



الجامعة الإسلامية العالمية ماليزيا
INTERNATIONAL ISLAMIC UNIVERSITY MALAYSIA
يُونِيسَيْتِي اِسْلَامُ اَنْتَارَايَحْسَا مِلْدِسِيَا
Garden of Knowledge and Virtue

MCTA 3203

MECHATRONIC SYSTEM INTEGRATION

SECTION 1

SEMESTER 2 2024/2025

LECTURER:

SIR WAHJU SEDIONO

SIR ZULKIFLI BIN ZAINAL

LAB REPORT 3A:

SERIAL COMMUNICATION (POTENTIOMETER)

GROUP 2

DATE OF SUBMISSION:

24 MARCH 2025

GROUP MEMBERS	MATRIC NUMBERS
AMIRUL IMAN BIN MAT KHIR	2316633
AHMAD IRFAN FAHMI BIN AHMAD USRI	2318631
MUHAMAD NUR HAFIZ BIN ROSLY	2317001
FAKHRULLAH BIN YUZAIRI	2220289

TABLE OF CONTENTS

INTRODUCTION	2
ABSTRACT	2
MATERIALS AND EQUIPMENT	3
EXPERIMENTAL SETUP	4
METHODOLOGY	4
RESULT	5
QUESTION	9
RECOMMENDATION	10
1. Hardware Improvements	10
2. Software Improvements	10
CONCLUSION	10
ACKNOWLEDGEMENT	11
STUDENT'S DECLARATION	12

