

SMART PATIENT HEALTH MONITORING SYSTEM USING IOT

MINI PROJECT REPORT

Submitted by,

MATAM MANASWINI (BU21EECE0100545)

SHREE RAKSHA B N (BU21EECE0100552)

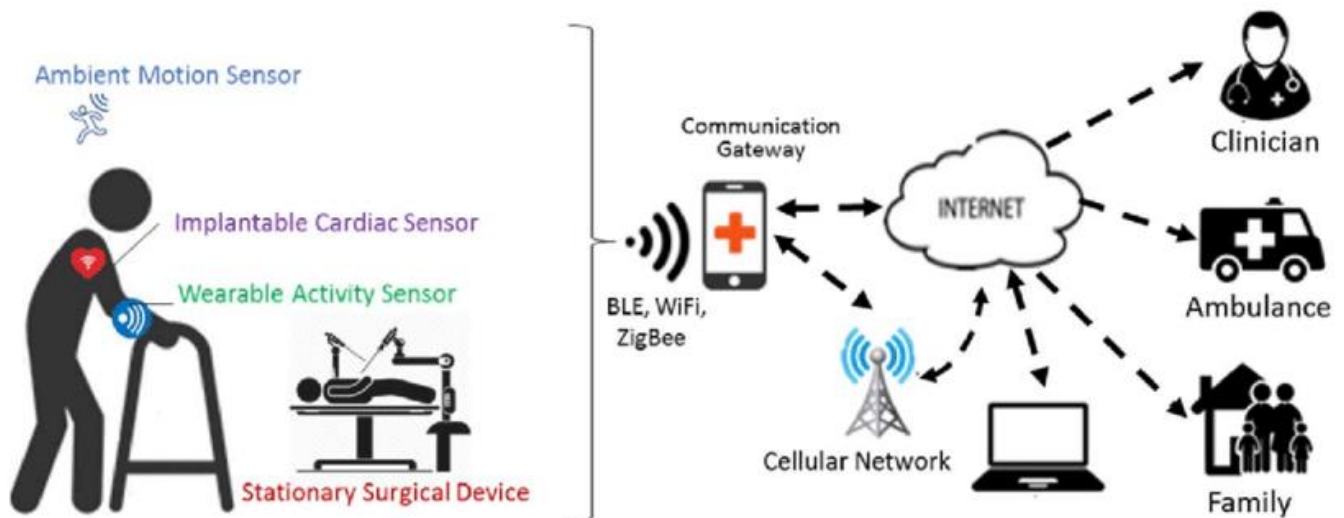
MEGHANA B (BU21EECE0100559)



AIM OF THE MINI-PROJECT:

“To create an integrated Internet of Things (IoT)-based patient health monitoring system that uses the DS18B20 temperature sensor to track body temperature and the MAX30100/102 pulse oxygen meter sensor to measure blood oxygen level (SpO2) and heart rate/pulse (BPM) in real-time. Incorporating the DHT11 Humidity & Temperature Sensor to control and maintain ideal environmental conditions in the patient's room is another way the project seeks to improve patient comfort. Continuous monitoring of critical health metrics and environmental elements will be made possible by the system, which will also guarantee a patient-friendly atmosphere and give healthcare providers timely alerts and insights for preventive intervention.”

PATIENT HEALTH MONITORING ON LARGER SCALE

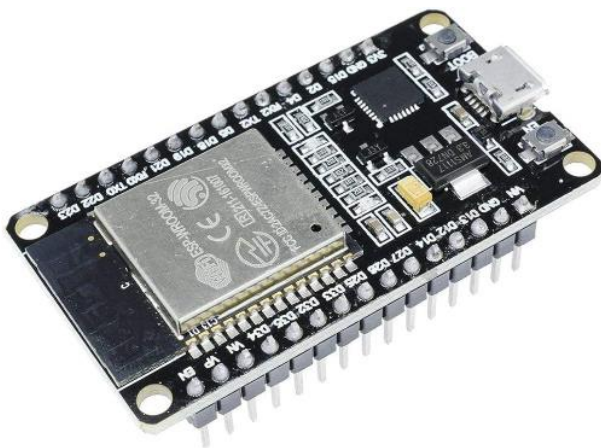


One of the most significant developments in the medical industry in the current digital era is smart health care. Traditional medicine, which is founded on bioengineering, has increasingly started to digitalize and informationize itself in response to scientific theory and technological advancements. Among these technical innovations is the Internet of Things. The ability to continuously monitor a patient by looking at a variety of metrics and predicting a positive outcome based on past results is what sets the Internet of Things apart in the healthcare sector. Hospitals commonly utilize this to share data with clinicians and continuously monitor patients in Intensive Care Units (ICUs). This helps in the early detection of abnormalities in patients and the timely delivery of care. Smart health care enables individuals from diverse types of backgrounds (e.g., doctors, health workers, physician caregivers, older relatives, and patients) to obtain the proper information and discover the right solutions, with the goal of minimizing medical errors, improving efficiency, and lowering costs at the proper moment in the health-care profession. Smart health care employs a variety of techniques, including the use of mobile phones, computers, and televisions, as well as the use of different networks, including wide area networks, local area networks, and body area networks. Body temperature, pulse rate, blood pressure, and motion detection are the most often monitored metrics.

