Pick and Place Robotic Arm

August, 2023 to December, 2023

Aditya Mehta
Chemical Engineering
22110150

Parth Govale
Computer science Engineering
22110087

Pranjal Gaur
Computer science Engineering
22110201

Farhan Obaid
Mechanical Engineering
22110081

***Abstract*—This project report outlines the design and development of a pick-and-place robotic arm. Constructed using ABS black through 3D printing, the arm integrates theoretical models, numerical methods, and an Arduino Mega for control. The report covers material selection, fabrication, and workspace analysis. Theoretical foundations, implemented in Python, include forward and inverse kinematics with the Jacobian matrix. Simulation using Python libraries validates the theoretical models. The collaborative efforts of the team, along with guidance from the faculty advisor and teaching assistants, have been essential for the successful completion of the project.**

1. Introduction

In the contemporary landscape of technological innovation, the development of automated systems has become integral to enhancing efficiency across various industries. Our project, focused on the design and implementation of a pick-and-place robotic arm, embodies the convergence of cutting-edge engineering principles and real-world application.

The pick-and-place robotic arm serves as a testament to the potential of robotics to revolutionise manufacturing, logistics, and other sectors where repetitive tasks can be optimised for speed, accuracy, and reliability. As industries evolve, so do the challenges they face. The versatility and adaptability of our robotic arm make it a viable solution for tackling complex tasks across a spectrum of applications. By delving into the intricate details of its design, fabrication, and theoretical underpinnings, this report aims to provide a comprehensive understanding of the project's scope and significance.

The primary objective of this project is to design a robotic arm capable of performing intricate pick-and-place operations. It aims to provide a detailed overview, covering various crucial aspects such as the project's purpose, its significance, and the intricacies involved in its design. Furthermore, this report delves into the development process, shedding light on the various challenges encountered along the way.

2. Application of a Robotic Arm

The Pick and Place Robotic Arm has a wide range of applications in various industries and fields. Some of the industries and fields where the robotic arm can be used are:

2.1. Manufacturing

The robotic arm can be used in manufacturing processes to pick and place products on assembly lines, increasing efficiency and reducing labour costs. These robotic systems excel in handling repetitive and intricate tasks such as assembly, sorting, and packaging. By automating these processes, manufacturers can enhance production speed, reduce errors, and optimise resource utilisation.

2.2. Warehouse and Logistics

The robotic arm can be used in warehousing and logistics to move and stack boxes, reducing the risk of injury to workers and increasing productivity. The ability to swiftly and accurately move objects from one location to another streamlines logistics operations, resulting in increased efficiency and reduced operational costs.

2.3. Healthcare

Precision and repeatability are critical in healthcare settings, and the pick-and-place robotic arm finds applications in tasks like medical instrument handling, sample sorting, and pharmaceutical packaging. Its ability to operate in controlled environments makes it a valuable tool for enhancing safety and accuracy in healthcare processes.

2.4. Food and Beverage

The robotic arm can be used in food and beverage industries to sort and package products, ensuring consistent quality and reducing waste. The robotic arm can be employed for tasks such as sorting, packaging, and palletising.

3. Concepts and Principles Used

The Pick and Place Robotic Arm is designed and operated using a combination of mechanical, electrical, and control concepts and principles. These concepts and principles work together to ensure that the robotic arm operates efficiently and effectively, allowing it to perform its intended tasks with precision and accuracy.

3.1. Mechanical Concepts and Principles

The mechanical concepts and principles used in the Pick and Place Robotic Arm relate to the design and construction of the physical components of the arm. These components include the joints, links, and end effectors, which work together to enable the arm to move and manipulate objects.

3.1.1. Kinematics. Kinematics is at the heart of the robotic arm's motion. It involves the study of the arm's geometry, motion, and spatial positioning without considering the forces that cause the motion. Our project employs forward kinematics to determine the end-effector's position based on the joint angles and inverse kinematics to compute the required joint angles for a desired end-effector position.

3.1.2. Dynamics. Dynamics deals with the forces and torques that cause motion. Understanding the dynamics of the robotic arm is crucial for optimising its performance and ensuring structural integrity.

3.2. Electrical Concepts and Principles

3.2.1. Micro - Controller. The Arduino Mega serves as the central processing unit, executing logic that coordinates the operation of the entire robotic system. Understanding digital signals and algorithm implementation are essential principles guiding the development of the control logic. This ensures efficient communication and coordination among various components.

3.2.2. Power Supply. The foundation of our electrical system lies in a robust power supply. Understanding voltage requirements and a well-regulated power supply is necessary for the functioning of the servo motors and the control system.

3.3. Programming and Algorithms

The programming of the robotic arm involves the development of algorithms that dictate its behaviour. The programming logic on the Arduino Mega revolves around numerical methods for servo motor control. Utilising iterative algorithms, the microcontroller calculates the step sizes and sequences required for desired joint movements. This approach facilitates control, allowing the robotic arm to move with precision in step sizes of angles.

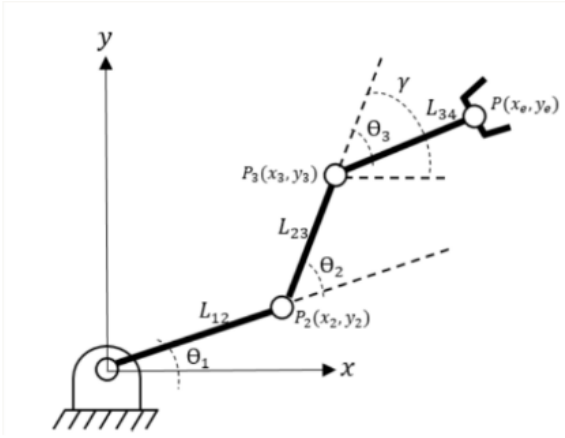
3.4. Control Systems

Feedback control is a pivotal aspect of our robotic arm's operation, involving continuous adjustments to the control inputs for precise end-effector positioning. Implemented on the Arduino Mega, this control mechanism dynamically calculates the joint angles necessary to place the end-effector at the desired position in every step. This iterative refinement involves adjusting the control inputs, i.e. the end-effector coordinates in every step, which in turn calculates the joint angles and updates the servo motors accordingly to place the end-effector at the desired position.

4. Mathematical Calculations

4.1. Forward Kinematics

Forward Kinematics describes the relationship between the joint angles of the robotic arm and the position and orientation of the end effector. This model is essential for predicting the movement of the arm and ensuring that it can reach the desired target locations. In our pick-and-place robotic arm, the forward kinematics model involves the transformation of joint angles into Cartesian coordinates. Using trigonometry, we mapped the joint angles with end-effector coordinates.

