KULLIYAH OF ENGINEERING

MCTA 3203

MECHATRONICS SYSTEM INTEGRATION

SEMESTER 1 2025/2026

SECTION 1

GROUP 10

SERIAL COMMUNICATION TASK 1 LAB REPORT

NAME	MATRIC NO.
Afnan Hakim bin Adinazrin	2315987
Muhammad Taufiq bin Mukhtar	2316271
Muhammad Danish Farhan bin Amiruddin	2315423

INSTRUCTED BY:

DR ZULKIFLI BINZAINAL ABIDIN

TABLE OF CONTENTS

ABSTRACT

This experiment demonstrates the use of serial communication between an Arduino microcontroller and a Python program to transmit and display potentiometer readings in real time. The system converts analog voltage signals from a potentiometer into digital data, which are then sent through the Arduino's serial interface to Python for monitoring and analysis. An LED was incorporated as a visual indicator to show when the potentiometer reading exceeded half of its range. The experiment successfully verified the functionality of real-time data transmission, data visualization, and basic control using Python. This serves as a foundation for developing more complex data acquisition and control systems in mechatronics applications.

INTRODUCTION

Serial communication is one of the most fundamental methods for transferring data between microcontrollers and computers. It allows information such as sensor readings to be transmitted efficiently for real-time monitoring and control. In this task, the focus is on transmitting analog data from a **potentiometer** connected to an **Arduino** to a **Python script** running on a computer.

The potentiometer acts as a variable resistor that provides an analog voltage corresponding to its position. The Arduino converts this analog input into a digital value ranging from **0 to 1023** using its **10-bit analog-to-digital converter (ADC)**. These digital values are then sent via **USB serial communication** at a baud rate of **9600 bits per second (bps)** to the Python program, which displays them on the screen.

An **LED indicator** was added to provide a simple feedback mechanism—turning ON when the potentiometer's value exceeded half its range, and OFF otherwise. This combination of sensor input, microcontroller processing, and Python visualization demonstrates a complete **data acquisition and monitoring system**, which is a core concept in mechatronics and automation engineering.

MATERIALS AND EQUIPMENT

1. Arduino board
2. Potentiometer
3. LED
4. 220 Ω resistor
5. Jumper wires

6. Breadboard

7. USB cable

EXPERIMENTAL SETUP

This setup allows real-time acquisition of potentiometer data from an Arduino board and display through Python running in PyCharm.
Hardware used:

- Arduino UNO board (connected via USB)
- Potentiometer
- LED + 220 Ω resistor
- Breadboard and jumper wires

Connections:

- Potentiometer:
    - One outer pin → 5 V
    - Other outer pin → GND
    - Middle pin → A0
- LED:
    - Anode → pin 13
    - Cathode → 220 Ω resistor → GND

The Arduino continuously sends analog readings (0–1023) to Python via the serial port. Python displays the readings and controls the LED logic based on the value received.

METHODOLOGY **Coding in python (Pycharm)**

```
importmatplotlib.pyplot as plt


importmatplotlib
import serial
matplotlib.use('TkAgg')

ser=serial.Serial('COM3', 9600)
#---Liveplotsetup ---
plt.ion()
fig,ax=plt.subplots()
x_vals,y_vals= [], []

try:
        whileTrue:
        line=ser.readline().decode().strip()
        ifline.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='green', 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)
exceptKeyboardInterrupt:
        print("\nStopped by user.")
finally:
        ser.close()
   plt.ioff()
        plt.show()
        print("Serial connection closed.")
```

DATA COLLECTION

- The Arduino was connected to a potentiometer, LED, and 220 Ω resistor on a breadboard.
- Analog input from the potentiometer (A0) was read and transmitted via USB serial communication at 9600 baud to Python.
- The potentiometer readings were displayed in the Python terminal and visualized in the Arduino Serial Plotter.
- The LED indicator was programmed to turn **ON** when the potentiometer reading exceeded half of its maximum range (~512/1023).
- Observed data (sample):

Trial	Potentiometer Reading	LED Status
1	150	OFF
2	520	ON
3	800	ON
4	320	OFF
5	980	ON

