Mechatronics System Integration (MCTA3203)

Week **3a**: Parallel, Serial and USB interfacing with microcontroller and computer based system (1): Sensors and actuators.

Serial communication between Python and an Arduino is a common way to exchange data between a computer and a microcontroller. To establish serial communication, you'll need to write code on both the Python side and the Arduino side. In this experiment, we want to send potentiometer readings from an Arduino to a Python script through a USB connection. To achieve this, you can proceed with the following steps:

1. Hardware Setup:

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).

2. Arduino Code:

Write an Arduino 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 sketch to your Arduino.

3. Python Script:

You can use Python 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.")
```

4. Run the Python Script:

Run your Python script, and 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.

THE TASK

Materials Needed:

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

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, such as A0. An example of the circuit setup is shown in **Fig. 1**.

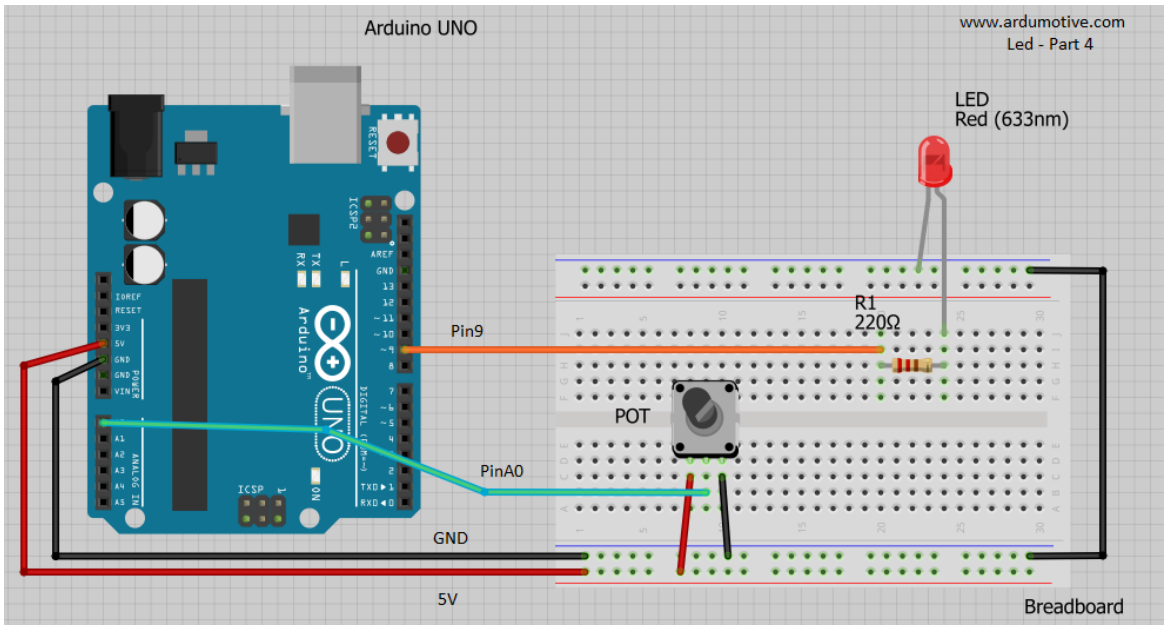

Fig. 1

Experiment Steps:

1. Connect the Arduino to your computer via a USB cable.
2. Power on the Arduino (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.

Using the Arduino Serial Plotter (Please note that 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.

Questions:

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).