



الجامعة الإسلامية العالمية ماليزيا
INTERNATIONAL ISLAMIC UNIVERSITY MALAYSIA
يُونِيسَيْتِي اِسْلَامْ اِنْتَارَايَغُسِيَا مِلَيْسِيَا
Garden of Knowledge and Virtue

KULLIYAH OF ENGINEERING
DEPARTMENT OF MECHATRONICS ENGINEERING

Mechatronics System Integration (MCTA3203)

WEEK 3: Serial Communication

Section 1
(Group E)

Adam Mikhail Bin Khairul Effendy	2117613
Muhammad Afiq bin Mazhalimi	2110491
Muhammad Fahmi Ikhwan bin Ahmad Suparjo	2111167
Eiman Bin Azzam	2110565
Zulhilmi bin Abdul Rahman	2117679

Date of submission: 25/10/2023
Prepared for: Dr. Zulkifli bin Zainal Abidin

Abstract

Data interchange between a computer and a microcontroller is frequently accomplished via serial communication between an Arduino and Python. In order to create serial communication, code must be written for both the Arduino and Python platforms. We wish to send potentiometer readings in this experiment.

Table of Contents

1.1 Introduction.....	3
1.2 Materials & Equipments	3
1.3 Experimental setup	4
1.4 Methodology	6
1.5 Data Collection	7
1.6 Data Analysis.....	7
1.7 Results	8
1.8 Discussion.....	9
1.9 Conclusion	11
1.10 Recommendations	11
1.11 References	11
1.12 Appendices.....	12
1.13 Acknowledgments	12
1.14 Student's Declaration.....	13

1.1 Introduction

Serial communication between Python and Arduino is a common way to exchange data between a computer and a microcontroller. To establish serial communication, the code must be written on both the Python side and the Arduino side. In this experiment, we will send potentiometer readings from an Arduino to a Python script through a USB connection.

Objectives:

- 1) To know how to set up a serial communication.
- 2) To know the purpose of using serial communication.

1.2 Materials & Equipments

PART A

- 1) Arduino Board
- 2) Potentiometer
- 3) Jumper Wires
- 4) LED
- 5) 220 resistor
- 6) Breadboard

PART B

- 1) Arduino Board
- 2) Servo motor
- 3) Jumper wires
- 4) Potentiometer

- 5) USB cable for Arduino
- 6) Computer with Arduino IDE and Python installed

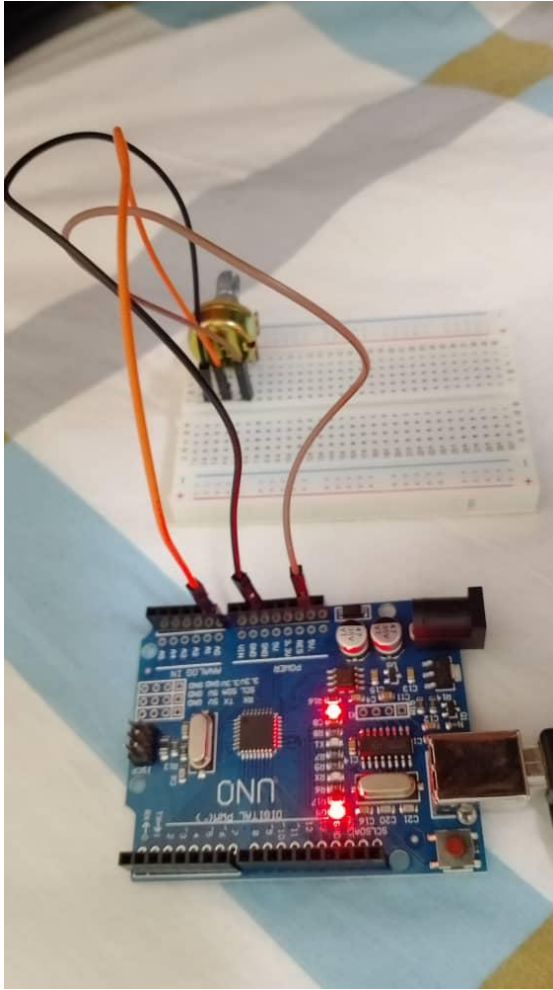
1.3 Experimental Setup

PART A

Circuit 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.

