

# IoT For Healthcare Monitoring

## Aim

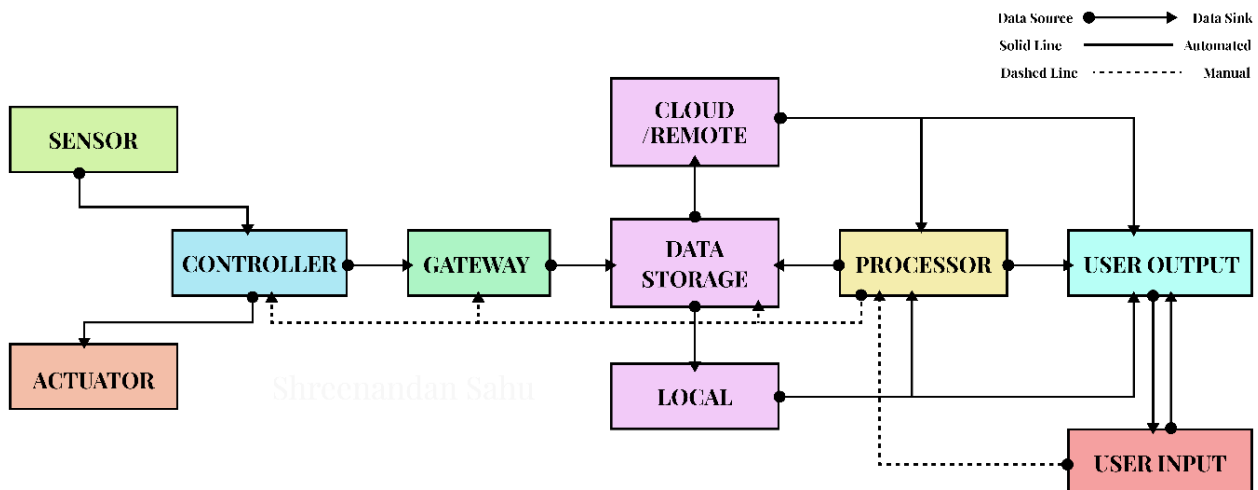
To Develop IoT system for monitoring Healthcare using ARDUINO compatible board with temperature sensor (TMP36) and Bluetooth module (HC05).

## 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 **interconnected network 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.*

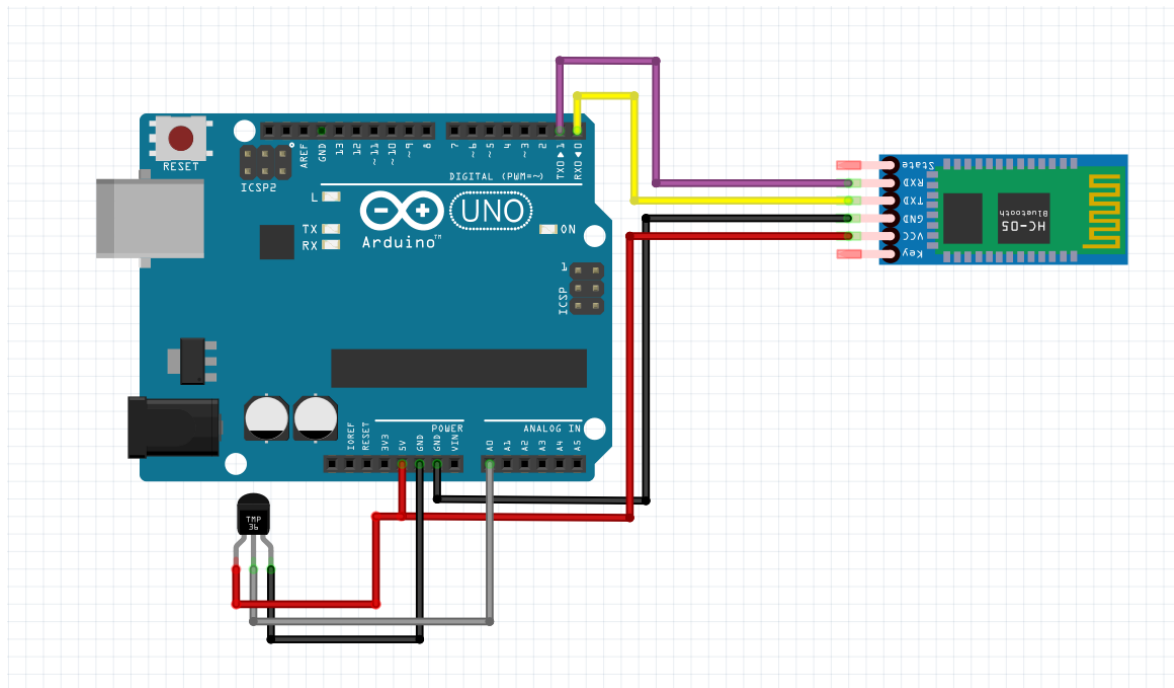
**Bluetooth:** Bluetooth is a **wireless communication** technology that allows electronic devices to exchange data **over short distances**. It was developed to replace traditional wired connections, such as USB cables, for connecting devices like smartphones, headphones, keyboards, and more. Bluetooth 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 processor.*

