Universidad de Alcalá
Escuela Politécnica Superior

Grado en Ingeniería Electrónica de Comunicaciones

Trabajo Fin de Grado

Sistema Remoto de adquisición de variables meteorológicas con monitorización en un entorno web

Autor: Sergio Lluva Plaza

Tutor: José Manuel Villadangos Carrizo

2018

UNIVERSIDAD DE ALCALÁ
Escuela Politécnica Superior

GRADO EN INGENIERÍA ELECTRÓNICA DE COMUNICACIONES

Trabajo Fin de Grado

Sistema Remoto de adquisición de variables meteorológicas con monitorización en un entorno web

Autor: Sergio Lluva Plaza

Tutor: José Manuel Villadangos Carrizo

TRIBUNAL:

Presidente: José Manuel Rodríguez Ascariz

Vocal 1º: César Mataix Gómez

Vocal 2º: José Manuel Villadangos Carrizo

FECHA: 8 de enero de 2018

Índice

Índice de figuras

Índice de tablas

Índice de códigos

1. Resumen

El presente Trabajo Fin de Grado pretende documentar el diseño llevado a cabo para realizar una plataforma autónoma de adquisición de variables relacionadas con la meteorología, que serán graficadas en un entorno web.

El sistema remoto está formado por un microcontrolador que controla la adquisición de datos mediante distintos sensores y coordina el envío de los datos obtenidos a una base de datos, para lo que dispone de dos métodos de envío de los datos, mediante una comunicación WiFi o a través de GPRS.

Una vez que los datos están almacenados se procede a mostrarlos al usuario de forma gráfica, mediante la librería de representación de datos de Highcharts, a través de una página web con la que se puede interactuar.

2. Abstract

The present End of Degree Project aims to document the design carried out to realize an autonomous platform for the acquisition of variables related to meteorology, which will be graphed in a web environment.

The remote system is formed by a microcontroller that controls the acquisition of data by means of different sensors and coordinates the sending of the obtained data to a database, for which it has two methods of sending the data, through a WiFi communication or through of GPRS.

Once the data is stored, it is presented to the user in graphic form, through the Highcharts data representation library, through a web page with which it can interact.

3. Resumen extendido

La realización de proyectos en el campo de IoT (Internet of Things) está teniendo un gran auge en los últimos años debido a sus infinitas posibilidades y a las múltiples utilidades que pueden obtener.

En la actualidad existen numerosas cuestiones por las cuales monitorizar las condiciones climatológicas de un determinado lugar. Por ejemplo, sería de utilidad saber estos parámetros en lugares donde la actividad principal sea la agricultura.

El proyecto consiste en realizar la adquisición de un conjunto de variables relacionadas con la meteorología, para tal fin se usa el microcontrolador LPC17868. Para la transmisión de los datos obtenidos, se usan dos módulos de comunicaciones, dependiendo si el sistema se encuentra localizado en un lugar remoto sin acceso a un punto de conexión WiFi se utilizará la transmisión vía GPRS, mientras que si el sistema se encuentra en un área urbana con acceso a un punto WiFi se utilizará un módulo WiFi para la transmisión de los datos, de esta forma se ahorrará dinero frente a la transmisión GPRS.

Las variables a medir serán: radiación ultravioleta, temperatura, presión atmosférica, humedad, luminosidad, dirección y velocidad del viento, lluvia. También se utilizará un módulo GPS para obtener las coordenadas en las que se encuentra el sistema. Esta característica puede ser de utilidad, por ejemplo, si el sistema se coloca en el campo y para volver a recogerlo no se encuentra debido a que ha crecido mucho la vegetación de la zona, de esta forma sería localizable con facilidad.

Los datos serán enviados hacia un servidor de Internet, en el que se realizará su tratamiento para poder representarlos en un entorno gráfico en una página web. Con ello se pretende que los datos sean interpretables fácilmente por cualquier usuario.

El sistema podrá emplazarse en cualquier lugar, ya que será autónomo. Para ello se realizará un estudio del consumo de los dispositivos que componen el sistema para hacer un correcto dimensionamiento del subsistema de alimentación, que está compuesto por una batería, un panel solar y su regulador.

Existen muchas plataformas IoT con las que se pueden graficar los datos de manera sencilla, por ejemplo, ThingSpeak, Cayene MyDevices, ThingsBoard, etc, pero estas plataformas no permiten o lo permiten en un bajo grado la modificación del aspecto visual de sus gráficas, siendo bastante sencillas, por los que se ha decidido realizar las gráficas mediante HighCharts para que sean más atractivas visualmente mediante una interfaz sencilla. El incluir este método de representación implica almacenar los datos en una base de datos para poder disponer de ellos cuando se vayan a graficar, esto en las anteriores plataformas es totalmente transparente para el usuario.

El diagrama que representa de forma general los bloques que compondrán el sistema se muestra a continuación:

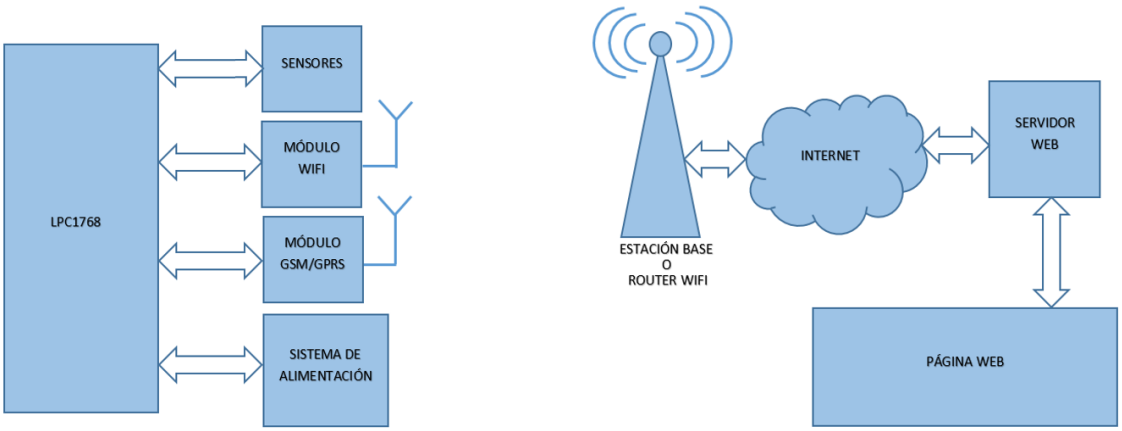

Figura 1. Diagrama de bloques general.

4. Palabras clave

Palabras clave: microcontrolador, estación meteorológica, HighCharts, IoT, página web

5. Memoria

5.1. Introducción

Este capítulo consiste en la explicación de la funcionalidad de todos los componentes que forman el sistema completo. Primero se explica el funcionamiento del microcontrolador utilizado, así como de sus periféricos usados en este proyecto. Después se explica cada sensor y módulo de comunicación utilizados con un enfoque Hardware (explicación de sus características) y un enfoque Software que describe el proceso seguido para que el microcontrolador pueda leer sus datos. También se explica la forma en la que se ha realizado la página web. Una vez explicadas las partes que componen el sistema se procede a explicar cómo funciona el sistema de manera conjunta.

5.2. Microcontrolador LPC1768

El elemento central que coordina todo el sistema es el microcontrolador LPC1768 del fabricante NXP Semiconductor, basado en un núcleo ARM Cortex-M3, con una arquitectura ARMv7-M y maneja un set de instrucciones Thumb-2.

Las principales características de este microcontrolador son:

- Reloj de hasta 100 MHz.
- 512 kB de memoria flash.
- 64 kB de memoria de datos.
- Controlador DMA de 8 canales.
- Bus CAN, SSP, SPI e I2C.
- 4 UART´s
- 8 canales ADC de 8 bits.
- DAC de 10 bits.
- Módulo PWM.
- QEI (Quadrature Encoder Inteface).
- 4 Timers de propósito general de 32 bits.
- Real Time Clock.
- MPU (Unidad de protección de memoria).

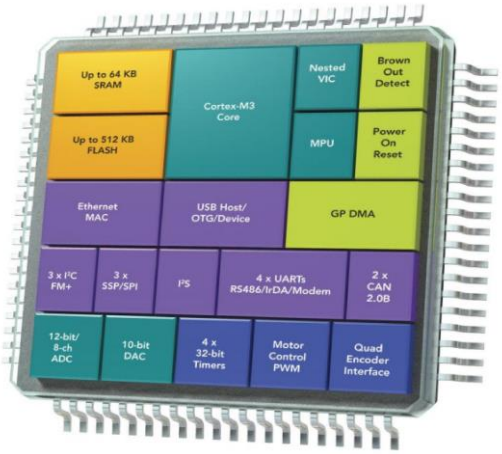

Figura 2. Microcontrolador LPC1768.

Las funcionalidades de este microcontrolador que se usan en este proyecto se explican con mayor detenimiento en los apartados siguientes.

5.2.1. GPIO.

Cada pin del microcontrolador LPC1768 puede tener varias funciones en relación con los periféricos que se usen, debido a ello requiere de registros para su configuración, que forman el bloque de conexión de los pines. Estos registros son:

- Registros PINSELx: como cada pin puede tener hasta cuatro funciones distintas, se requieren dos bits para indicar la función de cada pin. Existen los registros desde PINSEL0 hasta PINSEL9.
- Registros PINMODEx: con estos registros se configura el modo de entrada de cada puerto. Se pueden configurar con resistencia de pull-up, con resistencia de pull-down o en drenador abierto.

Los puertos de entrada/salida de propósito general (GPIO) para ser utilizados se deben configurar como GPIO mediante los bits y el registro PINSELx correspondientes. El LPC1768 tiene cinco puertos de propósito general. Los registros que se describen a continuación tienen todos 32 bits.

- Registros FIOxDIR: con un bit se configura el pin que tiene función de GPIO como entrada o salida. Siendo “x” el número de puerto, del 0 al 4.
- Registros FIOxPIN: tienen funciones distintas si se escribe en ellos o se lee de ellos. El valor que se escribe en cada bit del registro equivale al valor que tendrá cada pin (un 0 será un nivel bajo en el pin correspondiente y un 1 será un nivel alto). Mientras que, si se lee de él, devuelve el valor que tiene cada pin correspondiente con su bit.
- Registros FIOxSET: si el pin está configurado como GPIO, escribir un 1 en un bit de estos registros se traduce en un nivel alto en su correspondiente pin. Escribir un 0 no provoca efecto alguno.
- Registros FIOxCLR: si el pin está configurado como GPIO, escribir un 1 en un bit de estos registros se traduce en un nivel bajo en su correspondiente pin. Escribir un 0 no provoca efecto alguno.
- Registros FIOxMASK: registros de enmascaramiento. Escribir un 1 en un bit del registro enmascara en el pin correspondiente las acciones de los registros FIOxPIN, FIOxSET y FIOxCLR asociados a ese pin. Escribir un 0 en un bit del registro permite usar los registros FIOxPIN, FIOxSET y FIOxCLR asociados al pin.

5.2.2. Interrupciones.

Una excepción es un evento que provoca una pausa en el curso actual de ejecución del programa para pasar a ejecutar el código específico de atención a la excepción. Dentro de las excepciones se encuentran las interrupciones que son provocadas por eventos producidos por periféricos del microcontrolador o por interrupciones externas. Cada excepción tiene asociado un número de excepción en la tabla de vectores que la identifica.

Los estados en los que puede estar una interrupción son tres:

- Inactiva: el evento asociado a la interrupción no se ha disparado.
- Pendiente: se ha disparado el evento de la interrupción, pero no se está ejecutando la función de atención a la interrupción correspondiente.
- Ejecutándose: se está ejecutando el servicio de atención a la interrupción correspondiente.

Si hay varias interrupciones pendientes, la próxima en ejecutarse será la que mayor prioridad tenga, es decir, la que tenga mayor número de prioridad.

Se usa la biblioteca de funciones CMSIS (Cortex Microcontroller Software Interface Standard), que facilita el acceso a recursos del microprocesador Cortex M3 para la gestión de interrupciones y otros aspectos. Con las funciones que proporciona esta biblioteca la configuración de los registros asociados al manejo de interrupciones es transparente para el programador.

El LPC1768 dispone de cuatro entradas dedicadas de interrupción externas. Mediante el registro EXTMODE se puede configurar si la interrupción es activa por flanco o por nivel y mediante el registro EXTPOLAR se configura si es activa por nivel alto o bajo, o por flanco de subida o bajada.

Se usan las siguientes funciones de la biblioteca CMSIS relacionadas con la configuración de interrupciones:

- *NVIC_SetPriorityGrouping (uint32_t PriorityGroup)*: configura el grupo de prioridad
- *NVIC_EnableIRQ (IRQn_Type IRQn)*: habilita la interrupción en el NVIC.
- *NVIC_DisableIRQ (IRQn_Type IRQn)*: deshabilita la interrupción en el NVIC.
- *NVIC_SetPriority (IRQn_Type IRQn, uint32_t priority)*: configura el nivel de prioridad asociado a una interrupción.

5.2.3. Comunicación serie asíncrona.

La comunicación entre el microcontrolador y los módulos WiFi y SIM808, para el envío de datos, se realiza mediante una comunicación serie asíncrona. Este tipo de comunicación es muy sencilla y solo se necesitan dos líneas de conexión, además de la señal de referencia de tensión. Si la comunicación se realizase en un solo sentido, es decir, del microcontrolador hacia un sensor o viceversa, solo sería necesaria una línea de conexión. En este proyecto se usan las dos líneas debido a que se utiliza una comunicación en los dos sentidos, para ello se necesita una línea de transmisión de la información, TX, y otra de recepción, RX.

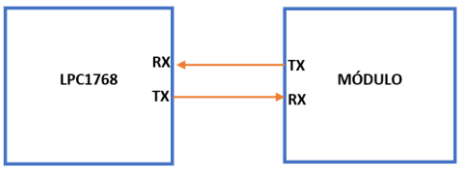

Figura 3. Conexionado comunicación serie asíncrona.

La UART (Universal Asynchronous Receiver-Transmitter) es un periférico que está incorporado en el microcontrolador LPC1768, en concreto tiene cuatro periféricos de

este tipo. La información se transmite y recibe mediante tramas, las cuales tienen las características siguientes:

- La línea de datos debe estar a nivel alto en estado de reposo.
- El envío y recepción de una trama comienza por el bit de START, representado con un nivel bajo en la línea.
- Después del bit START, se envía el dato de ocho bits empezando por el bit menos significativo hasta el bit más significativo.
- La trama termina con el bit de STOP, que se representa mediante un nivel alto en la línea.
- El tiempo de cada bit viene definido por el inverso de la velocidad de comunicación, especificada en bits/segundo o baudios.
- La configuración descrita en los anteriores puntos es la que se va a utilizar en este proyecto. Destacar que el dato puede tener longitud de 5, 6, 7 u 8 bits, puede existir un bit de paridad y la condición de STOP puede ser doble.

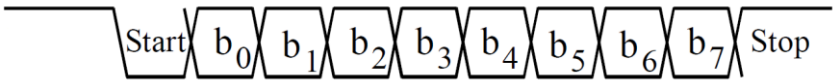

Figura 4. Estructura de una trama UART.

Tras la definición de las características de la UART se procede a explicar de forma general mediante los archivos utilizados para la programación de la UART 2, cómo se debe configurar en el LPC1768 para un correcto funcionamiento:

- En primer lugar, la UART que se quiere utilizar debe estar alimentada, lo cual se realiza con los bits correspondientes del registro PCONP.
- Los pines de transmisión y recepción de los datos deben estar configurados como RxD y TxD en el registro PINSELx.
- Como la comunicación se realiza mediante interrupción, se debe habilitar en el registro NVIC.

El archivo "*uart2.h*" contiene la definición de las constantes asociadas a los desplazamientos de bits para escribir en los bits de los registros correspondientes. En él también se definen las siguientes variables:

- *buffer2[255]*: buffer de recepción de 255 caracteres.
- **ptr_rx2*: puntero de recepción.
- *rx_completa2*: flag de recepción de cadena, se activa al recibir la tecla return CR (ASCII=13).
- **ptr_tx2*: puntero de transmisión.
- *tx_completa2*: flag de transmisión de cadena, se activa al transmitir el carácter null (fin de cadena).

```
#ifndef uart2_H_
#define uart2_H_

#define UART_ACCEPTED_BAUDRATE_ERROR    3
```

```
#define CHAR_8_BIT                      (3 << 0)
#define STOP_1_BIT                      (0 << 2)
#define PARITY_NONE                     (0 << 3)
#define DLAB_ENABLE                     (1 << 7)
#define FIFO_ENABLE                     (1 << 0)
#define RBR_IRQ_ENABLE                  (1 << 0)
#define THRE_IRQ_ENABLE                 (1 << 1)
#define UART_LSR_THRE                   (1 << 5)
#define RDA_INTERRUPT                   (2 << 1)
#define CTI_INTERRUPT                   (6 << 1)

extern char buffer2[255];
extern char *ptr_rx2;
extern char rx_completa2;
extern char *ptr_tx2;
extern char tx_completa2;

void UART2_IRQHandler(void);
void tx_cadena_UART2(char *cadena);
static int uart2_set_baudrate(unsigned int baudrate);
void uart2_init(int baudrate);

#endif
```

Código 1. Fichero uart2.h

Las funciones del archivo “uart2.c” son:

- *uart2_init(int baudrate):* función de configuración de la UART a la que se le pasa como argumento la velocidad de comunicación en baudios. En primer lugar, se seleccionan los pines cuya funcionalidad será RX y TX mediante el registro PINSEL. Posteriormente se activa la alimentación de la UART activando el bit correspondiente en el registro PCONP. En el registro LCR de la UART2 se configura la estructura de las tramas, en este caso con ocho bits de datos, sin bit de paridad y con un bit de stop. Después, se llama a la función *“uart2_set_baudrate()”* y se le pasa como argumento la velocidad de comunicación. Por último, se habilita la interrupción mediante los registros del NVIC y se le asigna una prioridad.

```
void uart2_init(int baudrate)
{
    LPC_PINCON->PINSEL0|=(1<<20)|(1<<22);
    LPC_SC->PCONP|=(1<<24);
    LPC_UART2->LCR &= ~STOP_1_BIT & ~PARITY_NONE;
    LPC_UART2->LCR |= CHAR_8_BIT;

    uart2_set_baudrate(baudrate);

    LPC_UART2->IER = THRE_IRQ_ENABLE|RBR_IRQ_ENABLE;
    NVIC_EnableIRQ(UART2_IRQn);
    NVIC_SetPriority(UART2_IRQn, 0);
}
```

Código 2. Función uart2_init.

- *uart2_set_baudrate():* función que se le pasa como argumento la velocidad de comunicación y calcula la configuración de los registros involucrados en base a una fórmula especificada en la hoja de características del LPC1768.

$$UARTn_{baudrate} = \frac{PCLK}{16 \cdot (256 \cdot UnDLM \cdot UnDLL) \cdot \left(1 + \frac{DivAddVal}{MulVal}\right)} \quad [1]$$

```
static int uart2_set_baudrate(unsigned int baudrate)
{
```

```
    int errorStatus = -1;

    unsigned int uClk =SystemCoreClock/4;
    unsigned int calcBaudrate = 0;
    unsigned int temp = 0;

    unsigned int mulFracDiv, dividerAddFracDiv;
    unsigned int divider = 0;
    unsigned int mulFracDivOptimal = 1;
    unsigned int dividerAddOptimal = 0;
    unsigned int dividerOptimal = 0;

    unsigned int relativeError = 0;
    unsigned int relativeOptimalError = 100000;

    uClk = uClk >> 4; /* div by 16 */

    for (mulFracDiv = 1; mulFracDiv <= 15; mulFracDiv++) {
        for (dividerAddFracDiv = 0; dividerAddFracDiv <= 15;
dividerAddFracDiv++) {
            temp = (mulFracDiv * uClk) / (mulFracDiv +
dividerAddFracDiv);
            divider = temp / baudrate;
            if ((temp % baudrate) > (baudrate / 2))
                divider++;
            if (divider > 2 && divider < 65536) {
                calcBaudrate = temp / divider;

                if (calcBaudrate <= baudrate) {
                    relativeError = baudrate - calcBaudrate;
                } else {
                    relativeError = calcBaudrate - baudrate;
                }

                if (relativeError < relativeOptimalError) {
                    mulFracDivOptimal = mulFracDiv;
                    dividerAddOptimal = dividerAddFracDiv;
                    dividerOptimal = divider;
                    relativeOptimalError = relativeError;
                    if (relativeError == 0)
                        break;
                }
            }
        }

        if (relativeError == 0)
            break;
    }

    if (relativeOptimalError < ((baudrate *
UART_ACCEPTED_BAUDRATE_ERROR) / 100)) {

        LPC_UART2->LCR |= DLAB_ENABLE;  // importante poner a 1
        LPC_UART2->DLM = (unsigned char) ((dividerOptimal >> 8) &
0xFF);
        LPC_UART2->DLL = (unsigned char) dividerOptimal;
        LPC_UART2->LCR &= ~DLAB_ENABLE; // importante poner a 0

        LPC_UART2->FDR = ((mulFracDivOptimal << 4) & 0xF0) |
(dividerAddOptimal & 0x0F);

        errorStatus = 0; //< Success
    }

    return errorStatus;
}
```

Código 3. Función uart2_set_baudrate.

- *UART2_IRQHandler():* función de atención a la interrupción de la UART2. Se consulta la causa que provoca la interrupción mediante el registro IIR. Si la interrupción se debe a que hay un dato para ser leído, se va almacenando en el puntero de recepción el valor del registro RBR hasta que se recibe la cadena completa y se activa el flag de cadena completa recibida. Mientras que, si la

interrupción es debida a la transmisión de datos a través de la UART, se van cargando los datos a transmitir en el registro THR hasta finalizar la cadena y se activa el flag de transmisión de cadena completada.

```
void UART2_IRQHandler(void)
{
    switch(LPC_UART2->IIR&0x0E) {
    case 0x04:
    *ptr_rx2=LPC_UART2->RBR;
    if (*ptr_rx2++ ==13)
    {
    *ptr_rx2=0;
    rx_completa2 = 1;
    ptr_rx2=buffer2;
    }
    break;

    case 0x02:
    if (*ptr_tx2!=0) LPC_UART2->THR=*ptr_tx2++;
    else tx_completa2=1;
    break;
    }
}
```

Código 4. Función UART2_IRQHandler

- *tx_cadena_UART2():* función que transmite por la UART2 la cadena apuntada por el argumento que se le pasa o transmite directamente la cadena de texto entre comillas.

```
void tx_cadena_UART2(char *cadena)
{
    ptr_tx2=cadena;
    tx_completa2=0;
    LPC_UART2->THR=*ptr_tx2++;
}
```

Código 5. Función tx_cadena_UART2.

5.2.4. Bus I2C.

En este proyecto se usan sensores cuya salida es digital y también con salida analógica.

Para todos los sensores con salida digital se utiliza el protocolo de comunicación I2C (Inter-Integrated Circuit). Este estándar de comunicación fue desarrollado por Philips en el año 1982. Destaca por su sencillez de implementación, pocos cables de interconexión y reducido tamaño de los componentes que usan dicha comunicación. La principal desventaja de este bus frente a otros buses paralelos es la velocidad de transmisión de datos, la cual se ve afectada debido a su arquitectura serie.

Los dispositivos que forman este bus deben tener una dirección única en la línea de transmisión para poder realizar una comunicación exitosa.

Este bus requiere solamente dos líneas para su implementación, SDA y SCL, aparte de la línea común de masa que deben compartir los dispositivos que forman el bus.

El esquema que se va a utilizar en este proyecto se puede resumir en la siguiente imagen:

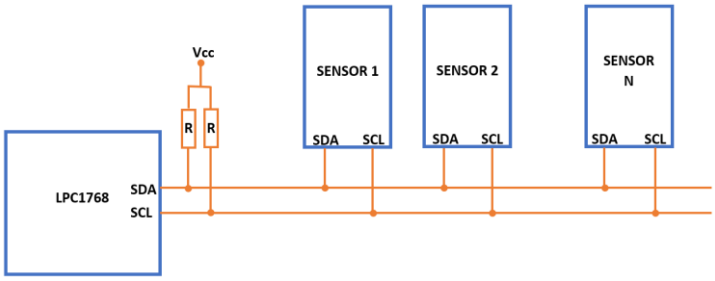

Figura 5. Esquema del bus I2C.

