MECHATRONICS SYSTEM INTEGRATION (MCTA 3203)

SECTION 2, SEMESTER 1, 2024/2025

ACTIVITY REPORT

Week :

3a-Serial Communication

Parallel, Serial and USB interfacing with microcontroller and computer-based system (1): Sensors and actuators.

No	Name (Group 6)	Matric No.
1	MUHAMMAD BASIL BIN ABDUL HAKIM	2215315
2	MUHAMMAD SYAFIQ BIN NOR AZMAN	2213187
3	MUHAMMAD HAFIZ HAMZAH BIN FANSURI	2212803
4	MUHAMMAD AMMAR ZUHAIR BIN NOR AZMAN SHAH	2110259
5	MUHAMMAD RAZIQ BIN KAHARUDDIN	2120225

Table of content

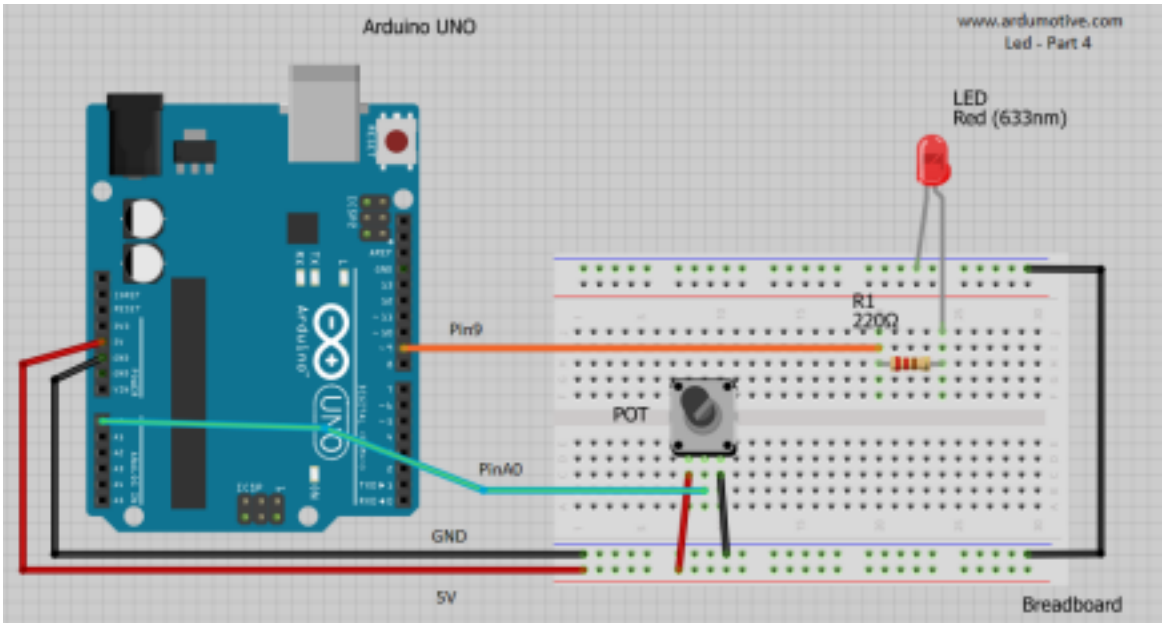
Abstract

This experiment showcased data exchange between computer and microcontroller by using the potentiometer as the controlled variable. The signal reading sent from the Arduino board to the Python script on the computer via a USB connection, allows the user to interpret the data into more readable and understandable information such as in the form of graphs.

Material and equipment

1. Arduino UNO
2. Breadboard
3. LED light
4. Jumper wires
5. Potentiometer

Experimental Setup and Methodology



1. Connected the Arduino board to the computer via a USB cable.
2. Upload the sketch from the Arduino IDE on the computer into the Arduino board.
3. Open the Serial Plotter in the Arduino IDE to monitor the signal of the potentiometer.
4. Closed the Serial Plotter and Arduino IDE before running the Python to avoid any signal interference.
5. Run the Python script on the computer.
6. Monitor the potentiometer readings displayed on the Phyton terminal by turning the knob of the potentiometer on the circuit.

