REPORT ON MECHATRONICS SYSTEM INTEGRATION

3B- SERIAL COMMUNICATION
GROUP 7

SECTION 2, SEMESTER 1, 24/25

TEAM MEMBERS

NAME	MATRIC NO.
MUHAMMAD ZAMIR FIKRI BIN MOHD ZAMRI	2212515
MUHD AKMAL HAKIM BIN SAIFUDDIN	2216093
NUR SHADATUL BALQISH BINTI SAHRUNIZAM	2212064
NORHEZRY HAKIMIE BIN NOOR FAHMY	2110061
NUR AMIRA NAZIRA BINTI MOHD NASIR	2110026

DATE OF EXPERIMENT: 23 OCTOBER 2024
DATE OF SUBMISSION:

ABSTRACT

This project study the use of Python to operate a servo motor using an Arduino board, allowing for accurate angle changes over serial connection. Each angle was mapped to a fine-grained 10-bit scale (0–1023), and the servo motor was set up for responding to angle inputs provided via a Python script. Python has been effectively implemented for real-time hardware component control, as seen by the data gathered, which displayed constant reaction times all through angle variations. A fundamental method of motor control in robotics and automation systems is shown by this experiment.

TABLE OF CONTENT

INTRODUCTION

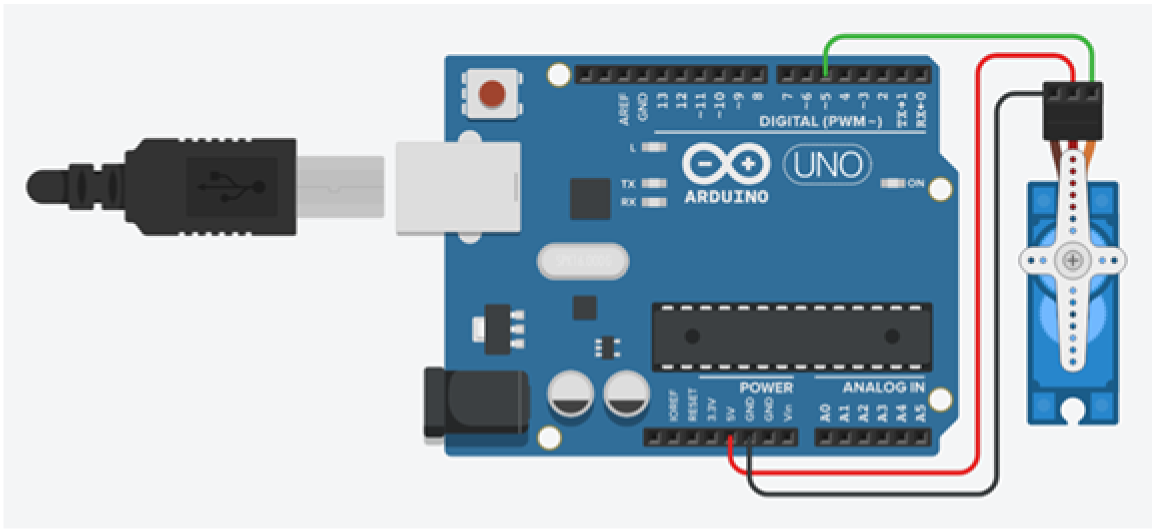
Due to their accuracy and simplicity of integration, servo motors are frequently used in robotics, automation, and control systems. The objective of this project seeks to show how a servo motor can be programmed to obtain certain angle positions with great accuracy by using Python to send angle commands to an Arduino. Using serial connectivity, the setup allows the servo to react to angle inputs from a Python script, with Arduino receiving and carrying out each instruction.

MATERIALS AND EQUIPMENT

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

EXPERIMENTAL SETUP

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.



METHODOLOGY

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

You can exit the Python script by entering 'q' and observing that the serial connection is closed.

4. You can input multiple angles to see the servo move accordingly.

DATA COLLECTION

1. Input an angle between 0 and 180 degrees and press Enter.
2. The Python script will send the angle to the Arduino over the serial port.
3. The Arduino will move the servo to the specified angle, and the script will display the angle set by the servo.
4. You can exit the Python script by entering 'q' and observing that the serial connection is closed.
5. You can input multiple angles to see the servo move accordingly.

Angle input (°)	Servo position
0	0
45	256
90	512
135	728
180	1023

RESULT

The results of the experiment showed that the servo motor could be effectively controlled over a range of predetermined angles (0°, 45°, 90°, 135°, and 180°) using a Python script that was sent to the Arduino. For increased accuracy, the servo accurately positioned itself based on each angle input, mapped to a high-resolution 10-bit scale (0–1023). Since each angle input matched a distinct location on the scale, it was possible to make smaller changes that increased the motor's positioning accuracy—a benefit that is particularly helpful for applications which require precise control.