Overview on hardware and software requirements

S.N.	Components Name	Quantity
1	ESP32 Board	1
2	MAX30100 Pulse Oximeter Sensor	1
3	DS18B20 Sensor	1
4	DHT11 Sensor	1
5	Resistor 4.7K	1
6	Connecting Wires	10
7	Breadboard	1

ESP32 BOARD



The ESP32 is a versatile and powerful microcontroller and Wi-Fi/Bluetooth module that has gained popularity in the world of embedded systems and IoT (Internet of Things) projects. Developed by Espressif Systems, the ESP32 builds upon the success of its predecessor, the ESP8266, and brings several improvements and additional features. Here are some key properties and characteristics of the ESP32 board:

1. Microcontroller Unit (MCU):

- The ESP32 integrates a dual-core Xtensa LX6 microcontroller, which allows for efficient multitasking and processing capabilities. The dual-core architecture enables the execution of multiple tasks simultaneously.

2. Clock Frequency:

- The ESP32 typically operates at clock frequencies of up to 240 MHz, providing substantial computing power for a wide range of applications.

3. Memory:

- It is equipped with both volatile and non-volatile memory. The ESP32 usually has various configurations with different amounts of Flash memory (for program storage) and RAM (for data storage). This enables users to choose a suitable configuration based on the requirements of their projects.

4. Wireless Connectivity:

- One of the standout features of the ESP32 is its built-in Wi-Fi and Bluetooth capabilities. It supports 802.11 b/g/n Wi-Fi and Bluetooth Classic (BR/EDR) as well as Bluetooth Low Energy (BLE). This makes it suitable for IoT applications requiring wireless communication.

5. Peripheral Interfaces:

- The ESP32 includes a rich set of peripheral interfaces, such as UART, SPI, I2C, I2S, PWM, and GPIO pins. These interfaces facilitate communication with various external devices and sensors.

6. Analog-to-Digital Converter (ADC):

- The ESP32 features a built-in ADC that allows it to read analog signals. This is useful for interfacing with analog sensors and acquiring data from the physical world.

7. Security Features:

- The ESP32 includes security features such as hardware-accelerated encryption (AES, SHA-2, RSA), secure boot, and flash encryption. These features enhance the overall security of the device, making it suitable for applications where data integrity and confidentiality are crucial.

8. Low Power Consumption:

- The ESP32 is designed with power efficiency in mind, making it suitable for battery-powered applications. It includes various low-power modes to minimize energy consumption when the device is in a standby or sleep state.

9. Development Environment:

- The ESP32 is well-supported by the Arduino IDE and the Expressive IDF (IoT Development Framework). This ensures a user-friendly development environment with a vast community and extensive documentation.

10. Modularity:

- The ESP32 is available in various module formats, including development boards, system-on-chip (SoC) modules, and compact surface-mount modules. This modularity allows users to choose the form factor that best fits their project requirements.

11. Community Support:

- The ESP32 has gained significant popularity in the maker and developer communities, resulting in extensive community support. This includes forums, online resources, and a wealth of contributed libraries and examples.

In summary, the ESP32 is a feature-rich microcontroller with built-in Wi-Fi and Bluetooth capabilities, making it an excellent choice for a wide range of IoT and embedded systems projects. Its combination of processing power, wireless connectivity, and a rich set of peripherals has contributed to its widespread adoption in the world of electronics and programming.

MAX30100 Pulse Oximeter Sensor



1. Functionality:

- Combines two LEDs, a photodetector, optimized optics, and low-noise analog signal processing. Designed for pulse and heart-rate signal detection.

2. Power Supply:

- Operates within the range of 1.8V to 3.3V. Features software-controlled power-down mode with negligible standby current.

3. LEDs:

- Equipped with two LEDs - one emitting red light and the other emitting infrared light. Red light and infrared light used for both pulse rate and oxygen level measurement.

4. Oxygen Level Measurement:

- In oxygen level measurement, only the infrared light is utilized. Oxygenated blood absorbs more infrared light, while deoxygenated blood absorbs red light. The sensor measures absorption levels to determine pulse rate and oxygen levels in the blood.

5. Pulse Rate Determination:

- Pulse rate is determined by analysing the time between the increase and decrease of oxygenated blood during the heart's pumping and relaxing phases.

6. Operating Principle:

- When the heart pumps blood, oxygenated blood increases, leading to higher infrared light absorption. As the heart relaxes, the volume of oxygenated blood decreases, causing lower infrared light absorption. The sensor reads absorption levels for both light sources (red and infrared) and stores them in a buffer accessible via I2C.

