REPORT 2: SERIAL COMMUNICATION

GROUP 2

MCTA 3203

SEMESTER 1 2024/2025

MECHATRONICS SYSTEM INTEGRATION

DATE OF SUBMISSION: 30/10/2024

	NAME	MATRIC NUMBER
1.	AKMAL ZARIF BIN NAJIB	2211971
2.	AMIR FARHAN BIN GHAFFAR	2115617
3.	AMIRAH HUDA BINTI JAMALULAIL ASRI	2210776
4.	AMIRUL AIZAD BIN AMIR	2211263
5.	ANAS BIN BADRULZAMAN	2219945

TABLE OF CONTENTS

ABSTRACT

This experiment demonstrates the use of serial communication between an Arduino microcontroller and a computer using Python language. Before setting up the circuit, the programming code must be present on both devices to establish a serial communication. A potentiometer is connected to the Arduino along with its code, sending it over the serial port to read the potentiometer value. The Python script reads the data from the Arduino via the serial port and displays the values. The values can be graphically visualised in the form of a graph using the "matplotlib" feature in the Python script. The second part of this experiment involves controlling a servo motor that has been interfaced with an Arduino board. Similar steps are taken where the servo's wires are connected to the Arduino, and code is written and read over the serial port. The objective was to establish a reliable communication channel, enabling data exchange between the two platforms.

INTRODUCTION

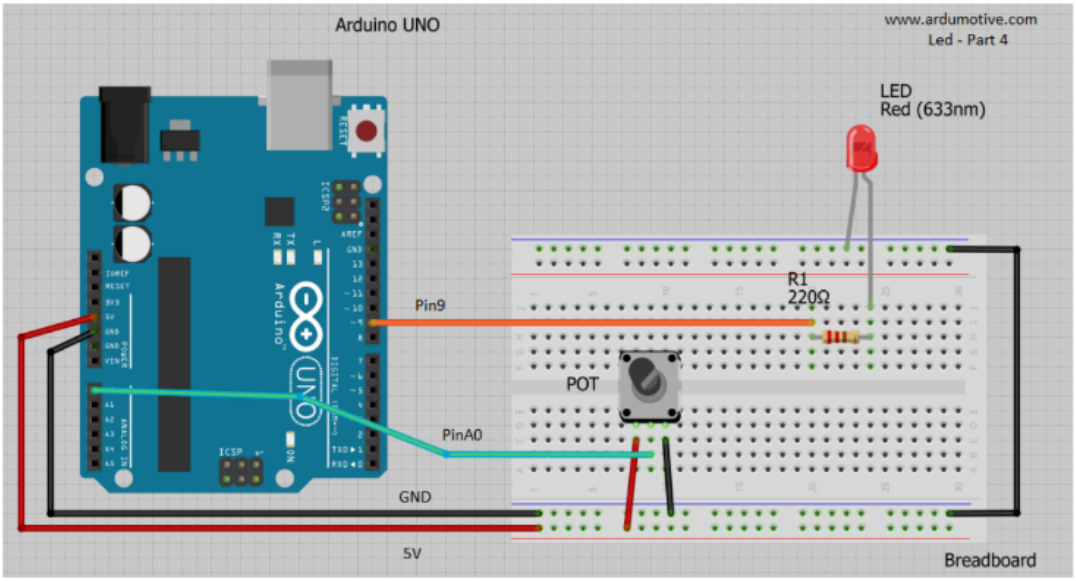
In data transmission, serial communication is a method that uses transmission lines to continuously send and receive data one bit at a time over a communication channel. A common way of exchanging data between a computer and a microcontroller is by establishing serial communication between Python and Arduino. This is done by programming a code and having it written on both the Python side and the Arduino side. Real-life interaction can then be demonstrated and analysed as the communication between the two devices has been established. The primary objective of this experiment is to understand this efficient method of data transmission as well as explore practical scenarios and applications that could be useful. Understanding this interaction is crucial for developing systems where microcontrollers need to communicate with data-logging devices and other external hardware.

EXPERIMENT 3A - <u>Sending potentiometer readings from an Arduino to a Python script through a USB connection.</u>

MATERIALS AND EQUIPMENT

- Arduino Uno board
- Potentiometer
- LED
- 220 Ω resistor
- Jumper wires
- Breadboard

EXPERIMENTAL SETUP



METHODOLOGY

1. Purpose: The aim is to demonstrate how an Arduino can send data to a computer. This process allows the Arduino to act as a “sender,” transmitting information from a connected sensor (potentiometer) to the computer. The computer, as the “receiver,” visualizes this data in real time.

2. How It Works Together:

Arduino IDE is used to program the Arduino to listen to changes from the potentiometer and then send these values to the computer.

On the computer side, Python reads the data sent by the Arduino and plots it on a graph, enabling real-time visualization.

3. Data Flow:

Test Setup: The potentiometer connected to the Arduino senses varying input values as it is adjusted. These values are read by the Arduino and sent as data to the computer.

Display: Python receives this data and continuously updates a graph to show the changes as a moving line.

