

ANEXOS

ANEXO I: Bibliografía

1. Referencias

Referencia	Enlace
Placa Arduino	http://www.arduino.cc
Sensor de TGS4161 (Figaro) datasheet	http://www.figarosensor.com/products/4161pdf.pdf
Amplificador operacional TLC271 datasheet.	http://www.ti.com/lit/ds/symlink/tlc271.pdf
Módem GSM Cinterion TC65T (hardware) datasheet	http://www.cinterion.com/tl_files/cinterion/downloads/hc65t_hd_v03000an.pdf
Módem GSM Cinterion TC65T (Comandos AT) datasheet.	http://webs.uvigo.es/jagfernandez/ET/CMOV/pr2/tc65_v02000.pdf
Comandos AT	http://www.wikilearning.com/curso_gratis/que_son_y_como_funcionan_los_modems-como_usar_los_comandos_at/3477-18
Sustancias contaminates	http://www.proyectosalohogar.com/Ciencias_ambientales/10CAtm1/200Conta.html
Tipos de sensores	www.alphasense.com

2. Artículos

Fuente	Autor	Enlace
Elektor		http://www.elektor.es/revistas/2012/febrero/controlar-con-un-arduino-y-un-pc.2055966.lynkx
Tendencias21		http://www.tendencias21.net/Doha-intenta-tomar-el-relevo-de-Kioto_a14320.html

3. Bibliografía

Autor	Título	Año
F. Reverter and R. Pallàs Areny	<i>Circuitos de interfaz directa sensor-microcontrolador,</i>	Marcombo, Barcelona, 2008
Pallàs-Areny R., Webster J. G.	<i>Sensors and Signal Conditioning</i>	2n Edition. 2001

ANEXO II: Índice de tablas

Número de tabla	Descripción
<i>Tabla 1</i>	<i>Concentraciones de gases en el aire.</i>
<i>Tabla 2</i>	<i>Efectos del sobre una persona.</i>
<i>Tabla 3</i>	<i>Comparativa de sensores de .</i>
<i>Tabla 4</i>	<i>Comparativa de placas Arduino.</i>
<i>Tabla 5</i>	<i>Fuerza electromotriz (EMF) en función del número de exhalaciones.</i>
<i>Tabla 6</i>	<i>Tensiones de alimentación de los componentes activos.</i>
<i>Tabla 7</i>	<i>Estados de funcionamiento del módem con sus consumos.</i>
<i>Tabla 8</i>	<i>Tiempos estimados de funcionamiento diario de cada dispositivo</i>
<i>Tabla 9</i>	<i>Consumos teóricos estimados del sistema.</i>
<i>Tabla 10</i>	<i>Consumos experimentales del sistema.</i>
<i>Tabla 11</i>	<i>Tipos de comandos AT</i>

ANEXO III: Índice de figuras

Número de figura	Descripción
<i>Figura 1</i>	<i>Diagrama de bloques del nodo sensor.</i>
<i>Figura 2</i>	<i>Diagrama esquemático de un pellistor y su montaje.</i>
<i>Figura 3</i>	<i>Esquema de funcionamiento de un sensor de gas electroquímico.</i>
<i>Figura 4</i>	<i>Método no dispersivo IR.</i>
<i>Figura 5</i>	<i>Esquema del sensor NDIR de dos canales.</i>
<i>Figura 6</i>	<i>Estructura del sensor electroquímico.</i>
<i>Figura 7</i>	<i>Sensor de estado sólido.</i>
<i>Figura 8</i>	<i>Sensor de , TGS 4161.</i>
<i>Figura 9</i>	<i>Disposición de los pines del sensor TGS4161.</i>
<i>Figura 10</i>	<i>Curva característica del TGS4161.</i>
<i>Figura 11</i>	<i>Tipos de placas Arduino.</i>
<i>Figura 12</i>	<i>Parte delantera de la placa ARDUINO UNO R3.</i>
<i>Figura 13</i>	<i>Parte trasera de la placa ARDUINO UNO R3.</i>
<i>Figura 14</i>	<i>Identificación de elementos Arduino UNO R3.</i>
<i>Figura 15</i>	<i>Módem Cinterion TC65T.</i>
<i>Figura 16</i>	<i>DS 1307.</i>
<i>Figura 17</i>	<i>Frontal y Bajo del Reloj.</i>
<i>Figura 18</i>	<i>Convertidor TTL-RS232</i>
<i>Figura 19</i>	<i>Conexión Arduino-Convertidor.</i>
<i>Figura 20</i>	<i>Tarjeta SD.</i>
<i>Figura 21</i>	<i>Lector de tarjeta SD.</i>
<i>Figura 22</i>	<i>Estructura de la red GSM.</i>
<i>Figura 23</i>	<i>Circuito de medida de la tensión del sensor TGS4161.</i>
<i>Figura 24</i>	<i>Circuito de medida para la caracterización de la impedancia del sensor TGS4161.</i>
<i>Figura 25</i>	<i>Circuito inversor para el acondicionamiento de señal.</i>
<i>Figura 26</i>	<i>Circuito para establecer la ganancia al A.O.</i>
<i>Figura 27</i>	<i>Circuito para eliminar la tensión de offset.</i>
<i>Figura 28</i>	<i>Curva de respuesta del sensor TGS4161.</i>
<i>Figura 29</i>	<i>Fuerza electromotriz (EMF) en función del número de exhalaciones.</i>
<i>Figura 30</i>	<i>Panel Solar.</i>
<i>Figura 31</i>	<i>Regulador de carga.</i>
<i>Figura 32</i>	<i>Etapas de carga del regulador de carga</i>
<i>Figura 33</i>	<i>Batería de plomo ácido.</i>
<i>Figura 34</i>	<i>Sistema de alimentación completo.</i>
<i>Figura 35</i>	<i>Circuito alimentación del sensor TGS4161.</i>

Figura 36	<i>Entorno de desarrollo.</i>
Figura 37	<i>Estructura de un programa Arduino.</i>
Figura 38	<i>Diagrama de flujo de la programación del Arduino.</i>

ANEXO IV: Código fuente del programa

```

// Librería incluidas
#include <Wire.h>
#include <SoftwareSerial.h>
// Declaración de constantes y variables
#define DS1307_I2C_ADDRESS 0x68
SoftwareSerial puerto(7,8); // pin 7 <--> RX y pin 8 <--> TX
boolean registro = true;
boolean email = true;
boolean cab = true;
String cuerpo;
// Bloque de inicialización y configuración
void setup() {
    // Inicialización del puerto serial
    Serial.begin(115200);
    Serial.print(F("Free mem: "));
    Serial.println(freeMemory());
    // Inicialización del reloj
    Wire.begin();
    // Inicialización del puerto serie virtual, módem
    puerto.begin(19200);
    // Configurar la tarjeta SIM para que lea los SMS en formato texto
    puerto.println(F("AT+CMGF=1"));
    while(puerto.read() != 'K');
    // Establecer la configuración a los pines digitales
    pinMode(0,OUTPUT); // calentador de placa de acondicionamiento
    pinMode(5,OUTPUT); // calentador de sensor de CO2
}
// Bucle principal del programa
void loop() {
    // Comprobación de envío de email
    if(EnviarEmail()) {
        EnvioDeEmail();
        cuerpo = "";
        cuerpo = "Fecha____Hora__Valor CO2(ppm)\n";
    }
    delay(20000);
    // Comprobación de registro de datos
    registrarDato();
    delay(20000);
    // Comprobación de envío de SMS
    comprobarSms();
    delay(20000);
}
// Método que comprueba si es necesario el envío de email
boolean EnviarEmail() {
    // Obtenemos la fecha y hora actual
    byte second, minute, hour, dayOfWeek, dayOfMonth, month, year;
    getDateDs1307(&second, &minute, &hour, &dayOfWeek, &dayOfMonth, &month, &year);
    Serial.print(F("Valor de hora: "));
    Serial.println(hour);
    // Se introduce la cabecera del fichero
    if(cab) {
        cuerpo = "Fecha____Hora__Valor CO2(ppm)\n";
        cab = false;
    }
}

```

```

        Serial.println(cuerpo);
    }
    // Comprobamos si es necesario el envío de email
    if(hour == 0 && email == false) {
        email = true;
        return true;
    } else if (hour == 0) {
        email = true;
        return false;
    } else {
        email = false;
        return false;
    }
}
// Método que comprueba si es necesario el registro de la medición de CO2
boolean registrarDato() {
    // Obtenemos la fecha y hora actual
    String minutos, hora, dia, mes, anio, linearegistro;
    byte second, minute, hour, dayOfWeek, dayOfMonth, month, year;
    getDateDs1307(&second, &minute, &hour, &dayOfWeek, &dayOfMonth, &month, &year);
    Serial.print(F("Valor de minutos: "));
    Serial.println(minute);
    // Comprobamos si es necesario realizar el registro del dato
    if(minute >= 30 && minute <= 39 && registro == false) {
        registro = true;
        if(minute >= 30 && minute <= 39) {
            minutos = "30";
        }
        if(0 <= hour && hour < 10) {
            hora = "0";
            hora += (String)hour;
        }
        else {
            hora = (String)hour;
        }
        if(0 <= dayOfMonth && dayOfMonth < 10) {
            linearegistro = "0";
            linearegistro += (String)dayOfMonth;
        }
        else {
            linearegistro = (String)dayOfMonth;
        }
        if(0 <= month && month < 10) {
            linearegistro += "0";
            linearegistro += (String)month;
        }
        else {
            linearegistro += (String)month;
        }
        linearegistro += "20";
        linearegistro += (String)year;
        linearegistro += " ";
        linearegistro += hora;
        linearegistro += ":";
        linearegistro += minutos;
        linearegistro += " ";
    }
}

```

```

        linearegistro += (String)obtenerDato();
        linearegistro += "\n";
        Serial.println(linearegistro);
        // Se añade la línea del registro al fichero
        cuerpo += linearegistro;
        Serial.println(cuerpo);
        return true;
    }
    else if(minute >= 30 && minute <= 39) {
        registro = true;
        return true;
    }
    else {
        registro = false;
        return true;
    }
}

// Método que obtiene el valor del sensor de CO2
int obtenerDato()
{
    int valorSensorCO2;
    Serial.println(F("Obteniendo el dato....."));
    // Pin para calentar el sensor de CO2 --> 5
    digitalWrite(5, HIGH);
    // Pin de acondicionamiento de señal --> 0
    digitalWrite(0, HIGH);
    delay(20000);
    valorSensorCO2 = analogRead(A0);
    // Se apaga el calentamiento del sensor de CO2
    digitalWrite(5, LOW);
    digitalWrite(0, LOW);
    return valorSensorCO2;
}

// Método que comprueba si existe algún SMS en la tarjeta SIM del módem
boolean comprobarSms()
{
    String respuesta, telefono;
    puerto.println("AT+CMGR=1");
    delay(3000);
    while(puerto.available()) {
        respuesta += String((char)puerto.read());
    }
    Serial.println(respuesta);
    Serial.print(F("Longitud de la respuesta del modem: "));
    Serial.println(respuesta.length());
    delay(1000);
    // Numero de caracteres recibidos en respuesta al comando AT+CMGR=1
    if(respuesta.length() > 35)
    {
        Serial.print(F("Valor de respuesta[34]: "));
        Serial.println(respuesta[34]);
        if(respuesta[34] == '6'){
            telefono = respuesta.substring(34,43);
            Serial.print(F("Numero de telefono READ: "));

```

```

Serial.println(telefono);
Serial.print(F("componentes de telefono: "));
Serial.println(telefono.length());
if(telefono.length() != 9){
  return false;
}
enviarSms(telefono);
return true;
}
else if(respuesta[34] == '6') {
  telefono = respuesta.substring(34,43);
  Serial.print(F("Numero de telefono UNREAD: "));
  Serial.println(telefono);
  return false;
}
else {
  Serial.println(F("NO hay SMS que enviar"));
  return false;
}
}
else {
  Serial.println(F("No hay SMS que enviar"));
  return false;
}
}
// Método que realiza el envío del SMS
void enviarSms(String telefono) {
  String textoSms;
  delay(500);
  // Leemos el valor del sensor y lo almacenamos
  int valorSensorCO2 = obtenerDato();
  // Creamos el cuerpo del SMS en función de la concentración de CO2.
  if(valorSensorCO2 <= 5 || valorSensorCO2 >= 512) {
    textoSms = "Dato incorrecto, el sistema no ha podido obtener el dato";
  }
  else if(valorSensorCO2 >= 25 && valorSensorCO2 < 512) {
    textoSms = "La concentracion de CO2 es de 380 ppm. Esta dentro de los margenes
permitidos";
  } else if(valorSensorCO2 > 5 && valorSensorCO2 < 25) {
    textoSms = "La concentracion de CO2 es mayor de 380 ppm. Supera los máximos
permitidos";
  }
  Serial.println(textoSms);
  delay(1000);
  puerto.println(F("AT+CMGF=1")); // Selecciona modo texto
  delay(1000);
  while(puerto.read() != 'K');
  puerto.print(F("AT+CMGS=\\"));
  puerto.print(telefono); // Envía el número de teléfono
  puerto.println(F("\\"));
  delay(3000);
  puerto.print(textoSms); // Envía el cuerpo del SMS
  delay(500);
  puerto.write(0x1A); // Mandamos un Ctrl+Z
  puerto.write(0x0D); // Mandamos un CR
  puerto.write(0x0A); // Mandamos un LF
}

```



```

delay(10000);
// Borrar de la memoria el SMS recibido
puerto.println(F("at+cmgd=1"));
delay(1000);
while(puerto.available()){
Serial.print((char)puerto.read());
}
}

// Método que proporciona la fecha y hora a partir del ds1307
void getDateDs1307(byte *second, byte *minute, byte *hour, byte *dayOfWeek, byte *dayOfMonth, byte
*month, byte *year) {
// Resetea el registro puntero
Wire.beginTransmission(DS1307_I2C_ADDRESS);
Wire.write(0);
Wire.endTransmission();

Wire.requestFrom(DS1307_I2C_ADDRESS, 7);

*second = bcdToDec(Wire.read() & 0x7f);
*minute = bcdToDec(Wire.read());
*hour = bcdToDec(Wire.read() & 0x3f);
*dayOfWeek = bcdToDec(Wire.read());
*dayOfMonth = bcdToDec(Wire.read());
*month = bcdToDec(Wire.read());
*year = bcdToDec(Wire.read());
}

// Método que convierte números normales decimales a BCD (binario decimal codificado)
byte decToBcd(byte val)
{
return ( val/10*16) + (val%10) );
}

// Método que convierte BCD (binario decimal codificado) a números normales decimales
byte bcdToDec(byte val)
{
return ( (val/16*10) + (val%16) );
}

```

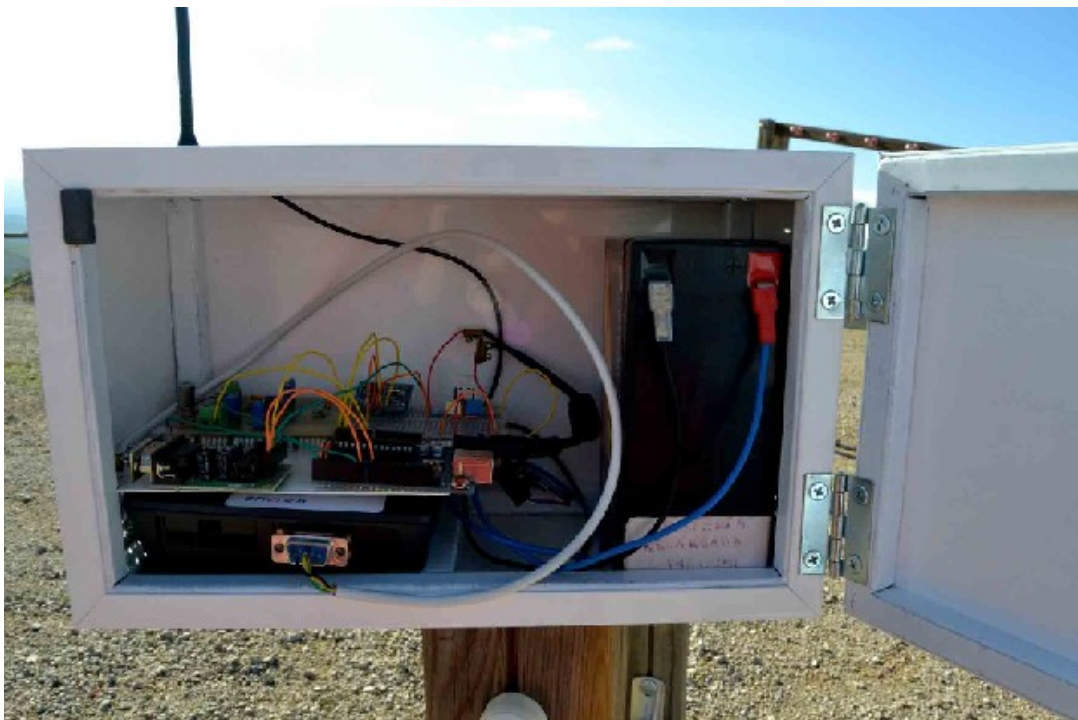
ANEXO V: Encapsulado

Se ha realizado una búsqueda de la caja que alojará al sistema. Las consideraciones que se han tenido en cuenta para la fabricación de la misma han sido que esté preparada para la intemperie, que deje pasar el aire y que proteja al sistema de la radiación solar.

Las principales características son:

- Construcción en un material ligero e impermeable como el aluminio.
- Dimensiones: 300 mm x 170mm x 150 mm.
- Incluye todas las piezas y tornillería para su montaje.

El dispositivo se puede ubicar en el lugar que nosotros queramos, preferiblemente en lugares donde se quiera controlar las concentraciones de . A continuación se muestran unas fotos del proyecto terminado.



Parte interior del sistema.



Montaje del dispositivo en el exterior.