MECHATRONICS SYSTEM INTEGRATION (MCTA 3203)

SEMESTER 1 2024/2025

WEEK 3A: SERIAL COMMUNICATION

SECTION 2

GROUP 8

LECTURER: ZULKIFLI BIN ZAINAL ABIDIN & WAHJU SEDIONO

NO.	GROUP MEMBERS	MATRIC NO.
1.	AHMAD DARWISH BIN AHMAD TERMIZI	2212089
2.	SYABAB ALHAQQI BIN JAAFAR	2211117
3.	WAN NAIMULLAH BIN MOHD AZRUDDIN	2214837
4.	ZAIMUDIN ZAKWAN BIN NORZAMRI	2217513
5.	ZAMIR MUTTAQIN BIN MUHAMMAD BALYAN	2212985

Date of Experiment: Wednesday, 23 October 2024

Date of Submission: Wednesday, 30 October 2024

ABSTRACT

This experiment aims to demonstrate the principles of microcontroller-to-computer communication by interfacing a potentiometer with an Arduino and establishing real-time serial communication with a Python script. The primary goal is to monitor the analog sensor data from the potentiometer, transmitted through USB, and to visualise it graphically using Python. By achieving this, the experiment provides an understanding of basic sensor interfacing, serial communication, and data visualisation, which are essential skills in embedded systems. This setup supports applications that require data analysis, control, and monitoring, serving as an introductory exploration into mechatronic system integration.

TABLE OF CONTENTS

NO	CONTENT	PAGE
1	INTRODUCTION	4
2	MATERIALS AND EQUIPMENTS	5
3	EXPERIMENTAL SETUP	5
4	METHODOLOGY	6-8
5	DATA COLLECTION	8-9
6	DATA ANALYSIS	9
7	RESULTS	10
8	DISCUSSION	10
9	CONCLUSION	10
10	RECOMMENDATIONS	11
11	REFERENCES	11
12	ACKNOWLEDGEMENT	11
13	DECLARATION	12

INTRODUCTION

In mechatronics, the ability to effectively communicate between microcontrollers and computers is fundamental to the design of interactive and data-driven systems. This experiment aims to interface an Arduino with a potentiometer and utilise serial communication to transmit real-time data from the potentiometer to a computer for visualisation. Key goals include establishing reliable serial communication between the Arduino and Python, reading sensor data through an analog input, and graphically displaying this data using Python's libraries. This hands-on approach provides practical exposure to concepts critical in automation and control applications, enhancing understanding of sensor integration, data transmission protocols, and real-time data analysis.

MATERIALS AND EQUIPMENTS

1. **7** SEGMENT DISPLAY
2. BREADBOARD
3. ARDUINO MEGA 2560
4. PUSHBUTTON
5. MALE TO MALE JUMPER WIRE
6. RESISTOR

EXPERIMENTAL SETUP

1. Connect the potentiometer to the Arduino with one leg to 5V, the other to GND, and the middle (wiper) to the analog input pin A0.
2. Connect the Arduino to the computer via USB, as shown in Figure 1, and run the Python script to begin reading potentiometer values in the terminal.
3. Adjust the potentiometer knob to observe the values displayed in real-time.
4. To view real-time data on the Arduino Serial Plotter, close the Python script.
5. Open the Serial Plotter from "Tools" and "Serial Plotter" in the Arduino IDE.
6. Ensure the correct COM port and baud rate (9600) are selected.
7. Observe and customise the data graph as the potentiometer is turned.

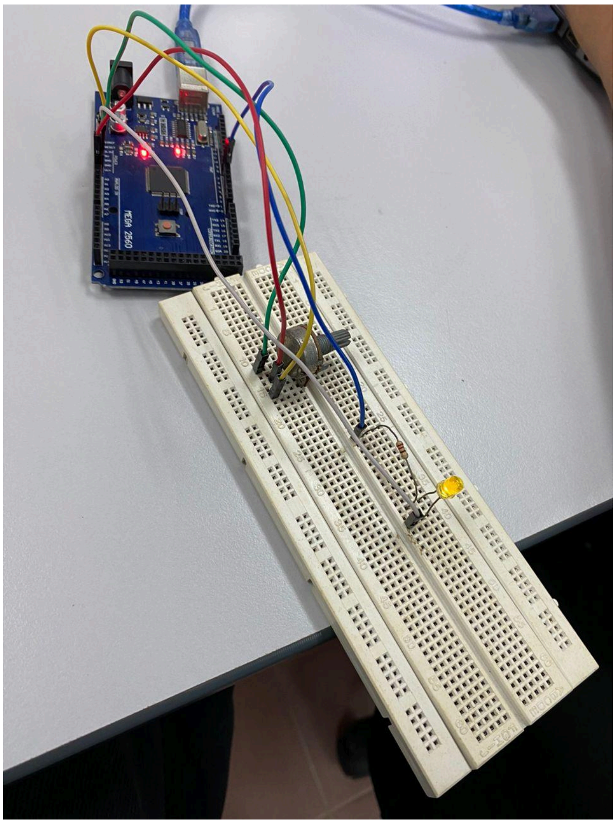

Figure 1: Hardware setup

METHODOLOGY

1. Setup the Arduino Mega 2560
2. IDE code implementation
3. Testing the terminal in python
4. Use the serial plotter in IDE
5. Present the plot in python
6. Code snippet

