MECHATRONICS SYSTEM INTEGRATION

EXPERIMENT 2: PARALLEL, SERIAL AND USB INTERFACING WITH MICROCONTROLLER AND COMPUTER BASED SYSTEM (1):

GROUP NUMBER: A

PROGRAMME: MECHATRONICS

GROUP MEMBERS	MATRIC NO
AISYAH SOFEA BINTI OTHMAN	2115386
SITI ALIAA BINTI IBRAHIM	2112618
MUHAMMAD ADLI FAHMI BIN TAJUL ARIS	2113095
ETHAR ABDALLA ABDELKARIM OSMAN	2111282
NURAIN AINAA AQILAH BINTI ROSLI	2114560

DATE OF EXPERIMENT:

Wednesday, 25th October 2023

DATE OF SUBMISSION:

Wednesday, 1st November 2023

Abstract

There are two parts to this week's experiment which are the first part for the LED and potentiometer while the second part is for the servo motor. For the first part, the objective of this experiment was to make sure we can turn on the LED using potentiometer. The potentiometer reading from Arduino will be sent to the Python to complete the interaction.This laboratory experiment will focus on data interchange between a computer and a microcontroller via serial communication between Python and Arduino. For the second part, the objective was to control the servo motor using Arduino-python interaction. Through a serial connection, a Python script communicates the servo's angle to the Arduino. The servo motor will move to a specified angle after the Arduino is done interpreting the data.

Table of Contents

Objectives **4**
Introduction **4**
Material & Equipment **5**
PART 3A: LED and Potentiometer 5
PART 3B: Servo Motor 5
Experimental setup **5**
PART 3A: LED & Potentiometer 5
PART 3B: Servo Motor 7
Methodology **9**
PART 3A: LED & Potentiometer 9
PART 3B: Servo Motor 12
Data collection **15**
PART 3B: Servo Motor 15
Data analysis **16**
PART 3A: LED & Potentiometer 16
PART 3B: Servo Motor 16
Result **16**
Discussion **17**
PART 3A: LED & Potentiometer 17
PART 3B: Servo Motor 18
PART 3A: LED & Potentiometer 20
PART 3B: Servo Motor 20
Recommendations **21**
Appendices **24**
Acknowledgement **26**

Objectives

1. To gain hands-on experience in setting up and implementing serial communication between an Arduino and a Python script (3A)
2. To send potentiometer readings from the Arduino to the Python script via a USB connection (3A)
3. To demonstrate of how to remotely control a physical device (the servo motor) using Python and Arduino (3B)

Introduction

This experiment is to send the readings from Arduino to the Python script via a USB connection by implementing serial communication between the two. The experiments that apply this implementation have two parts which is the first part, A is to send potentiometer readings whereupon 'Run', the python script should display potentiometer values as we turn its knob. The theory behind it is since the potentiometer is a variable resistor, as its knob is turned, it acts as a voltage divider. The Arduino will read the analog voltage generated by the potentiometer and send these readings to the Python script. The python is also capable of generating and showcasing a graph in the python by using 'matplotlib' library. Having that, it is to observe the readings of the potentiometer and the graph in python upon turning the potentiometer knob and have a smooth and responsive representation of the potentiometer readings.

In part B, the implementation of serial communication is used to control a servo's motor that was connected to the Arduino board where the transmitted angle data from a Python script to an Arduino is to actuates the servo motor to specified angle. We can have the potentiometer for real-time adjustments of the servo motor's angle by enhancing the code of Arduino and Python, where Arduino also has the ability to halt execution by pressing the 's' key from the computer's keyboard. Serial communication relies on the principles of asynchronous data transmission, where data is sent in a sequential manner, making it ideal for real-time applications. It is to observe the real-time adjustment of the servo motor's angle using the potentiometer and the ability to halt the execution using the 's' key will be successfully demonstrated.

Material & Equipment

PART 3A: LED and Potentiometer

- Arduino
- Potentiometer
- Jumper Wires
- LED
- 220Ω resistor
- Breadboard

PART 3B: Servo Motor

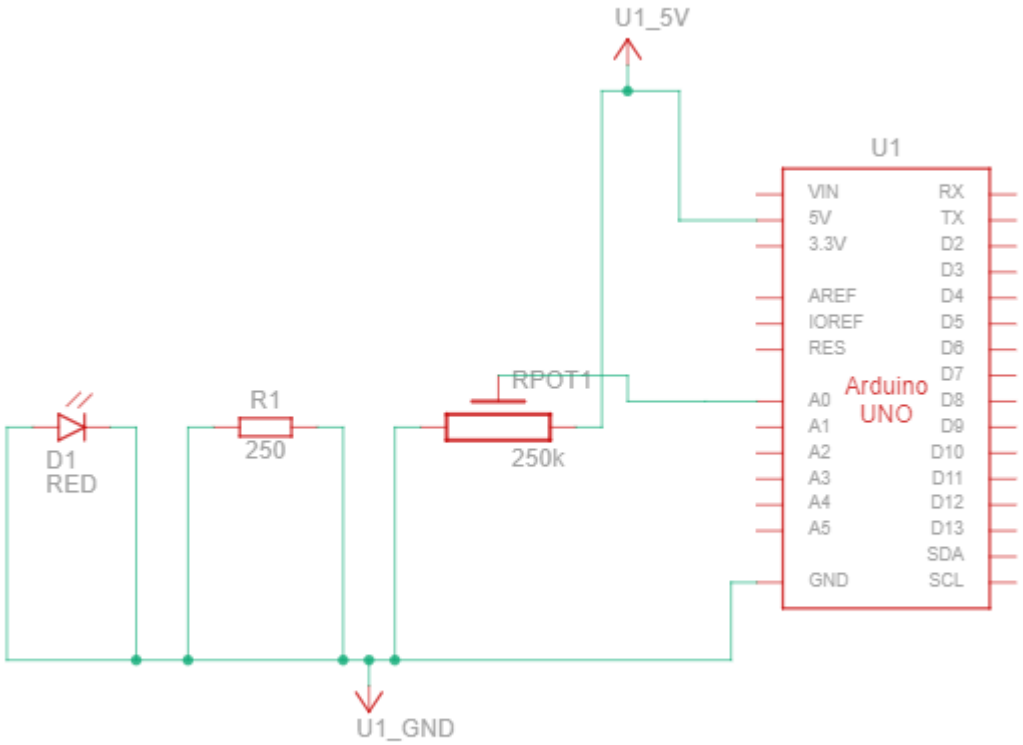
- Arduino Board
- Servo Motor
- Jumper Wires
- Potentiometer
- USB cable for Arduino
- Computer with Arduino and Python installed

