



HealthSense User Manual

**West Visayas State University
College of Information and Communications
Technology**

FELASOL FORMOSO GAQUIT MARBEBE UNRAR

HealthSense: A medical device with an embedded sensor system for
vital signs monitoring and Heart Disease prediction

An Undergraduate Thesis
Presented to the Faculty of the
College of Information and Communications Technology
West Visayas State University
La Paz, Iloilo City

In Partial Fulfillment
of the Requirements for the Degree
Bachelor of Science In Information Technology

by
Ana Patricia F. Felasol
Thrys J. Formoso
Patricia Anne A. Gaquit
John Kate E. Marbebe
Luke Ian B. Undar

June 2023

DISCLAIMER

This software project and its corresponding documentation entitled "HealthSense: A medical device with an embedded sensor system for vital signs monitoring and Heart Disease prediction" is submitted to the College of Information and Communications Technology, West Visayas State University, in partial fulfillment of the requirements for the degree, Bachelor of Science in Information Technology. It is the product of our own work, except where indicated text.

We hereby grant the College of Information and Communications Technology permission to freely use, publish in local or international journal/conferences, reproduce, or distribute publicly the paper and electronic copies of this software project and its corresponding documentation in whole or in part, provided that we are acknowledged.

Ana Patricia F. Felasol

Thrys J. Formoso

Patricia Anne A. Gaquit

John Kate E. Marbebe

Luke Ian B. Undar

Table of Contents

DISCLAIMER	3
Table of Contents	4
INTRODUCTION	4
SYSTEM REQUIREMENTS	5
HOW TO DEPLOY THE WEB APP	7
INSTALLATION	9
USAGE	13
WEB APP GUIDE	16
SCHEMATIC DIAGRAM	18
TROUBLESHOOTING	19
Frequently Asked Questions (FAQs)	19
CONTACT DETAILS	20

INTRODUCTION

Technology plays a vital role in healthcare. Health practitioners can use Information Technology (IT) to save and access data from a patient's health records. It also improves patient information transmission by providing a comprehensible form that anyone may use.

The researchers have decided to create a healthcare device connected to an embedded system that will monitor vital signs such as heart and pulse rate, oxygen rate, blood pressure and body temperature of the person wearing it, and the device will forecast the possibility of a cardiovascular disease from the result of the analyzation of information coming from the vital signs and data input of the user then will display the susceptibility of acquiring a heart disease.

SYSTEM REQUIREMENTS

Minimum mobile requirements

Single processor (not necessarily newest generation; at least 1 GHz)

Med-res display technology

3G capable

1GB of RAM or less

Android version 2.3

Recommended mobile requirements

OS: Android - Version 7.0 or later

Processor: Dual core cpu, at least 1.5 Ghz.

