LAB REPORT 3a :
SERIAL COMMUNICATION (POTENTIOMETER)

GROUP 5

MCTA 3203

SEMESTER 1 2024/2025

MECHATRONICS SYSTEM INTEGRATION

DATE OF SUBMISSION: 23 OCTOBER 2024

NO	NAME	MATRIC NUMBER
1.	MUHAMMAD AFIQ ADHAM BIN MOHD NADZRI	2227531
2.	MUHAMMAD AMIN BIN MOHAMAD RIZAL	2217535
3.	MUHAMMAD AFIQ BIN MOHD ASRI	2212541
4.	MUHAMMAD AKMAL BIN MOHD FAUZI	2214077
5.	MUHAMMAD AFIQ IKHWAN BIN NOR SHAHRIZAL	2215897

TABLE OF CONTENT

INTRODUCTION

The objective of this experiment is to transfer data between a computer and a microcontroller using a potentiometer and an LED. By sending the readings from the potentiometer connected to an Arduino to a Python script via a USB connection, we aim to learn how data is exchanged with the code uploaded to the microcontroller.

ABSTRACT

This experiment investigates the data transfer mechanisms between a computer and a microcontroller using a potentiometer and an LED. By connecting the potentiometer to an Arduino and transmitting its analog readings to a Python script via USB, the study demonstrates the key principles of serial communication within embedded systems. The Arduino processes the readings from the potentiometer, while the Python script visualizes this data in real-time, facilitating effective monitoring and control. This practical experience enhances our comprehension of the interactions between hardware and software, emphasizing how microcontrollers can interface with external devices to facilitate data exchange.

MATERIALS AND EQUIPMENT

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

EXPERIMENTAL SETUP

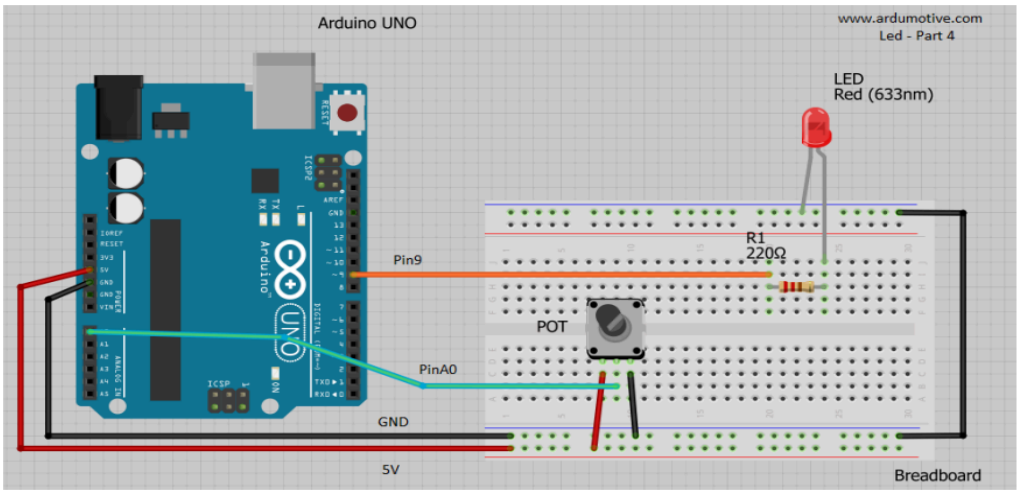


Fig 1

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.

METHODOLOGY

1. Setup for Arduino Board
    - Connect the potentiometer to the Arduino. Wire one end of the potentiometer to 5V, the other end to GND, and the centre pin to an analog input pin on the Arduino A0
    - Connect the LED to the Arduino, the negative part to GND, and the positive part to a resistor that is connected to pin 9.
    - Connect the Arduino to your computer via a USB cable.

2. Software Programming Arduino IDE
    - Write a program in Arduino IDE to:
        - Define the pins that connected to the potentiometer and LED as potPin=A0 and ledPin = 9
        - Initialize serial communication at pin 9 to read the potentiometer value and send it over the serial port.
        - Include a potentiometer function to control the LED using a potentiometer.
        - Upload the Arduino sketch to Arduino Uno

3. Arduino Execution
    - Control LED brightness using the potentiometer.
    - Open Serial Plotter in Arduino IDE to monitor the data received from Arduino board using 9600 baud rate.
    - Record the graph of the Serial Plotter.