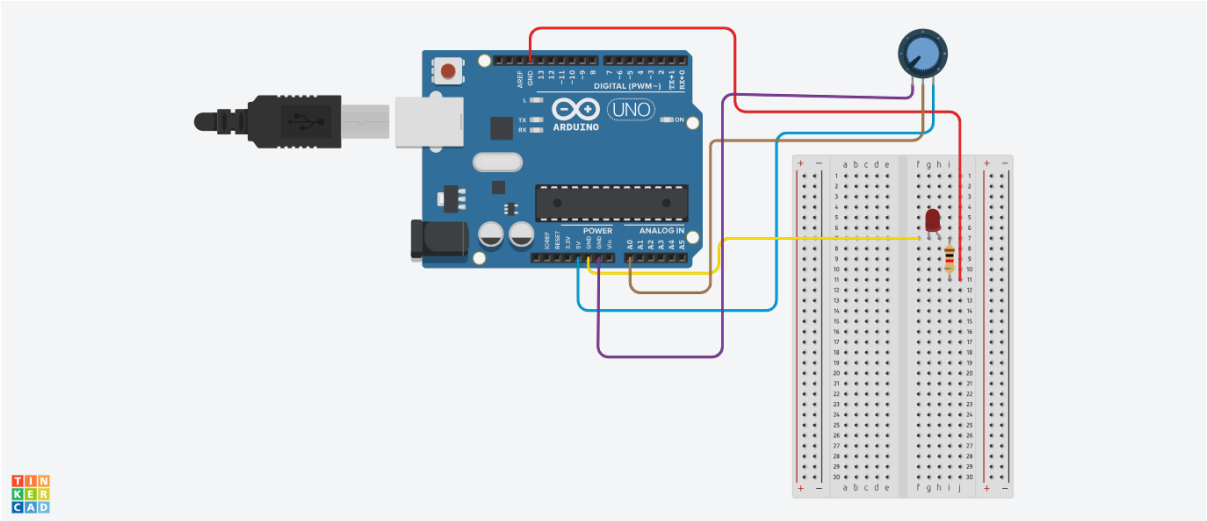
Experimental setup

PART 3A: LED & Potentiometer

To perform the experiments, two of the legs of the potentiometer are connected to 5V and ground respectively on the Arduino. Meanwhile the middle leg connected to the A0 which is an analog input pin on Arduino. On the other hand, the negative leg, called the cathode, with its shorter leg, connects to ground. All of the components are connected using jumpers. Finished with the connection, the Arduino is connected to the computer via a USB cable. Arduino IDE is used to code the instructions and upload it into the Arduino to execute it. After that, the Python script is 'Run' on the computer to display the potentiometer readings as the potentiometer knob is turned. This implies the serial communication between the Arduino and Python in which the python used 'serial' library for it.

There is no extra setup to read the potentiometer readings graphically required. Only the Python script was updated by installing the 'matplotlib' library.

