A Mini–Project Report on

ROBOTIC ARM

Bachelor Of Technology

In

Artificial Intelligence

By

Name	Roll No.
Shantanu Kumar	21911A35B6
Momeenah	21911A35B5
Ganesh	21911A35B4
Aishwarya	21911A35B7

Under the Guidance of

Dr. V. Phanindra

Assistant Professor

Department Of Artificial Intelligence

Vidya Jyothi Institute Of Technology

(Accredited by NAAC, Approved by AICTE New Delhi & Permanently Affiliated to JNTUH)

Aziz Nagar Gate, C.B. Post, Hyderabad-500 075

Overview

The 'Robotic Arm' project involves creating a functional robotic arm controlled by four servo motors for joint movement and four potentiometers for adjusting the positions of the joints. The end effector at the arm's tip enables interaction with objects.

Components

1. Micro Servo Motors (4): Control joint movement.
2. Potentiometers (4): Adjust servo positions.
3. Jumper Wires: Connect components.
4. Cardboard and Wood: Construct the arm and base.
5. Arduino Nano
6. Breadboard
7. Soldering machine
8. Insulation tape
9. Glue gun

Assembly Instructions

1. Constructing the Arm Framework

- Cut cardboard pieces to create the base, lower arm, upper arm, and end effector.
- Assemble the base to house the first servo motor securely.
- Attach servo motors at each joint: base, lower arm, upper arm, and end effector.

2. Wiring and Electronics Setup

- Connect potentiometers to the analog pins (A0-A3) on the Arduino.
- Solder the potentiometer wires to ensure secure and stable connections.
- Wire the potentiometers to the corresponding servo motors.
- Connect the servo motors to the designated digital pins (3, 5, 6, 9) on the Arduino.

3. System Testing and Calibration

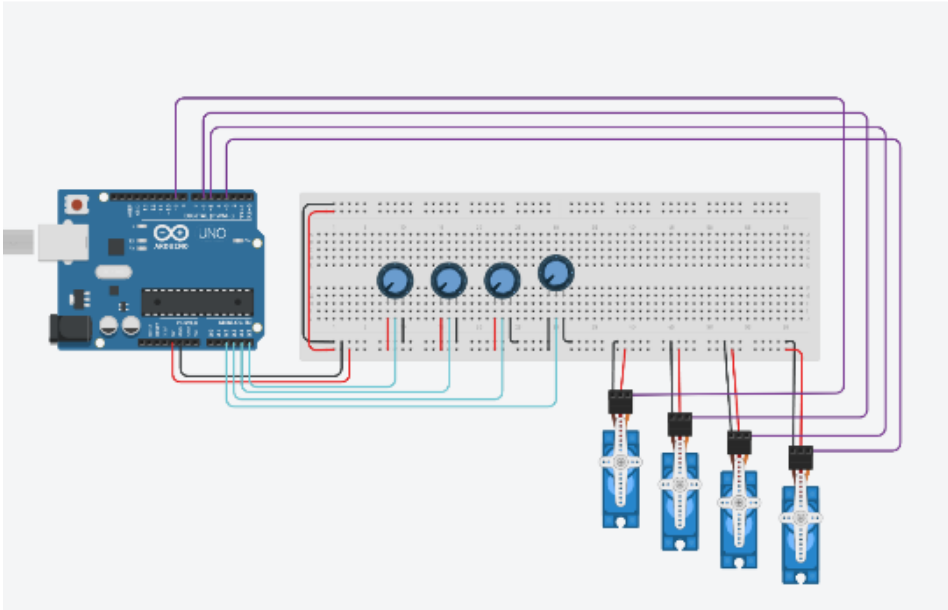
- Power on the system and adjust each potentiometer to verify the corresponding servo movement.
- Calibrate the system to achieve smooth, proportional movements and desired positions for each servo.

4. Final Assembly

- Attach the Gripper to the final servo motor.
- Make final adjustments for stability and functionality.

Circuit Diagram

Refer to the following image for the complete circuit diagram. It includes connections between the potentiometers, Arduino Nano, servos and Breadboard.



