

**LAB 2:**

**Parallel, Serial and USB interfacing with microcontroller and computer based system: Sensors and Actuators**

**MCTA 3202**

GROUP F

GROUP MEMBERS MATRIC NO

1. Muhammad Naufal bin Mohammad Bakri 2110333
2. Muhammad Khairul Aiman bin Mohamad Basari 2113571
3. Muhammad Nabil Ikhmal Bin Khairil Rizal 2114577
4. Muhammad Zakwan Zikri bin Irwan Affandi 2111033
5. Che Muhammad Danial Bin Che Suhaimi 2119075

DATE OF SUBMISSION :

1st November 2023

# **Table of Contents**

| CONTENT | PAGE | REMARK |
| --- | --- | --- |
| **PART A** | | |
| INTRODUCTION | 4 |  |
| MATERIALS AND EQUIPMENT | 4 |  |
| EXPERIMENTAL SETUP | 4 |  |
| METHODOLOGY | 5 |  |
| RESULT (OBSERVATION) | 6 |  |
| DISCUSSION | 6 |  |
| CONCLUSION | 7 |  |
| REFERENCES | 7 |  |
| APPENDICES | 8-9 |  |
| **PART B** | | |
| ABSTRACT | 10 |  |
| INTRODUCTION | 11 |  |
| MATERIALS AND EQUIPMENT | 11 |  |
| EXPERIMENTAL SETUP | 12 |  |
| METHODOLOGY | 13 |  |
| RESULT (OBSERVATION) | 14 |  |
| DISCUSSION | 14 |  |
| CONCLUSION | 15 |  |
| REFERENCES | 16 |  |
| APPENDICES | 17-18 |  |
| STUDENT’S DECLARATION | 19 |  |

# **PART A**

## **Abstract**

The objective of this experiment is to establish a serial communication link between Python and an Arduino, enabling the exchange of data from a potentiometer on the Arduino to a computer through a USB connection. This helps understand the process of setting up serial communication and allows utilization of real-time data from the Arduino in Python applications.

To establish a serial communication an Arduino will be programmed to send potentiometer data continuously via serial. On the other side, a Python script will use the **pyserial** library to establish a serial connection and read the data sent by the Arduino.

The key findings of this experiment reveal the successful establishment of serial communication link between Python and an Arduino, enabling the real-time transfer of potentiometer readings. The experiment demonstrated the possibility for reliable communication between Arduino and Python allowing for further use of the data such as plotting the data.

In conclusion, this experiment demonstrates the successful establishment of a serial communication link between Python and an Arduino, allowing data exchange, specifically potentiometer readings. By configuring both ends, this communication setup proves to be effective for real-time data utilization in Python applications. This project's outcomes contribute to a foundational understanding of serial communication between microcontrollers and computers, opening doors for various applications and projects in the future.

## **Introduction**

### Overview of the experiment's purpose and objectives

The objective of this experiment is to create a serial communication link between an Arduino and Python so that data can be sent from the Arduino's potentiometer to a computer via a USB connection. This makes it easier to comprehend how to set up serial connectivity and enables the use of Arduino real-time data in Python programs.

### Background information and relevant theory or concepts

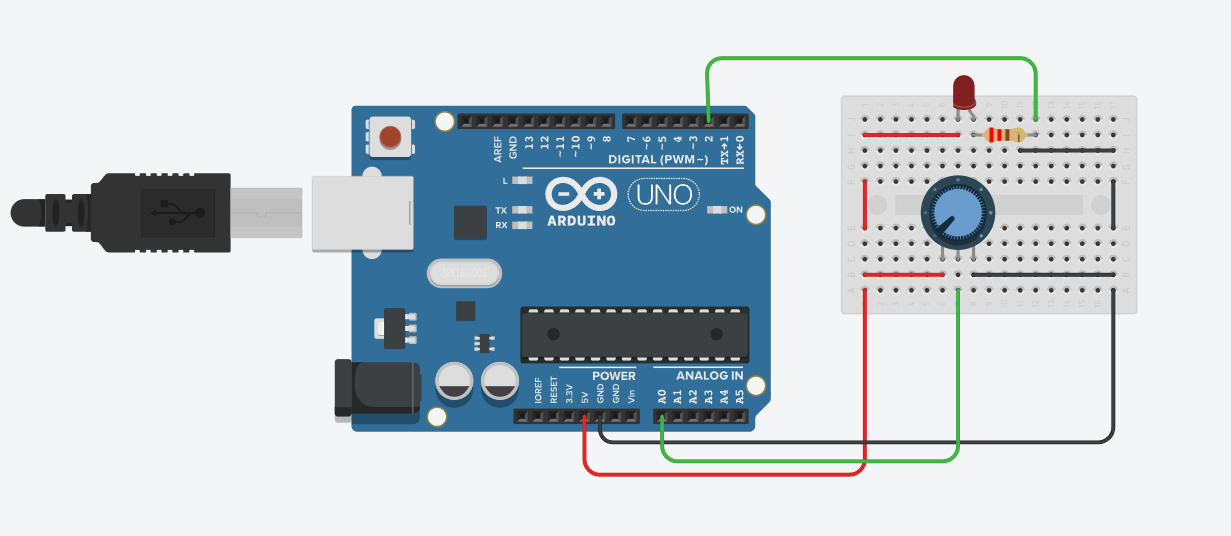
Arduino is a microcontroller used to control hardware and read data from sensors. Since a computer cannot access these hardware and sensors directly, an arduino would be used to interface with these hardware and send data to the computer. Moreover, Arduino boards are a lot cheaper than the average computer, so if an Arduino destroyed from a malfunctioning motor or an actuator is less devastating than if it is a computer instead.

