REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
الجمهورية الجزائرية الديمقراطية الشعبية

MINISTRY OF HIGHER EDUCATION
AND SCIENTIFIC RESEARCH

HIGHER SCHOOL IN APPLIED SCIENCES
--T L E M C E N--

المدرسة العليا في العلوم التطبيقية
École Supérieure en
Sciences Appliquées

وزارة التعليم العالي والبحث العلمي

المدرسة العليا في العلوم التطبيقية
-تلمسان-

Mémoire de fin d'étude

Pour l'obtention du diplôme d'Ingénieur

Filière : Automatique
Spécialité : Automatique

Présenté par : BENDI-OUIS Mustapha Réda
RABAH Mohammed Abdennour

Thème

Design, Simulation and Analysis of a 4 DOF Scara Robot Manipulator

Soutenu publiquement, le 03 / 07 / 2022 , devant le jury composé de :

M L. MERAD	Professeur	ESSA. Tlemcen	Président
M B. CHERKI	Professeur	ESSA. Tlemcen	Directeur de mémoire
M S. BENMANSOUR	MCB	ESSA. Tlemcen	Co- Directeur de mémoire
M C. BENSALAH	MCA	Université de Tlemcen	Examinateur 1
M F. ARICHI	MCA	ESSA. Tlemcen	Examinateur 2
M R.A. ADJIM	Ingénieur de laboratoire	ESSA. Tlemcen	Invité

Année universitaire : 2021 / 2022

DEDICATION

At the conclusion of 17 years of studies, I hereby wish to write a few words to express my sentiment at the twilight of my academic journey. I am pleased to dedicate this work to a handful of people dear to my heart :
To the man, colleague and brother who collaborated with me in this work **RABAH Mohammed Abdennour**, I would like to thank you for your collaboration and friendship during all those years. You have been the model companion throughout the years and common project and I could not have wished for a better partner.
To my ever loving parents **BENDI-OUIS Karim** and **Pr. BENCHOUK Assia**, I would like to express my gratitude for making me the person that I am today, for the affection and sacrifices made throughout my upbringing.
To my younger sister **Meriem** for her continuous support, wishing her to achieve even higher feats than myself. And to all my extended family members.
To **BELLAMRI Ikram** for her extended help and precious advise as well as her company for all those years.
To **DRISSI Naila Sarra** and **KARA MOHAMMED Selma** for their uninterrupted support and friendship.
To my mates **LATRECHE Mohammed Cherif** and **MIHOUBI Yakoub** for unforgettable memories together and even more to come.
To **KHELIL Ghizlene** for whom I wish success for next year and beyond.
To my oldest friend **OUAFI Mohammed Zakaria** who will likely accompany me to the grave.
And to my friends **Alaa eddine**, **Amine**, **Bouchra**, **Nessrine**, **Omar** and anyone involved in this project and my journey whom I express my deep gratitude to.
I would like to thank every single one of you for contributing to this project or ultimately to the person that I am today.

BENDI-OUIS Mustapha Réda

Already 6 years have passed, and here I am writing my last words in this work, a work that I am proud of. Not only because we worked hard on it, but also because it follows a passion. I was known as a child of destroying toys out of curiosity. Here I am a part of a team that built a modest robot. This job would never have been done without Allah's blessing, so thank Allah which with his greatness, goodness are done.
I dedicate this work to myself who have always believed in me, and always aimed big.
I dedicate this work to my colleague and friend Réda for his dedication, hard work and motivation.
My family words can not describe emotions yet here is a simple word for you, Mom you are the greatest thank you for being supportive all the time. Dad, thank you for raising me and always pushing me to never settle for less than I deserve. Anes you are one of the people I aspire to be like, you have always been an advisor. Thank you. Sisters you are the best, thank you for your support and motivation. This work is dedicated to you.
My Friends, Zaki, Omar, Lotfi, Fakhro, Ahmed, Yacine, Nadir and El-Hadi without you my friends i am nothing. Thank you for always being there for me. Thank you for being you. I dedicate this work to you.
My female colleagues, Bouchra thank you for being there at my lowest, Sarah, Ghizlene. I dedicate this work to you too.
I dedicate this work to all the beloved people.

RABAH Mohammed Abdennour

ACKNOWLEDGEMENT

First and foremost, we praise the almighty Allah for his benediction and blessing that guided us throughout our lived and permitted us to accomplish this project.
We would like to address our most sincere thanks to our supervisor **Pr. CHERKI Brahim** who was an inspiration and immense source of knowledge as a teacher ; who accepted to guide us in the pursuit of our dream in robotics and whose invaluable advise and wisdom was key in the accomplishment of our project.
We would also like to thank our co-supervisor **Dr. BENMANSOUR Sid Ahmed** for his major contribution in our work as well as his wisdom and precious recommendations.
Our most grateful thanks to **Mr. ADJIM Ramz-Eddine Abderrezak** for his invaluable help and experience over the years with the realisation presented in this project and during our previous projects.
We would like to express all our gratitude to **Pr. MERAD Lotfi** for accepting to preside over the jury as well as his guidance and continuous help to us throughout the years ; as well as to **Dr. BENSALAH Choukri** and **Dr. ARICHI Fayssal** for honouring us with their presence in our jury.
Finally, we would like to address our thanks to **Dr. ROUISSAT Boucherit** in addition to everyone at ESSAT for making the entirety of the school's infrastructure available to us namely the Fablab and other parts of the establishment.

CONTENTS

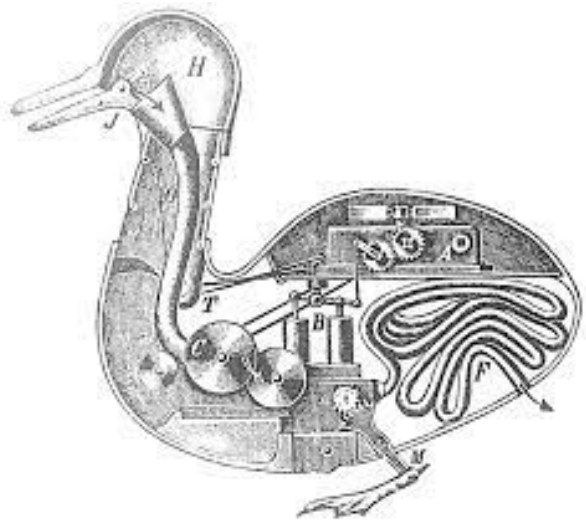
LIST OF FIGURES

LIST OF TABLES

GENERAL INTRODUCTION

The ambition to build skilled and intelligent machines has existed since the dawn of time ; and humans have continually sought alternatives capable of interacting with their surroundings in the same manner people do. In Greek mythology, the story of the Titan *Prometheus* corresponds to this desire ; and so do many other imaginative historical realisations. From *Frankenstein's* monster to *Star Wars*, science fiction films and novels have used the theme of robots and their interactions with humans. The term "*robot*" means "subordinate labour" and it was first presented by *Karel Čapek* [1].
Automata were incredibly complex clock-work machines loaded with gears and cams in the 18^{th} and 19^{th} centuries 1a. Vaucanson's mechanical duck is a famous example 1b, *"The Duck stretches out its neck to take corn out of your hand; it swallows it, digests it, and discharges it digested by the usual passage."* Jacques Vaucanson, 1738.

(a) 18^{th} and 19^{th} century automata



(b) Vaucanson's mechanical duck

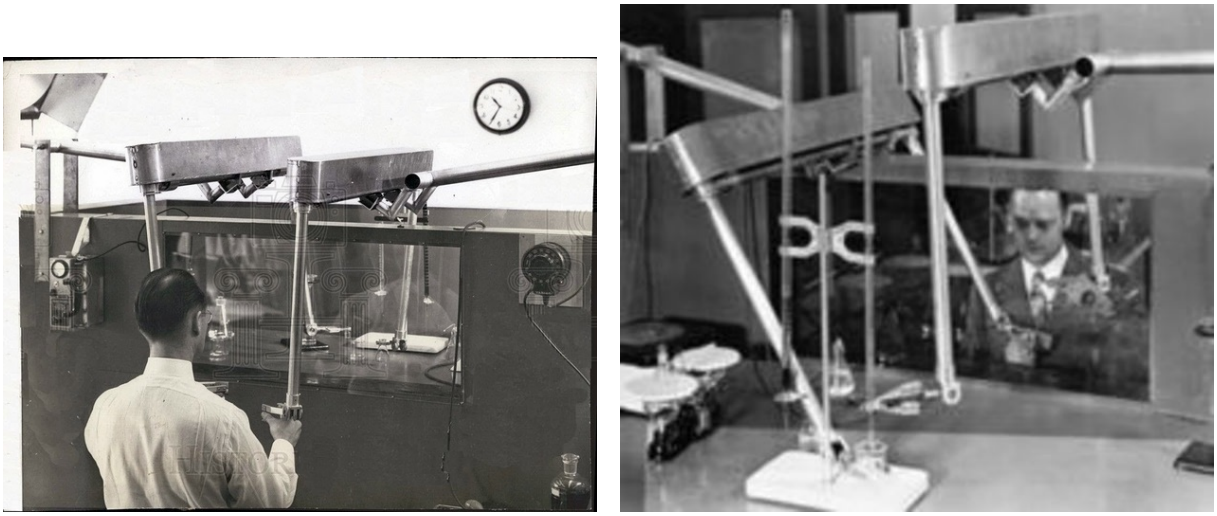
Figure 1: Automata : complex clock-work machines

Following that, *Joseph Jacquard* relied on previous works to develop the so called *Jacquard loom* see 2 ; which is a device that manages the weave pattern through a sequence of

punched wooden cards. This was a type of automated machine that could be programmed. The pattern that emerged was determined by the holes punched in those wooden cards [1].

Figure 2: Jacquard loom - 1801

Another significant achievement in the history of robotics was made at the *Argonne National Laboratory in the United States*, and it is the notion of *teleoperation*. This allowed them to handle radioactive materials with two mechanical hands see 3.



(a) (b)

Figure 3: Argonne National Laboratory robot

In 1956 *Devol and Joseph Engelberger* founded ***Unimation,*** the first robotic firm, to manufacture their first industrial robot arm which would later go to work in General Motors plants [1]. Since then many complex and advanced robots have been developed by numerous companies.
Science fiction novels and movies have heavily shaped the way in which many people

perceive robots to be and what they can perform. Robots, on the other hand, are machines that intelligently link perception and action [2]. Robotics is an academic field that studies how to design, model, plan and control those machines ; with living creatures and how they perform complex tasks as its main inspiration.
Robots are categorised into two classes: fixed basis robots, *robot manipulators*, and moving base robots, *mobile robots* 4a. Robot manipulators are an interconnection of links *(rigid bodies)* by means of joints *(articulations)*. This type of robots can be an open kinematic chain *(serial)* or a closed kinematic chain *(loop)* 4b. The joints can either be prismatic that present translational motion along its axis or revolute that present rotational motion along its axis ; each of them presents one degree of freedom. Different robot structures were developed by configuring different connections and joints [2], such as the SCARA configuration which is the topic of this work.

(a) TurtleBot3 Waffle Pi (b) Delta robot by Omron

Figure 4: Different types of robots

The modelling of a robots consist of the kinematics analysis of their structure that is to say the description of their motion with respect to a fixed Cartesian frame ignoring the cause of motion. It determines the analytical relationship between the end-effector pose (position and orientation) and joint position, not to be confounded with differential kinematics which studies that relationship in term of velocities [2]. Modelling also studies dynamics that consist of the derivation of the equations of motion as functions of forces and moments acting on the robot. The dynamic equations are complex and rely on accurate knowledge of each link's mass and inertia, which may not be readily available. Even if it were, the dynamic equations would still fail to account for physical phenomena like friction, elasticity, backlash, and hysteresis [3].
According to the task assigned to the robot e.g painting ; a desired trajectory is needed. The aim of planning a trajectory for a robot is to generate a geometric sequence of configurations of a robot with respect to a time scaling [3]. To achieve the wanted behaviour for a robot, controls strategies are developed namely motion or position control, force control, hybrid motion-force control, impedance control and computed torque control. The control problem is complex [2] ; however, they compensate for uncertainties by using feedback from different sensors.

In this project, we aimed to design and realise a 4 degrees of freedom robot manipulator type SCARA for pick-and-place applications. Additionally, we established the objective to obtain modest to decent performances as well as making it a platform for the application of control theory by ourselves and future users as a didactic robot.
In this work composed of 3 main chapters, we will first introduce in chapter 1 the history of robotics from etymology and a fruit of the imagination of mankind to modern robotics and its many applications. We will then in chapter 2 walk the reader step by step on the process of design of the mechanical structure and the electronics as well as the assembly of the robot. In chapter 3, we will first derive forward and inverse kinematic models in addition to deriving the velocity equations. We will then introduce the notion of trajectory planning with its subsequent simulation on Matlab Simulink using the URDF model of our robot. Additionally, we will showcase the intricacies of the manipulator's workspace using a Matlab simulation.

CHAPTER

1

STATE OF THE ART

1.1 Introduction

The word robot is known to most people and projects a certain image in their minds ; mostly that of a sentient humanoid machine. However, an empirical definition of the term is more generalized and free from the influence of the cultural depiction of the term.
A robot is defined by the *ISO*[1] as an

> *'Automatically controlled, reprogrammable, multipurpose manipulator, programmable in three or more axes, which can be either fixed in place or mobile for use in industrial automation applications'* (ISO 8373)

Since the dawn of robotics in the 1960s, robots have been used for a multitude of tasks that fit the term "**4D tasks**" : ***D**angerous, **D**ull, **D**irty, **D**umb* [4]. In other words, we send robots to get us out of harm's way on a mine field for example ; or for repetitive tasks that might be too boring for us. Additionally, robots might perform dirty tasks like painting or soldering especially where gases and particle emissions might constitute a health hazard with prolonged exposition.
The main focus of this chapter is to relate the evolution of robotics across history from a fantasy to today's advancements and what lies beyond.

1.2 History of robotics

1.2.1 A long-lived dream

For millennia, mankind has dreamt of an autonomous machine that would help it in its tasks. A relic of this dream can be found in the form of a sentient man-made humanoid machine in multiple mythologies : The Greeks had *Talos* 1.1 and *Galatea*, the Hebrews spoke about the *Golem* and the Inuits in Greenland fantasized about *Tupilak* [5].

[1] International Organization for Standardization

Figure 1.1: Talos - Jason and the Argonauts, 1963

Over the centuries, humanity developed more and more ways to automate their tasks. For instance, the invention of the first automated loom using punched cards in 1801 by the Frenchman *Joseph Jacquard*. Further steps were made towards automation by the Serbian inventor *Nikola Tesla* who invented amongst other things the first radio-controlled boat in the late 19^{th} century.
The first use of the word *robot* is credited to the Czech *Joseph Čapek* who in 1920 came up with the term to help his brother *Karel Čapek* writing his play *Rossum's Universal Robots* [5] better known as *RUR*. It is derived from the Czech words *robotnik* meaning peasant and *robota* meaning servitude. This etymology replaced the previously used word *automaton*.
Robots would still remain only a fruit of the imagination of authors and scenarists at the time ; fascinating the public with stories and the first movies including robots in the golden age of cinema 1.2.

Figure 1.2: First robot in film history - Metropolis, 1927

Among the most influential people in the early history of robotics was *Isaac Asimov* (1920-1992), one of the first science-fiction writers who was the pioneer of robotics in popular culture. He introduced the term *robotics* as the science and technology behind the design and construction of a robot. Additionally, we owe him the famous three laws of robotics published in 1942 and further corrected in 1985 to add a "zeroth" law that read as follows [5] :

‘

- ***Zeroth Law:*** *A robot may not injure humanity, or, through inaction, allow humanity to come to harm.*
- ***First Law:*** *A robot may not injure a human being, or, through inaction, allow a human being to come to harm, unless this would violate a higher order law.*
- ***Second Law:*** *A robot must obey the orders given to it by human beings, except where such orders would conflict with a higher order law.*
- ***Third Law:*** *A robot must protect its own existence, as long as such protection does not conflict with a higher order law.*

’

(Isaac Asimov, 1985)

The impact of Azimov's work on the technical side of robotics cannot be understated as his stories inspired a generation that would be the pioneers of the dawn of robotics in the late 1950s and early 1960s.

1.2.2 History of manipulator robots

The advent of robotics coincided with the post-world war II technological boom. The remarkable advances made in electronics and computing suddenly made an automated machine controlled by computer a possibility.
In 1946, an inventor named *George C.Devol, Jr.*, made his first leap in the world of automation by patenting a magnetic playback design used for the control of machines. His continuous pursuit of this path led him to another invention in 1954 of a multipurpose mechanical arm that he attempted patenting under the name *universal automation*, better known as *unimation*. It was able to move objects from one place to another via programming only ; a truly remarkable feat for the technology of the time.
Two years later, he met *Joseph F. Engelberger*, a physics major from Columbia University who worked as a nuclear physicist. Their conversation revolved around Devol's invention. Engelberger, who was a grand admirer of Azimov's stories immediately associated the invention to the robots that Azimov described. This meeting marked the beginning of a partnership between the two men that led to the dawn of industrial robotics.
They worked together on developing that invention for an industrial purpose and gave birth to the *Unimate* whose first prototype the Unimate #001 1.3 was completed in 1959 and greatly attracted General Motors.
In 1961, Engelberger founded **Unimation.Inc** and in the same year, which later became the worldwide leader in robotics. They achieved fame in 1966 with the first demonstration of the Unimate on live television at the tonight show on NBC live from New York. Later that year, they sold a licence to **Nokia** and **Kawasaki** in order to make and commercialize the Unimate on the European and Asian markets [6]. For all his contributions to the

Figure 1.3: Unimate - The first industrial robot in history, 1959

field, Joseph Engelberger is wildly considered today as the father of robotics and the Engelberger Robotics Awards are presented annually by the *Robotic Industries Association* since 1977 [5].
The advent of this new age of robotics quickly passed down to universities and research laboratories all racing to fund the creation of their first prototype. The Stanford university in California, USA emerged as the leader in this field by founding the *Stanford Research Institute* (SRI) in 1946 [5] which then saw the creation within it in 1966 of the *Artificial Intelligence Center* (*AIC*). A key revolution in the field was the introduction of sensors in robots which some consider to be the dawn of the second generation of robots [7]. This major change permitted to better control the robot by the use of PLCs[2].
One of the early projects at the AIC and a pioneer to this new generation was the development of the robot *Shakey* 1.4 by the team of the *Dr.Charles Rosen* between 1966 and 1972. It was equipped with a camera and multiple sensors to determine its position and avoid collisions (although it was very slow due to the computing speed of its time) It is still today regarded as one of the major works in the early days of robotics, computer vision and artificial intelligence as a whole.

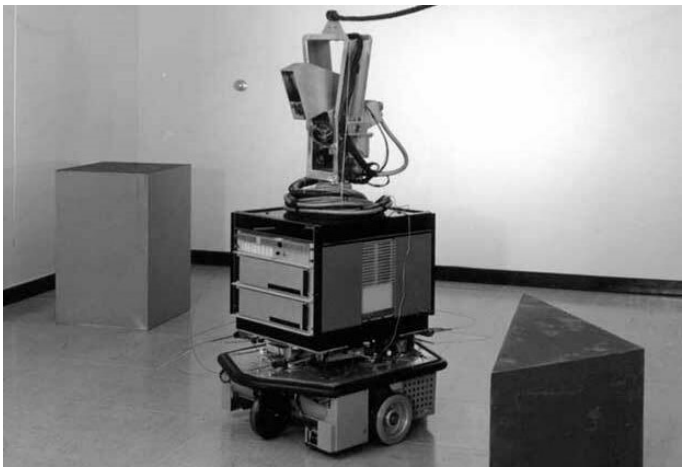


Figure 1.4: Shakey - The first autonomous mobile robot, 1972

[2] Programmable Logic Controller

Still at Stanford University in 1969, *Victor Scheinman*, a mechanical engineering student developed the first prototype of the Stanford Arm 1.5. It consists of a 6 degrees of freedom manipulator with a $RRPRRR$[3] structure 1.6. This design is still one of the major robot structures known to this day. Moreover, this first prototype was equipped with brakes on all joints to maintain position in order for the computer to calculate the next move [5].

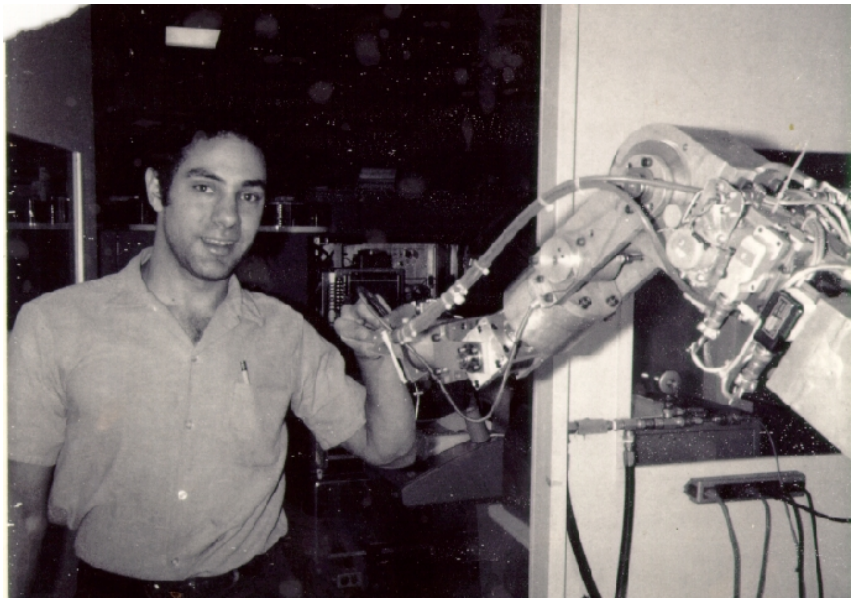


Figure 1.5: Victor Scheinman with an early prototype of the Stanford arm, 1968

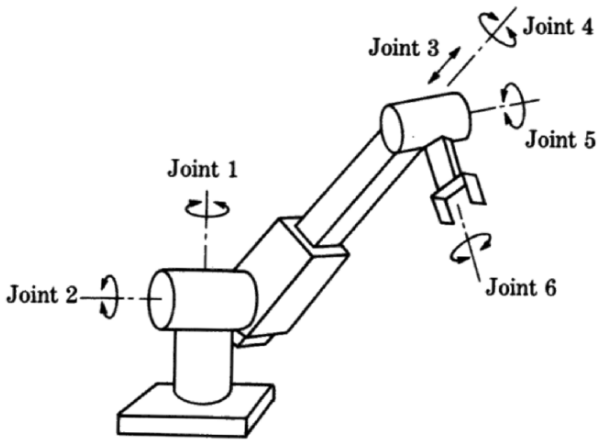


Figure 1.6: The Stanford arm structure

Research in this new field generalized to multiple countries in the late 1960s and early 1970s. A new structure was proposed in 1978 by Japanese scientist Hiroshi Makino from Yamanashi University [8]. The $SCARA$[4] was an $RRPR$[5] structure with 3 revolute joints with vertical axes and a prismatic joint at the end of the kinematic chain 1.7. This new structure was particularly suitable for pick and place applications and assembly tasks.