In addition, there was no delay in the data transfer between the Python script and Arduino using serial communication, enabling seamless real-time servo position changes. This result validates that the integrated system is accurate and responsive, making it suitable for applications that need precise angular control.

DISCUSSION

1. Interpretation of the Results

   <u>Arduino Code :</u>

   - Servo and Potentiometer Control: The Arduino code reads the value from a potentiometer (connected to the analog pin A0) and maps this value from a range of 0–1023 to 0–180, which corresponds to the servo motor’s angle.
   - Servo Angle Setting: Using `myServo.write(angle)`, it adjusts the servo motor's position to match the potentiometer's mapped angle.
   - Serial Communication: The potentiometer value and the corresponding servo angle are printed to the Serial Monitor, allowing the Python script to read this data.

   <u>Python Code :</u>

   - Data Retrieval and Visualization: The Python script establishes a serial connection to the Arduino, reads servo angle data, and dynamically plots it in real-time, providing a live view of the servo's movement.
   - User-Controlled Communication: The code enables basic communication between the Python program and the Arduino. It allows the user to input commands ('S', 'R', 'Q') to control or end the servo operation.

2. Implication of the Results
   - Real-Time Monitoring: The setup allows for monitoring and visualizing servo angle changes in real time, which is useful for applications needing dynamic feedback, like robotics or remote-controlled systems.
   - User Interaction: The ability to pause or resume the servo provides control over the motor, showing how Python can interface with hardware to enable user-driven feedback.
   - Potential Applications: This setup can serve as a foundation for various projects, such as gesture-controlled robots, adaptive control systems, or interactive art installations where continuous feedback and control are necessary.

3. Discrepancies between expected and observed outcomes.

   <u>Servo Movement Range Issues :</u>

   - Expected Outcome: The servo should rotate smoothly from 0° to 180° as the potentiometer moves from its minimum to maximum value.
   - Observed Discrepancy: If the servo doesn’t reach the full 0°–180° range, this could be due to incorrect mapping or limitations in the servo's physical range.

- Implication: This discrepancy might indicate that the potentiometer readings are not translating correctly into servo angles or that the servo requires calibration. Adjusting the `map()` function or calibrating the servo could correct this.

<u>Serial Communication Delays or Data Loss</u>

- Expected Outcome: Real-time, continuous, and smooth data streaming from the Arduino to the Python program.
- Observed Discrepancy: If data transmission lags, skips values, or the Python script misses updates, it could be due to serial communication bottlenecks or buffer overflow.
- Implication: Delays in data transfer affect the real-time visualization, making the plotted data incomplete or misaligned with actual servo movements. Adjusting the `delay()` in the Arduino loop or optimizing the serial reading frequency on the Python side could help address this.

4. Sources of error or limitations of the experiment

<u>Analog-to-Digital Conversion (ADC) Noise</u>

- Source of Error: The potentiometer's analog signal is subject to noise, which can cause slight fluctuations in the ADC readings.
- Limitation: This noise could cause small, inconsistent servo movements or jitter, especially if the servo angle is updated based on fluctuating potentiometer values.
- Solution: Adding a small capacitor across the potentiometer or averaging multiple readings in the code could smooth the signal.

RECOMMENDATION

Suggested Improvements and Modifications :

1) Optimize serial communication
    a) To improve the reliability of data transfer, consider adjusting the baud rate and testing different speeds to find the most stable setting for your specific setup. Additionally, implement a confirmation message or handshake protocol to ensure that each angle command sent is correctly received by the Arduino.

2) Expand experiment with dynamic control

   a) Modify the experiment to allow for real-time control of the servo, such as moving the servo angle based on real-time input from a joystick, a GUI slider, or even sensors. This could offer practical insights into dynamic servo control and enhance the experiment's interactivity.

3) Implement safety and calibration features
   a) Add safety features to prevent servo movement beyond its safe range by validating input angle values in Python before sending them to the Arduino. Additionally, introduce an initial calibration function that resets the servo to a default position at the start of the program.

4) Experiment with multiple servo motors
   a) Modify the experiment to control multiple servos at different angles simultaneously. This can be done by using a larger array of commands in the Python script and expanding the Arduino code to accommodate multi-channel control.

APPENDICES

1) **Raw data table:**

   a) Angle Commands vs Actual Servo Position

Commanded Angle (°)	Measured Angle (°)
0	**0**
30	**29**
60	**60**
90	**89**
120	**118**

