LABORATORY REPORT

LAB 2: SERIAL COMMUNICATION

GROUP B

PROGRAMME: MECHATRONIC ENGINEERING

GROUP MEMBERS:	MATRIC NO:
1. MUHAMMAD AFIQ BIN AHMAD	2115203
2. HANIS BINTI MOHD IZANI	2020590
3. SARAH AISHAH BINTI SA'AID HAZLEY	2117600
4. NUR AYU NAZIERA BINTI ROSLI	2119202
5. NOORUL GHOUSIAH BINTI NOORDEEN SAHIB	2118298

DATE OF SUBMISSION:
Wednesday, 1st November 2023

ABSTRACT

TABLE OF CONTENTS

Table of Contents

INTRODUCTION

The experiment consists of a hardware setup by connecting a potentiometer to an Arduino board, and the potentiometer's analog values are transmitted to a Python script for processing. By carefully following the steps outlined in this experiment, we establish a robust communication link that allows us to continuously monitor and graphically visualize potentiometer readings in real-time. We used the matplotlib library to create dynamic graphs that represent the evolving data from the potentiometer.

PROCEDURE

PART A

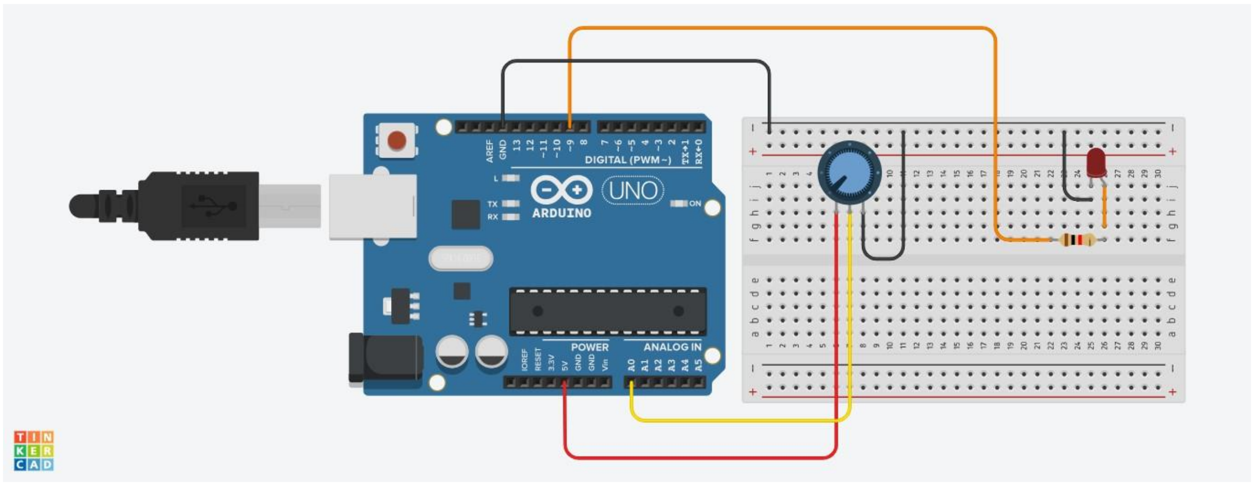
Materials And Equipment

- Arduino Uno Board
- Potentiometer
- Jumper Wires
- LED
- 220Ω resistor
- Breadboard

Experimental Setup

1. One leg of the potentiometer is connected to 5V on the Arduino.
2. The other leg of the potentiometer is connected to GND on the Arduino.
3. The middle leg (wiper) of the potentiometer is connected to an analog input pin on the Arduino, that is A0.
4. One LED is connected to the circuit by connecting the cathode leg to the ground and the anode leg to the digital pin 9 through a 220Ω resistor.

