



slington college
(इस्लिङ्टन कलेज)

Module Code & Module Title
CS5053NI/CC5068NI– Cloud Computing & IoT

Remote Patient Health Monitoring System using ESP32

Assessment Type
10% Proposal Report

Semester
2023 Spring/Autumn

Group members

London Met ID	Student Name
22067813	Dibbeshwor Acharya
22068180	Kirtan Maharjan
22067827	Nitika Suwal
22067503	Prabesh Shrestha
21036954	Swyambhu Sthapit

Assignment Due Date: 12/01/2024
Assignment Submission Date: 12/01/2024
Submitted to: Mr. Sugat Man Shakya
Word Count: 3000

I confirm that I understand my coursework needs to be submitted online via Google Classroom under the relevant module page before the deadline in order for my assignment to be accepted and marked. I am fully aware that late submissions will be treated as non-submission and a mark of zero will be awarded.

Acknowledgement

We would like to wholeheartedly thank our module leader Mr. Sugat Man Shakya and our tutor Mr. Ayush Pradhanang, without whom this project would not have been possible. Without their constant teaching and dedication, we would not have been able to work on our Remote Patient Health Monitoring System and make this project successful. They have assisted us throughout the whole development process of this project, providing us with invaluable insights and knowledge. We would also like to thank Sishir sir and Islington College Resource Department for providing us with materials that were available. At last, a thank you to all our group members and friends for helping us.

Abstract

Constant monitoring of patient vitals is an essential pillar of modern healthcare. Hospitals are equipped with devices costing hundreds of thousands of rupees so that patients can be monitored constantly, making sure immediate action is taken whenever vitals stray from normal range. The main goal of this project is to create a remote patient health monitoring system using readily available materials at the lowest cost possible.

Being able to constantly monitor the state of patients at minimum cost can benefit a huge number of people throughout the world. This system will especially be useful to people living in poverty, who cannot afford the cost of modern healthcare. Institutions and organizations can use this system to advance their healthcare.

Table of Contents

1. Introduction	1
1.1. Current Scenario	1
1.2. Problem Statement and Project as a Solution.....	2
1.3. Aim and Objectives	2
2. Background.....	3
2.1. System Overview	3
2.2. Design Diagrams.....	4
2.3. Requirement Analysis	8
2.4. Classification of Sensors and Actuators of our project.	13
3. Development.....	14
3.1. Planning and Design	14
3.2. Resource Collection	14
3.3. System Development	15
4. Result and Findings	18
4.1. Testing	18
4.1.1. Check if the Local Webserver starts and can be accessed by devices.	18
4.1.2. Check if body temperature sensor works.....	20
4.1.3. Check if website shows alert if pulse is below 60.	21
4.1.4. Checking the LED and Website Alert System.....	23
5. Future Works.....	25
6. Conclusion	26
7. References.....	27
8. Appendix	29
8.1. Work Breakdown Structure	29
8.2. Issue with the MAX30100 Sensor	29

8.3. Video of the device working	30
8.4. Code	30

Table of Figures

Figure 1: Block Diagram	4
Figure 2: Circuit Diagram	5
Figure 3: Schematic Diagram	5
Figure 4: System Architecture	6
Figure 5: Flowchart.....	7
Figure 6: ESP32 (Amazon Store)	8
Figure 7: DHT11 Sensor (Cytron.io).....	8
Figure 8: MAX30100/102 (Breadfruit Electronics)	9
Figure 9: DS18B20 (Robu.in)	9
Figure 10: Breadboard (Ubuy).....	10
Figure 11: Jumper Wires (Probots)	10
Figure 12: Arduino IDE	11
Figure 13: Tinkercad	11
Figure 14: Circuitio.io	12
Figure 15: Draw.io	12
Figure 16: Luid Chart.....	12
Figure 17: Connecting Body Temperature Sensor	15
Figure 18: Connecting Pulse Oximeter.....	16
Figure 19: Connect Humidity Sensor.....	16
Figure 20: Uploading Code to ESP32	17
Figure 21: Getting the IP Address	19
Figure 22: Accessing the local webserver.	19
Figure 23: Sensor values before applying heat	20
Figure 24: Sensor Value After Applying Heat.	21
Figure 25: While Pulse Normal.....	22
Figure 26: Removing Finger from Pulse Sensor.	22
Figure 27: Pulse data with finger on the device.....	23
Figure 28: LED alert and Website Alert shown.....	24
Figure 29: WBS (Canva)	29

Table of Tables

Table 1: Sensors used classification.	13
Table 2: Test 1	18
Table 3: Test 2	20
Table 4: Test 3	21
Table 5: Test 4.	23

1. Introduction

Peter Morville, the president of Semantic Studios one said, *“We have a deep need and desire to connect. Everything in the history of communication technology suggests we will take advantage of every opportunity to connect more richly and deeply. I see no evidence for a reversal of that trend.”* IoT since its inception has been growing rapidly with no evidence of slowing down. All because it addresses the innate human desire to connect, our desire of information and knowledge.

IoT is a network of connected devices that can communicate with each other and to the cloud. The reduced costs of computing devices and rapid advancement of technology has catapulted the development of IoT, causing more than billion devices currently to be connected to the Internet of Things (Amazon Web Services, inc. , 2023).

Patient health monitoring system is a simple IoT device that monitors patient vitals and sends the data to a local web server that can be accessed by devices remotely without the need of any physical connection. It leverages the power of IoT to make patient health monitoring more accessible.

1.1. Current Scenario

Sick people need to have their vitals checked on a regular basis to ensure that immediate action can be taken if their vitals fluctuate from the normal range. Regular checking can be cumbersome, especially for patients with chronic illnesses or reduced mobility. Data shows that an estimated 65% of patients are spot checked by nurses every four to eight hours a day and are not monitored constantly. This number rises even more when we look at countries like Nepal where the healthcare system has still not caught up with the western world. Being able to automate this process can be a huge help to both healthcare institutions and patients alike (O'Malley, 2020).

1.2. Problem Statement and Project as a Solution

A study conducted by by Susan McGarth showed that out of 111,488 continuously monitored patients no one died while out of 15,209 unmonitored patients, 3 died. The lack of continuous health monitoring can prove to be fatal to patients (Sotera Digital Health, 2022). Even in the context of Nepal, overcrowding and lack of management is a major issue among hospitals especially at times of distress like the recent COVID-19 pandemic (The Kathmandu Post, 2023). The main barrier to proper health monitoring among patients especially in developing countries like Nepal is the cost of modern healthcare where most people pay for the cost out of their pocket (Ending Pandemics, 2023).

This project aims to make continuous healthcare monitoring system more accessible and reduce avoidable fatalities. Easily accessible devices like this can promote healthcare equity at large, preventing the need for increased manpower, cost, and infrastructure. Patients can be monitored remotely by professionals or their loved ones without the need for expensive devices. Low risk patients can be monitored directly from their home, reducing overcrowding at hospitals and financial burden to patients (Jessica, et al., 2024).

1.3. Aim and Objectives

The major aim of this project is to develop a remote patient health monitoring system that will measure the vitals (Temperature, SP02 and Pulse) of a patient and remotely display it through a local webserver that can be accessed by any device on the same network and will show alert if the health data is not normal.

The objectives are:

- Successfully connect all the sensors and devices to gather data from them.
- Display the gathered data in a meaningful way in the serial monitor.
- Deploy a local webserver using the node MCU.
- Send data gathered by the sensor to the web server.
- Make sure that devices in the local network can successfully connect to the web server and view the data.
- Show an alert on the website and blink LED if the health data is not normal.

2. Background

This system was developed after a careful discussion and research on the various domains of IoT. It has been designed to show how IoT can be practically implemented to make an impact on our daily lives. After careful consideration of all IoT domains and systems, we decided to create the patient health monitoring system because it seemed the most impactful considering the current scenario. The field of healthcare is an extremely delicate but important domain in our modern lives. Our system aims to make a positive impact on this field.

2.1. System Overview

This system uses ESP32 as its heart, this is where all the data processing and outputting happens. It contains three main sensors, body temperature sensor for gathering patient body temperature. DHT11 to gather the current environmental conditions and a pulse oximeter to gather the oxygen concentration and pulse of the patient. Together these three sensors provide the major patient vitals.

The basic working principle of this system is very simple. The temperature system, humidity sensor and pulse oximeter continuously gather data from their environment. This data is then sent to the ESP32 module, which translates it to our known standard values and displays it in the serial monitor. The ESP32 will have started a local webserver on startup, to which then the data from the sensors is sent in HTML format. Users wanting to know the data can enter the web-servers IP address and access the current patient health condition. It also has an alert system where the website shows an alert to the user if the vitals are not normal, or the patient needs any kind of attention. The built-in LED in the ESP32 Module also turns on if the pulse becomes abnormal.

We need to refresh the webserver to get the latest data as the website is completely static and data is sent only after the user has refreshed the website.

2.2. Design Diagrams

This portion contains all the necessary diagrams which will help in understanding the working system better.