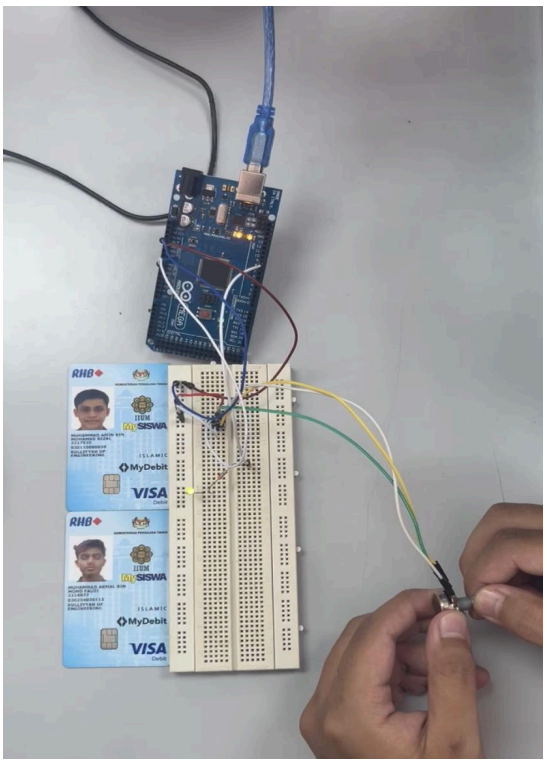
4. Python Execution
    - Write a program using Python to read the potentiometer data from the Arduino via the serial port.

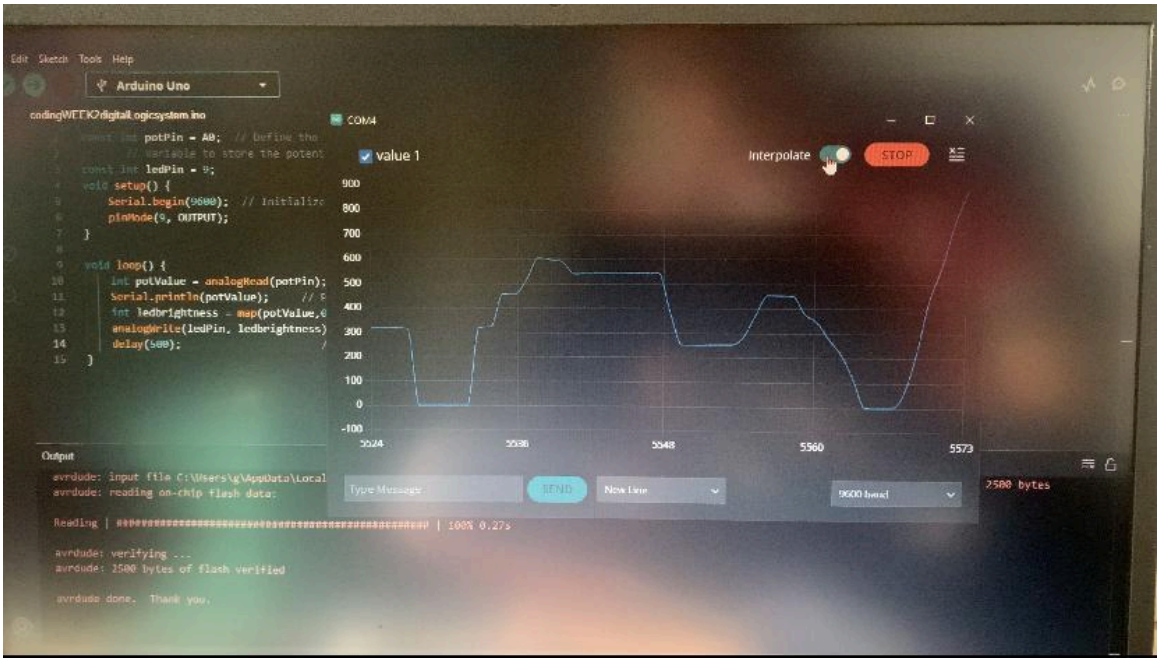
- Turn the potentiometer knob and record the potentiometer values.
- Compare the values of the potentiometer between Python and Arduino IDE.

