



الجامعة الإسلامية العالمية ماليزيا
INTERNATIONAL ISLAMIC UNIVERSITY MALAYSIA
يُؤْتِيهِ سَيِّدِي إِسْلَامُ أَنْبَارًا يُجْنِبُ مِلْمَسِنَا

MECHATRONICS SYSTEM INTEGRATION

WEEK 3A & 3B: SERIAL COMMUNICATION

NAME	MATRIC NUMBER
MUHAMAD IMRAN HAZIQ BIN HAIZAM	2216429
TANISHA ROWSHAN SALEH	2228290
MUHAMMAD YUSOF IRFAN BIN ABIDIN	2210915
ARIESHA DALIELA HANIM BINTI AMIR	2316834

DATE OF SUBMISSION: 24/03/25

TABLE OF CONTENTS

TITLE	PAGE NUMBER
ABSTRACT	2
INTRODUCTION	2
MATERIALS AND EQUIPMENT	3
EXPERIMENTAL SET-UP	4 - 5
METHODOLOGY	6 - 7
RESULTS	8-9
DISCUSSION	10
CONCLUSION	10
RECOMMENDATION	11
REFERENCE	11
APPENDIX	12 - 14
STUDENT DECLARATION FORM	15 - 17

ABSTRACT

experiment(3a) aimed to establish serial communication between a Python script and an Arduino microcontroller, by transmitting potentiometer readings via a USB connection. A potentiometer was connected to the Arduino, with one terminal to 5V, another to GND, and the middle pin to the analog input (A0). The Arduino was programmed to read the potentiometer values and send them over the serial port at a baud rate of 9600.

It was observed that the potentiometer values were successfully transmitted and displayed on the Python console in real-time. However, improper baud rate settings or selecting an incorrect COM port resulted in communication errors. Despite these potential issues, adjusting the settings allowed for smooth and accurate data transmission.

While experiment (3b) aimed to control a servo motor using Python by interfacing it with an Arduino board. The goal was to send angle data from a Python script to the Arduino, which then actuated the servo to move to the specified angle. A servo motor was connected to the Arduino, with its signal wire attached to a PWM-capable pin (digital pin 9), power wire connected to the 5V pin, and ground wire to the GND pin. The Arduino was programmed to receive angle data from the serial port and adjust the servo's position accordingly.

It was observed that the servo responded accurately to the input angles, demonstrating successful communication between Python and Arduino. However, entering invalid values outside the 0-180 range caused errors, requiring input validation. Additionally, selecting an incorrect serial port or baud rate resulted in communication failures.

INTRODUCTION

This experiment had two parts - 3A and 3B - both of which focused on serial communication between two different IDEs (integrated development environment): Arduino and Python, using a potentiometer and a servo motor. Data was transferred from Arduino to Python, using serial communication. Serial communication involves sending data from a sender to a receiver one bit at a time. Serial communication is better than parallel communication, as it's faster, and there's less electrical interference. The electrical interference happens due to multiple bits of data being transmitted at the same time, but since serial communication only involves sending one bit of data at any given time, there is relatively less electrical interference.

Experiment 3A aimed to show the varying values of the voltage controlled by the potentiometer on the serial monitor of the Arduino IDE, and the terminal of the Python IDE (Pycharm). As for experiment 3B, its purpose was to control a servo motor, using Arduino and Python.

