Low-Cost Robotic Arm

Client: []
Prepared by: [Sattanathan .P]
Date: July 27, 2025
Version: 1.0

Table of Contents

1. Introduction

The Low-Cost Robotic Arm is a four-degree-of-freedom (4-DOF) robotic arm designed for cost-effective automation, suitable for educational, hobbyist, and small-scale industrial applications. It features an ESP32 microcontroller for primary control via a WiFi-enabled web interface and an Arduino Uno for backup control using a joystick or auto mode. This documentation, prepared to the standards of top IT companies, provides comprehensive details on the system's design, setup, operation, and maintenance for our client.

2. Project Overview

The Low-Cost Robotic Arm offers:

- Manual Control: Adjust servo positions via a web interface (ESP32) or joystick (Arduino Uno).
- Record and Playback: Store and replay up to 50 servo positions (ESP32).
- Auto Mode: Execute a 16-step pick-and-place sequence (both ESP32 and Arduino Uno).
- WiFi Connectivity: Remote control via a web server (SSID: "low cost arm", Password: "12345678").
- Backup Control: Arduino Uno supports joystick-based manual control and auto mode without joystick.
- Cost-Effective Design: Utilizes affordable components like four servo motors, acrylic sheets, and M3 screws.

3. Hardware Requirements

Component	Description
4 Servo Motors	Control base (0-180°), right arm (40-100°), left arm (0-40°), gripper (0-28°).
ESP32 Microcontroller	Primary controller for WiFi and servo management.
Arduino Uno (Backup)	Secondary controller for joystick or auto mode.
Joystick	Analog input for manual control (X: A0, Y: A1).
Wire Set	Ensures reliable connections.
External Power Supply	Powers servos to avoid microcontroller overload.
6V Battery	Provides portable power.
Breadboard	Facilitates prototyping.
Acrylic Sheet (Cut Parts)	Forms the lightweight frame.
M3 Screws	Secures acrylic parts and motors.

4. System Architecture

- ESP32 Control Unit: Hosts a web server on port 80 (SSID: "low cost arm") for servo control, recording, and auto sequences.
- Arduino Uno Backup: Supports joystick control (manual mode) or a pre-programmed auto sequence.
- Actuators: Four servos connected to:
    - ESP32: Base (Pin 2), Right Arm (Pin 4), Left Arm (Pin 5), Gripper (Pin 18).
    - Arduino Uno: Base (Pin 9), Right Arm (Pin 10), Left Arm (Pin 11), Gripper (Pin 12).
- Power Management: External power supply and 6V battery for servos; ESP32/Arduino powered via USB or battery.

- **Frame: Acrylic sheets assembled with M3 screws.**

Diagram (Placeholder): ***Add hardware schematic*****.**

---

5. Software Implementation

5.1 ESP32 Code Explanation

The ESP32 code uses the Arduino framework with libraries:

- **WiFi.h: Creates a WiFi access point.**
- **WebServer.h: Hosts the web interface.**
- **ESP32Servo.h: Controls servos.**
- **EEPROM.h: Stores recorded positions.**

Key Features

- **WiFi SSID: "low cost arm".**
- **Web Page Title: "Low Cost Robotic Arm Controller".**
- **Functions:**
    - **setup(): Initializes serial, EEPROM, servos, WiFi, and server routes.**
    - **loop(): Handles client requests and executes playback/auto sequences.**
    - **handleMove(): Moves servos with angle constraints, records positions if enabled.**
    - **recordCurrentPosition(): Stores up to 50 positions.**
    - **playRecordedSequence(): Replays positions with 1-second delays.**
    - **runAutoSequence(): Executes a 16-step pick-and-place sequence.**
    - **moveServoSlowly(): Smooths servo transitions.**
    - **moveToDefaults(): Resets to default positions (Base: 0°, Right Arm: 90°, Left Arm: 0°, Gripper: 0°).**

5.2 Arduino Uno Code Explanation

The Arduino Uno code supports:

- **Manual Mode: Uses map() to convert joystick analog inputs (0-1023) to servo angles.**
- **Auto Mode: Executes the same 16-step pick-and-place sequence as the ESP32, without joystick input.**
- **Libraries: Servo.h for servo control.**

Key Features

- **Joystick Mapping: Uses map() to scale analog inputs to servo angles (e.g., X-axis for Base/Right Arm, Y-axis for Left Arm/Gripper).**
- **Auto Mode: Triggered by a button (Pin 7) or serial command, runs the 16-step sequence.**
- **Functions:**
  - **setup(): Initializes servos and joystick pins.**
  - **loop(): Checks for mode (manual/auto) and processes inputs.**
  - **moveServoSlowly(): Smooths servo transitions.**
  - **runAutoSequence(): Implements the 16-step sequence.**
  - **moveToDefaults(): Resets servos.**

5.3 Web Interface

The web interface (title: "Low Cost Robotic Arm Controller") is a responsive HTML page:

- **Modes: Manual (sliders for servo control) and Auto (16-step sequence).**
- **Controls: Record, playback, stop/reset, and auto sequence buttons.**
- **Status: Displays real-time servo positions and operation status.**
- **JavaScript: Sends HTTP requests and updates sliders every 2 seconds.**

6. Installation and Setup

1. **Assemble Frame:**
   - **Secure acrylic parts with M3 screws.**
   - **Attach servos to joints.**
2. **Connect Hardware:**
   - **ESP32: Wire servos to Pins 2, 4, 5, 18; connect external power supply and 6V battery.**
   - **Arduino Uno: Wire servos to Pins 9, 10, 11, 12; joystick to A0 (X), A1 (Y); button to Pin 7.**
3. **Upload Code:**
   - **Install Arduino IDE and libraries (WiFi.h, WebServer.h, ESP32Servo.h, EEPROM.h for ESP32; Servo.h for Arduino).**
   - **Flash ESP32 and Arduino codes.**
4. **WiFi Setup:**
   - **Connect to "low cost arm" (password: "12345678").**
   - **Access web interface via ESP32's IP (Serial Monitor).**
5. **Arduino Setup:**
   - **Use joystick for manual control or button/serial command for auto mode.**

---

7. Operation Guide

ESP32 (Primary)

- **Manual Mode: Use web interface sliders to adjust servos; record/playback sequences.**
- **Auto Mode: Start 16-step sequence via "Start Auto Sequence" button.**
- **Monitoring: Check status on web interface or Serial Monitor (115200 baud).**

Arduino Uno (Backup)

- **Manual Mode: Move joystick to control servos (X: Base/Right Arm, Y: Left Arm/Gripper).**
- **Auto Mode: Press button (Pin 7) or send 'A' via Serial to start 16-step sequence.**
- **Reset: Send 'R' via Serial or press reset button to return to defaults.**

---

8. Testing and Validation

- **Servo Accuracy: Verified full range of motion for all servos.**
- **WiFi Stability: Tested web interface across devices.**
- **Recording/Playback: Confirmed accurate sequence recording and playback (ESP32).**
- **Auto Sequence: Validated 16-step sequence on both ESP32 and Arduino.**
- **Joystick Control: Ensured smooth manual control with map() function.**

---

9. Maintenance and Troubleshooting

Maintenance

- **Inspect connections, battery levels, and frame integrity.**
- **Lubricate joints if stiff.**
- **Update Arduino libraries.**

Troubleshooting

Issue	Solution
WiFi Failure	Restart ESP32, verify SSID ("low cost arm") and password.
Servo Jitter	Check external power supply.
Joystick Failure	Verify Arduino connections, re-upload code, test joystick.

Issue	Solution
Web Interface Unresponsive	Refresh page, restart ESP32, check network.
Auto Mode Failure	Ensure button/serial input is correct; check code.

10. Conclusion

The Low-Cost Robotic Arm delivers a robust, affordable automation solution with dual control modes (web and joystick), WiFi connectivity, and a reliable backup system. This documentation ensures the client can operate and maintain the system effectively, meeting top IT company standards.

