REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE

Université de Mohamed El-Bachir El-Ibrahimi - Bordj Bou Arreridj

Faculté *des Sciences et de la technologie*

Département d'Electronique

Mémoire

Présenté pour obtenir

LE DIPLOME DE MASTER

FILIERE : Electronique

Spécialité : Electronique des Systèmes Embarqués

Par :

- **BELFAR Anes**
- **ANANE Youcef Abderrahim**

Intitulé

Elaboration d'un Langage de Programmation pour le Robot Educatif ED-7220C

Evalué le : … / … / 2022

Par la commission d'évaluation composée de* :

Nom & Prénom	*Grade*	*Qualité*	*Etablissement*
Mme. HAMADACHE Fouzia	*MAA*	*Président*	*Univ-BBA*
Dr. TALBI Mohamed Lamine	*MCA*	*Encadreur*	*Univ-BBA*
Pr. SARRA Mustapha	*PR.*	*Examinateur*	*Univ-BBA*

Année Universitaire 2021/2022

Dédicaces

A nos chers parents, pour tous leurs sacrifices, leur amour, leur tendresse, leur soutien et leurs prières tout au long de nos études.

A nos chères sœurs …… ... pour leurs encouragements permanents, et leur soutien moral.

A nos chers frères, ……, pour leur appui et leur encouragement.

A toute mes familles pour leur soutien tout au long de notre parcours universitaire.

Que ce travail soit l'accomplissement de vos vœux tant allégués, et le fuit de votre soutien infaillible.

Au Dr. TALBI Mohamed Lamine, pour sa supervision et son suivi de notre projet de fin d'études, et son soutien pour que nous nous efforcions et continuions à travailler.

Merci d'être toujours là pour nous.

Remerciements

Nous tenons à exprimer toute nos reconnaissances à notre directeur de mémoire, Dr. TALBI Mohamed Lamine. Nous le remercions de nous avoir encadré, orienté, aidé et conseillé.

Nous adressons nos sincères remerciements à tous les professeurs du département d'électronique.

Nous remercions nos très chers parents, qui ont toujours été là pour nous. Nous remercions nos sœurs, et nos frères, pour leurs encouragements.

Enfin, nous remercions nos amis qui ont toujours été là pour nous. Leur soutien inconditionnel et leurs encouragements ont été d'une grande aide.

À tous ces intervenants, nous présentons nos remerciements, nos respects et nos gratitudes.

Résumé

L'objectif de ce projet de fin d'études est la réhabilitation de la partie Software du robot pédagogique ED-7220C. Cette réhabilitation consiste essentiellement en la proposition d'un langage de programmation qui permet l'exploitation des ressources matérielles et logiciels du robot dans l'environnement de programmation MATLAB.

La communication avec la carte de commande du robot basée sur deux cartes de prototypage ARDUINO Méga 2560 a été assurée via la librairie « MATLAB support package for ARDUINO ».

Le langage proposé est composé de trois types d'instructions, à savoir, les instructions basées sur le MGD, les instructions basées sur le MGI et les instructions d'initialisation et de configuration.

L'un des avantages majeurs du jeu d'instruction proposé est que l'exécution se fait directement dans l'environnement MATLAB ce qui permet d'étendre le jeu d'instruction et d'implémentation des commandes avancées.

Mots-clés : *Robot pédagogique ED-7220C, réhabilitation, la partie Software, langage de programmation, l'environnement de programmation MATLAB, ARDUINO Méga 2560, librairie, MATLAB support package for ARDUINO, Model Géométrique Direct, Model Géométrique Inverse, commandes avancées.*

ملخص

الهدف من مشروع نهاية التخرج هذا هو إعادة تأهيل الجزء البرمجي للروبوت التعليمي *ED-7220C*. تتكون عملية إعادة التأهيل هذه أساسًا من اقتراح لغة برمجة تسمح باستغلال موارد الأجهزة والبرامج الخاص بالروبوت في بيئة برمجة *MATLAB*.

تمت عملية الاتصال مع لوحة التحكم في الروبوت باستعمال لوحتيّ *ARDUINO Mega 2560*, وتمت عملية الاتصال البرمجية عن طريق اعتماد حزمة « *MATLAB support package for Arduino hardware* ».

تتكون اللغة المقترحة من ثلاثة أنواع من التعليمات ، وهي التعليمات المستندة إلى النموذج الهندسي المباشر, والتعليمات المستندة إلى النموذج الهندسي العكسي, .وتعليمات التهيئة والتكوين.

تتمثل إحدى المزايا الرئيسية لمجموعة التعليمات المقترحة في أن التنفيذ يتم مباشرة في بيئة *MATLAB*, مما يجعل من الممكن تمديد مجموعة التعليمات وتنفيذ الأوامر المتقدمة.

الكلمات المفتاحية: روبوت تعليمي ED-7220C ، إعادة تأهيل ، الجزء البرمجي ، لغة برمجة ، بيئة برمجة MATLAB، ARDUINO Mega 2560 ، مكتبة ، حزمة دعم MATLAB لـ ARDUINO ، النموذج الهندسي المباشر ، النموذج الهندسي العكسي, أوامر متقدمة.

Abstract

The objective of this graduation project is the rehabilitation of the Software part of the ED-7220C educational robot. This rehabilitation essentially consists of the proposal of a programming language which allows the exploitation of the hardware and software resources of the robot in the MATLAB programming environment.

Communication with the robot control board based on two ARDUINO Mega 2560 prototyping boards was ensured via the «MATLAB support package for ARDUINO» library.

The proposed language is composed of three types of instructions, namely, Forward Kinematics based instructions, Inverse Kinematics based instructions, and initialization and configuration instructions.

One of the major advantages of the proposed instruction set is that the execution is done directly in the MATLAB environment, which makes it possible to extend the instruction set and implement advanced commands.

Keywords: *ED-7220C educational robot, rehabilitation, Software part, programming language, MATLAB programming environment, ARDUINO Mega 2560, library, MATLAB support package for ARDUINO, Direct Geometric Model, Inverse Geometric Model, advanced commands.*

Sommaire :

Liste des Figures :

Chapitre I : Généralités sur les robots manipulateurs

Chapitre II : Langage de programmation des Robots

Chapitre III : Programmation du robot pédagogique ED-7220C sous MATLAB

Liste des Tableaux :

Chapitre I : Généralités sur les robots manipulateurs

Chapitre III : Programmation du robot pédagogique ED-7220C sous MATLAB

Abréviations :

- ***DoF: Degree Of Liberty.***
- ***PWM: Puls Widht Modulation.***
- ***CC: Courant Continu.***
- ***DC: Direct Current.***
- ***AC: Alternativ Current.***
- ***RPM: Revolution Per Minute.***
- ***Arduino IO: Arduino Input Output.***
- ***USB: Universel Serial Bus.***
- ***PC: Personal Computer.***
- ***DHP: Denavit Hartenberg Parameters.***
- ***MGI: Model Géométrique Inverse.***
- ***MGD: Modèle Géométrique Direct.***
- ***IBM: International Business Machines.***
- ***MATLAB: MATrix LABoratory.***
- ***IDE: Integrated Development Environment.***
- ***CNC: Computer Numerical Control.***
- ***I2C: Inter-Integrated Circuits.***

Introduction Générale

Introduction Générale

Les robots manipulateurs sont l'un des systèmes mécatroniques les plus utilisés dans l'industrie, dont les applications comprennent l'assemblage d'éléments, ainsi que le soudage et la peinture des pièces. En raison de sa grande utilité dans l'industrie, il est très important d'étudier sa cinématique, sa dynamique ainsi que son contrôle automatique. Une caractéristique des robots manipulateurs est qu'ils sont généralement fabriqués avec une architecture fermée. Une fois qu'un robot atteint sa fin de vie, il est revendu, réutilisé ou recyclé.

L'objectif de ce projet de fin d'études est la réhabilitation de la partie Software du robot pédagogique ED-7220C disponible au niveau des laboratoires pédagogiques du département d'électronique.

Ce projet est la suite d'un projet de fin d'études dont l'objectif été la réhabilitation de la partie Hardware du robot. Dans le projet PFE de l'année passée, l'ancienne carte de commande du robot a été remplacée par une nouvelle architecture basée sur deux cartes Arduino Méga. [1]

Dans ce projet, nous proposons d'exploiter la nouvelle carte de commande développée pour créer des tâches robotisées, pour cela nous proposons une démarche à deux étapes. Dans la première étape, nous allons intégrer la carte de commande dans l'environnement Matlab. Dans la deuxième étape, nous allons proposer un jeu d'instructions d'un langage dédié à ce robot, afin de créer des tâches robotisées.

Ce mémoire est divisé en trois chapitres, dans le premier chapitre, nous allons présenter quelques notions de la robotique, une partie de ce chapitre sera consacré à une la description du robot ED 7220C.

Le deuxième chapitre est destiné aux notions générales sur les langages de programmation des robots, une partie de ce chapitre sera consacré à la description des deux logiciels (Matlab et Arduino) et aux plateformes de programmation que nous allons utiliser dans la programmation du langage du robot ED 7220 C.

Dans le dernier chapitre, nous allons présenter le langage de programmation que nous allons développer ainsi que les résultats obtenus.

Nous terminerons ce mémoire avec une conclusion générale et les perspectives.

Chapitre I

Généralités sur les robots manipulateurs.

1. Introduction

Un robot est un système mécanique articulé et actionné contrôlé par un ordinateur. La configuration d'un manipulateur est la description de la position de chaque point du manipulateur. Un robot aurait **N** degrés de liberté si sa configuration peut être définie par un minimum de **N** paramètres.[1]

Sur un robot parallèle, l'effecteur final est relié au sol par plusieurs chaînes cinématiques indépendantes. Sur un robot série, l'effecteur final est relié au sol par une seule chaîne cinématique.

2. Classification des architectures des robots manipulateurs

Vu le nombre important de tâches confiées aux robots, plusieurs architectures ont vu le jour depuis l'introduction de la robotique, nous citons dans cette section quelques-unes les plus utilisées [2] :

a) Robots SCARA :(Selective Compliance Articulated Robot for Assembly)

- 4 articulations série, RRPR, 4 DOF.
- Espace de travail cylindrique.
- Précise.
- Très rapide.

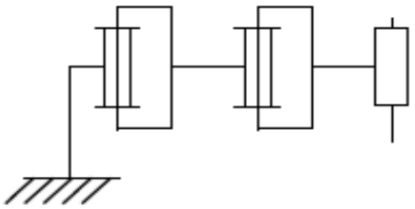


Figure I.1 : *Architectures des robots SCARA.*

b) Robots cartésiens :

- 3 articulations, série, PPP, 3 DoF.
- 3 articulations prismatiques perpendiculaires.
- Très bonne précision.
- Facile à contrôler.
- Lent.

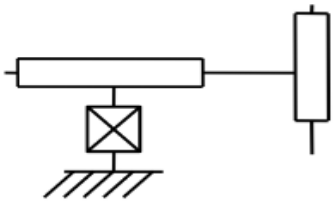


Figure I.2 : *Architectures des robots cartésiens.*

c) Robots parallèles :

- 4 articulations, parallèles, RP+RP, 3 DoF
- Espace de travail limité
- Haute rigidité
- Hautes performances dynamiques

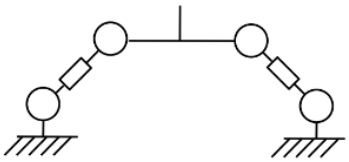


Figure I.3 : *Architectures des robots parallèles.*

3. Types des articulations

Il y a 2 types de articulations [3]

a) Transrationnelle / Prismatique

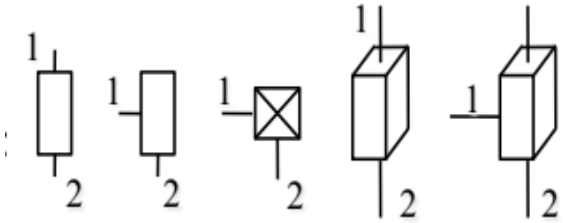


Figure I.4 : *Type d'articulation Transrationnelle / Prismatique.*

b) Rotationnel

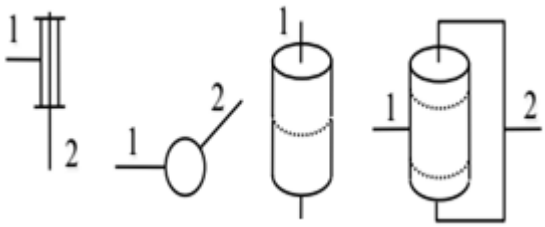


Figure I.5 : *Type d'articulation Rotationnel.*

4. Structure générale de robot manipulateur ED 7220C

Le bras robotique a été largement utilisé dans la recherche, le développement et l'enseignement. Il s'agit essentiellement d'un manipulateur en série ayant toutes les articulations comme rotoïde.

La configuration géométrique du bras est composée de la taille, de l'épaule, du coude et du poignet en correspondance avec les articulations du bras humain (Figure I.6). Chacune de ces articulations, à l'exception du poignet, a un seul DOF.

Le poignet peut se déplacer dans deux plans (roulis et tangage), rendant ainsi l'effecteur effecteur plus flexible en termes de manipulation d'objet.

Construit de manière articulée verticalement, le robot offre une observation visuelle du comportement mécanique de chaque articulation en un coup d'œil. Le bras est entièrement actionné avec chaque DOF réalisé par un servomoteur précis (DME 38B50 G-115) équipé d'un encodeur optique (Figure I.7).

L'effecteur terminal est une pince à deux états dotés de coussinets en caoutchouc [4]. Les limites de sécurité mécaniques intégrées limitent le mouvement de l'articulation en cas de problème dans l'algorithme de contrôle.

Tableau I.1 répertorie les principales caractéristiques du bras robotique ED7220C.

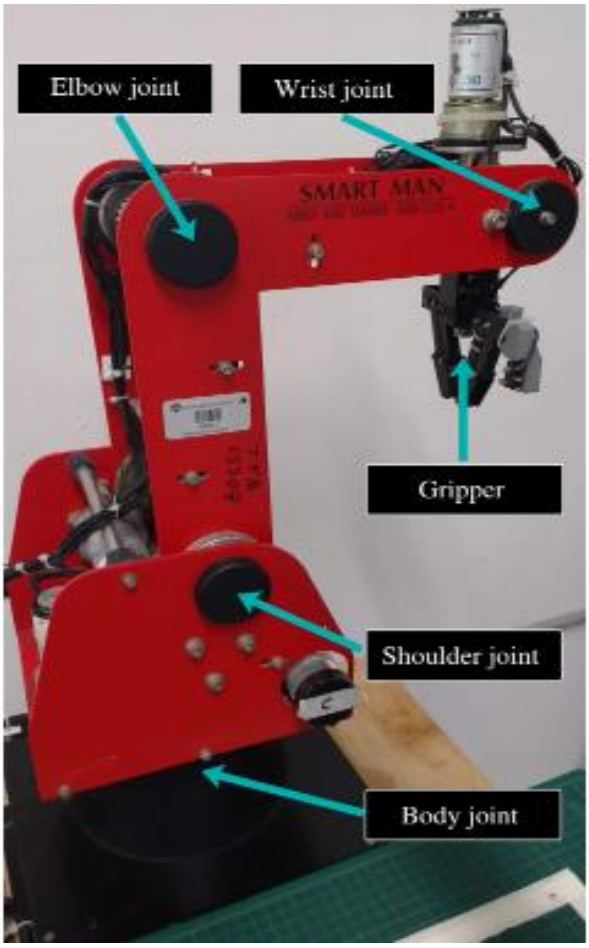


Figure I.6 *: La morphologie géométrique du bras robotique ED 7220C.*

Figure I.7 : *Un servomoteur précis (DME 38B50 G-115).*

Parameters	Specifcations	Units
Links length	Base: 385	mm
	Shoulder: 220	mm
	Elbow: 220	mm
	Wrist: 155	mm
Movement speed	Approx. 100	mm/s max
Range of motion (ROM)	Base: 310	deg
	Shoulder: $+130/-35$	deg
	Eblow: ± 130	deg
	Wrist: 360 rotation, up-down $\pm$ 130	deg
Precision (position)	± 0.5	mm
Weight	33	Kg
Load capacity	1	Kg
Construction	Vertical articulated arm	
Actuator	DC servo motor (optical encoder)	
Number of joints	5 joints + gripper	

Tableau I.1 : *Les principales caractéristiques du bras robotique ED7220C.*

5. Espace de travail

a) Espace de travail accessible

Ensemble de points accessibles par un point sur l'effecteur, généralement le point central de l'outil.

b) Espace de travail adroit

Ensemble de points qu'un point de l'effecteur peut atteindre sans limitation dans son orientation.