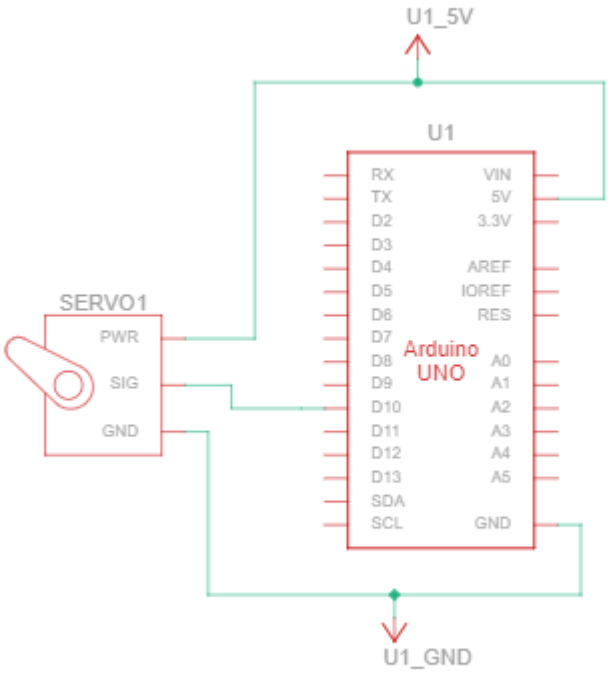


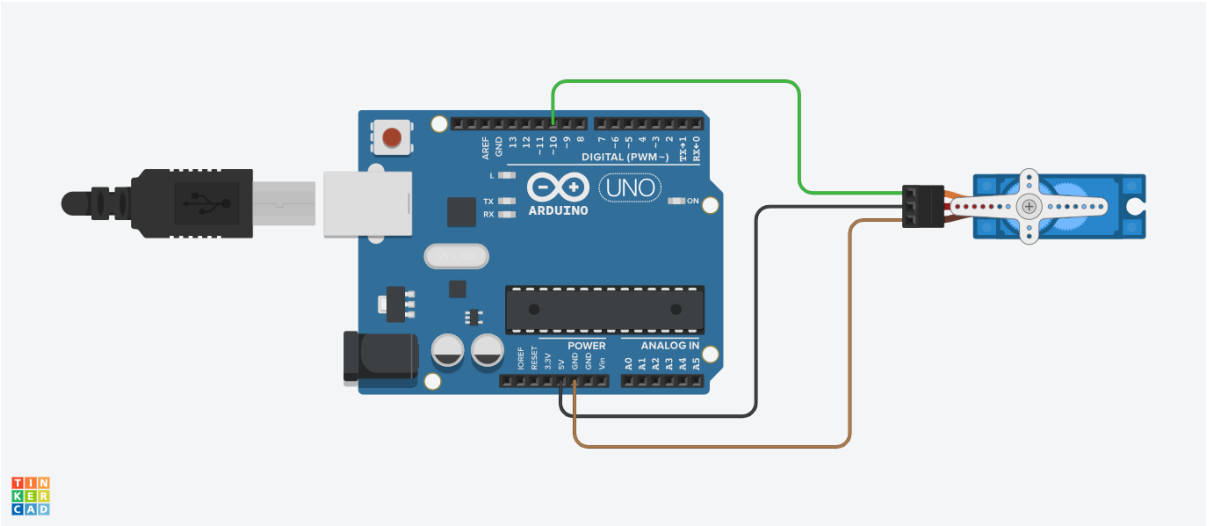


PART 3B: Servo Motor

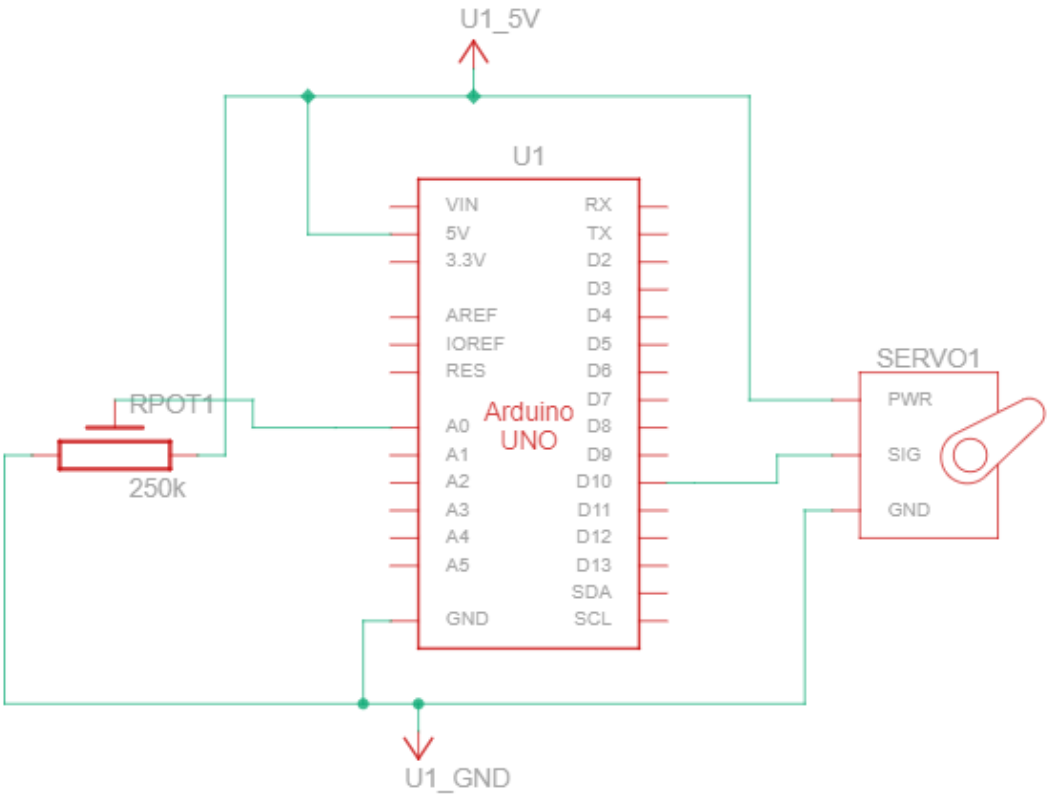
For part B, the servo motor had three signal wires which are PWM-capable pin (orange), power wire (red) and ground wire (brown). The PWM is connected to the digital pin 10, power wire to 5V output on the Arduino and ground wire to the ground. To code the

