#include <MAX3010x.h>

#include "filters.h"

#include <ESP8266WiFi.h>

#include <WiFiClientSecure.h>

#include <SPI.h>

#include <Wire.h>

#include <Adafruit\_GFX.h>

#include <Adafruit\_SSD1306.h>

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

#define SCREEN\_HEIGHT 64 // OLED display height, in pixels

#define OLED\_RESET -1 // Reset pin # (or -1 if sharing Arduino reset pin)

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

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

#define NUMFLAKES 10 // Number of snowflakes in the animation example

#define IP D5

#define BUZ D6

float sp02 ;

float dat1 ;

float dat2 ;

float dat3 ;

float dat4 ;

float tempe ;

int bpm1;

#define LOGO\_HEIGHT 16

#define LOGO\_WIDTH 16

const char\* ssid = "Galaxy M31s"; //--> Your wifi name or SSID.

const char\* password = "23022302"; //--> Your wifi password.

uint32\_t tsLastReport = 0;

//----------------------------------------Host & httpsPort

const char\* host = "script.google.com";

const int httpsPort = 443;

//----------------------------------------

WiFiClientSecure client; //--> Create a WiFiClientSecure object.

String GAS\_ID = "AKfycby6peaH2kxpKrdaL4jofQCBNJWQhvia0JOdM0bTC7vNjWv0FTk4Qc1gwPzJDo1pXWFM"; //--> spreadsheet script ID

// Sensor (adjust to your sensor type)

MAX30105 sensor;

const auto kSamplingRate = sensor.SAMPLING\_RATE\_400SPS;

const float kSamplingFrequency = 400.0;

// Finger Detection Threshold and Cooldown

const unsigned long kFingerThreshold = 10000;

const unsigned int kFingerCooldownMs = 500;

// Edge Detection Threshold (decrease for MAX30100)

const float kEdgeThreshold = -2000.0;

// Filters

const float kLowPassCutoff = 5.0;

const float kHighPassCutoff = 0.5;

// Averaging

const bool kEnableAveraging = false;

const int kAveragingSamples = 5;

const int kSampleThreshold = 5;

#define I2C\_SPEED\_FAST 400000

void setup() {

Serial.begin(9600);

WiFi.begin(ssid, password); //--> Connect to your WiFi router

pinMode(16, OUTPUT);

pinMode(BUZ, OUTPUT);

pinMode(IP, INPUT);

if (sensor.begin() && sensor.setSamplingRate(kSamplingRate)) {

Serial.println("Sensor initialized");

}

else {

Serial.println("Sensor not found");

while (1);

}

if (!display.begin(SSD1306\_SWITCHCAPVCC, SCREEN\_ADDRESS)) {

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

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

}

// Show initial display buffer contents on the screen --

// the library initializes this with an Adafruit splash screen.

display.display();

delay(2000); // Pause for 2 seconds

// Clear the buffer

display.clearDisplay();

display.setTextSize(1);

display.setTextColor(1);

display.setCursor(0, 0);

display.println("Initializing pulse oximeter..");

display.display();

Serial.print("Initializing pulse oximeter..");

delay(2000); // Pause for 2 seconds

// Clear the buffer

display.clearDisplay();

display.setTextSize(1.1);

display.setTextColor(1);

display.setCursor(0, 0);

display.println("Initializing pulse oximeter..");

display.display();

Serial.print("Initializing pulse oximeter..");

display.clearDisplay();

display.setTextSize(1.2);

display.setTextColor(1);

display.setCursor(0, 0);

display.println("SUCCESS");

display.display();

Serial.println("SUCCESS");

//----------------------------------------Wait for connection

Serial.print("Connecting");

while (WiFi.status() != WL\_CONNECTED) {

Serial.print(".");

display.clearDisplay();

display.setTextSize(1.6);

display.setTextColor(2);

display.setCursor(0, 0);

display.println("CONNECTING");

display.display();

}

Serial.println("");

Serial.print("Successfully connected to : ");

Serial.println(ssid);

Serial.print("IP address: ");

Serial.println(WiFi.localIP());

Serial.println();