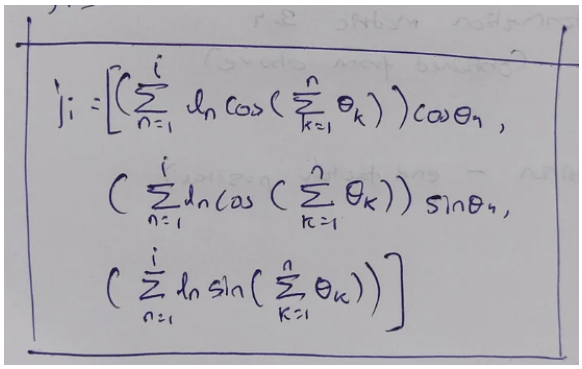


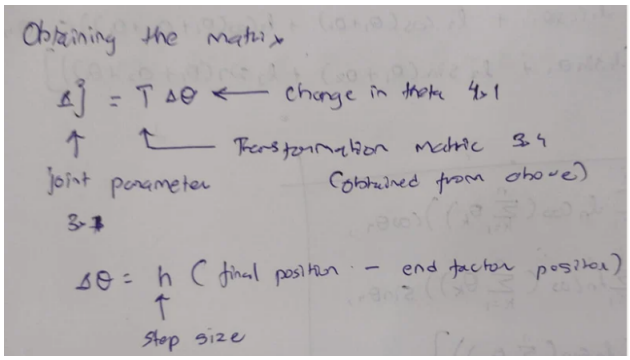
∴ Forward Kinematics,

$$j_1 = \left[l_1 \cos\theta_1 \cos\theta_4 \;,\; l_1 \cos\theta_1 \sin\theta_4 \;,\; l_1 \sin\theta_1 \right]$$

$$j_2 = \left[(l_1 \cos\theta_1 + l_2 \cos(\theta_1 + \theta_2)) \cos\theta_4 \;,\; (l_1 \cos\theta_1 + l_2 \cos(\theta_1 + \theta_2)) \sin\theta_4 \;,\; (l_1 \sin\theta_1 + l_2 \sin(\theta_1 + \theta_2)) \right]$$

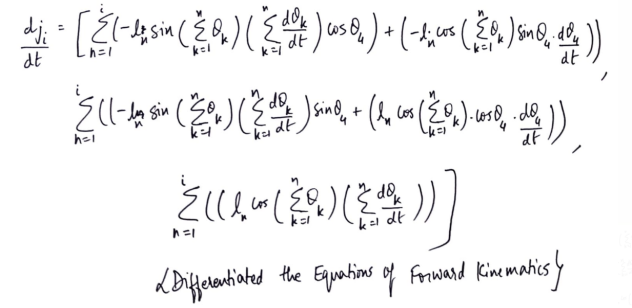
$$j_3 = \left[(l_1 \cos\theta_1 + l_2 \cos(\theta_1 + \theta_2) + l_3 \cos(\theta_1 + \theta_2 + \theta_3)) \cos\theta_4 \;,\; (l_1 \cos\theta_1 + l_2 \cos(\theta_1 + \theta_2) + l_3 \cos(\theta_1 + \theta_2 + \theta_3)) \sin\theta_4 \;,\; (l_1 \sin\theta_1 + l_2 \sin(\theta_1 + \theta_2) + l_3 \sin(\theta_1 + \theta_2 + \theta_3)) \right]$$





4.2. Inverse Kinematics

Inverse Kinematics describes the relationship between the position and orientation of the end effector and the joint angles of the robotic arm. This model is essential for determining the joint angles required to position the end-effector at a desired location in the workspace. Our approach involves solving the inverse kinematics problem through geometric and algebraic methods. Trigonometric relationships, geometric considerations and differential calculus are employed to derive the joint angles.



4.3. Jacobian Matrix

The Jacobian matrix serves as a bridge between joint velocities and end-effector velocities, providing insights into the manipulability and sensitivity of the robotic arm. In our analysis, the Jacobian matrix is derived by partial differentiation of the forward kinematics equations with respect to joint angles. The matrix allows us to relate changes in joint angles to changes in end-effector position and orientation. This relationship is instrumental for control strategies, as it enables the determination of optimal joint velocities to achieve desired end-effector velocities.

4.4. Torque Required Calculations

The actuator applies torque at the joint to overcome the resistance of the link to motion. Resistance of any link to motion is due to gravity and inertia effects.The effect of gravity on any link of the robot is to pull and accelerate it towards the center of the earth due to its own weight, thus exerting a resistive torque on it. Therefore, a portion of the torque generated by the actuator is needed to overcome this resistive torque (due to gravity). It was necessary to calculate the value of the gravity- induced resistive torque acting on each link of the arm so that an actuator with sufficient torque rating could be selected for each joint.

Intuitively the torque on the shoulder joint is much greater when the arm is stretched out horizontally. Thus, in order to calculate the torque required at each joint, the worst possibility case (arm completely stretched out) was chosen. The arm is subjected to the highest torque when the arm is stretched horizontally.

4.5. Power Required Calculations

After all the calculations the required came out to be 62.716 W with the voltage rating of 6 - 7 volts and current of 8 amperes.

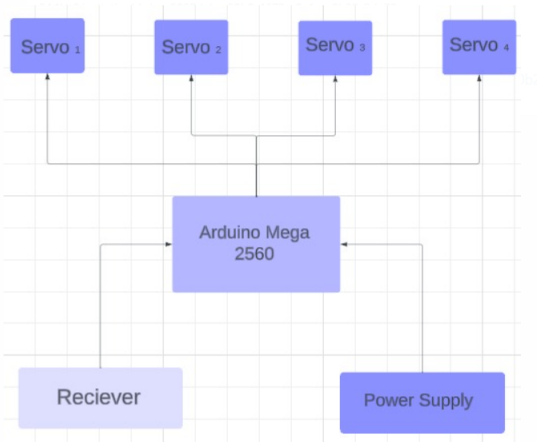
5. Code

```
#include <Servo.h>

Servo servo01;
Servo servo02;
Servo servo03;
Servo servo04;
Servo servo05;
Servo servo06;

void setup() {
  Serial.begin(9600); // Initialize
     serial communication
  // Attach servo pins
  servo01.attach(5);
  servo02.attach(6);
  servo03.attach(7);
```

```
  servo04.attach(8);
  servo05.attach(9);
  servo06.attach(10);
  int speed = 5;

}

void loop() {
    int angles[6]; // Array to store the
        angles
    // Move each servo to the
       corresponding angle
    angles[0];
    for (int i = 0; i < angles[0]+1; i+=
       speed) {
        servo01.write(angles[0]);
    }
    angles[1];
    for (int i = 0; i < angles[1]+1; i+=
       speed) {
        servo01.write(angles[1]);
    }
    angles[2];
    for (int i = 0; i < angles[2]+1; i+=
       speed) {
        servo01.write(angles[2]);
    }
    angles[3];
    for (int i = 0; i < angles[3]+1; i+=
       speed) {
        servo01.write(angles[3]);
    }
    angles[4];
    for (int i = 0; i < angles[4]+1; i+=
       speed) {
        servo01.write(angles[4]);
    }
    angles[5];
    for (int i = 0; i < angles[5]+1; i+=
       speed) {
        servo01.write(angles[5]);
    }

  }
}
```

6. Block Diagram

The block diagram for this circuit would look like:

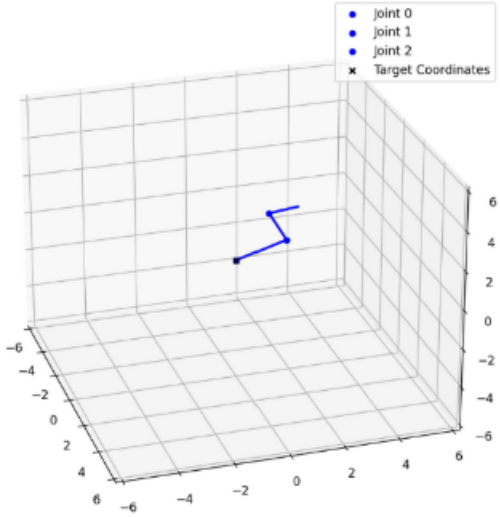


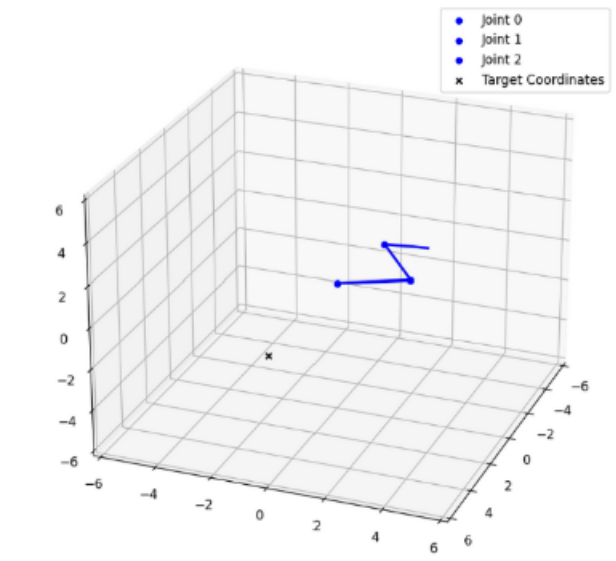
7. Simulation of the Arm

The simulation process involves utilising advanced software tools to model and visualise the operational envelope of the robotic arm in a virtual environment. The workspace simulation of our pick-and-place robotic arm is conducted using Python, leveraging libraries such as NumPy and Matplotlib.

The equations governing forward and inverse kinematics, along with the Jacobian Matrix, are translated into Python functions. This conversion facilitates the integration of theoretical models into the simulation. Matplotlib, a versatile plotting library in Python, is employed to visualise the simulated workspace. The 3D plotting capabilities of Matplotlib enable the representation of the robotic arm's motion in Cartesian space.

This figure shows us, how the torque can be calculated:





7.1. Code for Simulation

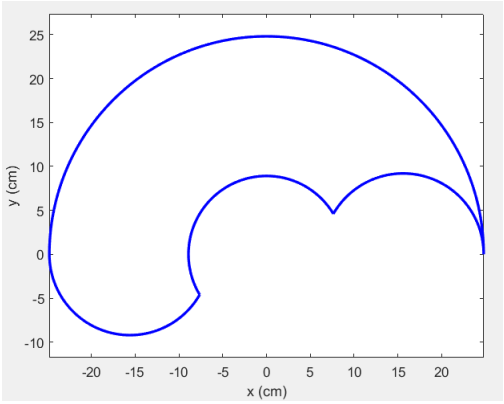
```
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.animation import
    FuncAnimation

# Forward Kinematics
def forward_kinematics(thetas):
    for i in range(1, joints.shape[0]):
        joints[i] = joints[i-1] + np.
            array([lengths[i-1]*np.cos(np
            .sum(thetas[:i]))*np.cos(
            thetas[-1]), lengths[i-1]*np.
            cos(np.sum(thetas[:i]))*np.
            sin(thetas[-1]), lengths[i
            -1]*np.sin(np.sum(thetas[:i])
            )])

# Inverse Kinematics
def inverse_kinematics(coordinates, phi)
    :
    thetas[-1] = np.arctan2(coordinates
        [1], coordinates[0])
    theta4_phi = np.array([np.cos(phi)*
        np.cos(thetas[-1]), np.cos(phi)*
        np.sin(thetas[-1]), np.sin(phi)])
    w = coordinates - lengths[2]*
        theta4_phi
    thetas[1] = np.arccos((np.sum(w**2)
        - np.sum(lengths[-2:]**2))/(2*np.
        prod(lengths[-2:])))
    x2 = np.sqrt(np.sum(w[:2]**2))
    c1 = (lengths[0] + lengths[1]*np.cos
        (thetas[1]))*x2 + lengths[1]*np.
        sin(thetas[1])*w[2]
    s1 = (lengths[0] + lengths[1]*np.cos
        (thetas[1]))*w[2] + lengths[1]*np
        .sin(thetas[1])*x2

    thetas[0] = np.arctan2(s1, c1)

    thetas[2] = phi - thetas[0] - thetas
        [1]

# Initailize required arrays and
    variables
lengths = np.array([2.5, 2.0, 1.5])
thetas = np.array([0.0, 0.0, 0.0, 0.0])
joints = np.zeros(shape=(4, 3))
step_size, space_limit = 0.01, 6.0

# Initialize using given conditions
initial = np.array(list(map(int, input("
    Enter initial coordinates: ").split()
    )))
target = np.array(list(map(int, input("
    Enter target coordinates: ").split())
    ))
phi = int(input("Angle of picking: "))*
    np.pi/180

# Adjustments according to initial
    conditions
inverse_kinematics(initial, phi)
thetas = np.array([0.0, 90.0, 180.0,
    90.0])
forward_kinematics(thetas)

print(thetas)

fig = plt.figure(figsize=(8, 8))

ax = fig.add_subplot(111, projection='3d
    ')
ax.set_xlim(-space_limit, space_limit)
ax.set_ylim(-space_limit, space_limit)
ax.set_zlim(-space_limit, space_limit)

# '''
def init():
    for i in range(joints.shape[0]-1):
        ax.plot([joints[i,0], joints[i
            +1,0]], [joints[i,1], joints[
            i+1,1]], [joints[i,2], joints
            [i+1,2]], 'b-', lw=2)

    for i in range(joints.shape[0]-1):
        ax.scatter(joints[i,0], joints[i
            ,1], joints[i,2], c='b',
            marker='o', label=f'Joint {i}
            ')
```

```
    ax.scatter(target[0], target[1],
        target[2], c='k', marker='x',
        label='Target Coordinates')

    ax.legend()
    return ax

def animate(i):
    global thetas

    dtarget = step_size*(target - joints
        [-1])

    inverse_kinematics(joints[-1] +
        dtarget, phi)
    forward_kinematics(thetas)

    ax.cla()
    ax.set_xlim(-space_limit,
        space_limit)
    ax.set_ylim(-space_limit,
        space_limit)
    ax.set_zlim(-space_limit,
        space_limit)
    init()

# ani = FuncAnimation(fig, animate,
    init_func=init, frames=200, interval
    =50)

# ax.set_xlabel('X')
# ax.set_ylabel('Y')
# ax.set_zlabel('Z')
# ax.set_title('3R Manipulator Animation
    in 3D')
# ax.grid()

init()
plt.show()

# '''
```

8. Workspace

The workspace of a robotic arm defines the region in which the end-effector can reach and operate. Analysing the workspace is crucial for understanding the limitations and capabilities of the pick-and-place robotic arm in performing tasks within its designated environment. Simulation tools are employed to visualise and validate the theoretical workspace analysis.

The Workspace for the 2R Manipulator will be shown as:



8.1. Code for Workspace

```
N = 1000;

% link lengths
l1 = 15.6;
l2 = 9.2;

% joint angle limitations
theta1_min = 0;
theta1_max = 180;
theta2_min = 0;
theta2_max = 150;

% limitations on theta1
theta1_start_end = [theta1_min,
    theta1_max];

% change the angle to Radians
theta1_start_end = theta1_start_end*pi
    /180;

% limitations on theta2
theta2_start_end =[theta2_min,theta2_max
    ];

% change the angle to Radians
theta2_start_end = theta2_start_end*pi
    /180;

% joint angles
theta1 = linspace(theta1_min,theta1_max,
    N);

% change the angle to Radians
theta1 = theta1*pi/180;
theta2 = linspace(theta2_min,theta2_max,
    N);

% change the angle to Radians
theta2 = theta2*pi/180;

%intialization of the x and y
```

```
x = zeros(2*length(theta1_start_end),
    length(theta2));
y = zeros(2*length(theta1_start_end),
    length(theta2));

% x and y are calculated using
    kinematics
for i = 1:2
    for j = 1:length(theta1)
        x(i,j) = l1*cos(theta1(j)) + l2*
            cos(theta1(j) +
            theta2_start_end(i));
        y(i,j) = l1*sin(theta1(j)) + l2*
            sin(theta1(j) +
            theta2_start_end(i));
    end
    for k = 1:length(theta1)
         x(i+2,k) = l1*cos(
             theta1_start_end(i)) + l2*
             cos(theta1_start_end(i) +
             theta2(k));
         y(i+2,k) = l1*sin(
             theta1_start_end(i)) + l2*
             sin(theta1_start_end(i) +
             theta2(k));
    end
end

x = x';
y = y';

%plotting
plot(x(:,1),y(:,1),'b','LineWidth',2)
hold on
plot(x(:,2),y(:,2),'b','LineWidth',2)
plot(x(:,3),y(:,3),'b','LineWidth',2)
plot(x(:,4),y(:,4),'b','LineWidth',2)

xlabel('x (cm)')
ylabel('y (cm)')
axis equal
```

8.2. What made the workspace better?

9. Choice of Material and Specifications

The selection of materials for our pick-and-place robotic arm is a critical aspect of ensuring durability, functionality, and efficiency.

9.1. Micro Controller

The microcontroller is the brain of the robotic arm. It is responsible for controlling the movement of the arm and its end effector. The microcontroller used in the Pick and Place Robotic Arm is an **Arduino Mega 2560 board**. It is programmed using the Arduino IDE. The Arduino Mega 2560 is a microcontroller board based on the ATmega2560 (datasheet). It has 54 digital input/output pins (of which 14 can be used as PWM outputs), 16 analogue inputs, 4 UARTs (hardware serial ports), a 16 MHz crystal oscillator, a USB connection, a power jack, an ICSP header, and a reset button.

9.2. Servo motors

Servo Motors of the following specifications are used in the arm:

1) Motor-1: Voltage = 7.4V, Current = 3.8V, Weight = 70g, Power = 28.12W

2) Motor-2: Voltage = 7.4V, Current = 3.5A, Weight = 72g, Power =25.9W

3) Motor-3: Voltage = 6V, Current = 1A, Weight = 56g, Power =6W

4) Motor-4: Voltage = 4.8V, Current = 0.27A, Weight = 14g, Power =1.29W

9.3. Power supply

For optimal operation, a 65W Switched Mode Power Supply (SMPS) unit with a voltage rating of 12V has been determined as sufficient to drive all the motors, facilitating the controlled movement of the robotic arm. This SMPS unit provides the necessary power for the motors' efficient performance. However, alternative power sources with slightly higher power ratings can also be used.

10. Bill of Materials

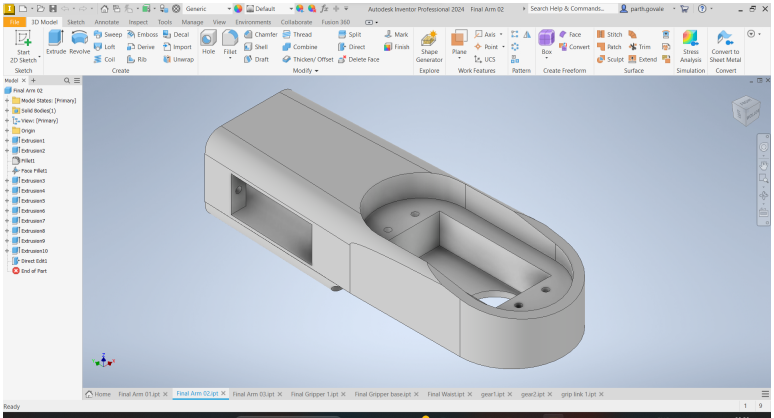
After the discussion on the specifications of the motors and materials used, now it was the time to order them.

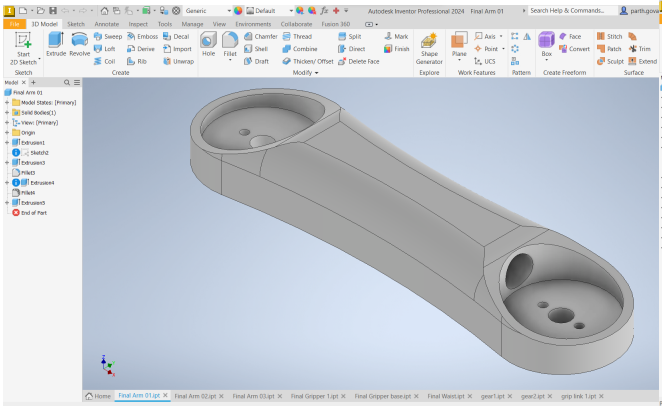
So, a bill of material was made.
This is my link: Bill-Link.

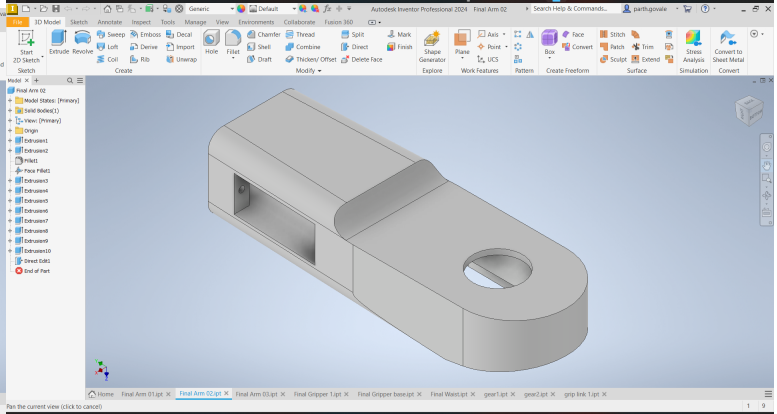
11. Designing the parts

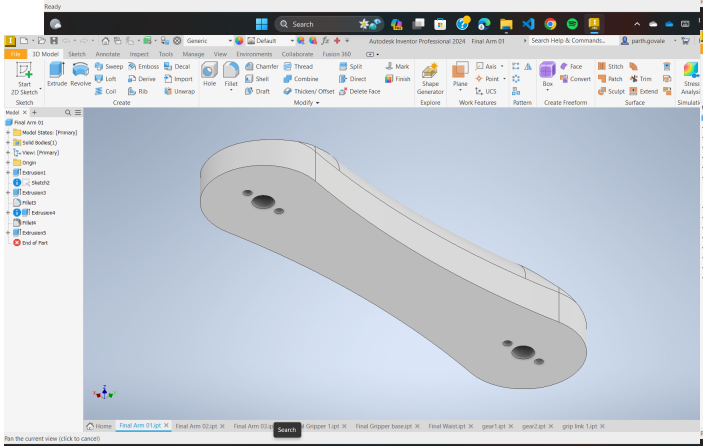
The design of the 3R manipulator took into consideration several factors, including the desired range of motion, payload capacity, and overall size and weight of the arm. The arm is required to lift a weight of 100 grams. The design of our pick-and-place robotic arm is a meticulous process undertaken with precision and innovation. We crafted each component to ensure structural integrity, functionality, and optimal performance. We used Autodesk Inventor, a cutting-edge design software. The design of each part takes into account the assembly process, ensuring that components fit seamlessly together. The components included the three arm segments, which are fundamental to the structural framework of the robotic arm, the waist, gripper, gripper base, grip link and gears.

