Microprocessors and Microcontrollers

A MINOR PROJECT REPORT

ON

Human Health Monitoring System

Submitted By

Sanket S.Sarmalkar (Roll No-20EEE1028)
Sanket Bhat (Roll No-20EEE1027)
Rahul Jalan (Roll No-20EEE1022)

November 2022

NATIONAL INSTITUTE OF TECHNOLOGY GOA

Farmagudi, Ponda, Goa – 403 401, India

Department of Electrical and Electronics Engineering

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

1. INTRODUCTION

The modern visionary of healthcare industry is to provide better healthcare to people in a more economic and patient friendly manner. Therefore, for increasing the patient care efficiency, there arises a need to improve the patient monitoring devices. The medical world today faces the most major problem that is the need of health care providers presence near the bedside of the patient. In this busy lifestyle, monitoring our health condition is becoming hectic so everyone expects to know about their health conditions using some smart technology which can be easily accessible and is effective.

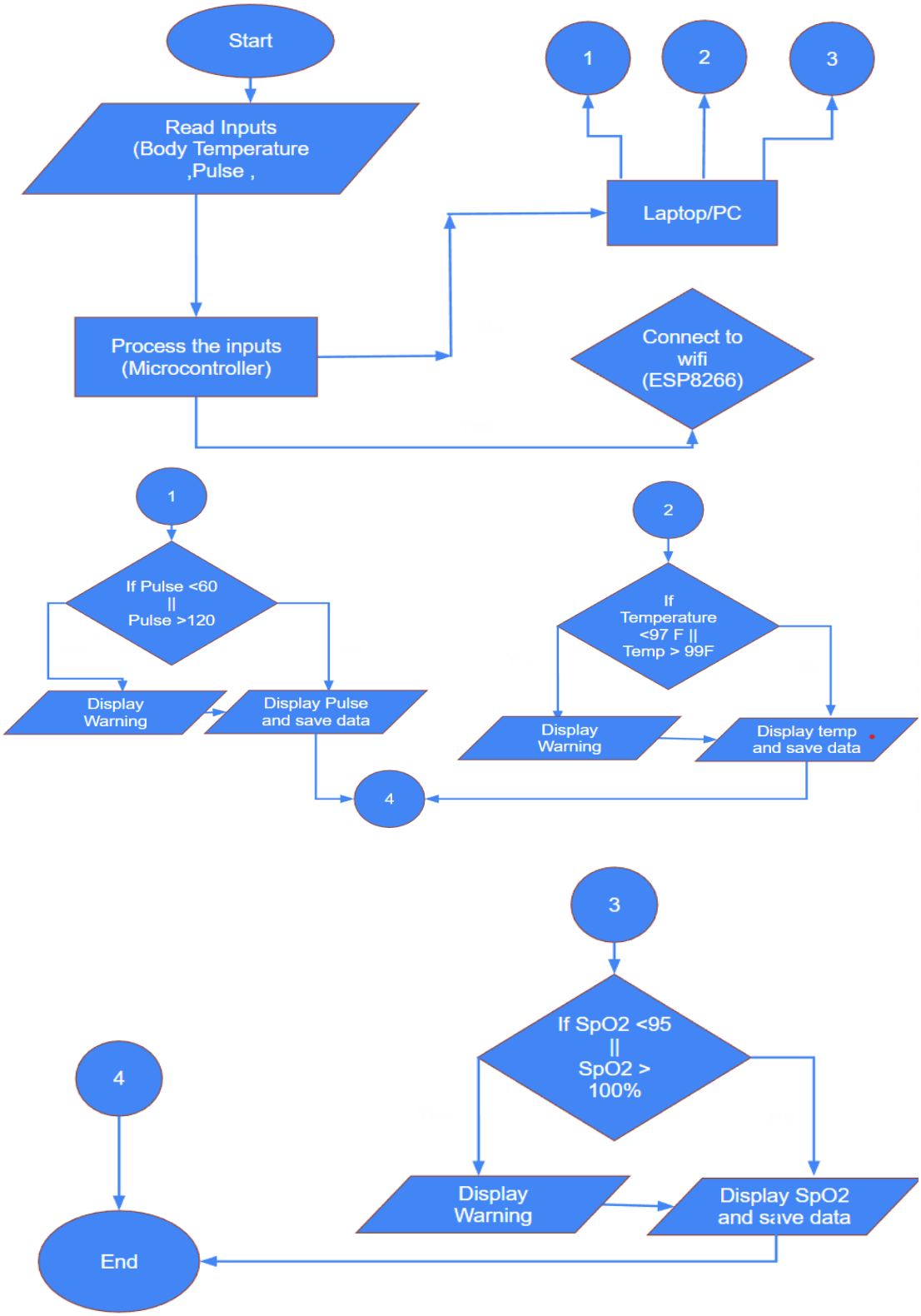
As the computers, bio instrumentation, and telecommunications technologies are being advanced, it has become feasible to design more the smart devices which help in tele monitoring systems to record data, acquire data, and display the data and to transmit the health signals from the human body to any location. Telemedicine benefits both the patients with efficient health care facility and even the doctors who can give better assistance to the people.