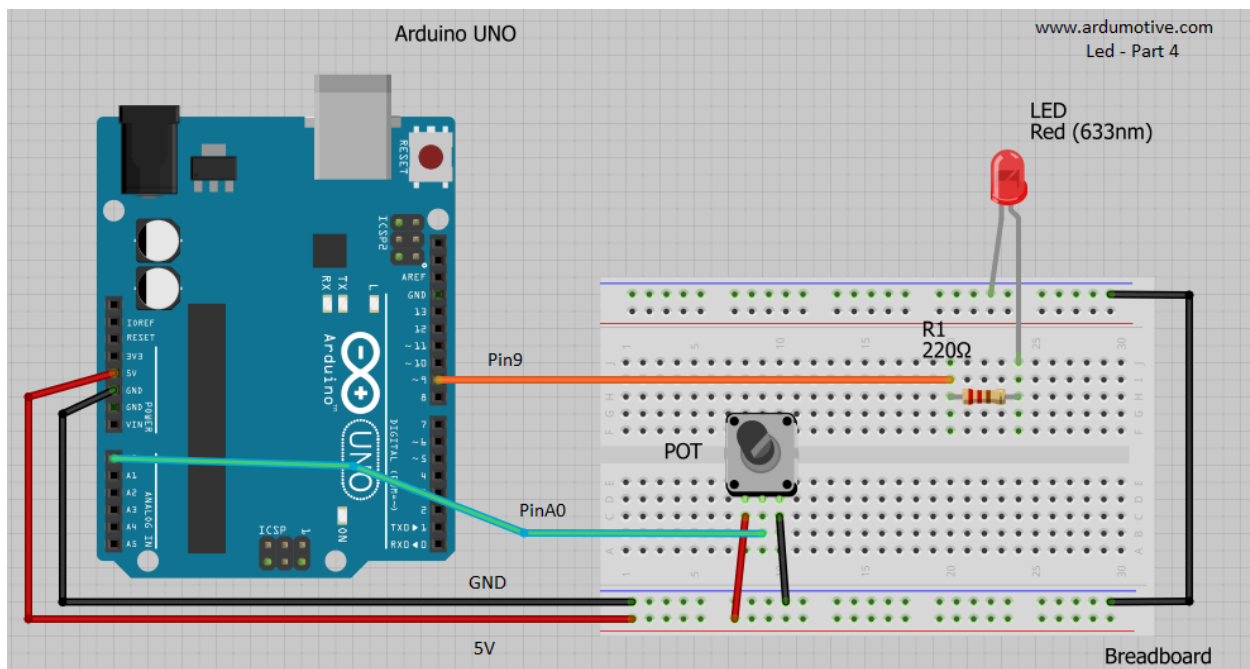
MATERIALS AND EQUIPMENT

Experiment 3A

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

Circuit Setup:

1. Connect one leg of the potentiometer to 5V on the Arduino.
2. Connect the other leg of the potentiometer to GND on the Arduino.
3. Connect the middle leg (wiper) of the potentiometer to an analog input pin on the Arduino, such as A0. An example of the circuit setup is shown in Fig. 1.



Experiment 3B

- 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 SET-UP

Experiment 3A

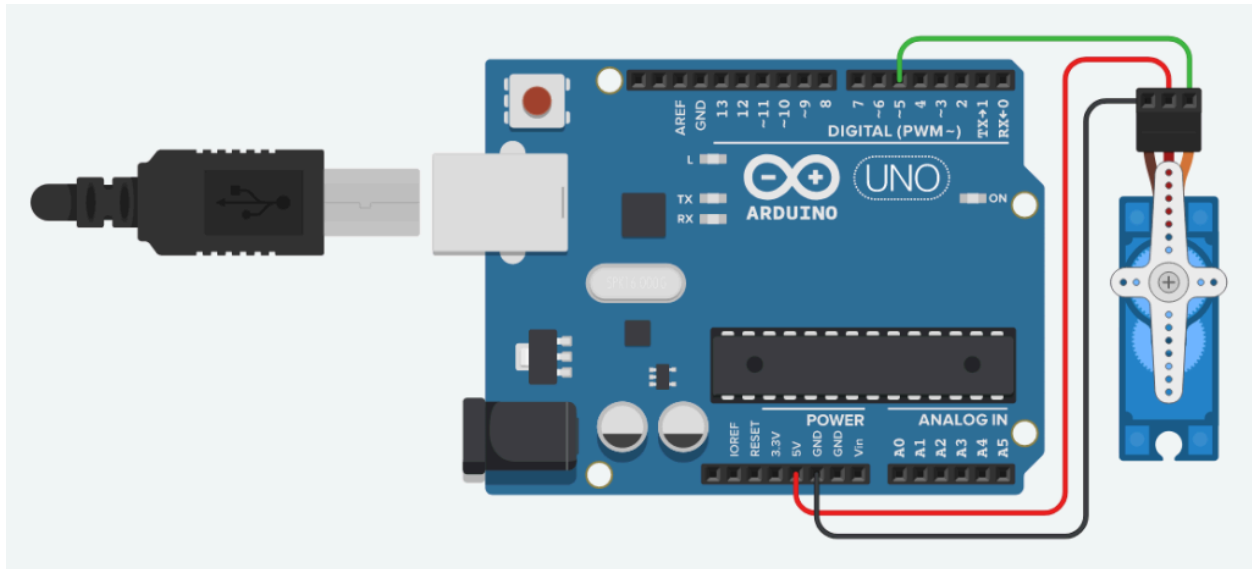
1. Connect the Arduino to your computer via a USB cable.
2. Power on the Arduino (upload the sketch to your Arduino using the Arduino IDE)
3. Run the Python script on your computer.
4. As you turn the potentiometer knob, you should see the potentiometer readings displayed in the Python terminal.
5. You can use these readings for various experiments, data logging, or control applications, depending on your project requirements.

