# 1. Purpose

Represent patient real time data (heart rate and blood oxygen level) on a Graphical User Interface (GUI).

# 2. Objectives

I. To help patients and nurses to track the patient measurements and the surrounding measurements periodically.

# 3. 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(ESpO2);
        //Serial.println(" %");
      }
      else  //Serial.println("---- %");
      if (checkForBeat(irValue) == true)
      {
        Serial.print(beatAvg);
        //Serial.println(" BPM");
        if (beatAvg > 30)
        {
          Serial.print(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() { }
```

**Processing IDE Code:**

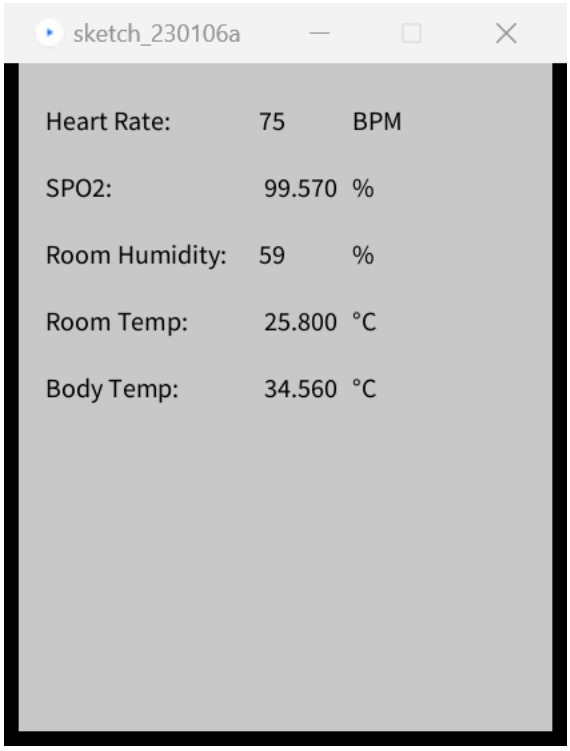
```
// Import the necessary libraries

import processing.serial.*;



// Declare global variables

Serial port;  // serial port object

int heartRate;  // heart rate value

float spo2;  // SPO2 value
```

```
int humidity;  // room humidity value
float roomTemp;  // room temperature value
float bodyTemp;  // body temperature value
float[] values;

void setup() {
  // Set the size of the window
  size(400, 500);

  // Set the background color
  background(200, 200, 200);

  // Set the font and text size
  textSize(20);
  //textFont("Arial");

  // Connect to the serial port
  port = new Serial(this, "COM3", 115200);

  // Set the initial values to zero
  heartRate = 0;
  spo2 = 0.0;
  humidity = 0;
  roomTemp = 0.0;
  bodyTemp = 0.0;

}
```

```
void draw() {
  // Clear the screen
  background(200, 200, 200);

  // Draw the heart rate value
  fill(0, 0, 0);  // set the text color to black
  text("Heart Rate: ", 20, 50);  // display the label
  text(heartRate, 180, 50);  // display the value
  text("BPM", 250, 50);  // display the unit

  // Draw the SPO2 value
  fill(0, 0, 0);  // set the text color to black
  text("SPO2: ", 20, 100);  // display the label
  text(spo2, 180, 100);  // display the value
  text("%", 250, 100);  // display the unit

  // Draw the room humidity value
  fill(0, 0, 0);  // set the text color to black
  text("Room Humidity: ", 20, 150);  // display the label
  text(humidity, 180, 150);  // display the value
  text("%", 250, 150);  // display the unit

  // Draw the room temperature value
  fill(0, 0, 0);  // set the text color to black
  text("Room Temp: ", 20, 200);  // display the label
  text(roomTemp, 180, 200);  // display the value
```

```
 text("°C", 250, 200);  // display the unit

 // Draw the body temperature value
 fill(0, 0, 0);  // set the text color to black
 text("Body Temp: ", 20, 250);  // display the label
 text(bodyTemp, 180, 250);  // display the value
 text("°C", 250, 250);  // display the unit

 values = new float[5];
 for (int i = 0; i < 5; i++) {
  values[i] = port.read();  // read a line from the serial port
  //values[i] = values[i].trim();  // remove leading and trailing whitespace
 }

 // Assign each value to a variable
 heartRate = int(values[0]);
 spo2 = values[1];
 humidity = int(values[2]);
 roomTemp = values[3];
 bodyTemp = values[4];

}
```

## 4. Testing


sketch_230106a
Heart Rate: 75 BPM
SPO2: 99.570 %
Room Humidity: 59 %
Room Temp: 25.800 °C
Body Temp: 34.560 °C