11. Contact Information

- Name: [Sattanathan .P]
- Email: [sattanathan2025@gmail.com]
- Phone: [+91- 9176529555]

12. Appendix: Code Listings

12.1 ESP32 Code

```
#include <WiFi.h>
#include <WebServer.h>
#include <ESP32Servo.h>
#include <EEPROM.h>

// WiFi credentials
```

```
const char* ssid = "low cost arm";
const char* password = "12345678";

// Create web server on port 80
WebServer server(80);

// Servo objects
Servo baseServo;
Servo rightArmServo;
Servo leftArmServo;
Servo gripperServo;

// Servo pins
const int basePIN = 2;
const int rightArmPIN = 4;
const int leftArmPIN = 5;
const int gripperPIN = 18;

// Current servo positions
int basePos = 0;
int rightArmPos = 90;
int leftArmPos = 0;
int gripperPos = 0;

// Recording variables
struct ServoPosition {
  int base;
  int rightArm;
  int leftArm;
  int gripper;
};
ServoPosition recordedPositions[50];
int recordedCount = 0;
bool isRecording = false;
bool isPlaying = false;
bool autoMode = false;

// HTML page
const char* htmlPage = R"rawliteral(
<!DOCTYPE html>
<html>
<head>
    <title>Low Cost Robotic Arm Controller</title>
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <style>
        body { font-family: Arial; text-align: center; background: #f0f0f0; margin: 0; padding: 20px; }
        .container { max-width: 800px; margin: 0 auto; background: white; padding: 20px; border-radius: 10px; box-shadow: 0 4px 6px rgba(0,0,0,0.1); }
        h1 { color: #333; margin-bottom: 30px; }
        .mode-buttons { margin-bottom: 30px; }
        .mode-btn { padding: 15px 30px; margin: 10px; font-size: 18px; border: none; border-radius: 5px; cursor: pointer; }
        .manual-btn { background: #4CAF50; color: white; }
        .auto-btn { background: #2196F3; color: white; }
        .servo-control { margin: 20px 0; padding: 15px; background: #f9f9f9; border-radius: 5px; }
        .servo-label { font-weight: bold; margin-bottom: 10px; }
        .slider { width: 80%; height: 25px; margin: 10px 0; }
        .value-display { font-size: 18px; font-weight: bold; color: #333; margin: 5px 0; }
        .control-buttons { margin: 20px 0; }
        .control-btn { padding: 12px 25px; margin: 10px; font-size: 16px; border: none; border-radius: 5px; cursor: pointer; }
        .record-btn { background: #FF5722; color: white; }
        .play-btn { background: #4CAF50; color: white; }
        .stop-btn { background: #f44336; color: white; }
        .auto-start-btn { background: #9C27B0; color: white; }
        .hidden { display: none; }
        .status { margin: 20px 0; padding: 15px; background: #e3f2fd; border-radius: 5px; font-weight: bold; }
    </style>
</head>
<body>
    <div class="container">
        <h1>🦾 Low Cost Robotic Arm Controller</h1>
        <div class="mode-buttons">
          <button class="mode-btn manual-btn" onclick="setMode('manual')">📱 Manual Control</button>
          <button class="mode-btn auto-btn" onclick="setMode('auto')">🤖 Auto Mode</button>
        </div>

        <div id="status" class="status">Ready to operate</div>

        <div id="manualSection">

          <div class="servo-control">

            <div class="servo-label">🔄 Base Servo (0-180°)</div>

            <input type="range" min="0" max="180" value="0" class="slider" id="baseSlider" oninput="updateServo('base', this.value)">

            <div class="value-display">Position: <span id="baseValue">0</span>°</div>

          </div>

          <div class="servo-control">

            <div class="servo-label">💪 Right Arm (40-100°)</div>

            <input type="range" min="40" max="100" value="90" class="slider" id="rightArmSlider" oninput="updateServo('rightArm', this.value)">

            <div class="value-display">Position: <span id="rightArmValue">90</span>°</div>

          </div>

          <div class="servo-control">

            <div class="servo-label">🤲 Left Arm (0-40°)</div>