Memory: 4gb Ram

4G capable

Arduino Hardware

NodeMCU ESP32

Max30102 Heart Rate and Pulse Oximeter Sensor Module

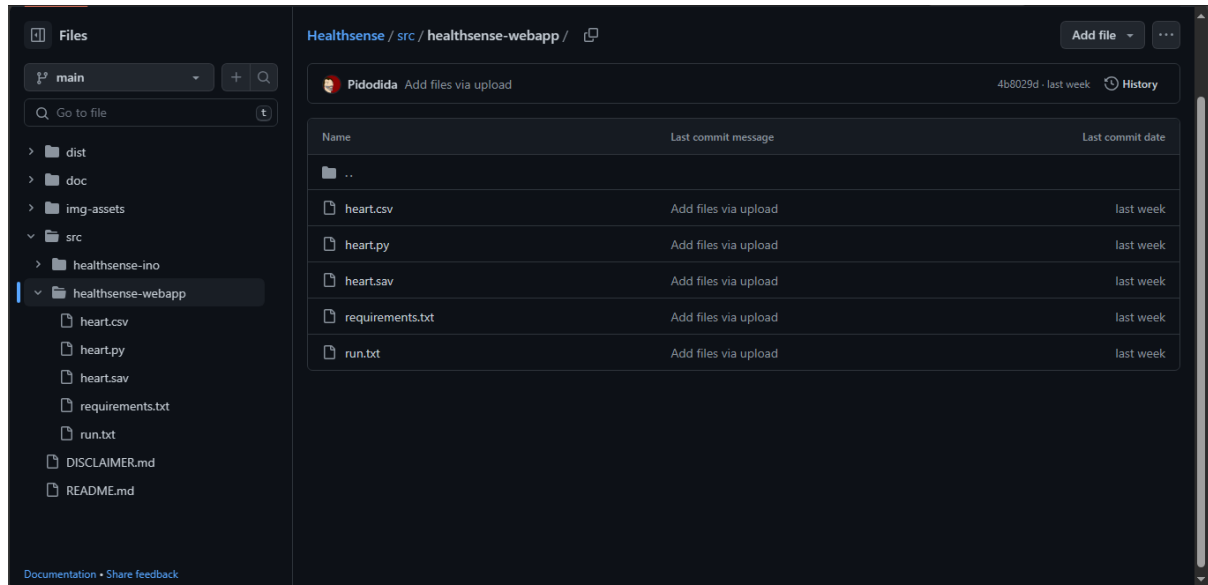
Digital Sphygmomanometer

Two AA batteries

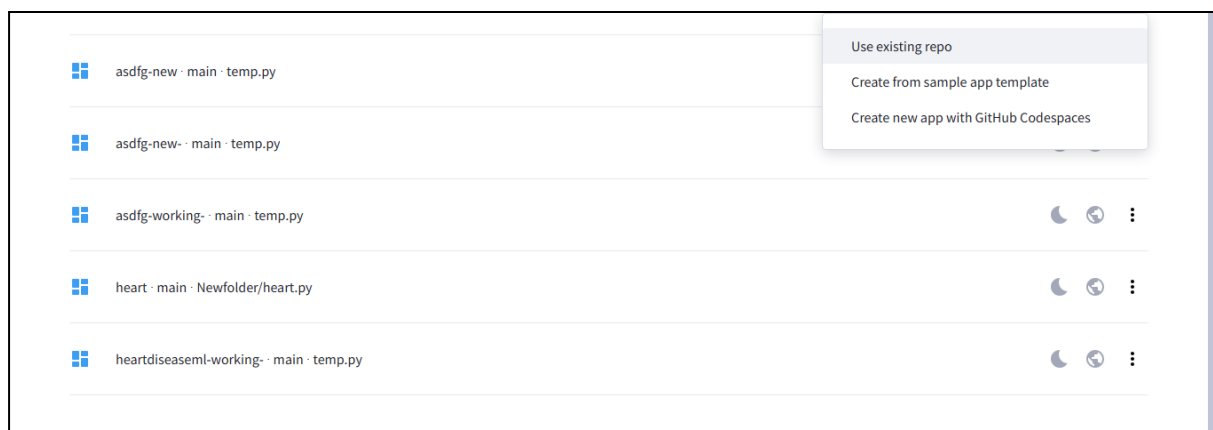
Jumper wires

HOW TO DEPLOY THE WEB APP

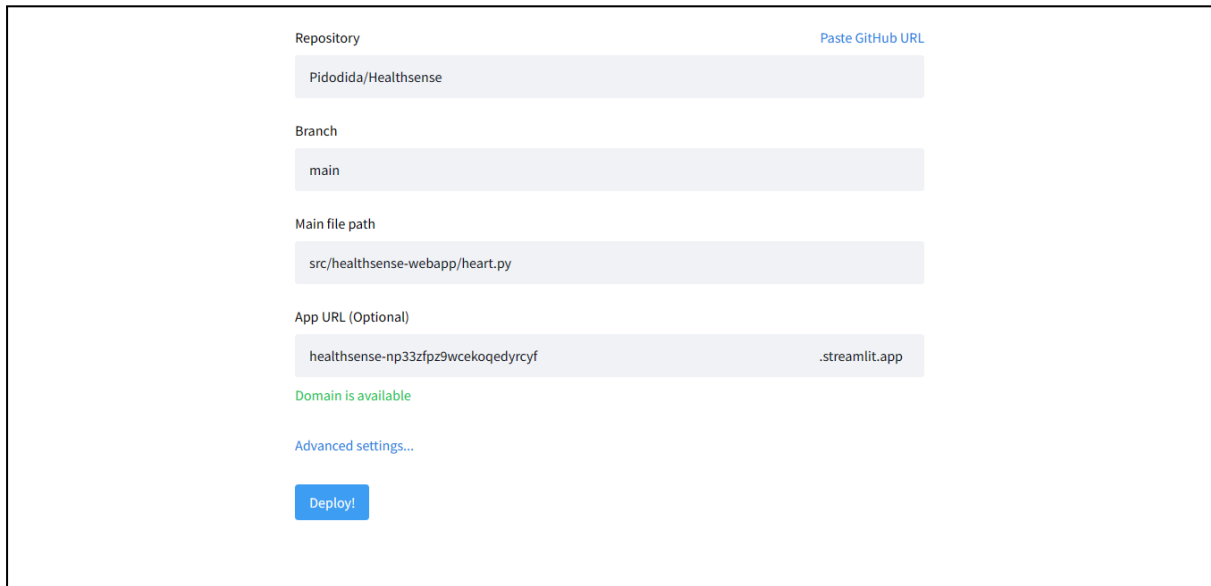
Step 1. Open the repository and copy the folder location where heart.py can be found (src/healthsense-webapp/**heart.py**)



