#include <Wire.h>

#include <SPI.h>

#include <Adafruit\_GFX.h>

#include <Adafruit\_SSD1306.h>

#define SCREEN\_WIDTH 128 // OLED display width, in pixels

#define SCREEN\_HEIGHT 32 // OLED display height, in pixels

#define SCREEN\_ADDRESS 0x3c ///< See datasheet for Address; 0x3D for 128x64, 0x3C for 128x32

#define OLED\_RESET 4

Adafruit\_SSD1306 display(SCREEN\_WIDTH, SCREEN\_HEIGHT, &Wire, OLED\_RESET);

#define maxperiod\_siz 80 // max number of samples in a period 80

#define measures 8 // number of periods stored 10

#define samp\_siz 4 // number of samples for average 4

#define rise\_threshold 3 // number of rising measures to determine a peak 3

// a liquid crystal displays BPM

//LiquidCrystal\_I2C lcd(0x3F, 16, 2);

int T = 20; // slot milliseconds to read a value from the sensor

int sensorPin = A1;

//int REDLed = 10;

//int IRLed = 11;

int REDLed = 6;

int IRLed = 7;

int SpO2;

int avBPM;

//indexfinger

const unsigned char myBitmap [] PROGMEM = {

0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,

0xff, 0xff, 0xff, 0xff, 0xff, 0x8f, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,

0xff, 0xe0, 0x0f, 0xff, 0xff, 0xc0, 0x7f, 0xe0, 0x1f, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,

0xff, 0xff, 0x8f, 0xfc, 0x7c, 0x00, 0xff, 0xff, 0x8f, 0xff, 0xff, 0xc7, 0xff, 0xff, 0xff, 0xff,

0xff, 0xfe, 0x3f, 0xff, 0xfc, 0x01, 0xff, 0xfc, 0x3f, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,

0xff, 0xfc, 0x78, 0x00, 0x31, 0xf8, 0x80, 0x1c, 0x71, 0xfc, 0x60, 0xe2, 0x00, 0x71, 0xc1, 0xff,

0xff, 0xfc, 0x7f, 0xf8, 0xf1, 0xf8, 0xff, 0x8c, 0x71, 0xfc, 0x70, 0x87, 0xfe, 0x18, 0xf8, 0xff,

0xff, 0xfc, 0x78, 0x00, 0xf1, 0xf8, 0xff, 0x84, 0x71, 0xfc, 0x63, 0xe2, 0x0f, 0x0c, 0x78, 0xff,

0xff, 0xfc, 0x71, 0xf8, 0xf1, 0xf0, 0xff, 0x8c, 0x71, 0xf8, 0x63, 0xe2, 0x3f, 0x1c, 0x38, 0xff,

0xff, 0xfc, 0x70, 0x78, 0xf0, 0x00, 0xf0, 0x1c, 0x78, 0x00, 0x60, 0x02, 0x00, 0x3e, 0x01, 0xff,

0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,

0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xf8, 0x00, 0x0f, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,

0xff, 0xff, 0xff, 0xff, 0xf8, 0x00, 0x00, 0x00, 0x00, 0x7f, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,

0xff, 0xff, 0xff, 0xff, 0x00, 0x0f, 0xff, 0xff, 0xfc, 0x0f, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,

0xff, 0xff, 0xff, 0xf0, 0x1f, 0xff, 0xff, 0xff, 0xff, 0xc1, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,

0xfc, 0x00, 0x3f, 0x03, 0xff, 0xff, 0xff, 0xff, 0xff, 0xf8, 0x3f, 0xff, 0xff, 0xff, 0xff, 0xff,

0xc0, 0x00, 0x00, 0x1f, 0xff, 0xff, 0xff, 0xff, 0xff, 0xfe, 0x00, 0x00, 0x00, 0x00, 0x0f, 0xff,

0xc3, 0xff, 0xc0, 0xff, 0xff, 0xff, 0xff, 0xf0, 0xff, 0xff, 0x0f, 0xf8, 0x00, 0x00, 0x00, 0x0f,

0x87, 0xff, 0xc3, 0xff, 0xff, 0xff, 0xff, 0xe0, 0x03, 0xff, 0x0f, 0xff, 0xff, 0xff, 0xff, 0x83,

0x87, 0xff, 0xc3, 0xff, 0xff, 0xff, 0xff, 0xc3, 0xc3, 0xff, 0x0f, 0xff, 0xff, 0xff, 0xff, 0xe1,

0x87, 0xff, 0xc3, 0xff, 0xff, 0xff, 0xfe, 0x00, 0x01, 0xff, 0x0f, 0xff, 0xff, 0xff, 0xfe, 0x03,

0x87, 0xff, 0xc3, 0xff, 0xff, 0xff, 0xf0, 0x3f, 0x80, 0xff, 0x80, 0x00, 0x00, 0x00, 0x00, 0x1f,

0x87, 0xff, 0xc3, 0xf8, 0x00, 0x00, 0x00, 0x7f, 0xf8, 0x3f, 0x87, 0xf0, 0xff, 0xff, 0xff, 0xff,

0x87, 0xff, 0xc3, 0xf8, 0x00, 0x01, 0xfc, 0x1f, 0xfe, 0x0f, 0x87, 0xf8, 0x7f, 0xff, 0xff, 0xff,

0x87, 0xff, 0xc3, 0xff, 0xff, 0xff, 0xf0, 0x00, 0xff, 0x80, 0x0f, 0xf0, 0xff, 0xff, 0xff, 0xff,

0x87, 0xff, 0xc3, 0xff, 0xff, 0xff, 0xe1, 0xf0, 0x03, 0xff, 0xfc, 0x01, 0xff, 0xff, 0xff, 0xff,

0x87, 0xff, 0xc3, 0xff, 0xff, 0xff, 0xe1, 0xff, 0xc0, 0x00, 0x00, 0x1f, 0xff, 0xff, 0xff, 0xff,

0x87, 0xff, 0xc3, 0xff, 0xff, 0xff, 0xf0, 0x3f, 0xff, 0xff, 0xfc, 0x3f, 0xff, 0xff, 0xff, 0xff,

0x87, 0xff, 0xc3, 0xff, 0xff, 0xff, 0x80, 0x0f, 0xff, 0xff, 0xfc, 0x3f, 0xff, 0xff, 0xff, 0xff,

0xc7, 0xff, 0xc1, 0xff, 0xff, 0xff, 0x87, 0xc0, 0xff, 0xff, 0xc0, 0x7f, 0xff, 0xff, 0xff, 0xff,

0xc3, 0xff, 0xc0, 0x1f, 0xff, 0xff, 0x83, 0xf8, 0x00, 0x00, 0x03, 0xff, 0xff, 0xff, 0xff, 0xff,

0xe0, 0x38, 0x07, 0x00, 0x3f, 0xff, 0xe0, 0xff, 0xff, 0xf8, 0x7f, 0xff, 0xff, 0xff, 0xff, 0xff,

0xf8, 0x00, 0x3f, 0xfc, 0x00, 0x1f, 0xf8, 0x1f, 0xff, 0xf0, 0x7f, 0xff, 0xff, 0xff, 0xff, 0xff,

0xff, 0xff, 0xff, 0xff, 0xfe, 0x00, 0x07, 0x00, 0x00, 0x01, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,

0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0x00, 0x00, 0x00, 0x3f, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff

};

byte sym[3][8] = {{B00000,B01010,B11111,B11111,B01110,B00100,B00000,B00000},

{B00000,B00000,B00000,B11000,B00100,B01000,B10000,B11100},

{B00000,B00100,B01010,B00010,B00100,B00100,B00000,B00100}

};

void setup() {

Serial.begin(115200);

Serial.flush();

pinMode(sensorPin,INPUT);

pinMode(REDLed,OUTPUT);

pinMode(IRLed,OUTPUT);

// turn off leds

digitalWrite(REDLed,LOW);

digitalWrite(IRLed,LOW);

// for(int i=0;i<8;i++) lcd.createChar(i, sym[i]);

if(!display.begin(SSD1306\_SWITCHCAPVCC, 0x3C)) { // Address 0x3C for 128x32

Serial.println(F("SSD1306 allocation failed"));

for(;;); // Don't proceed, loop forever

}}

void loop ()