[3] 2-Revolute Prismatic 3-Revolute
[4] Selective Compliance Assembly Robot Arm
[5] 2-Revolute Prismatic Revolute

Figure 1.7: One of Hiroshi Makino's first prototypes of the SCARA structure

Following those advances, the 1980s and 1990s came as the age of the generalization of robotics. More investments into the field along with the broader use of internet allowed for the adoption of the Ethernet communication which was quickly joined by the first Linux based operating systems [7].
Along with the dawn of the XXIst century came along a new vision of robotics. With industry no longer being the sole focus of the field, more diversity came into play as professional and home applications such as collaborative robots changed the way people look on robotics. Another decisive factor was the newly found accessibility with the advent of 3D printing, SBCs[6] and the hundreds of open source documents and code. This allowed countless hobbyist and tech enthusiasts such as ourselves to get into robotics without the need of a large budget and thus bringing more new and creative ideas on the table.

1.3 Current technologies

Since the beginning of the century, robotics design is facing somewhat of a saturation with few revolutionary changes being added in the last few decades from a design perspective. However, performances are getting better with improved speed, accuracy and repeatability [7]. The biggest improvements made over the last years was on the software side, as the rise of artificial intelligence allowed new control method to emerge. The focus of development also switched from the industrial side in favor of the humanoid and more generally bio-inspired robotics 1.9. These factors combined created a state of stagnation in modern manipulator robotics as the next steps in their evolution are facing some difficulties. One of the key challenges ahead being the cooperative nature of robot manipulators and their ability to function with each other is severely hampered by the current closed system nature of each constructor. This factor makes the communication between multiple differently manufactured robots very difficult and time consuming [7].

[6] Single Board Computers

Figure 1.8: A panel of modern KUKA robot manipulators

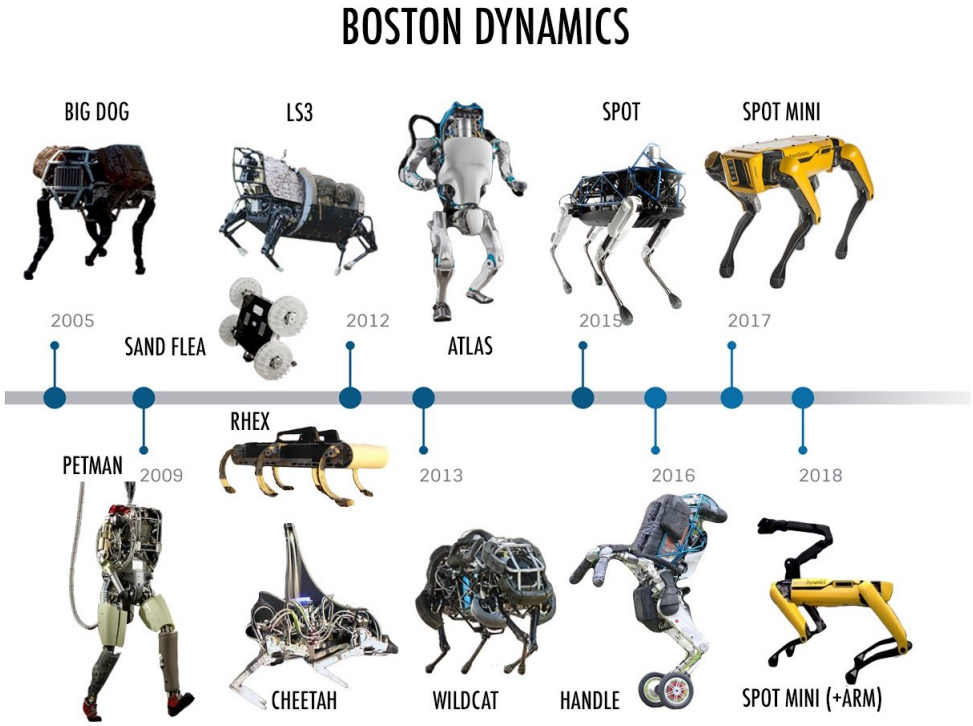


Figure 1.9: The evolution of mobile robots made by Boston Dynamics

1.4 Future challenges

- A solution to the aforementioned compatibility issues could be found in a new standardized protocol for all manipulators via for example *Robotics Operating System* (ROS) that would allow a common control and exploitation software to better handle the cooperation.
- The exponential rise of Artificial intelligence gives endless new possibilities for the control and supervision of manipulators. Multiple new exploitation methods such as our master project on visual servoing alongside many others will be key in the next steps of the evolution of the field.

As demonstrated in this chapter, constructing a robot has been a dream of mankind since the dawn of time. In the next chapter, we will showcase our own interpretation of this dream.

CHAPTER

2

DESIGN AND REALISATION

2.1 Introduction

Robotic manipulators have had a significant role in the industry over the past years due to their wide range of applications such as welding, painting, precision cutting, assembling, material transforming and pick-and-place.
Different types of robotic structures exist, among them the **SCARA** robot manipulator acronym for ***S**elective **C**ompliance **A**rticulated **R**obot **A**rm*. The SCARA configuration is unique and designed to manage various material handling operations. It consists of four *DOF*[1], three revolute joints and one prismatic joint. The SCARA is stiff in its vertical direction but, due to its parallel arranged axes, show compliance in its horizontal working plane, thus facilitating insertion processes typical in assembly tasks [9].
The goal of this chapter is devoted to the prototyping of a 3D printed SCARA robot arm.

Figure 2.1: Typical SCARA robot, made by FANUC

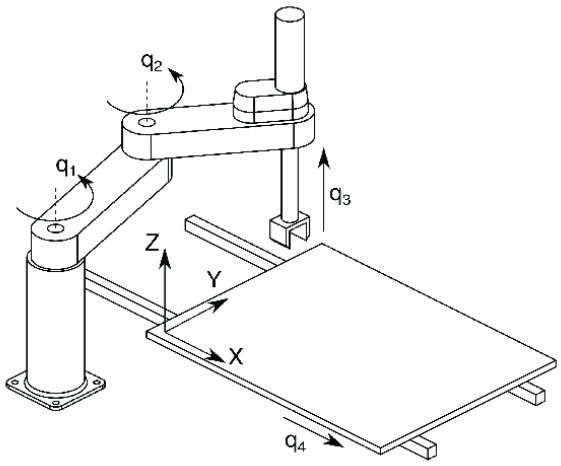
[1] Degrees Of Freedom

2.2 Prototyping and design goals

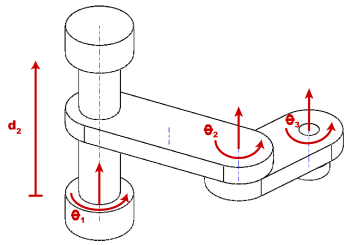
The design's concept was to create an educational platform to be used as a research tool with a simple implementation of new control algorithms. This is due to the control systems of such robots being designed for common industrial applications and in general are not suitable for high level research purposes [10]. The first goals were set to design a SCARA prototype with 400 mm maximum reach along the $\hat{x}, \hat{y}$ plane, 300 mm along the $\hat{z}$ axis and 1 kg of maximum payload. All while taking in consideration the limitation of the availability of the majority of the parts at ESSAT[2] and the national market in general on one hand ; and on the other hand most of the parts could be 3D printed, while keeping an eye on the general cost of the prototyping.
The design process consists of the choice of mechanical and electrical components, moreover the link and joint design that was inspired from [11].

2.3 Transmission Selection

The design is conceived to constrain the centre of gravity of the arm as near as possible to base, hence lowering the total inertia and static unbalancing of the system [12]. Thus, an RPRR[3] structure was introduced as shown in figure 2.2b.



(a) Industrial SCARA diagram

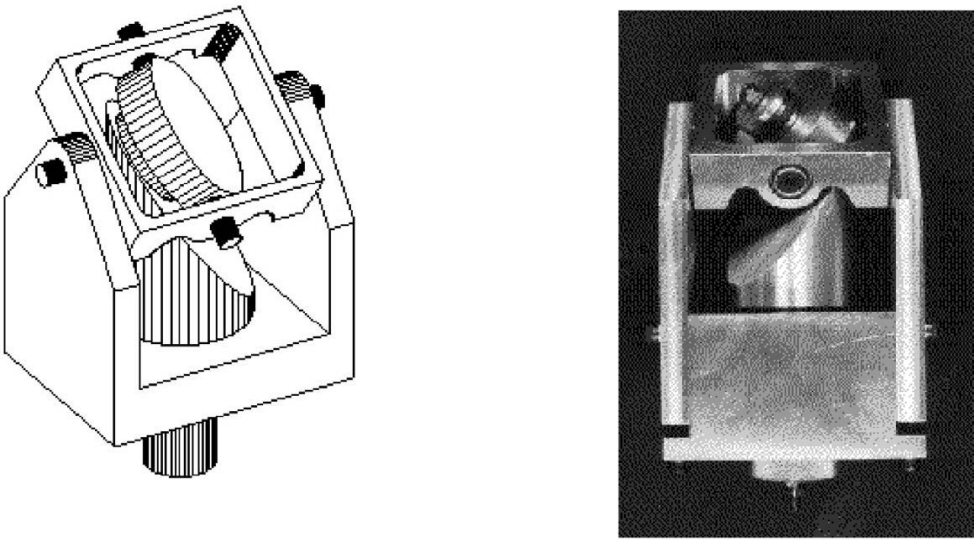


(b) The SCARA prototype diagram

Figure 2.2: SCARA robot diagram

[2] École Supérieure en Sciences Appliquées de Tlemcen
[3] Revolute Prismatic Revolute Revolute

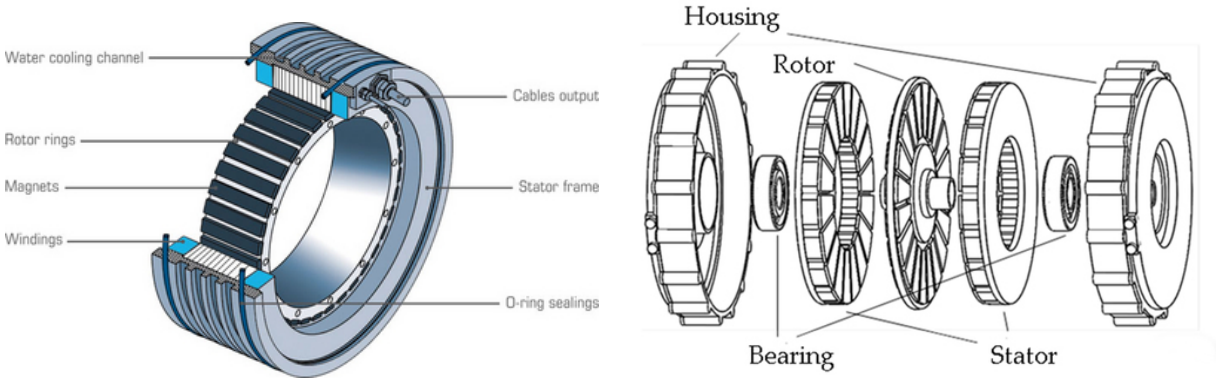
A variety of transmission systems exists. In [13], the authors demonstrated a new non-linear direct power transmission system, the *Gimbal Drive* see 2.3. This moves the motors from links to the base, thereby eliminating the weight of the motors in the robot's dynamics [10].



(a) (b)

Figure 2.3: The gimbal drive.

In Reference [14] various adaptive controls were compared on a *Direct Drive* SCARA, refer to 2.4.



(a) (b)

Figure 2.4: Direct drive rotary motors.

The paper in [15] investigates the effects of *Harmonic Drive* see 2.5 characteristics on the dynamic behaviour of industrial SCARA robot.

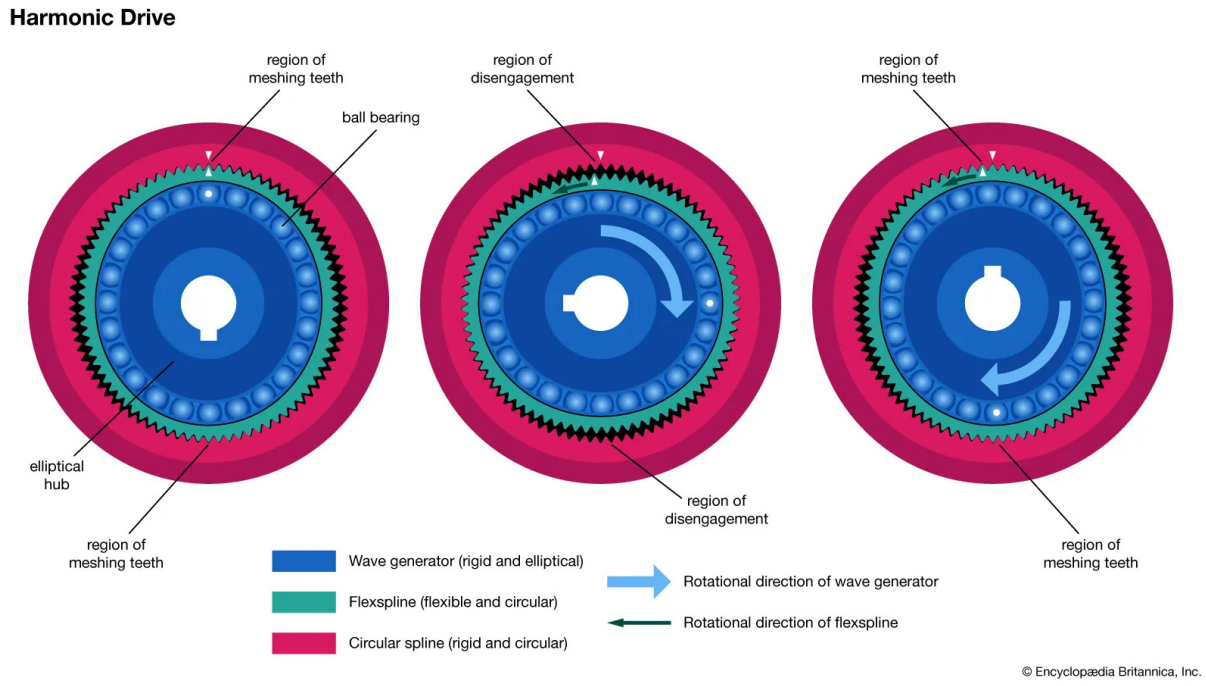


Figure 2.5: Harmonic drive
(Encyclopedia Britannica, Inc.)

Timing Belt Drive transmission system was selected as our transmission system to enable us to move the motors as near to the base as possible while minimising arm inertia. However, the backlash induced by differential and transmission mechanisms is a drawback [12]. Planetary gearboxes 2.6 and harmonic drives could be employed, nevertheless they would be complex to implement and would demand the use of specialised bearings, as well as increasing the link weight and inertia.

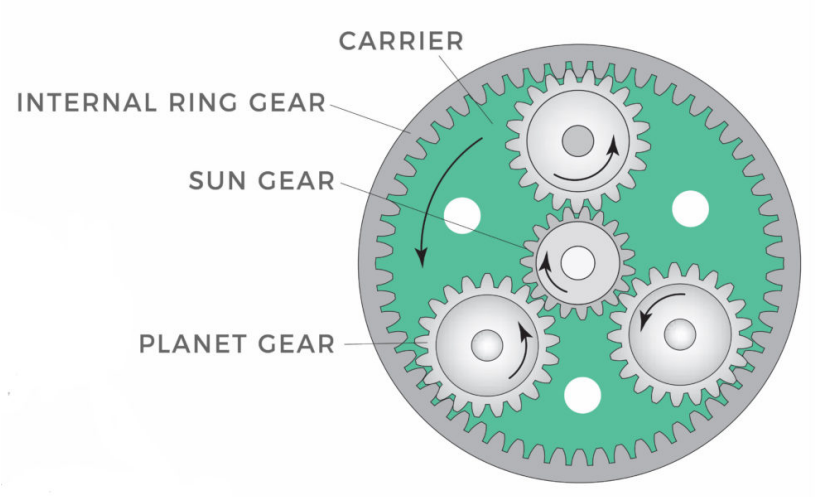


Figure 2.6: Planetary Gearbox

2.4 Joints and links design

The mechanical parts were created using a 3D modeling software then 3D printed using *Polylactic acid*-based material (PLA) 2.8, with assemblability, machinability, and the presence of dimensionally appropriate bearings taken into account.

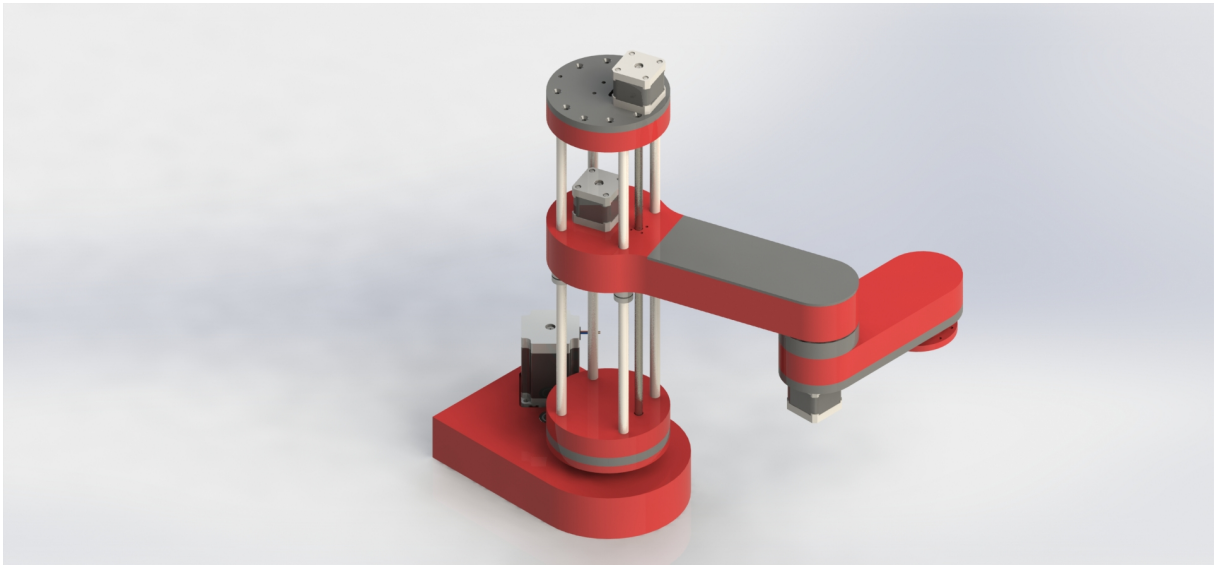


Figure 2.7: Rendered image of the SCARA prototype

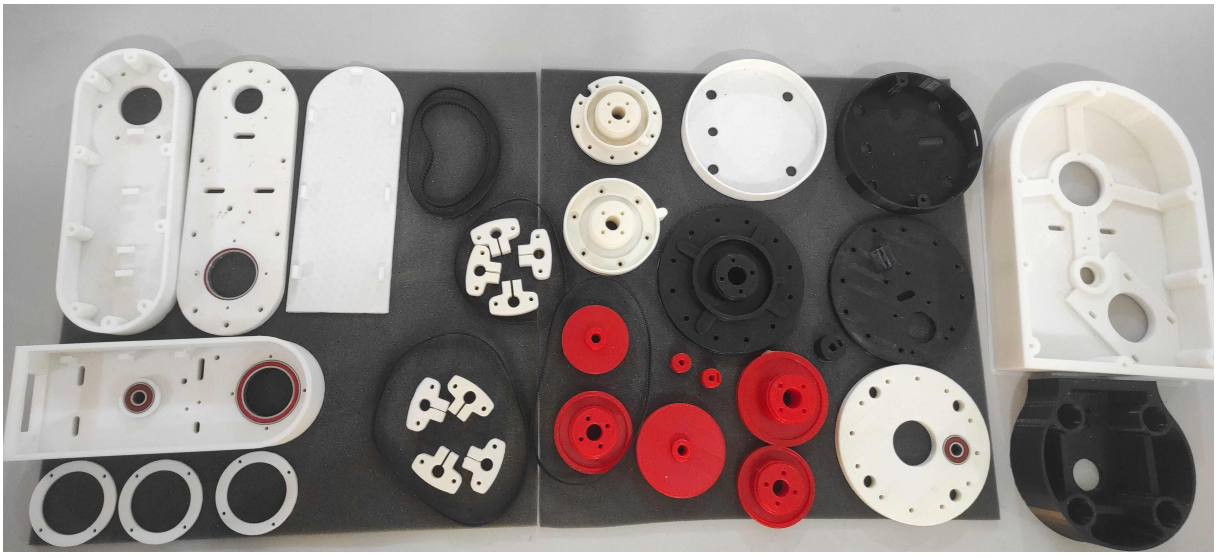
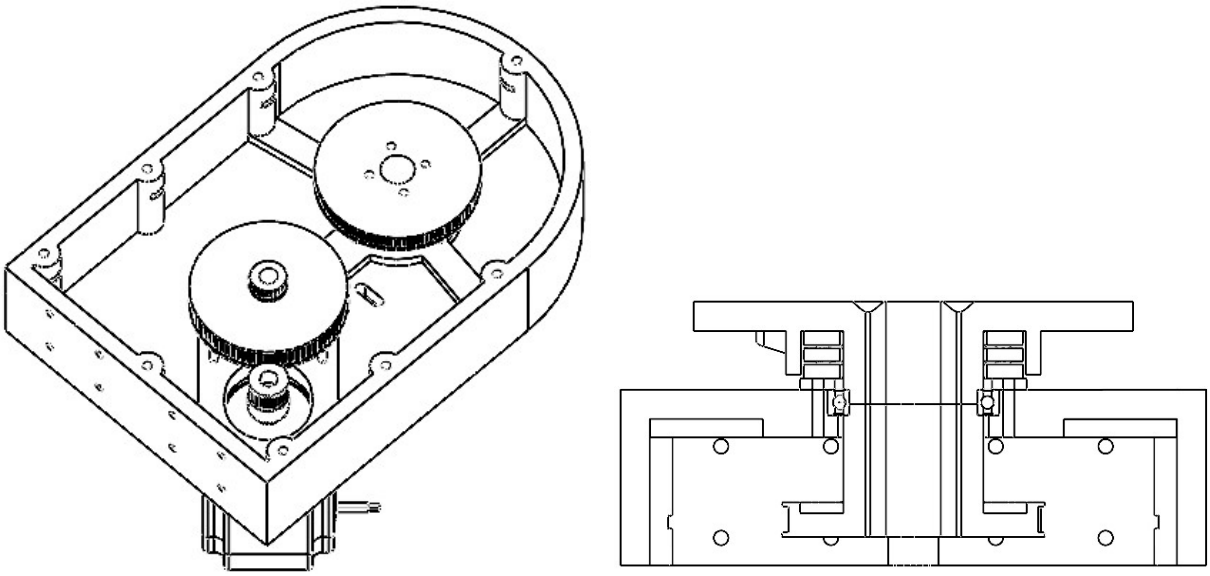


Figure 2.8: 3D printed parts at ESSAT's Fablab

The first joint achieves a 175:8 reduction ratio in two stages : the first with a 70:16 ratio driven by a *240 mm* GT3 timing belt and the second with a 110:22 ratio driven by a *300 mm* GT2 timing belt 2.9a. The reducer is controlled by a closed loop NEMA 23 with a holding torque of *2.0 N.m* in a full step configuration.
To support axial and radial efforts, an arrangement of thrust ball bearing and radial ball bearing has been used 2.9b.



