MCTA 3203

MECHATRONICS SYSTEM INTEGRATION

SEMESTER 1, 25/26

WEEK 03 : SERIAL COMMUNICATION BETWEEN ARDUINO AND PHYTON

SECTION 2

GROUP 19

LECTURER : DR WAHJU SEDIONO

DATE OF EXPERIMENT : 22 OCTOBER 2025

DATE OF SUBMISSION : 27 OCTOBER 2025

PREPARED BY :

NAME	MATRIC NUMBER
MUHAMMAD IMRAN BIN VEDDIN	2316409
ALYA KHAIRAH BINTI KHAIRUL JAMIL	2317100
MUHAMMAD IRFAN RUSHDI BIN ASRI	2319011

ABSTRACT

This experiment demonstrates how to create a serial communication link between an Arduino microcontroller and Python software to transmit potentiometer readings in real time. This experiment divided into 2 parts, part 3A and 3B. Part 3A focusses on transmitting potentiometer readings from an Arduino to a Python script via USB, showcasing real-time data acquisition. In part 3B, a servo motor is controlled by sending angle commands from Python to Arduino, demonstrating actuator control through serial communication. These experiments provide hands-on experience with protocols, enhancing our understanding of hardware-software interaction in embedded system.

TABLE OF CONTENTS

1.0 INTRODUCTION

Serial communication plays a crucial role in enabling interaction between microcontrollers and external systems in embedded applications. Through this laboratory exercise, the communication link between an Arduino and a computer was explored using USB as the interface. Two practical applications were demonstrated to deepen understanding of how data can be transferred both from sensors to software and from software to actuators.

Experiment 1 focused on transmitting real-time potentiometer readings from the Arduino to a Python script. The experiment highlighted how analog signals can be digitized, monitored, and visualized on a computer for analysis and logging.

Experiment 2 expanded the concept into actuator control. A Python script was used to send angle commands to the Arduino, which then actuated a servo motor to the specified position. This demonstrates bidirectional serial communication and showcases how software can directly influence physical hardware behavior.

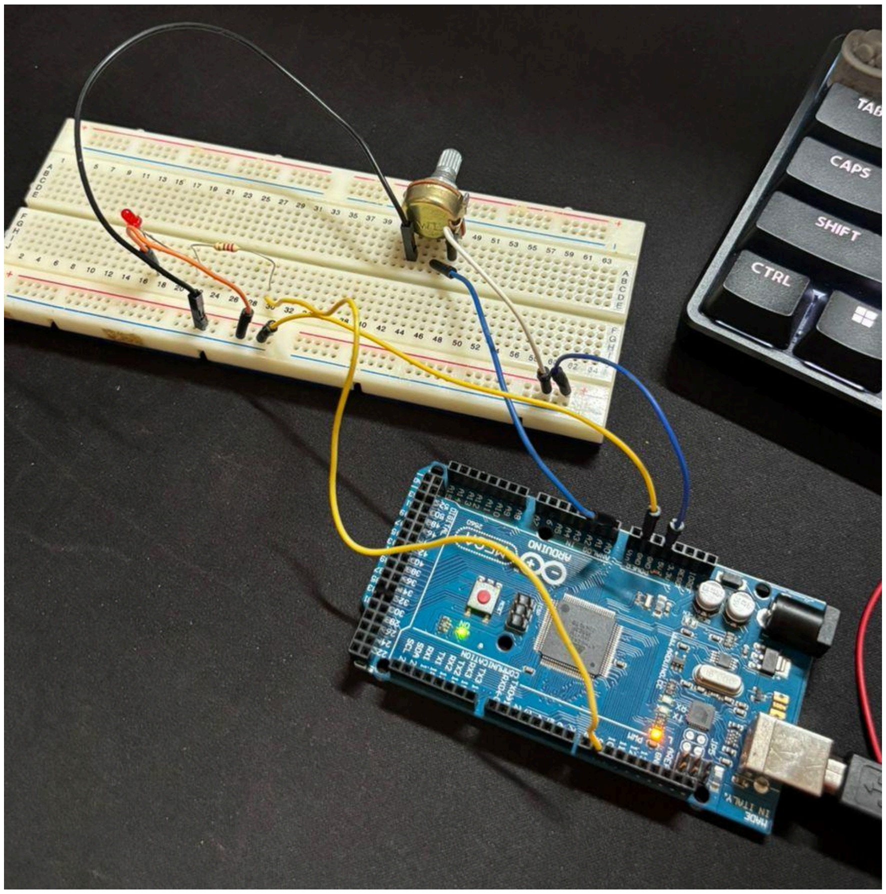
Through these tasks, the experiment emphasizes the integration of sensors, actuators, and microcontrollers, which is essential in robotics, automation, and IoT development. The hands-on activities help build a strong foundation for designing intelligent systems that rely on seamless hardware-software interaction.