1. Block Diagram:

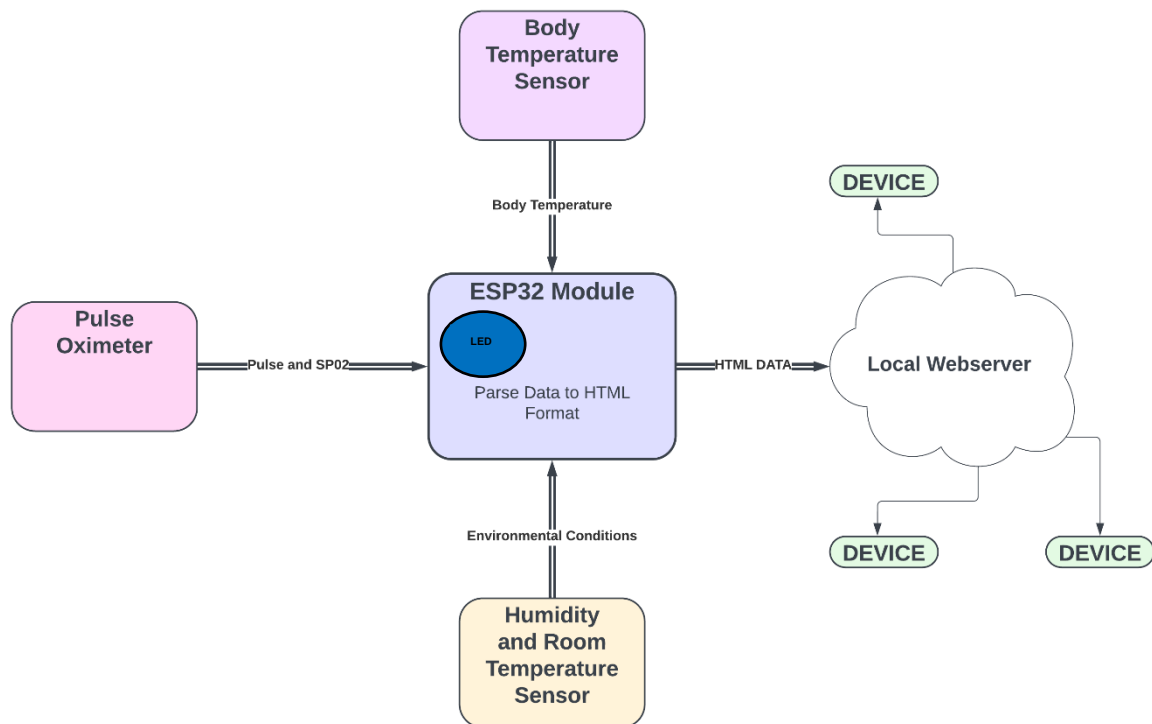
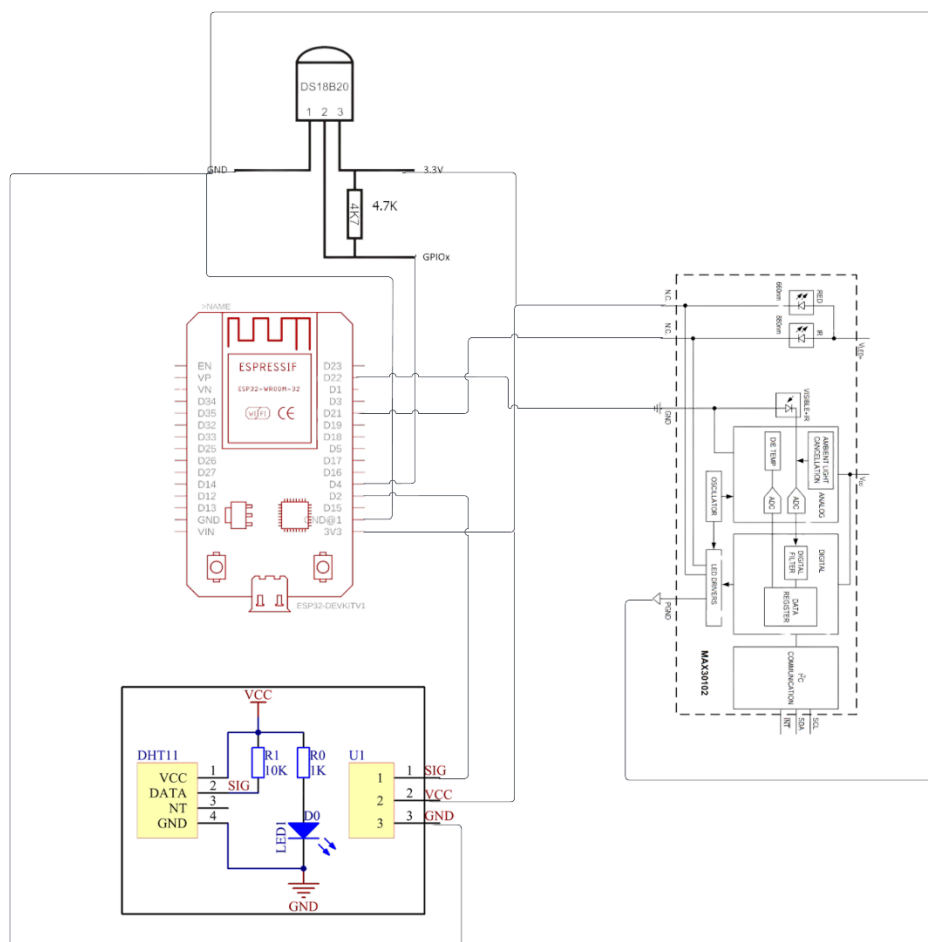
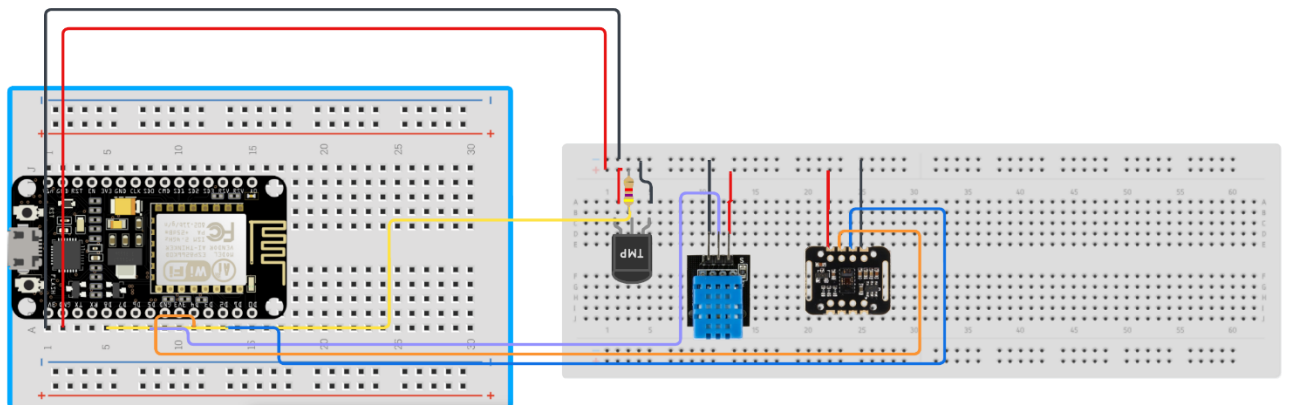


Figure 1: Block Diagram

The above block diagram shows the general arrangement of parts in the system. The ESP32 Module takes data from all the connected sensors and does the necessary processing according to the code. It then generates a local webserver that can be accessed by devices on the same network.



4. System Architecture

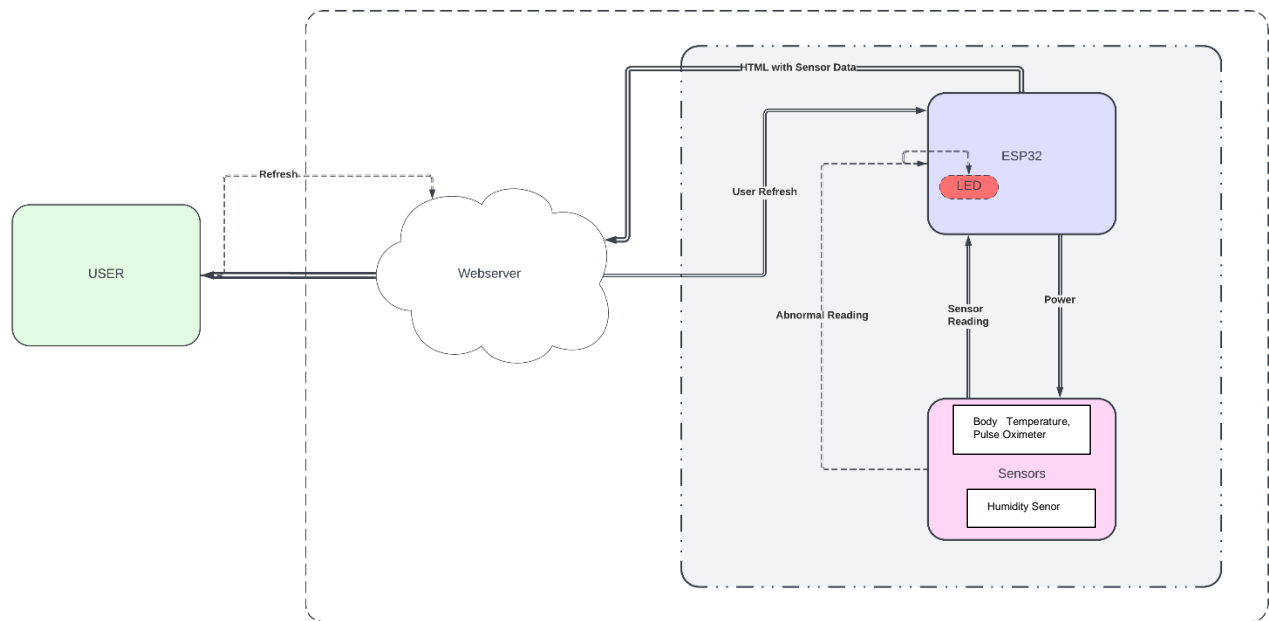


Figure 4: System Architecture

The above diagram shows the architecture of the patient health monitoring system. The innermost architecture shows the interaction between our microcontroller and the sensors. Then, the webserver is created by the ESP32 module which is then accessed by external users.

5. Flowchart

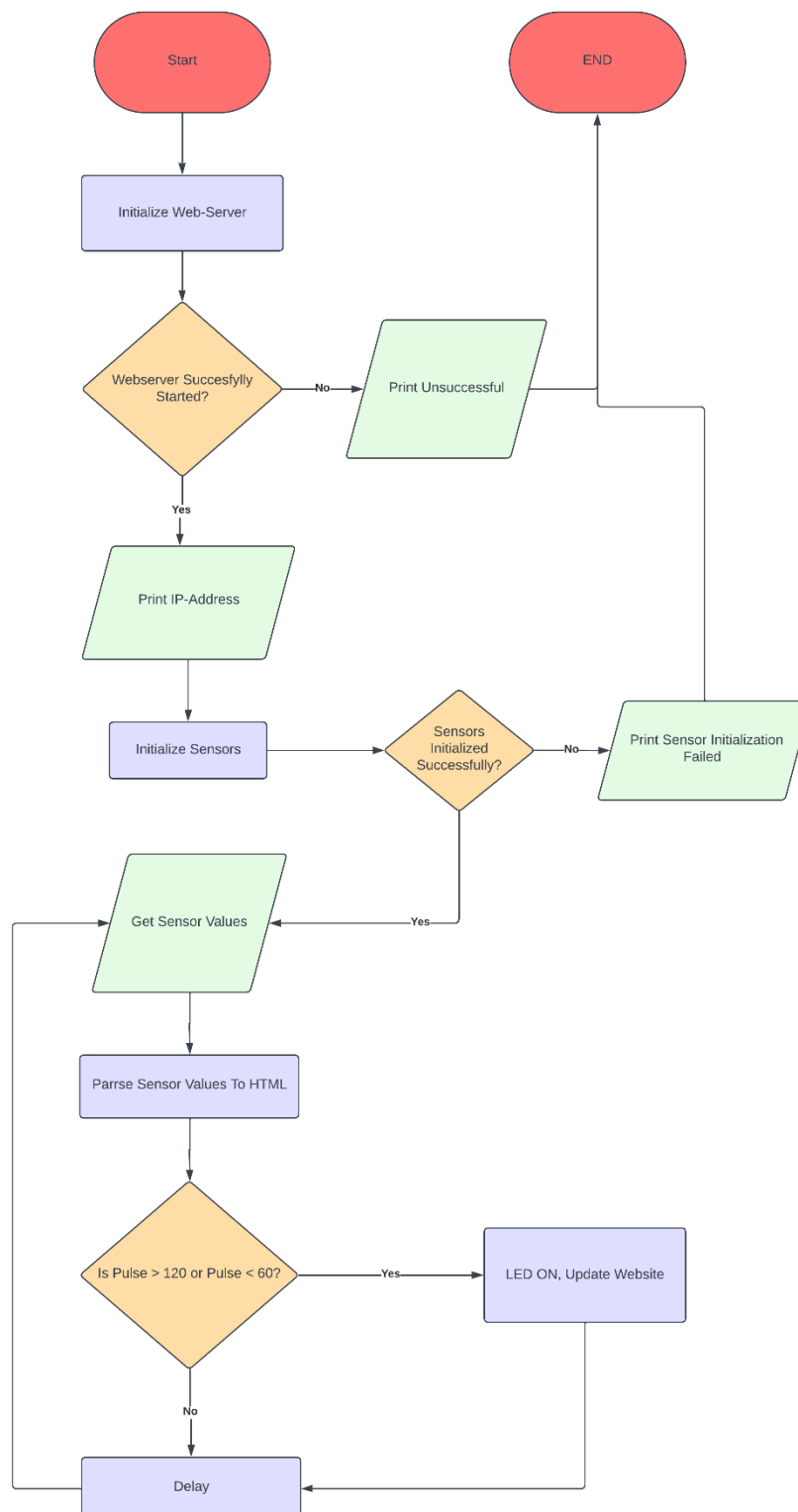


Figure 5: Flowchart

2.3. Requirement Analysis

We have made sure to use materials that are relatively easy to access and are not expensive to stay true to our goal of making healthcare more accessible to everyone. The basic requirements are very simple this system can be developed by anyone with knowledge on computing and IoT.

The hardware requirements of our system are as follows:

1. ESP32

ESP32 is a low-cost microcontroller designed by Espressif Systems. It has integrated Wi-Fi and even Bluetooth in some modules. The possibilities of systems that can be designed using this module are endless because of its ease of use (Espressif Systems (Shanghai) Co., Ltd., 2023).



Figure 6: ESP32 (Amazon Store)

2. DHT11 Sensor

The DHT11 Sensor is one of the most widely used and easy to integrate sensor used to gather the temperature and humidity data from the environment.

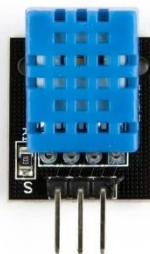


Figure 7: DHT11 Sensor (Cytron.io)

3. MAX30102/100 Sensor

The MAX30100/102 Sensors are integrated pulse and oxygen saturation sensors that combine two LEDs to measure the heart rate and oxygen saturation of patients. These sensors measure the reflection of light from our bloodstream to measure the relevant data (Maxim Integrated Products, inc., 2014).

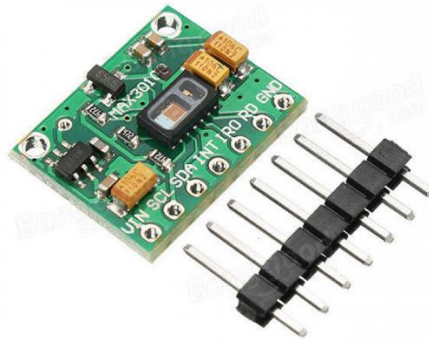


Figure 8: MAX30100/102 (Breadfruit Electronics)

4. DS18B20 Temperature Sensor

The DS18B20 is a digital thermometer that provides 9-to-12-bit Celsius temperature measurements. It does not require any analog to digital conversion and works using the one wire digital bus protocol (Arduino.cc, 2024).



Figure 9: DS18B20 (Robu.in)

5. Breadboard

A breadboard is an essential tool used in prototyping and designing to assemble various circuits and connect components without the need of soldering. It helps us test and prototype the system to find any errors and fix them before permanently assembling them (Circuito.io, 2018).

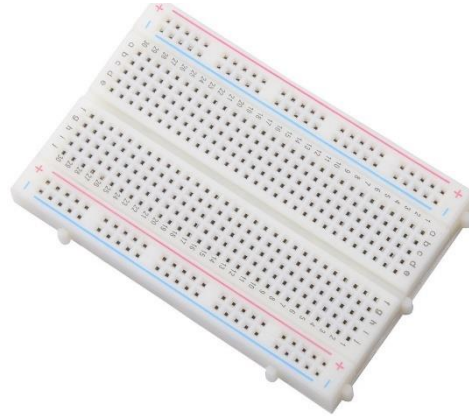


Figure 10: Breadboard (Ubuy)

6. Jumper Wires

Jumper wires are an essential tool in prototyping. They are small wires with male and female connections. They are used to connect components and create prototype without soldering (Hemmings, 2016).



Figure 11: Jumper Wires (Probots)

The software Requirements of our project are:

1. Arduino IDE

The Arduino IDE was used to write and upload all the code to the ESP32 module. It was also used to import the required libraries and dependencies to the module. The serial monitor of Arduino IDE was used to get the real time sensor values and debug any issues.

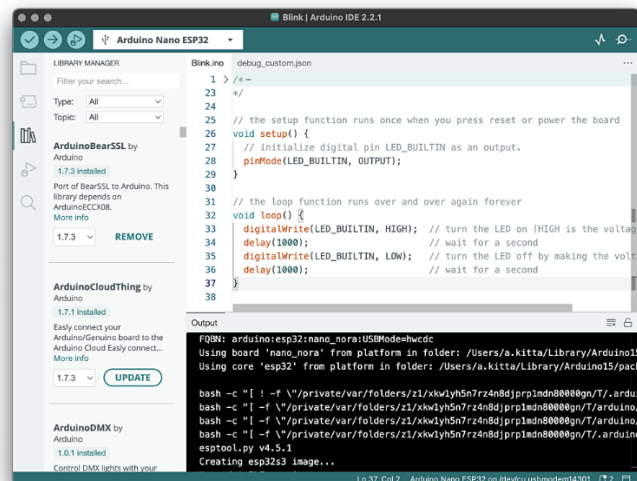


Figure 12: Arduino IDE

2. Tinkercad

Tinkercad is a free 3D Modelling and circuit designing software that runs directly in the browser. We used Tinkercad to design parts of our prototype as the ESP32 module is not available in Tinkercad.



Figure 13: Tinkercad

3. Circuito

Circuito is a free online tool used for designing circuits and diagrams. We used this to make our main circuit diagrams because it has most of the components we need.



Figure 14: Circuito.io

4. Draw.io

Draw.io is a free online graph drawing tool used to create various circuit diagrams. We used draw.io to add missing components in our diagrams.

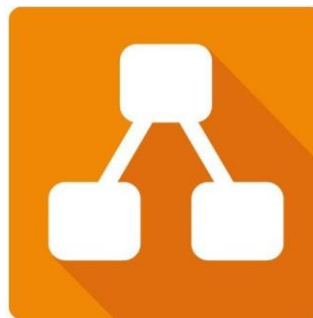


Figure 15: Draw.io

5. Lucid Chart

Lucid Chart is an online based intelligent diagramming system. It was used to create various diagrams including the block diagram of this project.



Figure 16: Lucid Chart

2.4. Classification of Sensors and Actuators of our project.

Sensors are devices that respond to stimuli from the physical environment. They can detect different kinds of stimuli based on their design and working principle (Jost, 2019). Actuators on the other hand are devices that help achieve physical movement in the real world by converting energy (Progressive Automations, 2024).

Sensors can be classified into various types based on energy conversion and nature of output signal.

Energy Conversion: Active sensors generate electrical signal as a function of physical measurements. Passive sensors modify electrical quantity as a function of physical measurements.

Output Signal: Analog Sensors produce continuous output signal. Digital sensors produce non-continuous signal.

Actuators can also be categorized into various types based on output movement and source of energy.

Output Movement: Linear Actuator move in a straight line whereas rotary actuators move in a circular motion.

Source of Energy: Electrical, Hydraulic, Pneumatic.

SENSOR	ENERGY CONVERSION	OUTPUT SIGNAL
DS18B20	Passive Sensor	Digital Signal
DHT11	Passive Sensor	Digital Signal
MAX30102/100	Passive Sensor	Digital Signal

Table 1: Sensors used classification.

3. Development

This section of the report shows how this idea was born and describes the step-by-step process of turning the idea into an actual tangible project. It also describes any hardships or problems we faced along the way and how we managed to solve it.