La transmisión de los datos (bits) se realiza por la línea SDA, mientras que por la línea SCL se transmite la señal de reloj, en la que se necesita un pulso por cada bit enviado por SDA.

Para implementar el protocolo I2C en este proyecto, se ha realizado mediante dos puertos de propósito general de entrada y salida.

La condición de START se define como una transición de nivel alto a nivel bajo de la línea de datos SDA, mientras la línea de reloj SCL se mantiene a nivel alto. Una vez que el maestro ha transmitido esta condición el bus pasa a estar ocupado. El byte siguiente que se transmite contiene la dirección del esclavo en los 7 bits más significativos.

La condición de STOP se define como una transición de nivel bajo a nivel alto en la línea SDA mientras la línea SCL se mantiene a nivel alto. Esta condición determina la finalización de la transmisión de datos.

A continuación, se muestra un diagrama típico del protocolo I2C utilizado en la transmisión de datos desde el maestro a un dispositivo esclavo para realizar una operación de escritura en alguno de sus registros:

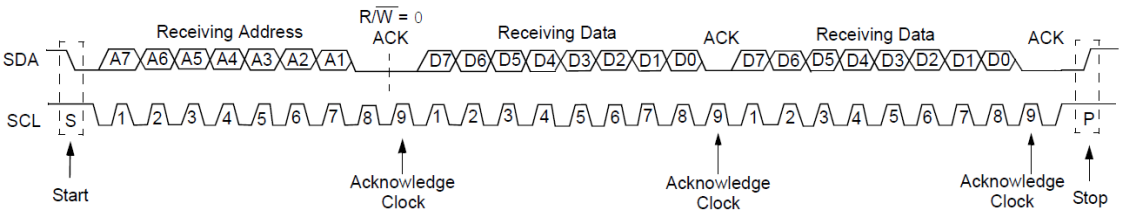

Figura 6. Diagrama de escritura mediante I2C.

Como se puede observar la secuencia comienza con la condición de START, después se envía la dirección de siete bits del esclavo empezando por el bit más significativo, seguido del bit R/W=0, indicando que se va a realizar una operación de escritura. Posteriormente se envían los bytes empezando por el bit más significativo. Cada byte enviado debe ser confirmado con un ACK, que fija la línea SDA a nivel bajo durante un pulso de SCL, excluyendo el último byte que se desee enviar que tiene que ser confirmado por un ACK y seguido a él se debe generar la condición de STOP.

A continuación, se muestra un diagrama típico del protocolo I2C utilizado en la transmisión de datos desde el esclavo a un dispositivo maestro para realizar una operación de lectura de alguno de sus registros:

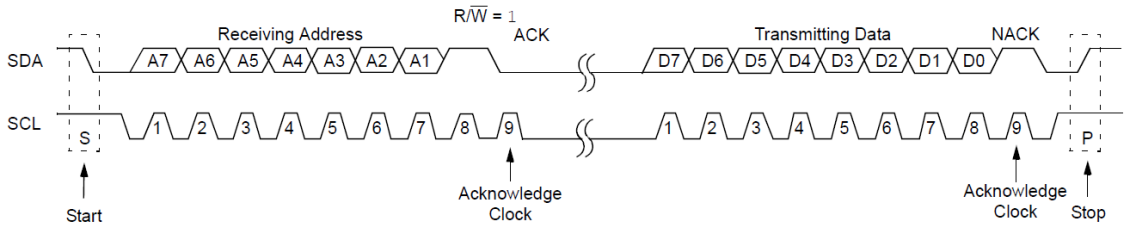

Figura 7. Diagrama de lectura mediante I2C.

Se comienza enviando la condición de START, seguida de la dirección de siete bits del esclavo, pero en este caso el bit R/W será 1, indicando al esclavo que se va a leer datos de alguno de sus registros y se recibe el bit ACK de confirmación y el esclavo comienza a transmitir los ocho bits comenzando por el más significativo. Tras transmitir el último byte se termina con un NACK para indicarlo y por último se envía el bit de STOP.

La librería utilizada para la comunicación I2C está basada en la implementación del bus mediante dos puertos de propósito general.

El fichero "*i2c_lpc17xx.h*" contiene la definición del número de pin que se usa como SDA y SCL y las funciones que se utilizan para implementar el protocolo I2C:

```
#ifndef _I2C_LPC17XX_H
#define _I2C_LPC17XX_H

#include <LPC17xx.h>

#define SDA 27 //pin 27
#define SCL 28 //pin 28

void I2Cdelay(void);
void pulso_SCL(void);
void I2CSendByte(unsigned char byte);
void I2CSendAddr(unsigned char addr, unsigned char rw);
unsigned char I2CGetByte(unsigned char ACK);
void I2CSendStop(void);

void I2CWrite8(uint8_t addr, uint8_t reg, uint8_t val);
uint8 t I2CRead8(uint8 t addr, uint8 t reg);

#endif
```

Código 6. Fichero i2c_lpc17xx.h

El fichero *"i2c_lpc17xx.c"* contiene la implementación de las funciones relacionadas con el bus I2C, las cuales se explican a continuación:

- *I2Cdelay():* genera un retardo que garantiza el cumplimiento de los tiempos del bus. Este retardo debe ser como mínimo de 4.7 µs según dictan las especificaciones del bus I2C.

```
void I2Cdelay(void)
{
  unsigned char i;
  for(i=0;i<120;i++);
}
```

Código 7. Función I2Cdelay.

- *pulso_SCL()*: genera un pulso en la línea de reloj mediante el puerto de propósito general asociado a la línea SCL.

```
void pulso_SCL(void)
{
    LPC_GPIO0->FIOSET=(1<<SCL);
    I2Cdelay();
```

```
    LPC_GPIO0->FIOCLR=(1<<SCL);
    I2Cdelay();
}
```

Código 8. Función pulso_SCL.

- *I2CSendByte():* envía cada byte empezando por el bit de mayor peso (MSB) a la vez que genera los pulsos de la señal de reloj correspondiente a cada bit enviado. Una vez enviado el dato se configura el pin SDA como entrada mediante el registro FIODIR para esperar la recepción del ACK que debe enviar el dispositivo esclavo, mientras el maestro genera un pulso de la señal de reloj. Por último, se vuelve a configurar el pin SDA como salida.

```
void I2CSendByte(unsigned char byte)
{
 unsigned char i;
 for(i=0;i<8;i++)
    {
      if (byte &0x80)  LPC_GPIO0->FIOSET=(1<<SDA);
      else  LPC_GPIO0->FIOCLR=(1<<SDA);
      byte = byte <<1;
      pulso_SCL();
    }

 LPC_GPIO0->FIODIR&=~(1<<SDA);
 pulso_SCL();

 LPC_GPIO0->FIODIR|=(1<<SDA);
}
```

Código 9. Función I2CSendByte.

- *I2CSendAddr()*: esta función se encarga de generar la condición de START y posteriormente envía la dirección del bus I2C del esclavo junto con el bit que indica si se va a realizar una operación de escritura o de lectura. Se genera la condición de START, primero se pone a nivel bajo la línea SDA y se espera el retardo para poner a nivel bajo la línea SCL. Después se envía el byte de la dirección de siete bits a la que se añade el bit de menor peso que indica operación de lectura o escritura.

```
void I2CSendAddr(unsigned char addr, unsigned char rw)
{
  LPC_GPIO0->FIODIR|=(1<<SDA)|(1<<SCL);
  LPC_GPIO0->FIOSET|=(1<<SDA)|(1<<SCL);
  I2Cdelay();
  LPC_GPIO0->FIOCLR = (1 << SDA);
  I2Cdelay();
  LPC_GPIO0->FIOCLR = (1 << SCL);
  I2Cdelay();
  I2CSendByte((addr=addr<<1) + rw);
}
```

Código 10. Función I2CSendAddr.

- *I2CGetByte()*: se encarga de leer un byte enviado por el esclavo. Al final de la lectura el máster envía el bit ACK=0 después de todos los bytes que no sean el último de la comunicación y ACK=1, equivalente a un NACK, después del último byte de la transmisión. En primer lugar, se configura el pin SDA como entrada ya que se van a recibir datos. Mediante un bucle se leen todos los bits empezando por el de mayor peso. Después de la lectura del byte se pone el pin SDA como salida para que el máster envíe el bit ACK o NACK según corresponda, a la vez que se generan los correspondientes pulsos de reloj. Por último, se retorna la variable “byte” que contiene la información leída del bus I2C.

```
unsigned char I2CGetByte(unsigned char ACK)
{
  unsigned char i, byte;

  LPC_GPIO0->FIODIR&=~(1<<SDA);
  for(i=0;i<8;i++)      {
        LPC_GPIO0->FIOSET=(1<<SCL);
      I2Cdelay();
      byte=byte<<1;
      if(LPC GPIO0->FIOPIN&(1<<SDA)) byte++;
      LPC_GPIO0->FIOCLR=(1<<SCL);
      I2Cdelay();
    }

  LPC_GPIO0->FIODIR|=(1<<SDA);
  if(ACK)LPC GPIO0->FIOSET=(1<<SDA);
    else LPC_GPIO0->FIOCLR=(1<<SDA);
  pulso_SCL();
  return (byte);
}
```

Código 11. Función I2CGetByte.

- *I2CSendStop():* se configura el pin SDA como salida para generar la condición de STOP. Primero se pone a nivel alto la línea SCL, se espera el retardo y después se sube la línea SDA para dejar el bus en reposo y que de esta forma esté disponible para otra transmisión.

```
void I2CSendStop(void)
{
  LPC_GPIO0->FIOCLR=(1<<SDA);
  I2Cdelay();
  LPC_GPIO0->FIOSET=(1<<SCL);
  I2Cdelay();
  LPC_GPIO0->FIOSET=(1<<SDA);
  I2Cdelay();
}
```

Código 12. Función I2CSendStop.

- *I2CWrite8():* función que se encarga de realizar el proceso de escritura de un byte (val) en el registro indicado (reg) del dispositivo del bus I2C (addr) deseado.

```
void I2CWrite8(uint8_t addr, uint8_t reg, uint8_t val)
{
    I2CSendAddr(addr, 0);
    I2CSendByte(reg);
    I2CSendByte(val);
    I2CSendStop();
}
```

Código 13. Función I2CWrite8.

- *I2CRead8()*: función que se encarga de realizar el proceso de lectura de un byte del registro indicado (reg) del dispositivo del bus I2C (addr) deseado.

```
uint8_t I2CRead8(uint8_t addr, uint8_t reg)
{
    uint8_t val;

    I2CSendAddr(addr, 0);
    I2CSendByte((uint8_t)reg);
    I2CSendAddr(addr, 1);
    val=I2CGetByte(1);
    I2CSendStop();

    return val;
}
```

Código 14. Función I2CRead8.

5.2.5. RTC.

Para tener la base de tiempo de envío de los datos se utiliza el RTC (Real Time Clock) que incorpora el microcontrolador LPC1768. Requiere de un oscilador externo de 32.768 kHz con el que se obtiene una base de tiempo de 1 Hz.

Este timer se utiliza para generar una interrupción cada diez minutos de forma que marca la periodicidad de envío de los datos sensados por en microcontrolador hacia la base de datos. Se utiliza este timer debido a que es el único timer del LPC1768 que sigue funcionando, aunque el microcontrolador este en modo de bajo consumo.

Para despertar al microcontrolador se programa una alarma mediante la comparación del valor actual del RTC con un registro configurado, si esto valores son iguales se produce una interrupción que despierta al micro para realizar las medidas de los parámetros climáticos y proceder a enviarlos al servidor.

Los pasos seguidos para poner en funcionamiento este periférico son:

- Habilitar su alimentación mediante los bits correspondientes del registro PCONP.
- Configurar el tiempo (hora, día, mes y año) inicial.
- Habilitar la cuenta mediante el registro CCR.
- Configurar la alarma.
- Habilitar la interrupción por alarma mediante el registro AMR.
- Habilitar la interrupción en el NVIC.

El fichero "*RTC.h*" contiene la siguiente información:

```
#ifndef __RTC_H
#define __RTC_H

extern int flag_rtc;

void RTC_Config(void);
void RTC_IRQHandler(void);

#endif
```

Código 15. Fichero RTC.h

El fichero "*RTC.c*" contiene las funciones de configuración del RTC y la función de atención a la interrupción:

- *RTC_Config():* se activa la alimentación del RTC con el bit correspondiente del registro PCONP. Se fija el tiempo actual a cero y posteriormente se fija el tiempo de los registros de alarma a los minutos que se le pasan como argumento a la función, de forma que interrumpa cada vez que la cuenta llegue a los minutos fijados. Se habilita la cuenta en el registro CCR y se habilitan la interrupción del RTC por alarmas y en el NVIC.

```
void RTC_Config(int minutos)
{
    LPC_SC->PCONP|= (1<<12);
    LPC_RTC->CCR=(1<<1);

    LPC_RTC->YEAR=0;
    LPC_RTC->MONTH=0;
    LPC_RTC->DOY=0;
    LPC_RTC->DOW=0;
    LPC_RTC->DOM=0;
```

```
    LPC_RTC->HOUR=0;
    LPC_RTC->MIN=0;
    LPC_RTC->SEC=0;

    LPC_RTC->CCR=(1<<4)|(1<<0);

    LPC_RTC->ALYEAR=0;
    LPC_RTC->ALMON=0;
    LPC_RTC->ALDOY=0;
    LPC_RTC->ALDOW=0;
    LPC_RTC->ALDOM=0;
    LPC_RTC->ALHOUR=0;
    LPC_RTC->ALMIN=minutos;
    LPC_RTC->ALSEC=0;

    LPC_RTC->CIIR=0;
    LPC_RTC->AMR=0;

    NVIC_EnableIRQ(RTC_IRQn);
    NVIC_SetPriority(RTC_IRQn, 1);

}
```

Código 16. Función RTC_Config.

- *RTC_IRQHandler():* función de atención a la interrupción provocada por el RTC. En primer lugar, se borra el flag de interrupción. Después se vuelve a configurar el RTC para que vuelva a interrumpir cada intervalo de tiempo configurado y por último se activa la variable "flag_rtc" para indicar que es el momento de realizar la adquisición de los datos de los sensores y enviarlos al servidor.

```
void RTC_IRQHandler(void)
{
    LPC_RTC->ILR|=(1<<1)|(1<<0);
    RTC_Config(minutos);
    flag_rtc=1;
}
```

Código 17. Función RTC_IRQHandler.

5.2.6. Modo de bajo consumo.

El microcontrolador LPC1768 dispone de varios modos de bajo consumo especificados en su hoja de características. Como la mayor parte del tiempo el sistema está en reposo, es decir, no realiza medidas con los sensores ni envía datos, es conveniente llevar el microcontrolador a un estado de baja energía para que su consumo sea el más bajo posible.

En este proyecto se utiliza en modo dormido (Sleep Mode) en el que el reloj del núcleo y la ejecución de instrucciones se paran hasta que se produce una interrupción o se produce un reset.

La forma más fácil de entrar a modo de bajo consumo es utilizar la instrucción en ensamblador WFI (Wait For Interrupt). Una vez el microcontrolador está dormido solo puede ser despertado mediante una interrupción. Cuando el microcontrolador ejecuta esta instrucción, deja de ejecutar más instrucciones y pasa a modo dormido.

La función que permite llamar a la instrucción en ensamblador WFI es:

```
__wfi();
```

Código 18. Función WFI.

Otra forma de reducir el consumo es deshabilitar la alimentación de los periféricos que no se vayan a utilizar, ya que algunos de ellos vienen activados por defecto tras un reset, para lo que se utiliza el registro PCONP, que está formado por 32 bits. Por tanto, se

deshabilitan los periféricos que no se van a utilizar en el proyecto. Si se fija el bit correspondiente a un periférico a 1, el periférico queda habilitado, mientras que, si se fija a 0, el periférico queda deshabilitado.

Una cuestión importante respecto al registro PCONP es que para trabajar con cualquier periférico es necesario habilitarlo inicialmente en este registro, sino cualquier lectura o escritura en el periférico no será válida.

A continuación, se muestran la relación de bits del registro PCONP asociados con su correspondiente periférico:

Bit	Symbol	Description	Reset value
0	-	Reserved.	NA
1	PCTIM0	Timer/Counter 0 power/clock control bit.	1
2	PCTIM1	Timer/Counter 1 power/clock control bit.	1
3	PCUART0	UART0 power/clock control bit.	1
4	PCUART1	UART1 power/clock control bit.	1
5	-	Reserved.	NA
6	PCPWM1	PWM1 power/clock control bit.	1
7	PCI2C0	The I^2C0 interface power/clock control bit.	1
8	PCSPI	The SPI interface power/clock control bit.	1
9	PCRTC	The RTC power/clock control bit.	1
10	PCSSP1	The SSP 1 interface power/clock control bit.	1
11	-	Reserved.	NA
12	PCADC	A/D converter (ADC) power/clock control bit. **Note:** Clear the PDN bit in the AD0CR before clearing this bit, and set this bit before setting PDN.	0
13	PCCAN1	CAN Controller 1 power/clock control bit.	0
14	PCCAN2	CAN Controller 2 power/clock control bit.	0
15	PCGPIO	Power/clock control bit for IOCON, GPIO, and GPIO interrupts.	1
16	PCRIT	Repetitive Interrupt Timer power/clock control bit.	0
17	PCMCPWM	Motor Control PWM	0
18	PCQEI	Quadrature Encoder Interface power/clock control bit.	0
19	PCI2C1	The I^2C1 interface power/clock control bit.	1
20	-	Reserved.	NA
21	PCSSP0	The SSP0 interface power/clock control bit.	1
22	PCTIM2	Timer 2 power/clock control bit.	0
23	PCTIM3	Timer 3 power/clock control bit.	0
24	PCUART2	UART 2 power/clock control bit.	0
25	PCUART3	UART 3 power/clock control bit.	0
26	PCI2C2	I2C interface 2 power/clock control bit.	1
27	PCI2S	I2S interface power/clock control bit.	0
28	-	Reserved.	NA
29	PCGPDMA	GPDMA function power/clock control bit.	0
30	PCENET	Ethernet block power/clock control bit.	0
31	PCUSB	USB interface power/clock control bit.	0

Figura 8. Registro PCONP.

5.3. Sensores utilizados.

La obtención de los datos físicos del lugar donde se emplace la estación meteorológica se realiza con sensores digitales y analógicos, los cuales serán descritos en los siguientes apartados con un enfoque hardware y software.

5.3.1. Sensor de temperatura.

Para medir la temperatura del lugar en el que se emplace el sistema completo, se utiliza el sensor MCP9808 del fabricante Microchip. Debido a que se quiere medir la temperatura ambiente, este sensor debe estar al aire libre, protegido con una estructura para que no lo deteriore o destruya las condiciones climatológicas adversas a las que pueda estar sometido. Este sensor viene integrado en un módulo, cuyo diagrama de conexionado se muestra a continuación:

Figura 10. Módulo MCP9808.

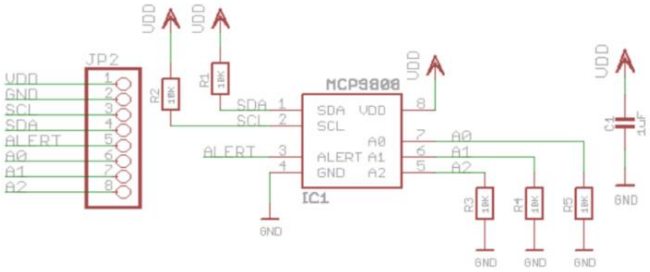

Figura 9. Esquemático del módulo MCP9808.

De este diagrama se pueden sacar varias conclusiones, el integrado principal tiene un condensador de desacoplo de la alimentación, dispone de dos resistencias de pull-up de 10 kΩ en las líneas de SDA y SCL, por tanto, su salida en reposo está fijada a nivel alto. Los pines A0, A1 y A2, están conectado mediante resistencias de 10 kΩ a masa, lo que indica que su entrada será masa, a no ser que se fije un valor de Vcc a través del conector, de forma que la dirección asociada a este dispositivo en el bus I2C es 0x18.

Este integrado, internamente utiliza un sensor band gap de temperatura. El funcionamiento de este tipo de sensores se basa en que la tensión directa de una unión P-N es dependiente de la temperatura cuya salida en voltaje es proporcional a la temperatura ambiente, este valor se introduce a un convertidor ADC, que proporciona una salida digital de 16 bits cargados en paralelo en el registro, de los cuales los 13 menos significativos contienen el valor digital de la temperatura ambiente en complemento A2.

Es un sensor que proporciona una salida digital, mediante el protocolo I2C. Las características más destacables de este sensor son:

- Alimentación de 2.7 V a 5.5 V.
- Resolución ajustable por el usuario.
- Precisión típica de ±0.25 en un rango de temperatura de -40ºC a 120ºC.
- Interfaz I2C de hasta 1 MHz.
- Corriente típica consumida durante medida de 200 µA.
- Corriente típica consumida en reposo 0.1 µA.

Este sensor tiene internamente 9 registros de 16 bits cada uno, divididos en 8 bits más significativos (MSB) y 8 bits menos significativos (LSB) excepto el último registro que está compuesto por 8 bits.

Register Pointer (Hex)	MSB/ LSB	Bit Assignment							
		7	6	5	4	3	2	1	0
0x00	MSB	0	0	0	0	0	0	0	0
	LSB	0	0	0	1	1	1	1	1
0x01	MSB	0	0	0	0	0	Hysteresis		SHDN
	LSB	Crt Loc	Win Loc	Int Clr	Alt Stat	Alt Cnt	Alt Sel	Alt Pol	Alt Mod
0x02	MSB	0	0	0	SIGN	2^{7}°C	2^{6}°C	2^{5}°C	2^{4}°C
	LSB	2^{3}°C	2^{2}°C	2^{1}°C	2^{0}°C	2^{-1}°C	2^{-2}°C	0	0
0x03	MSB	0	0	0	SIGN	2^{7}°C	2^{6}°C	2^{5}°C	2^{4}°C
	LSB	2^{3}°C	2^{2}°C	2^{1}°C	2^{0}°C	2^{-1}°C	2^{-2}°C	0	0
0x04	MSB	0	0	0	SIGN	2^{7}°C	2^{6}°C	2^{5}°C	2^{4}°C
	LSB	2^{3}°C	2^{2}°C	2^{1}°C	2^{0}°C	2^{-1}°C	2^{-2}°C	0	0
0x05	MSB	$T_A \geq T_{CRIT}$	$T_A > T_{UPPER}$	$T_A < T_{LOWER}$	SIGN	2^{7}°C	2^{6}°C	2^{5}°C	2^{4}°C
	LSB	2^{3}°C	2^{2}°C	2^{1}°C	2^{0}°C	2^{-1}°C	2^{-2}°C	2^{-3}°C	2^{-4}°C
0x06	MSB	0	0	0	0	0	0	0	0
	LSB	0	1	0	1	0	1	0	0
0x07	MSB	0	0	0	0	0	1	0	0
	LSB	0	0	0	0	0	0	0	0
0x08	LSB	0	0	0	0	0	0	1	1

Tabla 1. Mapa de registros del sensor MCP9808.

La mayoría de los bits que componen el mapa de registros están fijados a un valor (0 o 1), y otros no se usarán en este proyecto, por lo que a continuación se explican los registros que se utilizan:

- Del registro 0x01 se utiliza el bit 8 (SHDN) que controla el modo de funcionamiento del sensor, si se escribe un 0 estará alimentado para conversiones continuas y si se escribe un 1 el sensor entra en modo de bajo consumo.
- Del registro 0x05 se puede leer el dato digital en complemento A2 de la temperatura ambiente. Los bits 15 al 13 no se usan ya que proporcionan información sobre alertas de temperatura si el sensor se configura en modo comparador. El bit 12 (SIGN) proporciona información del signo de la temperatura medida, si es 0 la temperatura es mayor o igual que 0, y si es 1 la temperatura será menor que 0. Los bits 11 al 0 se denominan TA (Temperatura ambiente) y de ellos se obtiene el valor de la temperatura ambiente expresada en grados centígrados.
- Del registro 0x08 se utilizan los bits 1 y 0 para configurar la resolución de la medida de temperatura, que influye proporcionalmente en el tiempo de conversión de la medida realizada por el sensor. Los valores posibles junto con su resolución y tiempo de conversión son:

00 = +0.5°C (t_{CONV} = 30 ms typical)
01 = +0.25°C (t_{CONV} = 65 ms typical)
10 = +0.125°C (t_{CONV} = 130 ms typical)
11 = +0.0625°C (power-up default, t_{CONV} = 250 ms typical)

Figura 11. Bits de resolución del sensor MCP9808.

