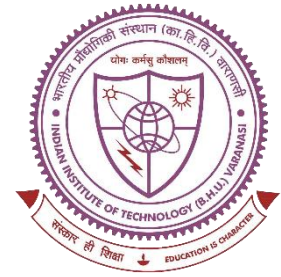


INDIAN INSTITUTE OF TECHNOLOGY
BANARAS HINDU UNIVERSITY
VARANASI



UNDERGRADUATE PROJECT

REPORT ON

Autonomous Robotic Arm

BY

Anushka Kumari (21135023)

Garvit Singhal (21135054)

Harshit Joshi (21135059)

Hemanth Patel (21135060)

Epsa (21134009)

Department of Mechanical Engineering

Under the esteemed guidance of :

Dr. Amit Tyagi

Department of Mechanical Engineering

Indian Institute of Technology (BHU) Varanasi

ACKNOWLEDGEMENT

We would like to express our sincere gratitude towards **Dr. AMIT TYAGI** (Department of Mechanical Engineering) for their guidance and constant supervision throughout this project. His willingness to share his vast knowledge helped us understand this project and its manifestations deeply. Without his support, we would not have been able to complete this project successfully.

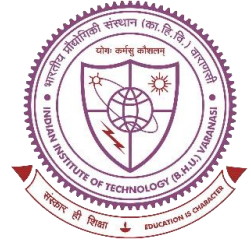
We also express our most profound obligation towards **ROBOTICS CLUB (IIT BHU)** for its significant expertise and resources that allowed us to bring our imagination into physical existence.

Finally, we would like to thank our friends, family, and people who supported us in our efforts during this project.

भारतीय
प्रौद्योगिकी
संस्थान
काशी हिन्दू विश्वविद्यालय



INDIAN
INSTITUTE OF
TECHNOLOGY
BANARAS HINDU UNIVERSITY



Department of Mechanical Engineering
यांत्रिक इंजीनियरिंग विभाग

CERTIFICATE

This is to certify that the project entitled “**Autonomous Robotic Arm,**” submitted for the undergraduate project in Part-3 Mechanical Engineering, Indian Institute of Technology (BHU) Varanasi - 221005, is a bonafide project work carried out by the following students under my guidance and supervision:

- 1) Anushka Kumari (21135023)
- 2) Garvit Singhal (21135054)
- 3) Harshit Joshi (21135059)
- 4) Hemanth Patel (21135060)
- 5) Epsa (21134009)

Project Supervisor:
Dr Amit Tyagi

Dr. Santosh Kumar
Head of Department
Mechanical Engineering
IIT-BHU, Varanasi

Contents:

❖ Introduction and Background	4
❖ Design and Fabrication	5 - 10
❖ Electronics and Control Systems	11 - 12
❖ Kinematics	13 - 15
❖ Analysis	16 - 18
❖ Programming	19 - 21
❖ Calibration	22 - 24
❖ Conclusion	25 – 26
❖ References	26

Introduction

Project Overview:

Our project focuses on developing an autonomous robotic arm capable of manipulating objects using advanced visual perception and voice command controls. Leveraging Arduino microcontrollers and servo drivers, we achieved precise joint angle control for intricate movements. The robotic arm, crafted with optimization and durability in mind, features a 3-degree-of-freedom (DOF) design with 60cm and 40cm link lengths, optimized using MATLAB's fmincon function. We chose lightweight aluminum over wood for its agility and durability, avoiding deformation issues. Innovative design features, like a glove and slot mechanism for base rotation and strategically placed ball bearings in arm joints, mitigate radial and axial forces on servo mechanisms, preserving torque capacity and extending servo lifespan. Aluminum construction also aids in effective heat dissipation, preventing servo overheating during extended operation.

Robotic arm and its applications:

A robotic arm is a mechanical device designed to mimic the functionality of a human arm, typically consisting of a series of interconnected segments or links with joints, controlled by motors or actuators. These arms are versatile tools with a wide range of applications across multiple industries. In manufacturing, robotic arms are employed for tasks such as assembly, welding, painting, and packaging, where they offer precision, consistency, and efficiency, leading to increased productivity and cost savings. In healthcare, robotic arms assist in surgical procedures, enabling surgeons to perform delicate operations with enhanced precision and minimally invasive techniques, reducing patient trauma and recovery time. Additionally, robotic arms find applications in space exploration, underwater exploration, agriculture, and more, demonstrating their adaptability and importance in advancing technological frontiers and improving human quality of life.

Design And Fabrication

The design process for the robotic arm encompassed several stages, beginning with conceptualization and concluding with prototyping and testing.

Materials and Components Used:

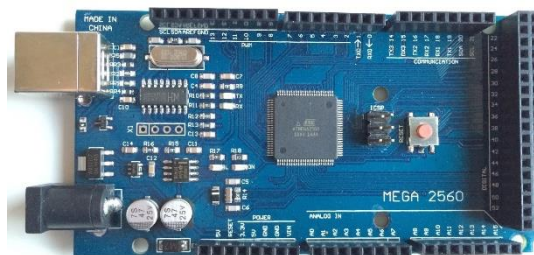
The choice of materials and components played a crucial role in shaping the robotic arm's functionality, durability, and performance. Key selections included:

Aluminium Alloys for Base: The base of the robotic arm was constructed using aluminum alloys, chosen for their lightweight yet robust properties. Aluminum alloys provided a sturdy foundation while minimizing overall weight, enhancing the arm's agility and maneuverability.

Aluminium Extrusion Bars for Arms Length: To achieve the desired reach and flexibility, aluminum extrusion bars were utilized for the arm's length. These bars offered versatility in design and assembly, allowing for precise customization to meet project specifications.



Arduino Microcontroller: The Arduino Mega 2560 is powered by an ATmega2560 microcontroller chip, which acts as the board's brain. It executes your programmed code and controls various input/output operations. It has a USB port allows you to connect it to a computer for programming and serial communication.



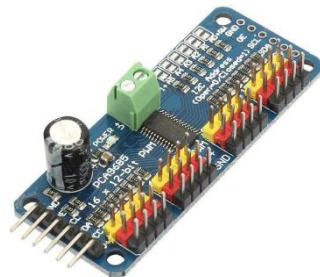
Power Supply: Wanptek KPS3020D 30V/20A High Precision Adjustable Digital DC Power Supply is used. This series has a single set of output and a high-precision double display switch DC power supply.



Servo Motors: Servo motors are commonly used in robotics and automation to control mechanisms' position, velocity, and acceleration precisely. They comprise a DC motor, a gear train, and a feedback control system.



Adafruit Servo Controller: To streamline the control and management of multiple servo motors, an Adafruit servo controller was employed. This compact and versatile controller facilitated seamless integration with the Arduino microcontroller, enabling synchronized control of servo motors with minimal wiring complexity.



