SEM 1 25/26

SECTION 1

COURSE CODE: MCTA 3202

LECTURER NAME:
ZULKIFLI BIN ZAINAL ABIDIN

WEEK 3: SERIAL COMMUNICATION BETWEEN ARDUINO AND PYTHON/ CONTROLLING SERVO MOTOR VIA SERIAL COMMUNICATION

PREPARED BY:
GROUP 1

NAME	MATRIC NUMBER
AMIR MUIZZUDDIN BIN NORAIDIL RAZMAN	2129579
MUHAMMAD HAIKAL HANIF BIN ABDUL RAZAK	2213297
AHMAD ILHAM BIN ABDUL AZIZ	2112109

TASK 1

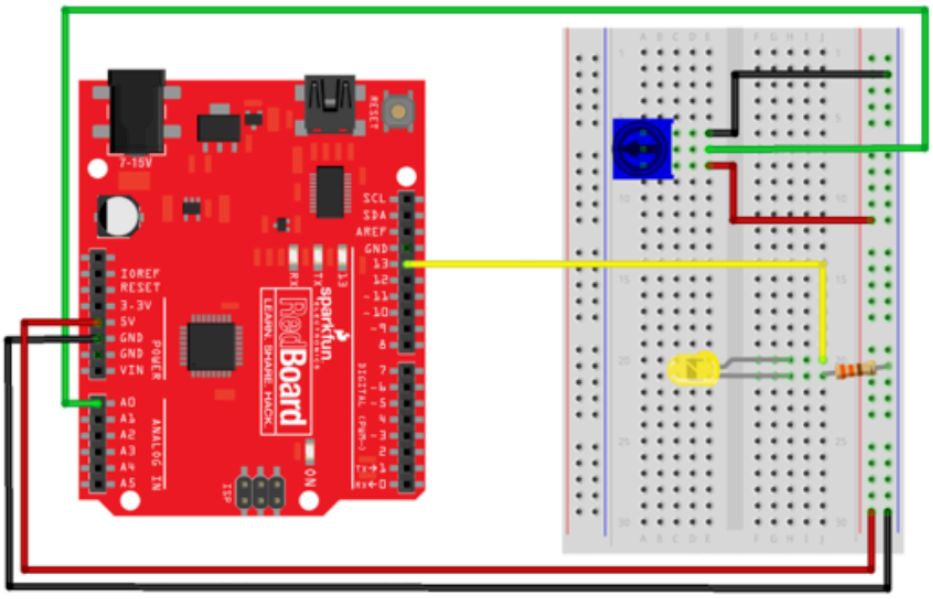
INTRODUCTION

In this task, a potentiometer is used to generate analog signals that are transmitted from Arduino to Python through serial communication. The purpose is to establish a data link where sensor readings can be continuously monitored in Python. This process introduces students to serial data handling and real-time visualization of analog input. The experiment also reinforces proper baud rate matching and communication synchronization between hardware and software platforms.

MATERIALS

- Arduino board
- Potentiomenter
- LED
- 220 ohm resistor
- Breadboard
- Jumper wires
- USB cable
- Computer with Python(Pyserial installed

EXPERIMENT SETUP



1. Circuit Setup
    - Connect one leg from the potentiometer to GND
    - Connect one leg from the potentiometer to 5v
    - Connect the middle leg of the potentiometer to an analog input pin on the Arduino, A0
    - Connect the LED with resistor with the anode being connected to the desired output pin (pin 13) and the cathode connected to GND
2. Computer Setup
    - Connect the arduino to the computer via USB cable
    - Upload the programming prompt into the Arduino

METHODOLOGY

Arduino Setup:

- The Arduino reads analog input from the potentiometer.
- The values (0–1023) are sent to the Python terminal through serial communication.
- The LED turns ON if the potentiometer value is above 512, and OFF otherwise.

Python Setup:

- Python reads and prints potentiometer values in real time.
- pyserial library is used for serial communication.
- matplotlib is used for plotting potentiometer readings dynamically.