3.1. Planning and Design

When this project was first mentioned, our team had initially planned to build a weather station using only the DHT11 sensor. But as we brainstormed and researched, we thought we could be a little more ambitious and develop a patient health monitoring system.

We first found out suitable resources online to make finalize the components we will need along with the code. After finalizing all the required components, a feasibility study was done as we did not want this project to cost a lot of money. Then we settled on which microcontroller to use. We had first decided to use an Arduino but ESP32 was much better suited for our project as it had built in Wi-Fi module and didn't need any extra effort to connect to the internet.

A healthcare related device seemed very impactful and had real world application that would help a lot of people if executed at scale.

3.2. Resource Collection

As instructed, we wrote a letter to the resource department of Islington College stating all the materials we required. Out of our requirements the materials we were provided are:

- ESP32 Module
- DHT11 Sensor
- USB Cable
- Breadboard and Jumper Wires

We then sourced the following materials ourselves after collecting money from the team members:

- Resistor
- Pulse Oximeter

We also gathered the necessary circuit diagram and code from the internet, combining various sources.

3.3. System Development

After all the necessary resources were collected. We put everything together to finalize our prototype.

STEP 1: Connect the body temperature sensor to ESP32 Module.

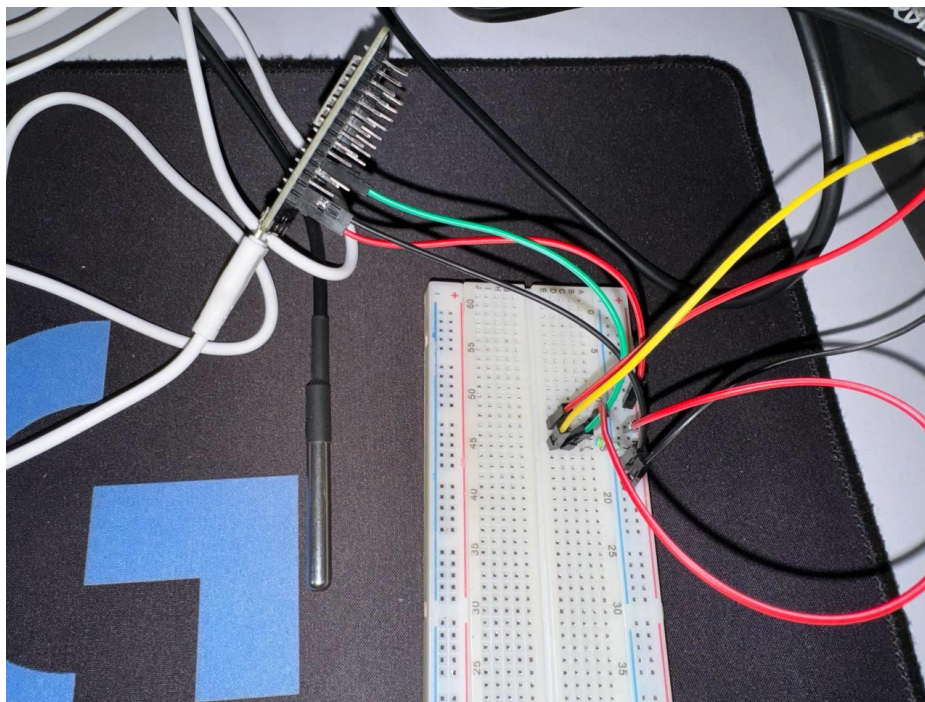


Figure 17: Connecting Body Temperature Sensor

In the Above Figure, the DS18B20 sensor has been connected to the ESP32 module. A common 3.3V and GND connection has been created in the positive and negative terminal of the bread board, and the data wire has been connected through a resistor.

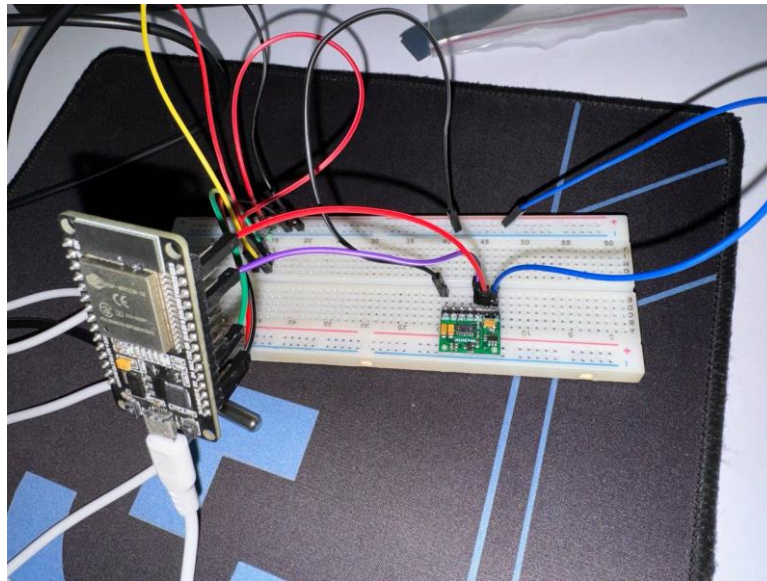
STEP 2: Connect the pulse oximeter.

Figure 18: Connecting Pulse Oximeter.

Here, the pulse oximeter is being connected next. It uses the same 3.3V and GND connection as the previous sensor from the common line. The two data pins have been connected to the ESP32 Module directly using jumper wires.

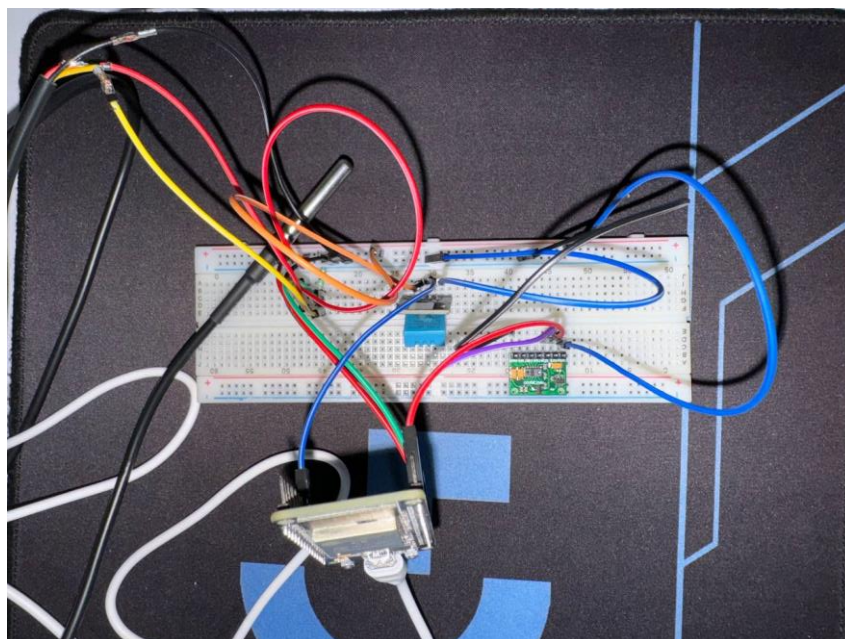
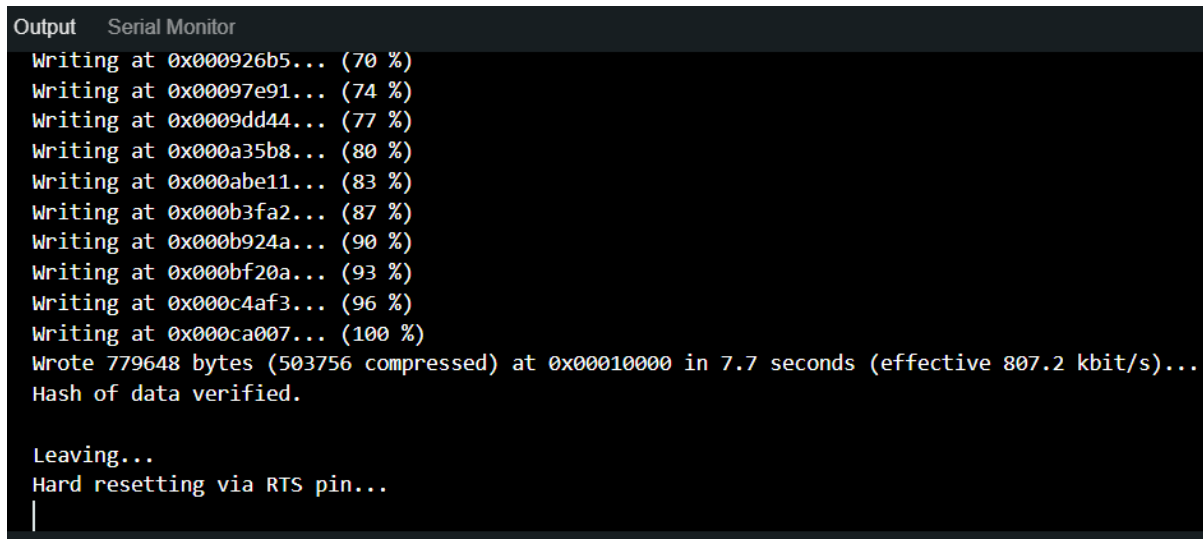
STEP 3: Connect humidity sensor.

Figure 19: Connect Humidity Sensor.

This is the final connection of our system. As the previous two sensors, it also uses the same 3.3V and GND connection. For now, there is enough power to run all three sensors, but we will be running this DTH sensor using the VIN pin if there are any power issues.

STEP 5: Upload Code to ESP32 Module.



```
Output  Serial Monitor
Writing at 0x000926b5... (70 %)
Writing at 0x00097e91... (74 %)
Writing at 0x0009dd44... (77 %)
Writing at 0x000a35b8... (80 %)
Writing at 0x000abe11... (83 %)
Writing at 0x000b3fa2... (87 %)
Writing at 0x000b924a... (90 %)
Writing at 0x000bf20a... (93 %)
Writing at 0x000c4af3... (96 %)
Writing at 0x000ca007... (100 %)
Wrote 779648 bytes (503756 compressed) at 0x00010000 in 7.7 seconds (effective 807.2 kbit/s)...
Hash of data verified.

Leaving...
Hard resetting via RTS pin...
|
```

Figure 20: Uploading Code to ESP32

At last, the code is uploaded to our ESP32 module after entering the correct wifi SSID and Password. The system assembly is then complete.