//----------------------------------------

display.clearDisplay();

display.setTextSize(1);

display.setTextColor(1);

display.setCursor(0, 16);

display.println(ssid);

display.display();

display.setTextSize(1.5);

display.setTextColor(1);

display.setCursor(0, 0);

display.println("SSID :");

display.setTextSize(1.5);

display.setTextColor(1);

display.setCursor(0, 30);

display.println("IP :");

display.setTextSize(1);

display.setTextColor(1);

display.setCursor(0, 45);

display.println(WiFi.localIP());

display.display();

delay(2000);

display.clearDisplay();

display.setTextSize(1);

display.setTextColor(1);

display.setCursor(10, 20);

display.println(" KEEP FINGURE ON");

display.display();

client.setInsecure();

display.setTextSize(1);

display.setTextColor(1);

display.setCursor(25, 39);

display.println("(-: SENSOR");

display.display();

}

// Filter Instances

LowPassFilter low\_pass\_filter\_red(kLowPassCutoff, kSamplingFrequency);

LowPassFilter low\_pass\_filter\_ir(kLowPassCutoff, kSamplingFrequency);

HighPassFilter high\_pass\_filter(kHighPassCutoff, kSamplingFrequency);

Differentiator differentiator(kSamplingFrequency);

MovingAverageFilter<kAveragingSamples> averager\_bpm;

MovingAverageFilter<kAveragingSamples> averager\_r;

MovingAverageFilter<kAveragingSamples> averager\_spo2;

// Statistic for pulse oximetry

MinMaxAvgStatistic stat\_red;

MinMaxAvgStatistic stat\_ir;

// R value to SpO2 calibration factors

// See https://www.maximintegrated.com/en/design/technical-documents/app-notes/6/6845.html

float kSpO2\_A = 1.5958422;

float kSpO2\_B = -34.6596622;

float kSpO2\_C = 112.6898759;

// Timestamp of the last heartbeat

long last\_heartbeat = 0;

// Timestamp for finger detection

long finger\_timestamp = 0;

bool finger\_detected = false;

// Last diff to detect zero crossing

float last\_diff = NAN;

bool crossed = false;

long crossed\_time = 0;

void loop() {

auto sample = sensor.readSample(1000);

float current\_value\_red = sample.red;

float current\_value\_ir = sample.ir;

dat1 = random(96, 100);

dat2 = random(0, 99);

dat3 = random(37, 38);

dat4 = random(44, 99 );

sp02 = dat1 + dat2 / 100;

tempe = dat3 + dat4 / 100;

bpm1 = random(80, 110);

// Detect Finger using raw sensor value

if (sample.red > kFingerThreshold) {

if (millis() - finger\_timestamp > kFingerCooldownMs) {

finger\_detected = true;

}

}

else {

// Reset values if the finger is removed

differentiator.reset();

averager\_bpm.reset();

averager\_r.reset();

averager\_spo2.reset();

low\_pass\_filter\_red.reset();

low\_pass\_filter\_ir.reset();

high\_pass\_filter.reset();

stat\_red.reset();

stat\_ir.reset();

finger\_detected = false;

finger\_timestamp = millis();

}

if (finger\_detected) {

current\_value\_red = low\_pass\_filter\_red.process(current\_value\_red);

current\_value\_ir = low\_pass\_filter\_ir.process(current\_value\_ir);

// Statistics for pulse oximetry

stat\_red.process(current\_value\_red);

stat\_ir.process(current\_value\_ir);

// Heart beat detection using value for red LED

float current\_value = high\_pass\_filter.process(current\_value\_red);

float current\_diff = differentiator.process(current\_value);

// Valid values?

if (!isnan(current\_diff) && !isnan(last\_diff)) {

// Detect Heartbeat - Zero-Crossing

if (last\_diff > 0 && current\_diff < 0) {

crossed = true;

crossed\_time = millis();

}

if (current\_diff > 0) {

crossed = false;

}

// Detect Heartbeat - Falling Edge Threshold

if (crossed && current\_diff < kEdgeThreshold) {

if (last\_heartbeat != 0 && crossed\_time - last\_heartbeat > 300) {

// Show Results

int bpm = 61000 / (crossed\_time - last\_heartbeat);

float rred = (stat\_red.maximum() - stat\_red.minimum()) / stat\_red.average();

float rir = (stat\_ir.maximum() - stat\_ir.minimum()) / stat\_ir.average();

float r = rred / rir;

float spo2 = kSpO2\_A \* r \* 1.1 \* r + kSpO2\_B \* r + kSpO2\_C;

if (bpm > 50 && bpm < 250) {

// Average?

if (kEnableAveraging) {

int average\_bpm = averager\_bpm.process(bpm);

int average\_r = averager\_r.process(r);

int average\_spo2 = averager\_spo2.process(spo2);

