VISVESVARAYA TECHNOLOGICAL UNIVERSITY

Jnana Sangama, Belagavi, Karnataka–590014

Mini-Project Report

"IMPLEMENTATION OF ROBOTIC ARM USING MATLAB AND SIMULINK"

Submitted in partial fulfilment of the requirements for the award of degree of

BACHELOR OF ENGINEERING

in

ELECTRONICS AND COMMUNICATION ENGINEERING

Submitted by

H P Jeevan	(1BI18EC051)
G Rohith	(1BI18EC045)
Emyl Varghese George	(1BI18EC044)

Under the guidance of

Dr.Sree Ranga Raju M N,

Professor & Head,

Dept. of ECE, BIT

DEPARTMENT OF ELECTRONICS & COMMUNICATION ENGINEERING

BANGALORE INSTITUTE OF TECHNOLOGY

K.R. Road, BANGALORE -560004

2020 -2021

BANGALORE INSTITUTE OF TECHNOLOGY

K.R. Road, V.V Puram, Bangalore - 560004

www.bit-bangalore.edu.in

Department of Electronics and Communication Engineering

CERTIFICATE

Certified that the mini-project work entitled "IMPLEMENTATION OF ROBOTIC ARM USING MATLAB AND SIMULINK" carried out by H P Jeevan (1BI18EC051), G Rohith (1BI18EC045) and Emyl Varghese George (1BI18EC044) bonafide students of **Bangalore Institute of Technology** in partial fulfilment for the award of **Bachelor of Engineering/ Bachelor of Technology** in **Electronics and Communication Engineering** of the Visvesvaraya Technological University, Belgaum during the year 2020- 2021. It is certified that all corrections/suggestions indicated for Internal Assessment have been incorporated in the Report deposited in the departmental library. The Mini- project report has been approved as it satisfies the academic requirements in respect of Mini-Project work prescribed for the above said Degree.

Dr.Sree Ranga Raju M N,
Professor & Head,
Dept. of ECE, BIT

Dr.Sree Ranga Raju M N,
Professor & Head,
Dept. of ECE, BIT

External Viva

Name of the examiners & Signature with date

1. Dr.Sree Ranga Raju M N
2. Sachin B M
3. Dr. Hemanth Kumar A R

Acknowledgement

I am overwhelmed in all humbleness and gratefulness to acknowledge my depth to all those who have helped me to put these ideas, well above the level of simplicity and into something concrete.

I would like to express my special thanks of gratitude to my guide as well as our principal who gave me the golden opportunity to do this wonderful project on the topic of Robotic Arm, which also helped me in doing a lot of Research and I came to know about so many new things. I am really thankful to them.

Any attempt at any level can't be satisfactorily completed without the support and guidance of my parents and friends.

I would like to thank my parents who helped me a lot in gathering different information, collecting data and guiding me from time to time in making this project, despite of their busy schedules, they gave me different ideas in making this project unique.

H P Jeevan (1BI18EC051),

G Rohith (1BI18EC045),

Emyl Varghese George (1BI18EC044)

ABSTRACT

Nowadays, robotics has been acknowledged as a mainstay in the industrial automation domain for decades. Robotic arms are being used in industries to minimize the human errors and increase efficiency, productivity, precision of the operations taking place.

One of the most important advantages of introducing Robotic arm in Industries is that it can work in crucial conditions like high temperatures, pressures where it's risky for humans to work. Since a manipulator comes under Flexible Automation, they can be updated and modified easily. An assistance robot is a robot which is self-governed and can work independently to perform the given tasks. Industries, military undertakings, medical sector are some of the fields where these robots are now being used. Working in assignments involving high temperatures or tasks like defusing bombs, handling molten metal might be fatal for people.

In this project we deal with the design and control of a robotic arm with 3 degrees of freedom. Our model has been constructed using four links and three revolute joints. Two types of grippers have been designed, a mechanical gripper and an electromagnetic gripper. A simulation of the robotic arm is done to visualize the joint movements using Robotics Toolbox for MATLAB which is also covered in detail. We generate a well-defined set of trajectory points for motion planning of the simulated model. The hardware implementation is achieved using Arduino Uno which is controlled by a Simulink model.

We have referred several research papers which have been experimentally verified to observe the different types of controllers used and various methodologies to decide the degrees of freedom of a manipulator used for the picking of an object and placing it at specified position. Thus, knowledge acquired after referring all these papers, has helped in designing the Robotic arm.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER I
INTRODUCTION

Chapter I

INTRODUCTION

1.1 Overview

Robotics is the highly advanced application of technology and automation which has provided humans with a set of helping hands to escape their problems and ease their efforts. Robotics has been used widely in the market which has allowed it to become a mainstay in the industrial automation domain for decades

Figure 1.1: Robotic Arm

A *robot* is a machine especially one programmable by a computer capable of carrying out a complex series of actions automatically. A robot can be guided by an external control device, or the control may be embedded within. The word "robot" was first coined in 1921 from the Czech word "Robata" which means slave. Since the beginning of the study of robotics, there has been some controversy in the definition of a robot. So long as the evolution of robotics continues, the definition of the robot will change from time to time, depending on the technological advances in its sensory capability and level of intelligence. However, the most widely accepted definition of a robot was given by the Robotic Institute of America (RIA) in 1979. RIA defines a robot as "a reprogrammable, multifunctional manipulator designed to move materials, parts, tools, or specialized devices through various programmed motions for the performance of a variety of tasks."

A *Robotic arm*, also called *Robotic Manipulator* is basically a machine which is very similar to a human hand, it consists of a combination of links attached in series or parallel as shown in Figure1.1. A manipulator, in general is a mechanical system aimed at manipulating objects. Manipulating, in turn, means to move something with one's hands, as the word is derived from the Latin word 'manus', meaning hand. Robotic arm is constituted by a structure consisting of robust links coupled by either rotational joints (also referred to as revolute joints) or translating joints (also referred to as prismatic joints). These structures are a concatenation of links, thereby forming an open kinematic chain. A robotic arm must be able to grab and hold an object, and transfer it to a new location. Most robotic arms have onboard controllers or translators to simplify communication, though they may be controlled directly or in a number of ways.

A common term that's used when a robot arm is designed is *DOF (degrees of freedom)*. The Figure1.2 demonstrates a 6-DOF robotic arm. The *configuration* of a robot is a complete specification of the position of every point of the robot. The minimum number of real-valued coordinates needed to represent the configuration is the number of degrees of freedom of the robot. The n-dimensional space containing all possible configurations of the robot is called the *configuration space (C-space).*

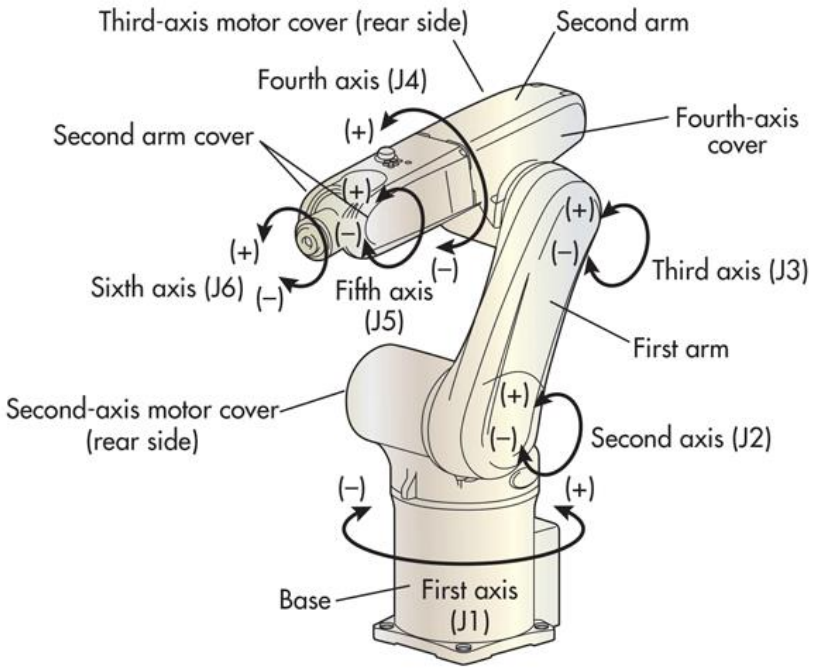
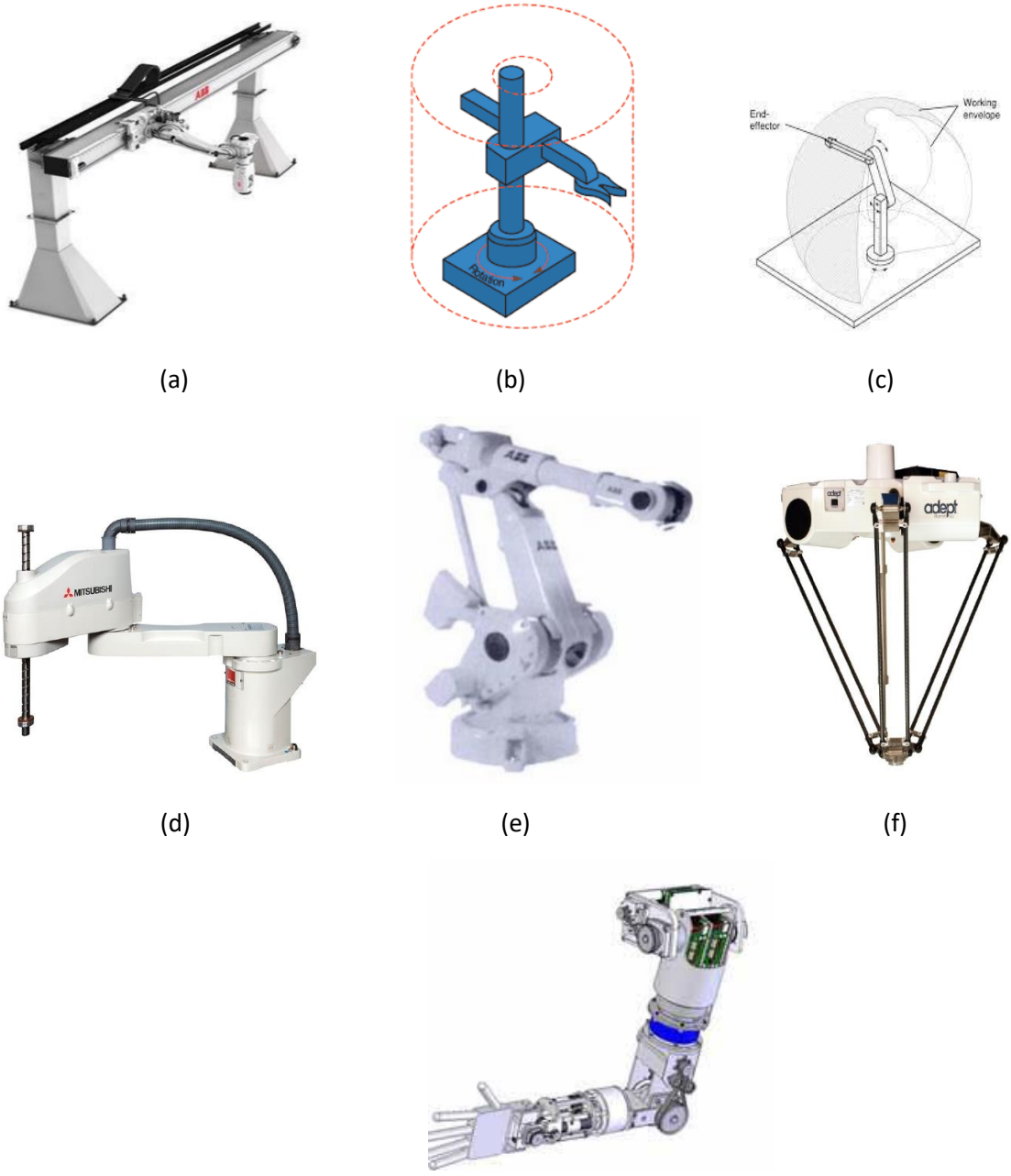


Figure 1.2: Degrees of freedom of a robotic arm

1.2 Types of Robotic Arms

Figure1.3 shows the different types of industrial robotic arms. Based on the joints used and their application, the robotic arm can be broadly classified as:



(a) (b) (c)

(d) (e) (f)

(g)

Figure 1.3: Types of robotic arms a) Cartesian b) Cylindrical c) Spherical d) Scara e) Articulated f) Parallel g) Anthropomorphic

- Cartesian robot: Three prismatic joints, whose axes are coincident with a cartesian co-ordinate constitute a cartesian robot. Arc welding, handling precision tools and pick and place work are some of its applications.

- Cylindrical robot: A robot having axes that forms a cylindrical co-ordinate system is called as cylindrical robot. Some of its applications include assembly operations, handling of machine tools, spot welding, and handling of diecasting machines.

- Spherical robot: A robot having an axis that forms a polar co-ordinate system is called a spherical robot. It is used for applications such as handling machine tools, spot welding, diecasting, fettling machines, gas welding and arc welding etc.

- Scara Robot: Two rotary joints which are parallel and are used to provide compliance in a plane constitutes a robot termed scara. Its applications include pick and place work, sealant, assembly operations and handling machine tools.

- Articulated robot: A robot consisting of an arm having at least 3 rotary joints is termed as Articulated. It is used in diecasting, assembly operations, fettling machines, gas welding, arc welding and spray painting.

- Parallel Robot: Arms having concurrent prismatic or rotary joints constitute a parallel robot. One of the uses is a mobile platform handling cockpit flight simulator.

- Anthropomorphic robot: A robotic arm which is similar to a human hand i.e., consists of independent fingers and thumbs is called as Anthropomorphic robot. The humanoid arm adopts an algorithm to avoid obstacles and the dexterous hand is capable of grasping objects.

1.3 Applications of the Robotic Arm

According to Angelo (2007), robots now play a prominent and indispensable role in modern manufacturing. The use of robots has led to increased productivity and improved product quality. It has helped to keep manufacturing industries viable in high-labour cost countries.

Robotic arms are typically used in industry. Repetitive autonomous robots perform one task repeatedly based upon predetermined movements and specifically located objects. Start and stop commands are determined by position, acceleration, deceleration, distance, and direction. More complex actions are executed based upon sensor processing. If object orientation or position is unknown, arms are often paired with computer vision and artificial intelligence to identify the object and subsequently control the arm.

Contemporary applications of the robotic arm range from doing an accurate and reliable job of spray-painting an automobile on an assembly line, to robotic surgery. The da Vinci surgical robot uses robotic arms equipped with scalpels and other instruments to more accurately target surgical objectives, allowing surgeons to use smaller, less invasive incisions.

Other applications of the robotic arm include:

- welding - arc welding, laser welding, plasma welding, welding automation.
- resistance welding and plasma cutting.
- material handling - dispensing, injection moulding, machine loading, pick and place, packaging, parts transfer and press tending.
- painting automation.
- fiberglass cutting.
- assembly operations.
- foundry operations.

In the automobile industry, robotic arms are used in diverse manufacturing processes including assembly, spot welding, arc welding, machine tending, part transfer, laser processing, cutting, grinding, polishing, deburring, testing, painting and dispensing. Robots have proved to help automakers to be more agile, flexible and to reduce production lead times. They can be programmed to perform precise intricate duties at much faster speeds than any human could be expected to. They also have the advantage of being able to withstand exposure to conditions adverse to humans such as extreme heat and presence of dangerous chemicals in the air. In many automotive plants, robots are assembling smaller components like pumps and motors at high speeds. Often, robots are performing tasks like windshield installation and wheel mounting to increase throughput.

1.4 Aim of the Project

The aim of the project is to design and build a hardware model of a robotic arm that can be used for demonstrative and educational purposes.

Bearing in mind that a similar project has been previously undertaken in the college Vidhyadeep Institute of Engineering & Technology (2018), this project work goes a step further by redesigning the robotic arm for improved accuracy by using servos to power the joints.

In our project, we have simplified the control system of the robotic arm by eliminating the use of complex algorithms such as inverse kinematics and instead we have relied on basic mathematical concepts. We want to build a cost effective and simple model. Our aim is to learn and understand the concepts used to build a completely automated robotic arm.

1.5 Project objectives

The objectives of this project are:

1. To simulate a virtual model of the 3R-robotic arm using MATLAB.
2. To design a control mechanism which can be used to move to a given position.
3. To implement an algorithm to move through a given set of waypoints.
4. To generate trajectory for motion planning.
5. To implement a hardware model of the robotic arm using Arduino.
6. Add a gripper to the end effector to pick and place an object.
7. To build a completely automated robotic arm.

1.6 Justification for the Project

The field of robotics has become indispensable to mankind. Robotics technologies are helping to improve healthcare (through the use of the da Vinci surgical system), national defence (through

the use of robotic bomb detonators), manufacturing (as witnessed in the automobile manufacturing industry), consumer goods sectors (for packaging finished goods) and so on.

The availability of a robotic arm that can be used for demonstrative and educational purposes in Engineering will go a long way in stimulating the interest of students in robotics. It will provide a tool to use for learning and experimenting with robotics. Students with a flair for programming can reprogram the robot to adapt it to different tasks.

One of the main advantages of a collaborative robotic arm in the factory is that they are highly precise and accurate. Robots work exactly how they are programmed to without any deviation, hence their accuracy and precision. Robotic arms help companies improve their production capacity. Robots are mechanical and do not require breaks and can work the whole day. They do not get sick or call in absent. In addition, they work faster and more accurately than any human worker. As a result, they will produce more per hour than any human worker. Speed and efficiency are the main competitive advantage that robots bring to an industrial operation. Robots are fast and are highly efficient in carrying out their tasks. They are accurate and precise; they can perform multiple tasks and can help organizations improve their financial status and market position.

1.7 Scope of the Project

This work covers design, simulation, programming and assembly of a basic robotic arm system. It also covers the implementation of simple algorithms that govern the motion of the arm and consider the details of the derivation of the equations of motion.

CHAPTER II
LITERATURE SURVEY

Chapter II

LITERATURE SURVEY

- A survey on Arduino Controlled Robotic Arm by Ankur Bhargava:

  In this paper, a 5 Degree of Freedom (DOF) robotic arm has been developed. It is controlled by an Arduino Uno microcontroller which accepts input signals from a user by means of a set of potentiometers. The arm is made from four rotary joints and end effector also, where rotary motion is provided by a servomotor. The servo-motors and links thus produced are assembled with fasteners to form the final shape of the arm. The Arduino has been programmed to provide rotation to each servo motor corresponding to the amount of rotation of the potentiometer shaft. A robot can be defined according to the nature of the relative movements between the links that constitute it.

- Survey on Design and Development of competitive low-cost Robot Arm with Four Degrees of Freedom by Ashraf Elfasahany:

  In this paper, the representation of the design, development and implementation of robot arm is done, which has the ability to perform simple tasks, such as light material handling. The robotic arm is designed and made from acrylic material where servo motors are used to create links between arms. However, the rotation range of the motor is less than 180º span, which greatly decreases the region reached by the arm and the possible positions. This design of the robot arm was for four degrees of freedom.

- Design and Fabrication of Pick and Place Robot to be used in Library by Anusha Ronanki, M. Kranthi:

  The use of robots in library is becoming more popular in recent years. The trend seems to continue as long as the robotics technology meets diverse and challenging needs in educational purpose. RFID is used for identifying the books and it has two IR Sensors for detecting the path. This robot is about 4 kg in weight and it is capable of picking and placing a book which weighs one kg.

- Pick and Place Robotic Arm Implementation Using Arduino Ashly Baby, Chinnu Augustine, Chinnu Thampi, Maria George, Abhilash A P, Philip C Jose:

  A robotic arm is designed using Arduino to pick and place the objects via user commands. It will pick and place an object from source to destination safely. The soft

catching gripper used in the arm will not apply any extra pressure on the objects. The robot is controlled using android based smart phones through Bluetooth. Based on the commands given by the user the robot moves accordingly. At the receiver end, there are four motors interfaced with the micro controller.

- Design, Simulation and Fabrication of A 5-Dof Robotic Arm by ORIDATE Ademola Ayodeji

  This report contains the design and implementation of a 5-DOF robotic arm. In this project, work goes a step further by redesigning the robotic arm for improved accuracy by using servos to power the joints as well as implementing the inverse kinematics of the robotic arm. A simulation of the robotic arm to visualise the joint movements using Robotics Toolbox for MATLAB is also covered in detail. A suitable servo controller was selected for the project and a control software for the robotic arm was developed using Microsoft's C# programming language. The software allows the gripper of the robotic arm to be positioned in space by specifying the coordinates of its centre position.

- Development Of Robotic Arm Using Arduino UNO by Priyambada Mishra, Riki Patel, Trushit, Upadhyaya, Arpan Desai:

  In this paper, they have used 4 servo motors to make joints of the robotic arm and the movement will be controlled with the help of potentiometer. The controller used is Arduino UNO. The analogue input signals of the Arduino’s are given to the Potentiometer. The arm has been built by cardboard and individual parts are attached to the respective servo motors. The arm is specifically created to pick and place light weight objects. So low torque servos, with a rotation of 0 to 180 degrees have been used. Programming is done using Arduino 1.6.10. Thus, the paper basically focuses on creating a robotic arm with non-useful materials and its application on small purposes.

- Wire Control Robotic Arm Ghodadra Nikesh S.1 Patel Dhruvil M.2 Luvani Maulik N.3Lalit Dhanani R.4 Mr. Paresh K. Katariya5

  In this report, they deal with robotic arm which has too much weight due to the hydraulic and pneumatic system so they have decided to remove them and replace with light weight object or actuator. They have used a mechanical cable which is light weight and has high breaking strength which can lift heavy weight with a small diameter.

As discussed in the above papers, most of the robotic arms are controlled manually by the use of potentiometers or by using gesture control mechanisms which may lead to inaccurate results. Complex algorithms such as inverse kinematics are being used in some

of the papers referred above, but in this project, we have used simple mathematical concepts that directly helps in obtaining the configurations of the robotic arm. This reduces the amount of calculation that is required which leads to simple control mechanism that is easy to understand and implement. For more details on inverse kinematics refer Appendix 1.

Also, the speed of the motion of the robotic arm in the papers discussed above is low due to use of hydraulic and pneumatic systems. This also leads to inaccurate results which makes them less applicable in high precision-driven tasks. We have instead implemented a mechanical system that makes use of light weight actuators like servo motors which gives accurate results.

In our project, we have tried to remove manual intervention by not using methods such as Bluetooth control or haptic control. The method that we have implemented involves equations that directly control the position of the arm in the coordinate space. This allows us to reduce human error and completely automate the movement of the system.

Generally, hardware models which involve Arduino are programed in C language. But, writing C code for a complex system such as robotic arm is a tedious process. Therefore, we have opted to use the Simulink software where we can use simple blocks to implement the same. This simplifies the process as the Simulink automatically produces the C code required to control the Arduino. Additionally, Simulink allows us to see the control flow between the blocks and allows for easier debugging. Simulink support package for Arduino helps us to directly deploy, test and debug on the hardware model.

CHAPTER III
METHODOLOGY

Chapter III

METHODOLOGY

In this section, we discuss the concepts that are required for the functioning of the robotic arm. There are two ways to move the robotic arm to a certain configuration, one where we know the end effector's position and the other, where we know the angles between the links also known as *joint angles*. First, we will discuss how to generate these joint angles for the robotic arm to reach a given point in its configuration space. There are four different joint angle configurations in which the robotic arm can reach the same point. We shall see how to get these configurations of the robotic arm.

The trajectory is a path defined by a set of waypoints that govern the motion of the robotic arm between two points. The generation of the trajectory points is seen in the later sections. At the end, we shall see how to use the serial communication between Simulink and Arduino to implement the hardware model.

3.1 The Joint angles

As we have seen earlier, the joint angles are the angles made by the two-consecutive links. We are using a revolute joint in our model which has 5 constraints and allows 1 DOF, therefore we get one joint angle between the two links. As we are using 4 links (including the base of the robot) we get 3 joint angles namely θ_1, θ_2 and θ_3. By varying these 3 joint angles, we can get the end effector to move to a specific position x, y and z in the C-space.

The basic structure of the robotic arm is seen in Figure 3.1. It consists of 4 links with 3 revolute joints and a gripper is attached to the end effector of the arm. The relation between the joint angles and the end effector position is given by the equations (3.1) to (3.3).

$$x = \{L_2*\cos(\theta_2) + L_3*\cos(\theta_2 + \theta_3)\} * \cos(\theta_1) \quad (3.1)$$

$$y = \{L_2*\cos(\theta_2) + L_3*\cos(\theta_2 + \theta_3)\} * \sin(\theta_1) \quad (3.2)$$

$$z = L_1 + L_2*\sin(\theta_2) + L_3*\sin(\theta_2 + \theta_3) \quad (3.3)$$

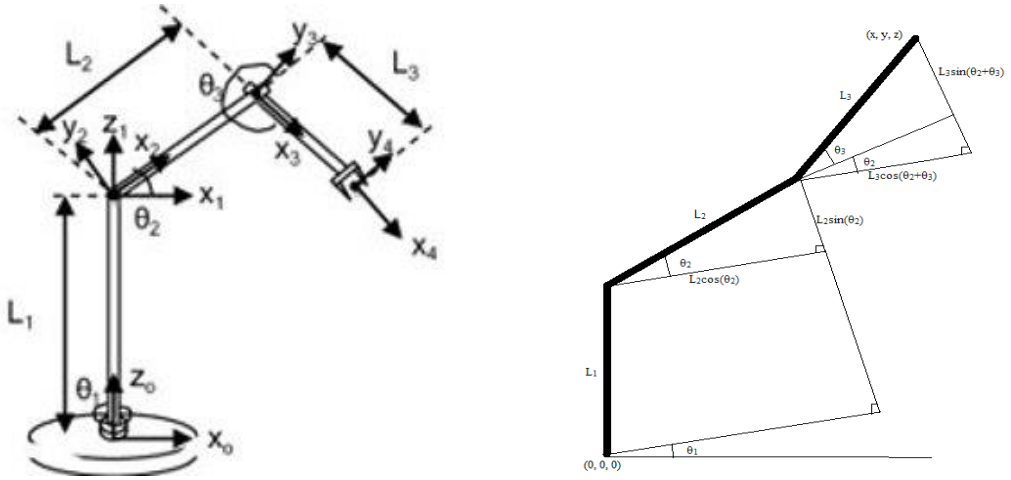


Figure 3.1: Schematic of robotic arm. L_1, L_2, L_3 are the link lengths; θ_1, θ_2, θ_3 are the joint angles; x, y, z is the position of the end effector

Considering L2 = L3 = L and solving the equations (3.1), (3.2) and (3.3) we obtain the joint angles as:

From equations (3.1) and (3.2) we get,

$$\tan(\theta_1) = y/x$$

$$\boxed{\theta_1 = \tan^{-1}(y/x)} \quad (3.4)$$

Squaring and adding both sides of the equations (3.1) and (3.3),

$$x^2/\cos^2(\theta_1) + (z - L_1)^2 = L^2 * \sin^2(\theta_2) + L^2 * \cos^2(\theta_2) + L^2 * \sin^2(\theta_2 + \theta_3) + L^2 * \cos^2(\theta_2 + \theta_3) + 2 * L^2 * \cos(\theta_2) * \cos(\theta_2 + \theta_3) + 2 * L^2 * \sin(\theta_2) * \sin(\theta_2 + \theta_3)$$

Using trigonometry identities → $\sin^2(A) + \cos^2(A) = 1$; $\cos(A - B) = \cos(A)\cos(B) + \sin(A)\sin(B)$

$$x^2/\cos^2(\theta_1) + (z - L_1)^2 = 2 * L^2 + 2 * L^2 * \cos(\theta_3)$$

Rearranging the equation,

$$\cos(\theta_3) = ((x / \cos(\theta_1))^2 + (z - L_1)^2 - 2 * L^2) / 2 * L^2$$

$$\boxed{\theta_3 = \cos^{-1}((x / \cos(\theta_1))^2 + (z - L_1)^2 - 2 * L^2) / 2 * L^2)} \quad (3.5)$$

From equation (3.1),

$$x / \cos(\theta_1) = L * \{\cos(\theta_2) + \cos(\theta_2 + \theta_3)\}$$

Using trigonometry identity → $\cos(A) + \cos(B) = 2*\cos((A+B)/2)* \cos((A-B)/2)$

$$x / \cos(\theta_1) = 2 * L * \cos(\theta_3/2) * \cos((2 * \theta_2 + \theta_3) / 2)$$

$$x / (2 * L * \cos(\theta_3/2) * \cos(\theta_1)) = \cos((2 * \theta_2 + \theta_3) / 2)$$

$$\theta_2 + (\theta_3 / 2) = \cos^{-1}(x / (2 * L * \cos(\theta_3/2) * \cos(\theta_1)))$$

$$\boxed{\theta_2 = \cos^{-1}(x / (2 * L * \cos(\theta_1) * \cos(\theta_3/2))) - (\theta_3 / 2)} \quad (3.6)$$

The equations (3.4) to (3.6) show that the joint angles can be obtained from the end-effector position.

Note: *In the equation (3.5), if $(z - L_1)$ is negative, the term $(z - L_1)^2$ becomes positive which gives incorrect value of θ_3. As shown in equation (3.6), θ_2 is dependent on θ_3 which causes inaccuracy in θ_2. This error can be fixed by making both the angles θ_2 and θ_3 negative at the end.*

3.2 The Four Joint Angle Configurations

From the equations (3.4), (3.5) and (3.6) we obtain one of the joint angle configurations of the robotic arm to move the end effector to a specific position. In the hardware model, the servo motors that have been used rotate in the range of 0 to 180 degrees, which may not be the range in which the obtained joint angles lie. Therefore, we need an alternate set of angles which lie in this range. There are four possible joint angle configurations which leads to the same end effector position. Hence, we calculate all of these possibilities and choose a configuration which suits our robot model.

Figure 3.2 shows the four possible joint angle configurations. The first configuration of the joint angle is found using the equations (3.4), (3.5) and (3.6) as shown in the Figure 3.2a. The other configurations are derived below using the first configuration.

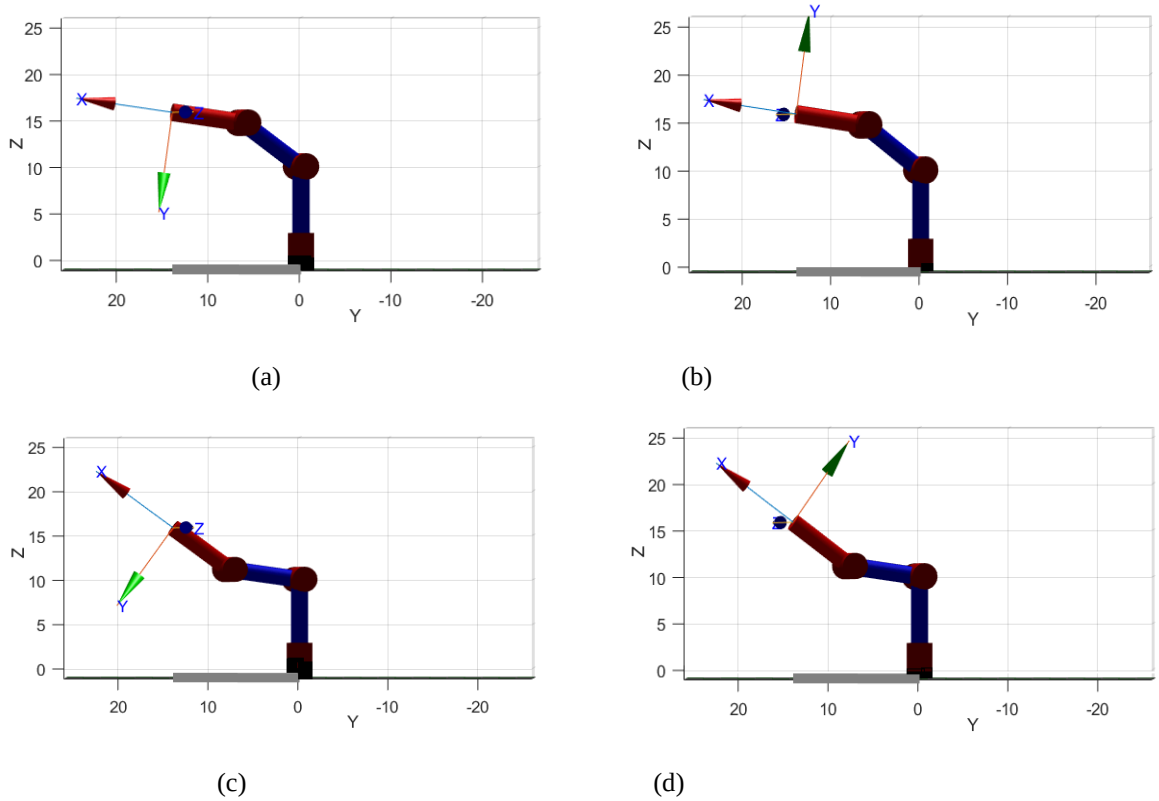


Figure 3.2: The four configurations of the robot a) from the equations; b) changing θ1, θ2 and θ3; c) & d) changing θ2 and θ3.

The configuration shown Figure 3.2b is found by changing the joint angles θ_1, θ_2 and θ_3.

$\theta_{1,\,config2} = 180° \pm \theta_{1,\,config1}$ (if $\theta_{1,\,config1} < 180$ add else subtract)

$\theta_{2,\,config2} = 180° - \theta_{2,\,config1}$

$\theta_{3,\,config2} = -\theta_{3,\,config1}$

The other two configurations shown in the Figure 3.2c and 3.2d are obtained by changing the joint angles θ_2 and θ_3.

$\theta_{3,\,config3} = 360° - \theta_{3,\,config1}$

$\theta_{2,\,config3} = \theta_{2,\,config1} - \theta_{3,\,config3}$

$\theta_{3,\,config3} = -\theta_{3,\,config1}$

$\theta_{3,\,config4} = 360° - \theta_{3,\,config2}$

$\theta_{2,\,config4} = \theta_{2,\,config2} - \theta_{3,\,config4}$

$\theta_{3,\,config4} = -\theta_{3,\,config2}$

After solving the above equations, we get 4 sets of joint angle configurations. Now we can choose the configuration which is suitable for the hardware model.

3.3 Trajectory

Trajectory planning is a subset of the problem of navigation or motion planning. Trajectory can be defined as "Generating a time schedule for how to follow a path given constraints such as position, velocity, and acceleration" or "Generating a feasible path from a start point to a goal point. A path usually consists of a set of connected waypoints". Trajectory generation computes the time evolution of motion for the robot. Trajectories can be defined in joint space or in Cartesian space. They are then directly provided as the input for the controller. Trajectories are important because they enable the system to ensure feasibility, safety and comfort.

Trajectories might also be classified as coordinated trajectories or independent trajectories, according to the synchronization property. For coordinated trajectories the motions of all joints (or of all Cartesian components) start and end at the same instant. It means that all joints have the same time law. While independent trajectories are timed independently according to the requested displacement and robot capabilities. In our model we start the motion of all the actuators at the same instant and the actuator is stopped when the specified angle is reached.

In our model, trajectory points are generated from the obtained joint angles. In order to obtain a smooth trajectory, we have defined constant angular velocities for each revolute joint of the robotic arm. Given the initial and final joint angles, a trajectory is created. The trajectory points are equally spaced. The trajectory points are generated by adding the angular velocity vector as specified by the user to the current position vector to get the next point. This process is repeated to get the complete sequence of the trajectory.

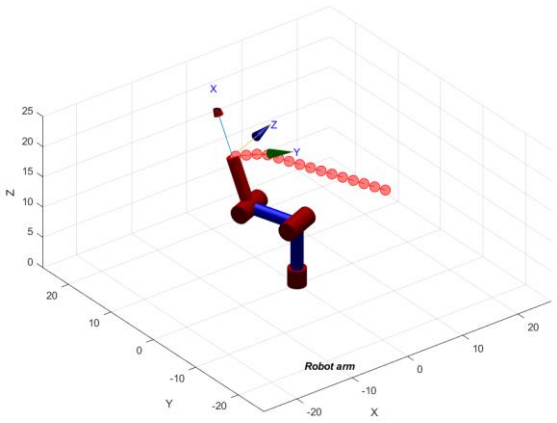


Figure 3.3: Trajectory

Hence, when we generate trajectory, we obtain complete control over the position, velocity and acceleration of the robot. The point to be noted is that number of waypoints change based on the angular displacement needed to reach the final destination which helps put less pressure on the motors as our model is an open loop control system. Since, generating the trajectory points of the robot helps in efficient motion planning, it helps reduce the power consumption of the actuators.

3.4 Flowcharts

In the flowchart shown in Figure 3.4, it can be seen that the desired position and the gripper activation are given as an input to the system. Using the equations (3.4), (3.5) and (3.6), we obtain one of the joint angle configurations of the robotic arm. This configuration is then used to derive the remaining joint angle configurations by using the equations mentioned in section 3.2.

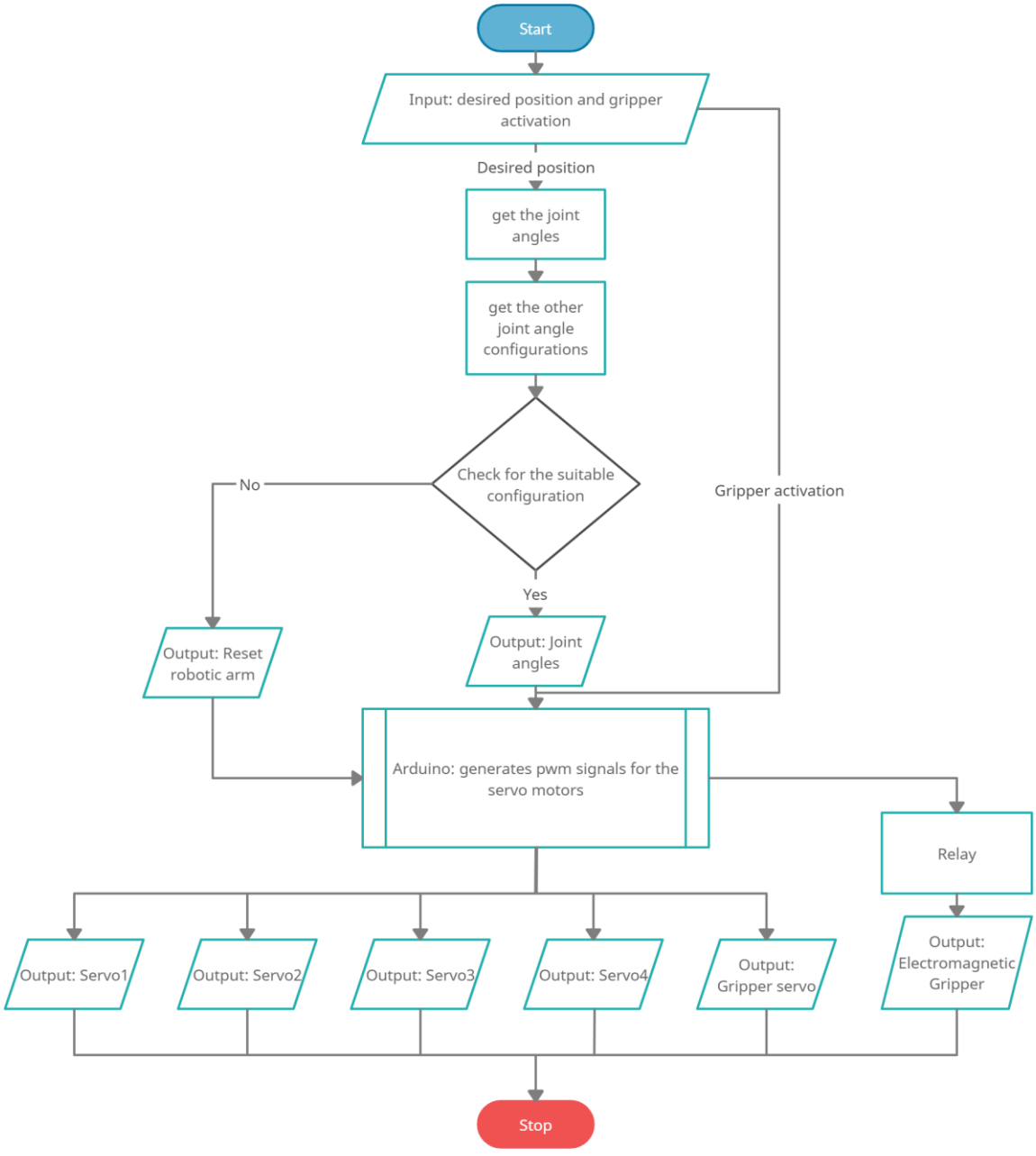


Figure 3.4: Flowchart describing the motion of robotic arm to move to a given point.

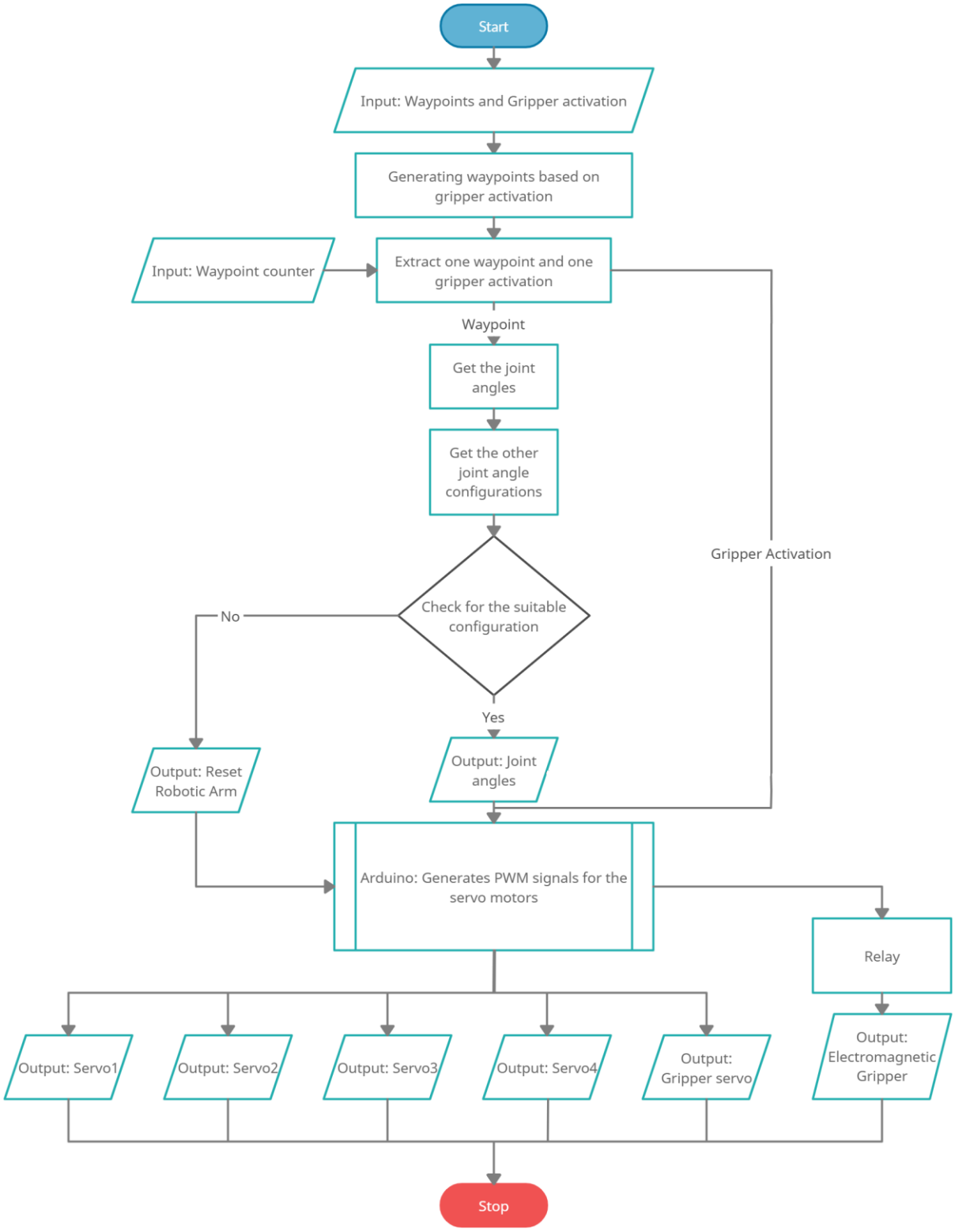


Figure 3.5: Flowchart describing the motion of the robotic arm through a set of waypoints.

We then choose the best suited configuration for the model and provide it as input to the Arduino. The servo motors then make the corresponding angles to reach the required configuration. The grippers are activated as per the user requirement.

The flowchart shown in Figure 3.5 is similar to the flowchart shown in Figure 3.4 with small differences. All the waypoints to be reached are given as the input to the system. These waypoints are manipulated using the gripper activation so as to activate the gripper in correspondence with the waypoints. Now, for each waypoint the respective joint angles are calculated and passed to the Arduino. The Arduino will activate the actuators which in turn controls the motion of the robotic arm and the gripper is activated at the user defined points.

CHAPTER IV
IMPLEMENTATION AND DESIGN

Chapter IV

IMPLEMENTATION AND DESIGN

In the first section of this chapter, we will be discussing about the simulation of the robotic arm. We will see how to build a serial link robotic arm using the robotics toolbox created by Peter Corke. We will go through each of the functions that are used to achieve two goals. The first goal is to move the robotic arm to a given position, and the second goal is to move the robotic arm through a set of waypoints.

In the later part of this chapter, we will design the hardware model of the robotic arm. We will see how to implement the same goals as those of the simulated model using Simulink and Arduino. We will look into the most important aspect of the model called the joint angles subsystem which is used to get the best joint angles for the given waypoint. We have used various electronics components that are controlled by Arduino such as servos and relay. These components work in a synchronised manner in order to build the complete system. We will see how to design and build two types of grippers – a mechanical gripper and an electromagnetic gripper. All the components are assembled to complete the structure of the robotic arm.

4.1 Simulation

Robot simulation capabilities are expanding every day, becoming a key part of the integration process for robot integrators with the technical savvy to leverage this technology. The main advantage of robot simulation is that it provides proof of concept and proof of design so that flaws aren't built into robotic systems. This benefit is realized every time a robot successfully processes a part. Proof of concept and design ensures the best robotic system is installed and that it works just as intended. Simulation allows us to study the structure, characteristics and the function of a robot system at different levels of details each posing different requirements for the simulation tools. As the complexity of the system under investigation increases the role of the simulation becomes more and more important.

Most robot simulation software is compatible with common programming languages like C++, Python, Java, LabVIEW, URBI or MATLAB which are easily accessible. The whole point is to work out any flaws in design before the system is installed. In this project we are using one of the most widely used programing language for robotics, MATLAB. MATLAB provides us with many toolboxes which simplifies the process of building robots. One of the most widely used

toolbox in MATLAB is the robotics toolbox which is created by Peter Corke. We will be using this toolbox to create our serially linked 3-R robotic arm.

4.1.1 Construction of the simulated Robotic Arm

The origin is located at the centre of the C-space. Our model consists of 4 links which are serially connected by 3 revolute joints. For our convenience we are going to use Cartesian coordinates in the rest of the report.

The robotics toolbox provides us with built-in functions to create simple robotic arm. The two functions that we have used to create the arm are Link and SerialLink functions. The Link function is used to create the individual links of the arm. It takes in parameters such as theta (initial joint angle), d(link offset), a(link lengths) and alpha(twist angle). The SerialLink function takes these links as an array and combines them to give a robotic arm.

Syntax – link = Link(theta, d, a, alpha);

Robot = SerialLink([link1 link2 link3 ……]);

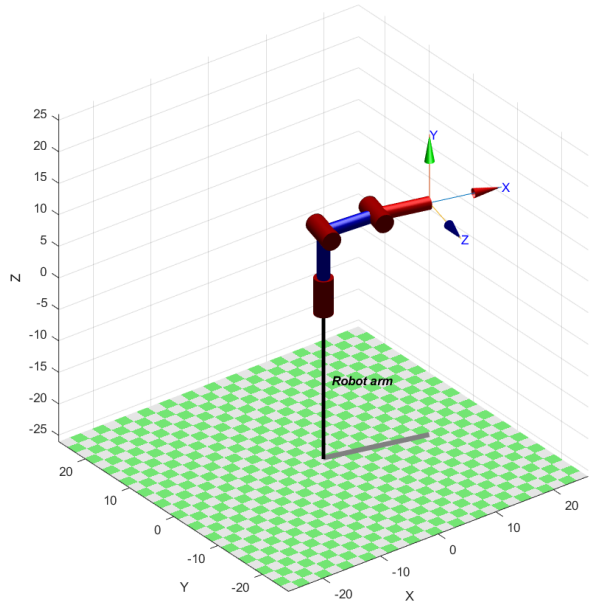


Figure 4.1: Initial position of the robotic arm

As we are using a robotic arm of 4 links with lengths 10cm, 8cm, 8cm and the base link, the initial position of the robotic arm is [16, 0, 10] in the Cartesian coordinates. The joint angles initially are [0 0 0] as shown in the figure 4.1. We have declared the angular velocities for the joint actuators as [0.008 0.004 0.004] respectively.

4.1.2 Go to Point

Our first goal is to move the end effector of the robotic arm to a given point in its C-space. We will see how to generate trajectory points between two points. Now we will look into the details of the functions which helps us achieve the task of go to point.

- get_postion – This function is used to get the position of the end effector. It takes in the input as joint angles and solves the equations (3.1), (3.2) and (3.3). This helps us to map the position of end effector based on the joint angles.

- get_joint_angles – In this function we get the joint angles based on the position of the end effector. This function is the inverse of the function get position. First, we check if the given point is reachable by the end effector using the distance formula. If it is reachable, we get the joint angles using the equations (3.4), (3.5) and (3.6). Then we will correct the error caused because of the equation (3.5) as mentioned in the note of section 3.1. If the position can't be reached then the robot arm is made to reset to its original position.

- get_trajactory – This function helps us to generate the trajectory points based on the two joint angles given to it. The trajectory points are created by incrementing the current joint angles with the given constant angular velocities till the desired angle is reached. Now we concatenate these trajectory points and return these points.

- animate – This function is used to visualize the robotic arm motion. We send in all the trajectory points need to reach the given point to the function. Using a for loop we go through all the waypoints and plot the robot arm for each of these configurations. We give a delay of 0.1 seconds between each trajectory points.

Now we look into how these functions are interlinked to achieve the task of go to point. When we input the desired position to the get_joint_angle function it generates the new joint angle

configuration that it has to reach. If the given point can be reached, then the current joint angle and the new joint angle are fed into the get trajectory function which gives the trajectory points between these two joint angles. Now these points are used to animate the robot model. The Figure 4.2 shows the robot arm after it reaches the final position passing through all the waypoints.

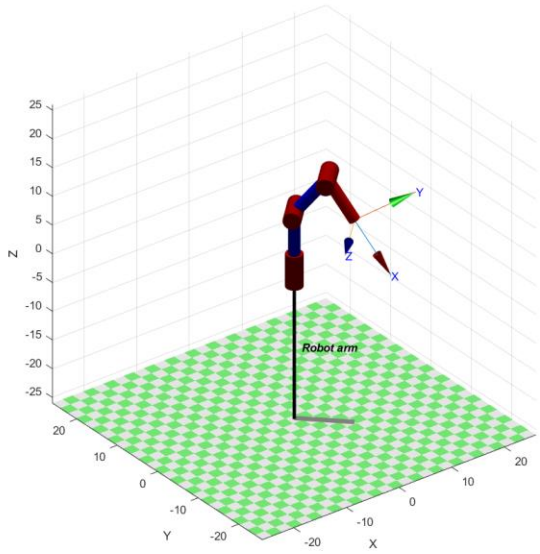


Figure 4.2: Robot arm at desired position

4.1.3 Repeat motion

In this subsection we look into the details of how the robotic arm can repeat its motion for the given waypoints. Based on the number of iterations given, the arm is made to repeat its motion through the defined path.

Repeat motion function – This function is used to get all the trajectory points. First the waypoints are sent as input to this function. Here, if it is possible to reach all the given waypoints then the joint angles are calculated. Based on these joint angles' trajectories are generated for each waypoint. These trajectory points are concatenated to form the entire trajectory which covers all the waypoints. Finally, the trajectory to the first waypoint from the last waypoint is concatenated to complete one cycle of its motion.

These trajectory points are sent to animate function to plot the arm at each of these points. The robotic arm moves through these waypoints for the given number of iterations. In the Figure 4.13 of the section 4.3.1 we will observe the motion of the robotic arm for a given set of waypoints.

4.2 Hardware Implementation

A hardware implementation means that the job is done using a physical device or electronic circuit as opposed to being done by a computer program. A hardware implementation often takes longer to create and that can make it more expensive. It is usually faster in operation and has the advantage that once built it cannot easily be tampered with or reprogrammed.

In this section we will see how we have implemented the hardware model using Simulink and Arduino. Here we will discuss how we have used the four joint angle configurations for a given point. This has been implemented using a subsystem in the Simulink models. We will then look into the details of the Simulink model that is used to achieve the motion of the robotic arm to a desired position. Then we move on to the next Simulink model which deals with the motion of the arm through a set of waypoints for a definite amount of time. In the end we will look into the details of the two grippers that have been designed which are, a mechanical gripper and an electromagnetic gripper.

4.2.1 Subsystem

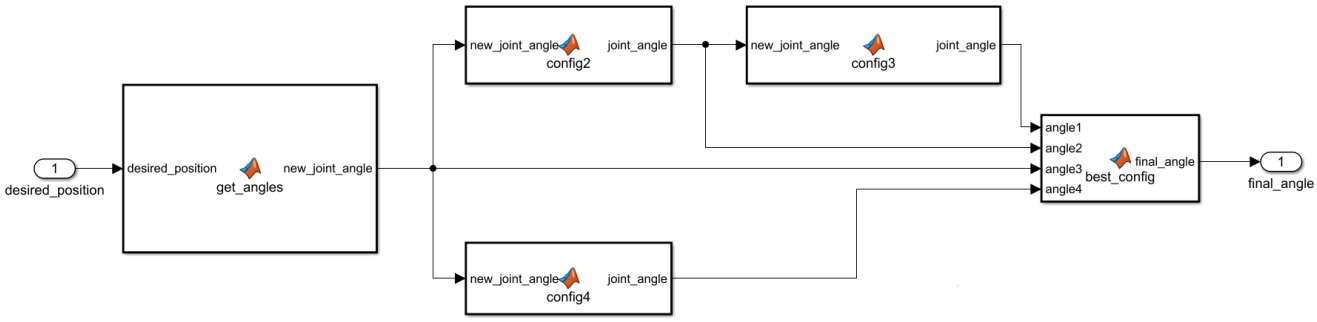


Figure 4.3: get_joint_angles subsystem

The subsystem that we have designed here is the most important aspect of this project. This subsystem is used to get the best joint angle configuration for a given desired position.

The desired position is given as an input to the get_angles function of this subsystem which gives one of the joint angle configurations required for the robotic arm to move to this position. Since

the servo motors can only rotate in a range of 180 degrees, the joint angles generated from the get_angles may not be valid. As discussed in section 3.2, three more configurations are possible and each of them are derived in a separate function using the equations (3.4), (3.5) and (3.6). Now, the configuration which is suitable for our servo motors is chosen in the best_config function which is given as the output of the subsystem. The figure 4.3 Shows the subsystem that has been designed to generate the joint angles for the hardware model.

4.2.2 Go to point

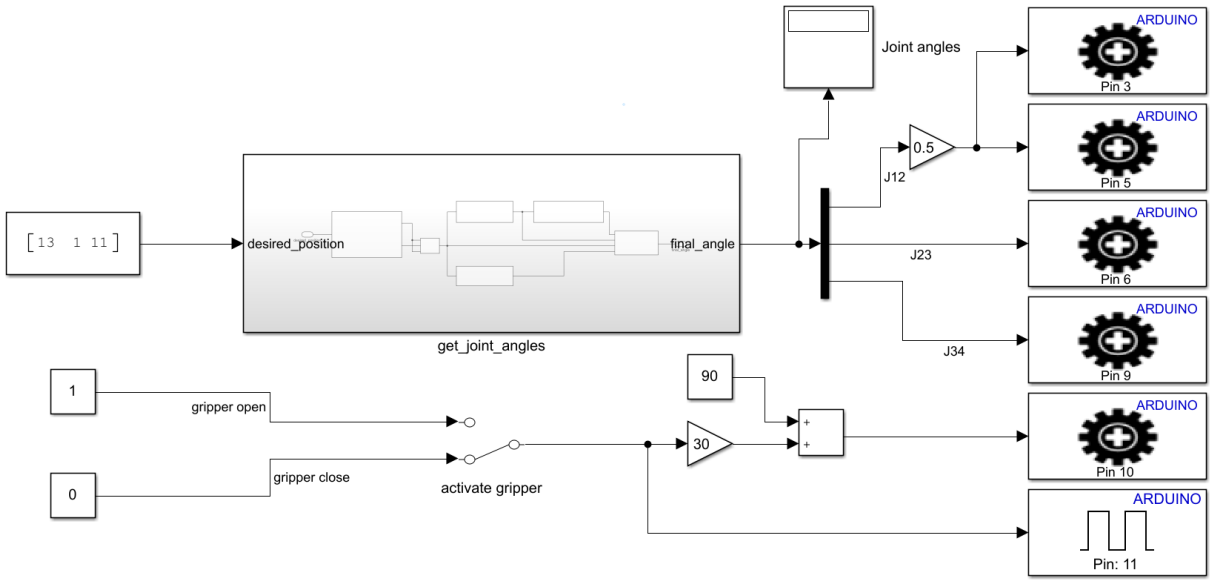


Figure 4.4: Simulink model to move the robotic arm to desired position with gripper

In this subsection, we will discuss the design of the Simulink model used to move the robotic arm to a given point. We will see how the data flows between the blocks and how each of these blocks interpret the data such that the result is obtained.

The desired position is given as an input to the get_joint_angle subsystem. The subsystem returns the best possible joint angle configuration using the methods discussed in the subsection 4.2.1. These joint angles are split into individual angles using a demux and are sent to Arduino to control the three corresponding servo motors. A manual switch is used to control both the grippers.

The Simulink model designed to implement the process described above is shown in Figure 4.4.

4.2.3 Repeat motion

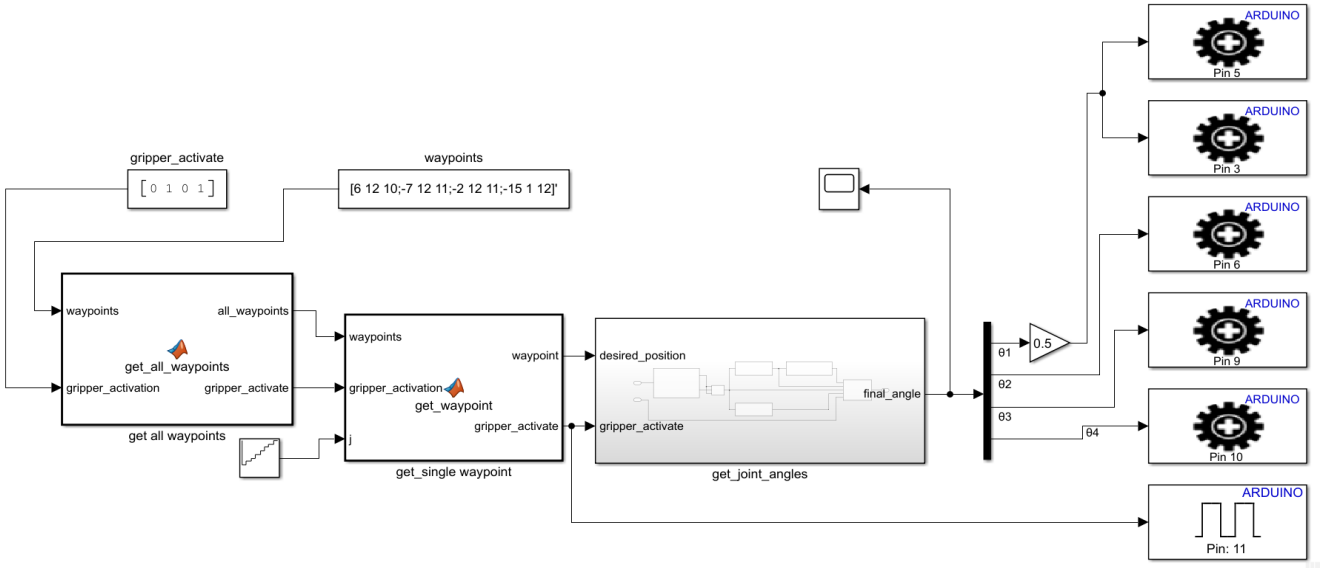


Figure 4.5: Simulink model to move the robotic arm through a set trajectory

In this subsection, we will see how to implement a Simulink model to move the robotic arm through a set of waypoints. We will look into the functions performed by all the blocks used in this model to completely understand it's working.

First, the waypoints and the gripper activation values are sent as an input to the get_all_waypoints function which generates the waypoints and the gripper activation values such that the grippers are activated at the required waypoints. These newly generated waypoints and the gripper activation values are fed to the get_waypoint function which extracts each waypoint and passes it to the subsystem. The subsystem then provides the best joint angle configuration which is given as an input to the Arduino which controls the corresponding servo motors. Each gripper activation value is extracted from the get_waypoint function which is then used to control both the grippers. The Figure 4.5 shows the Simulink model designed to implement the process discussed above.

4.2.4 Grippers

Grippers are one of the most important parts of the robotic arm. In industries many types of grippers are used according to their application. In this project we have designed two types of grippers, a mechanical gripper and an electromagnetic gripper.

The mechanical gripper is mainly controlled using a servo motor. We have designed this gripper in such a way that the gripper jaws are open when the angle of the motor is 120 degrees and closed when the angle of the motor is 90 degrees. The electromagnetic gripper is controlled using a relay. Since we have used an active low relay, the gripper is activated when 0 is sent as the input value and is deactivated when the value 1 is passed as the input. The figure 4.6(a) Shows the mechanical gripper in open position and (b) shows it in closed position.

(a) (b)

Figure 4.6: A mechanical gripper

4.2.5 Components

- Software

  - MATLAB – MATLAB is a programming language created by MathWorks. MATLAB is a numeric computing platform used by millions of engineers and scientists to analyse data, develop algorithms, and create models. We have used the robotic toolbox created by Peter Corke which makes building simple robots easier for simulation. The Toolbox provides many functions that are useful for the study and simulation of classical arm-type robotics.

    We have used MATLAB because it is one of the simplest programming languages for solving mathematical models. It contains many built-in functions which are application specific. As MATLAB is built for mathematical computations, it gives high precision values and faster results. MATLAB provides many features that help us visualize data such as 2D, 3D plots and robot models.

- Simulink - Simulink, an add-on product to MATLAB, provides an interactive, graphical environment for modelling, simulating, and analysing of dynamic systems. It enables rapid construction of virtual prototypes to explore design concepts at any level of detail with minimal effort. For modelling, Simulink provides a graphical user interface (GUI) for building models as block diagrams.

  The reason we are using Simulink is because the data flows between the blocks like signals. It includes a comprehensive library of pre-defined blocks which are used to construct graphical models of systems using drag-and-drop mouse operations. We can add the Simulink support package for Arduino to directly communicate with the Arduino which helps us to tune the parameters.

- Hardware

- Arduino Uno – The Arduino is an open-source electronics platform based on easy-to-use hardware and software which helps us build electronics projects. With the Arduino, we can design and build devices that can interact with the environment. They are able to read inputs with their onboard microcontroller and turn it into an output.

  Arduino boards are relatively inexpensive compared to other microcontroller platforms. As Arduino runs at 5V, it required less power to function. It contains 6 PWM pins which can be used to control different actuators. This allows us to run our hardware model by connecting the Simulink model to the Arduino board. For more details on Arduino refer the Appendix 2. Figure 4.7 shows the Arduino Uno microcontroller.

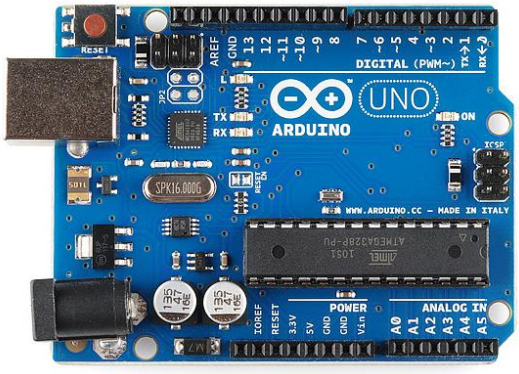


Figure 4.7: Arduino Uno microcontroller

- Servo Motors – In this project we are using TowerPro sg90 servo motors. They are small, powerful, easily programmable, and accurate. Most importantly, though, they allow for near perfect repeatability of motion. They are lightweight with high output power. Servo can rotate approximately 180 degrees (90 in each direction).

  The servo motors help us in getting accurate movement of the robotic arm. It can easily handle heavier weights which are picked by the gripper. The position of these motors can be easily controlled by connecting them to the PWM pins of the Arduino Uno. For more information on servo motors refer the Appendix 3.

Figure 4.8: sg90 Servo motor

- Relay –A relay is used to switch on a circuit by electromagnetic means. Triggering the relay operates the normally open or normally closed contacts. It is frequently used in an automatic control circuit. It is an automatic switch to control a high-current circuit with a low-current signal.

Figure 4.9: 5V single channel Relay module.

We are using a relay in our project to connect to an electromagnetic gripper. Arduino has a maximum voltage output of 5V which is not enough to connect it to a strong electromagnet. Therefore, the use of relay allows us to make an electromagnet which can

be used as a gripper to lift metallic objects. For more details on Relay module refer Appendix 4.

- Battery, breadboard, jumper wires, copper coil – We are using 2 AA batteries to power the electromagnet. Breadboard allows us to easily connect the electronic components used. We have used jumper wires to connect between components. The copper coil is used for making the electromagnetic gripper.

Figure 4.10: Breadboard and Jumper wires.

- Cardboard – For the structure of the entire robot we have made use of cardboard. As it is a very light material it helps in the weight management of the robot. As it is flexible and easy to cut, we don't require high end tools to build the robot body.

4.2.6 Simulink and Arduino Communication

Simulink Support Package for Arduino Hardware enables us to create and run Simulink models on Arduino board. The target includes a library of Simulink blocks for configuring and accessing Arduino sensors, actuators and communication interfaces. Additionally, the target enables you to monitor and tune algorithms running on Arduino board from the same Simulink models from which you developed the algorithms.

Simulink Support Package for Arduino Hardware provides an easy way to create algorithms that use Arduino sensors and actuators by using the blocks that can be added to your Simulink model. The Simulink Support Package generates C-code directly (bypassing the INO-file). Simulink's External mode feature enables you to accelerate the process of parameter tuning by letting you

change certain parameter values while the model is running on target hardware, without stopping the model. When you change parameter values from within Simulink, the modified parameter values are communicated serially to the target hardware immediately. The effects of the parameters tuning activity may be monitored by viewing algorithm signals on scopes or displays in Simulink.

4.2.7 Specifications

The Table 4.1 shows the specifications of joint angles. Here, S_1, S_2, S_3 and S_4 are the servo motors. The servo motors S_1 and S_2 are connected axially to create joint angle J_{12}, so that the joint can rotate complete 360 degrees. The servo motors S_3 and S_4 control the joint angles J_{23} and J_{34} respectively.

Joint	Servos connected	Mass (gm)	Torque at 5V (kg/cm)	Range (degrees)	Dimension (mm, mm, mm)	Operating speed (sec/degrees)
J_{12}	S_1, S_2	160	5	[0, 360]	[22.8, 12.6, 69]	$8.33*10^{-4}$
J_{23}	S_3	80	2.5	[0, 180]	[22.8, 12.6, 34.5]	$1.66*10^{-3}$
J_{34}	S_4	80	2.5	[0, 180]	[22.8, 12.6, 34.5]	$1.66*10^{-3}$

Table 4.1: specifications of joint angles

Link	Weight (gm)	Dimension (mm, mm, mm)
L_1	120	[70, 70, 100]
L_2	50	[80, 5, 20]
L_3	50	[80, 5, 20]

Table 4.2: specifications of links

The Table 4.2 shows the specifications of the links used. L_1, L_2 and L_3 represent the three links that are used in the robot arm. The link lengths are calculated based on the torque and the weight of the servo motors used

We have built two types of grippers – mechanical gripper and electromagnetic gripper.

- Mechanical gripper – This is an angular two jaw gripper which can lift an object with weight up to 150grams. When the servo motor is at 90 degrees the gripper is closed and when it is at 120 degrees it is open. It can hold an object of width up to 4cm. The dimensions of the gripper are 80mm, 70mm and 20mm.

- Electromagnetic gripper – The electromagnetic gripper is used to lift metallic objects of weight up to 100grams. The gripper coil consists of 25 turns. The gripper gets activated when a voltage of 3Volts and current of 50mA is passed through the coil. The dimensions of the gripper are 5mm, 5mm and 50mm.

4.2.8 Circuit diagram

The figure 4.11 shows the circuit connections between the Arduino, servo motors and relay. The V_{cc} of the Arduino is connected to one of the outer sections of the breadboard for all the actuators to activate. Each of the servo motors S_1 to S_5 are connected to the PWM pins 3, 5, 6, 9 and 10 of the Arduino.

The active low relay is connected to the digital pin 11 of the Arduino. the two terminals of the electromagnet are connected to the NO terminal of the relay and the positive terminal of the battery. The negative terminal of the battery is connected to the COM of the relay which completes the electromagnet circuit. The Arduino is connected to the PC where it communicates with the Simulink model. The table 4.3 shows the pin connections to the Arduino.

Pin Number	Type	Component
3	PWM	Servo 1
5	PWM	Servo 2
6	PWM	Servo 3
9	PWM	Servo 4
10	PWM	Servo 5
11	Digital	Relay

Table 4.3: Pin connections

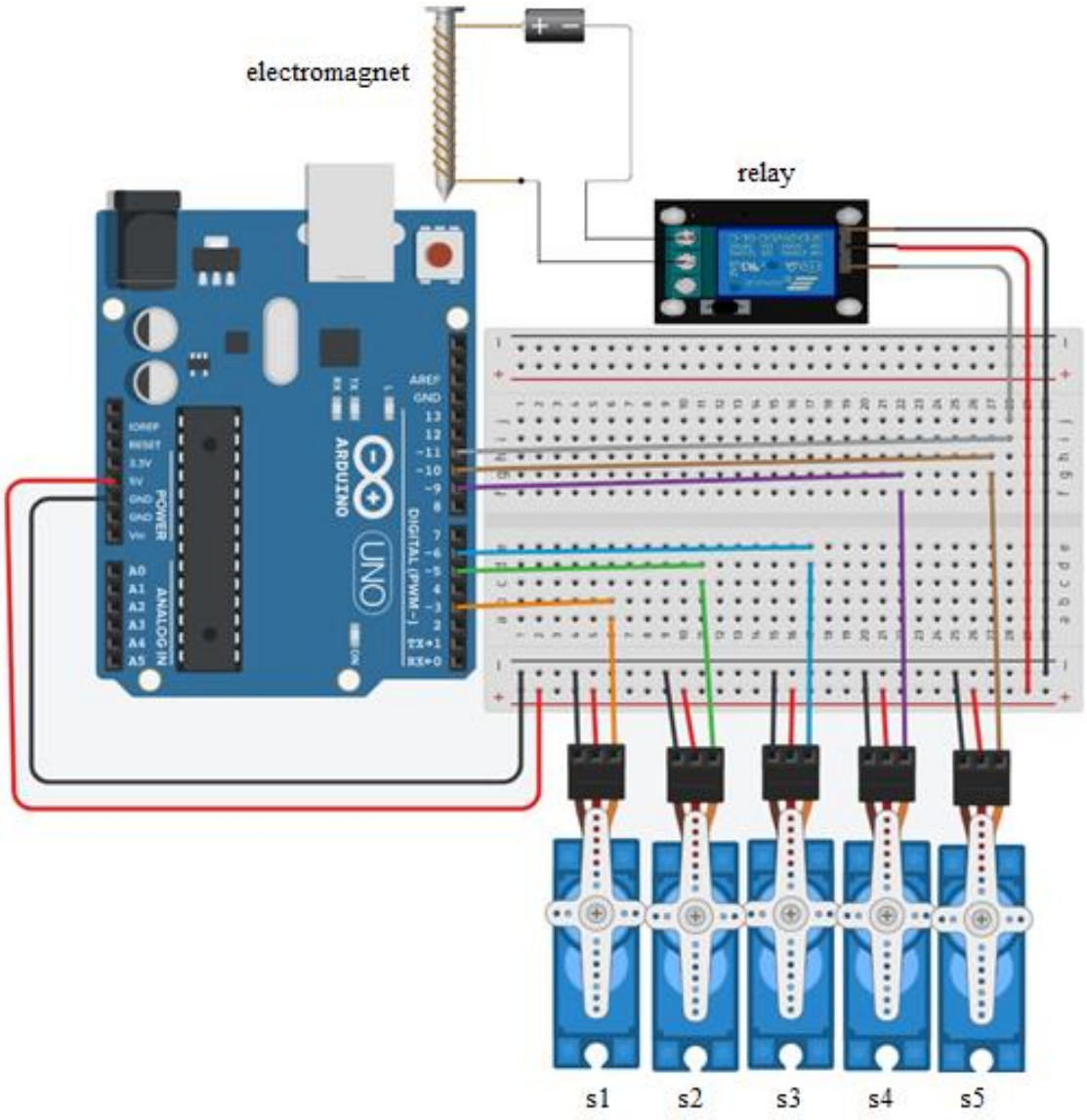


Figure 4.11: Circuit diagram of the hardware model.

4.2.9 Assembly of Robot

All parts of the robot arm are made using cardboard with the specified dimensions as mentioned in the Table 4.2. The base link houses two servo motors connected axially. For the remaining links, one end is connected to the servo motor and the other is connected to the other link. Therefore, the base joint angle (J_{12}) can rotate a total of 360 degrees.

The mechanical gripper has two well-spaced jaws that is supported by a small cardboard sheet. The servo motor is connected to one end of these jaws by a metal connector to execute the motion of the gripper. The electromagnet is made by winding a copper coil around an iron nail and is connected to the relay.

The remaining electronic components are connected to the Arduino using a breadboard as shown in figure 4.12

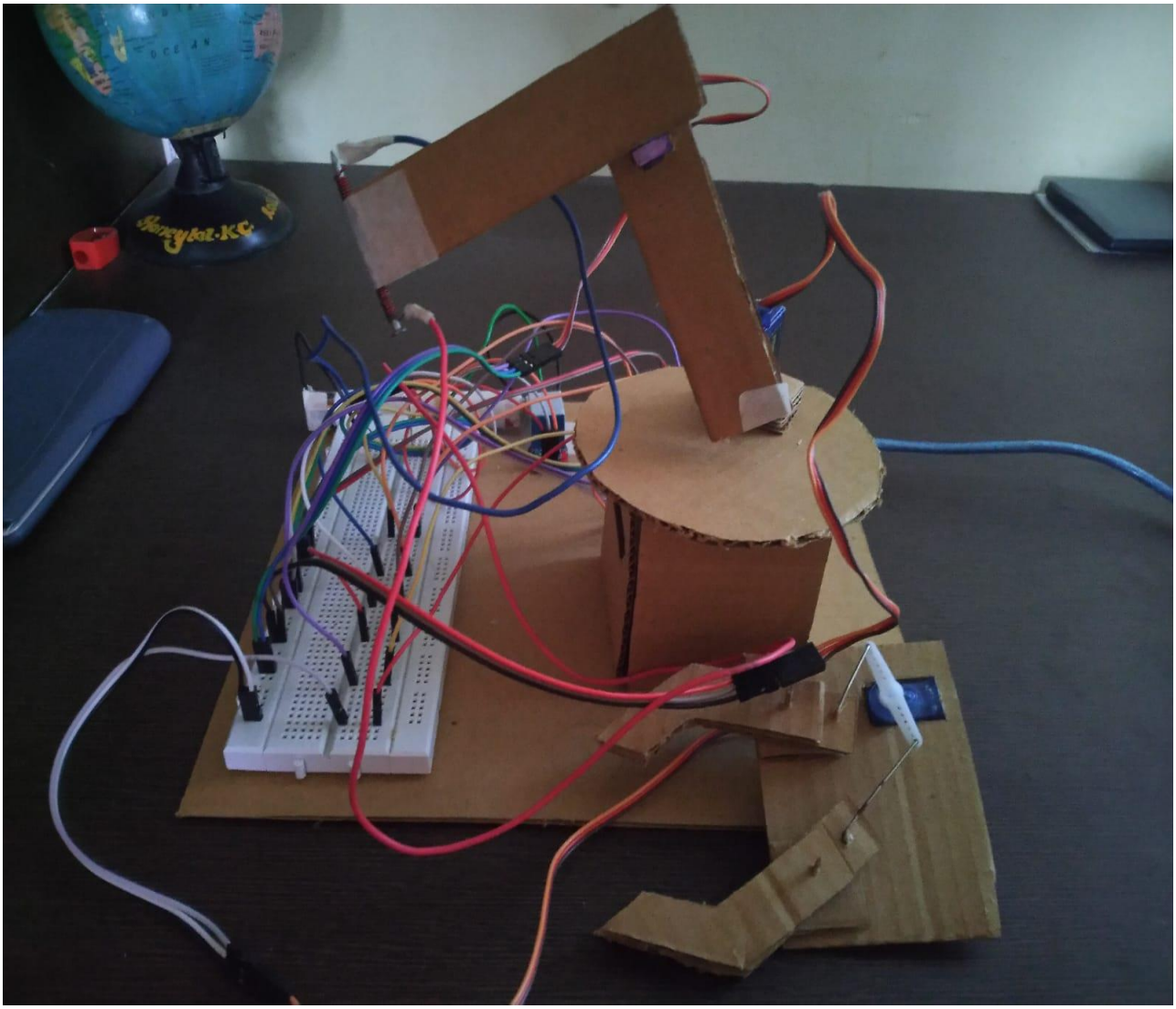


Figure 4.12: Assembly of the robotic arm

4.3 Observations

Observations act as the primary source of validation of a project as they quantify the results of the project. They allow researchers to publish their findings. Observations act as reference point for most researchers. It allows for them to have a quick glance at the findings of the project and hence allows them to judge if the project is worth looking further into. Observations tend to contain a comprehensive and detailed explanation of results.

In this section we will discuss about some of the observations done during the process of making the project. First, we shall observe the behavior of the simulated model when it is given a desired position to reach. Then we will see how the model repeats motion for a given set of waypoints. Next, we will observe the functioning of the hardware model to go to a point in its configuration space and to move through a given set of waypoints.

4.3.1 Simulation

- Here we will see how the model behaves when a desired position is given as input. For the given desired position, we will observe the joint angles, number of trajectory points generated and the overall motion of the robotic arm based on these parameters.

  When we input the desired point as x = 6 cm, y = -6 cm and z = 9 cm to the function, we get the joint angles as θ_1 = -45 deg, θ_2 = 51 deg and θ_3 = -115 deg. Initially the joint angle of the robotic arm is θ_1 = 0 deg, θ_2 = 0 deg and θ_3 = 0 deg. These initial and final joint angles are sent to the get trajectory function to generate 52 trajectory points.

  It has been observed that the robot arm is moving through the waypoints and at the end the simulated arm has reached desired position. The Figure 4.13 shows the motion of the simulated model of the robotic arm through some of the generated trajectory points.

- When the desired position cannot be reached such as x = 40cm, y = 30cm and z = 40cm then the joint angles that are generated by the function get_joint_angles is θ_1 = 0 deg, θ_2 = 0 deg and θ_3 = 0 deg which indicates that the robotic arm doesn't move. Hence, we don't animate the simulated model.

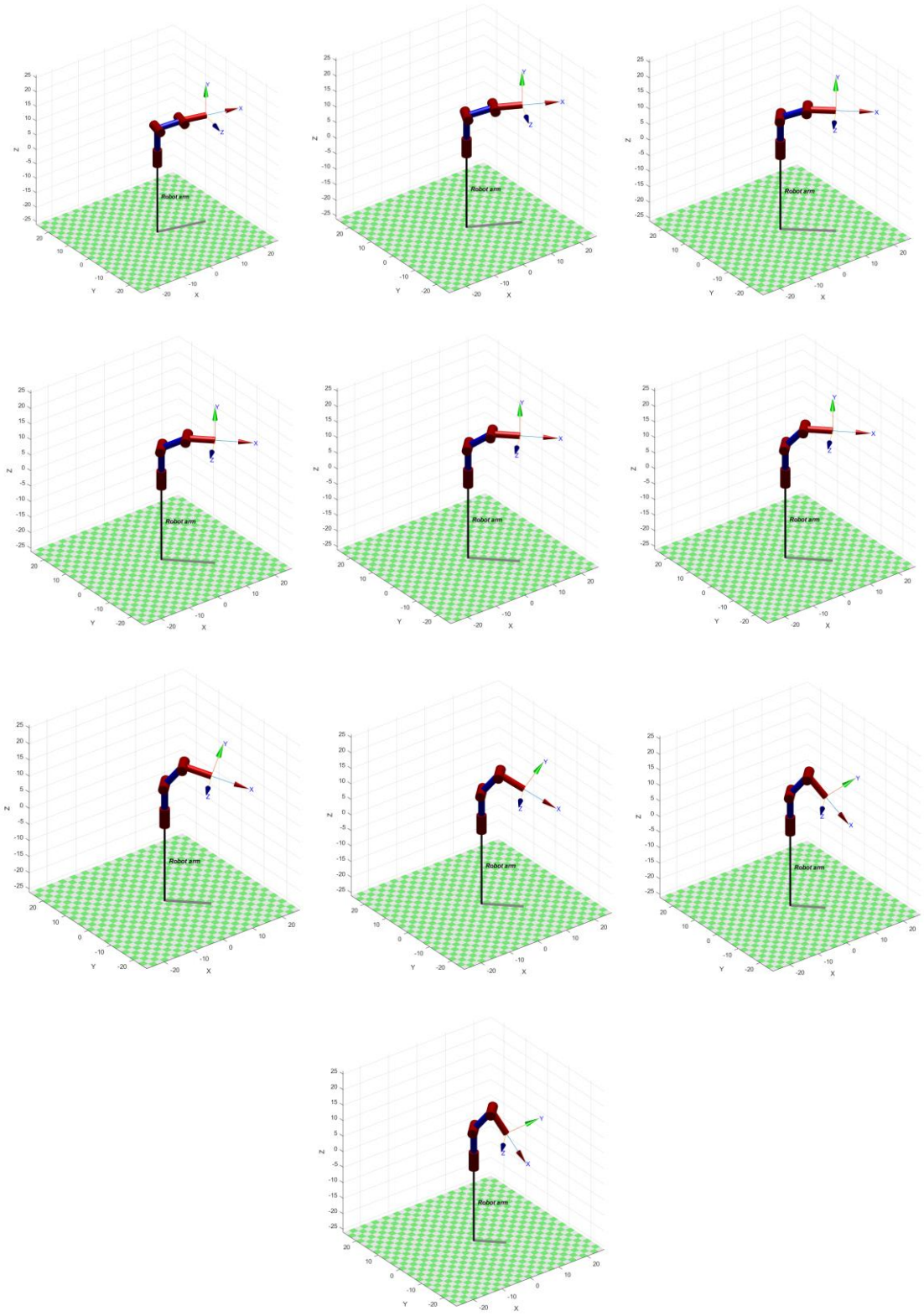


Figure 4.13: Motion of the robotic arm through some of the trajectory points to reach the given desired point

- Now we will observe the motion of the robot when a set of waypoints are given as input. We will see how the robot repeats the cycle for a given number of iterations.

  When the waypoints [11, 10, 15], [5, 6, 20], [-9, 8, 20], [-4, -8, 19] and [-4, 10, 18] are given as an input to the function repeat_motion, 177 trajectory points are generated for one cycle or iteration based on the angular velocites given. When the number of iterations is set to three the robot repeats its movement through the waypoints thrice. From the Figure 4.14 it can be observed that the robot arm moves through the waypoints when these trajectory points were provided.

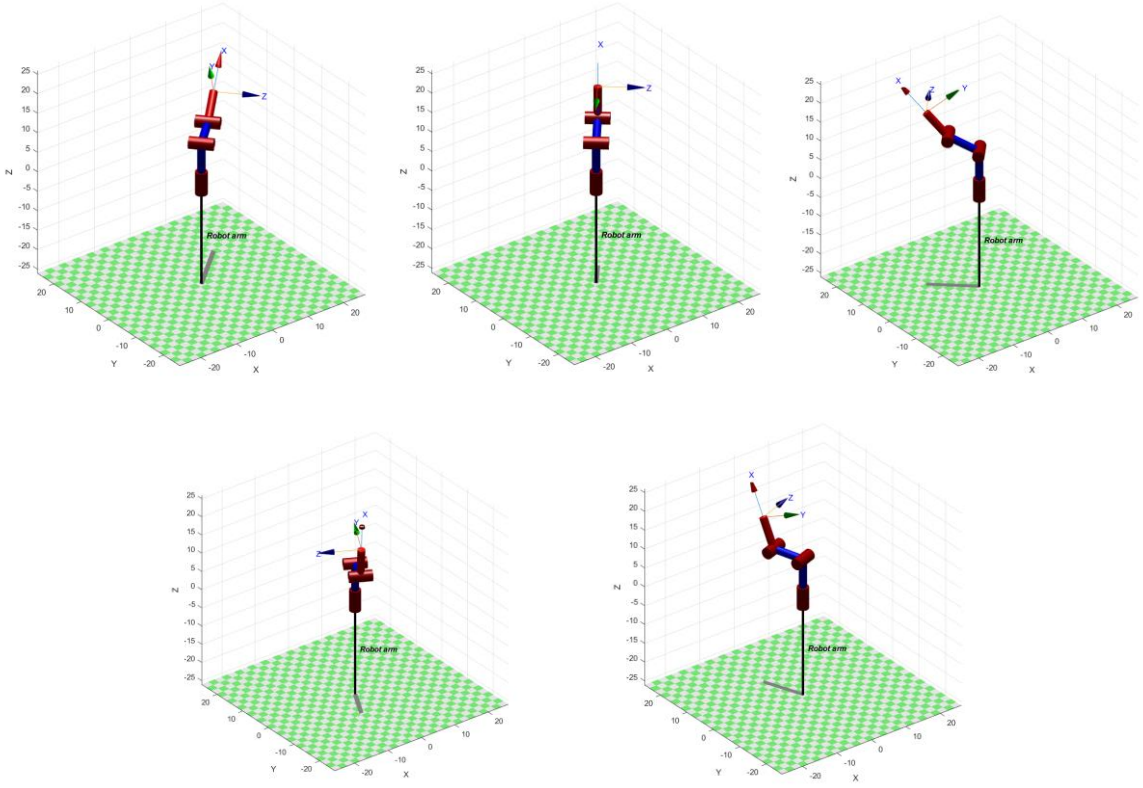


Figure 4.14: Robot arm at different waypoints

4.3.2 Hardware model

- Go to Point - Now we will see how the hardware model of the robotic arm behaves when it has to move to a given point. We will see how the gripper is activated or deactivated based on the status of the switch in the Simulink model.

When the desired position is given as x = -4 cm, y = -8 cm and z = 19 cm to the get_joint_angle subsystem, we get the joint angles as θ_1 = 242 deg, θ_2 = 97 deg and θ_3 = 165 deg. Now these angles are fed into the Arduino which controls the servo motors. When the manual switch in the Simulink model is on, the electromagnetic gripper is activated and the mechanical gripper is closed.

The Table lists the different joint angles corresponding to the given desired position. As we can see, the desired position mentioned at serial number 5 cannot be reached, so the robot arm is reset to its initial position.

Sl.No.	X (cm)	Y (cm)	Z (cm)	Θ1 (deg)	Θ2 (deg)	Θ3 (deg)
1.	6	12	10	62	56	156
2.	-7	12	11	120	64	148
3.	-2	12	11	98	54	170
4.	15	1	10	2	69	130
5.	-13	17	14	0	90	90
6.	13	1	11	4	59	160

Table 4.4: Position and corresponding joint angles

- Repeat motion – Now for the given set of waypoints we will see how the hardware model functions. We will also see how the motion of the robotic arm is synchronized with both the grippers.

When the set of waypoints are given as [11, 10, 15], [5, 6, 20], [-9, 8, 20], [-4, -8, 19] and [-4, 10, 18] and the gripper activate is given as [0, 1, 0, 1, 1] to the get_all_waypoints function, it generates nine waypoints which correspond to the gripper activation. The joint angles are generated using the joint angle subsystem discussed in the section 4.2.1. Now the joint angles and the gripper activation signals are sent to the Arduino which controls the servo motors and the relay.

The Figure 4.15 shows the graph of the joint angles produced by the Simulink model. The graph clearly show that the robotic arm is made to stay in its position until the gripper is activated or deactivated. We also observe that when we increase the sample time of these signals the motion of the robot arm is slowed down.

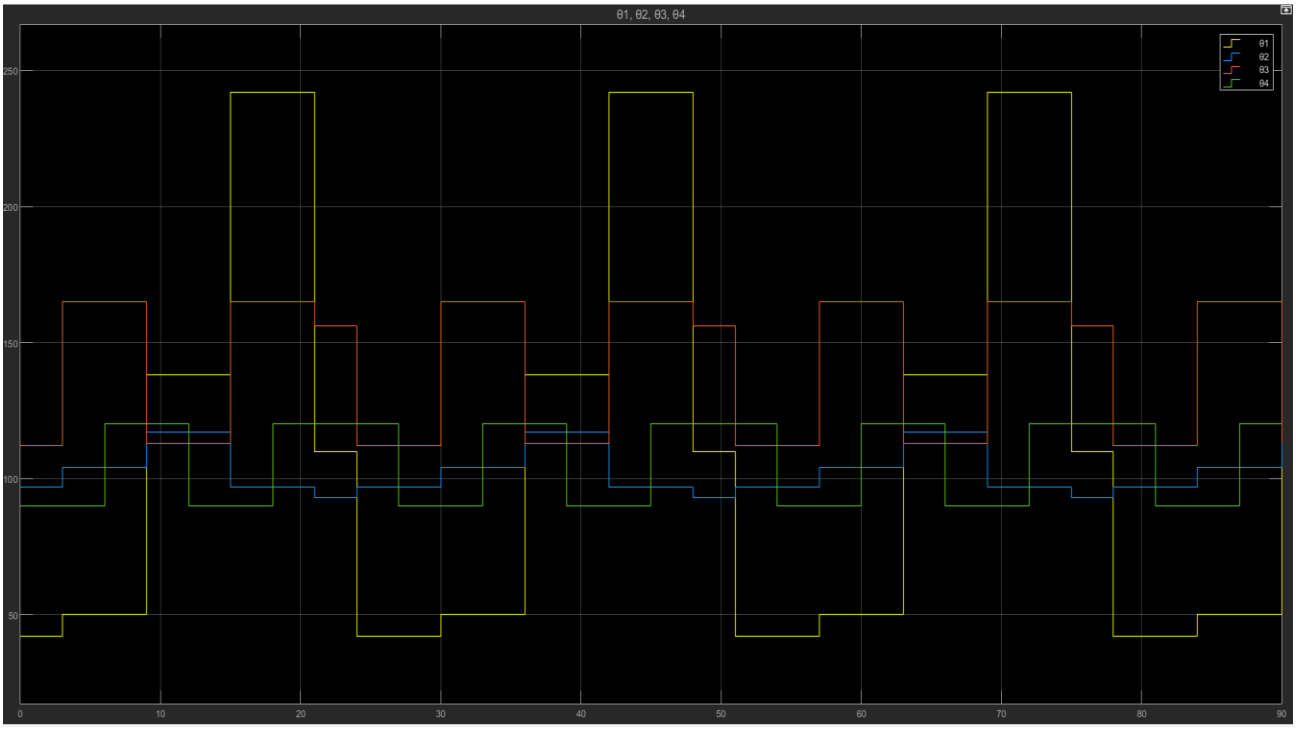


Figure 4.15: Variations in joint angles corresponding to the waypoints

CHAPTER V
CONCLUSION

Chapter V

CONCLUSION

Overall, the objectives of this project have been achieved which are developing the hardware and software for robotic arm, implementing the pick and place operation and also testing the robot so that it meets the purpose of the project. The study has further shown that the robotic arm can be reprogrammed in order to adapt it to a variety of tasks. This could prove to be very useful in an industrial environment, especially in manufacturing, packaging and mining industries. It is believed that the use of the arm for educational and instructional purposes will stimulate the interest of students in the field of robotics, especially in a developing country like India, where robotics is yet to enter into an advanced stage.

Design, manufacturing and analysis of robotic arm has been successfully completed. It is concluded that this robotic arm is working properly under the specified working envelope with good accuracy. During the process of making and developing the project, we have learnt Arduino, MATLAB and Simulink and how to interface between them. This knowledge has been put into practice and it has been ensured that it is suitable for the purpose of the project.

5.1 Result

1. We have successfully simulated the model of a 3R-robotic arm.

2. We have implemented a hardware model of the robotic arm with gripper.

3. We have completely automated the process of gripper activation and motion of the robotic arm through several waypoints.

5.2 Future Work

- The number of degrees of freedom of the robotic arm can be increased in order to expand its workspace, thereby making it more versatile. For example, if you have 6 DOF, a given end-effector pose is only reachable in a single configuration of the robot.

- We plan on using a stronger and durable material for the body of the robotic arm instead of cardboard. This improves weight management and can be used for tougher applications.

- We want to improve the accuracy by using more precise actuators. This will help in getting a smoother trajectory for the robotic arm. Presently we are using servo motors which takes in input in the range of 0 to 180 degrees as whole numbers. In the future we plan to use gear motors with encoders whose angles range from 0 to 360 degrees so that the angular displacement and angular velocities can be easily controlled.

- Incorporation of multiple styles of grippers each catered to a specific type of task. For example, creating gripper to pick and preserve samples of environment for labs, separating materials based on magnetism, telescopes, rubble clearing etc. We plan on building vacuum grippers, pneumatic grippers and hydraulic grippers.

- We would like to implement a closed loop system rather than an open loop system by using PID controllers. This leads to a faster and precise movement of the robotic arm. In the feedback loop, we would like to use sensors such as camera, inertial measurement unit (IMU) and accelerometers so that it can be applied in precision driven applications such as medical surgeries and telescopes.

- We would like to introduce more capabilities to the model by adding techniques such as object recognition and colour sensing to detect and removal faulty items. We can implement bar code and QR code scanners for better segregation of packages in assembly lines.

- We plan on installing the robotic arm on mobile robots which allows us to navigate into uncharted territories to discover and collect data which cannot be pursued by humans. Therefore, it can be used in deep sea exploration, space explorations and archaeological discoveries.

5.3 Links to our project

GitHub link: https://github.com/Electronics-Creed/Robotic-Arm

Drive link:

https://drive.google.com/drive/folders/1rfMxBnnn4IuiM7Yzvbfr_llIJidcPY3G?usp=sharing

YouTube: https://www.youtube.com/playlist?list=PLSWRiv_s7diOQ1Fk6TX-yL9LCNg870Yoa

REFERENCES

[1] Robotics toolbox, MATLAB robotics toolbox by Peter Corke

https://www.petercorke.com/RTB/r9/html/SerialLink.html

[2] Introduction to Robotics Toolbox for MATLAB, Yang Shen Ph.D. Candidate, Bionics Lab, UCLA MAE 263B

http://bionics.seas.ucla.edu/education/MAE_263D/RTB_MATLAB_Intro.pdf

[3] ROBOMECHTRIX, Forward Kinematics | Puma560 | Peter Corke Toolbox

https://www.youtube.com/watch?v=jFfUDp2Hh5w

[4] Robotics Toolbox for MATLAB® Release 10, Peter Corke

https://petercorke.com/wp-admin/adminajax.php?juwpfisadmin=false&action=wpfd&task=file.download&wpfd_category_id=27&wpfd_file_id=1050&token=&preview=1

[5] Work with Arduino Hardware

http://sti.tice.ac-orleans-tours.fr/spip2/IMG/pdf/working_with_arduino_hardware.pdf

[6] Simulink Support Package for Arduino Hardware User's Guide, Mathworks

https://in.mathworks.com/help/pdf_doc/supportpkg/arduino/arduino_ug.pdf

[7] How To Make Arduino Robotic Arm Controlled with Smartphone - Cardboard DIY, Ge Creative

https://www.youtube.com/watch?v=zs_keUVE26A&t=337s

[8] Simulink Support Package for Arduino Hardware, MathWorks, Simulink Team

https://www.mathworks.com/matlabcentral/fileexchange/40312-simulink-support-package-for-arduino-hardware

[9] Design, Simulation and Fabrication of A 5-Dof Robotic Arm (with implementation of inverse kinematics), ORIDATE Ademola Ayodeji

https://www.academia.edu/27913612/DESIGN_SIMULATION_AND_FABRICATION_OF_A_5_DOF_ROBOTIC_ARM_with_implementation_of_inverse_kinematics_

[10] Pick and Place Robotic Arm: A Review Paper, Sharath Surati, Shaunak Hedaoo, Tushar Rotti, Vaibhav Ahuja, Nishigandha Patel

https://www.irjet.net/archives/V8/i2/IRJET-V8I2311.pdf

[11] Survey on Robotic Arm Controlling Technique, V.Priyanka, E.Thangaselvi, Dept of ECE, PSNA College of Engineering and Technology, Dindigul, Tamil Nadu, India

https://www.ijeter.everscience.org/Manuscripts/Volume-5/Issue-2/Vol-5-issue-2-M-04.pdf

[12] Review on Development of Industrial Robotic Arm Rahul Gautam, Ankush Gedam, Ashish Zade, Ajay Mahawadiwar

https://www.irjet.net/archives/V4/i3/IRJET-V4I3402.pdf

[13] Wire Control Robotic Arm, Ghodadra Nikesh, Patel Dhruvil, Luvani Maulik, Lalit Dhanani, Mr. Paresh K. Katariya

http://www.ijsrd.com/articles/IJSRDV6I30693.pdf

[14] Design, Manufacturing and Analysis of Robotic Arm with SCARA Configuration, Kaushik Phasale, Praveen Kumar, Akshay Raut, Ravi Ranjan Singh, Amit Nichat

https://www.researchgate.net/publication/338621403_Design_Manufacturing_and_Analysis_of_Robotic_Arm_with_SCARA_Configuration

[15] Review on Design and Development of Robotic Arm Generation-1, Sneha Bire, Vaibhav Pawar, Shubham More, Komal More, Reshma Mule

https://ijisrt.com/wp-content/uploads/2018/04/%E2%80%9CReview-on-Design-and-Development-of-Robotic-Arm-Generation-1%E2%80%9D.pdf

[16] 3 Axis Robotic Arm , Abhivyakti Sharma, Kshitija Kanase, Vipul Pandey, C. K. Bhange

https://www.ijresm.com/Vol.2_2019/Vol2_Iss6_June19/IJRESM_V2_I6_25.pdf

[17] Controlling robot arm using Kinect

https://www.researchgate.net/publication/336676027_robot_arm_project

[18] Design, Analysis and Implementation of a Robotic Arm- The Animator Md. Anisur Rahman , Alimul Haque Khan , Dr. Tofayel Ahmed , Md. Mohsin Sajjad

http://www.ajer.org/papers/v2(10)/ZJ210298307.pdf

[19] Modeling and Control of 2-DOF Robot Arm Nasr M. Ghaleb and Ayman A. Aly

https://www.ijeert.org/papers/v6-i11/3.pdf

APPENDIX 1 – Inverse Kinematics

Kinematics is the science of motion. In a two-joint robotic arm, given the angles of the joints, the kinematics equations give the location of the tip of the arm. Inverse kinematics refers to the reverse process. Given a desired location for the tip of the robotic arm, what should the angles of the joints be so as to locate the tip of the arm at the desired location. There is usually more than one solution and can at times be a difficult problem to solve.

This is a typical problem in robotics that needs to be solved to control a robotic arm to perform tasks it is designated to do. In a 2-dimensional input space, with a two-joint robotic arm and given the desired coordinate, the problem reduces to finding the two angles involved. The first angle is between the first arm and the ground (or whatever it is attached to). The second angle is between the first arm and the second arm.

Depending on your robot geometry, IK can either be solved analytically or numerically. Analytic solutions mean that you can derive, in closed-form, an expression for the joint positions given the desired end effector position. This is beneficial because you do all the work offline and solving IK will be fast. As with everything in engineering: if you have an exact model of your system, you should take advantage of it!

Numerical solutions are generally slower and less predictable than analytic solutions, but they can solve harder problems than analytic solutions (we expand on this below). However, these solutions introduce uncertainty in the form of initial conditions, optimization algorithm choice, or even random chance. So, you may not get the answer you want.

The 3D pose of your end effector can be specified by 6 parameters: 3 for position and 3 for orientation. Technically, you can derive an analytical solution if there are up to 6 nonredundant joints in your manipulator, assuming the desired position is reachable.

APPENDIX 2 – Arduino Uno

Overview - The Arduino Uno is a microcontroller board based on the ATmega328 (datasheet). It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz ceramic resonator, a USB connection, a power jack, an ICSP header, and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started.

The Uno differs from all preceding boards in that it does not use the FTDI USB-to-serial driver chip. Instead, it features the Atmega16U2 (Atmega8U2 up to version R2) programmed as a USB-to-serial converter. Revision 2 of the Uno board has a resistor pulling the 8U2 HWB line to ground, making it easier to put into DFU mode.

Revision 3 of the board has the following new features:

- pinout: added SDA and SCL pins that are near to the AREF pin and two other new pins placed near to the RESET pin, the IOREF that allow the shields to adapt to the voltage provided from the board. In future, shields will be compatible both with the board that use the AVR, which operate with 5V and with the Arduino Due that operate with 3.3V. The second one is a not connected pin, that is reserved for future purposes.
- Stronger RESET circuit.
- Atmega 16U2 replace the 8U2.

"Uno" means one in Italian and is named to mark the upcoming release of Arduino 1.0. The Uno and version 1.0 will be the reference versions of Arduino, moving forward. The Uno is the latest in a series of USB Arduino boards, and the reference model for the Arduino platform; for a comparison with previous versions, see the index of Arduino boards.

Summary

Microcontroller: ATmega328

Operating Voltage: 5V

Input Voltage (recommended): 7-12V

Input Voltage (limits): 6-20V

Digital I/O Pins: 14 (of which 6 provide PWM output)

Analog Input Pins: 6

DC Current per I/O Pin: 40 mA

DC Current for 3.3V Pin: 50 mA

Flash Memory: 32 KB (ATmega328) of which 0.5 KB used by bootloader

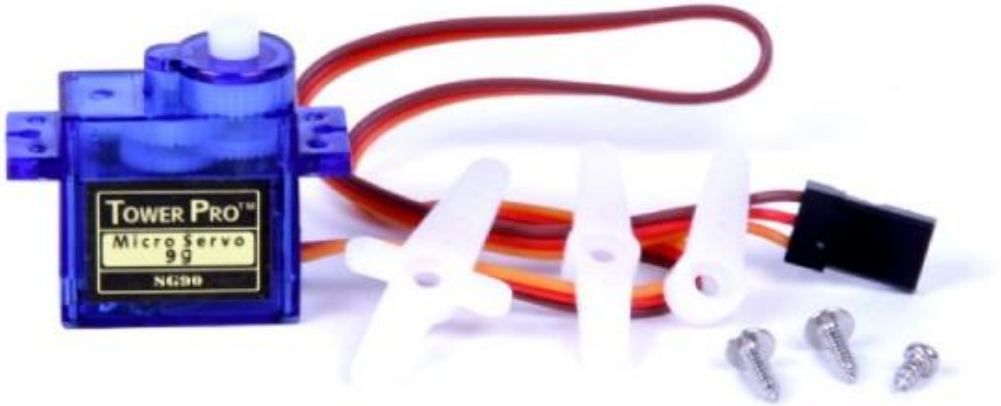
SRAM: 2 KB (ATmega328)

EEPROM: 1 KB (ATmega328)

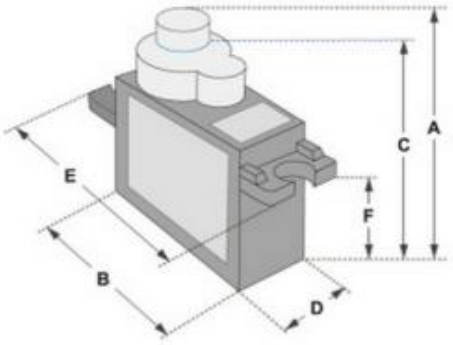
Clock Speed: 16 MHz

APPENDIX 3 – Servo Motor

SERVO MOTOR SG90 DATA SHEET

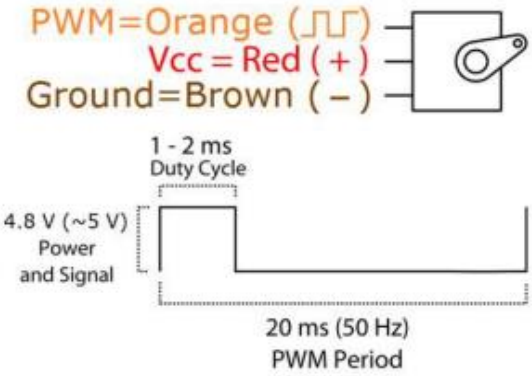


Tiny and lightweight with high output power. Servo can rotate approximately 180 degrees (90 in each direction), and works just like the standard kinds but smaller. You can use any servo code, hardware or library to control these servos. Good for beginners who want to make stuff move without building a motor controller with feedback & gear box, especially since it will fit in small places. It comes with a 3 horns (arms) and hardware.



Dimensions & Specifications
A (mm) : 32
B (mm) : 23
C (mm) : 28.5
D (mm) : 12
E (mm) : 32
F (mm) : 19.5
Speed (sec) : 0.1
Torque (kg-cm) : 2.5
Weight (g) : 14.7
Voltage : 4.8 - 6

Position "0" (1.5 ms pulse) is middle, "90" (~2ms pulse) is middle, is all the way to the right, "-90" (~1ms pulse) is all the way to the left.



APPENDIX 4 - Relay

Relays are most commonly used switching device in electronics. Let us learn how to use one in our circuits based on the requirement of our project.

Before we proceed with the circuit to drive the relay, we have to consider two important parameters of the relay. Once is the Trigger Voltage; this is the voltage required to turn on the relay that is to change the contact from Common->NC to Common->NO.

Features

- Trigger Voltage (Voltage across coil) : 5V DC
- Trigger Current (Nominal current) : 70mA
- Maximum AC load current: 10A @ 250/125V AC
- Maximum DC load current: 10A @ 30/28V DC
- Compact 5-pin configuration with plastic moulding
- Operating time: 10msec Release time: 5msec
- Maximum switching: 300 operating/minute (mechanically)