Step 2. Open this link [Streamlit • A faster way to build and share data apps](#) and sign in using your github account (be sure you already have a copy of the repository). Then click on create apps and use existing repo, select the repository which indicates the one mentioned above.



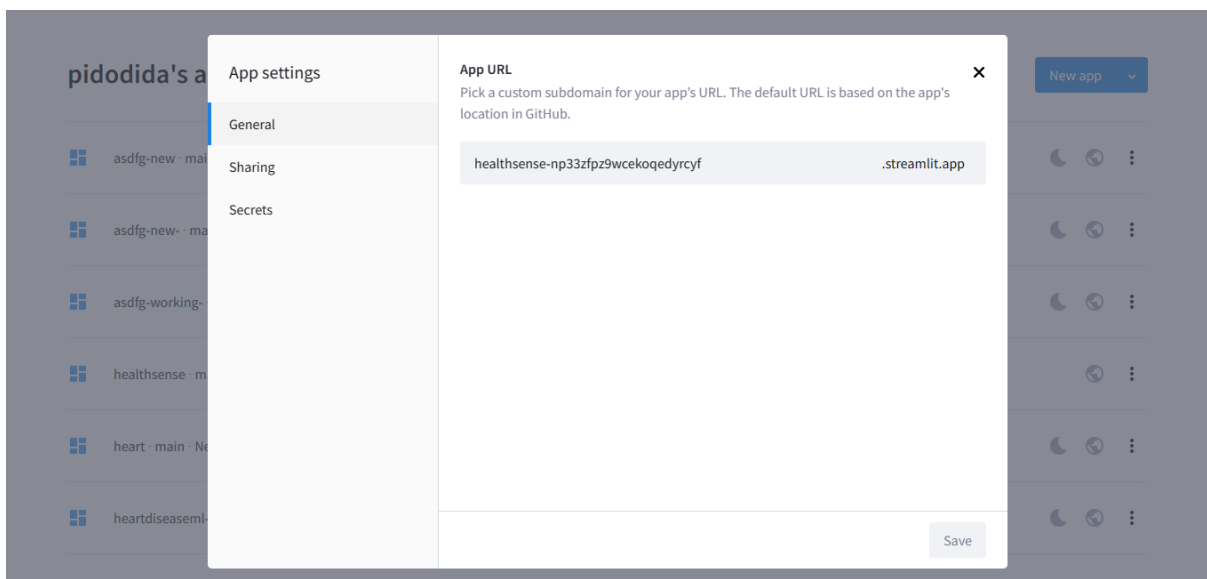
Step 3. Make sure the main file path for heart.py is correctly set and click Deploy!
(src/healthsense-webapp/**heart.py**)



A screenshot of the Streamlit deployment configuration form. It contains the following fields and options:

- Repository:** A text box containing "Pidodida/Healthsense" with a "Paste GitHub URL" link to its right.
- Branch:** A text box containing "main".
- Main file path:** A text box containing "src/healthsense-webapp/heart.py".
- App URL (Optional):** A text box containing "healthsense-np33zfpz9wcekoqedyrcyf" and a ".streamlit.app" suffix to its right.
- Domain is available:** A green text label below the App URL field.
- Advanced settings...** A blue link below the domain status.
- Deploy!** A blue button at the bottom.