{

/\*display.clearDisplay();

display.setTextSize(1); // Draw 2X-scale text

display.setTextColor(SSD1306\_WHITE);

display.setCursor(20, 10);

display.println("Insert Fingure");

display.display(); \*/

bool finger\_status = true;

float readsIR[samp\_siz], sumIR,lastIR, reader, start;

float readsRED[samp\_siz], sumRED,lastRED;

int period, samples;

period=0; samples=0;

int samplesCounter = 0;

float readsIRMM[maxperiod\_siz],readsREDMM[maxperiod\_siz];

int ptrMM =0;

for (int i = 0; i < maxperiod\_siz; i++) { readsIRMM[i] = 0;readsREDMM[i]=0;}

float IRmax=0;

float IRmin=0;

float REDmax=0;

float REDmin=0;

double R=0;

float measuresR[measures];

int measuresPeriods[measures];

int m = 0;

for (int i = 0; i < measures; i++) { measuresPeriods[i]=0; measuresR[i]=0; }

int ptr;

float beforeIR;

bool rising;

int rise\_count;

int n;

long int last\_beat;

for (int i = 0; i < samp\_siz; i++) { readsIR[i] = 0; readsRED[i]=0; }

sumIR = 0; sumRED=0;

ptr = 0;

while(1)

{

//

// turn on IR LED

digitalWrite(REDLed,LOW);

digitalWrite(IRLed,HIGH);

// calculate an average of the sensor

// during a 20 ms (T) period (this will eliminate

// the 50 Hz noise caused by electric light

n = 0;

start = millis();

reader = 0.;

do

{

reader += analogRead (sensorPin);

n++;

}

while (millis() < start + T);

reader /= n; // we got an average

// Add the newest measurement to an array

// and subtract the oldest measurement from the array

// to maintain a sum of last measurements

sumIR -= readsIR[ptr];

sumIR += reader;

readsIR[ptr] = reader;

lastIR = sumIR / samp\_siz;

//

// TURN ON RED LED and do the same

digitalWrite(REDLed,HIGH);

digitalWrite(IRLed,LOW);

n = 0;

start = millis();

reader = 0.;

do

{

reader += analogRead (sensorPin);

n++;

}

while (millis() < start + T);

reader /= n; // we got an average

// Add the newest measurement to an array

// and subtract the oldest measurement from the array

// to maintain a sum of last measurements

sumRED -= readsRED[ptr];

sumRED += reader;

readsRED[ptr] = reader;

lastRED = sumRED / samp\_siz;

//

// R CALCULATION

// save all the samples of a period both for IR and for RED

readsIRMM[ptrMM]=lastIR;

readsREDMM[ptrMM]=lastRED;

ptrMM++;

ptrMM %= maxperiod\_siz;

samplesCounter++;

//

// if I've saved all the samples of a period, look to find

// max and min values and calculate R parameter

if(samplesCounter>=samples){

samplesCounter =0;

IRmax = 0; IRmin=1023; REDmax = 0; REDmin=1023;

for(int i=0;i<maxperiod\_siz;i++) {

if( readsIRMM[i]> IRmax) IRmax = readsIRMM[i];

if( readsIRMM[i]>0 && readsIRMM[i]< IRmin ) IRmin = readsIRMM[i];

readsIRMM[i] =0;

if( readsREDMM[i]> REDmax) REDmax = readsREDMM[i];

if( readsREDMM[i]>0 && readsREDMM[i]< REDmin ) REDmin = readsREDMM[i];

readsREDMM[i] =0;

}

R = ( (REDmax-REDmin) / REDmin) / ( (IRmax-IRmin) / IRmin ) ;

}

// check that the finger is placed inside

// the sensor. If the finger is missing

// RED curve is under the IR.

//

if (lastRED < lastIR) {

if(finger\_status==true) {

finger\_status = false;

// lcd.clear();

// lcd.setCursor(0,0);

// lcd.print("No finger?");

//Serial.println("No finger?");

}

} else {

if(finger\_status==false) {

// lcd.clear();

finger\_status = true;

//lcd.setCursor(10,0);

//lcd.print("c=");

//Serial.println("c");

//lcd.setCursor(0,0);

//lcd.print("bpm");

// lcd.setCursor(0,1);

// lcd.print("SpO"); lcd.write(1); //2

// lcd.setCursor(10,1);

// lcd.print("R=");

}

}

float avR = 0;

avBPM=0;

if (finger\_status==true){

// lastIR holds the average of the values in the array

// check for a rising curve (= a heart beat)

if (lastIR > beforeIR)

{

rise\_count++; // count the number of samples that are rising

if (!rising && rise\_count > rise\_threshold)

{

// lcd.setCursor(3,0);

// lcd.write( 0 ); // <3

// Ok, we have detected a rising curve, which implies a heartbeat.

