Qatar University

College of Engineering

Department of Computer Science and Engineering

Course Project - CMPE 462 Computer Interfacing

IoT Based Patient Health Monitoring System

Submitted to: Prof. Muhammed Azeem

Project Group Member:

Marim Elhanafy 201803468
Hagar Elsayed 201805123
Tasfia Muslim Uddin 201800789

Date: 6-12-2022

Table of Contents

List Of Figures

List Of Tables

1. Problem Statement

Health monitoring is the primary issue in today's world. Patient has major health problems because of improper health monitoring. To track a patient's health these days, there are several IoT devices available. Health professionals are also using these smart devices for patients' health tracking. IoT is rapidly transforming the healthcare system, with hundreds of new digital healthcare start-ups. Our goal is to develop a real-time health system based on IoT using user-interface to monitor patient health parameters like heart rate and blood oxygen level along with body temperature. Using numerous sensors, it will measure patient data in real-time for keeping an eye on patient's health status. The device is designed to be used by the patient directly or by nurses in health centers.

2. Objectives

I. To develop a device for health tracking using different sensors to help patients and healthcare providers.
II. To help patients with chronic diseases to track their oxygen and heart rate level periodically.
III. To make an affordable User-friendly device.

3. High-Level Architecture

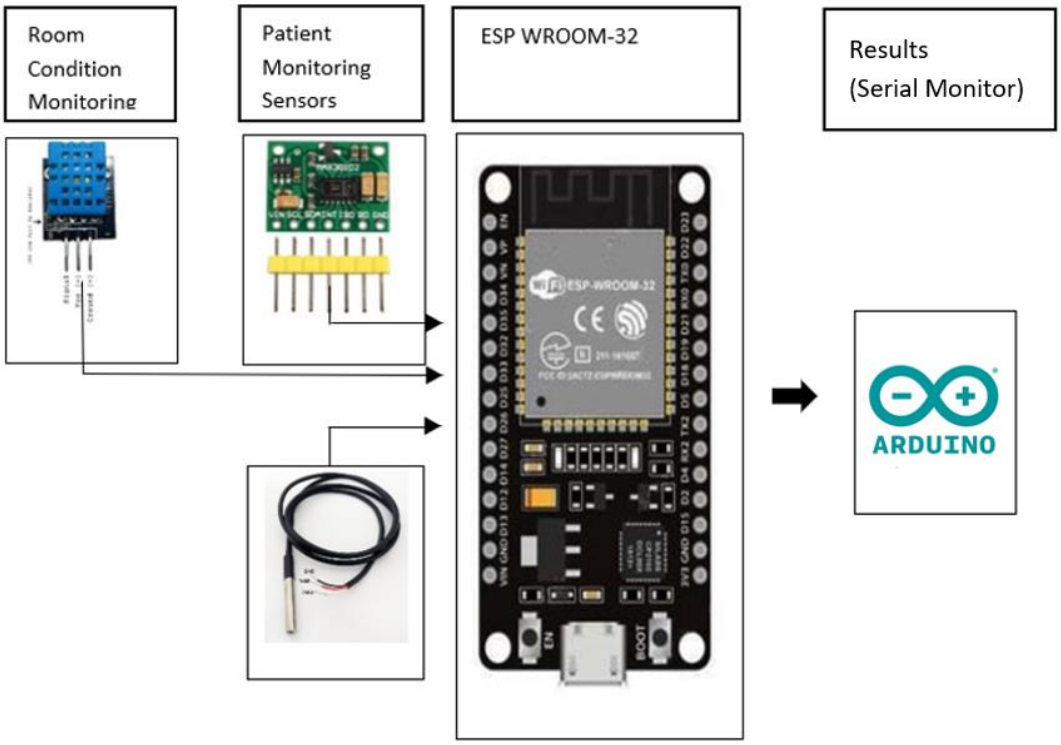


Figure 1: High-level Architecture

4. Design Details

4.1 Hardware

4.1.1 Connectivity Diagram

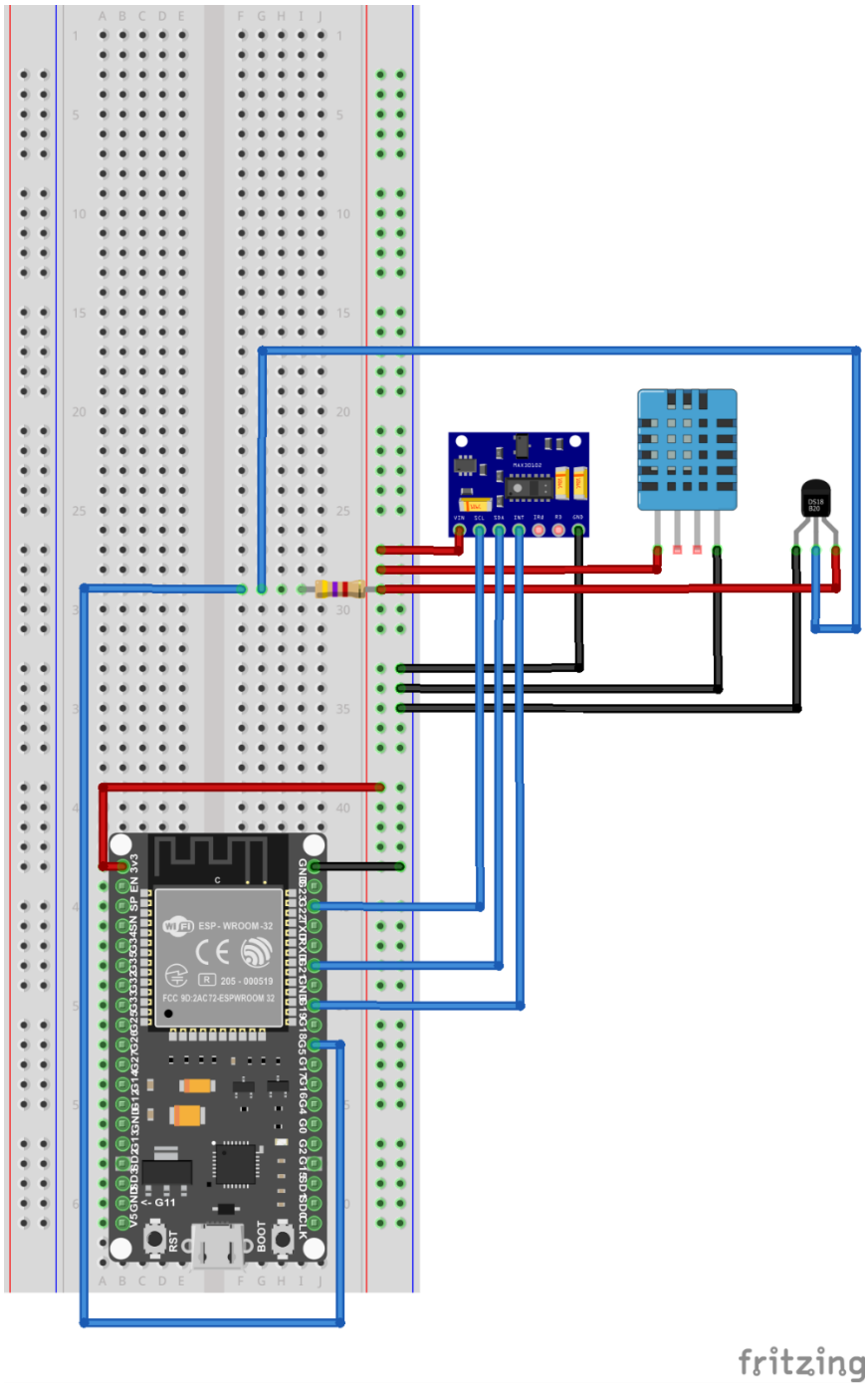
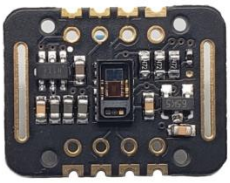
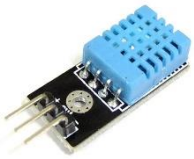
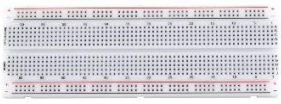


Figure 2: Overall connectivity of the system.

