IoT Based Real Time Patient Health Monitoring System

A report submitted in the partial fulfillment of the requirements for the course

on

ETE 3200: Project Design Based on Communication Systems

by

Saeedul Haque

Roll No.1904049

to the

Department of Electronics & Telecommunication Engineering

Faculty of Electrical & Computer Engineering

Rajshahi University of Engineering & Technology

Heaven's Light is Our Guide
Electronics & Telecommunication Engineering
Rajshahi University of Engineering & Technology

CERTIFICATE

This is to certify that the project entitled " IoT Based Real Time Patient Health Monitoring System" by Saeedul Haque, Roll No. 1904049 has been carried out under my supervision. To the best of my knowledge, this project work is an original one and was not submitted anywhere for any degree or diploma.

Supervisor

.....
(Md. Aslam Mollah)
Assistant Professor
Dept. of Electronics & Telecommunication Engineering
Rajshahi University of Engineering & Technology

Acknowledgement

This project has been submitted to the Department of Electronics and Telecommunication Engineering of Rajshahi University of Engineering & Technology (RUET), Rajshahi-6204, Bangladesh. Project title regards to "IoT Based Real Time Patient Health Monitoring System".

First and foremost, I offer my sincere gratitude and indebtedness to my project supervisor, Md. Aslam Mollah, Assistant Professor, Department of Electronics & Telecommunication Engineering, who has supported me throughout my project with his patience and knowledge. I shall ever remain grateful to him for his valuable guidance, advice, encouragement, and cordial and amiable contribution to my project.

I wish to thank once again Professor Dr. Mst. Fateha Samad, the head of the Department of Electronics & Telecommunication Engineering for her support and encouragement and also for providing all kinds of laboratory facilities.

Abstract

IoT based patient health monitoring system is a generic term given to any medical equipment that has internet capability and can measure one or more health data of a patient who is connected to the device such as heartbeat, body temperature, blood pressure etc. The equipment can record, transmit and alert if there is any abrupt change in the patient's health. By using this device, a doctor can monitor patient's heath in real time from anywhere. Based on the received data, the heath expert can prescribe a best treatment or take an immediate action in case of an emergency. The system can be designed to monitor various health parameters such as heart rate, blood pressure, body temperature, oxygen level, and many others. The data collected from the sensors can be transmitted to a webserver through which it can be monitored in real time. The system is also designed to send alerts or notifications to the individual or caregiver in case of abnormal readings or health issues. For instance, if the body temperature or heart beat reading goes beyond a certain threshold level, the system can send a notification to the individual and also to the healthcare provider for immediate attention.

Table of Contents

Chapter 1: Introduction

Introduction 1

1.1 Objectives 1

Chapter 2: Background and Preliminaries

2.1 Block Diagram 2

2.2 Component Used 3

2.3 Description of component

2.3.1 LM35 Temperature Sensor 3

2.3.2 DHT11 Temperature and Humidity Sensor 5

2.3.3 MAX 30100 Pulse Oximeter and Heart Rate Sensor 6

2.3.4 ESP-32 Board 8

2.3.5 20x4 LCD Display 10

2.3.6 I2C Module 11

2.3.7 Center Tapped Transformer 12

2.3.8 Diode 13

2.3.9 Capacitor 13

2.3.10 Voltage Regulator IC 14

2.4 Software Used 15

2.5 Description of Software

2.5.1 Arduino IDE 15

2.5.2 ExpressPCB 16

2.5.3 IFTTT 16

2.5.4 Local Webserver 16

Chapter 3: Project Details

3.1 Working Principle 17

3.2 Circuit Diagram 18

3.3 PCB Design 18

3.4 Code 19

3.5 Result 36

Chapter 4: Conclusion

4.1 Advantage 39

4.2 Disadvantage 39

4.3 Application 39

4.4 Conclusion 40

Chapter 1

Introduction

Over the last 15 years, interest in various health monitoring apps has rapidly increased. Healthcare has been transformed by the sector's technological revolution in recent years. a few of decades as a result of an IoT-based wireless patient monitoring system. A range of contemporary innovations, including mobile applications, smart gadgets, biosensors, wearable technology, home virtual assistants, and blockchain-based electronic medical record systems are ushering in a new age in healthcare. Because of advancements in information and communication technology, as well as an increase in the number of people reaching retirement age, there has been a rush in research on telehealth systems that can monitor persons in their homes and identify or forecast health concerns. Cloud-based wireless technologies will undoubtedly improve data storage and visualization capabilities. Telemedicine care standard and remove communication gaps. Continuous data gathering is especially important for chronic illnesses. Continuous manual data measurements are prone to error, which might result in the wrong drug prescription. Physician turnover is another factor contributing to the high death rate. However, if doctors lack sufficient patient information, they will prescribe incorrectly. We will be able to tackle a range of difficulties if we avoid manual data collecting and receive this data digitally in a short period of time. Some chronic conditions can be lethal if not regularly monitored.

1.1 Objectives

For this project some objectives were taken into account in order to monitor the physical condition of a patient in real time without the need of extra caretaker. The main objectives of this project are,

- Creating a cheap and affordable health monitoring system.
- Reduce the requirement of persons needed for monitoring the patient.
- Be able to monitor the patient's health in real time.