Show below is our circuit setup.



PART B

Circuit setup:

- 1) Connect the Servo's Signal Wire: Usually, you connect the servo's signal wire to a PWM capable pin on the Arduino (e.g., digital pin 9).
- 2) Power the servo using the Arduino's 5V and GND pins. Servos typically require a supply voltage of +5V. You can connect the servo's power wire (usually red) to the 5V output on the Arduino board.

3) Connect the Servo's Ground Wire: Connect the servo's ground wire (usually brown) to one of the ground (GND) pins on the Arduino.

1.4 Methodology

PART A

1. Connect the Arduino to your computer via a USB cable.
2. Powered on the Arduino (uploaded the sketch to our Arduino using the Arduino IDE).
3. Run the Python script on your computer.
4. As you turned the potentiometer knob, we saw the potentiometer readings displayed in the Python terminal.
5. We used these readings for various experiments, data logging, or control applications, depending on your project requirements.
6. Opened the Serial Plotter: In the Arduino IDE, went to "Tools" -> "Serial Plotter."
7. Selected the Correct COM Port: In the Serial Plotter, selected the correct COM port to which our Arduino was connected.
8. Set the Baud Rate: Ensured that the baud rate in the Serial Plotter matched the one set in our Arduino code (e.g., 9600).
9. Read Real-Time Data: As we turned the potentiometer knob, the Serial Plotter displayed the potentiometer readings in real-time, creating a graphical representation of the data. We could see how the values changed as you adjusted the potentiometer.
10. Customized the Plotter: We could customize the Serial Plotter by adjusting settings such as the graph range, labels, and colors.

PART B

1. The circuit was constructed according to the circuit setup instructions. (Appendix A)
2. The provided Arduino code was uploaded and modified to our Arduino Mega.
(Appendix B)
3. The Serial Monitor was opened using the Arduino IDE.
4. Python code was copied and modified from the module.
5. The user will put input to Python serial
6. The Arduino received a signal from Python and moved the servo.
7. The Servo's Signal Wire should be connected: Usually, the servo's signal wire is connected to a PWM-capable pin on the Arduino (e.g., digital pin 9).
8. The servo was powered using the Arduino's 5V and GND pins. Servos typically require a supply voltage of +5V. The servo's power wire (usually red) was connected to the 5V output on the Arduino board.
9. The Servo's Ground Wire was connected: The servo's ground wire (usually brown) was connected to one of the ground (GND) pins on the Arduino.

1.5 Data Collection

- No data collection

1.6 Data Analysis

- No data analysis

1.7 Results

As the potentiometer twisted, the reading was plotted by serial plotter from Arduino IDE.