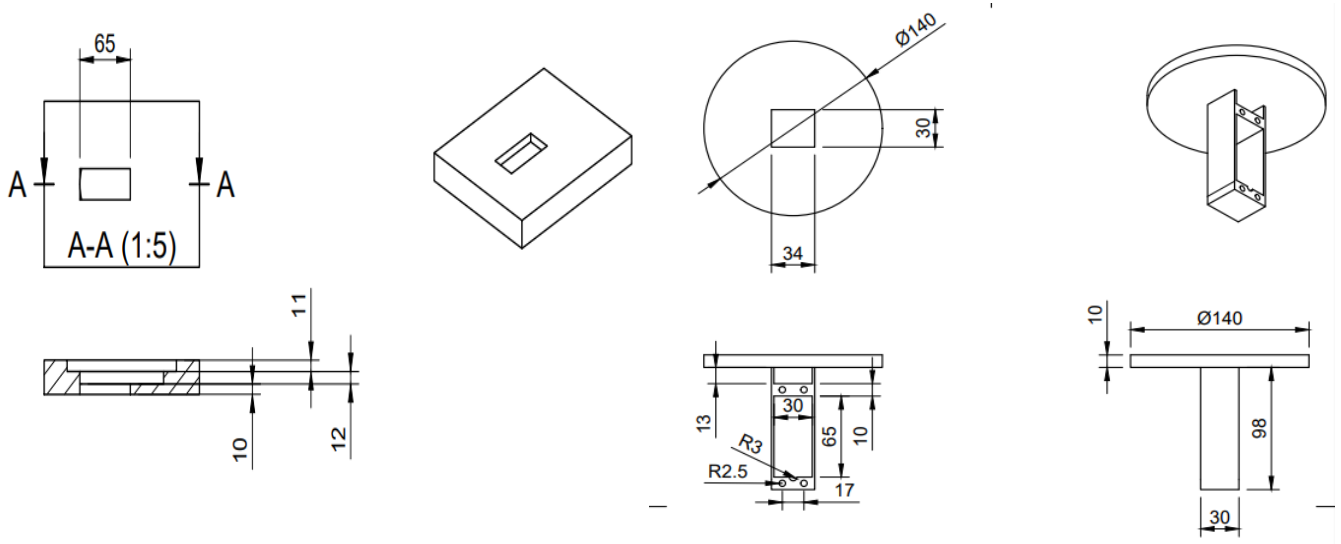
CAD DESIGN

The CAD (Computer-Aided Design) phase of the project involved the creation of detailed digital models to visualize and refine the robotic arm's mechanical structure and components. We used FUSION 360 for creating the CAD designs.

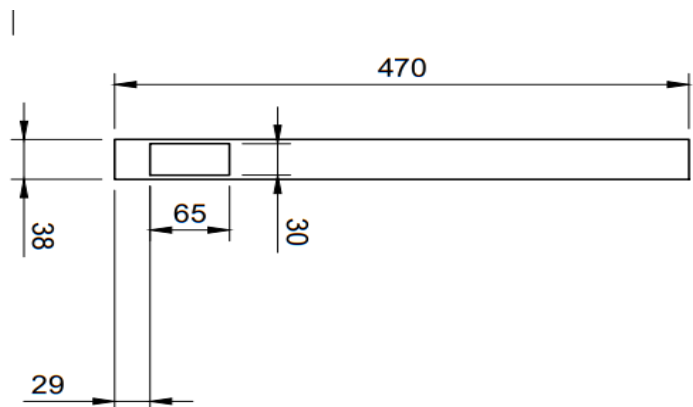
Visual Representation:

The design photos presented herein offer a comprehensive visual representation of the robotic arm's CAD design. These images showcase various aspects of the arm's architecture, including:

Base Structure: The foundational framework of the robotic arm, providing stability and support for the entire assembly. Detailed views highlight the arrangement of aluminium alloy components and the integration of mounting points for additional hardware.



Arm Segments: Each robotic arm segment, constructed from aluminium extrusion bars, is meticulously depicted in the CAD design. The modular nature of the arm allows for flexibility in configuration and extension, as illustrated in the provided images.



Joint Mechanisms: Close-up views of the joint mechanisms reveal the intricate details of servo motor mounting, gearing arrangements, and articulation points. These components are essential for facilitating precise motion control and coordination throughout the arm's range of motion.

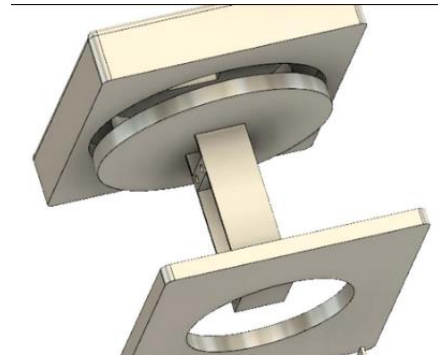
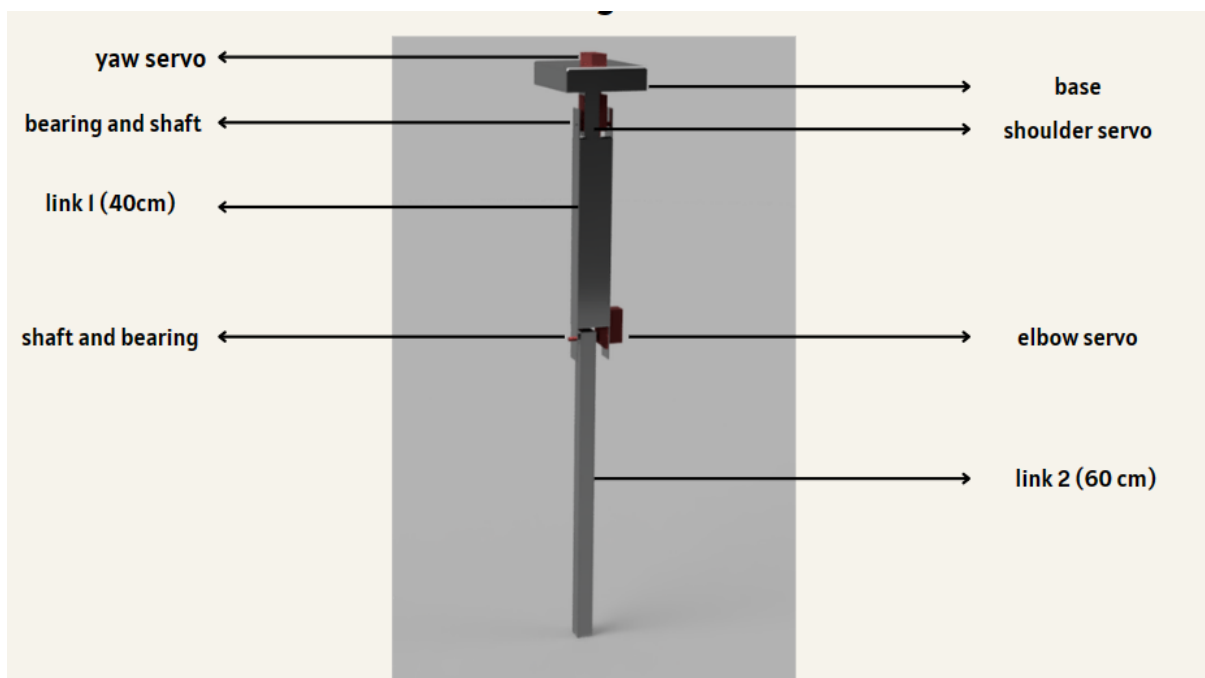


