} q_1 \\ q_2 \\ q_3 \\ 1 \end{bmatrix}_{R_1}$$

$$\mathbf{P}^{-1} = \begin{bmatrix} \cos(\theta) & \sin(\theta) & 0 \\ -\sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Nous proposons le contrôle de trajectoire de la figure 1.3 et le code Matlab/Simulink nécessaire pour effectuer toutes les simulations est donné sur la figure 1.4. Pour la synthèse des paramètres des 3 régulateurs (en cascade) de la trajectoire, nous avons utilisé l'application '**Tuner App**' disponible dans le bloc PID de Simulink [9].

A titre de démonstration nous déplacerons la base d'un état initial repère fixe (25m,20m) pour rejoindre l'origine du même repère (0m,0m) (Figures 1.5 et 1.6). L'évolution de la position en fonction du temps est donnée sur les figures 1.7 et 1.8. On constate que la base rejoint bien la position O dans des temps de 55 s avec écrêtage de vitesse contre 12s sans écrêtage de vitesses. Les vitesses sont représentées aussi sur les courbes 1.9 et 1.10.

Un dernier test est la planification d'une trajectoire circulaire de centre (0m,1m) et rayon 1m (Figure 1.11). Notons que dans ce cas, il faut ajouter une action intégrale dans les contrôleurs



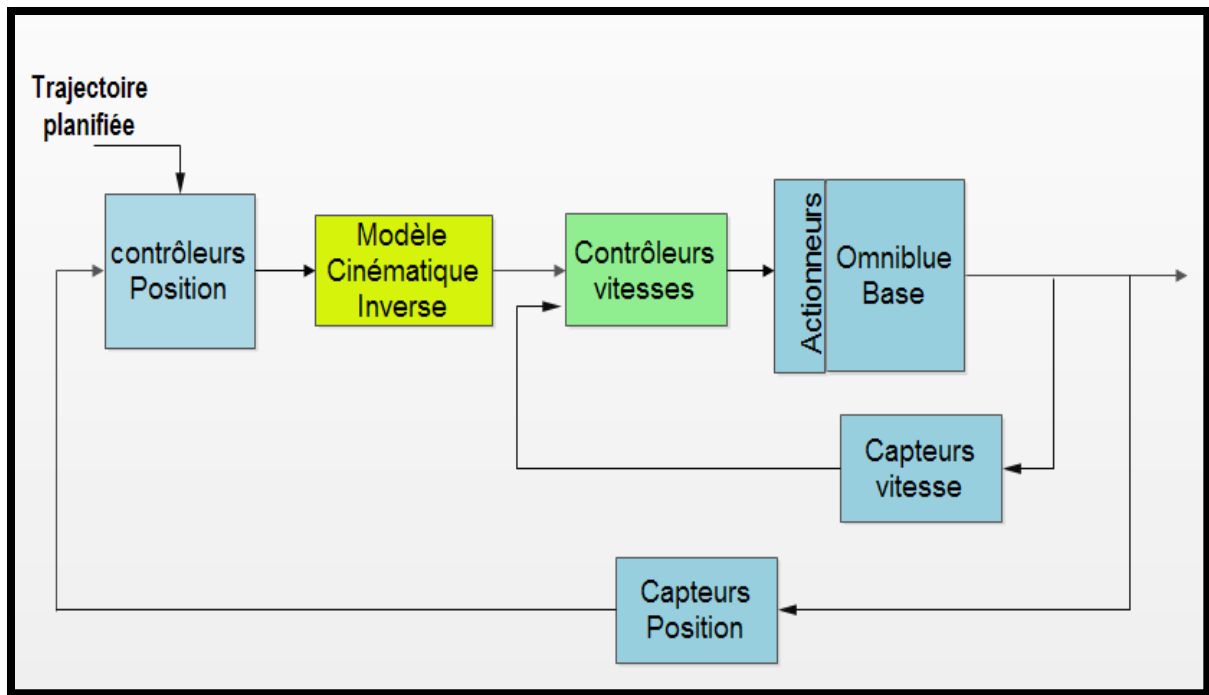


Figure 1.3 : Schéma global proposé du contrôle de trajectoire de la base mobile à roues mecanum

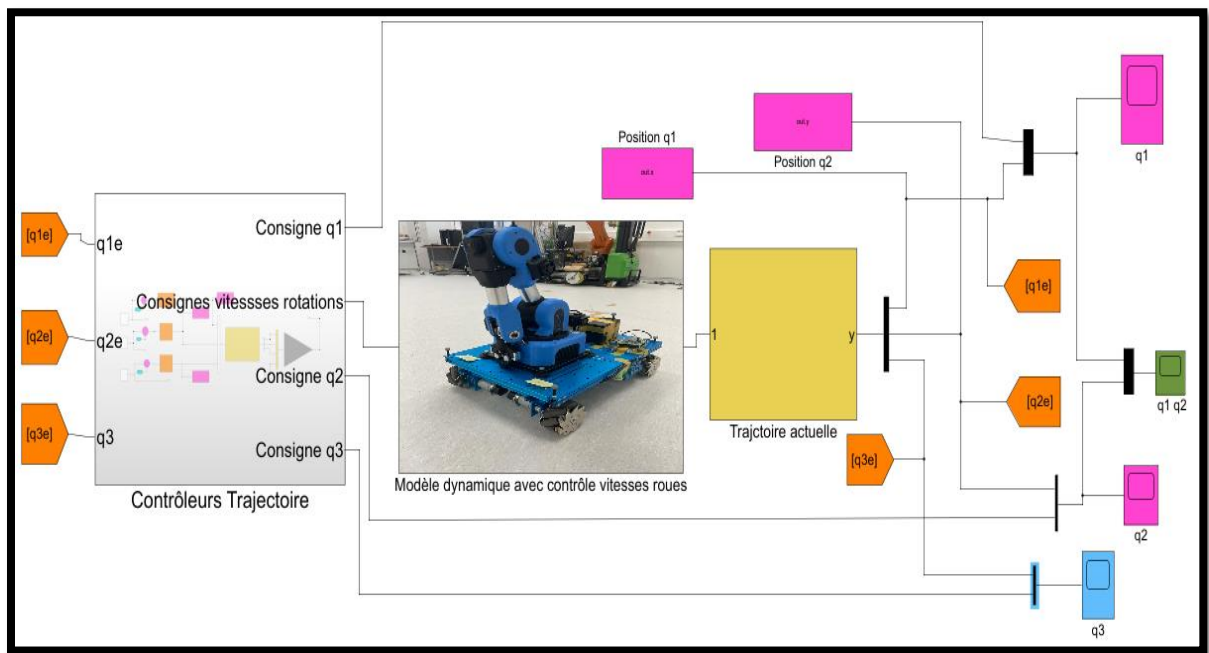


Figure 1.4 : Code Matlab/Simulink pour la simulation du contrôle de la base mobile

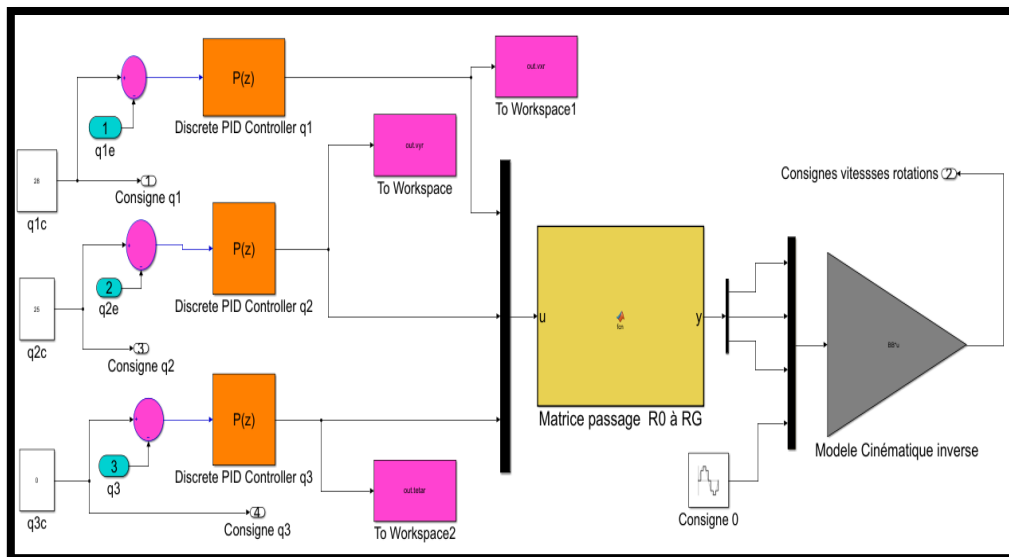


Figure 1.4.b : Codes Simulink : Bloc contrôleurs trajectoires

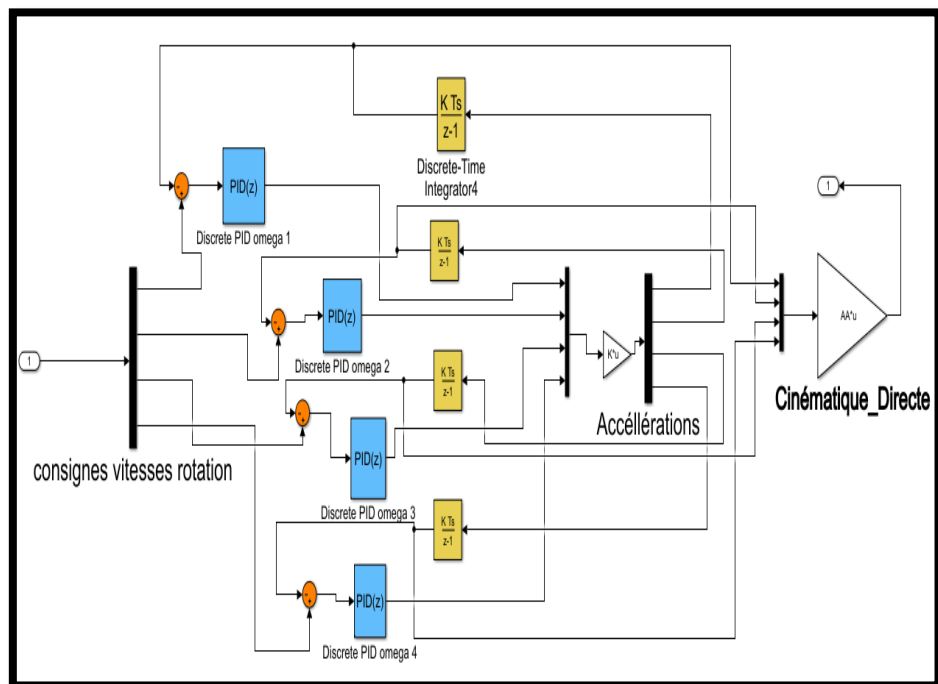


Figure 1.4.c : Codes Simulink : Modèle dynamique avec contrôle vitesses roues

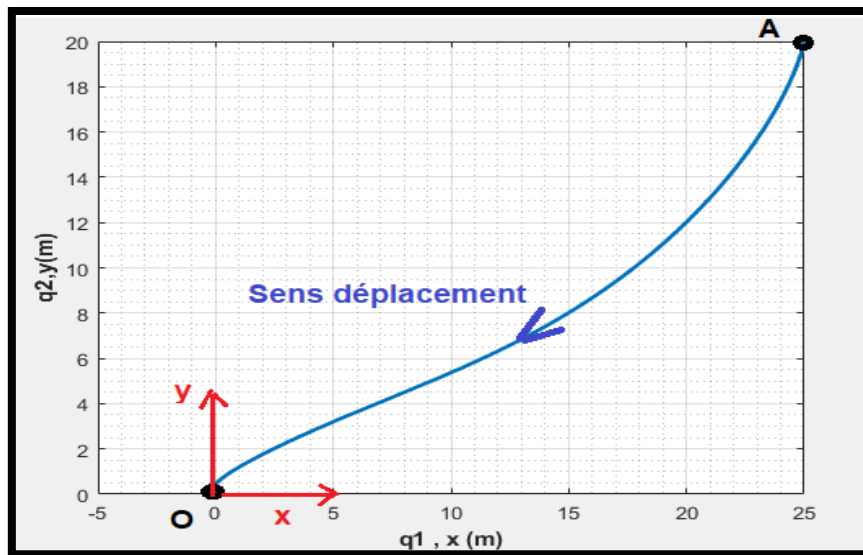


Figure 1.5 : Trajectoire effectuée par la base pour se déplacer du A à O sans écrêtage de vitesse

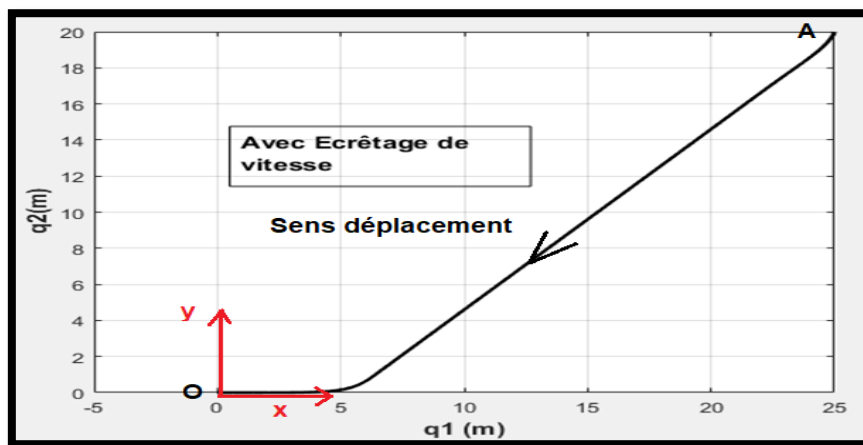


Figure 1.6 : Trajectoire effectuée par la base pour se déplacer du A à O avec écrêtage de vitesse

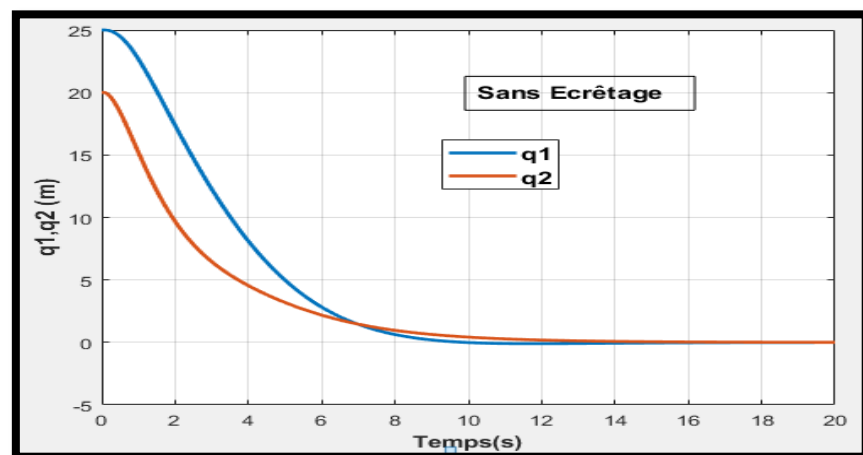


Figure 1.7 : Evolution des coordonnées de la base mobile en fonction du temps sans écrêtage de vitesses

