REPORT ON MECHATRONICS SYSTEM INTEGRATION

3A- SERIAL COMMUNICATION
GROUP 7

SECTION 2, SEMESTER 1, 24/25

TEAM MEMBERS

NAME	MATRIC NO.
MUHAMMAD ZAMIR FIKRI BIN MOHD ZAMRI	2212515
MUHD AKMAL HAKIM BIN SAIFUDDIN	2216093
NUR SHADATUL BALQISH BINTI SAHRUNIZAM	2212064
NORHEZRY HAKIMIE BIN NOOR FAHMY	2110061
NUR AMIRA NAZIRA BINTI MOHD NASIR	2110026

DATE OF EXPERIMENT: 23 OCTOBER 2024
DATE OF SUBMISSION: 30 OCTOBER 2024

ABSTRACT

Serial communication between Python and Arduino is a common way to exchange the data between a computer and microcontroller. This report investigates principles and application of serial communication in the system. This experiment was conducted by uploading the arduino code in the microcontroller and running the python script to observe the data collected and its effectiveness. We should be able to transmit the potentiometer reading from Arduino to python script. The findings indicate that serial communication can be a reliable data transmission. Serial communication has a lot of use in modern applications like embedded systems, industrial automation,data communication or telecommunications. As the technology keeps advancing, serial communication will evolve to meet more demands in the future.

TABLE OF CONTENT

INTRODUCTION

Serial communication is fundamental in data transmission. It has a lot of use especially in many modern applications. Embedded systems like arduino is one of the applications that use serial communication. This experiment is to observe the data that exchanges between microcontroller and computer by using Python and Arduino. Establishing the serial communication and reading the potentiometer value in Python that have been sent from Arduino. To establish communication on both sides, Arduino and Python code need to be written. The data will be transferred through USB to the computer. As the potentiometer knobs turn the reading will change according to whether we go higher or lower. The table of reading was recorded as inference. It is expected as we turn up the potentiometer, the higher reading goes as it will supply more voltage. (ranging from 0 to 1023, representing 0 to 5V on most Arduinos. It also can be observed through LED brightness. As the voltage supply increases, the brightness of the LED also increases.

MATERIALS AND EQUIPMENT

- Arduino Uno board
- Potentiometer
- Jumper wires
- LED
- 220 ohm resistor
- Breadboard

EXPERIMENTAL SETUP

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.

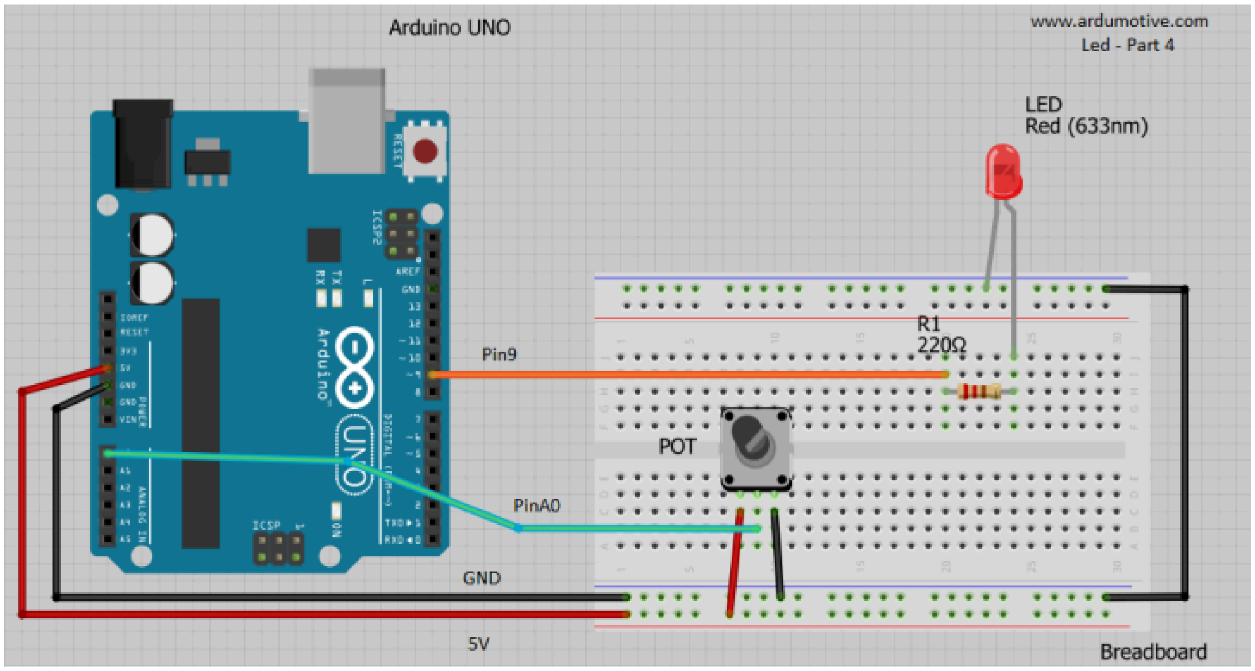


Figure 1.