The circuit setup is as shown:



The Arduino code:

```
int x;
void setup() {
  Serial.begin(9600);
  pinMode(9, OUTPUT);
}
void loop() {
  int potValue = analogRead(A0);
  Serial.println(potValue);


  if (potValue > 200) {
    digitalWrite(9, HIGH);
  } else {
    digitalWrite(9, LOW);
    }
  delay(1000);
}
```

The Python code:

```
import serial

ser = serial.Serial('COM5', 9600)

try:
 while True:
    pot_value = ser.readline().decode().strip()
    print("Potentiometer Value:", pot_value)

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

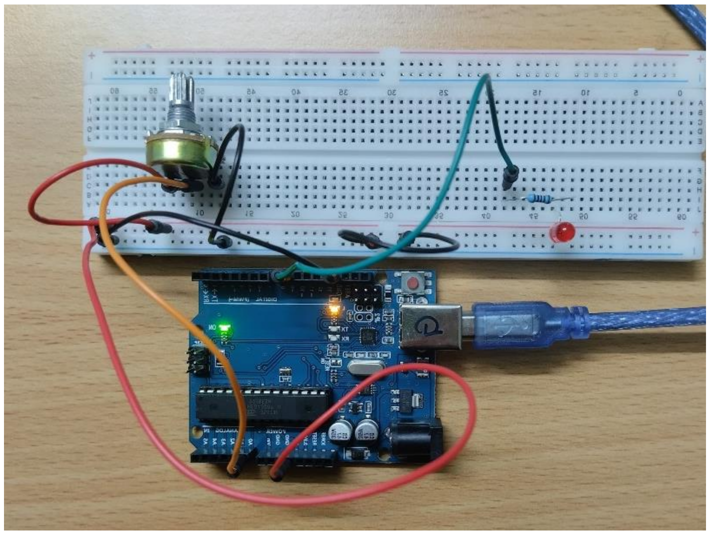
<u>Methodology</u>

1. The Arduino board is connected to your computer via a USB cable.
2. The Arduino board is powered on. The correct COM port and baud rate is selected.
3. The coding sketch is uploaded to the board using the Arduino IDE.
4. The Python script is run on the computer.
5. As the potentiometer knob turns, the potentiometer readings displayed in the Python terminal is observed.
6. The Python terminal is closed, and then, the serial plotter in Arduino IDE is opened.
7. The real-time potentiometer readings in the serial plotter are observed.

RESULTS

PART A

The following picture is the circuit set up:

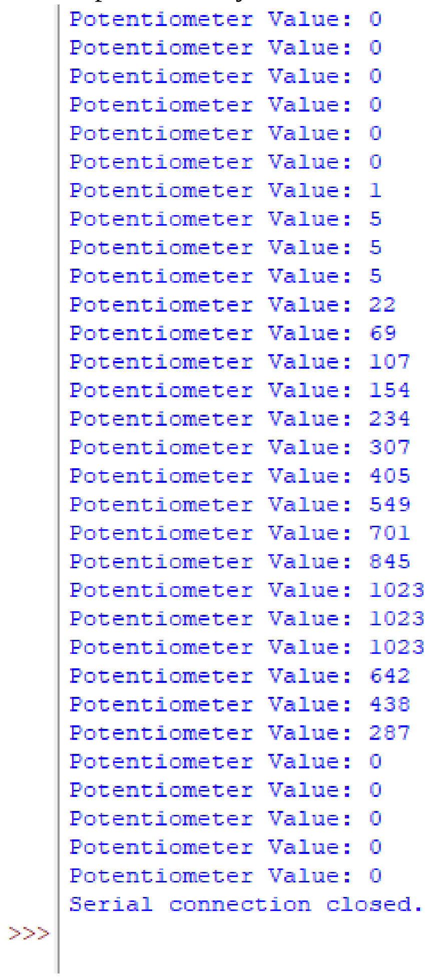


The serial communication between Python and Arduino for data exchange is established successfully.

When the knob of the potentiometer turns, the Python terminal displays the real-time value of the potentiometer. At the same time, LED turns on whenever the potentiometer value exceeds 200. Otherwise, the LED is turned off.

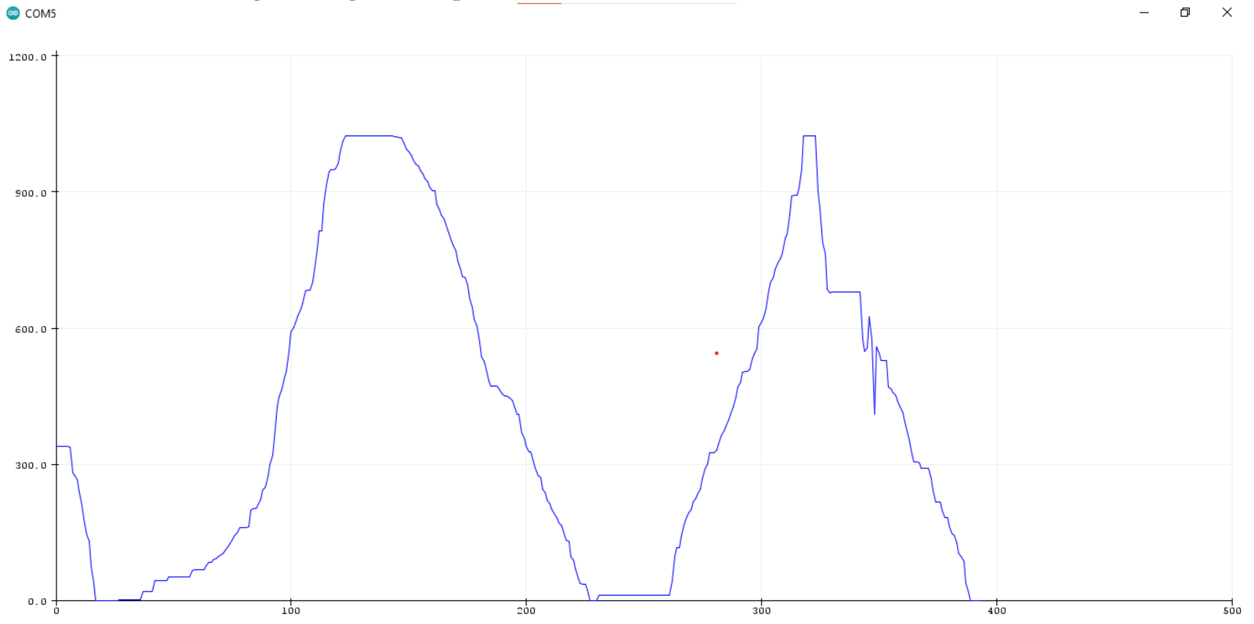
The real-time data of the potentiometer readings is also viewed graphically through the serial plotter of Arduino IDE.

The output on the Python terminal:



```
   Potentiometer Value: 0
   Potentiometer Value: 0
   Potentiometer Value: 0
   Potentiometer Value: 0
   Potentiometer Value: 0
   Potentiometer Value: 0
   Potentiometer Value: 1
   Potentiometer Value: 5
   Potentiometer Value: 5
   Potentiometer Value: 5
   Potentiometer Value: 22
   Potentiometer Value: 69
   Potentiometer Value: 107
   Potentiometer Value: 154
   Potentiometer Value: 234
   Potentiometer Value: 307
   Potentiometer Value: 405
   Potentiometer Value: 549
   Potentiometer Value: 701
   Potentiometer Value: 845
   Potentiometer Value: 1023
   Potentiometer Value: 1023
   Potentiometer Value: 1023
   Potentiometer Value: 642
   Potentiometer Value: 438
   Potentiometer Value: 287
   Potentiometer Value: 0
   Potentiometer Value: 0
   Potentiometer Value: 0
   Potentiometer Value: 0
   Potentiometer Value: 0
   Serial connection closed.
>>>
```

Potentiometer readings through serial plotter of Arduino:

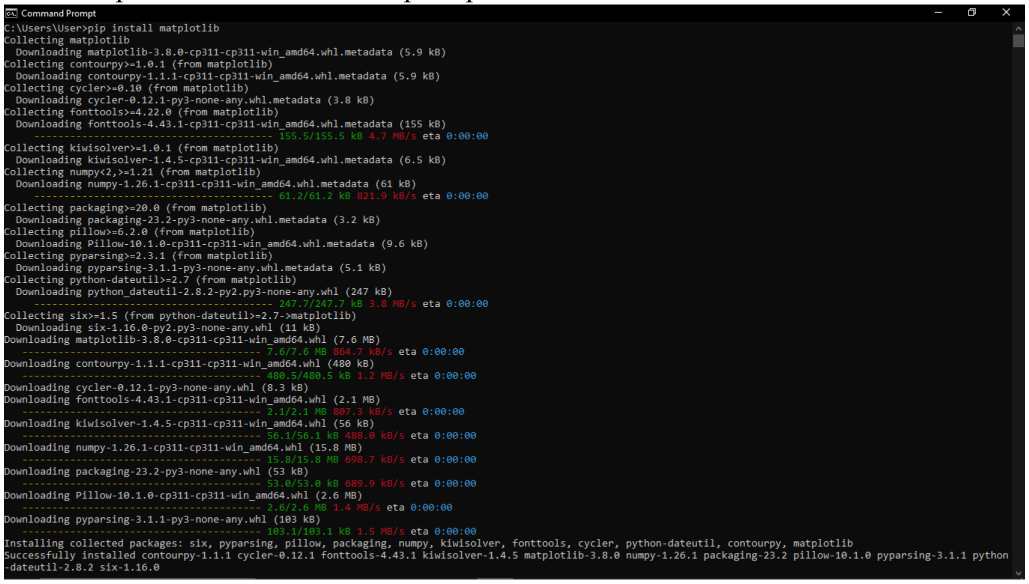


DISCUSSION

PART A

Matplotlib is graphing and data visualization library for Python programming language and its numerical mathematics extension NumPy. It provides an object-oriented API for embedding plots into applications using general-purpose GUI toolkits like Tkinter, wxPython, Qt, or GTK. These are the step-by-step of how to do it.

1. First thing first, make sure matplotlib has been installed. If not, we can install it using pip. Type 'pip install matplotlib' in the command prompt.



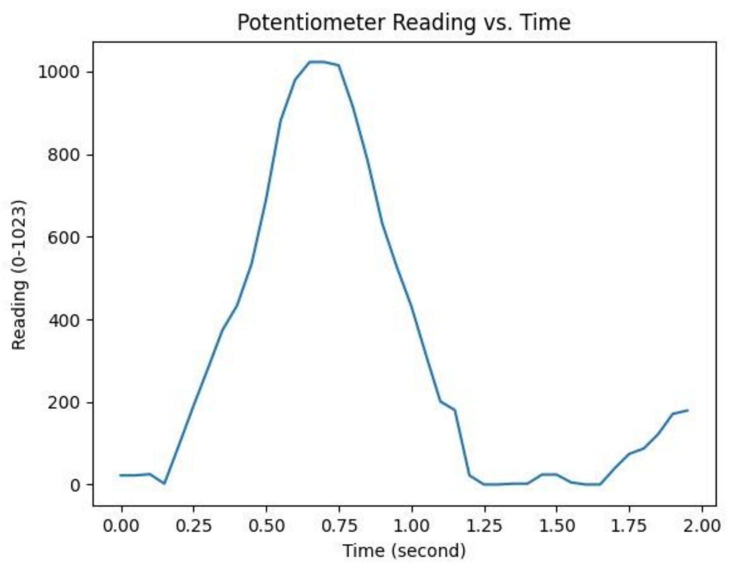
```
C:\Users\User>pip install matplotlib
Collecting matplotlib
  Downloading matplotlib-3.8.0-cp311-cp311-win_amd64.whl.metadata (5.9 kB)
Collecting contourpy>=1.0.1 (from matplotlib)
  Downloading contourpy-1.1.1-cp311-cp311-win_amd64.whl.metadata (5.9 kB)
Collecting cycler>=0.10 (from matplotlib)
  Downloading cycler-0.12.1-py3-none-any.whl.metadata (3.8 kB)
Collecting fonttools>=4.22.0 (from matplotlib)
  Downloading fonttools-4.43.1-cp311-cp311-win_amd64.whl.metadata (155 kB)
     ---------------------------------------- 155.5/155.5 kB 4.7 MB/s eta 0:00:00
Collecting kiwisolver>=1.0.1 (from matplotlib)
  Downloading kiwisolver-1.4.5-cp311-cp311-win_amd64.whl.metadata (6.5 kB)
Collecting numpy<2,>=1.21 (from matplotlib)
  Downloading numpy-1.26.1-cp311-cp311-win_amd64.whl.metadata (61 kB)