4.1.2 Hardware Used

Table 1 contains a list of all hardware components used and their purpose in the project.

Table 1: List of the components used in the project.

Component		Quantity	Usage
ESP-32S Development Board (ESP-WROOM-32) [1]		1	The ESP32 is a low-cost, low-power Microcontroller with Wi-Fi and Bluetooth built-in. ESP32 is used to implement the sensors used for tracking patients' health.
MAX30102 I2C Pulse Oximeter &Heart rate Sensor [2]		1	The MAX30102 pulse oximeter and heart rate sensor is an I2C-based low-power plug-and-play biometric sensor. It is used in two parts one is heart rate measurement and another is blood oxygen level measurement.
DS18B20 One-Wire Waterproof Temperature Sensor [3]		1	The DS18B20 sensor measures temperature with a minimal amount of hardware and wiring. It is used to send accurate temperature readings directly to the development board.
DHT11 Digital Humidity Temperature Sensor [4]		1	The DHT11 is a basic, ultra-low-cost digital temperature and humidity sensor. It uses a capacitive humidity sensor and a thermistor to measure the surrounding air.
Resistor		1	The resistors are predominantly used to lower the flow of current, divide voltages, block transmission signals, and bias active elements.
Connecting Wires		7-10	Wires are used to connect sensors with ESP32 in breadboard.
Breadboard		1	Breadboard is used in making all the sensors connect with the ESP32 using connecting wires.

4.1.3 Novel Aspects

The product is a real-time device for tracking the health status of patients, especially those with chronic diseases and the elderly. The product will serve the market because we need such devices that measure many vital and periodic measurements to ensure the patient's health. The product is a cost-effective approach and will save time for healthcare providers. It is using sensors which will obtain patients data which are commonly required, like body temperature, oxygen level and heart rate following with a sensor like temperature and humidity to know about the surrounding the patient is in.

4.2 Software

4.2.1 Flow Chart

The Logic flowchart represents the whole system logic of data flow as shown in Figure 3.

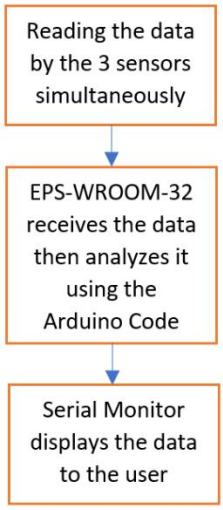


Figure 3: Proposed software architecture.

4.2.2 Software Used

In terms of software selection, Fritzing software is used to design the circuit. Also, Arduino IDE is used to code the Arduino and display the results. The idea is as follows, the ESP32 will take the measurements from three different sensors. MAX30102 measures the heart rate and the blood oxygen level, DS18B20 measures body temperature, and DHT11 measures room temperature and humidity. Then, ESP32 will receive and analyze the data of three sensors to display it to the user in the Serial Monitor.

Figure 4: Software IDEs used in the project.

