UNIVERSITAT POLITÈCNICA DE VALÈNCIA

ESCUELA POLITECNICA SUPERIOR DE GANDIA

Anexo 2: códigos de programación usados

TRABAJO FINAL DE MÁSTER:

"Desarrollo de un array de sensores para el estudio del tipo de gas y la variabilidad de la planta *Cistus Ladanifer*"

Autora:

Karen Gisset Zúñiga Ortiz

Tutor:

Jaime Iloret Mauri

Máster Universitario en Evaluación y Seguimiento Ambiental de Ecosistemas Marinos y Costeros

Gandía 2022

Contenido

Introducción

Arduino ofrece un entorno de desarrollo gratuito en el que realizar programas para que sean volcados al microcontrolador posteriormente vía USB. Arduino se programa mediante el uso de un lenguaje propio basado en el lenguaje de programación de alto nivel Processing. Sin embargo, se pueden emplear otros lenguajes y aplicaciones en Arduino, ya que emplea la transmisión serial de datos soportada por la mayoría de los lenguajes. Incluso es posible emplear software intermediario para aquellos lenguajes no compatibles con el formato serie (Arduino, 2022).

En este documento se presentarán los códigos de programación usados para los 5 sensores que se usaron para el desarrollo del trabajo de fin de master.

Sensor 1

```
#include <MQ135.h>
#include <DHT.h>
#include "MQUnifiedsensor.h"

#include <SPI.h>
#include <SD.h>

#include <Wire.h>
#include "RTClib.h"
RTC_DS1307 RTC;

#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>

/********************** Variables necesarias SD y reloj **************/
File myFile;
String fecha,fecha_dia,hora_dia;
int anyo,mes,dia,hora,minuto,segundo;
String nombre_diario;
String data_update;
boolean dato_ya_actualizado = false;
boolean dia_nuevo = false;
boolean SD_iniciada = false;
boolean actualizar_datos = false;
String cabecera_archivo =
"Timestamp;Temperatura;Humedad;MQ135(CO2);MQ2(H2);MQ3(Benceno);MQ4(Hum
o);MQ5(H2);MQ6(H2);MQ7(H2);MQ8(H2)";
int ultimo_seg=0;
int ultimo_min=0;
/*  Tiempo entre actualizaciones en tarjeta SD
    variable min_o_seg debe tener dos posibles valores:
        min_o_seg = 0 --> actualizacion en segundos
```

```
      min_o_seg = 1 --> actualizacion en minutos
   La variable updating_time hará referencia a minutos o segundos depeniendo del valor
   de la variable min_o_seg. Los valores permitidos son: 1, 2, 3, 4, 5, 6, 10, 12, 15, 20, 30 y 60.
*/
int min_o_seg = 0;
int updating_time = 60;

/**********************LCD**************/
#define ANCHO 128
#define ALTO 32

#define OLED_RESET 4
Adafruit_SSD1306 display(ANCHO,ALTO,&Wire,OLED_RESET);

/**********************placa usada**************/
#define       Board                 ("Arduino MEGA")
#define       Voltage_Resolution    (5)
#define       ADC_Bit_Resolution    (10) // For arduino UNO/MEGA/NANO

/************************CALIBRACION MQ2***********************************/
#define       Pin_MQ2               (A1)  //Analog input 1 of your arduino
/************************Software Related Macros***********************************/
#define       Type_MQ2              ("MQ-2") //MQ2
#define       RatioMQ2CleanAir      (9.83) //RS / R0 = 9.83 ppm
MQUnifiedsensor MQ2(Board, Voltage_Resolution, ADC_Bit_Resolution, Pin_MQ2, Type_MQ2);
/*****************************Globals***********************************************/


/************************CALIBRACION MQ3***********************************/
#define       Pin_MQ3               (A2)  //Analog input 2 of your arduino
/************************Software Related Macros***********************************/
```

```
#define         Type_MQ3                ("MQ-3") //MQ3
#define         RatioMQ3CleanAir        (60) //RS / R0 = 60 ppm
MQUnifiedsensor MQ3(Board, Voltage_Resolution, ADC_Bit_Resolution, Pin_MQ3, Type_MQ3);
/*****************************Globals***********************************************/

/*************************CALIBRACION MQ4*********************************/
#define         Pin_MQ4                 (A3)  //Analog input 3 of your arduino
/************************Software Related Macros************************************/
#define         Type_MQ4                ("MQ-4") //MQ4
#define         RatioMQ4CleanAir        (4.4) //RS / R0 = 60 ppm
MQUnifiedsensor MQ4(Board, Voltage_Resolution, ADC_Bit_Resolution, Pin_MQ4, Type_MQ4);

/*************************CALIBRACION MQ5*********************************/
#define         Pin_MQ5                 (A4)  //Analog input 4 of your arduino
/************************Software Related Macros************************************/
#define         Type_MQ5                ("MQ-5") //MQ5
#define         RatioMQ5CleanAir        (6.5) //RS / R0 = 60 ppm
MQUnifiedsensor MQ5(Board, Voltage_Resolution, ADC_Bit_Resolution, Pin_MQ5, Type_MQ5);

/*************************CALIBRACION MQ6*********************************/
#define         Pin_MQ6                 (A5)  //Analog input 5 of your arduino
/************************Software Related Macros************************************/
#define         Type_MQ6                ("MQ-6") //MQ6
#define         RatioMQ6CleanAir        (10) //RS / R0 = 60 ppm
MQUnifiedsensor MQ6(Board, Voltage_Resolution, ADC_Bit_Resolution, Pin_MQ6, Type_MQ6);

/*************************CALIBRACION MQ7*********************************/
#define         Pin_MQ7                 (A6)  //Analog input 6 of your arduino
/************************Software Related Macros************************************/
#define         Type_MQ7                ("MQ-7") //MQ7
```

```
#define         RatioMQ7CleanAir        (27.5) //RS / R0 = 60 ppm
MQUnifiedsensor MQ7(Board, Voltage_Resolution, ADC_Bit_Resolution, Pin_MQ7, Type_MQ7);

/************************CALIBRACION MQ8***********************************/
#define         Pin_MQ8                 (A7)  //Analog input 7 of your arduino
/************************Software Related Macros***********************************/
#define         Type_MQ8                ("MQ-8") //MQ8
#define         RatioMQ8CleanAir        (70) //RS / R0 = 60 ppm
MQUnifiedsensor MQ8(Board, Voltage_Resolution, ADC_Bit_Resolution, Pin_MQ8, Type_MQ8);

#define PIN_MQ135 A0 // MQ135 Analog Input Pin
#define DHTPIN 2 // DHT Digital Input Pin
#define DHTTYPE DHT11 // DHT11 or DHT22, depends on your sensor

MQ135 mq135_sensor(PIN_MQ135);
DHT dht(DHTPIN, DHTTYPE);
float temperature, humidity; // Temp and Humid floats, will be measured by the DHT
float MQ2_ppm,MQ3_ppm,MQ4_ppm,MQ5_ppm,MQ6_ppm,MQ7_ppm,MQ8_ppm;

void setup() {
  Serial.begin(9600);
  display.begin(SSD1306_SWITCHCAPVCC, 0x3C);
  display.clearDisplay();
  display.setTextSize(1);
  display.setTextColor(WHITE,BLACK);
  display.setCursor(0,0);

  while(!SD_iniciada){
    inicializar_SD();
  }
```

```
  dht.begin();


  Wire.begin();
  RTC.begin();
  //RTC.adjust(DateTime(__DATE__, __TIME__)); // Establece la fecha y hora
(Comentar una vez establecida la hora)


  display.begin(SSD1306_SWITCHCAPVCC, 0x3C);
  display.clearDisplay();
  display.setTextSize(1);
  display.setTextColor(WHITE,BLACK);
  display.setCursor(0,0);


  obtener_fecha();
  ultimo_seg = segundo;
  file_existe(nombre_diario);


// calibraciones del MQ2
  MQ2.setRegressionMethod(1); //_PPM =  a*ratio^b
  MQ2.setA(987.99); MQ2.setB(-2.162); // Configure the equation to to calculate LPG
concentration
  MQ2.init();


  Serial.print("Calibrating MQ2, please wait.");
  float calcR0_MQ2 = 0;
  for(int i = 1; i<=10; i ++)
  {
    MQ2.update(); // Update data, the arduino will read the voltage from the analog pin
    calcR0_MQ2 += MQ2.calibrate(RatioMQ2CleanAir);
    Serial.print(".");
  }
  MQ2.setR0(calcR0_MQ2/10);
  Serial.println("  done!.");
```

```
  if(isinf(calcR0_MQ2)) {Serial.println("Warning: Conection issue, R0 is infinite (Open circuit detected) please check your wiring and supply"); while(1);}
  if(calcR0_MQ2 == 0){Serial.println("Warning: Conection issue found, R0 is zero (Analog pin shorts to ground) please check your wiring and supply"); while(1);}
  /*****************************  MQ CAlibration ********************************************/

  MQ2.serialDebug(true);

// FIN calibraciones del MQ2

// calibraciones del MQ3
  MQ3.setRegressionMethod(1); //_PPM =  a*ratio^b
  MQ3.setA(4.8387); MQ3.setB(-2.68); // Configure the equation to to calculate Benzene concentration
  MQ3.init();

  Serial.print("Calibrating MQ3, please wait.");
  float calcR0_MQ3 = 0;
  for(int i = 1; i<=10; i ++)
  {
    MQ3.update(); // Update data, the arduino will read the voltage from the analog pin
    calcR0_MQ3 += MQ3.calibrate(RatioMQ3CleanAir);
    Serial.print(".");
  }
  MQ3.setR0(calcR0_MQ3/10);
  Serial.println("  done!.");

  if(isinf(calcR0_MQ3)) {Serial.println("Warning: Conection issue, R0 is infinite (Open circuit detected) please check your wiring and supply"); while(1);}
  if(calcR0_MQ3 == 0){Serial.println("Warning: Conection issue found, R0 is zero (Analog pin shorts to ground) please check your wiring and supply"); while(1);}
  /*****************************  MQ CAlibration ********************************************/
  MQ3.serialDebug(true);
```

```
// FIN calibraciones del MQ3


// calibraciones del MQ4
  MQ4.setRegressionMethod(1); //_PPM =  a*ratio^b
  MQ4.setA(30000000); MQ4.setB(-8.308); // Configure the equation to to calculate CH4 concentration
  MQ4.init();


Serial.print("Calibrating please wait.");
  float calcR0_MQ4 = 0;
  for(int i = 1; i<=10; i ++)
  {
    MQ4.update(); // Update data, the arduino will read the voltage from the analog pin
    calcR0_MQ4 += MQ4.calibrate(RatioMQ4CleanAir);
    Serial.print(".");
  }
  MQ4.setR0(calcR0_MQ4/10);
  Serial.println("  done!.");


  if(isinf(calcR0_MQ4)) {Serial.println("Warning: Conection issue, R0 is infinite (Open circuit detected) please check your wiring and supply"); while(1);}
  if(calcR0_MQ4 == 0){Serial.println("Warning: Conection issue found, R0 is zero (Analog pin shorts to ground) please check your wiring and supply"); while(1);}
  /*****************************  MQ CAlibration ********************************************/
  MQ4.serialDebug(true);
// FIN calibraciones del MQ4


// calibraciones del MQ5
  MQ5.setRegressionMethod(1); //_PPM =  a*ratio^b
  MQ5.setA(1163.8); MQ5.setB(-3.874); // Configure the equation to to calculate H2 concentration