Para convertir los bits obtenidos de la lectura del registro 0x05 a la temperatura en número decimal, el fabricante proporciona una fórmula en la hoja de características:

$$Temperatura\ T_A \geq 0: T_A = UpperByte \cdot 16 + \frac{LowerByte}{16} \quad [2]$$

$$Temperatura\ T_A < 0: T_A = 256 - \left(UpperByte \cdot 16 + \frac{LowerByte}{16}\right) \quad [3]$$

Donde T_A es la temperatura en grados centígrados, UpperByte son los bits 15 al 8 del registro de temperatura y LowerByte son los bits del 7 al 0 del registro de temperatura.

Una vez estudiadas las características de este sensor se procede a explicar los archivos utilizados para poder leer el valor de la temperatura del entorno.

El fichero de cabecera "*MCP9808.h"* contiene la definición de la dirección I2C asociada al sensor MCP9808 y las funciones que se utilizan:

```
#ifndef __MCP9808_H
#define __MCP9808_H

#define MCP9808_ADDR 0x18

void MCP9808_WakeUp(void);
void MCP9808_ShutDown(void);
float MCP9808_Temp(void);

#endif
```

Código 19. Fichero MCP9808.h

Las funciones del fichero "*MCP9808.c*" utilizadas son:

- *MCP9808_WakeUp():* función encargada de despertar el sensor para poder realizar una medida. Consiste en escribir en el registro 0x01 todo ceros de forma que el bit SHDN se pone a cero para poder realizar la medida de temperatura.

```
void MCP9808_WakeUp(void)
{
    I2CWrite16(MCP9808_ADDR, 0x01, 0x0000);
    delay_1ms(250);
}
```

Código 20. Función MCP9808_WakeUp.

- *MCP9808_ShutDown():* función que lleva al sensor a modo de bajo consumo, lo cual se realiza escribiendo un 1 en el bit SHDN del registro 0x01.

```
void MCP9808_WakeUp(void)
{
    I2CWrite16(MCP9808_ADDR, 0x01, 0x0000);
    delay_1ms(250);
}
```

Código 21. Función MCP9808_ShutDown.

- *MCP9808_Temp():* función que se usa para leer los datos de temperatura proporcionados por el sensor mediante el bus I2C. Se leen del registro 0x05 se lee en primer lugar el byte más significativo (UpperByte) y después el menos significativo (LowerByte). Tras leerlos, se procede a comprobar si la temperatura en positiva o negativa. Por último, se retorna el valor de la temperatura.

```
float MCP9808_Temp(void)
{
    float temp;
    uint8_t UpperByte, LowerByte;

    I2CSendAddr(MCP9808_ADDR, 0);
```

```
        I2CSendByte(0x05);
        I2CSendAddr(MCP9808_ADDR, 1);
        UpperByte = I2CGetByte(0);
        LowerByte = I2CGetByte(1);
        I2CSendStop();

        UpperByte = UpperByte & 0x1F;
        if((UpperByte & 0x10) == 0x10)
        {
            UpperByte = UpperByte & 0x0F;
            temp=(256-((UpperByte * 16.0) + (LowerByte / 16.0)));
        }
        else
        {
        temp = ((UpperByte * 16.0) + (LowerByte / 16.0));
        }

        return temp;
}
```

Código 22. Función MCP9808_Temp.

5.3.2. Sensor de humedad.

La humedad relativa del entorno se mide con el sensor SHT31-D del fabricante Sensirion. La humedad relativa se define como la cantidad de vapor de agua que contiene el aire. Al igual que ocurre con el sensor de temperatura, este sensor debe estar al aire libre para realizar una correcta medida de la humedad ambiente.

Figura 12. Módulo SHT31-D.

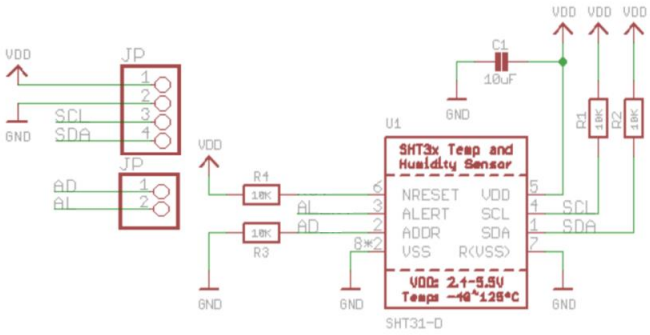

Figura 13. Esquemático del módulo SHT31-D.

Este módulo tiene los pines SDA y SCL para la comunicación I2C, los pines Vin y GND para la alimentación, el pin AD que sirve para establecer la dirección asociada a este sensor del bus I2C (por defecto está conectado a GND a través de una resistencia por lo que su dirección es 0x44, mientras que si se conecta a Vcc su dirección será 0x45) y el pin AI que sirve como pin de interrupción.

Sus principales características son:

- Alimentación de 2.4 V a 5.5 V.
- Interfaz I2C de hasta 1 MHz.
- Precisión típica de ±2%RH
- Resolución típica de 0.01%RH
- Rango de medida de 0%RH a 100%RH
- Corriente típica en reposo de 0.2 µA.
- Corriente típica durante una medida de 800 µA.

Para obtener la medida de humedad se utiliza el modo de adquisición de un solo disparo, con el que se obtiene una sola medida de temperatura y humedad, ambas de 16 bits, seguidas de un dato de 8 bits para comprobación. Los demás modos de adquisición que pueden ser configurados en este sensor se obviarán debido a que solo se usa el anterior. En la hoja de características del sensor se especifican los comandos a utilizar para el modo de adquisición de una sola medida junto con el diagrama de señales del bus I2C:

Condition		Hex. code	
Repeatability	Clock stretching	MSB	LSB
High	enabled	0x2C	06
Medium			0D
Low			10
High	disabled	0x24	00
Medium			0B
Low			16

Figura 14. Comandos modo de única medida.

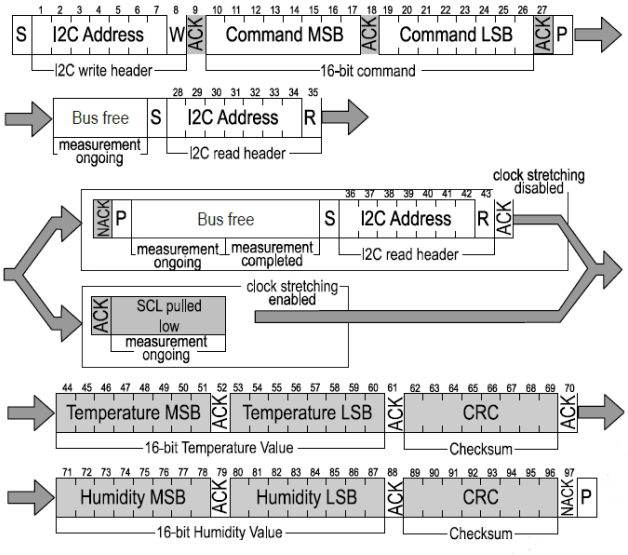

Figura 15. Diagrama de señales del modo de única medida.

Debido a que el orden de entrega de los datos de este sensor es primero la temperatura y después la humedad, es necesario leer ambos datos, aunque solo se vayan a usar los bits relativos al dato de humedad.

En este proyecto no se realizan comprobaciones con el bit de CRC, por lo que el fabricante recomienda leer los dos primeros bytes de temperatura junto con su CRC y posteriormente abortar la comunicación con un NACK tras leer los dos bytes de humedad.

Este sensor entra en modo de bajo consumo automáticamente después de realizar la adquisición de una medida de humedad.

La fórmula especificada en el datasheet del dispositivo para calcular la humedad relativa es:

$$RH = 100 \cdot \frac{S_{RH}}{2^{16} - 1} \qquad [4]$$

Donde SRH son los 16 bits obtenidos de la medida en el bus I2C.

Tras la explicación del funcionamiento del sensor SHT31D se procede a explicar los archivos utilizados en la programación para obtener el dato de la humedad.

El fichero de cabecera "*SHT31D.h"* contiene la definición de la dirección I2C asociada al sensor SHT31D y la función que se utiliza:

```
#ifndef __SHT31D_H
#define __SHT31D_H

#include <LPC17xx.h>

#define SHT31D_ADDR    0x44
```

```
float SHT31D_Hum(void);

#endif
```

Código 23. Fichero SHT31D.h

El fichero "*SHT31D.c*" contiene la función utilizada para leer la medida entregada por el sensor:

- *SHT31D_Hum():* en primer lugar, se envía en comando 0x2400 al sensor. Este comando, como se puede ver en la tabla de comandos, sirve para establecer la repetibilidad alta y deshabilitar la funcionalidad de "clock stretching". Posteriormente en un buffer de cinco elementos se almacenan los datos obtenidos del sensor a través de la comunicación I2C. Los bytes correspondientes a la medida de humedad son los almacenados en las posiciones tres y cuatro del buffer. Se alinean los bytes correspondientes y se procede a aplicar la formula descrita anteriormente para obtener el dato de la humedad relativa que es el argumento que retorna la función.

```
float SHT31D_Hum(void)
{
    uint8_t buffer[5];
    uint16_t SRH;
    float HR;

    I2CSendAddr(SHT31D_ADDR, 0);
    I2CSendByte(0x2400>>8);
    I2CSendByte(0x2400&0xFF);
    I2CSendStop();

    delay_1ms(100);

    I2CSendAddr(SHT31D_ADDR, 1);
    buffer[0] = I2CGetByte(0);
    buffer[1] = I2CGetByte(0);
    buffer[2] = I2CGetByte(0);
    buffer[3] = I2CGetByte(0);
    buffer[4] = I2CGetByte(1);
    I2CSendStop();

    SRH=buffer[3];
    SRH <<= 8;
    SRH |= buffer[4];

    HR = SRH;
    HR *= 100;
    HR /= 65535;

    return HR;
}
```

Código 24. Función SHT31D_Hum.

5.3.3. Sensor de luz ambiente.

Otro parámetro que se mide del entorno en el que se ubica la estación, es la luz ambiente, para ello se utiliza el sensor MAX44009 del fabricante Maxim integrated.

Figura 16. Módulo MAX44009.

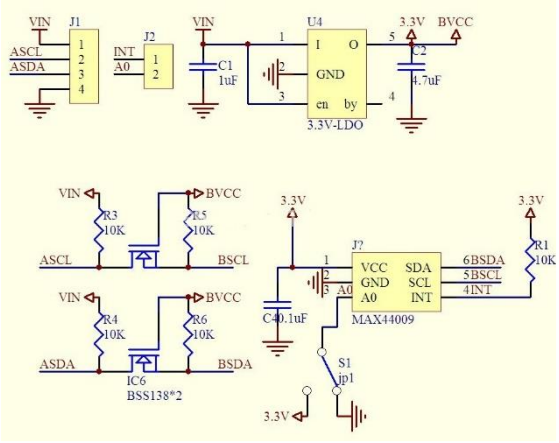

Figura 17. Esquemático del módulo MAX44009.

Este integrado, está compuesto internamente por un fotodiodo que genera una corriente eléctrica proporcional al nivel de luz captado del ambiente, la corriente se convierte en tensión mediante un conversor basado en un amplificador operacional para que esta tensión pueda ser introducida a un convertidor ADC, el cual su salida digital es transforma al protocolo I2C para poder ser leída por el microcontrolador.

Las principales características de este sensor son:

- Alimentación de 1.7V a 3.6V.
- Rango de temperatura de funcionamiento de -40ºC a 85ºC.
- Interfaz I2C.
- Rango de medida de 0.045 lux a 188000 lux.
- Corriente de operación típica de solo 0.65 µA.

La intensidad de la luz ambiente se mide en lux (lumen/m^2) que es una unidad de medida de la iluminancia.

La respuesta o sensibilidad de este sensor es muy parecida a la respuesta del ojo humano ante la luz ambiente, las cuales se pueden observar en la siguiente gráfica:

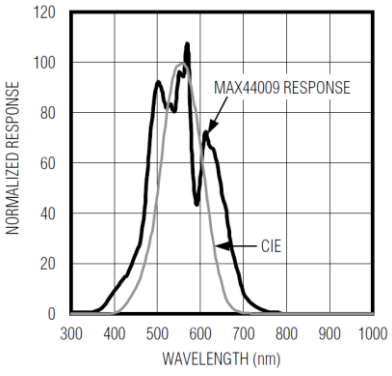

Figura 18. Respuesta del sensor MAX44009 vs ojo humano.

El ojo humano no es capaz de percibir las partes correspondientes al espectro de la radiación ultravioleta (<400 nm), ni la radiación infrarroja (>700 nm). También se observa que la máxima sensibilidad del ojo humano está aproximadamente en 555 nm, longitud de onda correspondiente a la luz verde, siendo menos sensible a las luces azul y roja.

En el momento de elegir este sensor hay que tener en cuenta los números límites de luxes que son necesarios captar para la aplicación en la que se utiliza. En este caso como se quiere medir la luz ambiente, el sensor puede estar al aire libre con incidencia directa de la luz solar, situación en la que se pueden alcanzar valores de 100000 lux.

El fabricante en su hoja de características recomienda usar el modo de alta resolución, ya que, de esta forma, el tiempo de medida o integración, es lo suficientemente largo para rechazar cualquier tipo de ruido que pueda afectar a la medida

El mapa de registros que posee este sensor es:

REGISTER	BIT								REGISTER ADDRESS	POWER-ON RESET STATE	R/W
	7	6	5	4	3	2	1	0			
STATUS											
Interrupt Status	—	—	—	—	—	—	—	INTS	0x00	0x00	R
Interrupt Enable	—	—	—	—	—	—	—	INTE	0x01	0x00	R/W
CONFIGURATION											
Configuration	CONT	MANUAL	—	—	CDR	TIM[2:0]			0x02	0x03	R/W
LUX READING											
Lux High Byte	E3	E2	E1	E0	M7	M6	M5	M4	0x03	0x00	R
Lux Low Byte	—	—	—	—	M3	M2	M1	M0	0x04	0x00	R
THRESHOLD SET											
Upper Threshold High Byte	UE3	UE2	UE1	UE0	UM7	UM6	UM5	UM4	0x05	0xFF	R/W
Lower Threshold High Byte	LE3	LE2	LE1	LE0	LM7	LM6	LM5	LM4	0x06	0x00	R/W
Threshold Timer	T7	T6	T5	T4	T3	T2	T1	T0	0x07	0xFF	R/W

Tabla 2. Mapa de registros del sensor MAX44009.

Los registros de estado por interrupción y los registros que configuran los límites de luxes para que ocurra la interrupción no se utilizan en el presente proyecto, por lo que no se explica su utilidad.

El registro de configuración (0x02) está compuesto por ocho bits, cuyas funcionalidades se explican a continuación:

- El bit 7 (CONT) sirve para configurar el modo de medida del sensor. Si se fija a 0, se realiza una sola medida, mientras que, si se fija a 1, se activa el modo de medida continuo.
- El bit 6 (MANUAL) se utiliza para configurar los bits CDR y TIM[2:0], es decir, si se fija a 0 dichos bits se determinan automáticamente por el circuito interno de auto rango, mientras que si se fija a 1 estos bits pueden ser configurados por el programador. En este proyecto se utilizará el modo de auto rango, debido a que el sensor estará expuesto a la luz ambiente que irá variando durante el transcurso de las horas del día.

En las direcciones 0x03 y 0x04 se encuentran los datos de la medida de luz ambiental, que están compuestos por bits de mantisa y bits de exponente. Para calcular la cantidad de lux se realiza con la siguiente fórmula:

$$Exponente = 8 \cdot E3 + 4 \cdot E2 + 2 \cdot E1 + E0 \quad [5]$$

$$Mantisa = 128 \cdot M7 + 64 \cdot M6 + 32 \cdot M5 + 16 \cdot M4 + 8 \cdot M3 + 4 \cdot M2 + 2 \cdot M1 + M0 \quad [6]$$

$$Lux = (2^{Exponente} \cdot Mantisa) \cdot 0{,}045 \quad [7]$$

A continuación, se explican los archivos utilizados en la programación para obtener el dato de la luz ambiente.

El fichero de cabecera "*MAX44009.h"* contiene la definición de la dirección I2C asociada al sensor MAX44009 y la función que se utiliza para obtener la medida:

```
#ifndef __MAX44009_H
#define __MAX44009_H

#include <LPC17xx.h>

#define MAX44009_ADDR    0x4A

float MAX44009_Lux(void);

#endif
```

Código 25. Fichero MAX44009.h

El fichero "*MAX44009.c*" contiene la función utilizada para leer la medida entregada por el sensor:

- *MAX44009_Lux():* en primer lugar, se configuran todos los bits del registro de configuración (0x02) a valor 0, de forma que se utiliza el modo de una sola medida y se utiliza en circuito de auto rango. Posteriormente, se leen los bytes de los registros 0x03 y 0x04 y se proceden a extraer los bits correspondientes al exponente y a la mantisa. Por último, se calcula el dato de lux obtenido en base a la fórmula indicada anteriormente, que es el dato que retorna la función.

```
float MAX44009_Lux(void)
{
    int exponente, mantisa;
    float lux;
    uint8_t lux_MSB, lux_LSB;

    //Configuración
    I2CWrite8(MAX44009_ADDR, 0x02, 0x00);

    //Leer registros
    I2CSendAddr(MAX44009_ADDR, 0);
    I2CSendByte(0x03);
    I2CSendAddr(MAX44009_ADDR, 1);
    lux_MSB = I2CGetByte(0);
    lux_LSB = I2CGetByte(1);
    I2CSendStop();

    exponente = (lux_MSB & 0xF0) >> 4;
    mantisa = ((lux_MSB & 0x0F) << 4);
    mantisa += lux_LSB & 0x0F;
    lux = mantisa * (1 << exponente) * 0.045;

    return lux;
}
```

Código 26. Función MAX44009_Lux.

5.3.4. Sensor de radiación ultravioleta.

El índice de radiación ultravioleta se mide con el sensor VEML6070 del fabricante Vishay Semiconductors con el que se puede medir la luz ultravioleta.

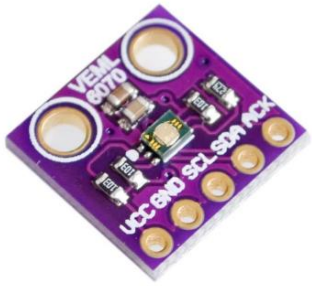

Figura 19. Módulo VEML6070.

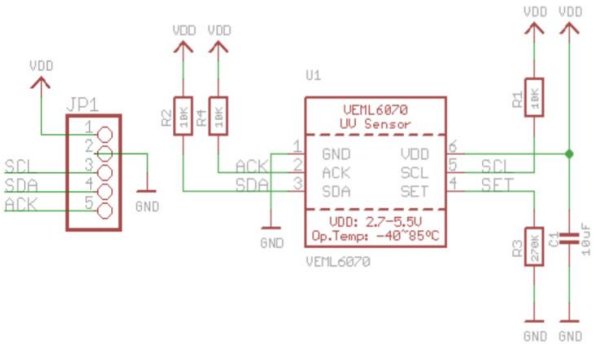

Figura 20. Esquemático del módulo VEML6070.

Este sensor incorpora un fotodiodo el cual es sensible a la luz ultravioleta, amplificadores, un ADC y la interfaz I2C. Dispone de un sensor de temperatura interno para compensar esta variable y que no afecte a la medida de la luz ultravioleta.

Las principales características son:

- Alimentación de 2.7V a 5.5V.
- Interfaz I2C.
- Temperatura de trabajo de -40ºC a 85ºC.
- Corriente típica de operación de 100 µA.
- Corriente típica en reposo de 1 µA.
- Sensibilidad máxima en la longitud de onda de 355 nm.

El módulo incorpora una resistencia RSET de valor 270 kΩ, la cual fija la configuración de la sensibilidad del sensor. La relación entre RSET y la sensibilidad a la luz solar ultravioleta es lineal.

El módulo también dispone de una salida ACK, la cual se activa al alcanzar un determinado valor de radiación ultravioleta configurable, esta característica no se utiliza en este proyecto.

Como este sensor se colocará en una caja protectora, esta debe dejar pasar la luz ultravioleta sin filtrarla para poder realizar una medida de forma correcta.

La salida de este sensor depende de la sensibilidad configurada con la resistencia RSET y el tiempo de lectura, con estas dos características se puede obtener los pasos equivalentes al índice UV.

Es necesario configurar el tiempo de integración, que es el tiempo durante el que el sensor realiza la medida. Para ello se configura el registro IT, que está formado por dos bits, IT1 e IT0. Este tiempo influye notablemente en la calidad de la medida, ya que, a mayor tiempo de integración, mayor tiempo está expuesto el fotodiodo a la radiación ultravioleta, con lo que se obtiene una medida de mayor calidad cuanto mayor sea este tiempo.

Este sensor tiene la posibilidad de configurarlo en modo de bajo consumo, para lo cual debe configurarse el registro SD.

El sensor utiliza la dirección 0x38 para comandos de escritura, y las direcciones 0x38 y 0x39 para comandos de lectura.

La radiación solar se puede dividir en longitudes de onda, las cuales determinan el tipo de radiación que emite el Sol. En la figura siguiente, se muestra dicha división:

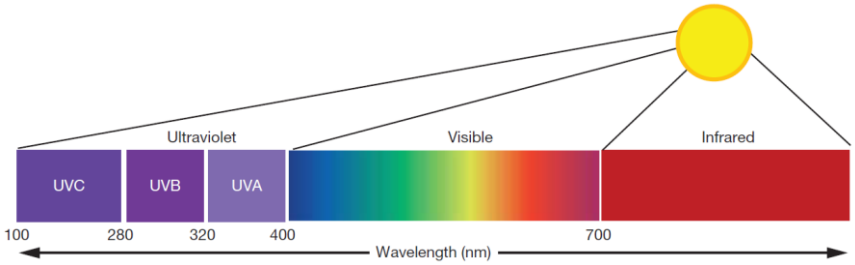

Figura 21. Espectro de radiación solar.

Como en este proyecto se quiere medir la radiación ultravioleta, el estudio se centra en este tipo de radiación, el cual se encuentra entre 100 nm y 400 nm.

La radiación UVC es bloqueada por la capa de ozono, por lo que no afecta a la medida de este sensor ya que no llega a la superficie terrestre.

Solamente un 0.1% de la energía solar que llega a la superficie terrestre es en forma de radiación UVB, mientras que la radiación que llega en forma de radiación UVA es del 4.9%.

El índice UV es una escala que sirve para relacionar el nivel de la radiación ultravioleta que llega a la superficie terrestre con el nivel de riesgo que conlleva.

Los valores de este índice van desde 0 hasta un valor de 11 o mayor, y está relacionado con la irradiancia (W/m2) como se muestra a continuación:

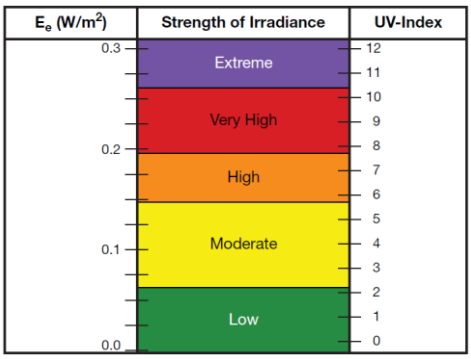

Figura 22. Relación entre irradiancia e índice UV.