5. Implementation and Testing

5.1 Actual Implementation

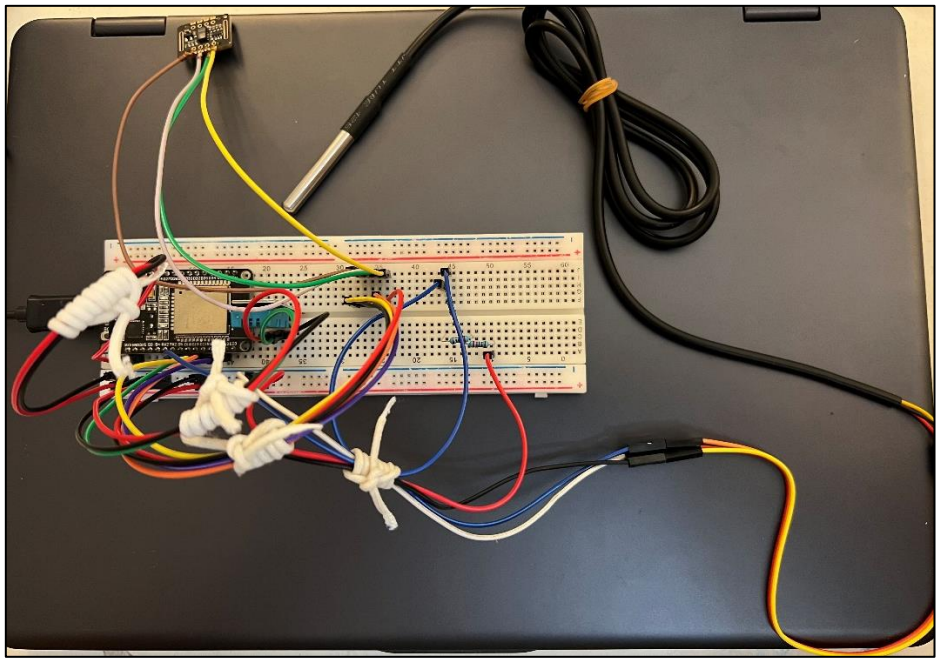


Figure 5: Actual implementation of the project.

*** Please consider comments on Appendix A during the hardware implementation and running the code.**

5.2 Complete Code

Arduino IDE Code:

```
//////////////////// 1. MAX30102  ////////////////////
#include "MAX30105.h"      //MAX3010x library
#include <Wire.h>
#include "heartRate.h"     //Heart rate calculating algorithm
#include "ESP32Servo.h"
const byte RATE_SIZE = 10;
MAX30105 particleSensor;
byte rates[RATE_SIZE];
byte rateSpot = 0;
long lastBeat = 0;         //Time at which the last beat occurred
float beatsPerMinute;
double avered = 0;
double aveir = 0;
double sumirrms = 0;
double sumredrms = 0;
double SpO2 = 0;
double ESpO2 = 90.0;
```

```
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
int beatAvg;
int i = 0;
int Num = 30;
#define FINGER_ON 7000
#define MINIMUM_SPO2 90.0
//////////////////// 2. DHT11     ////////////////////
#include "DHT.h"
#define DHTPIN 18         //Digital pin connected to the DHT sensor
#define DHTTYPE DHT11     //type we're using! (DHT 11)
DHT dht(DHTPIN, DHTTYPE); //Initialize DHT sensor.
//////////////////// 3. DS18B20   ////////////////////
#include <OneWire.h>
#include <DallasTemperature.h>
const int oneWireBus = 5;             //GPIO where the DS18B20 is connected to
OneWire oneWire(oneWireBus);          //Setup a oneWire instance to
communicate with any OneWire devices
DallasTemperature sensors(&oneWire);  //Pass our oneWire reference to Dallas
Temperature sensor
////////////////////////////////////////////////////////////////////////////////
////////////////////////////
void MAX30102_Function(void * parameter)
{
  for(;;)
  {
    //Reading the IR value
    //(it will permit us to know if there's a finger on the sensor or not)
    long irValue = particleSensor.getIR();
    if (irValue > FINGER_ON )
    {
      Serial.print(beatAvg);
      Serial.println(" BPM");
      if (beatAvg > 30)
      {
        Serial.print(String(ESpO2));
        Serial.println(" %");
      }
      else  Serial.println("---- %");
      if (checkForBeat(irValue) == true)
      {
        Serial.print(beatAvg);
        Serial.println(" BPM");
        if (beatAvg > 30)
        {
          Serial.print(String(ESpO2));
          Serial.println(" %");
```