// Record the time since last beat, keep track of the 10 previous

// peaks to get an average value.

// The rising flag prevents us from detecting the same rise

// more than once.

rising = true;

measuresR[m] = R;

measuresPeriods[m] = millis() - last\_beat;

last\_beat = millis();

int period = 0;

for(int i =0; i<measures; i++) period += measuresPeriods[i];

// calculate average period and number of samples

// to store to find min and max values

period = period / measures;

samples = period / (2\*T);

int avPeriod = 0;

int c = 0;

// c stores the number of good measures (not floating more than 10%),

// in the last 10 peaks

for(int i =1; i<measures; i++) {

if ( (measuresPeriods[i] < measuresPeriods[i-1] \* 1.1) &&

(measuresPeriods[i] > measuresPeriods[i-1] / 1.1) ) {

c++;

avPeriod += measuresPeriods[i];

avR += measuresR[i];

}

}

m++;

m %= measures;

// lcd.setCursor(12,0);

// lcd.print(String(c)+" ");

//Serial.println(String(c)+" ");

// bpm and R shown are calculated as the

// average of at least 5 good peaks

avBPM = 60000 / ( avPeriod / c) ;

avR = avR / c ;

// if there are at last 5 measures

//lcd.setCursor(12,1);

if(c==0) /\*lcd.print(" ");\*/ Serial.println(" ");

else /\*lcd.print(String(avR) + " ");\*/ Serial.println(" ");

// if there are at least 5 good measures...

if(c > 4) {

//

// SATURTION IS A FUNCTION OF R (calibration)

// Y = k\*x + m

// k and m are calculated with another oximeter

// SpO2 = -19 \* R + 99;

SpO2 = -19 \* R + 99 + 26;

//lcd.setCursor(4,0);

if(avBPM > 40 && avBPM <220) Serial.println(String(avBPM)+" ");

dis();

//lcd.print(String(avBPM)+" "); //else lcd.print("---");

//lcd.setCursor(4,1);

if(SpO2 > 70 && SpO2 <110) Serial.println( " " + String(SpO2) +"% "); //lcd.print( " " + String(SpO2) +"% "); //else lcd.print("--%

");

dis();

} else {

if(c <3) {

display.clearDisplay();

display.setTextSize(1); // Draw 2X-scale text

display.setTextColor(SSD1306\_WHITE);

display.setCursor(20, 10);

display.println("Insert Finger ");

//display.setTextSize(1); // Draw 2X-scale text

display.drawBitmap(0, 0, myBitmap, 128, 64, WHITE);

display.display();

// if less then 2 measures add ?

//lcd.setCursor(3,0); lcd.write( 2 ); //bpm ?

//lcd.setCursor(4,1); lcd.write( 2 ); //SpO2 ?

}

}

}

}

else

{

// Ok, the curve is falling

rising = false;

rise\_count = 0;

//lcd.setCursor(3,0);lcd.print(" ");

}

// to compare it with the new value and find peaks

beforeIR = lastIR;

} // finger is inside

// PLOT everything

//Serial.print(lastIR);

Serial.print(",");

// Serial.print(lastRED);

/\*

\* Serial.print(",");

Serial.print(R);

Serial.print(",");

Serial.print(IRmax);

Serial.print(",");

Serial.print(IRmin);

Serial.print(",");

Serial.print(REDmax);

Serial.print(",");

Serial.print(REDmin);

Serial.print(",");

Serial.print(avR);

Serial.print(",");

Serial.print(avBPM); \*/

Serial.println();

// handle the arrays

ptr++;

ptr %= samp\_siz;

} // loop while 1

}

void dis()

{

int ccc = 0;

display.clearDisplay();

display.setTextSize(1); // Draw 2X-scale text

display.setTextColor(SSD1306\_WHITE);

display.setCursor(10, 0);

display.println("SpO2%");

display.setCursor(90, 0);

display.println("BpM");

display.setTextSize(2);

display.setCursor(10, 11);

display.print(SpO2);

display.println("%");

display.setCursor(80, 11);

display.println(avBPM);

display.setTextSize(1);

for(ccc=0; ccc<1000; ccc++) {

display.setCursor(50, 41);

display.println("ccc");

}

display.display(); // Show initial text

// delay(100);

}