METHODOLOGY

1. Connect the Arduino to your computer via a USB cable.
2. Power on the Arduino (upload the sketch to your Arduino using the Arduino IDE)
3. Run the Python script on your computer.
4. As you turn the potentiometer knob, you should see the potentiometer readings displayed in the Python terminal.
5. You can use these readings for various experiments, data logging, or control applications, depending on your project requirements.

Using the Arduino Serial Plotter (Please note that if you are working with Python to read data from your Arduino, it's important not to open the Arduino Serial Plotter simultaneously. The Arduino Serial Plotter is a dedicated tool for visualizing data received from the Arduino board and may interfere with Python's access to the serial port. If you intend to use Python to read and process data from the Arduino, ensure that the Serial Plotter is closed or not in use while running your Python script to maintain uninterrupted communication between Python and the Arduino.):

6. Open the Serial Plotter: In the Arduino IDE, go to "Tools" -> "Serial Plotter."
7. Select the Correct COM Port: In the Serial Plotter, select the correct COM port to which your Arduino is connected.
8. Set the Baud Rate: Ensure that the baud rate in the Serial Plotter matches the one set in your Arduino code (e.g., 9600).
9. Read Real-Time Data: As you turn the potentiometer knob, the Serial Plotter will display the potentiometer readings in real-time, creating a graphical representation of the data. You can see how the values change as you adjust the potentiometer.
10. Customize the Plotter: You can customize the Serial Plotter by adjusting settings such as the graph range, labels, and colors.

DATA COLLECTION

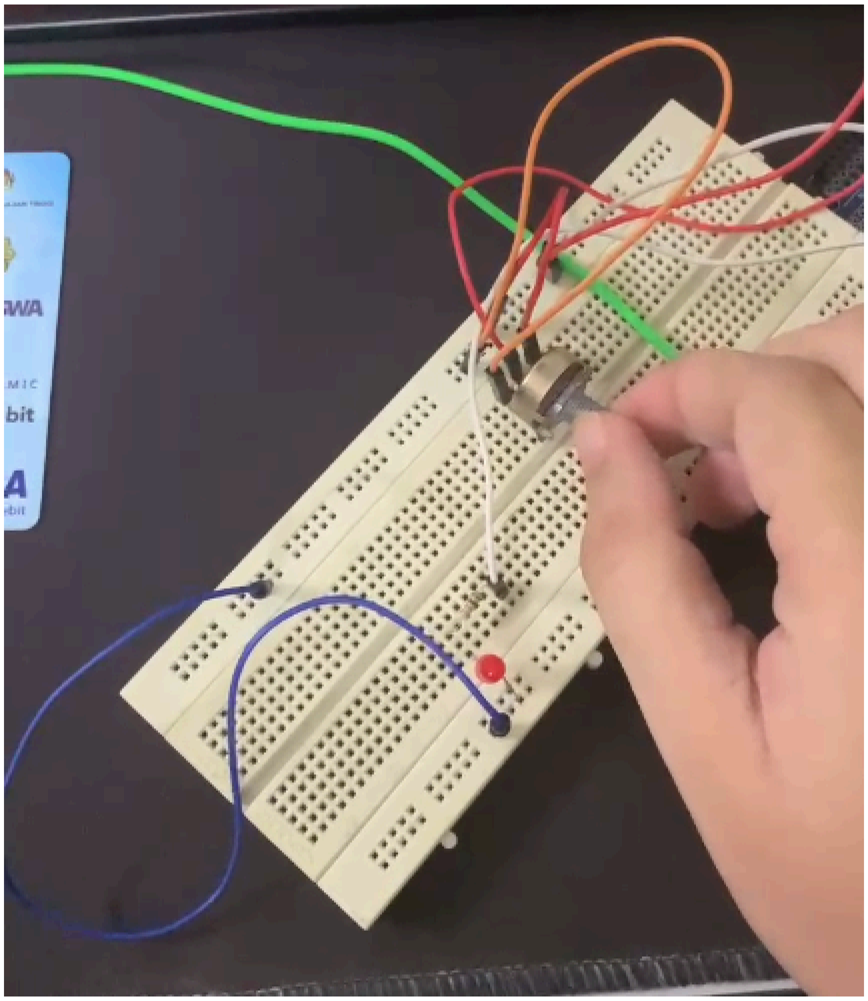
1) Instrument used
    a) Potentiometer
        i) Acts as a variable resistor to provide analog readings (ranging from 0 to 1023, representing 0 to 5V on most Arduinos.
    b) Arduino Board
        i) Reads the potentiometer values and sends them over USB to a computer.
    c) Python Script
        i) Receives serial data from the Arduino, stores it, and processes it for display.
2) Setup
    a) The potentiometer is connected to one of the analog pins on the Arduino
    b) The Arduino is programmed to read the potentiometer value and send this value over serial communication.
    c) Python script is designed to listen on the serial port, receive data, and record each reading for analysis.

