#CODE

#define USE\_ARDUINO\_INTERRUPTS true

#define DEBUG true

#define SSID "\*\*\*\*\*\*\*\*" // "SSID-WiFiname"

#define PASS "\*\*\*\*\*\*\*\*\*\*\*\*" // "password"

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

#include <SoftwareSerial.h>

#include "Timer.h"

#include <PulseSensorPlayground.h> // Includes the PulseSensorPlayground Library.

Timer t;

PulseSensorPlayground pulseSensor;

String msg = "GET /update?key=your api key";

SoftwareSerial esp8266(10,11);

//Variables

const int PulseWire = A0; // PulseSensor PURPLE WIRE connected to ANALOG PIN 0

const int LED13 = 13; // The on-board Arduino LED, close to PIN 13.

int Threshold = 550; //for heart rate sensor

float myTemp;

int myBPM;

String BPM;

String temp;

int error;

int panic;

int raw\_myTemp;

float Voltage;

float tempC;

void setup()

{

Serial.begin(9600);

esp8266.begin(115200);

pulseSensor.analogInput(PulseWire);

pulseSensor.blinkOnPulse(LED13); //auto-magically blink Arduino's LED with heartbeat.

pulseSensor.setThreshold(Threshold);

// Double-check the "pulseSensor" object was created and "began" seeing a signal.

if (pulseSensor.begin()) {

Serial.println("We created a pulseSensor Object !"); //This prints one time at Arduino power-up, or on Arduino reset.

}

Serial.println("AT");

esp8266.println("AT");

delay(3000);

if(esp8266.find("OK"))

{

connectWiFi();

}

t.every(10000, getReadings);

t.every(10000, updateInfo);

}

void loop()

{

panic\_button();

start: //label

error=0;

t.update();

//Resend if transmission is not completed

if (error==1)

{

goto start; //go to label "start"

}

delay(4000);

}

void updateInfo()

{

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="; //field 1 for BPM

cmd += BPM;

cmd += "&field2="; //field 2 for temperature

cmd += temp;

cmd += "\r\n";

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

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

Serial.println(cmd.length());

esp8266.println(cmd.length());

if(esp8266.find(">"))

{

Serial.print(cmd);

esp8266.print(cmd);

}

else

{

Serial.println("AT+CIPCLOSE");

esp8266.println("AT+CIPCLOSE");

//Resend...

error=1;

}

}

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"))

{

return true;

}

else

{

return false;

}

}

void getReadings(){

raw\_myTemp = analogRead(A1);

Voltage = (raw\_myTemp / 1023.0) \* 5000; // 5000 to get millivots.

tempC = Voltage \* 0.1;

myTemp = (tempC \* 1.8) + 32; // conver to F

Serial.println(myTemp);

int myBPM = pulseSensor.getBeatsPerMinute(); // Calls function on our pulseSensor object that returns BPM as an "int".

// "myBPM" hold this BPM value now.

if (pulseSensor.sawStartOfBeat()) { // Constantly test to see if "a beat happened".

Serial.println(myBPM); // Print the value inside of myBPM.

}

delay(20);

char buffer1[10];

char buffer2[10];

BPM = dtostrf(myBPM, 4, 1, buffer1);

temp = dtostrf(myTemp, 4, 1, buffer2);

}

void panic\_button(){

panic = digitalRead(8);

if(panic == HIGH){

Serial.println(panic);

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 += "&field3=";

cmd += panic;

cmd += "\r\n";

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

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

Serial.println(cmd.length());

esp8266.println(cmd.length());

if(esp8266.find(">"))

{

Serial.print(cmd);

esp8266.print(cmd);

}

else

{

Serial.println("AT+CIPCLOSE");

esp8266.println("AT+CIPCLOSE");

//Resend...

error=1;

}

}

}