L'espace de travail dépend principalement de :

- La géométrie du robot (Figure I.6)
- Les dimensions des liens
- Limitations du mouvement articulaire

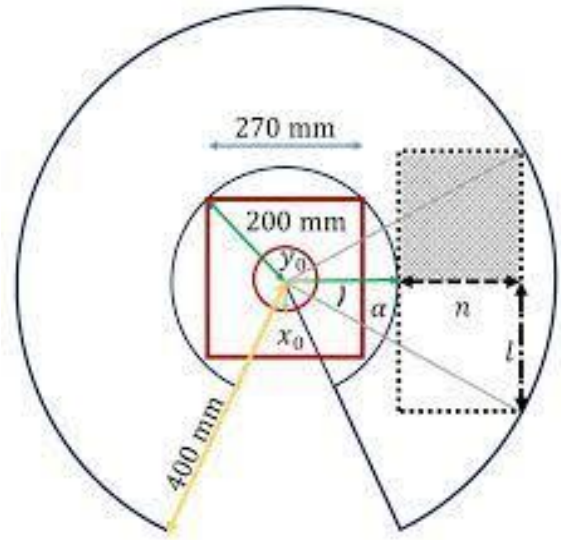


Figure I.8 : *L'espace de travail du bras manipulateur ED 7220C [4]*

6. Le fonctionnement du bras manipulateur ED 7220C

Toutes les articulations ainsi que la pince sont munies de fin de course qui indiquent leurs déplacements minimum et maximum. De plus, ces fins de course permettent d'établir la position initiale du manipulateur. Pour atteindre cette position, le manipulateur dispose également d'un capteur flexible de 11,43 cm [4], ce capteur est situé à l'articulation du coude. Les articulations du corps, de l'épaule et du coude sont couplées à des moteurs à courant continu permanents, modèle DME38B50G-116.

Les articulations sont indiquées comme q1, q2 et q3, respectivement. D'autre part, le poignet et la pince sont, respectivement, entraînés par des moteurs DME38B50-115 et DME33B37G-171 cc. L'articulation du poignet, appelée q4, est actionnée par un mécanisme

de transmission différentielle couplé à deux moteurs à courant continu. Chaque moteur du robot comprend un encodeur optique pour ***déterminer*** sa position.

- ***La carte de commande***

Deux cartes Arduino Mega, contrôlent la position des moteurs. Chaque carte acquiert les données de position de trois moteurs, et il est communiqué avec un ordinateur par une connexion USB (**Figure I.9**). Le signal de commande de chaque moteur est produit par le programme MATLAB. Ce signal est communiqué à la carte Arduino Mega via la boîte à outils Arduino IO, qui convertit le signal de commande en modulation de largeur d'impulsion (PWM).

Les signaux PWM sont transmis aux modules pilotes L298N Dual H-Bridge. Ces modules fournissent l'alimentation aux moteurs cc.

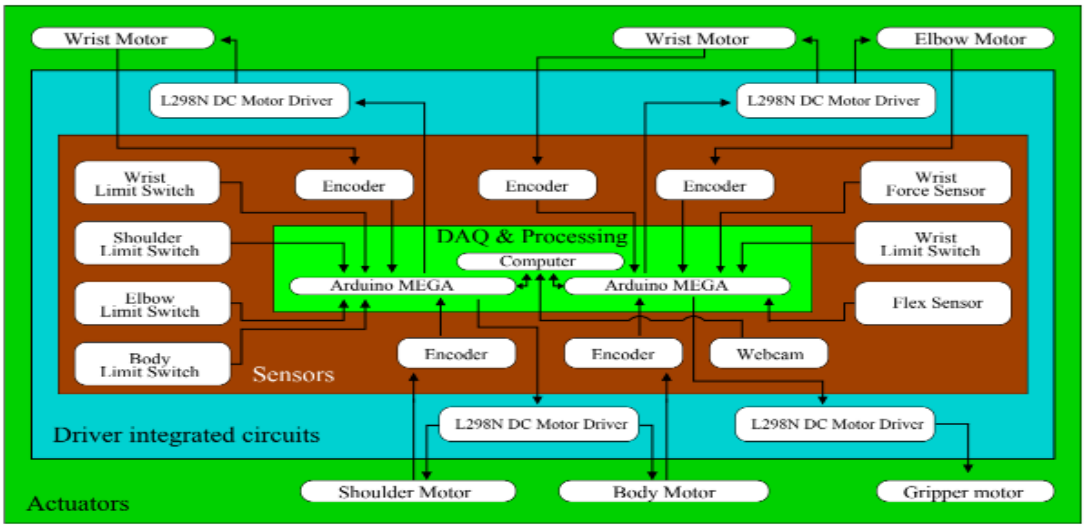


Figure I.9 : *Deux cartes Arduino Mega qui contrôlent les 6 moteurs. [4]*

a) Modélisation [5]

La conception et le contrôle des robots nécessitent le calcul de certains modèles mathématiques, tels que :

- Un modèle de transformation entre l'espace de manipulation (où est définie la situation de l'organe terminal) et l'espace articulaire (où est définie la configuration du robot). Nous distinguons :

- Des modèles géométriques directs et inverses, représentant les conditions d'organes terminaux en fonction de variables articulaires et inversement ;

- Des modèles cinématiques directs et inverses représentant la vitesse des organes terminaux en fonction de la vitesse articulaire et inversement ;

- Un modèle dynamique qui définit les équations de mouvement du robot, qui peut établir la relation entre le couple ou la force appliquée par l'actionneur et la position, la vitesse et l'accélération du robot découpeur.

b) Modélisation géométrique [5]

Modéliser un robot de manière systématique et automatique nécessite une méthode appropriée pour décrire sa morphologie. Plusieurs méthodes et notations ont été proposées.

La méthode la plus courante est Denavit-Hartenberg. Mais cette approche développée pour des structures ouvertes simples est ambiguë lorsqu'elle est appliquée à des robots à structures fermées ou arborescentes. La notation de Khalil et Klein Finger permet une description homogène d'architectures ouvertes simples et complexes de systèmes mécaniques articulés avec un nombre minimal de paramètres.

1- Modèle géométrique direct [6]

Le modèle géométrique direct (MGD) est l'ensemble des relations qui permettent d'exprimer la situation de l'organe terminal, c'est-à-dire les coordonnées opérationnelles du robot, en fonction de ses coordonnées articulaires. Dans le cas d'une chaîne ouverte simple, il peut être représenté par la matrice de transformation :

$$^{0}T_{n} = {}^{0}T_{n}(q_1)\ {}^{1}T_{2}(q_2) \ldots {}^{n-1}T_{n}(q_n)$$

Le modèle géométrique direct du robot peut aussi être représenté par la relation :

$$X = f(q)$$

q étant le vecteur des variables articulaires tel que :

$$q = [q1\ q2 \ldots qn$$

Les coordonnées opérationnelles sont définies par :

$$X = [x1\ x2 \ldots xm$$

2- Modèle géométrique inverse

Le modèle inverse est une technique qui calcule le mouvement requis ou optimal d'un système pour atteindre une destination particulière. En robotique, le modèle inverse peut déterminer comment le bras du robot se déplace afin qu'une pince soit correctement positionnée à l'extrémité du bras. En 3D, une cinétique inverse peut être activée dans l'espace, de sorte que le mouvement de la sous-articulation dans une manipulation à caractère hiérarchique affecte naturellement les objets parents. Dans ce processus, les paramètres de chaque expression, dans un corps élastique connecté (chaîne cinétique), seront

automatiquement calculés pour atteindre le mode souhaité, en particulier lorsque le point final se déplace.

Contrairement au modèle direct, les robots à multiples articulations révolutionnaires ont généralement plusieurs solutions pour la cinématique inverse, et différentes méthodes ont été proposées en fonction de l'objectif. En général, ils sont classés en deux méthodes, l'une est obtenue analytiquement (c'est-à-dire la solution analytique) et l'autre utilise l'arithmétique numérique.

7. Conclusion

Dans ce chapitre nous avons présenté une introduction sur les robots manipulateurs, notamment le robot ED 7220C que nous allons étudier dans ce mémoire.

Une partie de ce chapitre a été consacré à quelques notions générales relatives aux mécanismes des robots manipulateurs et les différents modèles utilisés pour décrire les mouvements des articulations d’un manipulateur et montrer comment calculer ces modélisations (géométrique, cinématique et dynamique) auxquelles ils sont nécessaires pour la commande des robots manipulateurs.

Chapitre II

Langage de programmation des Robots.

1. Introduction

La programmation n'est que le langage de communication entre l'homme et la machine. Ce langage se compose des nombreuses commandes permettant d'effectuer une tâche particulière.

Il existe peut-être autant de langages de robots que de fabricants de robots. Chaque fabricant conçoit son propre langage robotique, et donc, pour utiliser un robot particulier, sa marque de langage de programmation doit être appris.

De nombreux langages de la robotique sont basés sur des langages courants tels que C++ et autres. Ces derniers sont autonomes et ne se rapportent à aucune autre langue commune.

Les langages des robots sont à différents niveaux en fonction de leur conception et de leur application.

2. Les langages basés sur l'interpréteur [8]

Les langages basés sur l'interpréteur exécutent une ligne du programme à la fois. Chaque ligne du programme a une ligne ‘Numéro’. L'interpréteur interprète la ligne à chaque fois qu'elle est rencontrée (il convertit le programme du robot en un programme en langage machine que le processeur peut comprendre et exécuter) et exécute chaque ligne de manière séquentielle. L'exécution continue jusqu'à ce que la dernière ligne soit rencontrée, ou jusqu'à ce qu'une erreur soit détectée, moment auquel l'exécution s'arrête.

L'avantage d'un langage basé sur un interpréteur réside dans sa capacité à poursuivre l'exécution jusqu'à ce qu’une erreur soit détectée, ce qui permet à l'utilisateur d'exécuter et de déboguer le programme portion par portion. Par conséquent, le débogage d'un programme est beaucoup plus rapide et facile. Cependant, comme chaque ligne est interprétée à chaque fois, l'exécution est plus lente et peu efficace. De nombreux langages de robot tels que le V+ d'OMRON Adept sont basés sur ce concept.

3. Les langages basés sur un compilateur [8]

Les langages basés sur un compilateur utilisent un compilateur pour traduire l'ensemble du programme en langage machine (qui crée un code objet) avant son exécution. Comme le processeur exécute le code objet, ces programmes sont beaucoup plus rapides et efficaces. Cependant, comme tout le programme doit d'abord être compilé, il est impossible d’exécuter n'importe quelle partie du programme s'il y a des erreurs de syntaxe

présentes, avant même que la logique du programme ne soit testée. Par conséquent, le débogage d'un programme est plus difficile.

4. Les différents niveaux de langages robotiques

Ci-dessous une description générale des différents niveaux de langages des robots :

a) Niveau langage machine micro-ordinateur :

A ce niveau, les programmes sont écrits en langage machine. Ce niveau de programmation est le plus basique, et est très efficace, mais il est difficile à comprendre et difficile à suivre. Tous les langages seront éventuellement interprétés ou compilés à ce niveau. Cependant, dans le cas de programmes de niveau supérieur, l'utilisateur écrit les programmes dans un niveau supérieur. Langage plus facile à suivre et à comprendre.

b) Niveau point à point :

A ce niveau, les coordonnées des points sont saisies séquentiellement et les robots suivent les points spécifiés. Il s'agit d'un type de programme très primitif, simple et facile à utiliser, mais pas très puissant. Il manque également de ramification, d'informations sensorielles et conditionnelles.

c) Niveau mouvement primitif :

Dans ces langages, il est possible de développer des programmes plus élaborés, y compris les informations sensorielles, les branchements et les déclarations conditionnelles. La plupart des langues à ce niveau sont basées sur l'interprétation.

d) Niveau de programmation structuré :

La plupart des langages à ce niveau sont basés sur un compilateur, sont puissants et permettent une programmation plus sophistiquée. Cependant, ils sont également plus difficiles à apprendre.

e) Niveau axé sur les tâches :

Il n'existe pas encore de langues réelles à ce niveau. AUTOPASS, proposé par IBM dans les années 1980, jamais concrétisé. AUTOPASS était censé être axé sur les tâches, ce qui signifie qu'au lieu de programmer un robot pour effectuer une tâche en programmant chaque étape, l'utilisateur ne mentionnerait que la tâche, s'attendant à ce que le contrôleur crée le nécessaire séquence. Imaginez qu'un robot doit trier trois boîtes par taille. Dans toutes les langues existantes, le programmeur doit spécifier chaque mouvement et chaque étape, y compris comment aller à la plus grande boîte, comment ramasser la boîte, où la placer, où aller pour trouver la boîte suivante, et ainsi de suite, même si un système de vision ou d'autres dispositifs sensoriels sont utilisés. Dans AUTOPASS, l'utilisateur

n'indiquerait que "trier", tandis que le contrôleur du robot créerait cette séquence automatiquement. Cela ne s'est jamais produit.

5. Exemple de langage de robots *[9]*

a) Assembleur:

Le langage de programmation permet de programmer au "niveau zéro", c'est-à-dire au niveau de programmation le plus bas. Dans un passé récent, la plupart des sciences physiques de bas niveau nécessitaient une programmation en assemblage.

b) MATLAB « MATrix LABoratory » :

MATLAB et ses parents open source, comme Octave, sont très branchés avec certains ingénieurs en robotique pour l'analyse des connaissances et le développement de systèmes de gestion. Il y a aussi la mallette à outils AI très branchée pour MATLAB.

c) C/C++ :

C’est le langage de programmation le plus utilisé en robotique, puisque, il permet une interaction facile avec le bas niveau matériel.

6. Plateforme de développement Arduino *[10]*

Arduino est un ensemble matériel et logiciel en source libre qui permet d'apprendre l'électronique tout en se familiarisant avec la programmation informatique.

