#include <SoftwareSerial.h>

#define DEBUG true

SoftwareSerial esp8266(9,10);

#include <LiquidCrystal\_PCF8574.h>

#include <stdlib.h>

LiquidCrystal\_PCF8574 lcd(0x27); // set the LCD address to 0x27 for a 16 chars and 2 line display

#define SSID "ABC" // "SSID-WiFiname"

#define PASS "12345678" // "password"

#define IP "184.106.153.149"// thingspeak.com ip

String msg = "GET /update?key=5XVYVCOZLZK6OFI6"; //change it with your api key like "GET /update?key=Your Api Key"

//Variables

int switchState2 = 0;

int switchState6 = 0;

int switchState7 = 0;

float temp;

int hum;

String tempC;

int error;

int pulsePin = 0; // Pulse Sensor purple wire connected to analog pin 0

int blinkPin = 3; // pin to blink led at each beat

int fadePin = 5;

int fadeRate = 0;

// Volatile Variables, used in the interrupt service routine!

volatile int BPM; // int that holds raw Analog in 0. updated every 2mS

volatile int Signal; // holds the incoming raw data

volatile int IBI = 600; // int that holds the time interval between beats! Must be seeded!

volatile boolean Pulse = false; // "True" when heartbeat is detected. "False" when not a "live beat".

volatile boolean QS = false; // becomes true when Arduino finds a beat.

// Regards Serial OutPut -- Set This Up to your needs

static boolean serialVisual = true; // Set to 'false' by Default. Re-set to 'true' to see Arduino Serial Monitor ASCII Visual Pulse

volatile int rate[10]; // array to hold last ten IBI values

volatile unsigned long sampleCounter = 0; // used to determine pulse timing

volatile unsigned long lastBeatTime = 0; // used to find IBI

volatile int P =512; // used to find peak in pulse wave, seeded

volatile int T = 512; // used to find trough in pulse wave, seeded

volatile int thresh = 525; // used to find instant moment of heart beat, seeded

volatile int amp = 100; // used to hold amplitude of pulse waveform, seeded

volatile boolean firstBeat = true; // used to seed rate array so we startup with reasonable BPM

volatile boolean secondBeat = false; // used to seed rate array so we startup with reasonable BPM

void setup()

{

lcd.setBacklight(255);

lcd.begin(16, 2);

lcd.print("Group-1");

delay(100);

lcd.setCursor(0,1);

lcd.print("Connecting...");

Serial.begin(9600); //or use default 115200.

esp8266.begin(115200);

Serial.println("AT");

esp8266.println("AT");

delay(5000);

if(esp8266.find("OK")){

connectWiFi();

}

interruptSetup();

pinMode(3, OUTPUT);

pinMode(4, OUTPUT);

pinMode(5, OUTPUT);

digitalWrite(3, LOW); //Set lights to off initially

digitalWrite(4,LOW);

digitalWrite(5,LOW);

//lcd.noDisplay();

lcd.print("BPM= 0");

}

void loop(){

lcd.clear();

lcd.display();

lcd.setCursor(0, 0);

if(BPM<300 && BPM>50){ //Reset if over or below 150 or 50

lcd.print("BPM = ");

lcd.print(BPM);

}

else{

lcd.setCursor(0, 0);

lcd.print("BPM = 0");

}

delay (100);

lcd.setCursor(0, 1); // set the cursor to column 0, line 2

start: //label

error=0;

delay(500);

updatebeat();

//Resend if transmission is not completed

if (error==1){

goto start; //go to label "start"

}

delay(500);

}

void updatebeat(){

String cmd = "AT+CIPSTART=\"TCP\",\"";

cmd += IP;

cmd += "\",80";

Serial.println(cmd);

esp8266.println(cmd);

delay(2000);

if(esp8266.find("Error")){

return;

}

cmd = msg ;

cmd += "&field1=";

cmd += BPM;

cmd += "\r\n";

Serial.print("AT+CIPSEND=");

esp8266.print("AT+CIPSEND=");

Serial.println(cmd.length());

esp8266.println(cmd.length());

Serial.print(cmd);

esp8266.print(cmd);

}

boolean connectWiFi(){

Serial.println("AT+CWMODE=1");

esp8266.println("AT+CWMODE=1");

delay(2000);

String cmd="AT+CWJAP=\"";

cmd+=SSID;

cmd+="\",\"";

cmd+=PASS;

cmd+="\"";