7. Communication Protocol:

- Utilizes I2C for communication, allowing easy integration into various microcontroller platforms.

8. Applications:

- Suitable for applications requiring real-time monitoring of pulse rate and blood oxygen levels, such as fitness trackers, health monitoring devices, and wearable technologies.

9. MAX30100 Features:

- Optimized optics for accurate signal detection.
- Low-noise analog signal processing for enhanced reliability.
- Power-efficient design with software-controlled power-down mode.
- Compatibility with a range of power supplies for flexible usage.

In summary, the MAX30100 is a compact and versatile sensor that combines LEDs, a photodetector, and advanced signal processing to enable accurate and real-time monitoring of pulse rate and blood oxygen levels. Its low-power design and communication flexibility make it suitable for integration into a variety of electronic devices focused on health and fitness tracking.

DS18B20 Temperature Sensor



The DS18B20 sensor is a pre-wired and waterproofed temperature sensor, providing a convenient solution for measuring temperature in distant or wet conditions. Here are its key features:

1. Temperature Measurement Range:

- Capable of measuring temperatures within the range of -55 to 125°C (-67°F to $+257^{\circ}\text{F}$).

2. Waterproof Design:

- The sensor is waterproofed, making it suitable for applications in wet conditions or when measurement is required at a distance.

3. Cable Material:

- The cable is jacketed in PVC, enhancing its durability and providing protection in various environments.

4. Digital Output:

- The DS18B20 sensor utilizes digital communication, eliminating signal degradation even over long distances. This is especially advantageous for applications where analog signals might suffer from interference.

5. Precision:

- The sensor offers a high level of precision, with an accuracy of approximately $\pm 0.5^{\circ}\text{C}$ over a significant temperature range.

6. Resolution:

- Provides up to 12 bits of precision through the onboard digital-to-analog converter, enabling detailed temperature measurements.

7. Compatibility:

- Compatible with any microcontroller using a single digital pin for communication. This makes it versatile and easily integrable into a variety of projects and systems.

8. Communication Protocol:

- Utilizes the Dallas 1-Wire protocol for communication. While this protocol is known for its complexity, it ensures reliable and accurate data transmission from the sensor.

9. Pull-Up Resistor:

- The sensor requires a 4.7k pull-up resistor, included in the package. This resistor is essential for establishing a proper connection between the DATA and VCC lines during sensor operation.

10. Application:

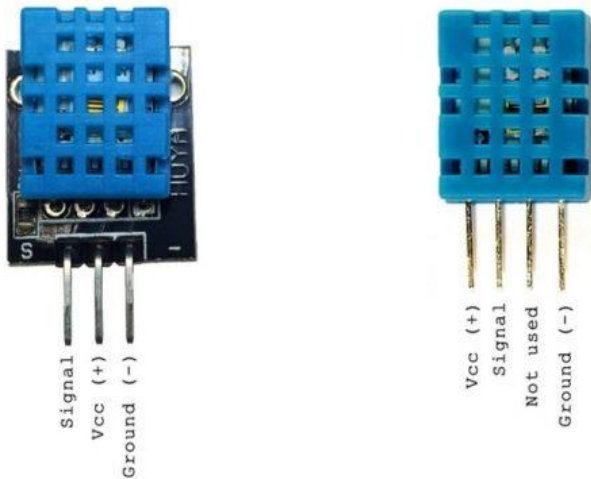
- Well-suited for applications where temperature measurement is critical, such as weather stations, industrial processes, agriculture, and other scenarios where a rugged and waterproof temperature sensor is necessary.

11. Ease of Use:

- Despite the complexity of the 1-Wire protocol, the sensor is made more user-friendly with the inclusion of a pull-up resistor and compatibility with a single digital pin, simplifying integration into various projects.

In summary, the DS18B20 sensor offers a reliable and waterproof solution for measuring temperature in challenging conditions. Its digital output, high precision, and compatibility with microcontrollers make it a versatile choice for a range of applications, despite the complexity associated with the Dallas 1-Wire protocol.

DHT11 Humidity & Temperature Sensor



The DHT11 is a commonly used and cost-effective sensor that provides readings of both temperature and humidity. Compact and cost-effective, it has become a popular choice in the electronics community for its relative accuracy and ease of integration. Operating on a simple one-wire communication protocol, the DHT11 can be seamlessly interfaced with a variety of microcontrollers, making it suitable for a wide range of applications.

Developed for digital systems, the DHT11 finds its applications in environments where there's a need to monitor or control atmospheric conditions, such as in smart homes, weather stations, and agricultural monitoring systems.

