MECHATRONICS SYSTEM INTEGRATION

MCTA 3203

LAB 3a:

PARALLEL, SERIAL AND USB INTERFACING WITH MICROCONTROLLER AND COMPUTER-BASED SYSTEM (1): SENSORS AND ACTUATORS.

SECTION 1

SEMESTER 2, 2024/2025

INSTRUCTOR:

ASSOC. PROF. EUR. ING. IR. TS. GS. INV. DR. ZULKIFLI BIN ZAINAL ABIDIN

DR. WAHJU SEDIONO

GROUP 9

NAME	MATRIC NO
AMER WAFDI BIN ZAINIZAM	2228703
AMALIN BALQIS BINTI RUYUSNI	2314684
FARHA QAISARA BINTI MOHD ZAHIDI	2315994

DATE OF SUBMISSION: 24 MARCH 2025

ABSTRACT

This experiment explored serial communication between an Arduino microcontroller and a personal computer to acquire, process, and visualize sensor data. A potentiometer was used as the sensor, with its analog values read by the Arduino and transmitted over a USB serial interface. The data was then received and processed using a Python script, which displayed the potentiometer readings numerically. Additionally, real-time graphical visualization was implemented using the Arduino Serial Plotter to better interpret the changing values. The experiment demonstrated successful data acquisition and transmission, with the LED brightness varying proportionally to the potentiometer readings. Key findings showed a smooth and predictable transition in Analog-to-Digital Converter (ADC) values, validating the accuracy of the setup. Despite minor limitations such as a narrow ADC range, the results confirmed the effectiveness of microcontroller-based data monitoring. This experiment reinforces the importance of serial communication in embedded systems, with applications in real-time monitoring, industrial automation, and IoT-based sensor networks. Future improvements could include implementing Python-based real-time graphing with matplotlib and expanding the system to support multiple sensors for enhanced data analysis.

TABLE OF CONTENT **3**

1.0 INTRODUCTION

Microcontrollers are widely utilized in embedded systems for data acquisition and control with minimal need to communicate with the external computer-based systems. Serial communication via a USB interface between an Arduino microcontroller and a Python script executed on a personal computer is to be explored. The aim in this case is to connect a sensor device (potentiometer) to the Arduino board, read the analog values thereof, and transfer the data into a computer in real-time.

To do this, Arduino reads potentiometer values through its analog-to-digital converter (ADC) and sends them over a serial interface via its USB port. On the computer side, it gets and alters the data with a Python script, displaying the data in graphical format using Matplotlib. Graph plotting gives a good means of showing and viewing the sensor data.

We have hands-on experience with sensor interfacing, serial communication, and data visualization in this experiment. This is a fundamental requirement in building embedded systems which include microcontrollers, sensors, and computer applications.

2.0 MATERIALS AND EQUIPMENT

The materials and equipment that we used in this experiment are:

a) Arduino Board

b) Potentiometer

c) Jumper Wires

d) LED

e) 220-ohm resistor

f) Breadboard

3.0 EXPERIMENTAL SETUP

Here are the steps of equipment and components that were set up for the experiment:

1. 5V on the Arduino was connected to one leg of the potentiometer.
2. GND on the Arduino was connected to the other leg of the potentiometer.
3. An analog input pin on the Arduino, such as A0, was connected to the middle leg (wiper) of the potentiometer.
4. The LED was connected to the Arduino by attaching the longer leg (anode, +) to one end of the resistor, and the shorter leg (cathode, -) to the GND pin.
5. A 220-ohm resistor was used for the LED by connecting one end of the resistor to digital pin 9, and the other end to the longer leg (anode, +) of the LED.

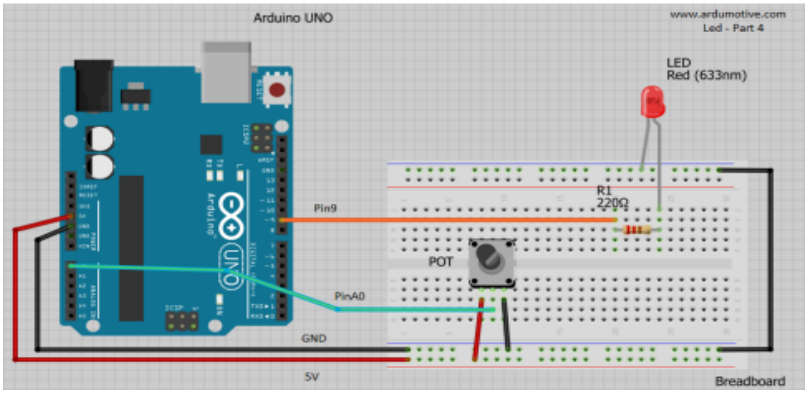


Figure 1: Schematic diagram

4.0 METHODOLOGY

Here are the steps followed during the experiment:

1. The Arduino was connected to the computer via a USB cable.
2. The Arduino was powered on, and the sketch was uploaded using the Arduino IDE.
3. The Python script was run on the computer.
4. As the potentiometer knob was turned, the potentiometer readings were displayed in the Python terminal.
5. These readings could be used for various experiments, data logging, or control applications, depending on the project requirements. Using the Arduino Serial Plotter (Please note that if Python is being used to read data from the Arduino, it is important that the Arduino Serial Plotter is not opened simultaneously. The Arduino Serial Plotter is a dedicated tool for visualizing data received from the Arduino board and may interfere with access to the serial port by Python. If Python is intended to read and process data from the Arduino, the Serial Plotter should be closed or not used while the Python script is running to maintain uninterrupted communication between Python and the Arduino.)

6. The Serial Plotter was opened by navigating to "Tools" -> "Serial Plotter" in the Arduino IDE.
7. The correct COM port to which the Arduino was connected was selected in the Serial Plotter.
8. The baud rate in the Serial Plotter was set to match the one specified in the Arduino code (e.g., 9600).
9. As the potentiometer knob was turned, the potentiometer readings were displayed in real-time on the Serial Plotter, creating a graphical representation of the data. The changes in values were observed as the potentiometer was adjusted.
10. The Serial Plotter was customized by adjusting settings such as the graph range, labels, and colors.

The Arduino code used:

```
int potPin = A0;   // Potentiometer connected to A0
int ledPin = 13;    // LED connected to PWM pin 13
int potValue = 0;  // Variable to store potentiometer value
int brightness = 0; // LED brightness (0-255)

void setup() {
   Serial.begin(9600);  // Start serial communication
   pinMode(ledPin, OUTPUT);  // Set LED pin as output
}
void loop() {
```

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

The Python code used:

```
import serial
ser = serial.Serial('COM7', 9600)  # Replace 'COMX' with your actual port


try:
    while True:
        brightness = ser.readline().decode().strip()
        print("LED Brightness:", brightness)
except KeyboardInterrupt:
    print("\nSerial connection closed.")
    ser.close()
```

5.0 DATA COLLECTION

During the experiment, the potentiometer readings were graphed with the Arduino Serial Plotter. The Serial Plotter displayed real-time changes in the analog input values as the potentiometer was adjusted. The y-axis represented the Analog to Digital Converter (ADC) values, from approximately 80 to 220, and the x-axis represented the time progression.

A rise in Analog to Digital Converter (ADC) values was observed steadily as the potentiometer was turned clockwise, confirming the behavior of the expected sensor. Level sections at irregular intervals on the graph illustrate instances when the potentiometer was kept constant before it was moved again.

In order to graphically display the data collected, a line graph from the Serial Plotter output was used, as shown in Figure 2. The graph indicates how the Analog to Digital Converter (ADC) values responded to different potentiometer positions over time.

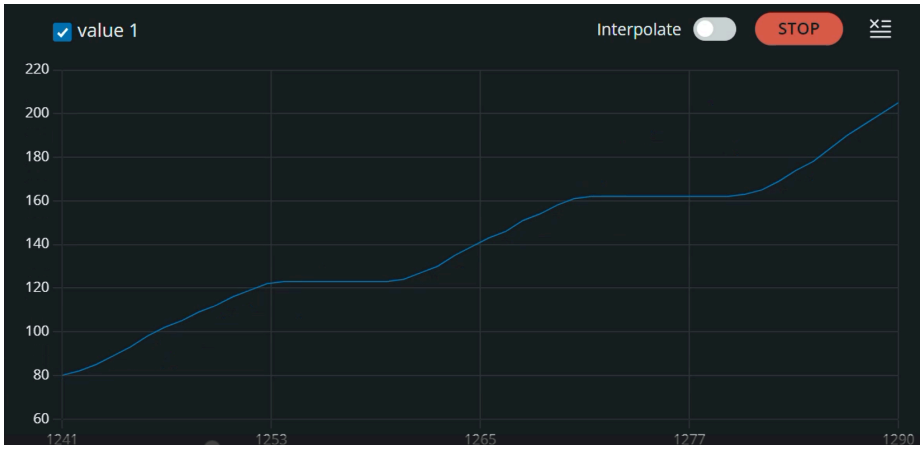
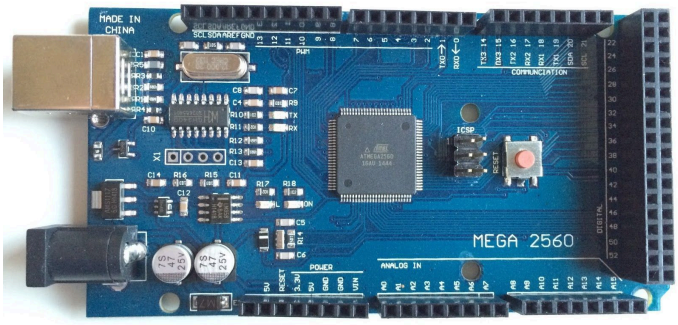