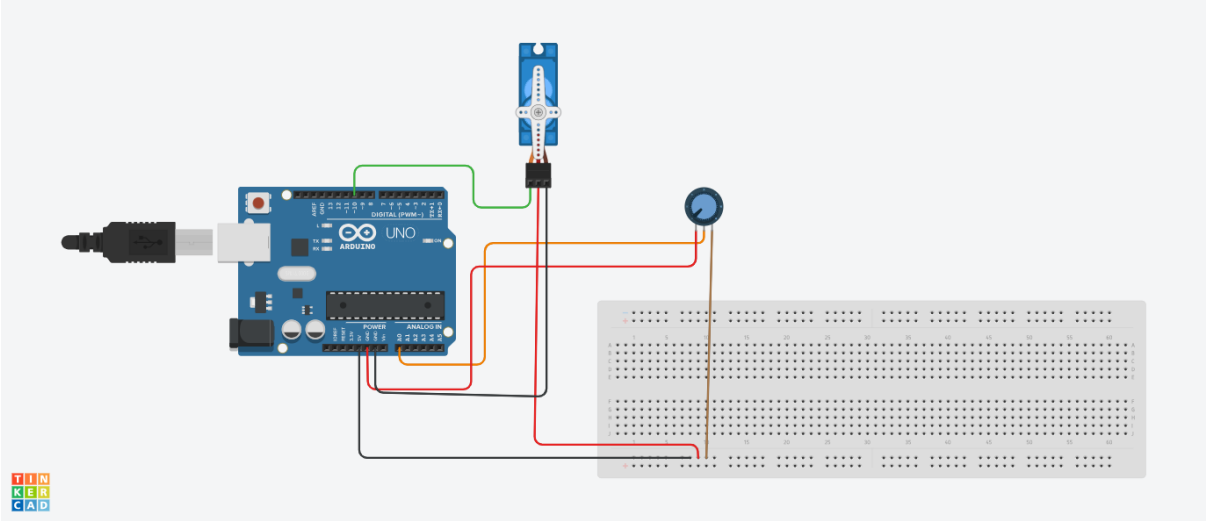
instructions, ‘Servo’ library is used where the code can read the angle data from the serial port and move the servo accordingly. Then, upload to the Arduino board. After that, a Python script is written to prompt the user to enter the servo angle from 0 to 180 degrees to control the servo. The range is due to restrictions, the type of servo motor used is in that range. By key in any value within the range and press the ‘enter’ key in the keyboard, the servo motor rotated to the specified angle.





The setup to incorporate a potentiometer for real-time adjustments of the servo motor's angle is simply by adding the potentiometer where two of the legs of the potentiometer are connected to 5V and ground respectively on the Arduino. Meanwhile the middle leg connected to the A0 which is an analog input pin on Arduino.





Methodology

PART 3A: LED & Potentiometer

For part 3A, it will focus on satisfying the Arduino-python interaction. The first step is to connect the potentiometer to the arduino. Wire the potentiometer so that the outer pins are connected to 5V and GND while the middle pin is to analog pins such as A0 pin. Arduino code will be used to read the potentiometer values and then transmitted over the serial port. The data at the serial port from the Arduino will be read by Python using the 'pyserial' library. The communication would be considered established if the real time potentiometer values were displayed in the python terminal while the potentiometer knob was turned. By following those steps, we can use the interpreted data from potentiometer to turn on the LED. The programming codes used as below:

Arduino IDE	Python

int sensorPin = A0; // the potentiometer is connected to analog pin 0
int ledPin = 13; // the LED is connected to digital pin 13
int sensorValue; // an integer variable to store the potentiometer reading
void setup() { // this function runs once when the sketch starts up
pinMode (ledPin, OUTPUT);
Serial.begin(9600);
}
 	
void loop() { // this loop runs repeatedly after setup() finishes
sensorValue = analogRead(sensorPin); // read the sensor
Serial.println(sensorValue); // output reading to the serial line
	import serial
	
	ser = serial.Serial('COM3', 9600)
	
	try:
	while True:
	pot_value = ser.readline().decode().strip()
	print("Potentionmeter Value:", pot_value)
	except KeyboardInterrupt:
	ser.close()
	print("Serial connection closed.")

```
  if (sensorValue < 500){
    digitalWrite(ledPin , LOW );}       // turn
the LED off
  else {
    digitalWrite(ledPin , HIGH );}      // keep
the LED on
  delay (1000);                         // Pause in
milliseconds before next reading
}
```

We can illustrate the potentiometer readings with a graph by updating the existing code using Matplotlib. The code is as below:

Updated Arduino code

```
import serial
import matplotlib.pyplot as plt
from time import sleep

# Set your serial port and baud rate
serial_port = "COM3"
baud_rate = 9600
dt = 0.05

# Open the serial port
ser = serial.Serial(serial_port, baud_rate)

if ser.is_open:
    print(f"Serial port {ser.name} is open.")
    data = []
    N = 100  # number of data
    for k in range(N):
```

```
        b = ser.readline()
        strn = b.decode()
        str1 = strn.rstrip()
        flt = float(str1)
        data.append(flt)
        sleep(0.05)

    # Close the serial port
    ser.close()
    if not ser.is_open:
        print(f"Serial port {ser.name} is closed.")

    # Plot the reading
    plt.plot(data)
    plt.xlabel('Time(seconds)')
    plt.ylabel('Potentionmeter Reading')
    plt.title('Potentionmeter Reading vs Time')
    plt.grid(True)
    plt.show()
else:
    print(f"Failed to open serial port {serial_port}.")
```

PART 3B: Servo Motor

For part 3B, this part demonstrates how to integrate a servo motor with an Arduino and use Python to control it. The first step is to make sure that the servo's signal wire was attached to an Arduino PWM-capable pin. For this part, we will connect it to digital pin 9. The GND and 5V pins on the Arduino are used to power the servo. The 5v is connected to the servo's power wire and the chosen ground pin will be connected to the servo's ground wire. For this part, we will focus more on the interaction between python and arduino than the fist part. The Python will prompt us to enter the desired angle between 0 to 180 only and then send the data to the serial port. Uploaded Arduino code will be used to read angle data from the serial port

and then drive the servo. The Arduino script will display the specified angle data. The python then will allow us to exit the script by clicking the ‘q’ character. The programming codes used as below:

Arduino IDE	Python

#include <Servo.h>
 	
Servo servo;
int servoPin = 9; //initiate pin on arduino
int angle = 90; //inital angle
 	
void setup() {
  servo.attach(servoPin);
  servo.write(angle);
  Serial.begin(9600);
}
 	
void loop() {
  if (Serial.available() > 0) {
  angle = Serial.parseInt(); //parseInt reads characters from the serial input until it encounters a non numeric charcter, then it converts the collected character into intigers
  servo.write(angle);
  delay(15); // Add a small delay for smoother motion
  }
}
	import serial
	import time
	
	ser = serial.Serial('COM4', 9600)
	try:
	 while True:
	  angle = input("Enter servo angle (0-180 degrees): ")
	  if angle.lower() == 'q':
	  break
	  angle = int(angle)
	  if 0 <= angle <= 180:
	  # Send the servo’s angle to the Arduino
	  ser.write(str(angle).encode())
	  else:
	  print("Angle must be between 0 and 180 degrees.")
	except KeyboardInterrupt:
	 pass # Handle keyboard interrupt
	finally:
	 ser.close() # Close the serial connection
	print("Serial connection closed.")