Python is a versatile programming language usually used in data science applications. By using the **pyserial** library, a Python script can read and write to a serial port. Thus, a communication between computer and Arduino can be established through a Python script. Furthermore, the versatility in Python at manipulating data allows for complex calculation with the access of higher compute power.

## **Materials and Equipment**

* Arduino Board
* Potentiometer
* Jumper Wires
* LED
* 220 resistor
* Breadboard

## **Experimental Setup**



## **Methodology**

1. The circuit was built as in the experimental setup.
2. Arduino code was uploaded to the provided Arduino Uno.
3. The port on the Python script was changed to match the Arduino’s port.
4. The Python script was run on the computer.
5. The output of the Python script was collected.Data Collection

## **Data Collection**

Output by Python script

Potentiometer Value: 0

Potentiometer Value: 3

Potentiometer Value: 9

Potentiometer Value: 18

Potentiometer Value: 24

Potentiometer Value: 30

Potentiometer Value: 41

Potentiometer Value: 47

Potentiometer Value: 54

Potentiometer Value: 56

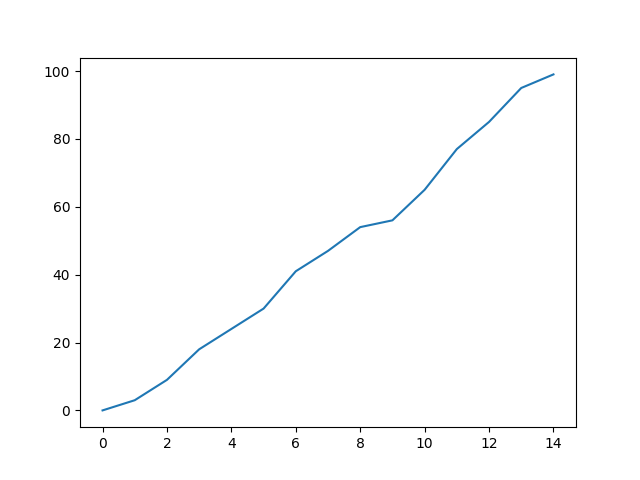
Potentiometer Value: 65

Potentiometer Value: 77

Potentiometer Value: 85

Potentiometer Value: 95

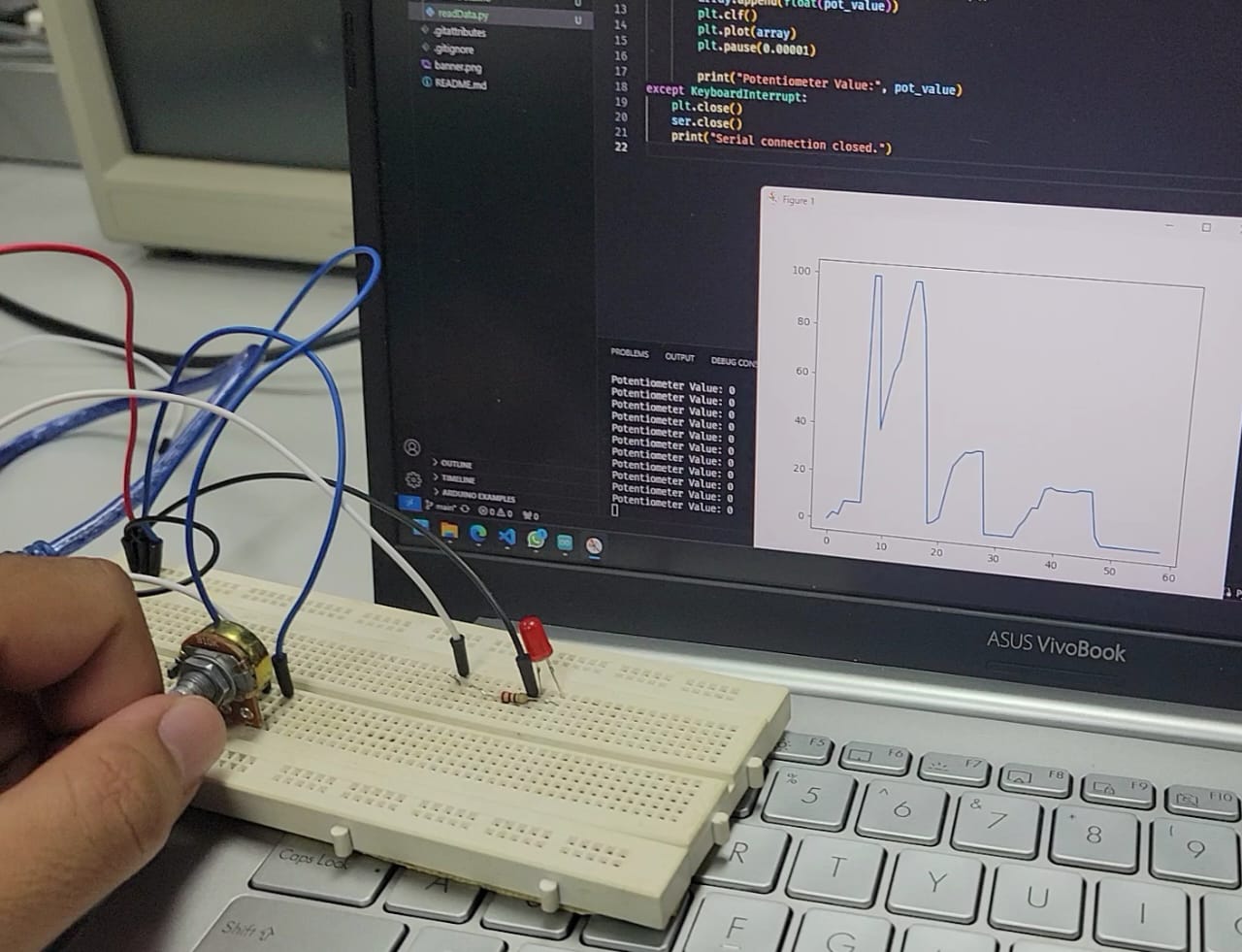
Potentiometer Value: 99



## **Data Analysis**

The potentiometer value shown by the Python script shows a changing value indicating an established serial communication. The value also corresponds to the current potentiometer position. As the potentiometer is turned, the value displayed by the script gradually ascends.

## **Results**



<https://github.com/NotLafuan/GROUP-F-MCTA-3203/raw/main/Week%203/Group%20F%20Week%203%20Part%20A.mp4>

## **Discussion**

The raw analog value of the potentiometer for 0 to 1023 some fluctuation due to long jumper wire and flimsy connection. We map the value from 0 to 1023 and change it into a percentage from 0 to 100. This removes the small fluctuation and makes the value change smoothly.

The serial port can only be accessed by one program. Whenever we try to upload the code, the uploader will spew out errors since we forgot to stop the Python script from using the port and vice versa. This causes a lot of frustration and time wasted. We solve this problem by closing any program using serial communication when we are done using them.

## **Conclusion**

In this experiment, the objective is to read data sent by Arduino using Python via serial. Additionally, this data can be visualized by changing the Python script to plot the values.

Through conducting this experiment, the serial communication established is robust and very reliable. Any fluctuation can be attributed to the electrical noise in the potentiometer and wires instead of the serial communication. This level of reliability can be achieved using only simple code on both sides. This simplicity provides the potential for more complex uses of the reading. In this experiment, we use the opportunity to visualize the reading changing by plotting it on a line graph.

Finally, this experiment is not only successful but it has opened the opportunity for more tinkering with the possibility of this basic knowledge of serial communication. This knowledge can be applied in complex systems where use of multiple microcontrollers and computers are commonplace. This is only the beginning of a bigger field of communication.

