HEALTH MONITORING SYSTEM

Manjusha S (21Z317)
Pradeesha S (21Z329)
Mydhilinayaki R (21Z324)
S Harini (21Z338)
Namita T J (21IZ014)
Varsha V (21IZ015)

19Z604 - EMBEDDED SYSTEMS

report submitted in partial fulfillment of the requirement for the award of degree of

BACHELOR OF ENGINEERING

Branch: COMPUTER SCIENCE AND ENGINEERING

Of Anna University

April 2024

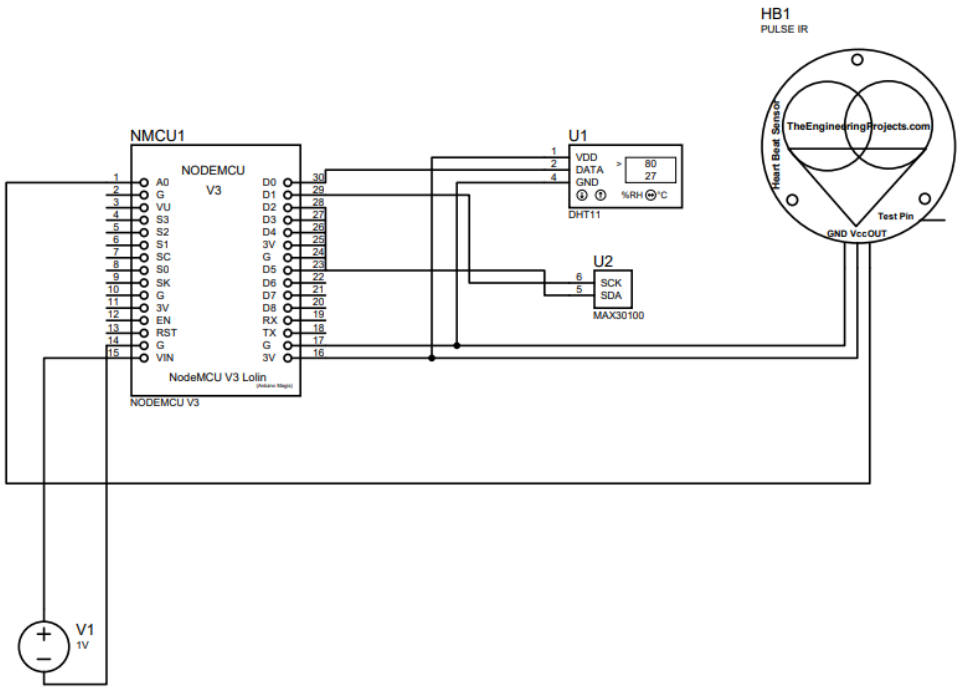
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
PSG COLLEGE OF TECHNOLOGY
(Autonomous Institution)

PROBLEM STATEMENT:

In the modern era, where health consciousness is on the rise, there's a growing need for accessible and affordable health monitoring solutions. To address this, we propose the development of a comprehensive health monitor system leveraging NodeMCU, PPG sensor, DHT sensor, and MAX30100 sensor.

The objective is to create a compact and versatile device capable of monitoring crucial health parameters such as pressure,spo2,bpm(beats per minute) and temperature in real-time. This device will provide users with valuable insights into their health status, enabling them to take proactive measures towards maintaining their well-being.
The primary challenge lies in seamlessly integrating multiple sensors with the NodeMCU development board and developing efficient data acquisition and processing algorithms. Additionally, ensuring accuracy and reliability of sensor readings, designing a user-friendly interface for data visualization, and implementing intelligent alerts for abnormal health conditions are key aspects to address.
Ultimately, the goal is to empower users with a cost-effective and user-friendly health monitoring solution that can be deployed at home, in healthcare facilities, or even in remote areas with limited access to medical infrastructure. This project aims to contribute towards democratizing healthcare and improving health outcomes for individuals worldwide.

SCHEMATIC DIAGRAM:



EMBEDDED C CODE:

```
#include <Wire.h>
#include <LCD_I2C.h>
#include <Wire.h>
float flt=0;
#include <ESP8266WiFi.h>
#include <SoftwareSerial.h>
#include <SimpleDHT.h>
int pinDHT11 = D0;
SimpleDHT11 dht11(pinDHT11);
float bpm,spo2;
String stat="";
float te,hu;
#include "ThingSpeak.h" // always include thingspeak header file after other header files and custom macros
char ssid[] = "Project";   // your network SSID (name)
char pass[] = "12345678";   // your network password
int keyIndex = 0;            // your network key Index number (needed only for WEP)
WiFiClient  client;
float d;
  String y="";
  int xt=0;
LCD_I2C lcd(0x27); // Initialize the LCD with I2C address 0x27
#include <Wire.h>
#include "MAX30100_PulseOximeter.h"
#define REPORTING_PERIOD_MS     1000
// PulseOximeter is the higher level interface to the sensor
// it offers: * beat detection reporting * heart rate calculation* SpO2 (oxidation level) calculation
PulseOximeter pox;
uint32_t tsLastReport = 0;
// Callback (registered below) fired when a pulse is detected
void onBeatDetected()
{Serial.println("Beat!");}
// Digital pin connected to IR transmitter
const int irReceiverPin = A0;       // Analog pin connected to IR receiver
void setup() {Serial.begin(9600);
   lcd.begin(); // If you are using more I2C devices using the Wire library use lcd.begin(false)
              // this stop the library(LCD_I2C) from calling Wire.begin()
             // Turn on the backlight
  pinMode(A0, INPUT);
```

```
  WiFi.mode(WIFI_STA);
  ThingSpeak.begin(client);
  // Connect or reconnect to WiFi
 for (int ji=0;ji<5;ji++){
      WiFi.begin(ssid, pass);  // Connect to WPA/WPA2 network.
      Serial.print(".");delay(3000);    } Serial.println("");delay(100);
   if (!pox.begin()) {Serial.println("FAILED");
       for(;;); } else {
Serial.println("SUCCESS");} pox.setOnBeatDetectedCallback(onBeatDetected);}
void loop() {// Make sure to call update as fast as possible
pox.update();
    // Asynchronously dump heart rate and oxidation levels to the serial
    // For both, a value of 0 means "invalid"
    if (millis() - tsLastReport > REPORTING_PERIOD_MS) {
       Serial.print("Heart rate:");bpm=pox.getHeartRate();
       spo2=pox.getSpO2();Serial.print(bpm);
       Serial.print("bpm / SpO2:");
       Serial.print(spo2);
       Serial.println("%");
  if((bpm>50)&&(spo2>90))
  {float pressureValue = measurepressure();
 // read without samples.
  byte temperature = 0;
  byte humidity = 0;
  int err = SimpleDHTErrSuccess;
  if ((err = dht11.read(&temperature, &humidity, NULL)) != SimpleDHTErrSuccess) {
    Serial.print("Read DHT11 failed, err="); Serial.println(err);delay(1000);
    return; }
Serial.print("Sample OK: ");
Serial.print((int)temperature); Serial.print(" *C, ");
Serial.print((int)humidity); Serial.println(" H");
te=temperature;hu=humidity;
ThingSpeak.setField(1,pressureValue);ThingSpeak.setField(2,bpm);
ThingSpeak.setField(3,spo2);ThingSpeak.setField(4,te);
int xuk = ThingSpeak.writeFields(549041, "1Y5AWUR4KI7Y1FMW");
  } tsLastReport = millis();}}
float measurepressure() {float pressureValue;float gv=0.0;
 for(int y=0;y<100;y++){pressureValue=analogRead(A0);
//Serial.println(glucoseValue);
if(pressureValue>600){gv++;}delay(100);}return gv;}
```

HTML CODE

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
                    <meta          name="viewport"
content="width=device-width,
initial-scale=1.0">
  <title>Patient Monitoring</title>
  <style>
    /* Your existing styles remain unchanged */
  </style>
</head>

<body>
<center>
  <div id="container">
       <h1>Patient  Monitoring</h1>      <table
id="aged-people-table">
     <!-- Table headers for the first seven fields
-->
     <thead>
       <tr>
         <th>Pressure</th>
         <th>BPM</th>
         <th>SPO2</th>
         <th>Temperature</th>
         </tr>
     </thead>
          <!-- Table body will be dynamically
populated using JavaScript -->
    </table>
      <!-- JavaScript to fetch ThingSpeak data
and update the table every 4 seconds -->
    <script>
  function fetchAgedPeopleData() {

fetch('https://api.thingspeak.com/channels/549
041/feeds.json?results=1')
         .then(response => response.json())
         .then(data => {
               const agedPeopleFields = ['field1',
'field2', 'field3', 'field4'];
                          const agedPeopleTable =
document.getElementById('aged-people-table'
);
                     // Clear existing table rows
agedPeopleTable.getElementsByTagName('tb
ody')[0]?.remove();
           // Add table body
                                  const tbody =
agedPeopleTable.createTBody();
           const row = tbody.insertRow();
           agedPeopleFields.forEach(field => {
             const td = row.insertCell();
             td.textContent = data.feeds[0][field];
           });
         })
              .catch(error => console.error('Error
fetching ThingSpeak data:', error));
     }
     fetchAgedPeopleData(); // Initial fetch
     // Auto-reload every 4 seconds
     setInterval(fetchAgedPeopleData, 4000);
    </script>
  </div>
</center>
</body>
</html>
```

SNAPSHOTS OF THE PROJECT: