gg

by Md. Chayan Ali

\6-DOF Robotic Arm Manipulator

Submitted By

Syed Hassan Mehmood Sherazi

Ammar Jamshed

Supervised By

Dr. M. Umair Khan

Dr. Kamran Javed

Department of Computer Engineering
National University Technology (NUTECH)
Islamabad, Pakistan
2023

6-DOF Robotic Arm Manipulator

By

Syed Hassan Mehmood Sherazi

Ammar Jamshed

A Project Report Submitted to the Department of Computer Engineering in partial fulfilment of the requirements for the degree of Bachelor of Science in Computer Engineering

Faculty of Computer Engineering

National University Technology,

Islamabad, Pakistan

July, 2023

BSCEN	**HASSAN, AMMAR**	**2023**

CERTIFICATE OF APPROVAL

It is certified that the project titled "6-DOF Robotic Arm Manipulator" carried out by Syed Hassan Mehmood Sherazi, Reg. No. F19604015 and Ammar Jamshed Iqbal, Reg. No. F19604016. under the supervision of Dr.Umair Khan, Assistant Professor and Dr. Kamran Javed, Associate Professor, National University of Technology (NUTECH), Islamabad.

Supervisor: ------------------------

Engr. Dr. M. Umair Khan

Assistant Professor

Dept. of Computer Engineering

National University of Technology (NUTECH), Islamabad

HOD: --------------------------

Engr. Dr. Kamran Javed

Head of Department

Dept. of Computer Engineering

National University of Technology (NUTECH), Islamabad

ACKNOWLEDGMENT

All the acclamation and appreciations are for Almighty ALLAH who created the universe and bestowed the mankind with knowledge and wisdom to search for its secrets. The Team feels great pleasure and honour to express our deepest sense of gratitude, sincere feelings and regards to our supervisor Dr. M. Umair Khan and Dr. Kamran Javed for their efficient guidance, tremendous help and special way of advice for the completion of my studies.

DEDICATED

TO

OUR PARENTS

Abstract

This project features the comprehensive design, fabrication and implementation of a 6-degree-of-freedom (DOF) robotic arm assembly incorporating custom 3D printed components. The goal is to create a cost-effective and user-friendly robotic arm that can be controlled via a graphical user interface (GUI). This project aims to provide versatile and practical tools for education, coding and prototyping. The first phase of the project involved the detailed design and modeling of the robot arm using computer-aided design (Solidworks) software. Special attention is paid to kinematics and mechanical aspects to ensure smooth and precise robot movements. Using 3D printing technology, the team was able to fabricate most parts of the robotic arm, resulting in a lightweight yet robust construction that significantly reduced manufacturing costs. The second phase focuses on assembling the robotic arm assembly, requiring careful calibration and alignment of all joints to achieve optimal performance. Safety precautions are also included to prevent potential hazards during operation. The highlight of the project was creating a user-friendly GUI interface to control the individual joints of the robot arm. The graphical interface is designed to provide intuitive controls that allow users to easily control and precisely command the robot's arm movements. The GUI is implemented using programming languages and programming tools for seamless integration with the robot arm's microcontroller. Extensive testing and evaluation was conducted to evaluate the performance and functionality of the robotic arm and GUI interface. The results show that the robotic arm exhibits smooth and precise movements, suitable for a variety of applications such as pick and place, simple assembly operations, and training demonstrations.

Table of Contents

LIST OF FIGURES:

LIST OF TABLES

Chapter 1

INTRODUCTION

A robotic arm is a piece of machinery that is meant to perform tasks and move in ways that are analogous to those of a human arm. It's a crucial part of robotic systems and finds widespread use in a wide range of fields and contexts. These devices can be used for anything from straightforward pick-and-place activities to intricate, delicate maneuvers. They have had a profound impact on the manufacturing, research, and healthcare industries. These robots can readily navigate complicated paths thanks to a combination of predetermined code and automatic trajectory construction. In 1961, the first robotic arm was introduced. In 1954, George Devol submitted a patent application for what would become the robotic arm. The robotic arm was patented by him in 1961. Unimate, which George Devol and Joseph Engelberger created, is widely recognized as the first company to commercially produce and sell articulated robotic arms for use in industry. The production, assembly, and pick/place operations all benefit greatly from the usage of articulated robotic arms. Along the assembly line, multiple articulated robots perform a wide variety of duties in a consistent, accurate manner.

1.1 Parameters of Robotic Arm

1.1.1 Load Bearing Capacity:

The term "carrying capacity" describes how much weight the robot arm can lift and carry by means of the grip. The robot's lifting capacity is established by the design of the arm's frame and the torque delivered to the joints by the motor. From 0.5 kg up to 1000 kg, this robot has you covered.

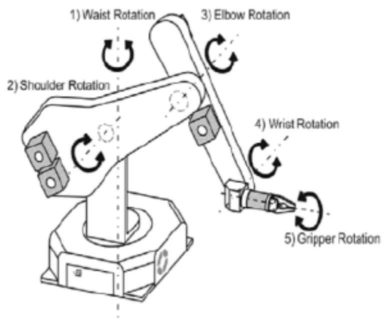


Figure 1.1 Manipulator Motion

1.1.2 Precision:

Precision refers to the ability of a fixed robot arm to reach a specific location or accurately follow a predetermined trajectory or path based on planned instructions. The level of accuracy is usually higher in well-built and well-designed machines. For a robot arm, the absolute positioning accuracy can be around +-1mm.

1.1.3 Repeatability:

The repeatability of a robotic arm refers to its ability to perform the same action consistently every time the Programmable Logic Controller (PLC) is run. Correct calibration and optimized programming are essential to ensure repeatability. This measure allows the robot to make the necessary adjustments before each cycle, preventing the accumulation of errors that can affect accuracy over time.

1.1.4 Speed:

Speed, in the context of a robot arm, refers to the robot's ability to travel a certain distance in a given time, usually measured in mm/s for an articulated robot arm. The speed of the robot is adjusted based on factors such as precision, accuracy and the weight of the load to be carried. When high precision and heavier loads are required, the speed of the robot is reduced, and on the contrary, it can be increased for tasks that prioritize fast movements and lower load requirements.

1.2 Robotic System Components:

Robot system components can be divided into three main groups:

1.2.1 Mechanical Structure:

The mechanical structure of the robot consists of interconnected links and mechanisms that allow movement from one link to another. Articulated Robotic Arms are ideal for fast and lightweight applications, or can adopt continuous joints often used in pick-and-roll operations.

1.2.2 Actuator:

Actuators include motors and mechanisms responsible for providing the power necessary to move mechanical structures. AC servo motors are preferred in modern articulated robot arms due to their lightweight nature, high power-to-weight ratio, closed-loop feedback capability, and good torque at high speeds. The power of the engine is then transmitted through gears, belt mechanisms, or transmission shafts that allow power to be transmitted between various planes.

1.2.3 Controller:

The controller becomes the brain of the Articulated Robotic Arm, providing instructions to move in fixed coordinates or follow a continuous trajectory. In industrial settings, Programmable Logic Controllers (PLCs) are often used as controllers for these robot arms. In the early days when the UNIMATE 001 was introduced, vacuum tubes and transistors were used, and standard PLC controllers for industrial robot arms.

1.2.4 End Effector:

An end effector is a device or handle attached to the end of a robot's arm that allows it to interact with the environment or perform certain tasks. A variety of effects are applied based on the application. For example, if the robot needs to pick up and pick up objects, a welding torch is attached for welding operations. There are a variety of grip mechanisms, with overall strength, opening/closing time, ease of manufacture, and other factors influencing the choice of grip mechanism for a particular application.

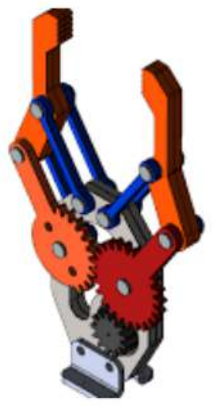


Figure 1.2 End Effector

1.3 Overview:

This FYP aims to create a fast-moving, lightweight robotic arm with a high degree of accuracy and precision. The robot arm will include an intuitive graphical user interface (GUI) for controlling each individual joint and autonomous trajectory capabilities for easy picking up and carrying out. The goal is to create a robotic arm that is inexpensive and amenable to mass production, with the ability to conduct quick movements that are well suited to a wide range of delivery and pickup applications. It would be impossible to exaggerate the value of robots to the world's manufacturing sectors. Increased productivity and quicker turnaround times are only two of the many benefits that have resulted from the widespread adoption of robotic arms in industry. As more and more processes are being automated in the manufacturing industry, robotic arms have become standard equipment on assembly lines and other production facilities. Therefore, both academic engineering programs and private industry are investing heavily in research and development to create robotic arms that meet or exceed current standards for precision, dependability, efficiency, and cost.

The expanding significance of robotic hands and their potential uses across industries is acknowledged in this FYP. One such industry is the medical area, where they could be used in procedures like surgery. The developed robot arm takes advantage of cutting-edge Topology / Shape Optimization methods to select and position its frame components for maximum structural efficiency. Beyond picking up and carrying, this lightweight and adaptable robotic arm can hold welding tools, cutting tools, paint supplies, and can even be utilized as a 3D printer. The robot's purpose is to promote the development of robotics and its function in academia and industry by satisfying these demands and meeting these needs.

1.4 Problem Statement

The complexity of today's technical products, such as cars, rocket engines, and cell phones, need very precise and reproducible manufacturing methods, especially for mass production. Therefore, there is a trend toward using robots to perform previously human tasks. However, the high cost of entry and ongoing upkeep, in addition to the complexity of these systems, provide obstacles. Therefore, **"there is a need for robust and efficient robotic arms that can be mass-produced**

to facilitate the widespread adoption of robotic automation in industry."

1.5 Specifications of Proposed Solution:

All the mechanisms in this final year project were picked because of their high accuracy, repeatability, simplicity, and ease of manufacture and assembly. Here are a few details to consider.

- Our design's rotatable joints simplify the construction process.
- All components are 3D printed using a printer that the author built from scratch.
- Incorporating 3D-printed components reduces the time and money needed for production and assembly.
- Easy-to-use graphical user interface for manipulating robotic arm.

Several motorized components provide a robot arm its mobility and speed. The horizontal motion of the arm is made possible by a rotating mechanism at its base, which is turned by a NEMA 17 stepper motor. A bearing fixed to a circular disc provides support for the main ring, allowing it to spin freely. For continuous rotation of the robot arm's base during pick and place operations, a Nema-17 motor was opted for due to its high torque output and capacity to drive the entire structure. In order to move quickly and smoothly, a powerful motor is required. A gear system transmits power from the Nema-17 motor to the robot arm's base. This gearbox is built around an internal gear and a Nema-17 motor. Gear ratios are chosen to optimize torque output and transmission efficiency. With this setup, the robot arm's base may rotate easily and precisely, facilitating its smooth and efficient performance during pick and place tasks.

The first articulating arm joint is powered by a pair of NEMA 17 stepper motors that coordinate their rotation. The range of motion in the shoulder is adequate because of this setup. Once again, gear mechanisms are employed here. The elbow joint is controlled by a second NEMA 17 stepper motor located in the second link. Because of this motor, the arm may extend and flex at the elbow, giving the user more mobility. The robotic arm's end effector is driven by a pair of MG996R servo motors for precise motion. The hands are given the freedom to control items and carry out complex activities thanks to the assistance of these motors. Finally, an SG90 servo motor is used to open and close the gripper's claws in order to grasp the object. This motor regulates the motion of the handle, guaranteeing precise control over the grip. When a NEMA 17 stepper motor is paired with a servo motor, and the entire assembly is mounted on a circular drive and supported by bearings, the resulting robot arm has the potential to perform a wide range of tasks with ease and precision.

1.6 Purpose of This Project:

Our 3D robotic arm kit was designed to give robotics fans, amateurs, and students a simple way to experiment with and learn about the many applications of robotic arms. Features include exact control of each robot joint via a user-friendly GUI interface, as well as automatic trajectory control and the ability to pick and place objects.

Educational tools: Kits for building a robotic arm can be used in the classroom to teach students about mechanics, automation, and robotics. Users can learn the fundamentals of robotics engineering through the installation procedure and hands-on practice.

Versatile functionality: The suite's many features, such as a pick-and-place simulator, help customers see how robots are employed in the workplace. This adaptability inspires risk-taking and novel approaches to problems.

Automated Trajectory: With Automated Trajectory, the user can instruct the robot arm to move in a specific pattern. It's a great tool for spreading knowledge about programming and automation.

GUI Interface: The graphic user interface offers a simple and straightforward interface for manipulating the robot arm. The user experience as a whole is enhanced by the simplicity with which users can set up and manage their various connections.

3D Printing and Cost-Efficiency: Assembling a robotic arm using 3D-printed parts is both cheap and simple to replicate. As a result, more people will be able to learn about and use robots.

Ease of Assembly: The design of the manipulators places an emphasis on simplicity of assembly so that people with varied degrees of technical expertise can utilize them. Easy assembly is ensured with clear instructions and intuitive parts.

Skill Development: By constructing and controlling robot arms, users can hone their abilities in mechanical assembly, electronics, programming, and resolving issues.

Personalization: Because the robot arm is 3D-printed, users can play around with different configurations and try out new ideas.

1.7 Applications of Robotic Arm:

Manipulators are used for a wide range of jobs in the industrial sector, particularly those that call for high levels of accuracy, speed, and reproducibility. They find applications in "pick and place" using autonomous trajectory and user based GUI, as well as in heavy industry like automotive production lines.

Before purchasing a programmable manipulator, it is vital to understand the unique needs of each situation and project. When deciding the kind of programmed manipulator to buy, it's important to think about the specifics of the job at hand. The jobs best suited for this manipulator include those that are highly repetitive, precise, and potentially dangerous for human employees. Programmable manipulators are versatile machines that can be instructed to perform a wide range of tasks with high efficiency, precision, and dependability. Industrial manipulators have become more accessible to a wider range of businesses thanks to a dramatic drop in their price over the past decade. These manipulators have found widespread application in a variety of settings, including laboratories, testing and sample management, factories, workplace automation, assembly robots, and machine accessibility.

Chapter 2

LITERATURE REVIEW

2.1 Related Technologies

Robotics and automation are at the heart of this endeavor, with a particular emphasis on the investigation of novel robot arm technologies and manipulator mechanisms. Some of these technologies are built in from the start, providing the framework for the whole thing. At the same time, the team serves as a source of inspiration and motivation, driving everyone to do their best and increase output.

2.1.1 Kuka KR 30-3

The KUKA KR 30-3 robot arm is made by the renowned German robotics company KUKA Robotics. Some of the characteristics of the KUKA KR 30-3 robot arm, as of my most recent revision are as follows

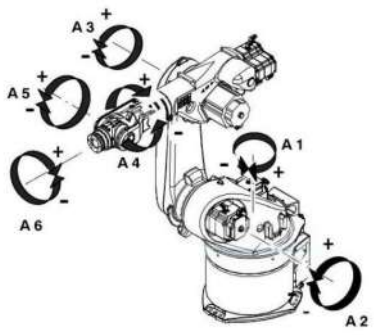


Figure 2.1 Kuka KR 30-3

1. Load capacity: KR 30-3 can hold up to around 30 kilograms (66 pounds) of weight. This means it has a weight capacity of 30 kilograms.
2. Measurements: Approximately 6.7 feet in height, the robot arm can reach a maximum height of 2033 millimeters. The arm's maximum reach and operational range are defined by these parameters.
3. Degrees of Freedom (DOF): The average KUKA KR 30-3 has 6 DOF. The robot arm's range

of motion is increased by the number of degrees of freedom (DOFs).

4. Weight: The robot's approximate weight is 1 466 grams. Without any payload or attachments, this is how much the robot arm weighs.

5. Controller: A KUKA robot controller is the device typically used to program and manage the functioning of a KUKA robot.

6. Application: Material handling, machining, welding, assembling, and other manufacturing activities that need exact and repeated typing frequently employ KR 30-3.

2.1.2 WLKATA Mirobot

Beijing Tsinew Technologies Co., Ltd. produces the WLKATA Mirobot, a six-axis compact industrial robot arm manipulator. Manipulator's Original Concept and Realization General characteristics and specifications of the WLKATA Mirobot, which was designed primarily for STEAM factories, K-12 schools, and universities:

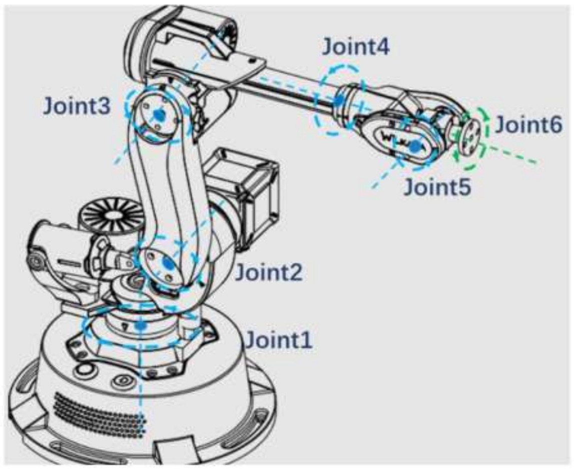


Figure 2.2 WLKATA Mirobot

1. Degrees of Freedom (DOF): Most Mirobot robots can move along four or six axes (called "degrees of freedom"). Model and revision choices can affect the precise DOF configuration.

2. Load carrying capacity: Mirobot can carry lightweight tools and things. While specific figures may vary, payload capacities are typically in the hundreds of grams.

3. Reach: The reach of desktop robot arms is restricted. It is often intended for use in tight spaces, with a maximum radius of two to three centimeters.

4. Materials: Mirobot is typically constructed from aluminum or other lightweight and sturdy materials. Its small footprint makes it perfect for placing on a desk.

5. Controller: The software can either come with its own control interface specific to the robot arm, or it can be compatible with a standard programming environment and language.**6. Application:** The primary audience for the Mirobot Guide is educational and recreational robot and programming novices. Simple manipulation and locating can be accomplished with the help of draw, select, and operations.

7. Programming: Mirobot may be programmed in a number of different ways, including graphical user interfaces and more traditional programming languages.

Name	**Parameter**
Number of Axes	6+1
Standard payload	250g
Maximum payload (movement near desktop)	400g
Workspace	315mm
Repeated positioning accuracy	0.2mm
Type of Six Joints	High accuracy stepping motor + reducer
Power supply voltage	100 V- 240 V, 50/60 HZ
Power input	12V / 4A DC
Power	50W Max
Working environment	-10°C~ 60°C

2.1.3 The Yasukawa Motoman L-3

The Yasukawa Motoman L-3 is an industrial robot arm produced by Yasukawa Electric Corporation, a well-known Japanese robotics manufacturer. The Yasukawa Motoman L-3 robot arm is a component of the company's Motoman range of industrial robots, and it has the following basic characteristics:

Load Capacity: Yasukawa Motoman L-3 has a maximum load capacity of 3 kilograms (about 6.6 pounds). This means it has the capability to transport and handle items up to 3 kilograms in weight.
Reach: The robot arm can extend to a maximum length of roughly 544 millimeters (about 21.4 inches). The arm's maximum reach and operational range are defined by these parameters.
Degrees of Freedom (DOF): The standard number of DOF for a Yaskawa Motoman L-3 is 5. The robotic arm's multiple degrees of freedom (DOFs) make it useful in a wide range of manufacturing settings.
Weight: The standard robot weight is around 286 pounds (130 kg). Without any payload or attachments, this is how much the robot arm weighs.
Controller: Advanced YRC1000 robot controllers are commonly found in Yasukawa robots. These controllers are used for programming and controlling the robot's movement and operation.
Application: The Yaskawa Motoman L-3 finds widespread use in a wide variety of assembly and processing lines, where the precision and ability to perform repetitive motions are invaluable.

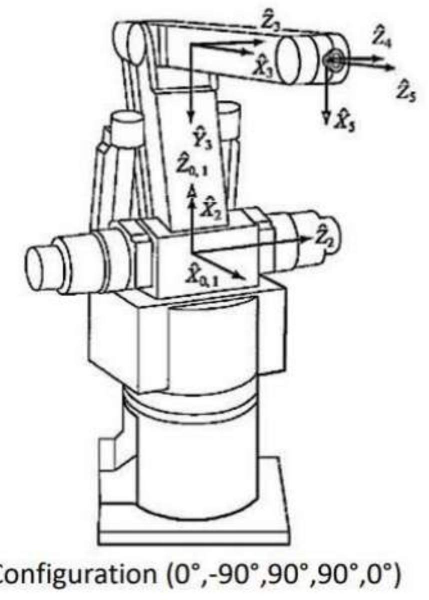


Figure 2.3 Yasukawa Motoman L-3

2.2 Related Projects

2.2.1 Modeling and Analysis of a 6 DOF Robotic Arm Manipulator

In this research [1], the authors create and examine a kinematic model for a six-DOF robotic arm. As this article demonstrates, complex manipulators present unique challenges that are best understood through the use of analytical models in robotics. The primary motivation for this study is the need for accurate robot control and positioning in unstructured environments, making the development of an analytical solution to the inverse kinematics problem of a robot arm a top priority. The proposed kinematic model enables the manipulator to move in the specified path. Denavit-Hartenberg (DH) parametric scheme is used to model the forward motion of the robot arm in the kinematic model. However, an inverse kinematic model was created to determine the necessary joint angle given the location and orientation of the robot's end effector. The researcher employed MATLAB's Robotics Toolbox to verify the accuracy of the complex kinematic model. They also experimented with a genuine robot arm after implementing an inverse kinematics model. The created model yields data that demonstrate the robotic arm's end effector can identify target coordinates to within 0.5 cm. This work is significant because it demonstrates how the same robotic manipulator can be utilized to address a variety of kinematics issues. The study adds to the enhancement of the control and performance of high-DOF robotic manipulators in a variety of practical applications by giving strong analytical solutions for inverse kinematics.

2.2.2 Development of a prototype 6 degree of freedom robot arm

This research paper [2] describes and evaluates a new 6-DOF robot arm design. The suggested robot is based on the design of the AR3 robot arm; however, it has been adapted to better fit the Vietnamese economy. To begin, the arm's construction was designed, strength tested, and optimized in Solidworks. Taking into account actual conditions in Vietnam, the researchers hope to lower production costs without sacrificing the robot's efficiency. In order to save money and run efficiently, we've optimized our material selection procedure. After the blueprint has been drawn up, researchers use a topology optimization technique to the machine parts to find the best layout for the robot arm. CNC (Computer Numerical Control) machines are used to create prototypes to test the design and optimization. The primary function of the robot arm was successfully tested in experimental settings. In conclusion, this research offers a practical design option for a 6 DOF robotic arm that accommodates Vietnam's unique needs and economic context. Using topology for optimization and Solidworks software for design, strength testing, and optimization increase the usefulness of this study. The proposed design will be more likely to succeed if it undergoes additional experimental validation, and it will serve as a helpful reference for any future work on developing and manufacturing a similar 6 DOF robot arm.

2.2.3 Design and Control of 6 DOF Robotic Manipulator

The goal of this research paper [3] is to create a robotic arm with six degrees of freedom (DOF). Similar multi-DOF robots are gaining traction in a variety of automation applications due to their superior precision when compared to human labor. The primary objective of this work is to develop a robot arm suitable for use in manufacturing, particularly for pick-and-place activities requiring delicate goods. The microcontroller-operated robotic arm consists of a base, shoulder, elbow, wrist rotation, and operational grip (end effector), and can perform a wide variety of tasks. The handle, which acts as an end effector, is built to securely grip a wide variety of items within the reach of a single hand. The robot arm's motors are supercharged with a PID controller for optimal performance. Precise control and placement of DC motors are made possible by advanced kinematics implemented by the microcontroller. The goal of this work is to keep the design straightforward while still allowing for straightforward selection and placement. This study aims to develop a 6 DOF robotic hand capable of performing manufacturing jobs requiring the careful handling of objects by combining mechanical systems, design concepts, and prototype implementation. Developing such robotic arms successfully can aid in the automation of processes and boost the productivity of pick-and-place tasks across a number of different sectors.

2.2.4 Kinematic Analysis of a 6-DOF Robotic Arm

The kinematic model of a six-DOF robotic arm is presented in this research [4]. Both forward and backward kinematic models were analyzed in this study, with the use of the Denavit-Hartenberg (D-H) parameters and the Partial Swarm Algorithm (PSO). The end-effector robot's location and movement are calculated using a feed-forward kinematic model. Allows one to determine the position and orientation of an end-effector given its joint angle of rotation. However, the inverse kinematics model is employed to determine the requisite joint angle for moving the end effector to the desired location and orientation. Inverse kinematics problems can be solved quickly and accurately with the help of the PSO algorithm. The simulation results were validated by experimental experiments. The kinematic model seems to accurately capture the robotic arm's behavior based on a comparison of the simulated data with the location and direction measured in the experiment. Finally, the findings of both forward and reverse kinematics are provided as part of the kinematic model for the 6 DOF robot arm. The research is significant because it establishes a dependable method for controlling and positioning a robot arm's end effector using D-H parameters and the PSO algorithm. Experimental verification of the suggested model's accuracy and efficiency is a significant advance in robotics and automation.

2.3 Related Studies/Research

2.3.1 A Dynamic Model Based Robot Arm Selection Criterion

How to choose a robot manipulator architecture, including link length, link direction, and link type (e.g. rotary, prismatic), is discussed in this article [1]. Conventionally, such choices are determined based on first-hand experience, touch, and kinematics, such as workspace and manipulation, whereas dynamics is typically overlooked despite its relevance in this context. The purpose of this research is to provide new criteria for selecting robot manipulator architecture based on system dynamics, with a special emphasis on the ease of solving the corresponding general inertia matrix (GIM). Fast computations and/or diagonals can increase the robot's speed, accuracy, and stability; GIM has a significant impact on control and simulation methods. The computational complexity of the GIM in terms of floating-point operations and the CPU time necessary for the evolving algorithms of the GIM, such as the inverse dynamics algorithm, are used to evaluate the GIM's ease of use. The proposed criteria are demonstrated by comparing the workspace and manipulation of two robot arms with rotary and prismatic joints. The advantages and benefits of using GIM-based metrics in robot manipulator design are highlighted through this comparison. Finally, the research provides instances of picking the RTX and Stanford spatial robot architectures by hand using the provided criteria, illustrating the usefulness and efficacy of the GIM-based approach in selecting robot manipulator architectures.

2.3.2 Current Designs of Robotic Arm Grippers

The research [2] investigates the potential and limitations of robotic hand grippers across a range of applications, focusing on the technological advancements that have made it possible for robots equipped with grippers to carry out previously human-only tasks. Because of their versatility and effectiveness in many applications, grippers are gaining prominence. Ideally, a gripper would be inexpensive to produce, efficient in its use of energy, and flexible in its application. Despite the plethora of grippers available, there are still obstacles that prevent them from replacing the hold of human hands. The purpose of this study is to compare and contrast several gripper designs for robot arms in order to identify their relative merits and shortcomings. Motion mechanisms, degrees of freedom, multi-object sensing capabilities, and individual applications are all 3 taken into account in this comparison. The research seeks to develop a gripper design that provides a number of benefits, including the ability to do a variety of activities, the elimination of some existing constraints, and so on. This in-depth analysis will help shape future developments in gripper design, leading to more sophisticated and efficient robot grippers that can be used in a wider variety of settings and industries.

2.4 Limitations and constraints in robotic manipulators

Creating a robotic arm manipulator capable of performing 53 tasks such as picking and placing and more is a complex engineering challenge. Although advances in the field of robotics have come a long way, there are some limitations and challenges in designing and using such robotic systems. Here are the general limitations:

1. Flexibility and Accuracy: Achieving a high degree of accuracy and precision in robotic arm movements can be difficult. Factors such as mechanical tolerances, sensor limitations, and environmental disturbances can affect the hand's ability to place objects accurately.

2. Load carrying capacity: There is a limit to how much weight a robot arm can carry and hold. Large and heavy objects may be beyond the capabilities of robotic arms, limiting their practical reach.

3. Reach and Workspace: Robots are limited by the physical design of the arms, which may limit their ability to reach objects in specific areas or limited areas.

4. Complexity of the Project: While robotic arms excel in repetitive and ready-made tasks, they may be more difficult to solve complex and unpredictable tasks. The programming and control system of the arm must be sophisticated enough to handle a wide range of scenarios.

5. Speed: In some applications, the speed of movement of the robotic arm may not be sufficient to meet production or operational requirements.

6. Sensitivity and Sensing: Robot arms rely on sensors and sensory systems to detect and interact with objects. Limitations in the accuracy and sensitivity of sensors can lead to errors in object detection or manipulation.

7. Safety: The interaction of the robotic arm with other people or objects can lead to safety risks. It is important to ensure the safety of robot arm movements and avoid collisions.

8. Cost: Building highly capable robotic arms can be expensive, limiting their availability for certain industries or applications.

9. Maintenance and Reliability: Complex robotic systems require regular maintenance to ensure proper operation. Unplanned downtime due to outages or technical issues can be costly and disruptive.

10. Cognitive Limitations: Robot arms without advanced AI capabilities may struggle with decision making and problem solving outside of pre-programmed tasks.

11. Environmental Limitations: Extreme environmental conditions such as high temperature, humidity or radiation can affect the performance and lifespan of the robot arm.

2.5 Summary

As a rapidly developing discipline of computer engineering, robotics is at the forefront of annual innovation thanks to the work of countless engineers. The initiative aims to use multiple types of research to eliminate or significantly lessen these restrictions. The ultimate aim is to produce a finished product that performs the designated task flawlessly. This project's success is built on the shoulders of the ventures that came before it.

Chapter 3

PROJECT DESIGN AND IMPLEMENTATION

3.1 Project Design:

The project has been split into two halves, each of which focuses on a different aspect of problem solving or robotics. In the first step, a focused region of the office is identified and evaluated. Moreover, the manipulator is modeled mathematically by analytical and software means in MATLAB. The mathematical modeling approach allows for the derivation and calculation of crucial design parameters. The second stage of the project involves moving into 3D modeling using the determined design parameters. Each part of the manipulator is built in SolidWorks according to a set of specifications. Assembling these components results in a sophisticated 3D model of the manipulator. With this two-step method, we can build the manipulator methodically, from the ground up, from the theoretical to the practical.

3.2 Analysis procedure:

After learning the fundamentals of robotics, you may model the problem and zero in on the best course of action by performing a forward kinematics analysis of the manipulator to determine the necessary speeds, torques, and forces.

3.3 Methodology

Following methodology has been followed for project design.

1. Literature Review
2. Mathematical modelling.
3. Calculating design Parameters.
4. Motors and power transmission mechanism selection
5. 3D modelling and printing
6. Assembling
7. Implementing Graphical user interface
8. System Testing and finishing

3.4 Detail about Proposed solution

The links that make up our (R-type) manipulator are connected using only revolute joints. Our end effector (Gripper) is a 4-bar link type, and it has force sensors on both sides. All of the links connecting the gripper to the rest of the robot are 3D printed.

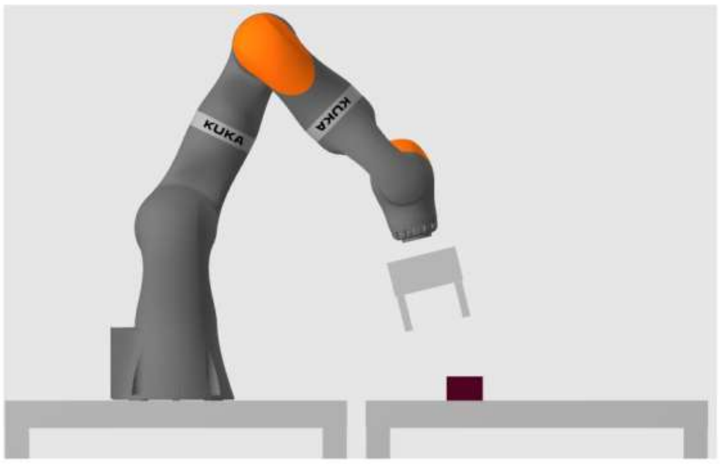


Figure 3.1 Pick and Place mechanism

3.5 Mathematical Modelling

Physical systems are represented by mathematical equations in the process of mathematical modeling of manipulators like robotic arms. This is useful for a variety of modeling, understanding, motion simulation, and control algorithm development purposes including manipulator behavior. There are typically two major phases to the modeling process: kinematic modeling and dynamic modeling.

1. Kinematic modeling:

Without taking into account force and torque, kinematic modeling focuses on the geometric relationship between the manipulator's numerous linkages and joints. The end effector's (the tool tip's) position and orientation relative to the robot's home coordinate frame is the focus of this method. When modeling kinematics, you can use one of two primary routes:

a. Forward kinematics

b. Inverse kinematics

3.5.1 Robotic Workspace

The term "workspace" is used to describe the set of points that can be reached by the end effectors of a robot manipulator. It's a simplified representation of the three-dimensional environment in which robots perform their tasks. If the end-effector is moved beyond the working space, there is no kinematic solution to move the manipulator or the joint space to accommodate the end-effector.

3.5.2 DH-Parameters

Modeling and evaluating the kinematics of robot manipulator arms is a common practice in robotics, and the Denavit-Hartenberg (DH) parameterization is a popular tool for doing so. It provides a logical method for representing the shape and location of a robot arm's many moving parts. Robot manufacturers have universally used the DH parameter first proposed by Denak Denavit and Richard Hartenberg in the 1950s. Each robot arm joint's coordinate system and the distance between joints are set by the DH parameter. These values are required for determining the robot's position, orientation, and motion in three-dimensional space using forward kinematics, inverse kinematics, and the Jacobian matrix. There are four DH parameters connected with each robot arm joint:

- **Link Length (d):** It is the distance between the z-axes of adjacent coordinate frames along the common normal, measured along the joint axis.
- **Link Offset (a):** It is the distance between the x-axes of adjacent coordinate frames along the common normal, measured perpendicular to the joint axis.
- **Joint Angle (θ):** It is the rotation angle about the joint axis, measured from the old z-axis to the new z-axis, along the joint axis.
- **Joint Twist (α):** It is the rotation angle about the common normal, measured from the old x-axis to the new x-axis, along the common normal.

A convention is defined for allocating the coordinate frame to the joints in order to implement the DH parameter. The right coordinate system consists of a z-axis that coincides with the joint axis, an x-axis that extends from the previous axis to the current joint, and a y-axis that is determined by the right hand rule. In order to calculate the position and orientation of the end effector, the DH parameters for each joint in the robot arm must be determined, and the transformations from the base to the end effector must be chained. Forward kinematics describes this action. Inverse kinematics, on the other hand, entails finding the joint angle necessary to orient the end effector at a specified position and orientation in space. The DH parameters provide a standardized foundation for the mathematical modeling of robotic manipulators, making kinematic analysis easier. Robot software, modeling, and control algorithms all make heavy use of this technique to accurately represent and analyze the motion of robot arms. It's worth noting, nevertheless, that alternate kinematic models or representations may be more suited for certain sophisticated robot designs.

3.1 DH-Table for CENTRON

No. of Links	Link Offset (a)	Joint Twist (α)	Link Length (d)	Joint Angle (θ)
Link 1	0	0	0	Θ_1
Link 2	90	0	0	Θ_2
Link 3	0	191	0	Θ_3
Link 4	90	172	0	Θ_4
Link 5	90	0	0	Θ_5

The above table is further use to compute the forward kinematics of our manipulator.

3.5.3 Forward Kinematics

With inputs like actuator rotation angle, forward kinematics may calculate the end effector's position and orientation. Here, the user sets the angle at which each joint motor rotates, and the robot then moves to the position indicated by those coordinates in the configuration file. The advanced kinematics joint angle is found in the Denavit-Hartenberg (DH) table, which details the kinematic parameters of the robot manipulator's joints and linkages. The forward kinematic algorithm determines the end effector's location and orientation with respect to the robot's base frame based on the DH table and the specified joint angles. This data is crucial for figuring out the robot's spatial layout and its capabilities in the workplace.

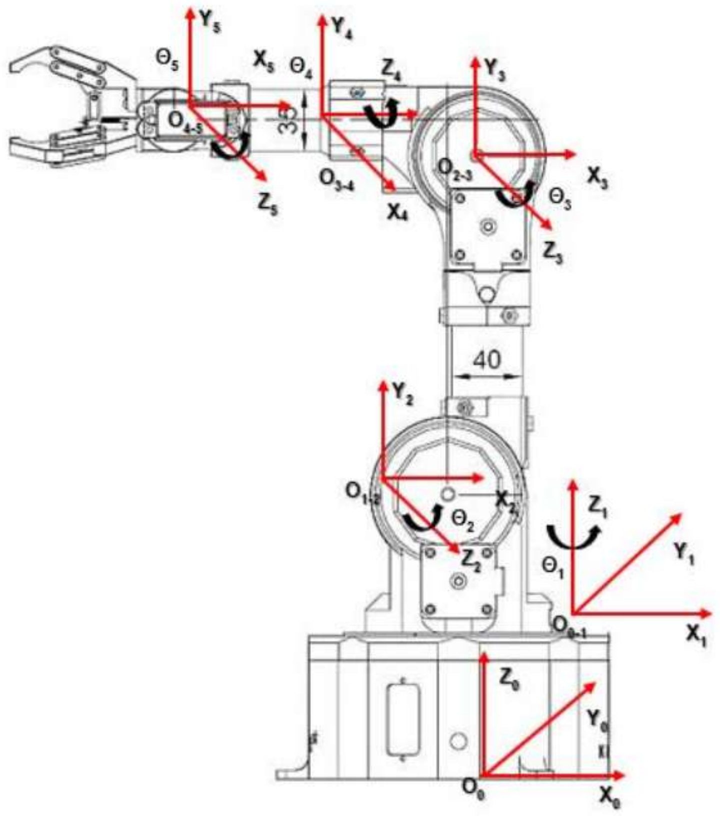


Figure 3.2 Assigning Reference Frames

From the above DH-Table we computed the following transformation matrix

$$ {}^{i-1}\mathbf{T}_{i} = \begin{pmatrix} C\theta_i & -S\theta_i C\alpha_i & S\theta_i C\alpha_i & a_i C\theta_i \\ S\theta_i & C\theta_i C\alpha_i & -C\theta_i S\alpha_i & a_i S\theta_i \\ 0 & S\alpha_i & C\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{pmatrix} $$

Here, Sθi=sin θi, Cθi =cos θi, Sαi=sin αi, Cαi=cos αi, Sijk=sin(θi+ θj+ θk), Cijk=sin(θi+ θj+ θk)

$$ {}^{0}\mathbf{T}_{1} = \begin{pmatrix} C_1 & -S_1 & 0 & 0 \\ S_1 & C_1 & 0 & 0 \\ 0 & 0 & 1 & d_1 \\ 0 & 0 & 0 & 1 \end{pmatrix} $$

$$ {}^{1}T_{2} = \begin{bmatrix} C_2 & 0 & S_2 & 0 \\ S_2 & 0 & -C_2 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

$$ {}^{2}T_{3} = \begin{bmatrix} C_3 & -S_3 & 0 & a_3{*}C_3 \\ S_3 & C_3 & 0 & a_3{*}S_3 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

$$ {}^{3}T_{4} = \begin{bmatrix} C_4 & -S_4 & 0 & a_4{*}C_4 \\ S_4 & C_4 & 0 & a_4{*}S_4 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

$$ {}^{4}T_{5} = \begin{bmatrix} C_5 & 0 & S_5 & 0 \\ S_5 & 0 & C_5 & 0 \\ 0 & 1 & 0 & d_5 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

$$ {}^{0}T_{5} = \begin{bmatrix} C_{12}\,C_{345} & S_{12} & C_{12}S_{345} & S_{12}d_5 + C_{12}a_4\,C_{34} + C_{12}a_3\,C_3 \\ S_{12}\,C_{345} & -C_{12} & S_{12}\,S_{345} & -C_{12}d_5 + S_{12}a_4\,C_{34} + S_{12}a_3\,C_3 \\ S_{345} & 0 & -C_{345} & a_4S_{34} + a_3S + d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

End effector Transformation Matrix

$$ T_{e} = \begin{bmatrix} n_x & O_x & \sigma_x & P_x \\ n_y & O_y & \sigma_y & P_y \\ n_z & O_z & \sigma_z & P_z \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

10

Where, $n_x = c12*c345;$

$n_y = s12*c345;$

$n_z = s345;$

$O_x = s12;$

$O_y = -c12;$

$O_z = 0;$

$\sigma_x = c12*s345;$

$\sigma_y = s12*s345;$

$\sigma_z = -c345;$ (10)

$P_x = s12*d5 + c12*a4*c34 + c12*a3*c3$

$P_y = -c12*d5 + s12*a4*c34 + s12*a3*c3$

$P_z = a4*s34 + a3*s3 + d1$

$$^{i-1}\mathbf{T}_i = \begin{pmatrix} C\theta_i & -S\theta_i C\alpha_i & S\theta_i C\alpha_i & a_i C\theta_i \\ S\theta_i & C\theta_i C\alpha_i & -C\theta_i S\alpha_i & a_i S\theta_i \\ 0 & S\alpha_i & C\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

43

By using the above equation we computed the forward kinematics for the given angles ($\theta_1 = 0$, $\theta_2 = 45$, $\theta_3 = 90$, $\theta_4 = 90$, $\theta_5 = 45$)

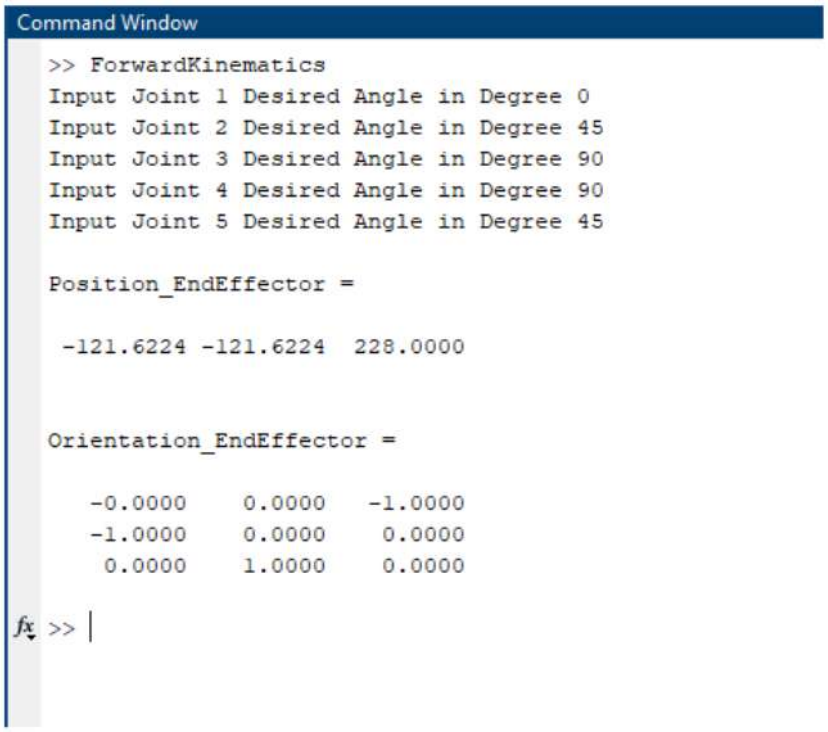
$$^{0}\mathbf{T}_5 = \begin{pmatrix} 0 & 0 & -1 & -121.622 \\ -1 & 0 & 0 & -121.622 \\ 0 & 1 & 0 & 191 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

37

In the above matrix the last column shows the position of the end effector and rest 3x3 matrix shows the orientation of the end effector.

3.5.4 Matlab Testing

For the given DH parameters we verify our testing through matlab


Command Window

```
>> ForwardKinematics
Input Joint 1 Desired Angle in Degree 0
Input Joint 2 Desired Angle in Degree 45
Input Joint 3 Desired Angle in Degree 90
Input Joint 4 Desired Angle in Degree 90
Input Joint 5 Desired Angle in Degree 45

Position_EndEffector =

 -121.6224 -121.6224  228.0000


Orientation_EndEffector =

   -0.0000    0.0000   -1.0000
   -1.0000    0.0000    0.0000
    0.0000    1.0000    0.0000

>>
```

Figure 3.3 Matlab Code Results

3.5.5 Trajectory Planning

30

Robot trajectory planning entails determining the most efficient and streamlined way for the end effector of a robotic arm to travel from its current position to a predetermined destination. Since different joint configurations call for different end effector positions and orientations, advanced kinematics plays a crucial part in this process. An overview of how to plan trajectories with sophisticated kinematics is as follows:

Define the Problem: Determine the problem's parameters, including the end effector's starting and stopping points and the direction it will travel in between. Defines checkpoints along the route that the robot must go.

Define Trajectory: Cut the entire path up into smaller pieces, or points. To achieve the required trajectory smoothness and execution speed, the number of waypoints and the time interval between them can be modified.

Forward Kinematics: Use the robot's forward kinematic equations to calculate the position and direction of the end effector for each point. For robots with multiple degrees of freedom, forward kinematic equations combine positions for the final effect.

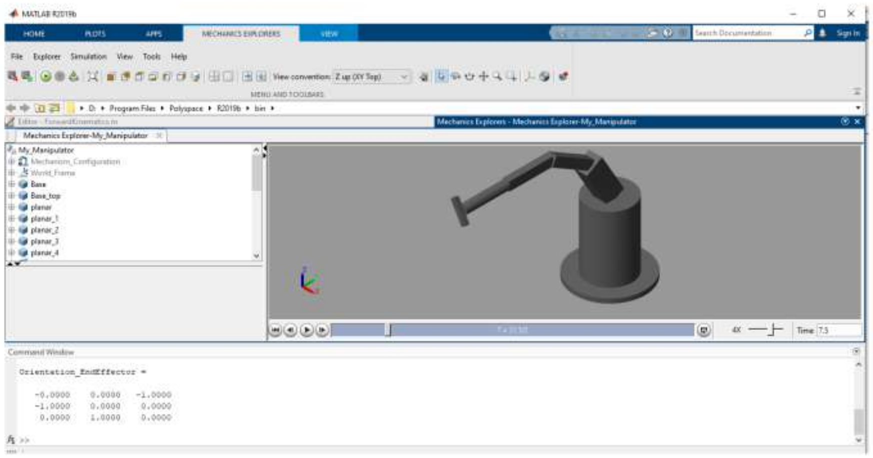
Interpolation: Path smoothing by interpolation between neighboring sites. Position, velocity, and acceleration profiles can be obtained in a continuous fashion using any of several interpolation techniques, including linear interpolation, cubic fraction, and quantic fraction.

Velocity and acceleration planning: Once the path is determined, a velocity and acceleration profile must be generated to ensure that the robot moves at a controlled speed without exceeding kinematic limits or causing dangerous movements.

Generation of joint trajectories: Inverse kinematics can be used to convert desired end effector trajectories into suitable joint trajectories. By solving inverse kinematics, joint angles or joint displacements are calculated for each point.

Smoothness and Avoidance of Obstacles: The trajectory can be smoothed out and obstacles avoided with some additional planning. Safe robot execution is guaranteed by taking constraints into account, such as joint restrictions and speed limits.

Simulation and Validation: Collisions, joint limit violations, and overall performance can all be checked by simulating the resulting trajectories in a robot simulation environment.

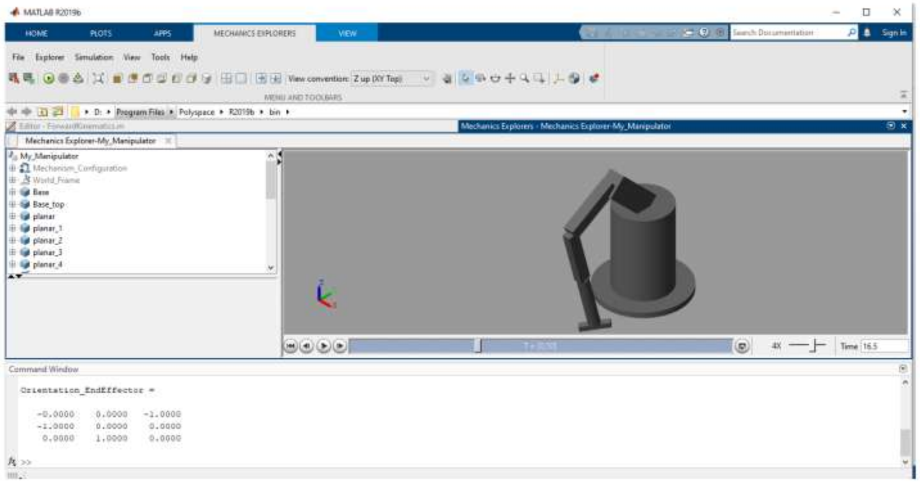


```
Orientation_EndEffector =

   -0.0000    0.0000   -1.0000
   -1.0000    0.0000    0.0000
    0.0000    1.0000    0.0000

>>
```

Figure 3.4 Manipulator Model (a)

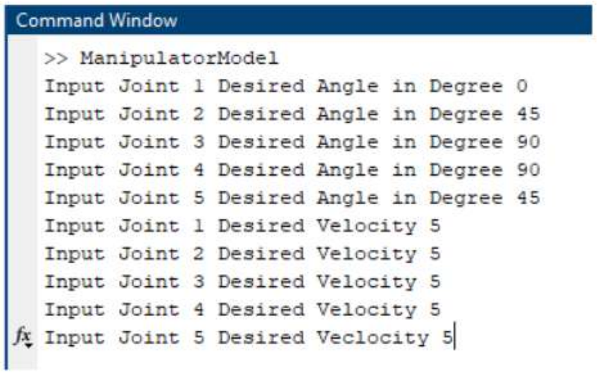


```
Orientation_EndEffector =

   -0.0000    0.0000   -1.0000
   -1.0000    0.0000    0.0000
    0.0000    1.0000    0.0000

>>
```

Figure 3.4.1 Manipulator Model (b)



```
Command Window
>> ManipulatorModel
Input Joint 1 Desired Angle in Degree 0
Input Joint 2 Desired Angle in Degree 45
Input Joint 3 Desired Angle in Degree 90
Input Joint 4 Desired Angle in Degree 90
Input Joint 5 Desired Angle in Degree 45
Input Joint 1 Desired Velocity 5
Input Joint 2 Desired Velocity 5
Input Joint 3 Desired Velocity 5
Input Joint 4 Desired Velocity 5
Input Joint 5 Desired Veclocity 5
```

Figure 3.4.2 Manipulator Model (c)

The above *figure 3.4* and *figure 3.4.1* and *3.4.2* show the manipulator model implememnted in the matlab for the given model we give the desired angles and velocities and it responses back by approacing towrads the target using **Forward Kinematics**

3.5.5 GUI (Grpahical User Interface)

We used USB serial communication at a baudrate of 9600 between the processing program and the arduino board, where our graphical user interface was built. We need to describe several parts and operations before we can build a graphical user interface. The specifics are as follows:

1. Software environment:

- Graphical user interface (GUI): Software is used for processing.
- Communication between devices and the Arduino Mega board is a constant feature. A component of the graphical user interface

2. Part of the GUI interface:

- There are a total of 13 buttons on the user interface.

3. Keypad to label robot arm joints:

- Primary link: This link has two buttons dedicated to it.
- Two buttons are indicated for the shoulder area.
- There are two buttons for the elbow.
- There are two buttons for the wrist.

4. References for Joints:

- Joints Literature Cited There are two buttons for controlling each joint.
- The clockwise arrow button does what you'd expect it to do.
- Clockwise direction: press the second button to move the joint in the opposite direction.

5. Gripper control:

- The Gripper features two buttons for control (buttons 5 and 6).
- Claw opening is as simple as pressing a button.
- The claw can be closed using the additional "Close Claw" button.

6. Trajectory control:

- Trajectory 1 has a button for autonomous trajectory control with sophisticated kinematics.

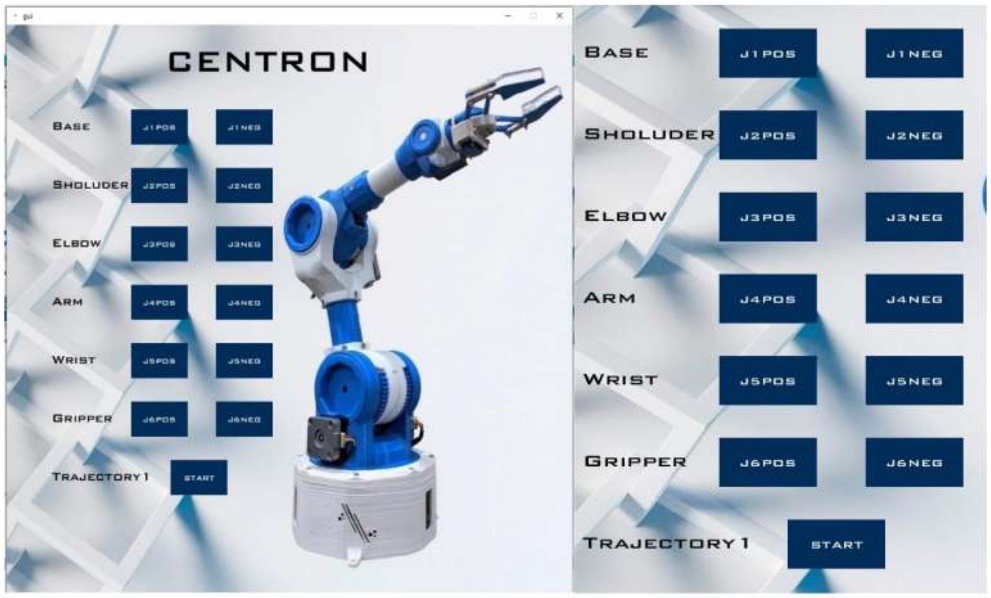


Figure 3.5 GUI Processing

3.6 Detail of Working Design Model

After completing the mathematical modelling, required parts are selected according to their calculated design parameters and availability in market. 3D model has been developed from the scratch by using Solid works. Each part has been made individually and then assembled in Solid work. Detail of all designed parts are following

3.6.1 Robotic Arm Link Lengths

The table 3.4 shows the link length measurements of our robotic arm. Through these measurements our manipulator's joint are linked together to form a refined structure.

Table 3.2 Link Lengths of CENTRON

No of Links	Measurements in (mm)
Base to Link 1	81 mm

Link 1 to Link 2	80 mm
Link 2 to Link 3	191 mm
Link 3 to Link 4	172 mm
Link 4 to End Effector	92 mm

3.6.1 Complete Assembly

Figure 3.6 shows the complete assembly of our robotic arm manipulator after assembling the 3D parts of individually.

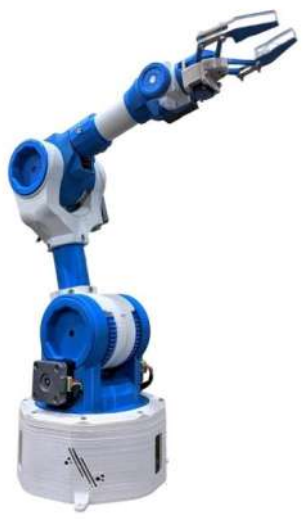


Figure 3.6 Complete Assembly

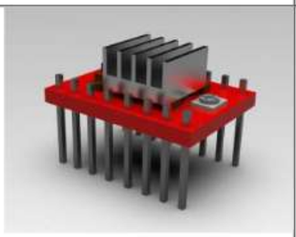
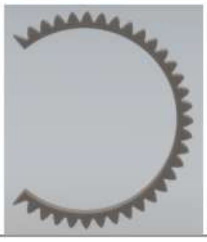
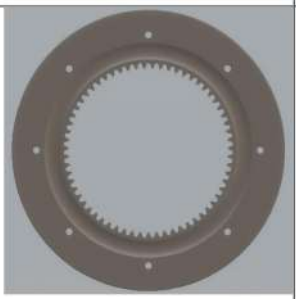
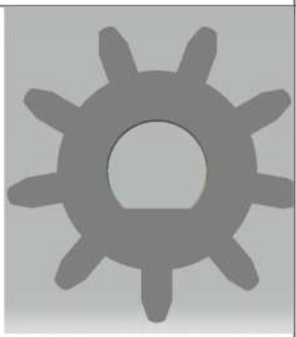
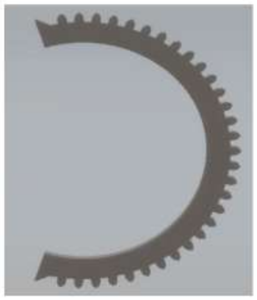
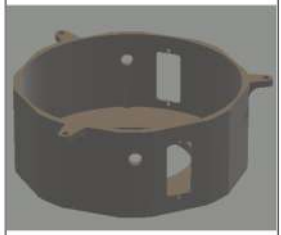
3.6.2 Complete Part detail and description

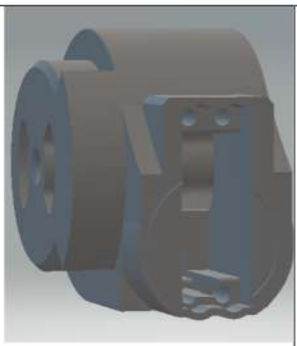
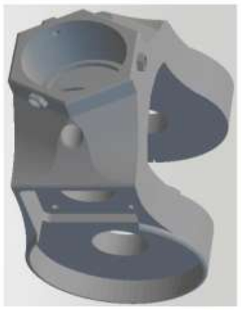
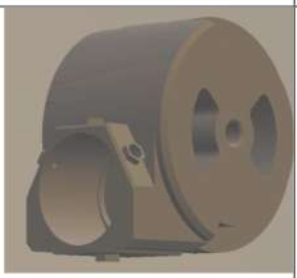
Table 3.3 and 3.4 shows the comprehensive details of the components that are involved in our robotic arm manipulator in addition to that the selection of gears through gear mechanism is also given below.

Table 3.3 Parts Detail

ITEM NO.	PART Name	QTY.	ITEM NO.	PART Name	QTY.
1	Base	1	**28**	Upper claw cover	2
2	Base sub part a	1	**29**	Gripper left claw gear	1
3	Base sub part b	1	**30**	Gripper right claw gear	1
4	Base upper disk	1	**31**	Gripper inner holding slit	2
5	Base upper ring part	1	**32**	Gripper back holder	1
6	Link 5 upper part	1	**33**	3D Printed Rod Ø8 x 80	1
7	Link 4 upper part	1	**34**	3D Printed Round tube Ø35 x 90	1
8	Link 3 upper part	1	**35**	3D Printed tube Ø40 x 110	1
9	Link 2 upper part	1	**36**	3D Printed rod Ø8 x 15	1
10	Link 1 upper part	1	**37**	M3 nut	20
11	Ball Bearing 8mm	20	**38**	M3 x 12 SHCS screw	20
12	Link-2 Right side cover	1	**39**	M3 x 20 SHCS screw	20
13	Link-2 Left side cover	1	**40**	M3 x 12 SHCS screw	20
14	Link-2 inner cover	2	**41**	3D printed Rod Ø3 mm	1
15	Link-4 left inner cover	1	**42**	Screw M3 x 10	10
16	Link-4 right inner cover	1	**43**	Screw M3 x 25	5
17	Link-4 left upper cover	1	**44**	Nut M3	8
18	Link-4 right upper cover	1	**45**	Nema 17	4
19	Link-2 upper gear	2	**46**	Mg996-R Servo	2
20	Link-4 upper gear	1	**47**	Sg90- Servo	1
21	Link-4 slit cover	1	**48**	Arduino Mega	1
22	Gripper Lower base	1	**49**	CNC Shield v3	1
23	Gripper left upper part	2	**50**	A4988 (Stepper Driver)	4
24	Gripper right upper part	2	**51**	12-V battery	1
25	Gripper upper cap	1	**52**	6-V battery	1
26	Sg90 holder	1	**53**	Stepper Motor gear	4
27	Bearings	3	**54**	DC to DC Buck convertor	1

Table 3.4 Parts List

Description	Parts	Description	Parts
Stepper Motor Driver A4988.		**Link-4 Upper gear** No of Teeth = 31	
Upper Disk Gear: No of teeth = 63		**3D Printed rod Ø8 x 15** Thickness: 8mm.Length: 15mm	
Nema 17 gear: No of Teeth = 9.		**3D Printed rod Ø8 x 80** Thickness: 8mm. Length: 80mm	
Link-2 upper gear: No of Teeth: 31		**Base** Length: 81mm	

Link 4 Upper Part Diameter = 48mm		**Link 3 Upper Part** Diameter = 30mm	
Link 2 upper part Inner diameter = 40mm		**Link 2 lower base part**	
Gripper right claw No of teeth = 26		**Gripper inner claw holder** No of teeth = 11	
Ball Bearing 8mm		**NEMA 17:** Holding Torque: 5.5kg/cm. Input Voltage: 12 V. Weight: 0.35 kg.	
Sg90 Servo **4.8V**		**Mg996-R Servo** **6V**	

CNC Shield		**Arduino Mega**	

3.6.3 System Electronics

Stepper Motors:

The robot arm in this project is moved with the help of a stepper motor. If you need accurate positioning, reliable speed regulation, and consistent motion, a stepper motor is the way to go. The lack of contact brushes also makes it incredibly durable and dependable, as they cause very little wear. Because it is an open-loop system, the stepper motor can respond quickly and repeat movements with high precision. Nema-17 stepper motors were employed because of their universality in robotics and automation and their standard dimensions.

Figure 3.7 Stepper Motor

Working:

A stepper motor has a stator, which remains in one place, and a rotor, which rotates. Permanent magnets and variable reluctance iron cores are also viable options for the rotor. The rotor is housed within the stator and is supplied with teeth wound from the coils. Creating a magnetic field with electricity flowing through the coils activates the stator phase. At first, during motor generation, the rotor is oriented so that it faces the magnetic field produced by a single coil—let's call it coil A. Coils B and C are then activated in a certain sequence to further enhance the motion. Activating coil B causes the rotor to rotate counterclockwise by 60 degrees to align with the new magnetic field. When coil C is switched on, the rotor rotates once again to face the coil's magnetic field. This process continues, and the final rotation of the rotor can be precisely controlled by activating the stator coils in a predetermined order and at a predetermined time. The color of the stator teeth in the figure indicates the direction of the magnetic field formed by the stator winding, which is useful for understanding the stepper motor's operating principle.

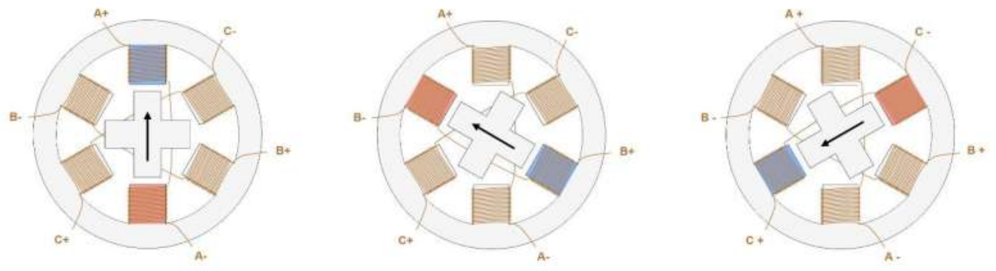


Figure 3.7.1 Operating Mechanism

SG90-Servo Motor

In the realms of robotics and hobbyists, the SG90 servo motor is a staple. The ability to precisely adjust the motor's angle, in a small, low-cost package. Robotics, RC vehicles, model airplanes, camera gimbals, and other applications that demand precise positioning all make use of servo motors.

Figure 3.7.2 Sg90 Servo Motor

Working principle:

The SG90 servo motor works on the principle of using a closed loop control system. It consists of several key components that work together to achieve accurate positioning:

DC Motor: The SG90 servo motor itself is a DC motor responsible for generating mechanical power.

Gear Train: The motor output shaft is connected to a gear set that reduces high-speed, low-torque rotation of the motor to low speed, high torque output.

Potentiometer (Position Feedback Device): The servo has a small potentiometer (variable resistor) attached to the output shaft. This component provides positional feedback to the control circuit that allows the control controller to know the current state of motor motion.

Control circuit: The servo motor includes a control circuit responsible for comparing the desired position (input signal) of the potentiometer with the actual position (feedback signal). The control circuit generates an error signal based on the difference between these two positions.

Positioning mechanism: Based on the error signal, the control circuit drives the motor in the appropriate direction to reduce the error and bring the shaft to the required position. This process continues until the desired destination is reached.

Working:

The SG90 servo motor has a built-in control circuit that reads input from the user and rotates the motor to the specified angle. The typical form of the control signal is a 50 Hz pulse width modulation (PWM) signal. In this signal, the target state is set by the duration of the pulse.

For example:

- A pulse width of 1.5 ms can correspond to the middle position (90 degrees).
- A pulse width of 1 ms can correspond to the lowest angle (0 degrees).
- A pulse width of 2 ms can correspond to a maximum angle (180 degrees).

Servo constantly modifies its position in response to command signals, allowing it to hold its position with high precision. The SG90 servo motor may not be up to the task of handling heavy loads due to its low torque and slow speed. However, the SG90 is a viable option for small-scale projects that need fine control because to its inexpensive price and widespread availability.

MG996-R Servo Motor:

Our manipulator has an additional degree of freedom thanks to two Mg996-R servo motors we installed in our robotic arm. The MG996-R servo motor is a common choice for applications that call for precise control of angular motion, such as robotics, remote-controlled vehicles, and radio-controlled aircraft. The MG996 servo motor has been replaced by this newer, more advanced model.

Figure 3.7.3 MG996-R Servo Motor

Working principle:

The MG996-R servo motor operates in the same way as the SG90 servo motor, using a closed loop control system. It also consists of components such as:

DC Motor: The MG996-R servo motor itself is a DC motor responsible for generating mechanical power.

Gear Train: Like the SG90, the MG996-R has a gear train that converts the engine's high-speed, low-torque rotation into low-speed, high-torque output.

Potentiometer (Positional Feedback Device): Inside the servo, there is a potentiometer attached to the output shaft that provides positional feedback to the control circuit.

Control circuit: the servo motor includes a control circuit that compares the desired position (input signal) and the actual position (feedback signal) from the potentiometer. The control circuit generates an error signal based on the state difference.

Positioning mechanism: The control circuit drives the motor in the right direction to reduce errors and align the shaft to the required position.

Working Process:

Similar to the SG90, the MG996-R servo motor is also compatible with a PWM control signal. In a servo motor, the desired position of the shaft is set by the control signal. It constantly receives, deciphers, and responds to servo control signals, adjusting its position accordingly. The servo motor attempts to attain and hold the intended position with the highest precision possible based on the width of the pulse in the control signal. The enhanced MG996-R version provides more power and stability than the previous SG90 model, making it ideal for demanding applications. Keep in mind that the MG996-R has improved performance over the SG90, but it is still not without its drawbacks. - In high-traffic situations, delays or position mistakes could occur. You should think about upgrading to a higher-end, more precise servo motor for mission-critical tasks. To prevent motor or train gear damage, it is also important to use the correct power supply and not subject the servo to servo power in excess of its rated capability.

A4988 (Stepper Motor Driver):

Four A4988 Stepper motor drivers are mounted on the CNC shield and are used in our circuitry. The A4988 motor driver IC (integrated circuit) is commonly used to operate a bipolar stepper motor. Popular applications include controlling stepper motors for 3D printers, CNC machines, robotics, and more.

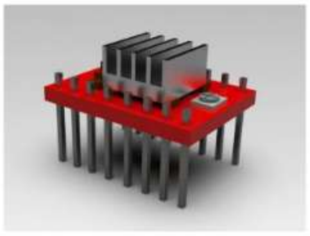


Figure 3.7.4 A4988 Stepper Motor Driver

Working principle:

A stepper motor is a type of motor that rotates at a constant rate but in discrete increments. The A4988 driver facilitates command over the stepper motor's direction and velocity. The A4988 driver operates by modulating the voltage applied to the stepper motor's coils. It regulates the motor's rotational speed and direction via a process called pulse width modulation (PWM).

Motor Connection: The A4988 driver has multiple terminals to connect four coils (for bipolar motors) or two coils (for unipolar motors) of a stepper motor. The coil pairs are labeled A, A', B and B'.

Current regulation: A4988 includes an internal current regulator that allows you to determine the maximum current flowing in the motor coils. This current determines the torque of the motor. The current is usually adjusted by connecting a potentiometer (current-adjusting trimpot) or an external resistor on the driver board.

Stepper motor and direction control: To control the stepper motor, the A4988 driver needs two control signals:

Step signal: A high-low transition of this pin causes the motor to take one step. The frequency of this pulse determines the rotation speed of the motor.

Direction signal: This pin controls the direction of motor rotation. Setting it up or down determines the direction of movement.

Microstepping: The A4988 driver supports microstepping, allowing smoother and more precise movement compared to full stepping. By sending intermediate step positions between full steps, microstepping can significantly reduce vibration and noise in the motor.

Working Process:

Setting the DIRECTION pin in the desired direction (high or low) and providing a high-to-low transition on the STEP pin will get the motor going. When the STEP pin changes, the motor advances by one step. It is possible to regulate the motor's rotational speed by adjusting the pulse's frequency. Setting the Direction pin high and applying a tiny pulse to the STEP pin will cause the motor to rotate in a clockwise direction. Setting the Direction pin low and giving a pulse on the STEP pin will cause the motor to spin counter-clockwise. The A4988 driver's microstepping solution is set by connecting Pins MS1, MS2, and MS3 to a high or low logic state.

CNC Shield:

The Arduino CNC Shield V3 is an extension board made specifically for operating CNC (Computer Numerical Control) equipment. Hobbyists and makers who want to construct their own CNC routers, engraving machines, or other CNC-based devices utilize it frequently.

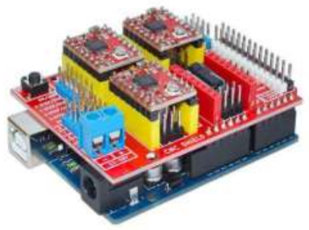


Figure 3.7.5 CNC Shield

Features and components:

Stepper Motor Driver: CNC Shield V3 has four slots for installing stepper motors, usually A4988 or DRV8825. This driver is responsible for controlling the stepper motors that move the axes (X, Y, Z and optionally A) of the CNC machine.

Microstepping Support: The A4988 and DRV8825 stepper motors used with the CNC Shield V3 support microstepping, which allows CNC machining to be smoother and more accurate.

Input Connector: the shield has terminals to connect limit switches and circuit breakers. A limit switch is used to limit the travel of the device and prevent over travel. The home key helps create a reference position for the device.

Laser Motion and Control: CNC Shield V3 provides several switches to control the motor (for cutting and milling) and the laser module (for engraving).

Power Supply: CNC Shield V3 requires an external power supply to power the stepper motor and other components.

Working:

The CNC shield was added to our project primarily to ensure that the second link of the robot arm always moved in unison with the two-step motor. The CNC shield allows us to precisely coordinate the motion of these parts, resulting in the robotic arm's smooth and coordinated operation. The tool can be optimized for size and efficiency with the inclusion of the CNC shield, leading to greater robot system performance and functionality.

3.6.4 Circuit Diagram

Complete circuit diagram for our manipulator is given below

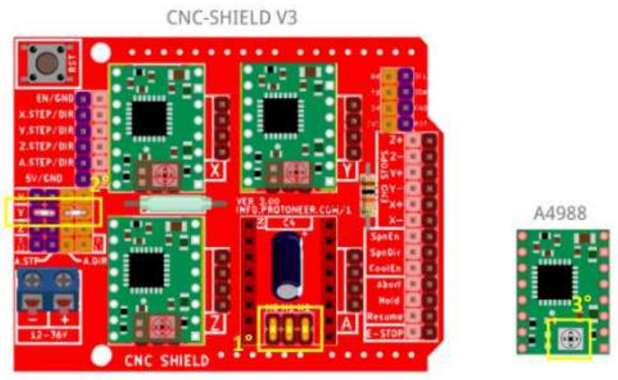


Figure 3.8 CNC Shield connection with A4988Stepper Motor Drivers

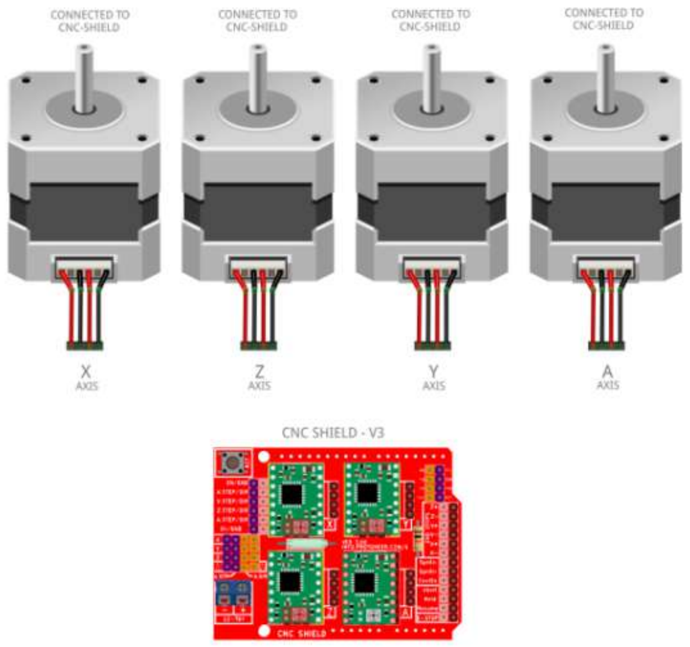


Figure 3.8.1 CNC Shield connection with Nema 17 Stepper Motor

A pre-wired connection connecting the four-step motor to the CNC shield's four specialized connectors is required. To ensure appropriate operation, jump the four pins depicted in the image such that axis A mirrors axis Y in its motion. The robot's A- and Y-axes must be perpendicular to the ground at all times. To ensure that the A and Y axes rotate in the other direction, it is necessary to connect them in the reverse orientation on the CNC board. The robot system will operate more smoothly and efficiently if the A and Y axes are rotated in the opposite direction, as shown in *Figure 3.8.1*. Note that in order to prevent damage to the A4988 driver, the stepper motor must be run on a precise 12V power supply with 1A current.

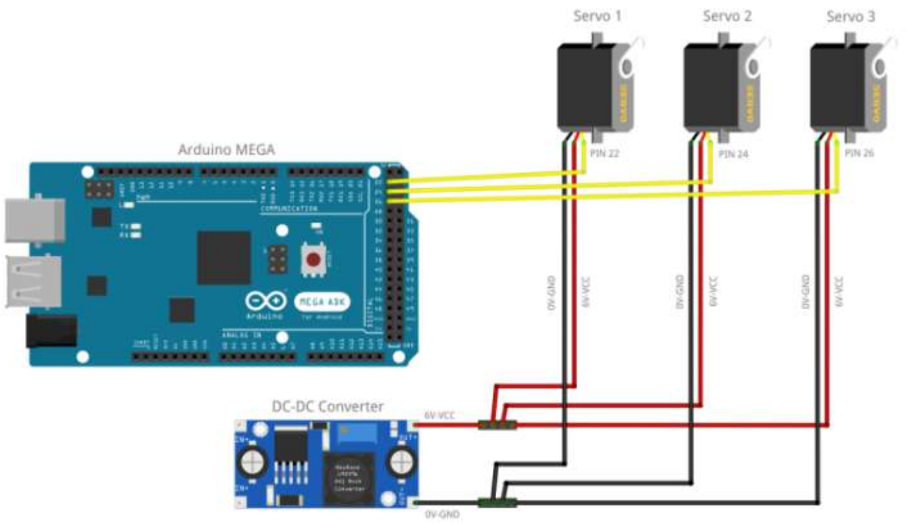


Figure 3.8.2 Arduino connection with Servo Motors

The three servomotors in the system should be connected to the splitters specified in the *figure 3.8.2* dedicated to the power supply (6V power supply side and 0.50A current). The signal cables from the servomotors must be connected to the Arduino Mega board. Here buck convertor is used to convert the 12V supply to 6V for the servos to behave ideally.

Chapter 4

TOOLS AND TECHNIQUES

4.1 Shape Optimization of Links:

For better 3D printing results, each component of the manipulator undergoes topology and image optimization processes. We anticipate a decrease in both publishing time and link weight by employing this optimization strategy. As a result, the connection is both lighter and stronger than before, allowing it to bear the specified weight and force.

Our MakerBot 3D-printed manipulator allowed us to accomplish this. Adjusting the 3D printed part's fill density, typically between 15% and 20% depending on the structure of each component, allows us to retain a weight-optimized design. The total performance of the manipulator can be enhanced by making better use of available resources and designing components with an optimum mix of strength and weight. Some benefits of 3D printing these components are as follows.

- Flexible Design. 3D printing allows for the design and print of more complex designs than traditional manufacturing processes.
- Rapid Prototyping.
- Print on Demand.
- Strong and Lightweight Parts.
- Fast Design and Production.
- Minimizing Waste.
- Cost Effective.
- Ease of Access.

4.2 Movement of the Manipulator:

There are two main contributors to our manipulator's success: first, the graphical user interface (GUI) built into the processing software, which we covered earlier.

Communication between hardware and software

Typically, a conventional graphical user interface (GUI) will use a communication protocol to transfer information to and from the microcontroller (Arduino or Raspberry Pi) that operates the manipulator arm of a robot. Here are the key components of our robotic arm's interface.

Custom GUI Development: We need to build unique GUIs with the help of programming languages and GUI frameworks like PyQt, Tkinter, Processing, or anything similar. To manipulate the robot arm, the user interface will have input fields, sliders, and buttons.

Create a communication channel: There must be some way for the GUI and the microcontroller to talk to one another. In this context, commonly used communication protocols include:

a. Serial communication (UART): For do-it-yourself robots, this is one of the most common and straightforward methods of interaction. The Universal Asynchronous Receiver/Transmitter (UART) protocol is used for communication between the microcontroller and the graphical user interface over a serial connection (such as USB).

b. Wi-Fi Communication: Wirelessly transmitting and receiving instructions and data is possible with the use of Wi-Fi communication protocols (such TCP/IP) and microcontrollers that have Wi-Fi capabilities, like the Raspberry Pi.

c. Bluetooth: Bluetooth communication can also be used for wireless control and data exchange, especially if your microcontroller supports Bluetooth connectivity.

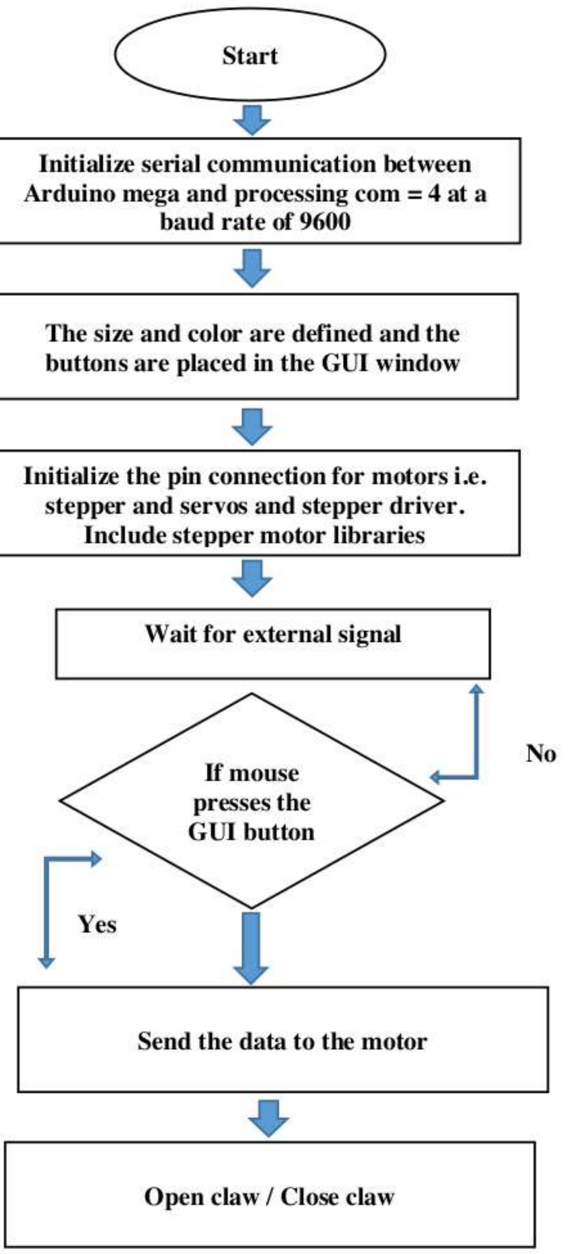
d. USB Communication: In some cases, you can use USB communication to connect the GUI directly to the microcontroller's USB port.

Sending commands and receiving data: The GUI sends commands to the microcontroller based on user input. For example, if the user wants the robot arm to move to a specific location, the GUI sends the appropriate action command to the microcontroller.

Interpret the commands in the microcontroller: The microcontroller software reads the commands coming from the communication channel. The software will have the logic necessary to interpret these commands and translate them into actions for the robot arm.

Controlling the robot arm: The microcontroller sends control signals to the motor of the robot arm based on the commands received. It calculates the joint angle required to reach the desired Position (forward kinematics) or receives a direct joint angle command (reverse kinematics) and activates the actuator.

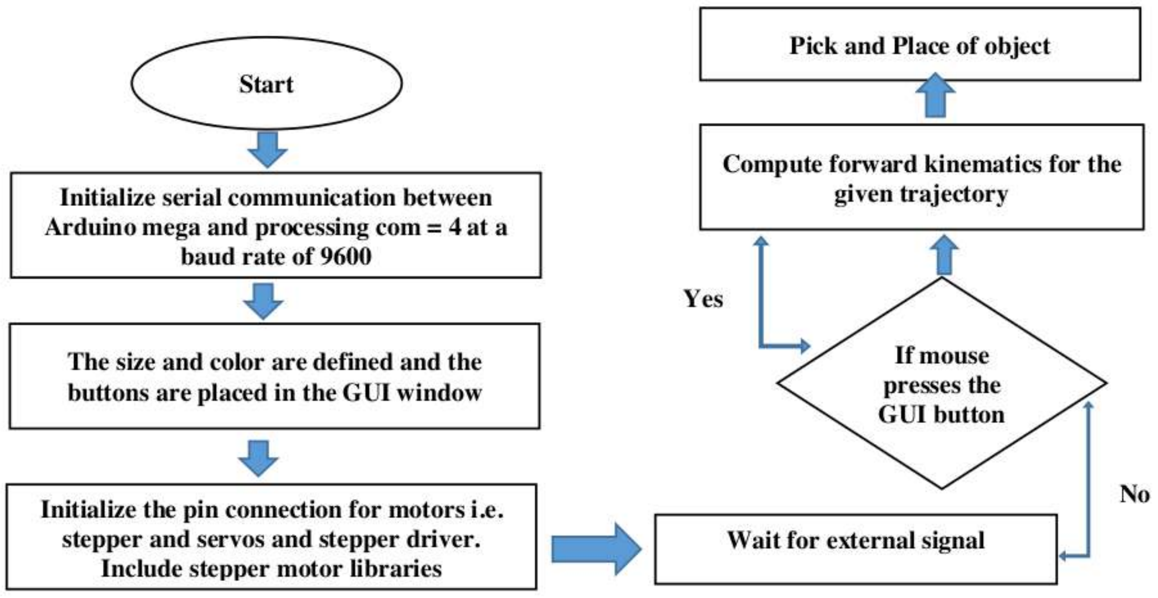
4.3 Flow Chart of Communication between GUI and Robotic Arm for opening and closing Gripper


Start
Initialize serial communication between Arduino mega and processing com = 4 at a baud rate of 9600
The size and color are defined and the buttons are placed in the GUI window
Initialize the pin connection for motors i.e. stepper and servos and stepper driver. Include stepper motor libraries
Wait for external signal
If mouse presses the GUI button
No
Yes
Send the data to the motor
Open claw / Close claw

4.2 Automated Trajectory:

To precisely move an end-effector from one spot to another, our manipulator uses feed-forward kinematics and an automated trajectory for a pick-and-place operation. The end effector's Cartesian coordinates (position and orientation) are determined with respect to the robot's base frame using the kinematic model's geometric characteristics and joint angles. The forward kinematics algorithm determines the joint angles required to reach the desired gripping and putting positions based on the user-entered positions and orientations. Using the determined joint angles, the robotic arm's end effector is guided to the object's location during the pickup phase. Once the object is in the robot's grasp, it will move along the estimated trajectory to the target location. Following this path guarantees a safe and direct journey free of accidents. The robotic arm is well-suited for a number of industrial and automation applications due to its ability to conduct pick-and-place activities efficiently and precisely using forward kinematics for automated trajectory planning.

4.4 Flow Chart of Automated Trajectory


Start
Initialize serial communication between Arduino mega and processing com = 4 at a baud rate of 9600
The size and color are defined and the buttons are placed in the GUI window
Initialize the pin connection for motors i.e. stepper and servos and stepper driver. Include stepper motor libraries
Wait for external signal
If mouse presses the GUI button
No
Yes
Compute forward kinematics for the given trajectory
Pick and Place of object

4.5 Microstepping

When compared to the more common full- or half-step driving methods, microstepping provides smoother and more precise motion when controlling stepper motors. The number of teeth on a stepper motor's rotor and stator determines the motor's step size. A predetermined angle is moved in each step. Microstepping is a more continuous and variable method of regulating the current via a motor's windings. The result is a situation where the motor is halfway between its normal steps. In this approach, the precision and accuracy of motor motions are greatly enhanced by microstepping. Pulse width modulation (PWM) is used to create microstepping by varying the amount of current flowing through the motor's windings at any one time. In order to make the motor transition smoothly between full steps, the driver modifies the present level to establish an intermediate position. Depending on the motor and controller, the standard microstepping mode can have steps as small as 1/2, 1/4, 1/8, 1/16, and even smaller. Microstepping allows for quieter operation, smoother motion, and more precise positioning. Microstepping may cause a marginal drop in motor torque relative to full stepping, though. Microstepping mode selection takes into account the trade-off between smoothness and torque for a given application. 3D printers, CNC machines, robotic systems, and other precision control systems all make use of microstepping to achieve smooth and exact motion.

4.5.1 Microstepping Calculations

We are using Nema 17 stepper motor which is commonly used stepper motor size, typically with 1.8 degrees of step per step. Since it is a 2-phase stepper motor, there are 200 complete steps per revolution (360 degrees per step / 1.8 degrees = 200 steps). And we are using 1/4 microstepping, this means that each subsequent step is divided into 4 microsteps. This effectively increases the number of steps per revolution. To calculate the number of microsteps per revolution, you can use this formula:

- **Microsteps per revolution =** (Complete steps per revolution) x (Value of Microstepping)
- **Microsteps per revolution =** 200 full steps x 4 microsteps per full step = 800 microsteps

4.6 Ball Bearing Disk

We used the ball bearings disk upon which our base is moving by using this application we achieved the desired advantages we were seeking in our structure some of them are mentioned below:

Reduced friction: ball bearings provide smooth, low-friction rotation that allows the robot arm to move with minimal resistance. This leads to increased efficiency and reduced component wear.

Increased bearing capacity: Ball bearings can support higher loads compared to plain bearings. This allows the swing arm to handle heavier loads while maintaining stability and accuracy.

Precise Positioning: Ball bearings offer precise and controlled motion that allows the base of the robotic arm to be positioned precisely at a specific angle or direction. This accuracy is essential to perform complex tasks and accurately reach the desired end point.

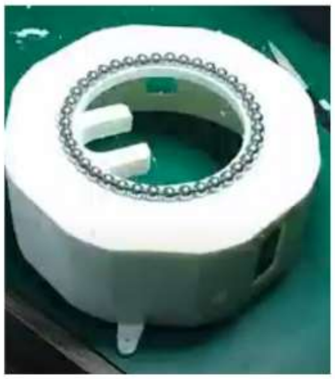
Long life and durability: Ball bearings are designed to withstand high-speed rotation and frequent movements, making them reliable and durable for long-term operation. They have a long life compared to some other types of bearings, reducing maintenance requirements.

Low maintenance: Ball bearings require minimal maintenance due to their robust design and reduced friction. This results in savings and less work time for the robot arm.

Smooth operation: the smooth and consistent rotation provided by the direction of the ball ensures stable and unobstructed movement of the robot arm, improving overall productivity.

Reduced heat output: Low friction in ball bearings produces less heat generation during rotation. This is especially important if the robot arm operates continuously or at high speeds.

Design flexibility: Ball bearings come in a variety of sizes and configurations, providing flexibility in designing the base of the robot arm. These adaptations make it easy to customize the robot arm to meet specific application requirements.

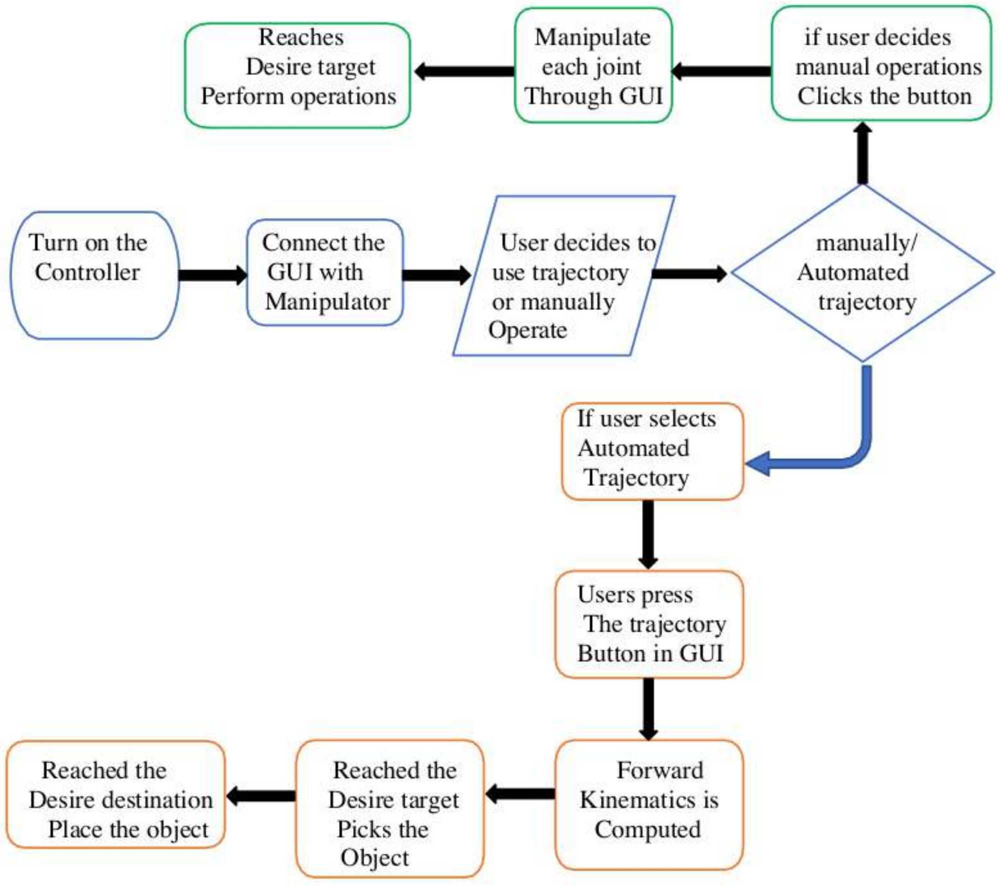


Chapter 5

Figure 3.9 Ball Bearing Disk

RESULTS AND DISCUSSION

5.1 Workflow Chart


Reaches Desire target Perform operations
Manipulate each joint Through GUI
if user decides manual operations Clicks the button
Turn on the Controller
Connect the GUI with Manipulator
User decides to use trajectory or manually Operate
manually/ Automated trajectory
If user selects Automated Trajectory
Users press The trajectory Button in GUI
Reached the Desire destination Place the object
Reached the Desire target Picks the Object
Forward Kinematics is Computed

5.2 Control System

Following are the types of controllers and control systems are used in this project.

1. Arduino for sensor data accusation and motion control (Kinematic codes)

2. GUI (Graphical User interface)

5.2.1 Micro-Controller (Arduino Mega):

Arduino was chosen for this project due to its low cost, wide availability, and ease of programming and interfacing with other electronic components. In addition to their versatility in terms of programming and use, Arduinos are easy to link between many languages and applications. In this setup, we employ not one but two Arduino Mega to manage the system. The manipulator's movement and trajectory are managed by an Arduino. Forward kinematics is used to plot a course to a predetermined destination based on the input of angles and velocities.

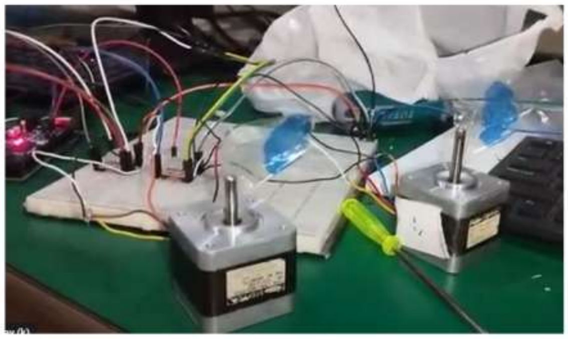


Figure 5.1 Arduino mega controlling stepper using CNC shield

5.2.2 GUI (Graphical User interface)

The GUI gives some extra leverage to the user if they want to manipulate the joints according to their desired requirements

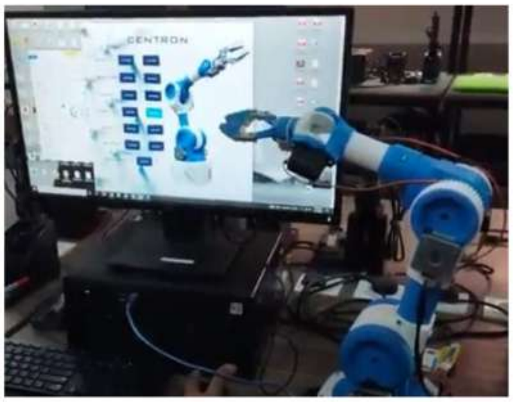


Figure 5.2 Manipulator Controlled by custom GUI

5.3 Presentation of Findings:

Following are the findings of project includes both hardware and software results in detail manner.

5.3.1 Hardware Results:

In order to prevent any issues with the manipulator's motion while in use, thorough research and meticulous design were performed prior to manufacture. The manipulator's motion is consistent and smooth in all directions, regardless of the operating speed. The load on the manipulator motor is high enough to supply the torque required for the device to function smoothly. The numbers are as follows.

The link's weight and the motor's load were both reduced thanks to the employment of form optimization techniques. Because of this, the manipulator can move smoothly while using less energy. The servo-driven handle's motion is determined by direct angle calculation. The manipulator's size and carrying capacity informed the design of the gripper. The rubberized grip on the manipulator's handle makes it easy to open and close the jaws, and prevents it from slipping out of the user's hand. Effective and dependable manipulator implementation is a byproduct of an integrated design strategy, shape optimization, and the use of high-quality motors and components.

5.3.1.1 Workspace Calculations:

The maximum workspace use in this project is around 3.5 Feet and the designed Workspace is about 2 feet as it depends on the tilt angle of End-factor. The End-factor usein this project is a gripper with gear mechanism.

5.3.2 Software Results:

In the same way that the goal of this project is to collect various objects and transport them to various storage areas, depending on what they are. Both manipulators' movements and logic are managed by an Arduino micro-controller. Because Arduino allows for precise control over the manipulator's velocity and acceleration at each link.

5.3.3 System Results and Accuracy:

The reliability of the system has been put through a variety of tests. Some of the tests involve evaluating the precision of the manipulator under varying illumination conditions. Findings from the lighting analysis are explained under the project's constraints. Pick-and-place testing consists of four sets, each of which consists of twenty attempts to move an item from its starting position at the target to its final resting place at the destination.

Table 5.1 Test Results

No of tests	Successful Tests	Failed Tests
1	**16**	**4**
2	**17**	**3**
3	**18**	**2**
4	**18**	**4**

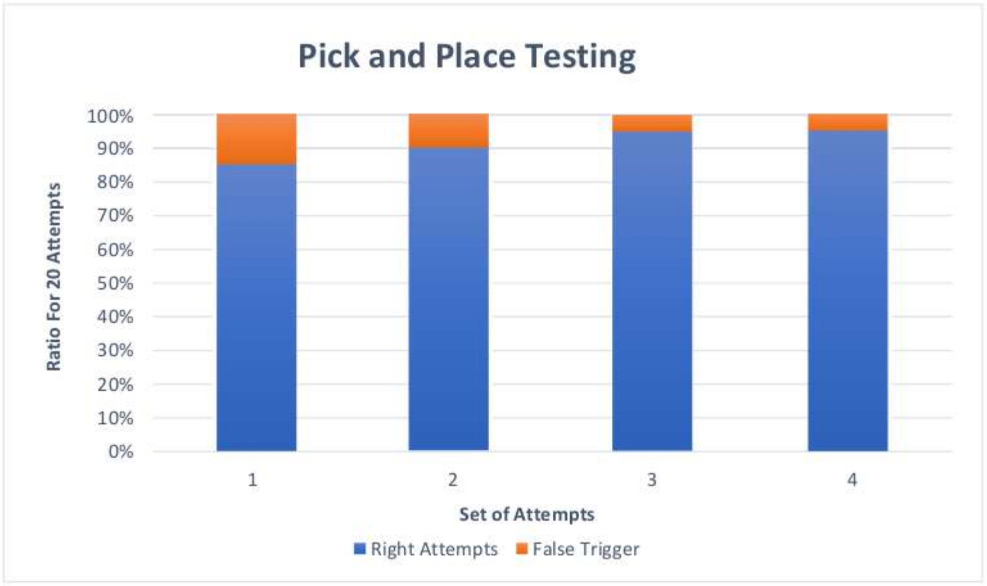


Figure 5.3 Success and Error Graph

5.4 Limitations & Solutions:

Some limitations of our projects are following:

5.4.1 Trajectory Plan:

Proper predefined trajectory plan is required for manipulator motion. As if an object is needed to pick and place from a specific location. A predefine trajectory must be calculated and store in the Arduino.

5.4.3 Arduino Limitations

Arduino can provide small level flexibility to facilitate your hardware otherwise it results in getting destroyed. As in this project a person is working with six motors and other equipment's to so high processing power be required for data processing and controlling. Therefore, it is preferable to use Raspberry Pie controller.

5.3.4 Load Capacity

The motors calculations are done for maximum carrying load of 500g. This limit can be increase by using higher torque stepper motors like Nema-23 or Nema-44 stepper motor instead of Nema-17. Also, the gripper servo motor can be change for higher torque motor.

5.3.5 Material Limitation

As most of this manipulator is 3D printed with PLA, so it works in temperatures below 45 Degree Celsius. So, for operation in temperature 45 Degree Celsius and above other material should be use. Where ABS can be used to increase the operation temperature range.

41

Chapter 6

CONCLUSION AND FUTURE RECOMMENDATION

6.1 Conclusion

The team behind this project set out to educate Pakistan's manufacturing sector and engineering elite on the importance of embracing automation and robotics in order to remain competitive in the global marketplace. The goal of this research was to develop an error- and fatigue-free robotic manipulator capable of performing the aforementioned tasks. Thus, this industrial manipulator features a 4-link construction with a rotating base. In order to keep costs low and facilitate widespread application, a basic design was embraced. The tabletop is 3.5 ft. in diameter. Motors were selected after careful consideration of the workspace, torques, velocities, and forward kinematics. A CAD model of the project was created using the link lengths and data calculated in SolidWorks. The prototype was 3D printed with a focus on weight minimization to cut down on bulk without sacrificing durability. Once again, solid works were used to fine-tune the design. The honeycomb-like structure of the manipulator was 3D printed to decrease its overall mass and boost its strength and rigidity. The manipulator is easy to put together thanks to its straightforward design. Arduino, Nema 17, Mg996-R servo, sg90 servo motor The manipulator's power and motion were both controlled by microcontrollers. A standard servo motor controlled the gripper's motions. Both a graphical user interface (GUI) with individual buttons for operating the manipulator and an automated trajectory using forward kinematics to approach the target and perform pick and place have been built for the pick and place mechanism. Due to increased processing capability and memory, upgrading the microcontroller from Arduino to Raspberry-pi will further allow us to do jobs with much more accuracy. The project's goals have been accomplished thus far, and the manipulator is successfully carrying out its assigned tasks.

6.2 Future Recommendations

There are several modification and improvement that can added in future, some of them are following.

6.2.1 Pick and Place through Object Detection

The manipulator is currently manipulated through GUI to perform pick and place in future we can introduce object detection feature in which the manipulator should sort the object and do pick and place.

6.2.2 Proper GUI with Multiple Modes

The Robotic Arm currently operated through GUI made on the processing in future we can implement a GUI based on G-Codes enhancing its versatility.

6.2.3 Mound on Mecanum wheel

The robotic arm currently operating in its desired workspace in further future we can mound our manipulator on the Mecanum wheel to enhance its efficiency.

6.2.4 Path Trajectory through inverse Kinematics

Currently our robotic arm manipulator is performing the path trajectory through forward kinematics as we are using the stepper motors. For future we can use the servos instead of stepper to optimized our path trajectory through inverse kinematics

6.2.5 Modular End-Factor:

Currently the Robotic Arm has one gripper attached to manipulator which enables it to pick and place objects. In future a universal clamp can be designed which will be fixed on manipulator. A number of attachments can then be used with this clamper such as welder, cutter, FDM nozzle for 3D-Printing and laser cutting etc. This modular end-effector design will allow for easy swap between different end-effectors and increase application of Robotic Arm.

Reference

[1] Iqbal, J., Islam, R. U., & Khan, H. (2012). Modeling and analysis of a 6 DOF robotic arm manipulator. *Canadian Journal on Electrical and Electronics Engineering*, *3*(6), 300-306.

[2] Tung, T. T., Van Tinh, N., Thao, D. T. P., & Minh, T. V. (2023). Development of a prototype 6 degree of freedom robot arm. *Results in Engineering*, *18*, 101049.

[3] Bilal, M., Khan, M. O., Mughal, A., & Ali, N. (2018). Design and Control of 6 DOF Robotic Manipulator. *Mechatronics & Manufacturing Engineering University of Engineering and Technology Lahore Faisalabad Campus*.

[4] Nguyen, M. T., Yuan, C., & Huang, J. H. (2019). Kinematic analysis of A 6-DOF robotic arm. In *Advances in Mechanism and Machine Science: Proceedings of the 15th IFToMM World Congress on Mechanism and Machine Science 15* (pp. 2965-2974). Springer International Publishing.

[5] Bhangale, P. P., Saha, S. K., & Agrawal, V. P. (2004). A dynamic model based robot arm selection criterion. *Multibody System Dynamics*, *12*, 95-115.

[6] Hernandez, J., Sunny, M. S. H., Sanjuan, J., Rulik, I., Zarif, M. I. I., Ahamed, S. I., ... & Rahman, M. H. (2023). Current designs of robotic arm grippers: a comprehensive systematic review. *Robotics*, *12*(1), 5.

[7] H. R. R. B. T. K. S. Z. F. S. K. Rajashekar K, “Robotic Arm Control Using Arduino,” 2020. [Online https://www.jetir.org/view?paper=JETIR2006065.

[8] A. G. Rahul Gautam, “Review on Development of Industrial Robotic Arm,” 2017. [Online]. Availa https://www.researchgate.net/publication/322926656_Review_on_Development_of_Industrial_Rob

[9] R. E. S. B. S. K. I. Mohd Ashiq Kamaril Yusoff, “WIRELESS MOBILE ROBOTIC ARM,” [Onlin https://cyberleninka.org/article/n/443852.

[10] R. E. S. B. S. K. I. Mohd Ashiq Kamari Yusoff, “Wireless Mobile Robotic Arm,” 2012. [Online]. A https://www.sciencedirect.com/science/article/pii/S1877705812026859.

[11] A. H. K. T. A. M. M. S. Md Anisur Rahman, “Design, Analysis and Implementation of a Robotic A Animator,” 2013. [Online]. Available: https://www.researchgate.net/publication/269690255_Design_Analysis_and_Implementation_of_a_ The_Animator.

[12] B. K. K. S. L. K. Kruthika, “Design and development of a robotic arm,” [Online]. Available: https://ieeexplore.ieee.org/document/8053274.

[13] A. M. L. L. T. Jean Jiang, “Development of a motion controlled robotic arm,” 2017. [Online]. Avail https://ieeexplore.ieee.org/document/8248998.

[14] J. Iqbal, “Modern Control Laws for an Articulated Robotic Arm: Modeling and Simulation,” 2019. [Available: https://etasr.com/index.php/ETASR/article/view/2598.

Appendix:

Arduino Code

```
#include <Servo.h>
Servo servo;
Servo servo1;
Servo servo_sg;

int servoPosition1 = 0;
int servoIncrement1 = 5;

int servoPosition2 = 0;
int servoIncrement2 = 5;

#define EN1 8
#define X_DIR1 5
#define X_STP1 2

#define EN2 9
#define Y_DIR2 6
#define Y_STP2 3

#define EN3 10
#define Z_DIR3 7
#define Z_STP3 4

#define EN4 11
#define A_DIR4 12
#define A_STP4 13
```

```
int delayTime = 700; //this is the delay between pulses, changing this will change speed of motor

void step(boolean dir, byte dirPin, byte stepperPin, int steps)
{
  digitalWrite(dirPin, dir);
  delay(100);
  for (int i = 0; i < steps; i++)
  {
    digitalWrite(stepperPin, HIGH);
    delayMicroseconds(delayTime);
    digitalWrite(stepperPin, LOW);
    delayMicroseconds(delayTime);
  }
}

void setup()
{
  servo.attach(30);
  servo.write(0);
  servo1.attach(28);
  servo1.write(45);
  servo_sg.attach(26);
  pinMode(X_DIR1, OUTPUT);
  pinMode(X_STP1, OUTPUT);
  pinMode(EN1, OUTPUT);
  digitalWrite(EN1, LOW);

  pinMode(Y_DIR2, OUTPUT);
  pinMode(Y_STP2, OUTPUT);
  pinMode(EN2, OUTPUT);
  digitalWrite(EN2, LOW);
```

```
 pinMode(Z_DIR3, OUTPUT);
 pinMode(Z_STP3, OUTPUT);
 pinMode(EN3, OUTPUT);
 digitalWrite(EN3, LOW);

 Serial.begin(9600);    //start serial communication @9600 bps




}

void loop()
{
  if(Serial.available()){  //id data is available to read

   char val = Serial.read();

   if(val == 'A'){
   step(true, X_DIR1, X_STP1, 200);
  }
   if(val == 'a'){
   step(false, X_DIR1, X_STP1, 200);
  }
   if(val == 'B'){
   step(true, Y_DIR2, Y_STP2, 200);
    }
   if(val == 'b'){
   step(false, Y_DIR2, Y_STP2, 200);
    }
   if(val == 'C'){
```

```
step(true, Z_DIR3, Z_STP3, 200);
  }
if(val == 'c'){
step(false, Z_DIR3, Z_STP3, 200);
  }
if(val == 'D'){     //if b received
servoPosition1 += servoIncrement1;
servo.write(servoPosition1);
  }
if(val == 'd'){       //if y received
servoPosition1 -= servoIncrement1;
servo.write(servoPosition1);
  }
if(val == 'E'){     //if b received
servoPosition2 += servoIncrement2;
servo.write(servoPosition2);
  }
if(val == 'e'){       //if y received
servoPosition2 -= servoIncrement2;
servo.write(servoPosition2);
  }
if(val == 'F'){       //if b received
servo_sg.write(0);
  }
if(val == 'f'){       //if y received
servo_sg.write(90);
  }
// if(val == 'f'){       //if f received
//   digitalWrite(11, LOW); //turn off all led
//   digitalWrite(12, LOW);
//   digitalWrite(10, LOW);
```

```
  //   }
  }


//  step(true, Y_DIR2, Y_STP2, 500);
//  delay(1000);
//  step(false, Y_DIR2, Y_STP2, 500);
//  delay(1000);

//  step(false, Z_DIR3, Z_STP3, 300);
//  delay(1000);
//  step(true, Z_DIR3, Z_STP3, 300);
//  delay(1000);

}
```

GUI Processing Code

```
import controlP5.*; //import ControlP5 library
import processing.serial.*;
Serial port;
ControlP5 cp5; //create ControlP5 object
PFont font;
PFont font1;
PFont font2;
PImage img;
void setup(){ //same as arduino program
  size(1000, 1000);    //window size, (width, height)
  printArray(Serial.list());   //prints all available serial ports
  port = new Serial(this, "COM5", 9600);//i have connected arduino to com3, it would be different
in linux and mac os
  size(1000,1000);
```

```
img = loadImage("Robotic2.jpeg");
//lets add buton to empty window
cp5 = new ControlP5(this);
font = createFont("Bank Gothic Medium BT", 65);    // custom fonts for buttons and title
font1 = createFont("Bank Gothic Medium BT", 25);
font2 = createFont("Bank Gothic Medium BT", 15);
cp5.addButton("J1Pos")     //"red" is the name of button
  .setPosition(220, 145)  //x and y coordinates of upper left corner of button
  .setSize(100, 60)      //(width, height)
  .setFont(font2)
;
cp5.addButton("J1Neg")     //"yellow" is the name of button
  .setPosition(370, 145)  //x and y coordinates of upper left corner of button
  .setSize(100, 60)      //(width, height)
  .setFont(font2)
;
  cp5.addButton("J2Pos")     //"yellow" is the name of button
  .setPosition(220,245 )  //x and y coordinates of upper left corner of button
  .setSize(100, 60)      //(width, height)
  .setFont(font2)
;
  cp5.addButton("J2Neg")     //"yellow" is the name of button
  .setPosition(370, 245)  //x and y coordinates of upper left corner of button
  .setSize(100, 60)      //(width, height)
  .setFont(font2)
;
cp5.addButton("J3Pos")     //"blue" is the name of button
  .setPosition(220, 345)  //x and y coordinates of upper left corner of button
  .setSize(100, 60)      //(width, height)
  .setFont(font2)
;
```

```
 cp5.addButton("J3Neg")     //"blue" is the name of button
  .setPosition(370, 345)  //x and y coordinates of upper left corner of button
  .setSize(100, 60)      //(width, height)
  .setFont(font2)
;
cp5.addButton("J4Pos")     //"alloff" is the name of button
  .setPosition(220, 445)  //x and y coordinates of upper left corner of button
  .setSize(100, 60)      //(width, height)
  .setFont(font2)
;
   cp5.addButton("J4Neg")     //"blue" is the name of button
  .setPosition(370, 445)  //x and y coordinates of upper left corner of button
  .setSize(100, 60)      //(width, height)
  .setFont(font2)
;
cp5.addButton("J5Pos")     //"alloff" is the name of button
  .setPosition(220, 545)  //x and y coordinates of upper left corner of button
  .setSize(100, 60)      //(width, height)
  .setFont(font2)
;
  cp5.addButton("J5Neg")     //"alloff" is the name of button
  .setPosition(370,545)  //x and y coordinates of upper left corner of button
  .setSize(100,60)      //(width, height)
  .setFont(font2)
;
cp5.addButton("J6Pos")     //"alloff" is the name of button
  .setPosition(220, 645)  //x and y coordinates of upper left corner of button
  .setSize(100, 60)      //(width, height)
  .setFont(font2)
;
 cp5.addButton("J6Neg")     //"alloff" is the name of button
```

```
    .setPosition(370, 645)  //x and y coordinates of upper left corner of button
    .setSize(100, 60)      //(width, height)
    .setFont(font2)
  ;
    cp5.addButton("Start")     //"alloff" is the name of button
    .setPosition(290, 745)  //x and y coordinates of upper left corner of button
    .setSize(100, 60)      //(width, height)
    .setFont(font2)
  ;

}

void draw(){  //same as loop in arduino
  background(255, 255 , 255); // background color of window (r, g, b) or (0 to 255)
  image(img, 0, 0);
  //lets give title to our window
  fill(0, 0, 0);               //text color (r, g, b)
  textFont(font);
  text("CENTRON", 280, 80);  // ("text", x coordinate, y coordinat)
  textFont(font1);
  text("Base", 80, 180);  // ("text", x coordinate, y coordinat)
  text("Sholuder", 80, 280);  // ("text", x coordinate, y coordinat)
  text("Elbow", 80, 380);  // ("text", x coordinate, y coordinat)
  text("Arm", 80, 480);  // ("text", x coordinate, y coordinat)
  text("Wrist", 80, 580);  // ("text", x coordinate, y coordinat)
  text("Gripper", 80, 680);  // ("text", x coordinate, y coordinat)
  text("Trajectory1", 80, 780);  // ("text", x coordinate, y coordinat)
}
//lets add some functions to our buttons
//so whe you press any button, it sends perticular char over serial port
void J1Pos(){
  port.write('A');
}
void J1Neg(){
  port.write('a');
}
void J2Pos(){
  port.write('B');
}
void J2Neg(){
  port.write('b');
}
void J3Pos(){
  port.write('C');
}
void J3Neg(){
  port.write('c');
}
void J4Pos(){
  port.write('D');
}
void J4Neg(){
  port.write('d');
}
void J5Pos(){
  port.write('E');
}
void J5Neg(){
  port.write('e');
}
void J6Pos(){
  port.write('F');
```

```
}
void J6Neg(){
  port.write('f');
}
void Start(){
  port.write('S');
}
```

gg

ORIGINALITY REPORT

11%	9%	4%	6%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	repository.tcu.edu Internet Source	2%
2	forum.arduino.cc Internet Source	1%
3	"Advances in Mechanism and Machine Science", Springer Science and Business Media LLC, 2019 Publication	1%
4	Submitted to Higher Education Commission Pakistan Student Paper	1%
5	fabacademy.org Internet Source	1%
6	dokumen.pub Internet Source	<1%
7	edge.rit.edu Internet Source	<1%
8	"Advances in Robot Kinematics 2022", Springer Science and Business Media LLC,	<1%

	2022 Publication	
9	Submitted to Temple University Student Paper	<1 %
10	issuu.com Internet Source	<1 %
11	slideheaven.com Internet Source	<1 %
12	link.springer.com Internet Source	<1 %
13	documents.mx Internet Source	<1 %
14	www.hackster.io Internet Source	<1 %
15	"Machines, Mechanism and Robotics", Springer Science and Business Media LLC, 2022 Publication	<1 %
16	doaj.org Internet Source	<1 %
17	Submitted to Virginia Polytechnic Institute and State University Student Paper	<1 %
18	Submitted to Durban University of Technology Student Paper	<1 %

19	M. Shimizu. "Analytical Inverse Kinematic Computation for 7-DOF Redundant Manipulators With Joint Limits and Its Application to Redundancy Resolution", IEEE Transactions on Robotics, 10/2008 Publication	<1 %
20	uotechnology.edu.iq Internet Source	<1 %
21	www.wlkata.com Internet Source	<1 %
22	journal.umy.ac.id Internet Source	<1 %
23	www.dol.gov Internet Source	<1 %
24	www.wlkata.shop Internet Source	<1 %
25	www.tjprc.org Internet Source	<1 %
26	dspace.dtu.ac.in:8080 Internet Source	<1 %
27	Tran Thanh Tung, Nguyen Van Tinh, Dinh Thi Phuong Thao, Tran Vu Minh. "Development of a prototype 6 degree of freedom robot arm", Results in Engineering, 2023 Publication	<1 %

28	Submitted to University of Western Sydney Student Paper	<1 %
29	djvu.online Internet Source	<1 %
30	peerj.com Internet Source	<1 %
31	Cleghorn, William. "Mechanics of Machines", Oxford University Press Publication	<1 %
32	Submitted to Montana State University, Bozeman Student Paper	<1 %
33	Submitted to University of Strathclyde Student Paper	<1 %
34	Submitted to University of Leeds Student Paper	<1 %
35	Submitted to Vietnam Maritime University Student Paper	<1 %
36	Submitted to Liverpool Hope Student Paper	<1 %
37	Mangesh Saraf, A Agarwal, A Chaudhary, A Ganthale. "Kinematic Modelling and Motion Mapping of Robotic Arms", Journal of Physics: Conference Series, 2021 Publication	<1 %

38	pt.scribd.com Internet Source	<1 %
39	Submitted to Piri Reis University Student Paper	<1 %
40	Submitted to University of Bradford Student Paper	<1 %
41	eprints.utm.my Internet Source	<1 %
42	psasir.upm.edu.my Internet Source	<1 %
43	Péter Balázs. "An Evolutionary Approach for Object-Based Image Reconstruction Using Learnt Priors", Lecture Notes in Computer Science, 2009 Publication	<1 %
44	www.researchgate.net Internet Source	<1 %
45	Mechanisms and Machine Science, 2015. Publication	<1 %
46	elibrary.tucl.edu.np Internet Source	<1 %
47	infostore.saiglobal.com Internet Source	<1 %
48	patents.google.com Internet Source	<1 %

49	www.machineseeker.by Internet Source	<1 %
50	V G Pratheep, M Chinnathambi, E B Priyanka, P Ponmurugan, Pridhar Thiagarajan. "Design and Analysis of six DOF Robotic Manipulator", IOP Conference Series: Materials Science and Engineering, 2021 Publication	<1 %
51	cv.archives-ouvertes.fr Internet Source	<1 %
52	docksci.com Internet Source	<1 %
53	ozrobotics.com Internet Source	<1 %
54	repository.usd.ac.id Internet Source	<1 %
55	www.mesj.ukim.edu.mk Internet Source	<1 %
56	T. Doi. "Teleoperation system of ETS-VII robot experiment satellite", Proceedings of the 1997 IEEE/RSJ International Conference on Intelligent Robot and Systems Innovative Robotics for Real-World Applications IROS 97 IROS-97, 1997 Publication	<1 %

57	Hina Raja, M. Usman Akram, Sajid Gul Khawaja, Muhammad Arslan, Aneeqa Ramzan, Noman Nazir. "Data on OCT and fundus images for the detection of glaucoma", Data in Brief, 2020 Publication	<1 %

Exclude quotes	On	Exclude matches	Off
Exclude bibliography	On		