It is cost effective. It can increase the efficiency through better management of patient monitoring, shared health professional staffing. Tele monitoring involves remotely monitoring the patient health care. These devices keep track of blood pressure, heart rate, weight, blood glucose etc. of the patient. The Telemedicine system consists of customized hardware and software at both the patient and specialist doctor ends.

This project discusses the advantages of using android technology and Arduino for patient health monitoring system. In this technology the data is collected from a patient, to feed the same to separate interfaces in which the patient parameters and details is displayed on the local server. The person then can connect to the local IP and can get, analyse and can take preventive measures before he reaches the hospital in serious case.

Hence, this project can be used to help the patients monitoring and assistance by using the trending technology.

2. BLOCK DIAGRAM



3. CIRCUIT SCHEMATIC DIAGRAM

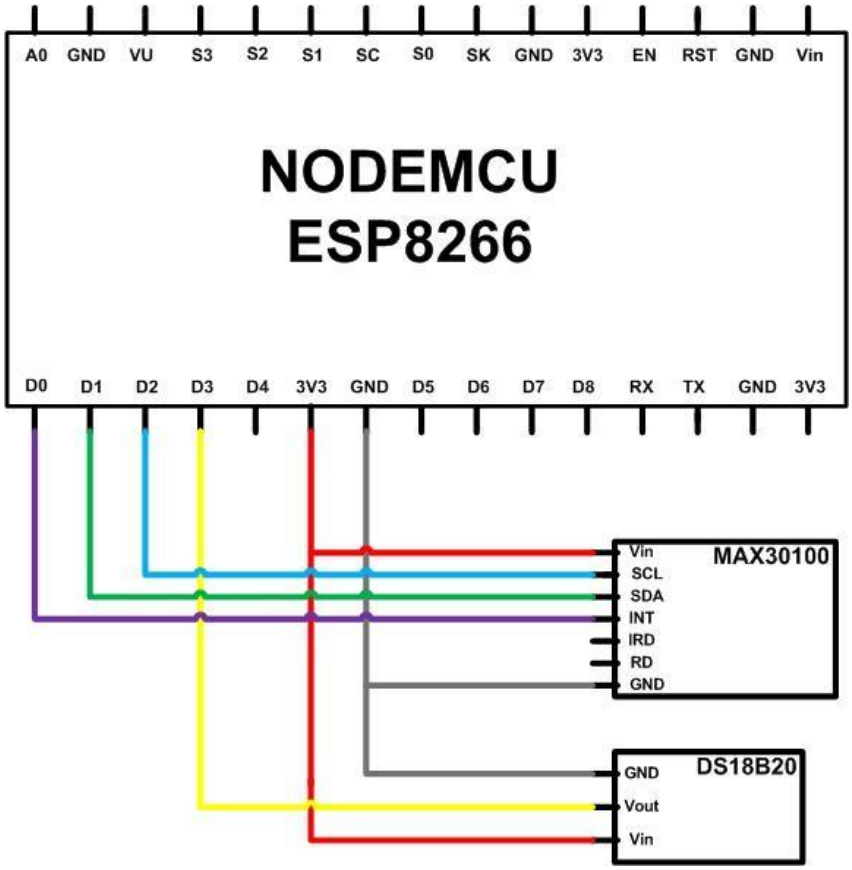


Figure 3-1: Circuit Diagram

Table 3-1 List of components

SL. No	Component	Part ID
1	NodeMCU	NodeMCU
2	Temperature sensor	DS18B20
3	Pulse and Oximeter sensor	MAX30102

4. SOFTWARE DESCRIPTION

Code for this project can be found here

https://github.com/SanketSarmalkar/HumanHealthMonitoringSystem

CODE :

```
#include <ESP8266WiFi.h>

#include <ESP8266WebServer.h>

#include <OneWire.h>

#include <DallasTemperature.h>

#include <Wire.h>

#include "MAX30105.h"

#include "MAX30100_PulseOximeter.h"

#include <string>

#include "heartRate.h"


#define DS18B20 2


PulseOximeter pox;

//for ds18b20

OneWire oneWire(DS18B20);

DallasTemperature sensors(&oneWire);


// max30102

MAX30105 particleSensor;


const byte RATE_SIZE = 4;  //Increase this for more averaging. 4 is good.

byte rates[RATE_SIZE];     //Array of heart rates

byte rateSpot = 0;