            <input type="range" min="0" max="40" value="0" class="slider" id="leftArmSlider" oninput="updateServo('leftArm', this.value)">

            <div class="value-display">Position: <span id="leftArmValue">0</span>°</div>

          </div>

          <div class="servo-control">

            <div class="servo-label">✋ Gripper (0-28°)</div>

            <input type="range" min="0" max="28" value="0" class="slider" id="gripperSlider" oninput="updateServo('gripper', this.value)">

            <div class="value-display">Position: <span id="gripperValue">0</span>°</div>
          </div>
          <div class="control-buttons">
            <button class="control-btn record-btn" id="recordBtn" onclick="toggleRecord()">🔴 Start Recording</button>
            <button class="control-btn play-btn" onclick="playRecording()">▶ Play Recording</button>
            <button class="control-btn stop-btn" onclick="stopAll()">⏹ Stop & Reset</button>
          </div>
        </div>
        <div id="autoSection" class="hidden">
          <h3>🤖 Automatic Operation</h3>
          <p>Pre-programmed 16-step pick and place sequence</p>
          <div class="control-buttons">
            <button class="control-btn auto-start-btn" onclick="startAuto()">🚀 Start Auto Sequence</button>
            <button class="control-btn stop-btn" onclick="stopAll()">⏹ Stop Auto</button>
          </div>
        </div>
      </div>
      <script>
        let isRecording = false;
        let currentMode = 'manual';
        function setMode(mode) {
          currentMode = mode;
          if (mode === 'manual') {
```

```
    document.getElementById('manualSection').classList.remove('hidden');
    document.getElementById('autoSection').classList.add('hidden');
  } else {
    document.getElementById('manualSection').classList.add('hidden');
    document.getElementById('autoSection').classList.remove('hidden');
  }
  updateStatus('Mode switched to ' + mode);
}
function updateServo(servo, value) {
  document.getElementById(servo + 'Value').innerText = value;
  fetch('/move?servo=' + servo + '&value=' + value)
    .then(response => response.text())
    .then(data => console.log(data));
}
function toggleRecord() {
  const btn = document.getElementById('recordBtn');
  if (!isRecording) {
    isRecording = true;
    btn.innerText = '⏺️ Stop Recording';
    btn.style.background = '#FF9800';
    fetch('/record?action=start');
    updateStatus('Recording started - adjust servos to record positions');
  } else {
    isRecording = false;
    btn.innerText = '🔴 Start Recording';
    btn.style.background = '#FF5722';
    fetch('/record?action=stop');
    updateStatus('Recording stopped');
  }
}
function playRecording() {
  fetch('/play')
    .then(response => response.text())
    .then(data => {
      updateStatus('Playing recorded sequence...');
    });
}
function startAuto() {
  fetch('/auto')
    .then(response => response.text())
    .then(data => {
      updateStatus('Running automatic pick & place sequence...');
    });
}
function stopAll() {
  fetch('/stop');
  updateStatus('Stopping all operations - returning to default position');
  document.getElementById('baseSlider').value = 0;
  document.getElementById('rightArmSlider').value = 90;
  document.getElementById('leftArmSlider').value = 0;
  document.getElementById('gripperSlider').value = 0;
  document.getElementById('baseValue').innerText = 0;
```

```
      document.getElementById('rightArmValue').innerText = 90;
      document.getElementById('leftArmValue').innerText = 0;
      document.getElementById('gripperValue').innerText = 0;
    }
    function updateStatus(message) {
      document.getElementById('status').innerText = message;
    }
    setInterval(() => {
      fetch('/status')
        .then(response => response.json())
        .then(data => {
          document.getElementById('baseSlider').value = data.base;
          document.getElementById('rightArmSlider').value = data.rightArm;