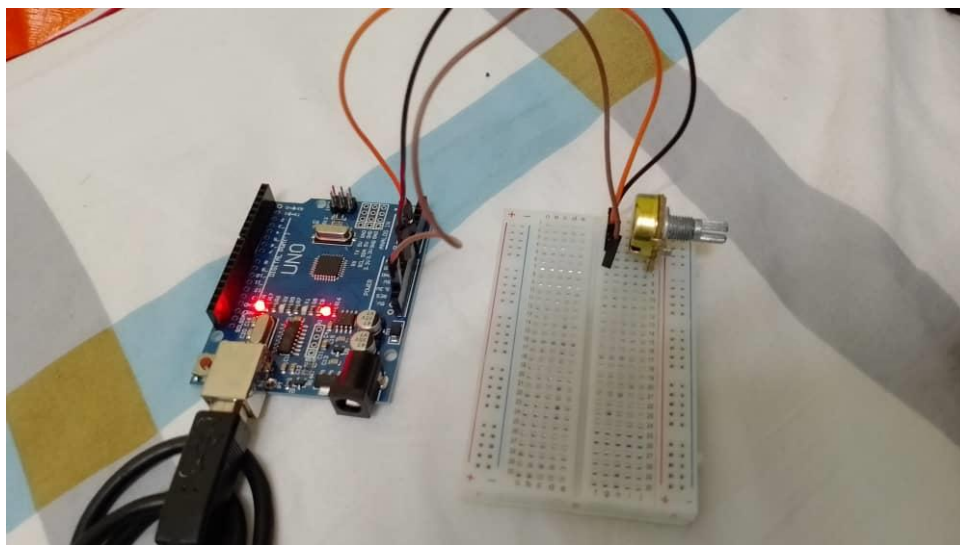


Figure 1

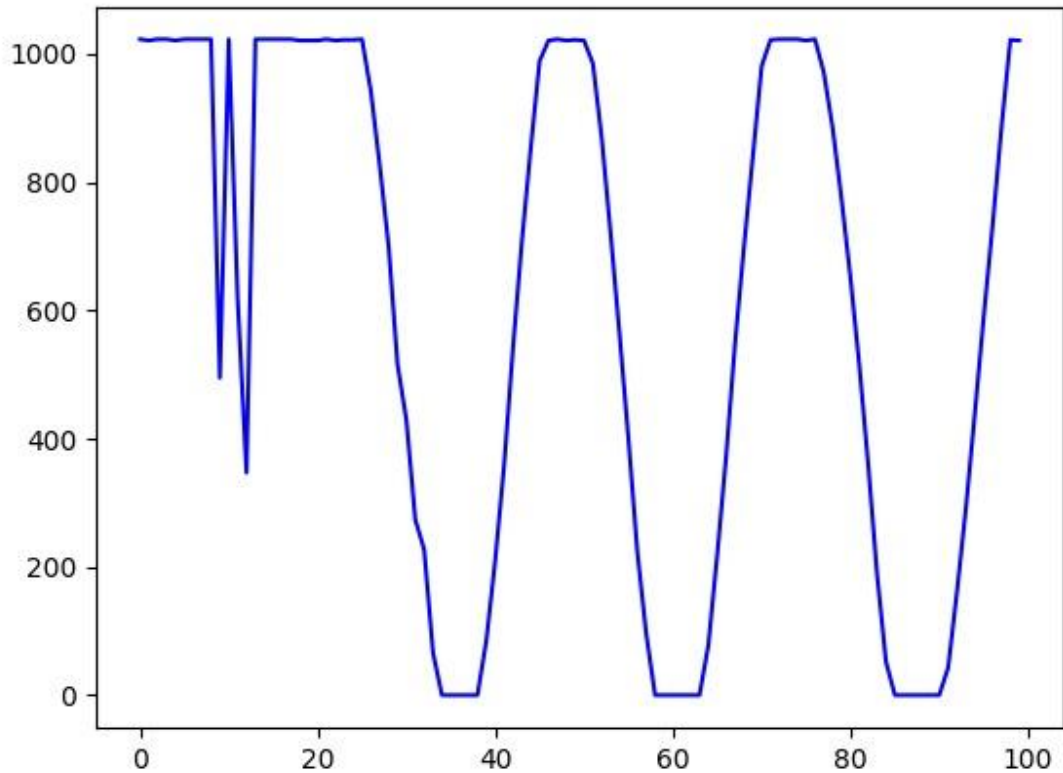
The angle that users enter into the Python prompt, which spans from 0° to 180° , determines the servo motor's movement. The Python script then transfers this angle to the Arduino using the serial connection.

1.8 Discussion

PART A

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).



PART B

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.

We successfully managed to control the angle of the servo angle in real-time by adjusting the potentiometer.