Les Arduino sont des cartes électroniques programmables sur lesquelles nous pouvons brancher des capteurs de température, d'humidité, de vibration ou de lumière, une caméra, des boutons, des potentiomètres de réglage, des contacts électriques…etc. Il y a aussi des connecteurs pour brancher des LED, des moteurs, des relais, des afficheurs, un écran…

Une carte Arduino est un cerveau qui permet de rendre intelligents des systèmes électroniques et d'animer des dispositifs mécaniques.

Figure II.1 : *Une carte Arduino Uno avec ses connecteurs.*

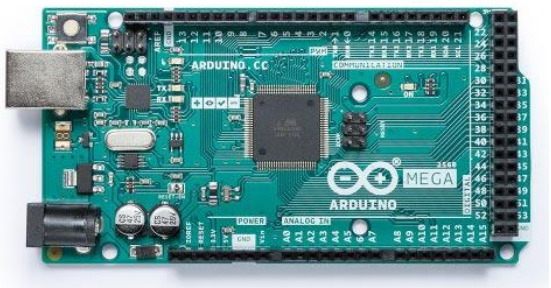


Figure II.2 : *Une carte Arduino MEGA avec ses connecteurs.*

a) Le logiciel IDE d'Arduino

Les créateurs d'Arduino ont développé un logiciel pour que la programmation des cartes Arduino soit visuelle, simple et complète à la fois. C'est ce que l'on appelle une IDE, qui signifie « Integrated Development Environment » ou Environnement de Développement « Intégré » en français (donc EDI).

L'IDE affiche une fenêtre graphique qui contient un éditeur de texte et tous les outils nécessaires à l'activité de programmation. Vous pouvez donc saisir votre programme, l'enregistrer, le compiler, le vérifier, le transférer sur une carte Arduino…

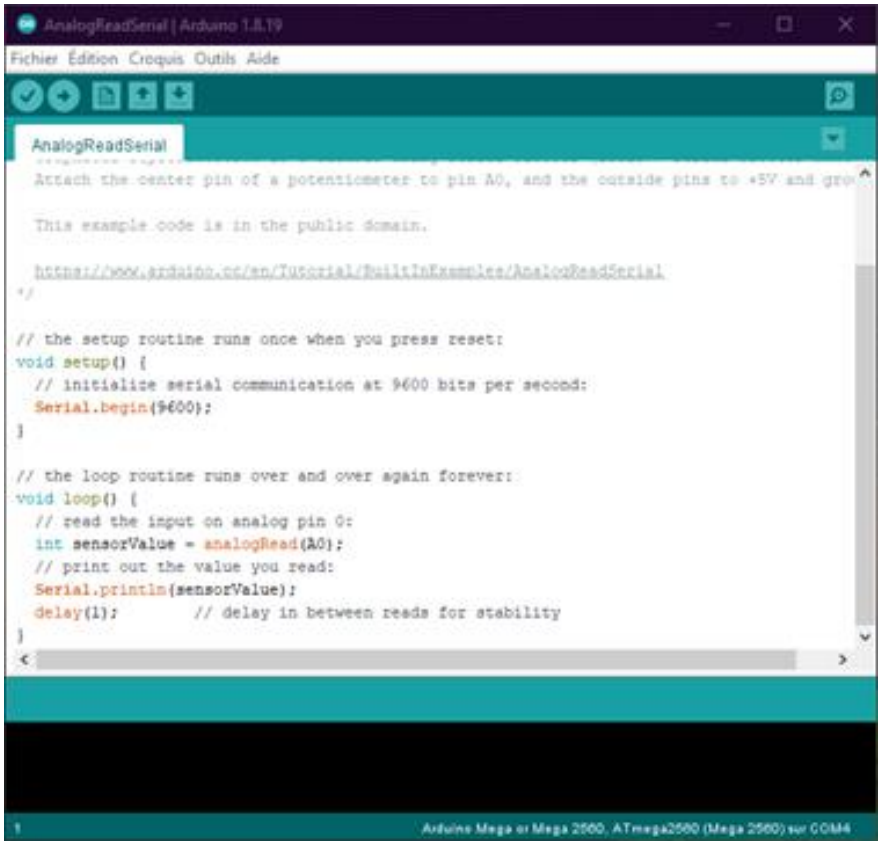


Figure II.3 : *L'écran principal de l'IDE Arduino au démarrage.*

b) Réaliser des projets avec Arduino

En plus de la facilité de programmation, l'autre grande caractéristique d'un Arduino est la capacité du microcontrôleur sur lequel elle est basée.

Avec quelques Shields supplémentaires facilement disponibles, un large choix de modules capteurs économiques et d'actionneurs, il n'y a vraiment pas beaucoup de choses que vous ne puissiez faire avec un Arduino.

Voici quelques applications possibles pour une Arduino :

- **Mesure et détection :**
  - Station météorologique automatisée,
  - Détecteur de foudre,
  - Suivi du soleil pour orientation des panneaux solaires,
  - Moniteur de radiation,
  - Détecteur automatique de la faune,
  - Système de sécurité domestique ou professionnel.
- **Contrôle :**
  - Petits robots,
  - Maquette de fusée ou d'avion,
  - Drones multirotor,
  - CNC simple pour petites machines-outils.
- **Automatisation :**
  - Serre automatisée,
  - Aquarium automatisé,
  - Robot navette d'échantillon de laboratoire,
  - Chambre thermique de précision (couveuse, yaourtière, étuve, séchoir…),
  - Système de test électronique automatisé.

c) *Exemple d'un montage avec Arduino*

La figure 4, montre un exemple d'une application de la carte Arduino

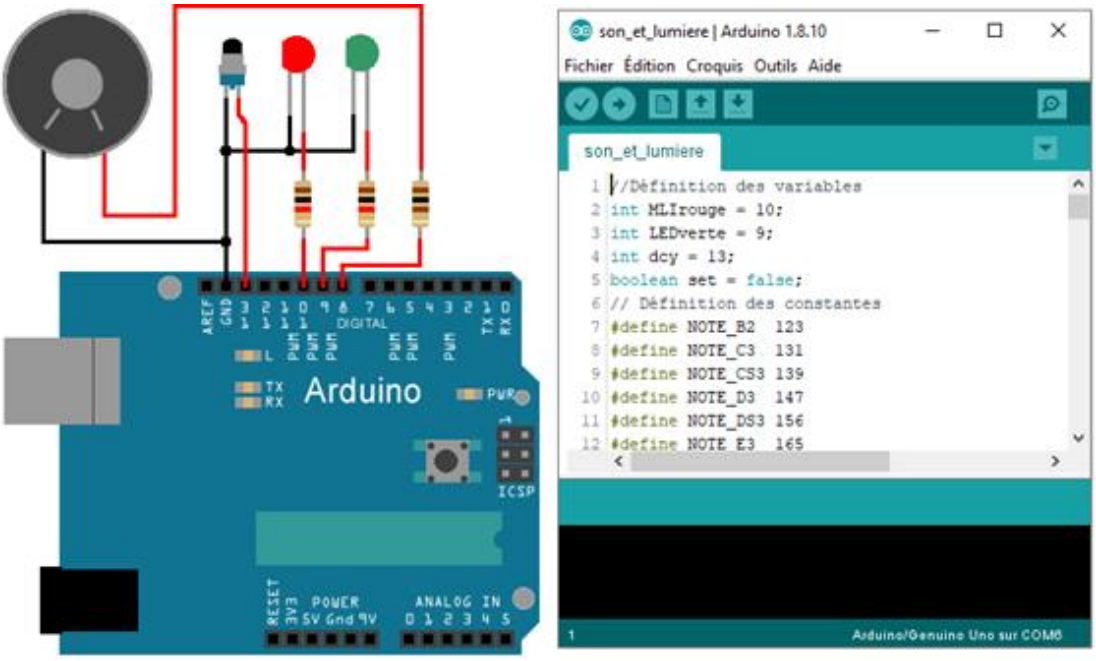


Figure II.4 : *Un montage câblé avec une carte Arduino et son logiciel de programmation.*

7. MATLAB [11]

MATLAB est **une** plate-forme de programmation et de calcul numérique utilisée par des millions d'ingénieurs et de scientifiques pour analyser des données, développer des algorithmes et créer des modèles.

Des millions d'ingénieurs et de scientifiques du monde entier utilisent MATLAB® pour analyser et concevoir les systèmes et les produits qui transforment notre monde. Le langage MATLAB basé sur des matrices est le moyen le plus naturel au monde d'exprimer des mathématiques computationnelles. Les graphiques intégrés facilitent la visualisation et l'obtention d'informations à partir des données. Le code MATLAB peut être intégré à d'autres langages, ce qui permet de déployer des algorithmes et des applications dans des systèmes Web, d'entreprise et de production.

Avec MATLAB, vous pouvez développer des algorithmes beaucoup plus rapidement que dans les langages traditionnels, tels que C, C++ ou Fortran, sans avoir à déclarer de variables, allouer de la mémoire ou compiler du code.

8. Interfaçage de MATLAB et ARDUINO :

L'idée est d'utiliser les fonctions au sein du langage Arduino pour envoyer et acquérir des informations binaires via le port (USB). Ainsi, sous Simulink, nous pouvons développer des programmes pour visualiser, stocker et/ou traiter ces informations.

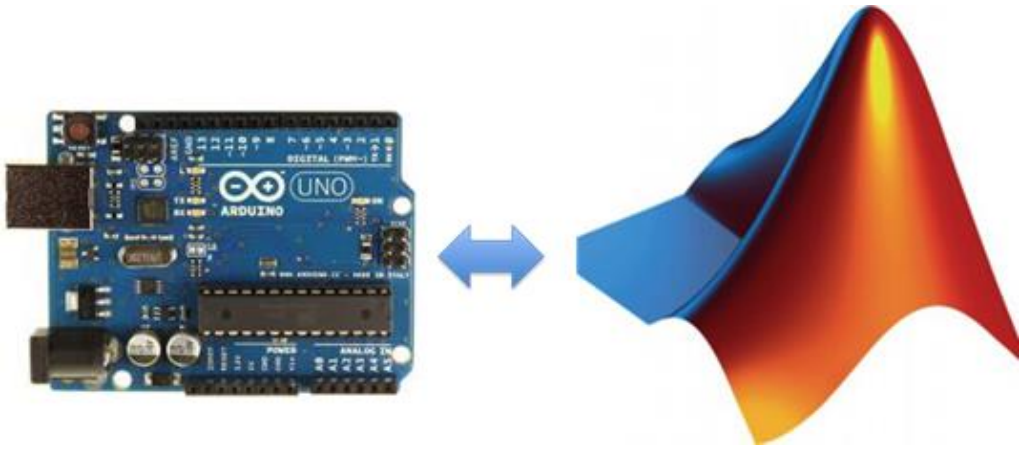


Figure II.5 : *Interfaçage de MATLAB et ARDUINO.*

MATLAB et Simulink relèvent plusieurs défis avec la programmation Arduino traditionnelle. Les produits prennent en charge deux workflows principaux :

- Lire, écrire et analyser les données des capteurs Arduino.
- Développer des algorithmes qui s'exécutent de manière autonome sur l'appareil Arduino.

a) Lire, écrire et analyser les données des capteurs ARDUINO

« MATLAB support package for Arduino » permet d'écrire des programmes MATLAB qui lisent et écrivent des données sur la carte Arduino et permet d'accéder à des appareils connectés tels que des moteurs, des LED et des appareils I2C. Parce que MATLAB est un langage interprété de haut niveau, le prototypage et l'affinement des algorithmes sont faciles. MATLAB comprend des milliers de fonctions mathématiques, d'ingénierie et de traçage.

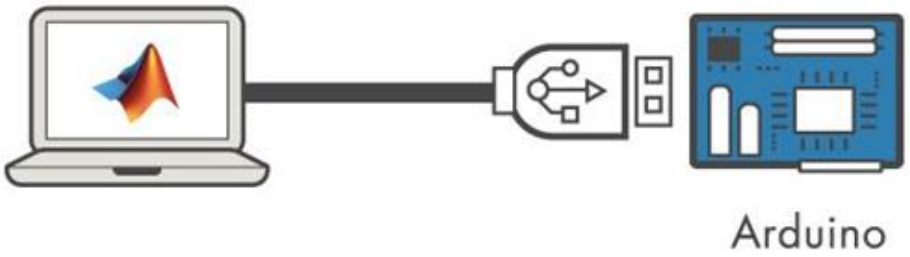


Figure II.6 : *Connection d'Arduino avec un ordinateur exécutant MATLAB.*

b) Avantages de l'utilisation de MATLAB pour la programmation Arduino

- Lisez et écrivez les données des capteurs de manière interactive sans attendre la compilation de votre code.
- Développez des algorithmes et analysez les données des capteurs à l'aide de milliers de fonctions prédéfinies pour le traitement du signal, l'apprentissage automatique, la modélisation mathématique, etc.
- Visualisez rapidement vos données à l'aide de la vaste gamme de types de tracés dans MATLAB.

c) Développer des algorithmes qui s'exécutent de manière autonome sur l'Arduino

« Simulink support package for Arduino » vous permet de développer des algorithmes dans Simulink, un environnement de diagramme de blocs pour la modélisation des systèmes dynamiques et le développement d'algorithmes, et de les exécuter de manière autonome sur la carte Arduino. Le package de support étend Simulink avec des blocs pour configurer et accéder aux capteurs, actionneurs et interfaces de communication Arduino. Après avoir créé votre modèle Simulink, vous pouvez le simuler, régler les paramètres de l'algorithme jusqu'à ce que vous l'obteniez parfaitement et télécharger l'algorithme terminé pour une exécution autonome sur l'appareil. Avec le bloc MATLAB Function, vous pouvez incorporer du code MATLAB dans votre modèle Simulink.

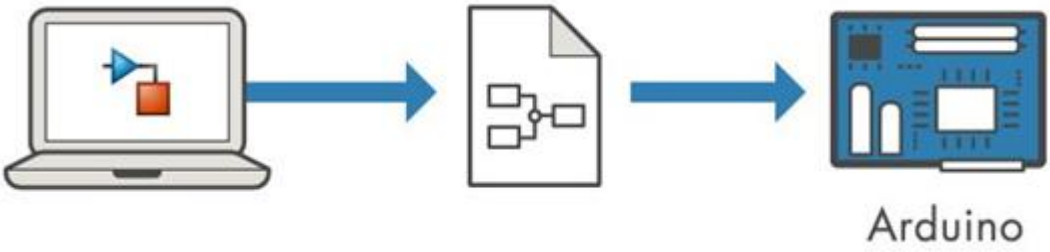


Figure II.7 : *Connection d'Arduino avec un ordinateur exécutant MATLAB/Simulink.*

d) Avantages de l'utilisation de Simulink pour la programmation Arduino :

- Développez et simulez vos algorithmes dans Simulink et utilisez la génération automatique de code pour les exécuter sur l'appareil.
- Incorporez le traitement du signal, la conception des commandes, la logique d'état et d'autres routines mathématiques et d'ingénierie avancées dans vos projets matériels.
- Ajustez et optimisez de manière interactive les paramètres dans Simulink lorsque votre algorithme s'exécute sur votre Arduino.
- Modifiez facilement les algorithmes pour qu'ils s'exécutent sur d'autres plates-formes matérielles à faible coût et commerciales.