Para relacionar la salida que proporciona el sensor con el índice UV, se utiliza la siguiente tabla:

UVI	R_{SET} = 270 kΩ; IT = 1T	R_{SET} = 270 kΩ; IT = 2T	R_{SET} = 270 kΩ; IT = 4T	UV-INDEX
≥ 11	≥ 2055	≥ 4109	≥ 8217	Extreme
8 to 10	1494 to 2054	2989 to 4108	5977 to 8216	Very High
6, 7	1121 to 1494	2242 to 2988	4483 to 5976	High
3 to 5	561 to 1120	1121 to 2241	2241 to 4482	Moderate
0 to 2	0 to 560	0 to 1120	0 to 2240	Low

Tabla 3. Relación entre RSET, tiempo de integración e índice ultravioleta.

El tiempo de integración es proporcional a la resistencia RSET, y viene dado por la siguiente gráfica que proporciona el fabricante:

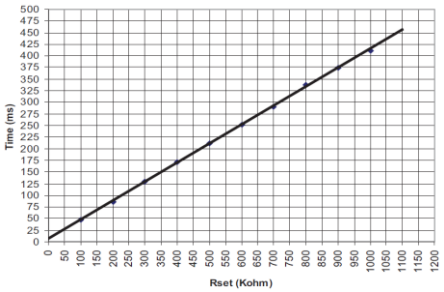

Figura 23. Relación entre Rset y el tiempo de integración.

En ella se puede observar que para un valor de RSET de 270 kΩ se corresponde con un tiempo de integración de 125 ms aproximadamente.

Tiene un registro de configuración con la siguiente estructura:

Reserved		ACK	ACK_THD	IT		Reserved	SD
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	0	ACK	THD	IT1	IT0	1	SD

Figura 24. Registro de configuración del sensor VEML6070.

Solamente se utiliza el bit 0 (SD), con el que se puede configurar el modo de funcionamiento del sensor, si se fija a 0 el sensor está listo para poder realizar una medida, mientras que, si se configura a 1, el sensor entra en modo de bajo consumo. También se utiliza el bit 2 (IT0) y el bit 3 (IT1), con los que se configura el tiempo de integración.

A continuación, se explican los archivos utilizados en la programación para obtener el dato del índice de radiación ultravioleta.

El fichero de cabecera "*VEML6070.h*" contiene la definición de la dirección I2C asociada al sensor VEML6070 y las funciones que se utilizan para obtener la medida:

```
#ifndef __VEML6070_H
#define __VEML6070_H

#define VEML6070_ADDRH 0x39
#define VEML6070_ADDRL 0x38

void VEML6070_Config(void);
int VEML6070_UV(void);
void VEML6070_ShutDown(void);

#endif
```

Código 27. Fichero VEML6070.h

El fichero "*VEML6070.c*" contiene las funciones utilizadas:

- *VEML6070_Config():* función encargada de configurar el sensor VEML6070. Escribe en su registro de configuración de 8 bits el dato 0x0A, con el que se fija a cero el bit SD, necesario para poder realizar una medida y se fija el tiempo de integración equivalente a un periodo.

```
void VEML6070_Config(void)
{
    I2CSendAddr(VEML6070_ADDRL, 0);
    I2CSendByte(0x0A);
    I2CSendStop();
    delay_1ms(250);
}
```

Código 28. Función VEML6070_Config.

- *VEML6070_ShutDown():* función encargada de llevar al sensor a modo de bajo consumo lo cual se realiza escribiendo un 1 en el bit SD del registro de configuración.

```
void VEML6070_ShutDown(void)
{
    I2CSendAddr(VEML6070 ADDRL, 0);
    I2CSendByte(0x0A|0x01);
    I2CSendStop();
}
```

Código 29. Función VEML6070_ShutDown.

- *VEML6070_UV():* función que calcula el índice de radiación ultravioleta, que es el valor que retorna la función. Para ello se obtienen los dos bytes del sensor a través del bus I2C y se alinean para obtener un dato de 16 bits. Este dato se divide entre el número 373.67 que equivale al número de pasos entre un índice y el siguiente para una Rset de 270 kΩ y un tiempo de integración de un periodo.

```
int VEML6070_UV(void)
{
    uint16_t uv;
    int uvi;

    I2CSendAddr(VEML6070_ADDRH, 1);
    uv=I2CGetByte(1);
    I2CSendStop();
    uv<<=8;
    I2CSendAddr(VEML6070_ADDRL, 1);
    uv|=I2CGetByte(1);
    I2CSendStop();
    uvi=uv/373.67;

    return uv;
}
```

Código 30. Función VEML6070_UV.

5.3.5. Sensor de presión atmosférica.

La presión atmosférica se mide con el sensor MPL3115A2 del fabricante NXP. La presión atmosférica o barométrica es la presión que ejerce el aire en un punto cualquiera de la atmosfera, depende ligeramente de las condiciones climatológicas y es inversamente proporcional al aumento de la altitud. Este dispositivo se basa en un sensor piezoresistivo que traduce las variaciones de presión en una tensión que es digitalizada con un ADC y mediante una interfaz I2C se lleva a sus pines de salida para que pueda ser leída por un microcontrolador.

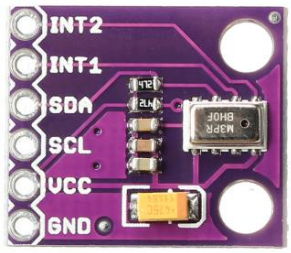

Figura 25. Módulo MPL3115A2.

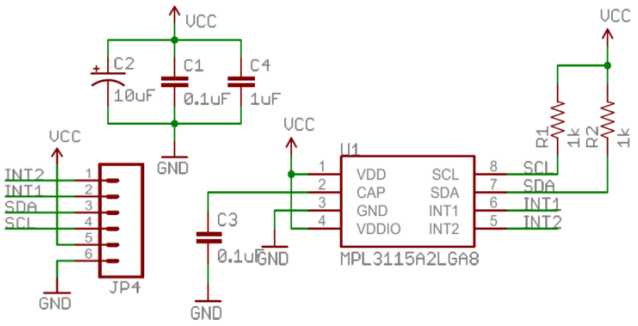

Figura 26. Esquemático del módulo MPL3115A2.

Sus principales características son:

- Rango de medidas de 20 kPa a 110 kPa
- Compensación del efecto de la temperatura mediante un sensor de temperatura interno.
- Precisión típica de ±0.05 kPa
- Alimentación 3.3 V
- ADC de 24 bits

Los registros que se utilizan son:

- CTRL_REG1 (0x26):

	7	6	5	4	3	2	1	0
R W	ALT	RAW	OS2	OS1	OS0	0 RST	OST	SBYB
Reset	0	0	0	0	0	0	0	0

Figura 27. Registro CTRL_REG1 del sensor MPL3115A2.

El bit ALT si está a 0 el sensor pasa a modo barómetro y si está a 1 pasa a modo altímetro.

Con el bit RAW activo, el dato leído es la salida del ADC sin procesamiento.

Con los bits OS[2:0] se selecciona la relación de sobremuestreo, a mayor sobremuestreo en menor medida afecta el ruido a la medida realizada.

Con el bit RST activado se produce un reset mediante software.

El bit OST se utiliza para hacer una sola medida. Si el sensor está en modo standby y se activa OST, se pasa a modo activo para realizar una medida y después se borra el bit OST y se vuelve a pasar a modo standby, todo ello automáticamente. Antes de escribir de nuevo en el bit OST se debe leer su valor.

Con el bit SBYB activo el sensor pasa a modo activo y si se fija a cero el sensor pasa a modo standby.

- OUT_P_MSB (0x01), OUT_P_CSB (0x02) y OUT_P_LSB (0x03): registros de ocho bits cada uno, de los que se obtiene la medida del sensor dependiendo del modo de uso del sensor (bit ALT). Como en este proyecto se usa el modo barómetro se obtendrá la medida en Pascales. El dato de presión se almacena como un entero de 18 bits formado por OUT_P_MSB, OUT_P_CSB y los bits 7 y 6 de OUT_P_LSB, mientras que los bits 5 y 6 de OUT_P_LSB son la parte fraccionaria.

A continuación, se explican los archivos utilizados en la programación para obtener el dato del índice de radiación ultravioleta.

El fichero de cabecera "*MPL3115A2.h*" contiene la definición de la dirección I2C asociada al sensor MPL3115A2 y las funciones que se utilizan para obtener la medida:

```
#ifndef __MPL3115A2_H
#define __MPL3115A2_H

#include <LPC17xx.h>

#define MPL3115A2_ADDR          0x60
#define MPL3115A2_CTRL_REG1     0x26
#define MPL3115A2_OUTP_MSB      0x01

void MPL3115A2_Standby(void);
void MPL3115A2_ModoBarometro(void);
void MPL3115A2_OneShot(void);
void MPL3115A2_Oversample(uint8_t OS);
float MPL3115A2_Presion(void);

#endif
```

Código 31. Fichero MPL3115A2.h

El fichero "*MPL3115A2.h*" está formado por las siguientes funciones:

- *MPL3115A2_Standby():* función encargada de llevar al sensor a modo Standby para que consuma menos energía. Para ello, se leen los ochos bits del registro CTRL_REG1 y se le enmascara el bit 0 (SBYB) a valor cero para posteriormente escribirlo en el registro para que tenga efecto.

```
void MPL3115A2_Standby(void)
{
    uint8_t temp_CTRL_REG1;
    temp_CTRL_REG1 = I2CRead8(MPL3115A2_ADDR, MPL3115A2_CTRL_REG1);
    temp_CTRL_REG1 &= ~(1<<0);
    I2CWrite8(MPL3115A2_ADDR, MPL3115A2_CTRL_REG1, temp_CTRL_REG1);
}
```

Código 32. Función MPL3115A2_Standby.

- *MPL3115A2_ModoBarometro():* procediendo de igual forma que en la función anterior se pone a cero el bit 7 (ALT) para que el sensor funcione en modo barómetro.

```
void MPL3115A2_ModoBarometro(void)
{
    uint8_t temp_CTRL_REG1;
  temp_CTRL_REG1 = I2CRead8(MPL3115A2_ADDR, MPL3115A2_CTRL_REG1);
  temp_CTRL_REG1 &= ~(1<<7); //Poner a 0 el bit ALT
  I2CWrite8(MPL3115A2_ADDR, MPL3115A2_CTRL_REG1, temp_CTRL_REG1);
}
```

Código 33. Función MPL3115A2_ModoBarometro.

- *MPL3115A2_OneShot ():* en la hoja de característica se especifica que antes de escribir un nuevo valor en el bit OST es necesario hacer una lectura del mismo. En concreto esta función pone el bit OST a 1 para que el sensor funcione en modo de única medida.

```
void MPL3115A2_OneShot(void)
{
  uint8_t temp_CTRL_REG1;
  temp_CTRL_REG1 = I2CRead8(MPL3115A2_ADDR, MPL3115A2_CTRL_REG1);
  temp_CTRL_REG1 &= ~(1<<1); //Poner a 0 el bit OST
  I2CWrite8(MPL3115A2_ADDR, MPL3115A2_CTRL_REG1, temp_CTRL_REG1);

  temp_CTRL_REG1 = I2CRead8(MPL3115A2_ADDR, MPL3115A2_CTRL_REG1);
  temp_CTRL_REG1 |= (1<<1); //Poner a 1 el bit OST
  I2CWrite8(MPL3115A2_ADDR, MPL3115A2_CTRL_REG1, temp_CTRL_REG1);
}
```

Código 34. Función MPL3115A2_OneShot.

- *MPL3115A2_Oversample():* lee el valor del registro CTRL_REG1 y pone a valor 0 los bits correspondientes a OS[2:0] con una máscara para conservar el valor de los demás bits del registro. Después, introduce en el dato de tipo byte el valor que se le pasa por parámetro a la función que se corresponde con la tasa de sobremuestreo para volver a escribir el nuevo valor en el registro.

```
void MPL3115A2_Oversample(uint8_t OS)
{
    uint8_t temp_CTRL_REG1;
  OS <<= 3; //Alinearlo en el bit 3
  temp_CTRL_REG1 = I2CRead8(MPL3115A2_ADDR, MPL3115A2_CTRL_REG1);
  temp_CTRL_REG1 &= 0xC7; //Poner a 0 los bits OS[2:0]
  temp_CTRL_REG1 |= OS; //Nuevo valor del registro CTRL_REG1
  I2CWrite8(MPL3115A2_ADDR, MPL3115A2_CTRL_REG1, temp_CTRL_REG1);
}
```

Código 35. Función MPL3115A2_Oversample.

- *MPL3115A2_Presion():* desde esta función se llama a las funciones que establecen el modo barómetro, tasa de sobremuestreo máxima y modo de medida única. Después se obtienen los bits de los registros OUT_P_MSB, OUT_P_CSB y OUT_P_LSB para calcular la parte entera y la parte decimal del valor de la presión atmosférica en unidades de kPascales.

```
float MPL3115A2_Presion(void)
{
    uint32_t pres_entera;
    float pres_decimal, pres;
    uint8_t OUT_P_MSB, OUT_P_CSB, OUT_P_LSB;

    MPL3115A2_ModoBarometro(); //Modo bar□ro
    MPL3115A2_Oversample(7); //Mḥma tasa de sobremuestreo
    MPL3115A2_OneShot(); //Realizar una medida

    //Leer registros OUTP_MSB, OUTP_CSB y OUTP_LSB
    I2CSendAddr(MPL3115A2_ADDR, 0);
    I2CSendByte(MPL3115A2_OUTP_MSB);
    I2CSendAddr(MPL3115A2_ADDR, 1);
    OUT_P_MSB = I2CGetByte(0);
    OUT_P_CSB = I2CGetByte(0);
    OUT_P_LSB = I2CGetByte(1);
    I2CSendStop();

    MPL3115A2_Standby();

    pres_entera = OUT_P_MSB<<16 | OUT_P_CSB<<8 |
OUT_P_LSB;
    pres_entera >>= 6; //Alinear bits a la derecha

    OUT_P_LSB &= 0x30; //Bit 5 y 4
```

```
        OUT_P_LSB >>= 4; //Alinear bits a la derecha
        pres_decimal = OUT_P_LSB/4.0;

        pres = pres entera + pres decimal;

        return pres;
}
```

Código 36. Función MPL3115A2_Presion.

5.3.6. Sensor de velocidad del viento.

La lectura de la velocidad del viento se realiza mediante un anemómetro del fabricante PCE. Este sensor está formado por tres cazoletas, que al incidir el viento sobre ellas giran, lo cual hace que gire un imán que lleva incorporado en la parte superior del anemómetro y este activa un interruptor magnético cuando pasa sobre él. Se ha añadido un circuito externo al interruptor magnético para evitar los rebotes en las transiciones del mismo.

Figura 29. Anemómetro.

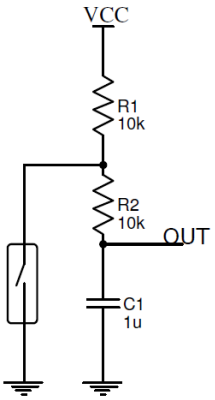

Figura 28. Circuito anti rebotes anemómetro.

El interruptor magnético tiene un funcionamiento muy sencillo, si está en presencia de un campo magnético se cierra el interruptor, mientras que si no hay un campo magnético presente el interruptor magnético está en circuito abierto.

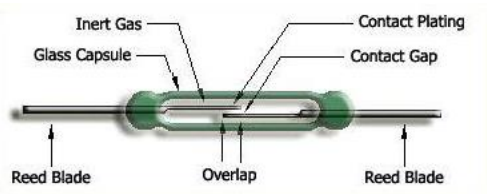

Figura 30. Interruptor magnético.

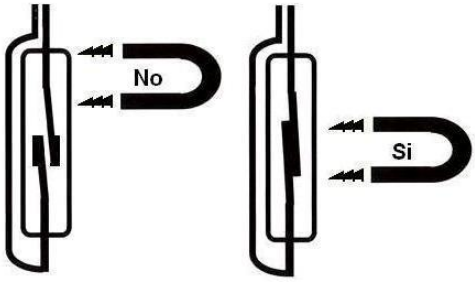

Figura 31. Funcionamiento interruptor magnético.

Una forma de cuantificar la velocidad del viento es la escala de Beaufort que indica la relación entre las velocidades del viento y los posibles efectos que pueden producir:

Nº de Beaufort	Velocidad del viento (km/h)	Denominación	Efectos en tierra
0	0 a 1	**Calma**	Calma, el humo asciende verticalmente
1	2 a 5	**Ventolina**	El humo indica la dirección del viento

2	6 a 11	**Flojito** (Brisa muy débil)	Se caen las hojas de los árboles, empiezan a moverse los molinos de los campos
3	12 a 19	**Flojo** (Brisa Ligera)	Se agitan las hojas, ondulan las banderas
4	20 a 28	**Bonancible** (Brisa moderada)	Se levanta polvo y papeles, se agitan las copas de los árboles
5	29 a 38	**Fresquito** (Brisa fresca)	Pequeños movimientos de los árboles, superficie de los lagos ondulada
6	39 a 49	**Fresco** (Brisa fuerte)	Se mueven las ramas de los árboles, dificultad para mantener abierto el paraguas
7	50 a 61	**Frescachón** (Viento fuerte)	Se mueven los árboles grandes, dificultad para caminar contra el viento
8	62 a 74	**Temporal** (Viento duro)	Se quiebran las copas de los árboles, circulación de personas muy difícil, los vehículos se mueven por sí mismos.
9	75 a 88	**Temporal fuerte** (Muy duro)	Daños en árboles, imposible caminar con normalidad. Se empiezan a dañar las construcciones. Arrastre de vehículos.
10	89 a 102	**Temporal duro** (Temporal)	Árboles arrancados, daños en la estructura de las construcciones. Daños mayores en objetos a la intemperie.
11	103 a 117	**Temporal muy duro** (Borrasca)	Destrucción en todas partes, lluvias muy intensas, inundaciones muy altas. Voladura de personas y de otros muchos objetos.
12	más de 118	**Temporal huracanado** (Huracán)	Voladura de vehículos, árboles, casas, techos y personas. Puede generar un huracán o tifón

Tabla 4. Escala de Beaufort.

Según el circuito anti rebotes diseñado, la salida OUT en reposo será Vcc mientras que cuando el imán pase por encima del interruptor magnético se obtienen cero voltios en la salida OUT. Por tanto, en la salida OUT del circuito anti rebotes, se obtienen dos pulsos a nivel bajo por cada vuelta que da la parte superior del anemómetro cuando el viento incide sobre las cazoletas y sabiendo que el radio desde el centro hasta las cazoletas mide 0.09 metros se puede calcular la velocidad del viento con la siguiente formula:

$$v = \frac{longitud\ de\ una\ vuelta}{tiempo\ en\ dar\ una\ vuelta} = \frac{2 \cdot \pi \cdot R}{T} = 2 \cdot \pi \cdot R \cdot (f/2) =$$

$$= \pi \cdot 0.09 \cdot f\ [m/s] = \pi \cdot 0.09 \cdot 3.6 \cdot f\ [km/h] \quad [8]$$

A continuación, se explican los archivos utilizados en la programación para obtener el dato del índice de la velocidad del viento.

El fichero de cabecera "*ANEMOMETRO.h*" contiene las funciones que se utilizan para obtener la medida:

```
#ifndef __ANEMOMETRO_H
#define __ANEMOMETRO_H

extern float velocidad;
```

```
void config_TIMER2(void);
void vel_ANEMOMETRO(void);

#endif
```

Código 37. Fichero ANEMOMETRO.h

El fichero "*ANEMOMETRO.c*" contiene la descripción de las funciones utilizadas:

- *Config_TIMER2():* función encargada de configurar el Timer 2. Habilita la alimentación del TIMER2 en el registro PCONP. Mediante el registro PINSEL se configura el pin P0.4 como entrada CAP2.0. El prescaler del timer se fija a cero. Por último, se habilita la interrupción del Timer2 en el NVIC y se le asigna una prioridad.

```
void config_TIMER2(void)
{
    LPC_SC->PCONP|=(1<<22);
    LPC_PINCON->PINSEL0|=(3<<8);

    LPC_TIM2->PR = 0x0000;
    LPC_TIM2->MCR = 0x0000;
    LPC_TIM2->CCR = 0x0005;
    LPC_TIM2->EMR = 0x0000;
    LPC_TIM2->TCR = 0x01;

    NVIC_SetPriority(TIMER2_IRQn, 2);
    NVIC_EnableIRQ(TIMER2_IRQn);
}
```

Código 38. Función config_TIMER2.

- *TIMER2_IRQHandler():* rutina de atención a la interrupción provocada por el Timer 2. En primer lugar, se borra el flag de interrupción. En el registro CR0 se encuentra el valor del contador TC, por tanto, para saber el número de cuentas entre dos flancos de subida se almacena la captura y se resta de la anterior. La frecuencia viene dada por la división de la frecuencia de funcionamiento del Timer 2 entre el número de cuentas entre dos flancos. Por último, se calcula la velocidad aplicando la fórmula mencionada anteriormente y se descartan posibles valores de error.

```
void TIMER2_IRQHandler(void)
{
    static uint32_t  captura;
    double frec;
    float vel;

    LPC_TIM2->IR|= (1<<4);
    N=LPC_TIM2->CR0-captura;
    frec=F_pclk/N;
    captura=LPC_TIM2->CR0;

    vel=(1.018*frec); // pi*0.09*3.6=1.018

    if((vel>1)&&(vel<100))
    {
        velocidad=vel;
    }
    else velocidad=0;
}
```

Código 39. Función TIMER2_IRQHandler.

5.3.7. Sensor de dirección del viento.

La dirección del viento se mide con una veleta del fabricante PCE, cuya salida es analógica. Su funcionamiento es similar al del anemómetro, con la salvedad que en este sensor lleva incluidas unas resistencias junto con varios interruptores magnéticos en su interior, que junto con unos componentes exteriores dará a su salida un voltaje determinado dependiendo de la posición de la veleta.

Figura 32. Veleta.

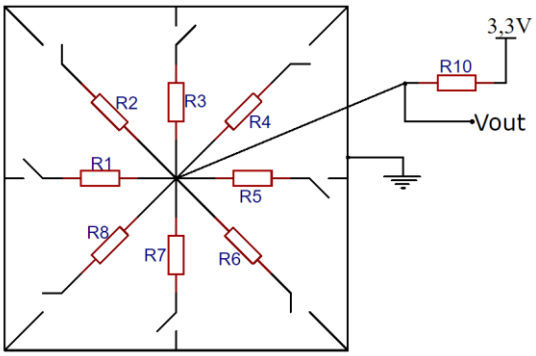

Figura 33. Esquemático interno de la veleta.

La tensión de salida (Vout) que se mide con el ADC del microcontrolador es la resultante de un divisor resistivo formado por la resistencia (Rx) que cierra el circuito cuando un interruptor magnético se cierra en presencia de un campo magnético y la resistencia R10 externa, cuyo valor es de 10 kΩ. Con estos datos se obtiene la tensión de salida que determina la dirección a la que apunta la veleta, que es la dirección del viento.

$$V_{out} = V_{cc} \cdot \frac{Rx}{Rx+R10} = 3.3\ V \cdot \frac{Rx}{Rx+10\ k\Omega} \qquad [9]$$

Aplicando la fórmula anterior a las distintas resistencias se obtiene el valor del voltaje de salida de la veleta, resumido en la tabla siguiente:

Resistencia	Vout
R1=33 kΩ	2.53 V
R2=8.2 kΩ	1.49 V
R3=1 kΩ	0.30 V
R4=2.2 kΩ	0.60 V
R5=3.9 kΩ	0.93 V
R6=16 kΩ	2.03 V
R7=120 kΩ	3.05 V
R8=64.9 kΩ	2.86 V

Tabla 5. Resistencia interna vs voltaje se salida.

También hay que tener en cuenta que el dato obtenido de este sensor no proporciona una medida válida si no hay viento, por lo que se ve condicionado a la medida tomada por el anemómetro, es decir, si el dato de velocidad del viento obtenido por el anemómetro es de 0 km/h no se tendrá en cuenta la dirección del viento.

La veleta descrita anteriormente tiene una problemática para medir la dirección del viento de forma autónoma debido a que su salida no está relacionada con ningún punto cardinal (Norte, Sur, Este u Oeste) de referencia porque el módulo se limita a entregar una tensión a su salida en función del lugar a donde apunte la veleta. De forma que el usuario tendría que ubicar la veleta respecto a una posición geográfica concreta que tendría que estar definida en el manual de usuario.