Serial.println(cmd);

esp8266.println(cmd);

delay(5000);

if(esp8266.find("OK")){

Serial.println("OK");

return true;

}else{

return false;

}

}

void interruptSetup(){

TCCR2A = 0x02; // DISABLE PWM ON DIGITAL PINS 3 AND 11, AND GO INTO CTC MODE

TCCR2B = 0x06; // DON'T FORCE COMPARE, 256 PRESCALER

OCR2A = 0X7C; // SET THE TOP OF THE COUNT TO 124 FOR 500Hz SAMPLE RATE

TIMSK2 = 0x02; // ENABLE INTERRUPT ON MATCH BETWEEN TIMER2 AND OCR2A

sei(); // MAKE SURE GLOBAL INTERRUPTS ARE ENABLED

}

ISR(TIMER2\_COMPA\_vect){ // triggered when Timer2 counts to 124

cli(); // disable interrupts while we do this

Signal = analogRead(pulsePin); // read the Pulse Sensor

sampleCounter += 2; // keep track of the time in mS

int N = sampleCounter - lastBeatTime; // monitor the time since the last beat to avoid noise

// find the peak and trough of the pulse wave

if(Signal < thresh && N > (IBI/5)\*3){ // avoid dichrotic noise by waiting 3/5 of last IBI

if (Signal < T){ // T is the trough

T = Signal; // keep track of lowest point in pulse wave

}

}

if(Signal > thresh && Signal > P){ // thresh condition helps avoid noise

P = Signal; // P is the peak

} // keep track of highest point in pulse wave

// NOW IT'S TIME TO LOOK FOR THE HEART BEAT

// signal surges up in value every time there is a pulse

if (N > 250){ // avoid high frequency noise

if ( (Signal > thresh) && (Pulse == false) && (N > (IBI/5)\*3) ){

Pulse = true; // set the Pulse flag when there is a pulse

digitalWrite(blinkPin,HIGH); // turn on pin 13 LED

IBI = sampleCounter - lastBeatTime; // time between beats in mS

lastBeatTime = sampleCounter; // keep track of time for next pulse

if(secondBeat){ // if this is the second beat

secondBeat = false; // clear secondBeat flag

for(int i=0; i<=9; i++){ // seed the running total to get a realistic BPM at startup

rate[i] = IBI;

}

}

if(firstBeat){ // if it's the first time beat is found

firstBeat = false; // clear firstBeat flag

secondBeat = true; // set the second beat flag

sei(); // enable interrupts again

return; // IBI value is unreliable so discard it

}

word runningTotal = 0; // clear the runningTotal variable

for(int i=0; i<=8; i++){ // shift data in the rate array

rate[i] = rate[i+1]; // and drop the oldest IBI value

runningTotal += rate[i]; // add up the 9 oldest IBI values

}

rate[9] = IBI; // add the latest IBI to the rate array

runningTotal += rate[9]; // add the latest IBI to runningTotal

runningTotal /= 10; // average the last 10 IBI values

BPM = 60000/runningTotal; // how many beats can fit into a minute? that's BPM!

QS = true; // set Quantified Self flag

// QS FLAG IS NOT CLEARED INSIDE THIS ISR

if(BPM>170){ //Red

digitalWrite(4, HIGH);

digitalWrite(5, LOW);

}

else if(BPM>99){

digitalWrite(4, LOW);

digitalWrite(5, LOW);

}

else{ //green

digitalWrite(4, LOW);

digitalWrite(5, HIGH);

}

}

}

if (Signal < thresh && Pulse == true){ // when the values are going down, the beat is over

digitalWrite(blinkPin,LOW); // turn off pin 3 LED

Pulse = false; // reset the Pulse flag so we can do it again

amp = P - T; // get amplitude of the pulse wave

thresh = amp/2 + T; // set thresh at 50% of the amplitude

P = thresh; // reset these for next time

T = thresh;

}

if (N > 2500){ // if 2.5 seconds go by without a beat

thresh = 512; // set thresh default

P = 512; // set P default

T = 512; // set T default

lastBeatTime = sampleCounter; // bring the lastBeatTime up to date

firstBeat = true; // set these to avoid noise

secondBeat = false; // when we get the heartbeat back

}

sei();

// enable interrupts when youre done!

}// end isr