LAB REPORT 2: PARALLEL, SERIAL AND USB INTERFACING (Part 1)

GROUP 1

MCTA 3203

SEMESTER 1 2024/2025

MECHATRONICS SYSTEM INTEGRATION

NO	NAME	MATRIC NO
1.	ABDUL A'LIM BIN MOHD RAJIB	2119687
2.	AHMAD HAZAMI BIN MOHD RAZIP	2211203
3.	ABDUL HADI BIN ZAWAWI	2210739
4.	AIN MAISARA BINTI ABDULLAH	2217856
5.	ADIBAH BINTI MOHD AZILAN	2212670

DATE OF SUBMISSION: 30TH OCTOBER 2024

ABSTRACT

This report showcases the implementation of serial communication, established between Arduino and Python using PySerial library, which is used to facilitate the control and monitoring of electronic components in real-time data exchange. Two experiments were carried out: Part A involved displaying potentiometer readings on the Arduino using Python, and Part B which focused on controlling a servo motor through Python commands. The results obtained in both of these experiments indicate the success of the implementation of serial communication, meeting the primary objectives of real-time data exchange and hardware control. These findings demonstrate the effectiveness of integrating Python with Arduino, providing a bigger framework for future applications in mechatronics.

TABLE OF CONTENTS

1. INTRODUCTION

Mechatronic is known as an interdisciplinary field made up of the merging of mechanical, electronics and programming aspects of engineering. The integration of hardware and software has become increasingly popular and crucial to the world as technology continues to advance. Arduino microcontrollers, which are known for their flexibility and user-friendly interfaces, serve as a key player for the advancement of integration systems.This report aims to construct an effective interaction, focusing on the serial communication between Python and Arduino with the electronic components.

The first experiment, Part A, focused on displaying potentiometer readings from an Arduino board using Python. A potentiometer works as a variable resistor, producing a different range of voltage based on its position. Serial communication link via PySerial library allows the real-time data of potentiometer readings to be transferred from the Arduino and Python interface. This setup enabled us to visualize the produced readings and succeeded in demonstrating the practicality of Python usage for data acquisition and analysis.

In the second experiment, Part B, focus was given in controlling the servo motor by prompting instructions in Python to the Arduino. Servo motor is known to be used as a converter which converts the control signal received by the controller into rotational angular displacement or angular velocity of the output shaft of the motor, providing a precise in position control. The commands received from the Python script are transmitted to adjust the position of the servo motor. From this experiment, Python's capability in controlling the movement of hardware is proven, demonstrating a pathway for autonomous tasks and improving system responsiveness.

PART A : DISPLAYING POTENTIOMETER READINGS FROM ARDUINO USING PYTHON

1. MATERIALS AND EQUIPMENT

- Arduino Uno Board
- Potentiometer
- Jumper Wires
- LED
- 220 resistor
- Breadboard

2. EXPERIMENTAL SETUP

a. Connect one leg of the potentiometer to 5V on the Arduino.

b. Connect the other leg of the potentiometer to GND on the Arduino.

c. 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**.

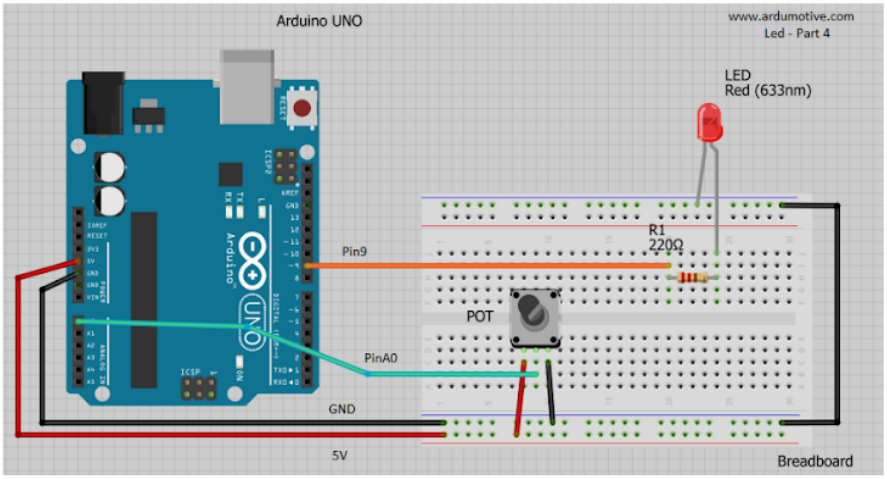


Fig. 1

3. METHODOLOGY

PROCEDURES

a. Connect the Arduino to the computer via a USB cable.