long lastBeat = 0;  //Time at which the last beat occurred
```

```
float beatsPerMinute;

int beatAvg;


float bodytemp = 0;


/*Put WiFi SSID & Password*/

const char* ssid = "Galaxy M518B8E";  // Enter SSID here

const char* password = "jzni3896";    // Enter Password here


ESP8266WebServer server(80);


bool LEDstatus = LOW;


// SPO2

double aveRed = 0;     //DC component of RED signal

double aveIr = 0;      //DC component of IR signal

double sumIrRMS = 0;   //sum of IR square

double sumRedRMS = 0;  // sum of RED square

unsigned int i = 0;    //loop counter

#define SUM_CYCLE 100

int Num = SUM_CYCLE;  //calculate SpO2 by this sampling interval

double eSpO2 = 95.0;  //initial value of estimated SpO2

double fSpO2 = 0.7;   //filter factor for estimated SpO2

double fRate = 0.95;  //low pass filter for IR/red LED value to eliminate AC component

double SpO2 = 0;
```

```
#define TIMETOBOOT 3000  // wait for this time(msec) to output SpO2
#define SCALE 88.0       //adjust to display heart beat and SpO2 in Arduino serial plotter at the same time
#define SAMPLING 1       //if you want to see heart beat more precisely , set SAMPLING to 1
#define FINGER_ON 50000  // if ir signal is lower than this , it indicates your finger is not on the sensor
#define MINIMUM_SPO2 80.0
#define MAX_SPO2 100.0
#define MIN_SPO2 80.0



void setup() {
 Serial.begin(115200);
 delay(100);
 pinMode(16, OUTPUT);



 if (!particleSensor.begin(Wire, I2C_SPEED_FAST))  //Use default I2C port, 400kHz speed
 {
  Serial.println("MAX30105 was not found. Please check wiring/power. ");
  while (1)
    ;
 }
 Serial.println("Place your index finger on the sensor with steady pressure.");

 particleSensor.setup();                    //Configure sensor with default settings
 particleSensor.setPulseAmplitudeRed(0x0A);  //Turn Red LED to low to indicate sensor is running
 particleSensor.setPulseAmplitudeGreen(0);   //Turn off Green LED
```