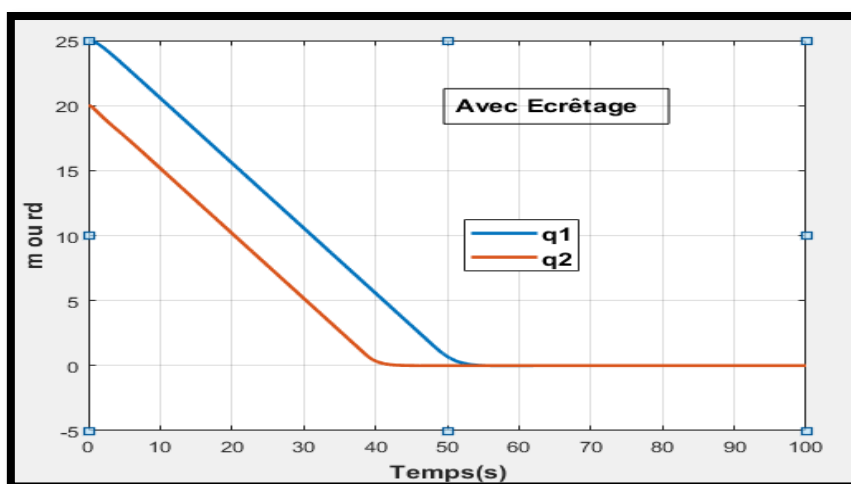


Figure 1.8 : Evolution des coordonnées de la base mobile en fonction du temps avec écrêtage de vitesses

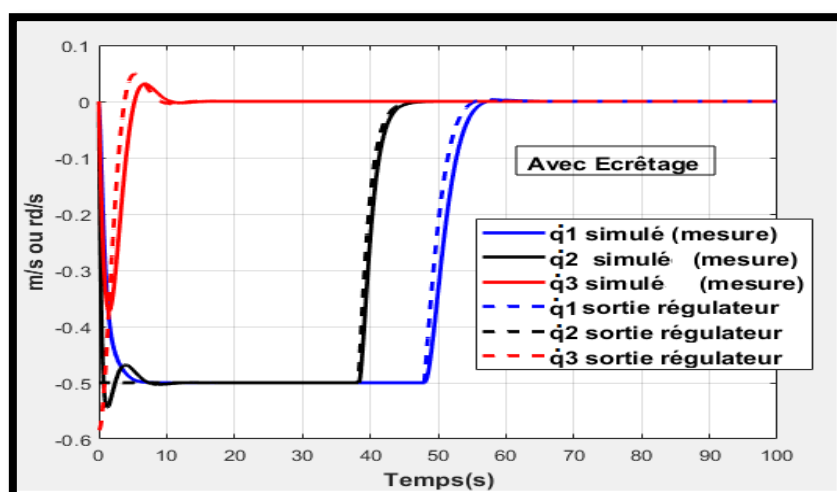


Figure 1.9 : Evolution des vitesses de la base mobile en fonction du temps avec écrêtage de vitesses (vitesse maxi 0.5m/s).

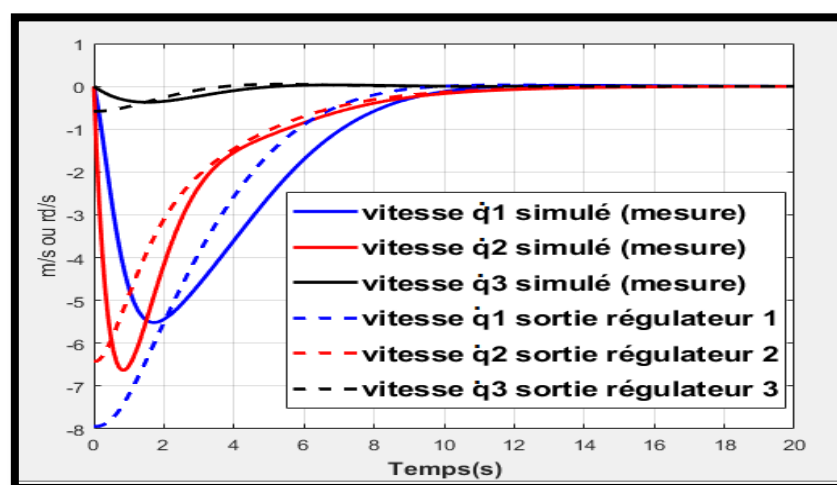


Figure 1.10 : Evolution des vitesses de la base mobile en fonction du temps sans écrêtage de vitesses (vitesse maxi 0.5m/s).

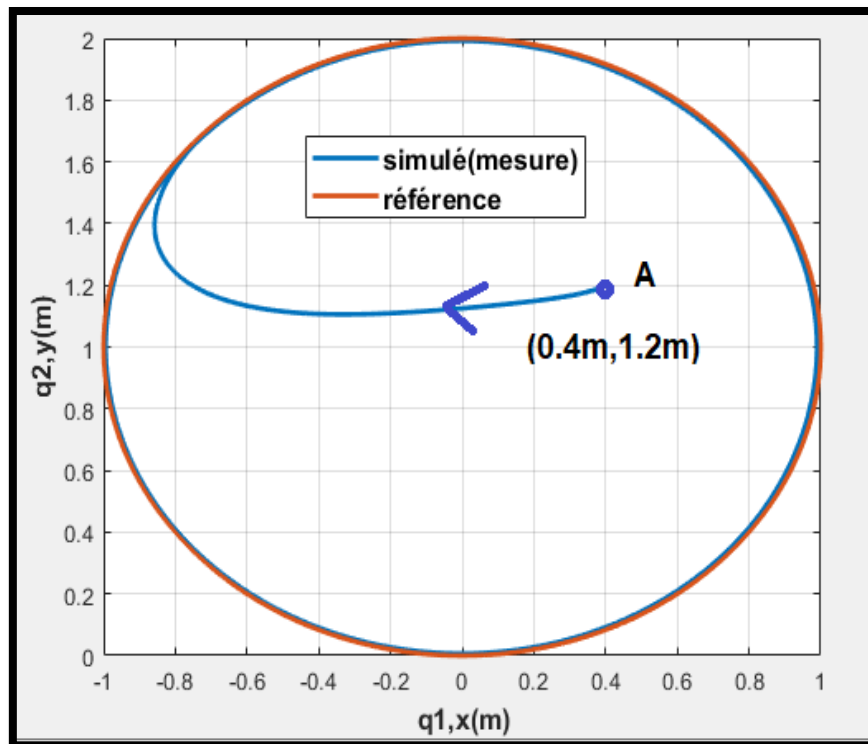


Figure 1.11 : Commande en trajectoire circulaire de la base : centre  $(0,1)$  et rayon=1

## 2. Programmation de l'algorithme du contrôle de la base/Validation

### 2.1. Programmation

Pour la partie développement du projet, nous avons utilisé **ROS2 (Robot operating system v2)** pour programmer notre robot. Un README.md (ANNEXE) pour la réutilisation du projet ainsi que le code sont déposés sur notre Gitlab de l'université (<https://gitlab.univ-lille.fr/pretil/cirmap/omniblue>).

Afin d'implémenter un algorithme de contrôle sur la base, il a fallu tout d'abord développer la partie bas niveau pour rendre la base compatible avec ROS2. Ce premier nœud subscriber « *base\_mobile* » reçoit des consignes vitesses soit par appui sur les touches du clavier par le nœud « *teleop\_twist\_keyboard* », soit par le nœud « *controller* » (figure 2.2), les messages communiqués entre les nœuds sont appelés les topics, ils contiennent des vitesses sous forme de message Twist (voir figure 2.1).

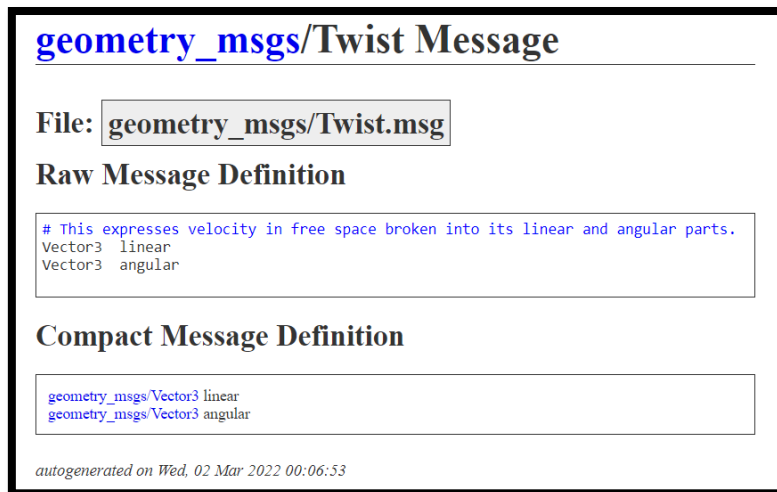


Figure 2.1: Structure d'un message Twist dans ROS  
(les noeuds en bleu / les topics en vert)

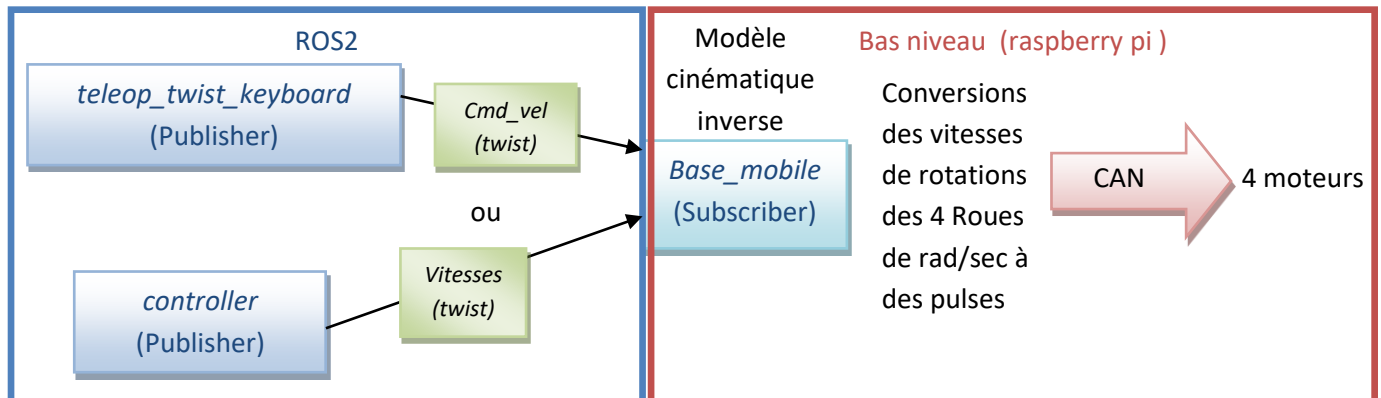


Figure 2.2 : Fonctionnement du bas niveau du robot avec ROS2

Une fois la partie programmation bas niveau est terminée, il faut récupérer une mesure pour fermer la boucle de contrôle. Pour cela, nous avons développé un nœud « *odometry* » qui permet d'estimer la trajectoire parcourue de la base en temps réel, à partir des vitesses mesurées par les encodeurs (figure 2.4). Les messages « *odom* » publiés sont de type Odometry (voir figure 2.3).



Figure 2.3: Structure d'un message Odometry dans ROS

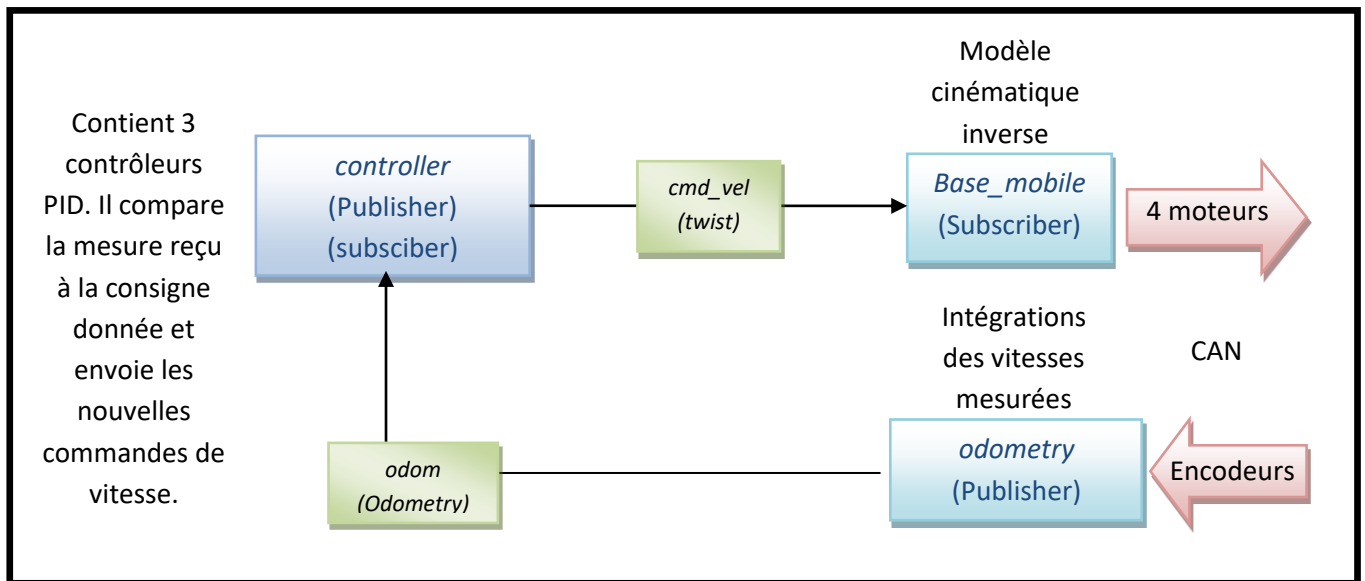


Figure 2.4: Fonctionnement en boucle fermée, mesure reçue de l'odométrie

Comme l'odométrie ne suffit pas pour contrôler la base, il a fallu utiliser l'Optitrack pour tracker la position du robot. On a utilisé un projet Optitrack/ROS2 trouvé sur Git [10] qui comporte un nœud qui envoie un topic « *mocap\_rigid\_body* » de type *Rigid\_body\_array* (figure 2.5) avec la position en temps réel d'un corps solide défini à partir des marqueurs, dans ce cas c'est la base mobile.

```

hrabi@hrabi-HP:~/base_control_ros2$ ros2 interface show mocap_optitrack_interfaces
/msg/RigidBodyArray
#Array of rigid bodies
RigidBody[] rigid_bodies
  int64 id
  bool valid
  float64 mean_error
  geometry_msgs/PoseStamped pose_stamped
    std_msgs/Header header
      builtin_interfaces/Time stamp
        int32 sec
        uint32 nanosec
      string frame_id
    Pose pose
      Point position
        float64 x
        float64 y
        float64 z
      Quaternion orientation
        float64 x 0
        float64 y 0
        float64 z 0
        float64 w 1

```

Figure 2.5 : Exemple d'un message Rigidbodyarray

Un réseau local a été configuré (voir README.md) pour établir la **communication** entre le bas niveau (base\_mobile et odometry) implémenté dans la Raspberry Pi et le haut niveau (algorithme de control et mesure optitrack) sur un pc Linux.