Arduino code

```
void setup() {
  Serial.begin(9600);
}

void loop() {
  int potValue = analogRead(A0);
  Serial.println(potValue);
  delay(100);
}
```

Python code

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

# Serial port configuration for Arduino Mega
arduino_port = 'COM7'
baud_rate = 9600

# Establish serial communication
ser = serial.Serial(arduino_port, baud_rate, timeout=1)
time.sleep(1)  # Wait for the serial connection to initialize

# Function to read potentiometer value from Arduino
def read_potentiometer():
    if ser.in_waiting > 0:
        data = ser.readline().decode('utf-8').strip()  # Read line from serial, decode, and strip
whitespace
        try:
            return int(data)  # Convert reading to integer
        except ValueError:
```

```
        return None  # In case of an invalid reading
    return None

# Initialize plot
plt.ion()  # Enable interactive mode
fig, ax = plt.subplots()
readings = []
num_readings = 50  # Number of readings to display on the plot
line, = ax.plot(readings, label="Potentiometer Reading")
ax.set_ylim(0, 1023)  # Set y-axis range for Arduino analog values (0 to 1023)
ax.set_xlabel("Time (s)")
ax.set_ylabel("Potentiometer Reading")
plt.title("Real-time Potentiometer Readings from Arduino")
plt.legend()

# Function to update the plot
def update_plot():
    line.set_ydata(readings)
    line.set_xdata(range(len(readings)))
    ax.relim()  # Recalculate limits
    ax.autoscale_view()  # Rescale to fit new data
    plt.draw()
    plt.pause(0.1)

# Main loop for reading data and updating plot
try:
    for i in range(num_readings * 2):  # Adjust this for desired duration
        reading = read_potentiometer()

        if reading is not None:
            readings.append(reading)