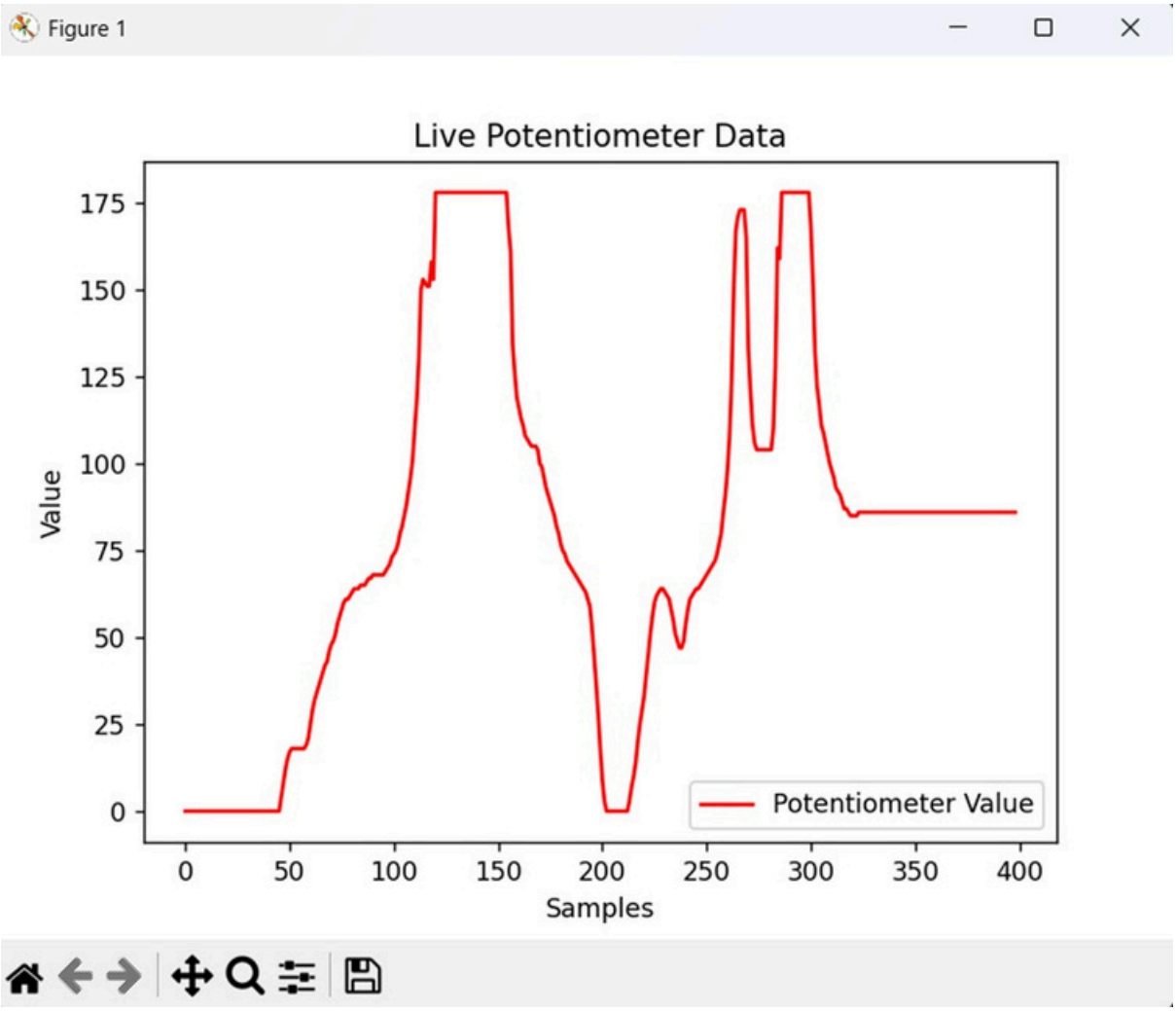
DATA ANALYSIS

- The potentiometer's analog output corresponds to values between 0 and 1023.
- As the knob is turned, the readings printed in PyCharm increase or decrease linearly.
- The Python logic turns the LED indicator message ON when readings exceed 512.
- This demonstrates successful data transmission, threshold-based decision-making, and serial synchronization between the Arduino hardware and Python software.

RESULTS

After setting up the Arduino circuit and running the Python script, the potentiometer readings were successfully transmitted and displayed in real time on the Python terminal. The system responded instantly to changes in the potentiometer's position, confirming proper serial communication between both devices.

The LED correctly responded to the threshold condition — turning ON when the potentiometer value was above half of its maximum (approximately 512) and turning OFF otherwise. Data readings were also verified using the Arduino Serial Plotter, which produced asmooth waveform corresponding to the potentiometer's rotation.



DISCUSSION

In this task, the Arduino was successfully used to read analog signals from a potentiometer and transmit them via serial communication to Python. The readings were displayed in real time, demonstrating the effectiveness of serial communication between hardware and software. When the potentiometer's wiper was rotated, the analog voltage varied between 0 and 5 V, producing corresponding digital readings between 0 and 1023.

The Python script accurately received and displayed the transmitted data, confirming proper synchronization of the baud rate (9600 bps) between both systems. Additionally, the LED indicator, controlled through an if–else condition, provided a simple yet effective feedback mechanism. When the potentiometer value exceeded half of its range, the LED turned ON, representing a threshold-based digital control derived from analog input.

This task effectively demonstrated data acquisition and transmission from Arduino to Python, emphasizing fundamental principles of mechatronic integration — **sensing, signal conversion, and data communication.** Minor variations in readings were observed due to analog-to-digital conversion noise and signal fluctuations, but overall, the response was stable and consistent.

CONCLUSION

The Arduino was able to collect analog data from a potentiometer and transmit it to Python via serial communication. The LED feedback system verified that analog data could be used effectively for threshold-based digital control. This task introduced key concepts in **data acquisition** and **serialdata transmission**, forming the foundation for real-time control applications.