**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 converts 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**

```
Cloud_Temp_c | Arduino 1.8.19
File Edit Sketch Tools Help

Cloud_Temp_c.g
// 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 milliseconds between readings
}
```

- Python Code
  - Live Temperature

```

cloud_data copy.py distance.py temperature_live.py X cloud_data.py cloud_col_row.py fft.py 2 sided.py apikey.json
Python > temperature_live.py > ...
1  import time
2  # pip install pyserial
3  import serial
4  import serial.tools.list_ports
5  # Get a list of available serial ports
6  available_ports = list(serial.tools.list_ports.comports())
7  # once obtaining available port
8  for port in available_ports:
9      # we store the data into variable
10     # baud rate should match the serial baudrate ie. 9600
11     ard_data=serial.Serial(port.device,9600)
12     # waiting 1 Sec for the communication to start
13     time.sleep(1)
14     while(ard_data.inWaiting()==0):
15         pass
16     data=ard_data.readline()
17     data=str(data,'utf-8')
18     data=data.strip('\r\n')
19     data=int(data)
20     print(data)
21

```

- Storing Temperature

```

cloud_data copy.py distance.py temperature_storing.py X temperature_live.py cloud_data.py cloud_col_row.py fft.py 2 sided.py apikey.json
Python > temperature_storing.py > ...
1  import time
2  import serial
3  import pandas as pd
4  import numpy as np
5  import serial.tools.list_ports
6  name=input("Name of the Patient:- ")
7  # Get a list of available serial ports
8  available_ports = list(serial.tools.list_ports.comports())
9  #initializing a numpy array to store 10000 temperature data points
10 data_array=np.zeros(10000)
11 t=0
12 # begins the counter
13 t1=time.time()
14 # once obtaining available port
15 for port in available_ports:
16     # we store the data into variable
17     # baud rate should match the serial baudrate ie. 9600
18     ard_data=serial.Serial(port.device,9600)
19     time.sleep(1)
20     #setting the loop for 10000 times
21     while t<10000:
22         while(ard_data.inWaiting()==0):
23             pass
24         #processing the data from serial port and converting it to integer
25         data=ard_data.readline()
26         data=str(data,'utf-8')
27         data=data.strip('\r\n')
28         data=int(data)
29         print(data)
30         data_array[t]=data
31         t=t+1
32     # ending the counter
33     t2=time.time()
34     print(t2-t1)
35     print("Collected")
36     print(data_array)
37     # making a data frame to hold the numpy data
38     df = pd.DataFrame(data_array)
39     # converting the dataframe to csv file
40     df.to_csv(name + '.csv')
41     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.
- $$\text{datapoints} = \frac{\text{time in minutes} * 60 * \text{time delay in milliseconds}}{1000 \text{ milliseconds}}$$

