VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

CHENNAI

SELF PULSE & BLOOD-OXYGEN MONITORING SYSTEM

by

Bera Rahul Dilip [18BCE1142]

Prakrit Raj [19BCE1865]

A Project Report submitted to

Dr. Pradeep Kumar TS

in partial fulfilment of the requirements of the course

CSE3009 - Internet of Things

in

B.Tech Computer Science Engineering

November 2020

CERTIFICATE

This is to certify that the Project work entitled **"Self Pulse and Blood-Oxygen Monitoring System"** that is being submitted by Bera Rahul Dilip [18BCE1142] and Prakrit Raj [19BCE1865] for CAL in B.Tech, Internet of Things (CSE3009) during the Fall Semester 2020-21 is a record of bonafide work done under my supervision and guidance for the course.

STUDENT SIGNATURE:

Bera Rahul Dilip (19BCE1142)

Prakrit Raj (19BCE1865)

FACULTY SIGNATURE:

Dr. Pradeep Kumar T S

Associate Professor

School of Computer Science Engineering

ABSTRACT

Covid-19 has now become a global pandemic and cases are on the rise in India, the Govt. is promoting self-isolation and home quarantine as the hospitals are running out of beds. There is a need for a self-monitoring system. In this Project, we offer an IoT-based Pulse and Blood-Oxygen Monitoring System which will allow doctors to monitor the pulse and Blood-Oxygen level of isolated people remotely through an app. Since Covid-19 has the symptoms of Pneumonia, where the blood oxygen level drops down our device sends a notification over the app informing the doctor to attend the person immediately. Also, it notifies the patient that they may need oxygen cylinders as early as possible to maintain the blood-oxygen level.

ACKNOWLEDGEMENT

We wish to express our sincere thanks and deep sense of gratitude to our project guide, **Dr. Pradeep Kumar T S,** Associate Professor, School of Computer Science Engineering, for his consistent encouragement and valuable guidance offered to us in a pleasant manner throughout the course of the project work.

We are extremely grateful to **Dr. Jagadeesh Kannan R,** Dean of School of Computer Science Engineering, VIT Chennai, for extending the facilities of the School towards our project and for his unstinting support and all the faculty of the School for their support and their wisdom imparted to us throughout the course.

We thank our parents, family, and friends for bearing with us throughout the course of our project and for the opportunity they provided us in undergoing this course in such a prestigious institution.

RAHUL BERA PRAKRIT RAJ

TABLE OF CONTENTS:

INTRODUCTION:

With the inevitable increase in the number of COVID-19 affected patients, the hospitals are finding it hard to allot beds for new patients and doctors are finding it impossible to monitor them. Since COVID-19 has the symptoms similar to that of Pneumonia, a sudden decrease in the Blood oxygen level and an irregular Heart Beat is noted for patients, who are in the recovery stage. As the doctors cannot monitor the recovering patients frequently and they are generally discharged to their homes, providing quick help becomes impractical.

In this Project, we will be trying to make an IoT-based Pulse and Blood-Oxygen Monitoring System which will allow doctors to monitor the pulse and Blood-Oxygen level of isolated people remotely through the Blynk IoT App. Our device will periodically collect the parameters from the sensors and send it to the Blynk IoT App. The app will send this data to a Cloud database. Whenever the blood oxygen level drops down below the general level, the App will notify the consulted doctor by sending notifications. Also, it signals the patient that they may need oxygen cylinders as early as possible to maintain the blood-oxygen level.

FEATURES:

- The Device setup consists of sensors, a controller and an Android application.
- The setup can support data access from multiple accounts.
- It also supports Bluetooth connectivity in addition to Wi-Fi connectivity.
- The sensor used, consumes very less power
    - Voltage- 1.8 to 3.3 V
    - Current- 0.7μA

SYSTEM DESIGN:

In this project, we are having a sensor node to sense the pulse and blood oxygen level. The data is converted into standard units by the Microcontroller and is sent to be stored on web servers and is also displayed in the mobile app as it is. Also, the app has to send the notification to the doctor as a message or prompt informing that the patient's oxygen level has dropped below or is reaching the risk. In addition to this, the data can be accessed from multiple devices.