Step 4. You can modify the link for the web app you set up here when signing on to streamlit



A screenshot of the Streamlit web interface showing the "App settings" dialog. The dialog has a sidebar with "General", "Sharing", and "Secrets" tabs. The "App URL" section is active, showing a text box with "healthsense-np33zfpz9wcekoqedyrcyf" and ".streamlit.app" to its right. A "Save" button is at the bottom right of the dialog. The background shows a list of apps on the left and a "New app" button on the right.

INSTALLATION

HealthSense App

Download the apk file found within the repository "Healthsense/dist/**Healthsense.apk**."

- Install the file, an alert will prompt for permission to install unknown sources , click "Allow" or "OK". This will allow the application to be installed. If not, make sure you turned off your antivirus or made an exception for the android package (apk) file.

To upload a sketch to the microcontroller these are the steps you need to take:

1. Install Arduino IDE via this link: <https://www.arduino.cc/en/software>
2. Download the necessary libraries used in this project.
 - a. Firebase ESP client downloadable here
<https://github.com/mobizt/Firebase-ESP-Client>
 - b. Firebase Arduino Client Library for ESP8266 and ESP32 within Arduino Libraries
 - c. Sparkfun MAX3010x Pulse and Proximity Sensor Library within Arduino libraries
3. In boards, download the esp32 board by Espressif Systems.
4. After setting up necessary prerequisites connect the ESP32 Microcontroller using a Micro usb type-b cable to your laptop's usb port.
5. After completing the steps above you can now download the sketch for the HealthSense system under Healthsense/ src/ healthsense-ino folder in github with a filename Healthsense-Code.ino using this link
<https://github.com/Pidodida/Healthsense/tree/main/src/healthsense-ino>.
6. Locate the file and then click upload to flash the firmware to the ESP32 Microcontroller.
7. Below attached are sample codes which are explained as guides on how to modify the code.

```

/* 1. Define the WiFi credentials */
#define WIFI_SSID "tests"
#define WIFI_PASSWORD "12345test"

/* 2. If work with RTDB, define the RTDB URL and database secret */
#define DATABASE_URL "iotdb-24ela-default-rtdb.firebaseio.com" //<databaseName>.firebaseio.com or <databaseName>.<region>.firebasedatabase.app
#define DATABASE_SECRET "AIzaSyBXdVIjqyhHqSb3Nt3pQNSZGqBQKQ5D7sk"

```

1. #define WIFI_SSID
 - a. Data type: String
 - b. This is to specify which Wi-Fi connection the system will connect to
2. #define WIFI_PASSWORD
 - a. Data type: String
 - b. This is to specify the password of the Wi-Fi connection we are using in the system.
3. #define DATABASE_URL
 - a. Data type: String
 - b. This is the URL where our system interchanges data
4. #define DATABASE_SECRET
 - a. Data type: String
 - b. This is the key code/secret key for the connection of the database to our system.
5. Attached below are the Codes to set up fixed variables used in the system.

```

/* 3. Define the Firebase Data object */
FirebaseData fbdo;

/* 4, Define the FirebaseAuth data for authentication data */
FirebaseAuth auth;

/* Define the FirebaseConfig data for config data */
FirebaseConfig config;

unsigned long dataMillis = 0;
int count = 0;

//===== Millis variable to send/store data to firebase database.
unsigned long sendDataPrevMillis = 0;
const long sendDataIntervalMillis = 10000; //--> Sends/stores data to firebase database every 10 seconds.
//=====

// Boolean variable for sign in status.
bool signupOK = false;

float store_random_Float_Val;
int store_random_Int_Val;
//
//int const PULSE_SENSOR_PIN = 34; // 'S' Signal pin connected to A0

int Signal;          // Store incoming ADC data. Value can range from 0-1024
int Threshold = 2000; // Determine which Signal to "count as a beat" and which to ignore.
int val;
int tempPin = A0;
PulseOximeter pox;