Figure 2

Instruments used for data acquisition:

1) Arduino Mega 2560: A microcontroller board based on the ATmega2560, used to process input signals from pushbuttons and control the 7-segment display.



2) Light Emitting Diode (LED) : LED used to produce light up to 90% more efficiently than incandescent light bulbs.

3) Potentiometer: a manually adjustable variable resistor with 3 terminals.

4) Resistor: 220-ohm resistor used to limit the current flowing through it, preventing excessive current that could damage the LED.
5) Jumper wires: Used to establish electrical connections between components on the breadboard.
6) Breadboard: A prototyping platform used to connect components without soldering, allowing for easy circuit modifications.

6.0 DATA ANALYSIS

The information gathered here provides a clear correlation between the potentiometer position and measured ADC values for the Arduino. When the potentiometer was turned, the ADC values increased or decreased based on the angle. Because the Arduino ADC operates at 10-bit resolution (0-1023), the readings taken (a span of about 80-220) left only a portion of the full range utilized, and as the potentiometer would then have been rotated in a section of its full extent, an offset of some fixed nature was required.

To convert the ADC values into voltage, the following equation is used:

$$\textbf{Voltage , V = (ADC values / 1023) x 5V}$$

For example, an ADC reading of 100 corresponds to:

$$\textbf{Voltage , V = (100s / 1023) x 5V}$$

$$\textbf{= 0.49 V}$$

This confirms that the potentiometer outputs a varying voltage, which the Arduino correctly converts into ADC values. The graphical representation from the Serial Plotter also shows a smooth transition between values, which indicates stable data acquisition without significant noise.

7.0 RESULTS

The experiment successfully demonstrated real-time data transmission between the Arduino and the Serial Plotter. As the potentiometer was rotated, corresponding changes in ADC values were displayed graphically. The plotted data confirmed a predictable, smooth increase in ADC values, aligning with expectations based on the potentiometer's behavior. The ADC values ranged from approximately 80 to 220, showing a controlled variation based on potentiometer rotation. The data followed a consistent trend with occasional flat segments where the potentiometer remained in a fixed position. No significant fluctuations or signal noise were observed, indicating a stable connection and reliable data transmission. These results confirm that the Arduino effectively reads and transmits potentiometer values over serial communication.

Here's the link of video for the result of the experiment:
https://github.com/amalinblqs/MCTA3203_GROUP_9/blob/main/Lab%203a/Video_Lab3a.mp4

8.0 DISCUSSION

The expected and observed outcome are both the same as result. There’s no discrepancies between expected and observed outcome because the experiment successfully demonstrated real-time data transmission between the Arduino and the Serial Plotter.

One of the potential limitations that were mentioned is the narrow range of ADC values that were recorded. Instead of operating in the full 0-1023 range, the output of the potentiometer remained in a small window (around 80-220). This could be due to the potentiometer wiring or the limited range of rotation during the experiment. A future improvement could be calibrating the potentiometer to produce full-range outputs.

