#include "Adafruit\_FONA.h"

#include<string.h>

#define FONA\_RX 2

#define FONA\_TX 3

#define FONA\_RST 4

#include <SoftwareSerial.h>

SoftwareSerial fonaSS = SoftwareSerial(FONA\_TX, FONA\_RX);

SoftwareSerial \*fonaSerial = &fonaSS;

Adafruit\_FONA fona = Adafruit\_FONA(FONA\_RST);

uint8\_t readline(char \*buff, uint8\_t maxbuff, uint16\_t timeout = 0);

uint8\_t type =FONA808\_V2;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*CO sensor\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#define MQ\_PIN (1) //define which analog input channel you are going to use

#define RL\_VALUE (1) //define the load resistance on the board, in kilo ohms

#define RO\_CLEAN\_AIR\_FACTOR (3.3) //RO\_CLEAR\_AIR\_FACTOR=(Sensor resistance in clean air)/RO,

#define GAS\_CO (1)

//which is derived from the chart in datasheet

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Software Related Macros\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#define CALIBARAION\_SAMPLE\_TIMES (50) //define how many samples you are going to take in the calibration phase

#define CALIBRATION\_SAMPLE\_INTERVAL (500) //define the time interal(in milisecond) between each samples in the

//cablibration phase

#define READ\_SAMPLE\_INTERVAL (50) //define how many samples you are going to take in normal operation

#define READ\_SAMPLE\_TIMES (5) //define the time interal(in milisecond) between each samples in

//normal operation

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Application Related Macros\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int co2ppm = 0; //int for calculated ppm

float COCurve[3] = {0.5,0.21,-4.79};

float Ro = 3.65;

#define anInput A2 //analog feed from

#define digTrigger 3 //digital feed from

#define co2Zero 55 //calibrated CO2 0 level

String nerestring;

int microphonValue = 0;

int n\_Samples = 1000; // numero de muestras para promediar

int sensorValueEskala = 0;

int sensorValue = 0;

void setup() {

pinMode(anInput,INPUT); //MQ135 analog feed set for input

Serial.begin(115200);

Serial.println(F("FONA basic test"));

Serial.println(F("Initializing....(May take 3 seconds)"));

fonaSerial->begin(4800);

if (! fona.begin(\*fonaSerial)) {

Serial.println(F("Couldn't find FONA"));

while (1);

}

type = fona.type();

Serial.println(F("FONA is OK"));

Serial.print(F("Found "));

switch (type) {

case FONA800L:

Serial.println(F("FONA 800L")); break;

case FONA800H:

Serial.println(F("FONA 800H")); break;

case FONA808\_V1:

Serial.println(F("FONA 808 (v1)")); break;

case FONA808\_V2:

Serial.println(F("FONA 808 (v2)")); break;

case FONA3G\_A:

Serial.println(F("FONA 3G (American)")); break;

case FONA3G\_E:

Serial.println(F("FONA 3G (European)")); break;

default:

Serial.println(F("???")); break;

}

}

void loop() {

// Print module IMEI number.

char imei[16] = {0}; // MUST use a 16 character buffer for IMEI!

uint8\_t imeiLen = fona.getIMEI(imei);

if (imeiLen > 0) {

Serial.print("Module IMEI: ");

Serial.println(imei);

}

fona.setGPRSNetworkSettings(F("internet.euskaltel.mobi"), F("CLIENTE"), F("EUSKALTEL"));

fona.setHTTPSRedirect(true);

// Unlock the SIM with a PIN code

char PIN[4]= "4998";

Serial.println(PIN);

Serial.print(F("Unlocking SIM card: "));

if (! fona.unlockSIM(PIN)) {

Serial.println(F("Failed"));

} else {

Serial.println(F("OK!"));

}

if (!fona.enableGPS(true))

Serial.println(F("Failed to turn on"));

delay (4000);

if (!fona.enableGPRS(false))

Serial.println(F("Failed to turn off"));

delay (4000);

if (!fona.enableGPRS(true))

Serial.println(F("Failed to turn on"));

delay (4000);

float latitude, longitude, speed\_kph, heading, speed\_mph, altitude;

boolean gps\_success = fona.getGPS(&latitude, &longitude, &speed\_kph, &heading, &altitude);

if (gps\_success) {

Serial.println(latitude, 6);

Serial.println(longitude, 6);

}

delay (2000);

Serial.print("CO:");

Serial.print(MQGetGasPercentage(MQRead(MQ\_PIN)/Ro,GAS\_CO) );

Serial.print( "ppm" );

Serial.print(" ");

delay(200);

CO2();

delay (4000);

float batezBestekoa = getbatezBestekoa (n\_Samples);

//Serial.print ("sensorValueEskala: ");

//Serial.println (sensorValueEskala);

Serial.print ("batezBestekoa: ");

Serial.println (batezBestekoa);

int dB\_tan = 20 \* log (batezBestekoa);

Serial.print ("dB\_tan: ");

Serial.println (dB\_tan);

delay (4000);