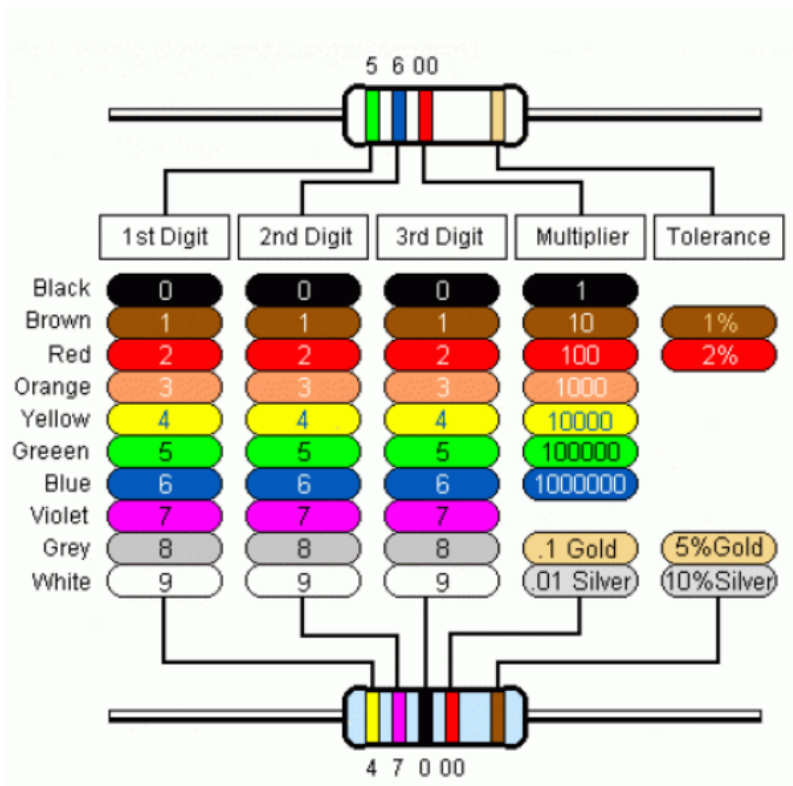
Pinout of DHT11 Sensor

The DHT11 typically comes with a 4-pin package, although only three of them are functionally used:

1. **VCC (Pin 1):** This is the power supply pin, which can accept voltages from 3.3V to 5V, making it compatible with most microcontroller systems.
2. **Data (Pin 2):** This is the pin through which the DHT11 communicates. It uses a proprietary single-wire protocol to transmit temperature and humidity data to the connected microcontroller.
3. **NC (Pin 3):** Not connected or used.
4. **GND (Pin 4):** Ground pin, used to complete the circuit.

RESISTOR [4.7K]

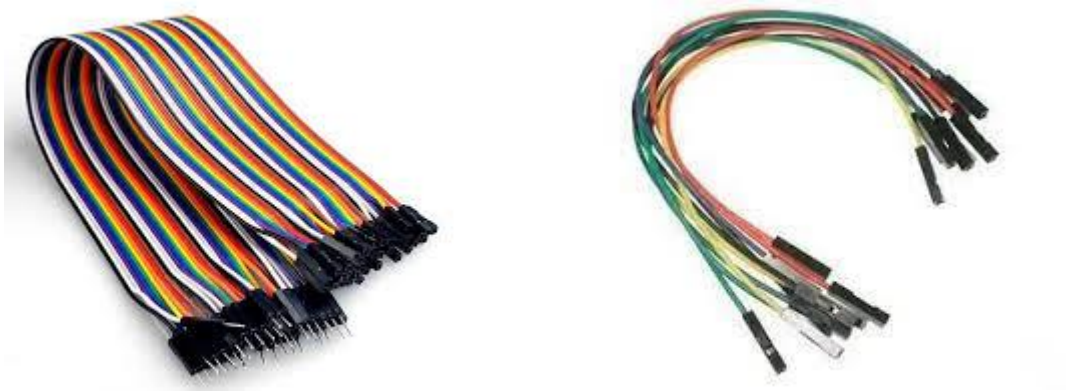
A resistor is an electrical component that limits or regulates the flow of electrical current in an electronic circuit. Resistors can also be used to provide a specific voltage for an active device such as a transistor.



Applications of Resistor

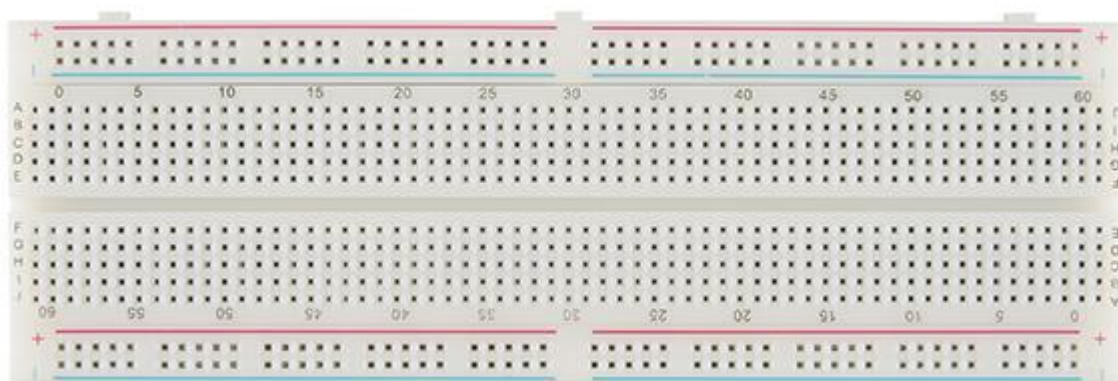
- 1) Wire wound resistors find applications where balanced current control, high sensitivity, and accurate measurement are required like in shunt with ampere meter.
- 2) Photoresistors find application in flame detectors, burglar alarms, in photographic devices, etc.
- 3) Resistors are used for controlling temperature and voltmeter.
- 4) Resistors are used in digital multi-meter, amplifiers, telecommunication, and oscillators.
- 5) They are also used in modulators, demodulators, and transmitters.