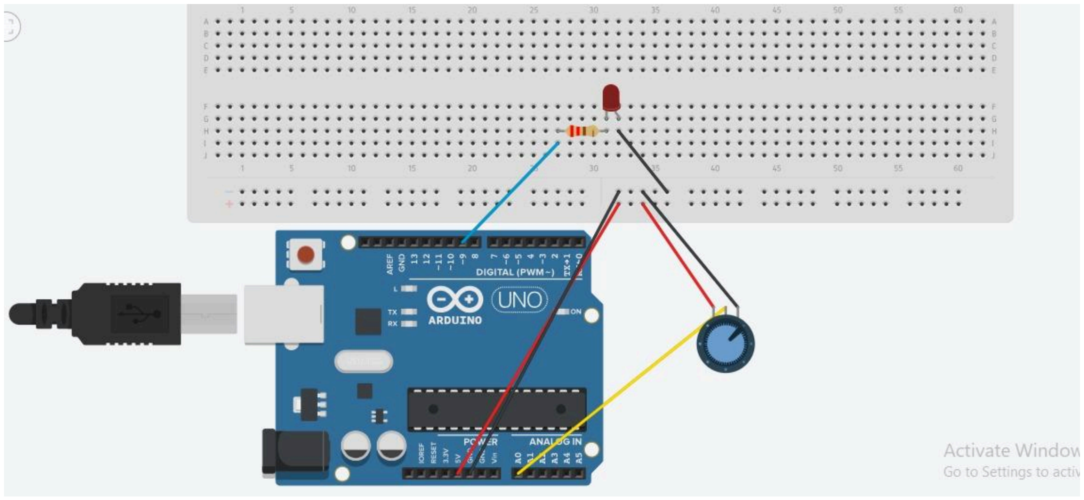
EXPERIMENT 1

2.0 MATERIALS AND EQUIPMENT

Compenent	**Quantity**	**Notes**
Arduino board	1	Arduino Mega.
USB cable	1	For connection and power.
Potentiometer	1	Variable resistor (e.g., 10k Ω).
LED	1	Standard light-emitting diode. (Red)
220 Ω resistor	1	220 Ω (for the LED).
Jumper wires	As needed	
Breadboard	1	For circuit assembly.
Computer	1	Running Python 3, Arduino IDE, and required libraries.

3.0 EXPERIMENTAL SETUP





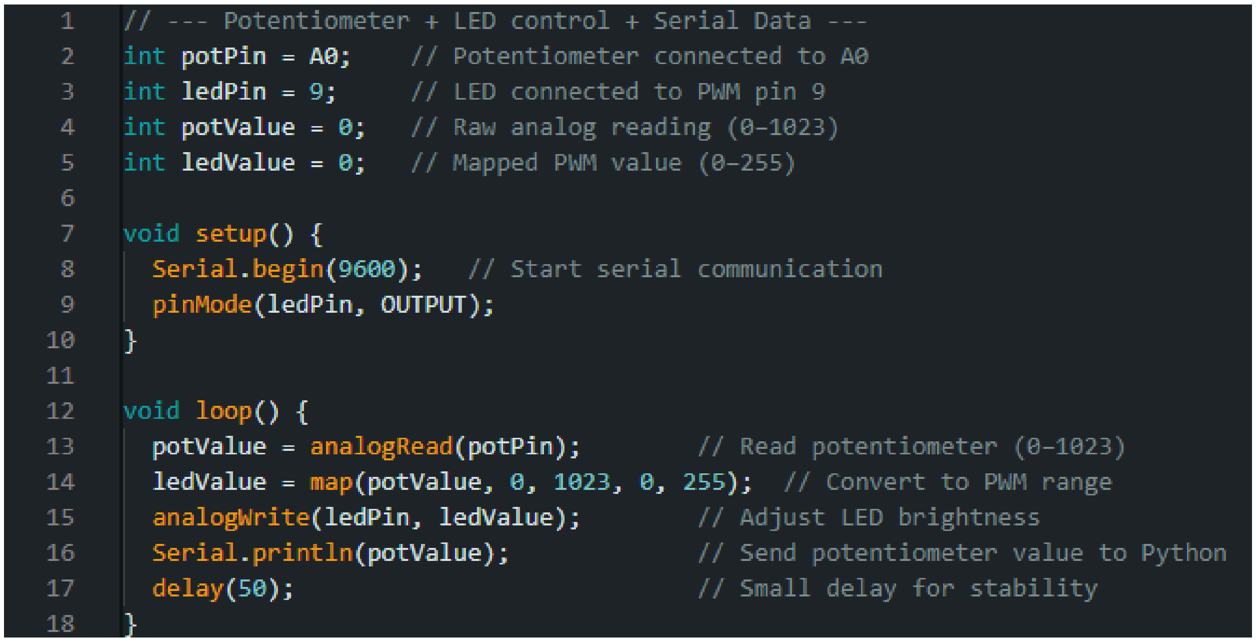
3.1 Circuit assembly :