4. Understanding:

The goal of this part is to evaluate how effectively and accurately the Arduino can read data from a sensor and transmit it to the computer for real-time monitoring.

PROCEDURE

1. Set Up the Arduino:

- Connect the potentiometer to the Arduino: one side of the knob to 5V, the other to GND, and the middle to an input pin.
- Connect the light and resistor to another pin on the Arduino.

2. Writing Commands for the Arduino:

- Program the Arduino to read numbers from the knob and send them out for the computer to pick up.

3. Computer Program Setup:

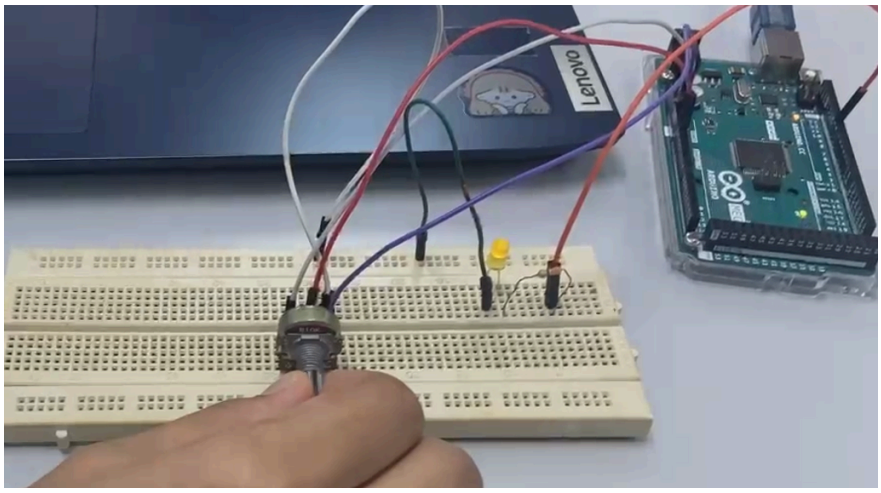
- Write a simple Python script that opens a channel to receive the numbers from the Arduino.
- Use a basic plotting tool to show these numbers on a graph.

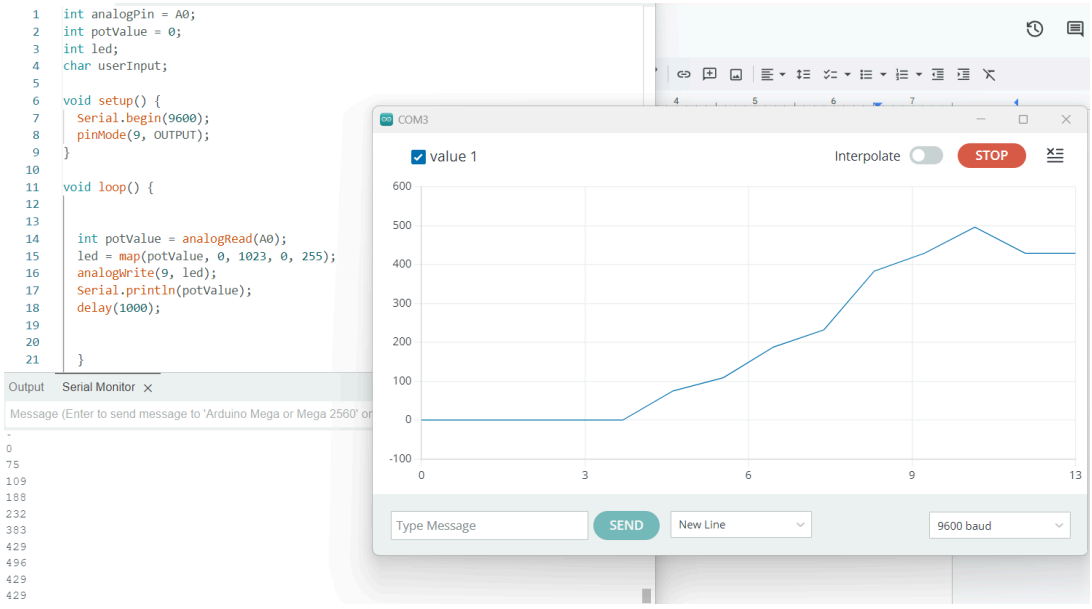
4. Starting the Test:

- Run both setups so the computer can show the knob's numbers in real-time on a graph.