Para solucionar la problemática anterior se utiliza el sensor HMC5883L del fabricante Honeywell junto con la veleta. Este chip está compuesto internamente por tres sensores magnetoresistivos, distribuidos en los ejes X, Y, Z, de forma que mediante una operación trigonométrica se puede saber su orientación respecto al campo magnético de la Tierra y así obtener su orientación magnética.

Figura 35. Módulo HMC5883L.

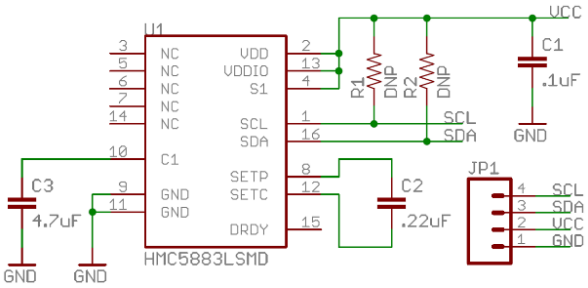

Figura 34. Esquemático del módulo HMC5883L.

En concreto, se pone el eje x del sensor HMC5883L en la misma dirección de la punta de la veleta, de esta forma se tiene una referencia, ya que se va a medir la orientación magnética del Norte respecto del eje X, por tanto, la punta de la veleta es la que marca la referencia. El norte magnético, que es el que se va a medir en este proyecto es distinto al norte geográfico, el cual viene cuantificado por la declinación magnética que es el ángulo que forman dichas orientaciones y depende de la ubicación en la Tierra.

Las principales características de este sensor son:

- Rango de medida de campo magnético de -8 Gauss a 8 Gauss.
- Interfaz I2C.
- Precisión de 1º a 2º.
- Alimentación de 2.16 V a 3.6 V.
- Rango de temperatura de trabajo de -30ºC a 85ºC.
- Consumo de corriente en reposo de 1 µA.
- Consumo de corriente en medida de 100 µA.
- ADC interno de 12 bits.

Este sensor dispone de tres modos de funcionamiento, modo continuo de medida, modo de única medida (por defecto) y modo reposo. La dirección I2C de 7 bits de este sensor es 0x1E.

El mapa de registros internos del sensor es:

Address Location	Name	Access
00	Configuration Register A	Read/Write
01	Configuration Register B	Read/Write
02	Mode Register	Read/Write
03	Data Output X MSB Register	Read
04	Data Output X LSB Register	Read
05	Data Output Z MSB Register	Read
06	Data Output Z LSB Register	Read
07	Data Output Y MSB Register	Read
08	Data Output Y LSB Register	Read
09	Status Register	Read
10	Identification Register A	Read
11	Identification Register B	Read
12	Identification Register C	Read

Tabla 6. Mapa de registros del sensor HMC5883L.

El registro de configuración A (0x00) está formado por ocho bits dejando todos sus valores a los fijados por defecto.

El registro de configuración B (0x01) está formado por ocho bits, de los cuales, los tres bits más significativos (GN[2:0]) sirven para configurar la ganancia del sensor. Este parámetro está relacionado con la resolución y el fabricante indica el rango de campo magnético adecuado a medir para cada valor de ganancia según la siguiente figura:

GN2	GN1	GN0	Recommended Sensor Field Range	Gain (LSb/ Gauss)	Digital Resolution (mG/LSb)	Output Range
0	0	0	± 0.88 Ga	1370	0.73	0xF800–0x07FF (-2048–2047)
0	0	1	± 1.3 Ga	1090 (default)	0.92	0xF800–0x07FF (-2048–2047)
0	1	0	± 1.9 Ga	820	1.22	0xF800–0x07FF (-2048–2047)
0	1	1	± 2.5 Ga	660	1.52	0xF800–0x07FF (-2048–2047)
1	0	0	± 4.0 Ga	440	2.27	0xF800–0x07FF (-2048–2047)
1	0	1	± 4.7 Ga	390	2.56	0xF800–0x07FF (-2048–2047)
1	1	0	± 5.6 Ga	330	3.03	0xF800–0x07FF (-2048–2047)
1	1	1	± 8.1 Ga	230	4.35	0xF800–0x07FF (-2048–2047)

Tabla 7. Configuración de la ganancia del sensor HMC5883L.

Como el campo magnético terrestre en la superficie está comprendido en el rango de 0.25G a 0.65G dependiendo del lugar en el que se mida. Se opta por elegir la ganancia que viene configurada por defecto (1090) ya que comprende el rango que se quiere medir con una buena resolución, de -1.3 Gauss a 1.3 Gauss.

El registro de modo de funcionamiento (0x02) permite configurar el modo de operación del sensor mediante sus dos bits menos significativos (MD[1:0]). Si se fijan a 00 el sensor funciona en modo de medida continua. Si se fijan a 01 funciona en modo de única medida y para el resto de valores el sensor entre en modo de reposo. En modo de una sola medida el sensor vuelve a reposo automáticamente tras realizar la medida.

Los registros de salida de las medidas en los diferentes ejes están formados por dos registros de 8 bits cada uno. Solamente se utilizan los valores obtenidos de los ejes X e Y, de los que se puede obtener la orientación respecto al Norte magnético del eje X.

$$\theta = \tan^{-1}\left(\frac{my}{mx}\right)\ [rad] \qquad [10]$$

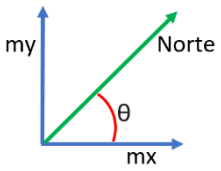

Figura 36. Norte magnético respecto del eje X.

El valor medido por este sensor es correcto si cerca de él no se encuentra ningún elemento que pueda alterar dicho campo, como un metal o un imán. En concreto en este proyecto se va a ubicar dentro de la carcasa de plástico que compone la veleta, por lo que no habrá ningún problema.

Tras la explicación de los dos sensores que se utilizan conjuntamente para medir la dirección del viento respecto al Norte magnético, se procede a explicar los archivos que se utilizan para obtener dicho dato.

En el fichero *"veleta.h"* se encuentran la declaración de las funciones utilizadas:

```
#ifndef __Veleta_H
#define __Veleta_H

#define HMC5883L_ADDR 0x1E

extern float voltios;

void Config_ADC(void);
float Veleta_Voltios(void);
void HMC5883L_PowerDown(void);
float HMC5883L_Angulo(void);
float Angulo_Norte(void);

#endif
```

Código 40. Fichero veleta.h

En el fichero *"veleta.c"* se describen las funciones:

- *Config_ADC():* función que se encarga de configurar el ADC del LPC1768 para medir el voltaje que entrega la veleta. Se habilita la alimentación en el registro PCONP. Se configura el pin P0.23 como AD0.0 y se deshabilitan las resistencias de pull-up y pull-down del pin

```
void Config_ADC(void)
{
    LPC_SC->PCONP|= (1<<12);
```

```
    LPC_PINCON->PINSEL1|= (1<<14);
    LPC_PINCON->PINMODE1|= (2<<14);
    LPC_SC->PCLKSEL0|= (0x00<<8);
    LPC_ADC->ADCR= (0x01<<0)|
                   (0x01<<8)|
                   (0x01<<21); // PDN=1
}
```

Código 41. Función Config_ADC.

- *Veleta_Voltios():* hace la lectura de los voltios que entrega la salida de la veleta, que es el dato que retorna la función.

```
float Veleta_Voltios(void)
{
    LPC_ADC->ADCR&=~(7<<24);
    LPC_ADC->ADCR|=(1<<24);
    voltios= ((LPC_ADC->ADGDR >>4)&0xFFF)*3.3/4095.0;

    LPC_ADC->ADCR&=~(7<<24);
    LPC_ADC->ADCR|=(1<<24);

    return voltios;
}
```

Código 42. Función Veleta_Voltios.

- *HMC5883L_PowerDown():* función que cambia a modo reposo el sensor HMC5883L, mediante la escritura en los bits de MD[1:0] = 10.

```
void HMC5883L_PowerDown(void)
{
    I2CWrite8(HMC5883L_ADDR, 0x02, 0x02);
}
```

Código 43. Función HMC5883L_PowerDown.

- *HMC5883L_Angulo():* calcula el ángulo del eje X respecto del Norte magnético. En primer lugar, se configura el modo de única medida fijando los bits MD[1:0]=01. Después, se leen los seis bytes correspondientes al campo magnético en los tres ejes. Se calcula la arcotangente del dato del eje Y entre el dato del eje X, obteniendo el ángulo en radianes. Se pasa el ángulo a grados y se comprueba que esté sea mayor de 0º, sino habrá que sumarle 360º.

```
float HMC5883L_Angulo(void)
{
    int16_t mx, my, mz;
    float angulo;

    I2CWrite8(HMC5883L_ADDR, 0x02, 0x01);

    //Leer datos
    I2CSendAddr(HMC5883L_ADDR, 0);
    I2CSendByte(0x03);
    I2CSendStop();

    I2CSendAddr(HMC5883L_ADDR, 1);
    mx = I2CGetByte(0)<<8;
    mx |= I2CGetByte(0);

    mz = I2CGetByte(0)<<8;
    mz |= I2CGetByte(0);

    my = I2CGetByte(0)<<8;
    my |= I2CGetByte(1);
    I2CSendStop();
```

```
  angulo = atan2(my, mx);
angulo = angulo * 57.2958; // 1rad=57.2958

  if(angulo < 0) angulo = angulo + 360;

  return angulo;
}
```

Código 44. Función HMC5883L_Angulo.

- *Angulo_Norte():* función que asocia el valor obtenido por el ADC y el valor del sensor HMC5883L para obtener la dirección real del viento. Como el eje X del sensor HMC5883L se encuentra apuntando a la dirección de la veleta que a su salida se obtienen 2.03 V se pueden relacionar ambos datos y obtener la dirección del viento respecto al Norte magnético.

```
float Angulo_Norte(void)
{
    float angulo, Angulo_Norte, voltios_ADC;

    angulo=HMC5883L_Angulo();
    voltios_ADC=Veleta_Voltios();
    if(voltios_ADC>1.93 && voltios_ADC<2.13) Angulo_Norte=angulo; // 2.03 V
    if(voltios_ADC>0.83 && voltios_ADC<1.03) Angulo_Norte=angulo+315; // 0.93 V
    if(voltios_ADC>0.5 && voltios_ADC<0.7) Angulo_Norte=angulo+270; // 0.6 V
    if(voltios_ADC>0.2 && voltios_ADC<0.4) Angulo_Norte=angulo+225; // 0.3 V
    if(voltios_ADC>1.39 && voltios_ADC<1.59) Angulo_Norte=angulo+180; // 1.49 V
    if(voltios_ADC>2.44 && voltios_ADC<2.64) Angulo_Norte=angulo+135; // 2.54 V
    if(voltios_ADC>2.76 && voltios_ADC<2.96) Angulo_Norte=angulo+90; // 2.86 V
    if(voltios_ADC>2.96 && voltios_ADC<3.15) Angulo_Norte=angulo+45; // 3.05 V
    if(Angulo_Norte>360)Angulo_Norte=Angulo_Norte-360;

    return Angulo_Norte;
}
```

Código 45. Función Angulo_Norte.

5.3.8. Pluviómetro.

La cantidad de agua precipitada se mide con un pluviómetro del fabricante PCE. La precipitación se mide en mm que equivale a metros de agua por metro cuadrado de terreno.

Su funcionamiento se basa en un recipiente con un hueco por el que cae el agua hacia un balancín el cual tiene a cada lado del eje un recipiente con una capacidad determinada, que, al llenarse, cambia de posición accionando, a través de un imán, un interruptor magnético produciendo un cambio en su salida.

En este caso cada vez que se llena una parte del recipiente, el balancín se mueve de posición lo que provoca un cambio en la salida debido a que el imán ha pasado por encima del interruptor magnético. La capacidad de cada parte que compone el balancín es de 0.2794 mm, por tanto, un cambio de estado a la salida de este sensor implica que ha caído una precipitación de 0.2794 mm. Para evitar los rebotes en la señal de salida que produce el interruptor magnético (lo que provocaría errores en la medición) se utiliza el mismo circuito anti rebotes diseñado para el anemómetro.

Figura 38. Pluviómetro.

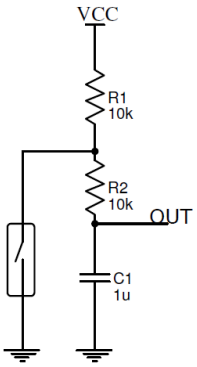

Figura 37. Circuito anti rebotes pluviómetro.

Debido a que la mayor parte del tiempo el microcontrolador estará en estado de bajo consumo, lo que implica que solo saldrá de este estado si se produce una interrupción. Por tanto, se utiliza como entrada de interrupción al microcontrolador la salida de este sensor, de forma que cuando se llene de agua una parte del balancín se producirá una interrupción que despertará al microcontrolador para poder almacenar la medida y proceder a enviarla cuando se produzca la interrupción generada por el RTC, tras la interrupción y almacenaje del valor se vuelve a dormir el microcontrolador.

Una vez definido el funcionamiento del pluviómetro se procede a explicar los archivos necesarios para conseguir realizar la medida del agua precipitada.

El archivo "*Pluviometro.h*" contiene la declaración de las funciones utilizadas:

```
#ifndef __Pluviometro_H
#define __Pluviometro_H

extern int lluvia_enviada;
extern float lluvia;

void EINTs_Config(void);
void
EINT2_IRQHandler(void);

#endif
```

Código 46. Fichero Pluviometro.h

El archivo "*Pluviometro.c*" contiene la descripción de las funciones utilizadas:

- *EINTs_Config():* se configura el pin P2.12 como entrada de interrupción externa (EINT2) y por flanco de bajada. Por último, se le asigna una prioridad y se habilita la interrupción en el NVIC.

```
void EINTs_Config(void)
{
    LPC_PINCON->PINSEL4|=(0x01<<24);
    LPC_SC->EXTMODE|=(1<<2);
    LPC_SC->EXTPOLAR=0;
    NVIC_SetPriority(EINT2_IRQn, 0x3);

    NVIC_EnableIRQ(EINT2_IRQn);
}
```

Código 47. Función EINTs_Config.

- *EINT2_IRQHandler():* es la función de atención a la interrupción de la EINT2. Al entrar en ella, se borra el flag de interrupción y se procede a comprobar el valor de la variable "*lluvia_enviada*", si esta variable vale 1 significa que se ha enviado el valor de la precipitación obtenida en el último rango de tiempo. Mientras que, si tiene valor 0, se debe ir acumulando el valor de las medidas para proceder a su envío la próxima vez que interrumpa el RTC.

```
void EINT2_IRQHandler(void)
{
    LPC_SC->EXTINT=(1<<2);
    if(lluvia_enviada==1)
    {
        lluvia_enviada=0;
        lluvia=0;
    }
  if(lluvia_enviada==0)
    {
        lluvia=lluvia+0.2794;
    }
}
```

Código 48. Función EINT2_IRQHandler.

5.3.9. Sensor de inclinación.

Como medida de seguridad ante posibles robos del sistema o para detectar una caída del sistema debido a fenómenos meteorológicos muy desfavorables se utiliza un sensor de inclinación. Su funcionamiento es sencillo, está compuesto por una bola de mercurio en su interior que se encarga de abrir (no existe contacto entre pines) o cerrar el circuito (la bola de mercurio pone en contacto los pines), dependiendo de la inclinación del mismo.

Figura 39. Sensor de inclinación.

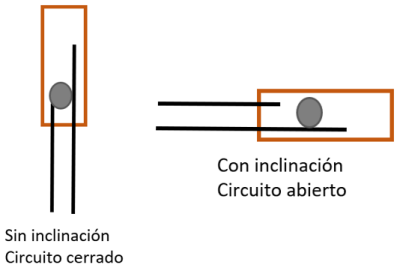

Figura 40. Funcionamiento del sensor de inclinación.

La obtención de la medida de este sensor se realiza conectado una de sus salidas a una entrada de interrupción del microcontrolador con resistencia de pull-up activada y la otra patilla a masa, de forma que si el sensor se inclina llegaría un flanco de subida al pin del microcontrolador que es el evento a detectar. Si se detecta dicho evento se entra en un estado de alarma.

5.4. Módulos de comunicación.

Para el envío de los datos hacia el servidor se puede optar por dos tipos de comunicación, mediante una comunicación WiFi si se tiene acceso a un punto WiFi o mediante una conexión GPRS si el sistema se encuentra en un lugar remoto que tenga cobertura telefónica.

El envío de los datos se hace mediante el método POST del protocolo HTTP. El mensaje para enviar los datos hacia el servidor debe tener la siguiente estructura:

POST datos.php HTTP/1.1\r\n
Host: tfgsergio.esy.es\r\n
*Accept: */*\r\n*
Content-Length: longitud\r\n
Content-Type: application/x-www-form-urlencoded\r\n
t=valor1&h=valor2&l=valor3&uv=valor4&p=valor5&lat=valor6&long=valor7&d=valor8&v=valor9&plu=valor10\r\n

Donde “longitud” es el tamaño de caracteres a enviar y “valorx” es el valor correspondiente a cada parámetro medido.

También se podrían enviar los datos mediante el método GET de HTTP, pero sería menos seguro, ya que desde un navegador se podrían insertar datos falsos por cualquier persona. La estructura de la petición GET sería:

GET/datos.php?t=valor1&h=valor2&l=valor3&uv=valor4&p=valor5&lat=valor6&long=valor7&d=valor8&v=valor9&plu=valor10\r\n
Host: tfgsergio.esy.es HTTP/1.1\r\n

El otro envío de datos se realiza al servidor de IFTTT. En este caso solo hay que enviar los datos relativos a la posición de la estación meteorológica (longitud y latitud).

POST trigger/alarma/with/key/”clave_ifttt” HTTP/1.1\r\n
Host: maker.ifttt.com\r\n
*Accept: */*\r\n*
Content-Length: longitud\r\n
Content-Type: application/x-www-form-urlencoded\r\n
value1=longitud&value2=latitud\r\n

Donde “clave_ifttt” es la clave privada de IFTTT, la cual identifica al usuario.

5.4.1. Comunicación vía WiFi.

Para dotar al sistema de comunicación WiFi se utiliza el módulo ESP8266-v01 del fabricante Expressif.

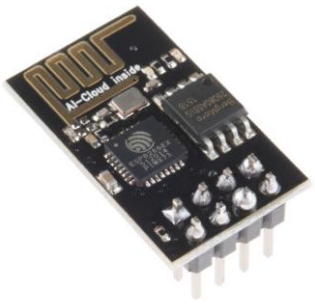

Figura 41. Módulo ESP8266-V01.

Funcionalidad de los pines que incorpora:

- GND y VCC para la alimentación.
- GP0 y GP2 son puertos de propósito general.
- CH_PD es el pin para llevar al chip a modo de bajo consumo.
- RST es el pin de reset, si se produce un pulso a nivel bajo se resetea el módulo.
- RX y TX son los pines utilizados para la comunicación serie asíncrona.

Las principales características técnicas de este dispositivo son:

- Soporta los protocolos 802.11 b/g/n
- Pila TCP/IP integrada
- Procesador interno de 32 bits.
- Rango de operación de -40ºC a 125ºC
- Consumo en modo de baja energía menor de 10 µA.

La versión del firmware que viene cargada de fábrica en este módulo está anticuada y no soporta algunas funciones como WPS. Por lo que se procede a actualizar el firmware del ESP8266, para ello se necesita instalar el programa “ESP8266 Flash Downloader” en el ordenador y un archivo con extensión .bin para descargar a través del USB del ordenador, pero como puente entre el USB y la UART del ESP8266 es necesario un convertidor de señal serie a TTL. Una vez realizadas las conexiones que indica el fabricante para poder actualizar la versión del firmware, se procede a su descarga.

El microcontrolador controla el envío de comando AT hacia este módulo a través de la UART2 a una velocidad de comunicación de 9600 baudios.

Los comandos que se usan junto con la respuesta a cada comando son:

- **AT+WPS=1**: activa la función WPS. Después de ejecutar este comando es el momento de pulsar el botón WPS del router para que se inicie la vinculación del ESP8266 al router. Si se establece la conexión correctamente responde con OK mientras que si se produce algún fallo responde con ERROR. La primera vez que se conecta al router es necesario que una persona pulse el botón WPS, pero en ocasiones siguientes ya no es necesario, ya que el ESP8266 guarda en memoria flash los datos de acceso al router y se conecta de forma automática.

- **AT+CWMODE=<modo>**: configura el modo de funcionamiento del ESP8266 y lo guarda en la memoria Flash. Si modo es 1 el ESP8266 actúa como cliente, si es 2 actúa como servidor y si es 3 puede ser cliente y servidor a la vez. En este proyecto solo se utiliza el ESP8266 en modo cliente modo=1.
- **AT+CIPMODE=0**: sirve para configurar el modo de transmisión.
- **AT+CIPMUX=<modo>**: sirve para fijar el modo de conexión simple o múltiple, si modo es 0 o 1 respectivamente.
- **AT+CIPSTART=<id>,<modo>,<ip>,<puerto>**: sirve para establecer una conexión TCP o UDP con un dominio de Internet a través del puerto indicado. Por ejemplo, en este proyecto se usa una conexión TCP, para realizar una conexión a la base de datos a través del puerto 80, por lo que el comando AT queda de la siguiente forma: AT+CIPSTART=”TCP”,”tfgsergio.esy.es”,”80”.
- **AT+CIPSEND=<longitud>**: prepara al módulo para el envío de datos, responde con el carácter “>” cuando está listo para el envío y procede a enviar el número de caracteres indicados en longitud.
- **AT+CIPCLOSE=<id>**: cierra una conexión activa identificada por el parámetro id.
- **AT+GSLP=<time>**: lleva el dispositivo a modo de bajo consumo durante el tiempo especificado en “time” en milisegundos. En la hoja de características del dispositivo se especifica que para entrar en este modo deben estar conectados XPD_DCDC con EXT_RST mediante una resistencia de cero ohmios. Responde con OK si entra correctamente a modo dormido.
- **AT+SLEEP=<sleep mode>**: comando para fijar el modo de baja energía del dispositivo. Por defecto está fijado a valor 2.

0 : disable sleep mode
1 : light-sleep mode
2 : modem-sleep mode

El tiempo máximo que el ESP8266 puede estar en estado de baja energía viene limitado por los 32 bits. El máximo rango es 2^{32} dando un valor de 4.294.967.293 especificado en microsegundos. Convirtiendo el valor anterior a minutos se obtiene un valor de 71,58 minutos.

Debido a que durante las pruebas de este módulo se detectó que tenía fallos de respuesta, se diseña una máquina de estados para poder solucionar este problema. Su funcionamiento se basa en que, si el microcontrolador no recibe la respuesta oportuna de cada comando enviado, se espera durante un periodo de tiempo fijo, si este no llega se envía otra vez el comando o se vuelve a comenzar la transmisión.

La transmisión de los comandos por la UART 2 para realizar el envío de los datos al servidor se realiza mediante una máquina de estados, que puede resumirse en el diagrama siguiente:

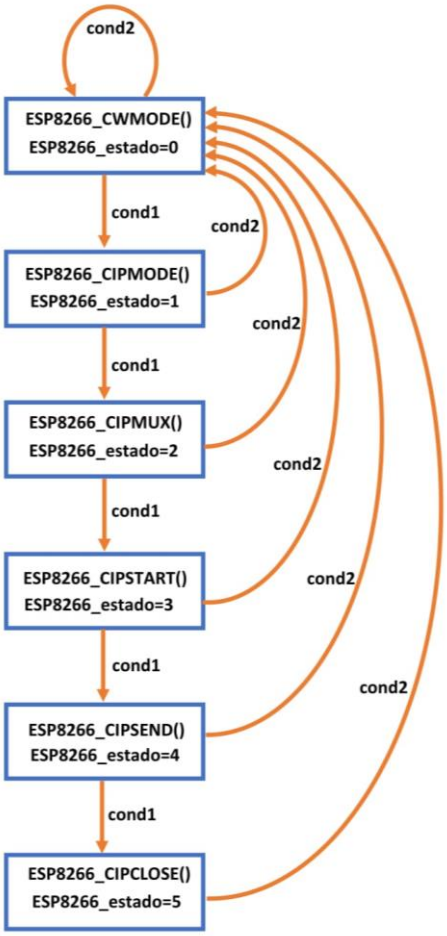

Figura 42. Máquina de estados ESP8266.