(a) Perspective view of the base (b) Section view of the base

Figure 2.9: Robot base

The second joint is the prismatic joint, which is made up of an *8 mm* lead screw, four *10 mm* smooth rods and linear ball bearings so that links 1 and 2 can slide 2.10.

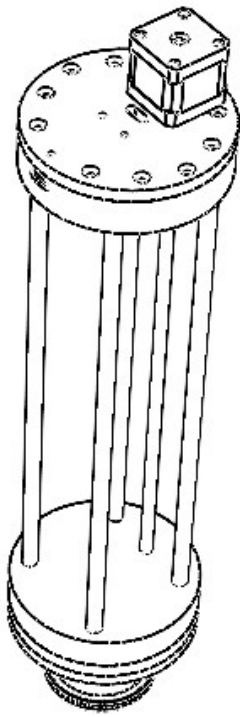
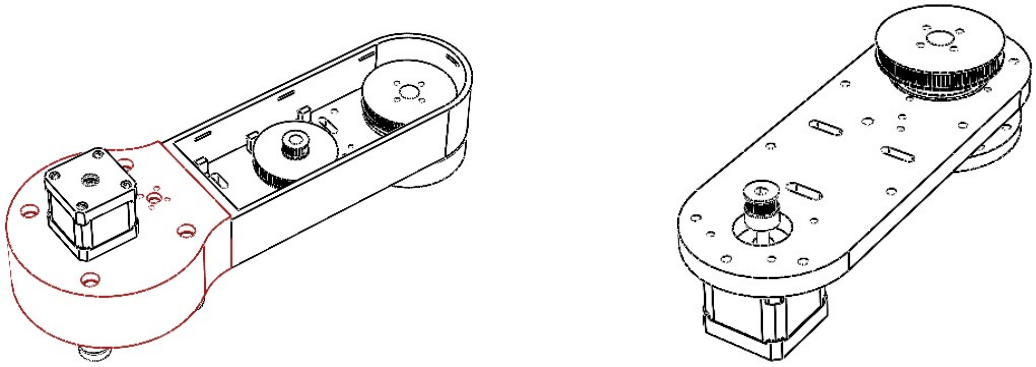


Figure 2.10: The prismatic joint

Joint three has a 16:1 reduction ratio and is actuated by 400 mm and 300 mm GT2 timing belts 2.11a. A 400mm GT2 timing belt drives the last joint that has a 9:2 reduction ratio 2.11b. Each of the last two revolute joints has a radial ball bearing and is operated by a closed loop NEMA 17 motor with a holding torque of 0.55 N.m in full step configuration.



(a) First link (b) Second link

Figure 2.11: Link one and two

The links are made of 3D printed slots with lengths of 228 mm for the first and 136,50 mm for the second 2.12. The joints are hollow so that the wires from the motors and micro switches may flow through.pulley idles can be placed to tighten the belts . Each joint may spin 161.57° in both directions in theory 2.14.

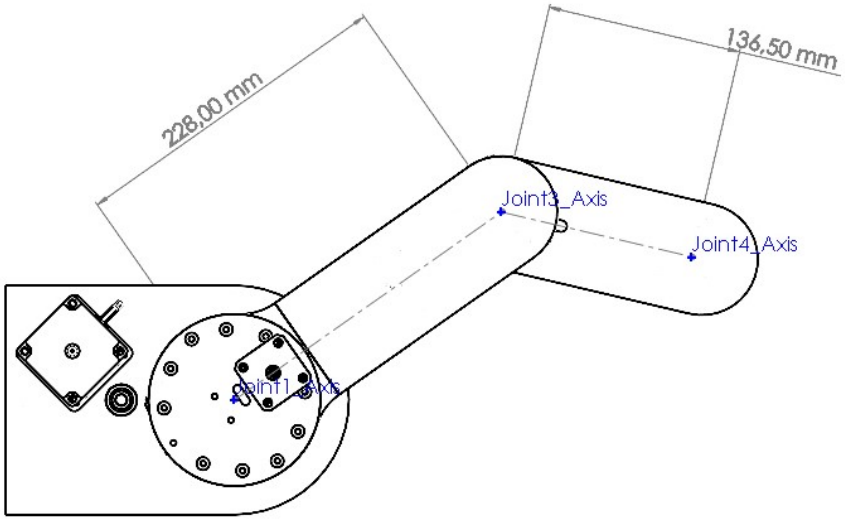


Figure 2.12: Links dimensions

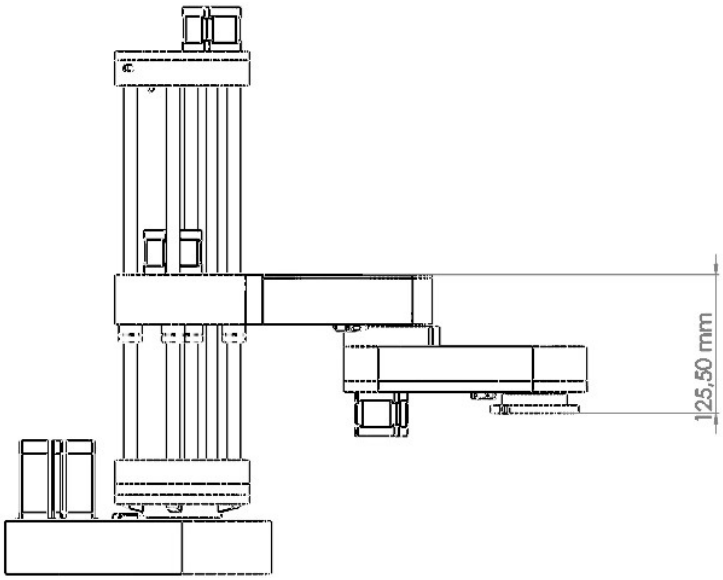


Figure 2.13: Z axis offset

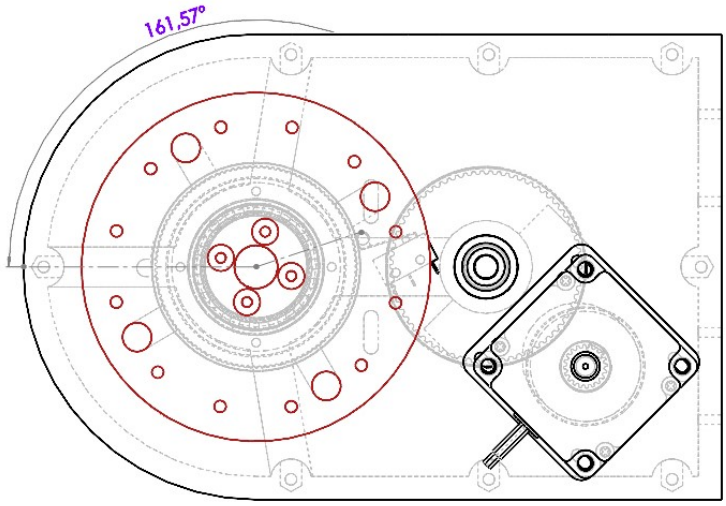


Figure 2.14: Base joint limits

2.4.1 The assembly

The robot was assembled according the the following steps :

- First, a radial ball bearing with a 35mm inner and 47mm outer diameter was placed, followed by the first thrust bearing with a 40mm inner and 60mm outside diameter. This bearing will be sandwiched between the joint coupler and the base.
- Following that, we used four M4 bolts of 55mm length to connect the pulley and top section. We need to utilise self-locking nuts and tighten them properly so that the connection is robust while still allowing for free rotation.
- The middle pulley was then fitted. This pulley is used in conjunction with the joint pulley and a 300mm GT2 belt. We utilised two 608 ball bearings, one on top and one on the bottom side of the base, to mount this pulley. The pulley was

then held in place using a 45mm M8 bolt, a washer, and a self-locking nut. The stepper motor for this joint was then mounted. A 240 mm GT3 belt was used to connect the stepper to the centre pulley. We stretched the belt as far as we could before tightening the nuts; if the belts were not tight enough, idler pulleys may be used to tighten them further. Following that, the micro switches for the joints were installed.

- The Z-axis was then put together. First, we had to mount the Z-axis bottom plate component on top of the joint coupler. On top of that, the four clamps for the smooth rods were fastened. The smooth rods were then inserted. At this stage, we needed to insert the lead screw bearing. To complete this portion, we simply put in a basic cover that will hide everything and give the robot a cleaner appearance.

- We then proceeded to assemble the robot's first link. The connection was made up of two bolted together sections. The linear bearings that will slide through the smooth rods were placed in the first section. We joined the two sections of the arm. The motor, bearing, belts, and pulleys for the third joint were then placed; the second link is connected to the preceding joint through a joint coupler.

- We continued putting together the second link. For the fourth joint, we first fitted a motor, bearing, and pulley. When we completed building both links, we attached them to the Z-axis rods.

- The Z-axis top plate, which supports the upper ends of the rods, was next prepared and installed. We installed the lead screw and used a 5mm to 8mm shaft coupler to connect the motor to the lead screw; cable management was taken into account throughout the assembly.

Nota bene: It should be noted that any type of end effector might be linked to the final joint.
When the SCARA assembly was finished, we fixed it in the centre of a piece of wood cut to the dimensions of the robot's safe zone. The pieces where 3D printed using Creality Ender 3, Creality CR-10, Flashforge Adventurer 3 3D printers at ESSAT Fablab, using 30% infill. the final result is shown in the figure 2.15

(a) (b)

Figure 2.15: The assembled SCARA prototype

2.5 Electronics

2.5.1 Initial version

Early on in the project's life-cycle, we made the initial plan on the electronics to use in our prototype according to their availability at our school and on the Algerian market as a whole.
First, to actuate the joints we chose Closed-loop stepper motor kits made by *Dewo motor* comprised of motors with integrated encoders and appropriated drivers.
For the base joint, we chose the NEMA 23 kit showcased in figure 2.16 due to the need for higher torque on that specific joint. The motor specifications are mentioned in table 2.1 alongside the tension requirement of its' particular driver.

Table 2.1: Closed loop NEMA 23 kit specifications

Model No.	Driver tension	Current
57HSE2N-D25	24-50 V	4.2 A
Step angle	Holding torque	Encoder resolution
1.8°	2.0 N.m	1000 ppr

Figure 2.16: NEMA 23 kit

As for the other 3 joints, we chose the NEMA 17 kit showcased in figure 2.17 with specifications mentioned in table 2.2 for weight reduction purposes.

Table 2.2: Closed loop NEMA 17 kit specifications

Model No.	Driver tension	Current
42HSE05N-D24	24-50 V	1.5 A
Step angle	Holding torque	Encoder resolution
1.8°	0.55 N.m	1000 ppr

Figure 2.17: NEMA 17 kit

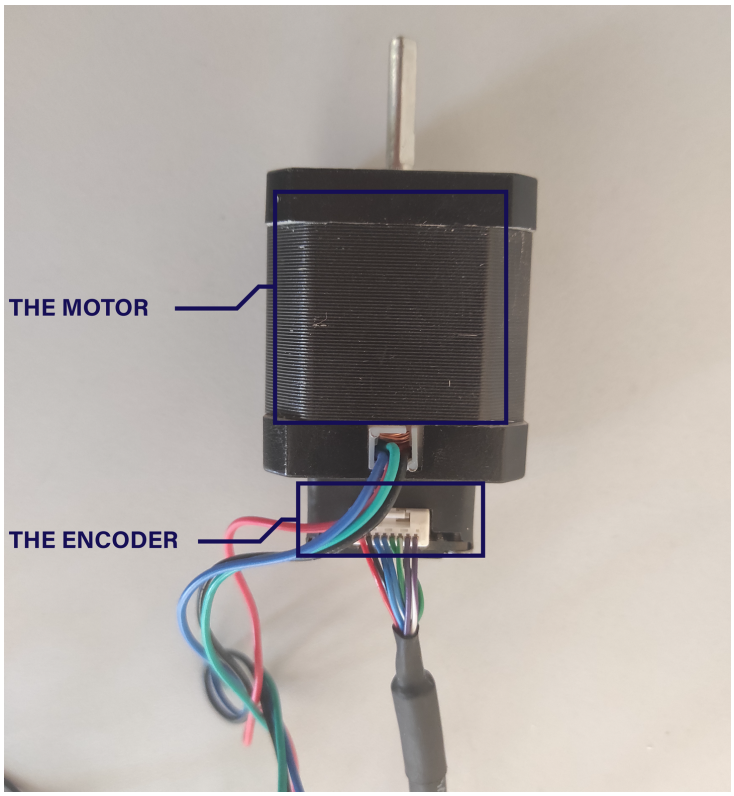


Figure 2.18: Motor with its encoder

As shown in figure 2.18, the motors are all equipped with magnetic encoders with 1000 ppr[4] resolution. These encoders detect the rotation of a cylindrical magnet fixed at the second end of the motor's shaft via a hall effect sensor.
Two square signal emitting channels are then derived with a + and - side that are 180° out of phase from each other. Ultimately, only one input signal from each channel is enough to read the information although the complementary signals are useful for troubleshooting. The A and B channels are for their part always $\pm 90°$ out of phase from each other, with the sign depending on sense of the rotation of the motor as described in figure 2.19. The speed is then proportional to the frequency of the signal.

[4]Pulse per revolution

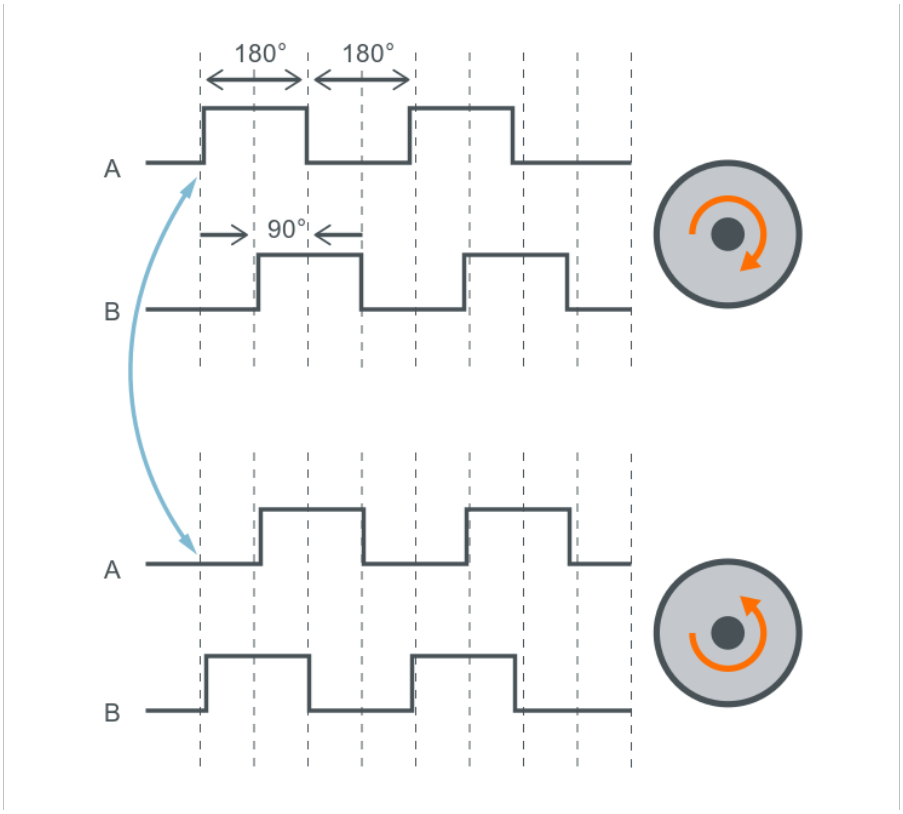


Figure 2.19: Magnetic encoder output

In order to power these motors we chose a 24V power supply displayed in figure 2.20 that would provide sufficient current for our needs.

Figure 2.20: 24V DC power supply

We also decided to use an Arduino mega 2560 shown in figure 2.21 due to its availability as well as ease of use with the availability of the *Accelstepper* library that would allow us to easily run our motors in parallel and implement the acceleration, deceleration and control speed.

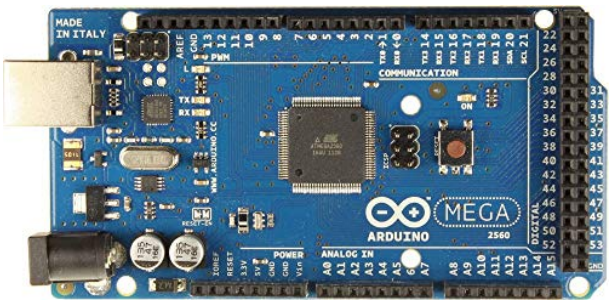


Figure 2.21: Arduino mega 2560

We also added limit switches showcased in figure 2.22 to each axis for safety purposes as well as allowing a homing process. We used the *Normally Open* mode meaning the signal was **HIGH** every time the switch was pressed and **LOW** at rest.

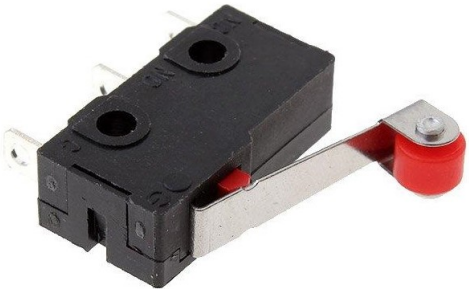


Figure 2.22: Limit switch

Finally we soldered extensions to the cables and tidied the cable management so that we arrived to the stage depicted in figure 2.23.

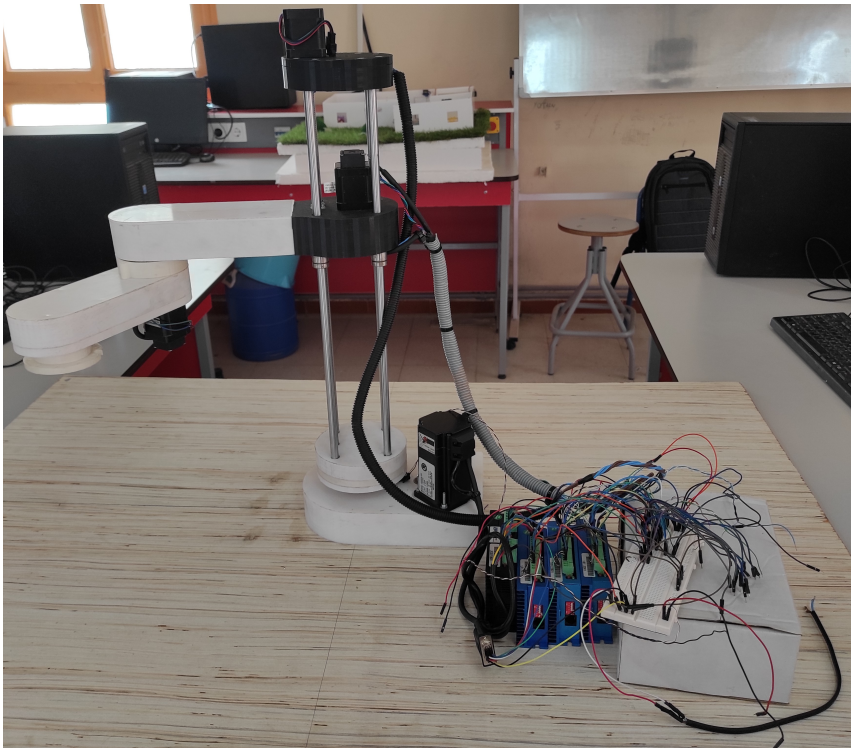


Figure 2.23: Initial version cable management

2.5.2 Final version

Along with our progress during the project's life-cycle, we encountered some hardships that forced us to deviate somewhat from our initial plans detailed in section 2.5.1.
The main issue we have faced was the total lack of documentation from the constructor *Dewo motor*, thus forcing us to blindly find ways to make the motors work as intended.
In this process, one of the NEMA 17 motors got mechanically damaged rendering it unusable and we had to replace it with an encoder-less NEMA 17 shown in figure 2.24 for the prismatic joint alongside an open loop driver showcased in figure 2.25.

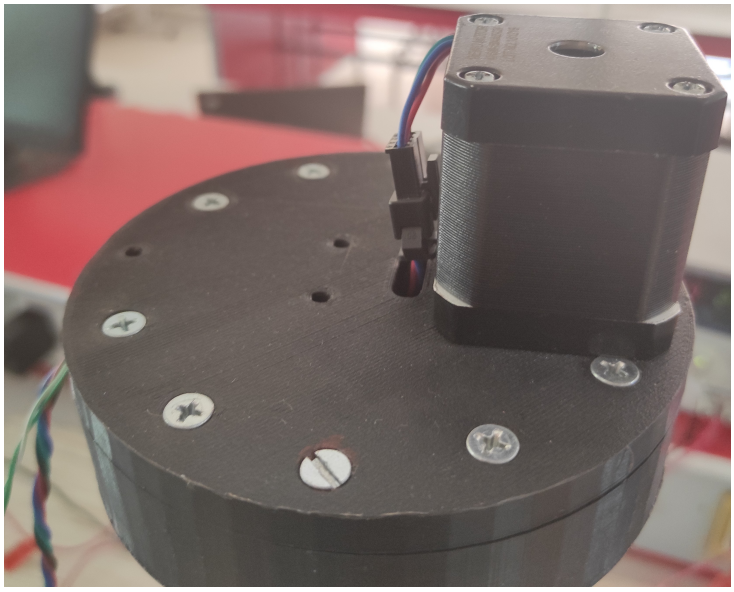


Figure 2.24: Replacement open loop motor

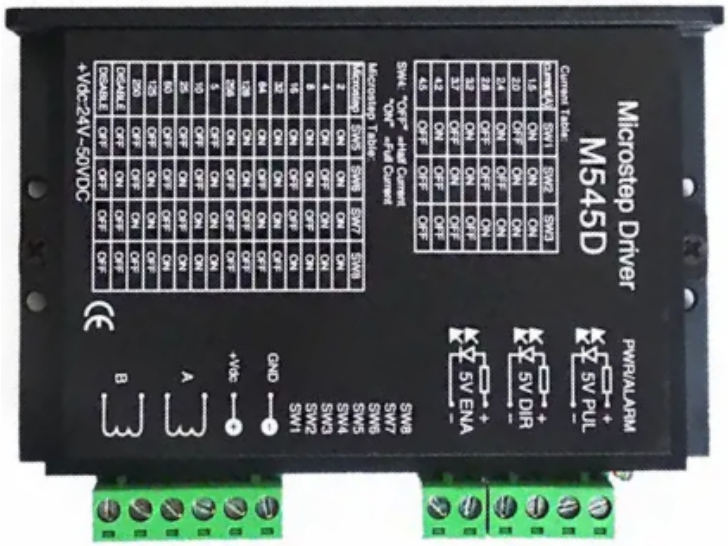


Figure 2.25: Open loop driver M545D

Furthermore, the lack of documentation left us to manually determinate the pinout of the encoder as showcased in figure 2.26 using an oscilloscope with 2 cables remaining

unknown. We also used trial and error to determine the wires of the motor as showed in figure 2.27

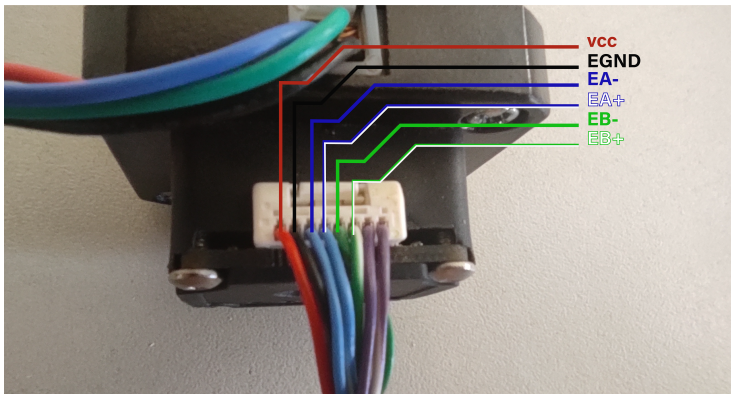


Figure 2.26: NEMA 17 Encoder pinout

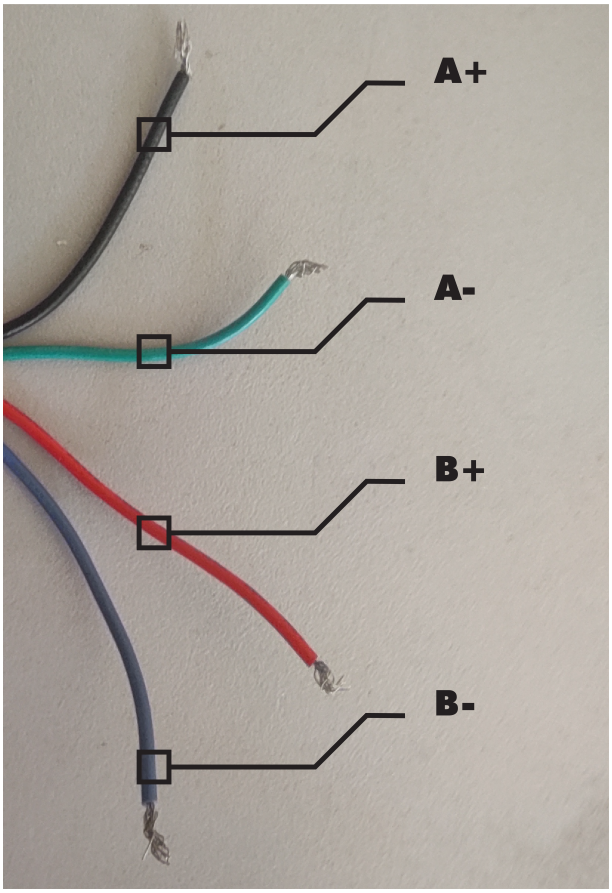


Figure 2.27: NEMA 17 Motor pinout

Additionally, the last 2 closed loop NEMA 17 motors presented unpredictable behaviour most likely due to internal regulation issues as well as signaling a '*Position ultra difference*' error that we did not manage to solve with the lack of documentation.
Thus, with the lack of time on our hands and no response from the constructor '*Dewo motor*' we decided to use the open loop drivers M545D showed in figure 2.25 and as such not using the encoder return directly (although the wiring is all accessible and operational if needed).

We also faced issues for the power supply as it was ultimately unavailable at our school, and thus we used a 30V/3A lab power supply showed in figure 2.28.

Figure 2.28: 30V/3A Lab power supply

Finally, our cable management solution showed in 2.23 turned out problematic as magnetic interference from the motor coils caused unpredictable behaviour from the end switches and as such forcing us to revert back on cable management in order for the prototype to be operational as shown in 2.29.

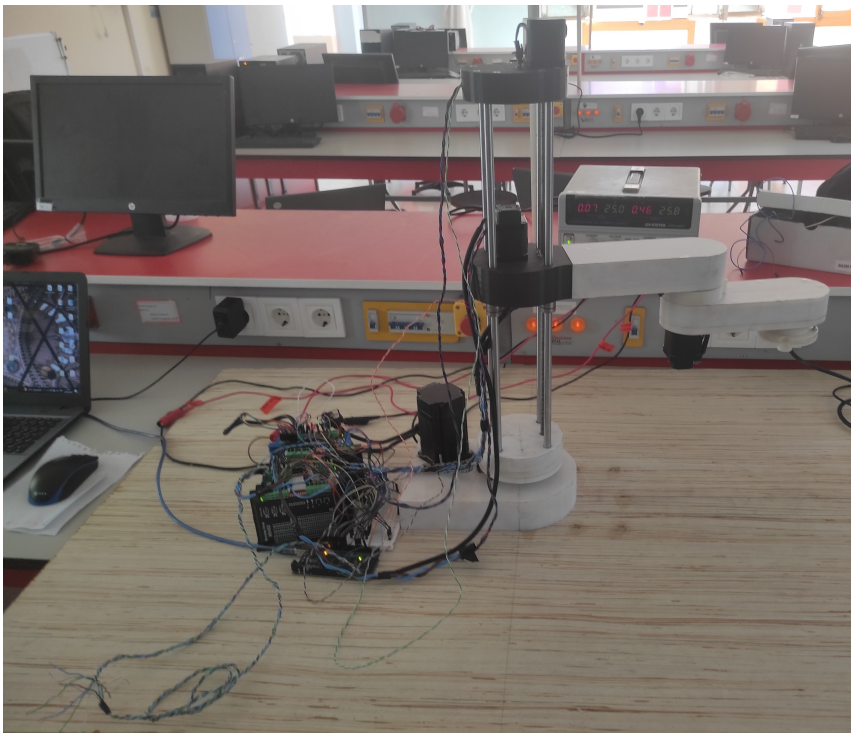


Figure 2.29: Final version cable management

2.5.3 Program

For the program of our prototype we have used two parts, one high level program written in Python in order to read and process the user command and make the necessary calculations for the forward and inverse kinematics that will be explained in more details in chapter 3 and a lower end program on Arduino in order to send the command to the motors and receive the signal from the limit switches.
We have implemented 3 main functionalities in our program :

- **Forward kinematics** : A calculation of the forward kinematics on Python to show the user the corresponding end-effector position and orientation for the inputted data and then sending the command to the Arduino via serial communication.
- **Inverse kinematics** : A calculation of the inverse kinematics on Python to show the user the corresponding joint angles for the inputted end-effector position and orientation then sending the appropriate command to the Arduino via serial communication while managing configuration duality.
- **Homing** : Allows the robot to go back to it's 0 point by using the limit switches to calculate the position. We have implemented this function as both a general way to home all axes and in separate manner for any individual axis if need be.
- **Moving the motors** : An Arduino implemented function which calculates the corresponding stepper motors steps required in order to fulfill the required command by taking account of the motor step angle, the microstepping and the reduction ratio specific to each joint ; and moves the motors accordingly with absolute positioning while implementing the speed and acceleration and deceleration.

Those functions are the basic blocs that could be used to build a complete program for this prototype with small adjustments and improvements.
However, two main issues remain in the absence complete parallelism in the motors' movements (in order for them to all start and stop at the same time) and also speed limitations due to the limitation of the frequency of the Arduino. This drawback can be mitigated with a change in the controller (such as a Raspberry Pi for example) at the cost of trading the ease of use of the *Accelstepper* library.
Nota bene: The microstepping used is 64 times for the NEMA 23 motor and 16 times for the NEMA 17 motors.

2.6 Conclusion

In this chapter we presented the design and realisation process of the SCARA prototype. During this process, we have made many changes due to difficulties met along the way from a lack of documentation, time and resources ; the last of which was the change in the base joint reduction by eliminating the intermediate stage and obtaining an overall 11:2 ratio through the addition of a custom length GT2 belt.
We have also managed to achieve one of our main design goals of keeping the center of mass close to the base as proved by our verification using a 3D modelling software.
Although this first prototype is far from perfection, with numerous flaws such as play, backlash, being open loop, desynchronization of the joint movements, inadequate cable management ; we achieved modest performances with our robot with $364.5\,mm$ maximum reach, along the $\hat{x}, \hat{y}$ plane; $323\,mm$ along the $\hat{z}$ axis. Calculated speeds of $10,227\ deg/s$ for the first joint, $10\ mm/s$ for the prismatic joint, $28.125\ deg/s$ for the third joint and $100\ deg/s$ for the fourth joint. No less than $1\ kg$ of payload tested with the maximum extension configuration alongside the maximum speed, an accuracy of about $1\ mm$ on the the $\hat{x}, \hat{y}$ plane and a repeatability of about $1\ mm$ on the the $\hat{x}, \hat{y}$ plane.
The following step will be to make the robot move and as such, the mathematical model is needed and will be presented in the next chapter.

CHAPTER

3

MODELING AND SIMULATION

3.1 Introduction

In order to manipulate objects in the space, the end-effector's configuration relative to the base must be studied, in addition to its velocities with respect to that frame. Therefore in this chapter the direct kinematics equations as well as inverse kinematics problems are derived. In the SCARA case, the direct kinematics equations are relatively easy to obtain using a geometrical approach, but a more systematic general approach was used. We introduced the trajectory planning problem with a simulation using the URDF model and Matlab alongside a simulation of the workspace of the robot.

3.2 Direct and differential kinematics

The forward kinematics of a robot refer to the calculation of the end-effector pose relative to a fixed frame given the joint variables. In order to tackle this problem we use the modified Denavit-Hartenberg convention which is a set group of rules for frame attachment rather than randomly assigning them. The forward kinematics are then derived form the knowledge of the transformation between two successive frames. The coordinate frames have to be assigned as follows :

- $\hat{z}_i$ axis coincides with joint axis i and the $\hat{z}_{i-1}$ axis coincides with joint axis $i-1$.
- The origin of frame $\{i-1\}$ is then located at the point where the common normal[1] intersects joint axis $i-1$.
- The $\hat{x}$ axis is to be chosen along the common normal pointing from the $i-1$ axis to the i axis.
- The $\hat{y}_i$ axis follows the right hand rule.

[1] The line segment that orthogonally intersects both the joint axes $\hat{z}_{i-1}$ and $\hat{z}_i$

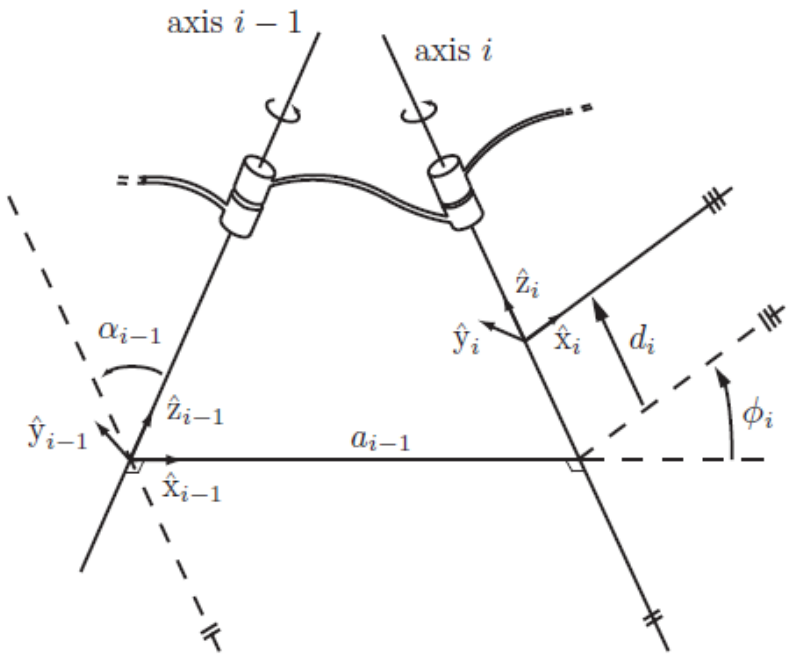


Figure 3.1: Denavit–Hartenberg kinematic parameters

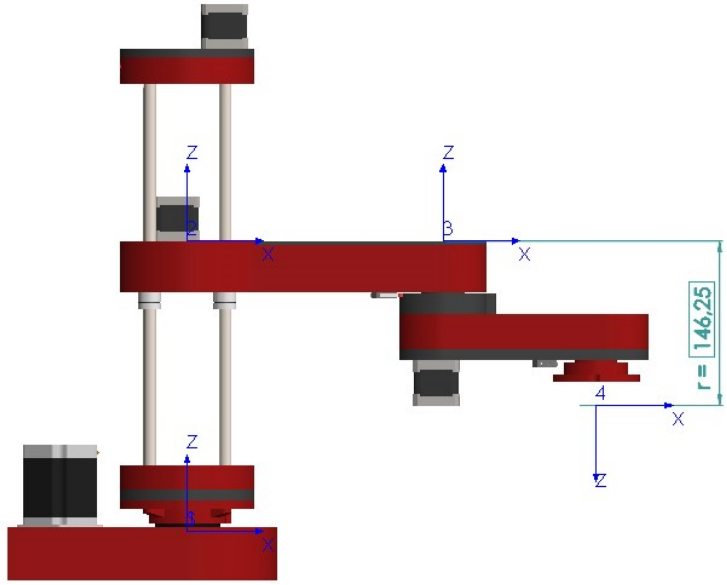
Now we can fully describe the rotation and translation using only four parameters :

- The length of the common normal, denoted by the scalar a_{i-1} ; is called the link length of link $i-1$.
- The link twist α_{i-1} is the angle from $\hat{z}_{i-1}$ and $\hat{z}_i$ measured about $\hat{x}_{i-1}$.
- The link offset d_i is the distance from the intersection $\hat{x}_{i-1}$ and $\hat{z}_i$ to the origin of the link i frame.
- The joint angle ϕ_i is is the angle from $\hat{x}_{i-1}$ to $\hat{x}_i$ measured about the $\hat{z}_i$ axis.

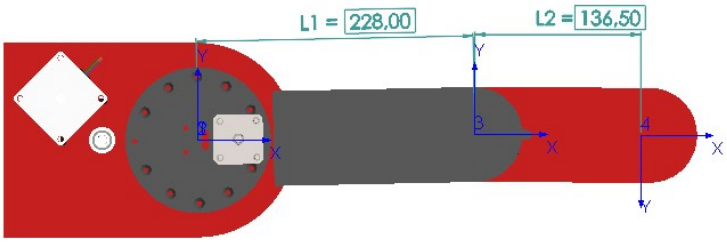
It is now possible to express the transformation between the frame i and $i-1$, by combining translation and rotation as follows :

$$T_{i-1,i} = Rot(\hat{x}_i, \alpha_{i-1})Trans(\hat{x}_i, a_{i-1})Trans(\hat{z}_i, d_i)Rot(\hat{z}_i, \phi_i) \tag{3.1a}$$

$$= \begin{bmatrix} cos\ \phi_i & -sin\ \phi_i & 0 & a_{i-1} \\ sin\ \phi_i cos\ \alpha_{i-1} & cos\ \phi_i cos\ \alpha_{i-1} & -sin\ \alpha_{i-1} & -d_i sin\ \alpha_{i-1} \\ sin\ \phi_i sin\ \alpha_{i-1} & cos\ \phi_i sin\ \alpha_{i-1} & cos\ \alpha_{i-1} & d_i cos\ \alpha_{i-1} \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.1b}$$



(a)



(b)

Figure 3.2: Frame assignment for the SCARA prototype

Now that we've covered the fundamentals, we can calculate the homogeneous transformation between the base and the last joint. The figure 3.2 represents the frame assignment for the SCARA prototype according to the *D-H* conversion where frame {0} corresponds to the base frame and it coincides with the coordinate frame {1} of the first joint. The *D-H* parameters are listed in the table 3.1 .

Table 3.1: SCARA prototype *D-H* parameters table

i	α_{i-1}	a_{i-1}	d_i	ϕ_i
1	0	0	0	ϕ_1
2	0	0	d_2	0
3	0	$L_1 = 228\ mm$	0	ϕ_3
4	π	$L_2 = 136.5\ mm$	$r = 146.25\ mm$	ϕ_4

Substituting the parameters in (3.1b) yield the following homogeneous transformation matrices

$$T_{01} = \left[\begin{array}{ccc:c} cos\ \phi_1 & -sin\ \phi_1 & 0 & 0 \\ sin\ \phi_1 & cos\ \phi_1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ \hdots & \hdots & \hdots & \hdots \\ 0 & 0 & 0 & 1 \end{array}\right] \tag{3.2a}$$

$$T_{12} = \left[\begin{array}{ccc:c} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_2 \\ \hdots & \hdots & \hdots & \hdots \\ 0 & 0 & 0 & 1 \end{array}\right] \tag{3.2b}$$

$$T_{23} = \left[\begin{array}{ccc:c} cos\ \phi_3 & -sin\ \phi_3 & 0 & L_1 \\ sin\ \phi_3 & cos\ \phi_3 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ \hdots & \hdots & \hdots & \hdots \\ 0 & 0 & 0 & 1 \end{array}\right] \tag{3.2c}$$

$$T_{34} = \left[\begin{array}{ccc:c} cos\ \phi_4 & sin\ \phi_4 & 0 & L_2 \\ sin\ \phi_4 & -cos\ \phi_4 & 0 & 0 \\ 0 & 0 & -1 & -r \\ \hdots & \hdots & \hdots & \hdots \\ 0 & 0 & 0 & 1 \end{array}\right] \tag{3.2d}$$

The forward kinematics is given by the equation 3.3 :

$$T_{04} = T_{01}T_{12}T_{23}T_{34} \tag{3.3}$$

Hence we have the final result after simplification :

$$T_{04} = \left[\begin{array}{ccc:c} cos(\phi_{134}) & sin(\phi_{134}) & 0 & L_1.cos\ \phi_1 + L_2.cos(\phi_{13}) \\ sin(\phi_{134}) & -cos(\phi_{134}) & 0 & L_1.sin\ \phi_1 - L_2.sin(\phi_{13}) \\ 0 & 0 & -1 & d_2 - r \\ \hdots & \hdots & \hdots & \hdots \\ 0 & 0 & 0 & 1 \end{array}\right] \tag{3.4}$$

The equation 3.4 represent the position and orientation of the last joint (frame $\{4\}$) with respect the frame $\{0\}$. It is clear that we only have a rotation according to the $\hat{z_0}$ axis which describes the orientation. The end-effect frame coincide the the frame $\{4\}$. We define θ as $\theta = \phi_1 + \phi_3 - \phi_4$ as it will become handy for the inverse kinematic problem.

3.3 Velocity Kinematics

The end-effector of a robot moves in a Cartesian space with a linear and angular velocity, yet that velocity is a result of the contribution of all joints velocities. The relationship between those two is the *Velocity Kinematics* that is described by a matrix termed the **Jacobian** which varies depending on the manipulator configuration. The Jacobian is one of the important characteristics of a manipulator. In fact it is useful for determining singularities, statics analysis, path planing as well as the dynamics of the robot [2]. In this section we will derive the geometric Jacobian in the space frame of the SCARA prototype and its inverse.
Let $\boldsymbol{q} = \begin{bmatrix} q_1 & q_2 \dots q_n \end{bmatrix}^T$ represent the joint variables, $\dot{\boldsymbol{p}}_{\boldsymbol{e}}(\boldsymbol{q})$ the end-effector linear velocity and the angular velocity $\boldsymbol{\omega_e(q)}$. Hence, $\boldsymbol{v_e}$ is the (6×1) vector defined as $\boldsymbol{v_e} = \begin{bmatrix} \dot{p}_{ex} & \dot{p}_{ey} & \dot{p}_{ez} & \omega_{ex} & \omega_{ey} & \omega_{ez} \end{bmatrix}^T$

$$\boldsymbol{v_e} = \begin{bmatrix} \dot{\boldsymbol{p}}_{\boldsymbol{e}}(\boldsymbol{q}) \\ \dots \\ \boldsymbol{\omega_e(q)} \end{bmatrix} = \begin{bmatrix} \boldsymbol{j_p(q)} \\ \dots \\ \boldsymbol{j_o(q)} \end{bmatrix} \dot{\boldsymbol{q}} = \boldsymbol{J(q)}\dot{\boldsymbol{q}} \tag{3.5}$$

The $\boldsymbol{J(q)}$ in 3.5 is a $(6 \times n)$ matrix that represents the geometric Jacobian and n is the number of joints. Depending on the joint type the Jacobian can be partitioned into (3×1) columns vectors $\boldsymbol{j_{pi}}$ and $\boldsymbol{j_{oi}}$ where :

$$\boldsymbol{j_{pi}} = \begin{bmatrix} \boldsymbol{z_{i-1}} \\ \boldsymbol{0} \end{bmatrix} \qquad \text{for prismatic joints} \tag{3.6a}$$

$$\boldsymbol{j_{oi}} = \begin{bmatrix} \boldsymbol{z_{i-1}} \times (\boldsymbol{p_e} - \boldsymbol{p_{i-1}}) \\ \boldsymbol{z_{i-1}} \end{bmatrix} \qquad \text{for revolute joints} \tag{3.6b}$$

Where :

- $\boldsymbol{z_{i-1}}$ is given by the third column of the rotation matrix $\boldsymbol{R}_{0,i-1}$
- $\boldsymbol{p_{i-1}}$ is given by the first three elements of the fourth column of the transformation matrix $\boldsymbol{T}_{0,i-1}$

For the SCARA manipulator prototype :

$$\boldsymbol{J(q)} = \begin{bmatrix} \boldsymbol{z_1} \times (\boldsymbol{p_e} - \boldsymbol{p_1}) & \boldsymbol{z_2} & \boldsymbol{z_3} \times (\boldsymbol{p_e} - \boldsymbol{p_3}) & \boldsymbol{z_4} \times (\boldsymbol{p_e} - \boldsymbol{p_4}) \\ \boldsymbol{z_1} & \boldsymbol{0} & \boldsymbol{z_3} & \boldsymbol{z_4} \end{bmatrix} \tag{3.7}$$

Nota Bene : In literature the coordinate frame $\{0\}$ is to be assigned to joint 1, we have assigned frame $\{1\}$ to joint 1 which justify the index shifting in the matrix 3.7 i.e. starting with $\boldsymbol{z_1}$ instead of $\boldsymbol{z_0}$.
The various vectors can be computed from the manipulator direct kinematics :

$$\begin{gathered} \boldsymbol{p_1} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, \boldsymbol{p_2} = \begin{bmatrix} 0 \\ 0 \\ d_2 \end{bmatrix}, \boldsymbol{p_3} = \begin{bmatrix} L_1 cos(\phi_1) \\ L_1 sin(\phi_1) \\ d_2 \end{bmatrix}, \\ \boldsymbol{p_4} = \boldsymbol{p_e} = \begin{bmatrix} L_1 cos(\phi_1) + L_2 cos(\phi_{13}) \\ L_1 sin(\phi_1) + L_2 sin(\phi_{13}) \\ d_2 - r \end{bmatrix} \end{gathered} \tag{3.8}$$

$$\boldsymbol{z_1} = \boldsymbol{z_2} = \boldsymbol{z_3} = \begin{bmatrix}0\\0\\1\end{bmatrix}, \boldsymbol{z_4} = \begin{bmatrix}0\\0\\-1\end{bmatrix} \tag{3.9}$$

Substituting 3.8 and 3.9 in 3.7 yield :

$$\boldsymbol{J(q)} = \begin{bmatrix} -L_1 sin(\phi_1) - L_2 sin(\phi_{13}) & 0 & -L_2 sin(\phi_{13}) & 0 \\ L_1 cos(\phi_1) + L_2 cos(\phi_{13}) & 0 & L_2 cos(\phi_{13}) & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & -1 \end{bmatrix} \tag{3.10}$$

In the equation 3.10 it is obvious to the reader that there is no rotation about $\hat{x}, \hat{y}$ axis, thus the (4×4) Jacobian can be expressed by eliminating the 4^{th} and 5^{th} rows:

$$\boldsymbol{J(q)_{4\times4}} = \begin{bmatrix} -L_1 sin(\phi_1) - L_2 sin(\phi_{13}) & 0 & -L_2 sin(\phi_{13}) & 0 \\ L_1 cos(\phi_1) + L_2 cos(\phi_{13}) & 0 & L_2 cos(\phi_{13}) & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & -1 \end{bmatrix} \tag{3.11}$$

The matrix 3.11 allow us to identify poses which leads to an inability of end-effector instantaneous movement in one or more directions, such inability is termed as a ***kinematic singularity***. Mathematically it is the set of $\boldsymbol{q}$ that causes the matrix 3.10 to be rank deficient i.e. $\boldsymbol{det(.)} = 0$, in our case the determinant $\boldsymbol{det(J(q)_{4\times4})}$ is fully depended on $sin(\phi_3)$ that means we only have a singularity at $\phi_3 = 0$. Due to the joint physical limit, other cases are not taken into account. This singularity is a **Boundary singularity** and it does not represent a true drawback. [2]. It is interesting to consider the following points :

- Another form of the Jacobian is the analytical Jacobian, that generally differs from the geometric one, and that can be derived by direct differentiation of the direct kinematics function with respect to the joint variables.
- A linear mapping between the joint velocity space and the operational velocity space is represented by the differential kinematics equation. This fact shows that the differential kinematics equation might be used to solve the inverse kinematics problem by using the inverse of the Jacobian matrix [2].

$$\dot{\boldsymbol{q}} = \boldsymbol{J^{-1}_{4\times4}(q)v_e}$$

$$\boldsymbol{J^{-1}_{4\times4}(q)} = \begin{bmatrix} \dfrac{cos(\phi_{13})}{L_1 sin(\phi_3)} & \dfrac{sin(\phi_{13})}{L_1 sin(\phi_3)} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ \dfrac{-L_2 cos(\phi_{13}) + L_1 cos(\phi_1)}{L_2 L_1 sin(\phi_1)} & \dfrac{-L_2 sin(\phi_{13}) + L_1 sin(\phi_1)}{L_2 L_1 sin(\phi_1)} & 0 & 0 \\ \dfrac{-cos(\phi_1)}{L_2 sin(\phi_3)} & \dfrac{-sin(\phi_1)}{L_2 sin(\phi_3)} & 0 & -1 \end{bmatrix} \tag{3.12}$$

- Using the principle of conservation of power, a static analysis of the robot manipulator can be done using the Jacobian [3].

- The manipulability ellipsoid describes the ease with which the robot can move in different directions. The eigenvectors of $\boldsymbol{JJ^T}$ define the primary axes of the manipulability ellipsoid for a Jacobian $\boldsymbol{J}$, and the corresponding lengths of the principal semi-axes are the square roots of the eigenvalues [3].

3.4 Inverse Kinematics

In section 3.2, we have discussed the *direct kinematics* problem that maps joint variables $\boldsymbol{q}$ to the end-effector pose $\boldsymbol{p_e}$. To perform the desired motion in the operational space, we have to determine the corresponding joint angles in the joint space from the end-effector position and orientation. This problem is known as *inverse kinematics* and it is a complex problem with respect to direct kinematics yet with high importance. The reason for its complexity goes back to the non-linearity of the problem, the existence of multiple solutions or infinite ones, or even non admissible solutions i.e. physically impossible. The solution of this problem requires either an algebraic or geometric intuition to get a close form solution. However, in the case of the non existence or difficulty to find the close form solution, numerical algorithms could be used, like the use of the inverse Jacobian [2].
For the SCARA manipulator, it is relatively easy to get a close form solution. The workspace in the $\hat{x}, \hat{y}$ plan is an annulus, according to the desired end-effector position there may exist one, two or no solution depending on if the x, y lies on the boundary, interior or the exterior of the workspace. In this section we will derive the inverse kinematic equations using the algebraic intuition.
Let $\boldsymbol{p_e} = \begin{bmatrix} p_x & p_y & p_z & \vartheta \end{bmatrix}^T$ be the desired end-effector position and orientation, we know from 3.4 that :

$$p_x = L_1.cos\ \phi_1 + L_2.cos(\phi_{13}) \tag{3.13a}$$
$$p_y = L_1.sin\ \phi_1 + L_2.sin(\phi_{13}) \tag{3.13b}$$
$$p_z = d_2 - r \tag{3.13c}$$
$$\vartheta = \theta \tag{3.13d}$$

Combining 3.13a and 3.13b :

$$\begin{aligned} p_x^2 + p_y^2 &= L_1^2 + L_2^2 + 2L_1L_2(cos(\phi_1)cos(\phi_{13}) + sin(\phi_1)sin(\phi_{13})) \\ &= L_1^2 + L_2^2 + 2L_1L_2cos(\phi_3) \end{aligned} \tag{3.14}$$

From 3.14 we have

$$cos(\phi_3) = \frac{p_x^2 + p_y^2 - L_1^2 - L_2^2}{2L_1L_2} = c_3 \tag{3.15}$$
$$sin(\phi_3) = \pm\sqrt{1 - c_3} = s_3 \tag{3.16}$$

So :

$$\phi_3 = atan2(s_3, c_3) \tag{3.17}$$

Manipulation equation 3.13a and 3.13b by developing $cos(\phi_{13})$ and $sin(\phi_{13})$ yield :

$$\begin{aligned} p_x &= (L_1 + L_2c_3)cos(\phi_1) - L_2s_3sin(\phi_1) \\ p_y &= L_2s_3cos(\phi_1) + (L_1 + L_2c_3)sin(\phi_1) \end{aligned} \tag{3.18}$$

Using Kramer's rule to solve the system 3.18 gives :

$$cos(\phi_1) = \frac{(L_1 + L_2c_3)p_x - L_2s_3p_y}{p_x^2 + p_y^2} = c_1 \tag{3.19a}$$

$$sin(\phi_1) = \frac{(L_1 + L_2c_3)p_y - L_2s_3p_x}{p_x^2 + p_y^2} = s_1 \tag{3.19b}$$

$$\phi_1 = atan2(s_1, c_1) \tag{3.20}$$

From 3.13d we know that :

$$\vartheta = \theta = \phi_1 + \phi_3 - \phi_4 \tag{3.21}$$

Then :

$$\phi_4 = -\vartheta + \phi_1 + \phi_3 \tag{3.22}$$

And

$$d_2 = p_z + r \tag{3.23}$$

Now we have the vector of joint variables $\boldsymbol{q} = \begin{bmatrix}\phi_1 & d_2 & \phi_3 & \phi_4\end{bmatrix}^T$ given the end-effector position and orientation $\boldsymbol{p_e}$. Notice how we have two type solutions depending on the sign of 3.16, it corresponds to the elbow-up and elbow-down configurations see the figure 3.3. Those two configurations have been used to avoid self collision of the prototype.

Nota Bene: the $atan2(y, x)$ function allows us to determine the quadrant of the angle.

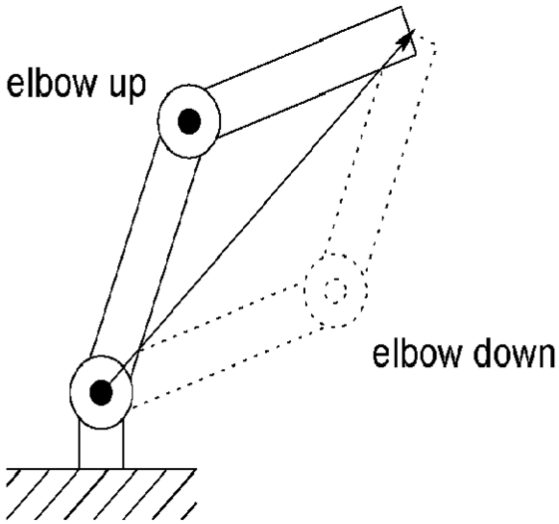


Figure 3.3: Elbow-up and elbow-down configurations

3.5 Trajectory planning

In order for the manipulator to move from point A to point B, a trajectory has to be determined from the infinite possibilities between the two points in order to minimize time, energy consumption or manage different constraints such as joint limits, collision prevention etc.
This subject is called trajectory planning and is a vast portion of robotics that we will briefly discuss in this section. We will also use the example given on Mathworks' website [16] of the trajectory planning of a 7 DOF manipulator to illustrate the different points in this section.
The basic principle is rather simple to understand, The initial and final points of the movement are needed alongside eventually some intermediate points called *waypoints* that help guide the general trajectory. The next step is to generate the trajectory alongside the resolution of the inverse kinematic problem in order to determine the joint position needed from the actuators to produce that movement.
The first major point to discuss is the use of Joint space or Task space for the interpolation of the trajectories [17].

- **Task space :** That is the most commonly used cartesian coordinates (x, y, z) alongside the orientations (ψ_x, ψ_y, ψ_z) around the (x, y, z) axes respectively. The trajectory generation being in the task space is more intuitive to the user rendering it easier to master and be used to properly plan the trajectory and easily avoid collisions. However, it also comes with the downside of more calculations to be made as inverse kinematics have to be calculated at every step [16].

- **Joint space :** Describing the coordinates relative to the actuators (in our case (q_1, q_2, q_3, q_4)). The main reason to use this approach is that the inverse kinematics have to be applied to the waypoints only, making it much faster for computation with the interpolation and command parts all done in the joint space directly. It also comes with smoother actuator motion compared to the task space but is lacking in terms of collision planning and overall trajectory visualisation [16].

We can easily verify this by plotting the trajectories given by task and joint space generations :

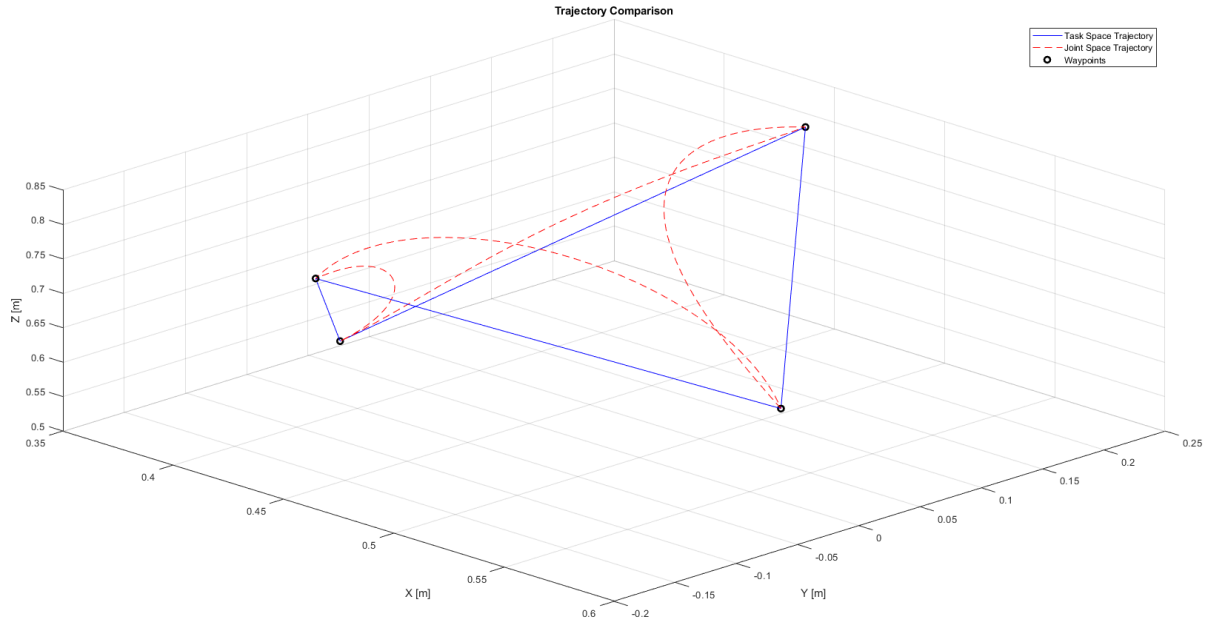


Figure 3.4: Comparison of joint space and task space generated trajectories

We can also clearly see the influence of this choice on the smoothness of joint movement :

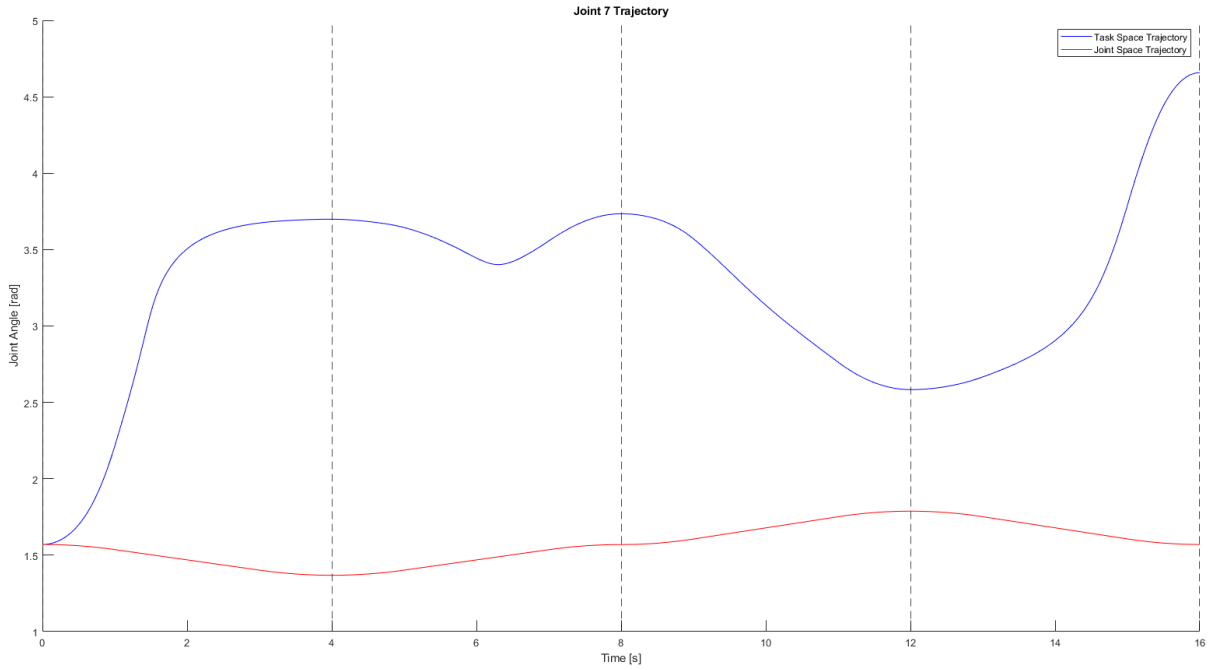


Figure 3.5: Comparison of joint space and task space joint smoothness

We must also note that the joint space generation was more than 31 times faster than the task space generation on this particular example.
The other major point in terms of trajectory planning is the type of trajectory generated, multiple ways exist each with their advantages and disadvantages.
In all cases, the general equation of such movement is as follows [17] :

Let there be :

$$\begin{cases} q \text{ a generalized coordinate.} \\ q^i \text{ the initial point.} \\ q^f \text{ the final point.} \\ D = q^f - q^i \\ r(t) \text{ the trajectory equation with } r(0) = 0 \text{ and } r(t_f) = 1. \end{cases} \tag{3.24}$$

$$\begin{cases} q_t = q^i + r(t).D & 0 \leq t \leq t_f \\ \dot{q}_t = \dot{r}(t).D \end{cases} \tag{3.25}$$

The determination of the trajectory equation is subject to different methods :

- **Linear :** The simplest method of resolution is to determine a polynomial solution to the problem. This problem has two inner constraints (the position at the beginning and the end), so a polynomial of degree 1 would fit. This solution, whilst possible in theory cannot be translated to a real robot with the speed discontinuities it would imply that the mechanical parts wouldn't be able to follow [17]. It is therefore in most cases a rejected possibility.

- **Cubic :** Imposing constraints on speed alongside the 2 previous ones amounts to a polynomial of degree 3. This solution is possible to function practically but still leaves acceleration discontinuities which isn't very recommended for less robust robots [17].

- **Quintic :** Adding constraints to the accelerations render the polynomial of degree 5 and as such become way more stable and predictable for most robotics manipulators [17].

- **Bang-Bang :** It consists of implementing the acceleration in form of 2 steps of equal amplitude and opposite signs in order to minimize the length of the movement of the robot with respect of time.

- **Trapezoidal :** An evolution of the Bang-Bang method by pushing the speed and acceleration to the saturation in order to truly minimize the time of the movement and resulting in a trapeze-shaped speed graph. It is also widely use due to its' ease of recognition and use.

- The introduction of waypoints requires these methods to be applied between each consecutive waypoints in order to form the complete trajectory. ***Spline*** is the method that generalises this approach by applying the cubic method between all of the waypoints while assuring the continuity of the speed and acceleration.

While the last 3 mentioned methods are used in most industrial applications, it would be very hard to obtain such results with our prototype as the discontinuities in speed and/or acceleration or the complexity of the calculations would be near-impossible to implement.

3.6 Simulation

3.6.1 Workspace simulation

One of the key factors in judging a robot arm's effectiveness is its workspace, which is the total domain in the 3D space that the end effector can reach in at least one configuration of the arm. Multiple methods exist to determinate the workspace of a robot manipulator : algebraic, geometric, algorithmic [18]. Each method having its advantages and inconvenients. In our case, we simulated all possible points reached by the end effector by using the forward kinematics equations and then plotting the results on Matlab (the code can be found in B).
SCARA robots being planar robots, meaning that the top view is sufficient to describe the full workspace of it. As with a simple translation along the $\hat{z}$ axis, the total workspace is a 3D extension of the planar workspace seen from the top view. Therefore, our study will be focused first on establishing that side. The resulting 3D workspace will be the extrusion of that 2D profile alongside the $\hat{z}$ axis. The defining factors are in our case the length of the 2 segments composing the arm, the angle limits of the two joints and the translational limit.
We first determined the rough reach the robot was supposed to have to be around 40 cm. Afterwards we tried out multiple combinations of arm lengths to envision the influence of the variation of the L_2/L_1 and θ_2/θ_1 ratios with a constant reach.