## **Recommendations**

The plots displayed are not quiet real-time. One second delay on the Arduino side is required to make sure there is no data missed during reading. This problem can be fixed by using multithreading where one thread will read the data and another thread will plot and display the collected data. With this implemented, the reading will not be hampered by the delay in plotting and displaying the data.

## **References**

<https://www.arduino.cc/reference/en/language/functions/communication/serial/>

<https://www.arduino.cc/reference/en/language/functions/analog-io/analogread/>

<https://pyserial.readthedocs.io/en/latest/>

<https://matplotlib.org/stable/index.html>

## **Appendices**

### Code Snippets

Arduino Code

void setup()

{

Serial.begin(9600);

}

void loop()

{

int potValue = analogRead(A0);

int percentage = map (potValue,0,1023,0,100);

Serial.println(percentage);

if (percentage>70)

digitalWrite(9,HIGH);

else

digitalWrite(9,LOW);

delay(1000);

}

Python Code

import serial

import matplotlib.pyplot as plt

ser = serial.Serial('COM6', 9600)

array= []

try:

while True:

pot\_value = ser.readline().decode().strip()

array.append(float(pot\_value))

plt.clf()

plt.plot(array)

plt.pause(0.00001)

print("Potentiometer Value:", pot\_value)

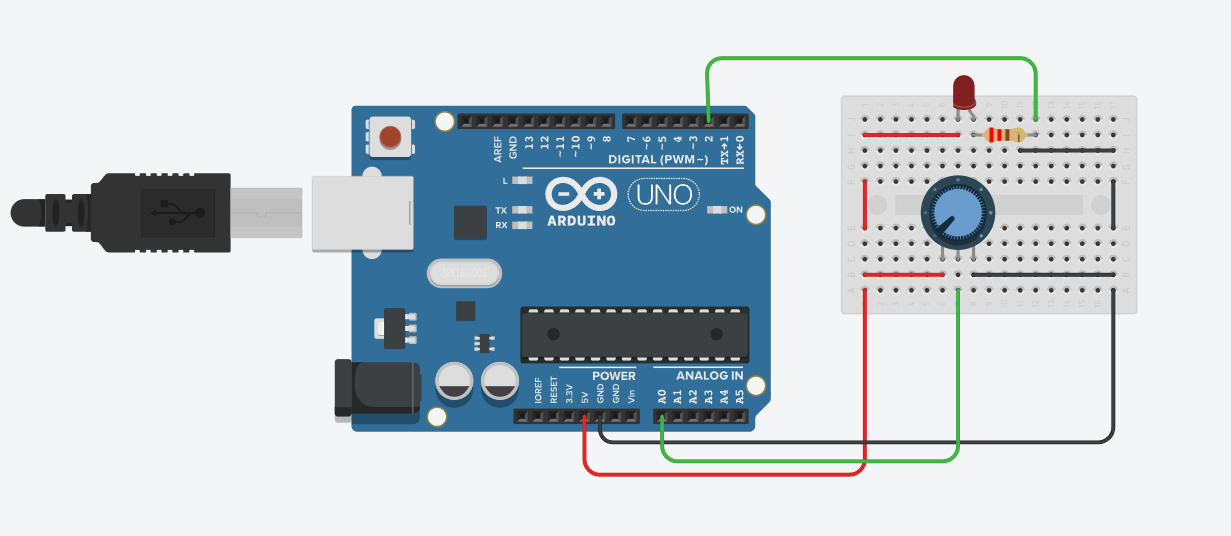
except KeyboardInterrupt:

plt.close()

ser.close()

print("Serial connection closed.")

### Circuit Diagram



# **PART B**

## **Abstract**

This lab report investigates the communication between an Arduino microcontroller and a Python program for data input. The objective of this experiment was to establish a bi-directional communication link between the Arduino and Python, allowing the exchange of data. The serial communication protocol was employed as the means to achieve this interactivity.

The setup involved connecting an Arduino board to a computer via USB and developing a Python script to communicate with it using the PySerial library. Data transmission was initiated from both ends, enabling data to be sent from the Arduino to the Python program and vice versa.

Various experiments were conducted to evaluate the performance of this communication method, including the successful transmission and reception of data in different formats, such as integers, strings, and sensor readings. The results showed reliable data transfer with minimal data loss and latency, making it a suitable solution for interfacing Arduino with Python.

Overall, this experiment demonstrated the successful establishment of communication between an Arduino microcontroller and Python, emphasizing the importance of serial communication in bridging the gap between hardware and software applications. This practical implementation holds great potential for numerous applications, including IoT projects, data logging, and real-time monitoring, offering a versatile and efficient way to input and output data between the two platforms.