3) Data Table

Time (s)	**Potentiometer Reading (Analog Value)**
0	512
1	640
2	300
3	800
4	1023
5	256
6	512

RESULT

The experiment's findings were obtained by building a simple circuit that used a potentiometer to change an LED's brightness. The brightness of the LED is then calculated when the potentiometer reads the value. To determine the brightness, divide the value of the potentiometer by four. After that, this number is sent as an input to the function that regulates LED brightness. The value was displayed on the serial monitor in order to debug the potentiometer. All things considered, the experiment effectively demonstrates how to use a potentiometer to control the LED brightness.



DISCUSSION

1. Interpretation of the Results
    - As you turn the potentiometer, the Arduino reads changes in voltage, converting it into a digital value from 0 to 1023 (corresponding to 0V to 5V). Each change in position produces a change in the output value, which is then sent via the serial connection to the Python script.
    - With the 0.1 second delay in place, the system shows a slower response rate, which means that rapid movements of the potentiometer may not be fully captured. The results will reflect a snapshot each second, offering a good overview but limiting finer, real-time precision.

2. Implication of the Results

- This system could be used for remote monitoring of any device controlled by a potentiometer, such as volume control, brightness adjustments, or other variable analog devices. The setup provides a basis for creating simple IoT monitoring systems, allowing users to observe and log the device’s settings or status remotely.
- The experiment demonstrates principles of data acquisition, serial communication, and analog-to-digital conversion. It’s a foundational setup for students or beginners in electronics and programming, offering insight into how physical inputs are digitized and interpreted.
- Adding real-time graphing could provide immediate visual feedback, making it easier to spot trends or anomalies as they occur.
- If greater precision is needed, a microcontroller with a higher ADC resolution could be used, capturing more nuanced changes in the potentiometer’s position.
- Implementing software-based data filtering techniques in Python could reduce the effects of noise and provide smoother data output, especially useful in systems where stability is crucial.

3. Discrepancies between expected and observed outcomes.
    - To reduce noise, we add capacitors near the potentiometer connections to smooth out fluctuations. Ensuring secure connections and using quality cables can also help stabilize readings.
    - If discrepancies remain unaddressed, it may lead to incorrect conclusions about the potentiometer’s behavior. For example, noise could be mistaken for fluctuations in the potentiometer's position.

4. Sources of error or limitations of the experiment
    - Electrical noise from nearby components, wiring, or even power sources can introduce small fluctuations in the analog readings.
    - Noise can cause inconsistent or fluctuating potentiometer readings even if the dial remains steady, leading to unreliable data.