Considering all these factors, it is implemented as a Level-2 IoT system.

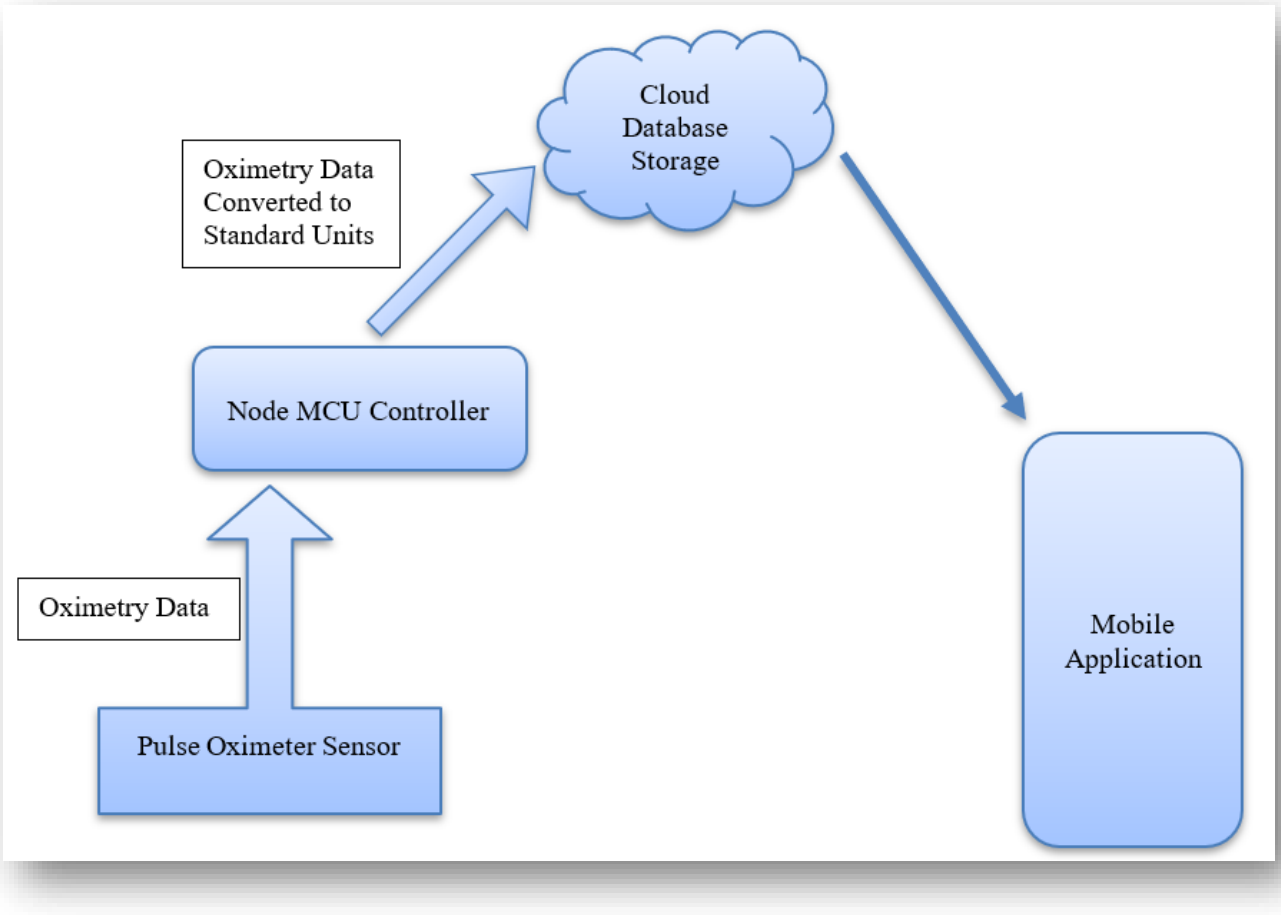


Figure 1: Block Diagram of the System

COMPONENTS:

HARDWARE:

- Node MCU ESP8266 Wi-Fi Module Ver 1.0
- MAX30102 Pulse Oximeter and Heart-Rate Sensor
- Bread Board
- Connection Wires
- Soldering Kit
- USB Data Cable
- LED

SOFTWARE:

- Blynk IoT Application (Android/ioS)
- Arduino IDE
- Blynk Libraries

ABOUT THE SENSOR:

The MAX30102 Pulse Oximeter and Heart-Rate Sensor is being used in the device to detect heart beats and Oxygen Level through the blood stream of a person.

WORKING:

- The detection of oxygen level is carried out by the red and infrared LEDs.
- When heart pumps blood, it is oxygenated (i.e. more concentration of O_2) whereas when the blood returns back it has less O_2 concentration.
- The oxygenated blood absorbs more infrared light while the deoxygenated blood absorbs more red light.
- Photoreceptors detect the amount of light absorbed to determine the blood-oxygen level.

- The time difference between the detection of Oxygenated and Deoxygenated blood helps in determining the pulse rate.

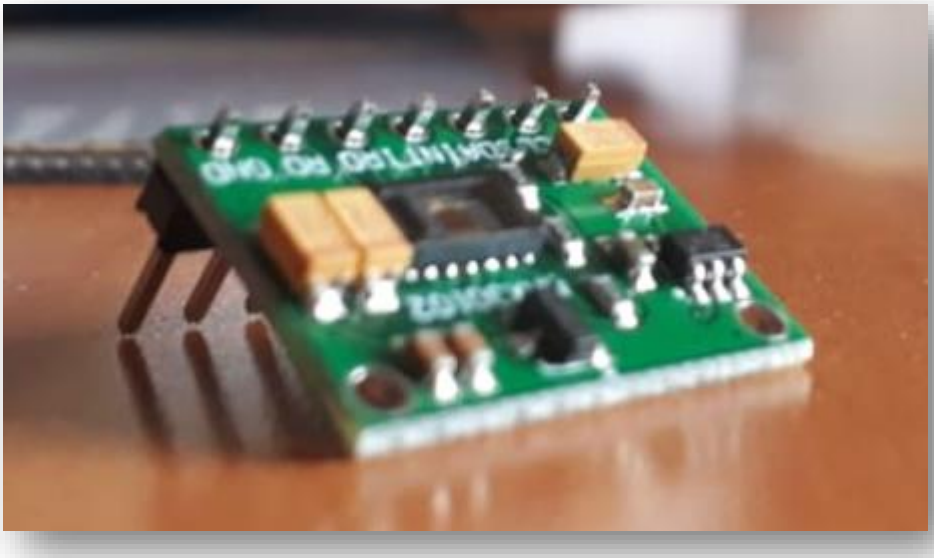


Figure 2: MAX30102 Pulse Oximeter and Heart-Rate Sensor

ABOUT THE MODULE:

The Node MCU is an IoT module which is based on the ESP8266 Wi-Fi Module, has a high processing power with a clock rate of 80-160 MHz and 128 kb RAM (flash memory of 4 Mb for programs). It also has in-built Bluetooth and Wi-Fi Modules to support Web based IoT Applications.

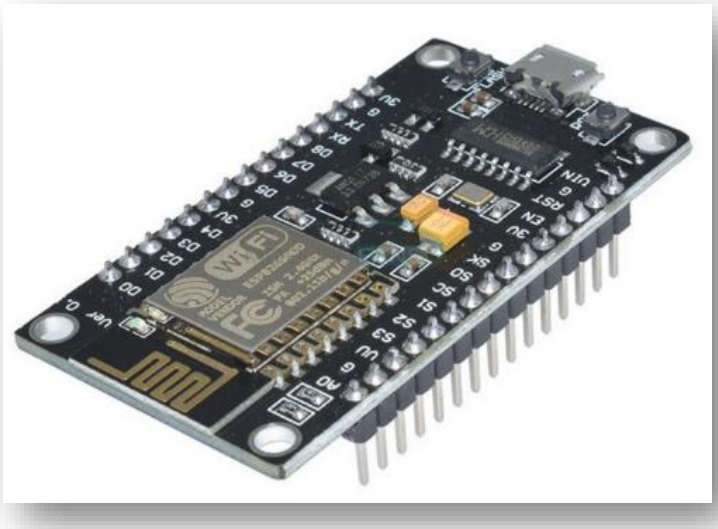


Figure 3: Node MCU ESP8266 Module

SYSTEM SETUP:

HARDWARE (INTERFACING MAX30102 WITH NODE MCU ESP8266):

- The NodeMCU is fitted in the bread board.
- It is then connected with the USB Cable to the Computer system to feed the code into the module.
- The SCL pin of the sensor is connected to the D1 pin slot of Node MCU.
- The SDA pin of the sensor is connected to the D2 pin slot of Node MCU.
- The INT pin of the sensor is connected to the D0 pin slot of Node MCU.
- The GND pin of the sensor is connected to the GND pin slot of Node MCU.
- The VIN pin of the sensor is connected to the 3.3V input slot of Node MCU.
- After setting up the hardware, we can proceed to set up the App.

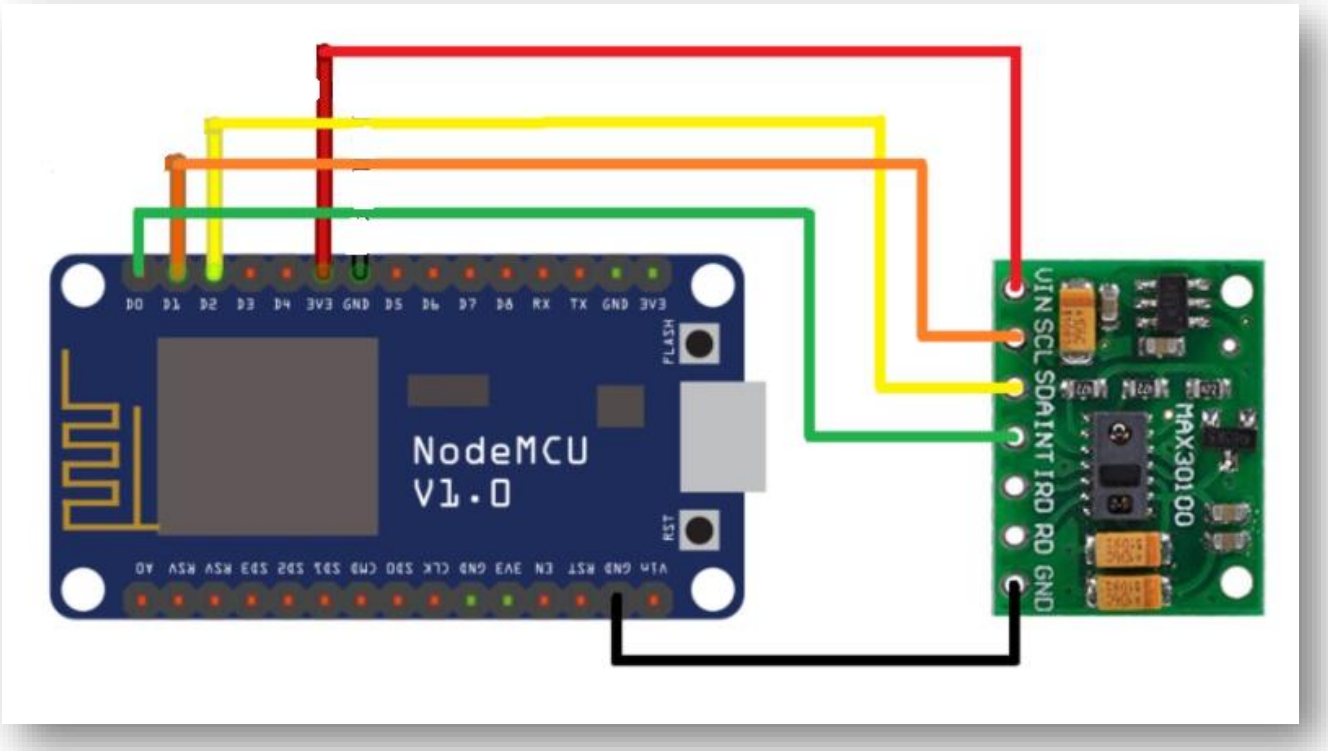


Figure 4: Circuit representation of Setup

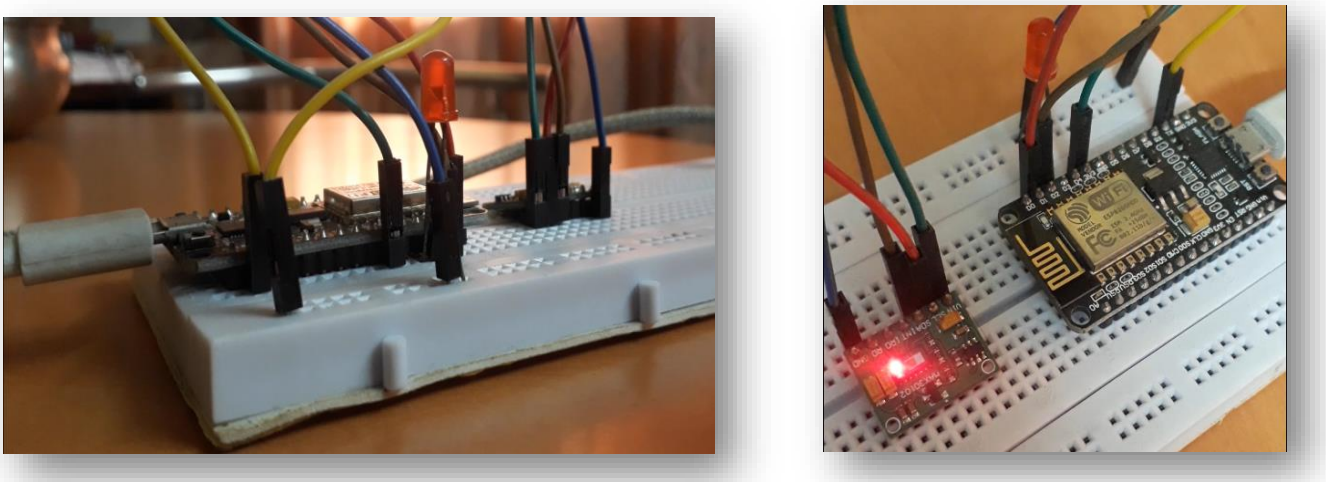


Figure 5: Hardware Setup on Bread Board

Software (setting up the Blynk IoT app [Android]):

- Download the "Blynk – IoT for Arduino, ESP8266/32, Raspberry Pi" from Google Play store and sign in with google account.
- Click on "New Project".
- Name the Project, select device as "NodeMCU" and connection type as "Wi-Fi". Then press Create.
- Click on the "⊕" sign on the top right corner to add the components to the Project.
- Add 1 LCD, 2 Value Display, 2 Gauge and 1 Notification. Individual components can be clicked upon to open component settings.
- For the LCD, select virtual pin V5.
- For the First value display
    - Keep the title "BPM" and range from 0 to 220.
    - Select virtual pin V1 and change reading rate to "push".
- For the Second value display
    - Keep the title "SpO2" and range from 0 to 100.
    - Select virtual pin V2 and change reading rate to "push".
- For the First Gauge
    - Keep the title "BPM" and range from 0 to 220.
    - Select virtual pin V1 and change reading rate to "push".

- For the Second Gauge
  - Keep the title “SpO2” and range from 0 to 100.
  - Select virtual pin V2 and change reading rate to “push”.
- Now click on the “Settings” icon beside the Project Title to open the Project settings.
- Click on “Email All” to receive all the Auth tokens.
- Also, multiple devices can be connected by clicking on “Devices” in the same menu and adding a new Device.