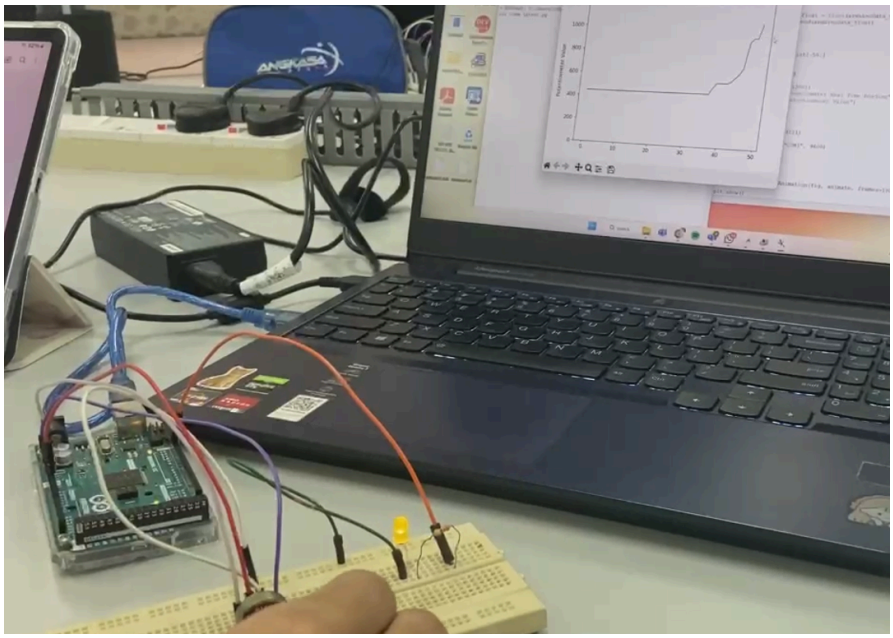
RESULTS

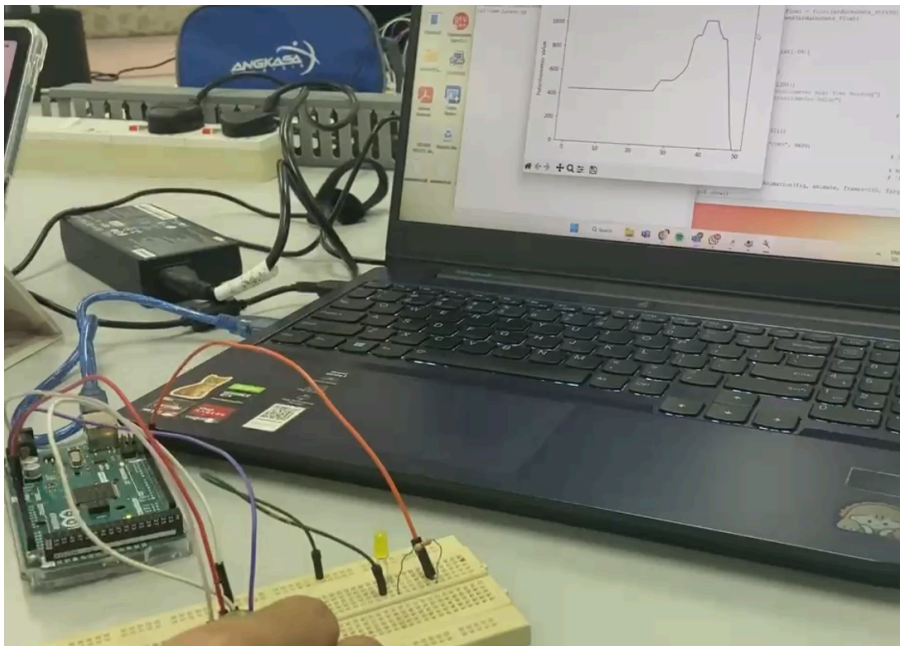
Reading the data using Arduino Serial Plotter :





Reading the data using Python :

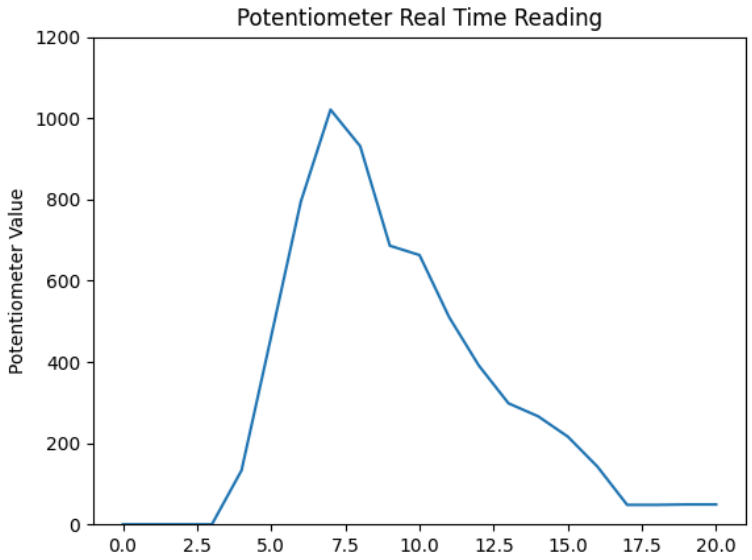




DISCUSSION

We enhance our coding in Python to present potentiometer readings graphically by using the matplotlib library. Firstly, we import the matplotlib library in our coding. Then, we convert the Arduino data point to float. Then, we design the plotting for the graph. For this project, we design the limit for the x-axis and y-axis, the title of the graph, and the label on the y-axis. For the real-time plotting, we utilise the Matplotlib Animation function.