Notations : notons les positions par  $X$ ,  $Y$  et  $\Phi$  et les vitesses par  $V_x$ ,  $V_y$  et  $V_\phi$  :



The diagram illustrates the ROS control architecture for a mobile robot. It shows the flow of data from sensors to the base mobile robot through various control nodes.

```
graph LR; /odometry([/odometry]) --> /tf[/tf/]; /odometry --> /odom[/odom/]; /teleop_twist_keyboard([/teleop_twist_keyboard/]) --> /cmd_vel[/cmd_vel/]; /odom --> /Controller([/Controller/]); /odom --> /world_to_base([/world_to_base/]); /mocap_rigid_bodies[/mocap_rigid_bodies/] --> /Controller; /mocap_rigid_bodies --> /world_to_base; /world_to_base --> /baseframe_rigid_bodies[/baseframe_rigid_bodies/]; /baseframe_rigid_bodies --> /inverse_kinematics([/inverse_kinematics/]); /Controller --> /cmd_vel_control[/cmd_vel_control/]; /cmd_vel --> /base_mobile([/base_mobile/]); /cmd_vel_control --> /base_mobile;
```

**Topic publié par optitrack**

Figure 2.7 : (*rqt\_graph*) Architecture ROS2 finale de la base mobile

Notons d’abord que c’est le deuxième prototype de base mobile que nous avons reçu à cause d’une panne du premier. Malheureusement, à cause de la nouvelle mise à jour, nous avons refait le modèle cinématique, pour pouvoir l’adapter à cette nouvelle base (Figure 1.1).

24



Le tracé de la position du mouvement longitudinal estimée par odométrie montre bien que le robot suit une trajectoire longitudinale de vitesse 0.4 m/s égale à la consigne imposée. Donc on peut conclure que le mouvement s'effectue correctement sans glissement (Figure 2.8 et 2.9).

Pour la trajectoire à 45°, nous avons tracé pour des consignes de vitesse de  $V_x=V_y= 0.3$  m/s les positions estimées ( $q_1= X$ ,  $q_2=Y$ ) par odométrie pour le robot soulevé du sol (Figures 2.11, 2.13) et posé sur le sol (Figures 2.10, 2.12). Sur cette dernière figure 2.12, on constate que les trajectoires mesurées sont légèrement déviées par rapport aux trajectoires consignes (Figure.213). Ceci est dû à l'existence d'un glissement et dans ce cas on peut conclure que l'odométrie basé sur les encodeurs est moins efficace et donc il faut utiliser des capteurs plus performants comme par exemple l'Optitrack.

En ajoutant du poids sur la base avec une répartition uniforme de la charge, nous avons constaté une diminution nette des glissements, une idée qui nous a été suggéré par notre tuteur.

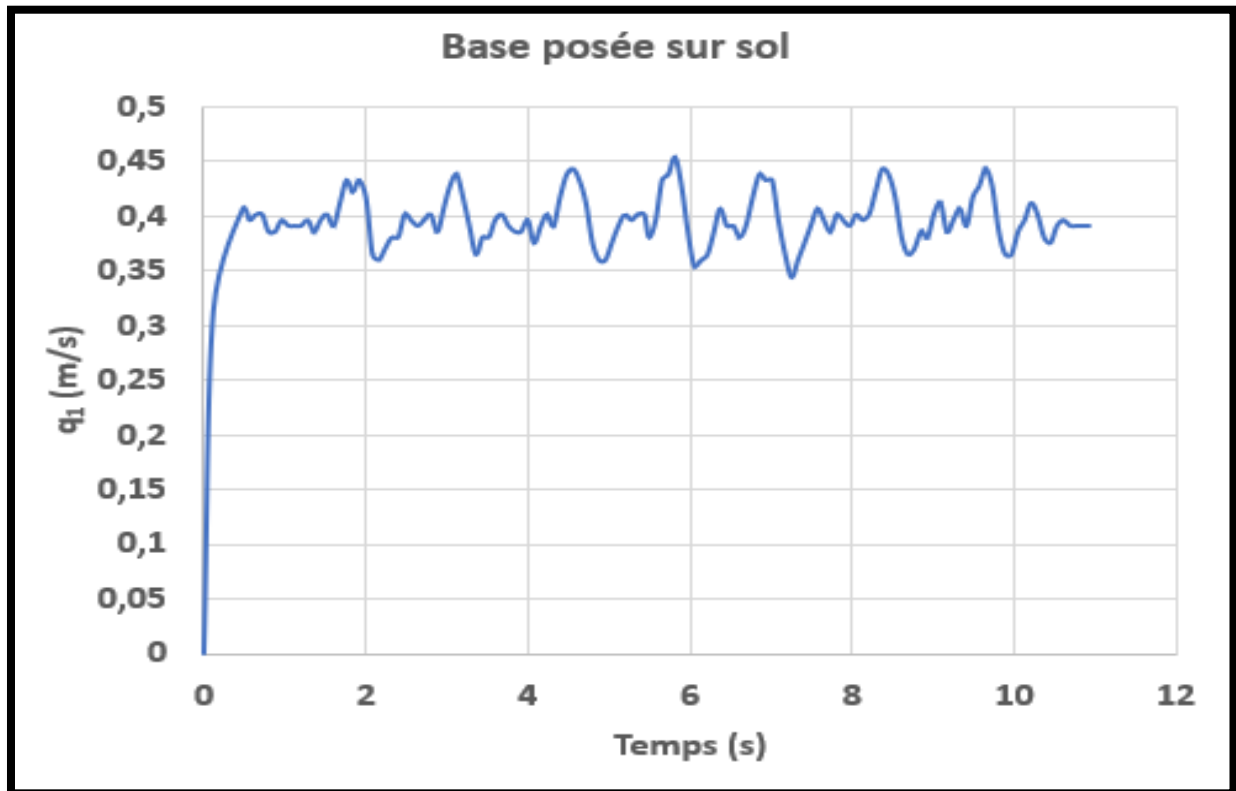


Figure 2.8 : Base posée sur sol : commande en vitesse (consigne  $V_x=0.4\text{m/s}$  ) pour un déplacement longitudinal de la base selon X.

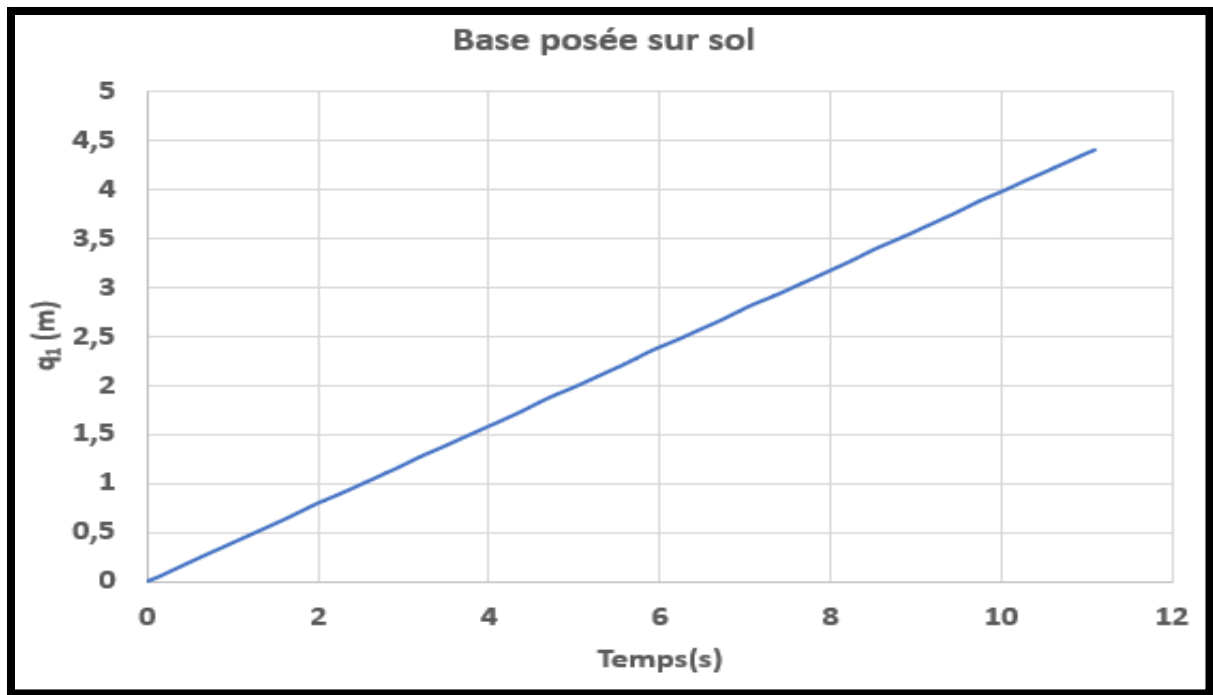


Figure 2.9 : Base posée sur sol : Trajectoire X suite à une commande en vitesse (consigne  $V_x=0.4\text{m/s}$  ) pour un déplacement longitudinal de la base selon X.

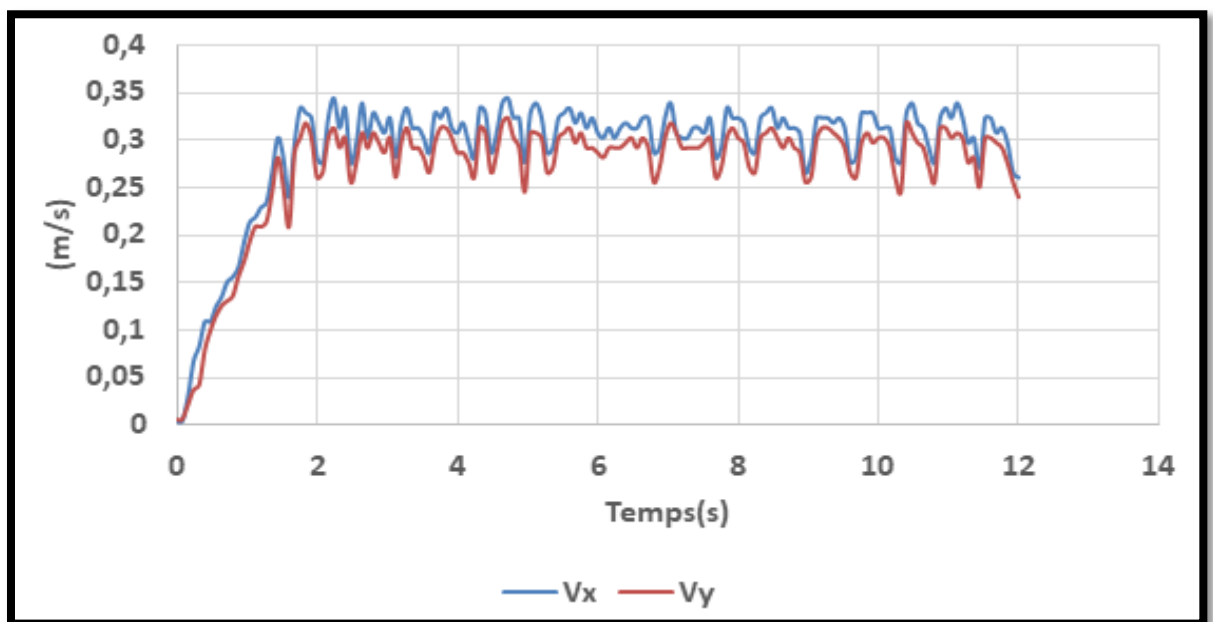


Figure 2.10 : Base posée sur sol : commande en vitesse (consigne  $V_x=0.3\text{m/s}$  ) pour un déplacement à droite en arrière de  $45^\circ$  .

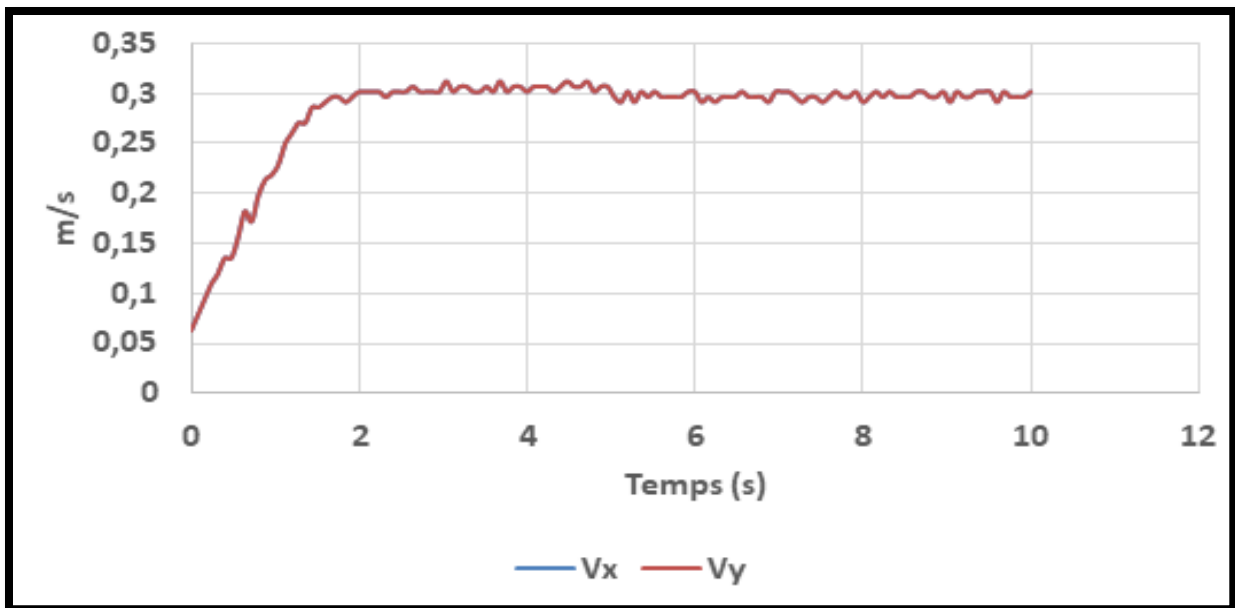


Figure 2.11 : Base soulevée du sol : commande en vitesse (consigne  $V_x = 0.3 \text{ m/s}$ ) pour un déplacement à droite en arrière de  $45^\circ$ .

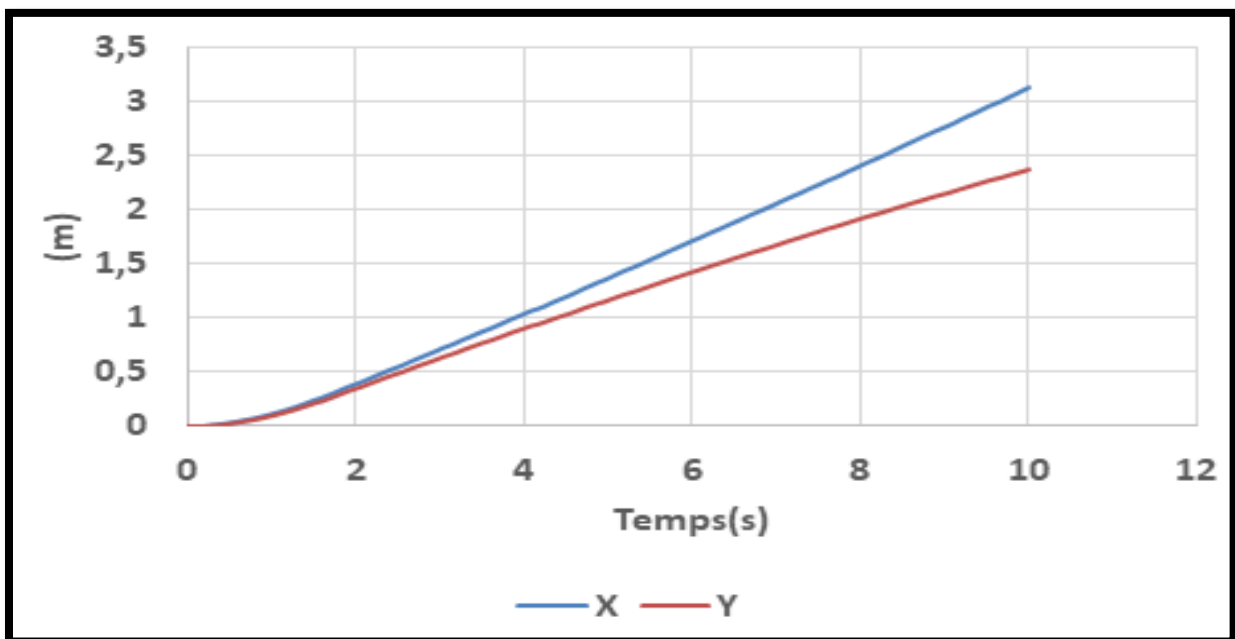


Figure 2.12 : Base posée sur sol : trajectoires  $X$  et  $Y$  suite à une commande en vitesse (consigne  $V_x = 0.3 \text{ m/s}$ ) pour un déplacement à droite en arrière de  $45^\circ$ .

