Mechatronics System Integration (MCTA3203)

Week **3**: Serial Communication (ver. 3.0)

Serial Communication between Arduino and Python

Table of Contents

Experiment 1: Reading Potentiometer Values via Serial Communication

Objective:
To transmit potentiometer readings from Arduino to Python via USB serial.

1. Required Hardware:

- Arduino board
- Potentiometer
- Jumper wires
- Breadboard

2. Connections:

Connect the potentiometer to the Arduino. Wire one end of the potentiometer to 5V, the other end to GND, and the *center pin* to an analog input pin on the Arduino, e.g., **A0** (see **Fig. 1**).

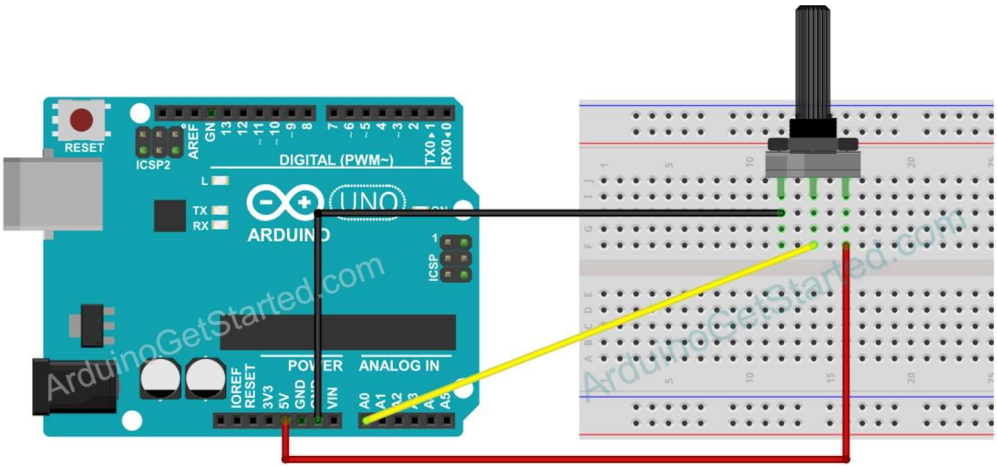

Fig. 1. Wiring Diagram

3. Arduino Code:
Write an Arduino code/sketch to read the potentiometer value and send it over the *serial port*. Here's a basic example:

```
void setup() {
      Serial.begin(9600);
}

void loop() {
      int potValue = analogRead(A0);
      Serial.println(potValue);
      delay(1000); // add a delay to avoid sending data too fast
}
```

Upload this code/sketch to your Arduino.

4. Python Script:
You can use Python (*IDLE* or *PyCharm*) to read the data from the Arduino via the *serial port*. You will need the *pyserial* library to establish communication. Install it if you haven't already:

```
pip install pyserial
```

Here's a *basic* Python script to read the potentiometer data:

```
import serial

# -- Replace 'COMx' with your Arduino's serial port
ser = serial.Serial('COMx', 9600)

try:
      while True:
            pot_value = ser.readline().decode().strip()
            print("Potentiometer Value:", pot_value)
except KeyboardInterrupt:
      ser.close()
      print("Serial connection closed.")
```

5. Run the Python Script:
Run your Python script. It should display the potentiometer values as you turn the potentiometer knob. Make sure that you've selected the correct baud rate in both the Arduino code and the Python script. If your Arduino uses a different baud rate, update it in both places.

This setup allows you to transmit the potentiometer readings from your Arduino to your Python script. You can then process or *visualize* this data in your Python application as needed.

Task 1

Required Hardware:

- Arduino board
- Potentiometer

- LED
- 220 Ω resistor
- Jumper wires
- Breadboard

Connections:

- See **Fig. 1**

Experiment Steps:

1. Connect the Arduino to your computer via a USB cable.
2. Upload the sketch to your Arduino using the Arduino IDE
3. Run the Python script on your computer.
4. As you turn the potentiometer knob, you should see the potentiometer readings displayed in the Python terminal.
5. You can use these readings for various experiments, data logging, or control applications, depending on your project requirements.

   Note on the use of Arduino *Serial Plotter*:
   If you are working with Python to read data from your Arduino, it's important NOT to open the Arduino Serial Plotter simultaneously. The Arduino *Serial Plotter* is a dedicated tool for visualizing data received from the Arduino board and may interfere with Python's access to the serial port. If you intend to use Python to read and process data from the Arduino, ensure that the Serial Plotter is closed or NOT in use while running your Python script to maintain uninterrupted communication between Python and the Arduino.

6. Open the *Serial Plotter*: In the Arduino IDE, go to **`Tools -> Serial Plotter`**.
7. Select the Correct COM Port: In the *Serial Plotter*, select the correct COM port to which your Arduino is connected.
8. Set the Baud Rate: Ensure that the baud rate in the *Serial Plotter* matches the one set in your Arduino code (e.g., 9600).
9. Read Real-Time Data: As you turn the potentiometer knob, the *Serial Plotter* will display the potentiometer readings in real-time, creating a *graphical* representation of the data. You can see how the values change as you adjust the potentiometer.
10. Customize the Plotter: You can customize the Serial Plotter by adjusting settings such as the graph range, labels, and colors.
11. **Extend** the circuit by adding an LED in series with a 220 Ω resistor. **Update** your Python code so that the LED turns ON automatically when the potentiometer value exceeds half of its maximum range, and turns OFF otherwise.

