1. Purpose

Represent patient real time data (heart rate and blood oxygen level) on a display screen.

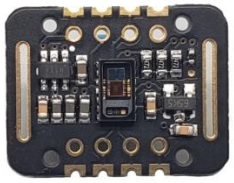
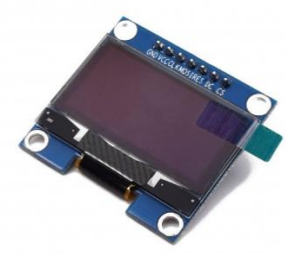
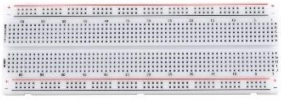
2. Objectives

I. To help patients and health care providers to track oxygen and heart rate level of the patient periodically.

3. Hardware Used and Connections

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

Table 1: List of the components used.

Component		Quantity	Usage
ESP-32S Development Board (ESP-WROOM-32)		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		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.
SH1106 Dot Matrix OLED Display		1	The display uses the OLED technology to display shapes using I2C Protocol. The OLED (Organic Light-Emitting Diode) is a self-light-emitting technology composed of a thin, multi-layered organic film placed between an anode and cathode. OLED does not require a backlight, which means that it saves a lot of power.
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.

The hardware connections are done as following:

1. Since both MAX30102 and SH1106 are using I2C Protocol, they require SDA and SCL pins to relate to the same pin numbers.
2. SDA is connected to GPIO4
3. SCL is connected to GPIO15
4. Both MAX30102 and SH1106 are connected to GND and 3V.

4. Actual Implementation

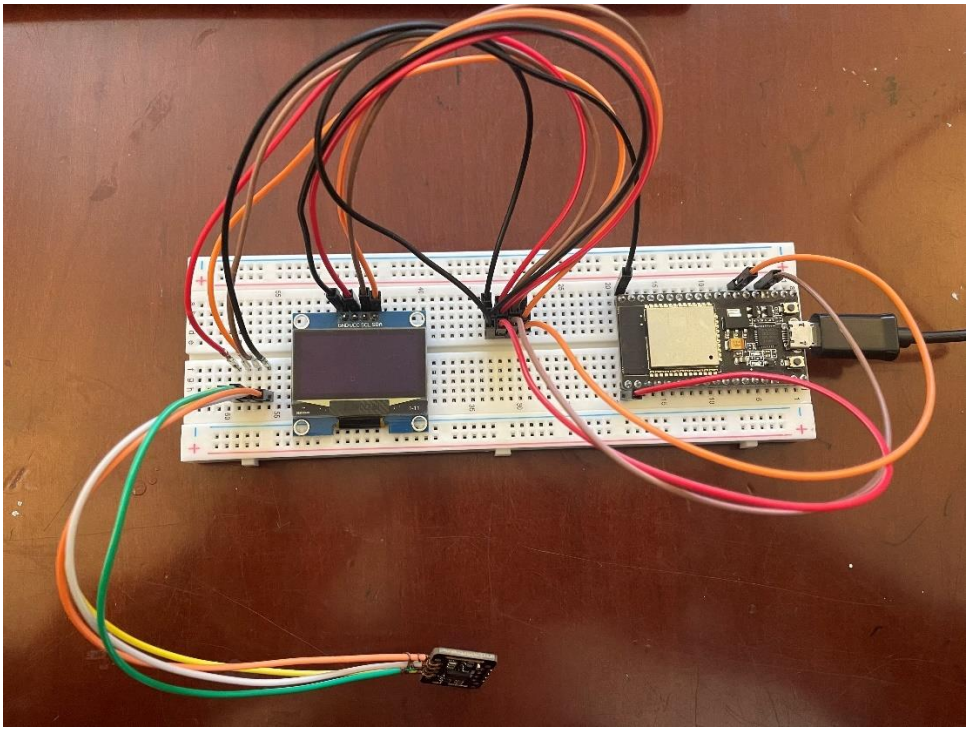


Figure 1: Actual implementation.

