LABORATORY PROJECT REPORT

SERIAL COMMUNICATION BETWEEN ARDUINO AND PYHTON

EXPERIMENT 3B

Section 1

Group 2

Semester 1 2025/2026

NO	NAME	MATRIC NO
1.	ADAM NURUDDIN BIN HELMI	2215783
2.	NUR FATIHAH HANNANI BINTI HASMAYADI	2227062
3.	NUR SYAHZMAN AZIQ BIN MOHD SHAHMADI	2310037

Date of submission: 29th October 2025

Abstract

This experiment investigates the use of serial communication between an Arduino Uno microcontroller and a Python-based computer interface to achieve real-time servo motor control. The objective is to transmit angle data and potentiometer readings through a serial link, allowing the servo motor to respond dynamically to user or sensor input. The Arduino is programmed to receive commands from Python, interpret the data, and adjust the servo's position accordingly, while the Python script handles user interaction, serial transmission, and optional data visualization. A potentiometer is integrated into the setup to enable continuous, real-time adjustment of the servo angle, enhancing system interactivity. The experiment emphasizes key principles such as baud rate synchronization, sensor-actuator interfacing, and bidirectional serial data exchange. The results demonstrate successful integration of hardware and software, illustrating a practical application of serial communication in sensor-based control and automation systems.

Table of Contents

1.0 **Introduction**

2.0 **Materials and Equipment**

    2.1 Electronic Components

    2.2 Equipment and Tools

3.0 **Experimental Setup**

4.0 **Methodology**

5.0 **Data Collection**

6.0 **Data Analysis**

7.0 **Results**

8.0 **Discussion**

9.0 **Conclusion**

10.0 **Recommendations**

11.0 **References**

1.0 Introduction

The purpose of this experiment is to explore serial communication between an Arduino Uno microcontroller and a Python interface for controlling a servo motor in real time. The experiment demonstrates how angle data or sensor input can be transmitted from a computer to the Arduino via a USB serial connection to achieve precise servo positioning. By integrating a potentiometer into the circuit, the servo's angle can be adjusted dynamically, providing continuous and interactive control without relying solely on user input through the Python console.

This experiment introduces key concepts in embedded systems such as serial data transmission, actuator control, and sensor feedback integration. The Arduino is programmed to interpret the incoming serial data and actuate the servo motor accordingly using the Servo library, while the Python script handles user commands, potentiometer data, and optional graphical visualization through matplotlib.

The hypothesis is that, with proper serial communication setup and baud rate synchronization, the servo motor will accurately respond to both Python inputs and potentiometer variations, reflecting smooth and real-time motion control. This experiment reinforces fundamental principles of mechatronic system integration, highlighting the seamless interaction between hardware and software for automated control applications.

