**Servo Motor Control Using Arduino**

**1. Introduction**

This project demonstrates how to control a servo motor using an Arduino microcontroller. Servo motors are widely used in robotics, automation, and control systems due to their precision and reliability. The Arduino platform makes it easy to interface and control servo motors with minimal hardware requirements.

**2. Objectives**

* Understand the basics of servo motor control.
* Learn how to use the Servo library in Arduino.
* Build a simple circuit to control the servo motor.

**3. Materials Required**

* Arduino Uno/Mega
* Servo motor (e.g., SG90 or MG996R)
* Jumper wires
* Breadboard (optional)
* Power supply (if required for servo motor)

**4. Circuit Diagram and Connections**

**Connections**

1. **Servo Motor Wires**:
   * **Red (VCC):** Connect to 5V pin on Arduino.
   * **Brown/Black (GND):** Connect to GND pin on Arduino.
   * **Orange/White (Signal):** Connect to a PWM pin (e.g., Pin 9) on Arduino.
2. **Power Supply (Optional)**:
   * For high-power servo motors, connect an external power supply to the servo's VCC and GND pins.
   * Ensure the Arduino GND is connected to the external power supply GND.

**5.Code:**

#include <Servo.h> // Include the Servo library

Servo myServo; // Create a Servo object

void setup() {

myServo.attach(9); // Attach the servo to pin 9

Serial.begin(9600); // Start serial communication for debugging

Serial.println("Servo Motor Control Initialized");

}

void loop() {

for (int pos = 0; pos <= 180; pos += 1) { // Move from 0° to 180°

myServo.write(pos); // Set the servo position

delay(15); // Wait for the servo to reach the position

}

delay(1000); // Pause at 180°

for (int pos = 180; pos >= 0; pos -= 1) { // Move from 180° to 0°

myServo.write(pos); // Set the servo position

delay(15); // Wait for the servo to reach the position

}

delay(1000); // Pause at 0°

}

**6. Working**

1. The Arduino controls the servo motor's position by generating PWM signals on the signal pin.
2. The Servo library simplifies the generation of PWM signals, allowing users to control the servo by specifying angles (0° to 180°).
3. The code moves the servo from 0° to 180°, pauses briefly, then moves back to 0°, repeating this cycle indefinitely.

**7. Testing and Observations**

1. Power the Arduino and observe the servo motor's movement.
2. Verify that the motor rotates smoothly between 0° and 180°.
3. Adjust the delay times in the code if the servo's movement appears too fast or too slow.

**8. Applications**

* Robotics and automation systems.
* Camera gimbals and pan-tilt systems.
* Actuator mechanisms in mechanical projects.

**9. Conclusion**

This project demonstrates how to control a servo motor using Arduino in a simple and effective way. By using the Servo library, we were able to move the servo to specific angles with minimal setup. This project provides a solid foundation for building more advanced applications in robotics and automation, while also helping to understand the basics of servo motor control and Arduino programming.