LABORATORY PROJECT REPORT

PARALLEL, SERIAL AND USB INTERFACING

EXPERIMENT 3A

DATE : 22TH MARCH 2025

SECTION : 1

GROUP : 1

SEMESTER 2, 2024/2025

NO	NAME	MATRIC NO
1.	AKMAL FAUZAN BIN AZHAR	2319531
2.	AIMAN SAIFULLAH BIN AMINNURLLAH	2319185
3.	SYAFIQ HELMIE BIN SAIFULLAH ADHA	2316049

DATE OF SUBMISSION :

Monday, 22TH March 2025

Abstract

This experiment explores the use of serial communication between an Arduino Uno microcontroller and a Python-based computer interface to transmit analog sensor data. The objective is to gain practical experience in interfacing hardware with software using serial protocols. The methodology involves connecting a potentiometer to an Arduino, reading its analog values, and sending the data over a USB serial connection to a Python script for real-time monitoring. The Arduino is programmed to read the analog input and transmit it via the serial port, while the Python script utilizes the pyserial library to receive and display the data. Additional visualization is achieved using the Arduino Serial Plotter and can be extended using Python's matplotlib library. Key findings highlight the importance of synchronized baud rates, proper serial port management, and the usefulness of graphical representation for sensor data interpretation. The results demonstrate the effectiveness of serial communication for simple sensor interfacing and lay the foundation for more advanced data logging and control system applications. This experiment serves as a valuable introduction for mechatronics students to sensor interfacing, serial communication, and real-time data visualization.

Table of Contents

1.0 Introduction

This experiment aims to demonstrate the principles of serial communication by establishing a data exchange link between an Arduino Uno and a computer using a USB interface. The objective is to read analog values from a potentiometer connected to the Arduino and transmit these readings to a Python script running on the computer. Through this experiment, students will gain hands-on experience with microcontroller programming, analog-to-digital conversion, and real-time data communication between hardware and software systems.

Serial communication is a fundamental concept in embedded systems, allowing for the transmission of data one bit at a time over a communication channel. It is widely used in interfacing microcontrollers with external devices, sensors, and computers. In this experiment, the Arduino reads the potentiometer's analog signal and sends the data via its serial port. A Python script using the pyserial library receives and displays the data, providing a platform for real-time monitoring and potential visualization. The experiment reinforces key electronic concepts, including analog signal acquisition, baud rate synchronization, and serial data handling.

The hypothesis for this experiment is that when the circuit is properly assembled and the respective codes are correctly uploaded and executed, the potentiometer readings will be successfully transmitted from the Arduino to the Python terminal, enabling accurate real-time data display and visualization. This outcome will validate the effectiveness of serial communication and the integration of hardware and software in embedded system applications.

2.0 Materials and Equipment

The following materials and components were used in the experiment:

2.1 Electronic Components

- Arduino Uno – Microcontroller board for controlling the display
- Potentiometer – Variable resistor used to generate varying analog voltage levels.
- 220-Ohm Resistors – Current-limiting resistors for each segment of the display
- LED – Optional visual indicator for circuit verification or output indication.
- Jumper Wires – For making connections between components
- Breadboard – For assembling the circuit without soldering

2.2 Equipment and Tools

- Arduino IDE – Software for writing and uploading code to the Arduino Uno
- USB Cable – For connecting the Arduino Uno to a computer
- Power Supply (5V via USB or external source) – To power the circuit
- Python with PySerial Library – Used to establish and manage serial communication with the Arduino and display real-time data.