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

ITEM	AMOUNT
LED	1
ARDUINO UNO MEGA	1
BREADBOARD	1
220 OHM RESISTANCE	7
JUMPER WIRES	9
POTENTIOMETER	1

EXPERIMENTAL SETUP

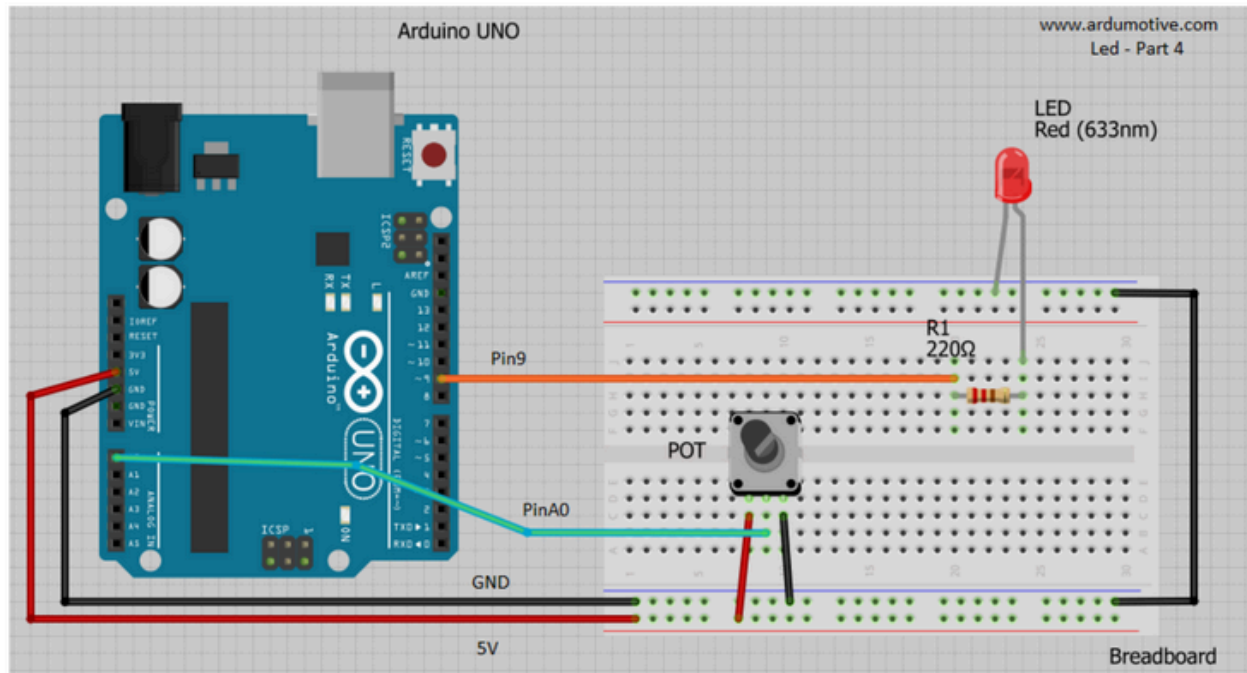


Fig 1

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

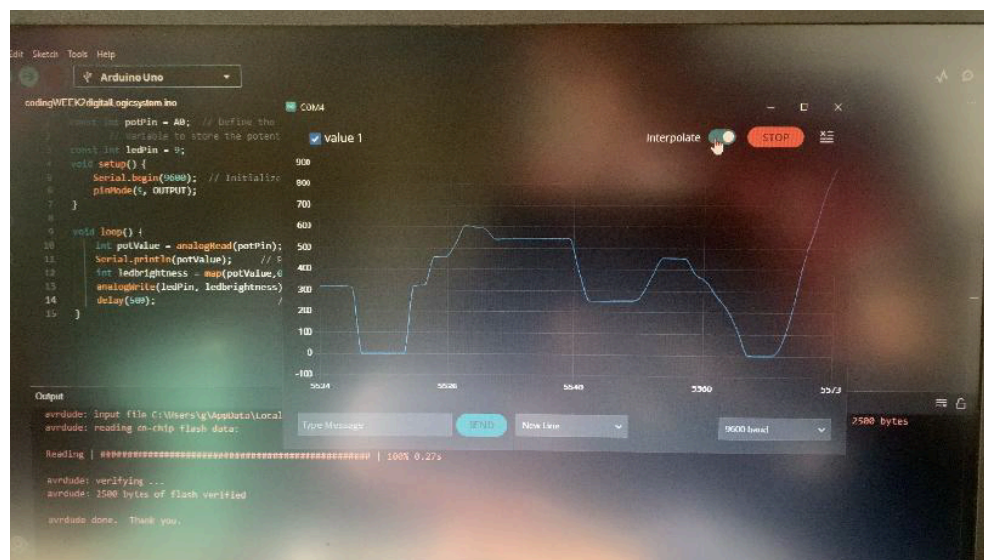
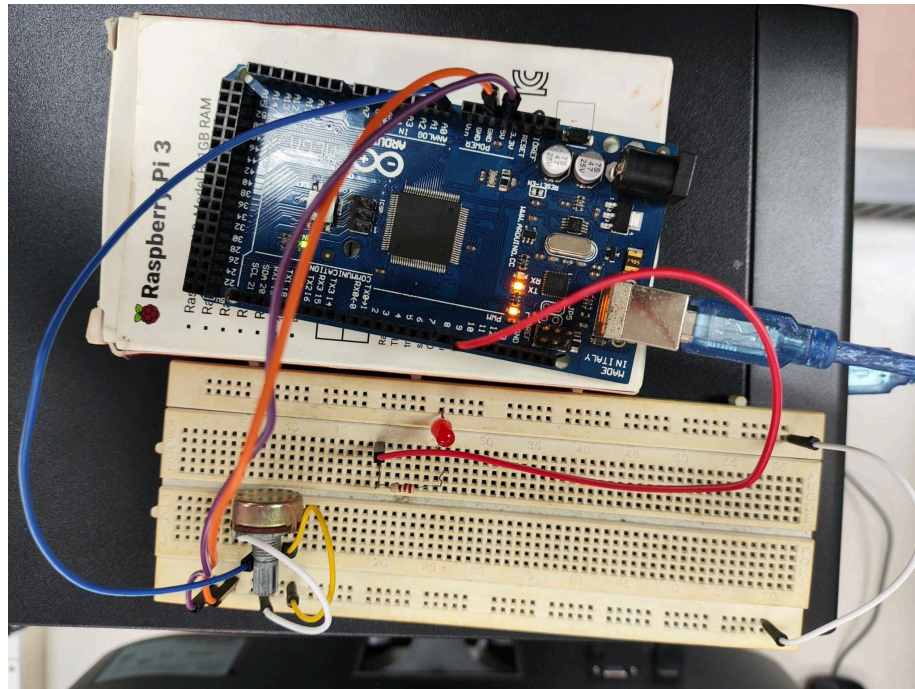
- Control LED brightness using the potentiometer.
- Open Serial Plotter in Arduino IDE to monitor the data received from Arduino board using 9600 baud rate.
- Record the graph of the Serial Plotter.

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.



Video Link:

(For the LED and potentiometer demonstration):

<https://github.com/Irfanf248/MSI-GROUP2-EXPERIMENT-Week3/blob/main/3A%20EXP%20VIDEO.mp4>

(For the Python/Visual Studio Code Output):

<https://github.com/Irfanf248/MSI-GROUP2-EXPERIMENT-Week3/blob/main/python%20code%2003A.mp4>

DISCUSSION

In this project, we investigated the principles of serial communication between a computer's Python script and an Arduino microcontroller. Sending potentiometer readings from the Arduino to Python so that the data could be analysed and shown was the aim. A crucial component of mechatronics and embedded systems design, this exercise illustrated how microcontrollers and computers may collaborate to develop interactive systems.

The hardware setup involved connecting a potentiometer to the Arduino's analog input pin. The potentiometer acted as a simple sensor, providing variable voltage readings based on the knob's position. The Arduino code read these values and sent them over the serial port at a defined baud rate. Proper configuration of the baud rate was crucial, as both the Arduino and Python script needed to match for accurate data transmission.

The pyserial library was utilised on the Python end to connect to the Arduino. The potentiometer values were shown in the terminal after the script read and decoded the incoming data. This demonstrated Python's hardware interface capabilities, particularly when paired with modules

that streamline communication protocols. Applications such as automation, control systems, and data visualisation are made possible by real-time data processing.

The significance of efficiently handling serial communication was one of the main lessons learnt. The Python script and Arduino Serial Plotter, for example, are unable to access the serial port at the same time, highlighting the necessity of cautious resource management to prevent conflicts. Nonetheless, the Serial Plotter helped with debugging and comprehending sensor behaviour by providing real-time data visualisation.

Overall, this experiment provided a solid introduction to integrating hardware and software in mechatronics systems. It showcased the practical aspects of serial communication and demonstrated how simple components like a potentiometer can be used to create interactive systems. The skills learned can be extended to more complex projects, such as controlling motors or building user interfaces, making Arduino and Python a versatile platform for prototyping and development.

QUESTION

To enhance the Python script for real-time graphical visualization of the potentiometer readings, we can use the `matplotlib` library. Below are the steps to implement this:

```
1  import serial
2  import matplotlib.pyplot as plt
3  from collections import deque
4  import time
5
6  # Initialize serial connection (replace 'COMX' with your Arduino's port)
7  ser = serial.Serial('COM3', 9600)
8
9  # Set up the plot
10 plt.ion() # Enable interactive mode
11 fig, ax = plt.subplots()
12 x_data = deque(maxlen=100) # Store last 100 readings
13 y_data = deque(maxlen=100)
14 line, = ax.plot([], [], 'b-') # Blue line for potentiometer values
15 ax.set_title("Real-Time Potentiometer Readings")
16 ax.set_xlabel("Time (samples)")
17 ax.set_ylabel("Analog Value (0-1023)")
18 ax.set_ylim(0, 1023) # Arduino analog range
19
20 try:
21     while True:
22         # Read and decode data from Arduino
23         raw_data = ser.readline().decode().strip()
24         if raw_data.isdigit(): # Ensure valid data
25             value = int(raw_data)
26             y_data.append(value)
27             x_data.append(len(y_data)) # Track sample count
28
29         # Update plot
30         line.set_data(x_data, y_data)
31         ax.relim()
32         ax.autoscale_view()
33         plt.pause(0.01) # Short delay for smooth updates
34
35 except KeyboardInterrupt:
36     ser.close()
37     print("Serial connection closed.")
38     plt.ioff()
39     plt.show() # Keep plot open after closing
```

Explanation of Key Steps

- `matplotlib.ion()`: Enables real-time interactive plotting.
- `deque`: Efficiently stores the last 100 readings (adjustable).
- `ser.readline().decode()`: Reads and decodes serial data from Arduino.
- **Plot Updates**: The graph dynamically updates as new data arrives.
- **KeyboardInterrupt Handling**: Ensures clean exit when stopping the script.