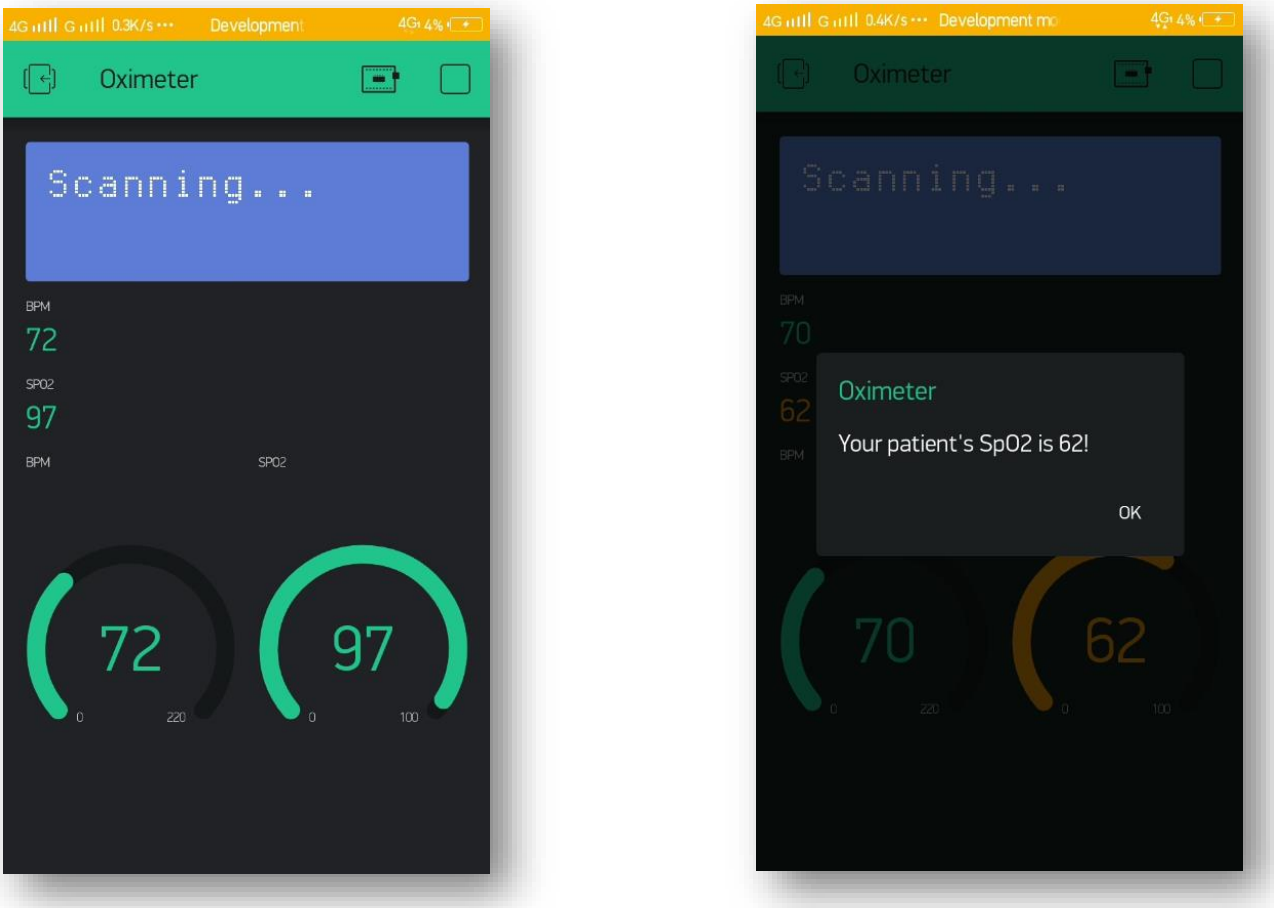
Figure 5: App Screenshot after setup

SYSTEM IMPLEMENTATION & RESULT ANALYSIS:

APP OUTPUT:

After the finger is placed on the sensor, it takes around a minute to calibrate the BPM values to adjust to the patient's resting BPM. During the calibration, no Warning message is displayed. After the BPM value comes within the normal resting range, the notifications can be sent. These notifications are sent:

- If the BPM value is less than 40.
- If the BPM value is more than 100.
- If the SpO_2 value is less than 85.