9. *Programmation Arduino [12]*

La carte Arduino est un microcontrôleur, c'est-à-dire une sorte de mini-ordinateur qui sert d'interface entre l'environnement (actions, mesures de grandeurs...) et un utilisateur. Elle se programme nativement dans un langage dérivé du C : le langage « Arduino »

Le logiciel utilisé : Arduino est le Logiciel permettant d'installer les drivers des cartes Arduino, de la programmer en langage C. Il est téléchargeable librement sur le site officiel.

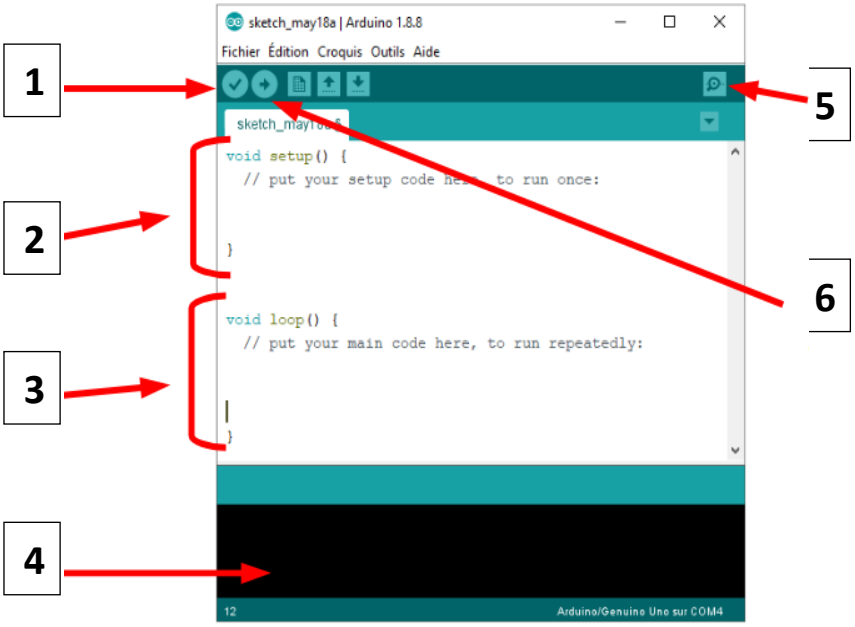


Figure II.8 : *Description de la page de démarrage pour le logiciel IDE de l'Arduino.*

1) Compile le programme : c'est-à-dire vérifie s'il n'y a pas d'erreur. Peut être utilisé, même si aucune carte n'est connectée, pour corriger la syntaxe d'un programme.
2) Cette boucle est exécutée une fois à l'initialisation du programme et quand la bouton reset de la carte est pressé.
3) La partie de programme se trouvant se répètent indéfiniment tant que la carte est alimentée.
4) Les messages d'erreur s'afficheront ici.
5) Lance le moniteur série qui affiche les valeurs et mesures renvoyée par la carte par la fonction :
   - `Serial.print()`
   - `Serial.println()` (pour sauter des lignes).
6) Téléverse le programme sur la carte Arduino, pour qu'il puisse tourner. Dès qu'une modification est faite dans le programme, il faut téléverser le programme de nouveau pour qu'elle soit prise en compte.

a) Exemple de la lecture de la position d'un moteur par interruption [13]

Un capteur qui renvoie un signal digital se branche de préférence sur les entrées digitales du microcontrôleur. L'encodeur rotatif possède 3 broches de sortie : une pour le bouton poussoir (SW) et deux pour la direction (DT) et la détection de rotation (CLK).

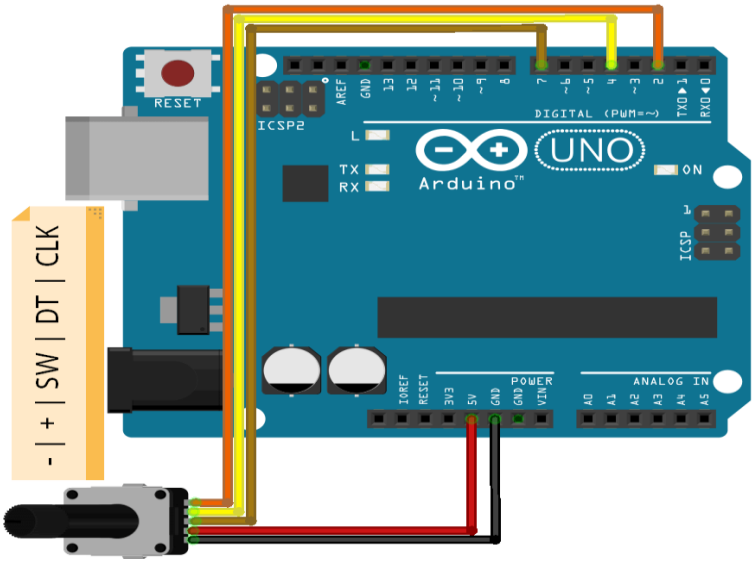


Figure II.9 : *Le câblage d'un encodeur rotatif à carte Arduino pour effectuer la lecture de la position d'un moteur.*

Pour récupérer les informations de l'encodeur rotatif, il va nous falloir repérer les changements d'état de chaque signal de sortie du capteur et définir les variables et leurs modifications en fonction de ces signaux. Plus précisément, l'algorithme détecte le passage de la broche de l'état HAUT à BAS. Ensuite, nous détectons le sens de rotation en testant si la broche DT est à l'état HAUT ou BAS, si le sens de rotation est antihoraire la variable rotVal est décrémentée sinon elle est incrémentée.

❖ **Code Arduino :**

```
//Parameters
const int clkPin  = 2;
const int dtPin  = 4;
const int swPin  = 7;
//Variables
int rotVal  = 0;
bool clkState  = LOW;
bool clkLast  = HIGH;
bool swState  = HIGH;
bool swLast  = HIGH;
void setup() {
 //Init Serial USB
 Serial.begin(9600);
 Serial.println(F("Initialize System"));
 //Init Rotary
 pinMode(clkPin,INPUT);
 pinMode(dtPin,INPUT);
 pinMode(swPin,INPUT_PULLUP);
}
void loop() {
 readRotary();
}
void readRotary( ) { /* function readRotary */
 ////Test routine for Rotary
 // gestion position
 clkState = digitalRead(clkPin);
 if ((clkLast == LOW) && (clkState == HIGH)) {//rotary moving
  Serial.print("Rotary position ");
  if (digitalRead(dtPin) == HIGH) {
   rotVal = rotVal - 1;
   if ( rotVal < 0 ) {
    rotVal = 0;
   }
```

```
    }
    else {
      rotVal++;
      if ( rotVal > 10 ) {
        rotVal = 10;
      }
    }
    Serial.println(rotVal);
    delay(200);
  }
  clkLast = clkState;
  //gestion bouton
  swState = digitalRead(swPin);
  if (swState == LOW && swLast == HIGH) {
    Serial.println("Rotary pressed");
    delay(100);//debounce
  }
  swLast = swState;
}
```

❖ **Résultat :**

On observe que la variable rotVal évolue en fonction de la rotation de l'encodeur et on détecte correctement la position sur l'axe de l'encodeur.

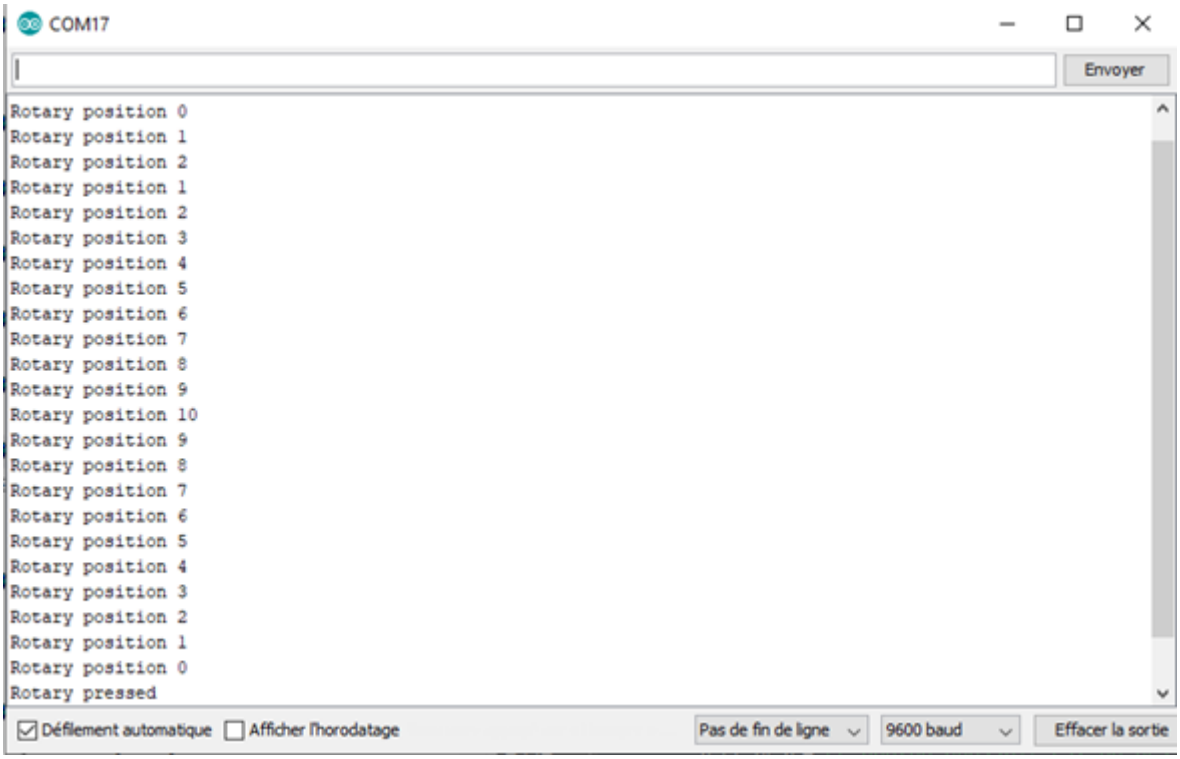


Figure II.10 : *Résultats d'interprétation donné simultanément à la rotation de l'encodeur rotatif*

10. Conclusion

Ce chapitre est consacré aux notions générales sur les langages de programmation des robots, une partie de ce chapitre a été destiné à description des deux logiciels (Matlab et Arduino) et aux plateformes de programmation que nous allons utiliser dans la réalisation du langage de programmation du robot manipulateur ED-7220C.

Chapitre III

Programmation du robot pédagogique ED-7220C sous MATLAB.

I. Introduction

Dans ce chapitre, nous allons présenter la partie pratique de notre projet de fin d'études qui consiste en la création d'un langage de programmation sous MATLAB dédié au robot pédagogique ED-7220C.

Ce langage de programmation une fois créé va nous permettre de réhabiliter le robot et le rendre opérationnel pour réaliser les travaux pratique ; cette réhabilitation concerne la partie software du robot. Comme il a été mentionné dans l'introduction de ce mémoire, ce projet est la suite d'un projet de fin d'études dont l'objectif été la réhabilitation de la partie Hardware du robot.

Donc, nous allons exploiter la carte de commande basée sur deux cartes Arduino Méga[1] pour créer notre langage de programmation. Mais avant de penser à la création du jeu d'instruction de notre langage de programmation, il faut tout d'abord assurer l'interfaçage matériel/logiciel entre la carte de commande du robot et le logiciel de programmation MATLAB. Pour cela, nous proposons une démarche en deux étapes.

Dans la première étape, nous allons intégrer la carte de commande dans l'environnement MATLAB. Dans la deuxième étape, nous allons proposer un jeu d'instructions d'un langage dédié à ce robot, afin de créer des tâches robotisées.

Ce chapitre comporte deux parties, la première partie consiste en la description de la configuration hardware et software de l'interface de communication (de programmation) entre l'environnement MATLAB et la carte de commande du robot ainsi que du robot.

La deuxième partie est la partie la plus importante de notre travail, elle est consacrée au développement d'un jeu d'instructions dédié à la programmation du robot pédagogique ED-7220C.

II. Partie I : Description de l'environnement de programmation

Dans cette partie nous allons décrire la structure mécanique articulé du robot, la carte de commande ainsi que les étapes d'installation package MATLAB support package for Arduino.

1. Robot Manipulateur ED-7220C :

Le robot est composé d'un bras robotisé de 6 degrés de liberté. La **Figure III.1** représente ses articulations, qui sont situées sur le corps, l'épaule, le coude et le poignet.

Le bras manipulateur dispose également d'une pince pour collecter des objets. Toutes les articulations ainsi que la pince sont équipées de fin de course, qui indique leurs

déplacements minimum et maximum. De plus, ces fins de course permettent d'établir la position initiale du manipulateur. Les articulations du corps, l'épaule, les coudes, le poignet et la pince sont couplées à des moteurs à courant continu. Chaque moteur du robot comprend un encodeur optique pour déterminer sa position.

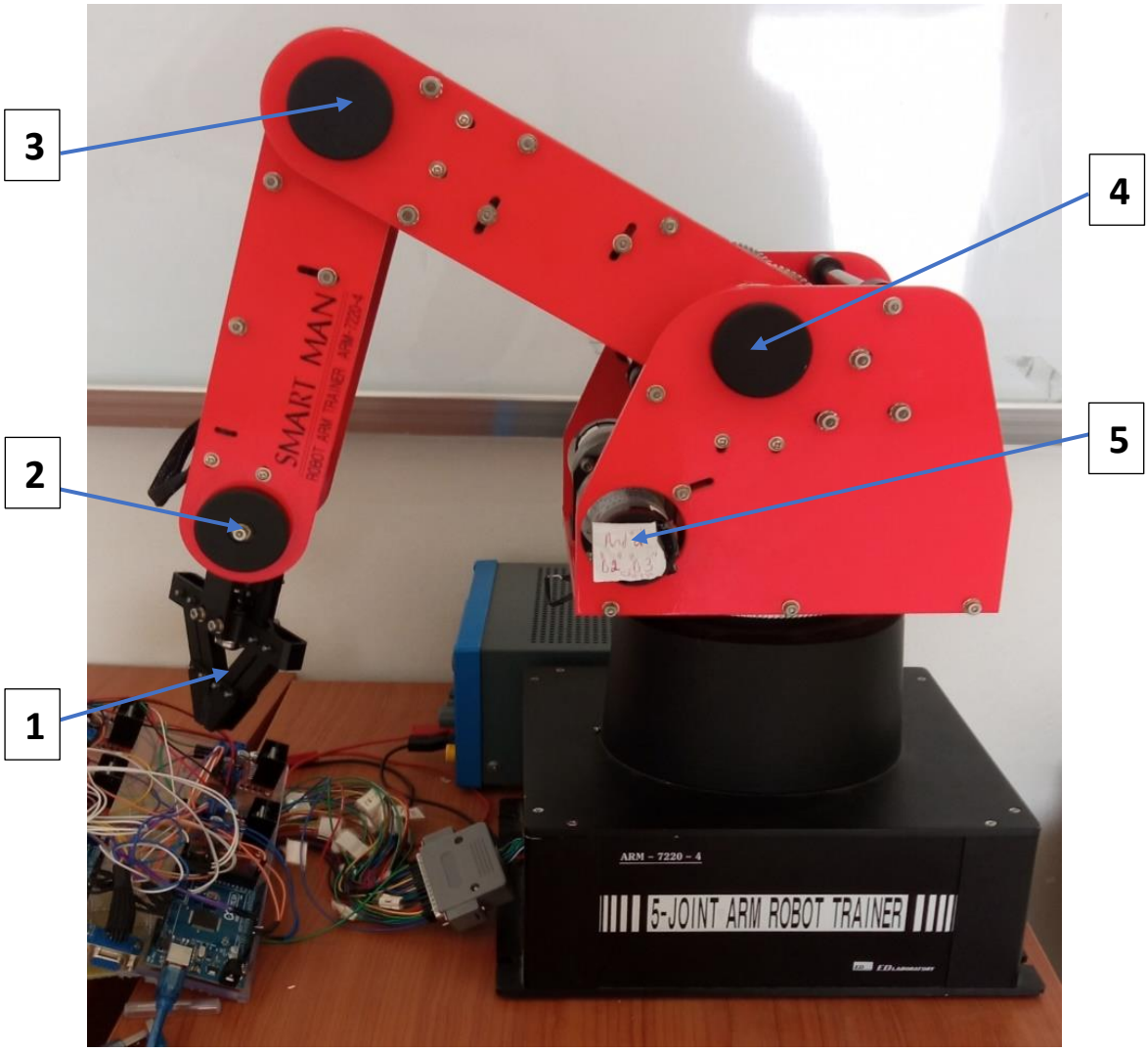


Figure III.1 : *Les articulations du robot ED-7200C.*

- **Terminologie français/anglais**

1) La pince « gripper ».
2) Le poignet « wrist ».
3) Le coude « elbow ».
4) L'épaule « shoulder ».
5) Le corps ou la base « body or base ».