Serial.println(dB\_tan);

delay(6000);

uint16\_t statuscode;

int16\_t length;

char topollution[]="https://topollution.herokuapp.com/api/device?device\_id=";

char id[]="1";

char lati[]="&latitud=";

char longi[]="&longitud=";

char dbs[]="&dbs=";

char co2[]="&co2=";

char co[]="&co=";

char net[]="&net=lora";

char url[180];

char data[180];

nerestring += String(topollution);

nerestring += String(id);

nerestring += String(lati);

nerestring += String(latitude, 6);

nerestring += String(longi);

nerestring += String(longitude, 6);

nerestring += String(dbs);

nerestring += String(dB\_tan);

nerestring += String(co2);

nerestring += String(co2ppm);

nerestring += String(co);

nerestring += String(MQGetGasPercentage(MQRead(MQ\_PIN)/Ro,GAS\_CO));

nerestring += String(net);

delay(1000);

nerestring.toCharArray(url, 180);

delay (6000);

Serial.println(F("URL to post (topollution.herokuapp.com/api/device?device\_id&latitud&longitud&dbs&co2&co&net):"));

Serial.print(F(""));

Serial.println(url);

delay(8000);

Serial.println(F("Data to post (e.g. \"foo\" or \"{\"simple\":\"json\"}\"):"));

Serial.println(data);

Serial.println(F("\*\*\*\*"));

if (!fona.HTTP\_POST\_start(url, F("text/plain"), (uint8\_t \*) data, strlen(data), &statuscode, (uint16\_t \*)&length)) {

Serial.println("Failed!");

}

while (length > 0) {

while (fona.available()) {

char c = fona.read();

#if defined(\_\_AVR\_ATmega328P\_\_) || defined(\_\_AVR\_ATmega168\_\_)

loop\_until\_bit\_is\_set(UCSR0A, UDRE0); /\* Wait until data register empty. \*/

UDR0 = c;

#else

Serial.write(c);

#endif

length--;

if (! length) break;

}

}

Serial.println(F("\n\*\*\*\*"));

fona.HTTP\_POST\_end();

delay (60000);

}

float MQResistanceCalculation(int raw\_adc){

return ( ((float)RL\_VALUE\*(1023-raw\_adc)/raw\_adc));

}

float MQCalibration(int mq\_pin){

int i;

float val=0;

for (i=0;i<CALIBARAION\_SAMPLE\_TIMES;i++) { //take multiple samples

val += MQResistanceCalculation(analogRead(mq\_pin));

delay(CALIBRATION\_SAMPLE\_INTERVAL);

}

val = val/CALIBARAION\_SAMPLE\_TIMES; //calculate the average value

val = val/RO\_CLEAN\_AIR\_FACTOR; //divided by RO\_CLEAN\_AIR\_FACTOR yields the Ro

return val;

}

float MQRead(int mq\_pin)

{

int i;

float rs=0;

for (i=0;i<READ\_SAMPLE\_TIMES;i++) {

rs += MQResistanceCalculation(analogRead(mq\_pin));

delay(READ\_SAMPLE\_INTERVAL);

}

rs = rs/READ\_SAMPLE\_TIMES;

return rs;

}

int MQGetGasPercentage(float rs\_ro\_ratio, int gas\_id){

if ( gas\_id == GAS\_CO ) {

return MQGetPercentage(rs\_ro\_ratio,COCurve);

}

return 0;

}

int MQGetPercentage(float rs\_ro\_ratio, float \*pcurve){

return (pow(10,( ((log(rs\_ro\_ratio)-pcurve[1])/pcurve[2]) + pcurve[0])));

}

float getbatezBestekoa(int n) { // hacemos "n" mediciones

float SUMA\_n = 0;

for (int i = 0; i < n; i++) {

sensorValue = analogRead(A3);

//alternoa DC-ra pasa (errektifikatu)

if (sensorValue < 248){

sensorValue = (248-sensorValue) + 248;

}

//sensorValue bariblea 0 eta 255 artean mapeatzea

sensorValueEskala = map (sensorValue, 248, 490, 0, 255);

SUMA\_n += sensorValueEskala;

}

return( SUMA\_n /n); // Promedio

}

void CO2()

{

int co2now[10]; //int lecturas co2

int co2raw = 0; //int valor raw co2

int co2comp = 0; //int para conpensacion co2

int co2ppm = 0; //int valor en ppm

int zzz = 0; //int para sumar mediciones

for (int x = 0;x<10;x++){ //samplpe co2 10x over 2 seconds

co2now[x]=analogRead(A2);

delay(200);

}

for (int x = 0;x<10;x++){ //add samples together

zzz=zzz + co2now[x];

}

co2raw = zzz/10; //divide samples by 10

co2comp = co2raw - co2Zero; //get compensated value

co2ppm = map(co2comp,0,1023,400,1600); //mapear el valor para ppm

Serial.print("CO2 "); //print title

Serial.println(co2ppm); //print co2 ppm

}