Fig. Robotic Arm CAD Model



Manufacturing of parts:

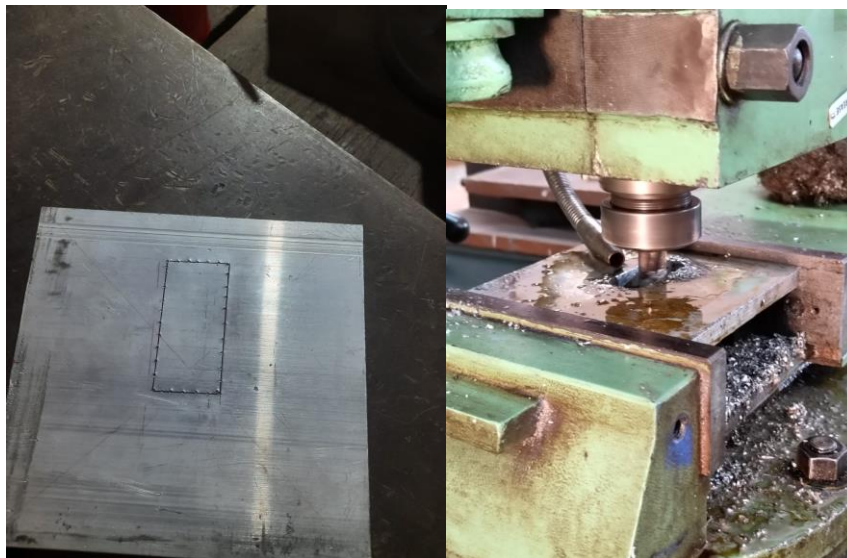
Wire EDM

The wire-cut EDM process is straightforward: the workpiece is submerged in dielectric fluid, clamped securely, and a wire carrying electric current passes through it, creating sparks that melt metal away. The process relies on the electric spark as the cutting tool. Deionized water is used to control the process and remove particles. Four discs were machined in total.



Milling Machine

As operators of the milling machine, we use it to craft the base part for our robotic arm, carving precise rectangle slots with its rotating cutter for assembly. The process involves securely mounting the workpiece, selecting the cutter, and guiding it along the intended path for accuracy. Safety measures, including protective gear and secure clamping, are prioritized to prevent accidents.



Vertical Drill

The vertical drilling machine, or drill press, is a precision tool used for drilling holes in various materials with superior accuracy, efficiency, and depth control compared to hand drilling. Its rigidity and fixed position ensure precise hole placement, enhancing productivity. While diesel is not commonly used as a cutting fluid for drilling aluminum, it can offer lubricating properties to reduce friction and heat. However, safety precautions are essential due to potential hazards such as fumes and fire risk.



Lathe Machine

We employed lathe machine to decrease the thickness of aluminium plates for our robotic arm. This versatile tool rotates the workpiece against a cutting tool, allowing us to shape cylindrical surfaces precisely. Diesel is used as a cutting fluid to lubricate the cutting tool and extend its lifespan by reducing friction and heat. While diesel can help maintain tool life, safety precautions such as proper ventilation are essential due to potential hazards like fumes and fire risks.



Cutting Machine

We use a cutting machine to cut aluminum plates for our robotic arm project. This machine employs rotating blades mounted on wheels to slice through the material, with water aiding in cooling and lubrication to prevent overheating and prolong blade life. While this method achieves accurate cuts efficiently, the cut edges' surface finish tends to be poor due to blade abrasion and water. To enhance surface quality, additional finishing processes like sanding are necessary. Despite this limitation, the cutting machine remains valuable for its efficiency and versatility in cutting aluminum and other materials.



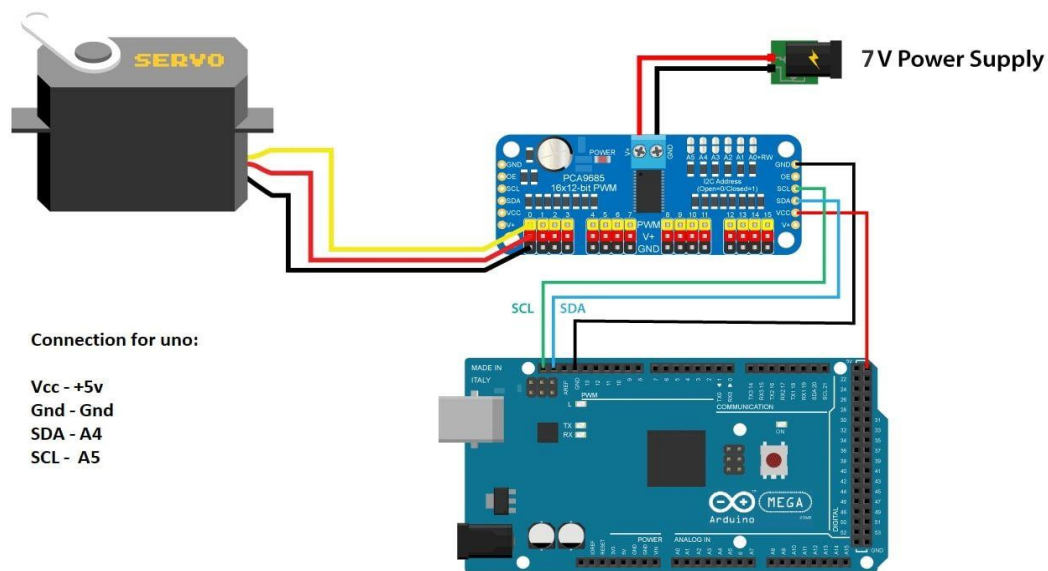
Electronics and Control System

The electronics and control systems serve as the backbone of the robotic arm, facilitating precise control and coordination of its movements. This section provides an overview of the integrated circuitry and control mechanisms employed in the project.

Integration of Arduino Microcontroller and Adafruit Servo Driver:

The Arduino microcontroller, in conjunction with the Adafruit servo driver, forms the core of the control system, enabling seamless communication and precise servo motor control.

Circuit Diagram:



The circuit diagram illustrates the interconnection of electronic components, including the Arduino microcontroller, Adafruit servo driver, power supply unit, and servo motors. Key elements of the circuitry include:

- Arduino Microcontroller
- Adafruit Servo Driver
- Servo Motors

Connections to the Arduino:

The PWM/Servo Driver uses I2C, so it takes 4 wires to connect to Arduino:

- +5v -> VCC
- GND -> GND
- Analog 4 -> SDA
- Analog 5 -> SCL

Working Principle:

The Arduino microcontroller communicates with the Adafruit servo driver via the I2C bus, sending commands to control the position and velocity of the servo motors. The Arduino orchestrates precise movements of the robotic arm by executing control algorithms and processing sensor feedback.