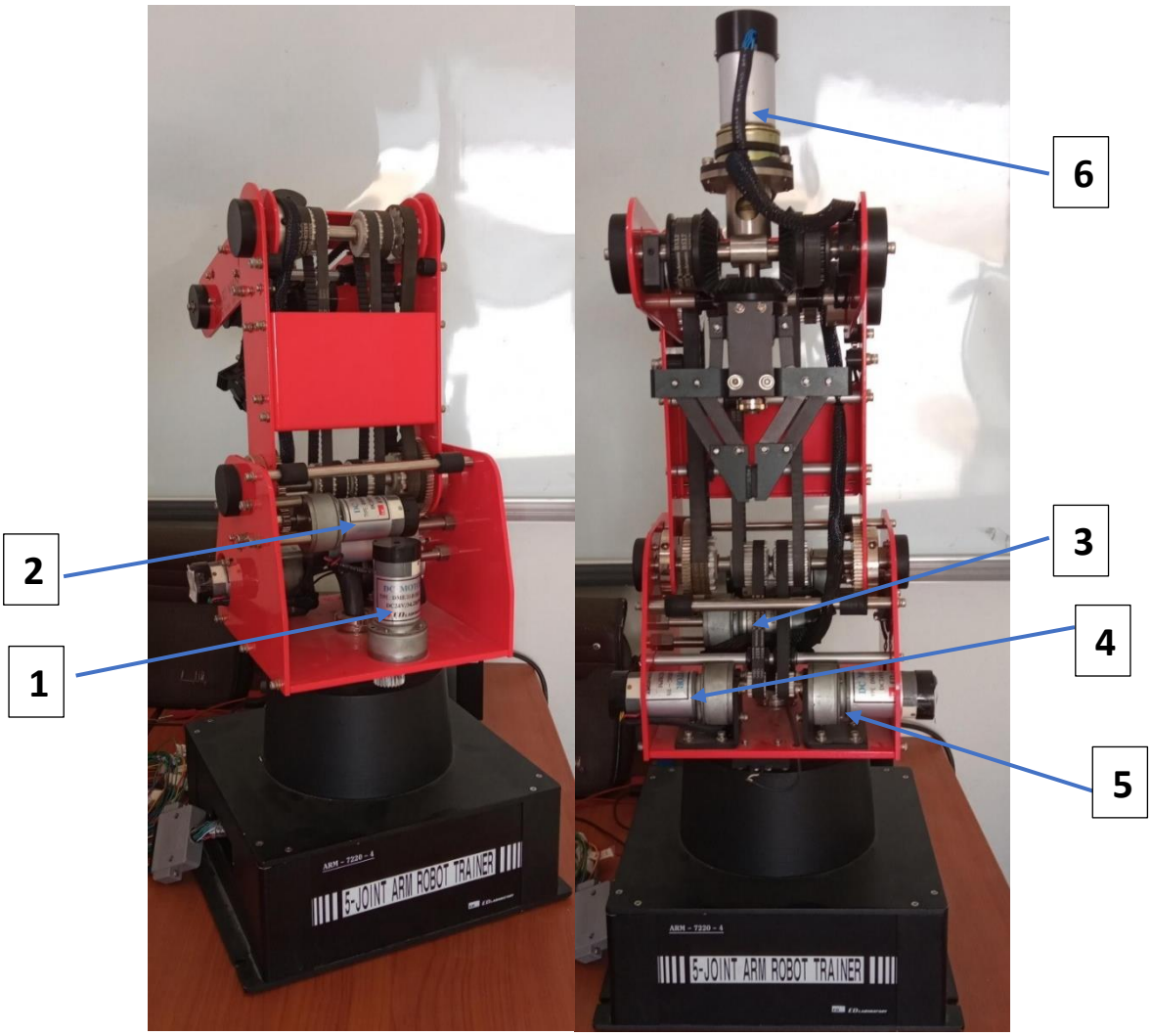


Figure III.2 : *Les moteurs du robot ED-7200C.*

- **Les moteurs du bras :**
  1) Le moteur de la base.
  2) Le moteur de l'épaule.
  3) Le moteur du coude.
  4) Le moteur du poignet droit.
  5) Le moteur de poignet gauche.
  6) Le moteur de la pince.

- **Plage de mouvement :**
  - Articulation du corps : 310°.
  - Articulation de l'épaule : +130°/-35°.
  - Articulation du coude : 130°.
  - Rotation du poignet : 360°.

- Ouverture de la pince 55 mm (sans tampon en caoutchouc : 68 mm).
- Dimension de la base : 220 x 180 (H) mm.
- Poids total : 33kg.

2. Arduino Mega2560 :

La carte Arduino Mega 2560 est basée sur un microcontrôleur ATMega2560 cadencé à 16 MHz. Elle dispose de 54 E/S dont 14 PWM, 16 analogiques et 4 UARTs. Elle est idéale pour des applications exigeant des caractéristiques plus complètes que la carte Uno.

Des connecteurs situés sur les bords extérieurs du circuit imprimé permettent d'enficher une série de modules complémentaires.

Le contrôleur ATMega2560 contient un bootloader qui permet de modifier le programme sans passer par un programmateur.

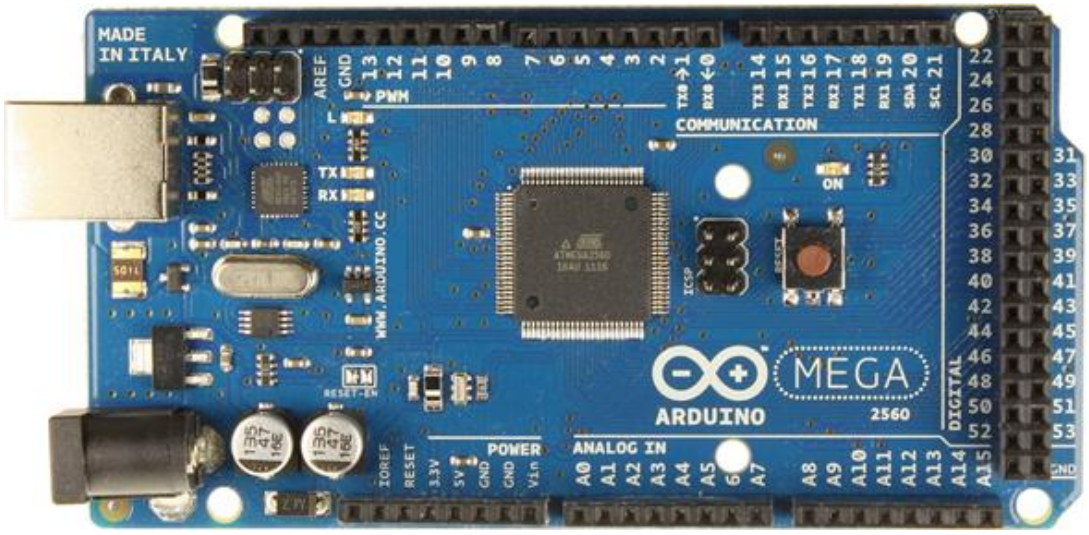


Figure III.3 *: La carte utilisée « Arduino Mega2560 ».*

➢ **Caractéristiques :**

Microcontrôleur	ATMega2560
Tension de fonctionnement	5V
Tension d'entrée (recommandée)	7-12V
Tension d'entrée (limite)	6-20V
Broches d'E/S numériques	54 (dont 15 fournissent une sortie PWM)
Broches d'entrée analogique	16
Courant CC par broche d'E/S	20 mA
Courant continu pour broche 3,3 V	50 mA

Mémoire flash	256 Ko dont 8 Ko sont utilisés par le bootloader
SRAM	8 KB
EEPROM	4 KB
Vitesse d'horloge	16 MHz
LED_BUILTIN	13
Longueur	101.52 mm
Largeur	53.3 mm
Poids	37

Tableau III.1 : *Les caractéristiques d'une carte Arduino Mega 2560.*

3. *Les étapes d'installation du 'MATLAB support package for Arduino '*

Pour installer le package, nous devons d'abord le télécharger à partir le site internet https://www.mathworks.com/. La version du MATLAB doit être R2014a ou ultérieure afin de pouvoir se connecter à la carte Arduino.

- **Installation de package :**

a) Tout d'abord, il faut démarrez MATLAB et cliquez sur le menu déroulant Add-Ons. Cela démarrera la fenêtre du programme d'installation du package. Comme montré sur la ***Figure III.4 :***

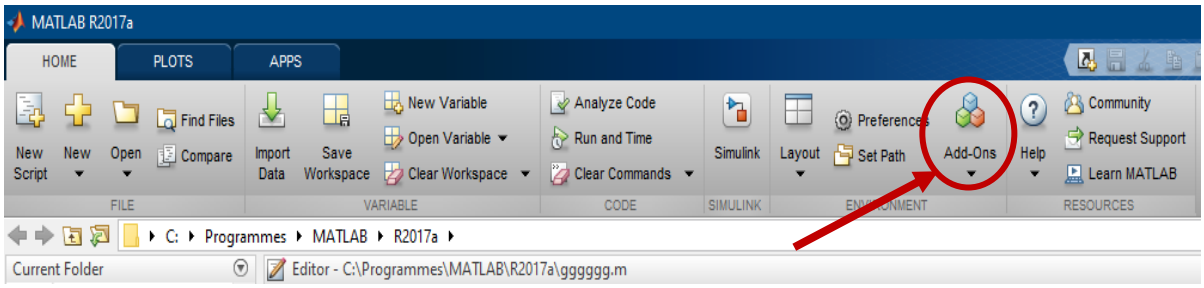


Figure III.4 : *Première étape de l'installation du package.*

b) Faire une rechercher sur la zone de texte, le terme « Matlab Support Package for Arduino Hardware ». Cela vous mènera directement à la page où se trouve le support package. Cliquer sur le premier lien comme suit :

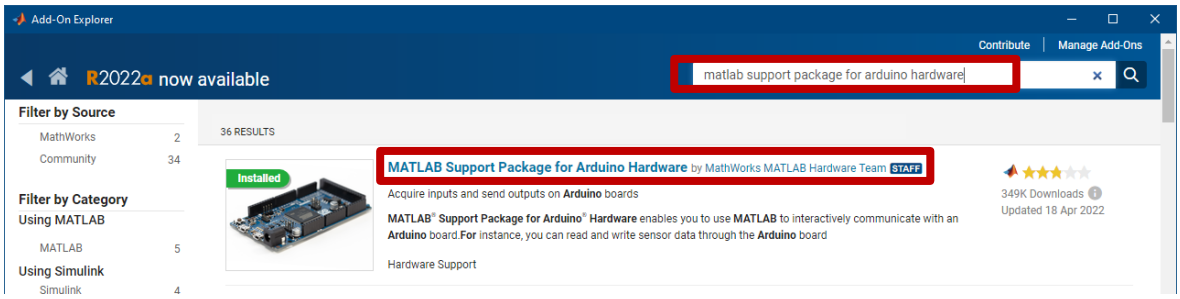


Figure III.5 : *Deuxième étape de l'installation du package.*

c) Dans la fenêtre suivante, vous verrez tous les packages disponibles. Sélectionner le package Arduino, puis cochez tous les packages affichés et cliquez sur Suivant pour poursuivre l'installation.

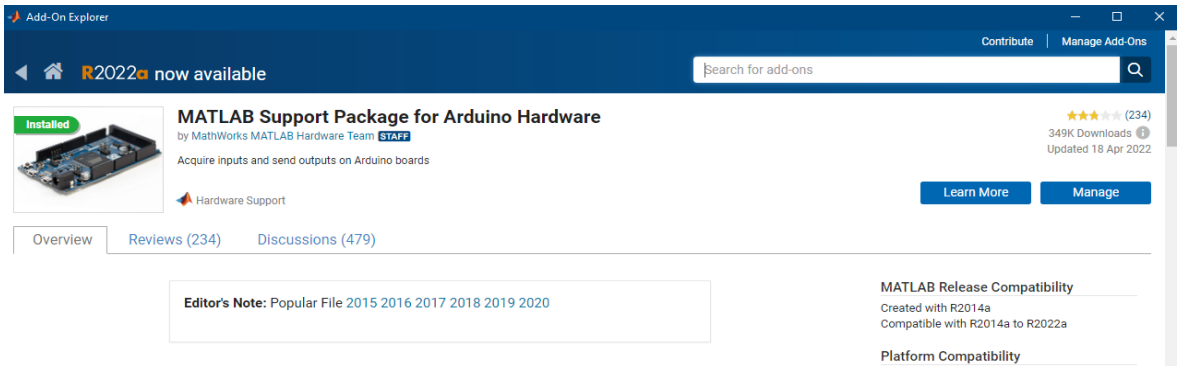


Figure III.6 : *Troisième étape de l'installation du package.*

d) Ensuite, le programme d'installation vous demandera de vous connecter à votre compte Math Works.

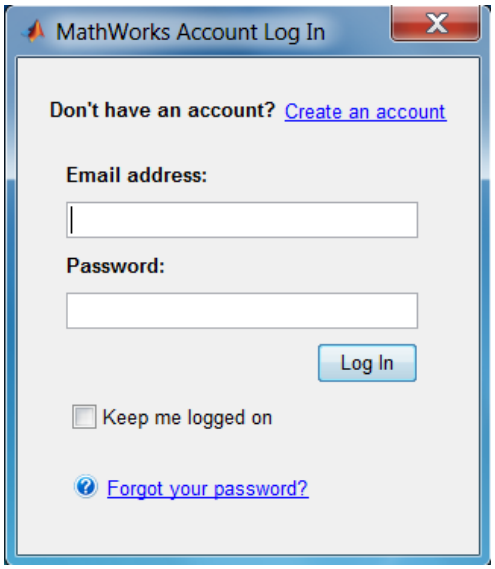


Figure III.7 : *Quatrième étape de l'installation du package.*

e) Acceptez le contrat de licence sur l'écran suivant et continuez à télécharger les packages. Vous devez maintenant attendre que MATLAB télécharge et installe tous les packages requis.