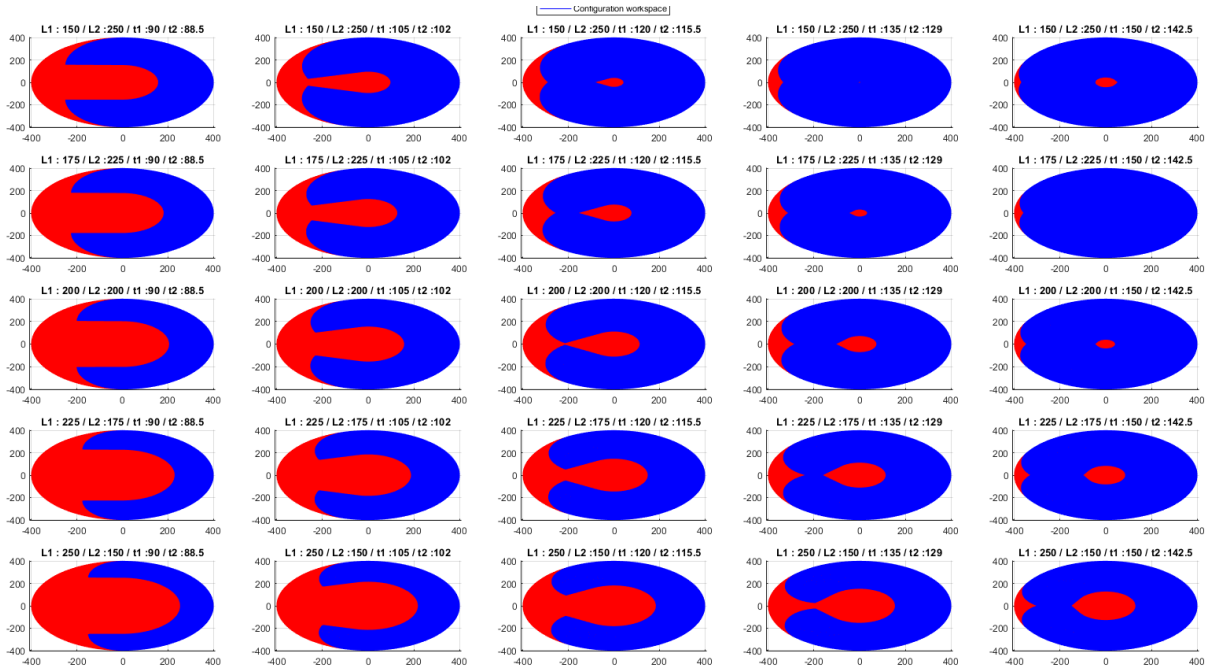


Figure 3.6: Impact of the L_2/L_1 and θ_2/θ_1 ratios on the total workspace

This simulation showed us that a $L_2/L_1 > 1$ is the best direction to have as it minimizes the dead zones. However, the aforementioned constrains of the realization obliged us to have a reach of 364.5 mm with $L_1 = 228$ mm and $L_2 = 136.5$ mm, $i.e$ $L_2/L_1 = 0.5987$. Therefore, the biggest influence possible was to play on the joint angle limits where we tested out multiple variations of the max angle as showcased with the following figure :

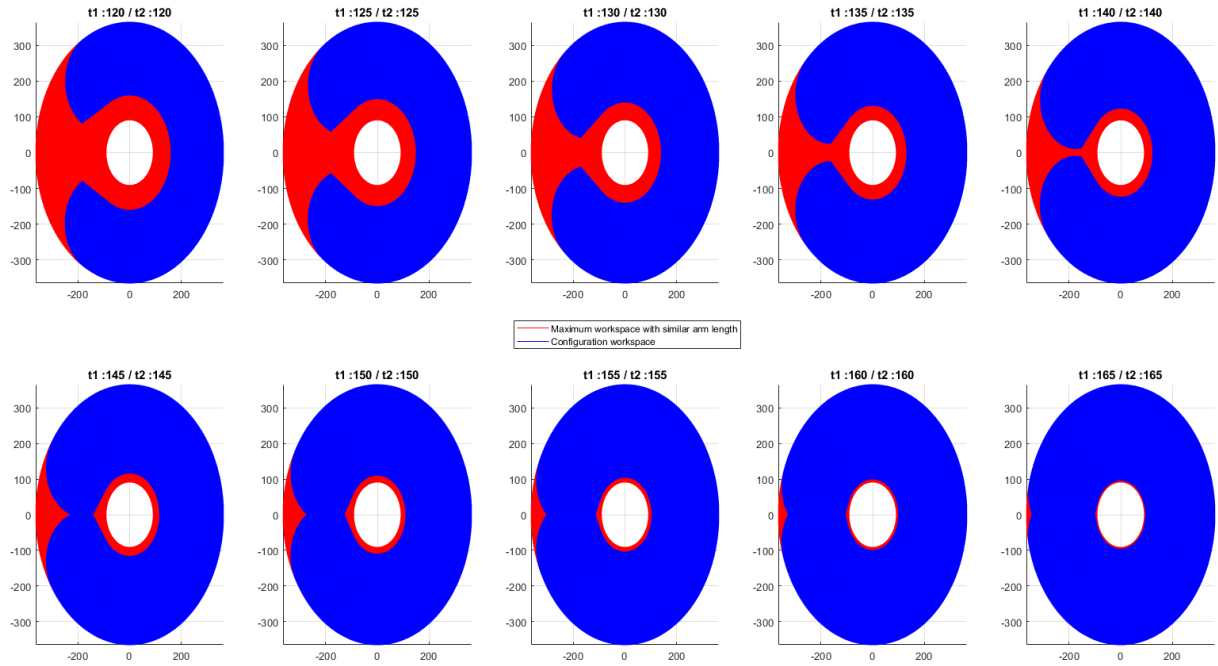


Figure 3.7: Impact of the maximum joints angle on the total workspace

The max angle being a compromise between maximizing workspace and a few cable management issues, we decided on a compromise at 161.74° that brings us a small centered dead zone of a diameter around 160 mm which is not an issue considering that area is occupied by the $\hat{z}$ axis mount structure. This compromise therefore appears to be good in theory.

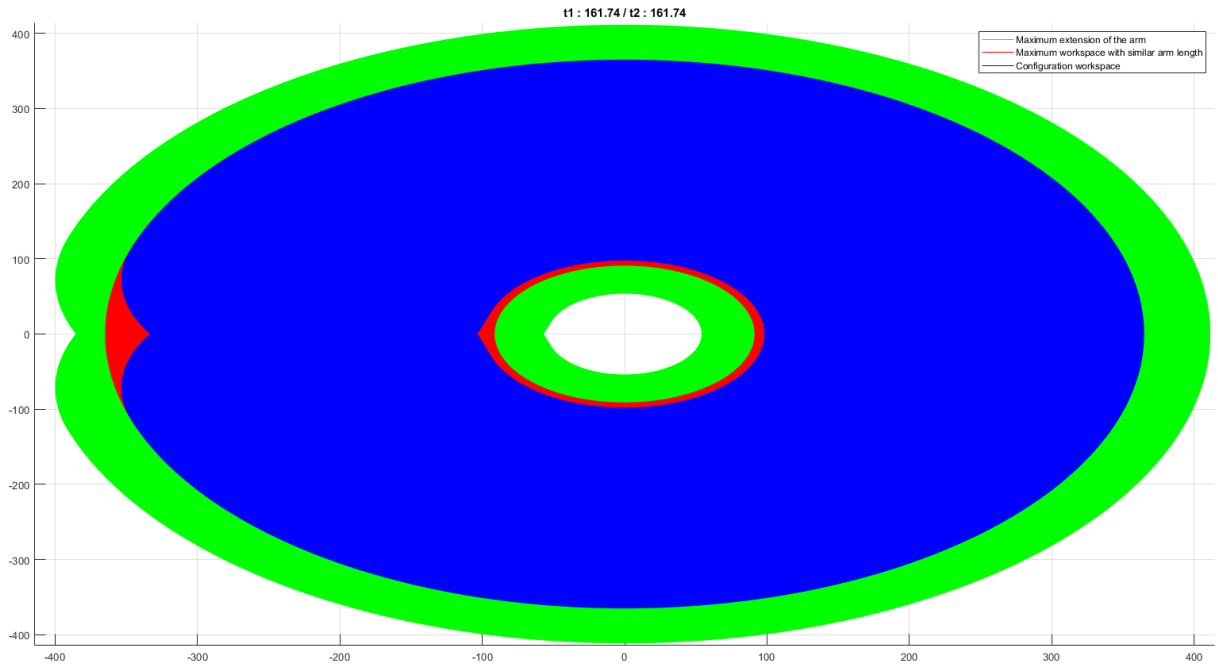


Figure 3.8: Total workspace, reachable zone and deadzone area

3.6.2 Trajectory planning simulation

The first requirement for the simulation of our manipulator is the creation of the URDF file (*Unified Robot Description Format*) that is an XML file generated by the 3D modelling software that describes a robot. It allows for the kinematic and dynamic description of the robot, it's visual representation and collision model.
The URDF description then allows us to get our model in a simulation environment in Matlab Simulink (Model and solver parameters can be found in C).
It generated the basic structure according to the URDF and we added position, speed, acceleration and torque sensing to have a full grasp at our manipulator's behaviour as shown in 3.9.

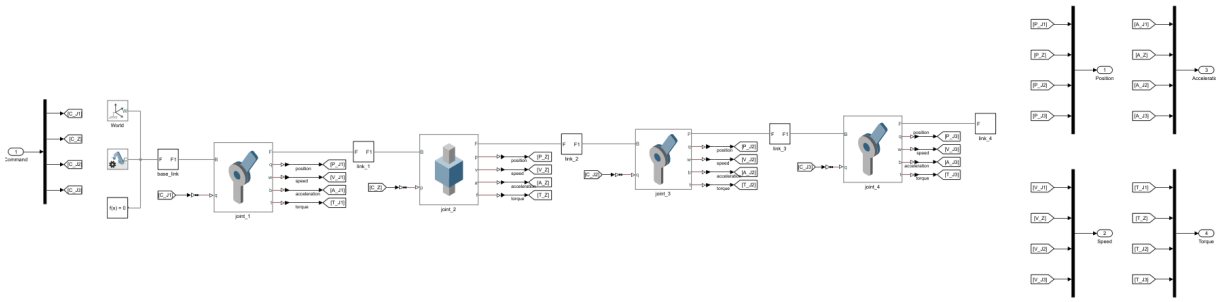


Figure 3.9: Robot model in Simulink

We then added two possibilities for the trajectory of our robot. First the simple command via signal builder and then the trajectory generation bloc with a switch to choose between what mode is required.
In the simulation 3.10, we chose to generate the trajectory in the task space to have better control over it's shape and avoid any collisions and/or getting close to the singularities. That also meant a compromise on time as with the 10^{-5} precision parameter chosen alongside the Levenberg-Marquardt discrete inverse kinematics solver, each simulation took approximately 10min (for a 10s total time simulated). We then transmitted the configuration to the robot model referenced above 3.9 and extracted all the previously cited parameters alongside the coordinates of the end effector in the task space via a forward kinematics bloc.

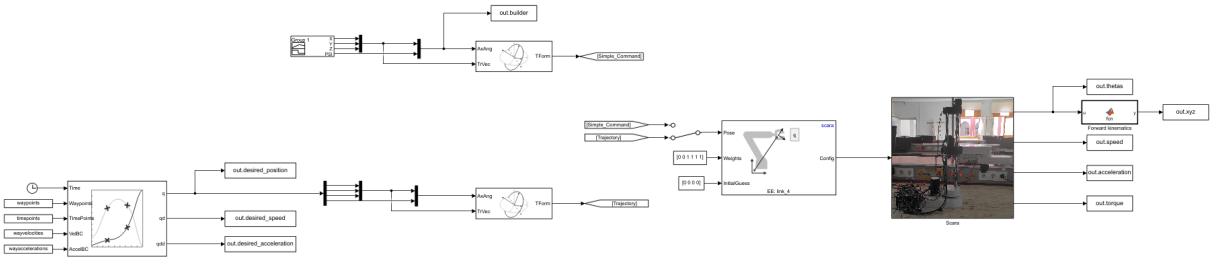


Figure 3.10: Trajectory planning simulation

We have also tried out both the cubic and quintic generation in order to compare their results with an emphasis on the quintic case as acceleration continuity is needed with our 3D printed prototype to limit the mechanical backlash [17].
The results can be viewed directly with an animation of the model as shown in 3.11.

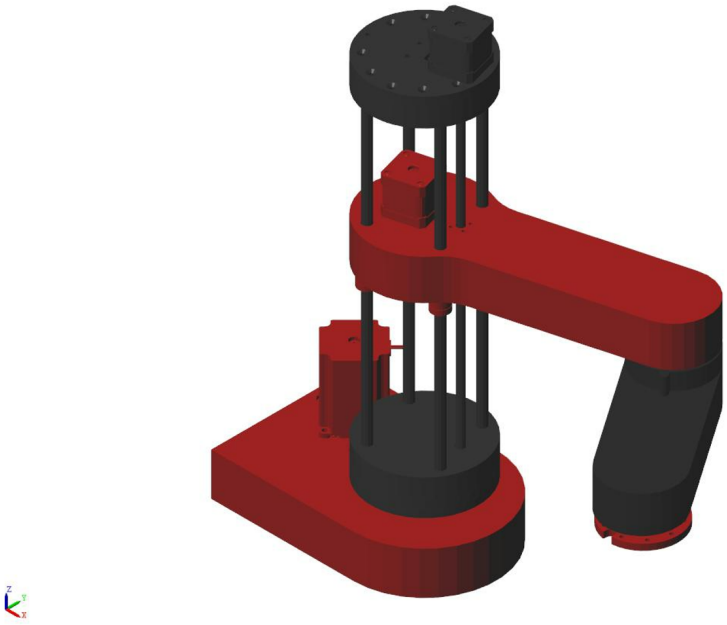


Figure 3.11: Simulation output

The 2D projection of the input trajectory on the (x, y) plane that is required to be followed back and forth is depicted in figure 3.12. We can clearly see that the trajectory was closely followed with a small error and linear paths between the waypoints as expected.
Nota Bene : It is important to note that we only managed to simulate the trajectory planning with the Cartesian coordinates.

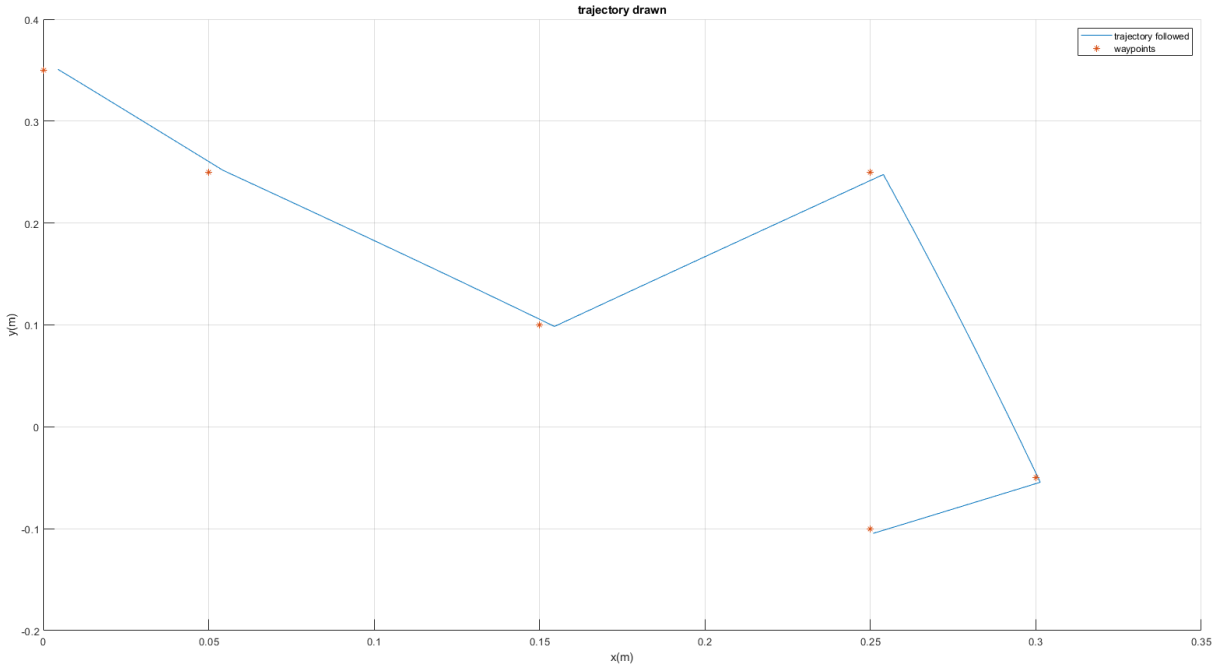


Figure 3.12: Input trajectory and resultant following

In figure 3.13, we can see that the position following is very close on all axes to the reference on all 3 axes with its' distinct polynomial shape.

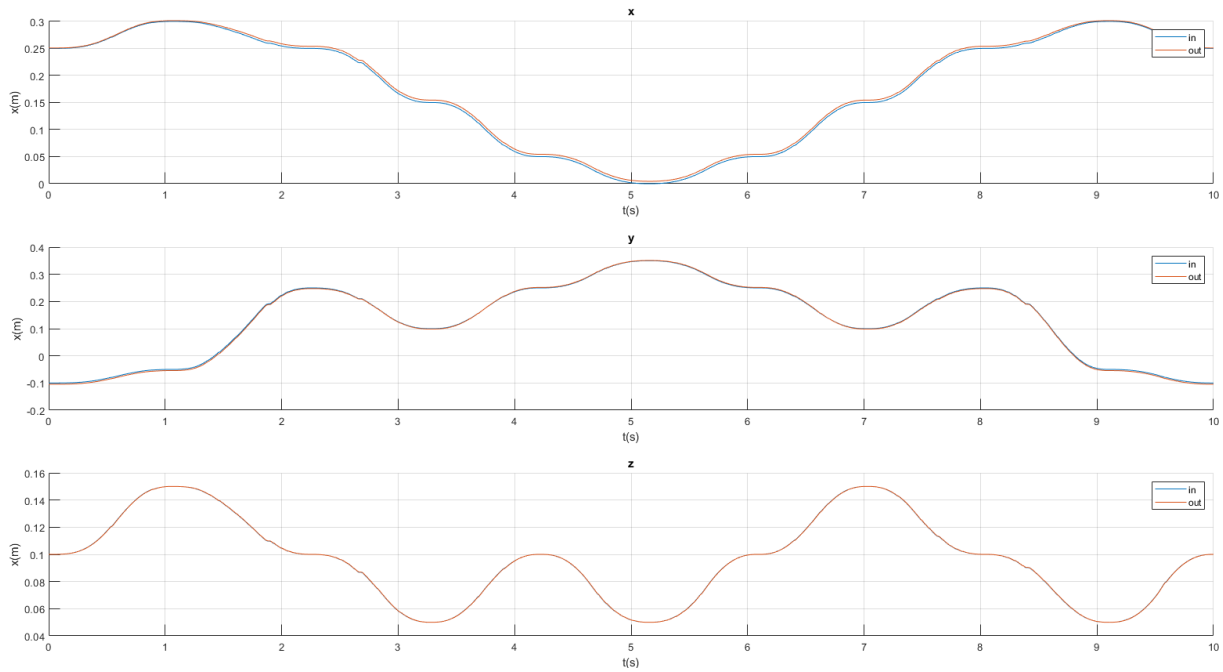


Figure 3.13: 3 axes position following

This is further demonstrated in figure 3.14 where we can clearly see that the error is of roughly a few millimetres or even micrometres in the case of the $\hat{z}$ axis.

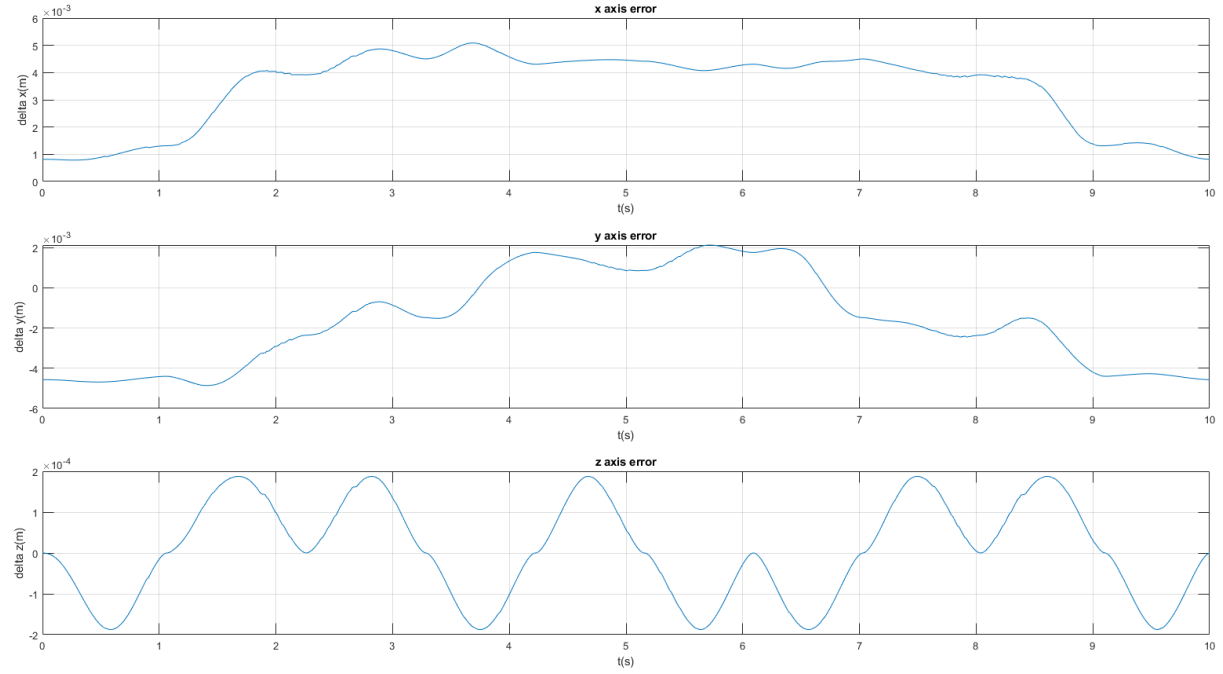


Figure 3.14: 3 axes position error

We also computed the overall position error show in figure 3.15 showing us an average value of less than 5 mm which is an acceptable error across 3 axes.

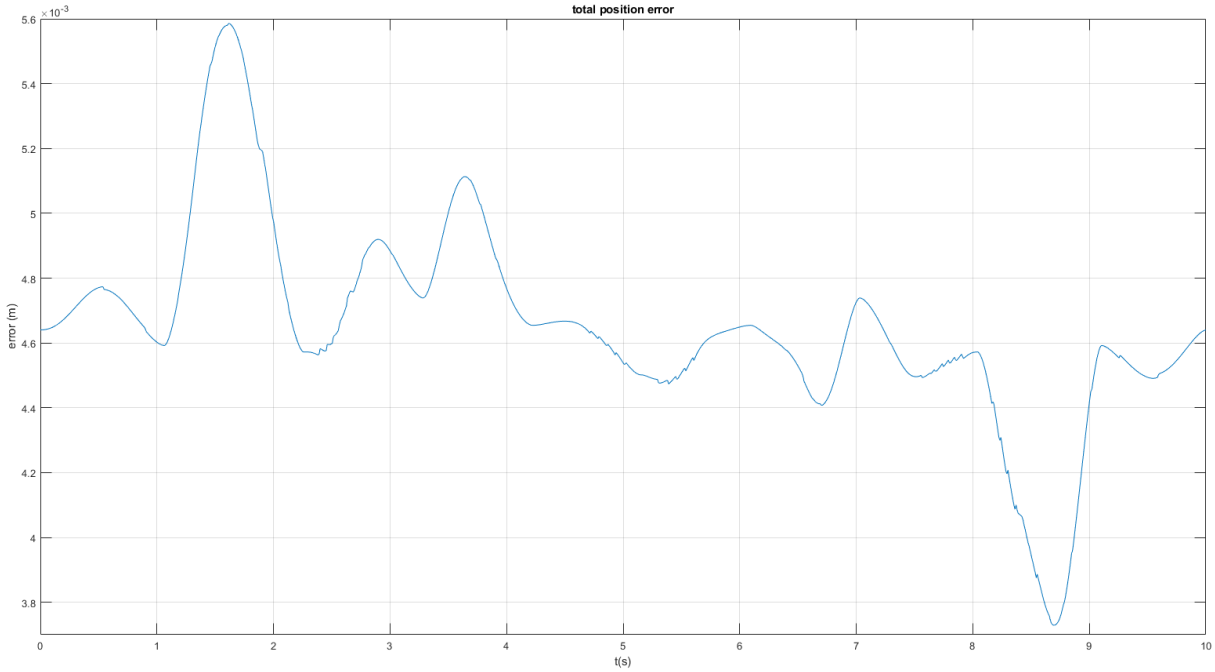


Figure 3.15: Total position error

Figure 3.16 shows the joint space movement of the 3 axes and it is clear that task space trajectory generation is quite demanding on the joints by the looks of this figure.