```
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 milliseconds 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.*

```
temperature_live.py
1 import time
2 # pip install pyserial
3 import serial
4 import serial.tools.list_ports
5 # Get a list of available serial ports
6 available_ports = list(serial.tools.list_ports.comports())
7 # once obtaining available port
8 for port in available_ports:
9     # we store the data into variable
10    # baud rate should match the serial baudrate ie. 9600
11    ard_data=serial.Serial(port.device,9600)
12    # waiting 1 Sec for the communication to start
13    time.sleep(1)
14    while True:
15        while(ard_data.inWaiting()==0):
16            pass
17        data=ard_data.readline()
18        data=str(data,'utf-8')
19        data=data.strip('\r\n')
20        data=float(data)
21        print(data)
22
```

Serial Monitor Output:

```
39.06
38.57
37.6
34.67
29.3
21.48
35.64
33.69
0.0
0.0
10.74
33.2
38.57
39.06
39.06
38.57
38.09
37.11
34.18
26.37
14.65
4.29
0.0
0.98
10.74
28.32
39.06
39.06
38.57
38.57
36.13
38.76
21.0
10.74
5.86
9.77
23.93
38.57
39.55
39.55
39.55
39.06
38.57
36.62
32.71
24.41
15.14
10.25
14.65
28.81
38.57
39.55
```

```

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*

```

File Edit Selection View Go Run Terminal Help
temperature_storing.py - ARDUINO PROJECT - Visual Studio Code

EXPLORER
ARDUINO PROJECT
  heart beat
  MED-IOT
  pulse
  Python
    2 sided.py
    cloud_col_row.py
    cloud_data copy.py
    cloud_data.py
    distance.py
    fft.py
    tempCodeRunnerFile.py
    temperature_live.py
    temperature_storing.py
  Schematics
  SPO_
  temp_sen
  webbapp
  10k.csv
  10k1.csv
  500 ECG.xlsx
  () spikekey.json
  BT_CAR_CONTROLapk
  ECG GRAPH CALIBRATE...
  file1.csv
  file2.csv
  file3.csv
  Sam.csv
  Samprat.csv
  shree.csv
  shree1000.csv
  shreenandan sahu 20000...
  shreenandan Sahu.csv
  test1.csv
  test2.csv
  Vikram 20000.csv
  Vikram.csv
  Vikram10000.csv
  vks00-3.csv
  vks00-3.csv

> OUTLINE
> TIMELINE

Python > temperature_storing.py > ...
1 import time
2 import serial
3 import pandas as pd
4 import numpy as np
5 import serial.tools.list_ports
6 name=input("Name of the Patient:- ")
7 # Get a list of available serial ports
8 available_ports = list(serial.tools.list_ports.comports())
9 #initializing a numpy array to store 10000 temperature data
10 points
11 data_array=np.zeros(10)
12 t=0
13 # begins the counter
14 t1=time.time()
15 for port in available_ports:
16     # we store the data into variable
17     # baud rate should match the serial baudrate ie. 9600
18     ard_data=serial.Serial(port.device,9600)
19     time.sleep(1)
20     #setting the loop for 10 times
21     while t<10:
22         while(ard_data.inWaiting()==0):
23             pass
24         #processing the data from serial port and converting it
25         #to integer
26         data=ard_data.readline()
27         data=str(data,'utf-8')
28         data=data.strip('\r\n')
29         data=float(data)
30         print(data)
31         data_array[t]=data
32         t=t+1
33     # ending the counter
34     t2=time.time()
35     print(t2-t1)
36     print("Collected")
37     print(data_array)
38     # making a data frame to hold the numpy data
39     df = pd.DataFrame(data_array)
40     # converting the dataframe to csv file
41     df.to_csv(name+'.csv')
42     print("Thank You")