2.0 Materials and Equipment

2.1 Electronic Components

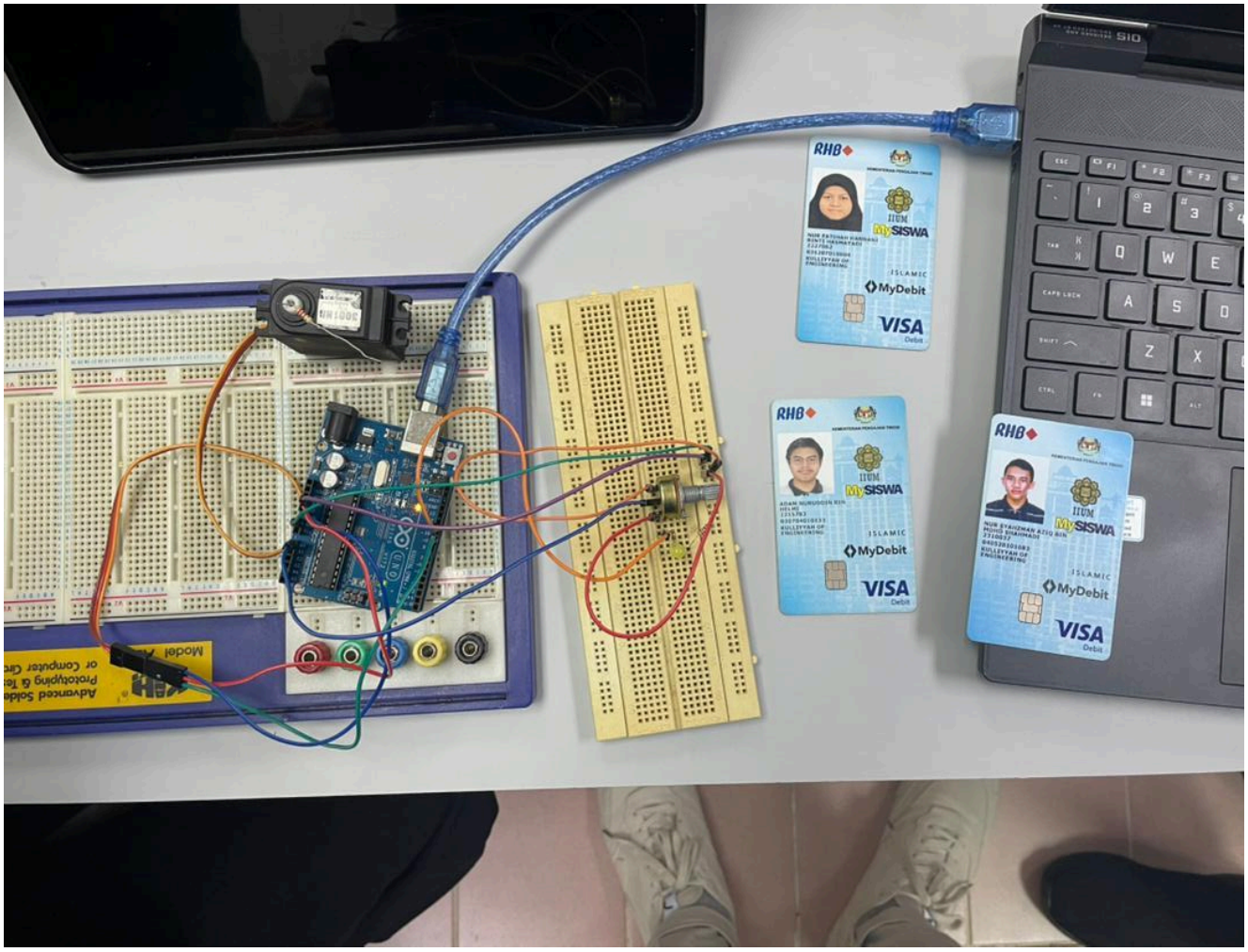
- Arduino Uno
- Servo motor
- Potentiometer
- Jumper Wires
- Breadboard
- USB cable

2.2 Equipment and Tools

- Arduino IDE
- Python with pyserial library
- Power Supply (5V via USB or external source)

3.0 Experimental Setup

1. The potentiometer was connected to the Arduino Uno with one leg to 5V pin on arduino, the other leg to gnd, and middle leg to pin A0.
2. Servo motor was connected to the Arduino with the red one to the 5V, the orange one to pin D9, and the brown one to the GND.
3. Arduino was connected to the computer via USB, as shown in Figure 1, and run the Python script to begin reading potentiometer values in the terminal.
4. The potentiometer knob was adjusted to observe the values displayed in real-time
5. Serial plotter in Arduino IDE was opened to view the real time data fron the potentiometer adjustment.
6. Ensure the correct COM port and baud rate (9600) are selected.
7. An LED connected to the pin 9 and 220-ohm resistor added as an optional output indicator connected to a digital pin to observe visual feedback after the potentiometer value exceeded half.
8. All components were connected using jumper wires on a breadboard for a secure and solderless setup.


RHB
IIUM
MySISWA
ISLAMIC
MyDebit
VISA
Debit

4.0 Methodology

1. Setup the Arduino Mega 2560
2. IDE code implementation
3. Python code implementation
4. Run the python code
5. Experiment the servo motor
6. Read the angle in python terminal
7. Code snippet

4.1 Detailed steps

1. Assemble the circuit as stated in the experimental setup.
2. Connect the arduino to the computer via usb cable, and upload the code to arduino IDE.
3. Close the arduino IDE and open python to run the python code.
4. Test the python code by inserting input angle of servo motor.
5. Adjust the potentiometer knob to observe the angle difference of servo motor and led light state.

4.2 Programming codes

Arduino

```
#include <Servo.h>


Servo myservo;

int potPin = A0;

int ledPin = 13;

int potValue = 0;

int angle = 0;

bool manualMode = false;  // start in potentiometer mode
```

```
void setup() {
  myservo.attach(9);
  pinMode(ledPin, OUTPUT);
  Serial.begin(9600);
  Serial.println("Arduino ready.");
}

void loop() {
  if (Serial.available() > 0) {
    String data = Serial.readStringUntil('\n');
    data.trim();

    if (data.equalsIgnoreCase("pot")) {
      manualMode = false;
      Serial.println("Switched to POT mode");
    }
    else if (data.equalsIgnoreCase("manual")) {
      manualMode = true;
      Serial.println("Switched to MANUAL mode");
    }
    else if (manualMode && data.toInt() >= 0 && data.toInt() <= 180) {
      angle = data.toInt();
      myservo.write(angle);
```

```
      Serial.print("Manual angle set to: ");
      Serial.println(angle);
    }
  }

  // If in potentiometer mode
  if (!manualMode) {
    potValue = analogRead(potPin);
    angle = map(potValue, 0, 1023, 0, 180);
    myservo.write(angle);

    if (angle > 90) digitalWrite(ledPin, HIGH);
    else digitalWrite(ledPin, LOW);

    Serial.println(angle);
    delay(100);
  }
}
```