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 pin in Arduino such as A0.
4. Connect cathode leg of the LED to the pin on the Arduino together with resistor and the anode leg to GND.

4.0 METHODOLOGY

1. The circuit was built according to the circuit setup instructions.
2. The Arduino code was uploaded into the Arduino Mega.
3. Rotating the potentiometer will increasing the brightnest of the LED
4. The output from the potentiometer was sent to Python, and Python displayed a graph.

4.1 Code used



```
// --- Potentiometer + LED control + Serial Data ---
int potPin = A0;    // Potentiometer connected to A0
int ledPin = 9;     // LED connected to PWM pin 9
int potValue = 0;   // Raw analog reading (0-1023)
int ledValue = 0;   // Mapped PWM value (0-255)

void setup() {
  Serial.begin(9600);   // Start serial communication
  pinMode(ledPin, OUTPUT);
}

void loop() {
  potValue = analogRead(potPin);         // Read potentiometer (0-1023)
  ledValue = map(potValue, 0, 1023, 0, 255);  // Convert to PWM range
  analogWrite(ledPin, ledValue);         // Adjust LED brightness
  Serial.println(potValue);              // Send potentiometer value to Python
  delay(50);                             // Small delay for stability
}
```

5.0 DATA COLLECTION

In this experiment , the data collected came from the potentiometer which provides a variable analog voltage depending on how far the knob is rotated. The Arduino converts this voltage into a digital value ranging from 0 to 1023 using its analog-to-digital converter (ADC). These readings were then sent to the computer via serial communication. Python received and displayed these values in real time, allowing observations of how the readings changed when the potentiometer was adjusted. Higher readings were recorded when the knob was turned toward the 5V side, and lower readings were recorded when turned toward the GND side.

6.0 DATA ANALYSIS

The collected data shows a direct relationship between the potentiometer position and the numerical values received on Python. As the resistance changed, the output voltage also changed, resulting in different ADC values. This confirms that the system successfully recorded analog variations and transmitted the data without major delay. Some small fluctuations were noticed even when the knob stayed still, which can be explained by electrical noise or imperfect sensor stability. Despite that, the overall data trend clearly followed the expected behavior which is smooth increase when rotating in one direction and decrease when rotating the other direction. This indicates that serial communication was stable and accurate for real-time sensor monitoring.

7.0 RESULT

The experiment successfully demonstrated the transmission of analog data from the Arduino to Python using serial communication.When the potentiometer was rotated, the LED brightness increased or decreased proportionally, and the corresponding numerical values were displayed in real time on the Python.

1. When turned fully counterclockwise, the values were near 0, and the LED was dim.
2. When turned fully clockwise, the values were near 1023, and the LED was brightest.
3. Intermediate positions produced proportional LED brightness and plotted values on the live graph.

8.0 DISCUSSION

Experiment 1 successfully demonstrated that the potentiometer's analog input can be read by the Arduino and transmitted to a Python script using serial communication. When the potentiometer knob was rotated, the numerical values shown in the Python terminal changed accordingly, proving that data was being sent correctly through the USB connection. This shows that the serial port is able to continuously deliver real-time information from the hardware to the computer.

Some minor issues were observed during testing. The potentiometer readings were not always stable and sometimes fluctuated even when the knob was not being moved. This happened because analog signals are sensitive to small electrical noise and the potentiometer does not always produce a perfectly steady output. The one-second delay in the Arduino code helped reduce rapid changes, but the data still showed slight variations.

Overall, this experiment confirmed that Python can be used as a monitoring tool to receive data from an Arduino in real time. This communication method is important for many future applications, such as data logging, monitoring system performance, and controlling devices based on sensor feedback. The results show a strong foundation for more advanced embedded system integration in upcoming experiments.