b. Power on the Arduino and upload the sketch to Arduino using the Arduino IDE.

c. Run the Python script on the computer.

d. As the potentiometer knob is turned, the potentiometer readings are displayed in the Python terminal.

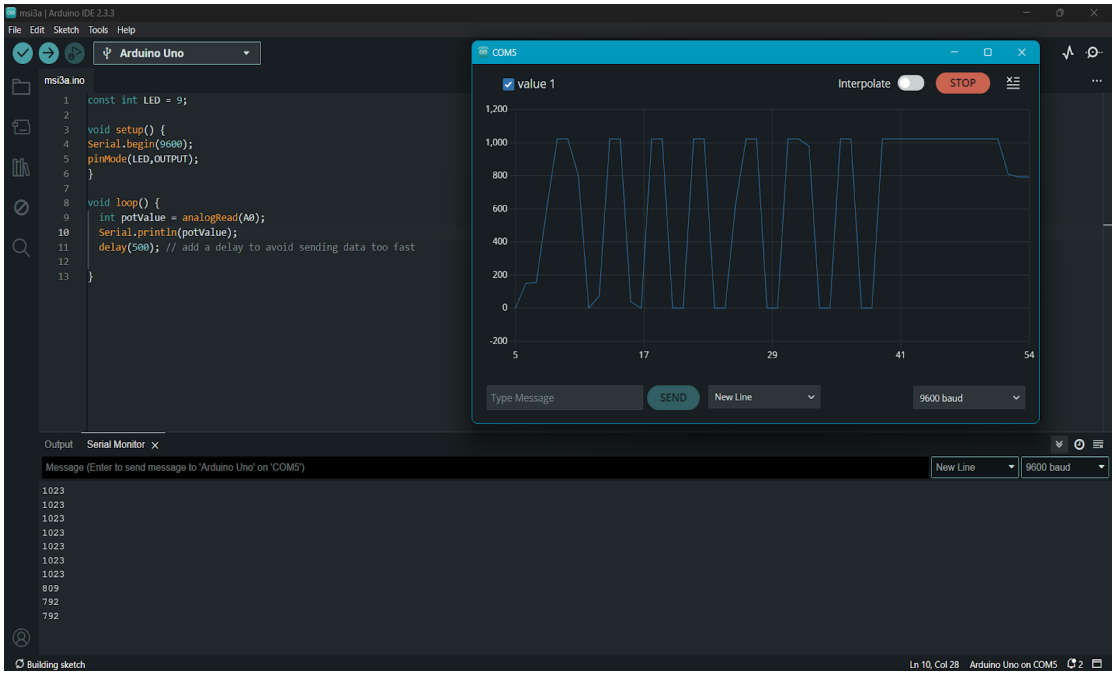
e. In the Arduino IDE, go to "Tools" -> "Serial Plotter."