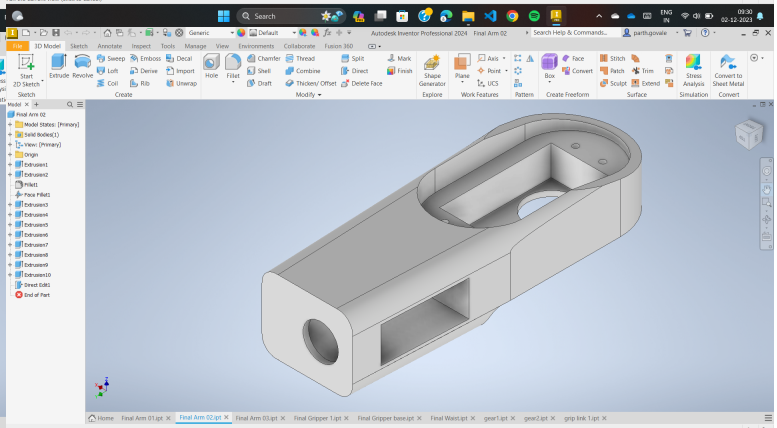
All the images of the parts are as follows:

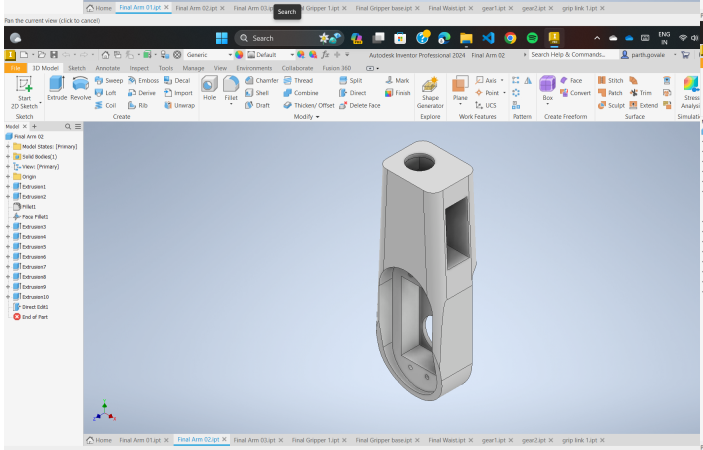


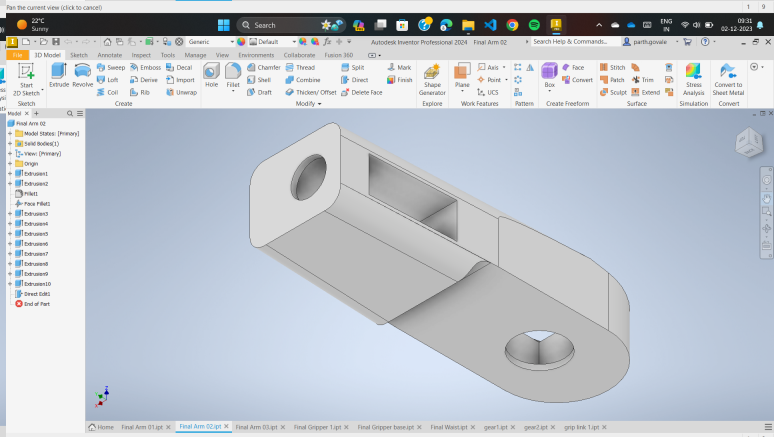


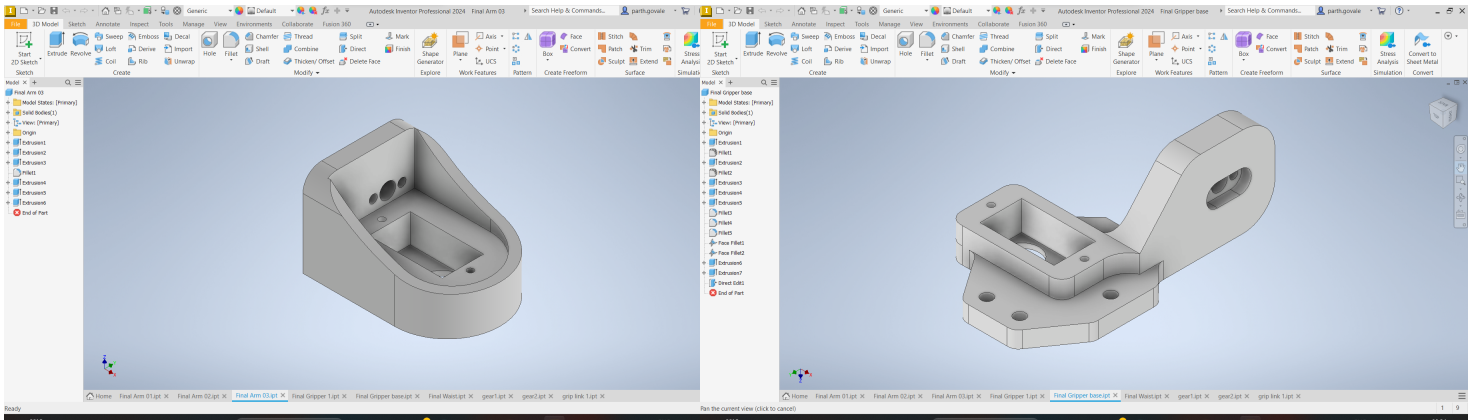


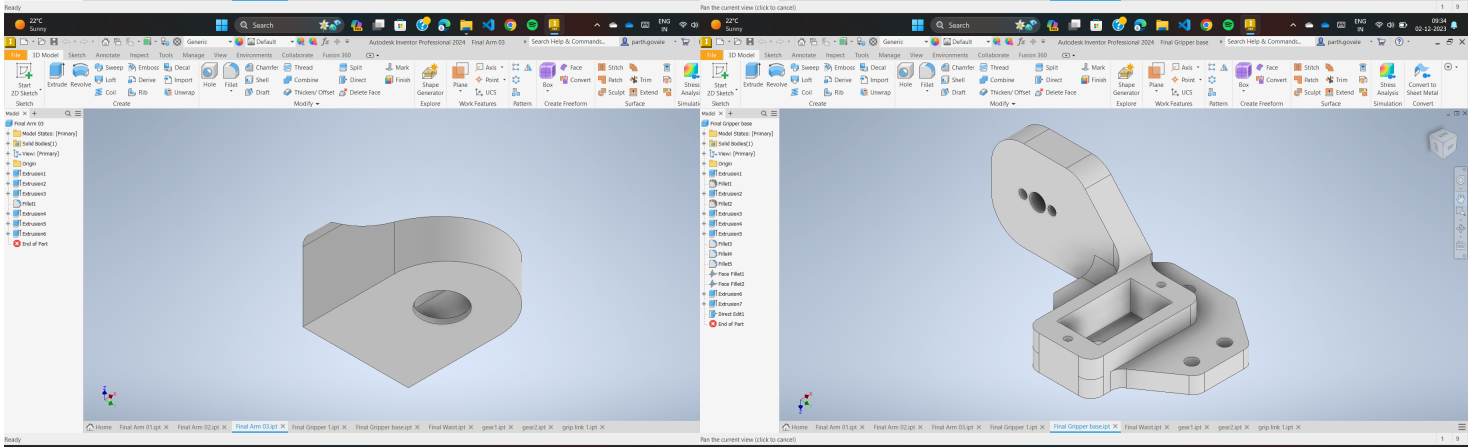


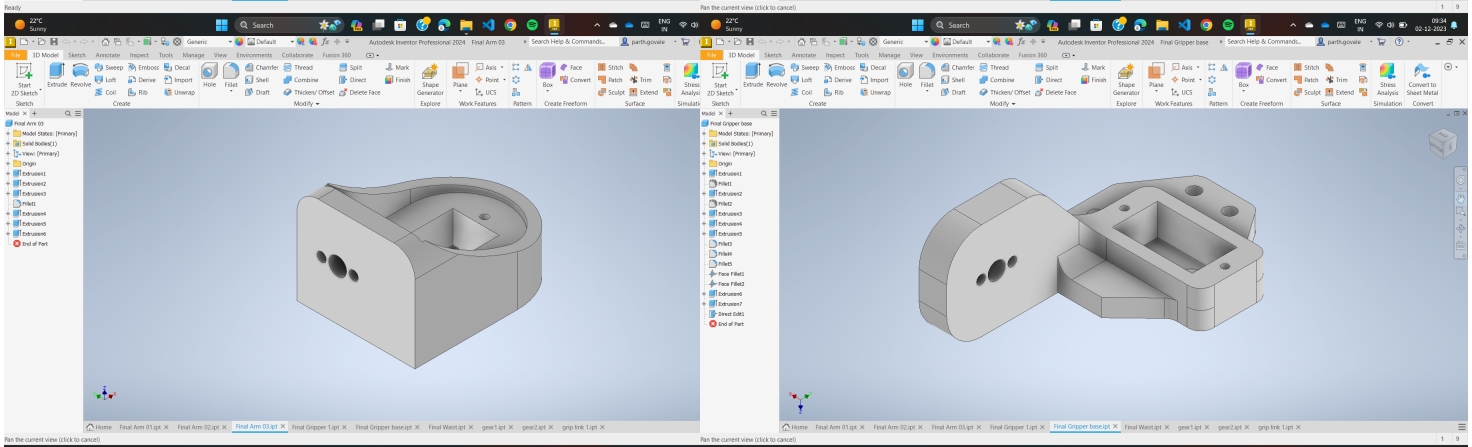


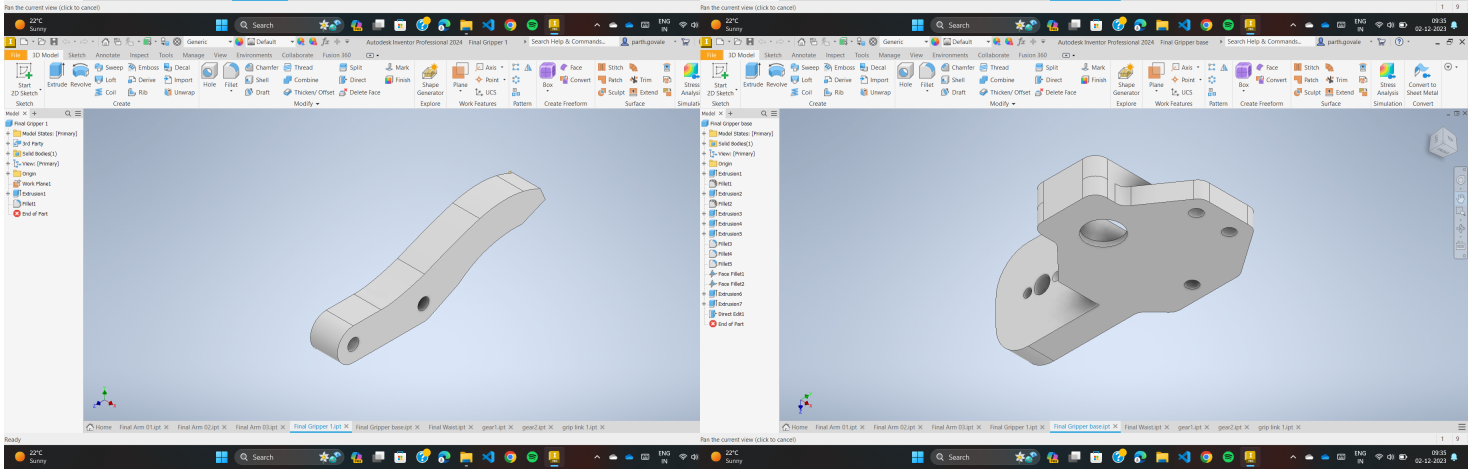


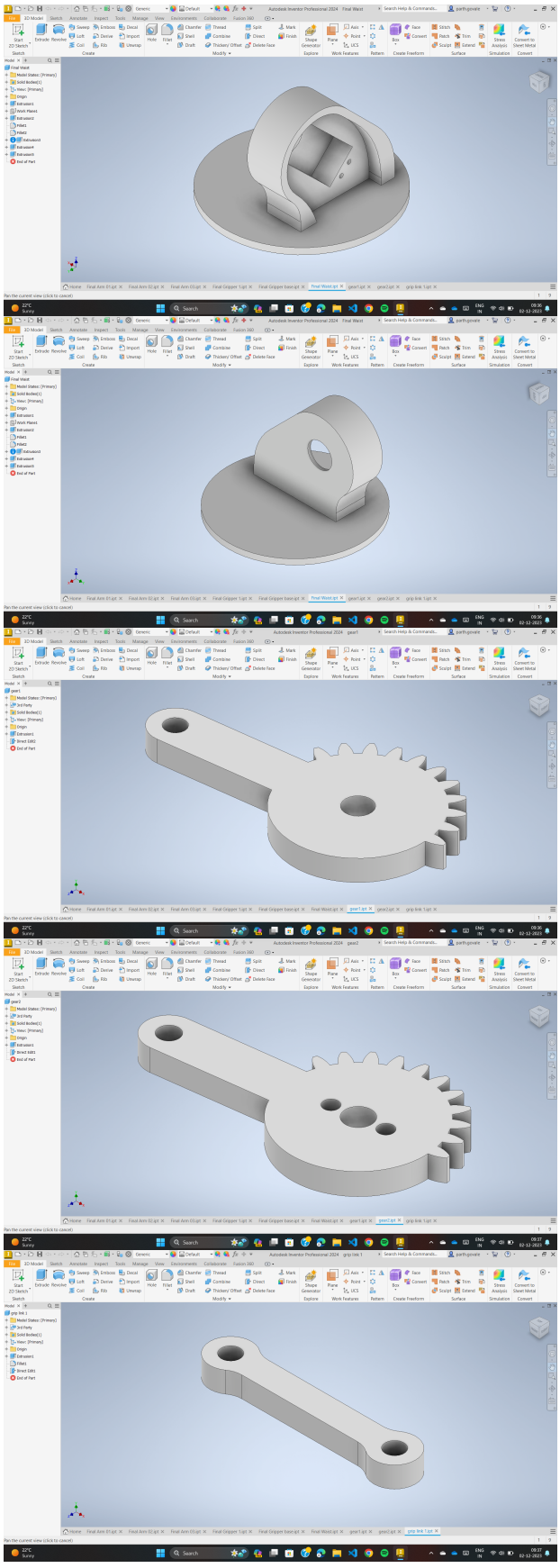












12. 3D Printing the Parts

ABS (Acrylonitrile Butadiene Styrene) in black was selected for its commendable mechanical properties, including high impact resistance, strength, and durability. The black colour enhances aesthetics while providing a professional and sleek appearance. ABS's favourable properties make it well-suited for the demanding conditions of our robotic arm.

12.1. Mechanical Strength

ABS offers high tensile and flexural strength, ensuring durability and longevity. It has a tensile strength in the range of 29.6 - 48 MPa.

12.2. Durability and Impact Resistance

The pick-and-place robotic arm is expected to perform repetitive tasks in dynamic settings. ABS's high impact resistance makes it resilient to sudden forces or collisions that may occur during operation.

12.3. Compactibility

The material's compatibility with this additive manufacturing process allows for precise layer-by-layer deposition, resulting in intricate and accurate part geometries. The ease of 3D printing with ABS facilitates the creation of complex shapes and customised components.

12.4. Material Availability and Cost Effective

ABS is widely available and is a cost-effective material for 3D printing applications. The accessibility and affordability of ABS contribute to the feasibility of the project.

13. Custom App

13.1. Code for Custom App

```
#include <SoftwareSerial.h>
#include <Servo.h>

Servo servo01;
Servo servo02;
Servo servo03;
Servo servo04;
Servo servo05;
Servo servo06;

SoftwareSerial Bluetooth(3, 4); //
    Arduino(RX, TX) - HC-05 Bluetooth (TX
    , RX)

int servo1Pos, servo2Pos, servo3Pos,
    servo4Pos, servo5Pos, servo6Pos; //
    current position
```

```
int servo1PPos, servo2PPos, servo3PPos,
    servo4PPos, servo5PPos, servo6PPos;
    // previous position
int servo01SP[50], servo02SP[50],
    servo03SP[50], servo04SP[50],
    servo05SP[50], servo06SP[50]; // for
    storing positions/steps
int speedDelay = 20;
int index = 0;
String dataIn = "";

void setup() {
  servo01.attach(5);
  servo02.attach(6);
  servo03.attach(7);
  servo04.attach(8);
  servo05.attach(9);
  servo06.attach(10);
  Bluetooth.begin(38400); // Default
     baud rate of the Bluetooth module
  Bluetooth.setTimeout(1);
  delay(20);
  // Robot arm initial position
  servo1PPos = 90;
  servo01.write(servo1PPos);
  servo2PPos = 150;
  servo02.write(servo2PPos);
  servo3PPos = 35;
  servo03.write(servo3PPos);
  servo4PPos = 140;
  servo04.write(servo4PPos);
  servo5PPos = 85;
  servo05.write(servo5PPos);
  servo6PPos = 80;
  servo06.write(servo6PPos);
}

void loop() {
  // Check for incoming data
  if (Bluetooth.available() > 0) {
    dataIn = Bluetooth.readString();  //
        Read the data as string

    // If "Waist" slider has changed
       value - Move Servo 1 to position
    if (dataIn.startsWith("s1")) {
      String dataInS = dataIn.substring
          (2, dataIn.length()); //
          Extract only the number. E.g.
          from "s1120" to "120"
      servo1Pos = dataInS.toInt();  //
          Convert the string into integer
      // We use for loops so we can
          control the speed of the servo
      // If previous position is bigger
          then current position
      if (servo1PPos > servo1Pos) {
        for ( int j = servo1PPos; j >=
            servo1Pos; j--) {   // Run
            servo down
          servo01.write(j);
          delay(20);    // defines the
              speed at which the servo
              rotates
        }
      }
      // If previous position is smaller
          then current position
      if (servo1PPos < servo1Pos) {
        for ( int j = servo1PPos; j <=
            servo1Pos; j++) {   // Run
            servo up
          servo01.write(j);
          delay(20);
        }
      }
      servo1PPos = servo1Pos;   // set
          current position as previous
          position
    }

    // Move Servo 2
    if (dataIn.startsWith("s2")) {
      String dataInS = dataIn.substring
          (2, dataIn.length());
      servo2Pos = dataInS.toInt();

      if (servo2PPos > servo2Pos) {
        for ( int j = servo2PPos; j >=
            servo2Pos; j--) {
          servo02.write(j);
          delay(50);
        }
      }
      if (servo2PPos < servo2Pos) {
        for ( int j = servo2PPos; j <=
            servo2Pos; j++) {
          servo02.write(j);
          delay(50);
        }
      }
      servo2PPos = servo2Pos;
    }
    // Move Servo 3
    if (dataIn.startsWith("s3")) {
      String dataInS = dataIn.substring
          (2, dataIn.length());
      servo3Pos = dataInS.toInt();
      if (servo3PPos > servo3Pos) {
        for ( int j = servo3PPos; j >=
            servo3Pos; j--) {
          servo03.write(j);
          delay(30);
        }
      }
```

```
  if (servo3PPos < servo3Pos) {
    for ( int j = servo3PPos; j <=
       servo3Pos; j++) {
      servo03.write(j);
      delay(30);
    }
  }
  servo3PPos = servo3Pos;
}
// Move Servo 4
if (dataIn.startsWith("s4")) {
  String dataInS = dataIn.substring
     (2, dataIn.length());
  servo4Pos = dataInS.toInt();
  if (servo4PPos > servo4Pos) {
    for ( int j = servo4PPos; j >=
       servo4Pos; j--) {
      servo04.write(j);
      delay(30);
    }
  }
  if (servo4PPos < servo4Pos) {
    for ( int j = servo4PPos; j <=
       servo4Pos; j++) {
      servo04.write(j);
      delay(30);
    }
  }
  servo4PPos = servo4Pos;
}
// Move Servo 5
if (dataIn.startsWith("s5")) {
  String dataInS = dataIn.substring
     (2, dataIn.length());
  servo5Pos = dataInS.toInt();
  if (servo5PPos > servo5Pos) {
    for ( int j = servo5PPos; j >=
       servo5Pos; j--) {
      servo05.write(j);
      delay(30);
    }
  }
  if (servo5PPos < servo5Pos) {
    for ( int j = servo5PPos; j <=
       servo5Pos; j++) {
      servo05.write(j);
      delay(30);
    }
  }
  servo5PPos = servo5Pos;
}
// Move Servo 6
if (dataIn.startsWith("s6")) {
  String dataInS = dataIn.substring
     (2, dataIn.length());
  servo6Pos = dataInS.toInt();
  if (servo6PPos > servo6Pos) {
    for ( int j = servo6PPos; j >=
       servo6Pos; j--) {
      servo06.write(j);
      delay(30);
    }
  }
  if (servo6PPos < servo6Pos) {
    for ( int j = servo6PPos; j <=
       servo6Pos; j++) {
      servo06.write(j);
      delay(30);
    }
  }
  servo6PPos = servo6Pos;
}
// If button "SAVE" is pressed
if (dataIn.startsWith("SAVE")) {
  servo01SP[index] = servo1PPos;  //
     save position into the array
  servo02SP[index] = servo2PPos;
  servo03SP[index] = servo3PPos;
  servo04SP[index] = servo4PPos;
  servo05SP[index] = servo5PPos;
  servo06SP[index] = servo6PPos;
  index++;                        //
     Increase the array index
}
// If button "RUN" is pressed
if (dataIn.startsWith("RUN")) {
  runservo();  // Automatic mode -
     run the saved steps
}
// If button "RESET" is pressed
if ( dataIn == "RESET") {
  memset(servo01SP, 0, sizeof(
     servo01SP)); // Clear the array
      data to 0
  memset(servo02SP, 0, sizeof(
     servo02SP));
  memset(servo03SP, 0, sizeof(
     servo03SP));
  memset(servo04SP, 0, sizeof(
     servo04SP));
  memset(servo05SP, 0, sizeof(
     servo05SP));
  memset(servo06SP, 0, sizeof(
     servo06SP));
  index = 0;  // Index to 0
}
  }
}

// Automatic mode custom function - run
   the saved steps
void runservo() {
  while (dataIn != "RESET") {   // Run
     the steps over and over again until
      "RESET" button is pressed
```

```
for (int i = 0; i <= index - 2; i++)
    {  // Run through all steps(
   index)
  if (Bluetooth.available() > 0) {
          // Check for incomding
     data
    dataIn = Bluetooth.readString();
    if ( dataIn == "PAUSE") {
                // If button "PAUSE
       " is pressed
      while (dataIn != "RUN") {
                // Wait until "RUN"
          is pressed again
        if (Bluetooth.available() >
           0) {
          dataIn = Bluetooth.
             readString();
          if ( dataIn == "RESET") {
            break;
          }
        }
      }
    }
    // If speed slider is changed
    if (dataIn.startsWith("ss")) {
      String dataInS = dataIn.
         substring(2, dataIn.length
         ());
      speedDelay = dataInS.toInt();
         // Change servo speed (
         delay time)
    }
  }
  // Servo 1
  if (servo01SP[i] == servo01SP[i +
     1]) {
  }
  if (servo01SP[i] > servo01SP[i +
     1]) {
    for ( int j = servo01SP[i]; j >=
        servo01SP[i + 1]; j--) {
      servo01.write(j);
      delay(speedDelay);
    }
  }
  if (servo01SP[i] < servo01SP[i +
     1]) {
    for ( int j = servo01SP[i]; j <=
        servo01SP[i + 1]; j++) {
      servo01.write(j);
      delay(speedDelay);
    }
  }

  // Servo 2
  if (servo02SP[i] == servo02SP[i +
     1]) {
  }
  if (servo02SP[i] > servo02SP[i +
     1]) {
    for ( int j = servo02SP[i]; j >=
        servo02SP[i + 1]; j--) {
      servo02.write(j);
      delay(speedDelay);
    }
  }
  if (servo02SP[i] < servo02SP[i +
     1]) {
    for ( int j = servo02SP[i]; j <=
        servo02SP[i + 1]; j++) {
      servo02.write(j);
      delay(speedDelay);
    }
  }

  // Servo 3
  if (servo03SP[i] == servo03SP[i +
     1]) {
  }
  if (servo03SP[i] > servo03SP[i +
     1]) {
    for ( int j = servo03SP[i]; j >=
        servo03SP[i + 1]; j--) {
      servo03.write(j);
      delay(speedDelay);
    }
  }
  if (servo03SP[i] < servo03SP[i +
     1]) {
    for ( int j = servo03SP[i]; j <=
        servo03SP[i + 1]; j++) {
      servo03.write(j);
      delay(speedDelay);
    }
  }

  // Servo 4
  if (servo04SP[i] == servo04SP[i +
     1]) {
  }
  if (servo04SP[i] > servo04SP[i +
     1]) {
    for ( int j = servo04SP[i]; j >=
        servo04SP[i + 1]; j--) {
      servo04.write(j);
      delay(speedDelay);
    }
  }
  if (servo04SP[i] < servo04SP[i +
     1]) {
    for ( int j = servo04SP[i]; j <=
        servo04SP[i + 1]; j++) {
      servo04.write(j);
      delay(speedDelay);
    }
  }
```

```
      // Servo 5
      if (servo05SP[i] == servo05SP[i +
          1]) {
      }
      if (servo05SP[i] > servo05SP[i +
          1]) {
        for ( int j = servo05SP[i]; j >=
            servo05SP[i + 1]; j--) {
          servo05.write(j);
          delay(speedDelay);
        }
      }
      if (servo05SP[i] < servo05SP[i +
          1]) {
        for ( int j = servo05SP[i]; j <=
            servo05SP[i + 1]; j++) {
          servo05.write(j);
          delay(speedDelay);
        }
      }

      // Servo 6
      if (servo06SP[i] == servo06SP[i +
          1]) {
      }
      if (servo06SP[i] > servo06SP[i +
          1]) {
        for ( int j = servo06SP[i]; j >=
            servo06SP[i + 1]; j--) {
          servo06.write(j);
          delay(speedDelay);
        }
      }
      if (servo06SP[i] < servo06SP[i +
          1]) {
        for ( int j = servo06SP[i]; j <=
            servo06SP[i + 1]; j++) {
          servo06.write(j);
          delay(speedDelay);
        }
      }
    }
  }
}
```


Acknowledgments

We express our sincere gratitude to Mr. Aniruddh, our esteemed faculty advisor, whose unwavering support and guidance have been instrumental throughout the development of this pick-and-place robotic arm project. His expertise, encouragement, and invaluable insights have significantly enriched our learning experience. Our heartfelt thanks also go to Pupul and Pankaj Bhaiya, our dedicated Teaching Assistants, for their continuous assistance and constructive feedback. Their commitment to our project, willingness to address queries, and insightful suggestions have been indispensable in navigating challenges and achieving milestones. A special appreciation is extended to our team members, whose collaborative efforts and dedication have been the driving force behind the success of this project. Each member's unique contributions and collective commitment have played a crucial role in overcoming hurdles and realising the vision of our pick-and-place robotic arm. We acknowledge the entire academic community and our peers for fostering an environment that promotes innovation and collaborative learning. The collective support from all these individuals has not only shaped this project but has also contributed to our overall growth as aspiring engineers.

References

[1] H. Kopka and P. W. Daly, *A Guide to LaTeX*, 3rd ed. Harlow, England: Addison-Wesley, 1999.