Chapter 2

Background and Preliminaries

2.1 Block diagram

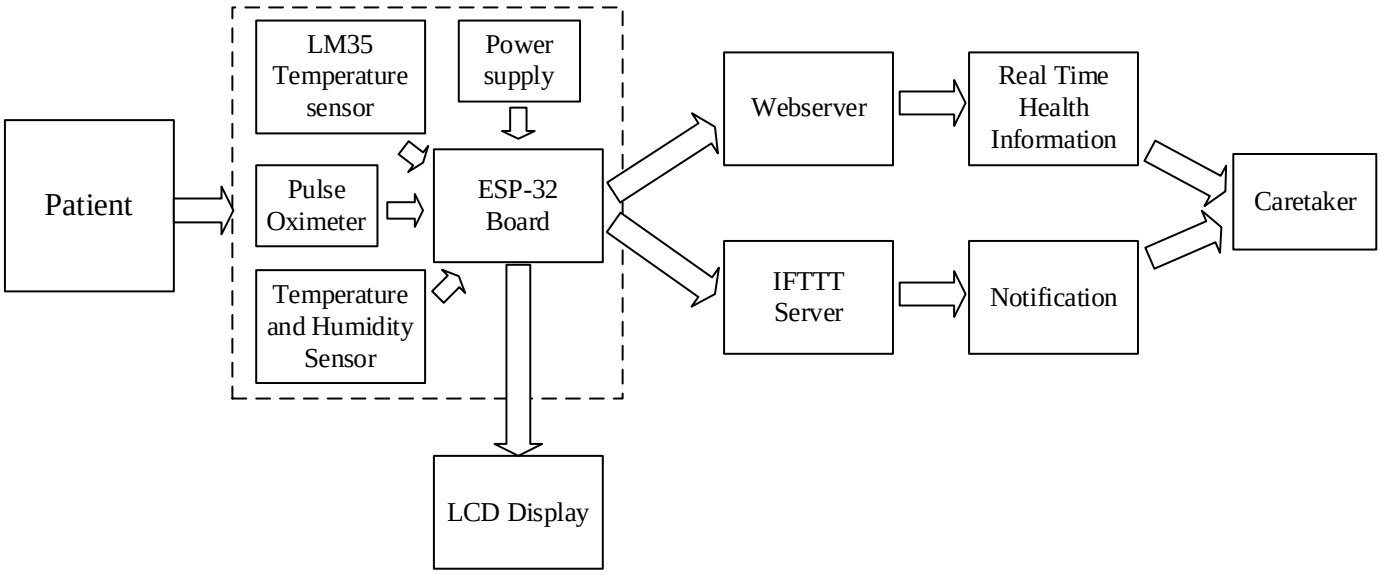


Figure 2.1: Block Diagram of the system

2.2 Component Used

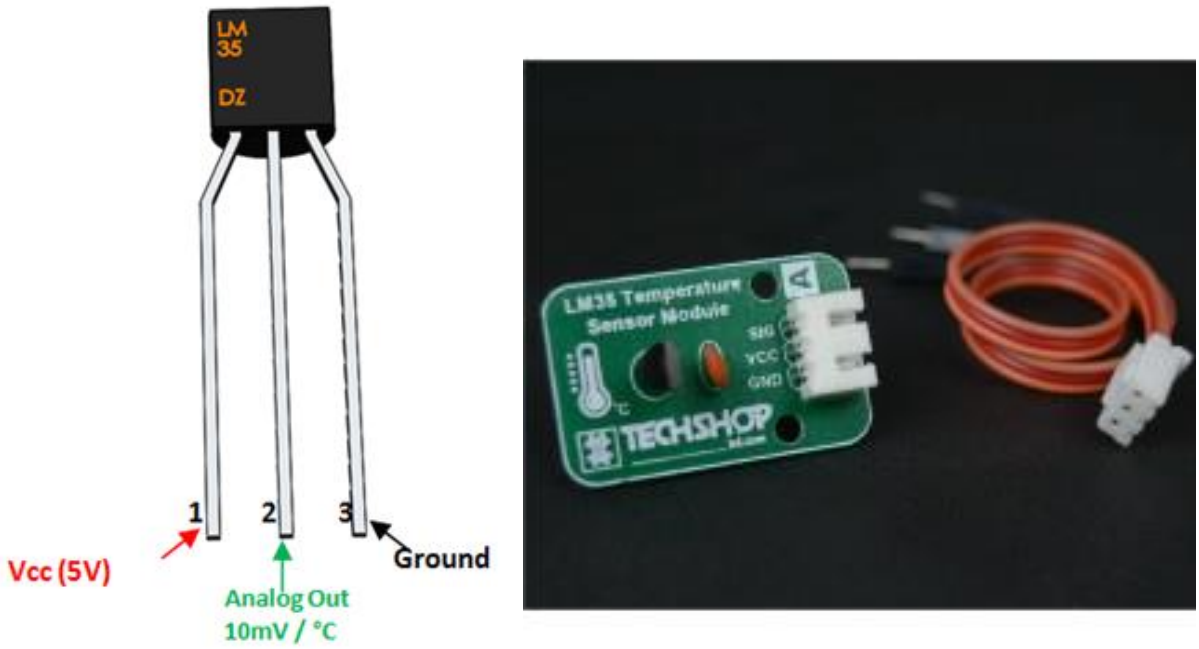
Component	Quantity
LM35 Temperature Sensor	01
DHT11 Temperature and Humidity Sensor	01
MAX30100 Pulse Oximeter Heart Rate Sensor	01
Center Tapped Transformer	01
ESP-32 board	01
20x4 LCD Display	01
I2C module	01
Capacitor	02
Diode	04
Voltage Regulator IC	03
PVC board	01

2.3 Description of component

2.3.1 LM35 Temperature Sensor

LM35 is a precession Integrated Circuit Temperature sensor, whose output voltage varies, based on the temperature around it. It is a small and cheap IC which can be used to measure temperature anywhere between -55°C to 150°C. It can easily be interfaced with any Microcontroller that has ADC function or any development platform like Arduino, ESP etc. The IC can be powered by applying a regulated voltage like +5V (VS) to the input pin and connected the ground pin to the ground of the circuit. Now, the temperature can be measured in form of voltage. If the temperature is 0°C, then the output voltage will also be 0V. There will be rise of 0.01V (10mV) for every degree Celsius rise in temperature. The voltage can be converted into temperature using the below formulae,

$$V_{out}=10mV/^{o}C * T$$



(a) (b)

Figure 2.2: LM35 Temperature Sensor (a) IC pinout (b) Module

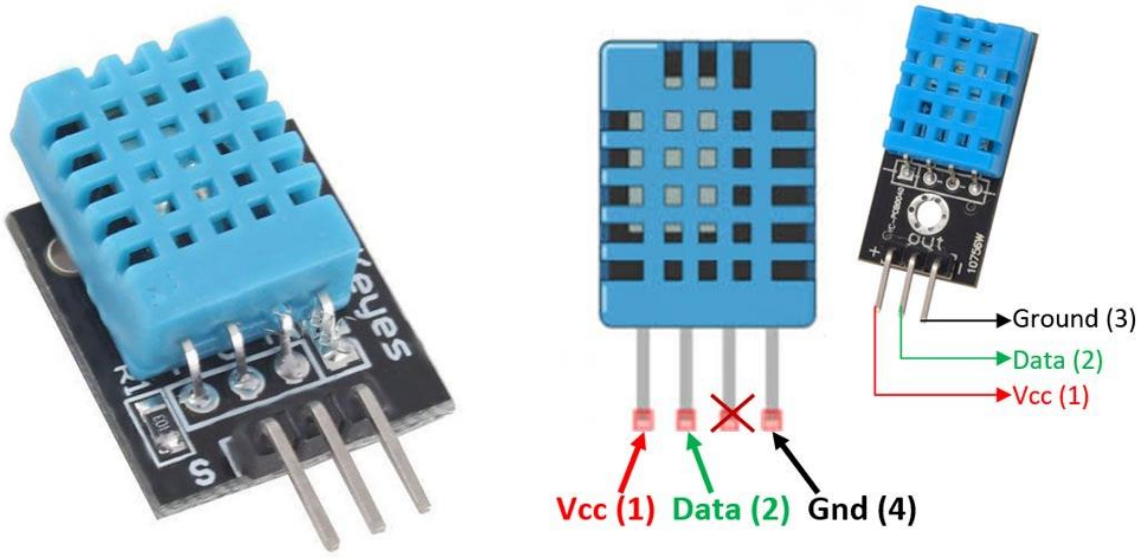
Features and Specifications:

- Minimum and Maximum Input Voltage is 35V and -2V respectively. Typically, 5V.
- Can measure temperature ranging from -55°C to 150°C
- Output voltage is directly proportional (Linear) to temperature (i.e.) there will be a rise of 10mV (0.01V) for every 1°C rise in temperature.
- ±0.5°C Accuracy
- Drain current is less than 60uA
- Low-cost temperature sensor
- Small and hence suitable for remote applications
- Available in TO-92, TO-220, TO-CAN and SOIC package

Pin Number	Pin Name	Description
1	V_{cc}	Input voltage is +5V for typical applications
2	Analog Out	There will be increase in 10mV for raise of every 1°C. Can range from -1V(-55°C) to 6V(150°C)
3	Ground	Connected to ground of circuit

2.3.2 DHT11 Temperature and Humidity Sensor

The DHT11 is a commonly used Temperature and humidity sensor. This sensor consists of a capacitive humidity sensor and a thermistor for measuring temperature. It provides digital output for both temperature and humidity readings. It also comes with a dedicated NTC to measure temperature and an 8-bit microcontroller to output the values of temperature and humidity as serial data. It is factory calibrated and hence easy to interface with other microcontrollers. The sensor can measure temperature from 0°C to 50°C and humidity from 20% to 90% with an accuracy of ±1°C and ±1%.



(a) (b)

Figure 2.3: DHT11 Temperature and Humidity Sensor (a) Module (b) IC pinout

Features and Specifications:

- Operating Voltage: 3.5V to 5.5V
- Operating current: 0.3mA (measuring) 60uA (standby)
- Output: Serial data
- Temperature Range: 0°C to 50°C
- Humidity Range: 20% to 90%
- Resolution: Temperature and Humidity both are 16-bit
- Accuracy: ±1°C and ±1%

Pin Number	Pin Name	Description
1	V_{cc}	Power supply 3.5V to 5.5V

2	Data	Outputs both Temperature and Humidity through serial Data
3	Ground	Connected to the ground of the circuit

2.3.3 MAX30100 Pulse Oximeter and Heart Rate Sensor

MAX30100 is a multipurpose sensor used for multiple applications. It is a heart rate monitoring sensor along with a pulse oximeter. The sensor comprises two light emitting diodes or LED, a photodetector, and a series of low noise signal processing devices to detect heart rate and to perform pulse oximetry.

The sensor consists of a pair of Light-emitting diode which emits monochromatic red light at a wavelength of 660nm and infrared light at a wavelength of 940 nm. These wavelengths are particularly chosen as at this wavelength oxygenated and deoxygenated hemoglobin have very different absorption properties. As shown in the graph below, it can be seen that there is a difference between HbO2 (oxygenated Hb) and Hb (deoxygenated Hb) when subjected to these specific wavelengths.

There are two parts to the sensor, an emitting diode, and a photoreceiver. As the photodiode emits the light, it falls over the finger which has to be placed steadily. The light emitted gets absorbed by the oxygenated blood and the rest of the light is reflected through the finger and falls over the detector whose output data is then processed and read through a microcontroller.

Below is the functional block for the MAX30100 module. The module consists of two LEDs (IR and RED) both of specific wavelengths, along with a photodetector to detect the received light.

