1. Introduction

- **Objective**: Enable communication between a Python script and an Arduino board.
- **Applications**: Home automation, sensor data logging, robotics, etc.

---

2. Basics of Serial Communication

2.1 What is Serial Communication?

- Transfer of data one bit at a time over a communication channel.
- Common in microcontrollers due to simplicity and low resource usage.

Serial communication refers to the process of transmitting data in a sequential manner, one bit at a time, over a single wire or channel. It is an important aspect of embedded systems, allowing for the exchange of information between different devices.

There are two main types of serial communication: synchronous, where data is transmitted in sync with a clock signal, and asynchronous, where data is transmitted without a clock signal.

Various standards and protocols exist for serial communication, and it is crucial to select the appropriate protocol for each application.

Types of Network Protocols and Their Uses | GeeksforGeeks

2.2 UART Protocol

- **UART (Universal Asynchronous Receiver/Transmitter)**:
    - Asynchronous: no shared clock; uses start and stop bits.
    - Full duplex communication.
    - Data format: 8-bit data, 1 start bit, 1 stop bit, optional parity.
    - Baud rate (bits per second): Must match on both ends (e.g., 9600, 115200).

Universal Asynchronous Receiver Transmitter (UART) Protocol | GeeksforGeeks

3. Arduino Side: Serial Communication

Serial Communication between Python and Arduino

3.1 Serial Object in Arduino

cpp

CODE :

```
void setup() {
  Serial.begin(9600); // Initialize serial communication
}

void loop() {
  Serial.println("Hello from Arduino!");
  delay(1000);
}
```

- `Serial.begin(baud_rate)` initializes communication.
- `Serial.print()` and `Serial.println()` send data over USB.

3.2 Reading from Serial

cpp

```
if (Serial.available() > 0) {
  char received = Serial.read();
  // Process received data
}
```

---

4. Python Side: PySerial

4.1 Introduction to PySerial

- Python library to access serial ports.
- Install via pip:

bash

```
pip install pyserial
```

4.2 Connecting to Arduino

python

```
import serial
arduino = serial.Serial(port='COM3', baudrate=9600, timeout=1)
```

- `COM3` for Windows; use `/dev/ttyUSB0` or `/dev/ttyACM0` on Linux.
- `baudrate` must match Arduino.
- `timeout` controls blocking behavior.

4.3 Reading Data

python

```
data = arduino.readline().decode('utf-8').strip()
print("Received:", data)
```

4.4 Writing Data

python

```
arduino.write(b'1')  # Send a byte
```

When you read data from the Arduino using `serial.readline()`, it returns the data as **bytes**, not a normal human-readable string.

Example:

python

```
data = arduino.readline()
print(data)
```

Might output:

bash

```
b'Hello from Arduino!\r\n'
```

This is a **bytes object**, indicated by the `b` prefix.

What `.decode('utf-8')` Does:

python

```
decoded_data = data.decode('utf-8')
```

This converts the bytes into a string:

bash

```
'Hello from Arduino!\r\n'
```

UTF-8 is the encoding used to map the bytes to characters. It supports all Unicode characters and is the standard encoding for text files and web data.

5. Example: LED Control via Python

Arduino Code

cpp

```
void setup() {
  pinMode(13, OUTPUT);
  Serial.begin(9600);
}

void loop() {
  if (Serial.available() > 0) {
    char command = Serial.read();
    if (command == '1') digitalWrite(13, HIGH);
    else if (command == '0') digitalWrite(13, LOW);
  }
}
```

Python Code

python

```
import serial
import time

arduino = serial.Serial('COM3', 9600)
time.sleep(2)  # Allow time for Arduino reset

arduino.write(b'1')  # Turn LED on
time.sleep(1)
arduino.write(b'0')  # Turn LED off
```

PROBLEM STATEMENT (yayyyyy)

Deadline : 10:15 pm

You have an 8 bit display. Your python program should be receiving a number 0-9 from the user and displaying that number on the 8 bit using the microcontroller. Essentially your python terminal will receive the number, send it to the arduino interface, and then your arduino interface will use the number to make the display glow :D

Some warnings :

Check the correct port and matching baud rate.

Running both programs at the same time may show issues with the port terminal being occupied. Ask GPT Baba for the correct order.

Both VS Code and Serial Monitor cannot access the same port at the same time

CODE :

Python -

import serial

import time

```
# Replace 'COM3' with your Arduino port ('/dev/ttyUSB0' or '/dev/ttyACM0' on Linux/macOS)
arduino = serial.Serial('COM3', 9600, timeout=1)
time.sleep(2)  # Give Arduino time to reset
def send_digit(n):
    if 0 <= n <= 9:
        arduino.write(str(n).encode())
        print(f"Sent: {n}")
    else:
        print("Only digits 0–9 are allowed")
while(1):
    ch=int(input("Enter the digit"))
    send_digit(ch)
```

Arduino -

```
const int segmentPins[7] = {2, 3, 4, 5, 6, 7, 8};
// Segment patterns for digits 0-9
const byte digits[10][7] = {
  {1,1,1,1,1,1,0}, // 0
  {0,1,1,0,0,0,0}, // 1
  {1,1,0,1,1,0,1}, // 2
  {1,1,1,1,0,0,1}, // 3
  {0,1,1,0,0,1,1}, // 4
  {1,0,1,1,0,1,1}, // 5
  {1,0,1,1,1,1,1}, // 6
  {1,1,1,0,0,0,0}, // 7
  {1,1,1,1,1,1,1}, // 8
  {1,1,1,1,0,1,1}  // 9
```

```
};
void setup() {
  for (int i = 0; i < 7; i++) {
    pinMode(segmentPins[i], OUTPUT);
  }
  Serial.begin(9600);
}
void displayDigit(int digit) {
  if (digit < 0 || digit > 9) return;
  for (int i = 0; i < 7; i++) {
    digitalWrite(segmentPins[i], digits[digit][i]);
  }
}
void loop() {
  if (Serial.available() > 0) {
    char ch = Serial.read();
    Serial.println(ch);
    if (ch >= '0' && ch <= '9') {
      displayDigit(ch - '0');
    }
  }
}
```