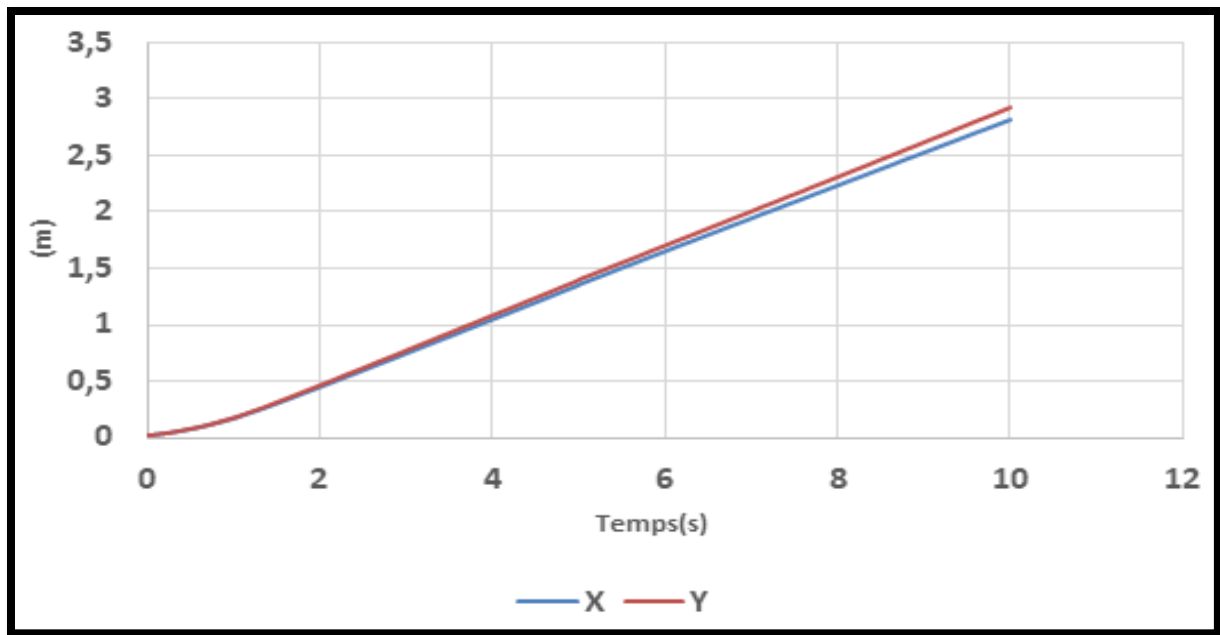


Figure 2.13 : *Base soulevée du sol* : trajectoires X et Y suite à une commande en vitesse pour un déplacement à droite en arrière de 45 °.

Pour la synthèse des paramètres des 3 régulateurs (en cascade) de la trajectoire, nous avons d'abord estimé les fonctions de transfert en boucle ouverte pour ces trois régulateurs. Ensuite, nous avons utilisé l'application '**Tuner App**' disponible dans le bloc PID de Simulink [9].

Les fonctions de transfert en boucle ouvertes pour x et y ont été estimées à partir de la courbe 2.12 (base posée sur sol). Pour la rotation nous avons utilisé la réponse en boucle ouverte avec base soulevée du sol. Ainsi, les fonctions de transfert en question sont respectivement pour x , y et  $\theta$   $G_x(p)=1.14/p$ ,  $G_y(p)=0.845/p$  et  $G_\theta(p)=1/p$ . Pour une période d'échantillonnage de 80 ms, nous avons trouvé les gains suivants :  $K_{px}=0.8771$ ,  $K_{py}=1.183$ ,  $K_{p\theta}=0.999$ .

Pour valider le contrôle proposé et illustré par simulation en régime dynamique, nous avons réalisée trois tests : commande de la base d'une position initial ( $q_1=X=1.514m$ ,  $q_2=Y=1.488m$ ,  $\phi=q_3=0$  rd)), donné vers une position finale ( $q_1=X=0m$ ,  $q_2=Y=0m$ ,  $\phi=q_3=0$  rd) donné (Figure 2.14 à 2 .16) puis une commande de la base pour effectuer un déplacement entre 4 positions formant un rectangle ABCD avec les coordonnées A (0m,0m), B (1m,0m), C (1m,2m), D (0m,2m) (Figures 2.17 à 2 .19).

Enfin, nous avons testé un contrôle de trajectoire circulaire de rayon  $R=1m$  et de centre (0,1) (Figures 2.20 à 2.22). Dans ce cas, nous avons ajouté des actions intégrales aux trois contrôleurs en question car la position n'est plus constante mais dépend du temps (il faut éliminer l'erreur de traînage). Par une même procédure du réglage [9], on trouve :  $K_{px}=0.8978$ ,  $I_x=0.22914$  ;  $K_{py}=1.0814$ ,  $I_y=0.23825$  ;  $K_{p\theta}=0.98723$ ,  $I_\theta=0.23974$ .

En analysant les résultats obtenus, on peut conclure que le contrôle de position fonctionne correctement car la base atteint une position de référence quel que soit le point de départ sans rotation (référence de rotation nulle). Par contre pour le contrôle de trajectoire, on observe une

erreur de trainage pour la position  $x$  et  $y$  (Figure 2.20) dans les zones correspondantes aux déplacements latéraux contrairement au contrôle de la rotation (erreur moyenne nulle, Figure 2.22). Ce qui démontre d'une autre manière l'existence de glissements latéraux de la base. Pour les zones où il n'y a pas de glissement le contrôle de trajectoire fonctionne correctement.

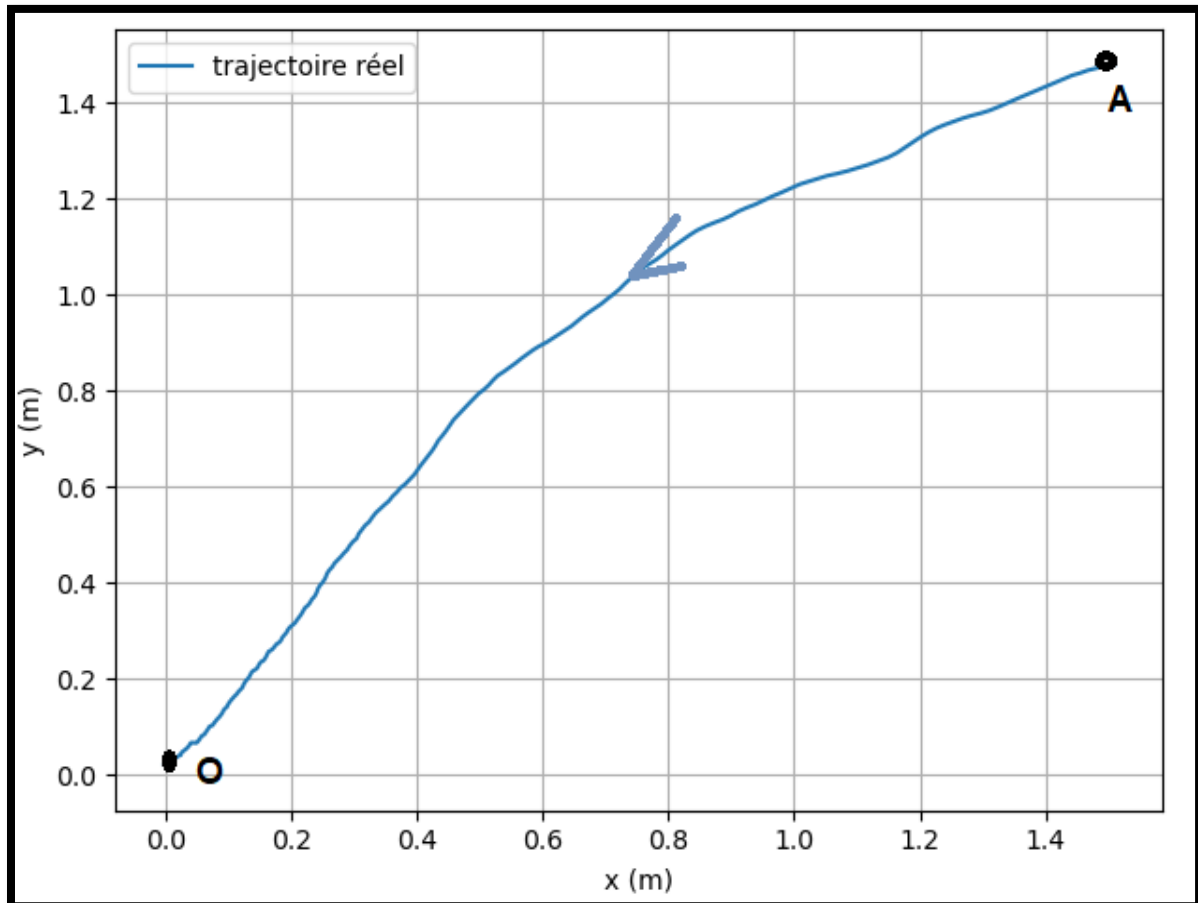


Figure 2.14 : Contrôle de position : déplacement de  $A(1.514m, 1.488m)$  vers  $O(0m, 0m)$ .

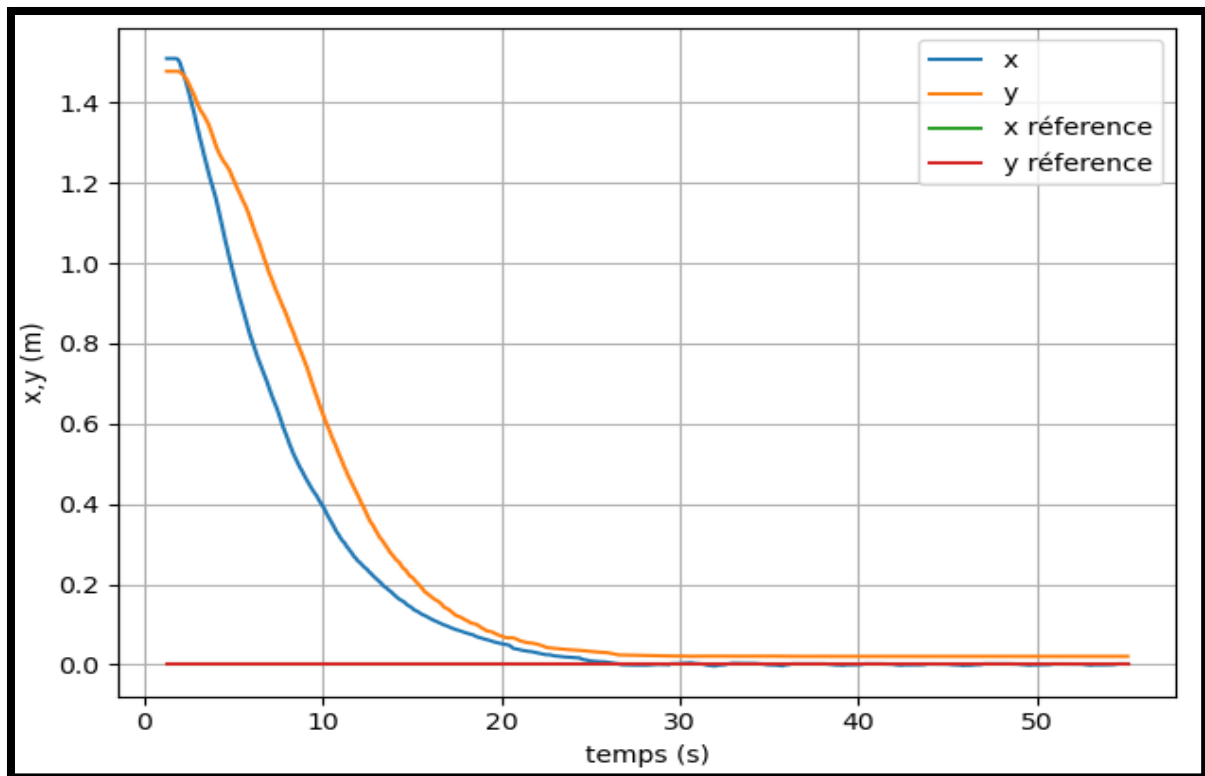


Figure 2.15 : Contrôle de position : Evolution de  $x$  et  $y$  en fonction du temps pour un déplacement de  $A(1.514m, 1.488m)$  vers  $O(0m, 0m)$

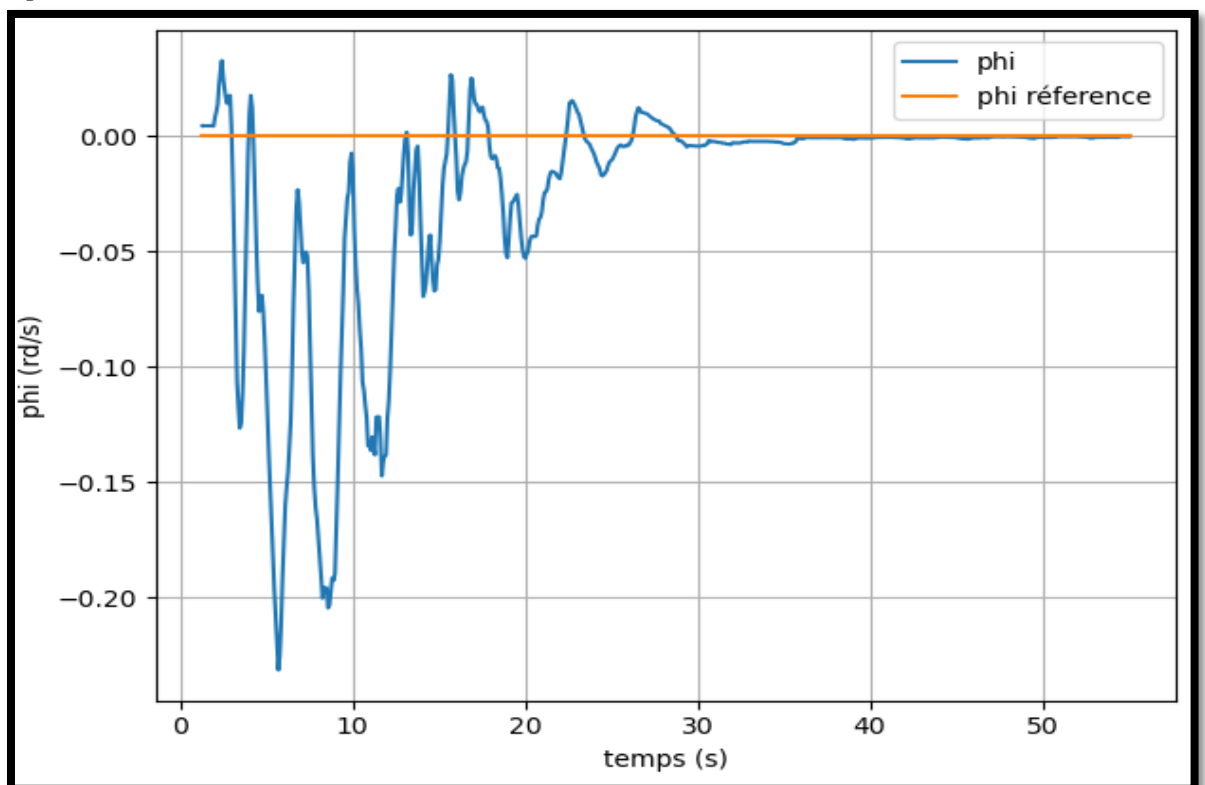


Figure 2.16 : Contrôle de position : Evolution de  $\phi = q_3$  en fonction du temps pour un déplacement de  $A(1.514m, 1.488m)$  vers  $O(0m, 0m)$