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

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

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

  Serial.begin(9600);
  Serial.println("Dallas Temperature IC Control Library Demo");
  // Start up the library
  sensors.begin();
}
Serial.println("");
Serial.println("WiFi connected..!");
Serial.print("Got IP: ");
Serial.println(WiFi.localIP());

server.on("/", handle_OnConnect);
server.on("/ledon", handle_ledon);
server.on("/ledoff", handle_ledoff);
server.onNotFound(handle_NotFound);
```

```
  server.begin();
  Serial.println("HTTP Server Started");
}
void loop() {
  server.handleClient();
  /*
  if(LEDstatus)
  {
    digitalWrite(16, HIGH);}
  else
  {
    digitalWrite(16, LOW);}
*/
  /*
  Serial.print(" Requesting temperatures...");
  sensors.requestTemperatures(); // Send the command to get temperature readings
  Serial.println("DONE");
/*****************************************************************************/
  /*  Serial.print("Temperature is: ");
  bodytemp = sensors.getTempCByIndex(0);
  Serial.print(sensors.getTempCByIndex(0));
*/
  long irValue = particleSensor.getIR();

  if (checkForBeat(irValue) == true) {

    Serial.print("Requesting temperatures...");
    sensors.requestTemperatures();  // Send the command to get temperature readings
```

```
Serial.println("DONE");

/**************************************************************************/

Serial.print("Body Temperature: ");

bodytemp = sensors.getTempCByIndex(0);

Serial.print(sensors.getTempCByIndex(0));

//We sensed a beat!

long delta = millis() - lastBeat;

//lastBeat = millis();


beatsPerMinute = 60 / (delta / 1000.0);


// beatsPerMinute = pox.getHeartRate();

/*Serial.print("delta");

Serial.print(delta);

Serial.print("bpm rahul");

Serial.print(beatsPerMinute);*/


if (beatsPerMinute < 255 && beatsPerMinute > 20) {

  rates[rateSpot++] = (byte)beatsPerMinute;  //Store this reading in the array

  rateSpot %= RATE_SIZE;                    //Wrap variable


  //Take average of readings

  beatAvg = 0;

  for (byte x = 0; x < RATE_SIZE; x++)

    beatAvg += rates[x];

  beatAvg /= RATE_SIZE;

}
```

```
//SPO2

uint32_t ir, red;  //raw data

double fred, fir;  //floating point RED ana IR raw values

double SpO2 = 0;   //raw SpO2 before low pass filtered

// double ir = particleSensor.getIR();

// double red = particleSensor.getRed();

red = particleSensor.getRed();  //Sparkfun's MAX30105

ir = particleSensor.getIR();    //Sparkfun's MAX30105


fred = (double)red;

fir = (double)ir;

aveRed = aveRed * fRate + (double)red * (1.0 - fRate);  //average red level by low pass filter

aveIr = aveIr * fRate + (double)ir * (1.0 - fRate);     //average IR level by low pass filter

sumRedRMS += (fred - aveRed) * (fred - aveRed);         //square sum of alternate component of red level

sumIrRMS += (fir - aveIr) * (fir - aveIr);              //square sum of alternate component of IR level

double R = (sqrt(sumRedRMS) / aveRed) / (sqrt(sumIrRMS) / aveIr);


SpO2 = -45.060 * R * R + 30.354 * R + 94.845 + 12;

//SpO2 = -23.3*(R - 0.4 )+ 100;

eSpO2 = fSpO2 * eSpO2 + (1.0 - fSpO2) * SpO2;




Serial.print(" IR=");

Serial.print(irValue);

Serial.print(", BPM=");
```

```
    Serial.print(beatsPerMinute);
    Serial.print(", Avg BPM=");
    Serial.print(beatAvg);
    Serial.print(", SpO2 =");
    Serial.println(eSpO2);

    delay(1000);
  }
  /*
  Serial.print("IR=");
  Serial.print(irValue);
  Serial.print(", BPM=");
  Serial.print(beatsPerMinute);
  Serial.print(", Avg BPM=");
  Serial.print(beatAvg);
*/
  if (irValue < 50000) {
    Serial.println("No finger?");
    delay(2000);
    beatsPerMinute = 0;
    eSpO2 = 95;
  }
  lastBeat = millis();
}

void handle_OnConnect() {
  //LEDstatus = LOW;
  //Serial.println("LED: OFF");
```