## 

## **Introduction**

The purpose of this experiment is to demonstrate and integrate hardware and software. The hardware in this case is the Arduino and the software is Python. It demonstrates how two distinct platforms can communicate and work together to control a physical device. It can help to prepare engineers to learn about the basics of hardware control using softwares.

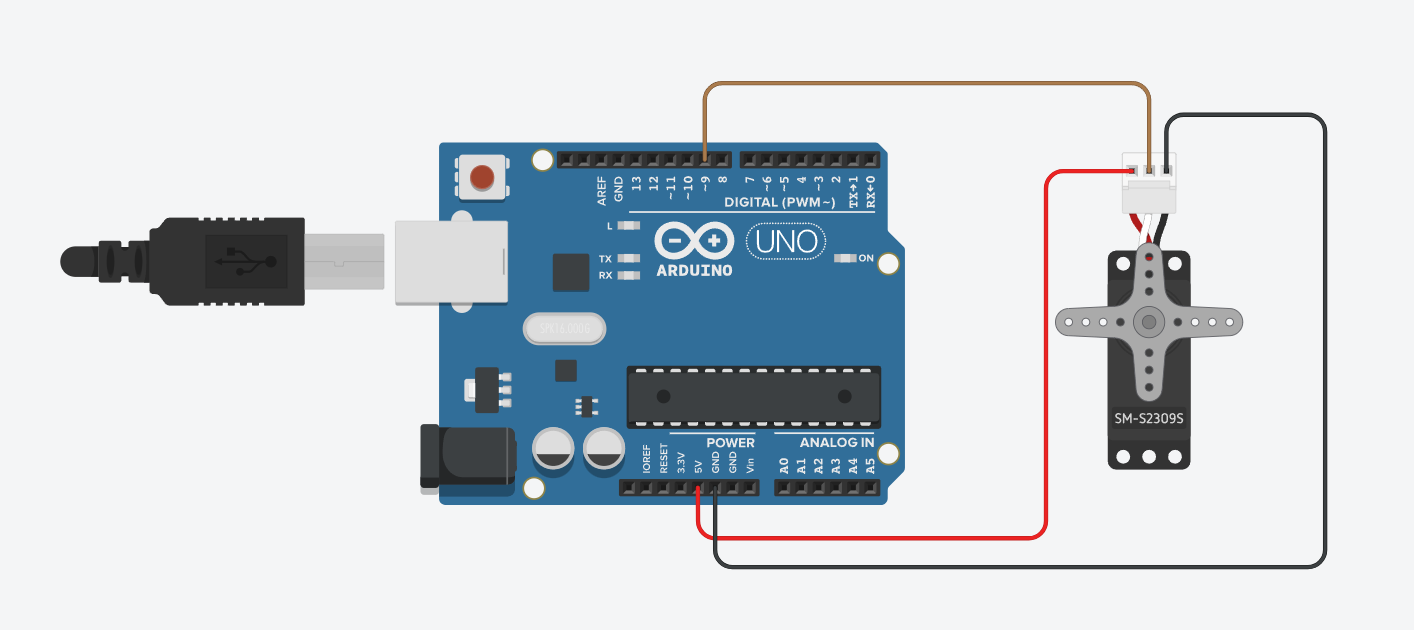
This experiment also introduces the concept of serial communication between devices such as Python script and the Arduino board using serial connection, and how the data can be sent and received between the two platforms. It will also help to learn how Servo Motor works and the required commands and signals needed to operate it.

This experiment is also a good start for future engineers to learn and master how to integrate different components into one complex engineering product. Furthermore, the knowledge can be implemented in a more complex creation which also involve different systems, incorporating sensors, actuators and other automated control systems.