Using the Arduino Serial Plotter (Please note that if you are working with Python to read data from your Arduino, it's important not to open the Arduino Serial Plotter simultaneously. The Arduino Serial Plotter is a dedicated tool for visualizing data received from the Arduino board and may interfere with Python's access to the serial port. If you intend to use Python to read and process data from the Arduino, ensure that the Serial Plotter is closed or not in use while running your Python script to maintain uninterrupted communication between Python and the Arduino.):

6. Open the Serial Plotter: In the Arduino IDE, go to "Tools" -> "Serial Plotter."
7. Select the Correct COM Port: In the Serial Plotter, select the correct COM port to which your Arduino is connected.
8. Set the Baud Rate: Ensure that the baud rate in the Serial Plotter matches the one set in your Arduino code (e.g., 9600).
9. Read Real-Time Data: As you turn the potentiometer knob, the Serial Plotter will display the potentiometer readings in real-time, creating a graphical representation of the data. You can see how the values change as you adjust the potentiometer.
10. Customize the Plotter: You can customize the Serial Plotter by adjusting settings such as the graph range, labels, and colors.

Experiment 3B

- 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).
- 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.
- Connect the Servo's Ground Wire: Connect the servo's ground wire (usually brown) to one of the ground (GND) pins on the Arduino. An example of the hardware setup is shown in Fig. 1



METHODOLOGY

a) Setup :

(3a) Connecting the potentiometer to Arduino with one wire on its end to 5V connection port and the other end to the GND. Centre pin to an analog input pin Arduino, in this case is A0.

(3b) - Connect Servo Signal wire to the PWM-capable pin on Arduino

- Connect 5V supply to the red wire , GND to the brown wire and pin 5 to the Arduino.

b) Codings:

(3a)(i) In Arduino code:

```
void setup()
{
  Serial.begin(9600);
}
void loop() {
  int potValue = analogRead (A0);
  Serial.println(potValue);
  delay(1000);
  // add a delay to avoid sending data too fast
}
```

(ii) in Python :

```
import serial

ser = serial.Serial(A0, 9600)
try:
  while True:
    pot_value = ser.readline().decode().strip()
    print("Potentiometer Value:", pot_value)
except KeyboardInterrupt:
  ser.close()
print("Serial connection closed.")
```

(3b)(i) in Arduino IDE:

```
#include <Servo.h>
Servo servo; int angle
= 90;
void setup () {
  servo.attach (9);}

```

```

void loop () {
  servo.write (angle);
  delay (1000);
  angle = 180
  - angle;
  90 degrees)}}
  // Set to the desired angle