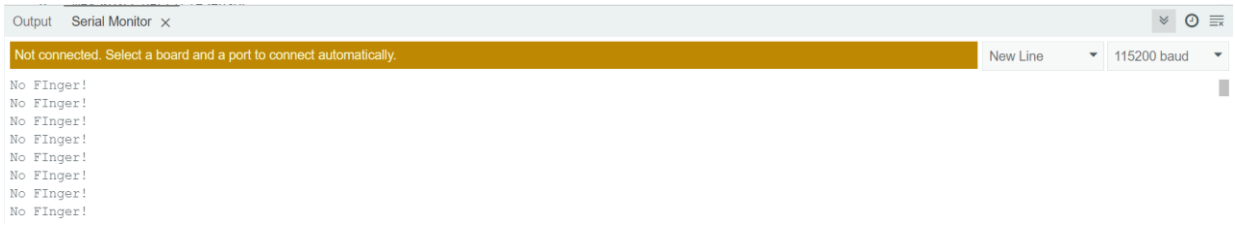
```
      }
      else  Serial.println("---- %");
      Serial.print("beatAvg=");
      Serial.println(beatAvg);
      long delta = millis() - lastBeat;
      lastBeat = millis();
      beatsPerMinute = 60 / (delta / 1000.0);
      if (beatsPerMinute < 255 && beatsPerMinute > 20)
      {
        rates[rateSpot++] = (byte)beatsPerMinute;
        rateSpot %= RATE_SIZE;
        beatAvg = 0;
        for (byte x = 0 ; x < RATE_SIZE ; x++) beatAvg += rates[x];
        beatAvg /= RATE_SIZE;
      }
    }
    uint32_t ir, red;
    double fred, fir;
    //Check the sensor, read up to 3 samples
    particleSensor.check();
    if (particleSensor.available())
    {
      i++;
      red = particleSensor.getFIFOIR();
      ir = particleSensor.getFIFORed();
      fred = (double)red; //double
      fir = (double)ir;   //double
      //average red level by low pass filter
      avered = avered * frate + (double)red * (1.0 - frate);
      //average IR level by low pass filter
      aveir = aveir * frate + (double)ir * (1.0 - frate);
      //square sum of alternate component of red level
      sumredrms += (fred - avered) * (fred - avered);
      //square sum of alternate component of IR level
      sumirrms += (fir - aveir) * (fir - aveir);
      if ((i % Num) == 0)
      {
        double R = (sqrt(sumredrms) / avered) / (sqrt(sumirrms) / aveir);
        SpO2 = -23.3 * (R - 0.4) + 100;
        ESpO2 = FSpO2 * ESpO2 + (1.0 - FSpO2) * SpO2;     //low pass filter
        if (ESpO2 <= MINIMUM_SPO2) ESpO2 = MINIMUM_SPO2;  //indicator for
finger detached
        if (ESpO2 > 100) ESpO2 = 99.9;
        Serial.print("Oxygen % = ");
        Serial.println(ESpO2);
        sumredrms = 0.0; sumirrms = 0.0; SpO2 = 0;
        i = 0;
      }
```

```
        particleSensor.nextSample();
      }
    }
    else
    {
      for (byte rx = 0 ; rx < RATE_SIZE ; rx++) rates[rx] = 0;
      beatAvg = 0; rateSpot = 0; lastBeat = 0;
      avered = 0; aveir = 0; sumirrms = 0; sumredrms = 0;
      SpO2 = 0; ESpO2 = 90.0;
      Serial.println("No FInger!");   //Finger Please
    }
  }
}
void DHT11_Function(void * parameter)
{
  for(;;)
  {
    delay(2000); //Wait a few seconds between measurements.
    //Reading temperature or humidity takes about 250 milliseconds!
    //Sensor readings may also be up to 2 seconds (its a very slow sensor)
    float h = dht.readHumidity();
    //Read temperature as Celsius (the default)
    float t = dht.readTemperature();
    //Check if any of the readings failed
    if (isnan(h) || isnan(t))
    {
      Serial.println(F("Failed to read from DHT sensor!"));
      return; //exit early (to try again)
    }
    Serial.print(F("Room Humidity: "));
    Serial.print(h);
    Serial.print(F("%  Room Temperature: "));
    Serial.print(t);
    Serial.println(F("°C "));
  }
}
void DS18B20_Function(void * parameter)
{
  for(;;)
  {
    sensors.requestTemperatures();
    float temperatureC = sensors.getTempCByIndex(0);
    Serial.print(F("Body Temperature: "));
    Serial.print(temperatureC);
    Serial.println(F("°C "));
    delay(5000);
  }
}
```

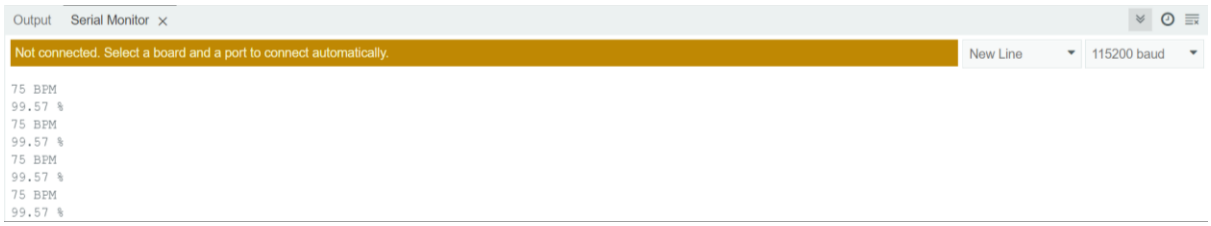
```
////////////////////////////////////////////////////////////////////////////
//////////////////////////
void setup()
{
  Serial.begin(115200); //Start the Serial Monitor
  Serial.println("System Start");
  ////////////////////// 1. MAX30102 ///////////////////
  if (!particleSensor.begin(Wire, I2C_SPEED_FAST))    //Use default I2C port,
400kHz speed
  {
    Serial.println("MAX30102");
    while (1);
  }
  //Set up the wanted parameters
  byte ledBrightness = 0x7F;
  byte sampleAverage = 4;
  byte ledMode = 2;
  int sampleRate = 800;
  int pulseWidth = 215;
  int adcRange = 16384;
  //Configure sensor with these settings
  particleSensor.setup(ledBrightness, sampleAverage, ledMode, sampleRate,
pulseWidth, adcRange);
  particleSensor.enableDIETEMPRDY();
  //Turn Red LED to low to indicate sensor is running
  particleSensor.setPulseAmplitudeRed(0x0A);
  //Turn off Green LED
  particleSensor.setPulseAmplitudeGreen(0);
  ////////////////////// 2. DHT11     ///////////////////
  Serial.println(F("DHTxx test!"));
  dht.begin();  //ask the sensor to start working
  ////////////////////// 3. DS18B20   ///////////////////
  sensors.begin(); //Start the DS18B20 sensor
  ////////////////////////////////////////////////////////////////////////////
//////////////////////////////
  //Using RTOS multitasking principle to run the three sensors simultaneously
  xTaskCreatePinnedToCore(MAX30102_Function, "beatAvg & ESpO2", 2000, NULL, 1,
NULL, 1);
  xTaskCreatePinnedToCore(DHT11_Function, "Room Humidity & Temperature", 1000,
NULL, 1, NULL, 1);
  xTaskCreatePinnedToCore(DS18B20_Function, "Body Temperature", 1000, NULL, 1,
NULL, 1);
}
////////////////////////////////////////////////////////////////////////////
//////////////////////////
void loop() { }
```

5.3 Testing

1. MAX30102:
    a. Without Finger:



    b. With Finger:



2. DHT11:

3. DS18B20:

5.4 Discussion

Table 2: Comparison between the tolerable ranges and the practical measurements

	Tolerable Range	Practical Measurement
Heart Rate	56 – 104 beats per minute (BPM) [5]	75 BPM
SPO2	94 – 98 % [6]	99.57 %
Room Humidity	between approximately 25% RH and 60% to 70% RH [7]	59 %
Room Temperature	25 – 28 °C [8]	25.8 °C
Body Temperature	35.5–37.0 °C [9]	34.56 °C

According to Table 2, most of the readings fall within their respective tolerable ranges. While it is demonstrated that the heart rate and room temperature readings follow the expected range, the SPO2, room humidity, and body temperature measurements were found to be slightly in error.

6. Author's Contributions

Table 3: Task's Distribution

Task	Done By
High-Level Architecture	All
Hardware Design	All
Software Design	All
Hardware Implementation	Marim Elhanafy – Hagar Elsayed
Arduino Code and Testing	Marim Elhanafy
Report	Marim Elhanafy – Tasfia Muslim Uddin
PowerPoint	Hagar Elsayed

References

[1] **ESP-WROOM-32 Datasheet:** esp32_datasheet_en.pdf (espressif.com)

[2] **MAX30102 Datasheet:** MAX30102--High-Sensitivity Pulse Oximeter and Heart-Rate Sensor for Wearable Health (smart-prototyping.com)

[3] **DS18B20 Datasheet:** DS18B20.pdf (shopify.com)

[4] **DHT11 Datasheet:** DHT11 Humidity & Temperature Sensor (mouser.com)

[5] Umetani K, Singer D, McCraty R, et al. Twenty-Four Hour Time Domain Heart Rate Variability and Heart Rate: Relations to Age and Gender Over Nine Decades. J Am Coll Cardiol. 1998 Mar, 31 (3) 593–601. https://doi.org/10.1016/S0735-1097(97)00554-8

[6] Aneman, A. (2012). SpO2 targets—How normal is normal?. Resuscitation, 83(10), 1175-1176. https://doi.org/10.1016/j.resuscitation.2012.07.026.

[7] Toftum, J., & Fanger, P. O. (1999). Air humidity requirements for human comfort. ASHRAE Transactions, 105, 641. Retrieved from http://0-search.proquest.com.mylibrary.qu.edu.qa/scholarly-journals/air-humidity-requirements-human-comfort/docview/192573404/se-2

[8] Meng ZL. [Studies on the health standard for room temperature in cold regions]. Zhonghua Yu Fang Yi Xue Za Zhi. 1990 Mar;24(2):73-6. Chinese. PMID: 2364801.

[9] Sund-Levander, M., Forsberg, C., & Wahren, L. K. (2002). Normal oral, rectal, tympanic and axillary body temperature in adult men and women: a systematic literature review. Scandinavian journal of caring sciences, 16(2), 122-128. https://doi.org/10.1046/j.1471-6712.2002.00069.x

Appendix A:

1. For MAX30102 please consider the following:
    a. The sensor is used to show heart rate and SPO2 levels and it uses the Optical SP02 Detection (SPK Algorithm) using the MAX30105 Breakout
    b. When you press your finger against the sensor it varies enough to cause the blood in your finger to flow differently which causes the sensor readings to go wonky.
    c. Hardware Connections (Breakout board to Arduino):
        i. 5V = 5V (3.3V is allowed)
        ii. GND = GND
        iii. SDA = A4 (or SDA)
        iv. SCL = A5 (or SCL)
2. For DHT11 please consider the following:
    a. To read from the DHT sensor, we will use the DHT library from Adafruit.
    b. Connect pin 1 (on the left) of the sensor to +5V
    c. Connect pin 2 of the sensor to whatever your DHTPIN is
    d. Connect pin 3 (on the right) of the sensor to GROUND
3. For DS18B20 please consider the following:
    a. To interface with the DS18B20 temperature sensor, you need to install the One Wire library by Paul Stoffregen and the Dallas Temperature library.
    b. New temperature readings are requested every 5 seconds.