EXPERIMENT 2

10.0 MATERIALS AND EQUIPMENT

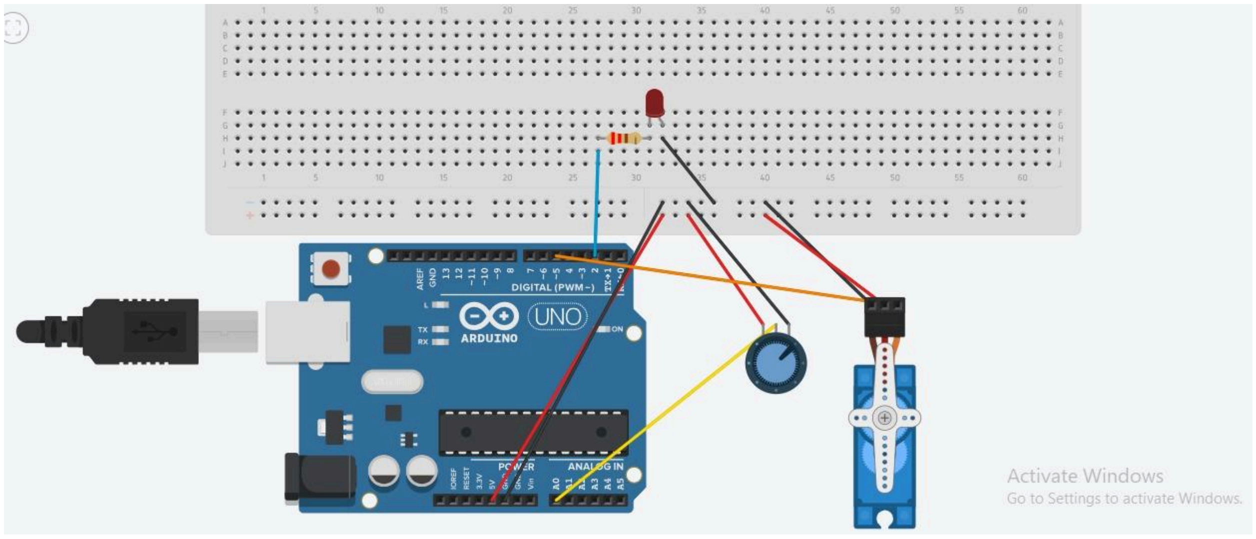
Component	**Quantity**	**Notes**
Arduino Board	1	Arduino Mega.
USB Cable	1	For connection and power.
Servo Motor	1	Standard servo motor (e.g., SG90).
Potentiometer	1	Variable resistor (e.g., 10k Ω).
LED	1	Standard light-emitting diode. (Red)
Resistor	1	220 Ω (for the LED).
Breadboard	1	For circuit assembly.
Jumper Wires	As needed	
Computer	1	Running Python 3, Arduino IDE, and required libraries.

11.0 EXPERIMENTAL SETUP

The circuit setup integrates the three main components (potentiometer, servo, and LED) with the Arduino board.

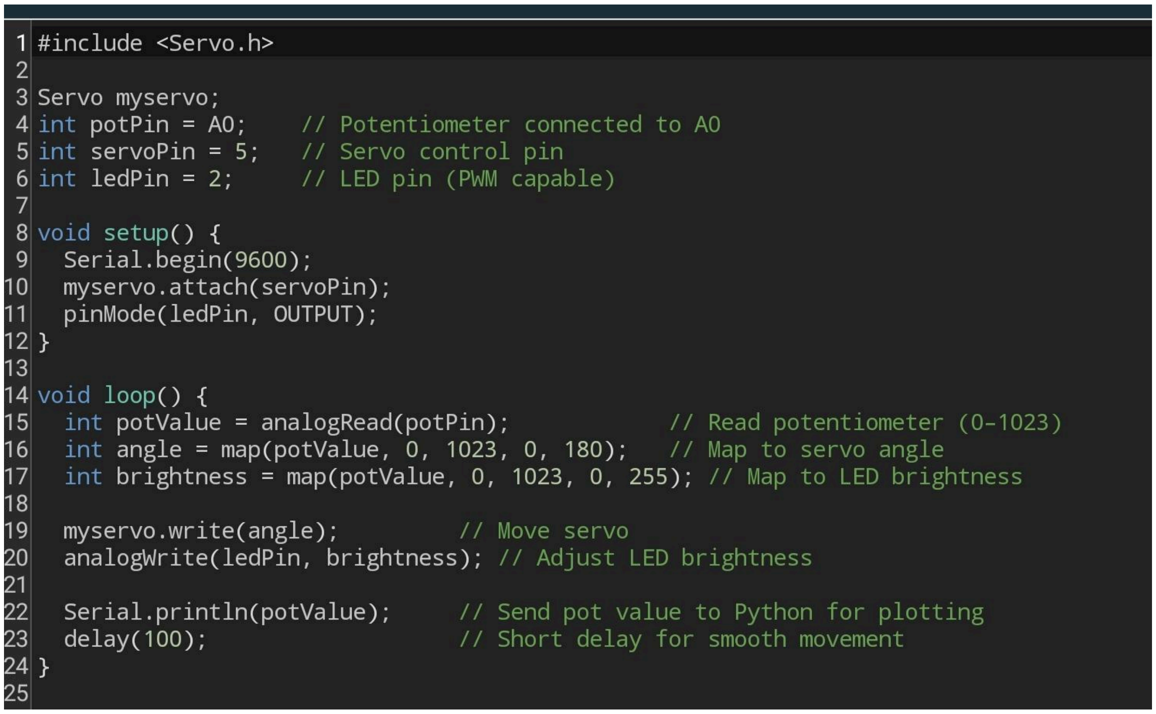
11.1 HARDWARE SETUP (CIRCUIT)