```

(ii) in python:

```

import serial
import time

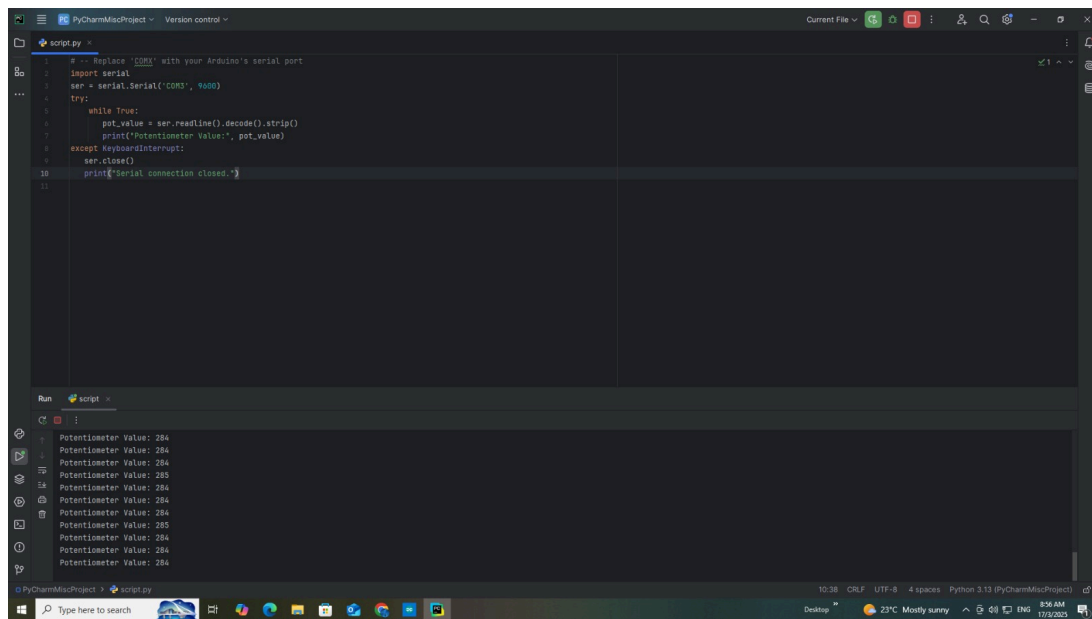
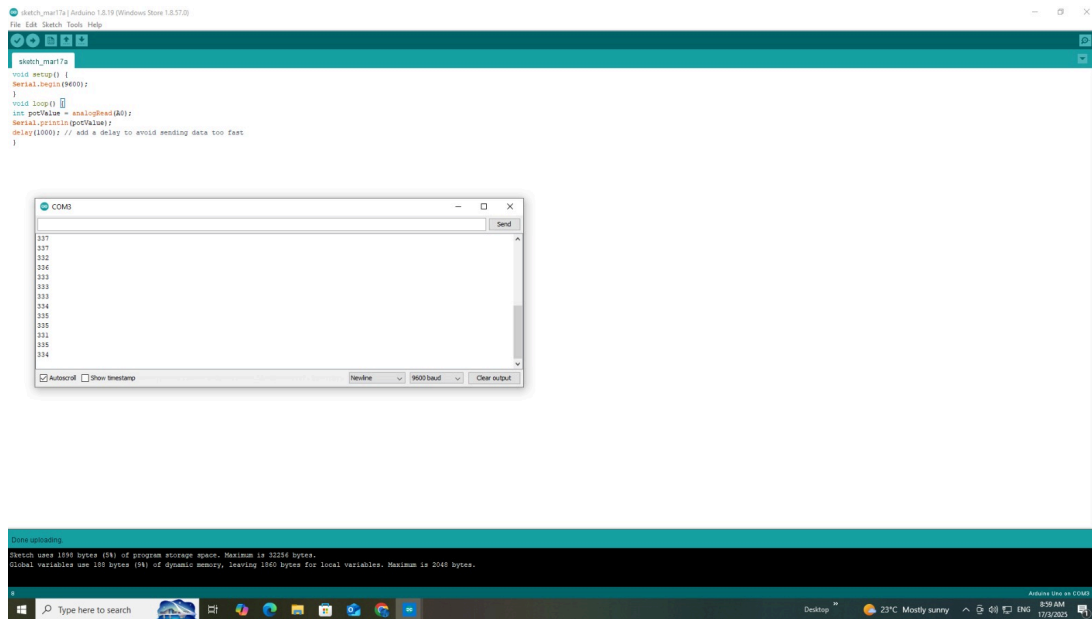
ser = serial.Serial (9, 9600)
try:
  while True:
    angle = input ("Enter servo angle (0-180 degrees): ")
    if angle. lower () == 'q':