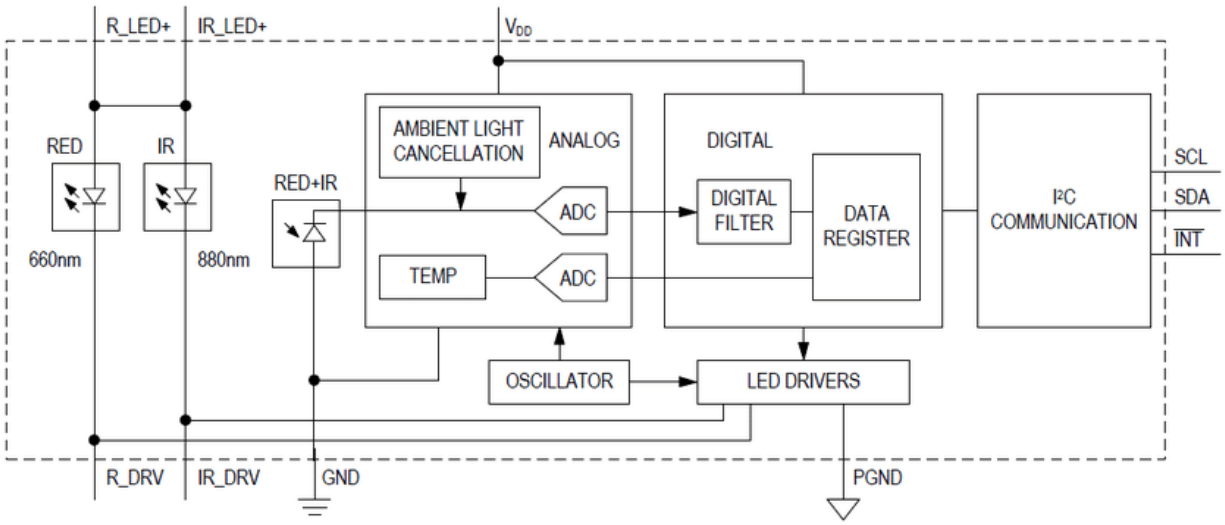
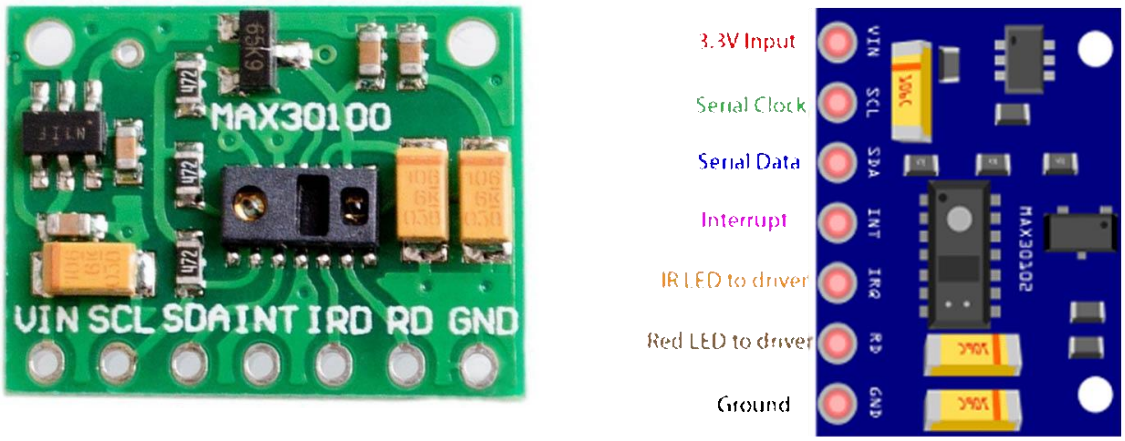


Figure 2.4: MAX30100 functional block diagram

The output from the photodiode is sent to the analog-to-digital converter from which the digital data is sent from a filter to the digital data register. The data can be collected from the register and can be sent to the microcontroller following the I2C communication protocol.



(a) (b)

Figure 2.5: MAX30100 Pulse Oximeter and Heart Rate Sensor (a) Module (b) IC pinout

Features and Specifications:

- Operating Voltage - 1.8V to 3.3V
- Input Current - 20mA
- Integrated Ambient Light Cancellation
- High Sample Rate Capability
- Fast Data Output Capability

Pin Type	Pin Function
VIN	Voltage Input
SCL	I2C - Serial Clock
SDA	I2C - Serial Data
INT	Active low interrupt
IRD	IR LED Cathode and LED Driver Connection Point(Leave floating in the circuit)
RD	Red LED Cathode and LED Driver Connection Point(Leave floating in the circuit)
GND	Ground pin

2.3.4 ESP-32 Board

The Esp32 DevKit v1 is one of the development boards created to evaluate the ESP-WROOM-32 module. It is based on the ESP-32 Microcontroller that boasts Wi-Fi, Bluetooth, Ethernet and Low Power support all in a single chip. ESP32 is already integrated antenna and RF balun, power amplifier, low-noise amplifiers, filters, and power management module. The entire solution takes up the least amount of printed circuit board area. This board is used with 2.4 GHz dual-mode Wi-Fi and Bluetooth chips by TSMC 40nm low power technology, power and RF properties best, which is safe, reliable, and scalable to a variety of applications. The internal flash of the ESP32 module is organized in a single flash area with pages of 4096 bytes each. The flash starts at address 0x00000, but many areas are reserved for Esp32 IDF SDK. There exist two different layouts based on the presence of BLE support. Power to the Esp32 DevKit v1 is supplied via the on-board USB Micro B connector or directly via the “VIN” pin. The power source is selected automatically. The device can operate on an external supply of 6 to 20 volts. If using more than 12V, the voltage regulator may overheat and damage the device. The recommended range is 7 to 12

volts. The Esp32 DevKit v1 comes with a serial-to-usb chip on board that allows programming and opening the UART of the ESP32 module.

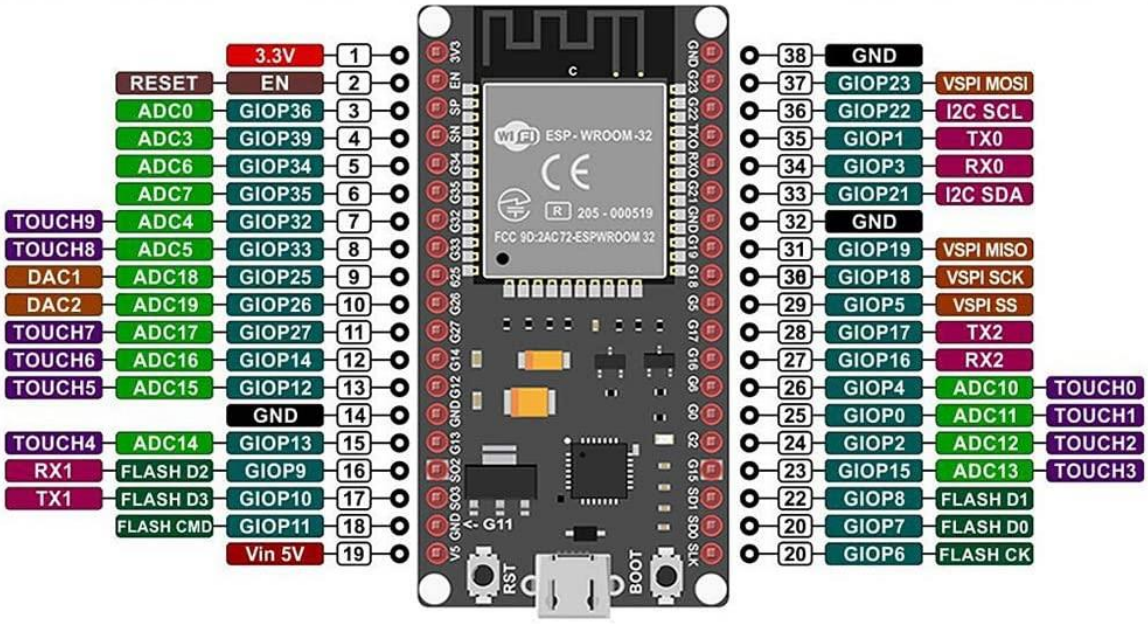


Figure 2.6: ESP-32 Devkit board

Features and Specifications:

- Microcontroller: Tensilica 32-bit Single-/Dual-core CPU Xtensa LX6
- Operating Voltage: 3.3V
- Input Voltage: 7-12V
- Digital I/O Pins (DIO): 25
- Analog Input Pins (ADC): 6
- Analog Outputs Pins (DAC): 2
- UARTs: 3
- SPIs: 2
- I2Cs: 3
- Flash Memory: 4 MB
- SRAM: 520 KB
- Clock Speed: 240 Mhz
- Wi-Fi: IEEE 802.11 b/g/n/e/i:
- Integrated TR switch, balun, LNA, power amplifier and matching network
- WEP or WPA/WPA2 authentication, or open networks

2.3.5 20x4 LCD Display

A 20x4 LCD display refers to a liquid crystal display with the capacity to show 20 characters per line and 4 lines in total. It's a commonly used display in various electronic devices due to its versatility and readability. The display operates by controlling liquid crystal molecules to either allow light to pass through or block it, creating characters or images. The display has a built-in controller that manages the display of characters. The controller interprets data sent to it and converts it into the appropriate signals to display characters on the screen. Each character position on the display has a unique address. For a 20x4 display, there are 80 such positions. The display is usually connected to a microcontroller or a computer via a communication protocol like I2C, SPI, or parallel interface. This connection allows the microcontroller to send data (characters) to be displayed. The microcontroller sends commands and data to the display through the communication interface. The display has its own character set built into its memory, allowing it to show predefined characters. Custom characters can also be created and stored in the display's memory for specialized applications. Many LCD displays come with a backlight to improve readability in different lighting conditions. The backlight is usually an LED array that can be controlled separately.