5. Complete Code

Arduino IDE Code:

```
// Marim Elhanafy
// 201803468
// Extra Part for Interfacing Project
////////////////////////////////////////////////////////////////////////////////
////
//////////////////////    MAX30102  //////////////////////
#include "MAX30105.h" //MAX3010x library
#include "heartRate.h" //Heart rate calculating algorithm
#include "ESP32Servo.h"
#include <Wire.h>
MAX30105 particleSensor;
```

```
int Tonepin = 4;
const byte RATE_SIZE = 10;
byte rates[RATE_SIZE];
byte rateSpot = 0;
long lastBeat = 0; //Time at which the last beat occurred
float beatsPerMinute;
int beatAvg;
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 i = 0;
int Num = 30;
#define FINGER_ON 7000
#define MINIMUM_SPO2 90.0
//////////////////    SH1106    //////////////////
#include <qrbits.h>
#include <qrcode.h>
#include <qrencode.h>
#include <SH1106.h> // Include the SH1106 library
#define OLED_ADDR 0x3C // OLED I2C address
#define OLED_SDA 4 // OLED SDA pin
#define OLED_SCL 15 // OLED SCL pin
SH1106 display(OLED_ADDR, OLED_SDA, OLED_SCL); // Create an SH1106 display
object
////////////////////////////////////////////////////////////////////////////
////
void setup() {
  Serial.begin(115200); //Start the Serial Monitor
  Serial.println("System Start");
  //////////////////    SH1106    //////////////////
  display.init(); // Initialize the display
  display.flipScreenVertically(); // Flip the display vertically
  display.setFont(ArialMT_Plain_10); // Set the font
  display.setTextAlignment(TEXT_ALIGN_LEFT); // Set the text alignment
  //////////////////    MAX30102  //////////////////
  //Use default I2C port, 400kHz speed
  if (!particleSensor.begin(Wire, I2C_SPEED_FAST))
  {
    Serial.println("MAX30102 did not work!");
    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);
}
////////////////////////////////////////////////////////////////////////////////
////
void loop() {
  //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 ) {
    display.clear(); // Clear the display
    display.drawString(10, 10, String(beatAvg));
    display.drawString(10, 20, "BPM");
    if (beatAvg > 30)
    {
      display.drawString(10, 30, String(ESpO2));
      display.drawString(10, 40, "%");
    }
    else
    {
      display.drawString(10, 30, "---- %");
    }
    display.display(); // Update the display
    if (checkForBeat(irValue) == true) {
      display.clear(); // Clear the display
      display.drawString(10, 10, String(beatAvg));
      display.drawString(10, 20, "BPM");
      if (beatAvg > 30)
      {
        display.drawString(10, 30, String(ESpO2));
        display.drawString(10, 40, "%");
      }
      else
      {
        display.drawString(10, 30, "---- %");
      }
```

```
      display.display(); // Update the display
      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;
        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;
```

```
    display.clear(); // Clear the display
    display.drawString(10, 10, "No Finger!"); //Finger Please
    display.display(); // Update the display
  }
}
```