By using the updated code below, we can incorporate a potentiometer for real-time adjustments to the servo motor's angle. Using the updated Arduino code, it will interact with updated python code by serial port connection so that the user could stop the servo motor data execution.

Updated Arduino code	Updated Python code

#include <Servo.h>
 	
Servo myservo; // Create a servo object
 	
int potPin = A0;   // Analog pin for potentiometer
int val;   // Potentiometer value
int servoPos = 90;   // Initial servo position
boolean haltFlag = false; // Flag to indicate if servo should be halted
 	
void setup() {
  myservo.attach(9); // Attach the servo to pin 9
  Serial.begin(9600); // Initialize serial communication
}
 	
void loop() {
  if (!haltFlag) { // Check if the servo should not be halted
    val = analogRead(potPin); // Read
	import serial
	import time
	
	ser = serial.Serial('COM8', 9600)
	try:
	    while True:
	        angle = input("Enter servo angle (0-180 degrees) or 's' to stop: ")
	        if angle.lower() == 's':
	            ser.write(b's') # Send 's' to stop the servo
	            break
	        angle = int(angle)
	        if 0 <= angle <= 180:
	            ser.write(str(angle).encode()) # Send the servo's angle to the Arduino
	        else:
	            print("Angle must be between 0 and 180 degrees.")
	except KeyboardInterrupt:
	    pass # Handle keyboard interrupt
	finally:
	    ser.close() # Close the serial

potentiometer value
servoPos = map(val, 0, 1023, 0, 180);
// Map potentiometer value to servo angle
 	
myservo.write(servoPos); // Set the servo position
}
 	
if (Serial.available() > 0) {
char key = Serial.read(); // Read a character from serial input
if (key == 's') { // Check if 's' character is received
haltFlag = !haltFlag; // Toggle the haltFlag
if (haltFlag) {
myservo.write(90); // Halt the servo at a specific position (90 degrees)
Serial.println("Servo halted.");
} else {
Serial.println("Servo resumed.");
}
}
}
}
	connection
	print("Serial connection closed.")

Data collection

PART 3A: LED & Potentiometer

LED Behavior: In this phase, there was close monitoring of how the LED responded to potentiometer knob adjustments. The objective was to understand how the LED responded to changes in the potentiometer resistance.

Potentiometer Readings: This involved the collection of analog readings using Arduino and sending them to Python. The objective was to assess the potentiometer performance and influence on the overall system.

Python Data Display: The Python script displayed real-time potentiometer data and confirmed that the data received from the Arduino was accurately displayed in the Python environment.

Graphical Visualisation: In order to visualize the graphs and understand the behavior of the potentiometer over time, Matplotlib was used to create graphs of potentiometer data.

PART 3B: Servo Motor

Servo Motor Control: In this phase, there was observation of the servo motor responses to Python-sent angle commands. The initial position of the servo motor was noted as well as its movements according to changes in angles.

The data collected was used for the following data analysis.

Data analysis

PART 3A: LED & Potentiometer

When we rotate and calibrate the potentiometer, it will influence the behaviour of an LED, causing it to illuminate. The resulting data is not only visualised and displayed within the

Python script but is also presented in the serial monitor, offering a graphical representation of this interaction.

PART 3B: Servo Motor

The servo motor responds to angle commands sent by the Python script through a serial connection with an Arduino UNO. It starts at 90 degrees and adjusts its position based on the Python input. The angle is limited to a range of 0 to 180 degrees, ensuring that the servo stops moving if a command exceeds these boundaries.

Result

Serial communication between the Arduino and Python script was successfully established. Data transmission occurred seamlessly, allowing for real-time interaction.The Arduino accurately read the potentiometer's analog values and transmitted them to the Python script. This enabled the Python script to access and utilise the potentiometer readings for further processing. The experiment effectively demonstrated the remote control of a servo motor using Python and Arduino. As potentiometer values changed, the Python script sent corresponding commands to the Arduino, instructing the servo motor to adjust its angle.

In conclusion, the experiment successfully met its goals, offering practical insight into setting up serial communication, sending potentiometer data, and showcasing remote servo motor control.

Discussion

PART 3A: LED & Potentiometer

- **LED Conduct**

When turning the potentiometer knob, the LED turned on, showing that the potentiometer was generating a voltage that was above a certain threshold. This behavior indicated that the Arduino and Python code correctly responded to changes in the potentiometer reading. This

experiment demonstrates how the Arduino can control the LED based on the potentiometer reading.

- **Python Data Display**

The Python code successfully read the data and displayed the information sent from the Arduino over the serial connection. It showed the potentiometer's real-time readings as the knob was adjusted. Python allowed us to collect and visualize the data from the Arduino in real-time.

- **Potentiometer**

The potentiometer acted as a voltage divider. Turning the knob changed the resistance, which changed the voltage input. The Arduino read this voltage and decided whether to turn on/off the LED based on the threshold set in the code.

- **Graph result**

The generated graph showed the trend in potentiometer readings over time. As you turned the knob, the graph showed fluctuations in the readings. This graph was useful for analyzing trends and patterns in the data. The graph generated by Python provided a visual representation of the data. This data could be further analyzed to identify patterns, trends, or anomalies, which were crucial in various data-driven applications.

Expected: The LED will turn off when the Potentiometer is turned on.
Observed: The LED does not turn on when the Potentiometer is turned on.

The first time we experimented, when we turned on the potentiometer knob the LED did not turn on. The potentiometer reading displayed was successfully shown at the Arduino and Python. So, we checked the wiring connection between the potentiometer, LED, and Arduino to ensure that the wiring was correct and there were no loose connections.

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 visualisation 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).

To generate and showcase the graph, we imported the necessary library first, including Matplotlib, which was used to create plots and graphs. When we turned on/off the potentiometer knob, the potentiometer reading was displayed in the Python script. The Python graph was displayed according to the result of the potentiometer knob value in the Python script. The code was included in the Methodology part.

PART 3B: Servo Motor

- **Servo Motor Control**

The experiment successfully demonstrates the control of a servo motor using Arduino and Python. The servo's movement corresponds to the angle values sent via the serial connection from the Python script. This indicates that the communication and control mechanisms are working as intended.

- **Arduino Uno**

The Arduino acts as an intermediary between the Python script and the servo motor. It receives angle commands from Python, processes them, and sends appropriate signals to the servo, resulting in its movement. This demonstrates the ability of microcontrollers like Arduino to interface with external devices.

There were several issues that we encountered when experimenting. We concluded the issues that we encountered and also the solutions to each issue:

Servo behaviour: The servo did not move as expected, sometimes overshooting and failing to reach the specified angle. We rechecked the code that we had written and made some improvements to it.

Python script error: The Python script encountered errors, causing it to terminate unexpectedly. We carefully reviewed the Python script for syntax and logic errors and implemented error handling to catch and gracefully handle exceptions.

Arduino code error: The Arduino code had issues processing the data received from Python or controlling the servo. We reviewed the Arduino code for data parsing errors and implemented error handling to prevent crashes.

Q: 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 can 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.

Using the updated Arduino and Python code that we attached at Methodology part, we could observe that when we turn on/off the potentiometer, the potentiometer value will be increased and decreased at the serial monitor. However, when we press the ‘s’ character, the Python script will halt the serial connection between Arduino and Python.

Conclusion

PART 3A: LED & Potentiometer

In this experiment, the group successfully showcased how a potentiometer, an LED, and Arduino-Python communication can interact with each other. The LED's response to changes in the resistance of the potentiometer confirmed that the system was working as intended. The Python script effectively presented real-time data. Incorporating Matplotlib for visualising data improved our understanding of the information. There was an issue with the

LED initially which highlighted the importance of wiring connections and emphasised how crucial attention to detail is in mechatronic systems.

PART 3B: Servo Motor

In this experiment, the group demonstrated control of a servo motor by coordinating Arduino and Python. The servo motor's dependable response to angular instructions sent through a serial connection validated the reliability of the communication system. The Arduino played a role as an intermediary facilitating command exchange between Python and the servo motor. Although there were challenges, the code was revised carefully and effectively resolved the issues. Both experiments offered insights into how components interact in mechatronic systems laying a solid groundwork for further exploration in this field.

Recommendations

Both part A and part B of the experiments apply serial communication in its implication. It is important to understand the way serial communication works. Most of the errors occurred while doing the experiments including not changing the port in Arduino IDE as the ports for each laptop are not the same, not having proper Python library to execute the code and delays for so long. While the delay is good to check the change in angle value, too long delays cause the timing to not be that much accurate with the readings and is hard to read. Hence the recommendations for future improvements:

1. **Understanding Serial Communication:**

Before starting any experiment involving serial communication between Python and Arduino, it's crucial to have a clear understanding of how serial communication works. Study the basics of serial communication, including baud rates, data bits, stop bits, and flow control, to ensure a reliable connection.

2. **Port Configuration:**

Be aware that the port assigned to your Arduino may vary depending on your computer. Always check and configure the correct COM port in the Arduino IDE under the "Tools" menu. Make a note of the port used for each setup to avoid confusion.

3. **Use the Right Python Libraries:**

Ensure that you have the necessary Python libraries installed to execute your code. For serial communication, the pySerial library is commonly used. Verify that you have it installed in your Python environment. If not, install it using pip.

4. **Optimise Delays:**

While delays can be useful for monitoring changes, long delays can affect the real-time accuracy of readings and responsiveness. Optimise the use of delays in your code to strike a balance between real-time monitoring and accuracy. Consider using small delays or alternative techniques, such as reading data on an event-driven basis.

5. **Code Modularity:**

Organise your Python and Arduino code into modular functions or classes. This improves code readability and maintainability, making it easier to troubleshoot and expand your projects.

By implementing these recommendations, you'll enhance the reliability and efficiency of your serial communication experiments and minimise potential errors.