Experiment 2: Controlling Servo Motor via Serial Communication

Objective:

- To control servo motor angles using Python input transmitted to Arduino.
- Visualize sensor data graphically on the computer.

Required Hardware:

- Arduino Board + USB cable
- Servo Motor
- Jumper Wires
- Potentiometer

An example of the hardware setup is shown in **Fig. 2** (Signal → pin **9**, Vcc → 5V, GND → GND)

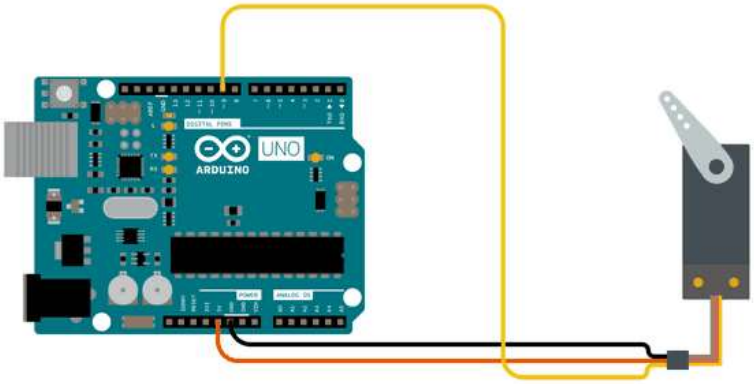

Fig. 2

Install Servo Libraries:

- Open the Arduino IDE.
- Go to **`Sketch > Include Library > Servo`** to install the `Servo` library.

Write the Arduino Code:

- Open a new sketch in the Arduino IDE.
- Write the Arduino code that reads angle data from the serial port and moves the servo accordingly.

```
#include <Servo.h>

Servo myservo;                  // to control a servo
int pos = 0;                    // variable to store the servo position

void setup() {
      myservo.attach(9);        // attaches the servo on pin 9
}

void loop() {
      for (pos = 0; pos <= 180; pos += 1) {
                                // goes from 0 to 180 degrees
            myservo.write(pos);
            delay(15);          // waits 15ms for the servo
```

```
                              // to reach the position
        }
        for (pos = 180; pos >= 0; pos -= 1) {
                              // goes from 180 to 0 degrees
            myservo.write(pos);
                              // tell servo to go to position in
                              // variable 'pos'
            delay(15);        // waits 15ms for the servo
                              // to reach the position
        }
    }
```

- Upload the code to your Arduino board

Install Required Python Libraries:

- Open a terminal or command prompt.
- Install the pyserial library using *pip*, if you haven't it:

```
pip install pyserial
```

Write the Python Script:

- Create a new Python script using a code editor (IDLE, PyCharm, or a text editor).

```
import serial

"""
Define the serial port and baud rate
(adjust the port COMx as per your Arduino)
"""
ser = serial.Serial('COMx', 9600)

try:
    while True:
       angle = input("Enter servo angle (0-180 deg, or 'q' to quit): ")
        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.")
```

Run the Python Script:

- Save the above Python script with a `.py` extension.
- Run the Python script, which will prompt you to enter the servo angle. You can enter an angle between 0 and 180 degrees to control the servo.

Exit the Python Script:

- When you're done with the experiment, enter 'q' to exit the Python script

Task 2

1. 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.

2. Enhance the experiment by adding an LED that lights up based on the potentiometer value or servo position. This adds a further layer of control and feedback.

3. Integrate *graphical visualization* using `matplotlib` to plot potentiometer values in real-time. **Modify** the following code to suit your specific requirements:

```
import serial
import matplotlib.pyplot as plt

ser = serial.Serial('COMx', 9600)
plt.ion()                       # Interactive mode for real-time plot
fig, ax = plt.subplots()
x_vals, y_vals = [], []

try:
    while True:
        pot_value = ser.readline().decode().strip()
        print("Potentiometer Value:", pot_value)
        x_vals.append(len(x_vals))
        y_vals.append(int(pot_value))

        ax.clear()
        ax.plot(x_vals, y_vals)  # Plot potentiometer readings
        plt.pause(0.1)          # Update plot

except KeyboardInterrupt:
    ser.close()
    plt.ioff()                  # Turn off interactive mode
    plt.show()                  # Final plot display
    print("Serial connection closed.")
```

References

[1] https://lastminuteengineers.com/servo-motor-arduino-tutorial/
How Servo Motor Works & Interface It With Arduino

[2] https://docs.arduino.cc/tutorials/
Tutorials - From beginner to Advanced

[3] https://www.instructables.com/Arduino-Servo-Motors/
Arduino Servo Motors