4. Result and Findings

After all the necessary connection have been made and the troubleshooting has been done, this project has worked without any issues. Although it may look simple from the outside, it took a lot of troubleshooting and minor fixes to get this project working properly. In the outcome of this project, the three sensors take relevant data from their environment. Body temperature, Room temperature, Humidity, Pulse and Oxygen Saturation. The sensor reading is then processed by the ESP32 and sent to our local webserver. This data can be accessed by anyone in the same network by entering the IP address. The system also alerts the user if their pulse is abnormal by showing an alert on the website and turning the build in LED of the ESP32 Module ON. The only change that needs to be done if we go to a new location is to update the Wi-Fi name and password in the code.

4.1. Testing

4.1.1. Check if the Local Webserver starts and can be accessed by devices.

TEST NO.	1
OBJECTIVE	Test if the local webserver starts and can be accessed.
ACTION	Copy the IP-Address and paste it in the browser.
EXPECTED RESULT	The webserver with the data will open
ACTUAL RESULT	The webserver with the data opened
CONCLUSION	Webserver accessed successfully

Table 2: Test 1

```
configsip: 0, SPIWP:0xee  
clk_drv:0x00,q_drv:0x00,d_drv:0x00,cs0_drv:0x00,hd_drv:0x00,wp_drv:0x00  
mode:DIO, clock div:1  
load:0x3fff0030,len:1344  
load:0x40078000,len:13964  
load:0x40080400,len:3600  
entry 0x400805f0  
Connecting to  
COMMON-Aashram  
..  
WiFi connected..!  
Got IP: 192.168.1.30  
HTTP server started
```

Figure 21: Getting the IP Address

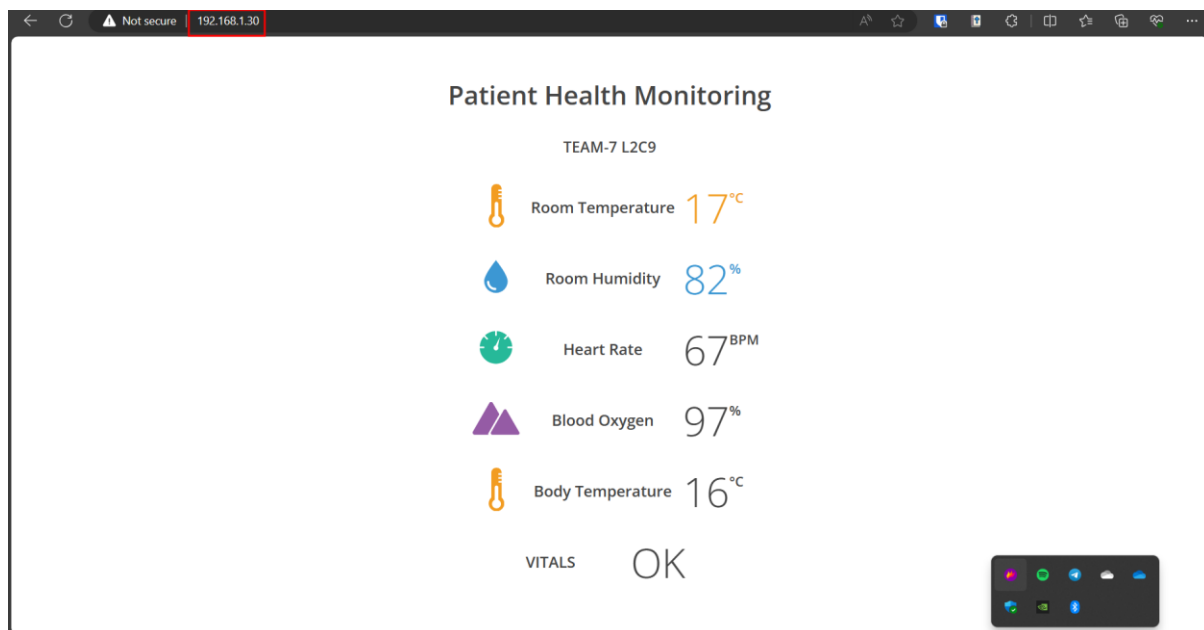


Figure 22: Accessing the local webserver.

4.1.2. Check if body temperature sensor works.

TEST NO.	2
OBJECTIVE	Check if the body temperature sensor works
ACTION	Heat the sensor and see if the values change
EXPECTED RESULT	The temperature will increase.
ACTUAL RESULT	The temperature increased.
CONCLUSION	Sensor works properly.

Table 3: Test 2

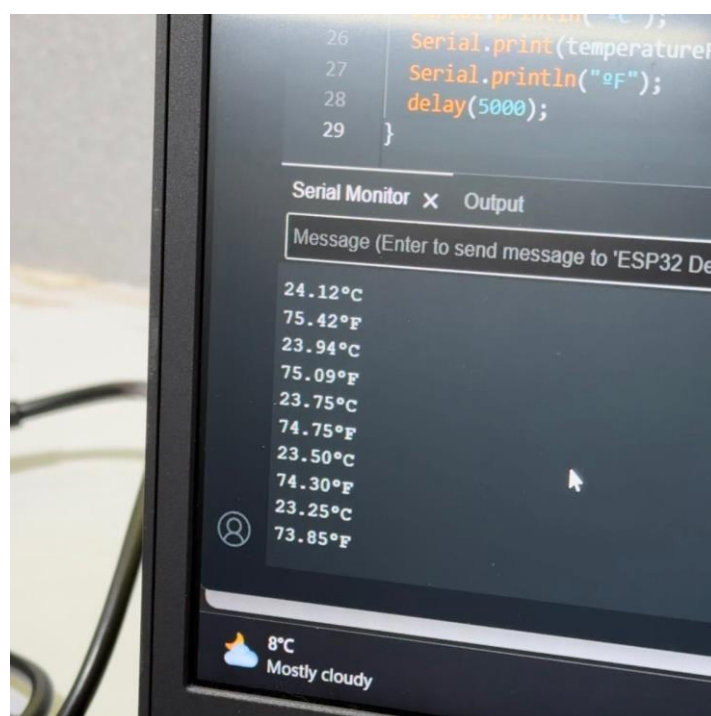


Figure 23: Sensor values before applying heat

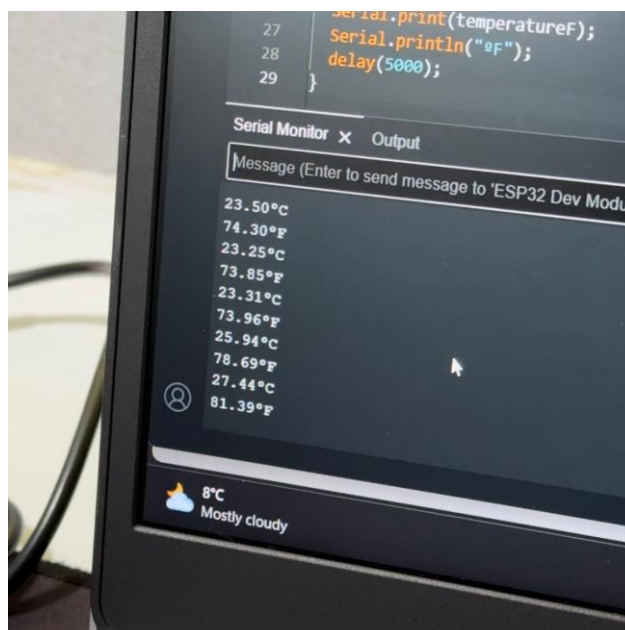


Figure 24: Sensor Value After Applying Heat.

4.1.3. Check if website shows alert if pulse is below 60.

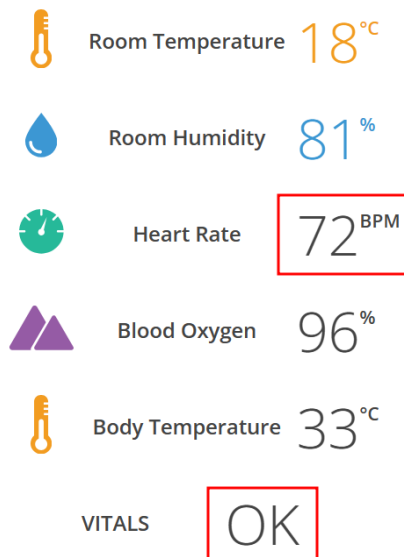
TEST NO.	3
OBJECTIVE	Check if website alert works.
ACTION	Remove finger from pulse sensor and drop pulse value to 0.
EXPECTED RESULT	Website will show alert.
ACTUAL RESULT	Website showed alert.
CONCLUSION	Alert function works properly.

Table 4: Test 3

192.168.1.30

Patient Health Monitoring

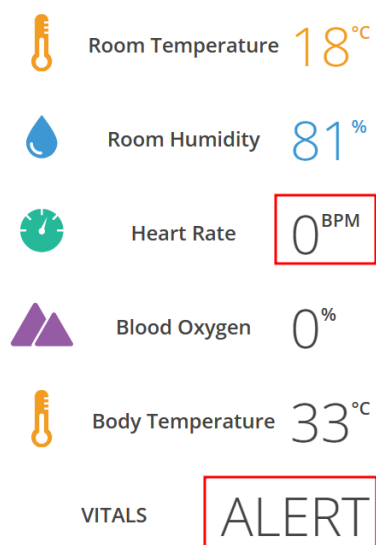
TEAM-7 L2C9

*Figure 25: While Pulse Normal*

192.168.1.30

Patient Health Monitoring

TEAM-7 L2C9

*Figure 26: Removing Finger from Pulse Sensor.*

4.1.4. Checking the LED and Website Alert System

TEST NO.	4
OBJECTIVE	Check if LED and Website Alert both work after removing finger.
ACTION	Remove finger from pulse sensor and drop pulse value to 0.
EXPECTED RESULT	Blue LED will turn on and Website will show Alert.
ACTUAL RESULT	LED turned on and Website showed error.
CONCLUSION	LED and Website Alert both functions properly.