Observation:

- Turning the potentiometer knob changes the displayed value on Python.
- LED lights up when the value exceeds half of its range.

Arduino Code :

```
void setup() {

  Serial.begin(9600);

  pinMode(13, OUTPUT);
```

```
}

void loop() {
  int potValue = analogRead(A0);
  Serial.println(potValue);
  delay(100); // prevents data flooding

  if (potValue > 512) {
    digitalWrite(13, HIGH);
  } else {
    digitalWrite(13, LOW);
  }
}
```

Python Code :

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

COM_PORT = 'COM4'
BAUD_RATE = 9600
ser = serial.Serial(COM_PORT, BAUD_RATE)
time.sleep(2)

plt.ion()
```

```
fig, ax = plt.subplots()
x_vals, y_vals = [], []
print("Reading potentiometer values... Press Ctrl+C to stop.\n")
try:
    while True:
        if ser.in_waiting > 0:
            data = ser.readline().decode().strip()
                    if data.isdigit():
                pot_value = int(data)
                print(f"Potentiometer Value: {pot_value}")


                    if pot_value > 512:
                   ser.write(b'1')  # Turn LED ON
                else:
                   ser.write(b'0')  # Turn LED OFF


                # Update live plot
                x_vals.append(len(x_vals))
                y_vals.append(pot_value)
                ax.clear()
                ax.plot(x_vals, y_vals, label="Potentiometer Value")
                ax.set_title("Real-Time Potentiometer Reading")
                ax.set_xlabel("Sample")
                ax.set_ylabel("Value (0–1023)")
                ax.legend()
```

```
            plt.pause(0.1)


except KeyboardInterrupt:
    print("\nProgram stopped by user.")
finally:
    ser.close()
    plt.ioff()
    plt.show()
    print("Serial connection closed.")
```

CONCLUSION

In conclusion, the given code demonstrates a basic but effective configuration using a potentiometer as input and LED as the output. By using the Arduino, it reads the analogue input from the potentiometer. After the potentiometer reaches the half threshold, the Arduino sends an output that turns on the LED. The Python software accepts the data and outputs the potentiometer value, demonstrating bidirectional connection between the Arduino and PC.

This configuration serves as a starting point for more complicated applications that require incorporating sensor inputs and output control. It also shows us the flexibility and capabilities using Arduino with external components, as well as the benefits of using Python's in presenting and processing data

TASK 2

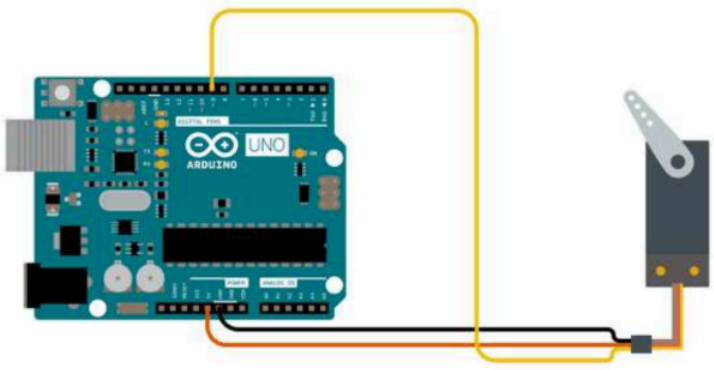
INTRODUCTION

In this task, a servo motor is controlled through Python commands sent to the Arduino using serial communication. By entering angle values in Python, the servo rotates accordingly, demonstrating bidirectional data exchange between computer and microcontroller. The experiment is later enhanced with a potentiometer for real-time angle adjustment, LED indicators for feedback, and graphical visualization using *matplotlib* to display sensor data. This activity strengthens understanding of hardware control, feedback systems, and interactive communication in mechatronic applications.

MATERIALS

- Arduino board
- Servo motor (signal pin - 9)
- Potentiometer
- LED
- Jumper wires
- Breadboard
- USB cable
- Computer with Python(pyserial and matplotlib libraries installed)

EXPERIMENT SETUP



1. Circuit Board Setup
    - Connect the servomotor to the arduino to their respective pins; GND, 5V, Any output pins available
    - Connect the LED with resistor with the anode being connected to the desired output pin (pin 13) and the cathode connected to GND
    - Connect one leg from the potentiometer to GND
    - Connect one leg from the potentiometer to 5v
    - Connect the middle leg of the potentiometer to an analog input pin on the Arduino, A0
2. Computer Setup
    - Set a Arduino program that can halt its process using input from the keyboard (in this experiment we use ‘S’), and can Light up the LED after reaching half of the threshold on the potentiometer
    - Upload it to the Arduino via USB cable
    - Set a python code that can plot the coordinate on the graph

METHODOLOGY

Arduino Code:

Read potentiometer values.

Maps analog input (0–1023) to servo angle (0°–180°).

Turn LED ON when servo angle > 90°.

Sends servo angle to Python via serial.

Pauses or resumes operation based on keyboard commands (s for stop, r for resume).

Python Code:

Reads live servo angle values and plots them in real time.

Sends control commands (s, r) to Arduino via serial.

Displays servo angle and LED status in the terminal.

Coding Arduino :