```

6.

```
void setup() {
  Serial.begin(115200);

  #if defined(ARDUINO_RASPBERRY_PI_PICO_W)
    multi.addAP(WIFI_SSID, WIFI_PASSWORD);
    multi.run();
  #else
    WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
  #endif

  Serial.print("Connecting to Wi-Fi");
  unsigned long ms = millis();
  while (WiFi.status() != WL_CONNECTED)
  {
    Serial.print(".");
    delay(300);
  }
  #if defined(ARDUINO_RASPBERRY_PI_PICO_W)
    if (millis() - ms > 10000)
      break;
  #endif
  Serial.println();
  Serial.print("Connected with IP: ");
  Serial.println(WiFi.localIP());
  Serial.println();

  Serial.printf("Firebase Client v%s\n\n", FIREBASE_CLIENT_VERSION);

  /* Assign the certificate file (optional) */
  // config.cert.file = "/cert.cer";
  // config.cert.file_storage = StorageType::FLASH;

  /* Assign the database URL and database secret(required) */
  config.database_url = DATABASE_URL;
  config.signer.tokens.legacy_token = DATABASE_SECRET;

  Firebase.reconnectWiFi(true);
```

```

/* Initialize the library with the Firebase authen and config */
Firebase.begin(&config, &auth);

// Or use legacy authenticate method
// Firebase.begin(DATABASE_URL, DATABASE_SECRET);
}

void loop() {

// Signal = analogRead(PULSE_SENSOR_PIN);
// int pulse = Signal / 30;
val = analogRead(tempPin);
float mv = (val / 1024.0)*500;
float cel = mv / 10;
Serial.println(cel);
pox.update();

    if (millis() - tsLastReport > REPORTING_PERIOD_MS) {
        Serial.print("Heart BPM:");
        Serial.print(pox.getHeartRate());
        Serial.print("-----");
        Serial.print("Oxygen Percent:");
        Serial.print(pox.getSpO2());
        Serial.println("\n");
        tsLastReport = millis();
    }
    if (Firebase.ready() && signupOK && (millis() - sendDataPrevMillis > 5000 || sendDataPrevMillis == 0)){
        sendDataPrevMillis = millis();
        // Write an Int number on the database path test/int
        if (Firebase.RTDB.setInt(&fbdo, "Database/pulserate", pulse)){
            Serial.println("PASSED");
            Serial.println("PATH: " + fbdo.dataPath());
            Serial.println("TYPE: " + fbdo.dataType());
        }

        else {
            Serial.println("FAILED");
            Serial.println("REASON: " + fbdo.errorReason());
        }
        if (Firebase.RTDB.setInt(&fbdo, "/Database/temperature", cel)){
            Serial.println("PASSED");
            Serial.println("PATH: " + fbdo.dataPath());
            Serial.println("TYPE: " + fbdo.dataType());
        }
        else {
            Serial.println("FAILED");
            Serial.println("REASON: " + fbdo.errorReason());
        }
        if (Firebase.RTDB.setInt(&fbdo, "/Database/temperature", cel)){
            Serial.println("PASSED");
            Serial.println("PATH: " + fbdo.dataPath());
            Serial.println("TYPE: " + fbdo.dataType());
        }
        else {
            Serial.println("FAILED");
            Serial.println("REASON: " + fbdo.errorReason());
        }
    }
    // if Signal
    // Serial.print("Pulse Rate = ");
    // Serial.println(pulse + " BPM");
    //                                     // Send the signal value to serial plotter
    //
    //
    //
}

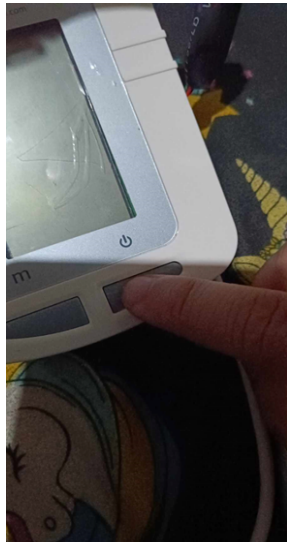
```

USAGE

1. Connect the device to power e.g. usb connector. The arduino device will be working promptly. It will automatically connect the device to the ISP provided in the Arduino code.