Table 5: Test 4.

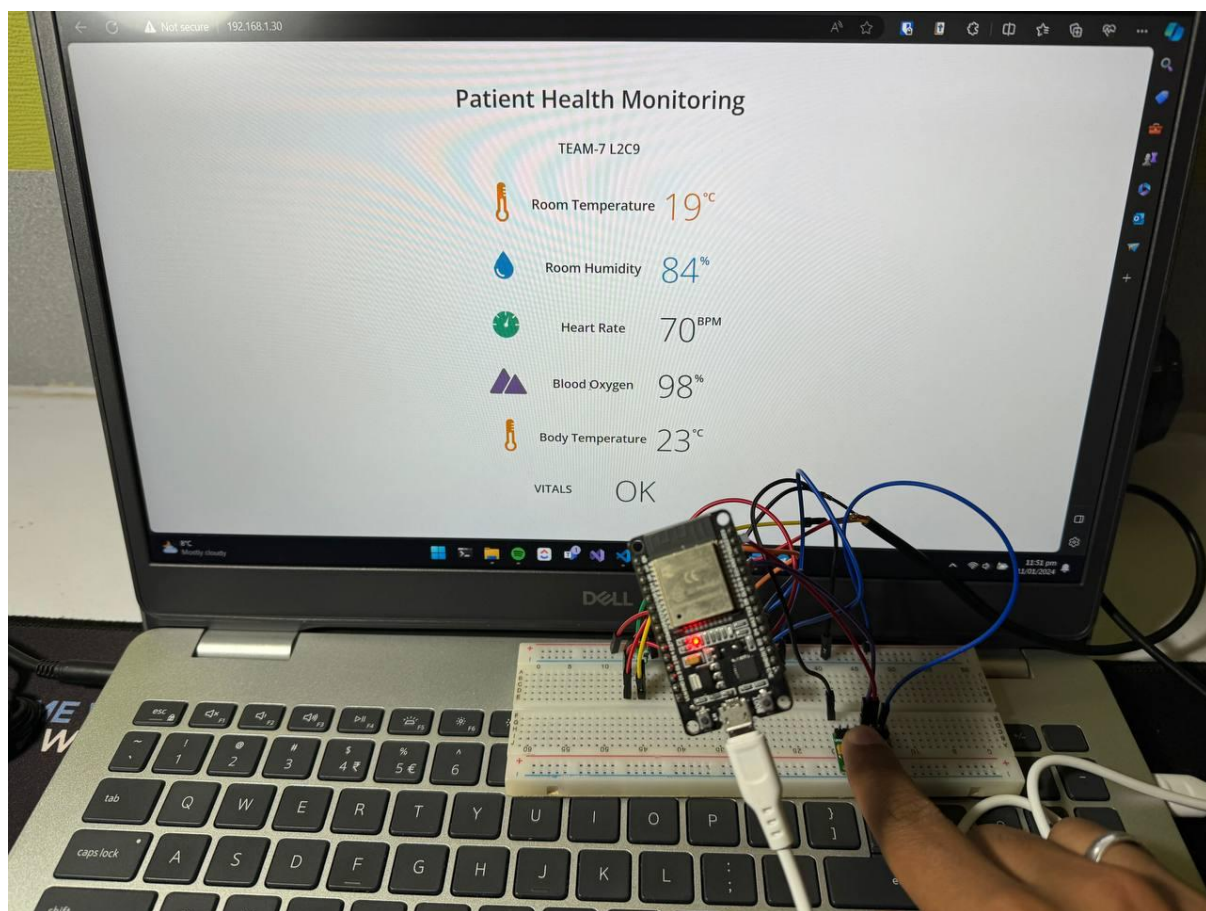


Figure 27: Pulse data with finger on the device.

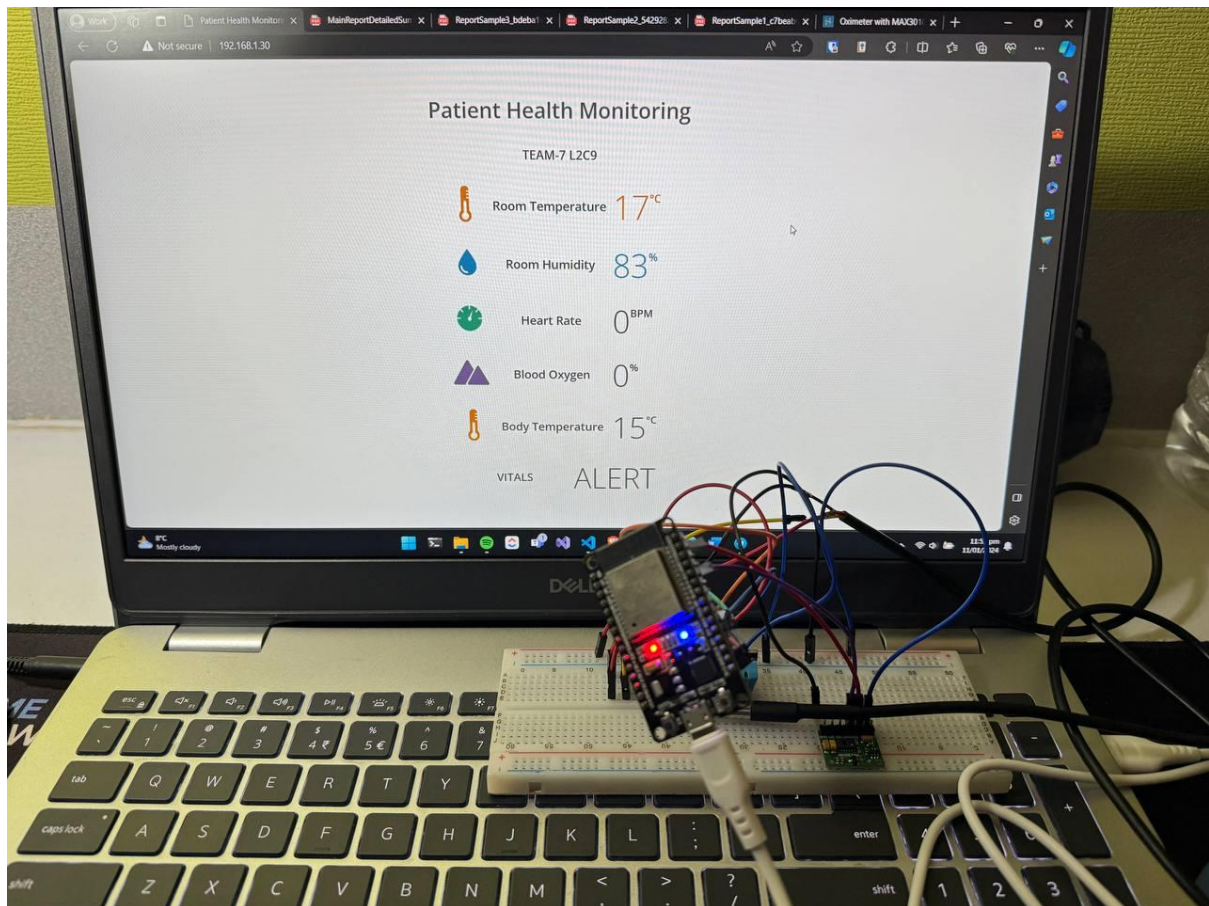


Figure 28: LED alert and Website Alert shown.

5. Future Works

Although this device is fully functional in this stage, there are various steps we can implement to improve the device.

- Integrate the device with any cloud service like AWS so that the patient data can be accessed from anywhere in the world rather than from the same network only.
- Create a better alert system where the device can send email or text message to concern people if the health data is not normal.
- Make the device battery powered so its portable.
- Store the medical health data in a database or in Excel so it can be accessed for future reference.
- Create a better user interface where the history of health data can also be accessed.
- Integrate the system with modern AI systems that can give suggestions to users based on their health data.

6. Conclusion

This project has been an immense challenge yet joy to work on. Being able to create such device has been a very rewarding experience and has helped us properly visualize the working as well as the mechanisms of IoT. Facing all the challenges and trying our hardest to overcome these challenges has helped us garner better knowledge and understanding of how systems like this work. During the start, this project seemed a very ambitious and we knew that we had to work diligently as a team and broaden our understanding of IoT systems.

The team has successfully created a patient health monitoring system and shown how IoT can be practically implemented in the real world. Along with that, the completion of prototype and evaluation report has given us an opportunity to get familiar with the real-world scenario, developing our critical thinking and teamwork skills.

7. References

Amazon Web Services, inc. , 2023. *What is IoT? - Internet of Things Explained - AWS*. [Online]

Available at: <https://aws.amazon.com/what-is/iot/>

[Accessed 3 January 2024].

Arduino.cc, 2024. *DS18B20 - Arduino Reference*. [Online]

Available at: <https://www.arduino.cc/reference/en/libraries/ds18b20/>

[Accessed 5 January 2024].

Circuito.io, 2018. *PCB Design for Beginners*. [Online]

Available at: <https://www.circuito.io/blog/pcb-design.amp/>

[Accessed 5 January 2024].

Ending Pandemics, 2023. *Continuous Surveillance Systems and Electronic Medical Record Integration in Nepal*. [Online]

Available at: <https://endingpandemics.org/projects/continuous-surveillance-systems-and-electronic-medical-record-integration-in-nepal/>

[Accessed 4 January 2024].

Espressif Systems (Shanghai) Co., Ltd., 2023. *ESP32 Series Datasheet*, Shanghai: Espressif Systems (Shanghai) Co., Ltd..

Hemmings, M., 2016. *What is a Jumper Wire?*. [Online]

Available at: <https://blog.sparkfuneducation.com/what-is-jumper-wire>

[Accessed 5 January 2024].