Python

```
import serial
import time

ser = serial.Serial('COM5', 9600, timeout=1)  # change COM3 to your real port
time.sleep(2)
print("Connected to Arduino.\n")
print("Commands:")
print("  manual  → switch to manual angle input mode")
print("  pot     → switch back to potentiometer mode")
print("  0–180   → set servo angle (manual mode only)")
print("  x       → exit\n")

try:
    while True:
        cmd = input("Enter command or angle: ")
        if cmd.lower() == 'x':
            break
        ser.write((cmd + '\n').encode())  # send with newline
        time.sleep(0.1)

        if ser.in_waiting > 0:
            print("Arduino:", ser.readline().decode().strip())
```

```
except KeyboardInterrupt:
    pass
finally:
    ser.close()
    print("Serial connection closed.")
```

5.0 Data Collection

Angle	LED
0	OFF
90	OFF
91	ON

6.0 Data Analysis

In potentiometer mode, the servo angle changes from 0° to 180° based on the analog input. The LED acts as an indicator, turning on when the servo angle exceeds 90° and off when the servo angle exceeds 90°. This confirms that the analog-to-digital conversion, angle mapping, and conditional LED control are working properly. The serial monitor displays the actual timing angle value, confirming that the servo movement directly corresponds to the potentiometer position.

7.0 Results

The system works well in both potentiometer and manual modes. In potentiometer mode, the servo moves smoothly from 0° to 180° when the knob is turned, and the LED lights up when the angle exceeds 90°. In manual mode, the servo moves quickly to the angle entered via Python without any delay or error. The LED also responds correctly, indicates that the wiring and coding were working as expected.

8.0 Discussion

In this experiment, a Python application and an Arduino board communicated serially to operate a servo motor. In order for the servo to rotate to a predetermined point, the goal was to transmit angle data from Python to Arduino over USB. The servo motor was adjusted by the Arduino when an angle between 0° and 180° was entered in the Python interface.

Through serial connectivity, this configuration effectively illustrated two-way interaction between software and hardware. The Arduino carried out the servo's actual motion, while the Python application served as the command interface. The system's precise response to the entered angles demonstrated both the efficacy of serial data transfer and the accuracy of servo control. Later addition of a potentiometer allowed for real-time angle modification, and the LED functioned as a visible servo position indication. This improved knowledge of how various parts of a control system may cooperate.

9.0 Conclusion

By effectively operating a servo motor with Python inputs sent to an Arduino via serial connection, the experiment met its goals. Reliable data transfer and control were demonstrated by the servo motor's rotation, which matched the angle values supplied in the Python application exactly. The experiment also demonstrated how real-time feedback and sensor inputs may be incorporated into microcontroller applications, which serve as the foundation for robotics and automation systems.

10.0 Recommendations

To avoid communication issues, it is advised to confirm the right COM port and baud rate before to executing the application. When operating continuously, the servo motor should be powered appropriately to prevent jitter or overheating. In Python, incorrect angle entries can be avoided by implementing limitations or input validation. Future enhancements may include adding sensors to allow for autonomous servo control or integrating graphical visualisation with matpotlib for real-time monitoring. Maintaining steady and dependable communication also requires proper error handling and serial port closure.

11.0 References

Using Python and an Arduino to Read a Sensor :
https://pythonforundergradengineers.com/python-arduino-potentiometer.html

12.0 Acknowledgements

Special thanks to ZULKIFLI BIN ZAINAL ABIDIN & WAHJU SEDIONO for their guidance and support during this experiment.

13, Student's Declaration

Certificate of Originality and Authenticity

We hereby certify that we are collectively responsible for the work presented in this report. The content reflects our original work, except where specific references or acknowledgements have been made. No part of this report has been completed or submitted by individuals or sources not identified within.

Furthermore, we confirm that this report is the result of a collaborative effort, with contributions made by all group members. The extent of each individual's contribution is documented 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: ADAM NURUDDIN BIN HELMI	Read	[/]
Matric Number: 2215783	Understand	[/]
Signature: *ADAM*	Agree	[/]

Name: NUR FATIHAH HANNANI BINTI HASMAYADI	Read	[/]
Matric Number: 2227062	Understand	[/]
Signature: *FAT*	Agree	[/]

Name: NUR SYAHZMAN AZIQ BIN MOHD SHAHMADI	Read	[/]
Matric Number: 2310037	Understand	[/]
Signature: *AZIQ*	Agree	[/]