Interfacing and Communication:

The integration of electronic components follows established protocols and hardware interfaces, such as I2C communication and GPIO pins, to facilitate seamless data exchange and control signal generation.

Scalability and Expandability:

The modular design of the control system allows for scalability and expandability, accommodating future enhancements and additions to the robotic arm's functionality with minimal integration effort.

Kinematics

Mechanism Used: "Groove and Pin" or "Slot and Pin" Mechanism:

The "groove and pin" or "slot and pin" mechanism is commonly employed in engineering and mechanics to constrain motion to a single degree of freedom, typically rotation. In this setup, a pin is inserted into a slot or groove, restricting motion in all directions except rotation around the pin.

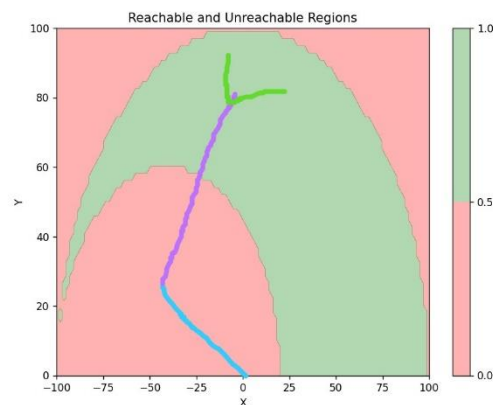
This mechanism is particularly useful in robotic arm design, where precise rotational motion control is essential. Incorporating the "groove and pin" mechanism into the robotic arm's joints prevents translational motion, ensuring smooth and accurate articulation of the arm segments.

Forward Kinematics:

Forward kinematics involves determining the position and orientation of a robotic arm's end-effector based on the joint angles.

For example, in a 3 Degree of Freedom (DOF) robotic arm with one yaw and two arms, forward kinematics involves calculating the end-effector's position and orientation using the angles of the three joints. The process typically involves defining Denavit-Hartenberg (DH) parameters, which describe the arm's geometry, including segment lengths and the angles between them. These parameters help

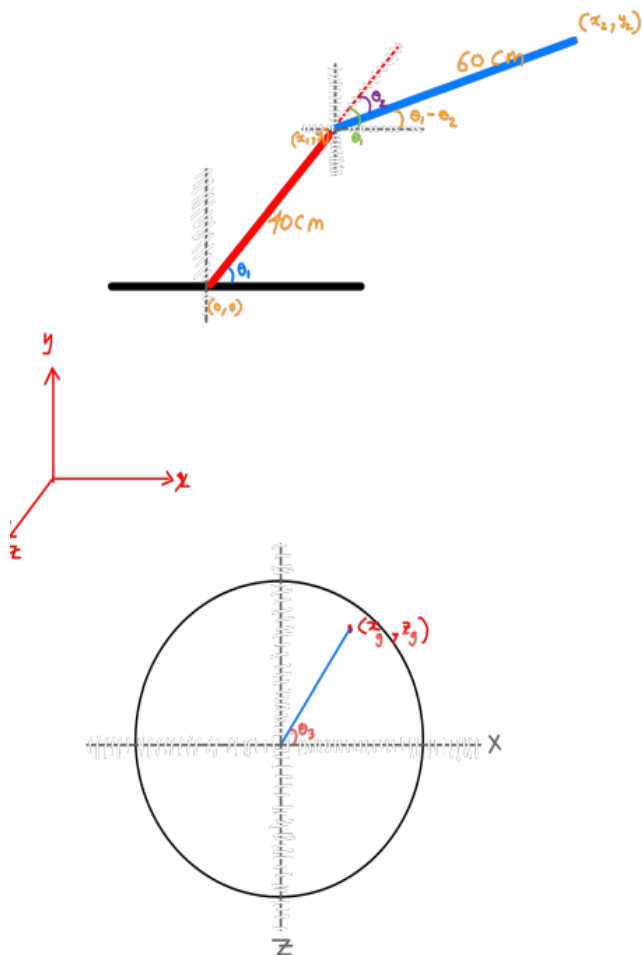
formulate transformation matrices representing the relationships between adjacent joints. The forward kinematics solution is obtained by multiplying these matrices and extracting the position (x, y, z coordinates) and the end-effector's orientation (Euler angles or quaternion) from the combined transformation matrix.



Inverse Kinematics:

Inverse kinematics is the process of finding the joint angles required for a robotic arm to reach a desired end-effector position and orientation. Different methods to solve inverse kinematics include analytical methods, which derive closed-form solutions based on mathematical equations; numerical methods, such as iterative algorithms like Jacobian-based approaches; geometric methods, which use geometric reasoning and constraints; and closed-loop control, which continuously adjusts joint angles based on sensor feedback until the desired pose is achieved.

In our project involving a 3 Degree of Freedom (DOF) robotic arm with 2 links and 1 yaw, we tackle the inverse kinematics using a hierarchy method. We begin by determining the yaw angle and then calculate the planar movement for the links. For this, we employ a numerical iterative method. Each link traverses a range of 180 degrees, and collectively, we select the best possible solution among the options available.



Forward kinematics

$$x_1 = 40 \cos \theta_1$$

$$y_1 = 40 \sin \theta_1$$

$$x_2 = 40 \cos \theta_1 + 60 \cos(\theta_1 - \theta_2)$$

$$y_2 = 40 \sin \theta_1 + 60 \sin(\theta_1 - \theta_2)$$

Inverse kinematics

$$\text{Goal} = (x_g, y_g)$$

For θ_1 in Range $(0, 180)$:

For θ_2 in Range $(0, 180)$:

$$x_2, y_2 = \text{Forward_kinematics}(\theta_1, \theta_2)$$

if (x_2, y_2) is closest to (x_g, y_g) :

$$\text{Inverse kinematics}(x_g, y_g) \Rightarrow \theta_1, \theta_2$$

$$\text{Yaw angle } (\theta_3) = \tan^{-1}\left(\frac{z_2}{x_2}\right)$$

$$x_2 = \sqrt{x_g^2 + z_g^2}$$

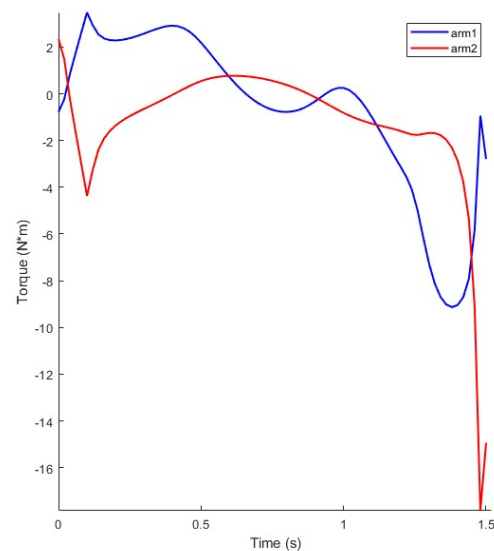
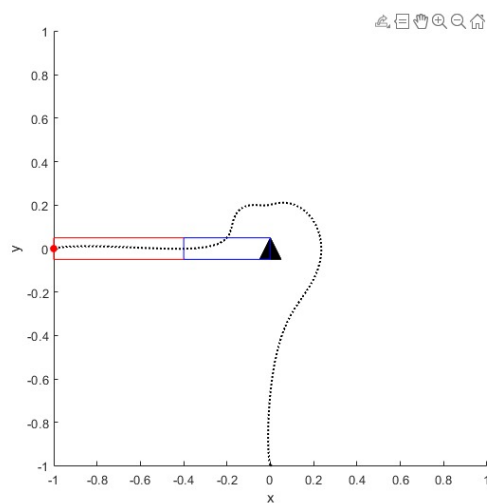
$$y_2 = y_g$$

Link Length Optimization:

In MATLAB, the `fmincon` function is a powerful tool for constrained optimization. It is particularly useful when dealing with problems where you must minimize (or maximize) a function while adhering to certain constraints. This capability aligns perfectly with the task of optimizing the link lengths of a robotic arm to reach a set of goal points while minimizing the clearance distance.