2. Press the power button on the digital blood sphygmomanometer to start reading your blood pressure.



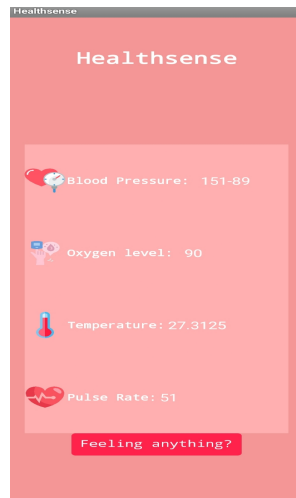
3. Secure the strap of the digital sphygmomanometer to your arm. Be calm and relaxed when getting your blood pressure.



4. Place your finger directly on the IR sensor to record your pulse and heart rate.



5. In the mobile application, the vital signs of the user, namely, Blood pressure, Oxygen Level, Temperature and Pulse rate are on display and are updated real-time.



6. The "Feeling anything?" button leads to the WebApp where the user can learn if they are susceptible to heart disease or not.



WEB APP GUIDE

The screenshot shows a web application titled "Heart Disease Prediction System". On the left is a sidebar with a "Home" button. The main area is titled "Heart Disease Prediction using ML" and contains several input fields for user data: Age (1), Fasting Blood Sugar > 120mg/dl (0), Exercise induced Angina (0-1) (0), Sex (0), Resting ECG (0-2) (0), Thalassemia (1-3) (1), Chest Pain (0-3) (0), and Maximum Heart rate (0). Below these fields is a "See Results" button. A callout box points to the "See Results" button with the text: "See Results" displays the text prompt below according to gathered input from the user. Below the button, a green box contains the text: "IF THIS TEXT IS SHOWING PLEASE INPUT YOUR DATA". In the top right corner, there are links for "Share", "link for github repository", and "Settings and developer options". A "Manage app" button is in the bottom right corner.

The user will be using the medical device to gather data needed using sensors embedded in the system. It will then be directly saved in the database. After being saved in the database, the data will be shown in the mobile app. Followed by the user having to input the remaining data needed to predict his/her probability of acquiring a heart disease.

This block shows a close-up of the first three input fields from the web app. Each field is labeled with a circled number: 1. "Age" with a value of 1. 2. "Sex" with a value of 0. 3. "Chest Pain (0 - 3)" with a value of 0. Each input field has minus and plus buttons for adjustment.

1.) age (Ages exceeding 77 will have less accurate results)

2.) sex (Male=1, Female=0)

3.) chest pain is determined by four values ranging from highest to lowest in pain level (0-4)

8	Fasting Blood Sugar > 120mg/dl	6	Exercise induced Angina (0 - 1)
	0 - +		0 - +
4	Resting ECG (0 - 2)	7	Thalassemia (1 - 3)
	0 - +		1 - +
5	Maximum Heart rate		
	0 - +		

