IoT For Healthcare Monitoring

Aim

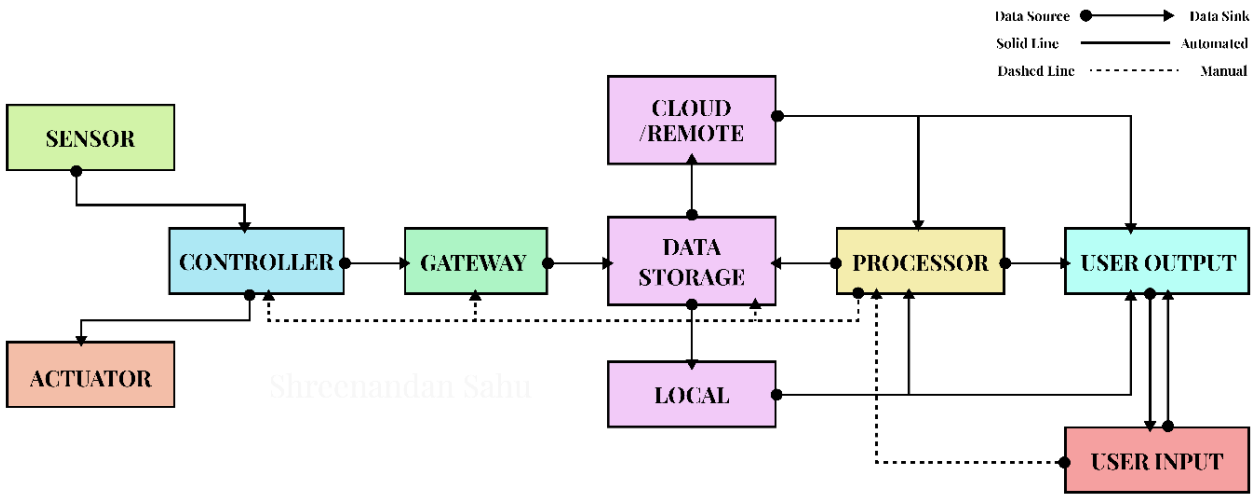
To Develop IoT system for monitoring Healthcare using ARDUINO compatible board with temperature sensor (TMP36) and Bluetooth module (HC05).

Apparatus/Software/Module Required

- Arduino compatible microcontroller
- HC05 Bluetooth Module
- TMP36 IC
- USB 2 wire
- Laptop/PC/Raspberry Pi
- Arduino IDE
- Python
- Pandas Module
- NumPy Module
- Serial Module
- Time Module
- Jumper Wires
- Bread Board

Theory

Internet of Things (IoT): - IoT It refers to a network of **interconnected physical objects** or "things" embedded with sensors, software, and other technologies that enable them to **collect, exchange, and transmit data over the internet**. (*Internet here always **does not mean** the web it simply means the **inter**connected **net**work of things.)* The primary goal of IoT is to enable these objects to communicate, interact, and provide value by sharing information and performing tasks autonomously or with minimal human intervention.



- **Sensor:** Sensors are devices that gather data from the physical world. They detect changes in their environment and convert these changes into electrical signals. *Eg. Temperature sensors, motion detectors, heart rate monitors.*
- **Actuator:** Actuators are devices that take signals from the IoT system and perform physical actions in the real world based on those signals. *Eg. Motors, servos, solenoids, and relays.*

- **Controller:** Controllers, often microcontrollers or microprocessors, are the "brains" of IoT devices. They process data, execute algorithms, and control the behaviour of the device. *Eg. Arduino boards, Raspberry Pi, ESP8266.*
- **Communication Gateway:** Gateways facilitate communication between IoT devices and the internet or other devices. They often bridge different communication protocols. *Eg. IoT routers, Serial COM, Bluetooth.*
- **Data Storage:** Data storage components store the data collected by sensors for later analysis or retrieval. This data can be stored locally or in the cloud. *Eg. Local databases, cloud storage services.*
- **Processor:** Processors analyse the data collected by sensors, making sense of it and often making decisions or predictions based on algorithms. *Eg. CPUs, GPUs, specialized AI processors.*
- **User Input:** User input refers to data or commands provided by users to the IoT system. This input can configure settings or trigger specific actions. *Eg. Buttons, touchscreens, voice commands.*
- **User Output:** User output components provide information or feedback from the IoT system to users. This can include notifications, displays, or auditory signals. *Eg. LEDs, displays, speakers.*