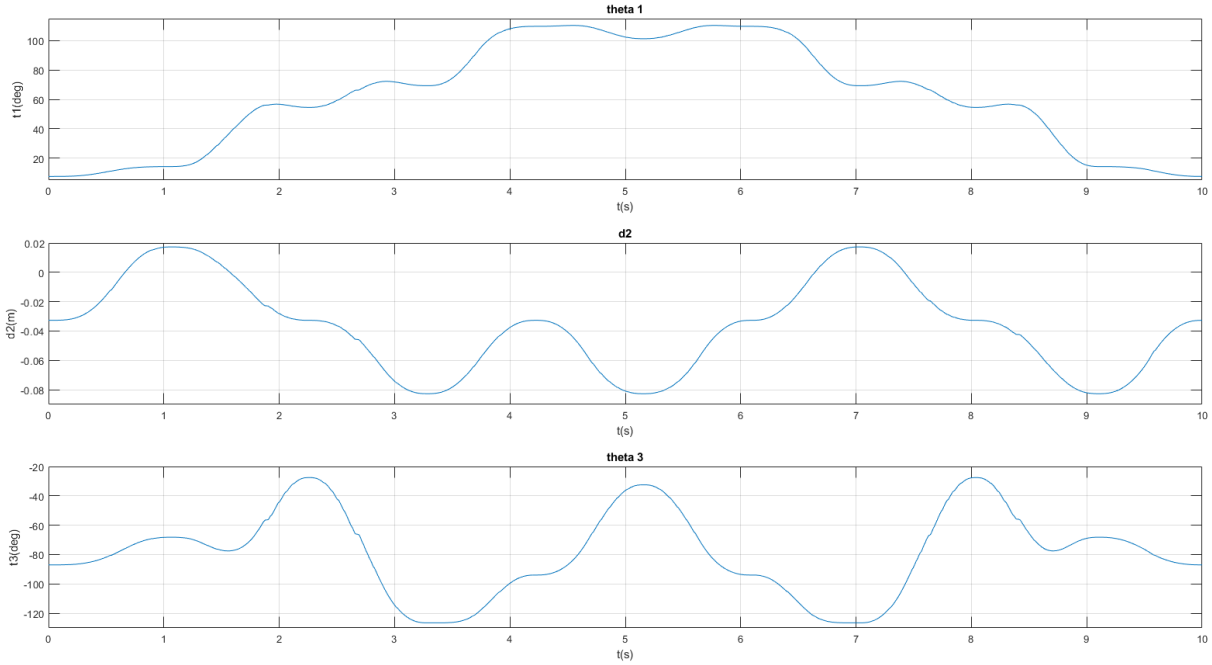


Figure 3.16: Joint space movement

Other interesting figures to look at are the speed 3.17, acceleration 3.18 and torque 3.19 graphs. We can see that that bar some spikes likely generated from the friction model, the profiles present a continuity that is specific to the quintic polynomial generation. We can also note the abrupt speed and acceleration profiles that characterise the task space generation.

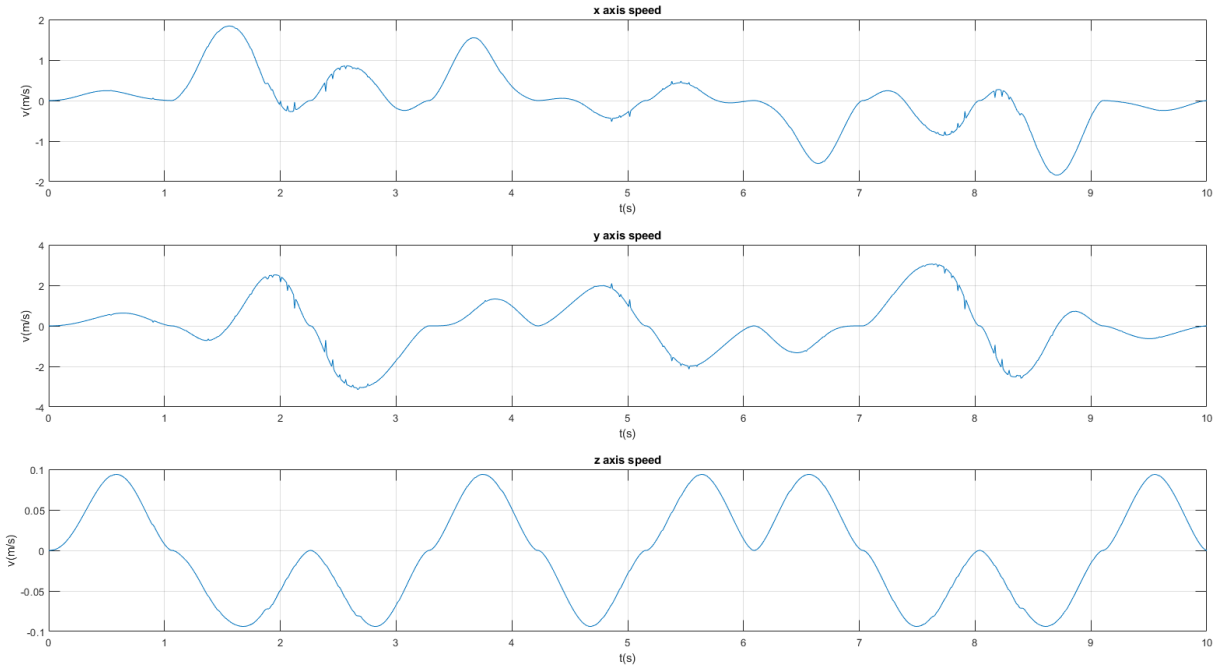


Figure 3.17: 3 axes speed

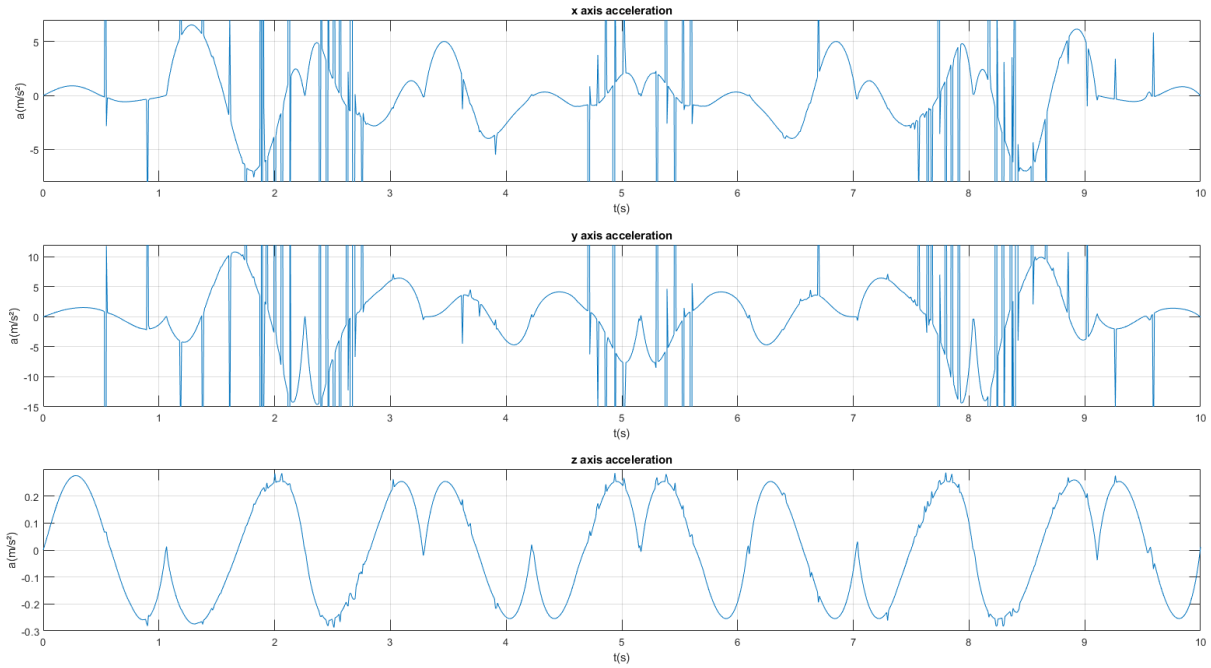


Figure 3.18: 3 axes acceleration

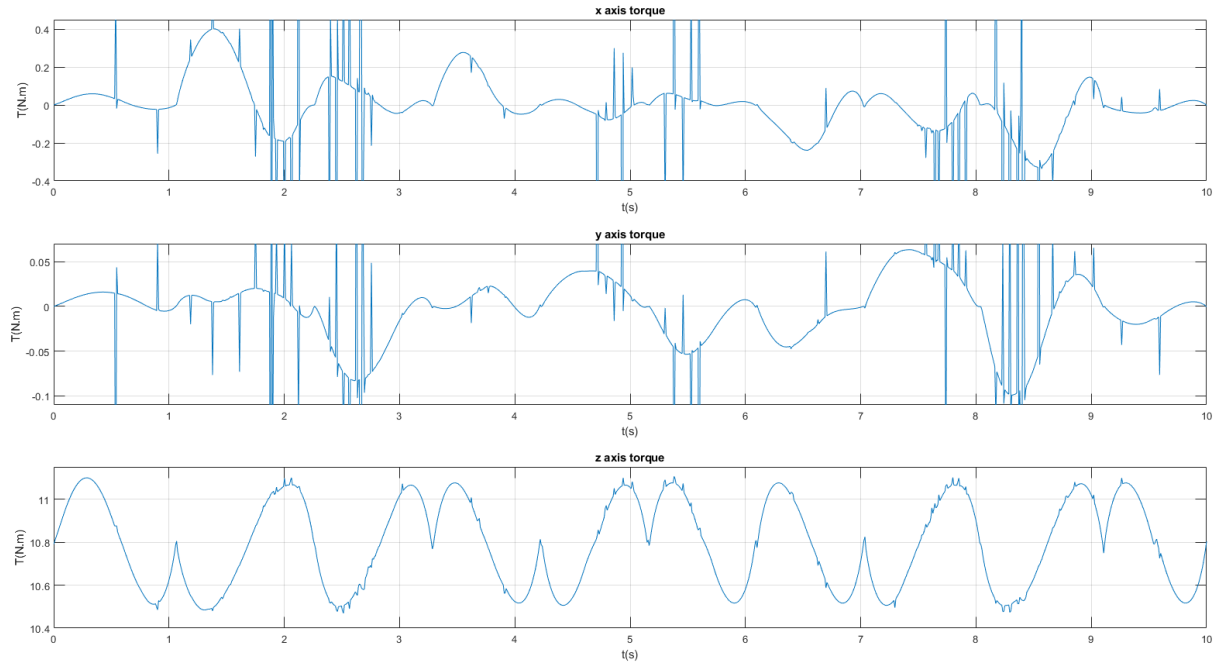


Figure 3.19: 3 axes torque

Regarding the trial of the cubic method, and while most of the position graphs were nearly identical, one might take a better look at the acceleration 3.20 and torque graphs 3.21. We can clearly see that the acceleration and torque profiles present discontinuities that would in our case be hard on the mechanical parts and such render it a sub-optimal choice for practical reasons.

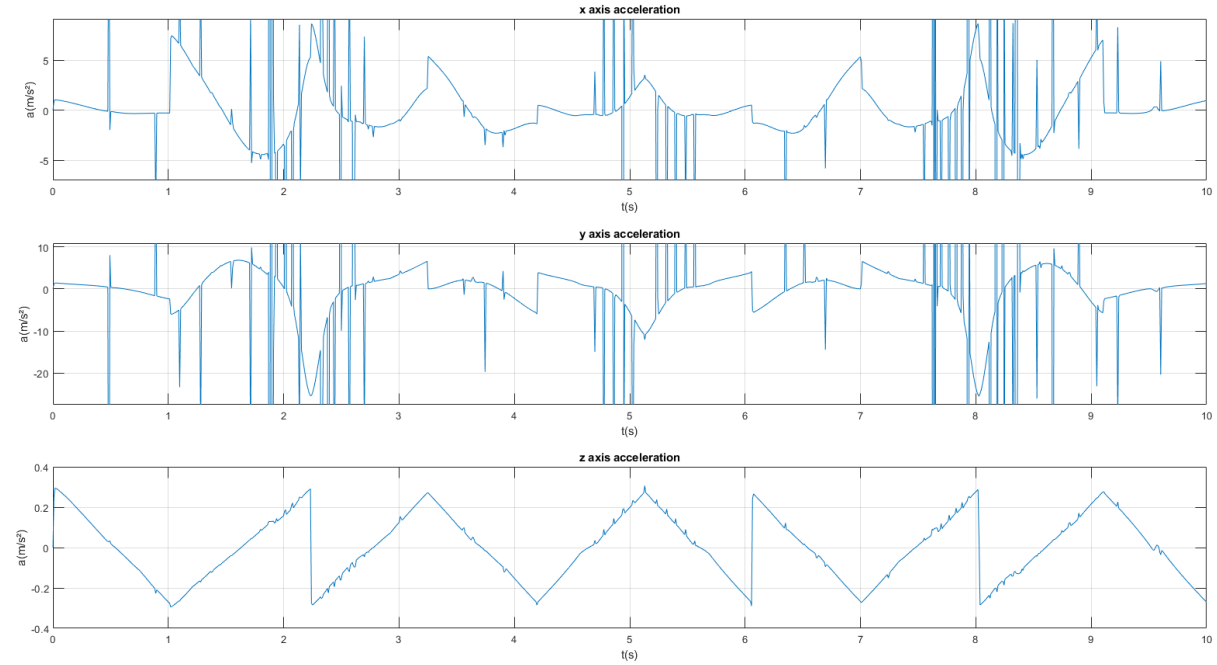


Figure 3.20: 3 axes acceleration - Cubic generation

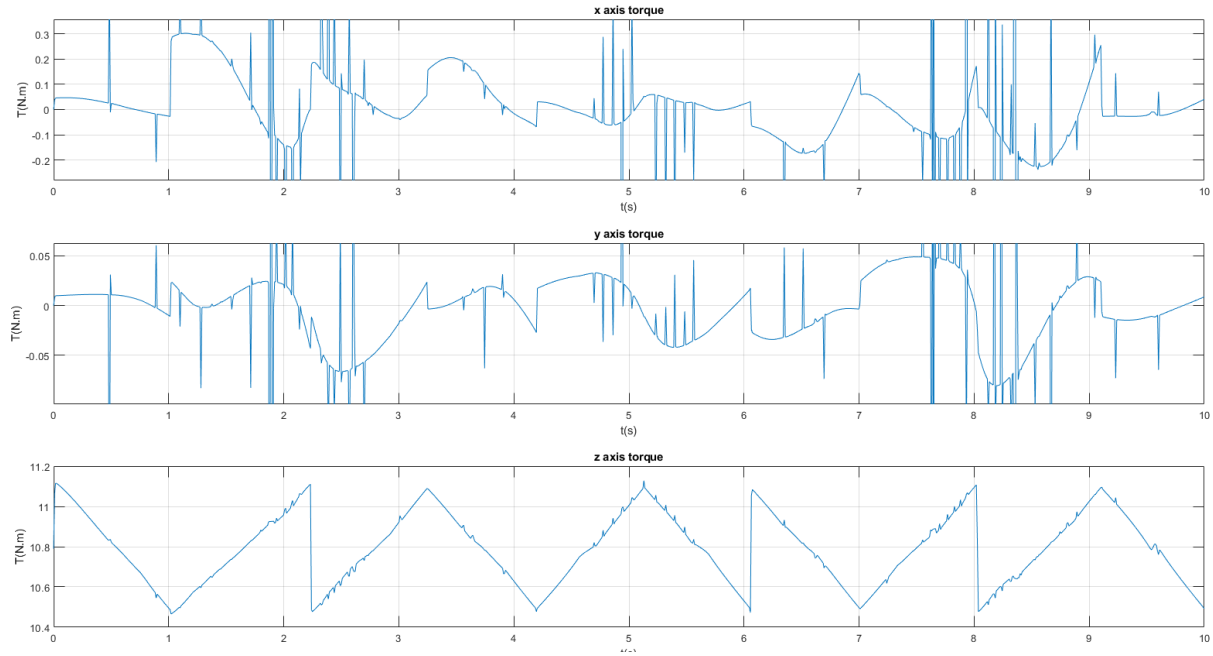


Figure 3.21: 3 axes torque - Cubic generation

3.7 Conclusion

The determination of the forward and inverse kinematics in the case of the SCARA prototype was straight-forward. This allowed us to implement them on our realisation and obtain decent performances.
As for the workspace simulation, we learned how to maximize the workspace with given link lengths and joint limits constraints. As well as determining the workspace of our final realisation and thus knowing its blind spots and safe zone which allowed us to adapt accordingly.
Finally, the trajectory planning simulation showcased the difference between multiple trajectory generation schemes as well as obtaining decent performances in trajectory following.

GENERAL CONCLUSION

In this work, we first introduced the history of robotics over time ; going over the origin of the words *robot* and *robotics* to the historical roots of the field in mythology and popular culture. We then transcribed the beginnings and evolution of robot manipulators over the last few decades until present times and presented a few challenges for robotics to face in the near-future.
We then presented our own manipulator robot and the complete process to realise it spanning from the choice of transmission systems to the 3D modelling and printing and subsequent assembly. We then went through the electronics choices and evolution as well as the program to run the robot permitting to implement the motors' movement as well as the forward and inverse kinematics along with the homing function.
Moreover, we discussed the mathematical model of our manipulator from deriving the forward kinematics that maps the set of joint variables $\mathbf{q}$ to the Cartesian space as well as the inverse kinematics problem that transform the end-effector pose from the task space to the joint space. The Jacobian matrix that describes the end-effector's linear and angular velocity in term of joint variables was then derived. We then introduced the concept of trajectory planning along with its subsequent simulation, alongside the workspace simulation varying from the design constraints.
Ultimately, we achieved modest performances with our robot with $364.5\,mm$ maximum reach, along the $\hat{x}, \hat{y}$ plane; $323\,mm$ along the $\hat{z}$ axis. Calculated speeds of $10,227\ deg/s$ for the first joint, $10\ mm/s$ for the prismatic joint, $28.125\ deg/s$ for the third joint and $100\ deg/s$ for the fourth joint. No less than $1\ kg$ of payload tested with the maximum extension configuration alongside the maximum speed, an accuracy of about $1\ mm$ on the the $\hat{x}, \hat{y}$ plane and a repeatability of about $1\ mm$ on the the $\hat{x}, \hat{y}$ plane.
Additionally, over the course of our project, we identified many areas that our prototype could be improved upon and that we could not implement by either a lack of resources, time or both. Among those we can cite the following alongside a few recommendations to possibly solve them :

- **Play in the base joint :** Improving the design of the base joint to improve the disruption tolerance.
- **Backlash :** Replacing the pulleys with metallic versions or more resistant 3D

printed version to minimise the wear.

- **Power supply :** Replacing the lab power supply by a more portable version as shown in 2.20 that also offers higher current.
- **Encoders :** Getting the documentation of the current encoders or trying to exploit the available encoders signal showcased in 2.19 without passing by the driver. Alternatively getting another type of encoder such as the *AS5600* magnetic position encoders.
- **Cable management :** Adding shielding to the wires in order to prevent the magnetic disruption.
- **Parallelism of the motors' movement :** Figuring out the exact speed profile to set the timing of the motors.
- **Speed :** Changing the microstepping value by trading off some precision and smoothness or changing the controller for one with a higher frequency at the expense of the simplicity of use of the *Accelstepper* library.
- **End-effector :** The space is left empty and fitting for any kind of end-effector with space present for its cable management as well.
- **Control of the robot :** The implementation of the control can easily be done once the encoders issue is fixed in order to improve the performances of the robot.

APPENDIX

A

KINEMATICS PROGRAM

This code will calculate the forward kinematics in Matlab for our robot as well as the Jacobian matrix and its inverse :

```
clc,clear all ,close all;

%  L1 = 228 ;
%  L2 = 136.5 ;
%  theta1 = deg2rad(90) ;
%  theta3 = deg2rad(0) ;
%  theta4 = deg2rad(0) ;
%  d2 = 0 ;
%  r = 125.5 ;



syms theta1 d2 theta3 theta4
syms L1 L2 r

disp('------FK------')
T_0_1 = [cos(theta1) -sin(theta1) 0 0;
         sin(theta1) cos(theta1)  0 0;
         0       0      1 0;
         0       0      0 1] ;
T_1_2 = [1 0 0 0 ;
         0 1 0 0 ;
         0 0 1 d2 ;
         0 0 0 1 ];
 T_2_3 =[cos(theta3) -sin(theta3) 0 L1;
        sin(theta3) cos(theta3)  0 0;
```

```
27          0       0       1  0;
28          0       0       0  1] ;
29  T_3_4 = [cos(theta4) sin(theta4) 0 L2;
30          sin(theta4) -cos(theta4)  0  0;
31          0       0      -1 -r;
32          0       0       0  1] ;
33  T_0_2 = T_0_1 * T_1_2;
34  ST_0_2= simplify(T_0_2)
35  T_0_3 = T_0_2 * T_2_3;
36  ST_0_3 =simplify(T_0_3)
37  T_0_4 = T_0_3 * T_3_4;
38  ST_0_4 = simplify(T_0_4)
39
40
41  disp('-----jacobian-----')
42  T_0_0 = eye(4,4);
43
44  R_0_0 = T_0_0([1:3],[1:3]);
45  d_0_0 = T_0_0([1:3],4);
46
47  R_0_1 = T_0_1([1:3],[1:3]);
48  d_0_1 = T_0_0([1:3],4);
49
50  R_0_2 = ST_0_2([1:3],[1:3]);
51  d_0_2 = ST_0_2([1:3],4);
52
53  R_0_3 = ST_0_3([1:3],[1:3]);
54  d_0_3 = ST_0_3([1:3],4);
55
56  R_0_4 = ST_0_4([1:3],[1:3]);
57  d_0_4 = ST_0_4([1:3],4);
58
59  Z0 = [0  0  1]';
60  J1 = [cross(R_0_1*Z0,(d_0_4-d_0_1));R_0_1*Z0] ;
61  J2 = [R_0_2*Z0;0;  0  ;0];
62  J3 = [cross(R_0_3*Z0,(d_0_4-d_0_3));R_0_3*Z0] ;
63  J4 = [cross(R_0_4*Z0,(d_0_4-d_0_4));R_0_4*Z0];
64  J  = [J1  J2  J3   J4]
65  J44 = J([1  2  3  6],:)
66 D = simplify(det(J44))
67  invJ = simplify(inv(J44))
```

APPENDIX

B

WORKSPACE SIMULATION

This is the Matlab code used for the workspace simulation in 3.6.1.

```
function [out] = workspace(l1,l2,tmin1,tmax1,tmin2,tmax2)
% input the constraints
resolution=0.001; % resolution of the points generation
x=[];
y=[];
for i=tmin1 : resolution : tmax1
    for j=tmin2 : resolution : tmax2
        x=[x;l2*cos(i+j)+l1*cos(i)];
        y=[y;l2*sin(i+j)+l1*sin(i)];
        % forward kinematics equations
    end
end
out=[x y];
end
```

APPENDIX

C

TRAJECTORY PLANNING

Here is the *Levenberg-Marquart* solver parameters used in the trajectory planning simulation, see 3.6.2.