Results

In the experiment, we managed to make a serial connection between Arduino and Python via a USB cable. Once the codes have been uploaded from Arduino IDE into an Arduino board, using Python, the value of the potentiometer is successfully displayed in Python's terminal in real-time as the potentiometer's knob is turned.

The potentiometer is also used to control the current flow in the circuit. Adjusting the knob of the potentiometer, the LED brightness can be dimmed or brightened depending on the value of the potentiometer shown in Python's terminal.

The results are reliable as the value of the potentiometer is the same as in Arduino IDE. This method can be used in other projects such as data logging and control applications. This method is also great for visualising data by adjusting the settings in the serial plotter.

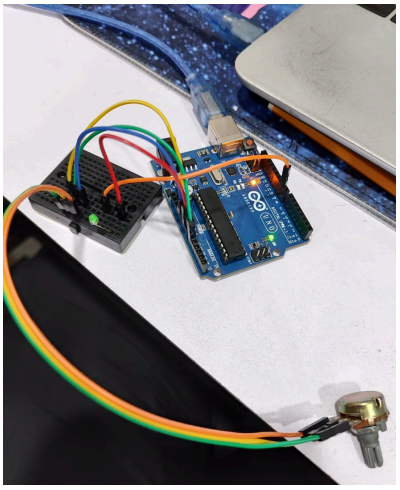


Figure 1: The brightness of LED when the potentiometer is set to low

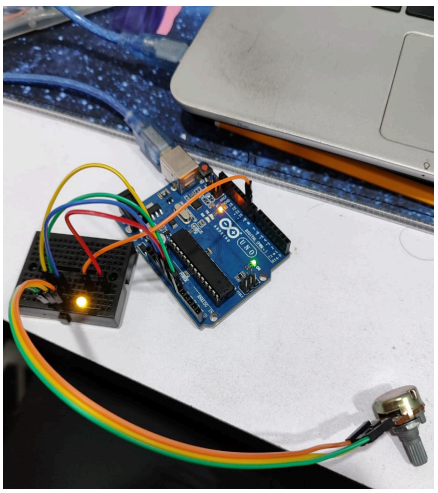
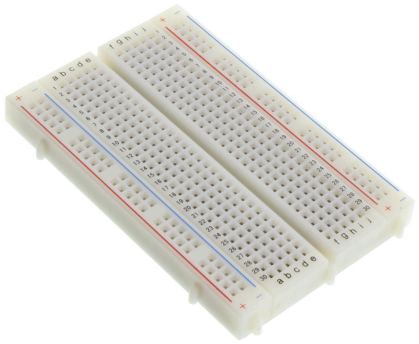


Figure 2: The brightness of the LED when the potentiometer is set to high

Discussions

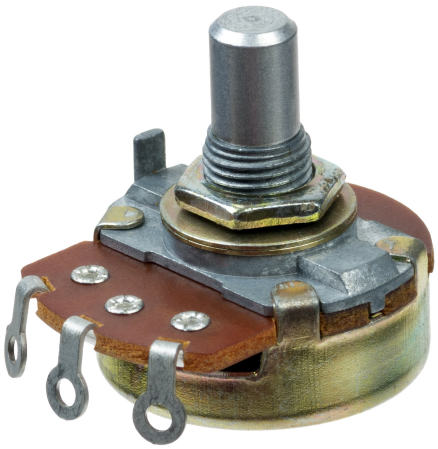
Hardware Discussion

1. **Breadboard**:



A breadboard provides a convenient way to build temporary circuits without soldering. It has interconnected rows and columns, allowing you to insert components and connect them through jumpers.

2. **Potentiometer**:



A potentiometer is a variable resistor. By turning its knob, you can adjust the resistance, which will control the amount of voltage supplied to the LED. In this experiment, it will adjust the brightness of the LED.

3. **LED**

The LED is the output component that lights up when voltage is applied. LEDs have polarity, so you need to connect the longer leg (anode) to the positive side and the shorter leg (cathode) to the ground.