La máquina de estado anterior va evolucionando si la respuesta que da el ESP8266 a cada comando es la esperada, esta condición viene especificada en el diagrama como “cond1”. Mientras que si la respuesta es de fallo salta la interrupción del Timer y vuelve al estado inicial para volver a intentar el envío, lo que viene determinado en el diagrama por “cond2”. Por último, se envía a estado de bajo consumo con el comando correspondiente.

5.4.2. Comunicación vía GPRS.

Para el envío de los datos mediante GPRS se utiliza el módulo SIM808 del fabricante SimCom.

Figura 44. Vista superior módulo SIM808.

Figura 43. Vista inferior módulo SIM808.

El envío de los datos de los sensores cuando la estación meteorológica se encuentra en un lugar remoto se realiza mediante el módulo SIM808 del fabricante SimCom, que permite establecer conexiones GSM (Global System for Mobile communications) y GPRS (General Packet Radio Service) a través de la red de un operador de telefonía. Para ello hay que poner una tarjeta SIM al módulo. Como se van a enviar los datos hacia la base de datos es necesario una conexión GPRS. El módulo SIM808 también permite establecer llamadas de voz o el envío de mensajes SMS mediante la red GSM, características que no serán abordadas en el presente proyecto. Este módulo se controla mediante comandos AT a través de la UART3 del microcontrolador.

La velocidad de comunicación por defecto es de 115200 baudios, es posible cambiar esta velocidad mediante el comando AT+IPR=BAUDIOS. Los comandos que se usan junto con la respuesta a cada comando son:

- **AT+CFUN=1**: comando que sirve para activar todas las funcionalidades del dispositivo. El módulo responde con “OK” si ha tenido éxito el comando.
- **AT+CSTT=<apn>,<username>,<password>**: sirve para indicar al módulo el APN (Access Point Name) del operador de telefonía al que se va a conectar. Por ejemplo, para la operadora Jazztel el APN es “jazzinternet”, mientras que el usuario y la contraseña se deben dejar en blanco.
- **AT+CIICR**: con este comando se activa el perfil de datos inalámbrico de GPRS. Responde con OK si ha tenido éxito o con ERROR en caso contrario
- **AT+CIFSR**: se usa para obtener una dirección IP local. Responde con OK si ha tenido éxito o con ERROR en caso contrario
- **AT+CIPSTART=<mode>,<domain name>,<port>**: sirve para establecer una conexión TCP o UDP con un dominio de Internet a través del puerto indicado. Por ejemplo, en este proyecto se usa una conexión TCP, para realizar una conexión a la base de datos a través del puerto 80, por lo que el comando AT queda de la siguiente forma: AT+CIPSTART=”TCP”,”tfgsergio.esy.es”,”80”.

- **AT+CIPSEND=<length>**: sirve para enviar a través de una conexión establecida un número determinado de caracteres. Responde con SEND OK si los datos se han enviado correctamente.
- **AT+CIPCLOSE**: se usa para cerrar la conexión TCP o UDP establecida. Responde con CLOSE OK si la conexión se ha cerrado correctamente o con ERROR si se ha producido algún fallo.
- **AT+CSCLK=1**: con este comando se activa el modo de bajo consumo del dispositivo, que está controlado por el pin DTR. Si DTR está a nivel alto el dispositivo entra en modo dormido, mientras que si está a nivel bajo se despierta.

Todos los comandos deben acabar con los caracteres CR y LF (\r\n).

El diagrama de estados que describe el envío de datos hacia el servidor mediante GPRS con el módulo SIM808 es igual al descrito para el módulo ESP8266 identificando los parámetros de ambos módulos que tienen la misma funcionalidad.

5.4.3. Ubicación GPS.

El módulo SIM808 también incorpora la posibilidad de geolocalización mediante GPS (Global Positioning System).

Los comandos AT utilizados para obtener la posición del sistema son:

- AT+CGNSPWR=<mode>: comando que sirve para controlar la alimentación de la parte GPS del módulo SIM808. Si se pone a 1, se activa, mientras que si se fija a 0 se desactiva.
- AT+CGNSINF: comando que responde con la información obtenida de los satélites. La información que proporciona se estructura de la siguiente manera:

Índice	Parámetro	Unidades	Rango	Longitud
1	GPS run status	--	0-1	1
2	Fix status	--	0-1	1
3	UTC date & Time	yyyyMMddhh mmss.sss	yyyy: [1980,2039] MM : [1,12] dd: [1,31] hh: [0,23] mm: [0,59] ss.sss:[0.000,60.999]	18
4	Latitude	±dd.dddddd	[-90.000000,90.000000]	10
5	Longitude	±dd.dddddd	[-180.000000,180.000000]	11
6	MSL Altitude	meters		8
7	Speed Over Ground	Km/hour	[0,999.99]	6
8	Course Over Ground	degrees	[0,360.00]	6
9	Fix Mode	--	0,1,2	1
10	Reserved1			0
11	HDOP	--	[0,99.9]	4

12	PDOP	--	[0,99.9]	4
13	VDOP	--	[0,99.9]	4
14	Reserved2			0
15	GPS Satellites in View	--	[0,99]	2
16	GNSS Satellites Used	--	[0,99]	2
17	GLONASS Satellites in View	--	[0,99]	2
18	Reserved3			0
19	C/N0 max	dBHz	[0,55]	2
20	HPA	meters	[0,9999.9]	6
21	VPA	meters	[0,9999.9]	6

Tabla 8. Información proporcionada por el GPS.

En este proyecto solo interesan los datos de posición de latitud y longitud.

Uno de los motivos por los que se ha decidido dotar de geolocalización al sistema ha sido debido a que se puede emplazar en lugares aislados remotos durante periodos de tiempo largos, al volver a recoger el sistema una vez tomadas todas las medidas durante el tiempo deseado la vegetación de la zona puede haber crecido, dejando la estación un poco escondida y no visible, por lo que con la geolocalización se puede saber su ubicación.

5.5. Sistema de alimentación

El sistema debe ser autónomo en cuanto a forma de alimentación se refiere, lo cual quiere decir que debe ser capaz de ser alimentado de forma aislada, por lo que debe usar un medio de almacenamiento, en este caso se utiliza un batería. Al ser autónomo, la batería se tiene que recargar sin intervención del usuario, para lo que se utiliza un panel solar y un módulo de carga.

De todas las opciones de baterías que existen en el mercado se han elegido del tipo Li-ion (Iones de litio) debido a su reducido tamaño y a que son menos peligrosas que las Li-Po (Polímero de Litio). Este tipo de baterías son capaces de almacenar bastante carga en un reducido tamaño.

Si se quiere ampliar el tiempo de uso del sistema se deben asociar baterías de las mismas características en paralelo de esta forma se consigue aumentar la capacidad del conjunto obteniendo una mayor duración.

El sistema de alimentación está formado por un panel solar, una batería Li-Ion, un módulo de carga de batería y un convertidor Step Up:

- Panel solar: la energía para cargar la batería del sistema viene proporcionada por un panel solar con una potencia máxima de 5.5 W. En condiciones de pleno Sol, su salida en voltaje (Vmp) es de 18 V proporcionando una corriente de máxima (Imp) de 0.3 A. Otras características son: corriente de cortocircuito (Isc) de 0.34 A, que es la intensidad que circula por el panel cuando se cortocircuita su salida, y voltaje en circuito abierto (Voc) de 21 V, que es el voltaje a la salida del panel cuando no hay carga.

Figura 45. Panel solar.

- Batería 18650 de Li-Ion: es una batería con una capacidad de 4200 mAh y que proporciona una tensión nominal de salida de 3.7V. La carga y descarga de este tipo de baterías debe ser controlado ya que son muy peligrosas si se sobrepasan determinados limites, por lo que es necesario un módulo de control de carga adecuado.

Figura 46. Batería 18650.

- Módulo de carga: la batería 18650 se carga a través de un módulo basado en el integrado TP4056. Este circuito protege la batería de forma que si la tensión de esta está por debajo de 2.4 V deja de dar corriente, mientras que durante el proceso de carga al llegar a 4.2 V para de inyectar corriente a la batería, de esta forma se protege la batería en unos límites adecuados.

Figura 47. Módulo de carga de baterías.

- Convertidor Step Up: la batería proporciona una salida nominal de 3.7 V pero el sistema requiere de 5V para funcionar correctamente, por lo que se utiliza un subidor de tensión, el cual proporciona una eficiencia de conversión sobre el 95%.

Figura 48. Convertidor Step Up.

El diagrama completo del sistema de alimentación queda de la siguiente forma:

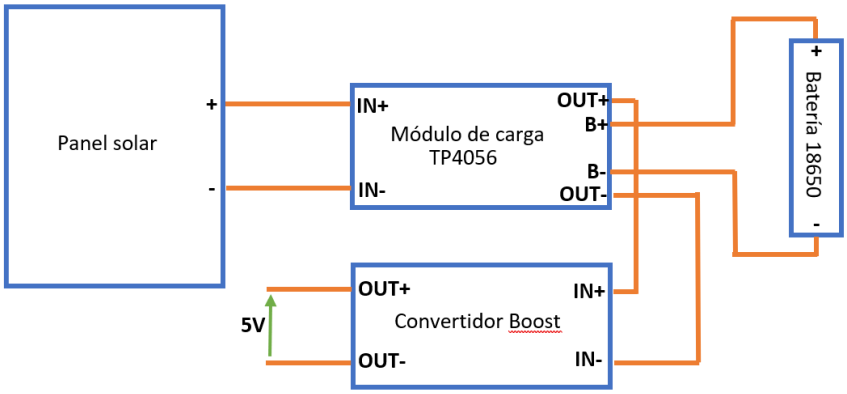

Figura 49. Diagrama de bloques del sistema de alimentación.

5.6. Integración hardware del sistema.

Una vez que se tienen todos los sensores y módulos funcionando de forma adecuada, es el momento de integrar todos los componentes que forman el sistema. Para ello se diseña una máquina de estados especificada en el siguiente diagrama:

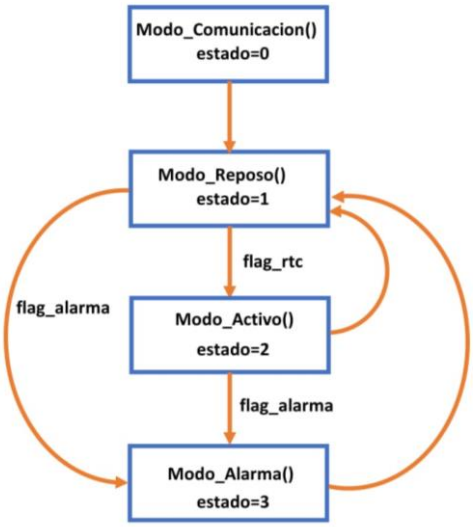

Figura 50. Máquina de estado general.

Se ha diseña una máquina de estados que gobierna el funcionamiento del programa cargado en el microcontrolador, el cual tiene varios estados, que pueden verse como modos de funcionamiento.

- Modo_Comunicacion: en este estado se detecta el modo de envío de los datos, si el switch está activo (ON) funciona mediante WiFi mientras que si está en posición OFF enviará los datos hacia el servidor con GPRS. Tras realizar la medida pasa automáticamente a estado de reposo. También se configura el intervalo de tiempo de envío de datos según sea el estado de dos switch:

Switch 1	Switch 2	Intervalo (minutos)
ON	**ON**	5
ON	**OFF**	10
OFF	**ON**	30
OFF	**OFF**	50

Tabla 9. Configuración del intervalo de envío de datos.

- Modo_Reposo: modo en el que todos los dispositivos deben permanecer en estado de baja energía durante el periodo configurado (periodo de interrupción del RTC). Cuando se produce la interrupción del RTC se pasa a estado activo. O si se detecta un cambio en la salida del sensor de inclinación se pasa a modo alarma.
- Modo_Activo: consiste en dar la orden de inicio de medida a todos los sensores uno por uno. Tras almacenar los parámetros meteorológicos sensados se procede a su envío hacía el servidor, ya sea mediante WiFi o GPRS, según se haya configurado. Tras el envío de los datos pasa automáticamente a estado de

reposo. Si se detecta un cambio en la salida del sensor de inclinación se pasa a modo alarma

- Modo_Alarma: entra en este modo si se detecta un cambio en la salida del sensor de inclinación, lo que indica que el sistema está siendo movido. Se empieza a enviar la posición de la estación cada cierto tiempo.

Una vez que se tiene el sistema completo funcionando se procede a la interconexión de sus componentes según se muestra en la imagen siguiente:

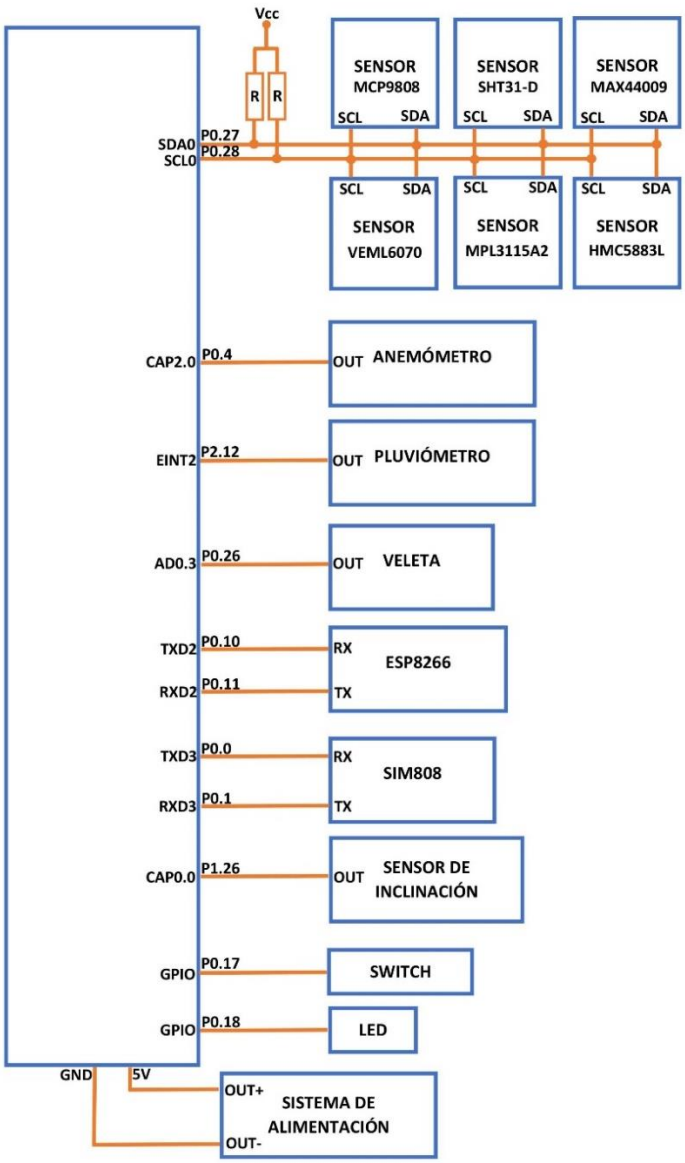

Figura 51. Integración Hardware del sistema.

Para ello se usa una caja con la tapa transparente, la cual es resistente al agua, parámetro importante debido a que el sistema final puede estar a la intemperie.

Se ha optado por la tapa transparente debido a que los sensores de radiación UV e iluminación deben estar protegidos a la vez que les tiene que llegar la luz ambiente o solar.

Aparte de los sensores de velocidad y dirección del viento y el pluviómetro, también deben estar fuera de la caja los sensores de humedad y de temperatura para obtener las medidas correctas de estos parámetros. Pero también deben estar protegidos, por lo que se utiliza una carcasa que permite que los sensores estén aireados a la vez que aislados de la lluvia y otros factores que provocarían su deterioro o destrucción.

Figura 52. Estación meteorológica ensamblada.

5.7. Sistema de alarmas mediante IFTTT.

Una funcionalidad que se ha dotado al proyecto es un pequeño sistema de alarma contra posibles robos del sistema. Para ello se utiliza un sensor de inclinación descrito anteriormente junto con IFTTT. Si la salida del sensor cambia es que la estación se está moviendo, esto puede estar causado por condiciones atmosféricas adversas (por ejemplo, ráfagas de viento fuerte) o debido a que la estación ha sido cogida por alguna persona. Si se activa la salida de este sensor se comprobará la posición de la estación y si esta cambia considerablemente respecto a la última medida, se envía un correo electrónico al usuario mediante IFTTT.

IFTTT (If This, Then That), que puede traducirse al español como “Si ocurre esto, haz esto otro” es un servicio web gratuito con el que se pueden automatizar distintas tareas mediante la creación de recetas (Applets).

En primer lugar, hay que registrarse en su página web. Tras acceder, se procede a crear la receta (Usuario -> New Applet). Aparece la siguiente pantalla:

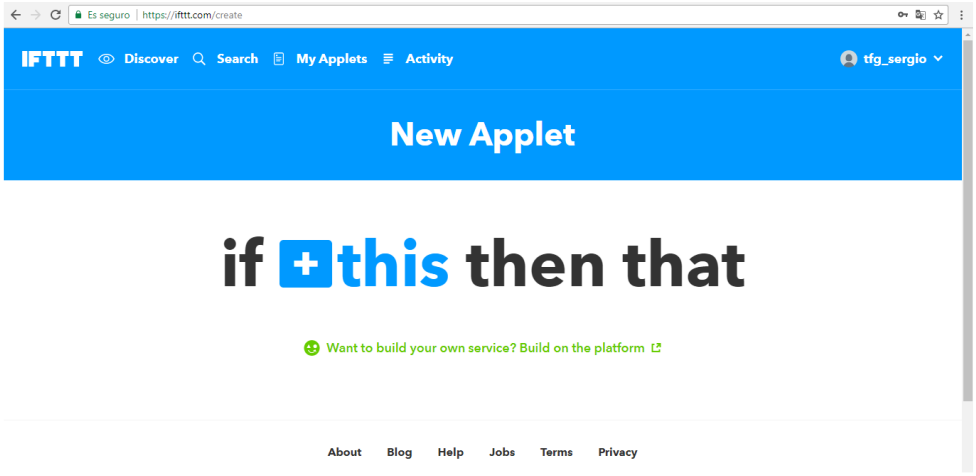

Figura 53. Nueva receta en IFTTT.

Primero se selecciona el evento que tiene que ocurrir para que desencadene una acción, para lo que se utiliza el servicio WebHooks, que es básicamente la recepción de una serie de datos mediante peticiones GET. Se le da un nombre al evento, en este caso “alarma”.

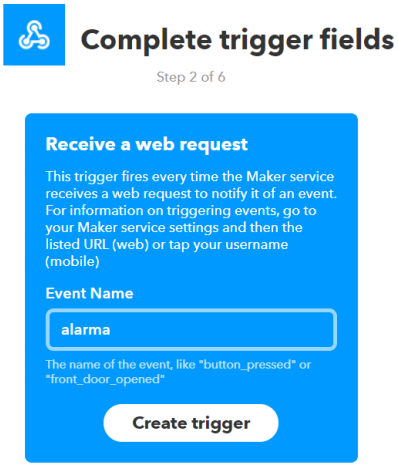

Figura 54. Trigger en WebHooks.

A continuación, se procede a crear la acción que se lleva a cabo si se produce el evento "alarma", que consiste en el envío de un correo electrónico.

Figura 55. Configuración IFTTT.

En la figura siguiente, se puede observar los parámetros que hay que configurar para el envío del correo electrónico, un asunto de mensaje y un cuerpo de mensaje.

Figura 57. Acción en WebHooks.

Figura 56. Acción en WebHooks configurada.

Por último, tras lanzar el evento se comprueba que se recibe un correo electrónico como el siguiente:

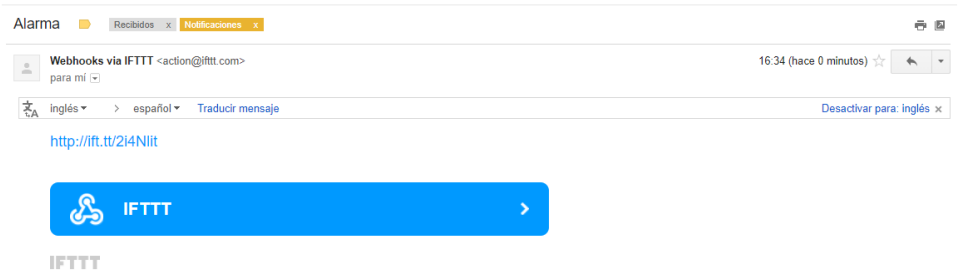

Figura 58. Email recibido cuando se produce la alarma.

El mensaje recibido en el correo electrónico del usuario contiene un link que redirige a la ubicación de la estación en Google Maps, de forma que se puede saber en qué lugar está.

5.8. Página web.

El entorno web diseñado para representar los datos pretende ser sencillo, de forma que la información aparezca graficada para extraer los datos necesarios visualmente.

Para el diseño de la página web donde se grafican los datos obtenidos se utilizan distintos lenguajes de programación:

- HTML (HiperText Markup Languaje) es un lenguaje que sirve para determinar la estructura o contenido de una página web, pero no sirve para determinar la funcionalidad que tiene una página web. Es un lenguaje que está basado en el uso de etiquetas.
- CSS (Cascading Style Sheets): herramienta de programación que se utiliza para determinar la apariencia de una página web. Junto con HTML pretende dar una mejora estética de la página web.
- JavaScript: es el lenguaje de programación que permite programar el comportamiento de una página web, de forma que sea interactiva y dinámica con el usuario.
- PHP (Hypertext Preprocessor): es un leguaje para el desarrollo web de código abierto. El código PHP se ejecuta en el servidor.

Otros elementos necesarios son:

- Hosting: es un servicio que provee una empresa para almacenar una página web y tenerla disponible mediante internet. A la hora de elegir un hosting para la web del proyecto se opta por uno gratuito proporcionado por Hostinger. Los hostings gratuitos están limitados en cuanto recursos se refiere, pero es más que suficiente para la página web de este proyecto.
- Dominio: es el nombre que identifica a una página web en Internet. Viene incluido con el hosting y es “tfgsergio.esy.es”. El objetivo de un dominio junto con DNS (Domain Name System) es la traducción del nombre de dominio de un sitio web (fácilmente memorizable) a su dirección IP.
- Base de datos: es un servicio que permite almacenar datos de forma organizada para posteriormente utilizarlos en alguna aplicación. Cada base de datos está compuesta por una o más tablas y cada tabla está formada por filas y columnas. En este proyecto se usa una base de datos de MySQL.
- Programa phpMyAdmin: programa utilizado para la manipulación de bases de datos remotamente a través de una interfaz web. De forma que permite la administración y el control de las tablas que forman las bases de datos (crear, editar y borrar) y la información que tienen (seleccionar, actualizar, borrar).
- Programa FileZilla: software gratuito empleado para la subida de archivos al servidor. En primer lugar, hay que conectarse al servidor mediante FTP (File Transfer Protocol) para lo cual se accede al Gestor de sitios y se introducen los datos del servidor FTP, el usuario y la contraseña:

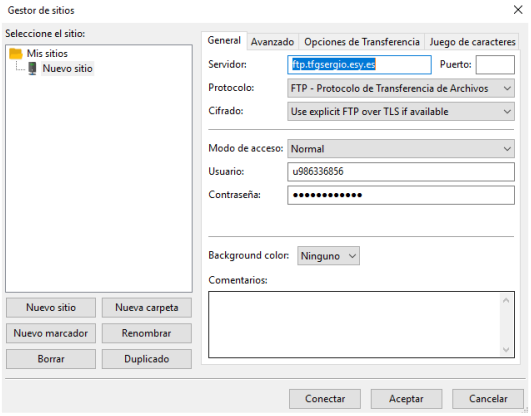

Figura 59. Configuración de FileZilla.