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

def animate(i, dataList, ser):
    ser.write(b'g')                                     # Char 'g' to receive the Arduino data point
    arduinoData_string = ser.readline().decode('ascii') # Decode receive Arduino data as a formatted string


    try:
        arduinoData_float = float(arduinoData_string)   # Convert to float
        dataList.append(arduinoData_float)              # Add to the list holding the fixed number of points to animate

    except:
        pass

    dataList = dataList[-50:]                           # List size

    ax.clear()                                          # Clear last data frame
    ax.plot(dataList)                                   # Plot new data frame

    ax.set_ylim([0, 1200])
    ax.set_title("Potentiometer Real Time Reading")
    ax.set_ylabel("Potentiometer Value")

dataList = []                                           # Create list variable

fig = plt.figure()
ax = fig.add_subplot(111)

ser = serial.Serial("COM3", 9600)
time.sleep(2)                                           # Time delay

                                                        # Matplotlib Animation Fuction takes care of real time plot.
                                                        # 'fargs' parameter is where we pass in our dataList and Serial object.

ani = animation.FuncAnimation(fig, animate, frames=100, fargs=(dataList, ser), interval=100)

plt.show()
ser.close()
```



- **Software**

```
1   int analogPin = A0;
2   int potValue = 0;
3   int led;
4   char userInput;
5
6   void setup() {
7     Serial.begin(9600);
8     pinMode(9, OUTPUT);
9   }
10
11  void loop() {
12
13
14    int potValue = analogRead(A0);
15    led = map(potValue, 0, 1023, 0, 255);
16    analogWrite(9, led);
17    Serial.println(potValue);
18    delay(1000);
19
20
21    }
```

Output Serial Monitor ×

Message (Enter to send message to 'Arduino Mega or Mega 2560'

```
1002
1002
953
802
430
110
1
```

- **Electrical**

An Arduino Mega and a potentiometer were interfaced to create a working circuit in this experiment's electrical setup. By twisting its knob, the potentiometer's resistance may be changed, acting as a variable resistor. This resistance fluctuation was essential since it made it easier to generate analog voltage signals that corresponded to the potentiometer knob's position.

The potentiometer was connected to the Arduino board in a simple configuration: one leg was linked to the 5V power supply, another to the ground (GND), and the middle leg (wiper) to an analog input pin, in this case, A0. With this configuration, a voltage divider circuit was produced, and depending on the position of the potentiometer, the analog voltage at the middle leg varied between 0 and 5 volts.

After receiving data from the potentiometer, the Arduino that we used with DAQ will transform the analog signal from the sensor into a digital signal so that the laptop can show the data.

- **Hardware**

In an Arduino-based experiment involving serial communication, each component has a specific function that contributes to the setup. The Arduino board acts as the central microcontroller, processing data and handling communication with a computer or another device via serial communication. The potentiometer serves as an analog input device, allowing variable resistance that can adjust the voltage level read by the Arduino’s analog input pin; this value can be sent via serial communication to observe changes in real-time. Jumper wires connect various components on the breadboard and facilitate

electrical connections between the Arduino and the other components. The LED provides visual feedback, lighting up based on commands or analog input values to indicate when certain thresholds are met. The 220-ohm resistor prevents excessive current from damaging the LED by limiting its flow. Last but not least, the breadboard arranges and joins parts without the need for soldering, enabling a rapid and adaptable setup to test the circuit and monitor communication results. These parts work together to create an organised circuit that enables data measurement, display, and communication.

CONCLUSION

In conclusion, combining an Arduino with a potentiometer and using Python or the Arduino Serial Plotter to display the data yields useful information for a variety of projects. We can successfully read and show real-time potentiometer readings by following the instructions listed, whether we use the Serial Plotter for graphical representation or a Python script for data processing.

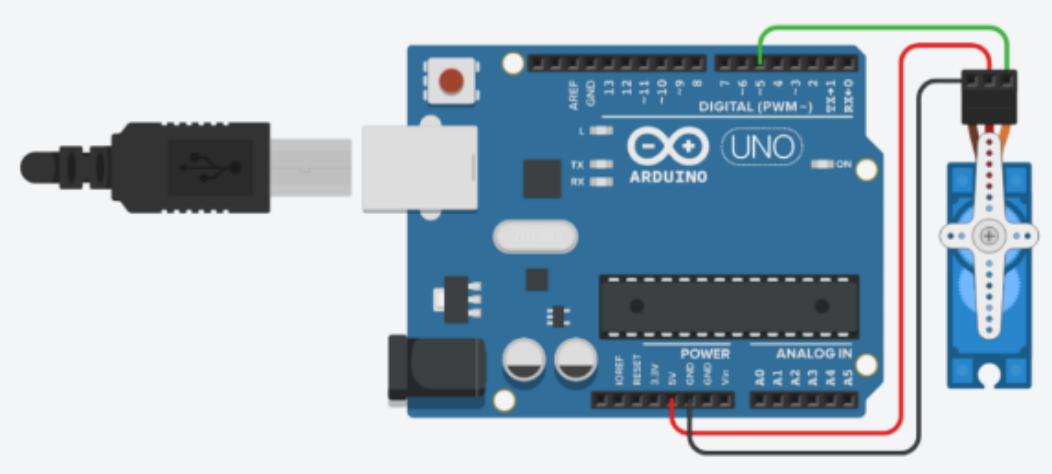
For a smooth connection, Python and the Serial Plotter must not access the serial port at the same time. This configuration makes it a flexible tool for learning and developing programming and electronics skills, opening the door to a variety of applications from basic experiments to complex control systems. We can further improve our data visualization and make it fit the requirements of your particular project by adjusting the plotter's parameters.