  MQ5.init();


  Serial.print("Calibrating please wait.");
```

```
  float calcR0_MQ5 = 0;
  for(int i = 1; i<=10; i ++)
  {
    MQ5.update(); // Update data, the arduino will read the voltage from the analog pin
    calcR0_MQ5 += MQ5.calibrate(RatioMQ5CleanAir);
    Serial.print(".");
  }
  MQ5.setR0(calcR0_MQ5/10);
  Serial.println("  done!.");

  if(isinf(calcR0_MQ5)) {Serial.println("Warning: Conection issue, R0 is infinite (Open circuit detected) please check your wiring and supply"); while(1);}
  if(calcR0_MQ5 == 0){Serial.println("Warning: Conection issue found, R0 is zero (Analog pin shorts to ground) please check your wiring and supply"); while(1);}
  /*****************************  MQ CAlibration ********************************************/
  MQ5.serialDebug(true);

// FIN calibraciones del MQ5

// calibraciones del MQ6
MQ6.setRegressionMethod(1); //_PPM =  a*ratio^b
  MQ6.setA(88158); MQ6.setB(-3.597); // Configure the equation to to calculate CH4 concentration
  MQ6.init();

  Serial.print("Calibrating please wait.");
  float calcR0_MQ6 = 0;
  for(int i = 1; i<=10; i ++)
  {
    MQ6.update(); // Update data, the arduino will read the voltage from the analog pin
    calcR0_MQ6 += MQ6.calibrate(RatioMQ6CleanAir);
    Serial.print(".");
  }
  MQ6.setR0(calcR0_MQ6/10);
```

```
  Serial.println("  done!.");


  if(isinf(calcR0_MQ6)) {Serial.println("Warning: Conection issue, R0 is infinite (Open circuit detected) please check your wiring and supply"); while(1);}

  if(calcR0_MQ6 == 0){Serial.println("Warning: Conection issue found, R0 is zero (Analog pin shorts to ground) please check your wiring and supply"); while(1);}

  /*****************************  MQ CAlibration ********************************************/

  MQ6.serialDebug(true);


// FIN calibraciones del MQ6


// calibraciones del MQ7

  MQ7.setRegressionMethod(1); //_PPM =  a*ratio^b

  MQ7.setA(69.014); MQ7.setB(-1.374); // Configure the equation to calculate CO concentration value

  MQ7.init();


  Serial.print("Calibrating please wait.");

  float calcR0_MQ7 = 0;

  for(int i = 1; i<=10; i ++)

  {

    MQ7.update(); // Update data, the arduino will read the voltage from the analog pin

    calcR0_MQ7 += MQ7.calibrate(RatioMQ7CleanAir);

    Serial.print(".");

  }

  MQ7.setR0(calcR0_MQ7/10);

  Serial.println("  done!.");


  if(isinf(calcR0_MQ7)) {Serial.println("Warning: Conection issue, R0 is infinite (Open circuit detected) please check your wiring and supply"); while(1);}

  if(calcR0_MQ7 == 0){Serial.println("Warning: Conection issue found, R0 is zero (Analog pin shorts to ground) please check your wiring and supply"); while(1);}

  /*****************************  MQ CAlibration ********************************************/

  MQ7.serialDebug(true);
```

```
// FIN calibraciones del MQ7


// calibraciones del MQ8
  MQ8.setRegressionMethod(1); //_PPM =  a*ratio^b
  MQ8.setA(976.97); MQ8.setB(-0.688); // Configure the equation to to calculate H2 concentration
  MQ8.init();


  Serial.print("Calibrating please wait.");
  float calcR0_MQ8 = 0;
  for(int i = 1; i<=10; i ++)
  {
    MQ8.update(); // Update data, the arduino will read the voltage from the analog pin
    calcR0_MQ8 += MQ8.calibrate(RatioMQ8CleanAir);
    Serial.print(".");
  }
  MQ8.setR0(calcR0_MQ8/10);
  Serial.println("  done!.");


  if(isinf(calcR0_MQ8)) {Serial.println("Warning: Conection issue, R0 is infinite (Open circuit detected) please check your wiring and supply"); while(1);}
  if(calcR0_MQ8 == 0){Serial.println("Warning: Conection issue found, R0 is zero (Analog pin shorts to ground) please check your wiring and supply"); while(1);}
  /*****************************  MQ CAlibration ********************************************/
  MQ8.serialDebug(true);


// FIN calibraciones del MQ8


}


void loop() {
  obtener_hora();
```

```
if(min_o_seg == 0){ //actualizacion en segundos
  if(segundo%updating_time==0){
    actualizar_datos = true;
  }else{
    actualizar_datos = false;
  }
}else{              //actualizacion en minutos
  if(minuto%updating_time==0){
    actualizar_datos = true;
  }else{
    actualizar_datos = false;
  }
}

if(actualizar_datos){

  humidity = dht.readHumidity();
  temperature = dht.readTemperature();

  // Check if any reads failed and exit early (to try again).
  if (isnan(humidity) || isnan(temperature)) {
    Serial.println(F("Failed to read from DHT sensor!"));
    return;
  }

  float rzero = mq135_sensor.getRZero();
  float correctedRZero = mq135_sensor.getCorrectedRZero(temperature, humidity);
  float resistance = mq135_sensor.getResistance();
  float ppm = mq135_sensor.getPPM();
  float correctedPPM = mq135_sensor.getCorrectedPPM(temperature, humidity);

  MQ2.update(); // Update data, the arduino will read the voltage from the analog pin
```

```
  MQ2.readSensor(); // Sensor will read PPM concentration using the model, a and b values set previously or from the setup

  MQ2_ppm = MQ2.get_ppm();


  MQ3.update(); // Update data, the arduino will read the voltage from the analog pin

  MQ3.readSensor(); // Sensor will read PPM concentration using the model, a and b values set previously or from the setup

  MQ3_ppm = MQ3.get_ppm();


  MQ4.update(); // Update data, the arduino will read the voltage from the analog pin

  MQ4.readSensor(); // Sensor will read PPM concentration using the model, a and b values set previously or from the setup

  MQ4_ppm = MQ4.get_ppm();


  MQ5.update(); // Update data, the arduino will read the voltage from the analog pin

  MQ5.readSensor(); // Sensor will read PPM concentration using the model, a and b values set previously or from the setup

  MQ5_ppm = MQ5.get_ppm();


  MQ6.update(); // Update data, the arduino will read the voltage from the analog pin

  MQ6.readSensor(); // Sensor will read PPM concentration using the model, a and b values set previously or from the setup

  MQ6_ppm = MQ6.get_ppm();


  MQ7.update(); // Update data, the arduino will read the voltage from the analog pin

  MQ7.readSensor(); // Sensor will read PPM concentration using the model, a and b values set previously or from the setup

  MQ7_ppm = MQ7.get_ppm();


  MQ8.update(); // Update data, the arduino will read the voltage from the analog pin

  MQ8.readSensor(); // Sensor will read PPM concentration using the model, a and b values set previously or from the setup

  MQ8_ppm = MQ8.get_ppm();


  data_update = hora_dia+";";
```

```
    data_update += (String)temperature+";";
    data_update += (String)humidity+";";
    data_update += (String)correctedPPM+";";
    data_update += (String)MQ2_ppm+";";
    data_update += (String)MQ3_ppm+";";
    data_update += (String)MQ4_ppm+";";
    data_update += (String)MQ5_ppm+";";
    data_update += (String)MQ6_ppm+";";
    data_update += (String)MQ7_ppm+";";
    data_update += (String)MQ8_ppm;

    escribir_en_SD(data_update);
  }
}




/* ------------------------------------------------------
 *  ---------------------- FUNCIONES----------------------
 *  ------------------------------------------------------
  */
void inicializar_SD(){
  Serial.print("INICIANDO TARJETA SD ...");    //MOSTRAMOS EL MENSAJE
  if (!SD.begin(7)) {                         //INICIAMOS LA SD EL PIN 7
    Serial.println("INICIACION FALLIDA");      //MOSTRAMOS EL MENSAJE EN CASO
QUE SD NO FUNCIONE CORRECTAMENTE
    display.clearDisplay();
    display.setCursor(0,0);
    display.println("INICIACION FALLIDA");
    display.display();
    return;
  }
```

```
  Serial.println("  INICIADO CORRECTAMENTE");  //SI LA SD SE HA INICIADO CORRECTAMENTE MOSTRAMOS EL MENSAJE
  SD_iniciada = true;
  Serial.println(" ");                    //DEJAMOS UN SALTO DE LINEA
  display.clearDisplay();
  display.setCursor(0,0);
  display.println("SD INICIADA CORRECTAMENTE");
  display.display();
}
void obtener_fecha(){
  DateTime now = RTC.now(); // Devuelve fecha de hoy YYYY-MM-DD y nombre de archivo del dia
  anyo = now.year();
  mes = now.month();
  dia = now.day();
  fecha_dia = (String)anyo+"-"+(String)mes+"-"+(String)dia;
  nombre_diario = (String)mes+"-"+(String)dia+".txt";
  if(dia_nuevo){
    file_existe(nombre_diario);
    Serial.println(nombre_diario);
    dia_nuevo = false;
  }
}
void obtener_hora(){
  DateTime now = RTC.now(); // Obtiene la fecha y hora del RTC
  boolean actualizar_hora = false;

  if(min_o_seg == 0){ //actualizacion en segundos
    if(ultimo_seg != now.second()){
      actualizar_hora = true;
    }else{
      actualizar_hora = false;
    }
```

```
  }else{              //actualizacion en minutos
    if(ultimo_min != now.minute()){
      actualizar_hora = true;
    }else{
      actualizar_hora = false;
    }
  }

  if(actualizar_hora){
    hora = now.hour();
    minuto = now.minute();
    segundo = now.second();
    ultimo_seg = segundo;
    ultimo_min = minuto;
    hora_dia = (String)hora+":"+(String)minuto+":"+(String)segundo;
    dato_ya_actualizado = false;
    if(hora == 0 && minuto == 0 && segundo == 0){
      dia_nuevo = true;
      obtener_fecha();
    }
  }else{
    dato_ya_actualizado = true;
  }
}
void file_existe(String nombre){ // EL NOMBRE DE UN ARCHIVO NO PUEDE TENER
MAS DE 8 CHARS
  myFile = SD.open(nombre);
  if (myFile) { // EL ARCHIVO EXISTE
    Serial.println("El archivo existe");
    myFile.close();
    display.println("El archivo existe");
    display.display();
  } else {
```

```
    myFile = SD.open(nombre, FILE_WRITE); //CREAAMOS UN ARCHIVO
    myFile.println(cabecera_archivo);     // ESCRIBIMOS EL TEXTO EN EL ARCHICO
    myFile.close();
    display.println("Archivo Creado");
    display.display();
  }
}
void escribir_en_SD(String texto){ // EL NOMBRE DE UN ARCHIVO NO PUEDE
TENER MAS DE 8 CHARS
  if(dato_ya_actualizado==false){
     //Serial.println(texto);
     Serial.println("Dato guardado: " + texto);
     myFile = SD.open(nombre_diario, FILE_WRITE);
     myFile.println(texto);              // ESCRIBIMOS EL TEXTO EN EL ARCHICO
     myFile.close();                          // CERREMOS EL ARCHIVO

     display.clearDisplay();
     display.setCursor(0,0);
     display.println(nombre_diario);
     display.println(texto);
     display.display();
  }
}
```

Sensor 2

```
#include <MQ135.h>
#include <DHT.h>
#include "MQUnifiedsensor.h"

#include <SPI.h>
#include <SD.h>

#include <Wire.h>
#include "RTClib.h"
RTC_DS1307 RTC;

#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>

/********************** Variables necesarias SD y reloj **************/
File myFile;
String fecha,fecha_dia,hora_dia;
int anyo,mes,dia,hora,minuto,segundo;
String nombre_diario;
String data_update;
boolean dato_ya_actualizado = false;
boolean dia_nuevo = false;
boolean SD_iniciada = false;
boolean actualizar_datos = false;
String cabecera_archivo =
"Timestamp;Temperatura;Humedad;MQ135(CO2);MQ2(LPG);MQ3(LPG);MQ4(LPG);M
Q5(LPG);MQ6(LPG);MQ7(LPG);MQ8(LPG)";
int ultimo_seg=0;
int ultimo_min=1;
/*  Tiempo entre actualizaciones en tarjeta SD
    variable min_o_seg debe tener dos posibles valores:
        min_o_seg = 0 --> actualizacion en segundos
```

```
      min_o_seg = 1 --> actualizacion en minutos
   La variable updating_time hará referencia a minutos o segundos depeniendo del valor
   de la variable min_o_seg. Los valores permitidos son: 1, 2, 3, 4, 5, 6, 10, 12, 15, 20, 30 y 60.
*/
int min_o_seg = 1;
int updating_time = 1;

/**********************LCD**************/
#define ANCHO 128
#define ALTO 32

#define OLED_RESET 4
Adafruit_SSD1306 display(ANCHO,ALTO,&Wire,OLED_RESET);

/**********************placa usada**************/
#define       Board                 ("Arduino MEGA")
#define       Voltage_Resolution    (5)
#define       ADC_Bit_Resolution    (10) // For arduino UNO/MEGA/NANO

/************************CALIBRACION MQ2*********************************/
#define       Pin_MQ2               (A1)  //Analog input 1 of your arduino
/************************Software Related Macros************************************/
#define       Type_MQ2              ("MQ-2") //MQ2
#define       RatioMQ2CleanAir      (9.83) //RS / R0 = 9.83 ppm
MQUnifiedsensor MQ2(Board, Voltage_Resolution, ADC_Bit_Resolution, Pin_MQ2, Type_MQ2);
/*****************************Globals***********************************************/


/************************CALIBRACION MQ3*********************************/
#define       Pin_MQ3               (A2)  //Analog input 2 of your arduino
/************************Software Related Macros************************************/
```

```
#define         Type_MQ3                ("MQ-3") //MQ3
#define         RatioMQ3CleanAir        (60) //RS / R0 = 60 ppm
MQUnifiedsensor MQ3(Board, Voltage_Resolution, ADC_Bit_Resolution, Pin_MQ3, Type_MQ3);
/*****************************Globals***********************************************/

/*************************CALIBRACION MQ4*********************************/
#define         Pin_MQ4                 (A3)  //Analog input 3 of your arduino
/************************Software Related Macros************************************/
#define         Type_MQ4                ("MQ-4") //MQ4
#define         RatioMQ4CleanAir        (4.4) //RS / R0 = 60 ppm
MQUnifiedsensor MQ4(Board, Voltage_Resolution, ADC_Bit_Resolution, Pin_MQ4, Type_MQ4);

/*************************CALIBRACION MQ5*********************************/
#define         Pin_MQ5                 (A4)  //Analog input 4 of your arduino
/************************Software Related Macros************************************/
#define         Type_MQ5                ("MQ-5") //MQ5
#define         RatioMQ5CleanAir        (6.5) //RS / R0 = 60 ppm
MQUnifiedsensor MQ5(Board, Voltage_Resolution, ADC_Bit_Resolution, Pin_MQ5, Type_MQ5);

/*************************CALIBRACION MQ6*********************************/
#define         Pin_MQ6                 (A5)  //Analog input 5 of your arduino
/************************Software Related Macros************************************/
#define         Type_MQ6                ("MQ-6") //MQ6
#define         RatioMQ6CleanAir        (10) //RS / R0 = 60 ppm
MQUnifiedsensor MQ6(Board, Voltage_Resolution, ADC_Bit_Resolution, Pin_MQ6, Type_MQ6);

/*************************CALIBRACION MQ7*********************************/
#define         Pin_MQ7                 (A6)  //Analog input 6 of your arduino
/************************Software Related Macros************************************/
#define         Type_MQ7                ("MQ-7") //MQ7
```

```
#define         RatioMQ7CleanAir        (27.5) //RS / R0 = 60 ppm
MQUnifiedsensor MQ7(Board, Voltage_Resolution, ADC_Bit_Resolution, Pin_MQ7, Type_MQ7);

/************************CALIBRACION MQ8***********************************/
#define         Pin_MQ8                 (A7)  //Analog input 7 of your arduino
/************************Software Related Macros***********************************/
#define         Type_MQ8                ("MQ-8") //MQ8
#define         RatioMQ8CleanAir        (70) //RS / R0 = 60 ppm
MQUnifiedsensor MQ8(Board, Voltage_Resolution, ADC_Bit_Resolution, Pin_MQ8, Type_MQ8);

#define PIN_MQ135 A0 // MQ135 Analog Input Pin
#define DHTPIN 2 // DHT Digital Input Pin
#define DHTTYPE DHT11 // DHT11 or DHT22, depends on your sensor

MQ135 mq135_sensor(PIN_MQ135);
DHT dht(DHTPIN, DHTTYPE);
float temperature, humidity; // Temp and Humid floats, will be measured by the DHT
float MQ2_ppm,MQ3_ppm,MQ4_ppm,MQ5_ppm,MQ6_ppm,MQ7_ppm,MQ8_ppm;

void setup() {
  Serial.begin(9600);
  display.begin(SSD1306_SWITCHCAPVCC, 0x3C);
  display.clearDisplay();
  display.setTextSize(1);
  display.setTextColor(WHITE,BLACK);
  display.setCursor(0,0);

  while(!SD_iniciada){
    inicializar_SD();
  }
```

```
  dht.begin();


  Wire.begin();
  RTC.begin();
  RTC.adjust(DateTime(__DATE__, __TIME__)); // Establece la fecha y hora
(Comentar una vez establecida la hora)


  display.begin(SSD1306_SWITCHCAPVCC, 0x3C);
  display.clearDisplay();
  display.setTextSize(1);
  display.setTextColor(WHITE,BLACK);
  display.setCursor(0,0);


  obtener_fecha();
  ultimo_seg = segundo;
  file_existe(nombre_diario);


// calibraciones del MQ2
  MQ2.setRegressionMethod(1); //_PPM =  a*ratio^b
  MQ2.setA(574.25); MQ2.setB(-2.222); // Configure the equation to to calculate LPG
concentration
  MQ2.init();


  Serial.print("Calibrating MQ2, please wait.");
  float calcR0_MQ2 = 0;
  for(int i = 1; i<=10; i ++)
  {
    MQ2.update(); // Update data, the arduino will read the voltage from the analog pin
    calcR0_MQ2 += MQ2.calibrate(RatioMQ2CleanAir);
    Serial.print(".");
  }
  MQ2.setR0(calcR0_MQ2/10);
  Serial.println("  done!.");
```

```
  if(isinf(calcR0_MQ2)) {Serial.println("Warning: Conection issue, R0 is infinite (Open circuit detected) please check your wiring and supply"); while(1);}
  if(calcR0_MQ2 == 0){Serial.println("Warning: Conection issue found, R0 is zero (Analog pin shorts to ground) please check your wiring and supply"); while(1);}
  /*****************************  MQ CAlibration ********************************************/

  MQ2.serialDebug(true);

// FIN calibraciones del MQ2

// calibraciones del MQ3
  MQ3.setRegressionMethod(1); //_PPM =  a*ratio^b
  MQ3.setA(44771); MQ3.setB(-3.245); // Configure the equation to to calculate Benzene concentration
  MQ3.init();

  Serial.print("Calibrating MQ3, please wait.");
  float calcR0_MQ3 = 0;
  for(int i = 1; i<=10; i ++)
  {
    MQ3.update(); // Update data, the arduino will read the voltage from the analog pin
    calcR0_MQ3 += MQ3.calibrate(RatioMQ3CleanAir);
    Serial.print(".");
  }
  MQ3.setR0(calcR0_MQ3/10);
  Serial.println("  done!.");

  if(isinf(calcR0_MQ3)) {Serial.println("Warning: Conection issue, R0 is infinite (Open circuit detected) please check your wiring and supply"); while(1);}
  if(calcR0_MQ3 == 0){Serial.println("Warning: Conection issue found, R0 is zero (Analog pin shorts to ground) please check your wiring and supply"); while(1);}
  /*****************************  MQ CAlibration ********************************************/
  MQ3.serialDebug(true);
```

```
// FIN calibraciones del MQ3


// calibraciones del MQ4
  MQ4.setRegressionMethod(1); //_PPM =  a*ratio^b
  MQ4.setA(3811.9); MQ4.setB(-3.113); // Configure the equation to to calculate CH4 concentration
  MQ4.init();


Serial.print("Calibrating please wait.");
  float calcR0_MQ4 = 0;
  for(int i = 1; i<=10; i ++)
  {
    MQ4.update(); // Update data, the arduino will read the voltage from the analog pin
    calcR0_MQ4 += MQ4.calibrate(RatioMQ4CleanAir);
    Serial.print(".");
  }
  MQ4.setR0(calcR0_MQ4/10);
  Serial.println("  done!.");


  if(isinf(calcR0_MQ4)) {Serial.println("Warning: Conection issue, R0 is infinite (Open circuit detected) please check your wiring and supply"); while(1);}
  if(calcR0_MQ4 == 0){Serial.println("Warning: Conection issue found, R0 is zero (Analog pin shorts to ground) please check your wiring and supply"); while(1);}
  /*****************************  MQ CAlibration ********************************************/
  MQ4.serialDebug(true);
// FIN calibraciones del MQ4


// calibraciones del MQ5
  MQ5.setRegressionMethod(1); //_PPM =  a*ratio^b
  MQ5.setA(80.897); MQ5.setB(-2.431); // Configure the equation to to calculate H2 concentration
  MQ5.init();


  Serial.print("Calibrating please wait.");
```

```
  float calcR0_MQ5 = 0;
  for(int i = 1; i<=10; i ++)
  {
    MQ5.update(); // Update data, the arduino will read the voltage from the analog pin
    calcR0_MQ5 += MQ5.calibrate(RatioMQ5CleanAir);
    Serial.print(".");
  }
  MQ5.setR0(calcR0_MQ5/10);
  Serial.println("  done!.");

  if(isinf(calcR0_MQ5)) {Serial.println("Warning: Conection issue, R0 is infinite (Open circuit detected) please check your wiring and supply"); while(1);}
  if(calcR0_MQ5 == 0){Serial.println("Warning: Conection issue found, R0 is zero (Analog pin shorts to ground) please check your wiring and supply"); while(1);}
  /*****************************  MQ CAlibration ********************************************/
  MQ5.serialDebug(true);

// FIN calibraciones del MQ5

// calibraciones del MQ6
  MQ6.setRegressionMethod(1); //_PPM =  a*ratio^b
  MQ6.setA(1009.2); MQ6.setB(-2.35); // Configure the equation to to calculate CH4 concentration
  MQ6.init();

  Serial.print("Calibrating please wait.");
  float calcR0_MQ6 = 0;
  for(int i = 1; i<=10; i ++)
  {
    MQ6.update(); // Update data, the arduino will read the voltage from the analog pin
    calcR0_MQ6 += MQ6.calibrate(RatioMQ6CleanAir);
    Serial.print(".");
  }
  MQ6.setR0(calcR0_MQ6/10);
```

```
  Serial.println("  done!.");


  if(isinf(calcR0_MQ6)) {Serial.println("Warning: Conection issue, R0 is infinite (Open circuit detected) please check your wiring and supply"); while(1);}

  if(calcR0_MQ6 == 0){Serial.println("Warning: Conection issue found, R0 is zero (Analog pin shorts to ground) please check your wiring and supply"); while(1);}

  /*****************************  MQ CAlibration ********************************************/

  MQ6.serialDebug(true);


// FIN calibraciones del MQ6


// calibraciones del MQ7

  MQ7.setRegressionMethod(1); //_PPM =  a*ratio^b

  MQ7.setA(700000000); MQ7.setB(-7.703); // Configure the equation to calculate CO concentration value

  MQ7.init();


  Serial.print("Calibrating please wait.");

  float calcR0_MQ7 = 0;

  for(int i = 1; i<=10; i ++)

  {

    MQ7.update(); // Update data, the arduino will read the voltage from the analog pin

    calcR0_MQ7 += MQ7.calibrate(RatioMQ7CleanAir);

    Serial.print(".");

  }

  MQ7.setR0(calcR0_MQ7/10);

  Serial.println("  done!.");


  if(isinf(calcR0_MQ7)) {Serial.println("Warning: Conection issue, R0 is infinite (Open circuit detected) please check your wiring and supply"); while(1);}

  if(calcR0_MQ7 == 0){Serial.println("Warning: Conection issue found, R0 is zero (Analog pin shorts to ground) please check your wiring and supply"); while(1);}

  /*****************************  MQ CAlibration ********************************************/

  MQ7.serialDebug(true);
```

```
// FIN calibraciones del MQ7


// calibraciones del MQ8
  MQ8.setRegressionMethod(1); //_PPM =  a*ratio^b
  MQ8.setA(10000000); MQ8.setB(-3.123); // Configure the equation to to calculate H2 concentration
  MQ8.init();


  Serial.print("Calibrating please wait.");
  float calcR0_MQ8 = 0;
  for(int i = 1; i<=10; i ++)
  {
    MQ8.update(); // Update data, the arduino will read the voltage from the analog pin
    calcR0_MQ8 += MQ8.calibrate(RatioMQ8CleanAir);
    Serial.print(".");
  }
  MQ8.setR0(calcR0_MQ8/10);
  Serial.println("  done!.");


  if(isinf(calcR0_MQ8)) {Serial.println("Warning: Conection issue, R0 is infinite (Open circuit detected) please check your wiring and supply"); while(1);}
  if(calcR0_MQ8 == 0){Serial.println("Warning: Conection issue found, R0 is zero (Analog pin shorts to ground) please check your wiring and supply"); while(1);}
  /*****************************  MQ CAlibration ********************************************/
  MQ8.serialDebug(true);


// FIN calibraciones del MQ8


}


void loop() {
  obtener_hora();
```

```
if(min_o_seg == 0){ //actualizacion en segundos
  if(segundo%updating_time==0){
    actualizar_datos = true;
  }else{
    actualizar_datos = false;
  }
}else{              //actualizacion en minutos
  if(minuto%updating_time==0){
    actualizar_datos = true;
  }else{
    actualizar_datos = false;
  }
}

if(actualizar_datos){

  humidity = dht.readHumidity();
  temperature = dht.readTemperature();

  // Check if any reads failed and exit early (to try again).
  if (isnan(humidity) || isnan(temperature)) {
    Serial.println(F("Failed to read from DHT sensor!"));
    return;
  }

  float rzero = mq135_sensor.getRZero();
  float correctedRZero = mq135_sensor.getCorrectedRZero(temperature, humidity);
  float resistance = mq135_sensor.getResistance();
  float ppm = mq135_sensor.getPPM();
  float correctedPPM = mq135_sensor.getCorrectedPPM(temperature, humidity);

  MQ2.update(); // Update data, the arduino will read the voltage from the analog pin
```

```
  MQ2.readSensor(); // Sensor will read PPM concentration using the model, a and b values set previously or from the setup

  MQ2_ppm = MQ2.get_ppm();


  MQ3.update(); // Update data, the arduino will read the voltage from the analog pin

  MQ3.readSensor(); // Sensor will read PPM concentration using the model, a and b values set previously or from the setup

  MQ3_ppm = MQ3.get_ppm();


  MQ4.update(); // Update data, the arduino will read the voltage from the analog pin

  MQ4.readSensor(); // Sensor will read PPM concentration using the model, a and b values set previously or from the setup

  MQ4_ppm = MQ4.get_ppm();


  MQ5.update(); // Update data, the arduino will read the voltage from the analog pin

  MQ5.readSensor(); // Sensor will read PPM concentration using the model, a and b values set previously or from the setup

  MQ5_ppm = MQ5.get_ppm();


  MQ6.update(); // Update data, the arduino will read the voltage from the analog pin

  MQ6.readSensor(); // Sensor will read PPM concentration using the model, a and b values set previously or from the setup

  MQ6_ppm = MQ6.get_ppm();


  MQ7.update(); // Update data, the arduino will read the voltage from the analog pin

  MQ7.readSensor(); // Sensor will read PPM concentration using the model, a and b values set previously or from the setup

  MQ7_ppm = MQ7.get_ppm();


  MQ8.update(); // Update data, the arduino will read the voltage from the analog pin

  MQ8.readSensor(); // Sensor will read PPM concentration using the model, a and b values set previously or from the setup

  MQ8_ppm = MQ8.get_ppm();


  data_update = hora_dia+";";
```

```
    data_update += (String)temperature+";";
    data_update += (String)humidity+";";
    data_update += (String)correctedPPM+";";
    data_update += (String)MQ2_ppm+";";
    data_update += (String)MQ3_ppm+";";
    data_update += (String)MQ4_ppm+";";
    data_update += (String)MQ5_ppm+";";
    data_update += (String)MQ6_ppm+";";
    data_update += (String)MQ7_ppm+";";
    data_update += (String)MQ8_ppm;

    escribir_en_SD(data_update);
  }
}




/* ------------------------------------------------------
 *  ---------------------- FUNCIONES----------------------
 *  ------------------------------------------------------
  */
void inicializar_SD(){
  Serial.print("INICIANDO TARJETA SD ...");    //MOSTRAMOS EL MENSAJE
  if (!SD.begin(7)) {                         //INICIAMOS LA SD EL PIN 7
    Serial.println("INICIACION FALLIDA");      //MOSTRAMOS EL MENSAJE EN CASO
QUE SD NO FUNCIONE CORRECTAMENTE
    display.clearDisplay();
    display.setCursor(0,0);
    display.println("INICIACION FALLIDA");
    display.display();
    return;
  }
```

```
  Serial.println("  INICIADO CORRECTAMENTE");  //SI LA SD SE HA INICIADO CORRECTAMENTE MOSTRAMOS EL MENSAJE
  SD_iniciada = true;
  Serial.println(" ");                    //DEJAMOS UN SALTO DE LINEA
  display.clearDisplay();
  display.setCursor(0,0);
  display.println("SD INICIADA CORRECTAMENTE");
  display.display();
}
void obtener_fecha(){
  DateTime now = RTC.now(); // Devuelve fecha de hoy YYYY-MM-DD y nombre de archivo del dia
  anyo = now.year();
  mes = now.month();
  dia = now.day();
  fecha_dia = (String)anyo+"-"+(String)mes+"-"+(String)dia;
  nombre_diario = (String)mes+"-"+(String)dia+".txt";
  if(dia_nuevo){
    file_existe(nombre_diario);
    Serial.println(nombre_diario);
    dia_nuevo = false;
  }
}
void obtener_hora(){
  DateTime now = RTC.now(); // Obtiene la fecha y hora del RTC
  boolean actualizar_hora = false;

  if(min_o_seg == 0){ //actualizacion en segundos
    if(ultimo_seg != now.second()){
      actualizar_hora = true;
    }else{
      actualizar_hora = false;
    }
```

```
  }else{              //actualizacion en minutos
    if(ultimo_min != now.minute()){
      actualizar_hora = true;
    }else{
      actualizar_hora = false;
    }
  }

  if(actualizar_hora){
    hora = now.hour();
    minuto = now.minute();
    segundo = now.second();
    ultimo_seg = segundo;
    ultimo_min = minuto;
    hora_dia = (String)hora+":"+(String)minuto+":"+(String)segundo;
    dato_ya_actualizado = false;
    if(hora == 0 && minuto == 0 && segundo == 0){
      dia_nuevo = true;
      obtener_fecha();
    }
  }else{
    dato_ya_actualizado = true;
  }
}
void file_existe(String nombre){ // EL NOMBRE DE UN ARCHIVO NO PUEDE TENER
MAS DE 8 CHARS
  myFile = SD.open(nombre);
  if (myFile) { // EL ARCHIVO EXISTE
    Serial.println("El archivo existe");
    myFile.close();
    display.println("El archivo existe");
    display.display();
  } else {
```

```
    myFile = SD.open(nombre, FILE_WRITE); //CREAAMOS UN ARCHIVO
    myFile.println(cabecera_archivo);     // ESCRIBIMOS EL TEXTO EN EL ARCHICO
    myFile.close();
    display.println("Archivo Creado");
    display.display();
  }
}
void escribir_en_SD(String texto){ // EL NOMBRE DE UN ARCHIVO NO PUEDE
TENER MAS DE 8 CHARS
  if(dato_ya_actualizado==false){
     //Serial.println(texto);
     Serial.println("Dato guardado: " + texto);
     myFile = SD.open(nombre_diario, FILE_WRITE);
     myFile.println(texto);              // ESCRIBIMOS EL TEXTO EN EL ARCHICO
     myFile.close();                          // CERREMOS EL ARCHIVO

     display.clearDisplay();
     display.setCursor(0,0);
     display.println(nombre_diario);
     display.println(texto);
     display.display();
  }
}
```

Sensor 3

```
#include <MQ135.h>
#include <DHT.h>
#include "MQUnifiedsensor.h"

#include <SPI.h>
#include <SD.h>

#include <Wire.h>
#include "RTClib.h"
RTC_DS1307 RTC;

#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>

/********************** Variables necesarias SD y reloj **************/
File myFile;
String fecha,fecha_dia,hora_dia;
int anyo,mes,dia,hora,minuto,segundo;
String nombre_diario;
String data_update;
boolean dato_ya_actualizado = false;
boolean dia_nuevo = false;
boolean SD_iniciada = false;
boolean actualizar_datos = false;
String cabecera_archivo =
"Timestamp;Temperatura;Humedad;MQ135(CO2);MQ2(Propano);MQ3(Hexano);MQ4(
CH4);MQ5(CH4);MQ6(CH4);MQ7(CH4);MQ8(CH4)";
int ultimo_seg=0;
int ultimo_min=1;
/*  Tiempo entre actualizaciones en tarjeta SD
    variable min_o_seg debe tener dos posibles valores:
        min_o_seg = 0 --> actualizacion en segundos
```

```
      min_o_seg = 1 --> actualizacion en minutos
   La variable updating_time hará referencia a minutos o segundos depeniendo del valor
   de la variable min_o_seg. Los valores permitidos son: 1, 2, 3, 4, 5, 6, 10, 12, 15, 20, 30 y 60.
*/
int min_o_seg = 1;
int updating_time = 1;

/**********************LCD**************/
#define ANCHO 128
#define ALTO 32

#define OLED_RESET 4
Adafruit_SSD1306 display(ANCHO,ALTO,&Wire,OLED_RESET);

/**********************placa usada**************/
#define       Board               ("Arduino MEGA")
#define       Voltage_Resolution     (5)
#define       ADC_Bit_Resolution     (10) // For arduino UNO/MEGA/NANO

/************************CALIBRACION MQ2*********************************/
#define       Pin_MQ2                 (A1)  //Analog input 1 of your arduino
/************************Software Related Macros************************************/
#define       Type_MQ2                ("MQ-2") //MQ2
#define       RatioMQ2CleanAir       (9.83) //RS / R0 = 9.83 ppm
MQUnifiedsensor MQ2(Board, Voltage_Resolution, ADC_Bit_Resolution, Pin_MQ2, Type_MQ2);
/*****************************Globals***********************************************/



/************************CALIBRACION MQ3*********************************/
#define       Pin_MQ3                 (A2)  //Analog input 2 of your arduino
/************************Software Related Macros************************************/
```

```
#define         Type_MQ3                ("MQ-3") //MQ3

#define         RatioMQ3CleanAir        (60) //RS / R0 = 60 ppm

MQUnifiedsensor MQ3(Board, Voltage_Resolution, ADC_Bit_Resolution, Pin_MQ3,
Type_MQ3);

/*****************************Globals***********************************************/


/*************************CALIBRACION MQ4*********************************/

#define         Pin_MQ4                 (A3)  //Analog input 3 of your arduino

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

#define         Type_MQ4                ("MQ-4") //MQ4

#define         RatioMQ4CleanAir        (4.4) //RS / R0 = 60 ppm

MQUnifiedsensor MQ4(Board, Voltage_Resolution, ADC_Bit_Resolution, Pin_MQ4,
Type_MQ4);


/*************************CALIBRACION MQ5*********************************/

#define         Pin_MQ5                 (A4)  //Analog input 4 of your arduino

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

#define         Type_MQ5                ("MQ-5") //MQ5

#define         RatioMQ5CleanAir        (6.5) //RS / R0 = 60 ppm

MQUnifiedsensor MQ5(Board, Voltage_Resolution, ADC_Bit_Resolution, Pin_MQ5,
Type_MQ5);


/*************************CALIBRACION MQ6*********************************/

#define         Pin_MQ6                 (A5)  //Analog input 5 of your arduino

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

#define         Type_MQ6                ("MQ-6") //MQ6

#define         RatioMQ6CleanAir        (10) //RS / R0 = 60 ppm

MQUnifiedsensor MQ6(Board, Voltage_Resolution, ADC_Bit_Resolution, Pin_MQ6,
Type_MQ6);


/*************************CALIBRACION MQ7*********************************/

#define         Pin_MQ7                 (A6)  //Analog input 6 of your arduino

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

#define         Type_MQ7                ("MQ-7") //MQ7
```

```
#define         RatioMQ7CleanAir        (27.5) //RS / R0 = 60 ppm
MQUnifiedsensor MQ7(Board, Voltage_Resolution, ADC_Bit_Resolution, Pin_MQ7, Type_MQ7);

/************************CALIBRACION MQ8***********************************/
#define         Pin_MQ8                 (A7)  //Analog input 7 of your arduino
/************************Software Related Macros***********************************/
#define         Type_MQ8                ("MQ-8") //MQ8
#define         RatioMQ8CleanAir        (70) //RS / R0 = 60 ppm
MQUnifiedsensor MQ8(Board, Voltage_Resolution, ADC_Bit_Resolution, Pin_MQ8, Type_MQ8);

#define PIN_MQ135 A0 // MQ135 Analog Input Pin
#define DHTPIN 2 // DHT Digital Input Pin
#define DHTTYPE DHT11 // DHT11 or DHT22, depends on your sensor

MQ135 mq135_sensor(PIN_MQ135);
DHT dht(DHTPIN, DHTTYPE);
float temperature, humidity; // Temp and Humid floats, will be measured by the DHT
float MQ2_ppm,MQ3_ppm,MQ4_ppm,MQ5_ppm,MQ6_ppm,MQ7_ppm,MQ8_ppm;

void setup() {
  Serial.begin(9600);
  display.begin(SSD1306_SWITCHCAPVCC, 0x3C);
  display.clearDisplay();
  display.setTextSize(1);
  display.setTextColor(WHITE,BLACK);
  display.setCursor(0,0);

  while(!SD_iniciada){
    inicializar_SD();
  }
```

```
  dht.begin();


  Wire.begin();
  RTC.begin();
  RTC.adjust(DateTime(__DATE__, __TIME__)); // Establece la fecha y hora
(Comentar una vez establecida la hora)


  display.begin(SSD1306_SWITCHCAPVCC, 0x3C);
  display.clearDisplay();
  display.setTextSize(1);
  display.setTextColor(WHITE,BLACK);
  display.setCursor(0,0);


  obtener_fecha();
  ultimo_seg = segundo;
  file_existe(nombre_diario);


// calibraciones del MQ2
  MQ2.setRegressionMethod(1); //_PPM =  a*ratio^b
  MQ2.setA(658.71); MQ2.setB(-2.168); // Configure the equation to to calculate LPG
concentration
  MQ2.init();


  Serial.print("Calibrating MQ2, please wait.");
  float calcR0_MQ2 = 0;
  for(int i = 1; i<=10; i ++)
  {
    MQ2.update(); // Update data, the arduino will read the voltage from the analog pin
    calcR0_MQ2 += MQ2.calibrate(RatioMQ2CleanAir);
    Serial.print(".");
  }
  MQ2.setR0(calcR0_MQ2/10);
  Serial.println("  done!.");
```

```
  if(isinf(calcR0_MQ2)) {Serial.println("Warning: Conection issue, R0 is infinite (Open circuit detected) please check your wiring and supply"); while(1);}
  if(calcR0_MQ2 == 0){Serial.println("Warning: Conection issue found, R0 is zero (Analog pin shorts to ground) please check your wiring and supply"); while(1);}
  /*****************************  MQ CAlibration ********************************************/

  MQ2.serialDebug(true);

// FIN calibraciones del MQ2

// calibraciones del MQ3
  MQ3.setRegressionMethod(1); //_PPM =  a*ratio^b
  MQ3.setA(7585.3); MQ3.setB(-2.849); // Configure the equation to to calculate Benzene concentration
  MQ3.init();

  Serial.print("Calibrating MQ3, please wait.");
  float calcR0_MQ3 = 0;
  for(int i = 1; i<=10; i ++)
  {
    MQ3.update(); // Update data, the arduino will read the voltage from the analog pin
    calcR0_MQ3 += MQ3.calibrate(RatioMQ3CleanAir);
    Serial.print(".");
  }
  MQ3.setR0(calcR0_MQ3/10);
  Serial.println("  done!.");

  if(isinf(calcR0_MQ3)) {Serial.println("Warning: Conection issue, R0 is infinite (Open circuit detected) please check your wiring and supply"); while(1);}
  if(calcR0_MQ3 == 0){Serial.println("Warning: Conection issue found, R0 is zero (Analog pin shorts to ground) please check your wiring and supply"); while(1);}
  /*****************************  MQ CAlibration ********************************************/
  MQ3.serialDebug(true);
```

```
// FIN calibraciones del MQ3


// calibraciones del MQ4
  MQ4.setRegressionMethod(1); //_PPM =  a*ratio^b
  MQ4.setA(1012.7); MQ4.setB(-2.786); // Configure the equation to to calculate CH4 concentration
  MQ4.init();


Serial.print("Calibrating please wait.");
  float calcR0_MQ4 = 0;
  for(int i = 1; i<=10; i ++)
  {
    MQ4.update(); // Update data, the arduino will read the voltage from the analog pin
    calcR0_MQ4 += MQ4.calibrate(RatioMQ4CleanAir);
    Serial.print(".");
  }
  MQ4.setR0(calcR0_MQ4/10);
  Serial.println("  done!.");


  if(isinf(calcR0_MQ4)) {Serial.println("Warning: Conection issue, R0 is infinite (Open circuit detected) please check your wiring and supply"); while(1);}
  if(calcR0_MQ4 == 0){Serial.println("Warning: Conection issue found, R0 is zero (Analog pin shorts to ground) please check your wiring and supply"); while(1);}
  /*****************************  MQ CAlibration ********************************************/
  MQ4.serialDebug(true);
// FIN calibraciones del MQ4


// calibraciones del MQ5
  MQ5.setRegressionMethod(1); //_PPM =  a*ratio^b
  MQ5.setA(177.65); MQ5.setB(-2.56); // Configure the equation to to calculate H2 concentration
  MQ5.init();


  Serial.print("Calibrating please wait.");
```

```
  float calcR0_MQ5 = 0;
  for(int i = 1; i<=10; i ++)
  {
    MQ5.update(); // Update data, the arduino will read the voltage from the analog pin
    calcR0_MQ5 += MQ5.calibrate(RatioMQ5CleanAir);
    Serial.print(".");
  }
  MQ5.setR0(calcR0_MQ5/10);
  Serial.println("  done!.");

  if(isinf(calcR0_MQ5)) {Serial.println("Warning: Conection issue, R0 is infinite (Open circuit detected) please check your wiring and supply"); while(1);}
  if(calcR0_MQ5 == 0){Serial.println("Warning: Conection issue found, R0 is zero (Analog pin shorts to ground) please check your wiring and supply"); while(1);}
  /*****************************  MQ CAlibration ********************************************/
  MQ5.serialDebug(true);

// FIN calibraciones del MQ5

// calibraciones del MQ6
  MQ6.setRegressionMethod(1); //_PPM =  a*ratio^b
  MQ6.setA(2127.2); MQ6.setB(-2.526); // Configure the equation to to calculate CH4 concentration
  MQ6.init();

  Serial.print("Calibrating please wait.");
  float calcR0_MQ6 = 0;
  for(int i = 1; i<=10; i ++)
  {
    MQ6.update(); // Update data, the arduino will read the voltage from the analog pin
    calcR0_MQ6 += MQ6.calibrate(RatioMQ6CleanAir);
    Serial.print(".");
  }
  MQ6.setR0(calcR0_MQ6/10);
```

```
  Serial.println("  done!.");


  if(isinf(calcR0_MQ6)) {Serial.println("Warning: Conection issue, R0 is infinite (Open circuit detected) please check your wiring and supply"); while(1);}

  if(calcR0_MQ6 == 0){Serial.println("Warning: Conection issue found, R0 is zero (Analog pin shorts to ground) please check your wiring and supply"); while(1);}

  /*****************************  MQ CAlibration ********************************************/

  MQ6.serialDebug(true);


// FIN calibraciones del MQ6


// calibraciones del MQ7

  MQ7.setRegressionMethod(1); //_PPM =  a*ratio^b

  MQ7.setA(60000000000000); MQ7.setB(-10.54); // Configure the equation to calculate CO concentration value

  MQ7.init();


  Serial.print("Calibrating please wait.");

  float calcR0_MQ7 = 0;

  for(int i = 1; i<=10; i ++)

  {

    MQ7.update(); // Update data, the arduino will read the voltage from the analog pin

    calcR0_MQ7 += MQ7.calibrate(RatioMQ7CleanAir);

    Serial.print(".");

  }

  MQ7.setR0(calcR0_MQ7/10);

  Serial.println("  done!.");


  if(isinf(calcR0_MQ7)) {Serial.println("Warning: Conection issue, R0 is infinite (Open circuit detected) please check your wiring and supply"); while(1);}

  if(calcR0_MQ7 == 0){Serial.println("Warning: Conection issue found, R0 is zero (Analog pin shorts to ground) please check your wiring and supply"); while(1);}

  /*****************************  MQ CAlibration ********************************************/

  MQ7.serialDebug(true);
```

```
// FIN calibraciones del MQ7


// calibraciones del MQ8
  MQ8.setRegressionMethod(1); //_PPM =  a*ratio^b
  MQ8.setA(80000000000000); MQ8.setB(-6.666); // Configure the equation to to calculate H2 concentration
  MQ8.init();


  Serial.print("Calibrating please wait.");
  float calcR0_MQ8 = 0;
  for(int i = 1; i<=10; i ++)
  {
    MQ8.update(); // Update data, the arduino will read the voltage from the analog pin
    calcR0_MQ8 += MQ8.calibrate(RatioMQ8CleanAir);
    Serial.print(".");
  }
  MQ8.setR0(calcR0_MQ8/10);
  Serial.println("  done!.");


  if(isinf(calcR0_MQ8)) {Serial.println("Warning: Conection issue, R0 is infinite (Open circuit detected) please check your wiring and supply"); while(1);}
  if(calcR0_MQ8 == 0){Serial.println("Warning: Conection issue found, R0 is zero (Analog pin shorts to ground) please check your wiring and supply"); while(1);}
  /*****************************  MQ CAlibration ********************************************/
  MQ8.serialDebug(true);


// FIN calibraciones del MQ8


}


void loop() {
  obtener_hora();
```

```
if(min_o_seg == 0){ //actualizacion en segundos
  if(segundo%updating_time==0){
    actualizar_datos = true;
  }else{
    actualizar_datos = false;
  }
}else{              //actualizacion en minutos
  if(minuto%updating_time==0){
    actualizar_datos = true;
  }else{
    actualizar_datos = false;
  }
}

if(actualizar_datos){

  humidity = dht.readHumidity();
  temperature = dht.readTemperature();

  // Check if any reads failed and exit early (to try again).
  if (isnan(humidity) || isnan(temperature)) {
    Serial.println(F("Failed to read from DHT sensor!"));
    return;
  }

  float rzero = mq135_sensor.getRZero();
  float correctedRZero = mq135_sensor.getCorrectedRZero(temperature, humidity);
  float resistance = mq135_sensor.getResistance();
  float ppm = mq135_sensor.getPPM();
  float correctedPPM = mq135_sensor.getCorrectedPPM(temperature, humidity);

  MQ2.update(); // Update data, the arduino will read the voltage from the analog pin
```

```
  MQ2.readSensor(); // Sensor will read PPM concentration using the model, a and b values set previously or from the setup

  MQ2_ppm = MQ2.get_ppm();


  MQ3.update(); // Update data, the arduino will read the voltage from the analog pin

  MQ3.readSensor(); // Sensor will read PPM concentration using the model, a and b values set previously or from the setup

  MQ3_ppm = MQ3.get_ppm();


  MQ4.update(); // Update data, the arduino will read the voltage from the analog pin

  MQ4.readSensor(); // Sensor will read PPM concentration using the model, a and b values set previously or from the setup

  MQ4_ppm = MQ4.get_ppm();


  MQ5.update(); // Update data, the arduino will read the voltage from the analog pin

  MQ5.readSensor(); // Sensor will read PPM concentration using the model, a and b values set previously or from the setup

  MQ5_ppm = MQ5.get_ppm();


  MQ6.update(); // Update data, the arduino will read the voltage from the analog pin

  MQ6.readSensor(); // Sensor will read PPM concentration using the model, a and b values set previously or from the setup

  MQ6_ppm = MQ6.get_ppm();


  MQ7.update(); // Update data, the arduino will read the voltage from the analog pin

  MQ7.readSensor(); // Sensor will read PPM concentration using the model, a and b values set previously or from the setup

  MQ7_ppm = MQ7.get_ppm();


  MQ8.update(); // Update data, the arduino will read the voltage from the analog pin

  MQ8.readSensor(); // Sensor will read PPM concentration using the model, a and b values set previously or from the setup

  MQ8_ppm = MQ8.get_ppm();


  data_update = hora_dia+";";
```

```
    data_update += (String)temperature+";";
    data_update += (String)humidity+";";
    data_update += (String)correctedPPM+";";
    data_update += (String)MQ2_ppm+";";
    data_update += (String)MQ3_ppm+";";
    data_update += (String)MQ4_ppm+";";
    data_update += (String)MQ5_ppm+";";
    data_update += (String)MQ6_ppm+";";
    data_update += (String)MQ7_ppm+";";
    data_update += (String)MQ8_ppm;

    escribir_en_SD(data_update);
  }
}




/* ------------------------------------------------------
 *  ---------------------- FUNCIONES----------------------
 *  ------------------------------------------------------
  */
void inicializar_SD(){
  Serial.print("INICIANDO TARJETA SD ...");    //MOSTRAMOS EL MENSAJE
  if (!SD.begin(7)) {                         //INICIAMOS LA SD EL PIN 7
    Serial.println("INICIACION FALLIDA");      //MOSTRAMOS EL MENSAJE EN CASO
QUE SD NO FUNCIONE CORRECTAMENTE
    display.clearDisplay();
    display.setCursor(0,0);
    display.println("INICIACION FALLIDA");
    display.display();
    return;
  }
```

```
  Serial.println("  INICIADO CORRECTAMENTE");  //SI LA SD SE HA INICIADO CORRECTAMENTE MOSTRAMOS EL MENSAJE
  SD_iniciada = true;
  Serial.println(" ");                    //DEJAMOS UN SALTO DE LINEA
  display.clearDisplay();
  display.setCursor(0,0);
  display.println("SD INICIADA CORRECTAMENTE");
  display.display();
}
void obtener_fecha(){
  DateTime now = RTC.now(); // Devuelve fecha de hoy YYYY-MM-DD y nombre de archivo del dia
  anyo = now.year();
  mes = now.month();
  dia = now.day();
  fecha_dia = (String)anyo+"-"+(String)mes+"-"+(String)dia;
  nombre_diario = (String)mes+"-"+(String)dia+".txt";
  if(dia_nuevo){
    file_existe(nombre_diario);
    Serial.println(nombre_diario);
    dia_nuevo = false;
  }
}
void obtener_hora(){
  DateTime now = RTC.now(); // Obtiene la fecha y hora del RTC
  boolean actualizar_hora = false;

  if(min_o_seg == 0){ //actualizacion en segundos
    if(ultimo_seg != now.second()){
      actualizar_hora = true;
    }else{
      actualizar_hora = false;
    }
```

```
  }else{              //actualizacion en minutos
    if(ultimo_min != now.minute()){
      actualizar_hora = true;
    }else{
      actualizar_hora = false;
    }
  }

  if(actualizar_hora){
    hora = now.hour();
    minuto = now.minute();
    segundo = now.second();
    ultimo_seg = segundo;
    ultimo_min = minuto;
    hora_dia = (String)hora+":"+(String)minuto+":"+(String)segundo;
    dato_ya_actualizado = false;
    if(hora == 0 && minuto == 0 && segundo == 0){
      dia_nuevo = true;
      obtener_fecha();
    }
  }else{
    dato_ya_actualizado = true;
  }
}
void file_existe(String nombre){ // EL NOMBRE DE UN ARCHIVO NO PUEDE TENER
MAS DE 8 CHARS
  myFile = SD.open(nombre);
  if (myFile) { // EL ARCHIVO EXISTE
    Serial.println("El archivo existe");
    myFile.close();
    display.println("El archivo existe");
    display.display();
  } else {
```

```
    myFile = SD.open(nombre, FILE_WRITE); //CREAAMOS UN ARCHIVO
    myFile.println(cabecera_archivo);     // ESCRIBIMOS EL TEXTO EN EL ARCHICO
    myFile.close();
    display.println("Archivo Creado");
    display.display();
  }
}
void escribir_en_SD(String texto){ // EL NOMBRE DE UN ARCHIVO NO PUEDE
TENER MAS DE 8 CHARS
  if(dato_ya_actualizado==false){
     //Serial.println(texto);
     Serial.println("Dato guardado: " + texto);
     myFile = SD.open(nombre_diario, FILE_WRITE);
     myFile.println(texto);              // ESCRIBIMOS EL TEXTO EN EL ARCHICO
     myFile.close();                          // CERREMOS EL ARCHIVO

     display.clearDisplay();
     display.setCursor(0,0);
     display.println(nombre_diario);
     display.println(texto);
     display.display();
  }
}
```

Sensor 4

```
#include <MQ135.h>
#include <DHT.h>
#include "MQUnifiedsensor.h"

#include <SPI.h>
#include <SD.h>

#include <Wire.h>
#include "RTClib.h"
RTC_DS1307 RTC;

#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>

/********************** Variables necesarias SD y reloj **************/
File myFile;
String fecha,fecha_dia,hora_dia;
int anyo,mes,dia,hora,minuto,segundo;
String nombre_diario;
String data_update;
boolean dato_ya_actualizado = false;
boolean dia_nuevo = false;
boolean SD_iniciada = false;
boolean actualizar_datos = false;
String cabecera_archivo =
"Timestamp;Temperatura;Humedad;MQ135(CO2);MQ2(CO);MQ3(CO);MQ4(CO);MQ5
(CO);MQ6(CO);MQ7(CO);MQ8(CO)";
int ultimo_seg=0;
int ultimo_min=1;
/*  Tiempo entre actualizaciones en tarjeta SD
    variable min_o_seg debe tener dos posibles valores:
        min_o_seg = 0 --> actualizacion en segundos
```

```
      min_o_seg = 1 --> actualizacion en minutos
   La variable updating_time hará referencia a minutos o segundos depeniendo del valor
   de la variable min_o_seg. Los valores permitidos son: 1, 2, 3, 4, 5, 6, 10, 12, 15, 20, 30 y 60.
*/
int min_o_seg = 1;
int updating_time = 1;

/**********************LCD**************/
#define ANCHO 128
#define ALTO 32

#define OLED_RESET 4
Adafruit_SSD1306 display(ANCHO,ALTO,&Wire,OLED_RESET);

/**********************placa usada**************/
#define       Board                 ("Arduino MEGA")
#define       Voltage_Resolution    (5)
#define       ADC_Bit_Resolution    (10) // For arduino UNO/MEGA/NANO

/************************CALIBRACION MQ2*********************************/
#define       Pin_MQ2               (A1)  //Analog input 1 of your arduino
/************************Software Related Macros************************************/
#define       Type_MQ2              ("MQ-2") //MQ2
#define       RatioMQ2CleanAir      (9.83) //RS / R0 = 9.83 ppm
MQUnifiedsensor MQ2(Board, Voltage_Resolution, ADC_Bit_Resolution, Pin_MQ2, Type_MQ2);
/*****************************Globals***********************************************/


/************************CALIBRACION MQ3*********************************/
#define       Pin_MQ3               (A2)  //Analog input 2 of your arduino
/************************Software Related Macros************************************/
```

```
#define         Type_MQ3                ("MQ-3") //MQ3
#define         RatioMQ3CleanAir        (60) //RS / R0 = 60 ppm
MQUnifiedsensor MQ3(Board, Voltage_Resolution, ADC_Bit_Resolution, Pin_MQ3, Type_MQ3);
/*****************************Globals***********************************************/

/*************************CALIBRACION MQ4*********************************/
#define         Pin_MQ4                 (A3)  //Analog input 3 of your arduino
/************************Software Related Macros************************************/
#define         Type_MQ4                ("MQ-4") //MQ4
#define         RatioMQ4CleanAir        (4.4) //RS / R0 = 60 ppm
MQUnifiedsensor MQ4(Board, Voltage_Resolution, ADC_Bit_Resolution, Pin_MQ4, Type_MQ4);

/*************************CALIBRACION MQ5*********************************/
#define         Pin_MQ5                 (A4)  //Analog input 4 of your arduino
/************************Software Related Macros************************************/
#define         Type_MQ5                ("MQ-5") //MQ5
#define         RatioMQ5CleanAir        (6.5) //RS / R0 = 60 ppm
MQUnifiedsensor MQ5(Board, Voltage_Resolution, ADC_Bit_Resolution, Pin_MQ5, Type_MQ5);

/*************************CALIBRACION MQ6*********************************/
#define         Pin_MQ6                 (A5)  //Analog input 5 of your arduino
/************************Software Related Macros************************************/
#define         Type_MQ6                ("MQ-6") //MQ6
#define         RatioMQ6CleanAir        (10) //RS / R0 = 60 ppm
MQUnifiedsensor MQ6(Board, Voltage_Resolution, ADC_Bit_Resolution, Pin_MQ6, Type_MQ6);

/*************************CALIBRACION MQ7*********************************/
#define         Pin_MQ7                 (A6)  //Analog input 6 of your arduino
/************************Software Related Macros************************************/
#define         Type_MQ7                ("MQ-7") //MQ7
```

```
#define         RatioMQ7CleanAir        (27.5) //RS / R0 = 60 ppm
MQUnifiedsensor MQ7(Board, Voltage_Resolution, ADC_Bit_Resolution, Pin_MQ7, Type_MQ7);

/************************CALIBRACION MQ8***********************************/
#define         Pin_MQ8                 (A7)  //Analog input 7 of your arduino
/************************Software Related Macros***********************************/
#define         Type_MQ8                ("MQ-8") //MQ8
#define         RatioMQ8CleanAir        (70) //RS / R0 = 60 ppm
MQUnifiedsensor MQ8(Board, Voltage_Resolution, ADC_Bit_Resolution, Pin_MQ8, Type_MQ8);

#define PIN_MQ135 A0 // MQ135 Analog Input Pin
#define DHTPIN 2 // DHT Digital Input Pin
#define DHTTYPE DHT11 // DHT11 or DHT22, depends on your sensor

MQ135 mq135_sensor(PIN_MQ135);
DHT dht(DHTPIN, DHTTYPE);
float temperature, humidity; // Temp and Humid floats, will be measured by the DHT
float MQ2_ppm,MQ3_ppm,MQ4_ppm,MQ5_ppm,MQ6_ppm,MQ7_ppm,MQ8_ppm;

void setup() {
  Serial.begin(9600);
  display.begin(SSD1306_SWITCHCAPVCC, 0x3C);
  display.clearDisplay();
  display.setTextSize(1);
  display.setTextColor(WHITE,BLACK);
  display.setCursor(0,0);

  while(!SD_iniciada){
    inicializar_SD();
  }
```

```
  dht.begin();


  Wire.begin();
  RTC.begin();
  RTC.adjust(DateTime(__DATE__, __TIME__)); // Establece la fecha y hora
(Comentar una vez establecida la hora)


  display.begin(SSD1306_SWITCHCAPVCC, 0x3C);
  display.clearDisplay();
  display.setTextSize(1);
  display.setTextColor(WHITE,BLACK);
  display.setCursor(0,0);


  obtener_fecha();
  ultimo_seg = segundo;
  file_existe(nombre_diario);


// calibraciones del MQ2
  MQ2.setRegressionMethod(1); //_PPM =  a*ratio^b
  MQ2.setA(36974); MQ2.setB(-3.109); // Configure the equation to to calculate LPG
concentration
  MQ2.init();


  Serial.print("Calibrating MQ2, please wait.");
  float calcR0_MQ2 = 0;
  for(int i = 1; i<=10; i ++)
  {
    MQ2.update(); // Update data, the arduino will read the voltage from the analog pin
    calcR0_MQ2 += MQ2.calibrate(RatioMQ2CleanAir);
    Serial.print(".");
  }
  MQ2.setR0(calcR0_MQ2/10);
  Serial.println("  done!.");
```

```
  if(isinf(calcR0_MQ2)) {Serial.println("Warning: Conection issue, R0 is infinite (Open circuit detected) please check your wiring and supply"); while(1);}
  if(calcR0_MQ2 == 0){Serial.println("Warning: Conection issue found, R0 is zero (Analog pin shorts to ground) please check your wiring and supply"); while(1);}
  /*****************************  MQ CAlibration ********************************************/

  MQ2.serialDebug(true);

// FIN calibraciones del MQ2

// calibraciones del MQ3
  MQ3.setRegressionMethod(1); //_PPM =  a*ratio^b
  MQ3.setA(521853); MQ3.setB(-3.821); // Configure the equation to to calculate Benzene concentration
  MQ3.init();

  Serial.print("Calibrating MQ3, please wait.");
  float calcR0_MQ3 = 0;
  for(int i = 1; i<=10; i ++)
  {
    MQ3.update(); // Update data, the arduino will read the voltage from the analog pin
    calcR0_MQ3 += MQ3.calibrate(RatioMQ3CleanAir);
    Serial.print(".");
  }
  MQ3.setR0(calcR0_MQ3/10);
  Serial.println("  done!.");

  if(isinf(calcR0_MQ3)) {Serial.println("Warning: Conection issue, R0 is infinite (Open circuit detected) please check your wiring and supply"); while(1);}
  if(calcR0_MQ3 == 0){Serial.println("Warning: Conection issue found, R0 is zero (Analog pin shorts to ground) please check your wiring and supply"); while(1);}
  /*****************************  MQ CAlibration ********************************************/
  MQ3.serialDebug(true);
```

```
// FIN calibraciones del MQ3


// calibraciones del MQ4
  MQ4.setRegressionMethod(1); //_PPM =  a*ratio^b
  MQ4.setA(200000000000000); MQ4.setB(-19.05); // Configure the equation to to calculate CH4 concentration
  MQ4.init();


Serial.print("Calibrating please wait.");
  float calcR0_MQ4 = 0;
  for(int i = 1; i<=10; i ++)
  {
    MQ4.update(); // Update data, the arduino will read the voltage from the analog pin
    calcR0_MQ4 += MQ4.calibrate(RatioMQ4CleanAir);
    Serial.print(".");
  }
  MQ4.setR0(calcR0_MQ4/10);
  Serial.println("  done!.");


  if(isinf(calcR0_MQ4)) {Serial.println("Warning: Conection issue, R0 is infinite (Open circuit detected) please check your wiring and supply"); while(1);}
  if(calcR0_MQ4 == 0){Serial.println("Warning: Conection issue found, R0 is zero (Analog pin shorts to ground) please check your wiring and supply"); while(1);}
  /*****************************  MQ CAlibration ********************************************/
  MQ4.serialDebug(true);
// FIN calibraciones del MQ4


// calibraciones del MQ5
  MQ5.setRegressionMethod(1); //_PPM =  a*ratio^b
  MQ5.setA(491204); MQ5.setB(-5.826); // Configure the equation to to calculate H2 concentration
  MQ5.init();


  Serial.print("Calibrating please wait.");
```

```
  float calcR0_MQ5 = 0;
  for(int i = 1; i<=10; i ++)
  {
    MQ5.update(); // Update data, the arduino will read the voltage from the analog pin
    calcR0_MQ5 += MQ5.calibrate(RatioMQ5CleanAir);
    Serial.print(".");
  }
  MQ5.setR0(calcR0_MQ5/10);
  Serial.println("  done!.");

  if(isinf(calcR0_MQ5)) {Serial.println("Warning: Conection issue, R0 is infinite (Open circuit detected) please check your wiring and supply"); while(1);}
  if(calcR0_MQ5 == 0){Serial.println("Warning: Conection issue found, R0 is zero (Analog pin shorts to ground) please check your wiring and supply"); while(1);}
  /*****************************  MQ CAlibration ********************************************/
  MQ5.serialDebug(true);

// FIN calibraciones del MQ5

// calibraciones del MQ6
  MQ6.setRegressionMethod(1); //_PPM =  a*ratio^b
  MQ6.setA(1000000000000000); MQ6.setB(-13.5); // Configure the equation to to calculate CH4 concentration
  MQ6.init();

  Serial.print("Calibrating please wait.");
  float calcR0_MQ6 = 0;
  for(int i = 1; i<=10; i ++)
  {
    MQ6.update(); // Update data, the arduino will read the voltage from the analog pin
    calcR0_MQ6 += MQ6.calibrate(RatioMQ6CleanAir);
    Serial.print(".");
  }
  MQ6.setR0(calcR0_MQ6/10);
```

```
  Serial.println("  done!.");


  if(isinf(calcR0_MQ6)) {Serial.println("Warning: Conection issue, R0 is infinite (Open circuit detected) please check your wiring and supply"); while(1);}

  if(calcR0_MQ6 == 0){Serial.println("Warning: Conection issue found, R0 is zero (Analog pin shorts to ground) please check your wiring and supply"); while(1);}

  /*****************************  MQ CAlibration ********************************************/

  MQ6.serialDebug(true);


// FIN calibraciones del MQ6


// calibraciones del MQ7

  MQ7.setRegressionMethod(1); //_PPM =  a*ratio^b

  MQ7.setA(99.042); MQ7.setB(-1.518); // Configure the equation to calculate CO concentration value

  MQ7.init();


  Serial.print("Calibrating please wait.");

  float calcR0_MQ7 = 0;

  for(int i = 1; i<=10; i ++)

  {

    MQ7.update(); // Update data, the arduino will read the voltage from the analog pin

    calcR0_MQ7 += MQ7.calibrate(RatioMQ7CleanAir);

    Serial.print(".");

  }

  MQ7.setR0(calcR0_MQ7/10);

  Serial.println("  done!.");


  if(isinf(calcR0_MQ7)) {Serial.println("Warning: Conection issue, R0 is infinite (Open circuit detected) please check your wiring and supply"); while(1);}

  if(calcR0_MQ7 == 0){Serial.println("Warning: Conection issue found, R0 is zero (Analog pin shorts to ground) please check your wiring and supply"); while(1);}

  /*****************************  MQ CAlibration ********************************************/

  MQ7.serialDebug(true);
```

```
// FIN calibraciones del MQ7


// calibraciones del MQ8
  MQ8.setRegressionMethod(1); //_PPM =  a*ratio^b
  MQ8.setA(2000000000000000000); MQ8.setB(-8.074); // Configure the equation to to calculate H2 concentration
  MQ8.init();


  Serial.print("Calibrating please wait.");
  float calcR0_MQ8 = 0;
  for(int i = 1; i<=10; i ++)
  {
    MQ8.update(); // Update data, the arduino will read the voltage from the analog pin
    calcR0_MQ8 += MQ8.calibrate(RatioMQ8CleanAir);
    Serial.print(".");
  }
  MQ8.setR0(calcR0_MQ8/10);
  Serial.println("  done!.");


  if(isinf(calcR0_MQ8)) {Serial.println("Warning: Conection issue, R0 is infinite (Open circuit detected) please check your wiring and supply"); while(1);}
  if(calcR0_MQ8 == 0){Serial.println("Warning: Conection issue found, R0 is zero (Analog pin shorts to ground) please check your wiring and supply"); while(1);}
  /*****************************  MQ CAlibration ********************************************/
  MQ8.serialDebug(true);


// FIN calibraciones del MQ8


}


void loop() {
  obtener_hora();
```

```
if(min_o_seg == 0){ //actualizacion en segundos
  if(segundo%updating_time==0){
    actualizar_datos = true;
  }else{
    actualizar_datos = false;
  }
}else{              //actualizacion en minutos
  if(minuto%updating_time==0){
    actualizar_datos = true;
  }else{
    actualizar_datos = false;
  }
}

if(actualizar_datos){

  humidity = dht.readHumidity();
  temperature = dht.readTemperature();

  // Check if any reads failed and exit early (to try again).
  if (isnan(humidity) || isnan(temperature)) {
    Serial.println(F("Failed to read from DHT sensor!"));
    return;
  }

  float rzero = mq135_sensor.getRZero();
  float correctedRZero = mq135_sensor.getCorrectedRZero(temperature, humidity);
  float resistance = mq135_sensor.getResistance();
  float ppm = mq135_sensor.getPPM();
  float correctedPPM = mq135_sensor.getCorrectedPPM(temperature, humidity);

  MQ2.update(); // Update data, the arduino will read the voltage from the analog pin
```

```
  MQ2.readSensor(); // Sensor will read PPM concentration using the model, a and b values set previously or from the setup

  MQ2_ppm = MQ2.get_ppm();


  MQ3.update(); // Update data, the arduino will read the voltage from the analog pin

  MQ3.readSensor(); // Sensor will read PPM concentration using the model, a and b values set previously or from the setup

  MQ3_ppm = MQ3.get_ppm();


  MQ4.update(); // Update data, the arduino will read the voltage from the analog pin

  MQ4.readSensor(); // Sensor will read PPM concentration using the model, a and b values set previously or from the setup

  MQ4_ppm = MQ4.get_ppm();


  MQ5.update(); // Update data, the arduino will read the voltage from the analog pin

  MQ5.readSensor(); // Sensor will read PPM concentration using the model, a and b values set previously or from the setup

  MQ5_ppm = MQ5.get_ppm();


  MQ6.update(); // Update data, the arduino will read the voltage from the analog pin

  MQ6.readSensor(); // Sensor will read PPM concentration using the model, a and b values set previously or from the setup

  MQ6_ppm = MQ6.get_ppm();


  MQ7.update(); // Update data, the arduino will read the voltage from the analog pin

  MQ7.readSensor(); // Sensor will read PPM concentration using the model, a and b values set previously or from the setup

  MQ7_ppm = MQ7.get_ppm();


  MQ8.update(); // Update data, the arduino will read the voltage from the analog pin

  MQ8.readSensor(); // Sensor will read PPM concentration using the model, a and b values set previously or from the setup

  MQ8_ppm = MQ8.get_ppm();


  data_update = hora_dia+";";
```

```
    data_update += (String)temperature+";";
    data_update += (String)humidity+";";
    data_update += (String)correctedPPM+";";
    data_update += (String)MQ2_ppm+";";
    data_update += (String)MQ3_ppm+";";
    data_update += (String)MQ4_ppm+";";
    data_update += (String)MQ5_ppm+";";
    data_update += (String)MQ6_ppm+";";
    data_update += (String)MQ7_ppm+";";
    data_update += (String)MQ8_ppm;

    escribir_en_SD(data_update);
  }
}




/* ------------------------------------------------------
 *  ---------------------- FUNCIONES----------------------
 *  ------------------------------------------------------
  */
void inicializar_SD(){
  Serial.print("INICIANDO TARJETA SD ...");    //MOSTRAMOS EL MENSAJE
  if (!SD.begin(7)) {                         //INICIAMOS LA SD EL PIN 7
    Serial.println("INICIACION FALLIDA");      //MOSTRAMOS EL MENSAJE EN CASO
QUE SD NO FUNCIONE CORRECTAMENTE
    display.clearDisplay();
    display.setCursor(0,0);
    display.println("INICIACION FALLIDA");
    display.display();
    return;
  }
```

```
  Serial.println("  INICIADO CORRECTAMENTE");  //SI LA SD SE HA INICIADO CORRECTAMENTE MOSTRAMOS EL MENSAJE
  SD_iniciada = true;
  Serial.println(" ");                    //DEJAMOS UN SALTO DE LINEA
  display.clearDisplay();
  display.setCursor(0,0);
  display.println("SD INICIADA CORRECTAMENTE");
  display.display();
}
void obtener_fecha(){
  DateTime now = RTC.now(); // Devuelve fecha de hoy YYYY-MM-DD y nombre de archivo del dia
  anyo = now.year();
  mes = now.month();
  dia = now.day();
  fecha_dia = (String)anyo+"-"+(String)mes+"-"+(String)dia;
  nombre_diario = (String)mes+"-"+(String)dia+".txt";
  if(dia_nuevo){
    file_existe(nombre_diario);
    Serial.println(nombre_diario);
    dia_nuevo = false;
  }
}
void obtener_hora(){
  DateTime now = RTC.now(); // Obtiene la fecha y hora del RTC
  boolean actualizar_hora = false;

  if(min_o_seg == 0){ //actualizacion en segundos
    if(ultimo_seg != now.second()){
      actualizar_hora = true;
    }else{
      actualizar_hora = false;
    }
```

```
  }else{              //actualizacion en minutos
    if(ultimo_min != now.minute()){
      actualizar_hora = true;
    }else{
      actualizar_hora = false;
    }
  }

  if(actualizar_hora){
    hora = now.hour();
    minuto = now.minute();
    segundo = now.second();
    ultimo_seg = segundo;
    ultimo_min = minuto;
    hora_dia = (String)hora+":"+(String)minuto+":"+(String)segundo;
    dato_ya_actualizado = false;
    if(hora == 0 && minuto == 0 && segundo == 0){
      dia_nuevo = true;
      obtener_fecha();
    }
  }else{
    dato_ya_actualizado = true;
  }
}
void file_existe(String nombre){ // EL NOMBRE DE UN ARCHIVO NO PUEDE TENER
MAS DE 8 CHARS
  myFile = SD.open(nombre);
  if (myFile) { // EL ARCHIVO EXISTE
    Serial.println("El archivo existe");
    myFile.close();
    display.println("El archivo existe");
    display.display();
  } else {
```

```
    myFile = SD.open(nombre, FILE_WRITE); //CREAAMOS UN ARCHIVO
    myFile.println(cabecera_archivo);     // ESCRIBIMOS EL TEXTO EN EL ARCHICO
    myFile.close();
    display.println("Archivo Creado");
    display.display();
  }
}
void escribir_en_SD(String texto){ // EL NOMBRE DE UN ARCHIVO NO PUEDE
TENER MAS DE 8 CHARS
  if(dato_ya_actualizado==false){
     //Serial.println(texto);
     Serial.println("Dato guardado: " + texto);
     myFile = SD.open(nombre_diario, FILE_WRITE);
     myFile.println(texto);              // ESCRIBIMOS EL TEXTO EN EL ARCHICO
     myFile.close();                          // CERREMOS EL ARCHIVO

     display.clearDisplay();
     display.setCursor(0,0);
     display.println(nombre_diario);
     display.println(texto);
     display.display();
  }
}
```

Sensor 5

```
#include <MQ135.h>
#include <DHT.h>
#include "MQUnifiedsensor.h"

#include <SPI.h>
#include <SD.h>

#include <Wire.h>
#include "RTClib.h"
RTC_DS1307 RTC;

#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>

/********************** Variables necesarias SD y reloj **************/
File myFile;
String fecha,fecha_dia,hora_dia;
int anyo,mes,dia,hora,minuto,segundo;
String nombre_diario;
String data_update;
boolean dato_ya_actualizado = false;
boolean dia_nuevo = false;
boolean SD_iniciada = false;
boolean actualizar_datos = false;
String cabecera_archivo =
"Timestamp;Temperatura;Humedad;MQ135(CO2);MQ2(Alcohol);MQ3(Alcohol);MQ4(Al
cohol);MQ5(Alcohol);MQ6(Alcohol);MQ7(Alcohol);MQ8(Alcohol)";
int ultimo_seg=0;
int ultimo_min=1;
/*  Tiempo entre actualizaciones en tarjeta SD
    variable min_o_seg debe tener dos posibles valores:
        min_o_seg = 0 --> actualizacion en segundos
```

```
      min_o_seg = 1 --> actualizacion en minutos
   La variable updating_time hará referencia a minutos o segundos depeniendo del valor
   de la variable min_o_seg. Los valores permitidos son: 1, 2, 3, 4, 5, 6, 10, 12, 15, 20, 30 y 60.
*/
int min_o_seg = 1;
int updating_time = 1;

/**********************LCD**************/
#define ANCHO 128
#define ALTO 32

#define OLED_RESET 4
Adafruit_SSD1306 display(ANCHO,ALTO,&Wire,OLED_RESET);

/**********************placa usada**************/
#define       Board                 ("Arduino MEGA")
#define       Voltage_Resolution    (5)
#define       ADC_Bit_Resolution    (10) // For arduino UNO/MEGA/NANO

/************************CALIBRACION MQ2*********************************/
#define       Pin_MQ2               (A1)  //Analog input 1 of your arduino
/************************Software Related Macros************************************/
#define       Type_MQ2              ("MQ-2") //MQ2
#define       RatioMQ2CleanAir      (9.83) //RS / R0 = 9.83 ppm
MQUnifiedsensor MQ2(Board, Voltage_Resolution, ADC_Bit_Resolution, Pin_MQ2, Type_MQ2);
/*****************************Globals***********************************************/


/************************CALIBRACION MQ3*********************************/
#define       Pin_MQ3               (A2)  //Analog input 2 of your arduino
/************************Software Related Macros************************************/
```

```
#define         Type_MQ3                ("MQ-3") //MQ3
#define         RatioMQ3CleanAir        (60) //RS / R0 = 60 ppm
MQUnifiedsensor MQ3(Board, Voltage_Resolution, ADC_Bit_Resolution, Pin_MQ3, Type_MQ3);
/*****************************Globals***********************************************/

/*************************CALIBRACION MQ4*********************************/
#define         Pin_MQ4                 (A3)  //Analog input 3 of your arduino
/************************Software Related Macros************************************/
#define         Type_MQ4                ("MQ-4") //MQ4
#define         RatioMQ4CleanAir        (4.4) //RS / R0 = 60 ppm
MQUnifiedsensor MQ4(Board, Voltage_Resolution, ADC_Bit_Resolution, Pin_MQ4, Type_MQ4);

/*************************CALIBRACION MQ5*********************************/
#define         Pin_MQ5                 (A4)  //Analog input 4 of your arduino
/************************Software Related Macros************************************/
#define         Type_MQ5                ("MQ-5") //MQ5
#define         RatioMQ5CleanAir        (6.5) //RS / R0 = 60 ppm
MQUnifiedsensor MQ5(Board, Voltage_Resolution, ADC_Bit_Resolution, Pin_MQ5, Type_MQ5);

/*************************CALIBRACION MQ6*********************************/
#define         Pin_MQ6                 (A5)  //Analog input 5 of your arduino
/************************Software Related Macros************************************/
#define         Type_MQ6                ("MQ-6") //MQ6
#define         RatioMQ6CleanAir        (10) //RS / R0 = 60 ppm
MQUnifiedsensor MQ6(Board, Voltage_Resolution, ADC_Bit_Resolution, Pin_MQ6, Type_MQ6);

/*************************CALIBRACION MQ7*********************************/
#define         Pin_MQ7                 (A6)  //Analog input 6 of your arduino
/************************Software Related Macros************************************/
#define         Type_MQ7                ("MQ-7") //MQ7
```

```
#define         RatioMQ7CleanAir        (27.5) //RS / R0 = 60 ppm
MQUnifiedsensor MQ7(Board, Voltage_Resolution, ADC_Bit_Resolution, Pin_MQ7, Type_MQ7);

/************************CALIBRACION MQ8***********************************/
#define         Pin_MQ8                 (A7)  //Analog input 7 of your arduino
/************************Software Related Macros***********************************/
#define         Type_MQ8                ("MQ-8") //MQ8
#define         RatioMQ8CleanAir        (70) //RS / R0 = 60 ppm
MQUnifiedsensor MQ8(Board, Voltage_Resolution, ADC_Bit_Resolution, Pin_MQ8, Type_MQ8);

#define PIN_MQ135 A0 // MQ135 Analog Input Pin
#define DHTPIN 2 // DHT Digital Input Pin
#define DHTTYPE DHT11 // DHT11 or DHT22, depends on your sensor

MQ135 mq135_sensor(PIN_MQ135);
DHT dht(DHTPIN, DHTTYPE);
float temperature, humidity; // Temp and Humid floats, will be measured by the DHT
float MQ2_ppm,MQ3_ppm,MQ4_ppm,MQ5_ppm,MQ6_ppm,MQ7_ppm,MQ8_ppm;

void setup() {
  Serial.begin(9600);
  display.begin(SSD1306_SWITCHCAPVCC, 0x3C);
  display.clearDisplay();
  display.setTextSize(1);
  display.setTextColor(WHITE,BLACK);
  display.setCursor(0,0);

  while(!SD_iniciada){
    inicializar_SD();
  }
```

```
  dht.begin();


  Wire.begin();
  RTC.begin();
  RTC.adjust(DateTime(__DATE__, __TIME__)); // Establece la fecha y hora
(Comentar una vez establecida la hora)


  display.begin(SSD1306_SWITCHCAPVCC, 0x3C);
  display.clearDisplay();
  display.setTextSize(1);
  display.setTextColor(WHITE,BLACK);
  display.setCursor(0,0);


  obtener_fecha();
  ultimo_seg = segundo;
  file_existe(nombre_diario);


// calibraciones del MQ2
  MQ2.setRegressionMethod(1); //_PPM =  a*ratio^b
  MQ2.setA(3616.1); MQ2.setB(-2.675); // Configure the equation to to calculate LPG
concentration
  MQ2.init();


  Serial.print("Calibrating MQ2, please wait.");
  float calcR0_MQ2 = 0;
  for(int i = 1; i<=10; i ++)
  {
    MQ2.update(); // Update data, the arduino will read the voltage from the analog pin
    calcR0_MQ2 += MQ2.calibrate(RatioMQ2CleanAir);
    Serial.print(".");
  }
  MQ2.setR0(calcR0_MQ2/10);
  Serial.println("  done!.");
```

```
  if(isinf(calcR0_MQ2)) {Serial.println("Warning: Conection issue, R0 is infinite (Open circuit detected) please check your wiring and supply"); while(1);}
  if(calcR0_MQ2 == 0){Serial.println("Warning: Conection issue found, R0 is zero (Analog pin shorts to ground) please check your wiring and supply"); while(1);}
  /*****************************  MQ CAlibration ********************************************/

  MQ2.serialDebug(true);

// FIN calibraciones del MQ2

// calibraciones del MQ3
  MQ3.setRegressionMethod(1); //_PPM =  a*ratio^b
  MQ3.setA(0.3934); MQ3.setB(-1.504); // Configure the equation to to calculate Benzene concentration
  MQ3.init();

  Serial.print("Calibrating MQ3, please wait.");
  float calcR0_MQ3 = 0;
  for(int i = 1; i<=10; i ++)
  {
    MQ3.update(); // Update data, the arduino will read the voltage from the analog pin
    calcR0_MQ3 += MQ3.calibrate(RatioMQ3CleanAir);
    Serial.print(".");
  }
  MQ3.setR0(calcR0_MQ3/10);
  Serial.println("  done!.");

  if(isinf(calcR0_MQ3)) {Serial.println("Warning: Conection issue, R0 is infinite (Open circuit detected) please check your wiring and supply"); while(1);}
  if(calcR0_MQ3 == 0){Serial.println("Warning: Conection issue found, R0 is zero (Analog pin shorts to ground) please check your wiring and supply"); while(1);}
  /*****************************  MQ CAlibration ********************************************/
  MQ3.serialDebug(true);
```

```
// FIN calibraciones del MQ3


// calibraciones del MQ4
  MQ4.setRegressionMethod(1); //_PPM =  a*ratio^b
  MQ4.setA(60000000000); MQ4.setB(-14.01); // Configure the equation to to calculate CH4 concentration
  MQ4.init();


Serial.print("Calibrating please wait.");
  float calcR0_MQ4 = 0;
  for(int i = 1; i<=10; i ++)
  {
    MQ4.update(); // Update data, the arduino will read the voltage from the analog pin
    calcR0_MQ4 += MQ4.calibrate(RatioMQ4CleanAir);
    Serial.print(".");
  }
  MQ4.setR0(calcR0_MQ4/10);
  Serial.println("  done!.");


  if(isinf(calcR0_MQ4)) {Serial.println("Warning: Conection issue, R0 is infinite (Open circuit detected) please check your wiring and supply"); while(1);}
  if(calcR0_MQ4 == 0){Serial.println("Warning: Conection issue found, R0 is zero (Analog pin shorts to ground) please check your wiring and supply"); while(1);}
  /*****************************  MQ CAlibration ********************************************/
  MQ4.serialDebug(true);
// FIN calibraciones del MQ4


// calibraciones del MQ5
  MQ5.setRegressionMethod(1); //_PPM =  a*ratio^b
  MQ5.setA(97124); MQ5.setB(-4.918); // Configure the equation to to calculate H2 concentration
  MQ5.init();


  Serial.print("Calibrating please wait.");
```

```
  float calcR0_MQ5 = 0;
  for(int i = 1; i<=10; i ++)
  {
    MQ5.update(); // Update data, the arduino will read the voltage from the analog pin
    calcR0_MQ5 += MQ5.calibrate(RatioMQ5CleanAir);
    Serial.print(".");
  }
  MQ5.setR0(calcR0_MQ5/10);
  Serial.println("  done!.");

  if(isinf(calcR0_MQ5)) {Serial.println("Warning: Conection issue, R0 is infinite (Open circuit detected) please check your wiring and supply"); while(1);}
  if(calcR0_MQ5 == 0){Serial.println("Warning: Conection issue found, R0 is zero (Analog pin shorts to ground) please check your wiring and supply"); while(1);}
  /*****************************  MQ CAlibration ********************************************/
  MQ5.serialDebug(true);

// FIN calibraciones del MQ5

// calibraciones del MQ6
MQ6.setRegressionMethod(1); //_PPM =  a*ratio^b
  MQ6.setA(50000000); MQ6.setB(-6.017); // Configure the equation to to calculate CH4 concentration
  MQ6.init();

  Serial.print("Calibrating please wait.");
  float calcR0_MQ6 = 0;
  for(int i = 1; i<=10; i ++)
  {
    MQ6.update(); // Update data, the arduino will read the voltage from the analog pin
    calcR0_MQ6 += MQ6.calibrate(RatioMQ6CleanAir);
    Serial.print(".");
  }
  MQ6.setR0(calcR0_MQ6/10);
```

```
  Serial.println("  done!.");


  if(isinf(calcR0_MQ6)) {Serial.println("Warning: Conection issue, R0 is infinite (Open circuit detected) please check your wiring and supply"); while(1);}

  if(calcR0_MQ6 == 0){Serial.println("Warning: Conection issue found, R0 is zero (Analog pin shorts to ground) please check your wiring and supply"); while(1);}

  /*****************************  MQ CAlibration ********************************************/

  MQ6.serialDebug(true);


// FIN calibraciones del MQ6


// calibraciones del MQ7

  MQ7.setRegressionMethod(1); //_PPM =  a*ratio^b

  MQ7.setA(40000000000000000); MQ7.setB(-12.35); // Configure the equation to calculate CO concentration value

  MQ7.init();


  Serial.print("Calibrating please wait.");

  float calcR0_MQ7 = 0;

  for(int i = 1; i<=10; i ++)

  {

    MQ7.update(); // Update data, the arduino will read the voltage from the analog pin

    calcR0_MQ7 += MQ7.calibrate(RatioMQ7CleanAir);

    Serial.print(".");

  }

  MQ7.setR0(calcR0_MQ7/10);

  Serial.println("  done!.");


  if(isinf(calcR0_MQ7)) {Serial.println("Warning: Conection issue, R0 is infinite (Open circuit detected) please check your wiring and supply"); while(1);}

  if(calcR0_MQ7 == 0){Serial.println("Warning: Conection issue found, R0 is zero (Analog pin shorts to ground) please check your wiring and supply"); while(1);}

  /*****************************  MQ CAlibration ********************************************/

  MQ7.serialDebug(true);
```

```
// FIN calibraciones del MQ7


// calibraciones del MQ8
  MQ8.setRegressionMethod(1); //_PPM =  a*ratio^b
  MQ8.setA(76101); MQ8.setB(-1.86); // Configure the equation to to calculate H2 concentration
  MQ8.init();


  Serial.print("Calibrating please wait.");
  float calcR0_MQ8 = 0;
  for(int i = 1; i<=10; i ++)
  {
    MQ8.update(); // Update data, the arduino will read the voltage from the analog pin
    calcR0_MQ8 += MQ8.calibrate(RatioMQ8CleanAir);
    Serial.print(".");
  }
  MQ8.setR0(calcR0_MQ8/10);
  Serial.println("  done!.");


  if(isinf(calcR0_MQ8)) {Serial.println("Warning: Conection issue, R0 is infinite (Open circuit detected) please check your wiring and supply"); while(1);}
  if(calcR0_MQ8 == 0){Serial.println("Warning: Conection issue found, R0 is zero (Analog pin shorts to ground) please check your wiring and supply"); while(1);}
  /*****************************  MQ CAlibration ********************************************/
  MQ8.serialDebug(true);


// FIN calibraciones del MQ8


}


void loop() {
  obtener_hora();
```

```
if(min_o_seg == 0){ //actualizacion en segundos
  if(segundo%updating_time==0){
    actualizar_datos = true;
  }else{
    actualizar_datos = false;
  }
}else{              //actualizacion en minutos
  if(minuto%updating_time==0){
    actualizar_datos = true;
  }else{
    actualizar_datos = false;
  }
}

if(actualizar_datos){

  humidity = dht.readHumidity();
  temperature = dht.readTemperature();

  // Check if any reads failed and exit early (to try again).
  if (isnan(humidity) || isnan(temperature)) {
    Serial.println(F("Failed to read from DHT sensor!"));
    return;
  }

  float rzero = mq135_sensor.getRZero();
  float correctedRZero = mq135_sensor.getCorrectedRZero(temperature, humidity);
  float resistance = mq135_sensor.getResistance();
  float ppm = mq135_sensor.getPPM();
  float correctedPPM = mq135_sensor.getCorrectedPPM(temperature, humidity);

  MQ2.update(); // Update data, the arduino will read the voltage from the analog pin
```

```
  MQ2.readSensor(); // Sensor will read PPM concentration using the model, a and b values set previously or from the setup

  MQ2_ppm = MQ2.get_ppm();


  MQ3.update(); // Update data, the arduino will read the voltage from the analog pin

  MQ3.readSensor(); // Sensor will read PPM concentration using the model, a and b values set previously or from the setup

  MQ3_ppm = MQ3.get_ppm();


  MQ4.update(); // Update data, the arduino will read the voltage from the analog pin

  MQ4.readSensor(); // Sensor will read PPM concentration using the model, a and b values set previously or from the setup

  MQ4_ppm = MQ4.get_ppm();


  MQ5.update(); // Update data, the arduino will read the voltage from the analog pin

  MQ5.readSensor(); // Sensor will read PPM concentration using the model, a and b values set previously or from the setup

  MQ5_ppm = MQ5.get_ppm();


  MQ6.update(); // Update data, the arduino will read the voltage from the analog pin

  MQ6.readSensor(); // Sensor will read PPM concentration using the model, a and b values set previously or from the setup

  MQ6_ppm = MQ6.get_ppm();


  MQ7.update(); // Update data, the arduino will read the voltage from the analog pin

  MQ7.readSensor(); // Sensor will read PPM concentration using the model, a and b values set previously or from the setup

  MQ7_ppm = MQ7.get_ppm();


  MQ8.update(); // Update data, the arduino will read the voltage from the analog pin

  MQ8.readSensor(); // Sensor will read PPM concentration using the model, a and b values set previously or from the setup

  MQ8_ppm = MQ8.get_ppm();


  data_update = hora_dia+";";
```

```
    data_update += (String)temperature+";";
    data_update += (String)humidity+";";
    data_update += (String)correctedPPM+";";
    data_update += (String)MQ2_ppm+";";
    data_update += (String)MQ3_ppm+";";
    data_update += (String)MQ4_ppm+";";
    data_update += (String)MQ5_ppm+";";
    data_update += (String)MQ6_ppm+";";
    data_update += (String)MQ7_ppm+";";
    data_update += (String)MQ8_ppm;

    escribir_en_SD(data_update);
  }
}




/* ------------------------------------------------------
 *  ---------------------- FUNCIONES----------------------
 *  ------------------------------------------------------
  */
void inicializar_SD(){
  Serial.print("INICIANDO TARJETA SD ...");    //MOSTRAMOS EL MENSAJE
  if (!SD.begin(7)) {                         //INICIAMOS LA SD EL PIN 7
    Serial.println("INICIACION FALLIDA");      //MOSTRAMOS EL MENSAJE EN CASO
QUE SD NO FUNCIONE CORRECTAMENTE
    display.clearDisplay();
    display.setCursor(0,0);
    display.println("INICIACION FALLIDA");
    display.display();
    return;
  }
```

```
  Serial.println("  INICIADO CORRECTAMENTE");  //SI LA SD SE HA INICIADO CORRECTAMENTE MOSTRAMOS EL MENSAJE
  SD_iniciada = true;
  Serial.println(" ");                        //DEJAMOS UN SALTO DE LINEA
  display.clearDisplay();
  display.setCursor(0,0);
  display.println("SD INICIADA CORRECTAMENTE");
  display.display();
}
void obtener_fecha(){
  DateTime now = RTC.now(); // Devuelve fecha de hoy YYYY-MM-DD y nombre de archivo del dia
  anyo = now.year();
  mes = now.month();
  dia = now.day();
  fecha_dia = (String)anyo+"-"+(String)mes+"-"+(String)dia;
  nombre_diario = (String)mes+"-"+(String)dia+".txt";
  if(dia_nuevo){
    file_existe(nombre_diario);
    Serial.println(nombre_diario);
    dia_nuevo = false;
  }
}
void obtener_hora(){
  DateTime now = RTC.now(); // Obtiene la fecha y hora del RTC
  boolean actualizar_hora = false;

  if(min_o_seg == 0){ //actualizacion en segundos
    if(ultimo_seg != now.second()){
      actualizar_hora = true;
    }else{
      actualizar_hora = false;
    }
```

```
  }else{              //actualizacion en minutos
    if(ultimo_min != now.minute()){
      actualizar_hora = true;
    }else{
      actualizar_hora = false;
    }
  }

  if(actualizar_hora){
    hora = now.hour();
    minuto = now.minute();
    segundo = now.second();
    ultimo_seg = segundo;
    ultimo_min = minuto;
    hora_dia = (String)hora+":"+(String)minuto+":"+(String)segundo;
    dato_ya_actualizado = false;
    if(hora == 0 && minuto == 0 && segundo == 0){
      dia_nuevo = true;
      obtener_fecha();
    }
  }else{
    dato_ya_actualizado = true;
  }
}
void file_existe(String nombre){ // EL NOMBRE DE UN ARCHIVO NO PUEDE TENER
MAS DE 8 CHARS
  myFile = SD.open(nombre);
  if (myFile) { // EL ARCHIVO EXISTE
    Serial.println("El archivo existe");
    myFile.close();
    display.println("El archivo existe");
    display.display();
  } else {
```

```
    myFile = SD.open(nombre, FILE_WRITE); //CREAAMOS UN ARCHIVO
    myFile.println(cabecera_archivo);     // ESCRIBIMOS EL TEXTO EN EL ARCHICO
    myFile.close();
    display.println("Archivo Creado");
    display.display();
  }
}
void escribir_en_SD(String texto){ // EL NOMBRE DE UN ARCHIVO NO PUEDE
TENER MAS DE 8 CHARS
  if(dato_ya_actualizado==false){
     //Serial.println(texto);
     Serial.println("Dato guardado: " + texto);
     myFile = SD.open(nombre_diario, FILE_WRITE);
     myFile.println(texto);              // ESCRIBIMOS EL TEXTO EN EL ARCHICO
     myFile.close();                          // CERREMOS EL ARCHIVO

     display.clearDisplay();
     display.setCursor(0,0);
     display.println(nombre_diario);
     display.println(texto);
     display.display();
  }
}
```