Connecting Wires



A wire is a flexible strand of metal, usually cylindrical. Wires are used for establishing electrical conductivity between two devices of an electrical circuit. They possess negligible resistance to the passage of current. The wires are covered by an insulated coating of different colours. The colour codes are used to [distinguish between neutral and ground](#), and live wire, which differs from one country to another.

SOLDERLESS BREADBOARD

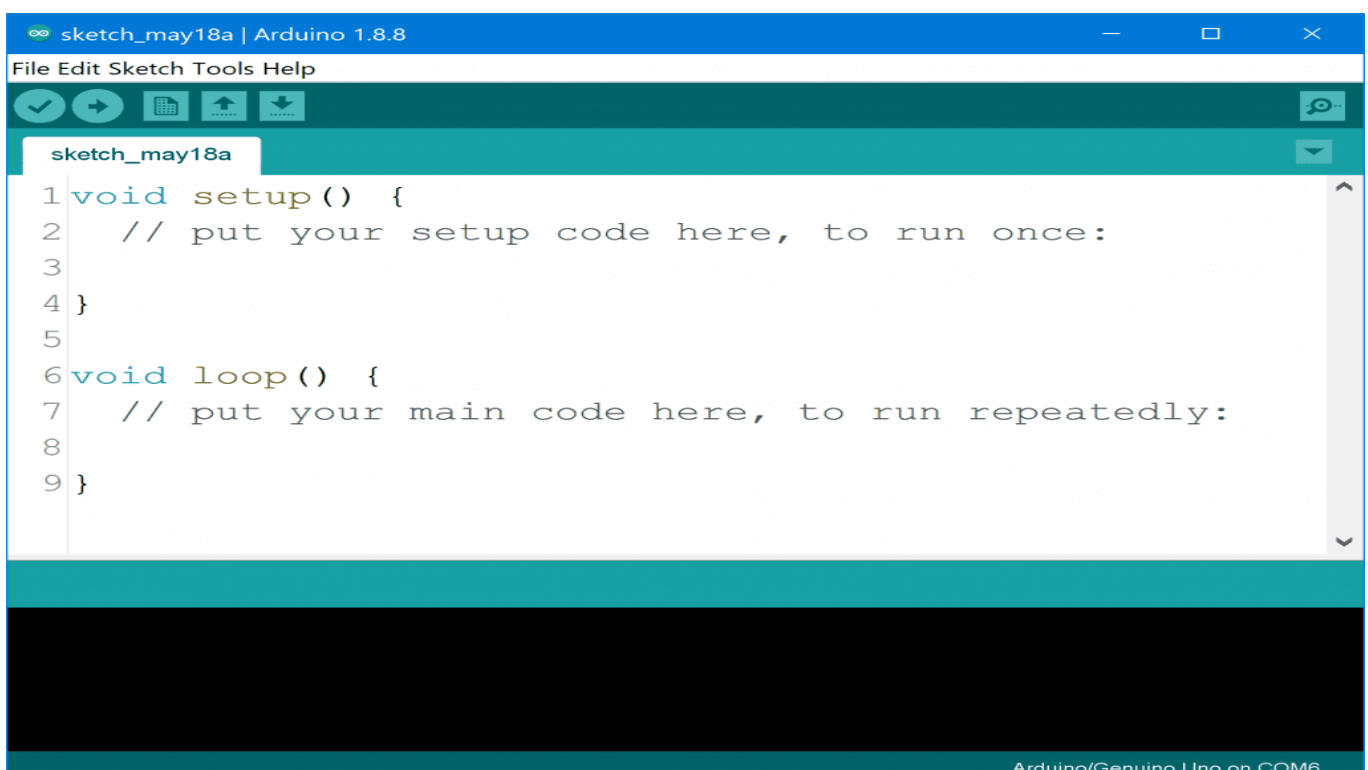


A solderless breadboard is as simple a device as it gets. It's the most used device when creating temporary circuits. It is called solderless because no soldering is required, you can just plug in the components. A component can easily be removed from a breadboard if you make a mistake, or when starting a new project. This makes it great for both beginners who are just starting to learn about electronics and seasoned professionals.