Below is the code written in the Arduino IDE:

```
#include <Arduino.h>
#include <Servo.h>

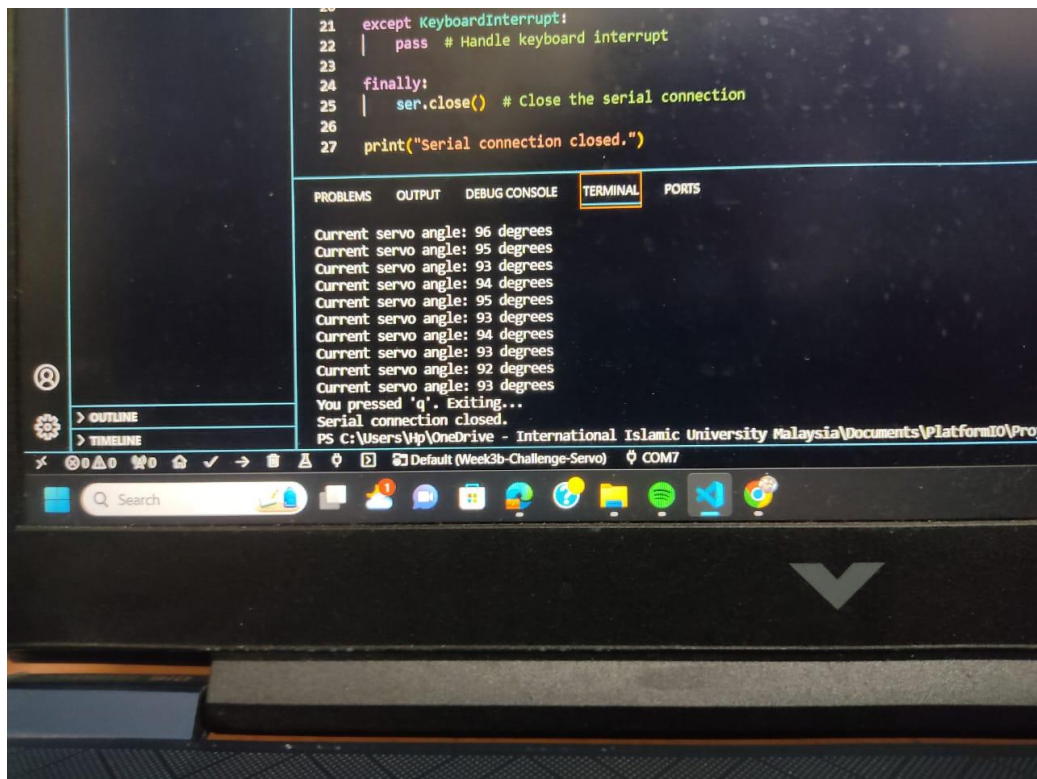
// Create a servo object
Servo myservo;

// Define the analog pin for the potentiometer
int potentiometerPin = A0;

// Variables to store the potentiometer value and the mapped servo position
int potValue;
int servoPosition = -1;
int currentServoPosition;
void setup() {
    // Attach the servo to pin 9
    myservo.attach(9);
```

```
Serial.begin(9600);  
}  
  
void loop() {  
  // Read the potentiometer value  
  potValue = analogRead(potentiometerPin);  
  
  // Map the potentiometer value (0-1023) to the servo angle range (0-180)  
  currentServoPosition = map(potValue, 0, 1023, 0, 180);  
  
  // Set the servo position based on the mapped potentiometer value  
  myservo.write(servoPosition);  
  
  // Send the servo angle over the serial port only when there's a significant change  
  if (currentServoPosition != servoPosition) {  
    servoPosition = currentServoPosition;  
    Serial.println(servoPosition);  
  }  
  
  // A short delay to stabilize the servo  
  delay(15);  
}
```

The current servo angle is printed on the python code in real-time as the potentiometer changes angle of the servo.