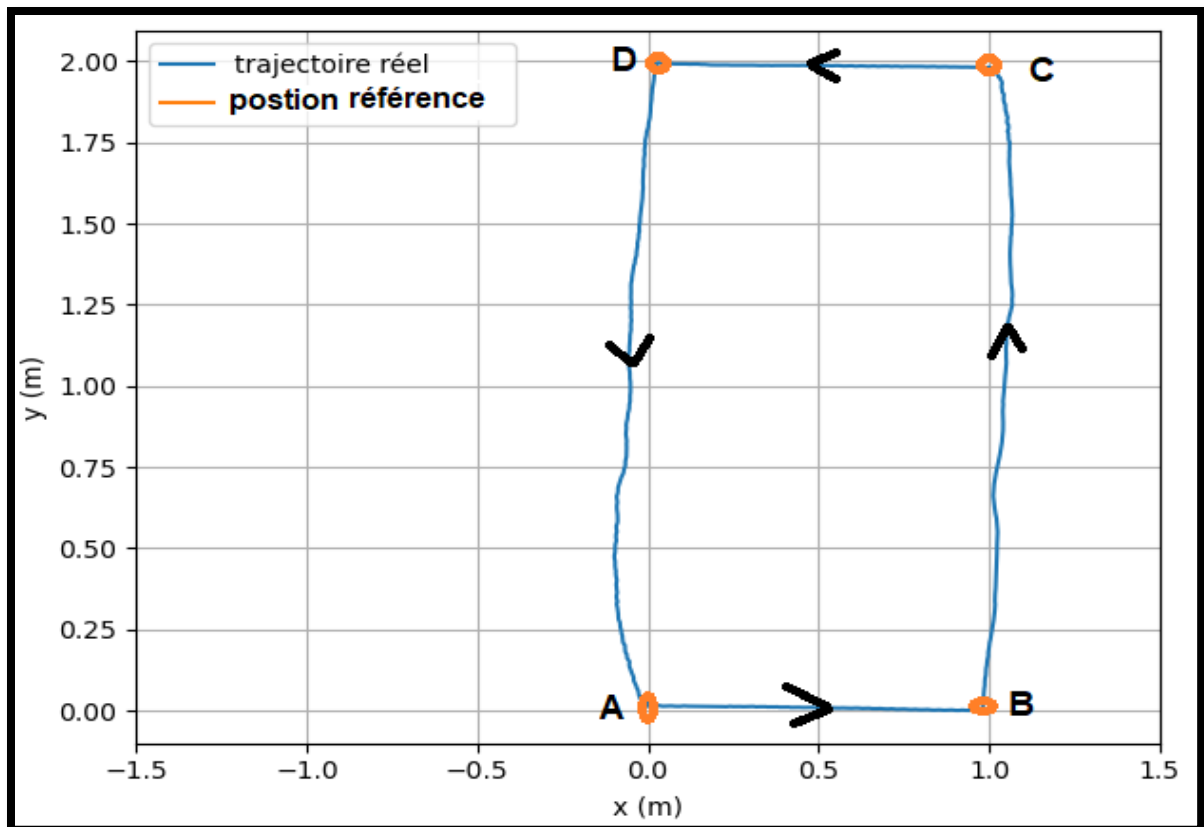


Figure 2.17: Contrôle de position : déplacement de A(0m,0m) vers B(1m,0m) vers C(1m,2m) vers D(0m,2m) vers O(0m,0m)

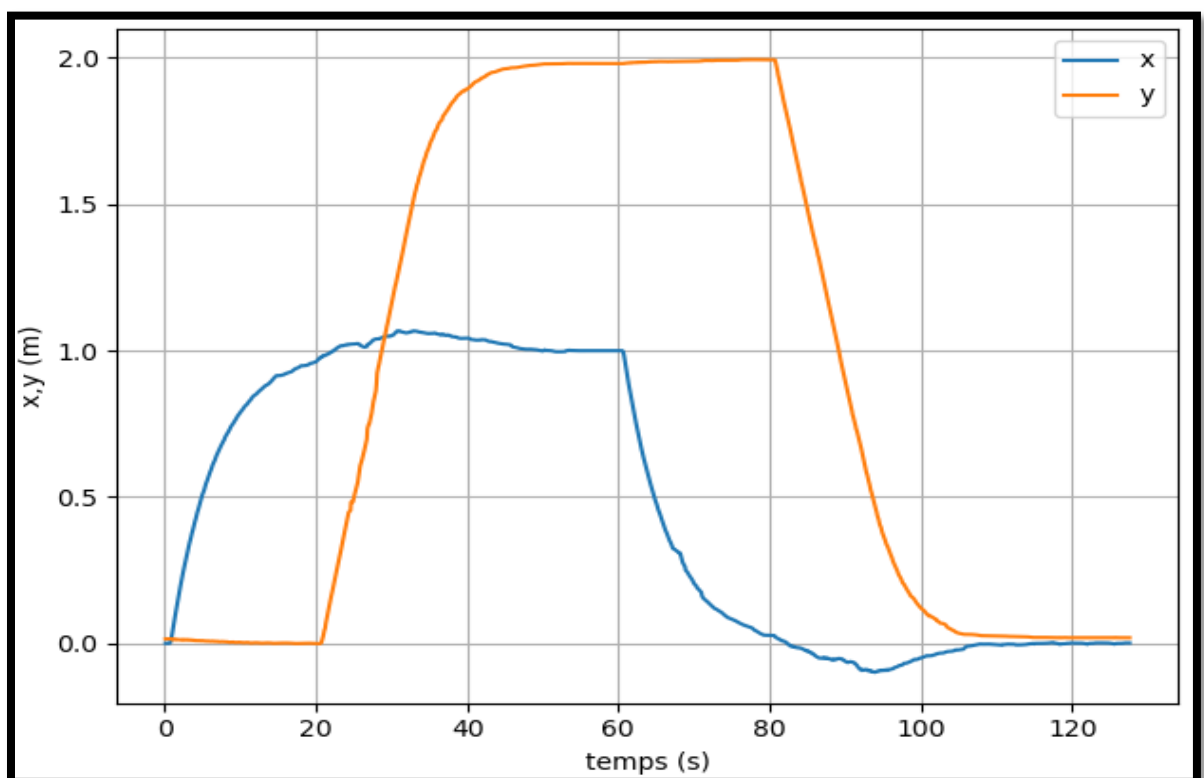


Figure 2.18: Contrôle de position Evolution de  $x,y$  en fonction du temps lors d'un déplacement de A(0m,0m) vers B(1m,0m) vers C(1m,2m) vers D(0m,2m) vers O(0m,0m)

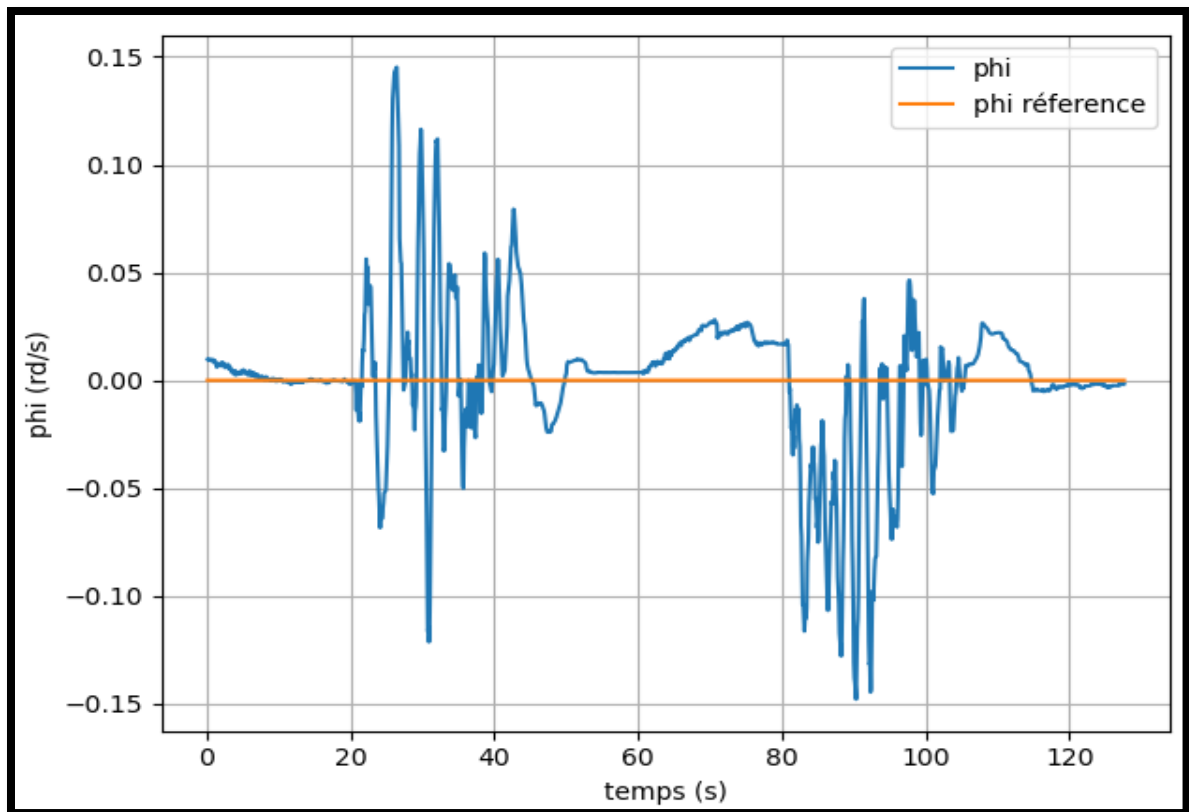


Figure 2.19: Contrôle de position Evolution de  $\phi=q_3$  en fonction du temps lors d'un déplacement de  $A(0m,0m)$  vers  $B(1m,0m)$  vers  $C(1m,2m)$  vers  $D(0m,2m)$  vers  $O(0m,0m)$

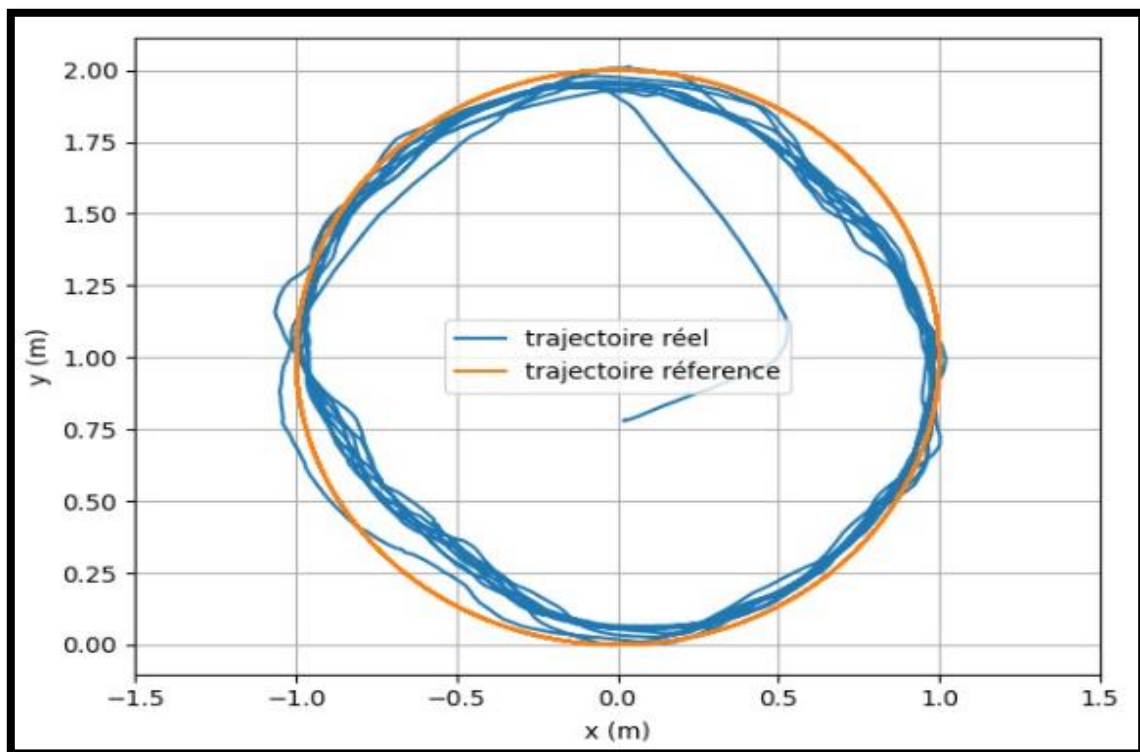


Figure 2.20: Contrôle trajectoire circulaire de rayon  $1m$  et du centre  $(0m,1m)$