Arduino Code

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

```
// Analog pins for potentiometer inputs
int base = A0;
int first = A1;
int second = A2;
int third = A3;

// Variables to store potentiometer values
int value_claw;
int value_base;
int value_joint1;
int value_joint2;

// Servo objects
Servo servo_claw;
Servo servo_base;
Servo servo_joint1;
Servo servo_joint2;

void setup() {
  // Attach servos to their respective pins
  servo_claw.attach(9);   // Claw servo connected to pin 9
  servo_base.attach(3);   // Base servo connected to pin 3
  servo_joint1.attach(5); // First joint servo connected to pin 5
  servo_joint2.attach(6); // Second joint servo connected to pin 6

  // Initialize serial communication
  Serial.begin(9600);
}

void loop() {
  // Read potentiometer values
  int pot_inputs[4];
  pot_inputs[0] = analogRead(base);   // Reading the base potentiometer
  pot_inputs[1] = analogRead(first);  // Reading the first potentiometer
  pot_inputs[2] = analogRead(second); // Reading the second potentiometer
  pot_inputs[3] = analogRead(third);  // Reading the third potentiometer

  // Determine claw state based on potentiometer value
  if (pot_inputs[3] > 511) {  // The third potentiometer controls the claw
    value_claw = 1;
  } else {
    value_claw = 0;
  }

  // Map potentiometer values to servo angles
  value_base = map(pot_inputs[0], 110, 800, 180, 0);   // Base potentiometer
to base servo
```

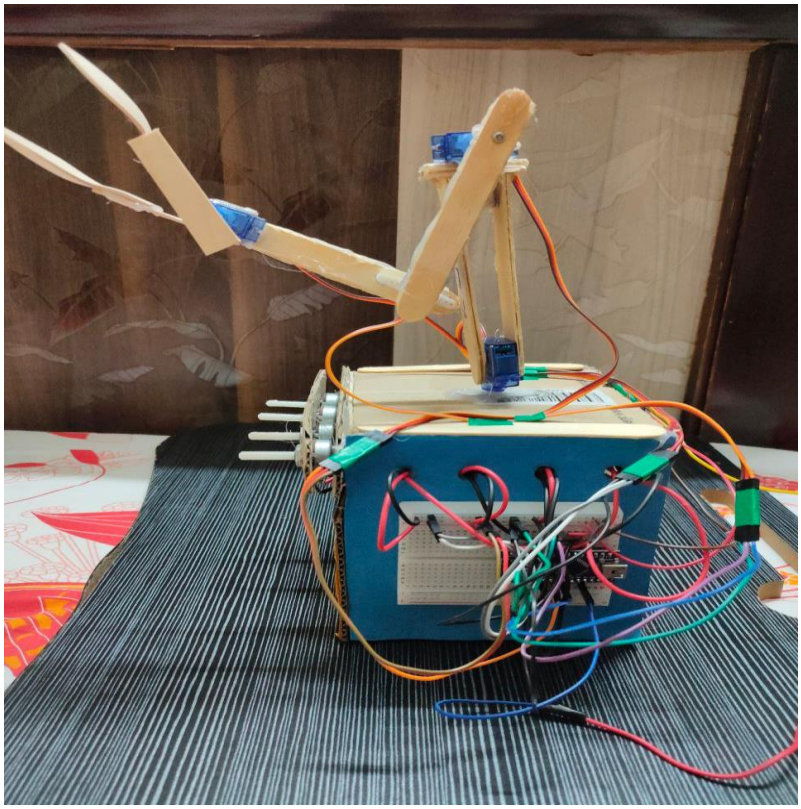
```
  value_joint1 = map(pot_inputs[1], 1023, 600, 30, 160);  // First
potentiometer to first joint servo
  value_joint2 = map(pot_inputs[2], 170, 880, 180, 0);  // Second
potentiometer to second joint servo

  // Set servo positions based on mapped values
  if (value_claw) {
    servo_claw.write(80);  // Claw open position
  } else {
    servo_claw.write(50);  // Claw close position
  }

  servo_base.write(value_base);      // Control base servo
  servo_joint1.write(value_joint1);  // Control first joint servo
  servo_joint2.write(value_joint2);  // Control second joint servo

  // Small delay to ensure smooth operation
  delay(5);
}
```

Model



Troubleshooting

1. **Servo Motor Not Moving:** Ensure that the wires are securely connected and correctly attached to the appropriate ports.

2. **Inconsistent Movement:** Recalibrate the potentiometers and check for any obstructions that might hinder the servo's motion.\

Use Cases

1. **Assembly Line Automation**
   - Description: Performs repetitive tasks like assembling components.
   - Benefits: Enhances efficiency and precision, reduces human error.

2. **Medical Assistance**
   - Description: Handles surgical tools, administers medication.
   - Benefits: Provides steady movements, maintains hygiene standards.

3. **Home Automation**
   - Description: Performs household tasks like picking up items.
   - Benefits: Enhances convenience, assists elderly or disabled individuals.