            # Keep the latest readings only
            if len(readings) > num_readings:
                readings.pop(0)

            # Update the plot
            update_plot()

            time.sleep(0.1)# Match Arduino's delay

except KeyboardInterrupt:
    print("Data collection stopped by user.")
```

```
finally:
    ser.close()  # Close serial connection
    plt.ioff()   # Turn off interactive mode
    plt.show()
```

DATA COLLECTION

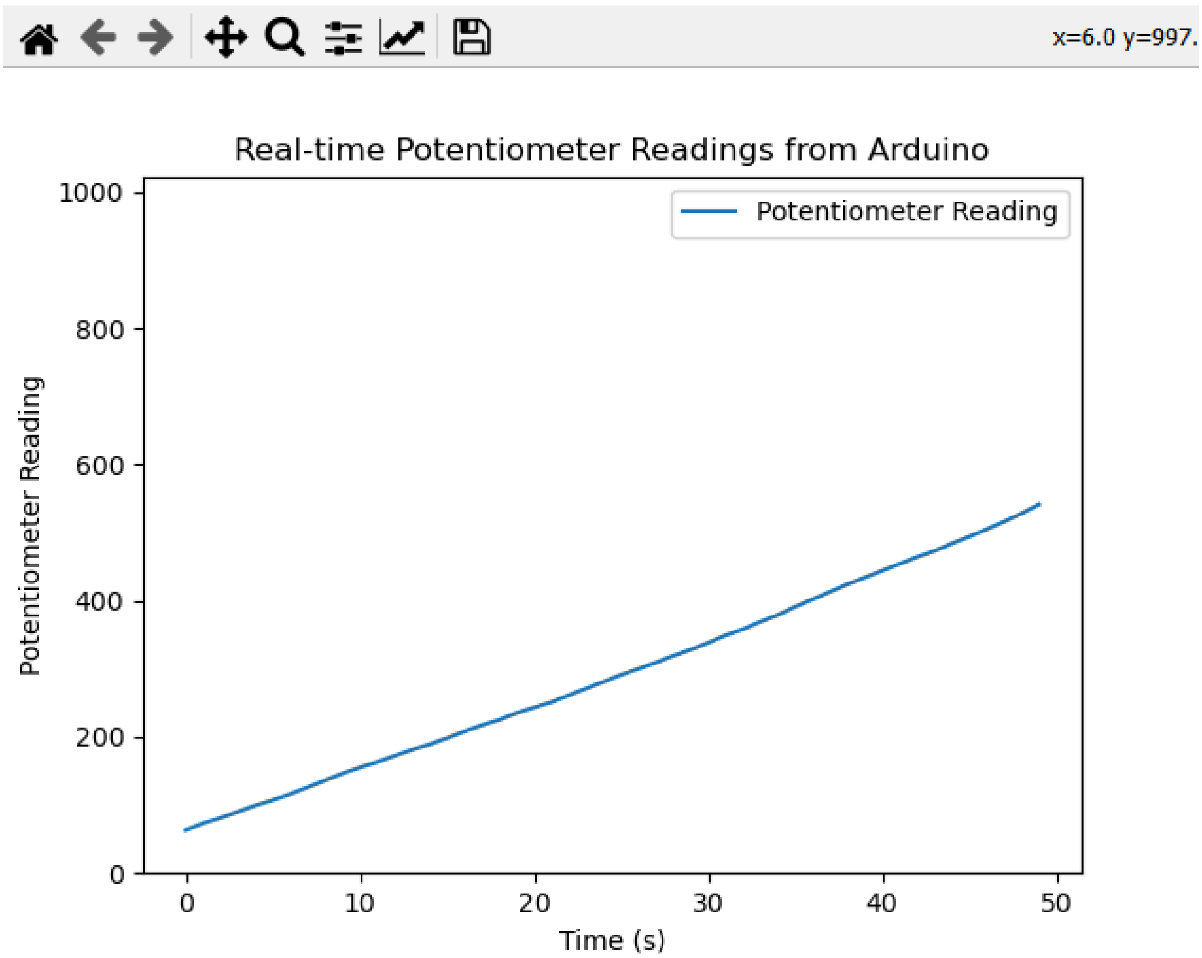

Figure 2

Time	Potentiometer Reading
0	79
10	187
20	230
30	368
40	479
50	592

Table 1

DATA ANALYSIS

This data represents the value of the potentiometer in real time. In figure 2 above, the value of the potentiometer increases as we rotate the potentiometer from 0 to 592 in the interval of 50 seconds. This is due to the change in resistance inside the potentiometer. The data collected is shown in table 1. The actual max value of the potentiometer is 1023. This shows that the potentiometer can be used in various devices such as changing the intensity of an LED or the volume of a buzzer. In a nutshell, the potentiometer is a useful component for a versatile change in output.

RESULT

The experiment successfully achieved its goal of capturing and displaying real-time potentiometer readings in the Python terminal. As the potentiometer was adjusted, the data was accurately transmitted via USB from the Arduino to the computer.

The Arduino Serial Plotter successfully displayed real-time potentiometer data as a graph, showing clear changes with each knob adjustment. The COM port and baud rate setup ensured synchronised communication, and plot customization allowed for a more detailed view of data changes.

DISCUSSION

The experiment successfully demonstrated that turning the potentiometer should produce analog values between 0 and 1023, where 0 represents 0V (ground) and 1023 represents approximately 5V. As the knob is rotated, the readings should vary smoothly, reflecting the voltage changes at the wiper pin.

Some potential sources of errors can to include:

- Electrical Noise: Analog signals can be prone to noise, which may cause minor fluctuations in readings even when the potentiometer is held steady. Ensuring stable connections and keeping wires short can help reduce this.

- Human Factors: If the potentiometer is rotated too quickly, the microcontroller may not capture all intermediate values accurately, causing the plot to appear "stepped" or less smooth.

CONCLUSION

The potentiometer readings changed smoothly from 0 to 1023 as expected, showing a direct relationship between knob position and output voltage. Serial communication successfully transferred data to Python for visualisation.The results aligned with the expected hypothesis, confirming that the code effectively gives the intended output.

RECOMMENDATIONS

We recommend to include some troubleshooting guidance for common issues such as: make sure the arduino serial monitor is closed when running the python and restarting the serial connection if data is not appearing as expected.

REFERENCES

Serial communication between python and arduino : https://projecthub.arduino.cc/ansh2919/serial-communication-between-python-and-arduino-663756

ACKNOWLEDGEMENTS

Special thanks to ZULKIFLI BIN ZAINAL ABIDIN & WAHJU SEDIONO for their guidance and support during this experiment.

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: darwish Read [/]
Name: AHMAD DARWISH BIN AHMAD TERMIZI Understand [/]
Matric Number: 2212089 Agree [/]
Contribution: Abstract, Introduction, Experimental Setup, Result

Signature: syabab Read [/]
Name: SYABAB ALHAQQI BIN JAAFAR Understand [/]
Matric Number: 2211117 Agree [/]
Contribution: Methodology, Data Collection, Data Analysis

Signature: naim Read [/]
Name: WAN NAIMULLAH BIN MOHD AZRUDDIN Understand [/]
Matric Number: 2214837 Agree [/]
Contribution: Discussion, Conclusion

Signature: zaim Read [/]
Name: ZAIMUDIN ZAKWAN BIN NORZAMRI Understand [/]
Matric Number: 2217513 Agree [/]
Contribution: Results

Signature: zamir Read [/]
Name: ZAMIR MUTTAQIN BIN MUHAMMAD BALYAN Understand [/]
Matric Number: 2212985 Agree [/]
Contribution: Recommendation, Reference