          document.getElementById('leftArmSlider').value = data.leftArm;
          document.getElementById('gripperSlider').value = data.gripper;
          document.getElementById('baseValue').innerText = data.base;
          document.getElementById('rightArmValue').innerText = data.rightArm;
          document.getElementById('leftArmValue').innerText = data.leftArm;
          document.getElementById('gripperValue').innerText = data.gripper;
        });
    }, 2000);
  </script>
</body>
</html>
)rawliteral";
```

```
void setup() {
  Serial.begin(115200);
  EEPROM.begin(512);
  baseServo.attach(basePIN);
  rightArmServo.attach(rightArmPIN);
  leftArmServo.attach(leftArmPIN);
  gripperServo.attach(gripperPIN);
  moveToDefaults();
  WiFi.softAP(ssid, password);
  IPAddress IP = WiFi.softAPIP();
  Serial.print("AP IP address: ");
  Serial.println(IP);
  server.on("/", handleRoot);
  server.on("/move", handleMove);
  server.on("/record", handleRecord);
  server.on("/play", handlePlay);
  server.on("/auto", handleAuto);
  server.on("/stop", handleStop);
  server.on("/status", handleStatus);
  server.begin();
  Serial.println("Web server started");
  Serial.println("Connect to WiFi: low cost arm");
  Serial.println("Password: 12345678");
  Serial.print("Open browser to: ");
  Serial.println(IP);
}
```

```
void loop() {
  server.handleClient();
  if (isPlaying) {
    playRecordedSequence();
  }
  if (autoMode) {
    runAutoSequence();
  }
}

void handleRoot() {
  server.send(200, "text/html", htmlPage);
}

void handleMove() {
  String servo = server.arg("servo");
  int value = server.arg("value").toInt();
  if (servo == "base") {
    basePos = constrain(value, 0, 180);
    baseServo.write(basePos);
  } else if (servo == "rightArm") {
    rightArmPos = constrain(value, 40, 100);
    rightArmServo.write(rightArmPos);
  } else if (servo == "leftArm") {
    leftArmPos = constrain(value, 0, 40);
    leftArmServo.write(leftArmPos);
  } else if (servo == "gripper") {
    gripperPos = constrain(value, 0, 28);
    gripperServo.write(gripperPos);
  }
  if (isRecording) {
    recordCurrentPosition();
  }
  server.send(200, "text/plain", "Moved " + servo + " to " + value);
  Serial.println("Moved " + servo + " to " + String(value));
}

void handleRecord() {
  String action = server.arg("action");
  if (action == "start") {
    isRecording = true;
    recordedCount = 0;
    server.send(200, "text/plain", "Recording started");
    Serial.println("Recording started");
  } else if (action == "stop") {
    isRecording = false;
    server.send(200, "text/plain", "Recording stopped. " + String(recordedCount) + " positions recorded");
    Serial.println("Recording stopped. Positions: " + String(recordedCount));
  }
}
```

```
void handlePlay() {
  if (recordedCount > 0) {
    isPlaying = true;
    server.send(200, "text/plain", "Playing recorded sequence");
    Serial.println("Playing recorded sequence");
  } else {
    server.send(200, "text/plain", "No recorded sequence found");
    Serial.println("No recorded sequence");
  }
}

void handleAuto() {
  autoMode = true;
  server.send(200, "text/plain", "Starting automatic sequence");
  Serial.println("Starting auto mode");
}

void handleStop() {
  isPlaying = false;
  isRecording = false;
  autoMode = false;
  moveToDefaults();
  server.send(200, "text/plain", "Stopped all operations");
  Serial.println("All operations stopped");
}
```

```
void handleStatus() {
  String json = "{";
  json += "\"base\":" + String(basePos) + ",";
  json += "\"rightArm\":" + String(rightArmPos) + ",";
  json += "\"leftArm\":" + String(leftArmPos) + ",";
  json += "\"gripper\":" + String(gripperPos);
  json += "}";
  server.send(200, "application/json", json);
}

void recordCurrentPosition() {
  if (recordedCount < 50) {
    recordedPositions[recordedCount].base = basePos;
    recordedPositions[recordedCount].rightArm = rightArmPos;
    recordedPositions[recordedCount].leftArm = leftArmPos;