3.0 Experimental Setup

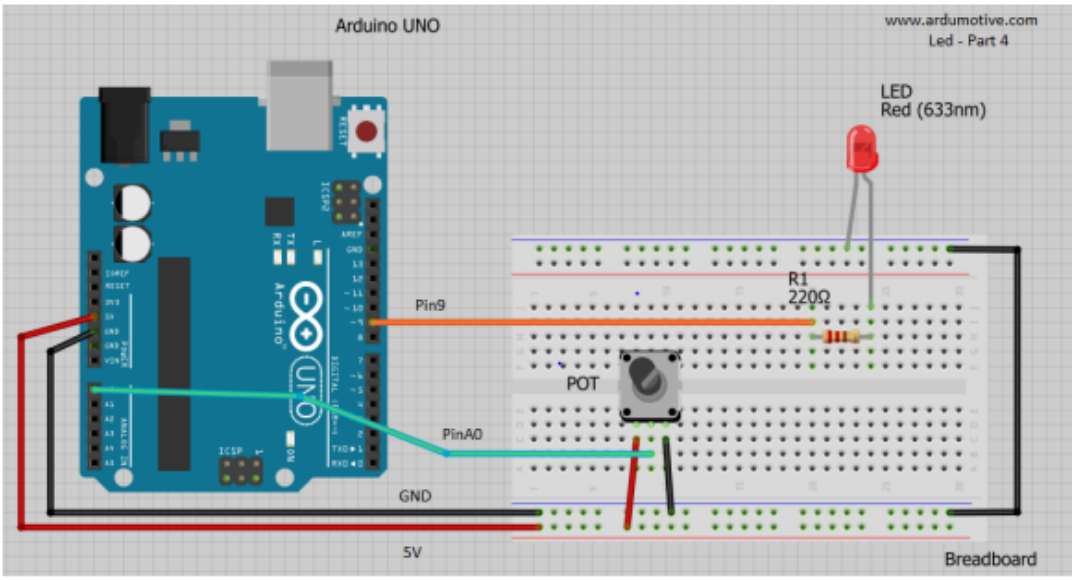


Figure 3.1 Diagram of the Arduino Setup

3.2 Circuit Setup

The potentiometer was connected to the Arduino Uno to allow analog signal input as follows:

1. One outer leg of the potentiometer was connected to the 5V pin on the Arduino to supply voltage.
2. The other outer leg was connected to the GND (ground) pin to complete the circuit.
3. The middle leg (wiper) of the potentiometer was connected to analog input pin A0 on the Arduino to read varying voltage levels based on the potentiometer's position.
4. An LED and 220-ohm resistor may be added as an optional output indicator connected to a digital pin if visual feedback is needed.
5. All components were connected using jumper wires on a breadboard for a secure and solderless setup.

4.0 Methodology

4.1 Implementation and Testing

1. The circuit was built using a breadboard, potentiometer, Arduino Uno, jumper wires, and optional LED with a 220-ohm resistor.
2. The potentiometer was connected with:
   -One end to 5V on the Arduino
   -The other end to GND
   -The middle (wiper) pin to analog input A0
3. The Arduino code was uploaded via the Arduino IDE to enable analog readings and serial communication.
4. A Python script using the pyserial library was run to receive and display the potentiometer readings on the computer terminal.
5. The potentiometer was rotated to test changes in analog values.
6. The Serial Monitor and Serial Plotter (in separate tests) were used to visualize data coming from the Arduino.
7. Readings varied smoothly in response to knob rotation, confirming successful communication.

4.2 Control Algorithm

The Arduino Uno was programmed using the Arduino IDE to read analog values from the potentiometer and send them via serial communication. The Python script handled the reception and display of this data. The control algorithm on the Arduino followed these steps:

1. Initialize the serial communication at a baud rate of 9600. Continuously read the analog input value from pin A0.
2. Transmit the potentiometer reading over the serial port using Serial.println().
3. Introduce a short delay to regulate data transmission.

The Python script followed these steps:

1. Establish a serial connection using the pyserial library.
2. Continuously read incoming data from the serial port.
3. Decode and display the potentiometer readings in the terminal.
4. Optionally, close the serial connection safely upon program termination.

7.0 Results

The experiment successfully demonstrated real-time serial communication between the Arduino Uno and a Python script through a USB interface. As the potentiometer was rotated, the Arduino accurately read the analog voltage values from the wiper pin and transmitted them over the serial port. These readings were successfully received and displayed in the Python terminal, showing a consistent and responsive change in values that reflected the potentiometer's position.

The data transmission was smooth and uninterrupted, with no noticeable lag or communication errors. When tested using the Arduino Serial Plotter, the potentiometer readings were visually represented as a continuous waveform that changed in real-time, further validating the accuracy of the analog-to-digital conversion and serial output. The baud rate of 9600 matched on both Arduino and Python sides ensured proper synchronization, while the delay ( ) function in the Arduino code effectively managed the transmission rate to prevent buffer overflow.

Throughout the experiment, the system remained stable, and the USB-powered Arduino provided a reliable power supply to the circuit. No interference or port conflicts occurred, provided that the Python script and Arduino Serial Plotter were not run simultaneously. These results confirm the successful implementation of the control algorithm and circuit setup, highlighting the practical application of serial communication for sensor monitoring and data visualization in embedded systems.

8.0 Discussion

The experiment successfully demonstrated serial communication between a microcontroller (Arduino) and a computer using Python. Potentiometer readings were accurately transmitted over the USB interface and displayed in the Python terminal, validating the proper setup and functionality of the serial connection. This confirms that the Arduino code and Python script were correctly implemented, with consistent and expected data exchange at the specified baud rate.

The potentiometer responded smoothly to manual adjustments, with real-time changes in resistance clearly reflected in the serial output. This verified the correct analog-to-digital conversion by the Arduino and the integrity of the wiring. The Python terminal consistently received and displayed readable data strings, showing that the pyserial library was effectively used for continuous data acquisition.