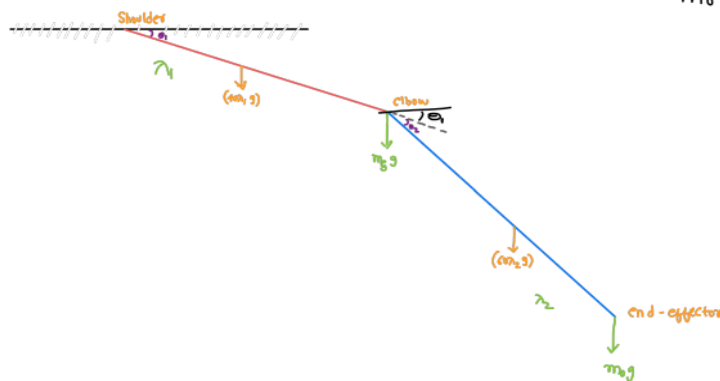
- **Objective Function:** Define the objective function as the sum of clearance distances to reach goal points.
- **Constraints:** Set constraints to ensure physical feasibility, like bounds on link lengths and joint limitations.
- **Optimization Setup:** Use `fmincon` to minimize the objective function while satisfying constraints. Provide initial link length guesses.
- **Run Optimization:** Call `fmincon` to adjust link lengths until the optimal solution iteratively, minimizing clearance distances.

This method efficiently tunes the robotic arm's geometry to enhance performance in reaching target points while maintaining safe distances from obstacles.



ANALYSIS

Free Body Diagram:



$\lambda_1 \rightarrow$ mass per unit length of arm 1

$\lambda_2 \rightarrow$ mass per unit length of arm 2

$m_s \rightarrow$ mass of servo motor

$m_o \rightarrow$ mass of obj at end-effector

$$\lambda_2 = \frac{200 \text{ gm}}{60 \text{ cm}} = 3.33 \times 10^{-3} \text{ kg/cm}$$

$$\lambda_1 = \frac{250 \text{ gm}}{40 \text{ cm}} = 6.25 \times 10^{-3} \text{ kg/cm}$$

$$m_s = 200 \text{ gm} \\ (\text{servo} + \text{bearing})$$

Moment on Servo at elbow:

$$M_2 = (60\lambda_2 g) \left(\frac{60}{2} \cos(\theta_1 + \theta_2) \right) + m_o g (60 \cos(\theta_1 + \theta_2))$$

$$M_2 = g \cos(\theta_1 + \theta_2) [1800\lambda_2 + 60m_o]$$

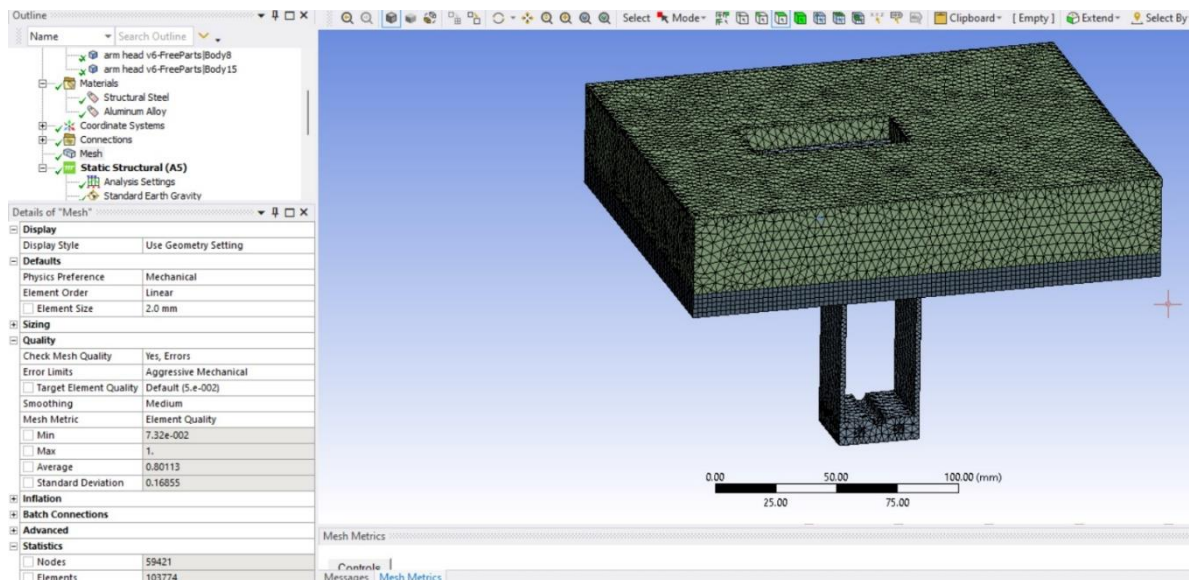
Moment on servo at Shoulder:

$$M_1 = (40\lambda_1 g) (20 \cos \theta_1) + m_s g (40 \cos \theta_1) + \\ (60\lambda_2 g) [40 \cos \theta_1 + 30 \cos(\theta_1 + \theta_2)] + \\ (m_o g) [40 \cos \theta_1 + 60 \cos(\theta_1 + \theta_2)]$$

This analysis determines that the arm can handle a maximum weight of 330 grams. Factoring in a safety margin, as the servo motor is rated for 60 kg cm but can handle up to 70 kg cm, the lifting capacity can be increased to 930 grams if two shoulder servo motors are utilized, providing a combined torque of 120 kg cm. Torque capacity is exceeded only at extreme position of arms which is outside our boundary region.

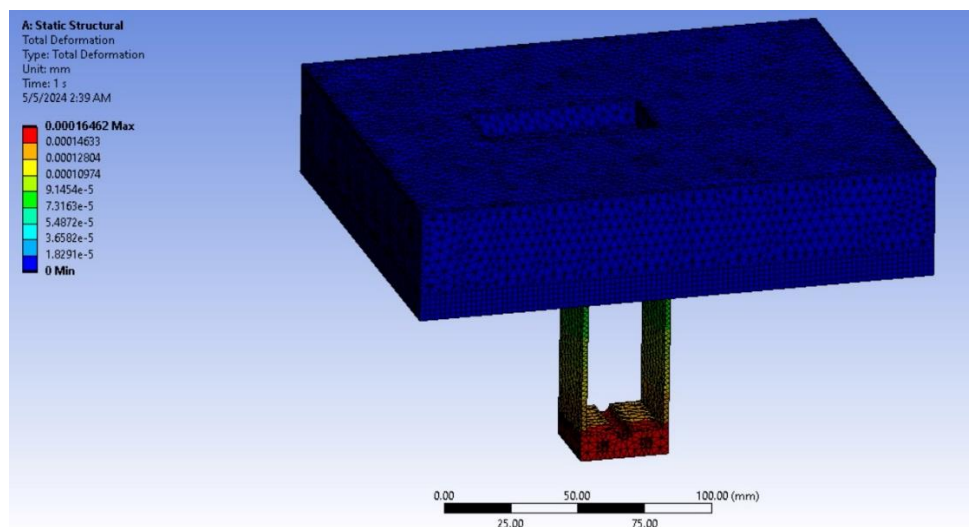
Mesh Analysis:

We simulated our structure using aluminum alloy with 2.7 g/cc density. We tried different mesh methods along with various element sizes to observe the trend of the number of elements and element quality. A global element size of 2 mm is ideal for modeling regarding computation efficiency and element quality, as we tried with 1.5 mm, which gave just 1% improvement in element quality, and total nodes increased to about 1 lakh, thus increasing the computation. We have used symmetry to reduce the number of nodes. Linear mechanical has been chosen as physics preference for depicting the linear properties of stress-strain.



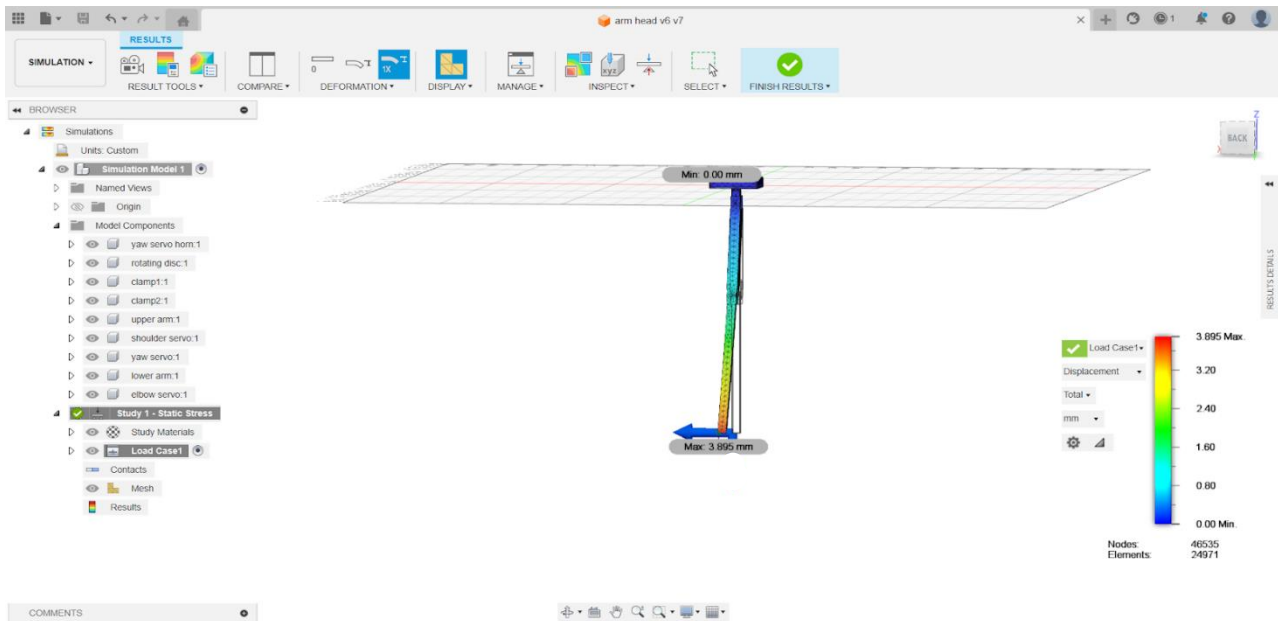
Boundary Conditions:

We used 60kg-cm moment and 5kg load on the disc for analysis as our servo can handle a maximum of this much torque, and the total mass of our links, servo, and payload is 2.5 kg. So, after considering a FOS 2, we used a 5kg load for analysis on our base disc.



Solutions and their importance:

We have chosen Von misses stress as it can consider multiaxial stresses and give us equivalent stress. Total deformation is our most important metric, as our objective is to maximize this metric.



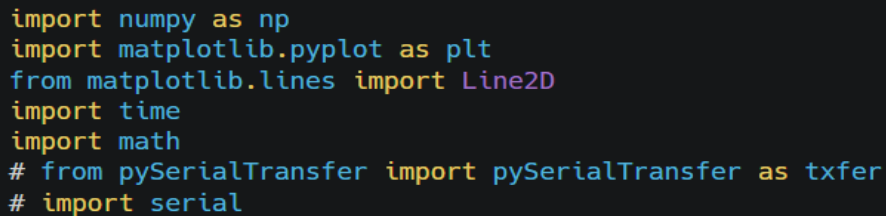
Programming

The programming aspect of the project involves developing the software to control and coordinate the robotic arm's movements.

Programming for the robotic arm involved Python and C languages. Implemented control algorithms, motor drivers, and sensor integration for precise movement and environmental interaction. The detailed code is discussed below:

1. Importing Libraries:

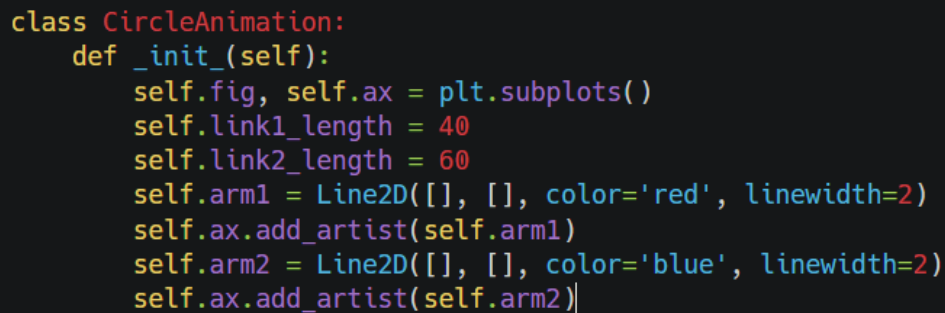
- The code imports necessary libraries for mathematical calculations, plotting graphs, handling time-related functions, and other functionalities.

A code editor window with a dark background and light green border. It contains Python code for importing libraries. The code is as follows:

```
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.lines import Line2D
import time
import math
# from pySerialTransfer import pySerialTransfer as txfer
# import serial
```

2. Class Definition - CircleAnimation:

- Defines the CircleAnimation class encapsulating functionality related to animating the robotic arm.
- Initializes the plot and sets up parameters such as the lengths of the robotic arm links (`link1_length` and `link2_length`).