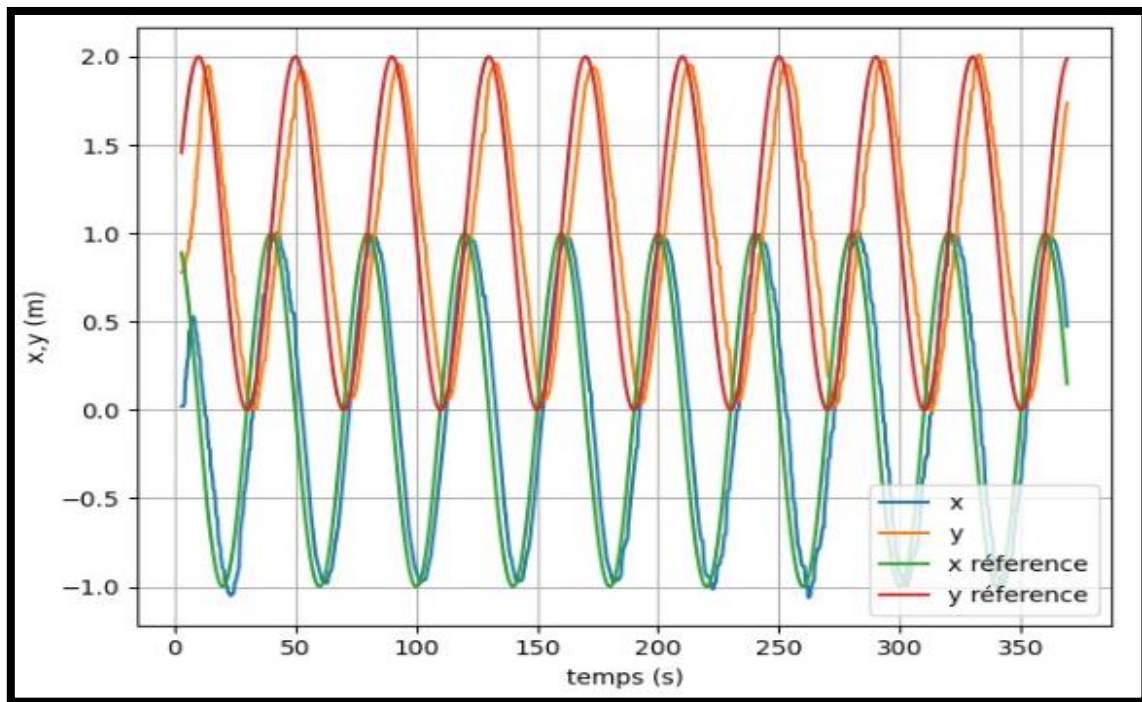


Figure 2.21: Contrôle trajectoire circulaire de rayon 1m et du centre (0m,1m) : Evolution de  $x,y$  en fonction du temps

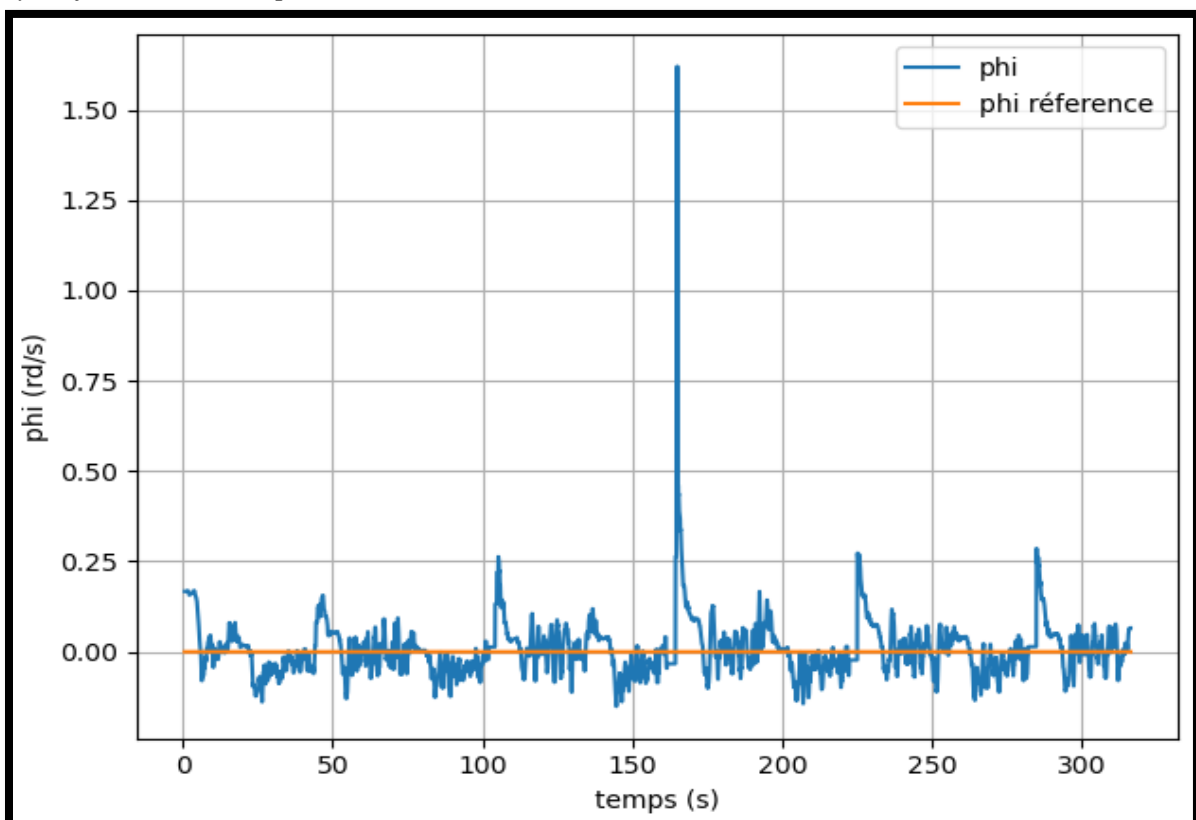


Figure 2.22: Contrôle trajectoire circulaire de rayon 1m et du centre (0m,1m) : Evolution de  $\phi=q_3$  en fonction du temps

## B- Bras manipulateur

### 1. Présentation du robot Niryo NED et type de programmation

#### 1.1 Présentation du robot Niryo NED

Niryo NED est un robot éducatif open-source qui vise à rendre l'apprentissage de la robotique accessible et amusant pour tous. Il est conçu pour être facile à utiliser, à monter et à programmer, et peut être utilisé dans une variété de contextes éducatifs, tels que les écoles, les centres de formation professionnelle et les universités. Il est basé sur une architecture robotique à 6 axes et est capable de saisir et de manipuler des objets. Il peut être programmé à l'aide de logiciels de programmation visuelle tels que Scratch, Python et ROS.



Figure 1.1.B : Logo de l'entreprise Niryo

Figure 1.2.B : Robot Niryo NED

#### 1.2 Type de programmation

Pour développer sur le robot Niryo NED, il y a plusieurs manières :

**Niryo Studio :** C'est une IHM (Interface homme machine) qui permet un contrôle rapide et direct du robot depuis un ordinateur externe.

**PyNiryo :** C'est un package de Niryo Ned qui est aussi une API (Application Programming Interface). Il permet de créer des programmes robotiques facilement et de contrôler les robots en communiquant à distance par le biais de leurs ordinateurs.

**ROS :** Niryo NED est entièrement basé sur ROS (Robot operating system), ce dernier est méta-système d'exploitation avec middleware qui est utilisé dans des projets robotiques. ROS est le moyen le plus direct de contrôler le robot car il permet d'envoyer une commande via le terminal mais aussi d'écrire un nœud Python/C++ entier dans un script afin de réaliser un processus complet.

**Modbus TCP :** Ce serveur permet de faire communiquer Ned avec un CLP (Control Language Procedure ) ou avec un autre ordinateur sur le même réseau.

Au début du projet, nous nous sommes focalisés sur ROS puisque c'est le moyen le plus direct pour contrôler le robot mais nous nous sommes aperçus que le robot est compatible avec une version du ROS datée (ROS Melodic) alors que la base mobile est programmée sous ROS2, nous avons donc contacté l'entreprise Niryo pour savoir si Niryo NED est compatible avec cette version de ROS.

La réponse étant négative ainsi que la mise en place d'une version ROS (ROS Neotic) dès Mai 2023, nous a changé d'avis afin d'explorer une autre manière pour programmer le robot, nous avons donc choisi Modbus TCP vu que la passerelle avec ROS2 pourra se faire facilement.

Notant que c'est possible de créer une passerelle entre ROS1 et ROS2 mais comme la mise en place d'une version ROS Neotic sur le robot Niryo NED est prévu pour Mai 2023, nos programmes ne seront plus valables dans ce cas-là et sera mieux de passer via Modbus TCP.

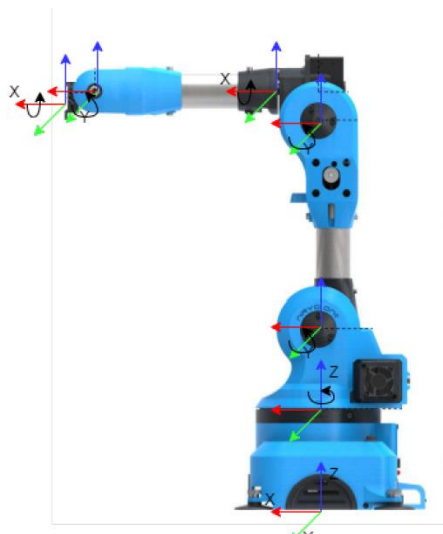
## **2. Modélisation géométrique directe du bras manipulateur et étude de l'espace du travail [1] [14] [15]**

### **2.1. Modélisation géométrique directe du bras manipulateur**

Comme beaucoup de robots manipulateurs, le robot décrit dans ce rapport est considéré comme un bras manipulateur sériel. Ces types de manipulateurs sont constitués d'un ensemble des corps rigides, appelés liens, reliés par des articulations qui permettent différents types de mouvements et qui ont différents types de restrictions, et contrairement aux manipulateurs parallèles, il n'y a pas de boucles fermées.

Pour établir le modèle géométrique direct, où le but est, étant donné un ensemble de valeurs des variables articulaires, calculer la configuration du robot (position + orientation), la solution pour la position de l'effecteur terminal est trouvée par une série de multiplications entre matrices de transformation. Pour cela, une première étape cruciale consiste à définir un ensemble de référentiels capables de définir le mouvement complet du manipulateur. Bien qu'il soit possible de définir ces référentiels selon de multiples conventions, dont certaines permettent non seulement une meilleure compréhension du mouvement du robot mais sont également plus faciles à calculer la chaîne de transformations nécessaire.

Pour ce projet spécifique, il a été défini un système de coordonnées directes pour chaque articulation, avec la particularité que tous les systèmes avaient la même orientation, comme le montre la figure, où (pour la vue du lecteur) chaque axe x pointait vers la gauche, chaque l'axe z pointait vers le haut et chaque axe y pointait vers le "spectateur", orthogonal aux axes x et y. Ci-dessous la figure montrant les repères de coordonnées pour les 6 articulations et le repère universel.



*Figure 2.1.B : Repères de coordonnées pour les 6 articulations et le repère universel*

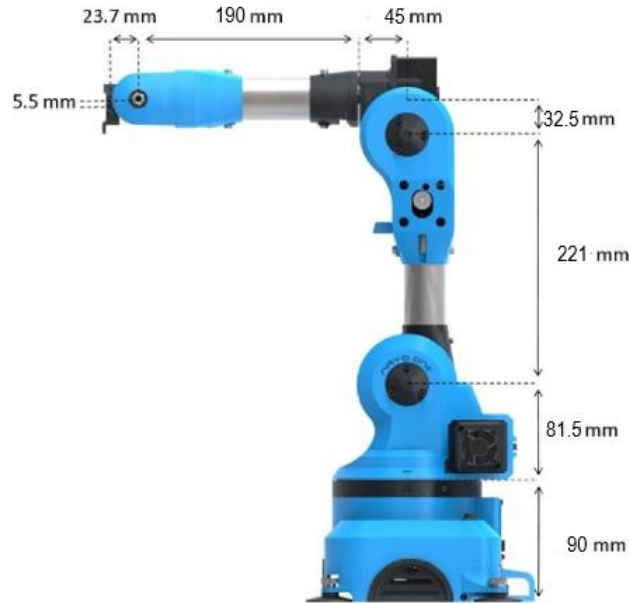


Figure 2.2.B : Dimensions entre chaque articulation

Pour la paire entre le repère universel et le repère 1 (le soi-disant lien 0), parce qu'il ne tourne que sur l'axe z, conduisant à :

$${}^0_1P = \begin{bmatrix} 0 \\ 0 \\ 90 \end{bmatrix} \quad {}^0R_z = \begin{bmatrix} \cos(\theta_1) & -\sin(\theta_1) & 0 \\ \sin(\theta_1) & \cos(\theta_1) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Nous précisons que les distances sont en mm. Ce qui mène à la matrice de transformation suivante :

$${}^0_1T = \begin{bmatrix} \cos(\theta_1) & -\sin(\theta_1) & 0 & 0 \\ \sin(\theta_1) & \cos(\theta_1) & 0 & 0 \\ 0 & 0 & 1 & 90 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Ensuite, pour la paire entre le repère 1 et le repère 2 (donc le lien 1 qui relie l'articulation 1 et l'articulation 2), car il n'y a qu'une rotation selon l'axe y, conduisant à :

$${}^1_2P = \begin{bmatrix} 0 \\ 0 \\ 81,5 \end{bmatrix} \quad {}^1R_y = \begin{bmatrix} \cos(\theta_2) & 0 & \sin(\theta_2) \\ 0 & 1 & 0 \\ -\sin(\theta_2) & 0 & \cos(\theta_2) \end{bmatrix}$$

Ce qui mène à la matrice de transformation suivante :

$${}^1_2T = \begin{bmatrix} \cos(\theta_2) & 0 & \sin(\theta_2) & 0 \\ 0 & 1 & 0 & 0 \\ -\sin(\theta_2) & 0 & \cos(\theta_2) & 81,5 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Pour la paire suivante, entre le repère 2 et le repère 3 (le lien 2 reliant les joints 2 et 3) il y a la même rotation autour de l'axe y comme décrit précédemment, conduisant à :

$${}^2_3P = \begin{bmatrix} 0 \\ 0 \\ 221 \end{bmatrix} \quad {}^2_3R_x = \begin{bmatrix} \cos(\theta_3) & 0 & \sin(\theta_3) \\ 0 & 1 & 0 \\ -\sin(\theta_3) & 0 & \cos(\theta_3) \end{bmatrix}$$

Ce qui mène à la matrice de transformation suivante :

$${}^2_3T = \begin{bmatrix} \cos(\theta_3) & 0 & \sin(\theta_3) & 0 \\ 0 & 1 & 0 & 0 \\ -\sin(\theta_3) & 0 & \cos(\theta_3) & 221 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

La paire suivante, où le repère 4 est positionné dans la quatrième articulation, une rotation autour de l'axe x est formée, conduisant à :

$${}^3_4P = \begin{bmatrix} 45 \\ 0 \\ 32,5 \end{bmatrix} \quad {}^3_4R_x = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\theta_4) & -\sin(\theta_4) \\ 0 & \sin(\theta_4) & \cos(\theta_4) \end{bmatrix}$$

Ce qui mène à la matrice de transformation suivante :

$${}^3_4T = \begin{bmatrix} 1 & 0 & 0 & 45 \\ 0 & \cos(\theta_4) & -\sin(\theta_4) & 0 \\ 0 & \sin(\theta_4) & \cos(\theta_4) & 32,5 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Puis une autre rotation autour de l'axe y pour la transformation entre les repères 4 et 5 (lien 4) et la rotation qui se produit dans la cinquième articulation, conduisant à :

$${}^4_5P = \begin{bmatrix} 190 \\ 0 \\ 0 \end{bmatrix} \quad {}^4_5R_y = \begin{bmatrix} \cos(\theta_5) & 0 & \sin(\theta_5) \\ 0 & 1 & 0 \\ -\sin(\theta_5) & 0 & \cos(\theta_5) \end{bmatrix}$$

Ce qui mène à la matrice de transformation suivante :

$${}^4_5T = \begin{bmatrix} \cos(\theta_5) & 0 & \sin(\theta_5) & 190 \\ 0 & 1 & 0 & 0 \\ -\sin(\theta_5) & 0 & \cos(\theta_5) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Pour la dernière transformation, il y a la liaison entre les repères 5 et 6 et la rotation autour de l'axe x dans l'articulation 6 ce qui conduit à

$${}^5_6P = \begin{bmatrix} 23,7 \\ 0 \\ -5,5 \end{bmatrix} \quad {}^5_6R_x = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\theta_6) & -\sin(\theta_6) \\ 0 & \sin(\theta_6) & \cos(\theta_6) \end{bmatrix}$$

Ce qui mène à la matrice de transformation suivante :

$${}^5_6T = \begin{bmatrix} 1 & 0 & 0 & 23,7 \\ 0 & \cos(\theta_6) & -\sin(\theta_6) & 0 \\ 0 & \sin(\theta_6) & \cos(\theta_6) & -5,5 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Maintenant que tous les cadres et transformations sont définis, le problème du modèle géométrique direct peut être divisé en deux parties, l'une pour trouver la position finale de l'effecteur final à l'aide d'une composition en série des matrices précédemment définies et l'autre pour trouver l'orientation de l'effecteur terminal, selon la convention des angles d'Euler Z-Y-Z, en utilisant certaines propriétés géométriques.