One potential source of error observed during the experiment was data lag or occasional missing readings. This may have resulted from an unstable serial connection or a high-frequency data transmission rate overwhelming the buffer. The use of a delay(1000) in the Arduino code helped mitigate this issue by pacing the data output. In future setups, further error handling could be implemented, such as adding checks for malformed data or timeouts in the Python script.

Another possible issue could arise from worn-out potentiometers or loose jumper wires, leading to inconsistent readings. Using newer components and ensuring secure connections can minimize these hardware-related errors.

Overall, the experiment effectively showcased the principles of serial communication and data interfacing between hardware and software systems. The successful execution reinforces the importance of synchronization in baud rates, clean wiring, and robust coding practices in mechatronic system integration. The findings also highlight the potential for extending this setup to more advanced applications, such as real-time plotting or control systems using potentiometer input.

9.0 Conclusion

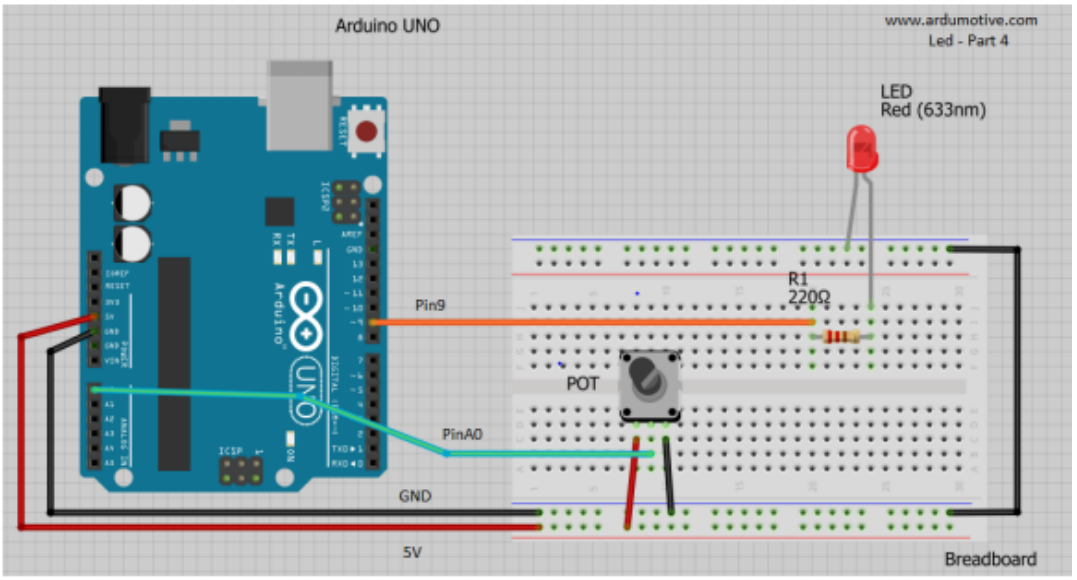
In this experiment, we successfully demonstrated serial communication between an Arduino and a computer using Python. The potentiometer readings were accurately transmitted and displayed in real-time, confirming proper wiring and correct implementation of both Arduino and Python code. This reinforced the fundamental concepts of sensor interfacing and serial data exchange. The experiment serves as a solid introduction to embedded system communication, with potential future improvements including real-time plotting, better error handling, and integration with additional sensors.

10.0 Recommendations

To improve the system's reliability and performance, functions should be used in the Arduino and Python code to simplify repetitive tasks and improve readability. Serial Monitor debugging can be helpful in identifying communication issues. It's recommended to ensure all hardware components, especially jumper wires and potentiometers, are in good condition to avoid erratic readings. Replacing worn-out parts can significantly enhance data accuracy. For future improvements, real-time data plotting using Python libraries, integrating an OLED or I2C display for better visualization, and exploring wireless communication options like Bluetooth or WiFi could expand the system's capabilities.

11.0 References

Weekly Module - Google Drive. (n.d.). Google Drive. https://drive.google.com/drive/folders/1rq0wLF6mA7jEoNsPWyAbWR9n0X3SQsQz?usp=sharing



12.0 Acknowledgement

A special thank you to Dr. Wahju Sediono and Dr. Zulkifli Bin Zainal Abidin, my teaching assistant, and peers, for their outstanding assistance and support in completing this report. Their advice, input, and expertise have had a significant impact on the quality and comprehension of this work. Their time, patience, and dedication to my academic progress are deeply appreciated.

13.0 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.

Name: Aiman Saifullah bin Aminnurllah	Read	/
Matric Number: 2319185	Understand	/
Signature:	Agree	/

Name: Akmal Fauzan Bin Azhar	Read	/
Matric Number:2319531	Understand	/
Signature:	Agree	/

Name: Syafiq Helmie bin Saifullah Adha	Read	/
Matric Number: 2316049	Understand	/
Signature:	Agree	/