     ---------------------------------------- 61.2/61.2 kB 821.9 kB/s eta 0:00:00
Collecting packaging>=20.0 (from matplotlib)
  Downloading packaging-23.2-py3-none-any.whl.metadata (3.2 kB)
Collecting pillow>=6.2.0 (from matplotlib)
  Downloading Pillow-10.1.0-cp311-cp311-win_amd64.whl.metadata (9.6 kB)
Collecting pyparsing>=2.3.1 (from matplotlib)
  Downloading pyparsing-3.1.1-py3-none-any.whl.metadata (5.1 kB)
Collecting python-dateutil>=2.7 (from matplotlib)
  Downloading python_dateutil-2.8.2-py2.py3-none-any.whl (247 kB)
     ---------------------------------------- 247.7/247.7 kB 3.8 MB/s eta 0:00:00
Collecting six>=1.5 (from python-dateutil>=2.7->matplotlib)
  Downloading six-1.16.0-py2.py3-none-any.whl (11 kB)
Downloading matplotlib-3.8.0-cp311-cp311-win_amd64.whl (7.6 MB)
   ---------------------------------------- 7.6/7.6 MB 864.7 kB/s eta 0:00:00
Downloading contourpy-1.1.1-cp311-cp311-win_amd64.whl (480 kB)
   ---------------------------------------- 480.5/480.5 kB 1.2 MB/s eta 0:00:00
Downloading cycler-0.12.1-py3-none-any.whl (8.3 kB)
Downloading fonttools-4.43.1-cp311-cp311-win_amd64.whl (2.1 MB)
   ---------------------------------------- 2.1/2.1 MB 807.3 kB/s eta 0:00:00
Downloading kiwisolver-1.4.5-cp311-cp311-win_amd64.whl (56 kB)
   ---------------------------------------- 56.1/56.1 kB 488.0 kB/s eta 0:00:00
Downloading numpy-1.26.1-cp311-cp311-win_amd64.whl (15.8 MB)
   ---------------------------------------- 15.8/15.8 MB 698.7 kB/s eta 0:00:00
Downloading packaging-23.2-py3-none-any.whl (53 kB)
   ---------------------------------------- 53.0/53.0 kB 689.9 kB/s eta 0:00:00
Downloading Pillow-10.1.0-cp311-cp311-win_amd64.whl (2.6 MB)
   ---------------------------------------- 2.6/2.6 MB 1.4 MB/s eta 0:00:00
Downloading pyparsing-3.1.1-py3-none-any.whl (103 kB)
   ---------------------------------------- 103.1/103.1 kB 1.5 MB/s eta 0:00:00
Installing collected packages: six, pyparsing, pillow, packaging, numpy, kiwisolver, fonttools, cycler, python-dateutil, contourpy, matplotlib
Successfully installed contourpy-1.1.1 cycler-0.12.1 fonttools-4.43.1 kiwisolver-1.4.5 matplotlib-3.8.0 numpy-1.26.1 packaging-23.2 pillow-10.1.0 pyparsing-3.1.1 python
-dateutil-2.8.2 six-1.16.0
```

2. Next is to create a simple line graph to visualize the reading that we want. This is a python script that we use,

```
from time import sleep
import serial

serialport="COM5"
baudrate=9600
dt=0.05

#open port
ser = serial.Serial(serialport, baudrate )
if ser.is_open :
    print(f"Serial port {ser.name} is open.")
    data =[]
    N=40

    for k in range(N):
        b = ser.readline()
        strn=b.decode()
        strl =strn.rstrip()
        flt = float(strl)
        print("Potentiometer Value:", flt)
        data.append(flt)
        sleep(dt)

#close port
ser.close()
if not ser.is_open :
    print(f"Serial port {ser.name} is closed")

#plot reading
import matplotlib.pyplot as plt
import numpy as np

#x = (0:(N-1))*dt
x = np.arange (0, N*dt, dt)
plt.plot( x, data)
plt.xlabel('Time (second)')
plt.ylabel('Reading (0-1023)')
plt.title('Potentiometer Reading vs. Time' )
plt.show()

exit()
```

‘Readings’ represents the potentiometer readings, and ‘x’ represents the corresponding x-axis values that our data can be placed. The ‘plot’ function is used to create the line graph with markers and can customize it with various options.

3. After we run the code, it will show the plotting. This code will display a simple line graph of the potentiometer readings using **matplotlib**. We can adjust the labels, titles, and other settings to suit our specific data and requirements.



Link to video:
https://youtu.be/WIJxwEoloQo?si=3TtdvT6a2VlN--yI

PART B

INTRODUCTION

The experiment begins with a hardware setup that involves connecting a servo motor to an Arduino board. The Arduino board serves as the intermediary between the user's input and the servo motor's movements. In addition, the setup allows for manual angle input via a potentiometer, which allows for an interactive method for controlling the servo's position. The Python script serves as the user interface and allows for real-time servo control. It allows users to input specific angles for the servo to move to.

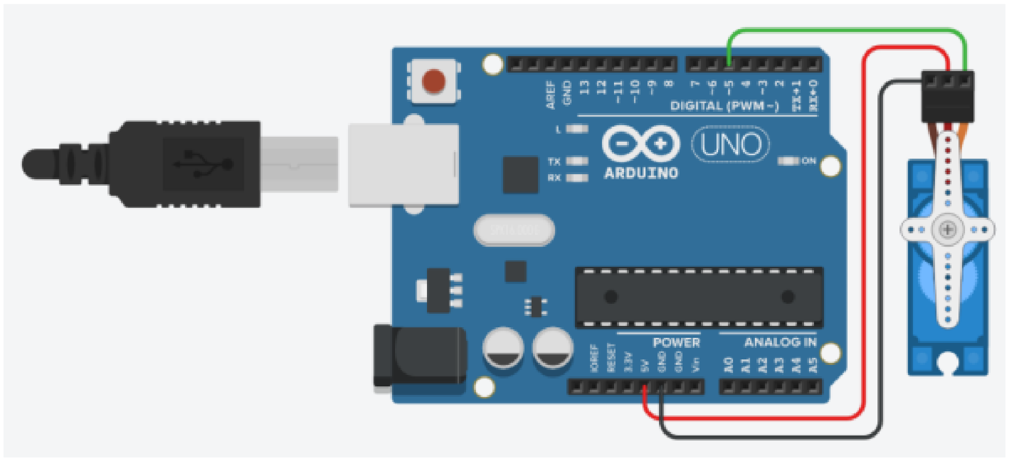
Materials And Equipment

- Arduino Uno Board
- Servo motor
- Jumper wires
- Potentiometer
- USB cable for Arduino
- Computer with Arduino IDE and Python installed

Experimental Setup

1. Servo's Signal Wire is connected.
2. Servo is powered using 5V and GND pins in Arduino.
3. Servo's Ground Wire is connected to GND.
4. One leg of the potentiometer is connected to 5V on the Arduino.
5. The other leg of the potentiometer is connected to GND on the Arduino.
6. The middle leg (wiper) of the potentiometer is connected to an analog input pin on the Arduino, that is A0.
7. Servo library is installed.
8. Pyserial library using pip is installed.

The circuit setup is as shown:



METHODOLOGY

1. The circuit was built according to the circuit setup instructions.
2. The Arduino code was uploaded to our board.

3. Write the python code and then run the programme.
4. The following was the code that we used in this experimental setup :

The Arduino code:

```
#include <Servo.h>
Servo servo;
int servoPin = 9; //initiate pin on arduino
int angle = 90; //inital angle

void setup() {
  servo.attach(servoPin);
  servo.write(angle);
  Serial.begin(9600);
}


void loop() {
  if (Serial.available() > 0) {
    angle = Serial.parseInt(); //parseInt reads characters from the serial input until it encounters a non
numeric charcter, then it converts the collected character into intigers
    servo.write(angle);
    delay(15);  // Add a small delay for smoother motion
  }
}
```

The python code:

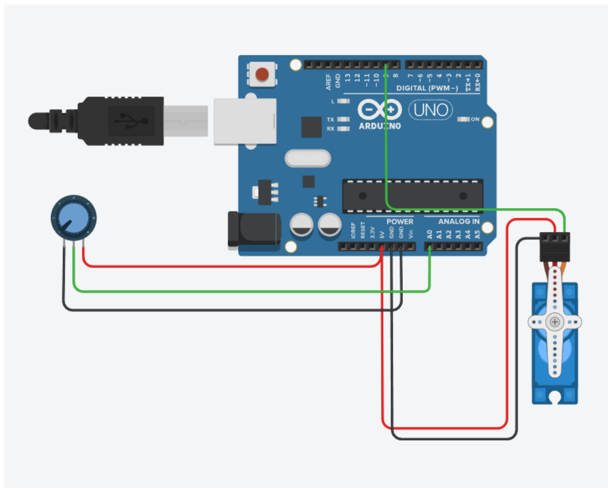
```
import serial
import time
```

```
ser = serial.Serial('COM9', 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.")
```

Link to video:

https://youtube.com/shorts/5YJvdQpr2PQ?si=jVHqOe3GOILlLhWp

PART B PART 2:
The circuit setup is shown as below:



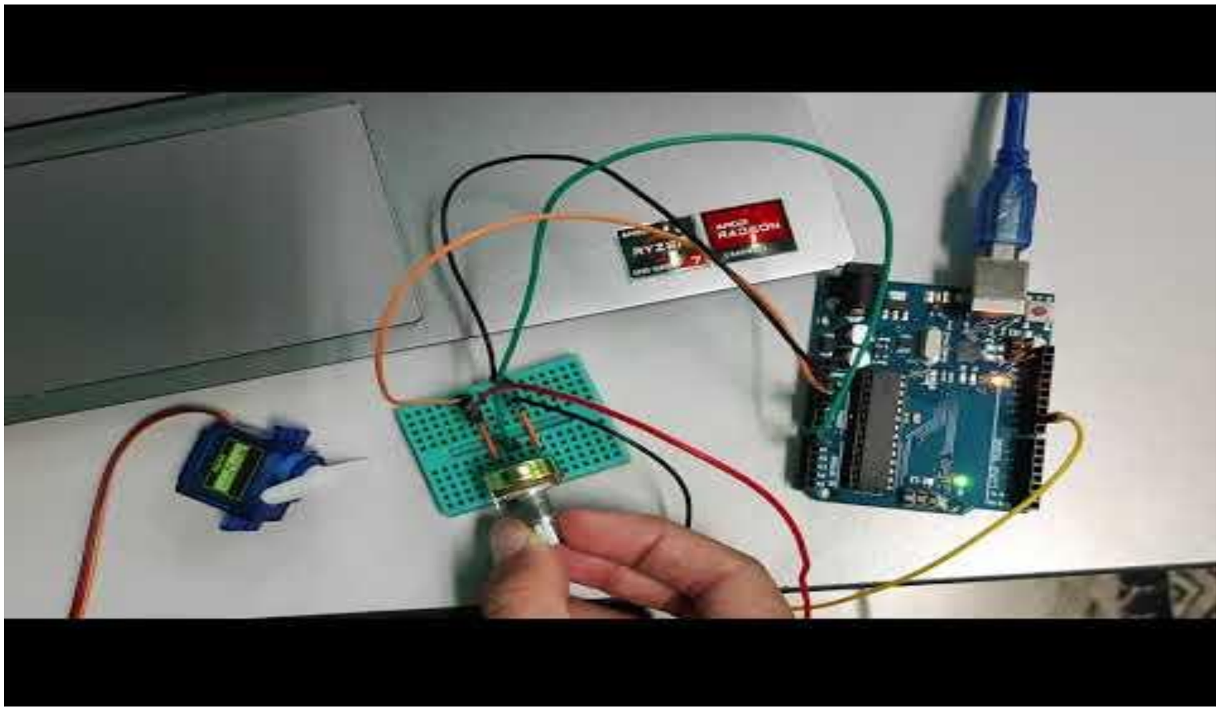
The Arduino Code:

```
#include <Servo.h>
Servo servo;
const int potentiometerPin = A0;  // Analog pin for the potentiometer
int potentiometerValue = 0;
int servoAngle = 0;  // Initial servo angle
```

```
char userInput = ' ';  // Variable to store user input

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

void loop() {
  // Read potentiometer value
  potentiometerValue = analogRead(potentiometerPin);

  // Map potentiometer value to servo angle (0-180 degrees)
  servoAngle = map(potentiometerValue, 0, 1023, 0, 180);


  // Control the servo
  servo.write(servoAngle);

  // Check for user input
  if (Serial.available() > 0) {
    userInput = Serial.read();

    if ((userInput >= 'A' && userInput <= 'Z') || (userInput >= 'a' && userInput <= 'z')) {
      // Stop the servo when a letter is received
      servo.write(0);  // Set the servo to the center position (90 degrees)
    }
  }
}
```

Link to video:

https://youtu.be/N76H668sHmw?si=TH_tMhQ2RwkIGBQ4



DISCUSSION

From the experiment above, we can see that the servo motor can be controlled by the potentiometer only using Arduino. To do this, we have to use a map function which resizes the potentiometer value to only be between 0 and 180 as this is the only acceptable value for the servo motor. So here, we have learnt the function of the map function.

RESULTS

The input angle that users enter in the Python prompt, ranging from 0 to 180, determines how the servo motor moves. Through the serial interface, the Python software can send the angle to the Arduino. After getting angle commands, the servo motor moves in the desired direction and stays there till the user gives new commands. This method makes it easier to operate the servo motor step-by-step, which enables accurate placement and regulated movements.

CONCLUSION

In conclusion, we were able to establish serial communication between Python and an Arduino. Then we were able to use the knowledge to transmit angle data from a Python script to an Arduino, which then actuates the servo to move to the specified angle. It shows that these experiments require a good understanding of both the hardware and software involved. Now, we have a better understanding of using python and Arduino. Also, we were able to understand how to use matplotlib in our Python script.

RECOMMENDATIONS

While carrying out the experiment, our group faced a problem with our python code, as we were not able to do the correct code to run the potentiometer. At first, we were not sure where the error was in the code. Then we finally found it when we asked our lecturer. In the future, a good understanding of python is important to facilitate the journey during conducting experiments. This would save time and allow us to finish both experiments of serial communication between Python and an Arduino in class time.

For the second part, we encounter problems in moving the servo motor. Fortunately, we know the cause of the problem and manage to fix it. For future reference, we will implement error handling in the Python script to handle unexpected situations like this. In addition, the setup needs to be tested and the servo motor needs to be calibrated to ensure it moves accurately to the desired positions.

REFERENCES

Arduino (2020, November 6). *Serial Communication between Python and Arduino*. Project Hub. Retrieved October 27, 2023, from https://projecthub.arduino.cc/ansh2919/serial-communication-between-python-and-arduino-663756

Stefan Maetschke (2020, November 6). *Positional versus Continuous Servos*. Maker Guides. Retrieved October 27, 2023, from https://www.makerguides.com/positional-vs-continuous-sg90-servos/

Nikhil Kumar (2023) *Graph plotting in Python: Set 1*, *GeeksforGeeks*. Available at: https://www.geeksforgeeks.org/graph-plotting-in-python-set-1/ (Accessed: 29 October 2023).

Jevtic, G. (2021) *How to install Pip for python on windows: Phoenixnap KB*, *Knowledge Base by phoenixNAP*. Available at: https://phoenixnap.com/kb/install-pip-windows (Accessed: 29 October 2023).

APPENDICES

ACKNOWLEDGEMENT

We would like to express our gratitude to Assoc. Prof. Dr. Zulkifli Bin Zainal Abidin, one of the lecturers of this course, mechatronic system integration 1 (MCTA 3203) for his guidance while the experiment is carried out.

STUDENT'S DECLARATION

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 person. 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 everyone is noted within this certificate. We also hereby certify that we have read and understand the content of the total report and no further improvement on the reports is needed from any of the individual's contributors to the report. We, therefore, agreed unanimously that this report shall be submitted for marking and this final printed report have been verified by us.

Signature: Ghousiah Name: Noorul Ghousiah Binti Noordeen Sahib Matric Number: 2118298 Contribution:	Read	✓
	Understand	✓
	Agree	✓
Signature: hanis Name: hanis Matric Number: 2020590 Contribution:	Read	✓
	Understand	✓
	Agree	✓
Signature: Afiq Name: Muhammad Afiq bin Ahmad Matric Number: 2115203 Contribution: Methodology, Results part B	Read	✓
	Understand	✓
	Agree	✓
Signature: ayunaziera Name: Nur Ayu Naziera Binti Rosli Matric Number: 2119202 Contribution: part A discussion, conclusion,recommendation	Read	✓
	Understand	✓
	Agree	✓
Signature:Sarah Name: Sarah Aishah binti Sa'aid Hazley Matric Number: 2117600 Contribution: Introduction, Materials and Setup	Read	✓
	Understand	✓
	Agree	✓