1. Potentiometer (Input):
- One outer pin to 5V, the other outer pin to GND.
- Center pin to Analog Input Pin A0 (potPin).
2. Servo Motor (Main Output):
- Signal wire to Digital Pin 9 (servoPin).
- Vcc and GND wires connected to 5V and GND.
3. LED (Secondary Output/Feedback):
- LED anode (long leg) to Digital Pin 6 (ledPin). *Note: Pin 6 is a PWM-capable pin, required for brightness control.*
- LED cathode (short leg) to the 220 *ohm* resistor.
- Resistor's other end to GND.



11.2 SOFTWARE SETUP

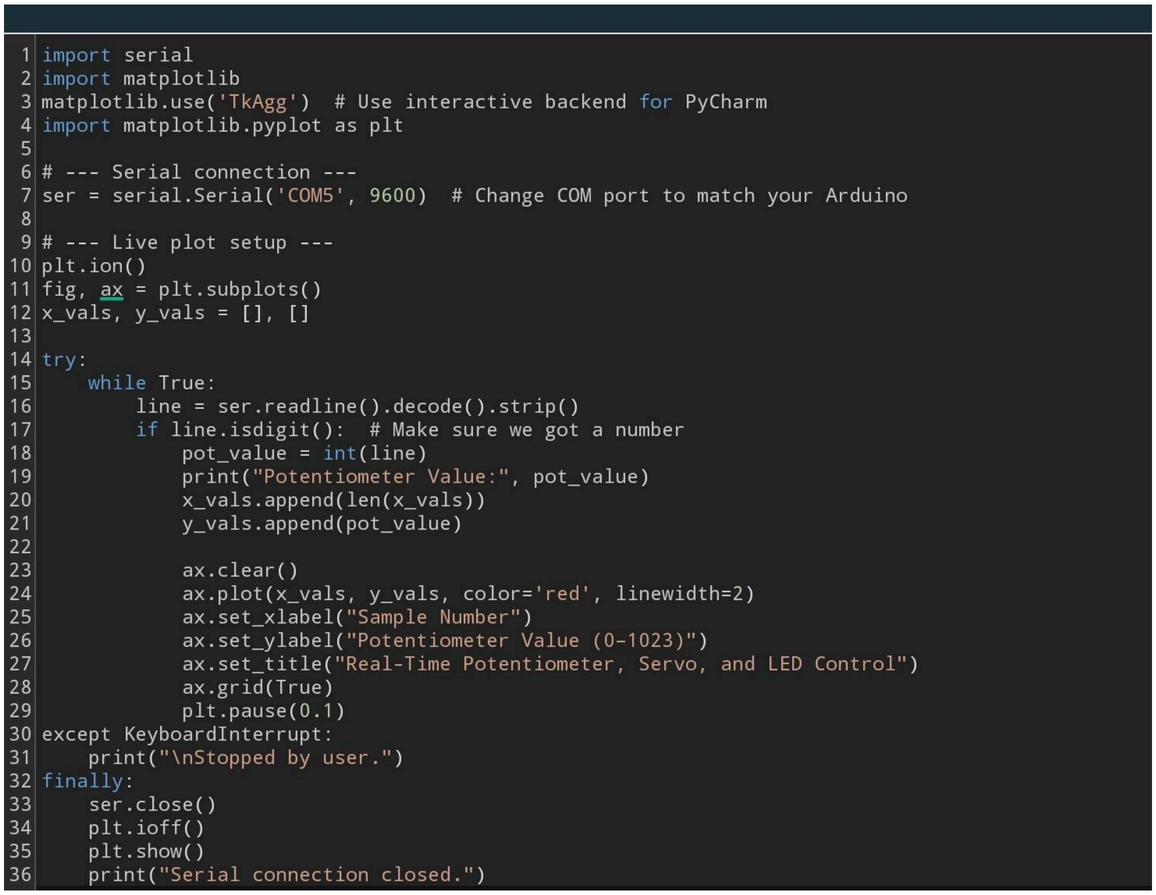
1. Arduino IDE: The Servo library was utilized.(Fig. 1)



```
#include <Servo.h>

Servo myservo;
int potPin = A0;    // Potentiometer connected to A0
int servoPin = 5;   // Servo control pin
int ledPin = 2;     // LED pin (PWM capable)

void setup() {
  Serial.begin(9600);
  myservo.attach(servoPin);
  pinMode(ledPin, OUTPUT);
}

void loop() {
  int potValue = analogRead(potPin);            // Read potentiometer (0-1023)
  int angle = map(potValue, 0, 1023, 0, 180);   // Map to servo angle
  int brightness = map(potValue, 0, 1023, 0, 255); // Map to LED brightness

  myservo.write(angle);         // Move servo
  analogWrite(ledPin, brightness); // Adjust LED brightness

  Serial.println(potValue);     // Send pot value to Python for plotting
  delay(100);                   // Short delay for smooth movement
}