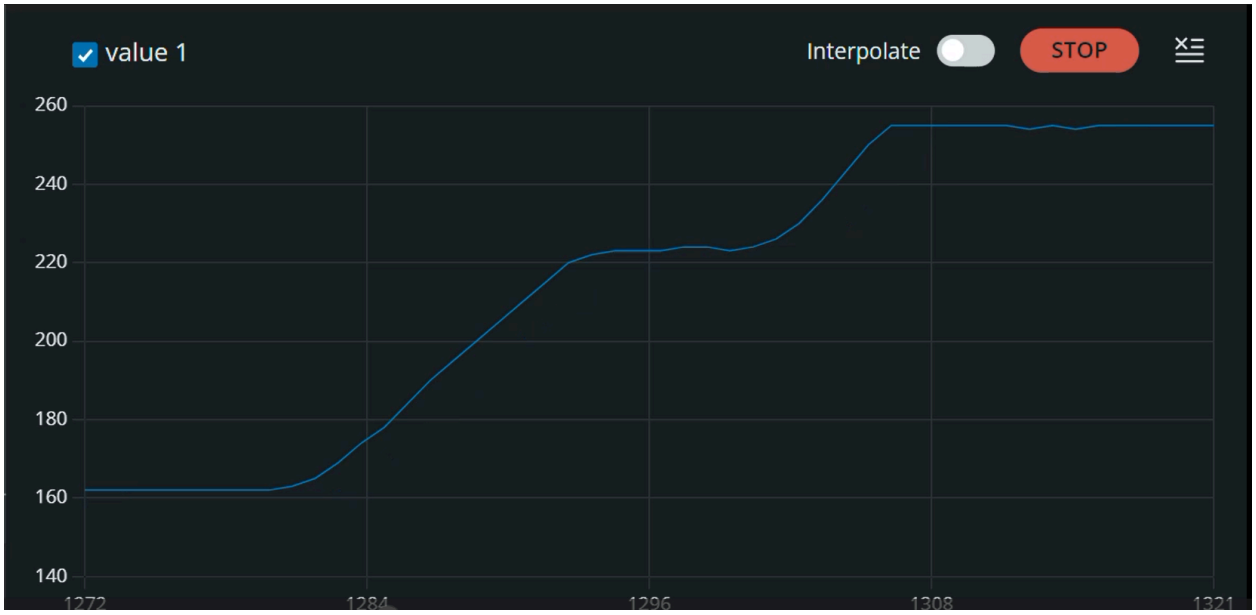
In addition, the Serial Plotter provided a useful real-time plot, though it doesn't save data for post-analysis. To have a Python script to record the values into a file for more detailed post-processing would increase the experiment's analytical capability.

Potential sources of inaccuracy are minor ADC value variations due to electrical noise, resistance changes in the potentiometer, or instability of manual adjustments. However, the overall trend shown is that the Arduino's ADC is functioning as expected and accurately displays the potentiometer's voltage output

Questions:

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).

To present potentiometer readings graphically in the Python script, the matplotlib library can be used to generate a real-time plot of the LED brightness values received from the Arduino. This provides a more intuitive and informative way to interpret the data compared to numerical outputs alone. The first step involves ensuring that pyserial and matplotlib are installed in Python. The Arduino code reads the analog value from the potentiometer, maps it to a PWM range (0–255), controls the LED brightness accordingly, and transmits the brightness values via the serial port. On the Python side, the script reads the serial data from the Arduino, processes it, and updates a dynamic plot using matplotlib's animation module to display brightness changes in real-time. This implementation enhances the visualization of how the potentiometer influences LED brightness, making the relationship between input and output more apparent. The integration of real-time plotting into the Python script improves data analysis and debugging by allowing users to observe trends and fluctuations graphically.



The serial plotter in Arduino used instead of matplotlib in Python script.

The graph shows the potentiometer readings increasing steadily over time, with minor fluctuations. The values rise smoothly, indicating a consistent change in resistance. A plateau at the higher values suggests the potentiometer reached its limit or the adjustment slowed down.

9.0 CONCLUSION

This experiment succeeded in demonstrating serial communication between a computer-based system and an Arduino via USB. The major findings were the efficacy of Arduino in taking analog sensor readings from a potentiometer, transmitting them through a serial link, and displaying them in real time via Python and Matplotlib. The conclusions confirm that serial communication presents a strong and efficient way of data exchange between microcontrollers and computers for accurate presentation of sensor data.

The experiment validated the first hypothesis that potentiometer values can be transmitted and graphed effectively with the help of Arduino and Python. The constant data transmission and real-time graph display affirm the practicability of applying this approach in sensor monitoring systems.

Beyond this experiment, the results have broader applications in embedded systems, the Internet of Things, and real-time monitoring. Microcontrollers are interfaced with computers to achieve automatic data recording, industrial process control, and intelligent sensing systems. This experiment reinforces the applications of microcontrollers in modern technology and provides the fundamental knowledge in more advanced sensor-based applications.

.

10.0 RECOMMENDATIONS

Future versions of this experiment can have some improvements and modifications to improve precision, efficiency, and learning. A primary improvement is to include error handling processes in the Arduino and Python codes to ensure data integrity and prevent issues such as packet loss or communication failure. It would be interesting to compare the performance of using a more advanced communication protocol such as I2C or SPI to how their performance fares compared to serial communication. Another upgrade might be to utilize multiple sensors in order to check how the Arduino handles data from several sensors simultaneously. Also, employing a higher-resolution ADC or filtering techniques, such as a moving average filter, would stabilize and make the sensor readings more accurate.