```
  server.send(200, "text/html", updateWebpage(LEDstatus));
}

void handle_ledon() {
  //LEDstatus = HIGH;
  //Serial.println("LED: ON");
  server.send(200, "text/html", updateWebpage(LEDstatus));
}

void handle_ledoff() {
  LEDstatus = LOW;
  Serial.println("LED: OFF");
  server.send(200, "text/html", updateWebpage(LEDstatus));
}

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

String updateWebpage(uint8_t LEDstatus) {
  String ptr = "<!DOCTYPE html> <html>\n";
  ptr += "<head><meta name=\"viewport\" content=\"width=device-width, initial-scale=1.0, user-scalable=no\"> <script src=\"https://cdn.tailwindcss.com\"><\/script>\n";
  ptr += "<title>LED Control</title>\n";
  ptr += "<style>html {font-family: Helvetica; display: inline-block; margin: 0px auto; text-align: center;}\n";
  ptr += "body{margin-top: 50px;} h1 {color: #444444;margin: 50px auto 30px;} h3 {color: #444444;margin-bottom: 50px;}\n";
  ptr += ".button {display: block;width: 80px;background-color: #1abc9c;border: none;color: white;padding: 13px 30px;text-decoration: none;font-size: 25px;margin: 0px auto 35px;cursor: pointer;border-radius: 4px;}\n";
```

```
  ptr += ".button-on {background-color: #3498db;}\n";
  ptr += ".button-on:active {background-color: #3498db;}\n";
  ptr += ".button-off {background-color: #34495e;}\n";
  ptr += ".button-off:active {background-color: #2c3e50;}\n";
  ptr += "p {font-size: 14px;color: #888;margin-bottom: 10px;}\n";
  ptr += "</style>\n";
  ptr += "</head>\n";
  ptr += "<body>\n";
  ptr += "<h1 class=\"text-amber-700 text-2xl\">ESP8266 Web Server</h1>\n";
  ptr += "<h3 class=\"mb-0\">Using Station(STA) Mode</h3>\n";
  /*ptr +=""


   if(LEDstatus){
    ptr +="<p>BLUE LED: ON</p><a class=\"button button-off\" href=\"/ledoff\">OFF</a>\n";
   }else{
    ptr +="<p>BLUE LED: OFF</p><a class=\"button button-on\" href=\"/ledon\">ON</a>\n";
   }
*/


  //Ajax Code Start
  ptr += "<script>\n";
  ptr += "setInterval(loadDoc,1000);\n";
  ptr += "function loadDoc() {\n";
  ptr += "var xhttp = new XMLHttpRequest();\n";
  ptr += "xhttp.onreadystatechange = function() {\n";
  ptr += "if (this.readyState == 4 && this.status == 200) {\n";
  ptr += "document.body.innerHTML =this.responseText}\n";
  ptr += "};\n";
```

```
ptr += "xhttp.open(\"GET\", \"/\", true);\n";

ptr += "xhttp.send();\n";

ptr += "}\n";

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

//Ajax Code END



ptr += "</body>\n";

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

//For Body Temperature

ptr += "<p >";

//ptr += "<i class='fas fa-thermometer-full' style='color:#d9534f'></i>";

ptr += "<span class='sensor-labels pt-4'> Body Temperature </span><br/>";

ptr += "<div class=\"rounded-2 bg-lime-100 p-2\">";

ptr += "<span class=\"text-2xl text-green-500 \">";

ptr += (float)bodytemp;

ptr += "<\span>";

ptr += "<sup class='units'>°C</sup>";

ptr += "<\div>";

ptr += "</p>";