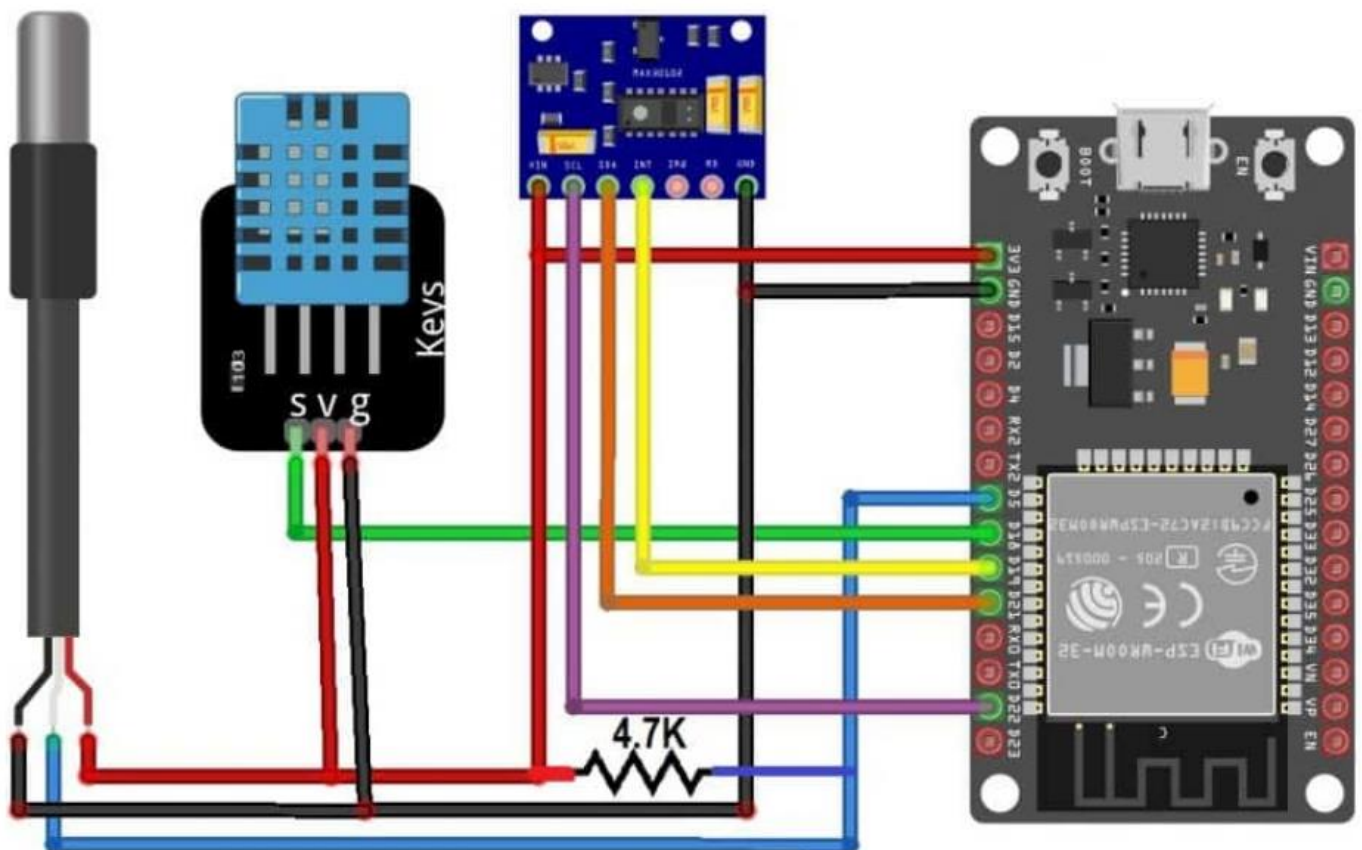
ARDUINO IDE SOFTWARE

The Arduino IDE website serves as the central platform for the Arduino Integrated Development Environment. The site is a go-to resource for programming Arduino microcontrollers, providing a user-friendly interface and supporting materials for electronic projects.

- **Code Development:** Write, edit, and compile code for Arduino microcontrollers.
- **Uploading Code:** Upload compiled code to Arduino boards for execution.
- **Hardware Prototyping:** Rapid prototyping of electronic projects.
- **Education:** Widely used for teaching programming and electronics.
- **IoT Projects:** Used in Internet of Things applications for data collection and control.
- **Home Automation:** Employed in DIY projects to control lights and appliances.
- **Robotics:** Used for programming robotic movements and sensing.
- **Interactive Art:** Enables the creation of interactive art installations.
- **Scientific Experiments:** Used in research to control and monitor sensors.
- **DIY Electronics:** Popular for various electronics projects and automation.
- **Open Source:** Arduino IDE is open-source, fostering a large community of users.