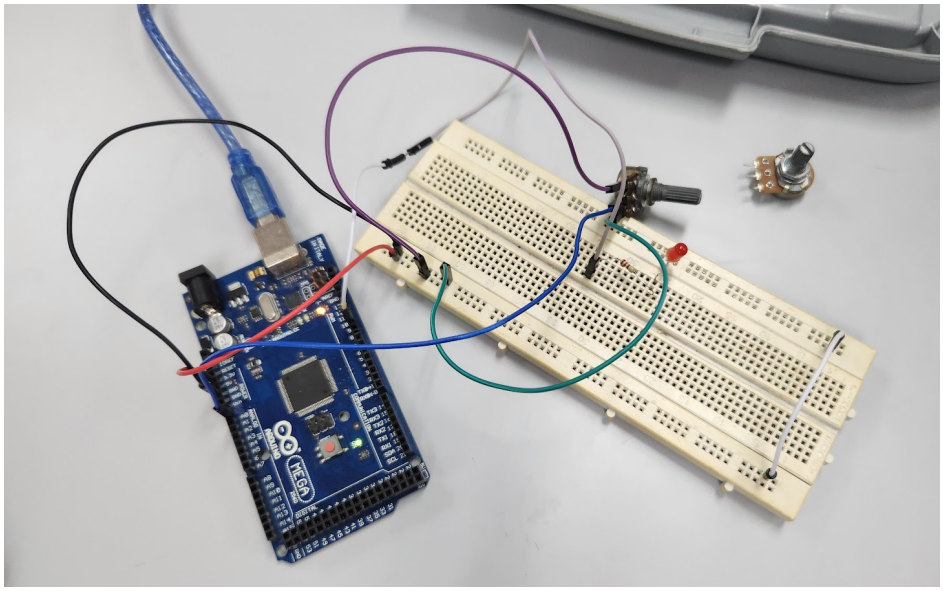
f. Select the correct COM port to which the Arduino is connected.

g. Ensure that the baud rate in the Serial Plotter matches the one set in the Arduino code, 9600 baud.

h. As the potentiometer knob is turned, the Serial Plotter will display the potentiometer readings in real-time, creating a graphical representation of the data. The value is then changed as the potentiometer is adjusted.

4. RESULTS





5. DISCUSSION

By using Python to set up serial connection between an Arduino and a computer, we were able to record and show real-time potentiometer readings in this experiment. We could

read changing voltage values based on the position of the potentiometer by connecting it to the Arduino's analog input. These values were then sent to the computer via a USB connection.

The Arduino sketch was set up to transmit these readings over serial at a baud rate of 9600, and a Python script using the PySerial library read them and showed them on the screen. We improved the visualization of these values with a graph using Matplotlib in Python, which made it easier to comprehend how the resistance of the potentiometer changed over time. This graphical method helps in the interpretation of data trends that could be overlooked when looking at raw statistics alone.

Challenges occurred when many applications attempted to use the serial port at the same time, such as when the Python script and Arduino Serial Plotter were both open. We resolved this by making sure that only one program used the port at a time. With applications in data recording, monitoring, and control systems, this configuration shows how microcontrollers can efficiently transfer sensor data to computers for real-time analysis.

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

Welcome | Release Notes: 1.95.0 | msi3a.py

C: > Users > Abdul Hadi > vs code > msi3a.py > ...

```python
import serial
import matplotlib.pyplot as plt
import matplotlib.animation as animation

# Set up the serial port
ser = serial.Serial('COM7', 9600)  # Update 'COM3' to your port
ser.flush()

# Initialize lists for storing data
data = []

# Create a figure for plotting
fig, ax = plt.subplots()
line, = ax.plot(data)
ax.set_ylim(0, 1023)  # Set the y-axis limit based on the potentiometer range

# Function to update the plot
def update(frame):
    if ser.in_waiting > 0:
        value = int(ser.readline().decode().strip())
        data.append(value)
        data[:] = data[-50:]  # Limit to the last 50 readings
        line.set_ydata(data)
        line.set_xdata(range(len(data)))
        ax.set_xlim(0, len(data))
        print(f"Potentiometer Value: {value}")  # Output to terminal
    return line,

# Set up animation
ani = animation.FuncAnimation(fig, update, blit=True, interval=100)

plt.show()
```

6. CONCLUSION

To conclude in Part A, the implementation of serial communication between the Arduino and Python successfully facilitated the display of potentiometer readings in real time. The PySerial library allowed for efficient data exchange, confirming the robustness of the communication protocol. The readings were accurately captured and transmitted, demonstrating that the system can reliably monitor analog signals. This experiment

highlights the potential for further applications in real-time monitoring systems, showcasing how Python can enhance the capabilities of Arduino in collecting and displaying data from various sensors.

PART B : TRANSMITTING ANGLE DATA FROM PYTHON SCRIPT TO ARDUINO TO ACTUATE A SERVO

1. MATERIALS AND EQUIPMENT

- Arduino Uno Board
- Servo motor
- Jumper wires
- Potentiometer (for manual angle input)
- USB cable for Arduino
- Computer with Arduino IDE and Python installed

2. EXPERIMENTAL SETUP

a. Connect the servo's signal wire to a PWM capable pin on the Arduino (e.g., digital pin 9).

b. Power the servo using the Arduino's 5V and GND pins. Connect the servo's power wire to the 5V output on the Arduino board.

c. Connect the servo's ground wire to one of the ground (GND) pins on the Arduino.

An example of the hardware setup is shown in **Fig. 1**

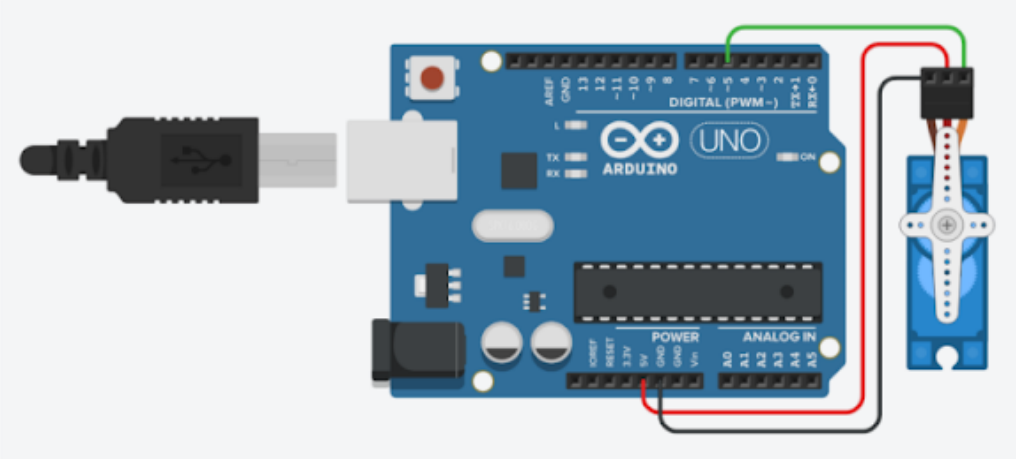


Fig. 1

3. METHODOLOGY

PROCEDURES

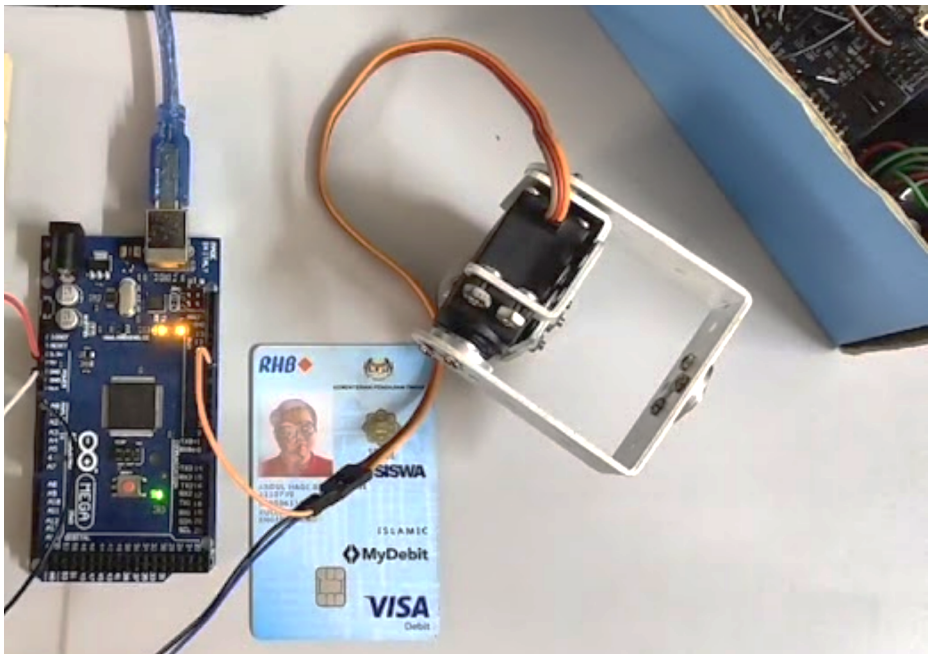
a. Open a new sketch in the Arduino IDE.

b. Write the Arduino code that reads angle data from the serial port and moves the servo accordingly.

c. Upload the code to the Arduino board. (refer

d. Write the Python script and run it.

4. RESULTS



5. DISCUSSION

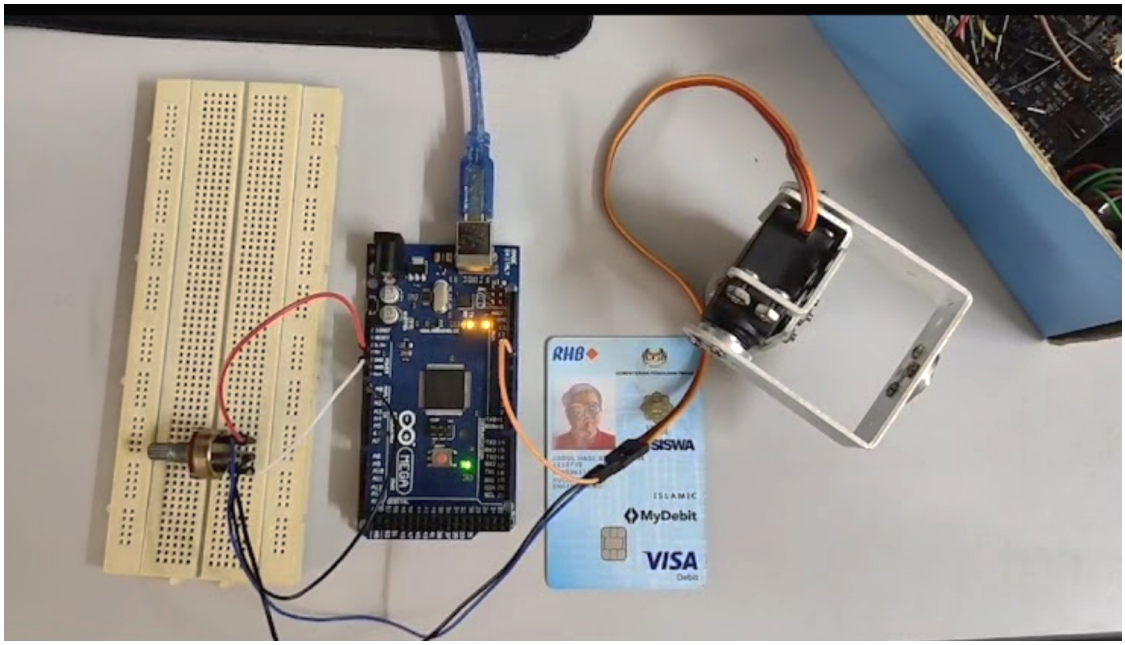
This experiment emphasizes the control of servo motors through the Python commands that are sent to Arduino. The connection setup was similar to the previous experiment in part A, in which the Arduino received commands through the established serial communication, translating the received Python commands into the actions of adjusting the servo's position. As a result, high level precision of the servo motor is displayed as the experiment is carried out.

In addition, this experiment demonstrated a significant impact, particularly in the field of robotics and automation. For applications requiring precise movement control—such as robotic arms and automated positioning systems— the integration of Python simplifies programming and has helped to enhance functionalities. This accessibility not only

benefits developers and engineers but also opens the door for broader use by educators, making advanced robotics more attainable for a wider audience.