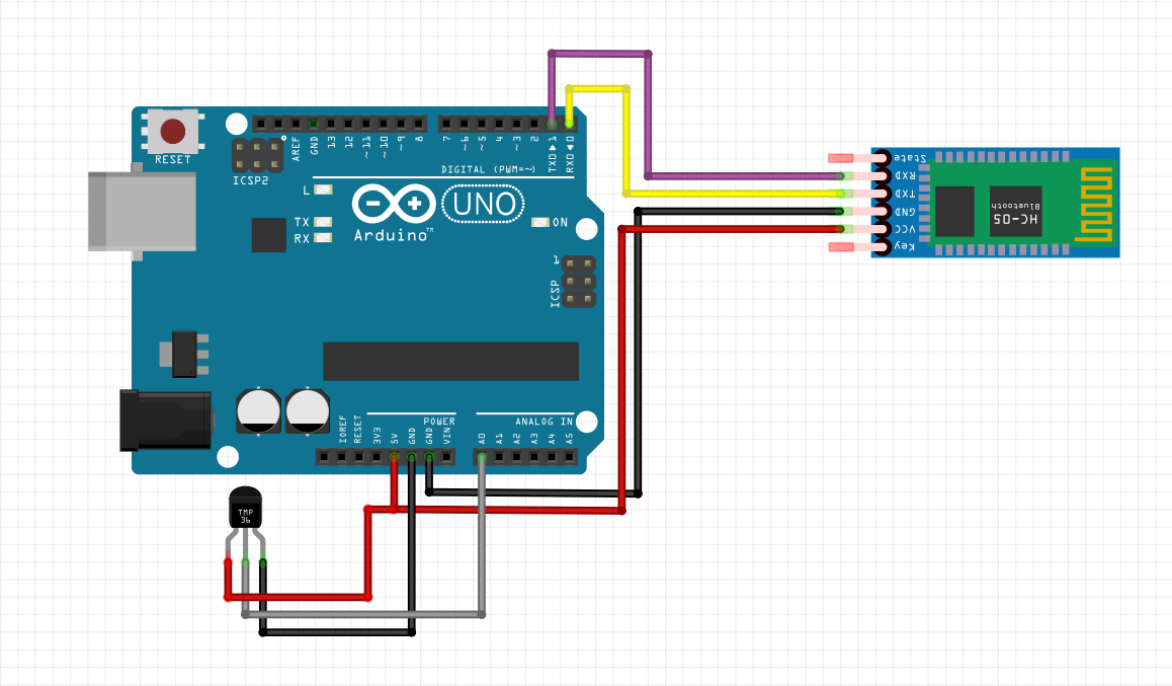
Bluetooth: Bluetooth is a **wireless communication** technology that allows electronic devices to exchange data **over short distances**. It was developed to replace traditional wired connections, such as USB cables, for connecting devices like smartphones, headphones, keyboards, and more. Bluetooth technology **uses radio waves in the 2.4 GHz ISM** (Industrial, Scientific, and Medical) band to establish connections between devices. *It provides a communication protocol to commute between the controller and processer.*

Arduino: Arduino is **an open-source** hardware and software platform designed for building and prototyping a wide range of electronic projects. It provides a simple and accessible way for individuals, including hobbyists, students, and professionals, to create interactive and programmable electronic devices. The core components of the Arduino platform include **Arduino Boards, Arduino Integrated Development Environment (IDE)** etc. *It is a controller which reads data from the sensor and coverts it to digital data and sends it to the processing unit.*

Procedure

1. Connect the TMP36 to the Arduino as shown below and connect the v_{out} Pin of the IC to the A0 pin.
2. Connect the Bluetooth Module HC-05 to the Arduino as shown in the diagram below make sure the TX and RX pin of module is connected to RX and TX pin of Arduino respectively.
3. Power up the Arduino using the USB 2.0 cable and use laptop/PC/raspberry pi as per your convenience to first code the Arduino compatible board and use it as a storage, processing, and visualising device.
4. Now open the Arduino IDE and code for obtaining the data from the sensor.
5. Now we will use the Serial COM port to push the data from the microcontroller to the laptop/pc/raspberry pi.
6. Here we have 2 option to send data to the processor *1. Wired USB 2. Wireless Bluetooth*.
7. Now open Code Editor and write a python code to obtain data from the serial port and display on the CLI (command line interface) and then store as a CSV.
8. Now to visualise the data we will use google spreadsheet or MS Excel.