```

Fig. 1

2. Python Environment: The pyserial library was used for communication, and matplotlib (specifically using the 'TkAgg' backend) was used for real-time visualization. (Fig. 2)



```python
import serial
import matplotlib
matplotlib.use('TkAgg')  # Use interactive backend for PyCharm
import matplotlib.pyplot as plt

# --- Serial connection ---
ser = serial.Serial('COM5', 9600)  # Change COM port to match your Arduino

# --- Live plot setup ---
plt.ion()
fig, ax = plt.subplots()
x_vals, y_vals = [], []

try:
    while True:
        line = ser.readline().decode().strip()
        if line.isdigit():  # Make sure we got a number
            pot_value = int(line)
            print("Potentiometer Value:", pot_value)
            x_vals.append(len(x_vals))
            y_vals.append(pot_value)

            ax.clear()
            ax.plot(x_vals, y_vals, color='red', linewidth=2)
            ax.set_xlabel("Sample Number")
            ax.set_ylabel("Potentiometer Value (0-1023)")
            ax.set_title("Real-Time Potentiometer, Servo, and LED Control")
            ax.grid(True)
            plt.pause(0.1)
except KeyboardInterrupt:
    print("\nStopped by user.")
finally:
    ser.close()
    plt.ioff()
    plt.show()
    print("Serial connection closed.")
```

Fig. 2

12.0 METHODOLOGY

The methodology focuses on a single Arduino sketch that handles all control logic, and a Python script dedicated solely to data reception and visualization.

12.1 ARDUINO SKETCH (CONTROL AND DATA TRANSMISSION)

The core of the Arduino code implements proportional mapping for both the servo and the LED.

1. Potentiometer Read: The analogRead(A0) function reads the sensor value, providing a range of 0 to 1023.
2. Servo Mapping: The map function scales the 0-1023 range to the servo's physical range of 0 to 180 degrees.

   angle = map(potValue, 0, 1023, 0, 180)

3. LED Mapping (PWM): The same map() function is used to scale the 0-1023 range to the analogWrite() range of 0 to 255 (for PWM brightness control).

   brightness = map(potValue, 0, 1023, 0, 255)

4. Serial Transmission: The raw potValue (the source data for both control actions) is sent via Serial.println() to the Python script.

12.2 PYTHON SCRIPT (DATA RECEPTION AND VISUALIZATION)

The Python script performs the following steps:

1. Establish Serial Connection: Connects to the Arduino's COM port at 9600 baud.
2. Real-Time Plot Setup: Initializes matplotlib in interactive mode (plt.ion()) to enable continuous plot updates.
3. Data Loop:

- Reads a line of serial data, decodes it, and verifies it is a digit.
- Converts the string to an integer (pot_value).
- Appends the pot_value to the y_vals list and an incrementing index to x_vals.
- Clears the previous plot (ax.clear()) and redraws the data (ax.plot()).
- Uses plt.pause(0.1) to refresh the graph and keep the window responsive.

4. Termination: Closes the serial connection and displays the final plot upon a KeyboardInterrupt (Ctrl+C).

13.0 DATA COLLECTION

Data collection was observational, verifying that the physical actions (potentiometer turn) correlated precisely with the three distinct outputs: servo angle, LED brightness, and the graphical plot value.