Question

Enhance your Arduino and Python code to incorporate a potentiometer for real-time adjustments of the servo motor's angle. Ensure that, in the updated Arduino code, you have the ability to halt its execution by pressing a designated key on your computer's keyboard. Following the modification, restart the Python script to receive and display servo position data from the Arduino over the serial connection. While experimenting with the potentiometer, observe the corresponding changes in the servo motor's position.



msi3b_arduino.ino

```ino
#include <Servo.h>

Servo myServo;
int potPin = A0; // Potentiometer connected to analog pin A0
int servoPin = 9; // Servo connected to digital pin 9
int angle = 0; // Variable to store the servo angle
int potValue = 0; // Variable to store the potentiometer value
int stopCommand = 0; // Variable to store the stop command from Python

void setup() {
  Serial.begin(9600); // Start serial communication
  myServo.attach(servoPin); // Attach the servo to pin 9
}

void loop() {
  // Check for a stop command from Python
  if (Serial.available() > 0) {
    stopCommand = Serial.read();
    if (stopCommand == 's') { // Check for 's' to stop
      return; // Stop the loop execution
    }
  }

  // Read the potentiometer value
  potValue = analogRead(potPin);
  // Map potentiometer value to servo angle range (0-180 degrees)
  angle = map(potValue, 0, 1023, 0, 180);
  // Set the servo to the mapped angle
  myServo.write(angle);

  // Send both the potentiometer value and servo angle to Serial Monitor
  Serial.print("Potentiometer Value: ");
  Serial.print(potValue);
  Serial.print(" | Servo Angle: ");
  Serial.println(angle);

  delay(200); // Small delay for stability
}
```

6. CONCLUSION

To sum up in part B, the control of a servo motor through Python commands illustrated the successful application of serial communication for hardware control. The ability to send precise commands from Python to the Arduino enabled smooth and responsive manipulation of the servo's position. This experiment confirms that the integration of

Python with Arduino is not only feasible but also effective for controlling electronic components. The findings pave the way for more complex automation and robotic applications, reinforcing the significance of combining software and hardware in mechatronic systems.

RECOMMENDATIONS

During the conduct of both of these experiments, several shortcomings were discovered, which include minor latency when the potentiometer is undergoing rapid adjustment, and occasional power supply fluctuations, which affect the servo's performance. Further improvements could be made in both of these experiments, for example, by optimizing the baud rate or applying a more efficient data handling in Python and exploring other alternative motor types for functionality enhancement of the motor..

REFERENCES

1. *Servo Motor - an overview | ScienceDirect Topics*. (n.d.). Www.sciencedirect.com. https://www.sciencedirect.com/topics/engineering/servo-motor

ACKNOWLEDGEMENTS

We would like to express our sincere gratitude to our lecturers, Dr. Wahju Sediono and Dr. Zulkifli bin Zainal Abidin for their invaluable guidance in this experiment. Their expertise in the mechatronics field and encouragement have been a big support in the successful integration of our system. Not forgetting our teaching assistants in the lab for helping us indirectly.

To add, we also wish to extend a special thank you to our fellow team members for their collaboration and hard work during the course of this project. Their contributions were pivotal in driving our efforts forward and enhancing the overall quality of our work.

STUDENTS' 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.**

Signature: **A'LIM**	**Read**	/
Name: ABDUL A'LIM BIN MOHD RAJIB	**Understand**	/
Matric No: 2119687	**Agree**	/

Signature: *hazami*	**Read**	/
Name: AHMAD HAZAMI BIN MOHD RAZIP	**Understand**	/
Matric No: 2211203	**Agree**	/

Signature: 	**Read**	/
Name: ABDUL HADI BIN ZAWAWI	**Understand**	/
Matric No: 2210739	**Agree**	/

Signature: 	**Read**	/
Name: AIN MAISARA BT. ABDULLAH	**Understand**	/
Matric No: 2217856	**Agree**	/

Signature: 	**Read**	/
Name: ADIBAH BINTI MOHD AZILAN	**Understand**	/
Matric No: 2212670	**Agree**	/