RECOMMENDATION

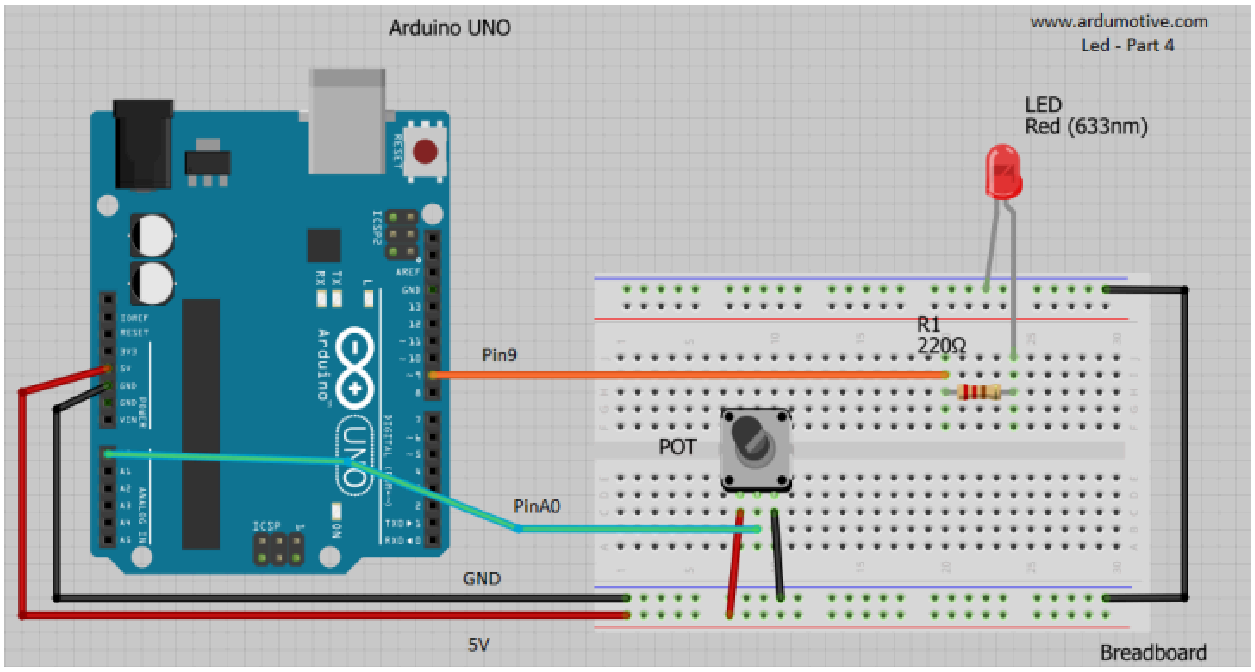
Plenty of improvement suggestions might be made to this Arduino potentiometer experiment to increase its accuracy, effectiveness, and usability. First, implementing data smoothing methods in the Arduino code, such a moving average filter, might help create more steady readings by eliminating electrical noise-induced variations. Another recommended step is calibration, which can handle variances between potentiometer values brought on by manufacturing tolerances. We might obtain more accurate and reliable readings by mapping the analogue input range of the potentiometer to a measured range.

The setup would be more portable and appropriate for separated monitoring if wireless data transfer was implemented using a Bluetooth or Wi-Fi module, this would remove the requirement for a cable connection. Using the Python Matplotlib package, which gives users greater flexibility than the Serial Plotter in terms of adding labels, adjusting axis sizes, and seeing real-time updates, can improve the presentation of information through customized graphs, which may also facilitate data understanding.

Lastly, interactivity would be introduced by automating some experiment controls. An LED indication that illuminates when the potentiometer reaches specific levels, for example, might be set up to provide visual feedback and enable more dynamic applications. All things considered, these improvements would increase the experiment's adaptability, precision, and interest for applications requiring data analysis and control.

APPENDICES

Circuit diagram



Arduino ide coding

```
void setup() {
  Serial.begin(9600);
  pinMode (LED_BUILTIN, OUTPUT);
 }
 void loop() {
   int potValue = analogRead(A0);
   Serial.println(potValue);
   delay(1000); }
   digitalWrite ( LED_BUILTIN, HIGH);
   delay(1000);
```

Python coding

```
import serial
import time

# Replace 'COMX' with your Arduino's serial port (e.g., 'COM3' on Windows or '/dev/ttyUSB0' on
Linux)
```

```
serial_port = 'COM5'
baud_rate = 9600  # Match this with the baud rate of the Arduino

try:
    # Establish the serial connection
    ser = serial.Serial(serial_port, baud_rate)
    time.sleep(2)  # Give some time for the serial connection to establish

    print("Serial connection established. Reading potentiometer values...")
    while True:
        # Read the potentiometer value from the serial port
        pot_value = ser.readline().decode('utf-8').strip()
        print(f"Potentiometer Value: {pot_value}")

except serial.SerialException as e:
    print(f"Error connecting to serial port: {e}")
except KeyboardInterrupt:
    print("Interrupted by user. Closing the serial connection.")
finally:
    # Ensure the serial connection is properly closed
    if 'ser' in locals() and ser.is_open:
        ser.close()
        print("Serial connection closed.")
```

STUDENT 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 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.**

Name : Muhammad Zamir Fikri Bin Mohd Zamri	**Read**	/
Matric No. : 2212515	**Understand**	/
[illegible] **Signatures :**	**Agree**	/

Name : Muhd Akmal Hakim Bin Saifuddin	**Read**	/
Matric No. : 2216093	**Understand**	/
Signatures : akmal	**Agree**	/

Name : Nur Shadatul Balqish Binti Sahrunizam	**Read**	/
Matric No. : 2212064	**Understand**	/
Signatures : shadatul	**Agree**	/

Name :NORHEZRY HAKIMIE BIN NOOR FAHMY	**Read**	/
Matric No. :2110061	**Understand**	/
Signatures :hezry	**Agree**	/

Name : Nur Amira Nazira Binti Mohd Nasir	**Read**	/
Matric No. :2110026	**Understand**	/
Signatures : amira	**Agree**	/