#### Résolution de la position :

En partant du premier problème de recherche de la position de l'effecteur terminal final dans l'espace, selon le repère universel défini, on peut dire que cette position, en coordonnées homogènes, est la dernière colonne de la matrice formée par la série de multiplication des précédents définis matrices, comme suivant :

$${}^0_6T = {}^0_1T * {}^1_2T * {}^2_3T * {}^3_4T * {}^4_5T * {}^5_6T$$

#### Résolution de l'orientation :

Pour la deuxième partie du modèle géométrique, qui consiste à trouver l'orientation de l'effecteur terminal, selon la convention Z-Y-Z utilisant les angles d'Euler, nous commençons par faire une corrélation entre la matrice de rotation précédemment définie à l'intérieur de la matrice de transformation,  ${}^0_6T$  entre le repère 6 et le repère universel,  ${}^0_6R$  et la matrice de rotation définie par les trois angles  $\alpha$ ,  $\beta$  et  $\gamma$ .

$$\begin{bmatrix} c_\alpha c_\beta c_\gamma - s_\alpha s_\gamma & -c_\alpha c_\beta s_\gamma - s_\alpha c_\gamma & c_\alpha s_\beta \\ s_\alpha c_\beta c_\gamma + c_\alpha s_\gamma & -s_\alpha c_\beta s_\gamma + c_\alpha c_\gamma & s_\alpha s_\beta \\ -s_\beta c_\gamma & s_\beta s_\gamma & c_\beta \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}$$

Par conséquent, à partir de l'inspection et du choix adéquat des équations à résoudre on obtient :

$$\begin{cases} \beta = \arctan2(\sqrt{r_{13}^2 + r_{23}^2}, r_{33}) \\ \gamma = \arctan2(\frac{r_{32}}{s_\beta}, \frac{-r_{31}}{s_\beta}), \quad \text{if } s_\beta \neq 0 \\ \alpha = \arctan2(\frac{r_{23}}{s_\beta}, \frac{r_{13}}{s_\beta}), \quad \text{if } s_\beta \neq 0. \end{cases}$$

Tout d'abord, l'angle est calculé parce que les deux autres angles dépendent de sa valeur. Si  $\sin \beta$  est différent de 0 alors la résolution des deux autres angles est facile en appliquant ce qui est avant. Si  $\sin \beta = 0$  alors  $\cos \beta = 1$  ce qui conduit à  $\alpha + \gamma = \text{atan2}(r_{21}; r_{11})$  (d'autres éléments de la matrice de rotation peuvent être choisis) et il est donc nécessaire de lever l'ambiguïté de leurs valeurs

$$\begin{cases} \gamma = 0 \\ \alpha = c_\beta \cdot \text{atan2}(r_{21}, r_{11}) \end{cases}$$

Nous tenons à rappeler que les valeurs marquées sont en radian.

## 2.2. Etude de l'enveloppe de travail du robot

Pour pouvoir valider le modèle géométrique, nous allons étudier l'enveloppe de travail du robot Niryo NED et le comparer via les capteurs de mouvement Optitrack. L'enveloppe (espace) du travail du robot trouvée sur le site du niryo est présentée ci-dessous :

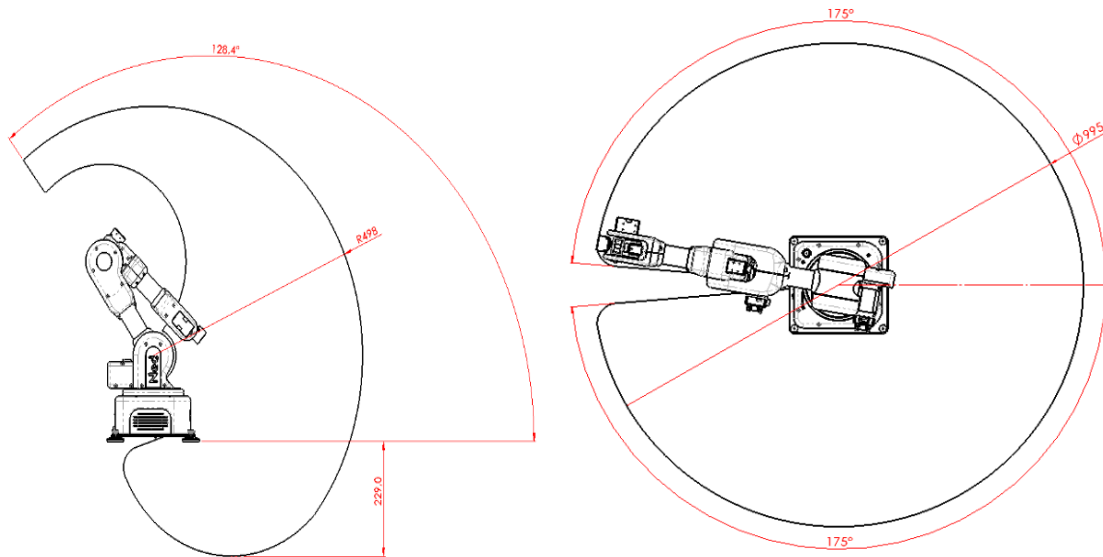


Figure 2.3.B : Enveloppe du travail du robot Niryo NED

## 3. Implémentation du modèle géométrique sous python, tests et validation des résultats avec Optitrack

### 3.1. Implémentation du modèle géométrique sous python :

Pour la partie développement du projet, nous avons utilisé **Modbus TCP** pour programmer notre robot. Un README.md (ANNEXE) pour la réutilisation du projet ainsi que le code sont déposés sur notre Gitlab de l'université [11].

Pour implémenter le modèle géométrique sous python, il fallait étudier et comprendre comment la génération des messages se fait sous Modbus TCP, en effet, sous ce dernier 4

registres sont implémentés, les coils, les entrées discrètes, les registres de maintien, ainsi que les registres d'entrée. Chaque banque de registres a un ensemble de fonctionnalités.

Les entrées discrètes et les registres d'entrée sont en lecture seule. Ils sont utilisés pour garder les états du robot. Tandis que les coils et les registres de maintien sont en lecture/écriture. Ils sont utilisés pour donner à l'utilisateur l'accès aux commandes du robot. Ces deux banques de registres ne contiennent pas les états du robot, mais les dernières commandes [12] .

Dans le cadre de ce projet nous avons dû utiliser les registres de maintien qui sont en lecture/écriture, un exemple de messages Modbus implémenté sous python est présenté ci-dessous ainsi qu'un extrait des adresses Modbus et leur description :

```
def auto_calibration():
    print ("Calibrate Robot if needed ...")
    client.write_register(310, 1)
    client.write_register(311, 1)
    # Wait for end of calibration
    while client.read_input_registers(402, 1).registers[0] == 1:
        time.sleep(0.05)
```

*Figure 3.1.B : Exemple de messages Modbus implémenté sous python*

|     |   |
|-----|---|
| 310 | Demande d'une nouvelle calibration      |
| 311 | Demarrage d'une calibration automatique |
| 312 | Demarrage d'une calibration manuelle    |

*Figure 3.2.B : Extrait des adresses Modbus ainsi que leur description*

Nous avons donc implémenté notre modèle géométrique du robot Niryo NED et nous obtenons des résultats que vous trouvez le détail dans la section suivante (3.2. Tests et validation des résultats avec Optitrack).

### **3.2. Tests et validation des résultats avec Optitrack :**

Pour tester et valider le modèle géométrique direct, nous avons dû générer une enveloppe de travail avec plusieurs points du robot Niryo NED, la première étape était alors de récupérer les limites de chaque articulation du robot dans notre cas  $q_1$ ,  $q_2$  et  $q_3$  (se référer au git pour plus de détails sur le code). Ensuite, la deuxième étape est de définir un pas d'incrément pour chaque articulation, dans notre cas on a choisi 0,1 rad pour avoir une bonne précision de l'enveloppe du travail. Finalement, nous avons utilisé « scatter » de matplotlib pour tracer chaque point dans l'enveloppe du travail, en utilisant les coordonnées x, y, z générées du modèle géométrique.

Pour le premier test, nous avons demandé à Niryo NED de générer une trajectoire de l'enveloppe du travail suivant  $q_3$  présentée dans la figure 3.3.B et nous obtenons à travers l'implémentation du modèle géométrique le résultat présenté dans la figure 3.4.B, notons qu'en **bleu** l'enveloppe du travail théorique et en **rouge** la mesure à travers le système de capteurs de mouvement Optitrack pour valider le modèle géométrique direct :



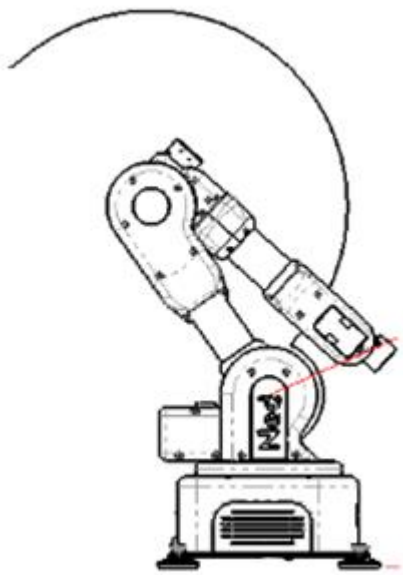


Figure 3.3.B : Trajectoire désirée

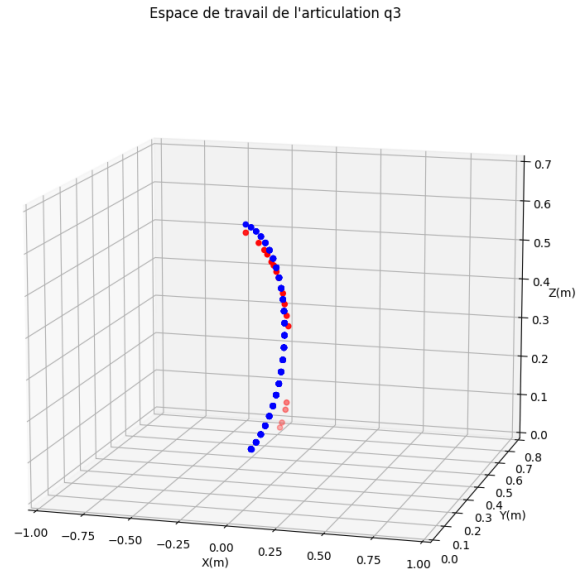


Figure 3.4.B : Enveloppe du travail suivant  $q_3$

Pour le deuxième test, nous avons demandé à Niryo NED de générer une trajectoire de l'enveloppe du travail suivant  $q_2$  présentée dans la figure 3.5.B et nous obtenons à travers l'implémentation du modèle géométrique le résultat présenté dans la figure 3.6.B, notons qu'en **bleu** l'enveloppe du travail théorique et en **rouge** la mesure à travers le système de capteurs de mouvement Optitrack pour valider le modèle géométrique direct:

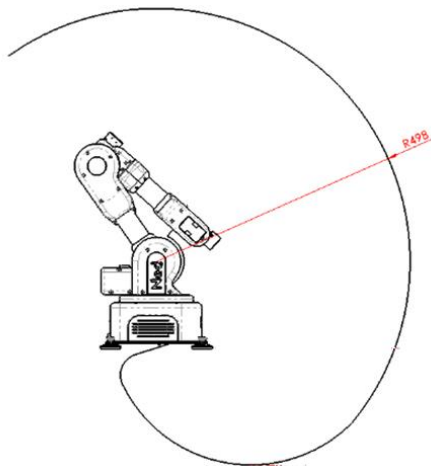


Figure 3.5.B : Trajectoire désirée

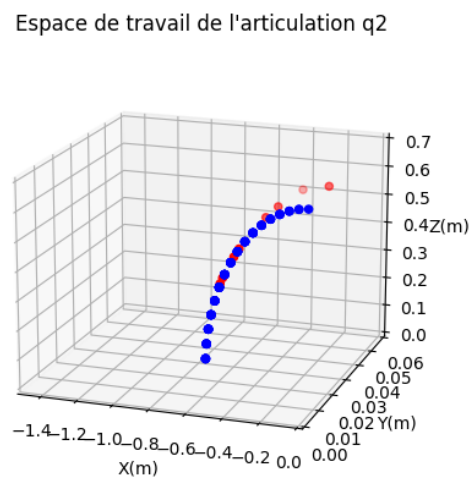
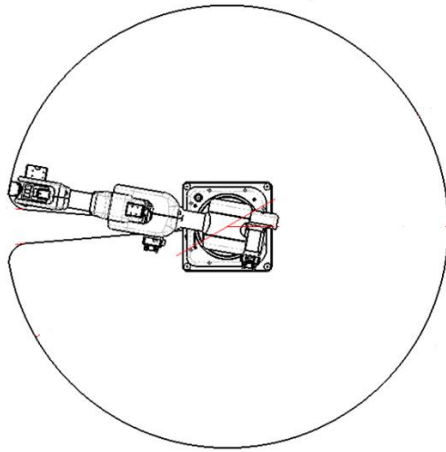
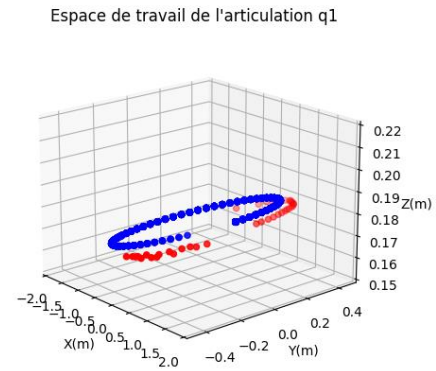


Figure 3.6.B : Enveloppe du travail suivant  $q_2$

Pour le dernier test, nous avons demandé à Niryo NED de générer une trajectoire de l'enveloppe du travail suivant  $q_1$  présentée dans la figure 3.7.B et nous obtenons à travers l'implémentation du modèle géométrique le résultat présenté dans la figure 3.8.B, notons qu'en **bleu** l'enveloppe du travail théorique et en **rouge** la mesure à travers le système de capteurs de mouvement Optitrack pour valider le modèle géométrique direct:



*Figure 3.7.B : Trajectoire désirée*



*Figure 3.4.B : Enveloppe du travail suivant  $q1$*

En comparant les résultats obtenus théoriquement avec ceux avec le système de capteurs de mouvement Optitrack, nous remarquons qu'il y a un petit décalage dû au manque de certaines dimensions entre les différentes articulations notamment les distances entre les articulations autrement dit les variables articulaires. [13]

## C- Contrôle en simultané

Pour le contrôle simultané nous proposons le schéma de contrôle de la figure 4 qu'il faut valider par la suite. Ceci ne peut être réalisé qu'après avoir développé et implémenté un modèle cinématique du bras du robot en question. Chose que nous n'avons pas pu faire par manque du temps et les contraintes rencontrées.