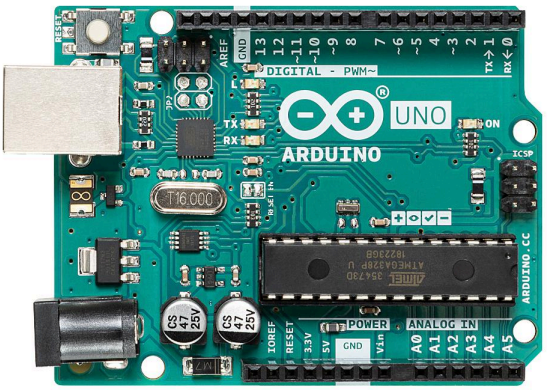
4. **Resistor**:

A resistor limits the amount of current flowing through the LED, preventing it from burning out. For an LED, a 220-ohm resistor is typically sufficient.

5. **Jumper Wires (Male-to-Female)**:

These wires connect components on the breadboard to the Arduino's pins. Male-to-female jumpers are useful for connecting the Arduino's pins to the breadboard.

6. **Arduino**:



The Arduino microcontroller board controls the circuit. It reads the input from the potentiometer and adjusts the voltage to the LED accordingly. You'll use analogue input to read the potentiometer's position and control the LED.

Software Discussion

ARDUINO CODE

```
int potvalue;
int led;
void setup() {
  Serial.begin(9600);
  pinMode(9, OUTPUT);
  pinMode(A0, INPUT);
}
void loop() {
```

```
  potvalue = analogRead(A0);
  led = map(potvalue, 0, 1023, 0, 255); // Corrected line
  Serial.println(potvalue);
  analogWrite(9, led);
  delay(1000);
}
```

A. **Potentiometer Input**:

   The potentiometer acts as a variable resistor. As you turn it, the resistance changes, which the Arduino reads as an input value between 0 and 1023. This value represents the potentiometer's position.

B. **Mapping for LED Brightness**:

C. The Arduino maps the potentiometer's input value (0-1023) to a new range (0-255). This conversion allows the Arduino to match the LED's brightness scale, where 0 is off and 255 is full brightness.

D. **PWM Control of LED Brightness**:

   The mapped value is sent to the LED pin using PWM . It rapidly switches the LED on and off at different levels, adjusting its perceived brightness according to the potentiometer setting.

E. **Serial Monitor Output**:

   The Arduino outputs the potentiometer's value to the Serial Monitor, providing real-time feedback so you can see how the dial position changes the input.

PYTHON CODE

```
import serial
# -- Replace 'COMX' with your Arduino's serial port
ser = serial.Serial('COM4', 9600)
try:
while True:
pot_value = ser.readline().decode().strip()
print("Potentiometer Value:", pot_value)
except KeyboardInterrupt:
ser.close()
print("Serial connection closed.")
```

A. **Setting Up the Serial Connection**:

   It creates a connection to the Arduino by specifying the correct serial port and the baud rate. The baud rate must match the rate used in the Arduino program to ensure proper communication.

B. **Starting a Continuous Loop**:

   The program enters an infinite loop, which allows it to continuously read data from the Arduino until the user decides to stop it.

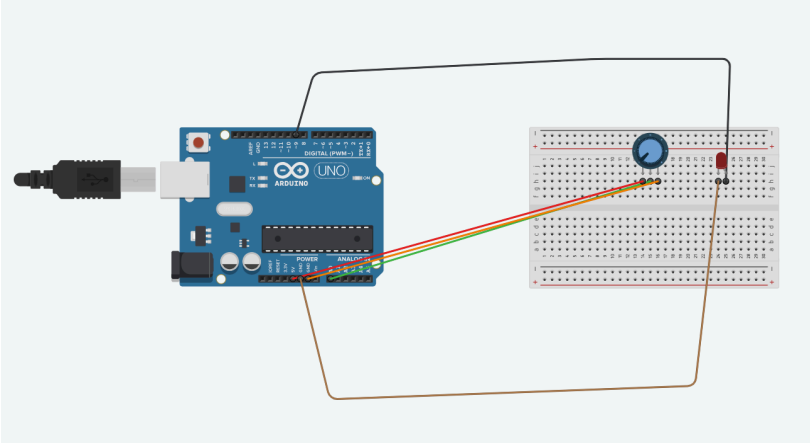
C. **Reading and Decoding the Potentiometer Value**:

   Inside the loop, the program reads a line of data from the Arduino. This data represents the current value of the potentiometer. The program then decodes the data from bytes to a string and removes any extra whitespace or newline characters.

D. **Displaying the Value**:

The program prints the cleaned potentiometer value to the console, allowing you to see how it changes as you adjust the potentiometer

Electrical discussion



In this experiment, an Arduino is configured to control the brightness of an LED using a potentiometer as an input device. The potentiometer is connected to the breadboard with its outer pins linked to the Arduino's 5V and GND, while the middle pin is connected to analogue input A0. The LED, connected to digital pin 9 on the Arduino, has its anode connected to the pin and its cathode connected to the ground through a 220Ω resistor to limit the current and prevent damage. When the potentiometer is adjusted, it alters the voltage sent to the Arduino on pin A0, allowing the Arduino to read the potentiometer value and adjust the brightness of the LED accordingly. This setup enables real-time control of the LED's brightness based on the user's input via the potentiometer.

Question

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).

```
import serial
import matplotlib.pyplot as plt
from matplotlib.animation import FuncAnimation

# Set up the serial port and baud rate
try:
    ser = serial.Serial('COM4', 9600, timeout=1)  # Ensure port matches your setup
except serial.SerialException:
    print("Could not open COM4. Check if the port is correct and available.")

# Initialize lists for data storage
x_data = []
y_data = []

# Set up the plot
fig, ax = plt.subplots()
ax.set_title("Real-Time Potentiometer Readings")
ax.set_xlabel("Time (s)")
ax.set_ylabel("Potentiometer Value")

# Update function for the live plot
def update(frame):
    if ser.in_waiting > 0:  # Check if data is available
```

```
                line = ser.readline().decode('utf-8').strip()  # Read a line from serial
            print("Received:", line)  # Debug: Print raw data from Arduino

        try:
                potvalue = int(line)  # Try converting line to integer
            x_data.append(len(x_data))  # Use the index as a proxy for time
             y_data.append(potvalue)  # Store potentiometer value

            # Clear and update plot with new data
          ax.clear()
                    ax.plot(x_data, y_data, color="green")
                # Set labels and limits
                    ax.set_title("Real-Time Potentiometer Readings")
          ax.set_xlabel("Time (s)")
                ax.set_ylabel("Potentiometer Value")
              ax.set_ylim(0, 900)  # Adjust as per potentiometer range
        except ValueError:
                print("Invalid data received; skipping line.")  # Debug message if data is invalid

# Start animation
ani = FuncAnimation(fig, update, interval=100)  # Update every 100 ms

# Show the plot
plt.show()
```