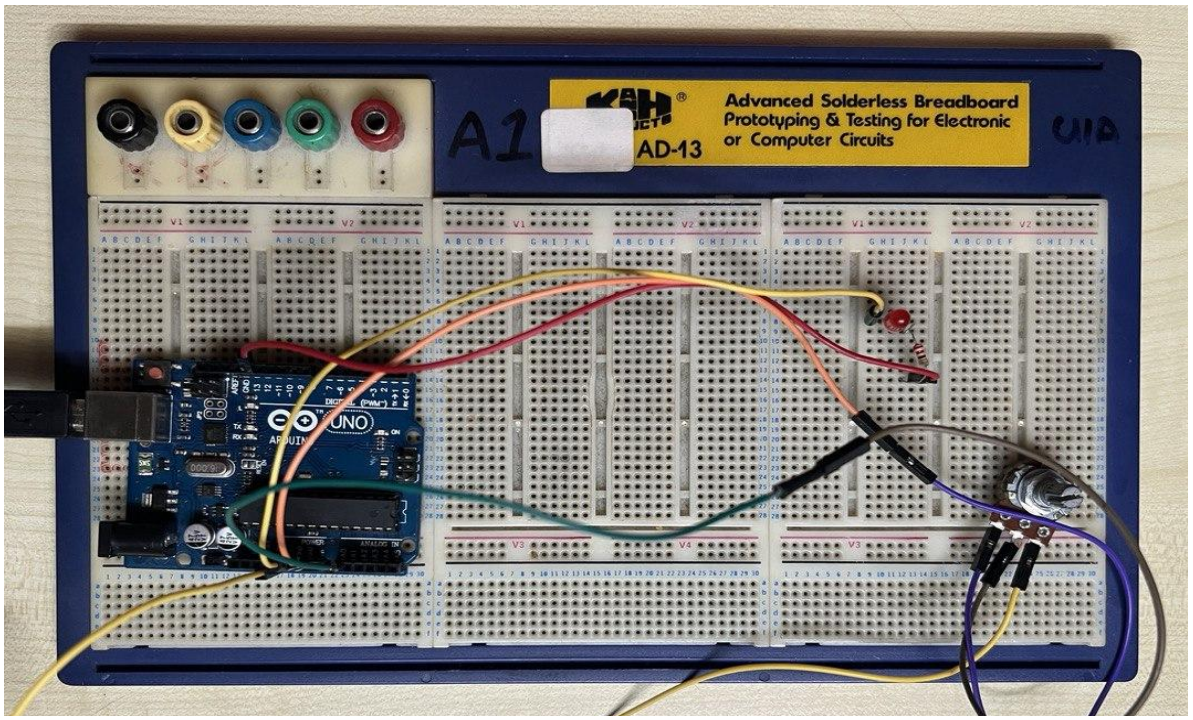
Reference

Kumar, S., Das, M. R., Kushalkar, R., Venkat, N., Gourshete, C., & Moudgalya, K. M. (2021, June). Microcontroller Programming with Arduino and Python. https://www.researchgate.net/profile/Sudhakar-Kumar-2/publication/353849431_Microcontroller_programming_with_Arduino_and_Python/links/61177e891ca20f6f861ecc9c/Microcontroller-programming-with-Arduino-and-Python.pdf

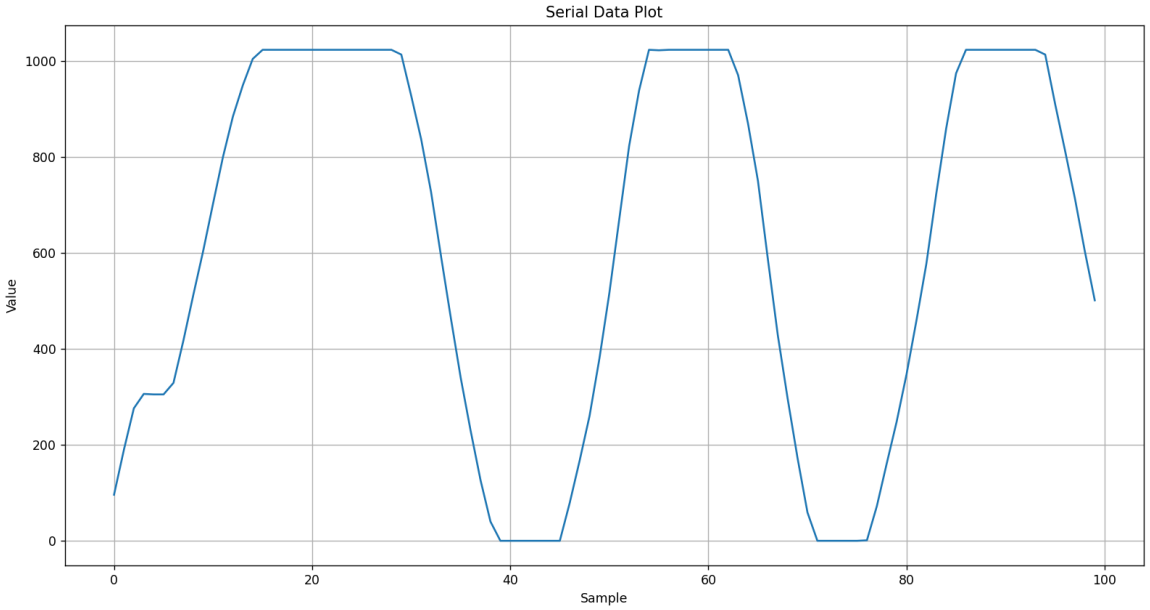
Zulkifli. (2014). Mechatronics Interfacing Lab Manual, (Rev. ed.). Unpublished Class Materials.

