```
#include <LiquidCrystal_I2C.h>
#include "MAX30105.h"
#include "spo2_algorithm.h"
#include "heartRate.h"
#include <OneWire.h>
#include <DallasTemperature.h>
///#include <WiFi.h>
LiquidCrystal_I2C lcd(0x27, 16, 2);
//----------CH_ID 2762659
//-----------WRITE_APIKEY =
WA1095K5AARFNTR5
//---------READ API = 9T547CK5OZTDMRS0
//String ssid =  "IOTHOME";
//String pass =  "ABCD098765432";

#define SerialAT Serial2

int keyIndex = 0;            // your network key
Index number (needed only for WEP)
const int s1 = 39;
const int s2 = 34;
```

```
const int s3 = 35;
const int s4 = 32;
const int m1 = 4;
const int m2 = 5;
const int m3 = 18;
const int m4 = 19;
const int trigPin = 15;
const int echoPin = 2;
const int buz = 13;
// Data wire is plugged into port 5 on the
Arduino
#define ONE_WIRE_BUS 23

// Setup a oneWire instance to
communicate with any OneWire devices
(not just Maxim/Dallas temperature ICs)
OneWire oneWire(ONE_WIRE_BUS);

// Pass our oneWire reference to Dallas
Temperature.
DallasTemperature sensors(&oneWire);
```

```
MAX30105 particleSensor;
#define MAX_BRIGHTNESS 255

uint32_t irBuffer[100]; //infrared LED
sensor data
uint32_t redBuffer[100];  //red LED sensor
data
//WiFiClient  client;

//String ChannelNumber = SECRET_CH_ID;
//String WriteAPIKey =
SECRET_WRITE_APIKEY;

int s1s=0;
int s2s=0;
int s3s=0;
int s4s=0;
int ecg;
int x=0;
int cntwifi;
```

```
int cnt=0;
int m =0;
float ecgf=0.0;
int spo=0;
long irValue;
const byte RATE_SIZE = 0; //Increase this for more averaging. 4 is good.
byte rates[RATE_SIZE]; //Array of heart rates
byte rateSpot = 0;
int deviceCount = 0;
float temp2=0.0;
byte lastBeat = 0; //Time at which the last beat occurred
int bpm;
float beatsPerMinute;
int beatAvg;
long duration;
float distanceCmF;
float SOUND_SPEED = 0.034;
```

```
void setup() {
  pinMode(buz, OUTPUT);
  pinMode(s1,INPUT);
  pinMode(s2,INPUT);
  pinMode(s3,INPUT);
  pinMode(s4,INPUT);
  pinMode(m1, OUTPUT);
  pinMode(m2, OUTPUT);
  pinMode(m3, OUTPUT);
  pinMode(m4, OUTPUT);
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
  digitalWrite(m1, LOW);
  digitalWrite(m2, LOW);
  digitalWrite(m3, LOW);
  digitalWrite(m4, LOW);
  Serial.begin(115200);  // Initialize serial
  SerialAT.begin(9600);

  lcd.init();
```

```
  lcd.backlight();
  lcd.setCursor(0,0);
  lcd.print("HEALTH  MONITOR ");
  lcd.setCursor(0,1);
  lcd.print("  SYSTEM    ");
  delay(1000);
  lcd.clear();

  //WiFi.mode(WIFI_AP);
  //WiFi.softAP(ssid, pass);
  //lcd.setCursor(0,0);
  //lcd.print("Connecting WiFi...");
  //while (WiFi.status() !=
WL_CONNECTED) {
  //  delay(500);
  //  return
  //}
  //Serial.println("");
  //Serial.println("WiFi connected");
  //lcd.clear();
```

```
  if (!particleSensor.begin(Wire,
I2C_SPEED_FAST))
  {
    Serial.println("SPO2 not found");
    while (1);
  }
  Serial.println("Place your index finger");
  particleSensor.setup();

particleSensor.setPulseAmplitudeRed(0x0
C);

particleSensor.setPulseAmplitudeGreen(1);