La relación de archivos utilizados para la creación de la página web se muestra en la siguiente figura, para su posterior explicación en apartados siguientes:

borrar_datos.php
cambio_datos.php
cierre_sesion.php
class.phpmailer.php
conexion_mysql.php
datos.php
desconexion_mysql.php
estilos.css
estilos_web.css
index.html
index_web.php
PHPMailerAutoload.php
recuperar_datos.html
recuperar_datos.php
validacion.php

Figura 60. Archivo que forman la página web.

5.8.1. Almacenaje de datos en MySQL.

Para almacenar los datos enviados desde el sistema remoto se utiliza una base de datos MySQL, la cual se crea mediante el programa phpMyAdmin. En concreto, para este proyecto se crean dos bases de datos la primera de nombre “u986336856_bd” y la segunda de nombre “u986336856_opt”.

Base de Datos MySQL	Usuario MySQL	Acciones
u986336856_bd	u986336856_ser	Ingresar phpMyAdmin
u986336856_opt	u986336856_ser1	Ingresar phpMyAdmin

Figura 61. Bases de datos usadas en el proyecto.

A continuación, en la base de datos “u986336856_bd” se crea una tabla llamada “datos”, en la que se almacenan los datos recibidos desde el sistema remoto, con la siguiente estructura:

#	Nombre	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado	Extra	Acción
1	id	int(32)			No	*Ninguna*	AUTO_INCREMENT	Cambiar Eliminar Navegar los valores distintivos Más
2	time2	datetime			No	*Ninguna*		Cambiar Eliminar Navegar los valores distintivos Más
3	time	timestamp			No	CURRENT_TIMESTAMP		Cambiar Eliminar Navegar los valores distintivos Más
4	temperatura	float			No	*Ninguna*		Cambiar Eliminar Navegar los valores distintivos Más
5	humedad	float			No	*Ninguna*		Cambiar Eliminar Navegar los valores distintivos Más
6	luminosidad	float			No	*Ninguna*		Cambiar Eliminar Navegar los valores distintivos Más
7	indiceUV	float			No	*Ninguna*		Cambiar Eliminar Navegar los valores distintivos Más
8	presion	float			No	*Ninguna*		Cambiar Eliminar Navegar los valores distintivos Más
9	latitud	float			No	*Ninguna*		Cambiar Eliminar Navegar los valores distintivos Más
10	longitud	float			No	*Ninguna*		Cambiar Eliminar Navegar los valores distintivos Más
11	dirV	float			No	*Ninguna*		Cambiar Eliminar Navegar los valores distintivos Más
12	velV	float			No	*Ninguna*		Cambiar Eliminar Navegar los valores distintivos Más
13	pluviometria	float			No	*Ninguna*		Cambiar Eliminar Navegar los valores distintivos Más

Figura 62. Estructura de la tabla “datos”.

El archivo “*datos.php*” es el encargado de recibir los datos enviados desde el sistema remoto mediante una petición POST y almacenarlos en los correspondientes apartados de la tabla de la base de datos se componen de las siguientes partes:

- Establecer la zona horaria de España de forma que el tiempo que se inserta en la base de datos es acorde a la ubicación.

```
// Zona horaria de Espa
date_default_timezone_set('Europe/Madrid');
$time2 = date("Y-m-d H:i:s");
```

Código 49. Zona horaria de España.

- Se capturan los datos en variables de la petición POST recibida.

```
$temperatura=$_POST['t'];
$humedad=$_POST['h'];
$luminosidad=$_POST['l'];
$indiceUV=$_POST['uv'];
$presion=$_POST['p'];
$latitud=$_POST['lat'];
$longitud=$_POST['long'];
$dirV=$_POST['d'];
$velV=$_POST['v'];
$pluviometria=$_POST['plu'];
```

Código 50. Captura de datos petición POST.

- Conexión a la base de datos. Con la función “*mysqli_connect()”* se abre una conexión con el servidor de la base de datos MySQL, pasándole como

parámetros el nombre del servidor, el nombre de la base de datos, el usuario de la base de datos y la contraseña asociada a la base de datos. Devuelve el identificador asociado a la conexión si la función se ejecuta con éxito, o devuelve FALSE en caso de error.

```php
// Database credentials
$servername = "mysql.hostinger.es";
$username = "u986336856_ser";
$dbname = "u986336856_bd";
$password = "sergio";

// Create connection.
$conn = mysqli_connect($servername, $username, $password, $dbname);
if (!$conn) {
    die("Connection failed: " . mysqli_connect_error());
}
```

Código 51. Conexión a la base de datos.

- Se insertan los valores almacenados en las variables en su correspondiente campo de la tabla "datos". Mediante la función *"mysqli_query()"* ejecuta la sentencia sql definida asociada al identificador de la conexión. Al ser una sentencia INSERT devuelve TRUE si se ejecuta con éxito.

```php
// Inserta los valores en la tabla
$sql = "INSERT INTO datos (time2, temperatura, humedad,
luminosidad, indiceUV, presion, latitud, longitud, dirV, velV,
pluviometria)
VALUES ('$time2', $temperatura, $humedad, $luminosidad,
$indiceUV, $presion, $latitud, $longitud, $dirV, $velV,
$pluviometria)";

if (mysqli_query($conn, $sql)) {
    echo "OK"; // Operacion correcta, responde OK
} else {
    echo "Fail: " . $sql . "<br>" . mysqli_error($conn);
}
```

Código 52. Insertar variables recibidas en la tabla "datos".

- Por último, se cierra la conexión con la base de datos. Con la función *"mysqi_close"* se cierra la conexión, asociada al identificador, con el servidor de MySQL.

```php
// Cerrar la conexión la base de datos
mysqli_close($conn);
```

Código 53. Cierre de conexión con la base de datos.

5.8.2. Hoja de estilos CSS.

Como se ha comentado anteriormente, se utiliza CSS para proporcionar un aspecto más visual a la página web, de forma que se pueden cambiar el tipo de letra, redondear los marcos de los contenedores de las estructuras de HTML.

En la página web se usan dos hojas de estilos, *"estilos.css"* para aplicar los estilos correspondientes a los documentos *"index.html"* y *"recuperar_datos.html"*. Y la hoja de estilos *"estilos_web.css"* que se utiliza en el archivo *"index_web.php"*.

Para trabajar con las hojas de estilos en primer lugar hay que incluirlas en el hosting y después referenciarlas en el archivo que se vayan aplicar.

```html
<link rel="stylesheet" href="estilos.css" type="text/css" media="all" />
```

Código 54. Insertar hoja de estilos CSS en HTML.

Los tres tipos de estilos que se aplican a distintos elementos de la estructura en HTML, son:

- Selector de id: permite aplicar estilo CSS a un único elemento de la página identificado por su id. Se usa el símbolo de la almohadilla (#) seguido del id del elemento. A continuación, se muestra un ejemplo en el que se aplica estilo CSS al elemento “cuerpo” .

```
#cuerpo{
    background: #3498DB;
    border-radius: 20px 20px 20px 20px;
    border: 3px solid #3A3A38;
    padding: 15px 30px;
    margin: 1px;
}
```

Código 55. Selector de id en CSS.

- Selector de tipo etiqueta: aplica estilo CSS a todos los elementos de la página cuya etiqueta HTML coincida con el valor del selector. En este caso el selector es “label” al que se le aplica el color y el tipo de letra especificados.

```
label{
    color: #3A3A38;
    font-weight: bold;
}
```

Código 56. Selector de tipo etiqueta en CSS.

- Selectores de clase: utilizados para dar estilo CSS a varios elementos diferentes que tienen el mismo valor de identificador en el atributo “class”

```
.boton{
    background: #85C1E9;
    color: #3A3A38;
    width: 110px;
    height: 45px;
    font-size: 20px;
}

.boton:active{
    position: relative;
    top: 3px;
}
```

Código 57. Selector de clase en CSS.

5.8.3. Acceso a la página principal.

La página web tiene un sistema de autenticación para acceder al contenido principal donde se muestra la información relevante. Se ha implementado dicha funcionalidad debido a que en la página web se muestra la ubicación del sistema remoto, por lo que podría ser robado si no se protege esta información.

En el archivo *“index.html”* se encuentra descrita la página principal a la que se accede al introducir el nombre de dominio de la página web en un navegador:

- La página está formada por un cuadro donde se presenta un formulario para introducir los datos de acceso pulsando el botón “Entrar” o para comenzar la recuperación de los datos de acceso pulsando sobre el botón “Recuperar datos”.

Figura 63. Página de autenticación.

- El código necesario para mostrar el aspecto de la página es sencillo. Basta con crear unos divisores dentro de otros e insertar las características del formulario, cuyos datos se envían mediante el método POST.

```
<html lang="es">

<head>
    <meta charset="utf-8">
    <title> Inicio </title>
    <link rel="stylesheet" href="estilos.css" type="text/css" media="all"
/>
</head>

<body>
    <div id="principal">
        <center>
        <label1> Introduzca los datos de acceso </label1>
        </center>
        <div id="cuerpo">
            <form id="formulario" method="POST" action="validacion.php">
                <p> <label> Usuario: </label> </p>
                <input id="usuario" type="text" name="usuario"
placeholder="Usuario" />
                <p> <label> Contraseña: </label> </p>
                <input id="contrasena" type="password" name="contrasena"
placeholder="Contraseña" />
                <center>
                <p id="boton"> <input type="submit" id="submit"
name="submit" value="Entrar" class="boton"> </p>
                </center>
            </form>
            <form id="formulario" method="POST"
action="recuperar_datos.html">
                <center>
                <p id="boton"> <input type="submit" id="submit"
name="submit" value="Recuperar datos" class="boton1"> </p>
                </center>
            </form>
        </div>
    </div>
</body>

</html>
```

Código 58. Fichero index.html

- Al pulsar en el botón “Enter”, se redirige al archivo “validación.php” para comprobar que los datos introducidos son correctos. Es un fichero que se ejecuta en el servidor, de manera transparente para el usuario. En primer lugar, guarda el usuario y la contraseña introducidos por el usuario y enviadas mediante una petición POST en sus correspondientes variables. Después, comprueba si alguno de los dos datos enviados y guardados en las variables es nulo, es decir el usuario no ha rellenado uno o los dos campos. Si ocurre esto, aparece el mensaje de advertencia “Es necesario que rellene los dos campos” como se muestra a continuación:

Figura 64. Mensaje "Es necesario que rellene los dos campos".

Mientras que, si se han rellenado los dos campos, se procede a realizar la conexión a la base de datos para consultar los campos "user" y "pass" almacenados en ella. Una vez almacenados los datos, se hace la desconexión de la base de datos y se procede a la verificación de los datos introducidos mediante la comparación con los datos de acceso de la base de datos.

Si el usuario introducido es incorrecto se muestra un mensaje como el siguiente:

Figura 65. Mensaje "El usuario introducido no está en la base de datos".

Mientras que si el usuario es el correcto se procede a comprobar si la contraseña introducida es correcta. Si es así, se procede a iniciar una sesión con el identificador del nombre de usuario introducido y se redirecciona al archivo *"index_web.php"*. Pero si la contraseña introducida es incorrecta se muestra el mensaje siguiente:

Figura 66. Mensaje "La contraseña es incorrecta".

- Si se pulsa sobre el botón de Recuperar datos se redirige al usuario al archivo *"recuperar_datos.html",* en el que podrá recuperar los datos de acceso a la página. Este archivo se explica en el apartado siguiente.
- El código utilizado para programar todo lo descrito en referencia al archivo "validación.php" se muestra a continuación:

```
<?php

//Se guardan los par□tros recibidos
$usuario = $_POST['usuario'];
$pass = $_POST['contrasena'];

//Algn campo vac
if(empty($usuario) || empty($pass)){
    echo 'Es necesario que rellene los dos campos.';
    exit();
}

//Conexi□ la base de datos
include("conexion_mysql.php");

$resultado = mysql_query("SELECT user, pass FROM opciones
WHERE user='" . $usuario . "'");
$fila = mysql_fetch_array($resultado);

include("desconexion_mysql.php");

if($fila['user'] ==  $usuario){
    if($fila['pass'] ==  $pass){
        session_start();
```

```
                $_SESSION['usuario'] = $usuario;
                header("Location: index_web.php");
            }else{
                echo  htmlentities('La contrase□s incorrecta.');
                exit();
            }
        }else{
            echo  htmlentities('El usuario introducido no est□n
        la base de datos.');
            exit();
        }

        ?>
```

Código 59. Fichero validación.php

5.8.4. Recuperar datos de inicio de sesión.

Otra funcionalidad que tiene la página web es la recuperación de datos de inicio de sesión debido a que el usuario se haya olvidado de ellos.

Para ello, en la página de acceso está el botón de "Recuperar datos", que, al pulsar sobre él, se redirige al usuario a otra página en la que se ejecuta el archivo "*recuperar_datos.html"* con el siguiente aspecto:

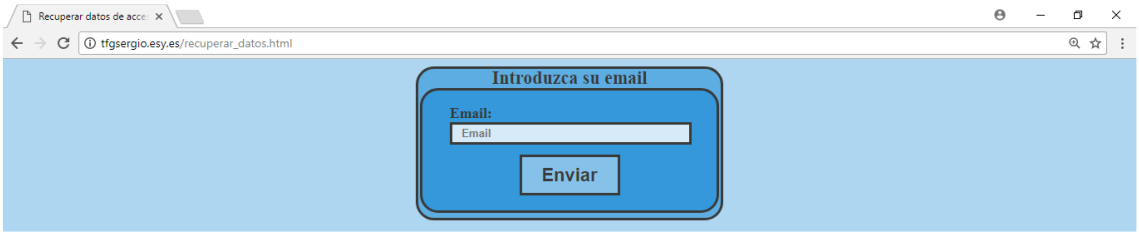

Figura 67. Página de recuperación de datos.

En la página se muestra un formulario para la recuperación de los datos de acceso si el usuario no se acuerda de los mismos. Para ello, se debe introducir el email en el campo indicado y pulsar sobre "Enviar".

La estructura del archivo *"recuperar_datos.html*" es muy parecida a la descrita del archivo "*index.html*". En este caso se envía mediante una petición POST el email introducido en el campo correspondiente para proceder a su comprobación en el fichero "*recuperar_datos.php*".

```
<html lang="es">

<head>
    <meta charset="utf-8">
    <title> Recuperar datos de acceso </title>
    <link rel="stylesheet" href="estilos.css" type="text/css" media="all" />
</head>

<body>
    <div id="principal">
        <center>
        <label1> Introduzca su email </label1>
        </center>
        <div id="cuerpo">
            <form id="formulario" method="POST" action="recuperar_datos.php">
                <p> <label> Email: </label> </p>
                <input id="usuario" type="text" name="email"
placeholder="Email" />
                <center>
                <p id="boton"> <input type="submit" id="submit" name="submit"
value="Enviar" class="boton"> </p>
                </center>
            </form>
        </div>
```

```
    </div>
</body>

</html>
```

Código 60. Fichero recuperar_datos.html

En el archivo “recuperar_datos.php” se almacena el email introducido en el formulario en una variable. Si el usuario ha dejado el campo vacío y ha pulsado en Enviar se advierte con el mensaje:

Figura 68. Mensaje “Es necesario que rellene el campo Email”.

Mientras que si se ha introducido algún dato se procede a realizar la conexión con la base de datos. Se hace una consulta de los campos “user”, “pass” y “email” de la tabla opciones donde el campo email coincida con el dato introducido en el formulario. Se procede a la desconexión de la base de datos. Si el email introducido se corresponde con el que hay en la base de datos se almacenan los datos de “user” y “pass” en variables y se incluye el archivo “*PHPMailerAutoload.php*”. A continuación, se crea la variable mail de la clase “PHPMailer” para poder enviar el correo electrónico con los datos de acceso.

En el campo ”addAddress” se introduce el email, en “Subjet” se pone el asunto del mensaje, en este caso “Datos de acceso” y en el campo “Body” el mensaje con los datos de acceso.

Si el email se ha enviado satisfactoriamente, se informa mediante un mensaje:

Figura 69. Mensaje “Datos enviados a su Email”.

El email recibido que contiene los datos de acceso tiene la siguiente estructura:

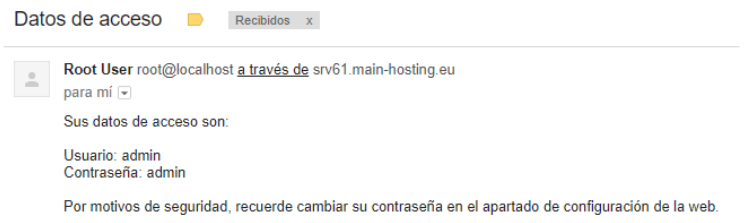

Figura 70. Email recibido con los datos de acceso.

Si el email introducido en el formulario no coincide con el que esta almacenado en la base de datos se informa con un mensaje:

Figura 71. Mensaje “El email introducido no está en la base de datos”.

Todo lo anterior se lleva a cabo mediante el archivo “recuperar_datos.php”:

```
<?php

$email = $_POST['email'];

if(empty($email)){
    echo 'Es necesario que rellene el campo "Email".';
    exit();
}

include("conexion_mysql.php");

$resultado = mysql_query("SELECT user, pass, email FROM opciones WHERE email='" .
$email . "'");
$fila = mysql_fetch_array($resultado);

include("desconexion_mysql.php");

if($fila['email'] ==  $email){
    $user=$fila['user'];
    $pass=$fila['pass'];
    require 'PHPMailerAutoload.php';
    $mail = new PHPMailer;
    $mail->addAddress("$email");
    $mail->Subject = 'Datos de acceso';
    $mail->Body = "Sus datos de acceso son:\n\nUsuario:
$user\nContrase□$pass\n\nPor motivos de seguridad, recuerde cambiar su contrase□n
el apartado de configuraci□e la web.";
    if($mail->send()) {
        echo 'Datos enviados a su email.';
        exit();
    }else{
        echo htmlentities('Error en el envde los datos. Vuelva a introducir su
email.');
    }
}else{
    echo htmlentities('El email introducido no est□n la base de datos.');
}
?>
```

Código 61. Fichero recuperar_datos.php

5.8.5. Página web principal.

Una vez que el usuario se ha autenticado correctamente en la página de acceso a la web se carga la página web principal en la que se representan los datos meteorológicos.

Esta página está estructurada en cuatro secciones bien diferenciadas:

- Sección de encabezado: muestra el título de la página web.

Figura 72. Título de la página principal.

- Sección de últimos datos medidos: en esta sección se muestra el mapa de ubicación de la estación meteorológica y los últimos datos recibidos del valor de los sensores.

Figura 73. Sección de últimos datos medidos.

- Sección de histórico de mediciones: apartado donde se muestran las gráficas correspondientes a todos los valores y los valores medios durante los días, meses o años. Se puede visualizar cada gráfica pulsando sobre el botón correspondiente.

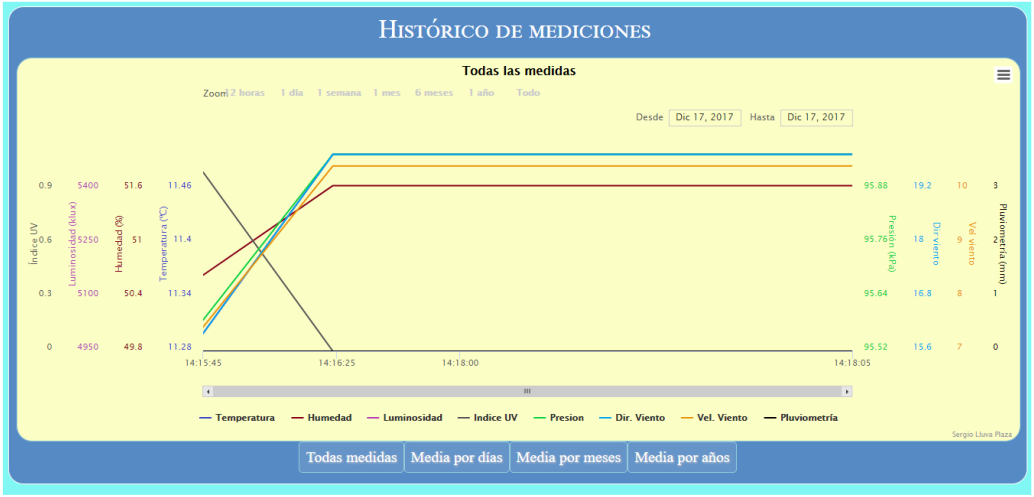

Figura 74. Sección de histórico de mediciones.

- Sección de configuración: en esta sección el usuario puede cambiar los datos de acceso a la página principal. También se puede cerrar la sesión o borrar todos los datos de la base de datos.

La acción de borrar todos los datos que contiene la base de datos es de utilizadas cuando el usuario cambia el sistema remoto de localización para obtener el reporte de datos climatológicos de otro lugar. Previo a este paso se recomienda utilizar la utilidad de exportar las gráficas de forma que los reportes anteriores pueden ser exportados por el usuario para ser almacenados.

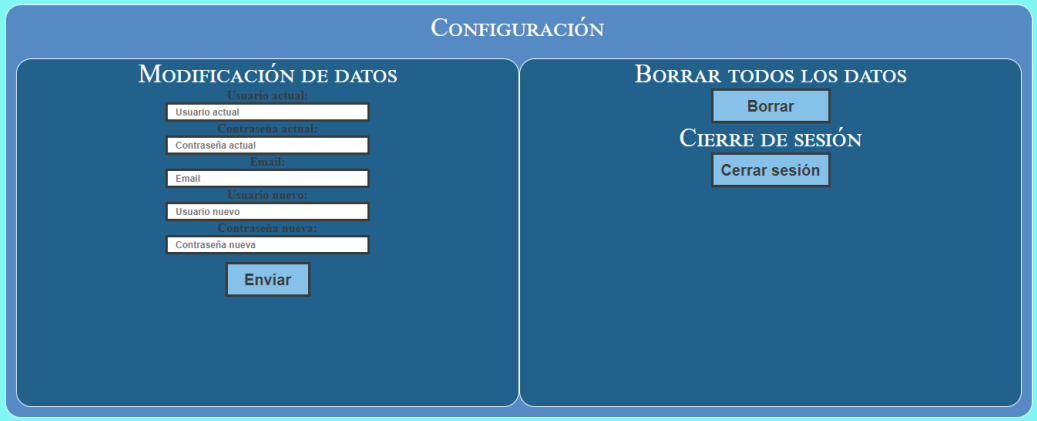

Figura 75. Sección de configuración.

5.8.6. Mapa mediante API de Google Maps.

El mapa que muestra la posición actual de la estación meteorológica se realiza con ayuda de la API de Google Maps.

Figura 76. Mapa de Google Maps.

A continuación, se muestras y explica el código programado para insertar el mapa en la estructura HTML de la página web:

```
<!-- Script para visualizar mapa  -->
<script>
function initMap() {
    var coordenadas = {lat: <?php echo $latitud; ?>, lng: <?php echo
$longitud; ?>}; // Ultimas coordenadas
    var map = new google.maps.Map(document.getElementById('map'), {
        center: coordenadas,
        zoom: 9,
        mapTypeId: google.maps.MapTypeId.ROADMAP
    });
```

```
    var icono=new google.maps.Marker({
        position: coordenadas,
        map: map,
    });

    var info = new google.maps.InfoWindow({
        content:"<center><strong>Estaci□eteorol□a <br><?php echo $latitud;
?>, <?php echo $longitud; ?></strong></center>",
    });

    info.open(map,icono);
}
```

Código 62. Función para mostrar el mapa.

- En primer lugar, mediante la etiqueta “script” se carga la API de Google Maps desde su servidor. El atributo “async” sirve para que el navegador vaya representando la página web durante el tiempo de carga de la API. La URL es desde donde se carga la API y debe contener la clave de API que ha tenido que ser solicitada previamente. El parámetro “callback” es el encargado de ejecutar la función “initMap” cuando la API de Google Maps está cargada por completo.
- La variable “coordenadas” almacena los últimos valores recibidos de latitud y longitud.
- En la variable “map” se crea un objeto con el mapa a cargar en el contenedor asociado al identificador obtenido mediante la función “getElementById”. El mapa se ubica en el lugar deseado de la página web mediante un elemento “div” que tenga asociado el identificador con el nombre del mapa.
- La propiedad “center” sirve para indicar el lugar donde se centra el mapa.
- La propiedad “position” indica la posición del marcador que se va a agregar.
- La propiedad “MapTypeId” sive para configurar el tipo de mapa que se muestra en la web. Hay cuatro tipos posibles, de los que se utiliza el tipo “ROADMAP” que muestra la vista del mapa de carreteras.
- La propiedad “zoom” se define mediante un numero entero y sirve para configurar el nivel de zoom con el que se representa inicialmente el mapa.
- La función “Map” crea un mapa nuevo dentro del contenedor deseado.
- El objeto “InfoWindow” sirve para incluir información mediante una ventana emergente en el mapa.
- Por defecto, la ventana de información no aparece en el mapa, por ello, para incluirla hay que llamar al método “open()” y pasarle como argumentos el identificador del mapa dónde se va a mostrar y el marcador donde debe fijarse.