## **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**



## **Methodology**

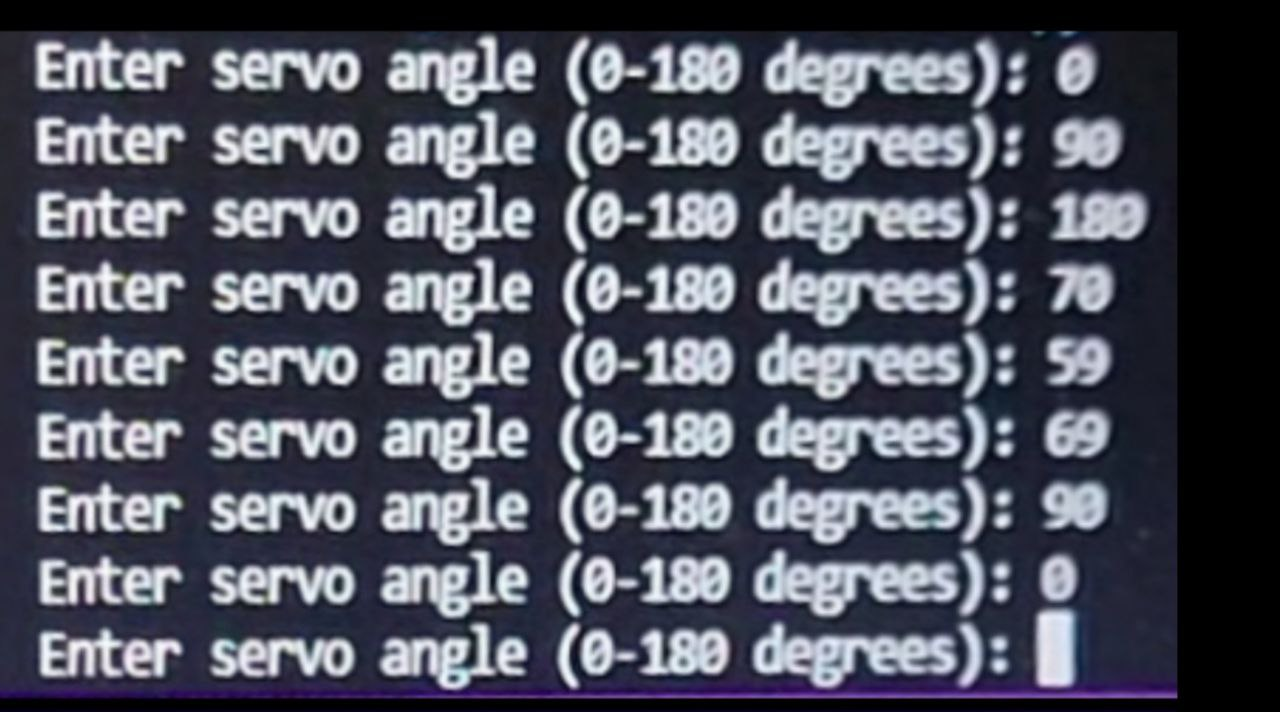
1. The circuit was built as in the experimental setup.
2. In the Arduino IDE,the servo was downloaded to use the function of the servo motor.
3. Arduino code that reads angle data from the serial port and moves the servo accordingly was being uploaded into the arduino.
4. The Python script that prompts the user to enter the servo is run on the laptop.
5. The movement of the servo motor based on the input angle from the user was being recorded.