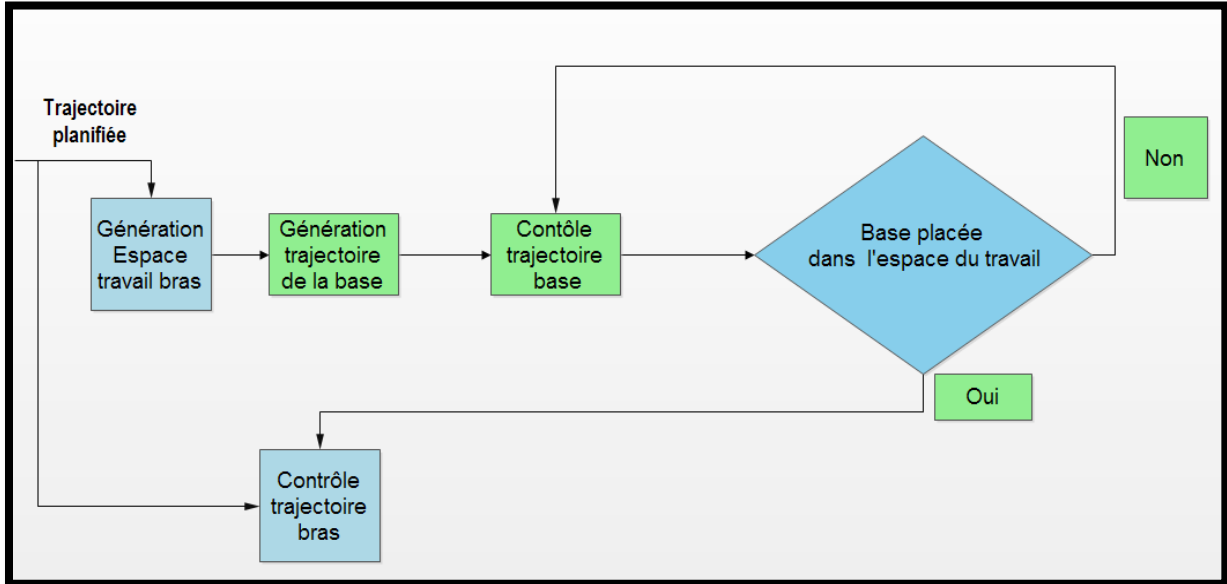


Figure 4.C: Organigramme global d'une éventuelle commande simultanée de la base et du bras robot

## Conclusion

Notre projet de fin d'étude a porté sur le contrôle de trajectoire d'un prototype de robot mobile composé d'une base mobile à roues mecanum et d'un bras manipulateur type Niryo. Une modélisation avec simulation de notre système est la première approche par laquelle nous avons commencé pour bien comprendre les différents mouvements possibles de notre système. Ce qui nous a permis de proposer un contrôle de la base que nous avons implémenté et testé en temps réel.

Les résultats obtenus ont permis de valider le modèle cinématique d'une part et de révéler l'existence de glissement lors de déplacements latéraux de la base d'autre part.

Aussi, ils ont montré le bon fonctionnement du contrôle de position de la base. Mais nous avons observé que le contrôle de trajectoire circulaire par exemple, s'effectue avec une erreur de trainage sur les positions  $x$  et  $y$  dans les zones où la base effectue des déplacements. Dans les autres zones le contrôle s'effectue correctement. Ce qui démontre encore une fois l'existence de glissement principalement pour les déplacements latéraux.

Il faut bien noter que rien que la modélisation cinématique et dynamique de la base mobile a fait l'objet de plusieurs travaux de recherche. Ceci pour montrer le temps que nous a pris notre travail pour ajouter en plus le contrôle de trajectoire de la base en question.

Une même étude doit être effectuée pour le bras du robot pour ensuite proposer un contrôle simultané de l'ensemble base et bras. A ce niveau, nous avons développé et implémenté un modèle géométrique du bras grâce auquel nous avons tracé en temps réel l'espace du travail du bras. Malheureusement par manque du temps, en plus des contraintes rencontrées (que nous détaillerons juste en dessous) nous n'avons pas pu terminer le contrôle du bras pour exécuter une trajectoire planifiée et ensuite proposer une stratégie de contrôle en simultané de l'ensemble. Ce qui nécessite une continuité de ce travail dans le cadre de PFE futurs.

## Les contraintes

Lors de ce projet, nous avons rencontré beaucoup de difficultés, tout d'abord la documentation sur la programmation du microcontrôleur de la base mobile était incomplète, de plus l'absence d'un travail existant sur la base mobile pour bien démarrer comme c'était le cas comme d'habitude, ce qui nous a engendré un temps de plus pour la partie bibliographie (se référer au GANTT réel tableau ci-dessus

Notons aussi la mise à jour régulière du contrôleur de la base mobile effectué par le chercheur ce qui nous implique la mise à jour aussi de la programmation ainsi que du modèle cinématique ; sans oublier les pannes survenues sur la base mobile qui nécessitaient l'intervention de M. Sanz Lopez à chaque panne.

Concernant la partie Niryo, nous avons pris beaucoup de temps à nous former sur ROS2 alors que le bras manipulateur n'est pas compatible avec ROS2 après avoir contacté NIRYO. Pour modéliser le robot Niryo NED, nous avons manqué des distances entre chaque articulation malgré le fait qu'on a contacté Niryo pour nous fournir les spécifications techniques du NED, nous avons essayé de mesurer ces distances en nous basant sur les spécifications fournies du robot Niryo One.

Au fil de ce projet, des étapes supplémentaires que nous n'avions pas prévu à l'origine sont apparues, mais nous ne sommes pas arrêtés en essayant toujours de trouver une autre solution alternative pour pouvoir avancer dans le projet.

## Les apports de notre Projet Ingénieur

Lors de ce projet ingénieur, nous avons pris le temps de déterminer en amont nos objectifs, ce qui nous a permis d'organiser notre travail. Nous avons également appris à faire face aux imprévus qui peuvent surgir à n'importe quel moment dans un projet. L'instauration des outils de gestion de projet et de suivi avec nos tuteurs, nous a permis de maintenir un rythme de travail constant tout au long du projet.

D'un point de vue technique, notre Projet Ingénieur nous a demandé une certaine polyvalence puisqu'il nécessite des compétences en informatique (ROS2, Python, C++), des compétences en réseaux/réseaux industriels, et une bonne communication et gestion de projet au sein de l'équipe. Nous avons également eu la chance d'étudier des technologies de capteur relativement récentes (les capteurs de mouvement Optitrack). La réalisation de ce projet nous a demandé beaucoup de rigueur et nous a nécessité un bon nombre de recherches et de renseignements auprès des chercheurs pour nous éclairer sur les difficultés rencontrées. Nous espérons que nos travaux et nos résultats aideront les chercheurs pour la suite du projet CIRMAP.

Nos remerciements vont à nos tuteurs, Gérald Dherbomez et Othman Lakhal, mais aussi à M. Sanz Lopez. Ainsi qu'aux autres chercheurs du lab CRISAL qui nous ont aidé lors de ce Projet Ingénieur.

## Critique et futur du projet

Malgré les difficultés surmontées, les conclusions tirées de notre projet présentent quelques défauts que l'on se doit de préciser. Tout d'abord, l'algorithme de contrôle de trajectoire de la base mobile sous ROS2 nécessite une amélioration pour tenir compte des glissements. Le code est disponible sur git avec une documentation en accompagnement du code prévu pour la réutilisation et la continuité du projet.

L'analyse des résultats obtenus nous a montré que la perte de précision est aussi due aux glissements. En effet, un nouveau prototype qui est plus adhérent au sol est en cours de fabrication. Ensuite, en ce qui concerne le bras Niryo, le modèle géométrique n'est pas tout à fait fiable. Malheureusement nous avons contacté l'entreprise Niryo, mais on n'a pas pu obtenir une documentation donc il faut revoir ce modèle.

Enfin, pour la partie contrôle en simultané, nous avons pu constater que lorsque le bras bouge, la répartition du centre de la masse bouge aussi, et donc fait apparaître des glissements et avec une perte de précision. Pour améliorer cette partie, on propose de développer un modèle dynamique avec un contrôle par la suite, ou sinon travailler avec un nouveau prototype de façon que le poids du bras soit négligeable devant le poids de la base.

## Annexes

### A. Suivi Projet (extrait du suivi)

- Suivre le tutoriel ROS et essayer de créer un noeud ROS
- Dépôt GIT

19/10/2022

- Réception du PC, installation du ROS2 et suivi des tutoriels ROS2
- Changement d'alimentation fonctionnel

20/10/2022

- Réussite de la communication ROS2 avec la base mobile

21/10/2022

- Noeud qui envoie une commande simple au robot.
- Noeud qui écoute la commande envoyée...
- Etude bibliographique Niryo

24/10/2022

- Problème accès internet [résolu]
- Code qui permet de contrôler la base mobile par un clavier (mouvement longitudinal avant/arrière, latéral gauche/droite, mouvement 45deg dans les 4 sens, rotation dans les 2 sens) / fréquence d'envoi des commandes 10Hz pour les tests et ça marche bien.
- Études bibliographique bras Niryo et test des différents mouvements du bras sur Niryo studio.

16/11/2022

- [Base mobile] Début de odométrie
- [Niryo] Installation du ROS Ubuntu 18 et les différents packages

17/11/2022

- [Base mobile] Décoder les messages CAN des encodeurs
- [Niryo] Problème de connexion du robot niryo avec le localhost
- Problème résolu: Dorénavant connexion par Ethernet

21/11/2022

- [Base mobile] Etalonnage de la base mobile.
- [Niryo] Début de connexion du niryo via SSH

22/11/2022

- [Base mobile] Problème détecté dans les paramètres Ki et Kp des contrôleurs des moteurs.
- [Niryo] Point avec Mario : Pas besoin de passer par Ubuntu 18 (Melodic) pour l'instant se concentrer sur la connexion du niryo via Modbus



## B. Dépôt git

- <https://gitlab.univ-lille.fr/pretil/cirmap/omniblue.git>

22/01/2023 22:57

pretil / CIRMMap / omniblue · GitLab



Update README.md

Haytham Rabi authored 3 weeks ago

| Name                      | Last commit                      | Last update |
|---------------------------|----------------------------------|-------------|
| <a href="#">README.md</a> | <a href="#">Update README.md</a> | 3 weeks ago |

README.md

### Omniblue

Omniblue est un robot mobile prototype du robot Matrice, conçue pour la fabrication additive. Il est composé d'une base mobile holonome (développée à central Lille) et d'un bras manipulateur Niryo Ned. La partie de la base mobile est développée par Haytham RABI, et la partie bras manipulateur par Ayman Moummadi.

Ce projet est sous la supervision de :

- Othman Lakhali
- Gerald Dherbomez
- Mario Sanz Lopez

POLYTECH LILLE - LABORATOIRE CRISTAL 2022/2023

### Description

Vous trouverez dans ce dépôt git les différents noeuds et packages ROS2 pour la mise en marche du robot Omniblue.

Ce dépôt est divisé en 3 branches :

- Main : Read\_Me.
- Base mobile : pour la base holonome.
- Bras manipulateur : pour le bras manipulateur.

La documentation et la mise en marche sont explicitées dans le Read\_me des deux branches.

## Référence bibliographique et webographique

[1] : Lakhal O.,2018. Contribution to the modeling and control of hyper-redundant robots : application to additive manufacturing in the construction. PH. D. THESIS. University of Lille.

[2] : Abdelrahman M. et al, 2014. A DESCRIPTION OF THE DYNAMICS OF A FOUR-WHEEL MECANUM MOBILE SYSTEM AS A BASIS FOR A PLATFORM CONCEPT FOR SPECIAL PURPOSE VEHICLES FOR DISABLED PERSONS. 58th ILMENAU SCIENTIFIC COLLOQUIUM Technische Universität Ilmenau.

[3] : HENDZEL Z. et RYKAŁA L, 2017. MODELLING OF DYNAMICS OF A WHEELED MOBILE ROBOT WITH MECANUM WHEELS WITH THE USE OF LAGRANGE EQUATIONS OF THE SECOND KIND. *Int. J. of Applied Mechanics and Engineering*, 2017, vol.22, No.1, pp.81-99

[4][https://perso.univ-lyon1.fr/marc.buffat/COURS/MECA\\_HTML/Cinematique\\_mecanum.html#Applications-num%C3%A9riques](https://perso.univ-lyon1.fr/marc.buffat/COURS/MECA_HTML/Cinematique_mecanum.html#Applications-num%C3%A9riques)

[5] : Renhui Z. et all, 2019. Trajectory tracking for omnidirectional mecanum robot with longitudinal slipping. MATEC Web of Conferences **256**, 0 (2019). <https://doi.org/10.1051/mateconf/201925602003>.

[6] : Taheri H. et all,2015. Kinematic Model of a Four Mecanum Wheeled Mobile Robot. International Journal of Computer Applications (0975 – 8887) Volume 113 – No. 3, March 2015.

[7] : Igor Z. et Klaus Z. , 2019. Dynamics of a four-wheeled mobile robot with Mecanum wheels. ZAMM - Journal of Applied Mathematics and Mechanics Published by Wiley-VCH Verlag GmbH & Co. KGaA.

[9] : <https://nl.mathworks.com/help/control/ref/pidtuner-app.html>

[10] : [https://github.com/tud-cor-sr/ros2-mocap\\_optitrack](https://github.com/tud-cor-sr/ros2-mocap_optitrack)

[11] (<https://gitlab.univ-lille.fr/pretil/cirmap/omniblue>).

[12] : [https://docs.niryo.com/dev/modbus/v3.2.0/fr/source/api\\_documentation.html](https://docs.niryo.com/dev/modbus/v3.2.0/fr/source/api_documentation.html))

[13] : <https://fr.wikipedia.org/wiki/Denavit-Hartenberg>

[14]: [https://github.com/Joao-Tiago-Almeida/Direct-and-Inverse-Kinematics-of-SerialManipulators--Nyrio-One-6-axis-Robotic-Arm/blob/main/Direct and Inverse Kinematics of Serial Manipulators \(Nyrio One 6-axis Robotic Arm\).pdf](https://github.com/Joao-Tiago-Almeida/Direct-and-Inverse-Kinematics-of-SerialManipulators--Nyrio-One-6-axis-Robotic-Arm/blob/main/Direct%20and%20Inverse%20Kinematics%20of%20Serial%20Manipulators%20(Nyrio%20One%206-axis%20Robotic%20Arm).pdf)

[15] <https://uvadoc.uva.es/bitstream/handle/10324/53862/TFG-I-2200.pdf?sequence=1&isAllowed=y>