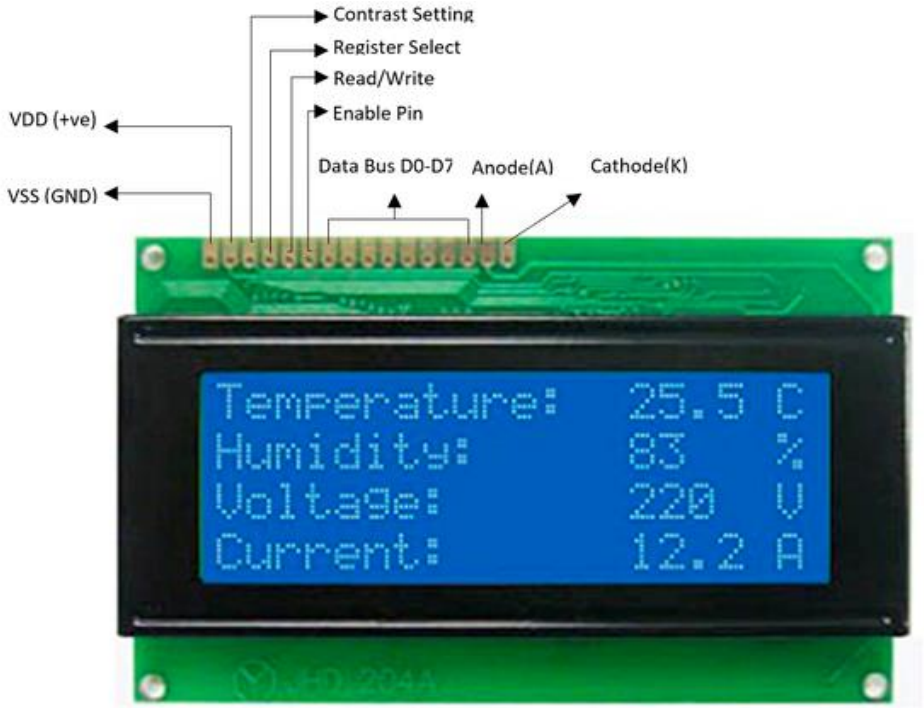


Figure 2.7: 20x4 LCD Display

Features and Specifications:

- Display: 80 characters at a time
- Cursor: 5x8 (40) dots
- Assembled controller of RW1063
- Input Voltage: 3.3V, 5V
- Duty cycle: 1/16

2.3.6 I2C Module

The I2C (Inter Integrated Circuit) standard was created to provide simple ways for digital information to be transformed between sensors and micro controllers. I2C has the advantage that it only needs two signal connections (clock and data) to microcontroller. One device on the I2C bus is considered the master (or primary) device. Its job is to coordinate the transfer of information between the other devices (slaves, secondary components) that are attached. Slave devices are identified by their address number. Each one must have a unique address. Some I2C devices have a fixed address while others allow to configure their address by setting pins high or low or by sending initialization commands.

I2C Module has an inbuilt PCF8574 I2C chip that converts I2C serial data to parallel data for the LCD display. These modules are currently supplied with a default I2C address of either Ox27 or 0x3F.

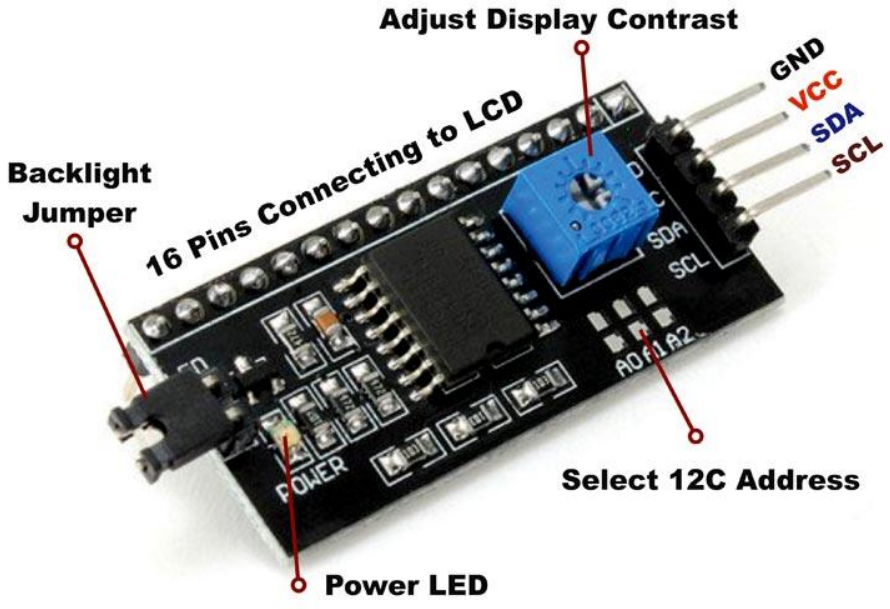


Figure 2.8: I2C module

- Operating Voltage: 5V
- Backlight and Contrast is adjusted by potentiometer
- Serial I2C control of LCD display using PCF8574
- Comes with 2 IIC interface, which can be connected by Dupont Line or IIC dedicated cable
- Compatible for 20x4 LCD
- This is another great IIC/I2C/TWI/SPI Serial Interface
- With this I2C interface module, one will be able to realize data display via only 2 wires.

2.3.7 Center Tapped Transformer

A center-tapped transformer also known as two phase three wire transformer is normally used for rectifier circuits. When a digital project has to work with AC mains a Transformer is used to step-down the voltage and then convert it to DC by using a rectifier circuit. In a center-tapped transformer the peak inverse voltage is twice as in bridge rectifier hence this transformer is commonly used in full wave rectifier circuits.

The operation and theory behind a Center tapped transformer is very similar to a normal secondary transformer. A primary voltage will be induced in the primary coil and due to magnetic induction, the voltage will be transferred to the secondary coil. Here in the secondary coil of a center tapped transformer, there will be an additional wire which will be placed exactly at the center of the secondary coil; hence the voltage here will always be zero.

If the zero potential wire is combined with either first or second wire, a voltage of 12V AC can be found. If this wire is ignored and voltage across first and second wire is considered then a voltage of 24V AC will be found. This feature is very useful for the function of a full wave rectifier.

By including the transformer as the power source of the project, it will be possible to get supply straight from the AC main without the need of any adapters. Using a rectifier, the AC will be converted to DC and the project will be able to operate.

Figure 2.9: Center Tapped Transformer

Features and Specifications:

- Step-down Centre tapped Transformer
- Input Voltage: 220V AC at 50Hz
- Output Voltage: 24V, 12V or 0V
- Output Current: 1A
- Vertical mount type
- Low cost and small package

2.3.8 Diode

A diode is a device which allows current flow through only one direction. That is the current should always flow from the Anode to Cathode. For 1N4007 Diode, the maximum current carrying capacity is 1A but it can withstand peaks up to 30A. Hence, we can use this in circuits that are designed for less than 1A. The reverse current is 5uA which is negligible. The power dissipation of this diode is 3W.