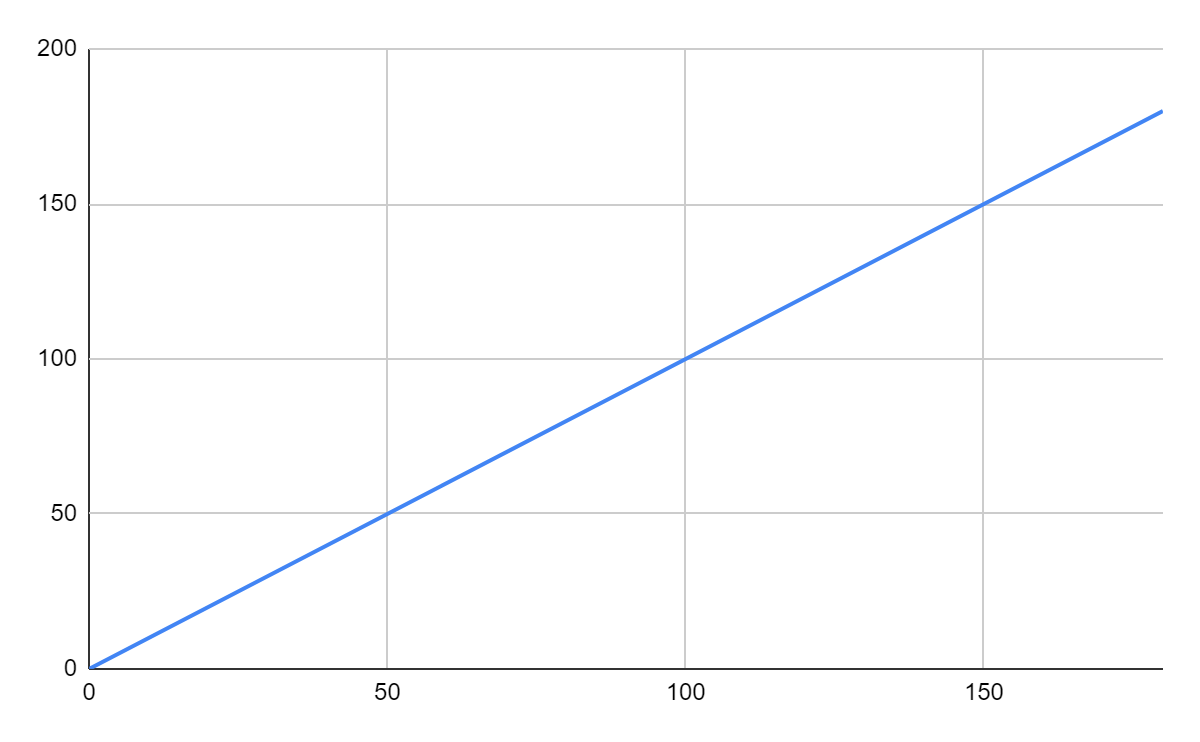
## 

## 

## 

## **Data Collection**





## **Data Analysis**

The movement of the servo motor is dependent on the input angle between 0 to 180 that users submit in the python prompt.The Python script will send the angle to the Arduino over the serial port.

## **Results**



<https://github.com/NotLafuan/GROUP-F-MCTA-3203/raw/main/Week%203/Group%20F%20Week%203%20Part%20B.mp4>

## **Discussion**

Upon execution, the Python script initializes a serial communication interface with the Arduino using the PySerial library, enabling bidirectional data exchange between the two platforms. The script is specifically designed to not only establish a connection but also to precisely manage the servo motor's angle once activated.

Within the Python script, the logic is structured to access the serial port, awaiting user-defined inputs to control the servo motor's angle. This interactivity allows users to command the servo motor to move at specific angles by inputting numerical values. Upon receiving these angle directives, the Arduino interprets and translates them into physical movements, effectively adjusting the servo's position as instructed.

The servo motor, upon receiving angle commands, executes the prescribed movements and holds its position until new instructions are relayed by the user. This approach facilitates a step-by-step control, allowing for precise positioning and controlled movements of the servo motor.

The Python program communicates with the Arduino via a USB serial connection, transmitting instructions in a specific format: "str(angle) + 'A'", where 'A' serves as a delimiter, marking the end of one angle's command and the start of the next angle for controlling the servo's position.

Successful execution of the program, alongside proper establishment of connections, facilitates a seamless interaction between the Python script and the Arduino board. This collaboration underscores the efficient communication and operational synergy between the two platforms, demonstrating how user inputs in Python influence physical movements via the Arduino-controlled servo motor. The experiment serves as an illustrative example of hardware-software integration and precise control of a physical device through programmatic inputs.

## **Conclusion**

The conclusion of a serial data experiment from Python to Arduino would depend on the specific goals and outcomes of the experiment. However, in a general sense, you could conclude that you have successfully established a communication link between Python and Arduino using serial communication. This allows you to send data from Python to Arduino for various applications, such as sensor data transfer or remote control. The success of your experiment would depend on factors like the reliability of data transfer, data processing on the Arduino side, and the achievement of your intended goals

Finally, this experiment was not only successful, but it also opened the door to further experimentation with the prospect of this fundamental understanding of serial communication. This information may be employed in complicated systems that make extensive use of several microcontrollers and computers. This is merely the beginning of a larger communication field.

**Recommendations**

The recommendation that our group suggested is to improvise this project in terms of improving the user interface.To enhance the user interface in Python code, prioritize clear and descriptive variable names, incorporate informative comments and documentation, and ensure user-friendly prompts. Implement robust error handling and input validation to guide users through potential issues. Consider incorporating progress indicators for time-consuming operations, maintain a consistent and intuitive design, and employ logging for debugging purposes. If applicable, explore graphical user interface (GUI) libraries for a more interactive experience. Develop menu systems for programs with multiple functionalities, provide user feedback on actions, and utilize color and formatting for improved visual appeal in console outputs. Implement unit tests to ensure ongoing code stability and offer user assistance through help messages or manuals. Additionally, consider internationalization for a global audience. Each recommendation aims to create a more accessible, efficient, and user-friendly interaction between users and the Python program.