/*

ptr +="</body>\n";

ptr +="</html>\n";*/

//For heartbeat

ptr += "<p class='sensor'>";

ptr += "<i class='fas fa-thermometer-full' style='color:#d9534f'></i>";

ptr += "<span class='sensor-labels pt-4'> Beats per Mins </span><br/>";

ptr += "<div class=\"rounded-2 bg-lime-100 p-2\">";

ptr += "<span class=\"text-2xl text-green-500 \">";
```

```
  ptr += (float)beatsPerMinute;

  ptr += "<\span>";

  //ptr += "<sup class='units'>BPM</sup>";

  ptr += "<\div>";

  ptr += "</p>";


  ptr += "<p class='sensor'>";

  ptr += "<i class='fas fa-thermometer-full' style='color:#d9534f'></i>";

  ptr += "<span class='sensor-labels pt-4'> Beats per Mins </span><br/>";

  ptr += "<div class=\"rounded-2 bg-lime-100 p-2\">";

  ptr += "<span class=\"text-2xl text-green-500 \">";

  ptr += (float)(beatsPerMinute==0)?0:eSpO2;

  ptr += "<\span>";

  ptr += "<sup class='units'> %</sup>";

  ptr += "<\div>";

  ptr += "</p>";


  ptr += "</div>";

  ptr += "</div>";

  ptr += "</div>";

  ptr += "</div>";

  ptr += "</div>";

  ptr += "</body>";

  ptr += "</html>";

  return ptr;