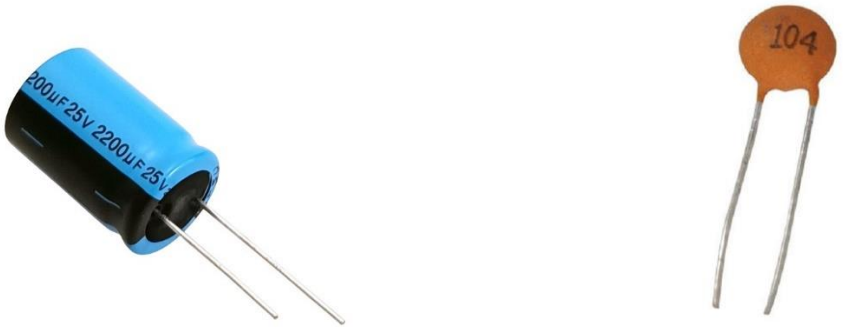
(a) (b)

Figure 2.10: Diode (a) Component (b) Pinout

2.3.9 Capacitor

Electrolytic Capacitors are the most recognizable types of capacitors. They come in distinctive metal cans with a plastic sheath, with clearly stated voltage and capacitance ratings and a white band to indicate the cathode. The name comes from the fact that, like mentioned above, the 'plates' are made of chemically etched aluminum foil. The etching process makes the aluminum and increases its surface area greatly, hence increasing capacitance. The dielectric is a thin layer of aluminum oxide. These capacitors are filled with oil that acts like an electrolyte, hence the name. Electrolytic capacitors are polarized because of their internal construction. They have large capacitance compared to other members of the capacitor family, but much lower voltages

There are also capacitors with a ceramic dielectric. Since the breakdown limit for the ceramic dielectric is quite high, it is expected to see ceramic capacitors with crazy breakdown voltages like 10kV. However, capacitance tend to be low, in the range of picofarads to a few tens of microfarads.



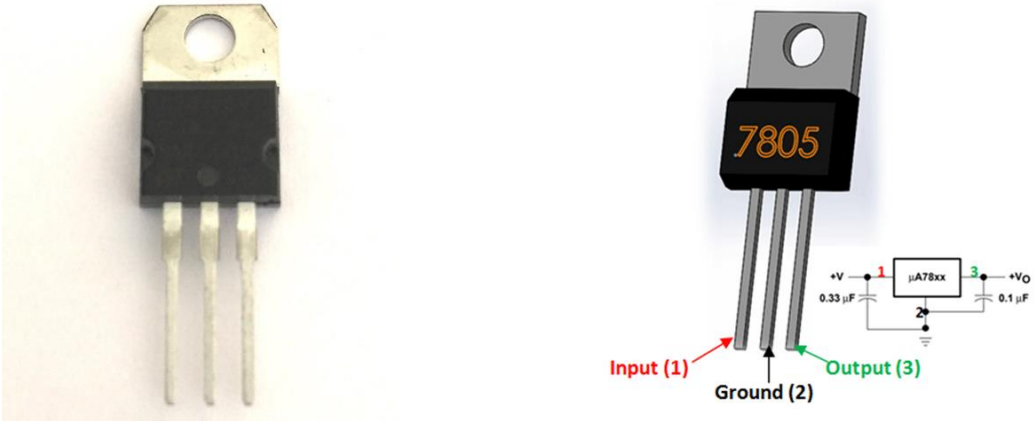
(a) (b)

Figure 2.11: Capacitor (a) Electrolytic (b) Ceramic

2.3.10 Voltage Regulator IC

Voltage regulators are very common in electronic circuits. They provide a constant output voltage for a varied input voltage. For example, the 7805 IC is an iconic regulator IC that finds its application in most of the projects. The name 7805 signifies two meaning, "78" means that it is a positive voltage regulator and "05" means that it provides 5V as output. So, the 7805 will provide a +5V output voltage. The output

current of this IC can go up to 1.5A. But, the IC suffers from heavy heat loss hence a heat sink is usually used for projects that consume more current.



(a) (b)

Figure 2.10: Voltage Regulator (a) IC (b) Pinout

2.4 Software Used

- Arduino IDE
- ExpressPCB
- IFTTT
- Local Webserver

2.5 Description of software

2.5.1 Arduino IDE

The Arduino IDE is an open-source software, which is used to write and upload code to the Arduino boards. The IDE application is suitable for different operating systems such as Windows, Mac OS X, and Linux. It supports the programming languages C and C++. Here, IDE stands for Integrated Development Environment. The program or code written in the Arduino IDE is often called as sketching. We need to connect the Genuino and Arduino board with the IDE to upload the sketch written in the Arduino IDE software. The sketch is saved with the extension '.ino.'

2.5.2 ExpressPCB

ExpressPCB is CAD software that is used to create layouts for printed circuit boards (PCB). The CAD is divided into the more helpful category such as ExpressSCH and ExpressPCB. ExpressSCH is used to design the schematics. ExpressPCB to create a PCB design. It is linked to the schematic.

2.5.3 IFTTT

IFTTT stands for "If This, Then That." It's a web-based service that allows users to create chains of simple conditional statements, called applets. These applets are triggered based on changes that occur within web services, apps, or devices.

IFTTT supports a vast array of services, from social media platforms like Facebook and Twitter to smart home devices, productivity tools, and more. It simplifies the automation of tasks by connecting different online services and devices through predefined triggers and actions, making it easier for users to set up personalized automations

2.5.4 Local Webserver

A local web server refers to a software application that allows the user to host and serve web content locally on the user’s computer. It's a piece of software that enables the user to run websites, web applications, or web services on the user’s own machine. Examples of local web server software include Apache, Nginx, Microsoft Internet Information Services (IIS), and more. These servers simulate the functionalities of a live web server but on a smaller scale, usually for development, testing, or personal use purposes.

Chapter 3

Project Details

3.1 Working Principle

The MAX30100 pulse oximeter operates based on the principle of photoplethysmography (PPG). It emits light into the skin and measures the amount of light absorbed by hemoglobin, providing information about heart rate and blood oxygen saturation levels. The DHT11 sensor is a digital temperature and humidity sensor that utilizes a capacitive humidity sensor and a thermistor to measure the surrounding air conditions. The LM35 temperature sensor provides an analog voltage output proportional to the Celsius temperature. It's used to measure the body temperature. These sensors are connected to the ESP-32 microcontroller. The microcontroller processes the sensor data and interfaces with an LCD display to show real-time readings.

Additionally, a local web server is set up on the microcontroller to create a user-friendly interface accessible through a web browser. This allows users to monitor their health parameters on devices connected to the same local network. The microcontroller continuously collects data from the sensors, updates the LCD display, and hosts a web page displaying the heart rate, blood oxygen level, room temperature, humidity, and body temperature. Users can access this information by entering the microcontroller's IP address in their web browser. The LCD display serves as a convenient local interface for users to check their health metrics in real-time, while the web server enhances accessibility and allows monitoring from various devices.

When the body temperature, blood oxygen level, heart rate or any other parameter is changed to a value that goes under the threshold value set by the code, the microcontroller will send a notification through the IFTTT service to the caretaker in order to notify the degradation of the patient's health condition so that necessary actions can be taken.

3.2 Circuit Diagram

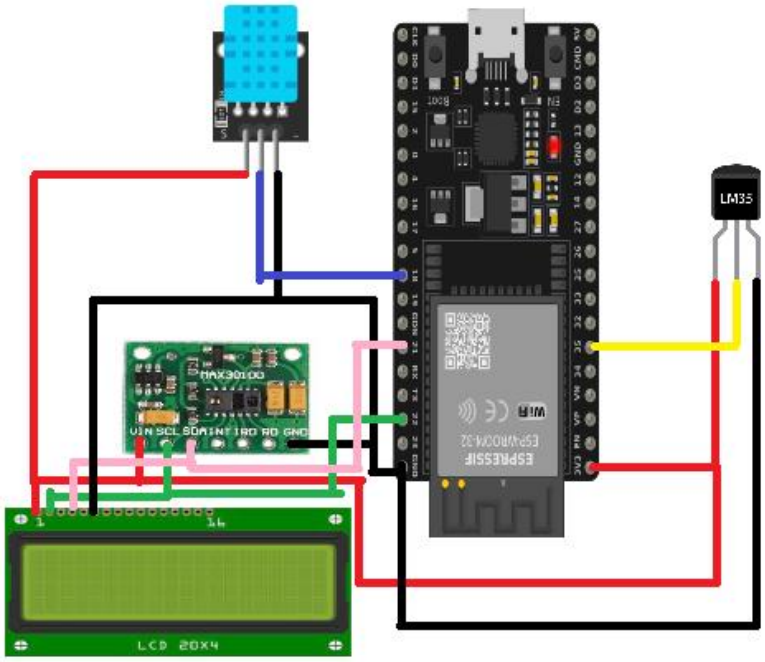


Figure 2.11: Circuit diagram

3.3 PCB Design

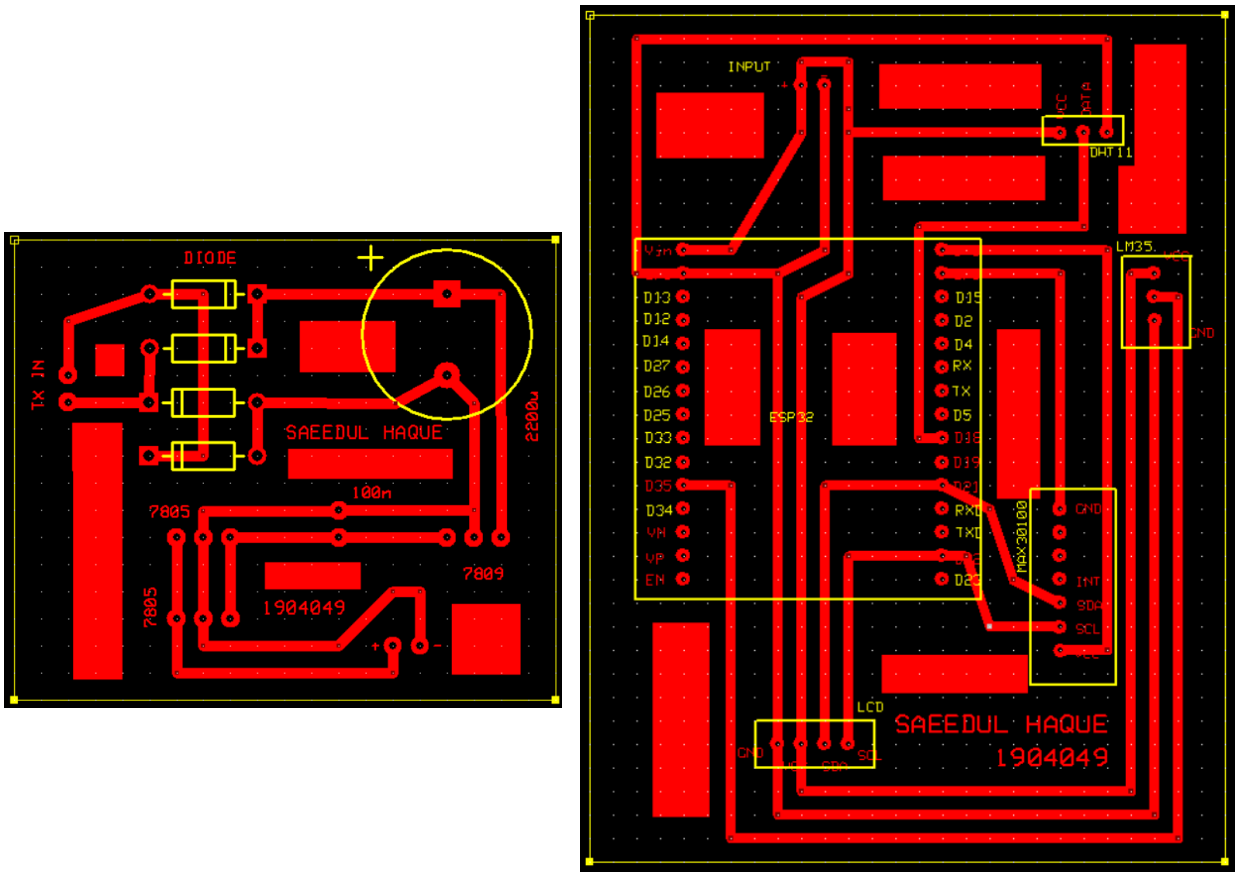


Figure 2.12: Design of the PCB

3.4 Code

```
#include <HTTPClient.h>
#include <Wire.h>
#include <WebServer.h>
#include <LiquidCrystal_I2C.h>
LiquidCrystal_I2C lcd(0x27, 20, 4); // change the value to 27 if display not working
#include "MAX30100_PulseOximeter.h"
#define REPORTING_PERIOD_MS     1000
```

```
PulseOximeter pox;
uint32_t tsLastReport = 0;

#include "DHT.h"

#define DHTTYPE DHT11
#define lm 35
#define DHTPIN 18

String server2 = "http://maker.ifttt.com";
String eventName = "Health alert";
String IFTTT_Key = "p4Eu6m4jrKpBIJQ60q-tXCstDWsUmTFfreAtAbUVxc8";
String IFTTTUrl="http://maker.ifttt.com/trigger/Health alert/with/key/p4Eu6m4jrKpBIJQ60q-tXCstDWsUmTFfreAtAbUVxc8";

int value1,value2,value3;
float temperature, humidity, BPM, SpO2,bodytemperature,b_p_m,s_p_o_2;
int lm_temp;
/*SSID & Password*/
const char* ssid = "M31";  // Enter SSID here
const char* password = "44448888";  //Enter Password here

DHT dht(DHTPIN, DHTTYPE);

WebServer server(80);
```

```
void sendDataToSheet(void)
{
  String url = server2 + "/trigger/" + eventName + "/with/key/" + IFTTT_Key + "?value1=" + String((int)value1) + "&value2="+String((int)value2) +"&value3=" + String((int)value3);
  Serial.println(url);
  //Start to send data to IFTTT
  HTTPClient http;
  Serial.print("[HTTP] begin...\n");
  http.begin(url); //HTTP

  Serial.print("[HTTP] GET...\n");
  // start connection and send HTTP header
  int httpCode = http.GET();
  // httpCode will be negative on error
  if(httpCode > 0) {
    // HTTP header has been send and Server response header has been handled
    Serial.printf("[HTTP] GET... code: %d\n", httpCode);
    // file found at server
    if(httpCode == HTTP_CODE_OK) {
      String payload = http.getString();
      Serial.println(payload);
    }
  } else {
    Serial.printf("[HTTP] GET... failed, error: %s\n", http.errorToString(httpCode).c_str());
```

```
  }
  http.end();
  loop;
}

void onBeatDetected()
{
    Serial.println("Beat!");
}
void setup() {
  Serial.begin(9600);

    //  Serial.println(F("DHTxx test!"));
    dht.begin();
    delay(500);

    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());
  delay(2000);

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

  server.begin();
  Serial.println("HTTP server started");
 lcd.init();
  byte degree[] ={
 B00111,
 B00101,
 B00111,
 B00000,
 B00000,
 B00000,
 B00000,
 B00000
};  lcd.createChar(0,degree); //we assign a number 1 to the degree sign created above
lcd.init();
```

```
  lcd.backlight();

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

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

  // Configure sensor to use 7.6mA for LED drive
  pox.setIRLedCurrent(MAX30100_LED_CURR_11MA);
  pox.setOnBeatDetectedCallback(onBeatDetected);
}

void loop() {
  server.handleClient();
  float t = dht.readTemperature();
    String Temperature_Value = String(t);
    float h = dht.readHumidity();
    String Humidity_Value = String(h);
      // Read temperature as Fahrenheit (isFahrenheit = true)
    float f = dht.readTemperature(true);
        if (isnan(h) || isnan(t)||isnan(f)) {
```

```
    Serial.println(F("Failed to read from DHT sensor!"));
    return;
  }
  temperature=t;
  humidity=h;
  lm_temp = analogRead(lm);
  float mv = ( lm_temp/1028.0)*5000 ; //1024
  float cel = mv/10 ;
  bodytemperature = ((cel-32)/9)*5;
pox.update();
  if (millis() - tsLastReport > REPORTING_PERIOD_MS)
   {
    lcd.clear();
      if(pox.getSpO2()>0)
      {
      Serial.print("Heart rate:");
      Serial.print(pox.getHeartRate());

      Serial.print("bpm / SpO2:");
      Serial.print(pox.getSpO2());
      Serial.println("%");

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

```
BPM=pox.getHeartRate();
SpO2=pox.getSpO2();

Serial.println("***********************************");
Serial.println();
lcd.setCursor(0, 0);
lcd.print("Health Monitoring");
lcd.setCursor(0,1);
lcd.print("H: ");
lcd.print(pox.getHeartRate());
lcd.print(" bpm");
lcd.setCursor(0,2);
lcd.print("SpO2: ");
lcd.print(pox.getSpO2());
lcd.print("%");
lcd.setCursor(0, 3);
lcd.print("Temp: ");
lcd.print(bodytemperature);
lcd.write((byte)0);
lcd.print("C");
}
else if (pox.getHeartRate()>0)
{

  lcd.setCursor(0,0);
```

```
        lcd.print("Finger Detected");

      }
      else
      {
        lcd.setCursor(0,0);
        lcd.print("Place Finger");
      }
      tsLastReport = millis();
   }
   if (t>32 && bodytemperature>35){

value1 = bodytemperature;
value2 = BPM;
value3 = SpO2;