CIRCUIT CONNECTIONS:



The whole sensor operates at 3.3V VCC. Thus, attach their 3.3V power supply to their VCC. Attach the GND to the GND. Since the MAX30100 is an I2C sensor, connect GPIO21 and GPIO22 to its SDA and SCL pins. Attach its INT pin to the ESP32's GPIO19. The GPIO18 of the ESP32 is linked to the output pin of the DHT11. In a similar manner, GPIO5 of the ESP32 is linked to the output pin of the DS18B20. The output pin of the DS18B20 is linked to the VCC pin via a 4.7K pull-up resistor.

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 18

#define DS18B20 5

#define REPORTING_PERIOD_MS 1000

float temperature, humidity, BPM, SpO2, bodytemperature;

const char* ssid = "gitam.edu";

const char* password = "Gitam$$123";

dht11 DHT;

PulseOximeter pox;

uint32_t tsLastReport = 0;

OneWire oneWire(DS18B20);

DallasTemperature sensors(&oneWire);

WebServer server(80);

void onBeatDetected()

{

    Serial.println("Beat!");

}

void setup() {
```

```
Serial.begin(115200);

pinMode(19, OUTPUT);

delay(100);

Serial.println("Connecting to ");

Serial.println(ssid);

WiFi.begin(gitam.edu,Gitam$$123);

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);  
}  
  
void loop() {  
  
    server.handleClient();  
  
    pox.update();  
  
    sensors.requestTemperatures();  
  
    int chk = DHT.read11(DHT11_PIN);  
  
    temperature = DHT.temperature;  
  
    humidity = DHT.humidity;  
  
    BPM = pox.getHeartRate();  
  
    SpO2 = pox.getSpO2();  
  
    bodytemperature = sensors.getTempCByIndex(0);  
  
    if (millis() - tsLastReport > REPORTING_PERIOD_MS)  
    {  
  
        Serial.print("Room Temperature: ");  
  
        Serial.print(DHT.temperature);  
  
        Serial.println("°C");  
  
        Serial.print("Room Humidity: ");  
  
        Serial.print(DHT.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));

}

void handle_NotFound(){

    server.send(404, "text/plain", "Not found");

}

String SendHTML(float temperature,float humidity,float BPM,float SpO2, float
bodytemperature){

    String ptr = "<!DOCTYPE html>";

    ptr += "<html>";

    ptr += "<head>";

    ptr += "<title>ESP32 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>ESP32 Patient Health Monitoring</h1>";
```

```
ptr += "<h3>www.how2electronics.com</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-
2h4.084V6.063C16.06,2.715,13.345,0,9.996,0H9.313C5.965,0,3.252,2.715,3.252,6.063v
30.982";
```

```
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
+="'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 += "<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='M14.618,0C14.618,0,17.95,0,26.022C0,34.096,6.544,40.64,14.618,40.64s14.617-
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-
0.202C14.39,0.279,9.936,2.256,6.544,5.385l3.576,3.577c0.391,0.391,0.391,1.024,0,1.41
4";
```

```
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='M11.976,8.82v2h4.084V6.063C16.06,2.715,13.345,0,9.996,0H9.313C5.965,0,3.252,2
.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+="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'>Body Temperature</div>";
ptr += "<div class='side-by-side reading'>";
ptr += (int)bodytemperature;
ptr += "<span class='superscript'>&deg;C</span></div>";
ptr += "</div>";
ptr += "</div>";
ptr += "</body>"; ptr += "</html>"; return ptr;