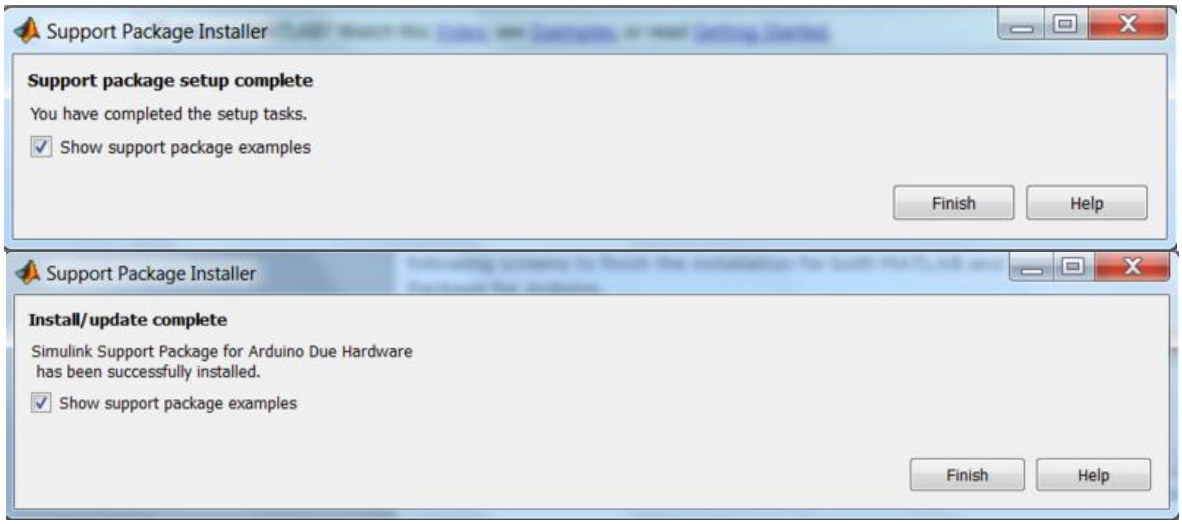


Figure III.8 : *Dernière étape de l'installation du package.*

Si l'installation est achevée sans aucun problème, vous pouvez commander la carte Arduino avec Matlab.

f) Test de la communication entre l'ordinateur hôte et la carte Arduino

Une fois les packages installés, connectez la carte Arduino à l'ordinateur et tapez la commande suivante dans la fenêtre de commande MATLAB :

```
>> a = arduino ()
```

MATLAB tentera alors de communiquer avec la carte. En cas de succès, MATLAB affichera les propriétés de la carte Arduino connectée à l'ordinateur comme indiqué ci-dessous :

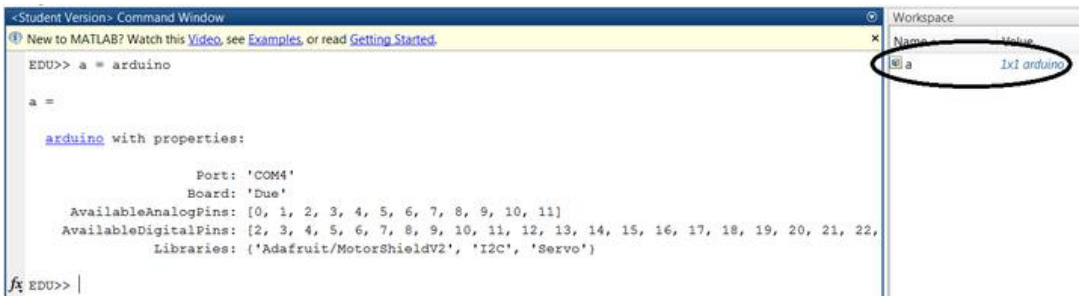


Figure III.9 : *Connexion établit avec succès.*

Ces informations affichent le port sur lequel votre carte est connectée, le modèle de la carte Arduino, ainsi que les broches et bibliothèques disponibles pour la carte.

Dans votre espace de travail, vous verrez une variable **a**, qui est l'objet que MATLAB a créer pour la carte Arduino connectée.

III. Partie II : Le jeu d'instruction du langage de programmation développée.

Dans cette partie de notre mémoire, nous allons décrire le jeu d'instruction que nous avons développé au cours de ce projet de fin d'études. Ce jeu d'instruction sera exécuté dans l'environnement MATLAB et il va nous permettre de commande le robot et de créer des tâches robotisées.

Mais avant de passer à la description de notre jeu d'instructions nous allons tous d'abord décrire les commandes fournis par **MATLAB support package for Arduino.**

1. Les commandes du package MATLAB support package for Arduino

Le package '**MATLAB support package for Arduino'** assure l'exploitation des ressources matériel et logiciel de la carte Arduino, pour cela nous allons exploiter les commandes fournies par ce package pour créer notre propre langage de programmation.

Dans ce qui suit nous allons décrire les commandes de ce package que nous allons utiliser.

i. `a = arduino (port, board)`

Cette instruction permet d'établir la connexion entre la carte de commande du robot et le port USB du de l'ordinateur hôte.

Si aucune connexion ne réussit ou si cette connexion échoue, il établit une connexion à l'une des appareils Arduino connectés à l'ordinateur hôte via USB en utilisant l'instruction `'a = arduino ()'`.

- **Exemple :** `arduino ('COM4' , 'Mega2560')`

Cette instruction permet d'établir la connexion entre l'ordinateur et la carte Arduino Mega2560 via le port **COM4** de cette dernière.

ii. `writePWMVoltage (a, pin, voltage)`

Cette instruction permet d'écrire la valeur de tension spécifiée sur la broche PWM des appareils Arduino. Ainsi, nous pouvons contrôler la valeur de la tension d'entrée de l'Arduino. Cette commande à 3 arguments d'entrée **`(a, pin, voltage)`**, où :

- **a** est la connexion matérielle créée à l'aide Arduino, spécifiée en tant qu'objet.
- **pin** est le numéro de broche numérique sur la carte Arduino.
- **voltage** est la tension des broches numériques PWM spécifiée sous forme de nombre entre 0 et 5 volts. Arduino méga accepte 0 – 5 V.

▪ **Exemple :** `writePWMVoltage (a, 'D11' , 5)`

Nous avons spécifié la broche numérique 11 sur l'Arduino pour avoir 5 volts.

iii. `WriteDigitalPin (a, pin, value)`

Cette instruction permet de d'écrire la valeur spécifiée sur la broche spécifiée sur le périphérique Arduino avec la connexion **a**, où :

- **a** est la **c**onnexion matérielle Arduino créée, spécifiée en tant qu'objet.
- **pin** est le numéro de broche numérique sur le Arduino.
- **value** est la valeur des données numériques à écrire sur la broche spécifiée sur la carte Arduino, spécifiée sous la forme d'une valeur logique de 0 et 1 ou vrai et faux.

▪ **Exemple :** `WriteDigitalPin (a, 'D10', 1)`

iv. `writePWMDutyCycle (a, pin, velo)`

Cette instruction permet de générer un signal PWM avec un rapport cyclique défini sur la broche numérique a de l'Arduino. L'intervalle de pwm est compris entre 0 et 255, et dans cette instruction, elle est spécifiée entre 0 et 1.

En utilisant cette instruction, nous pouvons contrôler la vitesse du moteur DC.

- **velo** est la valeur du rapport cyclique PWM de la broche numérique spécifiée sous la forme d'un nombre compris entre 0 et 1. Dans ce cas, la valeur entre 0 et 1 signifie la vitesse du moteur DC, qui est le rapport entre 0% et 100%.

▪ **Exemple :** `WritePWMDutyCycle (a, 'D10', 0.33)`

0,33 représente 33% de la vitesse maximale du moteur DC.

Étant donné une valeur de 1 signifie 100% de la vitesse du moteur.

v. `Parfor`

Cette instruction permet d'exécuter une série d'instructions MATLAB pour les valeurs `'loopvar'` entre `'initval'` et `'endval'`. La boucle s'exécute en parallèle lorsque on dispose du toolbox '**Parallel Computing Toolbox**'.

Les boucles sont exécutées en parallèle dans un ordre non déterministe.

Nous avons exploité ces instructions pour faire fonctionner deux moteurs simultanément,

vi. `rotaryEncoder (b, chA, chB)`

Cette instruction permet de créer une connexion à l'encodeur rotatif à l'aide de la carte Arduino **`b`**. **`chA`** et **`chB`** sont les broches d'interruption de la carte Arduino connectées à la sortie du canal A et du canal B de l'encodeur, avec,

- **`b`** est la connexion matérielle Arduino créée à l'aide d'Arduino, spécifiée en tant qu'objet.
- **`chA`** est la broche d'interruption de la carte Arduino, connectée à la sortie du canal A de l'encodeur, spécifiée sous la forme d'un vecteur de caractères de la forme 'Dx' où x est le numéro de broche Arduino.
- **`chB`** est la broche d'interruption de la carte Arduino, connectée à la sortie du canal B de l'encodeur, spécifiée sous la forme d'un vecteur de caractères de la forme 'Dx' où x est le numéro de broche Arduino.

Exemple : `rotaryEncoder (b, ‘D2’, ‘D3’)`

vii. `readVoltage(a,pin)`

Cette instruction permet de lire la tension sur les broches d'entrée analogiques spécifiées sur la carte Arduino.

Exemple : `readVoltage(b,'A0').`

Là, Matlab va lire la valeur de la tension actuelle dans le pin analogique A0 du carte Arduino.

viii. `resetCount(encoder)`

Cette instruction permet la remise à zéro de l’encoder.

ix. `readSpeed(encoder)`

Cette instruction permet de lire la vitesse de rotation actuelle de l’encodeur.

x. `readCount(encoder)`

Cette instruction permet de lire la valeur de comptage absolue actuelle qui a été incrémentée depuis la création de l'objet encodeur.

xi. `Rpm`

Cette instruction permet l'affichage des mesures de tours moteur par minute (RPM) en pourcentage de RPM.

xii. `configurePin (b,pin)`

Cette instruction permet d'afficher le mode de la broche spécifiée sur la carte Arduino dans la connexion b.

xiii. `MODE HOME :`

Pour envoyer le robot à la position d'origine, nous avons utilisé la commande HOME, est utilisé pour réinitialiser le robot et le ramener à sa position d'origine en trouvant réellement le centre des micro-interrupteurs sur tous les axes sous le contrôle du programme. Il se déplace vers la position où tous les comptages de l'encodeur sont nuls.

Nous avons utilisé l'instruction **HOME** à la fin des programmes pour ramener le robot à sa position d'origine.

2. Le jeu d'instructions du langage proposé

Les instructions proposées peuvent être regroupées en trois groupes, un premier groupe d'instruction qui permet de commande le robot directement, ce dernier est basé sur la géométrie directe c.-à-d. l'envoie des impulsions sur les différents moteurs du bras manipulateur. Le deuxième groupe d'instructions est basé sur la géométrie inverse, ce groupe d'instructions nécessite la programmation des équations du modèle géométrique inverse. Le troisième groupe d'instruction est consacré à la configuration et l'initialisation du robot.

Dans ce qui suit nous allons présenter les modèles géométriques directe et inverse du robot pédagogique ED-7220C. Ces modèles sont basés sur les travaux de

Les relations mathématiques de ces deux modèles seront exploitées pour créer les instructions des trois groupes.

La liste de ces instructions du langage proposée est regroupée dans le **Tableau III.2**.

❖ **Matrice de transformation du robot ED7220**

La matrice globale représentant la position de l'effecteur final du robot par rapport à sa base est donnée par l'équation suivante [14]:

$$ {}^{0}_{6}T = \begin{bmatrix} C_1C_5S_{234} + S_1S_5 & -C_1S_{234}S_5 + S_1C_5 & C_1C_{234} & C_1A \\ -S_1C_5C_{234} - C_1S_5 & S_1C_{234}C_5 + C_1C_5 & S_1C_{234} & S_1A \\ C_{234}C_5 & -C_{234}S_5 & -S_{234} & B \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

$$ A = L_2S_2 + L_3S_{23} + L_4C_{234} $$
$$ B = L_1 + L_2C_2 + L_3C_{23} - L_4S_{234} $$

La matrice 3 × 3 comprenant les trois premières lignes et les trois premières colonnes est la rotation tandis que la dernière colonne représente la position (x, y, z) de l'effecteur final par rapport à la base.

❖ **Le Modèle géométrique inverse du robot**

Le modèle inverse permet de calcule les angles d'articulation requis pour atteindre la position et l'orientation données. Pour implémenter la géométrie inverse du robot nous avons suivi une approche analytique pour développer par [14]. Cette approche garantit que pour tout objet dans l'espace de travail du robot, le modèle détermine les angles d'articulation corrects. Les quatre premiers angles d'articulation, c'est-à-dire, le corps (θ_1), l'épaule (θ_2), le coude (θ_3) et le poignet (θ_4) sont calculés à l'aide de cette approche tandis que le poignet rotatif (θ_5) est directement donné par l'orientation souhaitée pour la manipulation d'objet.

Étant donné que la transformation implique une rotation ainsi qu'une translation, la forme générale de la matrice de transformation de l'outil à la base est donnée par :

$$ {}^{\text{Base}}_{\text{Bool}}T = \begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

Où la première matrice 3x3 et $(p_x, p_y$ and $p_z)$ représentant la rotation et la translation de l'effecteur par rapport à la base du robot dans un problème de géométrie inverse connues.

Le modèle géométrique inverse analytique développer après des calculs mathématiques intensifs donne les équations pour les angles articulaires $\theta_1, \theta_3, \theta_2$ et θ_4 respectivement. Ces équations expriment les angles d'articulation requis en termes de coefficients donnés.

$$\theta_1 = \text{Atan } 2(p_x, p_y)$$

$$s_{234} = c_1 a_x + s_1 a_y$$

$$c_{234} = a_2$$

$$\theta_{234} = \text{Atan } 2(s_{234}, c_{234})$$

$$= \frac{\left(c_1 p_x + s_1 p_y + l_4 s_{234}\right)^2 + (p_z - l_1 + l_4 c_{234})^2 - l_2^2 - l_3^2}{2 l_2 l_3}$$

$$c_3$$

$$s_3 = \pm\sqrt{1 - c_3^2}$$

$$\theta_3$$

$$= \frac{\left(c_1 p_x + s_1 p_y + l_4 s_{234}\right)(c_3 l_3 + l_2) - (p_z - l_1 + l_4 c_{234}) s_3 l_3}{(c_3 l_3 + l_2)^2 + {s_3}^2 {l_3}^2}\Bigg)$$

$$s_2$$

$$= -\frac{\left(c_1 p_x + s_1 p_y + l_4 s_{234}\right) s_3 l_3 + (p_z - l_1 + l_4 c_{234})(c_3 l_3 + l_2)}{(c_3 l_3 + l_2)^2 + s_3^2 l_3^2}$$

$$\theta_2 = \text{Atan } 2(s_2, c_2)$$

$$\theta_4 = \theta_{234} - \theta_2 - \theta_3$$

a) Instructions basées sur le modèle géométrique direct

Ce groupe utilise la géométrie directe du robot, donc la commande se fait par l'envoie des impulsions sur les différents moteurs du bras manipulateur. Ce groupe comporte toutes les instructions qui permettent la commande directe des moteurs du bras manipulateur, dans ce groupe nous avons les instructions :

Comme montré dans le **Tableau III.2**.

b) Instructions basées sur le modèle géométrique inverse

Pour ce type d'instructions, nous entrons les coordonnées cartésien **x, y** et **z** de l'objet. Donc, l'instruction permet de calculer les angles des articulations en utilisant le modèle géométrique inverse, ces angles sont en ensuite traduit en impulsions, le moteur doit tourner jusqu'à ce qu'a que l'objet soit atteint.