A code editor window with a dark background and light green border. It contains the definition of the CircleAnimation class. The code is as follows:

```
class CircleAnimation:
    def __init__(self):
        self.fig, self.ax = plt.subplots()
        self.link1_length = 40
        self.link2_length = 60
        self.arm1 = Line2D([], [], color='red', linewidth=2)
        self.ax.add_artist(self.arm1)
        self.arm2 = Line2D([], [], color='blue', linewidth=2)
        self.ax.add_artist(self.arm2)|
```

3. Initializing Plot:

- Sets up the plot with Line2D objects representing the two links of the robotic arm.

4. Forward Kinematics Function (forward_kinematics):

- Calculates the forward kinematics of the robotic arm based on the provided joint angles.
- Computes the end-effector position (x, y) using trigonometric equations.

```
def forward_kinematics(self, joint_angles):
    x = self.link1_length * np.cos(joint_angles[0]) + self.link2_length *
np.cos(np.sum(joint_angles))
    y = self.link1_length * np.sin(joint_angles[0]) + self.link2_length *
np.sin(np.sum(joint_angles))
    return np.array([x, y])
```

5. Inverse Kinematics Function (inverse_kinematics):

- Computes the inverse kinematics of the robotic arm to determine the joint angles required to reach a specified target position.
- Iterates over a range of offsets to fine-tune the joint angles for accuracy.
- Returns the calculated joint angles as a numpy array.

```
def inverse_kinematics(self, target_position):
    x, y = target_position
    D = (x*2 + y*2 - self.link1_length*2 - self.link2_length*2) / (2 * self.link1_length *
self.link2_length)
    theta2 = np.arctan2(-np.sqrt(1 - D*2), D)
    theta1 = np.arctan2(y, x) - np.arctan2(self.link2_length * np.sin(theta2), self.link1_length +
self.link2_length * np.cos(theta2))
    dist = 100000
    for offset in range(50):
        theta2_check = np.arctan2(-np.sqrt(1 - D*2), D) + (offset(3.14/50))
        # print('D --> ', D)
        # print('theta 2 ---> ', theta2)
        theta1_check = np.arctan2(y, x) - np.arctan2(self.link2_length * np.sin(theta2_check),
self.link1_length + self.link2_length * np.cos(theta2_check))
        # print('theta 1 ---> ', theta1)
        reached = self.forward_kinematics([theta1_check, theta2_check])
        dist_check = math.sqrt((x-reached[0])*2 + (y-reached[1])*2)
        # print(dist_check)
        if dist_check < dist:
            dist = dist_check
            theta1 = theta1_check
            theta2 = theta2_check
    print(theta1, theta2+3.14)
    return np.array([theta1, theta2])
```

6. Angles Function:

- Defines a trajectory of target positions (trajectory) for the robotic arm to follow.
- The corresponding joint angles for each target position are calculated using the `inverse_kinematics` function and the results.

```
def angles(self):
    trajectory1 = [[0,50],[0,60],[0,70],[0,80]]
    for target in trajectory1:
        joint_angles = self.inverse_kinematics(target)
        print(joint_angles*57.3)
    ani = CircleAnimation()
    ani.angles()
```

7. Sending Angles to Arduino:

- To send the calculated joint angles to the Arduino, serial communication with the Arduino needs to be established.
- The code below indicates that serial communication was intended using libraries such as `pySerialTransfer` or `serial`.
- Typically, after calculating the joint angles, the code would establish a serial connection with the Arduino, encode them into a format suitable for transmission (e.g., bytes), and send them to the Arduino via the serial port.
- Upon receiving the joint angles, the Arduino would interpret them and control the robotic arm's actuators accordingly.

```
# Define the serial port and baud rate
serial_port = "COM6" # Replace with your actual serial port
baud_rate = 115200

# # Create a PySerialTransfer object
self.link = txfer.SerialTransfer(serial_port, baud_rate)

# # Open the serial port
self.link.open()
time.sleep(5)

self.ser = serial.Serial('COM6', 9600, timeout=1)
self.ser = serial.Serial('COM6', 115200, timeout=1)
```

CALIBRATION

Calibration of servo motors in the context of robotic arms involves aligning the robotic arm's physical zero position with the servo motors' zero position and determining the pulse widths corresponding to the minimum and maximum rotation angles.

Here's a step-by-step explanation of the calibration process:

- **Aligning Physical Zero Position:**

The physical zero position of the robotic arm is the reference point where all joint angles are considered zero.

The servo motors have their own zero position, typically corresponding to the centre of their range (e.g., 90 degrees for a 0-180 degree range).

To align the robotic arm's physical zero position with the servo motors' zero position, you manually position the robotic arm to match the zero position of the servo motors.

- **Determining Maximum and Minimum Angles:**

Identify the maximum and minimum rotation angles for each robotic arm joint. These angles depend on the mechanical design of the arm.

Move the robotic arm to its maximum and minimum limits for each joint while monitoring the corresponding angles reported by the servo motors or any encoders/position sensors attached.

Note down the angles corresponding to each joint's maximum and minimum positions.

- **Calculating Pulse Widths:**

Servo motors are typically controlled using pulse-width modulation (PWM) signals. The pulse width determines the position of the servo motor shaft.

Measure the pulse widths corresponding to the physical zero position and each joint's maximum and minimum rotation angles.

These pulse widths will serve as reference values for commanding the servo motors to specific positions.

- **Mapping Pulse Widths to Angles:**

Once you have the pulse widths corresponding to the zero, maximum, and minimum angles for each joint, you can create a mapping between pulse widths and joint angles.

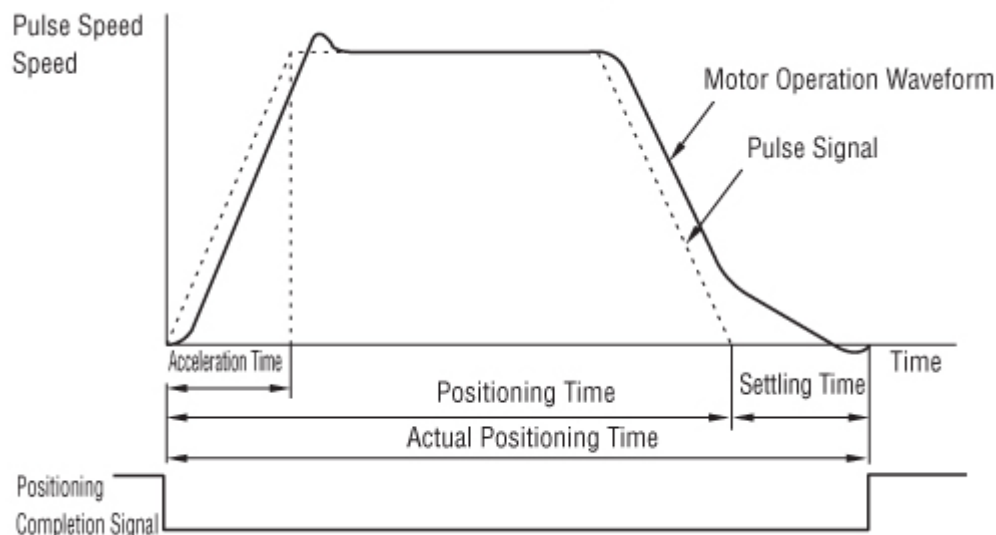
This mapping allows you to command the servo motors to move the robotic arm to desired positions by sending appropriate PWM signals.