4.) resting electrocardiographic results

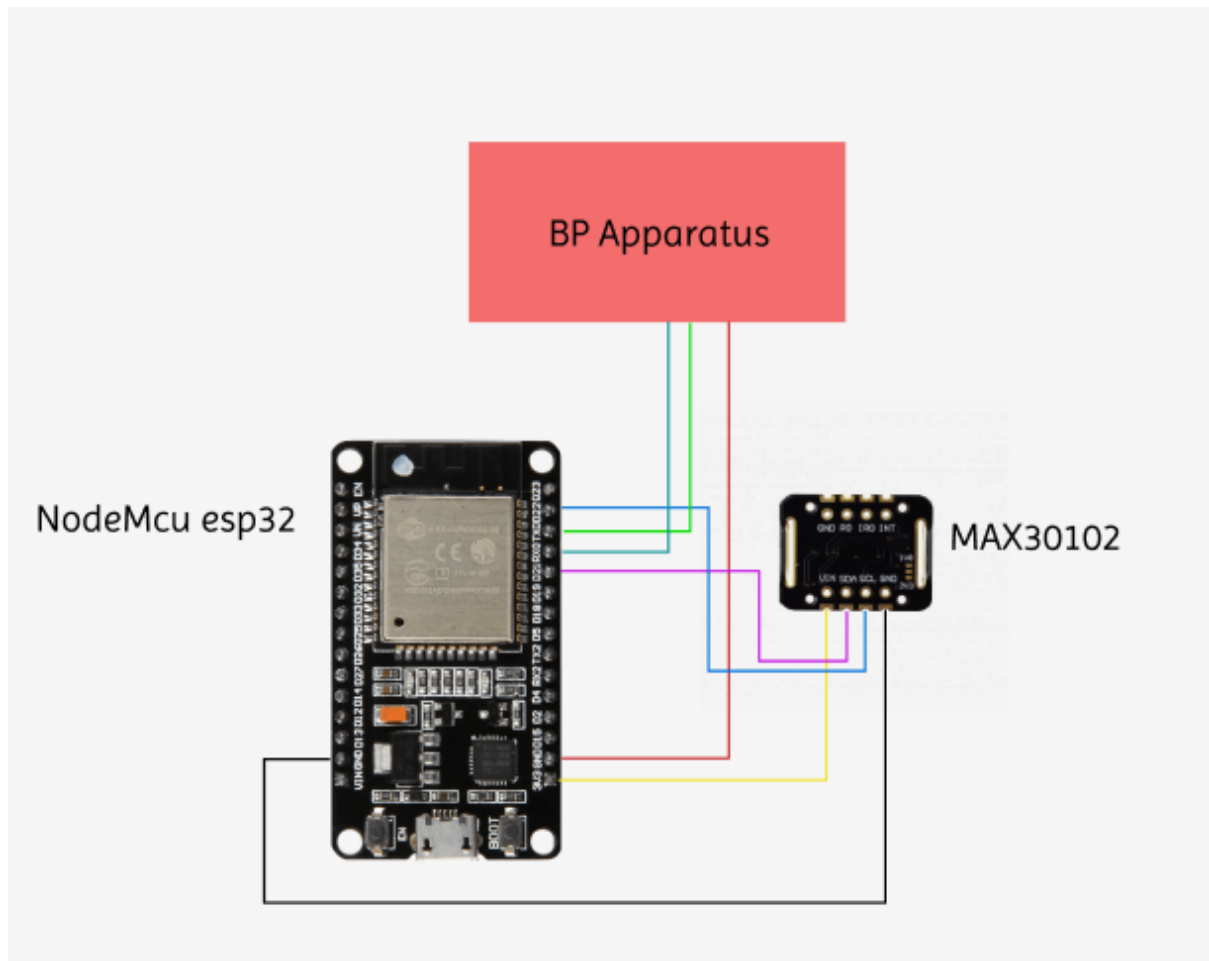
5.) maximum heart rate achieved

6.) exercise induced angina

7.) thal: 1 = normal; 2 = fixed defect; 3 = reversible defect

8. fasting blood sugar > 120 mg/dl

SCHEMATIC DIAGRAM



1. NodeMcu esp32: It is the main component where the processing part of information is cycling around and records the data which is then sent to the database.
2. Max30102: The module features the MAX30102. It combines two LEDs, a photodetector, optimized optics, and low-noise analog signal processing to detect pulse oximetry (SpO2) and heart rate (HR) signals.
3. BP Apparatus: records the blood pressure which is then sent to the microcontroller.

TROUBLESHOOTING

Device does not turn on.

Check if the power cord is plugged properly and double check if the AA battery is functional.

Vital signs are not showing on the mobile app.

Check if your phone is connected on the internet and restart the arduino device. It should work after that. Also, try a close contact of your finger to the IR sensor and a tight strap of digital BP apparatus on your arm.

Application is not launching/ not responding.

Re-launch the app and restart your device if the app is malfunctioning.

Webapp is not launching.

Refresh the webapp, if the problem persists, exit and open the application again.

Frequently Asked Questions (FAQs)

Is the application available for iOS?

No. HealthSense application is only available on Android devices.

Is HealthSense free to download and use?

Yes. The HealthSense application is free to use and download for everyone that can meet the app's system specifications.

Is the internet needed in order to use the app?

Yes, the Internet is needed to use the app.

CONTACT DETAILS

Ana Patricia Felasol

09077203108

anapatricia.felasol@wvsu.edu.ph

Thrys Formoso

09087611912

thrys.formoso@wvsu.edu

Patricia Anne Gaquit

09185653156

patriciaanne.gaquit@wvsu.edu.ph

John Kate Marbebe

09183444228

johnkate.marbebe@wvsu.edu.ph

Luke Ian Undar

09476395178

lukeian.undar@wvsu.edu.ph