      break
    angle = int (angle)
    If 0<= angle <= 180:
      # Send the servo's angle to the Arduino

  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.")

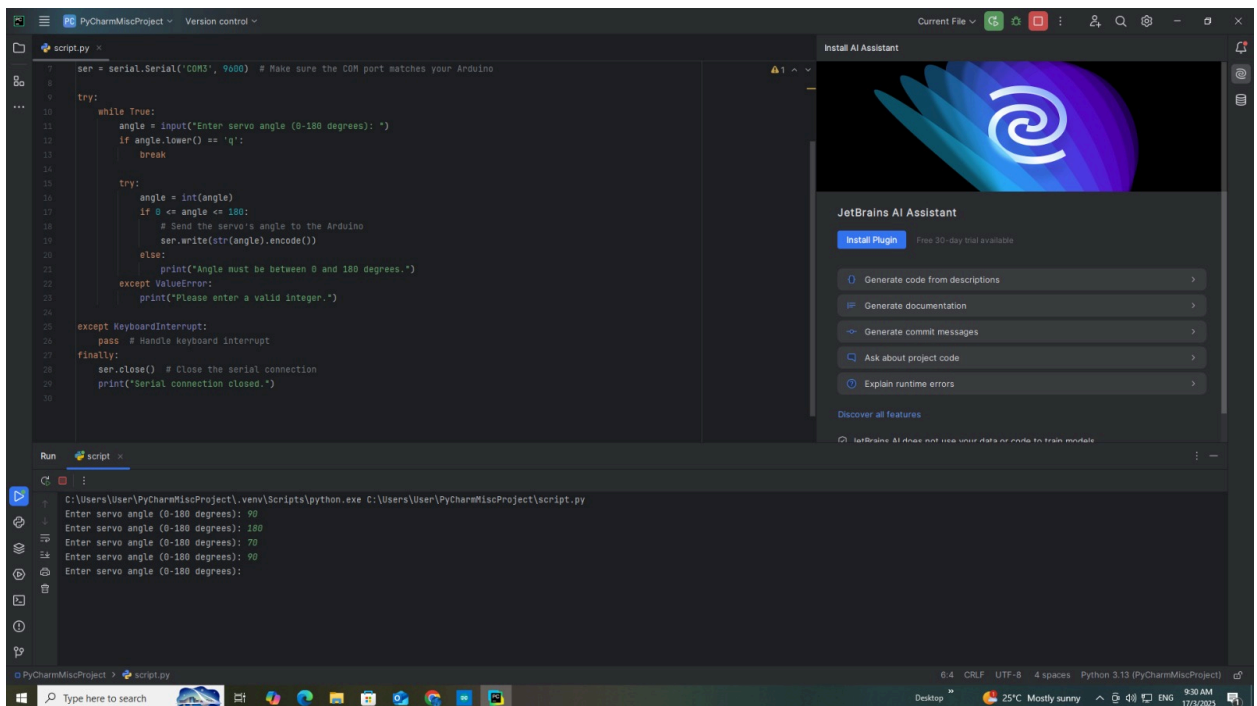
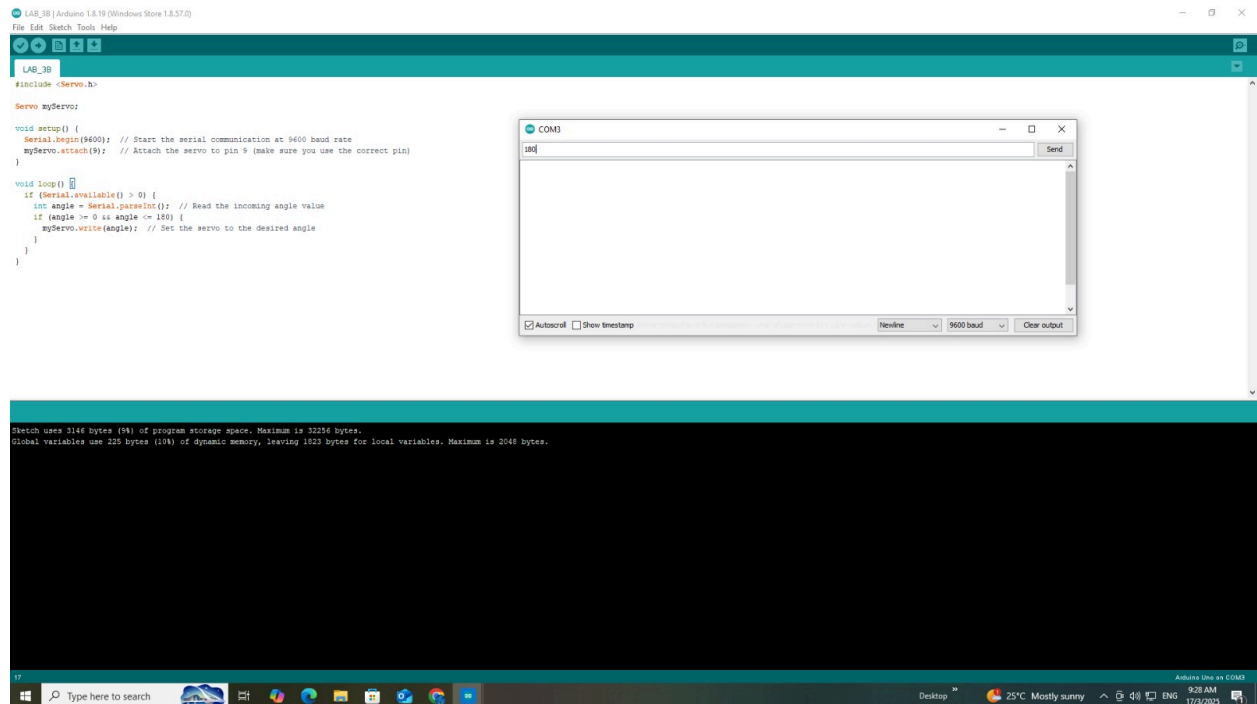
```