Potentiometer position	Pot. Value Sent (0−1023)	Servo Angle (0−180◦)	LED Brightness (0−255)	Plot Observation
Fully CCW (Min)	≈5	≈0	≈0	Plot line near the bottom axis (0)
Quarter Turn	≈256	≈45	≈64	Plot line rising steadily
Halfway (Center)	≈512	≈90	≈128	Plot line in the middle (512)
Three-Quarter Turn	≈768	≈135	≈192	Plot line near the top
Fully CW (Max)	≈1023	≈180	≈255	Plot line at the maximum (1023)

The real-time plot provided a clear, continuous visual record of the raw input data (Potentiometer Value) corresponding to the physical control actions.

14.0 DATA ANALYSIS

The analysis confirms the successful implementation of three parallel, proportional control systems driven by a single input source, with the data stream correctly routed for visualization.

14.1 PROPORTIONAL CONTROL

The Arduino's use of the map() function to translate the input range (0-1023) to the output ranges (0-180 for the servo and 0-255 for the LED) demonstrates Proportional Control. The output magnitude is directly proportional to the input magnitude across the entire range, providing fine, continuous control over the outputs.

14.2 DATA QUALITY AND VISUALIZATION

By sending the raw sensor data (potValue) over the serial port, the Python plot directly reflects the sensor's instantaneous state. The use of the isdigit() check in Python is a crucial addition that prevents the program from crashing if a non-numeric character (e.g., garbage data during initial connection or communication error) is received, improving the robustness of the data handling. The plot accurately displayed the data with appropriate labels and a fixed Y-axis range (0−1023), allowing for meaningful interpretation of the sensor's activity.

14.3 SIMPLIFICATION OF CONTROL

Unlike the previous task, this setup uses a simplified, one-way communication protocol (Arduino to Python) for data visualization. The control logic (servo and LED) is entirely managed by the Arduino, relying only on its onboard sensor input. This structure is common in data logging and monitoring applications where the host computer is purely a display and recording tool.

15.0 RESULT

The experiment successfully achieved a real-time, three-way system integration:

1. **Servo Control:** The potentiometer provided **smooth, proportional control** over the servo motor's angle from 0 to 180.
2. **LED Feedback:** The LED's brightness was adjusted proportionally using PWM (analogWrite on Pin 6), visually reinforcing the input level.
3. **Real-Time Visualization:** The raw potentiometer data was successfully transmitted to the Python script and displayed in a **live graphical plot** using matplotlib, providing accurate, continuous monitoring of the sensor state.

16.0 DISCUSSION

The method employed highlights an efficient way to manage multiple control outputs from a single analog input, using the Arduino's processing power effectively.

16.1 EFFICIENCY AND DECOUPLING

Placing all control logic on the Arduino (instead of relying on Python for control commands) significantly increases the reliability and responsiveness of the servo and LED. The Arduino can update the servo and LED much faster and more reliably than a Python script sending commands over serial, which could introduce latency and jitter. The Python script is then free to dedicate its resources purely to data acquisition and visualization, which it handles efficiently with matplotlib.

16.2 THE ROLE OF PWM

The use of analogWrite() on Pin 6 for the LED is essential. Without PWM (Pulse Width Modulation), the LED could only be ON or OFF, losing the ability to proportionally reflect the smooth change of the potentiometer. PWM simulates analog voltage by rapidly switching the digital pin on and off, controlling the average power delivered to the LED and thus its perceived brightness.

16.3 VISUALIZATION BACKEND

The specific inclusion of matplotlib.use('TkAgg') is a practical necessity for ensuring compatibility and optimal performance of the live plotting functionality within certain Python environments (like PyCharm), emphasizing the importance of configuring the visualization environment correctly for real-time applications.

17.0 CONCLUSION

This series of experiments successfully demonstrated essential aspects of serial communication between Arduino and Python

Part 1 successfully established real-time serial communication between the Arduino and Python for data monitoring. The system converted the potentiometer's analog input into digital values, transmitted them through USB, and visualized them in Python.

In part 2, the concept was extended to bidirectional interaction, where the potentiometer controlled both a servo motor and an LED while simultaneously transmitting the same data to Python for visualization. This showed the successful integration of input sensing, output actuation, and data monitoring in one synchronized system.

Overall, both experiments demonstrated the importance of serial communication in mechatronic systems, emphasizing how sensors, actuators, and software can work together.

18.0 RECOMMENDATIONS

This experiment can be improved in a few ways for better results and learning. Real-time graphs for both the potentiometer and servo motor would help us see the data changes more clearly. A noise-reduction method, such as a simple filter, could also make the readings more stable. Using a graphical user interface would make controlling the servo easier instead of typing the angle manually. Future upgrades may include wireless communication like Bluetooth or Wi-Fi to remove the need for cables. Adding more sensors or actuators can also help us explore more advanced applications. These improvements would help make the system more accurate, easier to use, and closer to real industry setups.