Le calcul de la position finale de l'effecteur est effectué à travers la résolution du système d'équation qui décrivant le la géométrie inverse du robot.

Ce groupe comporte toutes les instructions qui permettent la commande directe des moteurs du bras manipulateur, dans ce groupe nous avons les instructions :

Comme montré dans le **Tableau III.2**.

c) Instructions de configuration et d'initialisation

Ce groupe permet l'initialisation et la configuration du robot, il comporte les instructions suivantes, la description de ces instructions est donnée dans le **Tableau III.2**.

Les instructions sont mentionnées dans Annex 1.

<table>
<tr><th>N°</th><th colspan="2">Le regroupement des instructions proposées</th></tr>
<tr><td rowspan="4">1</td><td colspan="2">GROUPE 1 : Instruction de configuration et d'initialisation</td></tr>
<tr><td><code>mov_base.m :</code> Ce instruction permet de commander le moteur 1 (Figure III.2). Qui contrôle la vitesse et la direction de la base du bras manipulateur, cette base ne peut pas tourner que sur l'axe Z.</td><td><code>mov_shoulder.m :</code> Cette instruction permet de commander le moteur 2 (Figure III.2). Ce moteur par lequel l'épaule du bras manipulateur est déplacée selon l'axe Y.</td></tr>
<tr><td><code>mov_elbow.m :</code> Cette instruction permet de contrôler la vitesse et le sens de rotation du moteur 3 (Figure III.2). Ce moteur commande le coude du manipulateur lui permettant de monter et descendre selon l'axe y.</td><td><code>mov_grip.m :</code> Cette instruction permet de commander le moteur 6 (Figure III.2). Cet actionneur contrôle la pince du bras robotique. Le moteur permet à la pince de s'ouvrir et de se fermer à une vitesse que nous contrôlons à l'aide de cette instruction.</td></tr>
<tr><td><code>Parfor.m :</code> Cette instruction permet d'exécuter deux instructions en même temps, puisque vous pouvez faire marcher deux moteurs en parallèle. Lorsque les moteurs 4 et 5 tournent en même temps, ce processus permet au poignet de tourner sur lui-même et de rester immobile au même point.</td><td><code>position_art.m :</code> Cette instruction établit la connexion entre l'encodeur optique et les entraînements du bras, ce qui permet de lire la vitesse de rotation actuelle du moteur et de lire la valeur du comptage actuelle de l'encodeur à l'aide du décodeur quadratique. Ceci doit être utilisé pour déterminer la position de l'effecteur dans l'espace.</td></tr>
<tr><td colspan="2"><code>mov_wrist_mg.m/mov_wrist_md.m :</code> Cette instruction permet de contrôler deux moteurs. Ce sont le moteur 4 « moteur droit » et le moteur 5 « moteur gauche » illustré dans (Figure III.2). Ces deux moteurs contrôlent la vitesse et l'articulation et la rotation du poignet du bras manipulateur, où les moteurs 4 et 5 font le même travail en sens inverse, c'est-à-dire pour faire tourner le poignet, vous devez faire tourner les deux moteurs dans le même sens de travail, et puisque les deux moteurs sont opposés en sens de rotation, vous pourrez faire tourner le poignet à l'aide des engrenages et les courroies, ainsi que la possibilité de déplacer le poignet selon les axes Y et Z en utilisant un seul moteur. L'instruction permet uniquement aux deux moteurs de fonctionner en alternance.</td></tr>
<tr><td colspan="3"></td></tr>
</table>

2	***GROUPE 2 : Instructions basées sur le modèle géométrique directe « MGD »***
	`MGD.m` : Ce script, que nous avons créé à partir de l'étude géométrique des articulations du bras manipulateur, nous permet de calculer les coordonnées cartésiennes en saisissant les angles Thêta « Θ » pour chaque moteur du bras robotique, afin de les utiliser pour déterminer quel point à atteindre à partir du point de départ. Cela se fait en calculant la matrice de transformation correspond au bras robotique ED-7220C.
3	***GROUPE 3 : Instructions basées sur le modèle géométrique inverse « MGI »***
	`MGI.m` : Cette instruction, que nous avons créé à partir de l'étude géométrique des articulations du bras manipulateur [14], nous permet de calculer des angles thêta **"Θ"** pour chaque rotation du bras robotisé en saisissant des coordonnées cartésiennes du point à atteindre, afin de les utiliser pour déplacer précisément le bras vers les coordonnées de l'objet. Cela se fait en calculant la matrice de transformation générale **T** (4x4) d'un robot **4 DOF+.**
4	**`ROBOT.m` :** Cette instruction à double action peut fonctionner à la fois sur les modèles géométriques directs et inverses en utilisant les mêmes matrices et calculs que nous avons utilisés dans notre programme proposé. Elle fonctionne sur la base du Toolbox de Peter Corke « **Robotics Toolbox for MATLAB »** [15]

***Tableau III.2** : Le regroupement des instructions proposées.*

3. Test et validation du langage proposé

Pour valider le langage proposé nous allons créer une tâche robotisée en utilisant notre langage de programmation. Cette tâche est composée de six groupes d'instructions de manière suivante :

- Initialisation de la position du robot.
- Déplacement du l'effecteur (la pince) vers l'objet.
- Fermeture de la pince et monter de l'épaule et le coude.
- Déplacement vers la position du dépôt de l'objet.
- Ouverture de la pince, dépôt de l'objet et monté du bras.
- Retour à la position initiale.

❖ **Le diagramme :**

L'organigramme de la **Figure III.10**, décrit la répétions 5 fois de la tâche robotisée crée par notre langage programmation.

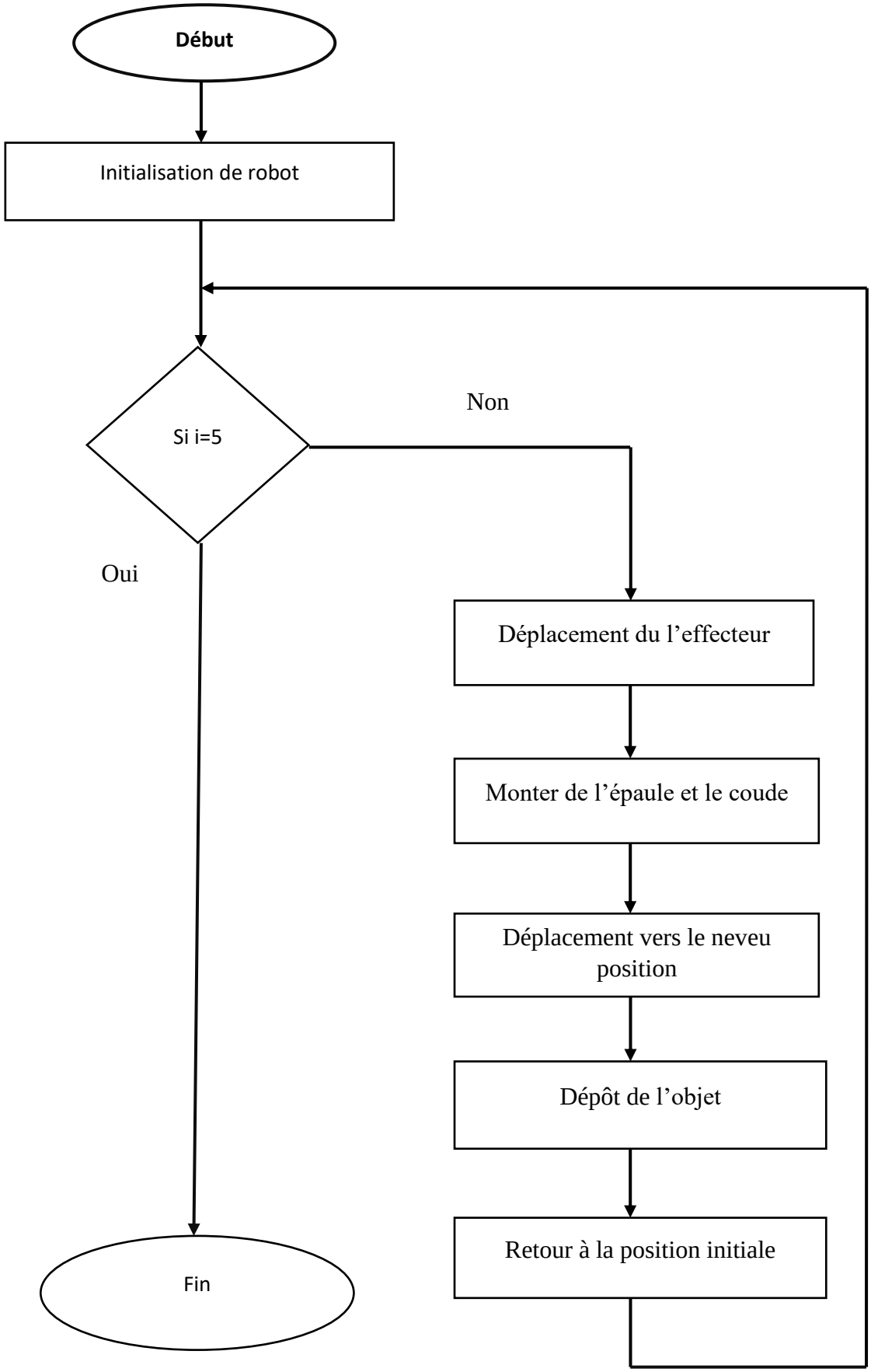
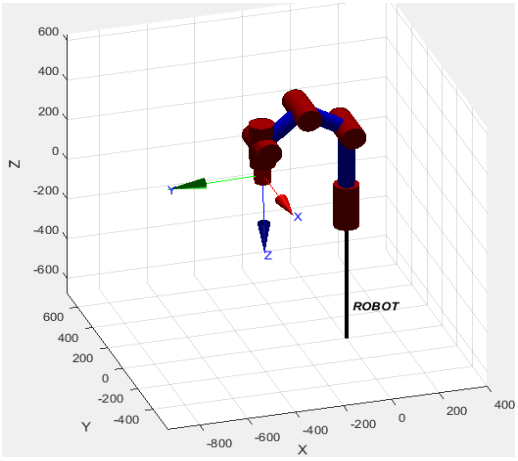
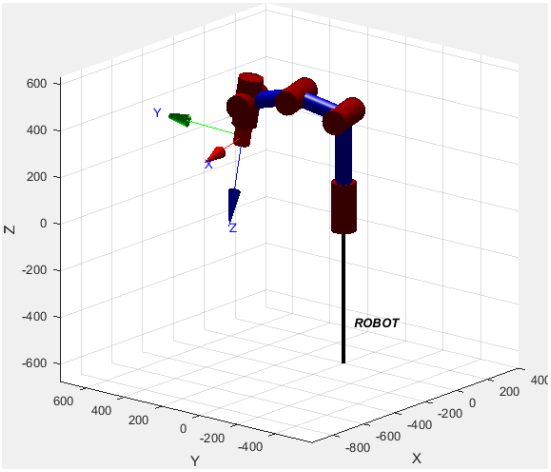


Figure III.10 : *Diagramme de la tache robotisée basé sur le langage Matlab proposé qui se répète 5 fois.*

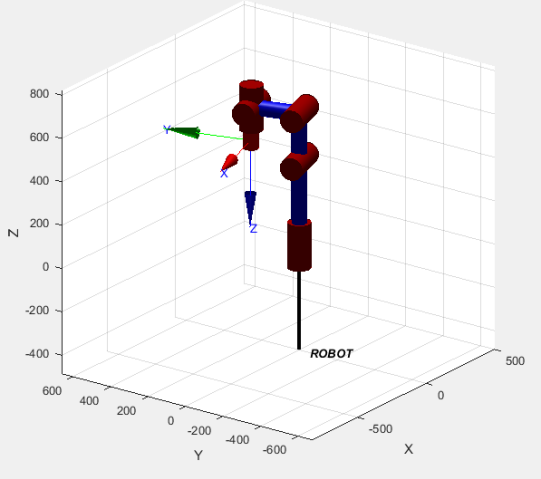
❖ **Le code :**

La description ainsi des animations réaliser avec 'Robotic Toolbox for MATLAB'[15] sont montré dans le **Tableau III.3**.

<table>
<tr><th>N°</th><th colspan="2">Instruction + Animation + Photo</th></tr>
<tr><td rowspan="2">1</td><td colspan="2">Le robot doit être en mode 'HOME'
« La Position Zéro »</td></tr>
<tr><td> 
</td><td>  </td></tr>
<tr><td rowspan="2">2</td><td colspan="2">mov_elbow(1,Th_4,velo);
mov_shoulder(1,Th_3,velo);
mov_base(1,Th_1,velo);
mov_shoulder(0,Th_3,velo);
mov_grip(1,70,velo); % L'ouverture du pince ne peut pas être montré dans l'animation.
mov_elbow(0,Th_4,velo);</td></tr>
<tr><td> 
</td><td>  </td></tr>
</table>

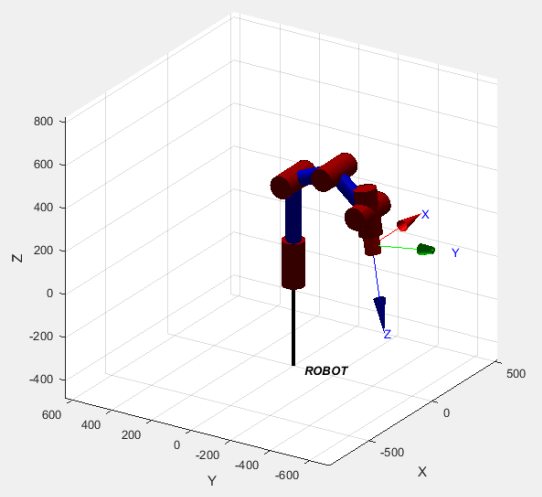
3

```
mov_grip(0,90,velo); % la fermeture aussi ne peut pas être montré dans l'animation.
mov_elbow(1,Th_4,velo);
mov_shoulder(1,Th_3,velo);
```



4

```
mov_base(1,Th_1,velo);
mov_shoulder(0,Th_3,velo);
mov_elbow(0,Th_4,velo);
```



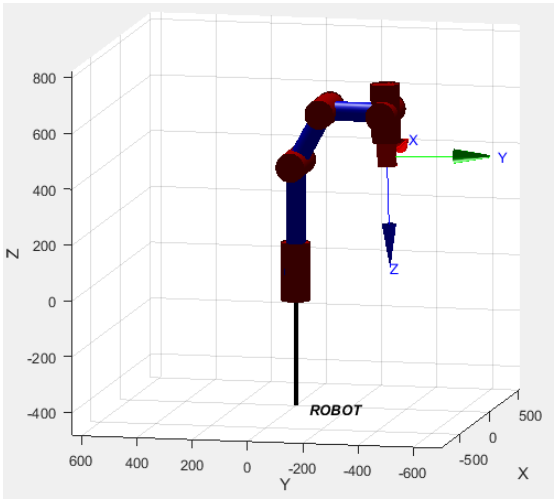
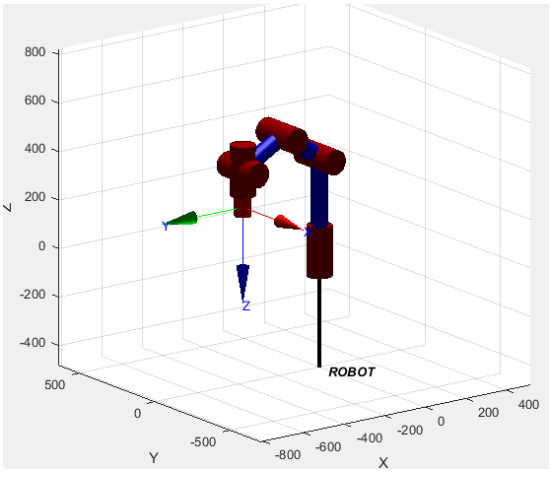
<table>
<tr><td rowspan="2">5</td><td colspan="2">mov_grip(1,70,velo);
mov_elbow(1,Th_4,velo);
mov_shoulder(1,Th_3,velo);</td></tr>
<tr><td> 
</td><td>  </td></tr>
<tr><td rowspan="2">6</td><td colspan="2">mov_base(0,Th_1,velo);
mov_shoulder(0,Th_3,velo);
mov_elbow(0,Th_4,velo);
mov_grip(0,100,velo);
% Le robot retourne à la position initial ‘HOME’</td></tr>
<tr><td> 
</td><td>  </td></tr>
</table>

Tableau III.3 : *Instruction + Animation + Photo de la tache robotisée basé sur le langage Matlab proposé.*

IV. Conclusion

Dans ce chapitre nous avons présenté notre langage de programmation, ce langage est composé de trois types d'instructions à savoir, les instructions basées sur le MGD, les instructions basées sur le MGI et les instructions d'initialisation et de configuration.

Dans la phase de validation nous avons créé une tâche robotisée où nous avons montré l'efficacité de notre langage dans l'exploitation des ressources matériel et logiciel du robot pédagogique ED 7220C.

Conclusion Générale

Conclusion Générale

L'objectif principal de ce projet de fin d'études est la réhabilitation de la partie Software du robot pédagogique ED-7220C disponible au niveau des laboratoires pédagogiques du département d'électronique. Une réhabilitation hardware a été proposé par [1]. Celle-ci est basée l'utilisation de deux carte Arduino Méga. Ce projet de fin d'études est une continuité du projet réaliser par [1].

Dans ce projet, nous avons passé par deux étapes, une étape de configuration et de préparation de l'environnement de programmation, et une étape de programmation de jeu d'instructions sous Arduino et Matlab.

L'objectif de la première étape est d'établir la liaison de communication entre la partie commande (carte Arduino Méga) et le logiciel de programmation Matlab à travers le 'package for Arduino hardware', cette étape est très importante puisqu'elle permet de configurer la carte Arduino.

Dans la deuxième étape, nous avons proposé et testé le jeu d'instructions pour la programmation du robot pédagogique ED-7220C. Ce jeu d'instructions est composé de trois groupes, à savoir, les instructions basées sur le modèle géométrique directe, les instructions basées sur le modèle géométrique inverse et les instructions de configuration et d'initialisation.

L'exécution du programme se fait directement dans l'environnement Matlab ce qui permet d'étendre le jeu d'instruction et l'implémentation des commandes avancée.

Des scripts permettons de programmé des tâches robotisées ont été proposées, pour permettre d'évaluer l'efficacité du jeu d'instructions.

Comme perspective de ce projet, nous proposons d'étendre et d'enrichir le jeu d'instructions, notamment l'implémentation des commandes avancées de la robotique.

REFERENCES BIBLIOGRAPH-IQUES

1. Chemseddine DJAAFERI et Mounir HERNOUF, Etude de la partie commande d'un bras de robot manipulateur éducatif (mise à jour avec une architecture open source), mémoire de master 2020.
2. Jacques GANGLOFF, Course of Robotics Manipulation and Control, Introduction, Cours master, Université de Strasbourg, 2017.
3. Remplacé par [2].
4. Remplacé par [2].
5. Antonio Concha SANCHEZ, Juan Felipe FIGUEROA-RODRIGUEZ, Recycling and Updating an Educational Robot Manipulator with Open-Hardware-Architecture, Robot Architecture, Article, 2020.
6. Wisama KHALIL et Etienne DOMBRE, Bases de la modélisation et de la commande des robots-manipulateurs de type série, Document, 2012.
7. Remplacé par [6].
8. Saeed B. NIKU, Introduction to Robotics, Analysis, Control, Applications Third Edition, Wiley, 2020.
9. Site web : https://www.almrsal.com/post/1118057 , consulté le 25 Mai 2022.
10. Site web : https://www.positron-libre.com/electronique/arduino/arduino.php , consulté le 29 Mai 2022.
11. Math. Graphics. Programming. The MathWorks, Inc. 2004-2022.
12. Programmer la carte Arduino en langage Arduino, L'interface du logiciel, Nouveaux programmes lycée Physique-chimie.
13. Site web : https://www.aranacorp.com/fr/utilisation-dun-encodeur-rotatif-avec-arduino , consulté le 06 Juin 2022.
14. Jamshed IQBAL, Raza UL ISLAM and Hamza KHAN, Modeling and Analysis of a 6 DOF Robotic Arm Manipulator, Canadian Journal on Electrical and Electronics Engineering Vol, 3, No, 6, July 2012.
15. P.I. CORKE, "Robotics, Vision & Control", Springer 2017.

Annex

- **function mov_base(horaire,x3,velo)**

```
% move the base of the robotic arm
with impulses, enter it according to
the range [0-400]
b = arduino('COM6','mega2560');
writePWMVoltage(b, 'D12',5);
%Determine pin 12 to carry 5 volts
if horaire==1   %If the user entre
condition 1

%code to run for case 1
for i=1:x3
writeDigitalPin(b, 'D7',1);
%Activate pin 7 by giving it the
Boolean value 1 .
writePWMDutyCycle(b, 'D7' , velo);
end
end
if horaire==0  %If the user entre
condition 2

%code to run for case 2
for i=1:x3
writeDigitalPin(b, 'D8',1);
%Activate pin 8 by giving it the
Boolean value 1 .
writePWMDutyCycle(b, 'D8' , velo);
%Control the speed of the DC motor
by entering its speed in the range
[0-1]
end
end
```

- **function mov_elbow(horaire,x1,velo)**

```
% move the elbow up and down with
impulses, enter it according to the
range [0-140]
a = arduino('COM4','mega2560');
writePWMVoltage(a, 'D11',5);
%Determine pin 11 to carry 5 volts
if horaire==1. %If the user entre
condition 1

%code to run for case 1
for i=1:x1
writeDigitalPin(a, 'D10',1);
%Activate pin 10 by giving it the
Boolean value 1 .
writePWMDutyCycle(a,'D10',velo);
%Control the speed of the DC motor
by entering its speed in the range
[0-1]

end
end
if horaire==0. %If the user entre
condition 2

%code to run for case 2
for i=1:x1
writeDigitalPin(a, 'D9',1);
%Activate pin 9 by giving it the
Boolean value 1 .
writePWMDutyCycle(a, 'D9' , velo);
%Control the speed of the DC motor
by entering its speed in the range
[0-1]
end
end
```

- **function mov_shoulder(horaire,x2,velo)**

```
% enter the direction of shoulder
lder, speed, and mouvement with
impulses to move the shoulder of the
robotic up and down.
a = arduino('COM4','mega2560');
writePWMVoltage(a, 'D12',5);
%Determine pin 12 to carry 5 volts
if horaire==1  %If the user entre
condition 1
%code to run for case 1
for i=1:x2
writeDigitalPin(a, 'D7',1);
%Activate pin 7 by giving it the
Boolean value 1 .
writePWMDutyCycle(a, 'D7' , velo);
%Control the speed of the DC motor
by entering its speed in the range
[0-1]
end
end
if horaire==0. %If the user entre
condition 2

%code to run for case 2
for i=1:x2
writeDigitalPin(a, 'D8',1);
%Activate pin 8 by giving it the
Boolean value 1 .
writePWMDutyCycle(a, 'D8' , velo);
%Control the speed of the DC motor
by entering its speed in the range
[0-1]
end
end
```

Annex

- **function mov_wrist_mg(horaire,x4,velo)**

```
% move and rotate the wrist of the
robotic arm using the left motor,
entre the pulses according to the
range [0-120]
a = arduino('COM4','mega2560');
writePWMVoltage(a, 'D13',5);
%Determine pin 13 to carry 5 volts
if horaire==1. %If the user entre
condition 1

%If the user entre condition 1
for i=1:x4
writeDigitalPin(a, 'D5',1);
%Activate pin 5 by giving it the
Boolean value 1 .
writePWMDutyCycle(a,'D5',velo);
%controling the speed of DC motor,
using pin 5, its speed set in the
range [0-1]
end
end
if horaire==0. %If the user entre
condition 2

%If the user entre condition 2
for i=1:x4
writeDigitalPin(a, 'D6',1);
%Activate pin 6 by giving it the
Boolean value 1 .
writePWMDutyCycle(a,'D6',velo);
%controlling the speed of DC motor,
using pin 6, its speed is set in the
range [0-1]
end
end
```

- **function mov_wrist_md(horaire,x5,velo)**

```
b = arduino('COM6','mega2560');
writePWMVoltage(b, 'D13',5);
%Determine pin 13 to carry 5 volts
if horaire==1 %If the user entre
condition 1

%If the user entre condition 1
for i=1:x5
writeDigitalPin(b, 'D5',1);
%Activate pin 5 by giving it the
Boolean value 1 .
writePWMDutyCycle(b, 'D5' , velo);
%controling the speed of DC motor,
using pin 5, its speed set in the
range [0-1]

end
end
if horaire==0  %If the user entre
condition 2

%If the user entre condition 2
for i=1:x5
writeDigitalPin(b, 'D6',1);
%Activate pin 6 by giving it the
Boolean value 1 .
writePWMDutyCycle(b, 'D6' , velo);
%controlling the speed of DC motor,
using pin 6, its speed is set in the
range [0-1]
end
end
```

- **function mov_grip(horaire,x6,velo)**

```
 %Opening and closing the gripper
b = arduino('COM6','mega2560');
writePWMVoltage(b, 'D11',5);
if horaire==1 %If the user entre
condition 1

%code to run for case 1
for i=1:x6
writeDigitalPin(b, 'D10',1);
writePWMDutyCycle(b, 'D10' , velo);
end
end
if horaire==0. %If the user entre
condition 2

%code to run for case 2
for i=1:x6
writeDigitalPin(b, 'D9',1);
writePWMDutyCycle(b, 'D9' , velo);
end
end
```

- **Fonction Parfor**

```
clc
clear all
b = arduino('COM6','mega2560')
a = arduino('COM4','mega2560')
horaireg=1; % Engine rotation
direction
xg=45; %The impulses that the arm
will move
horaired=1;
xd=50;
velo=0.8; %Engine speed of motor 4
velo=0.7; %Engine speed of motor 5
horaire=0;
x2=50;
velo=0.7;
parfor i=1:2 %Run two parallel
instructions at the same time
    if i==1
mov_wrist_mg(horaireg,xg,veloG);
```

```
    else
mov_wrist_md(horaired,xd,veloD);
    end
end
```

- **function position_art(rpm)**

```
%Read the current position of the
manipulator arm joints
b = arduino('COM6','mega2560');
Encoder=rotaryEncoder(b,'D2','D3');
%Setting pin 2 and 3 as
communication channels with the
Arduino
PotPin = configurePin(b,'A0');
HBA1 = configurePin(b,'D10');
resetCount(Encoder);
rpm = nan(1, 50);
h = figure(1);
while true
voltage=readVoltage(b,'A0'); %Read
the voltage on pin 0
writePWMVoltage(b,'D10',voltage);
%Determine pin 10 to carry the
voltage read on the previous pin
(pin 0)
 rpm=readSpeed(Encoder);  %reads the
current speed of the motor
pause(0.0000001);
rpm = [rpm(2:end),
readCount(Encoder)]; %read the
current absolute count value of DC
motor.
disp('rpm = '); %displays the value
of  rpm with printing the variable
name
disp(rpm);  %displays the value of
variable
    plot(rpm);
    grid on;
    grid minor;
    xlim([1, 50]);
    ylim([-2000,2000]);
    ylabel('rpm');
    drawnow;
end
```

- **function [theta1,theta2,theta3,theta4]=MGI(nx,ny,nz,ox,oy,oz,ax,ay,az,px,py,pz)**

```
%Enter the coordinates to calculates
the angles
impulse_max1=400; %the maximum value
of the impulses, that the joint can
move
theta1_max=310; %the maximum value
of the angle, that the joint can
move
impulse_max2=140;
theta2_max=160;
impulse_max3=120;
theta3_max=130;
impulse_max4=30;
theta4_max=50;
nx=1;ny=0;nz=0;ox=0;oy=1;oz=0;ax=0;a
y=0;az=1;px=5;py=0;pz=0; %Enter the
coordinates
        theta1=abs(rad2deg(
atan2(px,py)))
L1=385;
L2=220;
L3=220;
L4=155;
c234=az;
s234=ax*cos(theta1) +
ay*sin(theta1);
theta234=rad2deg(atan2(s234,c234));
%Calculating angles in radians
c_theta3=((cos(theta1)*px +
sin(theta1)*py + L4
)*((sin(theta234))^2) + (pz -L1
+L4*((cos(theta234))^2) -(L2^2)-
(L3^2)))/2*L2*L3;
s_theta3=  sqrt(1-
(cos(c_theta3))^2);
theta3=
abs(rad2deg(atan2((sin(s_theta3)),(c
os(c_theta3)))))
c_theta2=(cos(theta1)*px +
sin(theta1)*py + L4
*sin(theta234)*(cos(theta3)*L3+L2) -
(pz -L1
+L4*cos(theta234))*sin(theta3)*L3)
/(cos(theta3)*L3+L2)^2 +
(sin(theta3))^2 * L3^2;
s_theta2=-((cos(theta1)*px +
sin(theta1)*py + L4
*sin(theta234)*(sin(theta3) * L3) +
(pz -L1
+L4*cos(theta234))*(cos(theta3)*L3+L
2)))/(cos(theta3)*L3+L2)^2 +
(sin(theta3))^2 * L3^2;
        theta2=
abs(rad2deg(atan2((sin(s_theta2)),(c
os(c_theta2)))))
theta4= abs(rad2deg(theta234 -
(theta2 +theta3)))
    velo = 0.5
    horaire= 0
```