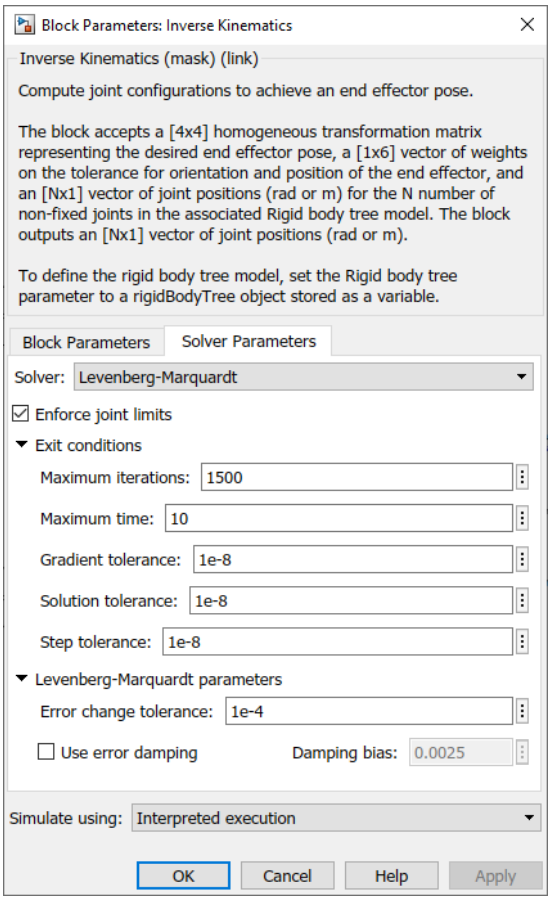


Figure C.1: *Levenberg-Marquart* parameters

Here are the simulation models of the trajectory planning simulation, see 3.6.2.

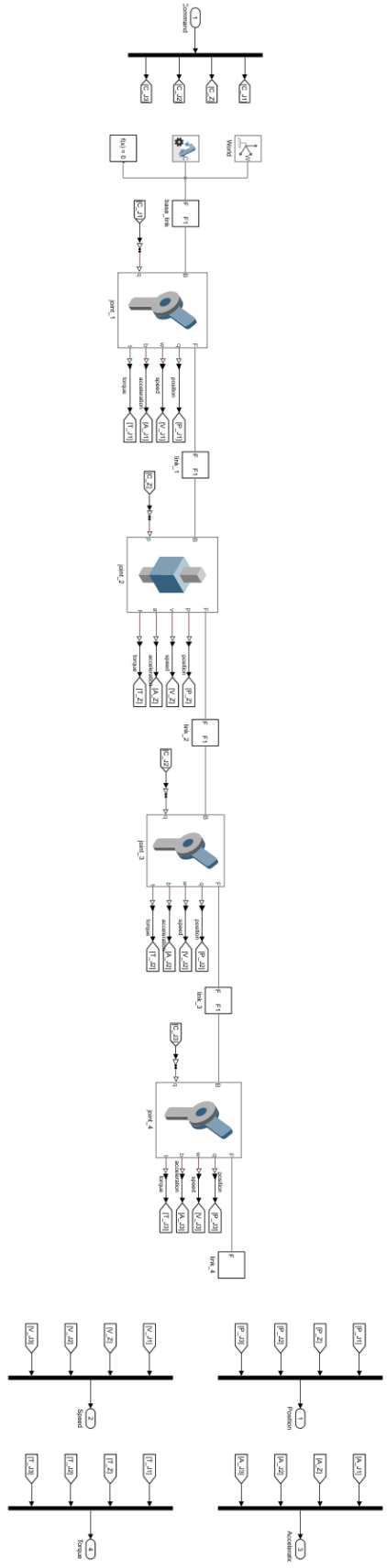


Figure C.2: Robot model in Simulink

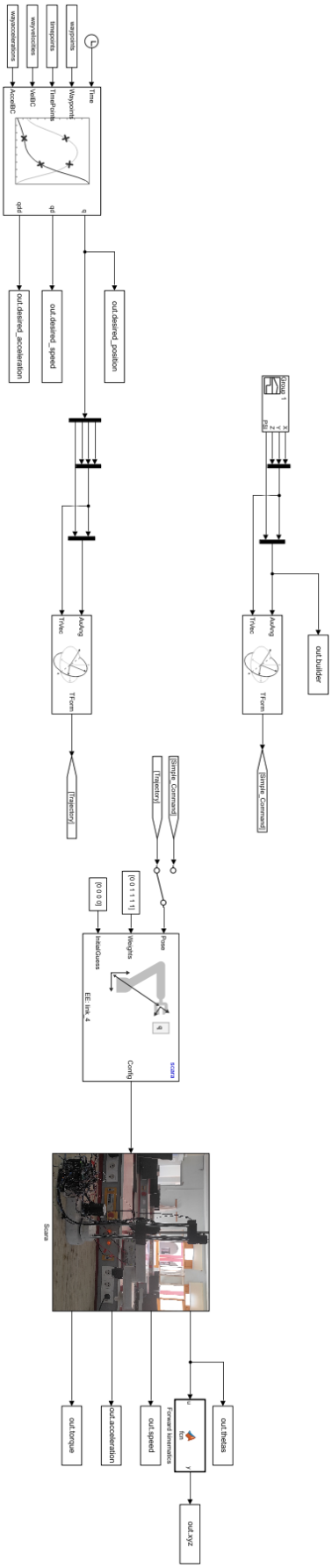


Figure C.3: Trajectory planning simulation

APPENDIX

D

PYTHON CODE

This python code will take commend from the user, calculate the forward kinematics or the inverse kinematics and sent the resulting data to the Arduino via serial communication:

```
#coding : utf-8
import serial
import time
import numpy as np
import math
mega = serial.Serial("com4",115200 , timeout= 1)
L1 = 228.0
L2 = 136.5
r = 170.0
def read_write_arduino(str,arduino):
    arduino.write(str.encode('utf-8'))
    time.sleep(0.05)
    lines = arduino.readlines()
    for line in lines :
        print(line.decode('ascii'))
def angleCalc(s,c):
    theta = math.asin(s)
    if c < 0 :
        if s > 0 :
            theta = math.pi - theta
        elif s < 0 :
            theta = -math.pi - theta
        else :
            theta = math.pi + theta
    theta = theta * 180 / math.pi
    return theta



while True :

```

```
    command = input("input somthing: ")
    commandList = command.split(" ")
    if commandList[0] == "FK" :
        print("FORWARD KINEMATICS")
        Ftheta1 = float (commandList[1])
        Fz      = float (commandList[2])
        Ftheta2 = float (commandList[3])
        Ftheta3 = float (commandList[4])
        T04 = np.array([[math.cos(Ftheta1+Ftheta2+Ftheta3), math.sin(
                                            Ftheta1+Ftheta2+Ftheta3),0,
                                            L1*math.cos(Ftheta1)+L2*math
                                            .cos(Ftheta1+Ftheta2)],
        [math.sin(Ftheta1+Ftheta2+Ftheta3),-math.cos(Ftheta1+Ftheta2+
                                            Ftheta3),0,L1*math.sin(
                                            Ftheta1)+L2*math.sin(Ftheta1
                                            +Ftheta2)],
        [0,0,-1.0,Fz-r],
        [0,0,0,1]])
        print(T04)
        serialSTR = "data" + " " + str(Fz) + " " + str(Ftheta1) + " " +
                                            str(Ftheta2)+" " + str(
                                            Ftheta3)
        read_write_arduino(serialSTR,mega)
        #print(serialSTR)



    elif commandList[0] == "IK" :
        print("INVERS KINEMATICS")
        px = float (commandList[1])
        py = float (commandList[2])
        pz = float (commandList[3])
        psiDgree= float (commandList[4])
        #psiRad= (psiDgree * math.pi)/180.0
        """
        R = np.array([[math.cos(psi), -math.sin(psi) ,0],
        [math.sin(psi), math.cos(psi),0],
        [0,0,1]])
        nx = R[0][0]
        ny = R[0][1]
        print(nx)
        print(ny)
        """


        c2 = (1/(2*L2*L1))*(px**2 + py ** 2 - L1**2 - L2**2 )
        s2 = math.sqrt(1- c2**2)
        s2v2 = -math.sqrt(1- c2**2)
        Itheta2 =  angleCalc(s2,c2)
        Itheta2v2 =  angleCalc(s2v2,c2)

        c1 = (((L1 + L2 * c2)*px)- (L2 * s2 *py)) / (px**2 + py**2)
        s1 = (((L1 + L2 * c2)*py) - (L2 * s2 *px)) / (px**2 + py**2)
        c1v2 = (((L1 + L2 * c2)*px)- (L2 * s2v2 *py)) / (px**2 + py**2)
        s1v2 = (((L1 + L2 * c2)*py) - (L2 * s2v2 *px)) / (px**2 + py**2)


        Itheta1 = angleCalc(s1,c1)
```

```
82          Itheta1v2 = angleCalc(s1v2,c1v2)
83
84          Itheta3 =   psiDgree - Itheta1 - Itheta2
85          Itheta3v2 = psiDgree - Itheta1v2 - Itheta2v2
86          Iz = pz
87          print("INVERS KINEMATICS v1 : theta1= {0}  theta2= {1} theta3= {
                                            2}  z = {3}".format(Itheta1,
                                            Itheta2,Itheta3,Iz))
88          print("INVERS KINEMATICS v2 : theta1= {0}  theta2= {1} theta3= {
                                            2}  Z = {3}".format(
                                            Itheta1v2,Itheta2v2,
                                            Itheta3v2,Iz))
89          if py > 0 :
90              serialSTR2 = "data" + " " + str(Iz) + " " + str(round(
                                                Itheta1,3)) + " " + str(
                                                round(Itheta2,3))+" " +
                                                str(round(Itheta3,3))
91              print(serialSTR2)
92              read_write_arduino(serialSTR2,mega)
93          else :
94              serialSTR2 = "data" + " " + str(round(Iz,3)) + " " + str(
                                                round(Itheta1v2,3)) + "
                                                " + str(round(Itheta2v2,
                                                3))+" " + str(round(
                                                Itheta3v2,3))
95              read_write_arduino(serialSTR2,mega)
96              print(serialSTR2)
97
98      elif commandList[0] == "homing" :
99          serialSTR= "homing"
100         read_write_arduino(serialSTR,mega)
101     elif commandList[0] == "homing" :
102         serialSTR= "homingAxis"+ commandList[1]
103         read_write_arduino(serialSTR,mega)
104     else :
105         print ("NOT VALID")
106
107     #print("FORWARD KINEMATICS : Ftheta1= {0}  Fz= {1} Ftheta2= {2}
                                        Ftheta3= {3} ".format(Ftheta1,Fz
                                        ,Ftheta2,Ftheta3))
```

APPENDIX

E

ARDUINO CODE

The following code is the constant set in the Arduino code for our robot :

```
/*-----------------------------*/
#define UP -1.0
#define DOWN 1.0
#define CW -1.0
#define CCW 1.0
#define motorInterfaceType 1

bool newData = false;
bool isAllowed = true;

double L1 = 228.0;
double L2 = 136.5;


/*-----------------------------*/

/*---------------J1-axis---------------*/

#define stepPinJ1 4
#define dirPinJ1 5
#define enablePinJ1 6
#define limitSwitchJ1 19
#define microSteppingJ1 25600.0 // microstepping * 200

double cstJ1 = (110.0)/(20.0*360.0);//(175.0)/(8.0*360.0);
double homingJ1 = CW * cstJ1 * microSteppingJ1 *176.5 ;  // 176 (angle)
double degreesJ1 = 0 ;


/*---------------J2-axis---------------*/
#define stepPinJ2 7
#define dirPinJ2 8
#define enablePinJ2 9
```

```
#define limitSwitchJ2 20
#define microSteppingJ2 3200.0  // microstepping * 200
double cstJ2 = (16.0)/(1.0*360.0);
double homingJ2 = CW * cstJ2 * microSteppingJ2 *153.0 ; // 157 (angle)
double degreesJ2 = 0 ;

/*--------------J3-axis--------------*/
#define stepPinJ3 10
#define dirPinJ3 11
#define enablePinJ3 12
#define limitSwitchJ3 21
#define microSteppingJ3 3200.0  // microstepping * 200
double cstJ3 = (9.0)/(2.0*360.0);
double homingJ3 = CCW * cstJ3 * microSteppingJ3 *148.0 ; // 157 (angle)
double degreesJ3 = 0 ;


/*--------------Z-axis--------------*/
#define stepPinZ 16
#define dirPinZ 15
#define enablePinZ 14
#define limitSwitchZ 18
#define microSteppingZ 3200.0 // microstepping * 200
double homingZ = UP*((microSteppingZ/8.0)*161.5); // 161.5 mm (distance)
                                        8.0 mm pitch of threaded rod
double degreesZ = 0 ;

// -161.5 mm , +154 mm

/*----------------------------*/
AccelStepper StepperJ1(motorInterfaceType, stepPinJ1, dirPinJ1);
AccelStepper StepperJ2(motorInterfaceType, stepPinJ2, dirPinJ2);
AccelStepper StepperJ3(motorInterfaceType, stepPinJ3, dirPinJ3);
AccelStepper StepperZ(motorInterfaceType, stepPinZ, dirPinZ);
/*----------------------------*/
```

The following code will read the serial data :

```
void serialread()
{
  String receivedCommand ;
  if (Serial.available() > 0  )
  {
        //printOnce = false  ;
        receivedCommand = Serial.readStringUntil(' '); // pass the value
                                                to the receivedCommad
                                               variable
        newData = true; //indicate that there is a new data by setting
                                               this bool to true
        if(newData == true )
        {
          if (receivedCommand == "data")
          {
            degreesZ = Serial.parseFloat(SKIP_ALL);
            degreesJ1 = Serial.parseFloat(SKIP_ALL);
            degreesJ2 = Serial.parseFloat(SKIP_ALL);
            degreesJ3 = Serial.parseFloat(SKIP_ALL);

          }
```

```
          else if (receivedCommand == "homing")
          {

            homing();

          }
          else if (receivedCommand == "homingAxis")
          {
            int axis = Serial.parseInt(SKIP_ALL);
            homingAxis(axis);
          }

        }
        newData = false;
  }
}
```

This is an example of the homing function :

```
 //---------------------------------HOMING J1-axis-----------------------
                                     -------------------//
    isAllowed = false ;
    Serial.println("HOMING SEQUANCE J1... ");
   StepperJ1.enableOutputs();

   while (digitalRead(limitSwitchJ1) != LOW)
   {
      StepperJ1.setSpeed(CW*4000);
      StepperJ1.runSpeed();
      StepperJ1.setCurrentPosition(homingJ1); // When limit switch
                                              pressed set position to 0
                                              steps
   }

  delay(20);

  StepperJ1.moveTo(0);

  while (StepperJ1.currentPosition() != 0)
  {
    StepperJ1.run();
  }

  StepperJ1.disableOutputs();
  Serial.println("DONE HOMING J1... ");
    isAllowed = true ;
```

This is how the motors are moved using **Accelstepper**:

```
 void moveMotors (double J1Degrees ,double Zmilimiter, double J2Degrees,
                                      double J3Degrees )
{


    double stepsJ1 =  cstJ1 * microSteppingJ1 *J1Degrees ;
    double stepsJ2 =  cstJ2 * microSteppingJ2 *J2Degrees ;
    double stepsJ3 =  cstJ3 * microSteppingJ3 *J3Degrees ;
    double stepsZ = ((microSteppingZ/8.0)*Zmilimiter);

```

```
10    StepperZ .moveTo (stepsZ);
11    StepperJ1.moveTo(stepsJ1);
12    StepperJ2.moveTo(stepsJ2);
13    StepperJ3.moveTo(stepsJ3);
14    if (StepperZ.distanceToGo() != 0 || StepperJ1.distanceToGo()!= 0 ||
                                          StepperJ2.distanceToGo()!= 0 ||
                                          StepperJ3.distanceToGo() != 0)
15    {
16      isAllowed = false ;
17
18        StepperZ.enableOutputs();
19        StepperJ1.enableOutputs();
20        StepperJ2.enableOutputs();
21        StepperJ3.enableOutputs();
22
23         StepperZ.run();
24        StepperJ1.run();
25        StepperJ2.run();
26        StepperJ3.run();
27
28    }
29
30    else if(StepperZ.distanceToGo() == 0 && StepperJ1.distanceToGo()== 0 &
                                          & StepperJ2.distanceToGo()== 0 &&
                                          StepperJ3.distanceToGo() == 0)
31    {
32
33        StepperZ.disableOutputs();
34        StepperJ1.disableOutputs();
35        StepperJ2.disableOutputs();
36        StepperJ3.disableOutputs();
37        isAllowed = true ;
38
39    }
40
41
42  }
```

BIBLIOGRAPHY

[1] Peter Corke. *Robotics, vision and control fundamental algorithms in MATLAB*. Vol. 118. Springer International Publishing, 2017.

[2] Bruno Siciliano, Lorenzo Sciavicco, and Luigi Villani. *Robotics modelling, planning and Control*. Springer, 2009.

[3] Kevin Lynch and Frank C. Park. *Modern Robotics: Mechanics, planning, and Control*. Cambridge University Press, 2017.

[4] George A Bekey et al. "Robotics: state of the art and future challenges". In: (2008).

[5] Thomas R Kurfess et al. *Robotics and automation handbook*. Vol. 414. CRC press Boca Raton, FL, 2005.

[6] Robot magazine. *Le robot Unimate : à l'origine d'une révolution l'industrielle*. 2020. URL: https://www.robot-magazine.fr/robot-unimate-revolution-industrielle/.

[7] Irati Zamalloa et al. "Dissecting Robotics-historical overview and future perspectives". In: *arXiv preprint arXiv:1704.08617* (2017).

[8] A Gasparetto and L Scalera. "A brief history of industrial robotics in the 20th century". In: *Advances in Historical Studies* 8.1 (2019), pp. 24–35.

[9] Shimon Y Nof. *Handbook of industrial robotics*. John Wiley & Sons, 1999.

[10] Morteza Shariatee et al. "Design of an economical SCARA robot for industrial applications". In: *2014 Second RSI/ISM International Conference on Robotics and Mechatronics (ICRoM)*. IEEE. 2014, pp. 534–539.

[11] Dejan Nedelkovski. *SCARA Robot — How To Build Your Own Arduino Based Robot*. 2020. URL: https://howtomechatronics.com/projects/scara-robot-how-to-build-your-own-arduino-based-robot/.

[12] Carmine Dario Bellicoso et al. "Design, modeling and control of a 5-DoF light-weight robot arm for aerial manipulation". In: *2015 23rd Mediterranean Conference on Control and Automation (MED)*. IEEE. 2015, pp. 853–858.

[13] PJ Turner, P Nigrowsky, and G Vines. “A new approach for the design of robot joint transmission”. In: *Mechatronics* 11.8 (2001), pp. 1053–1062.

[14] M Kemal Ciliz and M Ömer Tuncay. “Comparative experiments with a multiple model based adaptive controller for a SCARA type direct drive manipulator”. In: *Robotica* 23.6 (2005), pp. 721–729.

[15] Giovanni Legnani and Rodolfo Faglia. “Harmonic drive transmissions: the effects of their elasticity, clearance and irregularity on the dynamic behaviour of an actual SCARA robot”. In: *Robotica* 10.4 (1992), pp. 369–375.

[16] Sebastian Castro. *Trajectory Planning for Robot Manipulators*. 2019. URL: https://blogs.mathworks.com/student-lounge/2019/11/06/robot-manipulator-trajectory/.

[17] Wisama Khalil and Etienne Dombre. *Modeling identification and control of robots*. CRC Press, 2002.

[18] Chablat. Damien. “Domaines d’unicité et parcourabilité pour les manipulateurs pleinement parallèles”. In: *Automatique / Robotique. Ecole Centrale de Nantes (ECN)* (1998).

Abstract

In this work, we aimed to design and realise a 4 degrees of freedom robot manipulator type SCARA for *pick-and-place* applications with modest to decent performances. We first introduced the definition and history of robots from the antiquity to modern age. We then described the full process of the design, printing, assembly and programming of our robot prototype. The mathematical model of the robot is then derived alongside the introduction of the concept of trajectory planning subsequent to its own simulation in addition to the simulation of the total workspace of the robot.

Keywords : Robotics, SCARA, 3D modelling, Kinematics, Trajectory planning

Résumé

Dans cette œuvre, nous avions pour objectif de designer et réaliser un robot manipulateur à 4 DDL de type SCARA pour des applications *pick-and-place* avec des performances décentes. Nous avons tout d'abord introduit la définition et l'historique des robots de l'antiquité à l'ère moderne. Nous avons ensuite décris en détails le processus de design, impression, assemblage et programmation de notre prototype. Le modèle mathématique du robot est ensuite déduit en compagnie de l'introduction du concept de planification de trajectoire avec sa propre simulation ainsi que celle de l'espace de travail total du robot.

Mots clés : Robotique, SCARA, Modélisation 3D, Cinématique, Planification de trajectoire

ملخص:

في هذا العمل, كان هدفنا تصميم و صنع يد آلية من نوع SCARA و التي تتكون من 4 درجات حرية لتوضيفها في تطبيق pick-and-placeمع أداء متواضع إلى لائق.أولا قدمنا تعريف وتاريخ الروبوتات من العصور القديمة إلى العصر الحديث. تم وصفنا العملية الكاملة لتصميم وطباعة وتجميع وبرمجة النموذج الأولي للروبوت. تم تم اشتقاق النموذج الرياضي للروبوت جنبًا إلى جنب مع إدخال مفهوم تخطيط المسار(trajectory planning) بعد المحاكاة الخاصة به بالإضافة إلى محاكاة مساحة العمل الإجمالية للروبوت.

الكلمات المفتاحية : الروبوتات ، SCARA ، النمذجة ثلاثية الأبعاد ، الحركية ، تخطيط المسار