RECOMMENDATIONS

- Add real-time graphing (matplotlib) to visualize sensor behavior.
- Implement data smoothing (moving average) to minimize reading noise.

REFERENCES

Last Minute Engineers. (n.d.). *HowServoMotorWorks&Interface It With Arduino* . Retrieved from https://lastminuteengineers.com/servo-motor-arduino-tutorial/

Arduino Documentation. (n.d.). *Tutorials – From Beginner to Advanced.* Retrieved from https://docs.arduino.cc/tutorials/

Instructables. (n.d.). *Arduino Servo Motors*. Retrieved from https://www.instructables.com/Arduino-Servo-Motors/

APPENDICES

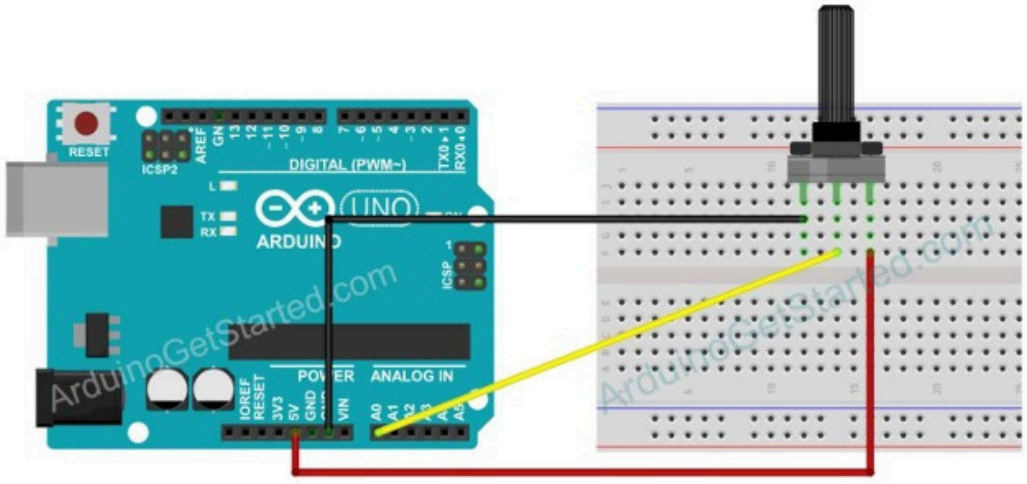


Fig. 1. Wiring Diagram

Python code used:

```
import matplotlib.pyplot as plt
import matplotlib
import serial
import time
matplotlib.use('TkAgg')
```

```
ser = serial.Serial('COM3', 9600)
time.sleep(2)

plt.ion()

times = []
values = []

figure,axes = plt.subplots()
line, = axes.plot(times, values, 'r-')



axes.set_xlabel("Samples")
axes.set_ylabel("Value")
axes.set_title("Live Potentiometer Data")
axes.legend()

sample = 0
try:
    while True:
        pot_value = ser.readline().decode().strip()
        values.append(int(pot_value))
        times.append(sample)
        sample += 1

        line.set_xdata(times)
        line.set_ydata(values)
        axes.relim()
        axes.autoscale_view()
        print("Potentiometer Value:", pot_value)
        plt.pause(0.01)
```

```
        time.sleep(0.5)


except KeyboardInterrupt:
    ser.close()
    print("Serial connection closed.")
    plt.ioff()
    plt.show()
```

ACKNOWLEDGMENTS

We would like to express our deepest gratitude to Sir Zulkifli bin Zainal Abidin , our course instructor for *Mechatronics System Integration (MCTA3203)*, for his invaluable guidance, encouragement, and continuous support throughout the completion of this Digital Logic System lab project.

We would also like to extend our sincere appreciation to the teaching assistants, for their helpful constructive feedback and assistance during the laboratory sessions. Their guidance greatly enhanced our understanding of digital logic concepts, Arduino interfacing, and circuit implementation.

Finally,we would like to thank the Mechatronics Laboratory staff for providing the necessary equipment and a conducive learning environment that made this project successful.

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.

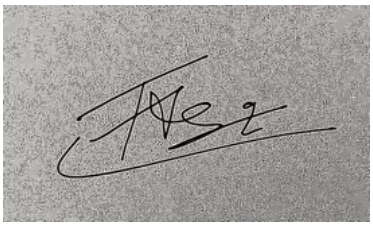
We hereby certify that this report has not been done by only one individual and all of us have contributed to the report. The length of contribution to the reports by each individual is noted within this certificate.

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:
Name: Afnan Hakim bin Adinazrin
Matric Number: 2315987
Contribution: Abstract,Introduction,Materials & Equipment, Results

Read	/
Understand	/
Agree	/



Signature:

Name: Muhammad Taufiq bin Mukhtar

Matric Number: 2316271

Contribution: Experimental Setup,Methodology,

Data Analysis,Recommendations

Read	/
Understand	/
Agree	/

Signature:

Name: Muhammad Danish Farhan bin Amiruddin

Matric Number: 2315423

Contribution: Data Collection,Discussion,Conclusion

Appendices

Read	/
Understand	/
Agree	/