Serial.print("Values are ");
Serial.print(value1);
Serial.print(' ');
Serial.print(value2);
Serial.print(' ');
Serial.println(value3);
Serial.print(' ');

sendDataToSheet();
```

```
    }
}
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 html = "<!DOCTYPE html>";
    html += "<html>";
    html += "<head>";
    html += "<title>Patient Health Monitoring</title>";
    html += "<meta name='viewport' content='width=device-width, initial-scale=1.0'>";
          // font awesome cdn
    html += "<link rel='stylesheet' href='https://cdnjs.cloudflare.com/ajax/libs/font-awesome/5.7.2/css/all.min.css'>";
           //google fonts
    html += "<link href='https://fonts.googleapis.com/css2?family=Londrina+Outline&display=swap' rel='stylesheet'>";
```

```
html += "<style>";

html += "body { center; background-color: #1b3f57; font-family: sans-serif; color: #333333; font: 14px Helvetica, sans-serif box-sizing: border-box;}";

html += "#page { margin: 20px; background-color: #7bc2e0;}";

html += ".header { padding: 5px;}";

html += ".header h1 { padding-bottom: 0.3em; color:#110930; font-size: 45px; font-weight: bold; font-family: 'Bebas Neue'; text-align: center;}";


html += ".datetime{color: #fff;background: #10101E;font-family: 'Segoe UI', sans-serif;width: 320px;height:60px; margin:auto;padding: 15px 10px; border: 3px solid #2E94E3; border-radius: 5px;}";

html += ".datetime:hover{ background: ##b6d3d4; box-shadow: 0 0 30px #8da7ba;}";

html += ".date{font-size: 15px;font-weight: 500;text-align: center;letter-spacing: 3px;}";

html += ".time{font-size: 40px;display: flex;justify-content: center;align-items: center;}";

html += ".time span:not(:last-child){position: relative;margin: 0 6px;font-weight: 500;text-align: center;letter-spacing: 3px;}";

html += ".time span:last-child{ background: #2E94E3; font-size: 20px; font-weight: 500; text-transform: uppercase;margin-top: 10px;padding: 0 5px;border-radius: 3px;}";


html += "h2 { padding-bottom: 0.2em; border-bottom: 1px solid #eee; margin: 2px; text-align: left;color:black;}";

html += ".box-full { padding: 20px; border 1px solid #ddd; border-radius: 1em 1em 1em 1em; box-shadow: 1px 7px 7px 1px rgba(0,0,0,0.4); background: #fff; margin: 20px; width: 300px;}";
```

```
html += "@media (max-width: 494px) { #page { width: inherit; margin: 5px auto; } #content { padding: 1px;} .box-full { margin: 8px 8px 12px 8px; padding: 10px; width: inherit;; float: none; } }";

html += "@media (min-width: 494px) and (max-width: 980px) { #page { width: 465px; margin 0 auto; } .box-full { width: 380px; } }";

html += "@media (min-width: 980px) { #page { width: 930px; margin: auto; } }";


html += ".sensor { margin: 12px 0px; font-size: 2.5rem;}";

html += ".sensor:hover{background: #05fff7;box-shadow: 0 0 30px #2E94E3;}";

html += ".sensor-labels { font-size: 1rem; vertical-align: middle; padding-bottom: 15px;}";

html += ".units { font-size: 1.2rem;}";

html += "hr{ height: 2px; color: green; background-color:green; border: normal;}";

html += ".footer{position: fixed; left: 0; bottom: 0px; height:55px; width: 100%; background-color: gray; color: white;text-align: center;}";


html += "</style>";


      //Ajax Code Start

html += "<script>\n";

html += "setInterval(loadDoc,1000);\n";

html += "function loadDoc() {\n";

html += "var xhttp = new XMLHttpRequest();\n";

html += "xhttp.onreadystatechange = function() {\n";

html += "if (this.readyState == 4 && this.status == 200) {\n";

html += "document.body.innerHTML =this.responseText}\n";
```

```
    html += "};\n";
    html += "xhttp.open(\"GET\", \"/\", true);\n";
    html += "xhttp.send();\n";
    html += "}\n";
    html += "</script>\n";
        //Ajax Code END


        //start Javascript logic for digital clock
    html += "<script type='text/javascript'>\n";
    html += "function updateClock(){\n";
    html += "var now = new Date();\n";
    html += "var dname = now.getDay(),mo = now.getMonth(),dnum = now.getDate(),yr = now.getFullYear(),hou = now.getHours(),min = now.getMinutes(),sec = now.getSeconds(),pe = 'AM';\n";


    html += "if(hou >= 12){pe = 'PM';}\n";
    html += "if(hou == 0){hou = 12;}\n";
    html += "if(hou > 12){hou = hou - 12;}\n";


    html += "Number.prototype.pad = function(digits){\n";
    html += "for(var n = this.toString(); n.length < digits; n = 0 + n);\n";
    html += "return n;\n";
    html += "}\n";


    html += "var months = ['January', 'February', 'March', 'April', 'May', 'June', 'July', 'August', 'September', 'October', 'November', 'December'];\n";
```

```
html += "var week = ['Sunday', 'Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday'];\n";

html += "var ids = ['dayname', 'month', 'daynum', 'year', 'hour', 'minutes', 'seconds', 'period'];\n";

html += "var values = [week[dname], months[mo], dnum.pad(2), yr, hou.pad(2), min.pad(2), sec.pad(2), pe];\n";

html += "for(var i = 0; i < ids.length; i++)\n";

html += "document.getElementById(ids[i]).firstChild.nodeValue = values[i];\n";

html += "}\n";


html += "function initClock(){\n";

html += "updateClock();\n";

html += "window.setInterval('updateClock()', 1);\n";

html += "}\n";

html += "</script>\n";

      // end Javascript logic for digital clock


html += "</head>";


html += "<body onload='initClock()'>";

html += "<div id='page'>";

html += "<div class='header'>";

html += "<h1>Real Time Health Monitoring System</h1>";

html += "</div>";


       //digital clock start
```

```
html += "<div class='datetime'>";
html += "<div class='date'>";
html += "<span id='dayname'>Day</span>,";
html += "<span id='month'>Month</span>";
html += "<span id='daynum'>00</span>,";
html += "<span id='year'>Year</span>";
html += "</div>";
html += "<div class='time'>";
html += "<span id='hour'>00</span>:";
html += "<span id='minutes'>00</span>:";
html += "<span id='seconds'>00</span>";
html += "<span id='period'>AM</span>";
html += "</div>";
html += "</div>";
      //digital clock end-

html += "<div id='content' align='center'>";
html += "<div class='box-full' align='left'>";
html += "<h2>  Sensors Readings</h2>";
html += "<div class='sensors-container'>";
       //For Temperature
html += "<div class='sensors'>";
html += "<p class='sensor'>";
html += "<i class='fas fa-thermometer-half' style='color:#0275d8'></i>";
html += "<span class='sensor-labels'> Room Temperature </span>";
```

```
html += (int)temperature;
html += "<sup class='units'>°C</sup>";
html += "</p>";
html += "<hr>";
html += "</div>";
      //For Humidity
html += "<div class='sensors'>";
html += "<p class='sensor'>";
html += "<i class='fas fa-tint' style='color:#5bc0de'></i>";
html += "<span class='sensor-labels'> Room Humidity </span>";
html += (int)humidity;
html += "<sup class='units'>%</sup>";
html += "</p>";
html += "<hr>";
      //For Heart Rate
html += "<p class='sensor'>";
html += "<i class='fas fa-heartbeat' style='color:#cc3300'></i>";
html += "<span class='sensor-labels'> Heart Rate </span>";
html += (int)BPM;
html += "<sup class='units'>BPM</sup>";
html += "</p>";
html += "<hr>";
      //For Sp02
html += "<p class='sensor'>";
html += "<i class='fas fa-burn' style='color:#f7347a'></i>";
```