Jessica, W., Shuai, X. & John A, R., 2024. From lab to life: how wearable devices can improve health equity. *Nature Communications*.

Jost, D., 2019. *What is a Sensor?*. [Online]

Available at: <https://www.fierceelectronics.com/sensors/what-a-sensor>

[Accessed 7 January 2024].

Maxim Integrated Products, inc., 2014. *Pulse Oximeter and Heart-Rate Sensor IC for Wearable health*, California : Maxim Integrated Products, inc..

O'Malley, T., 2020. *The Need for Continous Monitoring in Today's Healthcare System*. [Online]

Available at: <https://www.managedhealthcareexecutive.com/view/need-continuous-monitoring-todays-healthcare-system>

[Accessed 15 December 2023].

Progressive Automations, 2024. *ACTUATOR AS A KEYSTONE OF MOTION*.

[Online]

Available at: <https://www.progressiveautomations.com/pages/actuators>

[Accessed 7 January 2024].

Sotera Digital Health, 2022. *Improved Hospital Care Through Continuous Patient Monitoring*. [Online]

Available at: <https://soteradigitalhealth.com/blog/improved-hospital-care-through-continuous-patient-monitoring>

[Accessed 15 December 2023].

The Kathmandu Post, 2023. *Government Mismanagement Leads to Crowding at Hospitals*, Kathmandu: The Kathmandu Post.

8. Appendix

8.1. Work Breakdown Structure

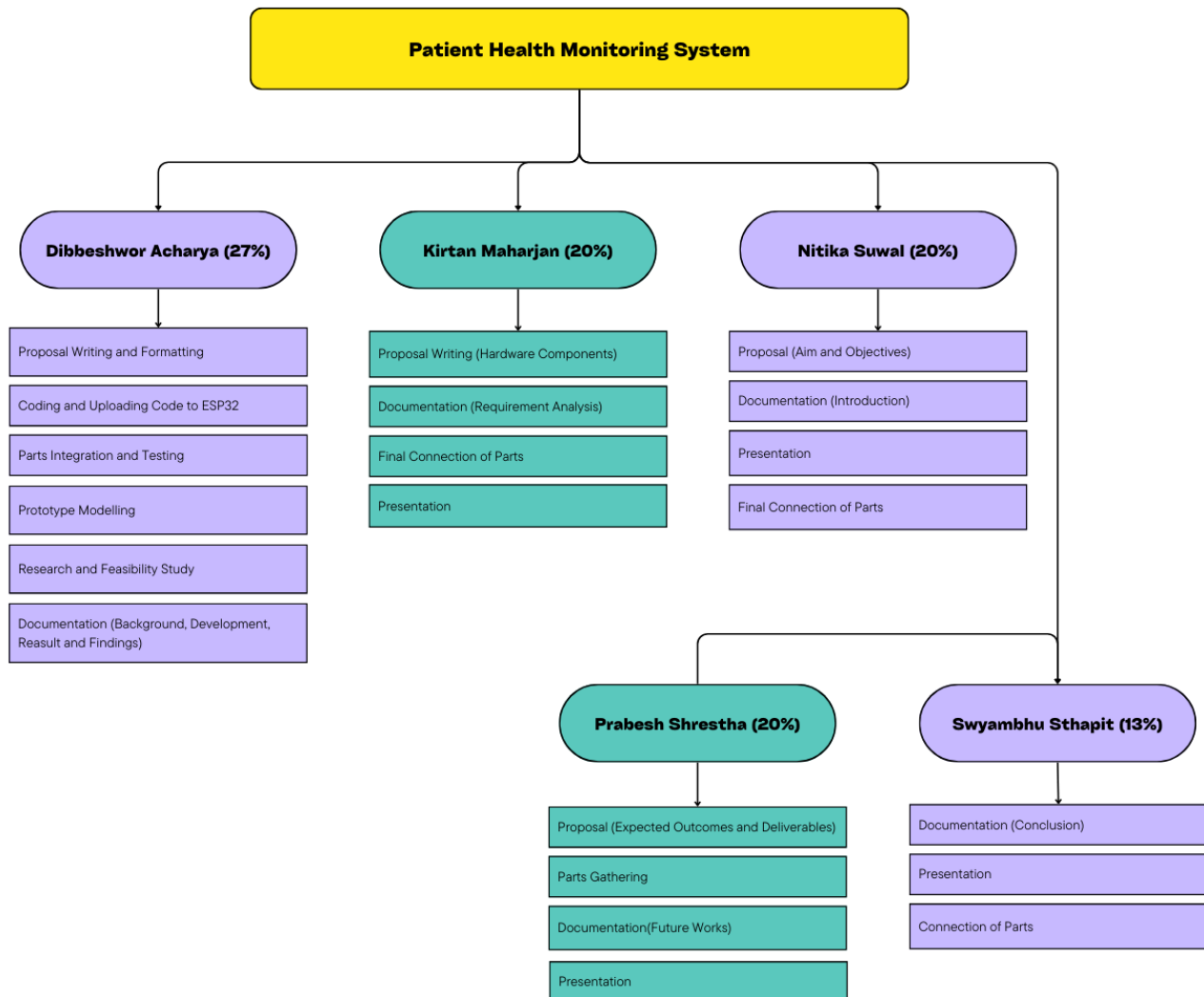


Figure 29: WBS (Canva)

8.2. Issue with the MAX30100 Sensor

We initially encountered a roadblock that we thought would change the entire scope of our project. The main issue was that no matter what we tried, we could not get the pulse oximeter to function. The sensor was not being initialized by ESP32. After hours of troubleshooting, we found out that it is a known design defect with the sensor and we needed to make changes to the PCB structure to make it work.

8.3. Video of the device working

The following link contains a folder that has all the photos and videos as proof of the device working and showing data.

[Shared Folder of Videos](#)