EXPERIMENT 3B - <u>Transmitting angle data from a Python script to an Arduino, which then actuates a servo to move to the specified angle.</u>

MATERIALS AND EQUIPMENT

- Arduino Uno board
- Servo motor
- Jumper wires

EXPERIMENTAL SETUP



METHODOLOGY

1. Purpose: The purpose of this test is to explore how a computer can control a device connected to an Arduino by sending it specific commands. Here, the computer acts as the “sender,” sending angle values to the Arduino, which controls a motor to match these angles.

2. How It Works Together:

In this setup, the computer uses Python to send angle commands to the Arduino.

The Arduino interprets these commands and adjusts the motor to move according to the specified angle.

3. Data Flow:

Test Setup: Python is programmed to send angle data to the Arduino based on user input or a predetermined pattern.

Motor Control: The Arduino receives the angle values and uses them to control the servo motor’s position, moving it to match the angle sent by the computer.

4. Understanding:

This part of the experiment assesses how smoothly and accurately the Arduino responds to incoming commands from the computer, as shown through the motor's movement in real-time.

PROCEDURE

1. Setting Up the Motor on the Arduino:

    - Attach the motor's power to the 5V on the Arduino, GND to GND, and the control to a different pin.

2. Commands for Motor Control:

    - Write commands for the Arduino to read angle numbers from the computer and use those numbers to adjust the motor's position.

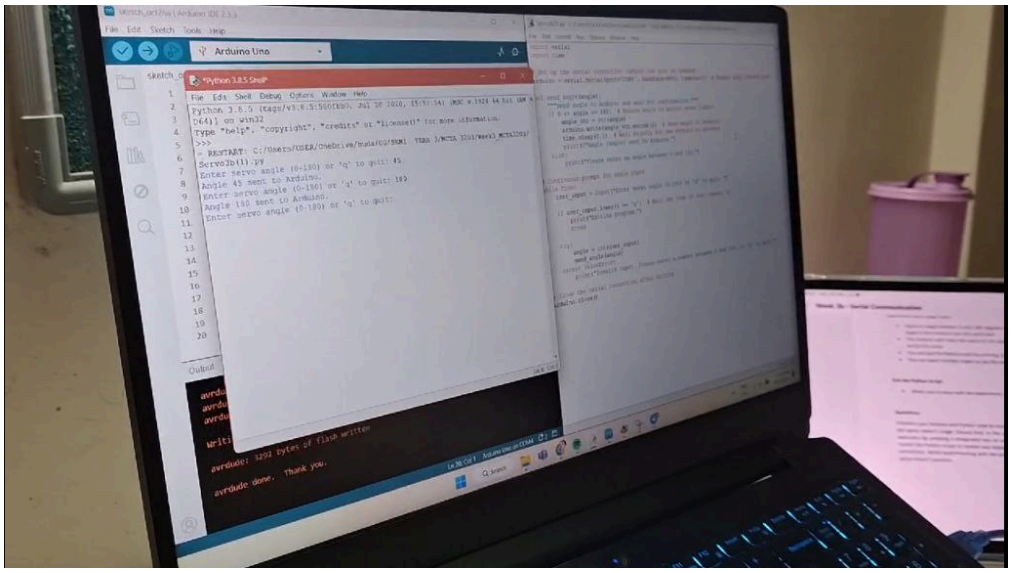
3. Computer Side Setup for Sending Angles:

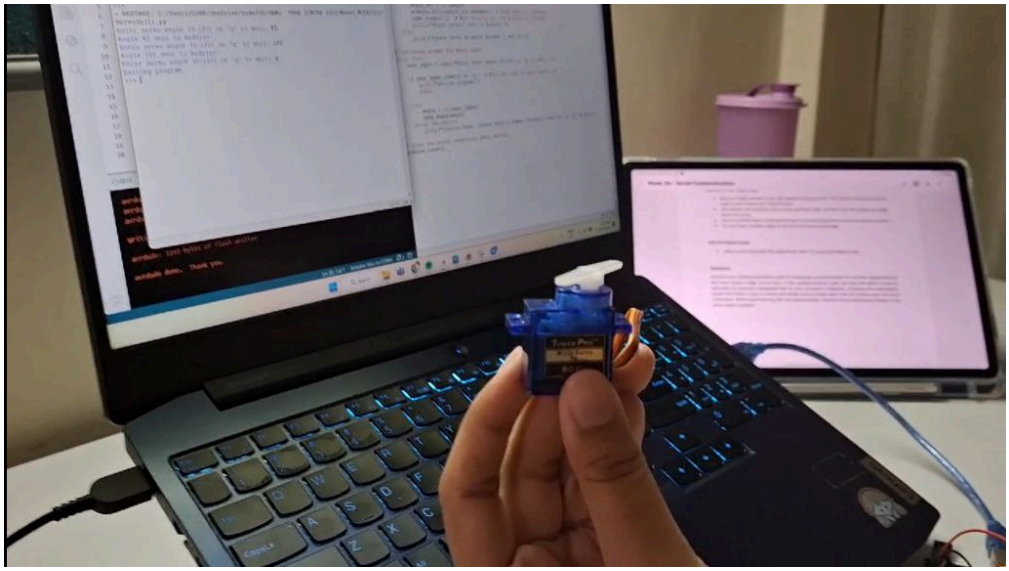
    - Create another Python script to send numbers to the Arduino based on user input or knob position.

4. Running the Setup:

    - Run both programs so the computer can send angles to the Arduino, which moves the motor as instructed.

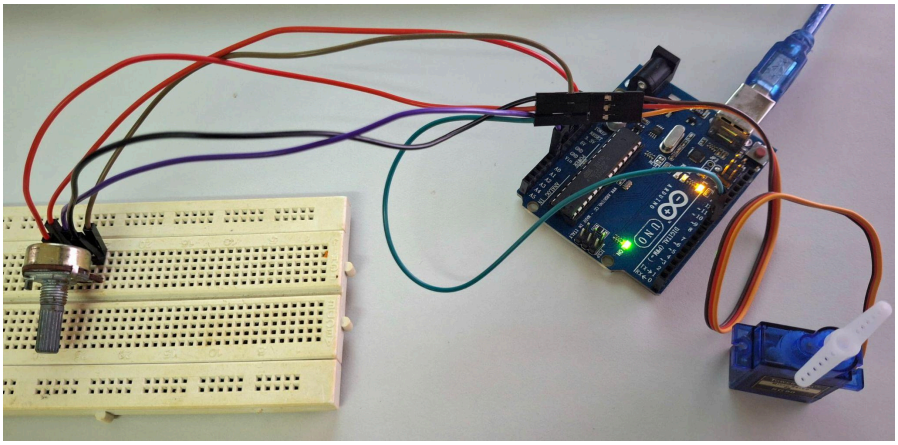
RESULTS


Arduino Uno
Python 3.8.5 Shell
File Edit Shell Debug Options Window Help
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:/Users/USER/OneDrive/Desktop/Servo1.py
ServoIo(1).py
Enter servo angle (0-180) or 'q' to quit: 45
Angle 45 sent to Arduino.
Enter servo angle (0-180) or 'q' to quit: 180
Angle 180 sent to Arduino.
Enter servo angle (0-180) or 'q' to quit:
avrdude: 4392 bytes of flash written
avrdude done. Thank you.



DISCUSSION

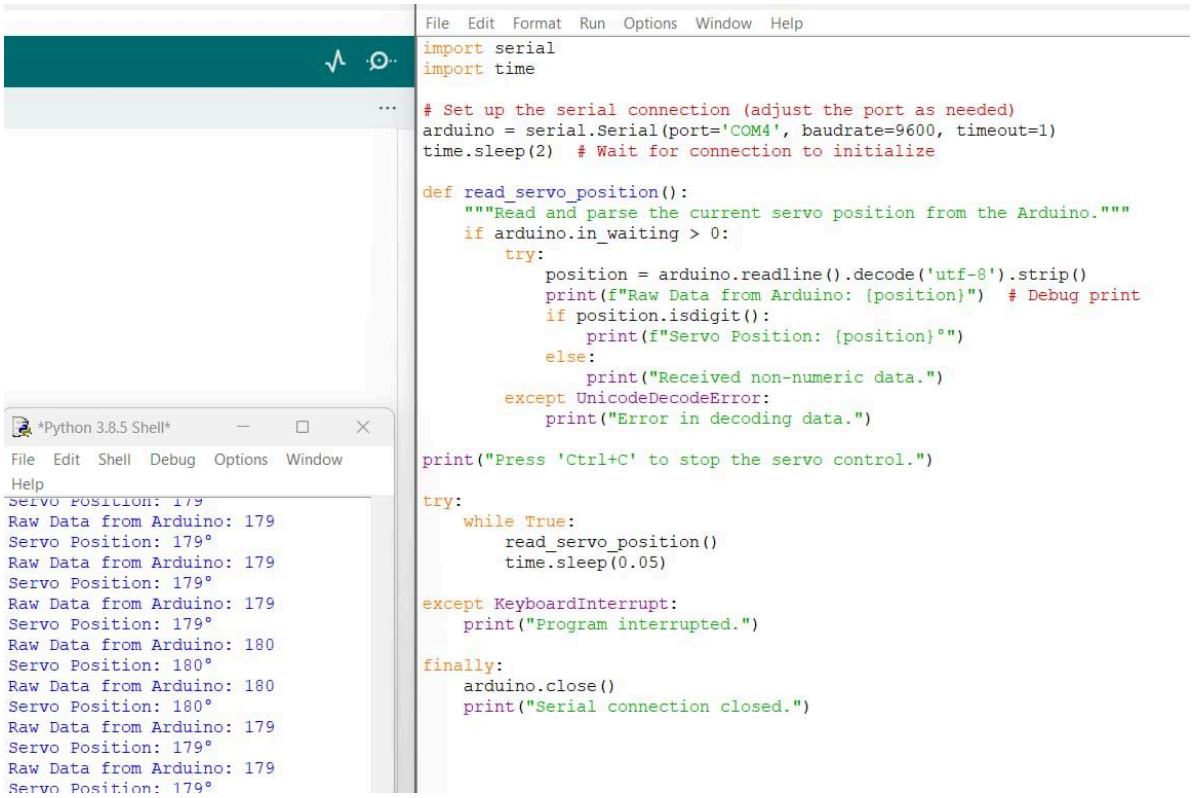
We are enhancing our Arduino and Python code to incorporate a potentiometer for real-time adjustments of the servo motor's angle. This is the new experimental set-up:



Reading the potentiometer value and sending the current angle to Python from the Arduino:

```
#include <Servo.h>

Servo myServo;
const int potPin = A0;
int potValue = 0;
int angle = 0;
int lastAngle = -1;

void setup() {
  Serial.begin(9600);
  myServo.attach(9);
}

void loop() {
  // Read potentiometer value
  potValue = analogRead(potPin);
  angle = map(potValue, 0, 1023, 0, 180);

  // Update servo only if the angle has changed significantly
  if (abs(angle - lastAngle) > 5) { // Deadband of 5 degrees for smoothing
    myServo.write(angle);
    lastAngle = angle;
  }

  // Send the current angle to Python
  Serial.println(angle);
  delay(50);  // Increased delay for stability
}

```

Receiving and displaying servo position data from the Arduino over the serial connection in Python:



```
File  Edit  Format  Run  Options  Window  Help
import serial
import time

# Set up the serial connection (adjust the port as needed)
arduino = serial.Serial(port='COM4', baudrate=9600, timeout=1)
time.sleep(2)  # Wait for connection to initialize

def read_servo_position():
    """Read and parse the current servo position from the Arduino."""
    if arduino.in_waiting > 0:
        try:
            position = arduino.readline().decode('utf-8').strip()
            print(f"Raw Data from Arduino: {position}")  # Debug print
            if position.isdigit():
                print(f"Servo Position: {position}°")
            else:
                print("Received non-numeric data.")
        except UnicodeDecodeError:
            print("Error in decoding data.")

print("Press 'Ctrl+C' to stop the servo control.")

try:
    while True:
        read_servo_position()
        time.sleep(0.05)

except KeyboardInterrupt:
    print("Program interrupted.")

finally:
    arduino.close()
    print("Serial connection closed.")
```

```
*Python 3.8.5 Shell*
File  Edit  Shell  Debug  Options  Window
Help
Servo Position: 179
Raw Data from Arduino: 179
Servo Position: 179°
Raw Data from Arduino: 179
Servo Position: 179°
Raw Data from Arduino: 179
Servo Position: 179°
Raw Data from Arduino: 180
Servo Position: 180°
Raw Data from Arduino: 180
Servo Position: 180°
Raw Data from Arduino: 179
Servo Position: 179°
Raw Data from Arduino: 179
Servo Position: 179°
```

- **Software**

```
import serial
import time

# Set up the serial connection (adjust the port as needed)
arduino = serial.Serial(port='COM4', baudrate=9600, timeout=1)  # Update with correct port

def send_angle(angle):
    """Send angle to Arduino and wait for confirmation."""
    if 0 <= angle <= 180:  # Ensure angle is within servo limits
        angle_str = str(angle)
        arduino.write(angle_str.encode())  # Send angle to Arduino
        time.sleep(0.1)  # Wait briefly for the Arduino to process
        print(f"Angle {angle} sent to Arduino.")
    else:
        print("Please enter an angle between 0 and 180.")

# Continuous prompt for angle input
while True:
    user_input = input("Enter servo angle (0-180) or 'q' to quit: ")

    if user_input.lower() == 'q':  # Exit the loop if user inputs 'q'
        print("Exiting program.")
        break

    try:
        angle = int(user_input)
        send_angle(angle)
    except ValueError:
        print("Invalid input. Please enter a number between 0 and 180, or 'q' to quit.")