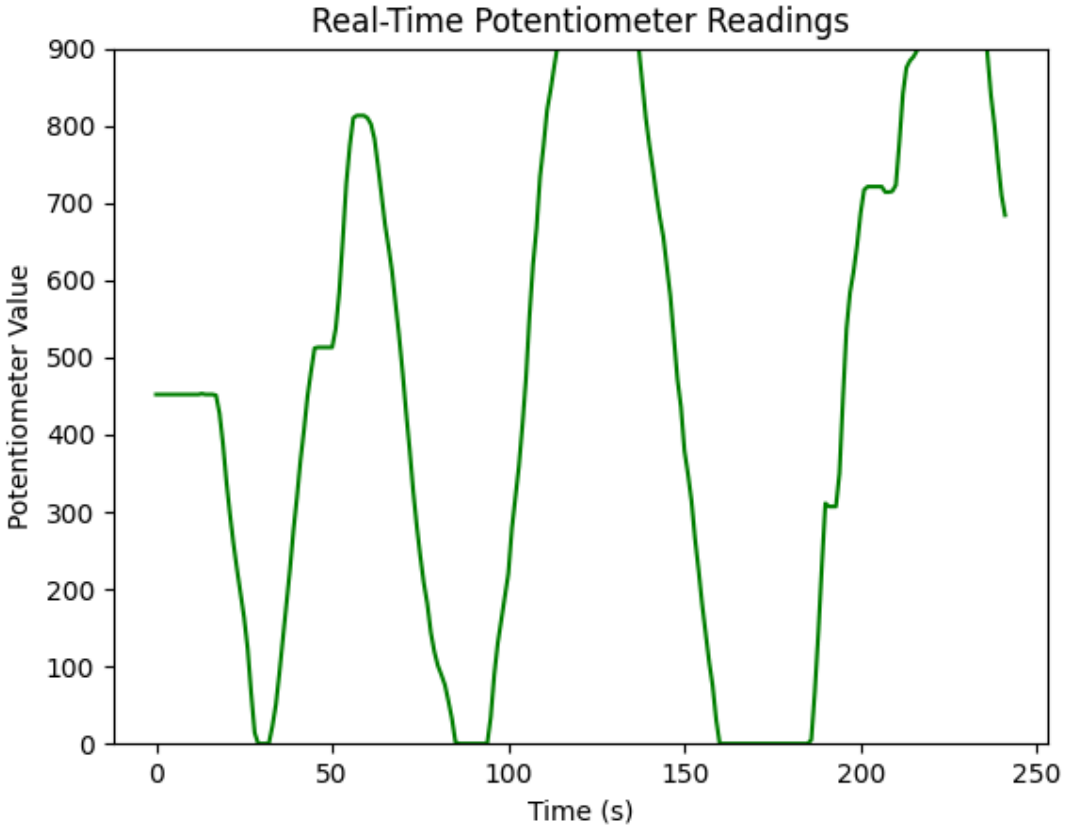


Figure 3: Graph of Real-Time Potentiometer Reading

Recommendation

To improve the experiment, it is recommended to make sure all of the components are working properly to avoid failure. After making sure the components are working, we can use filters such as low-pass filters to reduce noise that might occur during data reading. Another easily forgotten thing is to make sure delay is implemented in codes to make sure data reading does not jumble up and mess up the data shown.

To get more information from the data, we can plot graphs using matplotlib function in Python script to get a better visualization of the value for any project.

Conclusion

The experiment successfully demonstrated the interaction between an Arduino, a potentiometer, and an LED. By connecting the potentiometer to analog input A0 and the LED to digital pin 9, we were able to control the brightness of the LED based on the resistance adjusted by the potentiometer. The Arduino's ability to read the potentiometer's varying voltage and map it to LED brightness highlights the flexibility of microcontroller-based projects in real-time applications. Overall, the experiment reinforced key principles of circuit design and programming while providing a practical application of how simple components can work together to create dynamic and responsive systems.

Student Declaration

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 taken or done by unspecified sources or person.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 have been verified by us.

NAME: Muhammad Basil bin Abdul Hakim	READ	✔
MATRIC NO: 2215315	UNDERSTAND	✔
SIGNATURE: 	AGREE	✔

NAME: Muhammad Syafiq Bin Nor Azman	READ	✔
MATRIC NO: 2213187	UNDERSTAND	✔
SIGNATURE 	AGREE	✔

NAME:Muhammad Hafiz Hamzah Bin Fansuri	READ	✔
MATRIC NO:2212803	UNDERSTAND	✔
SIGNATURE : 	AGREE	✔

NAME: Muhammad Ammar Zuhair Bon Nor Azman Shah	READ	✔
MATRIC NO: 2110259	UNDERSTAND	✔
SIGNATURE: 	AGREE	✔

NAME: MUHAMMAD RAZIQ BIN KAHARUDDIN	READ	✔
MATRIC NO: 2120225	UNDERSTAND	✔
SIGNATURE	AGREE	✔