Password: Team7L2C9

8.4. Code

```
#include <WiFi.h>
#include <WebServer.h>
#include <Wire.h>
#include "MAX30100_PulseOximeter.h"
#include <OneWire.h>
#include <DallasTemperature.h>
#include "DHT.h"

#define DHT11_PIN 16
#define DS18B20 4
#define LED 2
#define REPORTING_PERIOD_MS 1000

float temperature, humidity, BPM, SpO2, bodytemperature;
char* alert = "OK";

/*Put your SSID & Password*/
const char* ssid = "COMMON-Aashram"; // Enter SSID here
const char* password = "heyuniverse"; //Enter Password here

PulseOximeter pox;
uint32_t tsLastReport = 0;
```

```
OneWire oneWire(DS18B20);  
DallasTemperature sensors(&oneWire);
```

```
WebServer server(80);
```

```
void onBeatDetected()  
{  
  Serial.println("Beat!");  
}
```

```
DHT dht(DHT11_PIN, DHT11);
```

```
void setup() {  
  Serial.begin(115200);  
  pinMode(19, OUTPUT);  
  pinMode(LED, OUTPUT);  
  delay(100);
```

```
  Serial.println("Connecting to ");  
  Serial.println(ssid);
```

```
  //connect to your local wi-fi network  
  WiFi.begin(ssid, password);
```

```
  //check wi-fi is connected to wi-fi network  
  while (WiFi.status() != WL_CONNECTED) {  
    delay(1000);  
    Serial.print(".");  
  }  
  Serial.println("");  
  Serial.println("WiFi connected..!");
```

```
Serial.print("Got IP: "); Serial.println(WiFi.localIP());

server.on("/", handle_OnConnect);
server.onNotFound(handle_NotFound);

server.begin();
Serial.println("HTTP server started");

Serial.print("Initializing pulse oximeter..");

if (!pox.begin()) {
    Serial.println("FAILED");
    for (;;)
} else {
    Serial.println("SUCCESS");
    pox.setOnBeatDetectedCallback(onBeatDetected);
}

pox.setIRLedCurrent(MAX30100_LED_CURR_7_6MA);

// Register a callback for the beat detection
pox.setOnBeatDetectedCallback(onBeatDetected);

dht.begin();
sensors.begin();

}

void loop() {
    server.handleClient();
    pox.update();
    // sensors.requestTemperatures();
    // int chk = DHT.read11(DHT11_PIN);
```

```
humidity = dht.readHumidity();
temperature = dht.readTemperature();
BPM = pox.getHeartRate();
SpO2 = pox.getSpO2();
bodytemperature = sensors.getTempCByIndex(0);

if (BPM < 60 or BPM > 120) {
    digitalWrite(LED,HIGH);
    alert = "ALERT!";
} else {
    digitalWrite(LED,LOW);
}

if (millis() - tsLastReport > REPORTING_PERIOD_MS)
{
    Serial.print("Room Temperature: ");
    Serial.print(temperature);
    Serial.println("°C");

    Serial.print("Room Humidity: ");
    Serial.print(humidity);
    Serial.println("%");

    Serial.print("BPM: ");
    Serial.println(BPM);

    Serial.print("SpO2: ");
    Serial.print(SpO2);
    Serial.println("%");
```



```
Serial.print("Body Temperature: ");
Serial.print(bodytemperature);
Serial.println("°C");

Serial.println("*****");
Serial.println();

    tsLastReport = millis();
}

}

void handle_OnConnect() {

    server.send(200, "text/html", SendHTML(temperature, humidity, BPM, SpO2,
bodytemperature, alert));
}

void handle_NotFound(){
    server.send(404, "text/plain", "Not found");
}

String SendHTML(float temperature,float humidity,float BPM,float SpO2, float
bodytemperature, char* alert){
    String ptr = "<!DOCTYPE html>";
    ptr += "<html>";
    ptr += "<head>";
    ptr += "<title> Patient Health Monitoring</title>";
    ptr += "<meta name='viewport' content='width=device-width, initial-scale=1.0'>";
    ptr += "<link href='https://fonts.googleapis.com/css?family=Open+Sans:300,400,600'
rel='stylesheet'>";
    ptr += "<style>";
```

```
ptr += "html { font-family: 'Open Sans', sans-serif; display: block; margin: 0px auto;
text-align: center; color: #444444; }";
ptr += "body { margin: 0px; } ";
ptr += "h1 { margin: 50px auto 30px; } ";
ptr += ".side-by-side { display: table-cell; vertical-align: middle; position: relative; }";
ptr += ".text { font-weight: 600; font-size: 19px; width: 200px; }";
ptr += ".reading { font-weight: 300; font-size: 50px; padding-right: 25px; }";
ptr += ".temperature .reading { color: #F29C1F; }";
ptr += ".humidity .reading { color: #3B97D3; }";
ptr += ".BPM .reading { color: #FF0000; }";
ptr += ".SpO2 .reading { color: #955BA5; }";
ptr += ".bodytemperature .reading { color: #F29C1F; }";
ptr += ".superscript { font-size: 17px; font-weight: 600; position: absolute; top: 10px; }";
ptr += ".data { padding: 10px; }";
ptr += ".container { display: table; margin: 0 auto; }";
ptr += ".icon { width: 65px; }";
ptr += "</style>";
ptr += "</head>";
ptr += "<body>";
ptr += "<h1>Patient Health Monitoring</h1>";
ptr += "<h3>TEAM-7 L2C9</h3>";
ptr += "<div class='container'>";

ptr += "<div class='data temperature'>";
ptr += "<div class='side-by-side icon'>";

ptr += "<svg enable-background='new 0 0 19.438 54.003' height=54.003px
id=Layer_1 version=1.1 viewBox='0 0 19.438 54.003' width=19.438px x=0px
xml:space=preserve xmlns=http://www.w3.org/2000/svg
xmlns:xlink=http://www.w3.org/1999/xlink y=0px><g><path d='M11.976,8.82v-
2h4.084V6.063C16.06,2.715,13.345,0,9.996,0H9.313C5.965,0,3.252,2.715,3.252,6.
063v30.982'>";

ptr
+= "C1.261,38.825,0,41.403,0,44.286c0,5.367,4.351,9.718,9.719,9.718c5.368,0,9.71
9-4.351,9.719-9.718";
```

```
ptr += "c0-2.943-1.312-5.574-3.378-7.355V18.436h-3.914v-2h3.914v-2.808h-4.084v-2h4.084V8.82H11.976z M15.302,44.833";

ptr += "c0,3.083-2.5,5.583-5.583,5.583s-5.583-2.5-5.583-5.583c0-2.279,1.368-4.236,3.326-5.104V24.257C7.462,23.01,8.472,22,9.719,22";

ptr += "s2.257,1.01,2.257,2.257V39.73C13.934,40.597,15.302,42.554,15.302,44.833z'fill=#F29C21 /></g></svg>";

ptr += "</div>";

ptr += "<div class='side-by-side text'>Room Temperature</div>";

ptr += "<div class='side-by-side reading'>";

ptr += (int)temperature;

ptr += alert;

ptr += "<span class='superscript'>&deg;C</span></div>";

ptr += "</div>";


ptr += "<div class='data humidity'>";

ptr += "<div class='side-by-side icon'>";

ptr += "<svg enable-background='new 0 0 29.235 40.64'height=40.64px id=Layer_1 version=1.1 viewBox='0 0 29.235 40.64'width=29.235px x=0px xml:space=preserve xmlns=http://www.w3.org/2000/svg xmlns:xlink=http://www.w3.org/1999/xlink y=0px><path d='M14.618,0C14.618,0,0,17.95,0,26.022C0,34.096,6.544,40.64,14.618,40.64s14.617-6.544,14.617-14.617'";

ptr += "C29.235,17.95,14.618,0,14.618,0z M13.667,37.135c-5.604,0-10.162-4.56-10.162-10.162c0-0.787,0.638-1.426,1.426-1.426";

ptr += "c0.787,0,1.425,0.639,1.425,1.426c0,4.031,3.28,7.312,7.311,7.312c0.787,0,1.425,0.638,1.425,1.425";

ptr += "C15.093,36.497,14.455,37.135,13.667,37.135z'fill=#3C97D3 /></svg>";

ptr += "</div>";

ptr += "<div class='side-by-side text'>Room Humidity</div>";

ptr += "<div class='side-by-side reading'>";

ptr += (int)humidity;

ptr += "<span class='superscript'>%</span></div>";

ptr += "</div>";
```

```

ptr += "<div class='data Heart Rate'>";
ptr += "<div class='side-by-side icon'>";

ptr += "<svg enable-background='new 0 0 40.542 40.541'height=40.541px
id=Layer_1 version=1.1 viewBox='0 0 40.542 40.541'width=40.542px x=0px
xml:space=preserve xmlns=http://www.w3.org/2000/svg
xmlns:xlink=http://www.w3.org/1999/xlink y=0px><g><path d='M34.313,20.271c0-
0.552,0.447-1,1-1h5.178c-0.236-4.841-2.163-9.228-5.214-12.593l-3.425,3.424";

ptr += "c-0.195,0.195-0.451,0.293-0.707,0.293s-0.512-0.098-0.707-0.293c-0.391-
0.391-0.391-1.023,0-1.414l3.425-3.424";

ptr += "c-3.375-3.059-7.776-4.987-12.634-
5.215c0.015,0.067,0.041,0.13,0.041,0.202v4.687c0,0.552-0.447,1-1,1s-1-0.448-1-
1v0.25";

ptr += "c0-0.071,0.026-0.134,0.041-
0.202C14.39,0.279,9.936,2.256,6.544,5.385l3.576,3.577c0.391,0.391,0.391,1.024,0,
1.414";

ptr += "c-0.195,0.195-0.451,0.293-0.707,0.293s-0.512-0.098-0.707-
0.293L5.142,6.812c-2.98,3.348-4.858,7.682-5.092,12.459h4.804";

ptr += "c0.552,0,1,0.448,1,1s-0.448,1-
1,1H0.05c0.525,10.728,9.362,19.271,20.22,19.271c10.857,0,19.696-8.543,20.22-
19.271h-5.178";

ptr += "C34.76,21.271,34.313,20.823,34.313,20.271z M23.084,22.037c-
0.559,1.561-2.274,2.372-3.833,1.814";

ptr += "c-1.561-0.557-2.373-2.272-1.815-3.833c0.372-1.041,1.263-1.737,2.277-
1.928L25.2,7.202L22.497,19.05";

ptr += "C23.196,19.843,23.464,20.973,23.084,22.037z'fill=#26B999 /></g></svg>";
ptr += "</div>";

ptr += "<div class='side-by-side text'>Heart Rate</div>";
ptr += "<div class='side-by-side reading'>";
ptr += (int)BPM;
ptr += "<span class='superscript'>BPM</span></div>";
ptr += "</div>";

ptr += "<div class='data Blood Oxygen'>";
ptr += "<div class='side-by-side icon'>";

ptr += "<svg enable-background='new 0 0 58.422 40.639'height=40.639px
id=Layer_1 version=1.1 viewBox='0 0 58.422 40.639'width=58.422px x=0px
xml:space=preserve xmlns=http://www.w3.org/2000/svg
xmlns:xlink=http://www.w3.org/1999/xlink y=0px><g><path

```

```
d='M58.203,37.754l0.007-0.004L42.09,9.935l-0.001,0.001c-0.356-0.543-0.969-0.902-1.667-0.902";
```

```
ptr += "c-0.655,0-1.231,0.32-1.595,0.808l-0.011-0.007l-0.039,0.067c-0.021,0.03-0.035,0.063-0.054,0.094L22.78,37.692l0.008,0.004";
```

```
ptr += "c-0.149,0.28-0.242,0.594-0.242,0.934c0,1.102,0.894,1.995,1.994,1.995v0.015h31.888c1.101,0,1.994-0.893,1.994-1.994";
```

```
ptr += "C58.422,38.323,58.339,38.024,58.203,37.754z'fill=#955BA5 /><path d='M19.704,38.674l-0.013-0.004l13.544-23.522L25.13,1.156l-0.002,0.001C24.671,0.459,23.885,0,22.985,0";
```

```
ptr += "c-0.84,0-1.582,0.41-2.051,1.038l-0.016-0.01L20.87,1.114c-0.025,0.039-0.046,0.082-0.068,0.124L0.299,36.851l0.013,0.004";
```

```
ptr += "C0.117,37.215,0,37.62,0,38.059c0,1.412,1.147,2.565,2.565,2.565v0.015h16.989c-0.091-0.256-0.149-0.526-0.149-0.813";
```

```
ptr += "C19.405,39.407,19.518,39.019,19.704,38.674z'fill=#955BA5 /></g></svg>";
```

```
ptr += "</div>";
```

```
ptr += "<div class='side-by-side text'>Blood Oxygen</div>";
```

```
ptr += "<div class='side-by-side reading'>";
```

```
ptr += (int)SpO2;
```

```
ptr += "<span class='superscript'>%</span></div>";
```

```
ptr += "</div>";
```

```
ptr += "<div class='data Body Temperature'>";
```

```
ptr += "<div class='side-by-side icon'>";
```

```
ptr += "<svg enable-background='new 0 0 19.438 54.003'height=54.003px id=Layer_1 version=1.1 viewBox='0 0 19.438 54.003'width=19.438px x=0px xml:space=preserve xmlns=http://www.w3.org/2000/svg xmlns:xlink=http://www.w3.org/1999/xlink y=0px><g><path d='M11.976,8.82v-2h4.084V6.063C16.06,2.715,13.345,0,9.996,0H9.313C5.965,0,3.252,2.715,3.252,6.063v30.982";
```

```
ptr += "C1.261,38.825,0,41.403,0,44.286c0,5.367,4.351,9.718,9.719,9.718c5.368,0,9.719-4.351,9.719-9.718";
```

```
ptr += "c0-2.943-1.312-5.574-3.378-7.355V18.436h-3.914v-2h3.914v-2.808h-4.084v-2h4.084V8.82H11.976z M15.302,44.833";
```

```
ptr += "c0,3.083-2.5,5.583-5.583,5.583s-5.583-2.5-5.583-5.583c0-2.279,1.368-4.236,3.326-5.104V24.257C7.462,23.01,8.472,22,9.719,22";
```

```
ptr
+="
```