Expected Output

When running the script:

- A live-updating graph will display the potentiometer values.
- Turning the knob will show immediate changes in the plot.
- The X-axis represents sample count, while the Y-axis shows the analog value (0–1023).

.Further Enhancements

- **Add Timestamps:** Use `time.time()` for real-time X-axis values.
- **Multiple Sensors:** Extend the script to plot multiple analog inputs.
- **Smoothing:** Apply a moving average to reduce noise in readings.

RECOMMENDATION

To enhance the performance and reliability of the potentiometer-controlled LED system, several improvements can be made across hardware, software, and user experience.

1. Hardware Improvements

-Component Replacement and Secure Connections: Ensure all components, especially jumper wires, are in good condition and properly connected to avoid performance issues.

-Verify Resistor Values: Use a 220-ohm resistor for current limiting to protect the LED and ensure consistent operation.

2. Software Improvements

-Optimize Code with Functions: Refactor the code to use functions for repetitive tasks like reading potentiometer values and controlling the LED, making the code cleaner and easier to maintain.

-Use Serial Monitor for Debugging: Utilize the Serial Monitor for real-time debugging and monitoring of potentiometer data.

CONCLUSION

The experiment on serial communication between Python and an Arduino successfully demonstrated the ability to transmit potentiometer readings from the microcontroller to a computer system. This process highlights the importance of establishing a reliable

communication link between hardware and software to collect, process, and visualize sensor data. Through this experiment, key concepts in embedded systems, serial communication, and real-time data monitoring were effectively explored.

One of the main findings was the seamless transmission of data using the serial interface. By employing a Python script and the `pyserial` library, the potentiometer readings were read and displayed in real time, proving the efficiency of serial communication in interfacing external sensors with a computer. The accurate reception and processing of analog sensor data provide a strong foundation for more advanced applications in automation, robotics, and control systems.

Furthermore, the experiment emphasized the significance of structured coding in both Python and Arduino environments. The integration of a simple script to read serial data demonstrated the practical use of software tools to interpret physical signals. The ability to visualize data through the Arduino Serial Plotter also reinforced the importance of graphical representation in understanding real-time sensor behavior.

In addition, the experiment lays the groundwork for future enhancements, such as incorporating graphical data representation using Python's `matplotlib` library. By extending the code to plot potentiometer readings dynamically, users can gain a more intuitive understanding of sensor variations over time. This capability is essential for applications that require precise monitoring and analysis of sensor data, such as industrial automation and scientific research.

In conclusion, this experiment provided valuable insights into the integration of sensors with microcontrollers and computer-based systems. The successful transmission and display of potentiometer readings highlight the practical applications of serial communication in various domains. By expanding on these principles, more sophisticated data acquisition and control systems can be developed, contributing to the advancement of mechatronics and embedded system technologies

ACKNOWLEDGEMENT

We would like to express our sincere gratitude to Sir Wahju Sediono for their guidance and support throughout this project. Our thanks also extend to the teaching assistants, for their constructive feedback and assistance, which greatly contributed to the completion of this work.

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**.

Name: Amirul Iman Bin Mat Khir

Matric Number: 2316633

Signature: 

Contribution: Title, introduction, equipment setup, acknowledgement

Read ☒

Understand ☒

Agree ☒

Name: Ahmad Irfan Fahmi Bin Ahmad Usri

Matric Number: 2319531

Signature: 

Contribution: Result, Discussion, abstract

Read ☒

Understand ☒

Agree ☒

Name: Muhamad Nur Hafiz Bin Rosly

Matric Number: 2317001

Read ☒

Understand ☒

Signature: 
Contribution: Recommendation, methodology


Agree ☒

Name: Fakhrolah Bin Yuzairi

Read ☒

Matric Number: 2220289

Understand ☒

Signature: 
Contribution: Question, Discussion,

Agree ☒