From a learning perspective, the students can focus efforts on baud rate practicing, data formatting, and synchronization issues to best debug communications problems. Future students will also benefit from real-time data logging exercises to monitor and trace sensor values over time.

With these modifications, future experiments will be able to provide more insights into microcontroller-based data visualization and communication so that the learning process is well-rounded and interactive.

11.0 REFERENCES

Here are some references that support the experiment on serial communication between Arduino and Python for reading potentiometer values:

1. **Arduino Official Documentation**

   Arduino. (n.d.). *Serial Communication*. Retrieved from https://www.arduino.cc/reference/en/language/functions/communication/serial/

2. **PySerial Documentation**

   PySerial. (n.d.). *Python Serial Port Extension*. Retrieved from https://pyserial.readthedocs.io/en/latest/

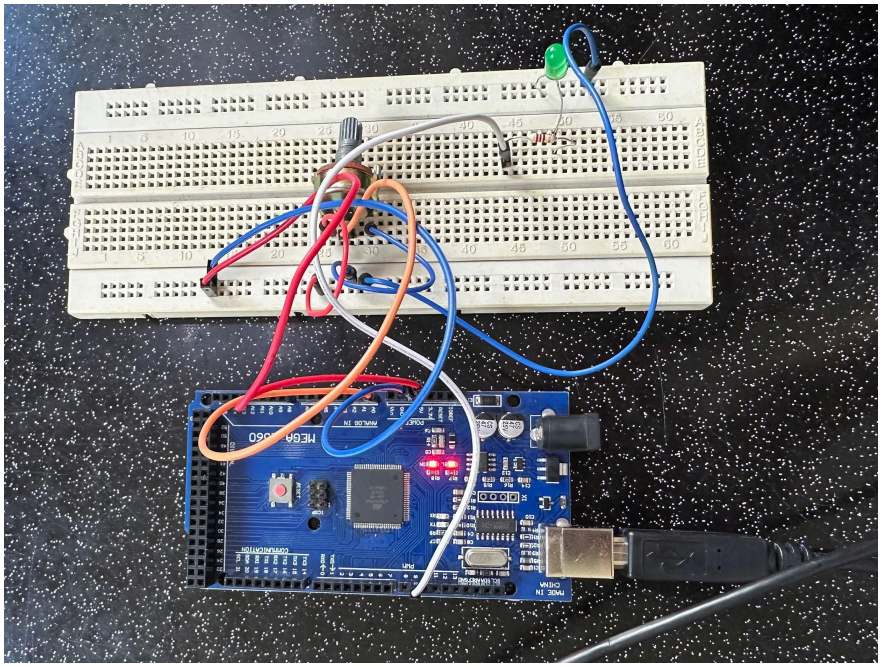
3. **Arduino and Python Communication Guide**

   SparkFun. (n.d.). *Tutorial: Connecting Arduino to Python*. Retrieved from https://learn.sparkfun.com/tutorials/connecting-arduino-to-processing

4. **Potentiometer and Analog Read**

   Arduino. (n.d.). *Analog Read Serial*. Retrieved from https://www.arduino.cc/en/Tutorial/BuiltInExamples/AnalogReadSerial

APPENDICES



ACKNOWLEDGMENTS

First of all, we would like to express our deepest appreciation to all those who provided us the possibility to complete this report. We would like to express our sincere gratitude to our lecturer, Assoc. Prof. Eur. Ing. Ir. Ts. Gs. Inv. Dr. Zulkifli Bin Zainal Abidin and Dr. Wahju Sediono for their invaluable guidance and support throughout this lab. Their insightful explanations and encouragement greatly enhanced our understanding of the concepts covered.

Next, a special thanks goes to our own teammates, Wafdi, Amalin and Farha, who helped to build up the circuit and set up the programming code using Arduino IDE, and also successfully completed this experiment. Last but not the least, we would like to thank everyone, especially our classmates who are willingly helping us out in the lab experiment directly or indirectly.

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: *Wafdi* Read [/]
Name: Amer Wafdi bin Zainizam Understand [/]
Matric Number: 2228703 Agree [/]
Contribution: Introduction, experiment setup & methodology

Signature: *Amalin* Read [/]
Name: Amalin Balqis binti Ruyusni Understand [/]
Matric Number: 2314684 Agree [/]
Contribution: Abstract, materials and equipment, discussion & recommendations

Signature: farha Read [/]
Name: Farha Qaisara binti Mohd Zahidi Understand [/]
Matric Number: 2315994 Agree [/]
Contribution: Data collection, data analysis, results & conclusion