    recordedPositions[recordedCount].gripper = gripperPos;
    recordedCount++;
    Serial.println("Position " + String(recordedCount) + " recorded");
  }
}

void playRecordedSequence() {
  static int currentStep = 0;
  static unsigned long lastMove = 0;
  if (millis() - lastMove > 1000) {
    if (currentStep < recordedCount) {
      baseServo.write(recordedPositions[currentStep].base);
      rightArmServo.write(recordedPositions[currentStep].rightArm);
      leftArmServo.write(recordedPositions[currentStep].leftArm);
      gripperServo.write(recordedPositions[currentStep].gripper);
      basePos = recordedPositions[currentStep].base;
      rightArmPos = recordedPositions[currentStep].rightArm;
      leftArmPos = recordedPositions[currentStep].leftArm;
      gripperPos = recordedPositions[currentStep].gripper;
      Serial.println("Playing step " + String(currentStep + 1));
      currentStep++;
      lastMove = millis();
    } else {
      isPlaying = false;
      currentStep = 0;
      Serial.println("Playback completed");
    }
  }
}

void runAutoSequence() {
  static int step = 0;
  static unsigned long lastMove = 0;
  if (millis() - lastMove > 2000) {
    switch (step) {
      case 0:
```

```
  moveServoSlowly(baseServo, basePos, 150);
  basePos = 150;
  Serial.println("Auto Step 1: Base to 150°");
  break;
case 1:
  moveServoSlowly(rightArmServo, rightArmPos, 35);
  rightArmPos = 35;
  Serial.println("Auto Step 2: Right arm to 35°");
  break;
case 2:
  moveServoSlowly(leftArmServo, leftArmPos, 30);
  leftArmPos = 30;
  Serial.println("Auto Step 3: Left arm to 30°");
  break;
case 3:
  moveServoSlowly(gripperServo, gripperPos, 27);
  gripperPos = 27;
  Serial.println("Auto Step 4: Gripper open to 27°");
  delay(2000);
  break;
case 4:
  moveServoSlowly(gripperServo, gripperPos, 10);
  gripperPos = 10;
  Serial.println("Auto Step 5: Gripper close to 10°");
  break;
case 5:
  moveServoSlowly(rightArmServo, rightArmPos, 90);
  rightArmPos = 90;
  Serial.println("Auto Step 6: Right arm to 90°");
  break;
case 6:
  moveServoSlowly(leftArmServo, leftArmPos, 0);
  leftArmPos = 0;
  Serial.println("Auto Step 7: Left arm to 0°");
  break;
case 7:
  moveServoSlowly(baseServo, basePos, 0);
  basePos = 0;
  Serial.println("Auto Step 8: Base to 0°");
  break;
case 8:
  moveServoSlowly(rightArmServo, rightArmPos, 30);
  rightArmPos = 30;
  Serial.println("Auto Step 9: Right arm to 30°");
  break;
case 9:
  moveServoSlowly(leftArmServo, leftArmPos, 30);
  leftArmPos = 30;
  Serial.println("Auto Step 10: Left arm to 30°");
  break;
case 10:
  moveServoSlowly(gripperServo, gripperPos, 27);
  gripperPos = 27;
  Serial.println("Auto Step 11: Gripper open to 27°");
  break;
case 11:
  moveServoSlowly(gripperServo, gripperPos, 0);
  gripperPos = 0;
  Serial.println("Auto Step 12: Gripper close to 0°");
  break;
case 12:
  moveServoSlowly(rightArmServo, rightArmPos, 90);
  rightArmPos = 90;
  Serial.println("Auto Step 13: Right arm to 90°");
  break;
case 13:
  moveServoSlowly(leftArmServo, leftArmPos, 0);
  leftArmPos = 0;
  Serial.println("Auto Step 14: Left arm to 0°");
  break;
case 14:
  moveServoSlowly(baseServo, basePos, 150);
  basePos = 150;
  Serial.println("Auto Step 15: Base to 150°");
  break;
case 15:
  step = -1;
  Serial.println("Auto Step 16: Looping sequence");
```

```
        break;
    }
    step++;
    lastMove = millis();
  }
}

void moveServoSlowly(Servo &servo, int startPos, int endPos) {
  int stepSize = (startPos < endPos) ? 1 : -1;
  for (int pos = startPos; pos != endPos; pos += stepSize) {
    servo.write(pos);
    delay(50);
  }
  servo.write(endPos);
}
void moveToDefaults() {
  basePos = 0;
  rightArmPos = 90;
  leftArmPos = 0;
  gripperPos = 0;
  baseServo.write(basePos);
  rightArmServo.write(rightArmPos);
  leftArmServo.write(leftArmPos);
  gripperServo.write(gripperPos);
  Serial.println("Moved to default positions");
}
```