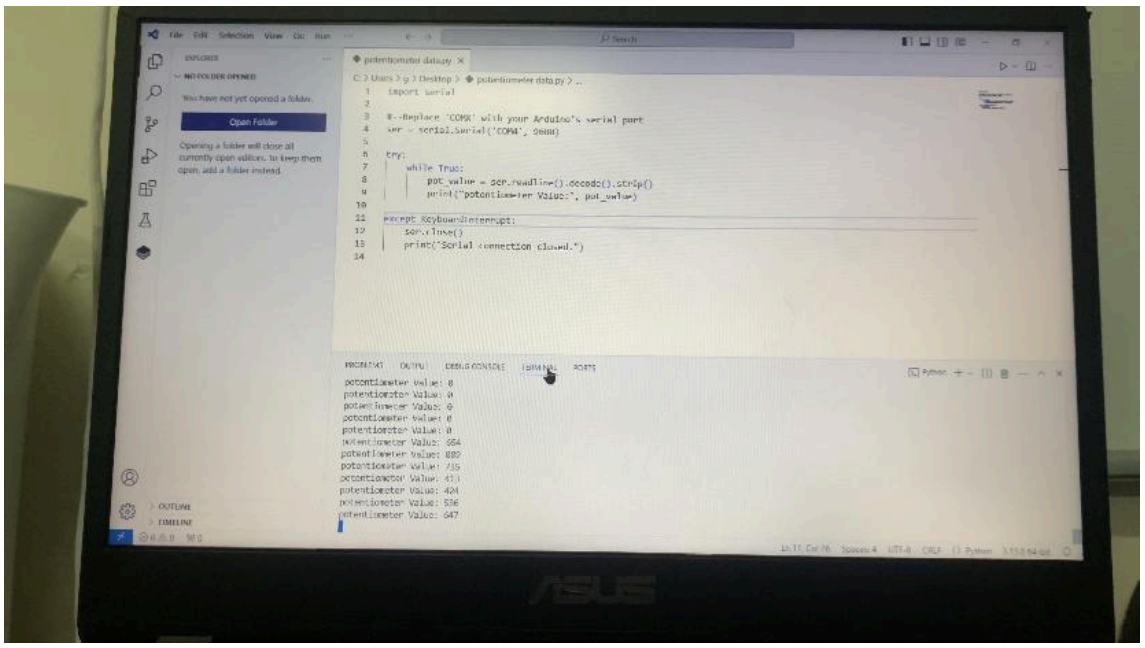
RESULT

The results of this experiment were acquired by constructing a basic circuit using a potentiometer to adjust the brightness of an LED. The potentiometer reads the value, which is then used to compute the LED's brightness. The brightness is calculated by dividing the potentiometer value by four. This value is then passed as an input to the function that controls the LED's brightness. To debug the potentiometer, the value was printed to the serial monitor. Overall, the experiment successfully shows how to utilize a potentiometer to regulate the brightness of an LED.

The following diagrams are the connection of the LED and potentiometer circuit, the Serial plotter from Arduino IDE, and the Output Serial from the Python (Visual Studio Code) respectively.




Edit Sketch Tools Help
Arduino Uno
codingWEEK2digitalLogicsystem.ino
const int potPin = A0; // Define the
// variable to store the potent
const int ledPin = 9;
void setup() {
Serial.begin(9600); // Initialize
pinMode(9, OUTPUT);
}
void loop() {
int potValue = analogRead(potPin);
Serial.println(potValue); //
int ledbrightness = map(potValue,
analogWrite(ledPin, ledbrightness)
delay(500);
}
COM4
value 1
Interpolate
STOP
900
800
700
600
500
400
300
200
100
0
-100
5524
5536
5548
5560
5573
Output
avrdude: input file C:\Users\g\AppData\Local
avrdude: reading on-chip flash data:
Reading | ## | 100% 0.27s
avrdude: verifying ...
avrdude: 1500 bytes of flash verified
avrdude done. Thank you.
Type Message
SEND
New Line
9600 baud
2500 bytes