9. There can be variations in the processes from her depending on showing or storing the data and usage of the available communication medium.
10. Once the code is written on the Arduino compatible board, we can unplug the board from processing device and use other power source to run the board in case of Bluetooth mode of communication.



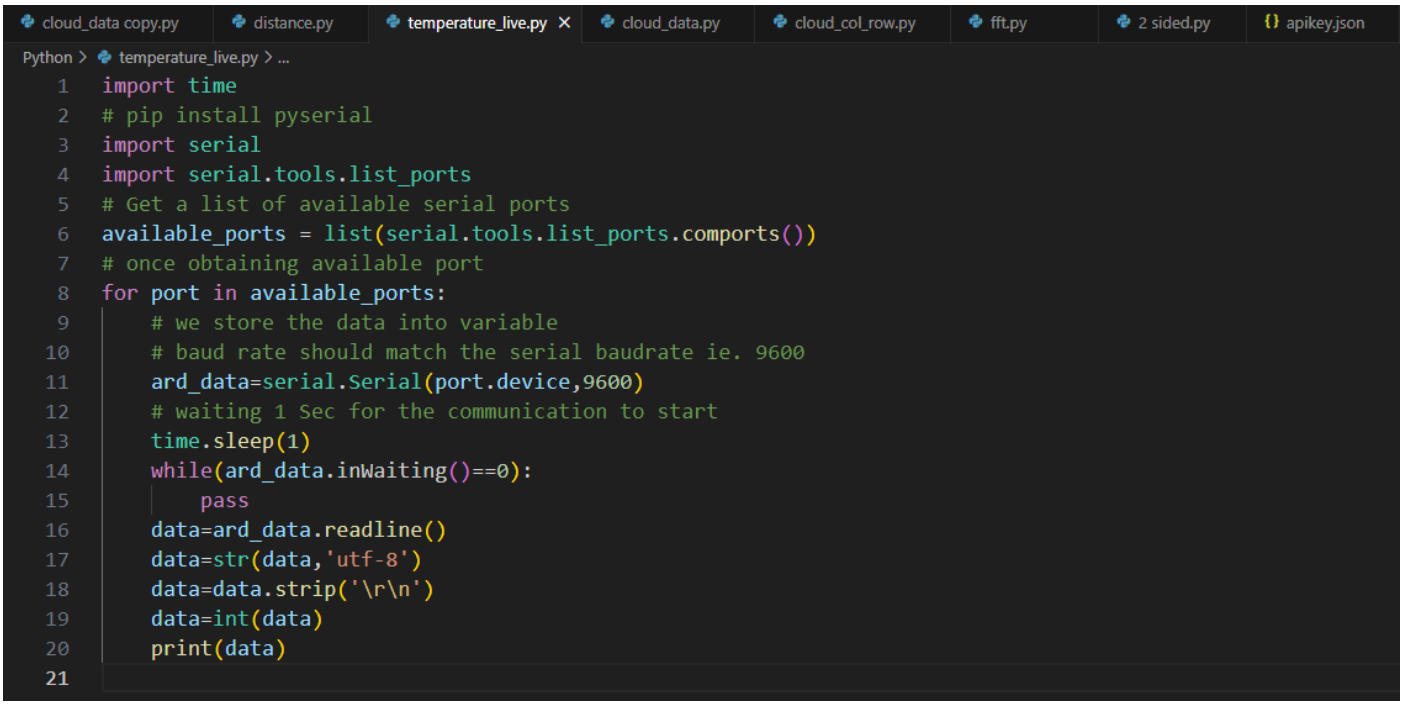
Schematic showing the component connection with the arduino compatible board.

Code

- **Arduino Code**

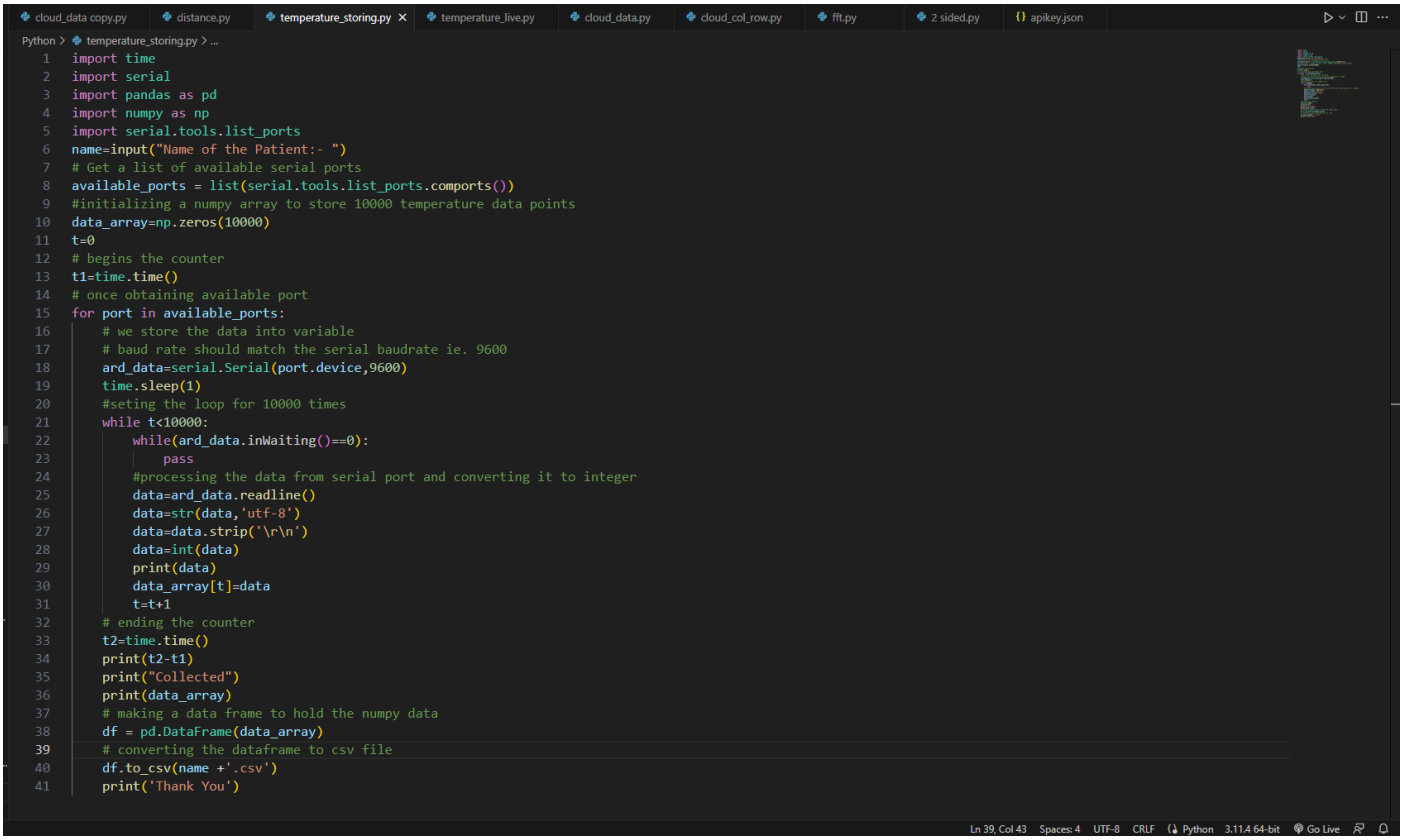
```
// Define the analog pin, the TMP36's Vout pin is connected to
#define sensorPin A0

void setup() {
  // Begin serial communication at 9600 baud rate
  Serial.begin(9600);
}

void loop() {
  // Get the voltage reading from the TMP36
  int reading = analogRead(sensorPin);

  // Convert that reading into voltage
  float voltage = reading * (5.0 / 1024.0);

  // Convert the voltage into the temperature in Celsius
  float temperatureC = voltage * 100;

  // Print the temperature in Celsius
  Serial.println(temperatureC);
  delay(10); // wait a 10 miliseconds between readings
}
```

- **Python Code**
  - **Live Temperature**



```python
import time
# pip install pyserial
import serial
import serial.tools.list_ports
# Get a list of available serial ports
available_ports = list(serial.tools.list_ports.comports())
# once obtaining available port
for port in available_ports:
    # we store the data into variable
    # baud rate should match the serial baudrate ie. 9600
    ard_data=serial.Serial(port.device,9600)
    # waiting 1 Sec for the communication to start
    time.sleep(1)
    while(ard_data.inWaiting()==0):
        pass
    data=ard_data.readline()
    data=str(data,'utf-8')
    data=data.strip('\r\n')
    data=int(data)
    print(data)

```

  - **Storing Temperature**



```python
import time
import serial
import pandas as pd
import numpy as np
import serial.tools.list_ports
name=input("Name of the Patient:- ")
# Get a list of available serial ports
available_ports = list(serial.tools.list_ports.comports())
#initializing a numpy array to store 10000 temperature data points
data_array=np.zeros(10000)
t=0
# begins the counter
t1=time.time()
# once obtaining available port
for port in available_ports:
    # we store the data into variable
    # baud rate should match the serial baudrate ie. 9600
    ard_data=serial.Serial(port.device,9600)
    time.sleep(1)
    #seting the loop for 10000 times
    while t<10000:
        while(ard_data.inWaiting()==0):
            pass
        #processing the data from serial port and converting it to integer
        data=ard_data.readline()
        data=str(data,'utf-8')
        data=data.strip('\r\n')
        data=int(data)
        print(data)
        data_array[t]=data
        t=t+1
    # ending the counter
    t2=time.time()
    print(t2-t1)
    print("Collected")
    print(data_array)
    # making a data frame to hold the numpy data
    df = pd.DataFrame(data_array)
    # converting the dataframe to csv file
    df.to_csv(name +'.csv')
    print('Thank You')
```

Observation

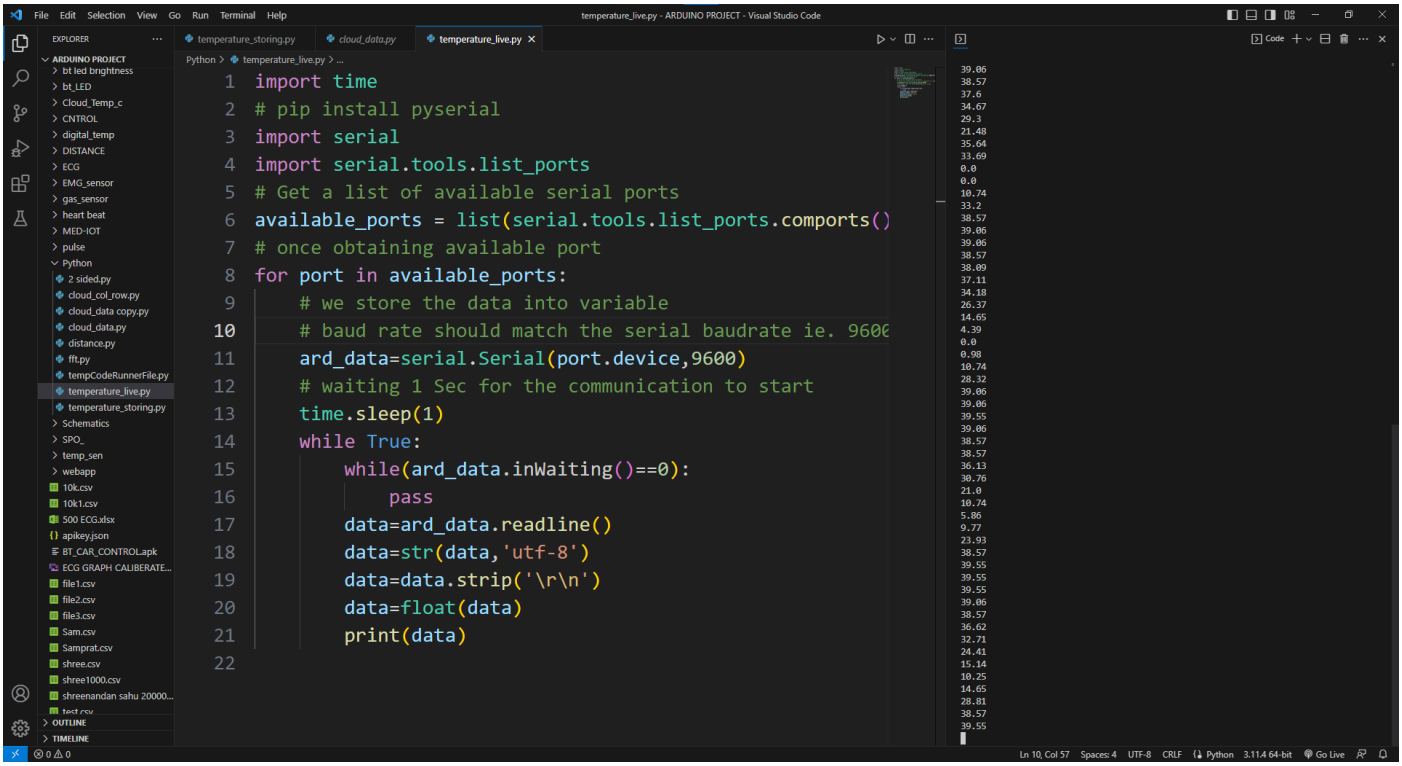
- Using the python code for LIVE data we obtained the live temperature of the patient.
- Using the python code for storing data we recorded the live temperature for 100 Seconds.

- Using the Same code, we can store the data of any desired amount of time using the simple calculation given below.
- $$datapoints = \frac{time\ in\ minutes * 60 * time\ delay\ in\ miliseconds}{1000\ miliseconds}$$

```
Cloud_Temp_c
// Define the analog pin, the TMP36's Vout pin is connected to
#define sensorPin A0

void setup() {
  // Begin serial communication at 9600 baud rate
  Serial.begin(9600);
}

void loop() {
  // Get the voltage reading from the TMP36
  int reading = analogRead(sensorPin);

  // Convert that reading into voltage
  float voltage = reading * (5.0 / 1024.0);

  // Convert the voltage into the temperature in Celsius
  float temperatureC = voltage * 100;

  // Print the temperature in Celsius
  Serial.println(temperatureC);
  delay(1000); // wait a 10 miliseconds between readings
}
```

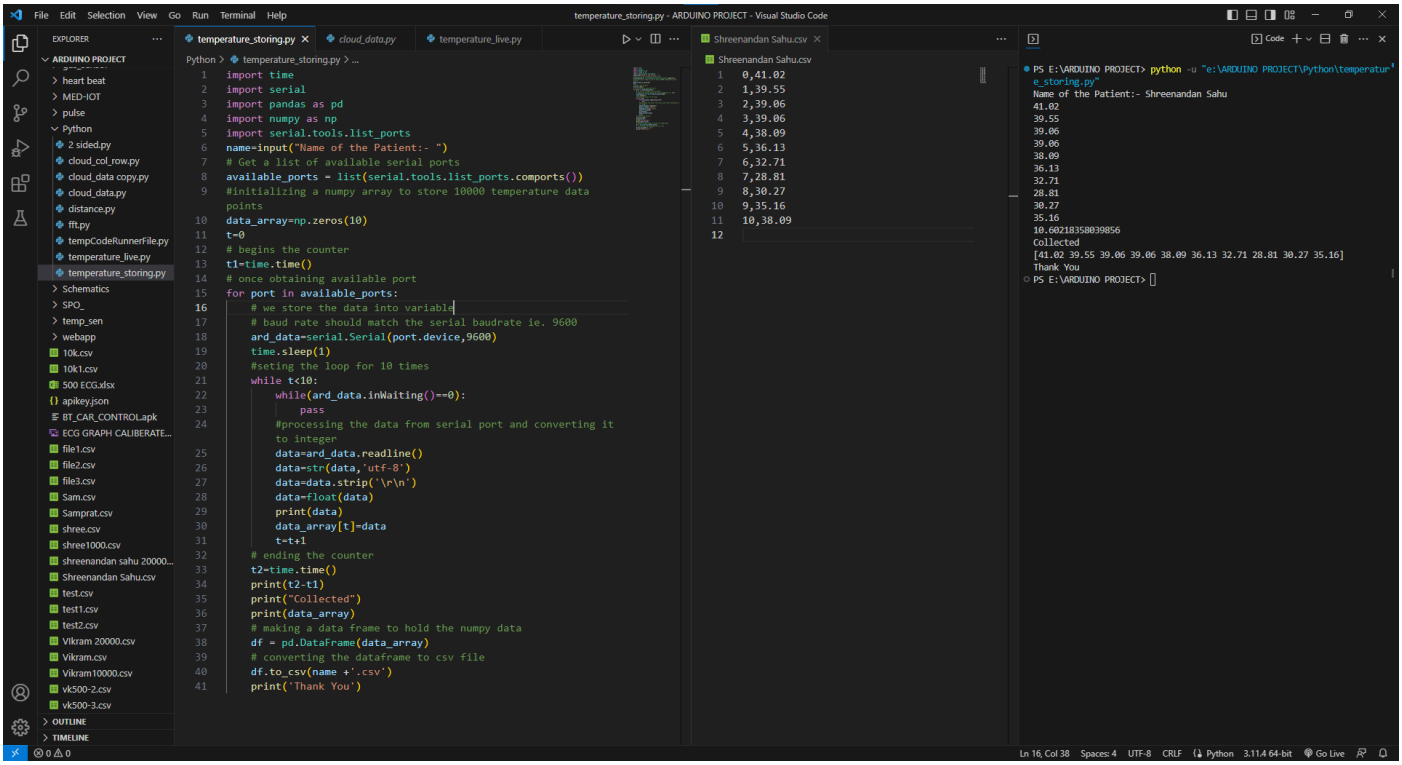
```
Done uploading.
Sketch uses 3152 bytes (9%) of program storage space. Maximum is 32256 bytes.
Global variables use 200 bytes (9%) of dynamic memory, leaving 1848 bytes for local variables. Maximum is 2048 bytes.
```