```
#include <Servo.h>

Servo myServo;
int potPin = A0;
int potValue = 0;
int angle = 0;
bool running = true; // control flag for servo operation

void setup() {
  Serial.begin(9600);
  myServo.attach(9); // servo signal pin connected to D9
  pinMode(potPin, INPUT);
  pinMode(13, OUTPUT); // Added: set pin 13 (LED) as output
}

void loop() {
  if (running) {
    potValue = analogRead(potPin);
```

```
  angle = map(potValue, 0, 1023, 0, 180);
  myServo.write(angle);
  Serial.println(angle); // send angle data to Python
  delay(100);

  // Added: LED control based on potentiometer value
  if (potValue > 512) { // Half of 1023 ≈ 512
    digitalWrite(13, HIGH); // Turn ON LED
  } else {
    digitalWrite(13, LOW); // Turn OFF LED
  }
}

// Check if there is any command from Python
if (Serial.available()) {
  char command = Serial.read();

  if (command == 's') { // stop command
    running = false;
    myServo.detach(); // stop servo
    Serial.println("Execution halted by user.");
  }
  else if (command == 'r') { // resume command
    if (!running) {
```

```
      myServo.attach(9); // reattach servo control
      running = true;
      Serial.println("Execution resumed.");
    }
  }
 }
}
```

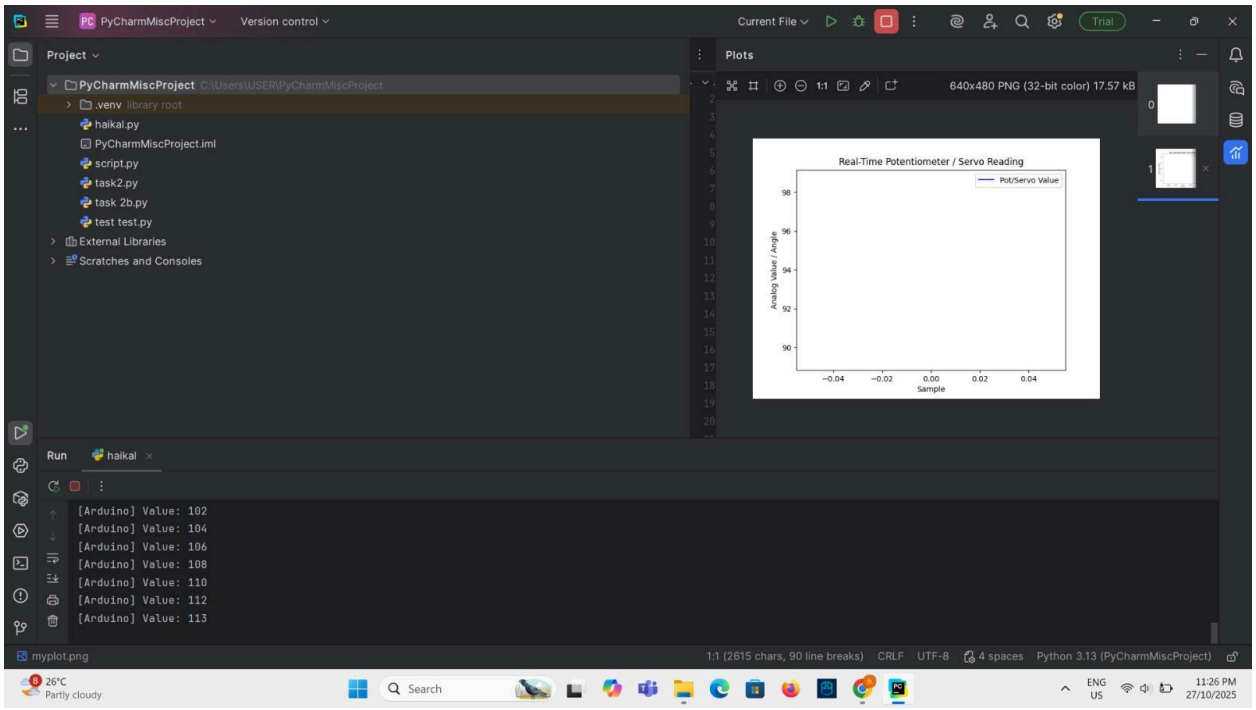
Python Coding :

```
import serial
import matplotlib.pyplot as plt
import keyboard  # pip install keyboard
import time
ser = serial.Serial('COM4', 9600, timeout=1)
time.sleep(2)  # wait for Arduino to reset
plt.ion()
fig, ax = plt.subplots()
x_vals, y_vals = [], []
print("========================================")
print(" Real-Time Potentiometer / Servo Plot ")
print("========================================")
print("Press 'r' to start/resume the Arduino process.")
print("Press 's' to stop the Arduino process.")
print("Close the plot window or press Ctrl+C to exit.\n")
```

```
def send_command(key, label, command_byte):