Appendices

PART 3A: LED & Potentiometer

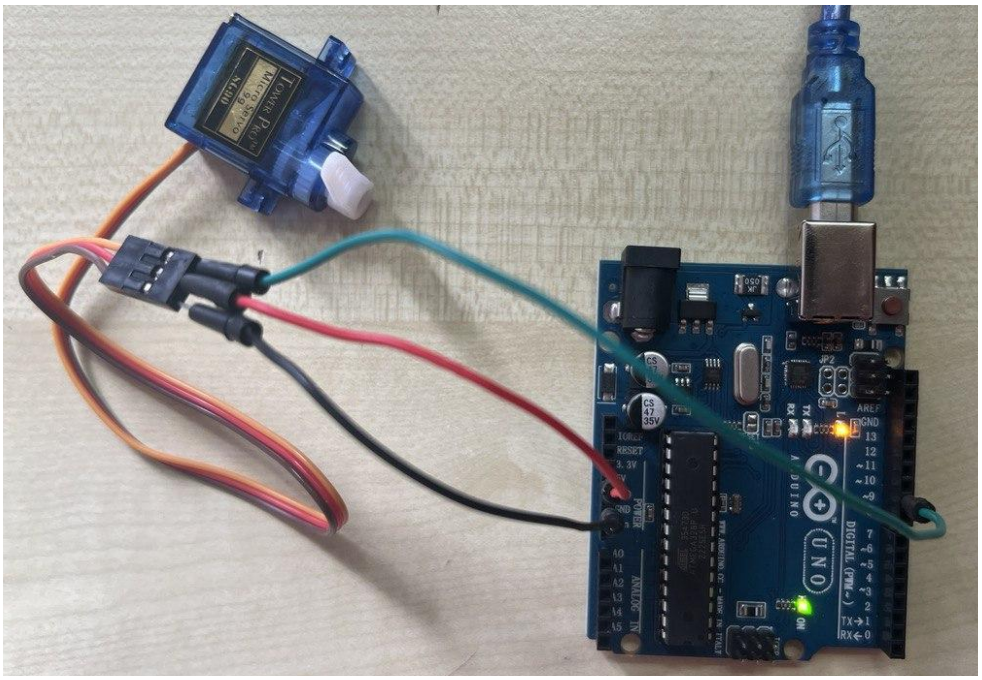


The final version of the circuit connection between the potentiometer and LED

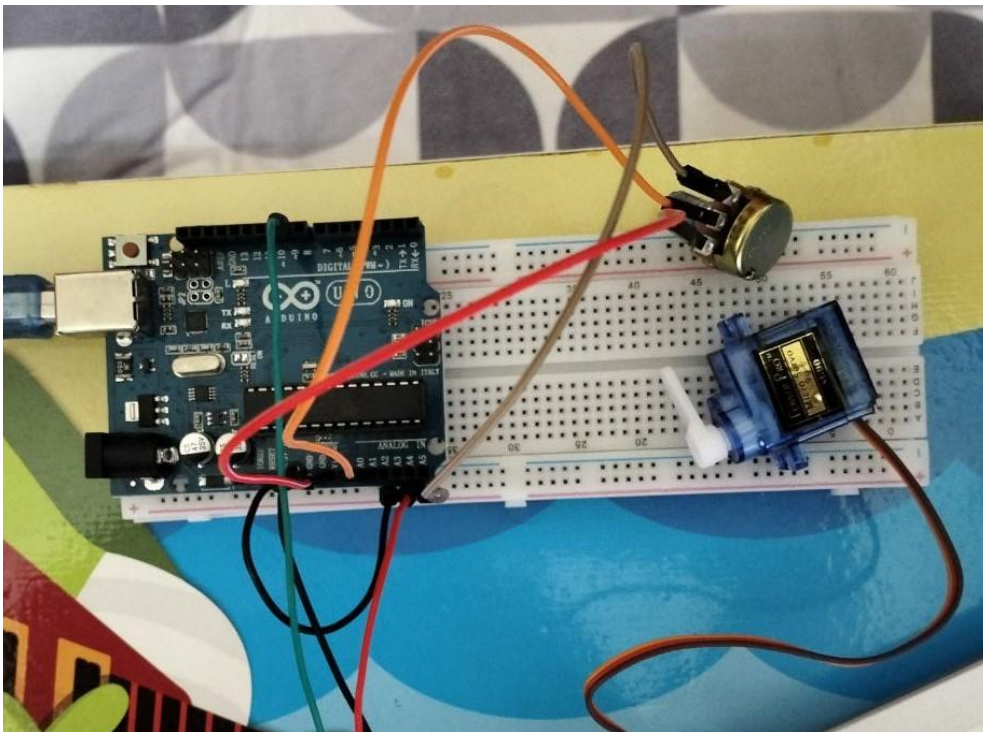


Potentiometer reading graphically using Matplotlib in Python script

PART 3B: Servo Motor



The final version of the circuit connection between the servo motor and Arduino



The final version of the circuit connection between the potentiometer and servo motor

Acknowledgement

We would like to express our sincere gratitude to the individuals who provided invaluable assistance, guidance, and support during this experiment. First and foremost, we extend our appreciation to Dr. Wahju Sediono for their comprehensive instruction and mentorship throughout the experiment. Their insights, feedback, and enthusiasm played an important role in our understanding of Arduino programming.

Our fellow group members also deserve special acknowledgment for their collaboration and support. Our discussions, knowledge-sharing, and problem-solving sessions greatly enriched our understanding of this experiment's concepts and enhanced the overall learning experience. The collective contributions of our group members have not only enriched our learning experience but have also significantly contributed to the successful completion of this project.

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 acknowledgment, and that the original work contained herein has 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 **read** and **understand** the content of the report and no further improvement on the report is needed from any of the individual’s contributions 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	AISYAH SOFEA OTHMAN	**Read** ✅
Matric Number	2115386	**Understand** ✅
Contribution	• Material & Equipment • Discussion	**Agree** ✅

Signature		
Name	SITI ALIAA BINTI IBRAHIM	**Read** ✅
Matric Number	2112618	**Understand** ✅
Contribution	• Abstract • Methodology	**Agree** ✅

Signature		
Name	ADLI FAHMI	**Read** ✅
Matric Number	2113095	**Understand** ✅
Contribution	• Result • Data Analysis	**Agree** ✅

Signature		
Name	ETHAR OSMAN	**Read** ✅
Matric Number	2111282	**Understand** ✅
Contribution	• Data Collection	**Agree** ✅

	• Conclusion	

Signature	***ainaa***	
Name	NURAIN AINAA AQILAH BINTI ROSLI	**Read** ✅
Matric Number	2114560	**Understand** ✅
Contribution	• Introduction • Experimental Setup • Recommendations	**Agree** ✅