```
html += "<span class='sensor-labels'> Sp02 </span>";
html += (int)SpO2;
html += "<sup class='units'>%</sup>";
html += "</p>";
html += "<hr>";
      //For Body Temperature celsius
html += "<p class='sensor'>";
html += "<i class='fas fa-thermometer-full' style='color:#d9534f'></i>";
html += "<span class='sensor-labels'> Body Temperature </span>";
html += (int)bodytemperature;
html += "<sup class='units'>°C</sup>";
html += "</p>";
html += "<hr>";

html += "</div>";
html += "</div>";
html += "</div>";
html += "</div>";
html += "</div>";

html += "<div class='footer'>";
html += " <br>Prepared by<br> Saeedul Haque [1904049]<br>.";
html += "</div>";

html += "</body>";
```

```
    html += "</html>";

    return html;
}
```

3.5 Result

I. Project Setup

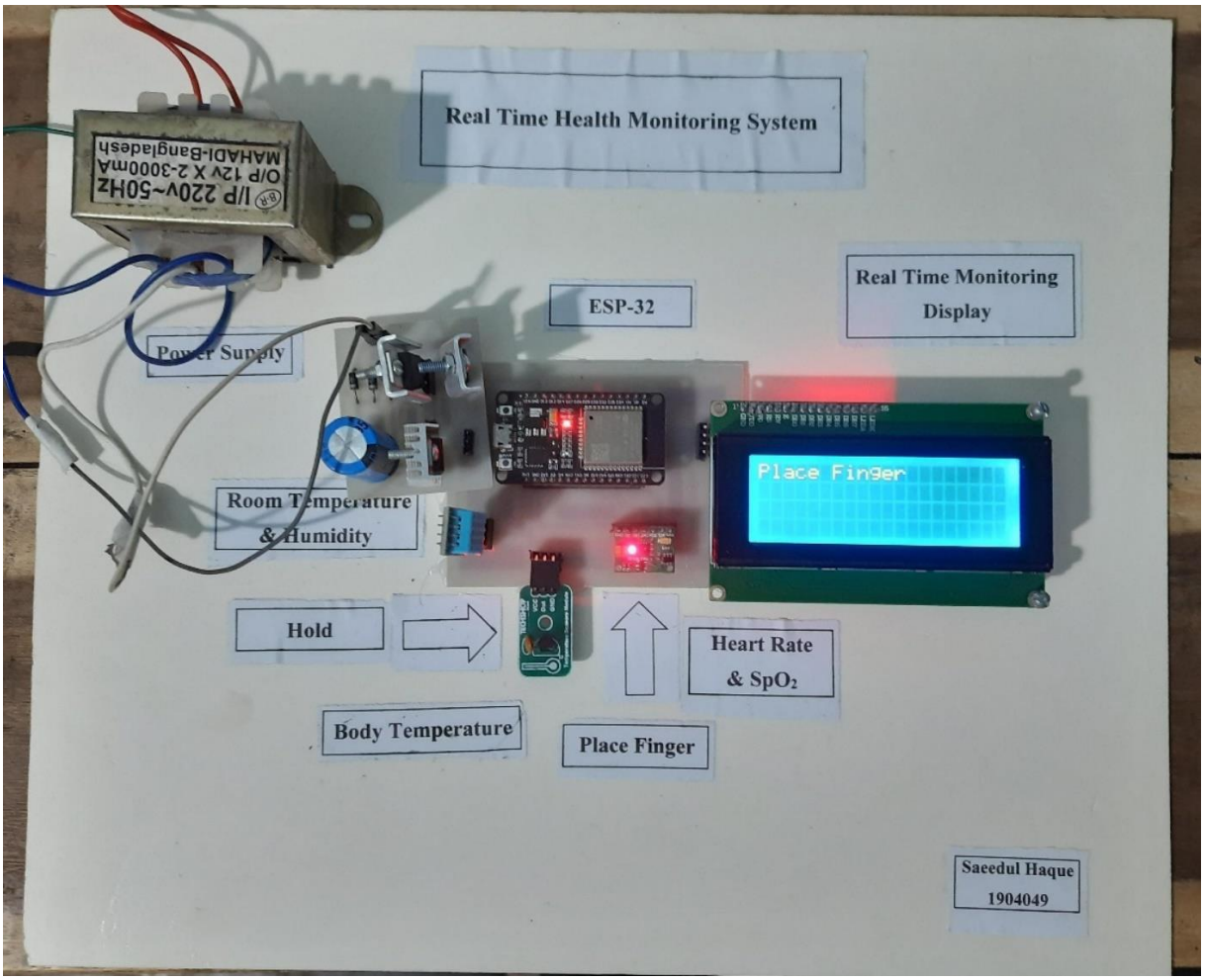
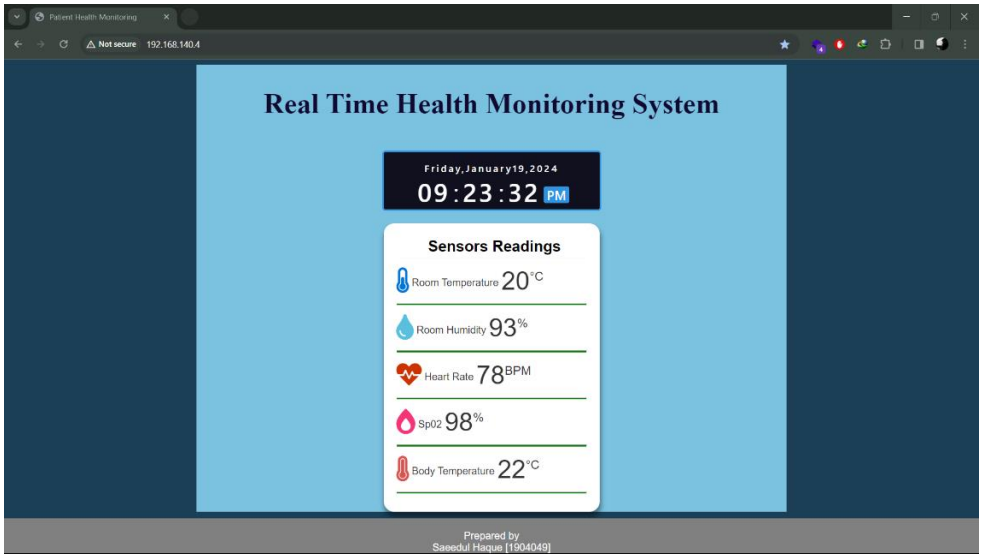
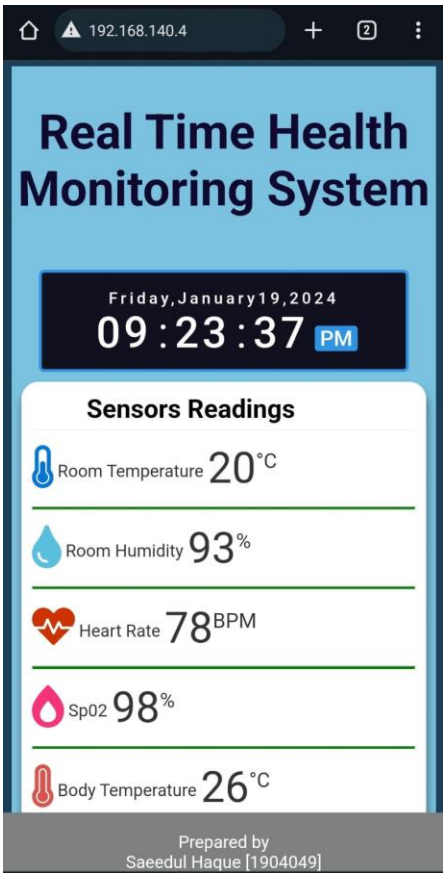


Figure 2.13: Prepared Project Setup

II. Output



(a)



(b)

Figure 2.14: Monitoring through local webserver (a) PC (b) Mobile

Figure 2.15: Monitoring through LCD Display

III. **Alert**

Figure 2.16: Alert to notify at certain condition

Chapter 4

4.1 Advantage

- Easy to operate
- User friendly
- Ability to monitor in real time
- Components easily available
- Low cost
- Modular
- Operable from AC power supply as well as DC source

4.2 Disadvantage

- Power supply failure will result in system shutdown
- Since the optical sensor can only operate in 100 Hz, the response time period can be maximum of 10ms. Hence it can not collaborate with IoT which requires higher delay.
- Requires AC power supply

4.3 Application

Some applications of this health monitoring project are:

• ICU Room.

• Rural Area.

• Tele-medicine Purpose

• Patient’s Room

• Automation in Medical Purpose

• IoT Controlled Hospital

• IoT controlled Patient’s Room

4.4 Conclusion

In conclusion, the health monitoring system incorporates the MAX30100 pulse oximeter for measuring the heart rate and blood oxygen level, DHT11 for room temperature and humidity, LM35 for body temperature, an LCD display, and a local web server, which offers solution for real-time health condition monitoring. By employing the principles of photoplethysmography, capacitive humidity sensing, and analog temperature measurement, the system provides accurate and reliable data on heart rate, blood oxygen levels, room temperature, humidity, and body temperature. The ESP-32, acting as the central processing unit, collects, processes, and displays this information to both a local LCD display and a user-friendly web interface accessible through a local network. This dual-display mechanism ensures immediate access to health metrics, allowing users to conveniently monitor their well-being in real-time. The project's utilization of a local web server extends its usability, enabling remote monitoring from various devices connected to the same network. This adaptability makes the system versatile and suitable for both personal health tracking and healthcare applications.