(a) Healthy Reading *(b) Warning Notification*

Figure 7: App Output Screenshots

Code analysis:

Number of lines: 193

Number of functions: 6

Number of libraries: 7

Conclusion & Future work:

This project components have been successfully tested and it suggests a better way to monitor patients who have contagious diseases. This approach can be very helpful in such situations. Also, this system enables the people to monitor their elderly parents or relatives having certain health conditions remotely from anywhere across the world.

The Blynk IoT App is currently being used to test the running of the app. An app customized for this particular use can be developed. The hardware components of the project can be customized into a wearable device so the one may not have to keep adjusting the finger upon the sensor. Also, a small touchscreen can be set up with the wearable device so as to customize and view the data locally.

Appendix:

Sample Code:

```
void setup(){
  pinMode(LED, OUTPUT); // Pinmode for LED
  Serial.begin(115200); // Initialize Serial
  blinkLED(5); // Blink LED for 5 seconds to signal start of
initialization
  Serial.println("Initializing...");
  if (!pox.begin(Wire, I2C_SPEED_FAST)) { //Use default I2C port, 400kHz
speed
    Serial.println("MAX30102 Sensor was not found. Please check
wiring/power.\nTry pressing the Reset key");
  } else {
    /* Configure the sensor */
```

```
    byte ledBrightness = 60; //Options: 0=Off to 255=50mA
    byte sampleAverage = 4; //Options: 1, 2, 4, 8, 16, 32
    byte ledMode = 2; //Options: 1 = Red only, 2 = Red + IR, 3 = Red + IR
+ Green
    byte sampleRate = 100; //Options: 50, 100, 200, 400, 800, 1000, 1600,
3200
    int pulseWidth = 411; //Options: 69, 118, 215, 411
    int adcRange = 4096; //Options: 2048, 4096, 8192, 16384
    pox.setup(ledBrightness, sampleAverage, ledMode, sampleRate,
pulseWidth, adcRange);
    Serial.println("Setup initialized! Enter any key to continue...");
    blinkLED(3); // Blink LED for 3 seconds to mark initialization
    while (Serial.available() == 0) ; //wait until user presses a key
    Serial.read();
  }
  Blynk.begin(blynkAuth, ssid, pass);
  // Blynk cloud server doesn't handle inline delays well
  // Hence, timer is used
  // Reason: http://help.blynk.cc/en/articles/2091699-keep-your-void-
loop-clean
  sendTimer.setInterval(250L, sendToBlynkServer);
  /* Initial Collection of Data */
  Serial.println("Collecting Data...");
  //read the first 100 samples, and determine the signal range
  for (byte i = 0 ; i < bufferLength ; i++){
    while (pox.available() == false) //do we have new data?
      pox.check(); //Check the sensor for new data
    redBuffer[i] = pox.getRed();
    irBuffer[i] = pox.getIR();
    pox.nextSample(); //We're finished with this sample so move to next
sample
    // Serial.print(F("red="));
    // Serial.print(redBuffer[i], DEC);
    // Serial.print(F(", ir="));
    // Serial.println(irBuffer[i], DEC);
  }
  Serial.println("Data Collected!");
}
void loop(){
  Blynk.run();
  sendTimer.run();
  getHeartRate();
  getSpO2();
}
```

REFERENCES:

- www.how2electronics.com
- www.blynk.io
- www.youtube.com
- www.maximintegrated.com
- www.components101.com
- www.one-tab.com/page/-9wokqlKSGaBABdh6wuIeg
- www.github.com/sparkfun/SparkFun_MAX3010x_Sensor_Library
- www.github.com/blynkkk/blynk-library