# Close the serial connection after exiting
arduino.close()
```

```
Python 3.8.5 Shell
File  Edit  Shell  Debug  Options  Window  Help
Python 3.8.5 (tags/v3.8.5:580fbb0, Jul 20 2020, 15:57:54) [MSC v.1924 64 bit (AM
D64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:/Users/USER/OneDrive/huda/UG/SEM1  YEAR 3/MCTA 3203/Week3_MCTA3203/
Servo3b(1).py
Enter servo angle (0-180) or 'q' to quit: 45
Angle 45 sent to Arduino.
Enter servo angle (0-180) or 'q' to quit: 180
Angle 180 sent to Arduino.
Enter servo angle (0-180) or 'q' to quit: q
Exiting program.
```

- **Electrical**

In this experiment, the Arduino board acts as the main controller, processing commands sent from a Python script on your computer via a USB connection. This USB connection not only powers the Arduino board with 5V but also enables data communication. The servo motor, which allows precise control over its angular position, is connected to the Arduino via a jumper wire. The Arduino's 5V output pin supplies power to the Vcc pin of the servo motor, while the ground connection (GND) ensures a complete circuit. The control signal is sent through a PWM-capable pin on the Arduino, usually pin 9 or 10, which is connected to the servo motor's signal input. Based on the width of the pulses it receives, the Arduino's PWM (Pulse Width Modulation) signal instructs the servo the angle it needs to maintain.

When you enter an angle in a Python script, it is sent to the Arduino via a USB cable, acting as a serial command. The Arduino then reads this command, generates a PWM signal corresponding to the specified angle, and sends it to the servo motor. The internal circuit of the motor interprets this PWM signal and moves to the specified angle. To end the session, the Python script closes the serial connection when you enter 'q', ending the control session and ensuring that the circuit is no longer active. This setup shows how electrical connections and commands from software can work together to dynamically control hardware.

- **Hardware**

In this experiment, the hardware setup includes an Arduino board, a servo motor, jumper wires, a USB cable, and a computer. The Arduino board is the core component, responsible for interpreting commands and controlling the servo motor’s angle. The servo motor, designed for

precise angular movement, has three main connections: Vcc (for power), GND (for grounding), and a signal pin that receives the PWM control signal from the Arduino. These connections are made using jumper wires, with the Arduino's 5V output pin supplying power to the servo's Vcc pin, and the GND pin completing the circuit by connecting to the servo's GND pin.

The USB cable connects the Arduino to the computer, supplying power and facilitating data transfer. The computer runs a Python script that allows you to input an angle, which is then sent over the USB connection to the Arduino. Inside the Arduino, this command is processed, and it generates the appropriate PWM signal to move the servo motor to the specified angle. The interaction between the computer, Arduino, and servo motor showcases how hardware components, such as the microcontroller, motor, and power connections, work in tandem to create a controlled motion system. This setup is simple yet powerful, allowing easy control over hardware through minimal components and clear data pathways.

CONCLUSION

The summary, which highlights the effective collaboration between hardware and software components, aligns well with the detailed description of the experiment. The Arduino board functions as the primary controller to accurately manage the movements of the servo motor. It receives angle commands from the Python script and produces corresponding PWM signals. This interaction illustrates the integration of programming, data connectivity, and electrical connections to achieve dynamic control of hardware.

The hardware setup, comprising the servo motor, Arduino, and necessary connections, provides a solid basis for understanding the function of each component within the overall system. This section enhances the educational value of the project by emphasizing the

simplicity and efficiency of utilizing a USB connection for both power supply and data transmission. It improves the understanding of concepts such as PWM and serial transmission while allowing for hands-on experimentation.

In summary, this setup not only demonstrates valuable applications in robotics and automation but also offers a comprehensive insight into embedded system principles, highlighting the collaboration between hardware and software to achieve controlled movement.

RECOMMENDATIONS

The findings from the experiment suggest the necessity of pursuing additional projects that enhance the foundational skills established in this framework. To facilitate the development of more complex interactions, it would be beneficial to incorporate additional sensors, such as temperature or ultrasonic distance sensors. For instance, we could program the Arduino to adjust the servo's movement in response to sensor data reflecting environmental changes.

Moreover, documenting our experiments and outcomes could reinforce the insights gained and provide valuable recommendations for future investigations. This hands-on experience not only solidifies theoretical concepts but also cultivates essential problem-solving skills pertinent to the fields of robotics and electronics. Overall, the project lays a robust groundwork for further exploration of embedded systems and fosters creativity and innovation in engineering endeavours**.**

REFERENCES

"Arduino Integrated Development Environment (IDE) v1 | Arduino Documentation | Arduino Documentation," *Arduino.cc*, 2022. https://docs.arduino.cc/software/ide-v1/tutorials/arduino-ide-v1-basics/

ACKNOWLEDGEMENT

We would like to express our gratitude to Dr. Wahju Sediono, Dr. Ali Sophian, Dr. Zulkifli bin Zainal Abidin, my teaching assistant, and my peers for their invaluable help and support in finishing this project. Their advice, feedback, and experience have greatly influenced the level of quality and understanding of this work.

STUDENT'S DECLARATION

Certificate of Originality and Authenticity

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 has 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.**

Signature: 	**Read**	/
Name: AKMAL ZARIF BIN NAJIB	**Understand**	/
Matric No: 2211971	**Agree**	/

Signature: 	**Read**	/
Name: AMIR FARHAN BIN GHAFFAR	**Understand**	/
Matric No: 2115617	**Agree**	/

Signature: 	**Read**	/
Name: AMIRAH HUDA BINTI JAMALULAIL ASRI	**Understand**	/
Matric No: 2210776	**Agree**	/

Signature: 	**Read**	/
Name: AMIRUL AIZAD BIN AMIR	**Understand**	/
Matric No: 2211263	**Agree**	/

Signature: 	**Read**	/
Name: ANAS BIN BADRULZAMAN	**Understand**	/
Matric No: 2219945	**Agree**	/