REPORTS ON MECHATRONICS SYSTEM INTEGRATION

REPORT 3B

SERIAL COMMUNICATION

SECTION 1, SEMESTER 2, 23/24

Date of Experiment:

Date of Submission: ***20/3/2024***

Team Members:

1. IKMAL FIKRI BIN KHAIRUL KHUBAIDILLAH (2218723)
1. MUHAMMAD IRFAN BIN ROSDIN (2214417)
2. ASYRAAF AIMAN MD HASSAN (2010705)
3. MUHAMMAD IKHWAN BIN MUHAMMAD FAUZI (22188845)

Table of Content

PART B : SERVO MOTOR

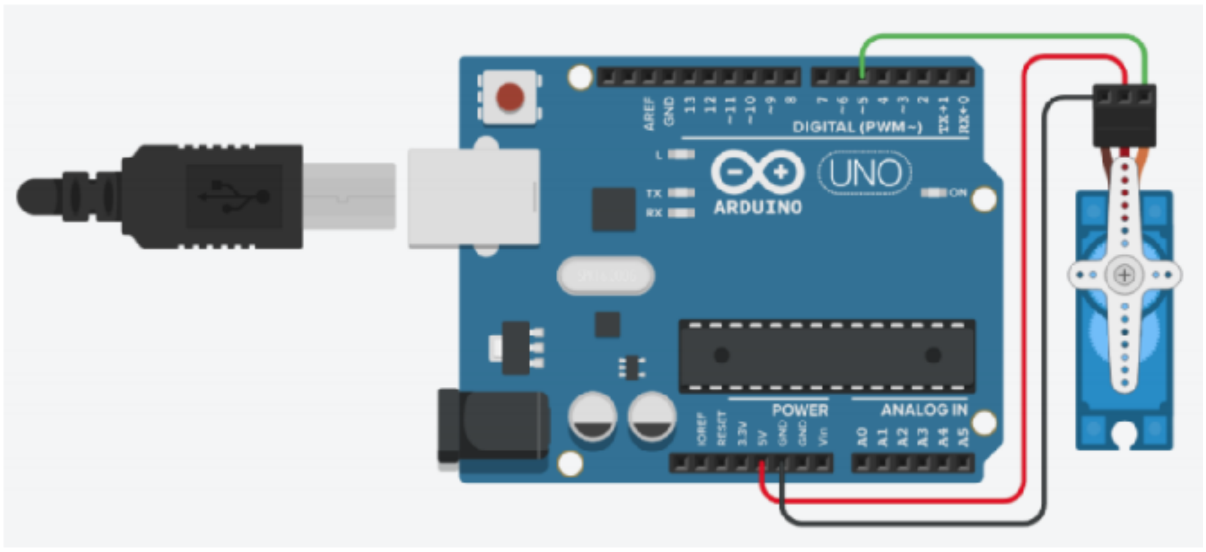
Introduction

The objective experiment is to control a servo motor using Python that involves interfacing with an Arduino board that's connected with the servo .

Materials and Equipment

Arduino board (e.g., Arduino Uno)	X 1
Servo motor	X 1
Jumper wires	X 3
Potentiometer (for manual angle input)	X 1
USB cable for Arduino	X 1
Computer with Arduino IDE and Python installed	X 1

Experimental Setup



Methodology/Procedures

1. Input an angle between 0 and 180 degrees and press Enter. The Python script will send the angle to the Arduino over the serial port.
2. The Arduino will move the servo to the specified angle, and the script will display the angle set by the servo.

3. Exit the Python script by entering 'q' and observing that the serial connection is closed.

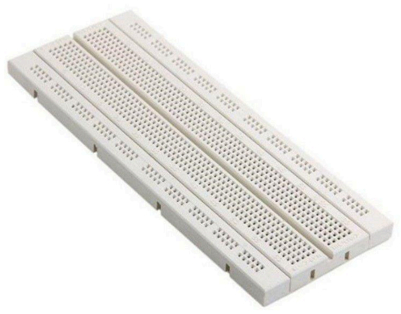
Results

The results of this experiment were acquired by setting and controlling the position of a servo motor using the input angle value received from the serial monitor. Upon startup, the servo motor is connected to digital pin 9 and serial connection is established. The loop checks for incoming input from the serial monitor and reads the angle value. The angle value is then limited to 0 to 180 degrees to ensure that it falls within the servo motor's range. Finally, the servo motor is adjusted to the new angle, and the angle value is shown on the serial monitor for verification. This experiment effectively illustrates how to control a servo motor's position using serial connection.

Discussion

Hardware

1) **Breadboard**



Breadboard is used to connect components with arduino through wire connection.

2) **Arduino Mega 2560**

Microcontroller used for our experiment is Arduino Mega 2560.

3) **Male to male jumper wires**

Male to male jumper wires are used to connect all of the components on the breadboard with Arduino Mega.

4) **Servo Motor**

To perform physical movement at particular angles.

Electrical

1. **Connect the Servo's Signal Wire:**
   Usually, you connect the servo's signal wire to a PWM-capable pin on the Arduino (e.g., digital pin 9).

2. **Power the servo using the Arduino's 5V and GND pins.**
   Servos typically require a supply voltage of +5V. You can connect the servo's power wire (usually red) to the 5V output on the Arduino board.

3. **Connect the Servo's Ground Wire:**
   Connect the servo's ground wire (usually brown) to one of the ground (GND) pins on the Arduino.