## **References**

<https://www.arduino.cc/reference/en/language/functions/communication/serial/>

<https://www.arduino.cc/reference/en/language/functions/analog-io/analogread/>

<https://pyserial.readthedocs.io/en/latest/>

## **Appendices**

### Code Snippets

Arduino Code

#include <Servo.h>

char incomingByte;

String data = "";

Servo servo;

void setup()

{

Serial.begin(9600);

servo.attach(9);

}

void loop()

{

if (Serial.available() > 0)

{

incomingByte = Serial.read();

if (incomingByte == 'A')

{

servo.write(data.toInt());

data = "";

}

else

{

data += incomingByte;

}

}

}

Python Code

import serial

"""

Define the serial port and baud rate

(adjust the port as per your Arduino)

"""

ser = serial.Serial('COM6', 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

data=str(angle)+"A"

ser.write(str(data).encode())

else:

print("Angle must be between 0 and 180 degrees.")

except KeyboardInterrupt:

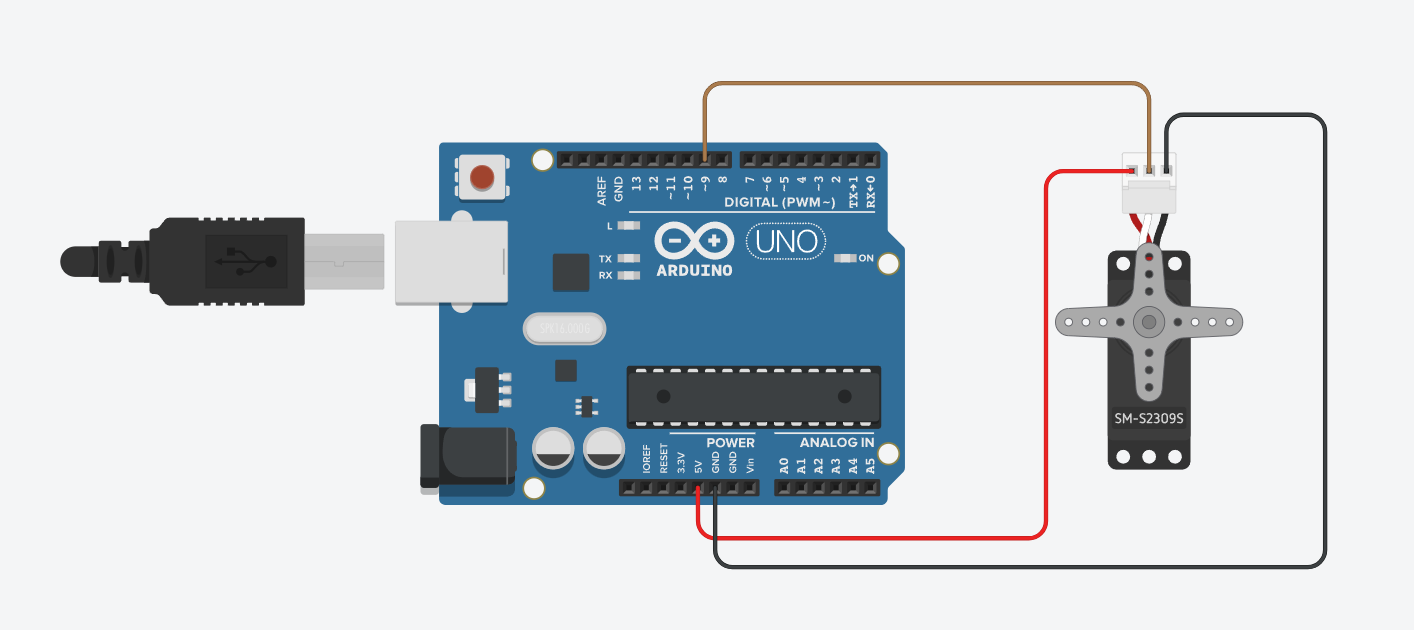
pass # Handle keyboard interrupt

finally:

ser.close() # Close the serial connection

print("Serial connection closed.")

### Circuit Diagram



# **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 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 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 has been verified by us.

Signature: Naufal Read

Name: Naufal Understand

Matric Number: 2110333 Agree

Contribution: Testing and Debugging

Signature: Aiman Read

Name: Aiman Understand

Matric Number: 2113571 Agree

Contribution : Programming

Signature: Nabil Read

Name: Nabil Understand

Matric Number: 2114577 Agree

Contribution: Circuit Design

Signature: Zakwan Read

Name: Zakwan Understand

Matric Number: 2111033 Agree

Contribution: Data Analysis

Signature: Che Read

Name: Che Understand

Matric Number: 21119075 Agree

Contribution : Discussion