The image shows a screenshot of a computer screen displaying a code editor (VS Code) with a Python script and a terminal window. The Python code is as follows:

```
20  
21 except KeyboardInterrupt:  
22 |     pass # Handle keyboard interrupt  
23  
24 finally:  
25 |     ser.close() # close the serial connection  
26  
27 print("Serial connection closed.")
```

The terminal window, titled "TERMINAL", shows the following output:

```
Current servo angle: 96 degrees  
Current servo angle: 95 degrees  
Current servo angle: 93 degrees  
Current servo angle: 94 degrees  
Current servo angle: 95 degrees  
Current servo angle: 93 degrees  
Current servo angle: 94 degrees  
Current servo angle: 93 degrees  
Current servo angle: 92 degrees  
Current servo angle: 93 degrees  
You pressed 'q'. Exiting...  
Serial connection closed.  
PS C:\Users\Hp\OneDrive - International Islamic University Malaysia\Documents\PlatformIO\Projects\Week3b-Challenge-Servo>
```

The terminal window also shows the file path: "PS C:\Users\Hp\OneDrive - International Islamic University Malaysia\Documents\PlatformIO\Projects\Week3b-Challenge-Servo". The terminal is connected to "COM7".

Below is a video regarding this question.

Video link:

https://drive.google.com/file/d/1Eb_Gut_oSwARMKrlu1Du0aRzwIQLcpnr/view?usp=sharing

1.9 Conclusion

The particular goals and outcomes that the experiment seeks to achieve will determine how a serial data experiment from Python to Arduino ends. However, in a broader sense, one could conclude that the serial connection between Python and Arduino has been established successfully. This achievement allows data to be transferred from Python to Arduino for a variety of uses, including remote control and sensor data transfer. The experiment's success depends on a number of variables, including the accuracy of data transfer, Arduino data processing, and the accomplishment of planned objectives.

In summary, this experiment was effective and opened the door for additional research that builds on the fundamental knowledge of serial communication. This information can be used in complex systems that make considerable use of several microcontrollers and processors. It marks the beginning of an even wider range of communication opportunities.

1.10 Recommendations

Make sure the Python and Arduino codes are performance-optimized. This entails reducing pointless loops, enhancing data structures, and steering clear of blocking processes. Enable thorough error handling in the Python and Arduino code to deal with unforeseen circumstances in an elegant manner. Provide users with informative error messages to aid in troubleshooting. Divide the code up into modules or functions that accomplish particular tasks. This improves the readability, reuse, and maintainability of the code.

1.11 References

1. Welcome to pySerial's documentation — pySerial 3.4 documentation. (n.d.).
<https://pyserial.readthedocs.io/en/latest/>
2. analogRead() - Arduino Reference. (n.d.).
<https://www.arduino.cc/reference/en/language/functions/analog-io/analogread/>

1.12 Appendices

- Refer to GitHub

1.13 Acknowledgments

We would like to acknowledge and give our thanks to our instructor, Dr. Wahyu Sediono, for helping us understand the topic of Serial Communication and basic interfacing while also giving us room to explore and experiment freely. May we meet again on more advanced topics, surely his advice will be beneficial in our future projects.

Also, we would like to thank Allah S.W.T for giving us guidance throughout our lives and helping us overcome the difficulties faced in making this experiment.

1.14 Student's Declaration

Declaration:

We certify that this project/assignment is entirely our own work, except where we have given fully documented references to the work of others, and that the material in this assignment has not previously been submitted for assessment in any formal course of study.

adam

Date: 1/11/2023

Name: Adam Mikhail Bin Khairul Effendy

fahmi

Name: Muhammad Fahmi Ikhwan Bin Ahmad Suparjo

Eiman

Name: Eiman Bin Azzam

afiq

Name: Muhammad Afiq bin Mazhalimi

Zul

Name: Zulhilmi bin Abdul Rahman