    """Send 's' or 'r' command once when pressed."""
    if keyboard.is_pressed(key):
        ser.write(command_byte)
        print(f"[Python] {label} command sent to Arduino.")
        while keyboard.is_pressed(key):
            pass


try:
    while True:
        send_command('s', 'Stop', b's')
        send_command('r', 'Resume', b'r')
            if ser.in_waiting > 0:
            data = ser.readline().decode(errors='ignore').strip()
            if data.isdigit():  # ensure it's numeric
                value = int(data)
                print(f"[Arduino] Value: {value}")
                # --- Append and plot live data ---
                x_vals.append(len(x_vals))
                y_vals.append(value)
                ax.clear()
                ax.plot(x_vals, y_vals, color='b', linewidth=2, label='Pot/Servo Value')
                ax.set_title("Real-Time Potentiometer / Servo Reading")
```

```
            ax.set_xlabel("Sample")
            ax.set_ylabel("Analog Value / Angle")
            ax.legend()
            plt.pause(0.05)  # refresh plot
except KeyboardInterrupt:
    print("\n[Python] Keyboard interrupt detected. Exiting...")
finally:
    ser.close()
    plt.ioff()
    plt.show()
    print("[Python] Serial connection closed.")
```



CONCLUSION

In conclusion, by using Arduino, we were shown that it managed to translate the analog input from the potentiometer into an angle. It shows the capability of Arduino to manage the potentiometer and turn the Servo motor as well as turn on the LED after reaching above 90 degree.We also manage to make a prompt where we manage to halt the servo motor using the keyboard input. By using the python also we could view the coordinates that have been plotted.

What we obtain during both of this task Arduino IDE is mostly used for translating input into an output and Python mostly used for plotting or presenting data