Arduino code being uploaded to the Arduino compatible board.

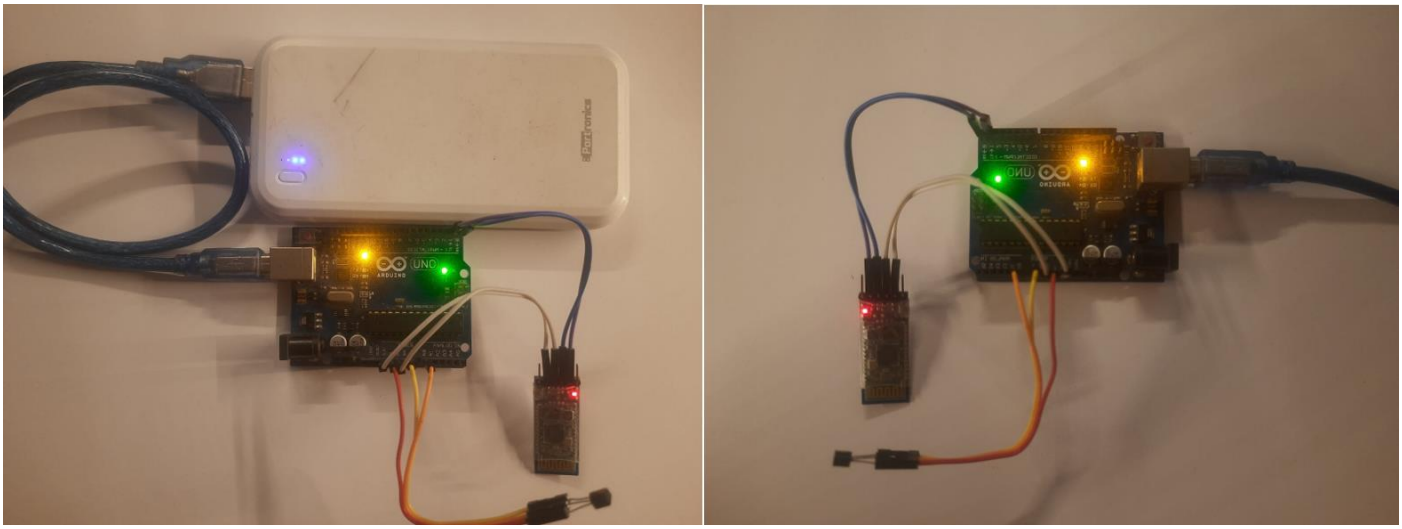


```
PS E:\ARDUINO PROJECT> python -u "e:\ARDUINO PROJECT\Python\temperature_live.py"
42.48
41.5
40.04
34.67
0.0
0.0
0.0
10.25
41.02
41.5
41.02
38.09
15.63
```

Running the live temperature code and obtaining the temperature in the Command line interface



Running the storing temperature code and obtaining the temperature in the CSV file format



Actual Demonstration of the practical

Conclusion

- After doing the experiment we can use sensor to obtain data and use microcontroller to send the data to the processing unit for storing, processing, and visualising the data.
- We can use different sensors to obtain various data regarding the patient in medical use of this.
- We can use different other modules of python to visualize the live data.

Discussion

- This simple demonstration of IoT in Healthcare monitoring can be extended to making apps and storing the data on cloud and processing it.
- In the further experiments we will make mobile apps and display and process data obtained in the cloud.

Shreenandan Sahu

120BM0806