19.0 REFERENCES

[1] https://lastminuteengineers.com/servo-motor-arduino-tutorial/

How Servo Motor Works & Interface It With Arduino

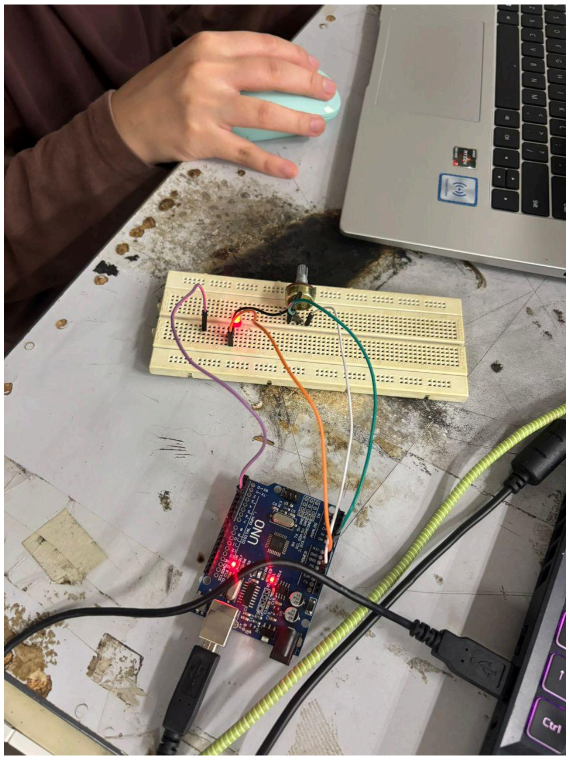
[2] https://docs.arduino.cc/tutorials/

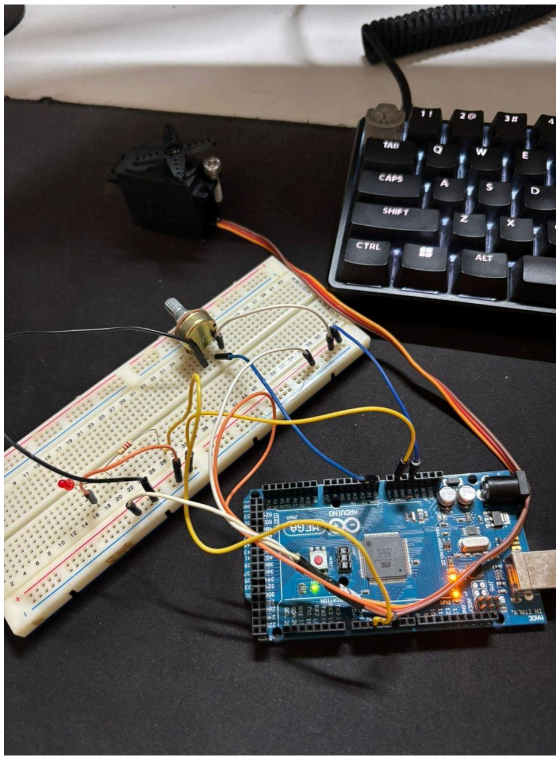
Tutorials - From beginner to Advanced

[3] https://www.instructables.com/Arduino-Servo-Motors/

Arduino Servo Motors

APPENDICES





ACKNOWLEDGEMENTS

Special thanks to DR WAHJU SEDIONO for their guidance and support during this experiment .

STUDENTS'S 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 specific in 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 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 individual's contributor 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:	[illegible]	Read	/
Name:	MUHAMMAD IMRAN BIN VEDDIN	Understand	/
Matric Number:	2316409	Agree	/
Contribution:	Abstract, Conclusion Experiment 1 : Materials and Equipment , Experiment Setup , Methodology, Result		

Signature:	[illegible]	Read	/
Name:	ALYA KHAIRAH BINTI KHAIRUL JAMIL	Understand	/
Matric Number:	2317100	Agree	/
Contribution:	Introduction , Recommendation Experiment 1 : Data Collection , Data Analysis and Discussion Experiment 2 : Results		

Signature:	[illegible]	Read	/
Name:	MUHAMMAD IRFAN RUSHDI BIN ASRI	Understand	/
Matric Number:	2319011	Agree	/
Contribution:	Assembly of components Experiment 2: Materials and Equipment, Experimental setup, Methodology, Data collection, Data Analysis, Discussion		