- **Fine-tuning and Verification:**

After setting up the initial calibration, you should fine-tune the mapping and check for discrepancies. Verify that the servo motors accurately move the robotic arm to the desired positions based on the commanded pulse widths.

Adjust the calibration parameters to ensure precise control over the robotic arm.

Calibration ensures that the servo motors accurately represent the positions of the robotic arm joints, allowing for precise control and movement. It establishes a reliable relationship between the control signals (pulse widths) and the physical positions of the arm, enabling effective operation in various tasks and applications.



Servo Motor

Servo motors are commonly used in robotics and automation to precisely control mechanisms' position, velocity, and acceleration. They comprise a DC motor, a gear train, and a feedback control system.

DC Motor: The core of a servo motor is a DC motor. When electric current is applied to the motor, it rotates.

Gear Train: Servo motors often have a gear train that reduces the speed of the motor while increasing its torque output. This gearing allows servo motors to provide high torque at low speeds, making them suitable for applications requiring precise control.

Feedback Control System: One of the distinctive features of servo motors is their built-in feedback control system, usually based on a potentiometer or an encoder. This system continuously monitors the position of the motor shaft and compares it to the desired position. If there's a difference between the actual and desired positions, the control system adjusts the motor's power input to minimize the error, bringing the motor shaft to the desired position.

Specifications of a 60 kg-cm Servo Motor with 0 to 180 Degree Angle Limits:

Torque Rating: 60 kg-cm refers to the servo motor's maximum torque. This specification indicates the amount of force the motor can apply at a given distance from the axis of rotation. In this case, the motor can exert a force equivalent to lifting a 60 kg weight with a lever arm of 1 cm.

Angle Limits: The angle limits specify the range over which the servo motor can rotate its shaft. In this case, the servo motor can rotate from 0 to 180 degrees. This means it can cover a half-circle rotation.

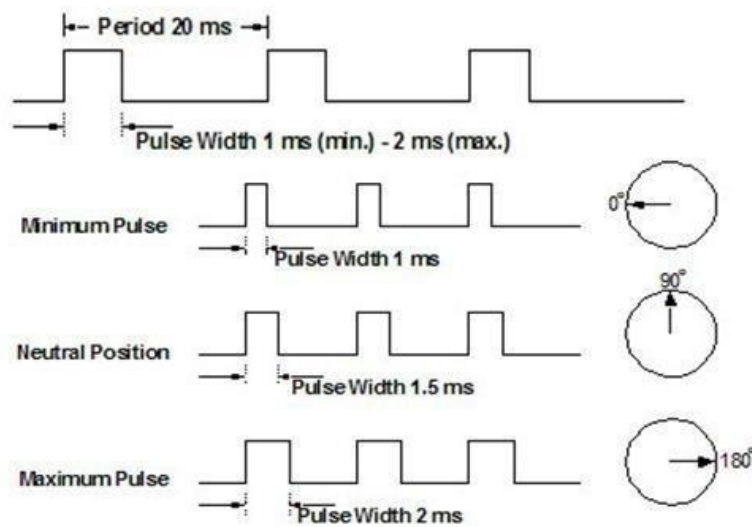
Time Pulse Graph:

The time pulse graph represents the relationship between the input signal (usually a pulse-width modulation, PWM, signal) and the resulting position of the servo motor shaft. It shows how the servo motor responds to different pulse widths over time.

X-axis: Represents time.

Y-axis: Represents the position of the servo motor shaft.

The graph typically shows that longer pulse widths correspond to larger rotation angles, allowing you to visualize how the servo motor responds to different control signals. This graph is crucial for understanding and fine-tuning the control of the servo motor to achieve desired movements accurately.



CONCLUSION

We have completed the fabrication of the arm. The base along with the arms is ready and attached to the servo motors, thus allowing the links to rotate.

Aluminum was selected for fabricating the arms due to its lightweight properties and cost-effectiveness. Wood was ruled out because impressions formed in it can induce vibrations, causing the end effector to deviate from its intended kinematic position. After evaluating the force of a 5 kg load and a maximum torque of 60 kg cm on the rotating disc, which is the motor's limit, we opted for a disc with a diameter of 140 mm and a thickness of 5 mm.

Utilizing fmincon optimization in Matlab, we determined the optimal link lengths to be 40 cm and 60 cm, respectively. By employing an Arduino and a servo driver, we successfully synchronized the movement of all servos. Our project effectively showcased the potential of leveraging advanced technologies for robotic arm applications.



DIFFICULTIES FACED

- Finding suitable aluminum grade for base and arms: 700mm x 200mm x 12mm pieces not readily available.
- Careful handling of wire in EDM machine crucial to prevent breakage during manufacturing.
- Difficulty in clamping workpiece on EDM machine due to its characteristics.
- Aluminum tends to melt and adhere to blade surface during cutting.
- Aligning arms with center of rotating disc challenging during assembly.
- Initial assumption of welding base parts changed to using nuts and bolts due to lack of sonic welding machine, resulting in vibrations.

FUTURE WORK PLAN

We aim to introduce a voice-controlled system for recognizing and interacting with a set of colored boxes, leveraging advanced natural language processing and artificial intelligence technologies. The primary goal is to empower users to issue voice commands to select specific coloured boxes based on their characteristics, utilising a robotic arm for physical interaction.

The system would initiate by employing a voice recognition module to interpret user commands. For instance, commands like "Pick the red-coloured box" or "Select the uniquely shaped box" trigger the system to identify the relevant box based on predefined criteria. Machine learning algorithms can be employed to classify and recognise the specified characteristics of each box.

Upon successful recognition, the system would convert the chosen box's identification into speech text using natural language processing techniques. This text would then be transmitted to the ChatGPT API, generating a response indicating the appropriate action. For example, if the user commands the system to pick a red box, the API-generated response might be, "Select the red box located at coordinates X, Y, Z."

Finally, the system would utilise the spatial coordinates (X, Y, Z) received from the ChatGPT API to guide a robotic arm into physically picking or interacting with the identified box. The robotic arm could be programmed to execute precise movements for efficient manipulation.

REFERENCES

- Matlab: <https://www.mathworks.com/help/optim/ug/fmincon.html>
 - Servo Driver: <https://learn.adafruit.com/16-channel-pwm-servo-driver/hooking-it-up>
 - Servo Motor Datasheet: <https://m.media-amazon.com/images/I/81EFGw8qkhL.pdf>
 - Aluminium Grade Datasheet: <https://www.aalco.co.uk/datasheets/?gId=1>
-