  deviceCount = sensors.getDeviceCount();
  digitalWrite(buz,
HIGH);delay(100);digitalWrite(buz,
LOW);delay(100);
}
```

```
void loop() {
  if(Serial.available()>0){
    int rec = Serial.read();
    Serial.println(rec);
    if(rec==65){
      digitalWrite(trigPin,
LOW);delayMicroseconds(2);digitalWrite(tri
gPin,
HIGH);delayMicroseconds(10);digitalWrite(
trigPin, LOW);
      duration = pulseIn(echoPin, HIGH);
      distanceCmF = duration *
SOUND_SPEED/2;
      Serial.print("Dist=");
      Serial.println(distanceCmF);
      delay(500);
    }
    if(rec==66){
      irValue=0;
      lastBeat=0;
      rateSpot=0;
```

```
    getbpm();
    getparam();
    Serial.print("BPM=");
    Serial.print(bpm);
    Serial.print(", spo=");
    Serial.print(spo);
    Serial.print(", Temp=");
    Serial.print(temp2);
    delay(3000);
  }
}
Serial.print("s1=");
Serial.print(s1s);
Serial.print(", s2=");
Serial.print(s2s);
Serial.print(", s3=");
Serial.print(s3s);
Serial.print(", s4=");
Serial.println(s4s);

lcd.setCursor(0,0);
```

```
  lcd.print("BPM:");lcd.print(bpm);lcd.print("
");lcd.print(temp2);lcd.print(char(223));lcd.p
rint("C ");


  s1s = digitalRead(s1);
  s2s = digitalRead(s2);
  s3s = digitalRead(s3);
  s4s = digitalRead(s4);
  lcd.setCursor(0,1);
  lcd.print(s1s);lcd.print("
");lcd.print(s2s);lcd.print("
");lcd.print(s3s);lcd.print(" ");;
  lcd.print("Sp:");lcd.print(spo);lcd.print("%
");

  if(s1s==HIGH){}
  if(s2s==HIGH){}
  if(s3s==HIGH){}
  if(s4s==HIGH){
    irValue=0;
```

```
    lastBeat=0;
    rateSpot=0;
    getbpm();
    getparam();
    Serial.print("BPM=");
    Serial.print(bpm);
    Serial.print(", spo=");
    Serial.print(spo);
    Serial.print(", Temp=");
    Serial.print(temp2);
    delay(3000);
    }

  cnt++;
  if(cnt>100){
  cnt=0;sensors.requestTemperatures();
  temp2 = sensors.getTempCByIndex(0);/
*updateserver();*/
  irValue=0;
    lastBeat=0;
    rateSpot=0;
```

```
    getbpm();
    getparam();
    }
}

void getparam(){
    cnt=0;sensors.requestTemperatures();
    temp2 = sensors.getTempCByIndex(0);
}

void getbpm(){
  for(x=0;x<500;x++){
  irValue = particleSensor.getIR();
  if (checkForBeat(irValue) == true)
  {
    int delta = millis() - lastBeat;
    lastBeat = millis();
    beatsPerMinute = 60 / (delta / 1000.0);
    if (beatsPerMinute < 255 &&
beatsPerMinute > 20)
    {
```

```
      rates[rateSpot++] =
(byte)beatsPerMinute; //Store this reading
in the array
      rateSpot %= RATE_SIZE; //Wrap variable
      beatAvg = 0;
      for (byte y = 0 ; y < RATE_SIZE ; y++)
        beatAvg += rates[y];
        beatAvg /= RATE_SIZE;
    }
  }
   bpm = beatsPerMinute;
   spo=irValue/1000;//Take spo  readings
}
}

/*void updateserver(){
  lcd.setCursor(0,0);
  lcd.print("UPDATING CLOUD.");
  ThingSpeak.Field(1, bpm);
  ThingSpeak.Field(2, spo);
  ThingSpeak.Field(3, ecg);
```

```
  ThingSpeak.Field(4, s1s);
  ThingSpeak.Field(5, s2s);
  ThingSpeak.Field(6, s3s);
  ThingSpeak.Field(7, s4s);

  delay(5000);
  lcd.clear();
}*/
```