Software

Arduino code

```
#include <Servo.h>

Servo servo;
int angle = 90;

void setup()
{
  servo.attach(9); // Attach the servo to dig. pin 9
  Serial.begin(9600); // Initialize serial communication
}

void loop()
{
  if (Serial.available() > 0) // Check if data is available to read
  {
    angle = Serial.parseInt(); // Read the incoming angle value
    angle = constrain(angle, 0, 180); // Constrain the angle value between 0 and 180
    servo.write(angle); // Set the servo to the new angle
    Serial.print("Set angle: ");
    Serial.println(angle);
  }

  delay(100); // Wait for a short time to avoid continuous reading
}
```

Python code

```
import serial
import time

# Define the serial port and baud rate
(adjust the port as per your Arduino)
ser = serial.Serial('COM4', 9600)

try:
    while True:
        angle = input("Enter servo angle
(0-180 degrees): ")
        if angle.lower() == 'q':
            break
        angle = int(angle)
        if 0 <= angle <= 180:
            # Send the servo’s angle to the
Arduino
            ser.write(str(angle).encode())
        else:
            print("Angle must be between 0
and 180 degrees.")
except KeyboardInterrupt:
    pass  # Handle keyboard interrupt
finally:
    ser.close()  # Close the serial connection
    print("Serial connection closed.")
```

Question

a) Enhance your Arduino and Python code to incorporate a potentiometer for real-time adjustments of the servo motor's angle. Ensure that, in the updated Arduino code, you have the ability to halt its execution by pressing a designated key on your computer's keyboard. Following the modification, restart the Python script to receive and display servo position data from the Arduino over the serial connection. While experimenting with the potentiometer, observe the corresponding changes in the servo motor's position.

Arduino code

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

```
Servo servo;
int angle = 90;

void setup() {
  servo.attach(9); // Attach the servo to
digital pin 9
  Serial.begin(9600); // Initialize serial
communication
}

void loop() {
  if (Serial.available() > 0) {
    char key = Serial.read(); // Read the
incoming key
    if (key == 'q') { // Halt if 'q' is pressed
      while (true) {} // Halt the program
    }
  }

  int potVal = analogRead(A0); // Read
potentiometer value
  int mappedVal = map(potVal, 0, 1023, 0,
180); // Map potentiometer value to servo
angle range
  servo.write(mappedVal); // Set servo
angle
  delay(50); // Delay for stability
}
```

Python Code

```
import matplotlib.pyplot as plt
import serial
import time
import keyboard

# Open serial port
ser = serial.Serial('COM3', 9600)  #
Update 'COM3' with your Arduino's port
time.sleep(2)  # Wait for serial connection
to establish

# Initialize empty lists to store data
x_data = []
y_data_pot = []
y_data_servo = []

# Set up the plot
plt.ion()  # Turn on interactive mode
fig, ax1 = plt.subplots()
ax2 = ax1.twinx()
line_pot, = ax1.plot(x_data, y_data_pot,
'b-', label='Potentiometer Reading')
line_servo, = ax2.plot(x_data,
y_data_servo, 'r-', label='Servo Position')
ax1.set_xlabel('Time (s)')
ax1.set_ylabel('Potentiometer Reading',
color='b')
ax2.set_ylabel('Servo Position', color='r')

# Main loop
try:
    while True:
        # Read potentiometer value and servo
position from Arduino
        data = ser.readline().decode().strip()
        if data:
            values = data.split(',')
            pot_val = int(values[0])
            servo_pos = int(values[1])
            print("Potentiometer Reading:",
pot_val, "Servo Position:", servo_pos)

            # Append data to lists
            x_data.append(time.time())  # Use
time as x-axis
            y_data_pot.append(pot_val)
            y_data_servo.append(servo_pos)

            # Update plot
            line_pot.set_xdata(x_data)
            line_pot.set_ydata(y_data_pot)
            line_servo.set_xdata(x_data)
            line_servo.set_ydata(y_data_servo)
            ax1.relim()
            ax1.autoscale_view()
            ax2.relim()
            ax2.autoscale_view()
            fig.canvas.draw()
            fig.canvas.flush_events()

        # Check for keyboard input to halt
        if keyboard.is_pressed('q'):
            break

except KeyboardInterrupt:
    ser.close()
    print("Serial connection closed")
```

The Arduino and Python code together show a realistic use of real-time control and visualisation with a servo motor and potentiometer. The Arduino code instructs the servo motor to change its angle based on the potentiometer reading, allowing for dynamic control. In addition, the Arduino code incorporates a function that allows it to cease execution when a certain key is pressed, increasing its adaptability and user engagement.

The Python code, on the other hand, creates a serial connection with the Arduino in order to receive and show real-time potentiometer readings as well as servo motor positions. The Python script uses the 'matplotlib' package to generate an interactive plot that graphically portrays the data, giving users a clear and easy method to examine the system's behaviour.

These scripts demonstrate the seamless integration of hardware (Arduino) and software (Python) for real-time control and visualisation, emphasising the versatility and power of employing these tools in tandem for a wide range of interactive applications.

Conclusion

In conclusion, the code arrangement above allows this experiment to use Python to operate a servo motor link to an Arduino Mega. The Arduino code initialises the servo and serial communication before continually listening for angle values from the serial port. When a new angle is received, the servo motor is set accordingly. The Python software sets up a serial connection with the Arduino, requests the user for a new angle, then transmits it to the Arduino. This arrangement allows for real-time control of the servo motor's position from the computer, exhibiting bidirectional communication between the Arduino Mega and PC.