6. Testing

1. Without Finger:

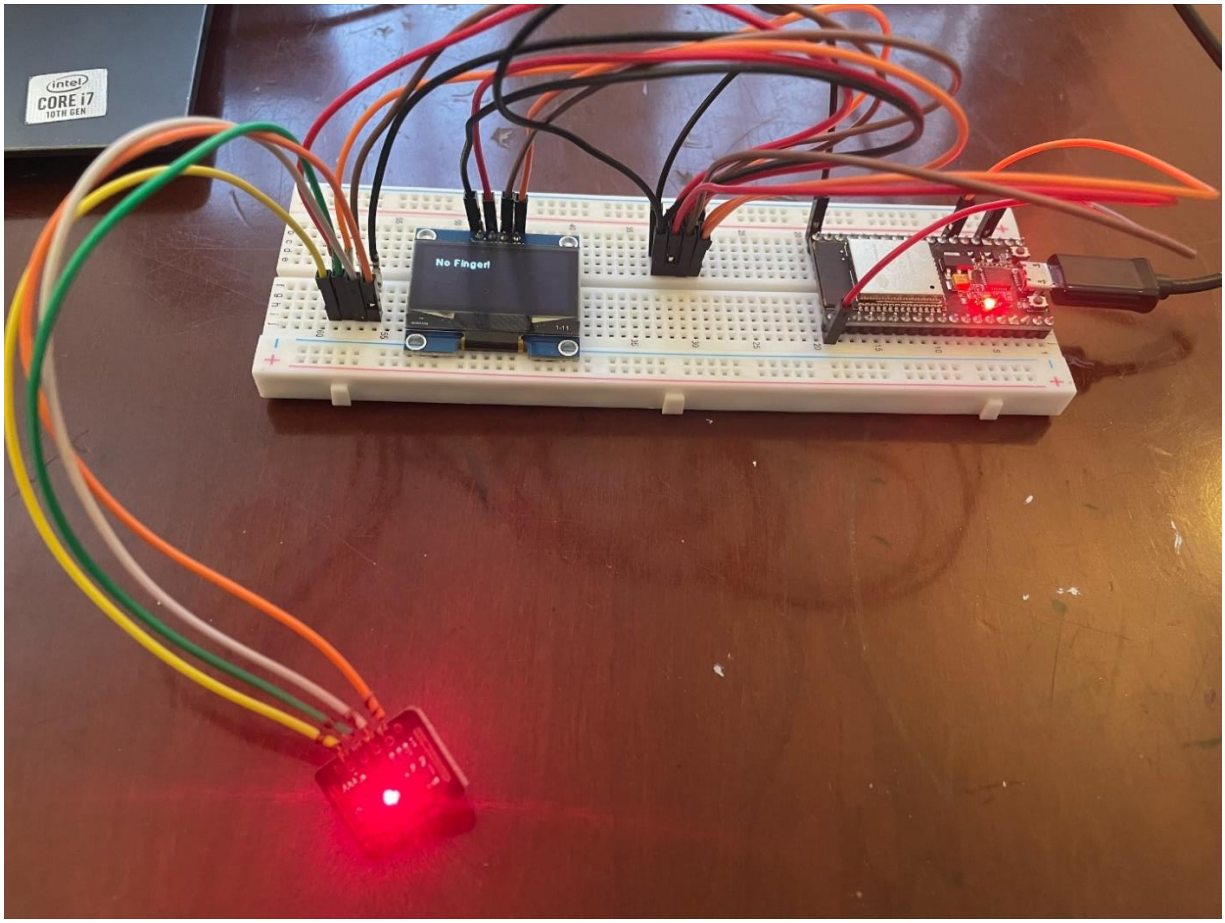


Figure 2: Testing the display without finger on the sensor.

2. With Finger:

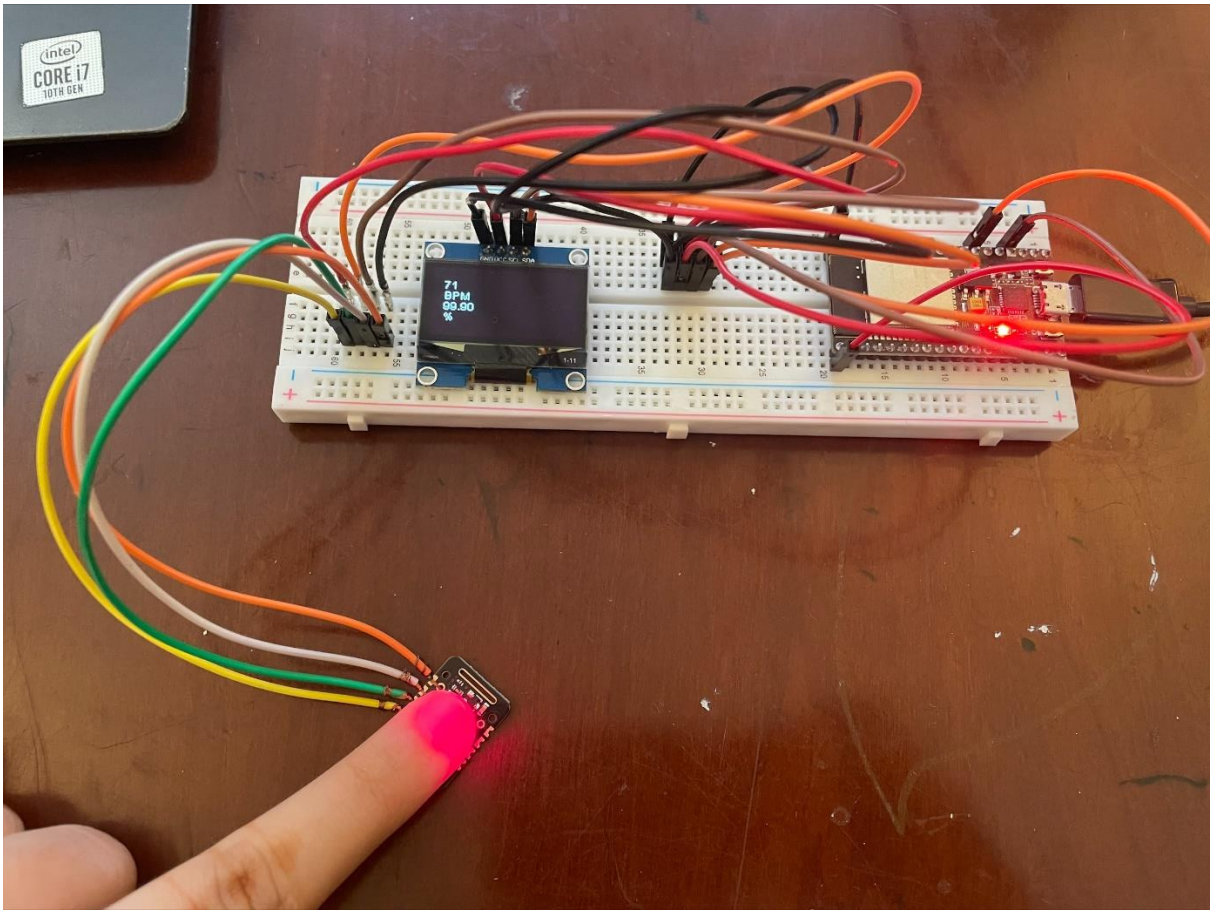


Figure 3: Testing the display with finger on the sensor.