RESULTS

Result 3A



Result 3B



DISCUSSION

The first part of the experiment, 3A, involved using a potentiometer to vary the source voltage across the Arduino board, displaying the values on the Arduino IDE's serial monitor as well as Pycharm's terminal. While turning the knob of the potentiometer, a set of values appeared on the Arduino IDE's serial monitor, ranging from 0 to 1024. This range was the digitized values of the analogous voltage input.

Once the Arduino set-up was interfaced with Pycharm, via Pyserial, the same range of values appeared on Pycharm's terminal, indicating a successful interfacing. A set-up like this can be extremely useful in data acquisition and data plotting. If Python proves to be better and more versatile than Arduino in certain aspects, but if the available components only work with Arduino, then being able to interface Arduino with Python would come in very handy. This is one of the many examples of why interfacing between different IDEs can be helpful.

As for the second part of the experiment, 3B, it had a similar purpose, but it involved a servo motor, instead of a potentiometer. After putting the code into the Arduino IDE, any value from 0 to 180-degrees could be input into the serial monitor, prompting the servo motor to turn correspondingly. A prompt message showed up, asking the user to input a number from 0 to 180-degrees, causing the servo motor to turn correspondingly. Generally, servo motors can't directly be interfaced with Python, which is why Arduino-Python interfacing is essential.

CONCLUSION

This experiment demonstrated the successful use of serial communication between Arduino and Python to transmit data and control a servo motor. **Experiment 3A** verified that potentiometer voltage readings could be displayed in both Arduino and Python, confirming accurate data transfer. **Experiment 3B** showed that Python could process the received data to control a servo motor in real time.

The experiment highlighted the advantages of serial communication, such as reduced electrical interference and reliable data transmission. These findings reinforce its importance in embedded systems, automation, and IoT applications. Future improvements could include testing different baud rates, integrating multiple sensors, or exploring wireless communication methods.

Overall, this experiment provided a practical understanding of how serial communication enables real-time interaction between hardware and software. By expanding on these concepts, more complex systems can be developed, allowing for advanced automation, robotics, and remote control applications.