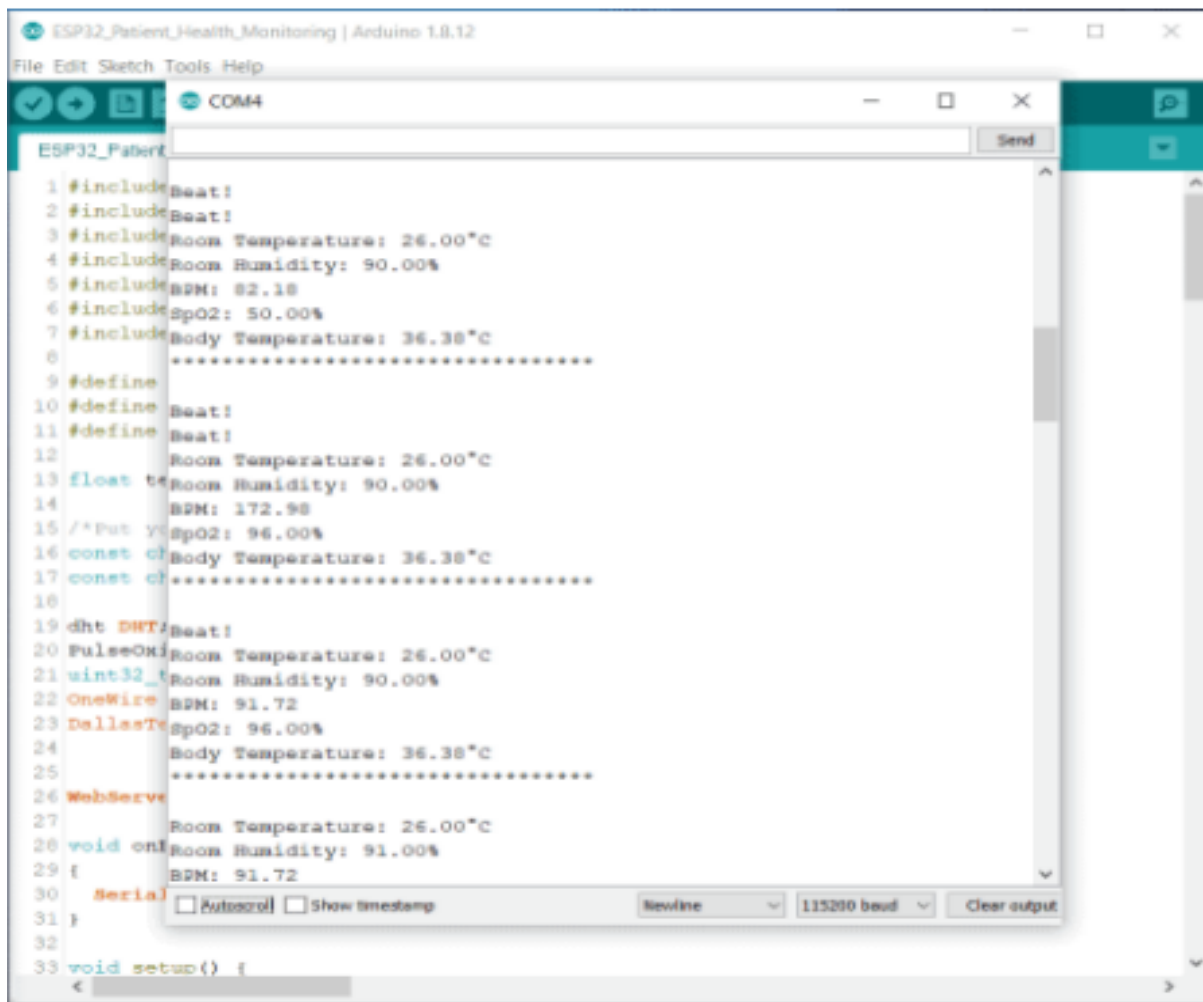
```

RESULTS

Once the code is uploaded, you can open the serial monitor. The ESP32 will try to connect to a network. Once connected, it will display the IP Address.

```
COM4
load:0x40080400,len:6352
entry 0x400806b8
Connecting to
BYNARK
...
WiFi connected..!
Got IP: 192.168.0.185
HTTP server started
Initializing pulse oximeter..SUCCESS
Room Temperature: 24.00°C
Room Humidity: 30.00%
BPM: 0.00
SpO2: 0.00%
Body Temperature: 25.62°C
*****

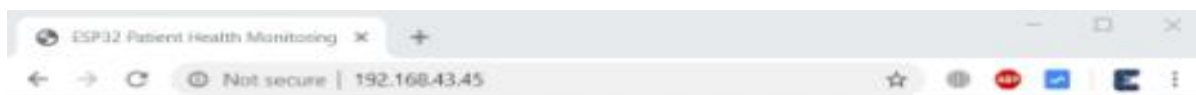
Room Temperature: 24.00°C
Room Humidity: 30.00%
BPM: 0.00
SpO2: 0.00%
Body Temperature: 25.50°C
*****
```



```

1 #include <Beat.h>
2 #include <Beat.h>
3 #include <Room_Temperature.h>
4 #include <Room_Humidity.h>
5 #include <BPM.h>
6 #include <SpO2.h>
7 #include <Body_Temperature.h>
8
9 #define
10 #define
11 #define
12 #define
13 float t
14 #define
15 /** Put Y
16 const ch
17 const ch
18
19 dht DHT
20 PulseOK
21 uint32_t
22 OneWire
23 DallasTe
24
25 WebServe
26
27 Room Temperature: 26.00°C
28 Room Humidity: 90.00%
29 BPM: 82.18
30 SpO2: 96.00%
31 Body Temperature: 36.38°C
32
33 void setup() {

```



ESP32 Patient Health Monitoring

www.how2electronics.com





The fundamental element of people's needs is health. Humans face a haul of surprising death and plenty of diseases because of varied diseases that are a result of lack of treatment to the patients at right time. The main objective of this project is to develop a reliable sensible patient health observance system victimization IoT so the attention professionals will monitor their patients.

Our goal is to implement a smart patient health monitoring system that can monitor the heartbeat and body temperature. The smart patient health care monitoring system developed by us has numerous applications.

The Internet of Things is considered now as one of the feasible solutions for any remote value tracking especially in the field of health monitoring. It facilitates that the individual prosperity parameter data is secured inside the cloud, stays in the hospital are reduced for conventional routine examinations and most important that the health can be monitored and disease diagnosed by any doctor at any distance.

Reference

Newspaper articles, Google (information, pictures and equipment description) and publications.