Video Link:

(For the LED and potentiometer demonstration):

https://github.com/GROUP5-MSI/WEEK-3-Serial-Communication/blob/main/Video%20for%20LED%20%26%20Potentiometer%20demo.MP4

(For the Arduino IDE Serial Plotter):

https://github.com/GROUP5-MSI/WEEK-3-Serial-Communication/blob/main/Video%20for%20Arduino%20IDE%20serial%20plotter%202.mp4

(For the Python/Visual Studio Code Output):

https://github.com/GROUP5-MSI/WEEK-3-Serial-Communication/blob/main/Video%20for%20Python%20Output%202.mp4

QUESTION

To present potentiometer readings graphically in your Python script, you may enhance your code by introducing the capability to generate and showcase a graph. This graphical visualization can deliver a more intuitive and informative perspective for data interpretation. Be sure to showcase the steps involved in your work (Hint: use matplotlib in your Python script).

Python Code:

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

# To define serial port and baud rate
ser = serial.Serial('COM4', 9600)

# Initialize lists to store data for plotting
x_data = []
y_data = []

# Set up the plot
plt.ion()  # Turn on interactive mode
fig, ax = plt.subplots()
line, = ax.plot(x_data, y_data, label='Potentiometer Reading')
ax.set_xlabel('Time (s)')
ax.set_ylabel('Potentiometer Value')
ax.set_ylim(0, 1023)  # Assuming potentiometer value range
ax.legend()

# Time tracking
start_time = time.time()

try:
    while True:
        # Read potentiometer value from Arduino
        if ser.in_waiting > 0:  # Check if data is available to read
            pot_value = ser.readline().decode().strip()  # Read and decode
            pot_value = int(pot_value)  # Convert to integer

            # Record time and potentiometer value
            current_time = time.time() - start_time
```

```
            x_data.append(current_time)
            y_data.append(pot_value)

            # Update the plot
            line.set_xdata(x_data)
            line.set_ydata(y_data)
            ax.relim()  # Recalculate limits
            ax.autoscale_view()  # Autoscale the view
            plt.draw()
            plt.pause(0.1)  # Pause to update the plot