RECOMMENDATION

To enhance the coding presentation in the report, it is important to properly format the Arduino and Python scripts with correct indentation and line breaks, making the code easier to read and understand. Adding short comments to explain key lines of code, such as defining variables, setting up serial communication, and processing data, will help clarify their functions. Additionally, some syntax errors in the current code, such as missing colons, incorrect capitalization, and improperly closed parentheses, should be corrected to ensure the scripts run without issues. Ensuring the accuracy of the copied code will prevent confusion and allow others to replicate the experiment without debugging unnecessary errors. Proper code formatting and explanations will make the report more professional and accessible, even to readers with minimal programming experience.

In addition to improving the code presentation, incorporating visual aids will greatly enhance the clarity of the report. Including circuit diagrams for both Experiment 3A and 3B will help illustrate the correct wiring of the potentiometer and servo motor, reducing ambiguity in the setup instructions. Furthermore, flowcharts outlining the sequence of data transmission and servo movement can provide a clearer overview of how the Arduino and Python scripts interact. A materials summary table should also be added to list all components used in the experiment, making it easier for readers to reference required items at a glance. By integrating these visual elements, the report will be more engaging, structured, and easier to follow.

REFERENCE

- 1)<https://projecthub.arduino.cc/ansh2919/serial-communication-between-python-and-arduino-663756>
- 2)<https://forum.arduino.cc/t/using-pyhton-to-communicate-with-arduino-to-control-2-servo-motors/645043>

APPENDIX

3A

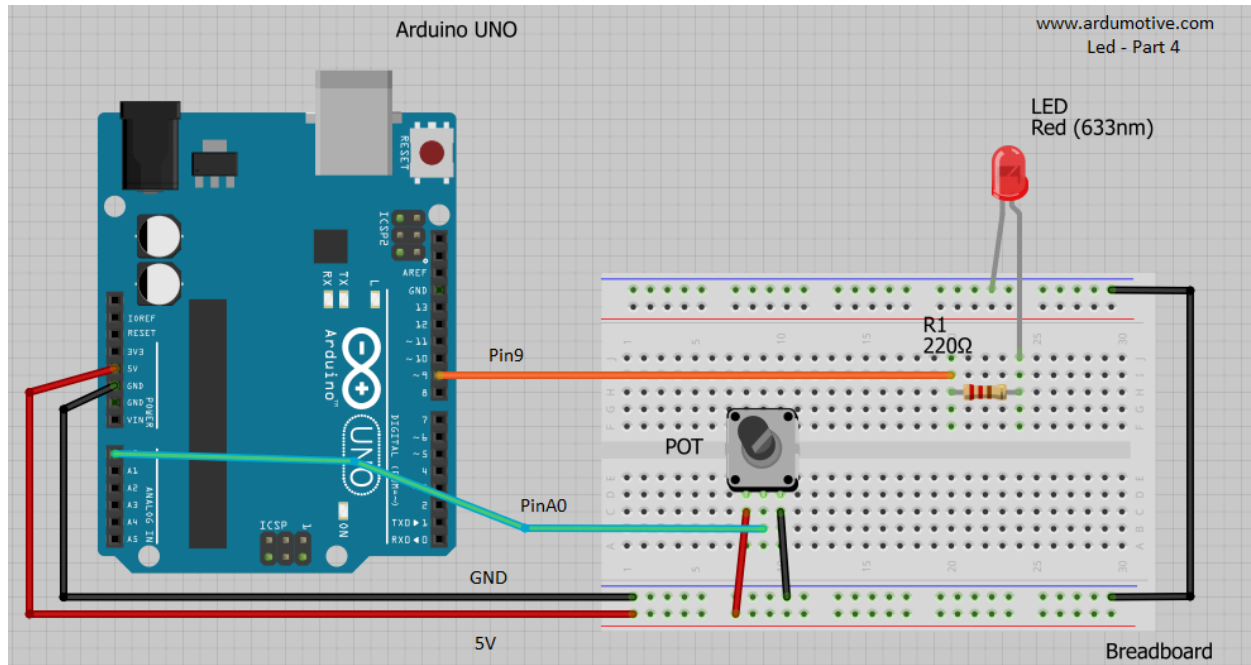
Arduino

```
void setup()
{
  Serial.begin(9600);
}
void loop() {
  int potValue = analogRead (A0);
  Serial.println(potValue);
  delay(1000);
  // add a delay to avoid sending data too fast
}
```