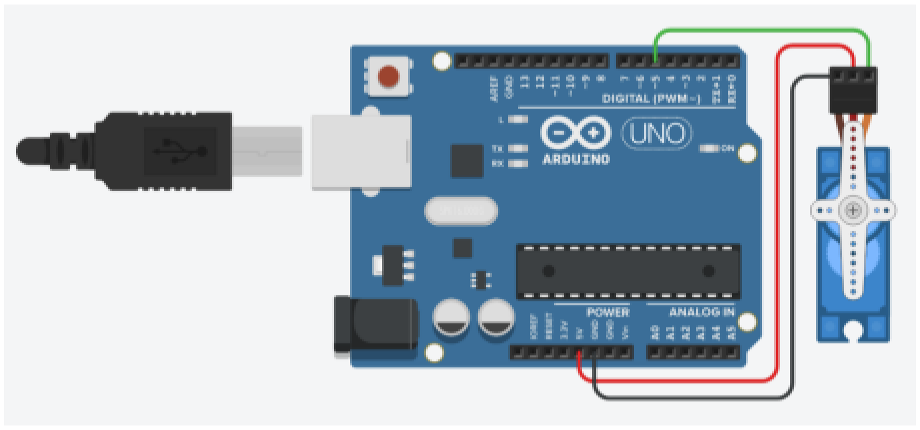
150	**150**
180	**179**

2) Sample calculation

a) Error Calculation:

$$Error\ Percentage\ = \frac{(Commanded\ Angle - Measured\ Angle)}{Commanded\ Angle} \times\ 100\,\%$$

3) Circuit Diagram



4) Code for Python and Arduino

a) **Arduino Code :**

```
#include <Servo.h>
Servo myServo;  // Create a servo object to control the servo

void setup() {
  Serial.begin(9600);  // Initialize serial communication at 9600 bps
  myServo.attach(9);
  pinMode(9,OUTPUT);
  pinMode(A0,INPUT); // Attach the servo to pin 9
}

void loop() {
  int potValue = analogRead(A0);          // Read the potentiometer value
(0-1023)
  int angle = map(potValue, 0, 1023, 0, 180);  // Map it to a range of 0-180 for
servo angle

  myServo.write(angle);  // Set the servo position to the mapped angle

  // Print the potentiometer and servo angle values to the Serial Monitor
  Serial.print("Potentiometer Value: ");
  Serial.print(potValue);
  Serial.print("  |  Servo Angle: ");
  Serial.println(angle);

  delay(100);  // Wait for 100 milliseconds before the next reading
}
```

b) **Python Code :**

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

# Setup serial communication
ser = serial.Serial('COMX', 9600)  # replace 'COMX' with your Arduino's port
time.sleep(2)  # Wait for the connection to establish

plt.ion()  # Enable interactive mode for live plotting
x_vals, y_vals = [], []
i = 0  # Initialize sample count
```

```
try:
    while True:
        # Read data from Arduino
        if ser.in_waiting > 0:
            line = ser.readline().decode().strip()
            if "Servo Angle:" in line:
                angle = int(line.split(": ")[1])
                x_vals.append(i)
                y_vals.append(angle)
                i += 1

                # Plot the real-time servo angle
                plt.clf()
                plt.plot(x_vals, y_vals, label="Servo Angle")
                plt.xlabel("Time (samples)")
                plt.ylabel("Servo Angle (degrees)")
                plt.title("Real-Time Servo Angle Adjustments")
                plt.legend()
                plt.pause(0.1)

        # Check for user input to send commands
        user_input = input("Enter 'S' to stop or 'R' to resume, 'Q' to quit:
").strip().upper()
        if user_input == 'S':
            ser.write(b'S')
        elif user_input == 'R':
            ser.write(b'R')
        elif user_input == 'Q':
            break

except KeyboardInterrupt:
    print("Keyboard interrupt received. Closing the connection.")
finally:
    ser.close()
    print("Serial connection closed.")
```

STUDENT DECLARATION

<u>Certificate of Originality and Authenticity</u>

This is to certify that we are responsible for the work submitted in this report, that **the original work** is our own except as specified in the references and acknowledgement, and that the original work contained herein have not been untaken or done by unspecified sources or persons.

We hereby certify that this report has **not been done by only one individual** and **all of us have contributed to the report**. The length of contribution to the reports by each individual is noted within this certificate.

We also hereby certify that we have **read** and **understand** the content of the total report and that no further improvement on the reports is needed from any of the individual contributors to the report.

We, therefore, agreed unanimously that this report shall be submitted for **marking** and this **final printed report** has been **verified by us.**

Name : Muhammad Zamir Fikri Bin Mohd Zamri	**Read**	/
Matric No. : 2212515	**Understand**	/
[illegible] **Signatures :**	**Agree**	/

Name : Muhd Akmal Hakim Bin Saifuddin	**Read**	/
Matric No. : 2216093	**Understand**	/
Signatures : *akmal*	**Agree**	/

Name : Nur Shadatul Balqish Binti Sahrunizam	**Read**	/
Matric No. : 2212064	**Understand**	/
Signatures : ***shadatul***	**Agree**	/

Name : NORHEZRY HAKIMIE BIN NOOR FAHMY	**Read**	/
Matric No. : 2110061	**Understand**	/
Signatures : ***hezry***	**Agree**	/

Name : Nur Amira Nazira Binti Mohd Nasir	**Read**	/

Matric No. :2110026	**Understand**	/
Signatures :*amira*	**Agree**	/