except KeyboardInterrupt:
    pass  # Handle keyboard interrupt

finally:
    ser.close()  # Close the serial connection
    plt.ioff()  # Turn off interactive mode
    plt.show()  # Show the final plot
    print("Serial connection closed.")
```

This Python code uses Matplotlib for real-time data visualization. It retrieves potentiometer values from an Arduino via serial communication, demonstrating Python's ability for interactive data analysis. The code updates a plot in real-time, clearly displaying the potentiometer readings. This method is useful not only for monitoring sensor data but also for teaching users how to understand and use real-time data visualization tools.

Moreover, the code highlights Python's easy integration with hardware, making it a strong tool for data processing and visualization. By showcasing this feature, users can use Python to create engaging visual representations of their data, improving their understanding of how hardware and software work together in real-time.

DISCUSSION

The experiment used several important hardware components, beginning with the breadboard, which acted as a foundation for connecting various electronic elements. The breadboard facilitated easy wiring and reconfiguration, allowing for a flexible setup when working with the Arduino Mega 2560. Serving as the central microcontroller for the project, the Arduino Mega 2560 was equipped with multiple input/output pins and could handle complex tasks. Male-to-male jumper wires connected the components on the breadboard to the Arduino, providing reliable electrical connections.

In addition to the breadboard and Arduino, the project featured a potentiometer, an adjustable variable resistor with three terminals. This component allowed users to modify the resistance, which in turn changed the voltage output as they rotated the knob. The experiment also included an LED, a semiconductor device that lights up when current passes through it, serving as a visual indicator of the potentiometer’s position. Furthermore, a 220-ohm resistor was added to limit the current flowing through the LED, safeguarding it from damage while ensuring it operates efficiently.

The electrical configuration established a direct connection between the potentiometer and the Arduino Mega. The potentiometer was connected by linking one terminal to the 5V power supply, another to the ground (GND), and the middle terminal (the wiper) to analog input pin A0. This arrangement created a voltage divider circuit that produced a variable voltage signal depending on the potentiometer's position. The Arduino utilized its analog-to-digital converter (ADC) to measure this analog voltage. Using the analogRead() function, it converted the voltage into a digital value ranging from 0 to 1023.

For communication with a computer, serial communication was implemented via USB. The Arduino transmitted the potentiometer readings to a Python script on the computer, which displayed these values. Maintaining a baud rate of 9600 in both the Arduino code and the Python script was essential to prevent communication problems. This configuration allowed for real-time monitoring of the potentiometer’s position and the brightness of the LED.

The Arduino code executed a straightforward series of tasks: it initialized serial communication, read the potentiometer value, and adjusted the LED brightness accordingly. The `analogRead(Pot_pin)` function gathered data from the potentiometer, while the `analogWrite(Led_pin, brightness)` function employed Pulse Width Modulation (PWM) to regulate the LED's brightness. At the same time, the Python script established a serial connection with the Arduino, continuously reading and displaying the potentiometer value in a user-friendly format. The design of both codes facilitated efficient and error-free data flow, enhancing the overall performance of the system.

Improvement:

Although the current configuration works well, there are a few possible improvements. Robustness could be increased by enhancing error handling in both the Python and Arduino code. For example, adding checks to guarantee successful serial connections or managing unexpected data types could stop problems during operation. Furthermore, creating a graphical user interface for the Python script might greatly improve user engagement and make the system more aesthetically pleasing and accessible. Increased participation in the project would result from such improvements, which would make the experience more polished and user-friendly. There are also the changes from components that we use, like Arduino uno and Arduino mega, both being used to test the functionality of code and circuit connection.

CONCLUSION

In conclusion, this series of experiments demonstrates the principles of serial communication between a computer and a microcontroller using Python and Arduino. By interfacing a potentiometer, we explored two fundamental aspects of sensors and actuators in embedded systems.

We could configure a potentiometer to send analog readings to a Python script using a USB connection in this experiment. We also gained a deeper understanding of the underlying principles of microcontroller programming, data transfer protocols, and the interdisciplinary nature of electronics and computer science. This setup allowed us to monitor real-time data, highlighting the ease with which analog inputs can be integrated into software applications for various purposes, such as data logging or control systems.

RECOMMENDATION

Several proposals would improve the experiment's reliability and effectiveness. First, double-check the connections before powering on the circuit, as incorrect wiring might result in poor functioning or damage. Next, the lab must provide a multimeter to examine voltage levels at various locations, which could reveal the source of problems in the power supply or improper resistor values. It is critical to ensure that the resistor values are accurate. Erroneous values will result in a dull display or damage. Use the resistor required to prevent the LED from being overpowered.

Once the fundamental functionality is built, testing various scenarios can identify potential issues far earlier. Following these criteria results in a much more minimal, productive, and successful experimental experience, as well as significant learning opportunities.

ACKNOWLEDGEMENTS

A special thank you to Dr. Wahju Sediono and Dr. Zulkifli Bin Zainal Abidin, my teaching assistant, and peers, for their outstanding assistance and support in completing this report. Their advice, input, and expertise have had a significant impact on the quality and comprehension of this work. Their time, patience, and dedication to my academic progress are deeply appreciated.

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 acknowledgment, 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:
Name: MUHAMMAD AFIQ ADHAM BIN MOHD NADZRI
Matric Number: 2227531

Read ☑
Understand ☑
Agree ☑

Signature:
Name: MUHAMMAD AMIN BIN MOHAMAD RIZAL
Matric Number: 2217535

Read ☑
Understand ☑
Agree ☑

Signature:
Name: MUHAMMAD AFIQ BIN MOHD ASRI
Matric Number: 2212541

Read ☑
Understand ☑
Agree ☑

Signature:
Name: MUHAMMAD AKMAL BIN MOHD FAUZI
Matric Number: 2214077

Read ☑
Understand ☑
Agree ☑

Signature: Afiq
Name: MUHAMMAD AFIQ IKHWAN BIN NOR SHAHRIZAL
Matric Number: 2215897

Read ☑
Understand ☑
Agree ☑