**Joystick controlled industrial automation system**

#include <Servo.h>

// Define servos

Servo servoX;

Servo servoY;

// Joystick pins

const int joyXPin = A0;

const int joyYPin = A1;

const int joyButtonPin = 2; // Optional button

// Servo positions

int posX = 90; // Initial position for servo X

int posY = 90; // Initial position for servo Y

// Threshold for joystick to detect significant movement

const int threshold = 10;

void setup() {

// Attach servos to pins

servoX.attach(9);

servoY.attach(10);

// Initialize servos to mid position

servoX.write(posX);

servoY.write(posY);

// Begin serial communication for debugging

Serial.begin(9600);

// Configure joystick button pin

pinMode(joyButtonPin, INPUT\_PULLUP);

}

void loop() {

// Read joystick values

int joyXValue = analogRead(joyXPin);

int joyYValue = analogRead(joyYPin);

int joyButtonState = digitalRead(joyButtonPin);

// Debugging: Print joystick values

Serial.print("X: "); Serial.print(joyXValue);

Serial.print(" Y: "); Serial.print(joyYValue);

Serial.print(" Button: "); Serial.println(joyButtonState);

// Map joystick values to servo ranges (0 to 180 degrees)

int mappedX = map(joyXValue, 0, 1023, 0, 180);

int mappedY = map(joyYValue, 0, 1023, 0, 180);

// Apply threshold to avoid jittering

if (abs(mappedX - posX) > threshold) {

posX = mappedX;

servoX.write(posX);

}

if (abs(mappedY - posY) > threshold) {

posY = mappedY;

servoY.write(posY);

}

// Optional: Use joystick button for additional control

if (joyButtonState == LOW) {

// Add your custom button-pressed action here

Serial.println("Button pressed!");

}

// Small delay to allow for servo movement

delay(20);

}