Shreenandan Sahu.csv
1 0,41.02
2 1,39.55
3 2,39.06
4 3,39.06
5 4,38.09
6 5,36.13
7 6,32.71
8 7,28.81
9 8,30.27
10 9,35.16
11 10,38.09
12

PS E:\ARDUINO PROJECT> python -u "e:\ARDUINO PROJECT\Python\temperatur
e_storing.py"
Name of the Patient:- Shreenandan Sahu
41.02
39.55
39.06
39.06
38.09
36.13
32.71
28.81
30.27
35.16
10,60218358039856
Collected
[41.02 39.55 39.06 39.06 38.09 36.13 32.71 28.81 30.27 35.16]
Thank You
PS E:\ARDUINO PROJECT>

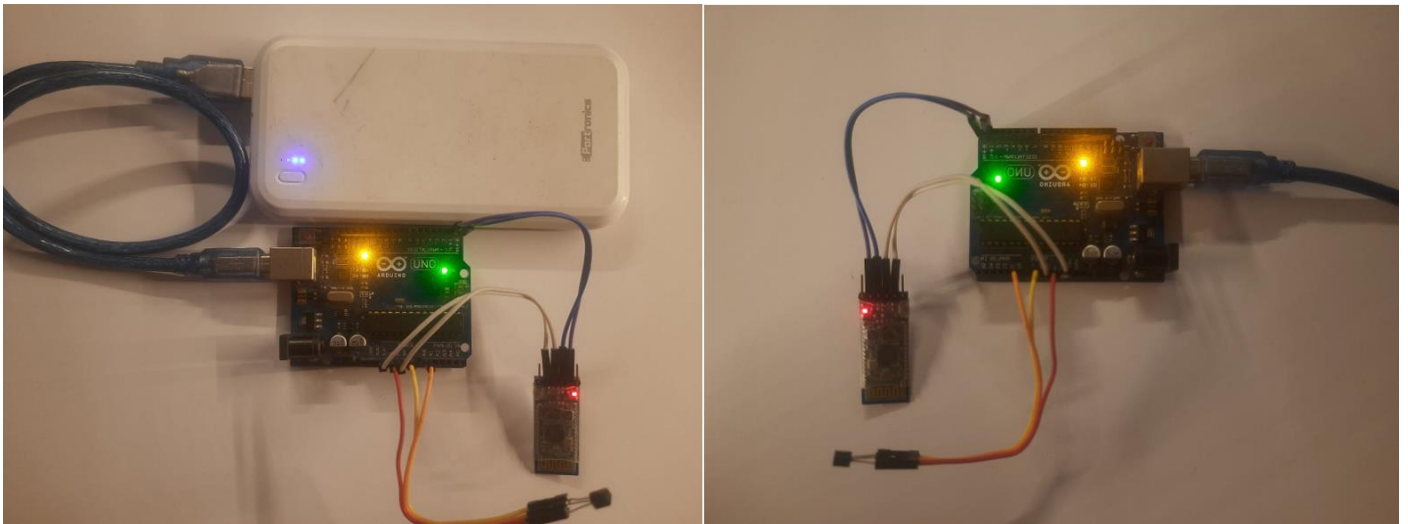
```

```

Shreenandan Sahu.csv
1 0,41.02
2 1,39.55
3 2,39.06
4 3,39.06
5 4,38.09
6 5,36.13
7 6,32.71
8 7,28.81
9 8,30.27
10 9,35.16
11 10,38.09
12

PS E:\ARDUINO PROJECT> python -u "e:\ARDUINO PROJECT\Python\temperatur
e_storing.py"
Name of the Patient:- Shreenandan Sahu
41.02
39.55
39.06
39.06
38.09
36.13
32.71
28.81
30.27
35.16
10,60218358039856
Collected
[41.02 39.55 39.06 39.06 38.09 36.13 32.71 28.81 30.27 35.16]
Thank You
PS E:\ARDUINO PROJECT>

```



*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