(ii) in Python :

```
import serial

ser = serial.Serial(A0, 9600)
try:
  while True:
    pot_value = ser.readline().decode().strip()
    print("Potentiometer Value:", pot_value)
except KeyboardInterrupt:
  ser.close()
  print("Serial connection closed.")
```



3A's circuit diagram

3B

```

Arduino
#include <Servo.h>
Servo servo; int angle
= 90;
void setup () {
  servo.attach (9);}

void loop () {
  servo.write (angle);
  delay (1000);
  angle = 180
  - angle;
  90 degrees)}
// Set to the desired angle

```

Python:

```

import serial
import time

ser = serial.Serial (5, 9600)

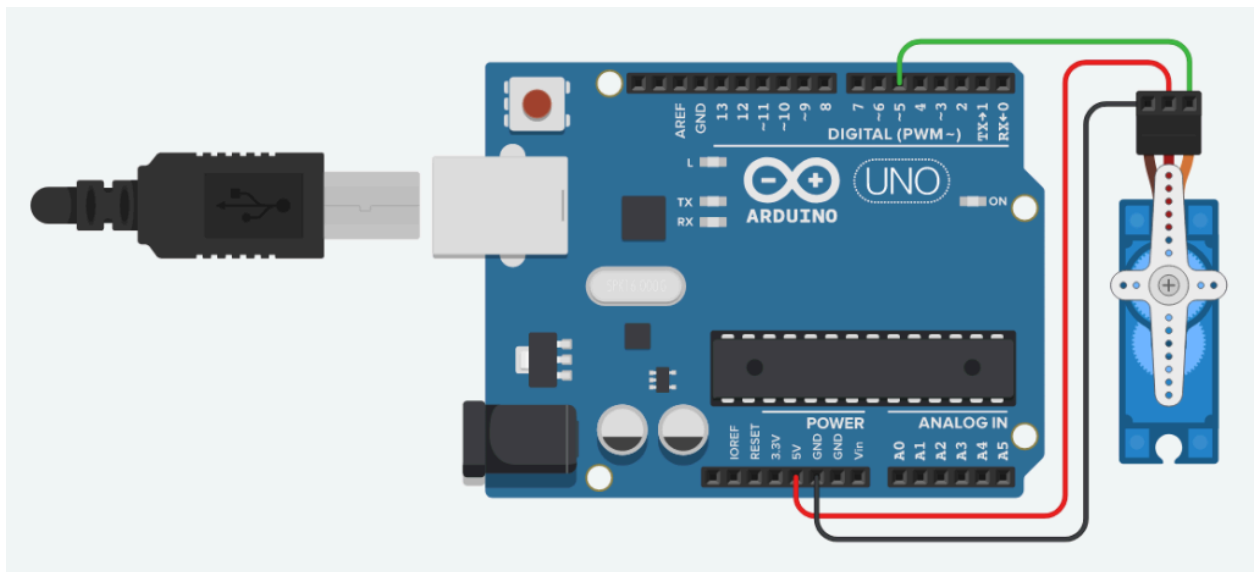
```

```

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.")

```



3B's circuit diagram

Student Declaration Form

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 that no further improvement on the reports is needed from any of the individual contributors to the report.

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

Signature: <i>Abidin</i>	Read	/
Name: Muhamad Yusuf Irfan bin Abidin	Understand	/
Matric No: 2210915	Agree	/
Contribution : Methodology, Result, Recommendation, Reference		

Signature: <i>Tanisha</i>	Read	/
Name: Tanisha Rowshan Saleh	Understand	/
Matric No: 2228290	Agree	/
Contribution : Discussion, Introduction		

Signature: 	Read	/
Name: Muhamad Imran Haziq bin Haizam	Understand	/
Matric No: 2216429	Agree	/
Contribution : Equipment, Conclusion		

Signature: 	Read	/
Name: Ariesha Daliela Hanim Binti Amir	Understand	/
Matric No: 2316834	Agree	/
Contribution : Equipment setup,		