}
```

5. PROJECT OUTCOME'S

The proposed patient health monitoring system can be used extensively in an emergency conditions as they can be monitored daily, recorded and stored as a database. In the future we can work more analyzing the data from the cloud and predict the outcomes in future, Emergency alert messages feature can be enabled and can be shared across intensive care and treatment hospitals. And, In case of a future pandemic this health monitoring system is very useful, we can avoid going to hospital regularly and check ourself at our house.

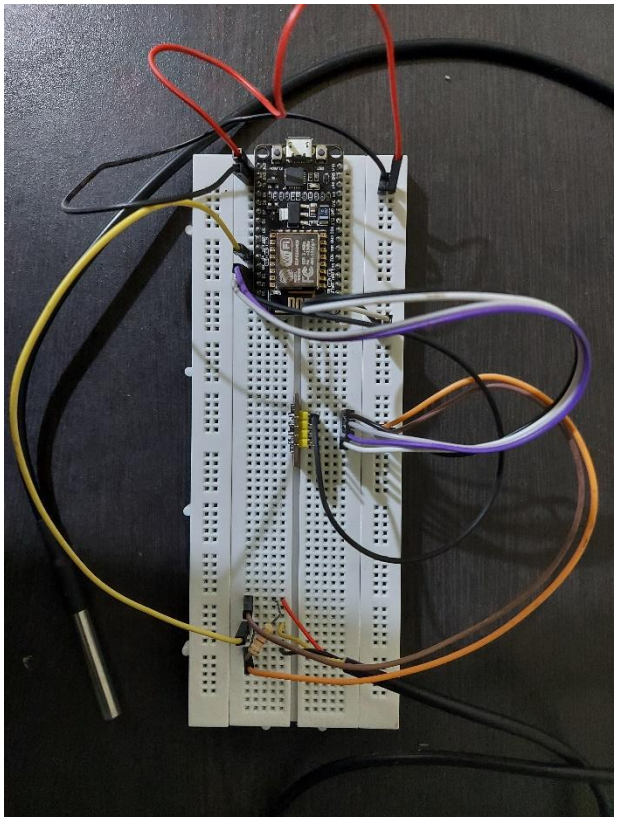


Fig 5.1 Top view of project

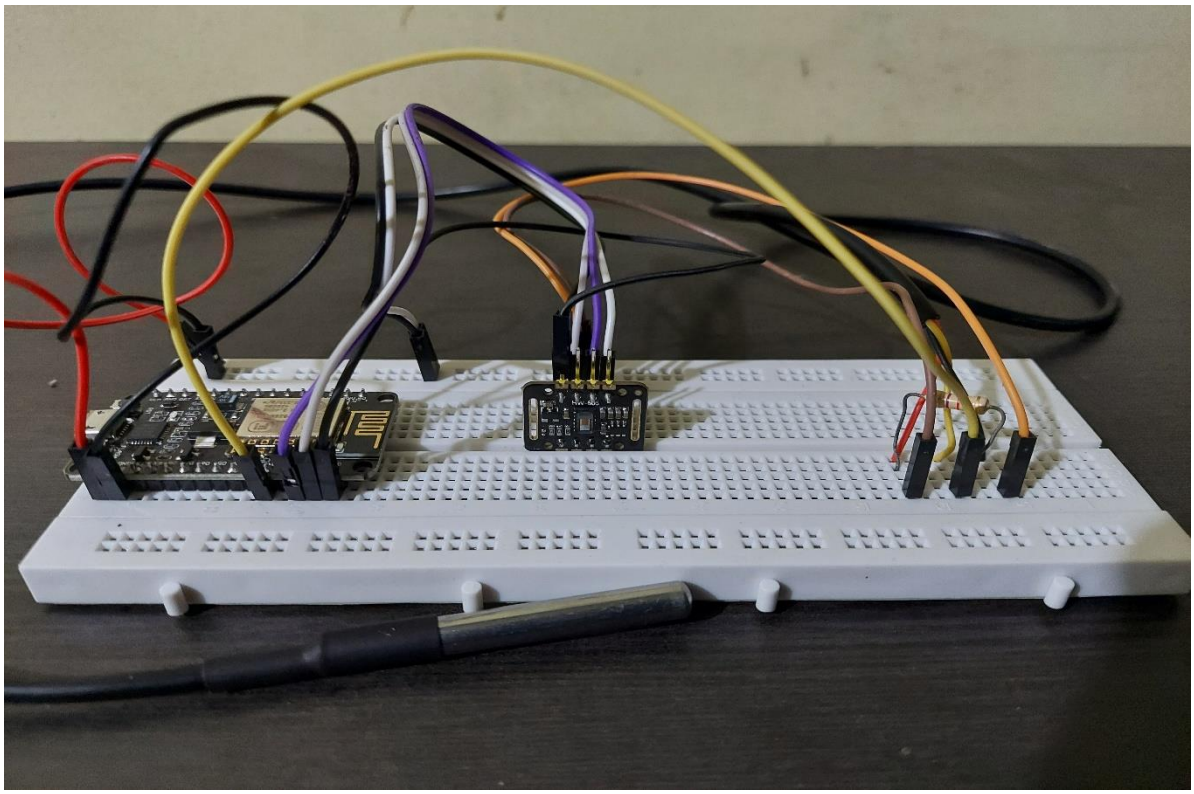


Fig 5.2 Side view of project

REFERENCES

- https://www.bing.com/ck/a?!&&p=14df5ee8ddd99488JmltdHM9MTY2OTc2NjQwMCZpZ3VpZD0wZjI4MzA2My0xMmMxLTY1NjUtMzJmNi0yMjIxMTM2YzY0ZTAmaW5zaWQ9NTE4OA&ptn=3&hsh=3&fclid=0f283063-12c1-6565-32f6-2221136c64e0&psq=nodemcu+documentation&u=a1aHR0cHM6Ly9ub2RlbWN1LnJlYWR0aGVkb2NzLmlvLw&ntb=1
- https://www.alldatasheet.com/datasheet-pdf/pdf/1338715/MAXIM/MAX30102.html
- https://github.com/coniferconifer/ESP32_MAX30102_simple-SpO2_plotter
- https://www.alldatasheet.com/datasheet-pdf/pdf/227472/DALLAS/DS18B20.html

	Evaluator	Signature
1		
2		