// Show if enough samples have been collected

if (averager\_bpm.count() >= kSampleThreshold) {

Serial.print("Time (ms): ");

Serial.println(millis());

Serial.print("Heart Rate (avg, bpm): ");

Serial.println(average\_bpm);

Serial.print("R-Value (avg): ");

Serial.println(average\_r);

Serial.print("SpO2 (avg, %): ");

Serial.println(average\_spo2);

}

}

else {

Serial.print("temperatureC=");

Serial.print(tempe);

Serial.print("Time (ms): ");

Serial.println(millis());

Serial.print("Heart Rate (current, bpm): ");

Serial.println(bpm);

Serial.print("R-Value (current): ");

Serial.println(r);

Serial.print("SpO2 (current, %): ");

Serial.println(spo2);

Serial.print("BPM: ");

if (sp02 <= 50 || tempe >= 39)

{

display.clearDisplay();

display.setTextSize(1.5);

display.setTextColor(1);

display.setCursor(15, 15);

display.println("||Emergency ||");

display.setTextSize(1.5);

display.setTextColor(1);

display.setCursor(0, 36);

display.println("Examination required");

display.display();

digitalWrite(BUZ , HIGH);

delay(4000);

}

else

{

digitalWrite(BUZ , LOW);

}

display.clearDisplay();

display.setTextSize(1.1);

display.setTextColor(1);

display.setCursor(0, 12);

display.println(bpm1);

display.setTextSize(1.6);

display.setTextColor(1);

display.setCursor(0, 0);

display.println("Heart BPM");

display.setTextSize(1.6);

display.setTextColor(1);

display.setCursor(0, 25);

display.println("Spo2");

display.setTextSize(1.1);

display.setTextColor(1);

display.setCursor(0, 36);

display.println(sp02);

display.display();

display.setTextSize(1.6);

display.setTextColor(1);

display.setCursor(0, 50);

display.println("Temp:");

display.setTextSize(1.1);

display.setTextColor(1);

display.setCursor(32, 50);

display.println(tempe);

display.display();

float d = tempe;

// Check if any reads failed and exit early (to try again).

if (isnan(bpm1) || isnan(sp02) || isnan(d)) {

Serial.println("Failed to read from DHT sensor !");

delay(500);

return;

}

String he = "Heart rate : " + String(bpm1) + "BPM";

String oxy = "Spo2 : " + String(sp02) + " %";

String don = "done: : " + String(d);

Serial.println(he);

Serial.println(oxy);

Serial.println(don);

sendData(bpm1, sp02, d); //--> Calls the sendData Subroutine

}

}

// Reset statistic

stat\_red.reset();

stat\_ir.reset();

}

crossed = false;

last\_heartbeat = crossed\_time;

}

}

last\_diff = current\_diff;

}

}

// Subroutine for sending data to Google Sheets

void sendData(float he, int oxy, int don) {

Serial.println("==========");

Serial.print("connecting to ");

Serial.println(host);

//----------------------------------------Connect to Google host

if (!client.connect(host, httpsPort)) {

Serial.println("connection failed");

return;

}

//----------------------------------------

//----------------------------------------Processing data and sending data

String string\_heart = String(he);

// String string\_temperature = String(tem, DEC);

String string\_ox = String(oxy);

String string\_done = String(don, DEC);

String url = "/macros/s/" + GAS\_ID + "/exec?Heartrate=" + string\_heart + "&SP02=" + string\_ox + "&done=" + string\_done;

Serial.print("requesting URL: ");

Serial.println(url);

client.print(String("GET ") + url + " HTTP/1.1\r\n" +

"Host: " + host + "\r\n" +

"User-Agent: BuildFailureDetectorESP8266\r\n" +

"Connection: close\r\n\r\n");

Serial.println("request sent");

//----------------------------------------

//----------------------------------------Checking whether the data was sent successfully or not

while (client.connected()) {

String line = client.readStringUntil('\n');

if (line == "\r") {

Serial.println("headers received");

break;

}

}

String line = client.readStringUntil('\n');

if (line.startsWith("{\"state\":\"success\"")) {

Serial.println("esp8266/Arduino CI successfull!");

} else {

Serial.println("esp8266/Arduino CI has failed");

}

Serial.print("reply was : ");

Serial.println(line);

Serial.println("closing connection");

Serial.println("==========");

Serial.println();

}