12.2 Arduino Uno Code

This code supports manual joystick control using the map() function and an auto mode triggered by a button or serial command, without joystick input.

```
#include <Servo.h>


// Servo objects
Servo baseServo;
Servo rightArmServo;
Servo leftArmServo;
Servo gripperServo;


// Servo pins
const int basePin = 9;
const int rightArmPin = 10;
const int leftArmPin = 11;
const int gripperPin = 12;


// Joystick pins
const int joystickX = A0;
const int joystickY = A1;
const int buttonPin = 7; // Button to toggle auto mode


// Servo positions
int basePos = 0;
int rightArmPos = 90;
int leftArmPos = 0;
int gripperPos = 0;

// Auto mode flag
bool autoMode = false;

void setup() {
  Serial.begin(9600);
  // Attach servos
  baseServo.attach(basePin);
  rightArmServo.attach(rightArmPin);
  leftArmServo.attach(leftArmPin);
  gripperServo.attach(gripperPin);
  // Set initial positions
  moveToDefaults();
  // Initialize button pin
  pinMode(buttonPin, INPUT_PULLUP); // Active low
  Serial.println("Low-Cost Robotic Arm: Ready (Manual Mode)");
  Serial.println("Press button or send 'A' for Auto Mode, 'R' to Reset");
}

void loop() {
  // Check for serial input
  if (Serial.available() > 0) {
    char command = Serial.read();
    if (command == 'A') {
```

```
  autoMode = true;
  Serial.println("Switching to Auto Mode");
 } else if (command == 'R') {
  autoMode = false;
  moveToDefaults();
  Serial.println("Reset to Manual Mode");
 }
}

// Check button for auto mode toggle
if (digitalRead(buttonPin) == LOW) {
 delay(50); // Debounce
 if (digitalRead(buttonPin) == LOW) {
  autoMode = !autoMode;
  if (autoMode) {
   Serial.println("Switching to Auto Mode");
  } else {
   moveToDefaults();
   Serial.println("Switching to Manual Mode");
  }
  while (digitalRead(buttonPin) == LOW); // Wait for release
 }
}

if (autoMode) {
 runAutoSequence();
```

```
  } else {
    // Manual mode: Read joystick and map to servo positions
    int xVal = analogRead(joystickX);
    int yVal = analogRead(joystickY);

    // Map joystick values to servo angles
    basePos = map(xVal, 0, 1023, 0, 180);
    rightArmPos = map(xVal, 0, 1023, 40, 100);
    leftArmPos = map(yVal, 0, 1023, 0, 40);
    gripperPos = map(yVal, 0, 1023, 0, 28);

    // Write to servos
    baseServo.write(basePos);
    rightArmServo.write(rightArmPos);
    leftArmServo.write(leftArmPos);
    gripperServo.write(gripperPos);

    Serial.print("Base: "); Serial.print(basePos);
    Serial.print(" | Right Arm: "); Serial.print(rightArmPos);
    Serial.print(" | Left Arm: "); Serial.print(leftArmPos);
    Serial.print(" | Gripper: "); Serial.println(gripperPos);

    delay(50); // Smooth control
  }
}
```

```
void runAutoSequence() {
 static int step = 0;
 static unsigned long lastMove = 0;
 if (millis() - lastMove > 2000) {
  switch (step) {
   case 0:
    moveServoSlowly(baseServo, basePos, 150);
    basePos = 150;
    Serial.println("Auto Step 1: Base to 150°");
    break;
   case 1:
    moveServoSlowly(rightArmServo, rightArmPos, 35);
    rightArmPos = 35;
    Serial.println("Auto Step 2: Right arm to 35°");
    break;
   case 2:
    moveServoSlowly(leftArmServo, leftArmPos, 30);
    leftArmPos = 30;
    Serial.println("Auto Step 3: Left arm to 30°");
    break;
   case 3:
    moveServoSlowly(gripperServo, gripperPos, 27);
    gripperPos = 27;
    Serial.println("Auto Step 4: Gripper open to 27°");
    delay(2000);
    break;
```

```
case 4:
  moveServoSlowly(gripperServo, gripperPos, 10);
  gripperPos = 10;
  Serial.println("Auto Step 5: Gripper close to 10°");
  break;
case 5:
  moveServoSlowly(rightArmServo, rightArmPos, 90);
  rightArmPos = 90;
  Serial.println("Auto Step 6: Right arm to 90°");
  break;
case 6:
  moveServoSlowly(leftArmServo, leftArmPos, 0);
  leftArmPos = 0;
  Serial.println("Auto Step 7: Left arm to 0°");
  break;
case 7:
  moveServoSlowly(baseServo, basePos, 0);
  basePos = 0;
  Serial.println("Auto Step 8: Base to 0°");
  break;
case 8:
  moveServoSlowly(rightArmServo, rightArmPos, 30);
  rightArmPos = 30;
  Serial.println("Auto Step 9: Right arm to 30°");
  break;
case 9:
  moveServoSlowly(leftArmServo, leftArmPos, 30);

  leftArmPos = 30;

  Serial.println("Auto Step 10: Left arm to 30°");

  break;

case 10:

  moveServoSlowly(gripperServo, gripperPos, 27);

  gripperPos = 27;

  Serial.println("Auto Step 11: Gripper open to 27°");

  break;

case 11:

  moveServoSlowly(gripperServo, gripperPos, 0);

  gripperPos = 0;

  Serial.println("Auto Step 12: Gripper close to 0°");

  break;

case 12:

  moveServoSlowly(rightArmServo, rightArmPos, 90);

  rightArmPos = 90;

  Serial.println("Auto Step 13: Right arm to 90°");

  break;

case 13:

  moveServoSlowly(leftArmServo, leftArmPos, 0);

  leftArmPos = 0;

  Serial.println("Auto Step 14: Left arm to 0°");

  break;

case 14:

  moveServoSlowly(baseServo, basePos, 150);
```

```
    basePos = 150;
    Serial.println("Auto Step 15: Base to 150°");
    break;
   case 15:
    step = -1;
    Serial.println("Auto Step 16: Looping sequence");
    break;
  }
  step++;
  lastMove = millis();
 }
}

void moveServoSlowly(Servo &servo, int startPos, int endPos) {
 int stepSize = (startPos < endPos) ? 1 : -1;
 for (int pos = startPos; pos != endPos; pos += stepSize) {
  servo.write(pos);
  delay(50);
 }
 servo.write(endPos);
}

void moveToDefaults() {
 basePos = 0;
 rightArmPos = 90;
 leftArmPos = 0;
```

```
  gripperPos = 0;

  baseServo.write(basePos);

  rightArmServo.write(rightArmPos);

  leftArmServo.write(leftArmPos);

  gripperServo.write(gripperPos);

  Serial.println("Moved to default positions");

}
```