5.8.7. Gráficas mediante Highcharts.

Highcharts es una librería escrita en JavaScript puro que permite la representación gráfica de datos. Highcharts tiene una licencia que es gratuita si el uso que se la va a dar es no comercial. Ofrece al diseñador una amplia gama de formas de representación de los datos. Es una herramienta compatible con todos los navegadores y está adaptada a entornos móviles, por lo que no se tendrá ningún problema a la hora de visualización en cualquier dispositivo

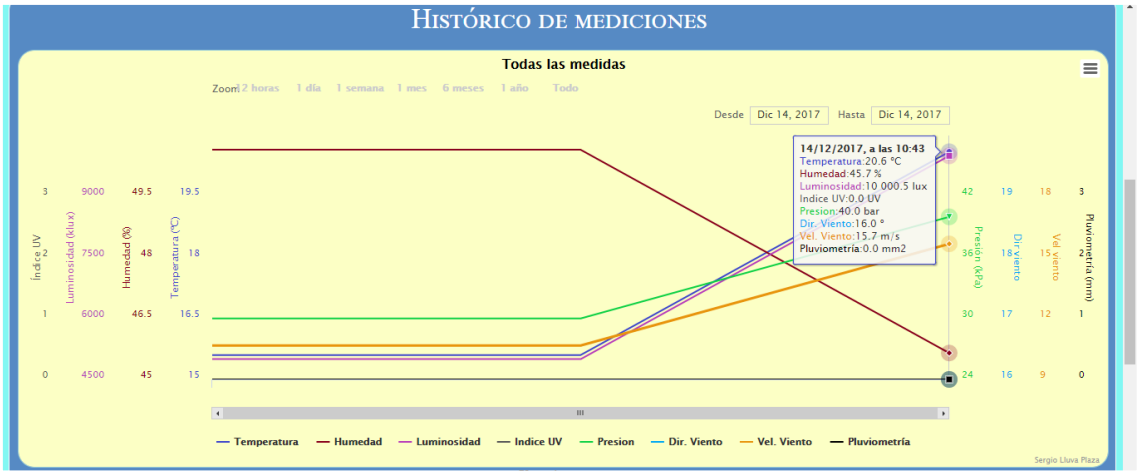

Figura 77. Gráfica mediante Highcharts.

También ofrece una API de forma que se simplifica el diseño de las gráficas, la cual hay que incluirla en el diseño para poder trabajar con ella:

```
<!-- Importar el archivo Javascript de Highcharts directamente desde su servidor -->
<script src="http://code.highcharts.com/stock/highstock.js"></script>
<script src="http://code.highcharts.com/modules/exporting.js"></script>
<script src="https://code.highcharts.com/highcharts.js"></script>
```

Código 63. Insertar archivos de Highcharts desde su servidor.

A continuación, se explica el código programado para ver la gráfica que muestra los datos de todos los sensores. Las demás gráficas se hacen igual exceptuando los valores representados.

En primer lugar, se definen las opciones generales de los gráficos que son aplicables a todos los gráficos que se representen en la página web. Para ello se hace uso de la estructura "*Highcharts.setOptions*" con la que mediante sus parámetros se puede modificar las palabras que aparecen en la zona del gráfico respecto a las que vienen por defecto que están en inglés.

```
    Highcharts.setOptions ({  // Opciones generales
        lang: {
            months: ['Enero', 'Febrero', 'Marzo', 'Abril', 'Mayo', 'Junio',
            'Julio', 'Agosto', 'Septiembre', 'Octubre', 'Noviembre',
'Diciembre'],
            weekdays: ['Domingo', 'Lunes', 'Martes', 'Mi鏗oles', 'Jueves',
'Viernes', 'Sabado'],
            shortMonths: ['Ene', 'Feb', 'Mar', 'Abr', 'May', 'Jun',
'Jul','Ago', 'Sept', 'Oct', 'Nov', 'Dic'],
            decimalPoint: ',',
            rangeSelectorFrom: "Desde",
            rangeSelectorTo: "Hasta",
            resetZoom: 'Reset zoom',
            resetZoomTitle: 'Reset zoom a 1:1',
            printChart: "Imprimir gr6co",
            downloadPNG: "Descargar imagen en formato PNG",
```

```
            downloadJPEG: "Descargar imagen en formato JPEG",
            downloadPDF: "Descargar documento en formato PDF",
            downloadSVG: "Descargar imagen en formato SVG",
            exportButtonTitle: "Opciones de descarga",
            printButtonTitle: "Imprimir gráfico",
            loading: "Cargando"
        }
    });
```

Código 64. Parámetro "lang".

Lo siguiente es definir una nueva variable que sirve para representar el nuevo gráfico junto con las propiedades que se le asignan posteriormente:

```
var chart = new Highcharts.stockChart('container',
```

Código 65. Nuevo gráfico.

Ahora es el momento de definir los parámetros del gráfico:

- Parámetro "*chart*": configuración de opciones generales del gráfico.

```
chart: {
    renderTo: 'container',
    zoomType: 'xy',
    alignTicks: true,
    backgroundColor: '#FCFFC5',
    height: 500,
    resetZoomButton: {
        theme: {
            fill: 'white',
            stroke: 'silver',
            r: 10,
            states: {
                hover: {
                    fill: '#417355',
                    style: {
                        color: 'white'
                    }
                }
            }
        }
    },
    type: 'line'
},
```

Código 66. Parámetro "chart".

- Parámetro "*credits*": con ella se pueden poner los créditos en la parte inferior de la gráfica.

```
credits: {
        enabled: true,
        text: "Sergio Lluva Plaza",
        href: false
    },
```

Código 67. Parámetro "credits".

- Parámetro "*navigator*": permite habilitar la barra de selección en la parte inferior.

```
navigator: {
        enabled: false
    },
```

Código 68. Parámetro "navigator".

- Parámetro "*rangeSelecto*r": permite habilitar los botones de selección del rango de tiempo que se muestran los valores.

```
rangeSelector: {
        buttonSpacing: 30,
        buttonTheme: { // styles for the buttons
            fill: 'none',
            stroke: 'none',
            'stroke-width': 0,
```

```
                r: 8,
                style: {
                    color: '#039',
                    fontWeight: 'bold'
                },
                states: {
                    hover: {
                    },
                    select: {
                        fill: '#039',
                        style: {
                            color: 'white'
                        }
                    }

                }
            },
            buttons: [{

                type: 'hour',
                count: 12,
                text: '12 horas' },
            {
                type: 'day',
                count: 1,
                text: '1 día
            },
            {
                type: 'day',
                count: 7,
                text: '1 semana'
            },
            {
                type: 'month',
                count: 1,
                text: '1 mes'
            },
            {
                type: 'month',
                count: 6,
                text: '6 meses'
            },
            {
                type: 'year',
                count: 1,
                text: '1 a□/span>
            },
            {
                type: 'all',
                count: 1,
                text: 'Todo'
            }],
            inputEnabled: true,
            selected: 0,    //Opcion marcada por
defecto
        },
```

Código 69. Parámetro "rangeSelector".

- Parámetro "*legend*": permite habilitar la leyenda de los puntos que forman la gráfica.

```
            legend: {
                    enabled: true,
                },
```

Código 70. Parámetro "legend".

- Parámetro "*title*": sirve para insertar un título en el gráfico.

```
        title: {
                text: 'Todas las medidas',
                align: 'center',
                margin: 0,
                style: {
                    color: '#000000',
                    fontWeight: 'bold'
```

```
        }
    },
```

Código 71. Parámetro "title".

- Parámetro "*xAxis*": sirve para definir la configuración del eje de abscisas. Se configura para que sea de tipo "datetime" debido a que es el eje que representa el tiempo y la fecha.

```
xAxis: {
        type: 'datetime',
        dateTimeLabelFormats: {
            hour: '%d-%m-%Y<br/>%H:%M',
            day: '%d-%m-%Y',
            month: '%m-%Y',
            year: '%Y'
        },
        tickPixelInterval: 50
    },
```

Código 72. Parámetro "xAxis".

- Parámetro "*yAxis*": con él se define el aspecto del eje de ordenadas. Hay que definir las propiedades de cada eje de los datos a representar. A continuación, solo se explica la configuración del eje de ordenadas para la variable de temperatura, siendo igual para las variables restantes.

```
yAxis: [{ // 1er yAxis (numero 0)
        gridLineWidth: 0,
            opposite: false,
            title: {
                margin: 0,
                text: 'Temperatura (C)',
                style: {
                    color: '#3F48CC'
                }
            },
            labels: {
                formatter: function() {
                    return this.value;
                },
                style: {
                    color: '#3F48CC'
                }
            },


    },
```

Código 73. Parámetro "yAxis".

- Parámetro "*tooltip*": opciones que muestra la información cuando el usuario se desplaza sobre la gráfica.

```
    tooltip: {
        valueDecimals: 2, // Dos decimales
        formatter: function() {

            var s = '<b>'+ Highcharts.dateFormat('%d/%m/%Y, a las %H:%M',
this.x) +'</b>';
            $.each(this.points, function(i, point) {
                var unit = {
                    'Temperatura': ' C',
                    'Humedad': ' %',
                    'Luminosidad': ' lux',
                    'Indice UV': ' UV',
                    'Presion': ' bar',
                    'Dir. Viento': ' ',
                    'Vel. Viento': ' m/s',
                    'Pluviometr: ' mm2',
                }[this.point.series.name];
                s = s + '<br>' + '<span style="color:'+ point.series.color
+'">'+ point.series.name +'</span>:'+Highcharts.numberFormat(point.y,1,"."," ")+ unit;
```

```
            });
            return s;
        },
        shared: true
    },
```

Código 74. Parámetro "tooltip".

- Parámetro "*plotOptions*": define un objeto de configuración para un conjunto de series. Se habilita un marcador que identifica a cada serie que se va a representar.

```
plotOptions: {
                series: {
                    marker: {
                    enabled:
false
                    }
                }
            },
```

Código 75. Parámetro "plotOptions".

- Parámetro "*series*": permite definir el conjunto de valores a graficar. Se muestra el código para el valor de la temperatura, siendo equivalente para el resto de medidas.

```
series: [{
        name: 'Temperatura',  //Variable a mostrar
        //zIndex: 1,
        yAxis: 0,
        color: '#3F48CC',
        data: (function() {
            var data = [];
            <?php for($i = 0 ;$i<count($rawdata);$i++){ ?>
            data.push([<?php echo
$rawdata[$i]["time2"];?>,<?php echo
$rawdata[$i]["temperatura"];?>]);
            <?php } ?>
            return data;
        })()
        },
```

Código 76. Parámetro "series".

Cuando se pulsa sobre un botón se borra el gráfico anterior y se carga el nuevo con la información solicitada.

6. Manual de usuario.

En este apartado se pretende explicar la puesta en marcha del sistema para poder empezar a recolectar los datos de los parámetros atmosféricos de una ubicación elegida.

Al principio es necesario indicar al sistema el modo de comunicación que se va a utilizar, lo que se indica mediante el switch 0, si está en posición ON el sistema funciona mediante WiFi y si está en posición OFF el sistema funciona mediante GPRS.

- Si el sistema solo se va a utilizar con WiFi, es necesario que el router al que se vaya a conectar para tener acceso a Internet tenga el botón de WPS.
  Antes de encender el sistema se debe pulsar el botón WPS del router, si el sistema se conecta correctamente al router aparecerá un led rojo encendido durante 5 segundos.
- Si el sistema funciona mediante GPRS es necesario haber introducido anteriormente la tarjeta SIM, sin protección mediante código PIN, al módulo SIM808. Si el sistema funciona correctamente, aparecerá un led rojo encendido durante 5 segundos.

También se debe configurar el intervalo de tiempo de envío de datos hacia el servidor, lo cual se realiza con los switch 1 y 2, según la tabla:

Switch 1	Switch 2	Intervalo (minutos)
ON	**ON**	5
ON	**OFF**	10
OFF	**ON**	30
OFF	**OFF**	50

Tabla 10. Configuración tiempo de envío de datos.

Una vez configurado el sistema remoto, comenzará el envío de los parámetros medidos hacia el servidor y podrán ser visualizados por el usuario. Para ello se debe introducir en un navegador web la URL “tfgsergio.esy.es”.

Una vez introducida la URL de la dirección de la página web aparece la pantalla de autentificación.

Tiene dos opciones, autentificarse con usuario y contraseña (para la primera vez será “admin” y ”admin” respectivamente) o la opción de recordar contraseña, en la que habrá que introducir el email para que se envíen los datos de acceso al email. Esta opción no está disponible si no se ha introducido por primera vez el dato del email en la sección de configuración de la web.

En ocasiones posteriores si se introduce el email que está en la base de datos se enviaran los datos al correo electrónico, si no está en la base de datos se informa al usuario.

La web está dividida en tres apartados, el primero muestra la ubicación en un mapa de Google Maps de la estación y a su lado están los últimos daos medidos de las variables climatológicas.

Seguidamente está el apartado de visualización de las gráficas con el histórico de datos medidos, con botones con los que se pueden mostrar medidas por separado de un periodo de tiempo o las medias de cada periodo de tiempo.

En la última sección de la página web se pueden modificar los datos de inicio de sesión actuales. También se pueden borrar todos los datos almacenados en la base de datos, para lo que es recomendable exportar los datos anteriores mediante la opción presente en la parte superior derecha de los gráficos.

7. Análisis de costes.

Este apartado consiste en un desglose del análisis de costes estimado para este proyecto en lo referido a hardware, software y mano de obra.

7.1. Coste total de equipo y software.

Concepto	Precio	Periodo de amortización (meses)	Uso del producto (meses)	Coste para el proyecto
Ordenador portátil (incluye S.O.)	899	48	6	112.38€
Keil uVision5 (versión de evaluación)	0	N/A	N/A	0€
Microsoft Office 365 (licencia universitaria)	69	48	6	8.63€
KiCad	0	N/A	N/A	0€
Coste total de equipo y software				121.01€

Tabla 11. Desglose del coste de equipo y software.

7.2. Coste de material.

Descripción	Precio [euros/ud]	Cantidad	Coste para el proyecto
Módulo MCP9808	6	1	6€
Módulo SHT31-D	5.19	1	5.19€
Módulo MAX44009	1.85	1	1.85€
Módulo VEML6070	4.41	1	4.41€
Módulo MPL3115A2	4.11	1	4.11€
Módulo HMC5883L	1.76	1	1.76€
Veleta	7	1	7€
Anemómetro	5	1	5€
Pluviómetro	6	1	6€
Módulo ESP8266	1.33	1	1.33€
Módulo SIM808	8.28	1	8.28€
Convertidor Boost	1	1	1€

Módulo de carga	1.5	1	1.5€
Panel solar	6.5	1	6.5€
Batería 18650	5	1	5€
Portapilas	0.45	1	0.45€
Caja estanca	3.72	1	3.72€
LPC1768	4.65	1	4.65€
Otros componentes (resistencias, condensadores, etc.)	N/A	N/A	10€
Fabricación PCB	10	1	10€
Coste total del material			93.69€

Tabla 12. Desglose del coste de material.

7.3. Coste de mano de obra.

Concepto	Categoría	€/hora	Total horas	Coste para el proyecto
Sergio Lluva Plaza	Ingeniero Junior	20	240	4800€
Coste total de mano de obra				4800€

Tabla 13. Desglose del coste de mano de obra.

7.4. Presupuesto de ejecución material.

Concepto	Coste
Coste de equipo y software	121.01€
Coste del material	93.69€
Coste de mano de obra	4800€
Total	5014.7€

Tabla 14. Desglose del presupuesto de ejecución de material.

7.5. Presupuesto de contrata.

Concepto	Coste
Presupuesto de ejecución material	5014.7€
Gastos generales (17%)	852.50€
Beneficio industrial (6%)	300.88€
Total	6168.08€

Tabla 15. Desglose del presupuesto de contrata.

7.6. Presupuesto total.

Concepto	Coste
Presupuesto de contrata	6168.08€
I.V.A. (21%)	1295.30€
Presupuesto Total	7463.38€

Tabla 16. Desglose del presupuesto total.

8. Conclusiones y trabajo futuro.

El objetivo principal de este proyecto era la creación de un sistema capaz de recoger los datos climatológicos de un entorno y graficarlos en un entorno web, lo cual se ha conseguido de forma satisfactoria.

Los objetivos conseguidos se enumeran a continuación:

- Adquisición de las variables meteorológicas mediante distintos sensores controlador por el microcontrolador LPC1768.
- Diseño del sistema de alimentación autónomo.
- Envío de los datos mediante un módulo de comunicación WiFi o GPRS hacia un servidor.
- Almacenamiento de los datos recibidos en base de datos.
- Diseño de una página web sencilla para la representación de los datos, integrando distintos lenguajes de programación.
- Manejo de la API de Google Maps y de la API de HighCharts.
- Diseño de la PCB basada en el microcontrolador LPC1768 con KiCad.

Para conseguir todo lo anterior, se han reforzado conocimientos aprendidos durante la etapa formativa de grado y se han adquirido otros nuevos como todo lo relacionado con el entorno web.

Aunque el resultado final del proyecto ha sido satisfactorio, se puede mejorar en varios aspectos, que por motivos de tiempo no se han podido implementar:

- Realizar una página web más compleja implementando sistemas de seguridad que no se han tenido en cuenta en el proyecto desarrollado.
- Añadir nuevos sensores de forma que se obtendrían más datos del entorno, por ejemplo, sensores de calidad del aire.
- Aunque se ha tenido en cuenta el consumo de energía de los sensores, llevando a modo de bajo consumo los módulos cuando no están realizando su función principal, se puede realizar un estudio práctico del consumo de todos los dispositivos involucrado en el proyecto para realizar un mejor dimensionamiento del sistema de alimentación.
- Insertar el sistema en una red de estaciones meteorológicas para monitorizar los datos meteorológicos de un lugar concreto.

9. Bibliografía

[1] Librería LPC1769 para KiCad.; https://github.com/ExploreEmbedded/Explore-Cortex-M3-LPC1768-Stick-DVB-14001/archive/master.zip

[2] Documentación API de Google Maps.; https://developers.google.com/maps/documentation/

[3] Escala de viento de Beaufort.; https://es.wikipedia.org/wiki/Escala_de_Beaufort

[4] Declinación magnética.; http://www.ign.es/web/ign/portal/gmt-declinacion-magnetica

[5] Interruptor magnético.; https://www.shoptronica.com/interruptor-magnetico-reed-switch/3981-que-son-los-interruptor-magnetico-reed-switch.html

[6] PHP.; http://php.net/

[7] HTML.; https://developer.mozilla.org/es/docs/Web/HTML

[8] HTML, CSS, PHP y JavaScript.; https://www.w3schools.com/

[9] Bajo consumo ESP8266.; http://www.esploradores.com/practica-9-modos-de-ahorro-de-energia-deep-sleep/

[10] LPC176x/5x User Manual.; https://www.nxp.com/docs/en/user-guide/UM10360.pdf

[11] KiCad.; http://kicad-pcb.org/

[12] Datasheet MCP9808.; http://ww1.microchip.com/downloads/en/DeviceDoc/25095A.pdf

[13] Datasheet SHT31-D.; https://www.sensirion.com/fileadmin/user_upload/customers/sensirion/Dokumente/2_Humidity_Sensors/Sensirion_Humidity_Sensors_SHT3x_Datasheet_digital.pdf

[14] Datasheet MAX44009.; https://datasheets.maximintegrated.com/en/ds/MAX44009.pdf

[15] Datasheet VEML6070.; https://www.vishay.com/docs/84277/veml6070.pdf

[16] Datasheet MPL3115A2.; https://www.nxp.com/docs/en/data-sheet/MPL3115A2.pdf

[17] Estación meteorológica PCE-FWS20.; http://www.pce-iberica.es/medidor-detalles-tecnicos/logger-de-datos/logger-datos-pce-fws20.htm

[18] Datasheet HMC5883L.; https://cdn-shop.adafruit.com/datasheets/HMC5883L_3-Axis_Digital_Compass_IC.pdf

[19] Documentos ESP8266.; http://espressif.com/en/support/download/documents?keys=&field_type_tid%5B%5D=14

[20] Documentos SIM808.; http://simcom.ee/documents/?dir=SIM808

[21] ThingSpeak.; https://thingspeak.com/

[22] Documentación API HighCharts.; https://api.highcharts.com/highcharts/accessibility

[23] Datasheet TP4056.; https://dlnmh9ip6v2uc.cloudfront.net/datasheets/Prototyping/TP4056.pdf

[24] Hostinger.; https://www.hostinger.es/

[25] Selectores básicos en CSS.;
https://librosweb.es/libro/css/capitulo_2/selectores_basicos.html

[26] PCB en SeedStudio.; https://www.seeedstudio.com/fusion_pcb.html

[27] Métodos de petición HTTP.; https://developer.mozilla.org/es/docs/Web/HTTP/Methods

[28] PHP+MySQL+Highchart: Mostrar varias gráficas dinámicamente.;
https://geekytheory.com/php-mysql-highchart-mostrar-varias-graficas-dinamicamente

[29] Reflash ESP8266.; http://www.xess.com/blog/esp8266-reflash/

10. Anexos.

10.1. Anexo 1. Keil uVision.

El entorno de desarrollo utilizado en este proyecto es el Keil uVision 5. Es una herramienta software que sirve para desarrollar proyectos que están basados en microcontrolador. Las principales funciones que lleva a cabo este software son, compilar, simular, depurar y cargar el código en el microcontrolador.

La carga en placa y depuración del programa se llevan a cabo mediante un adaptador JTAG (Joint Test Action Group) a través del puerto USB del ordenador.

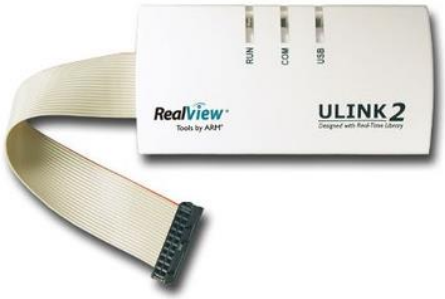

Figura 78. JTAG ULINK2 de ARM.

10.2. Anexo 2. KiCad.

Para el diseño de la PCB basada en el microcontrolador LPC1768, se ha utilizado el software de diseño hardware KiCad. Es un software libre y de código abierto para el diseño de circuitos impresos (PCB). Este software es muy sencillo de usar e intuitivo. La versión utilizada en este proyecto es la 4.0.6 en español.

Las principales partes que componen la PCB diseñada son el microcontrolador LPC1768, parte de alimentación, interfaz de depuración JTAG, osciladores para el reloj principal y para el RTC y un conjunto de pines.

Los pasos generales que se han seguido para el diseño de la placa PCB, son:

- Creación del esquemático mediante componentes que están en las librerías que proporciona Kicad y otros componentes añadidos.
- Asociación de cada componente con su footprint.
- Colocación de las huellas de cada componente en el lugar deseado dentro del contorno de la PCB y ruteo manual de las conexiones.
- Generación de los ficheros Gerber.
- Comprobación de los ficheros Gerber y realización del pedido de fabricación de la PCB.

A continuación, se explican detalladamente los pasos seguidos con el apoyo de imágenes que ilustren el proceso a la vez que se explica el funcionamiento del software KiCad.

En primer lugar, al abrir KiCad, hay que crear un nuevo proyecto, para ello se pulsa sobre “Nuevo proyecto”, se asigna una carpeta de trabajo y un nombre (en este caso “proyecto”) y aparece el siguiente menú:

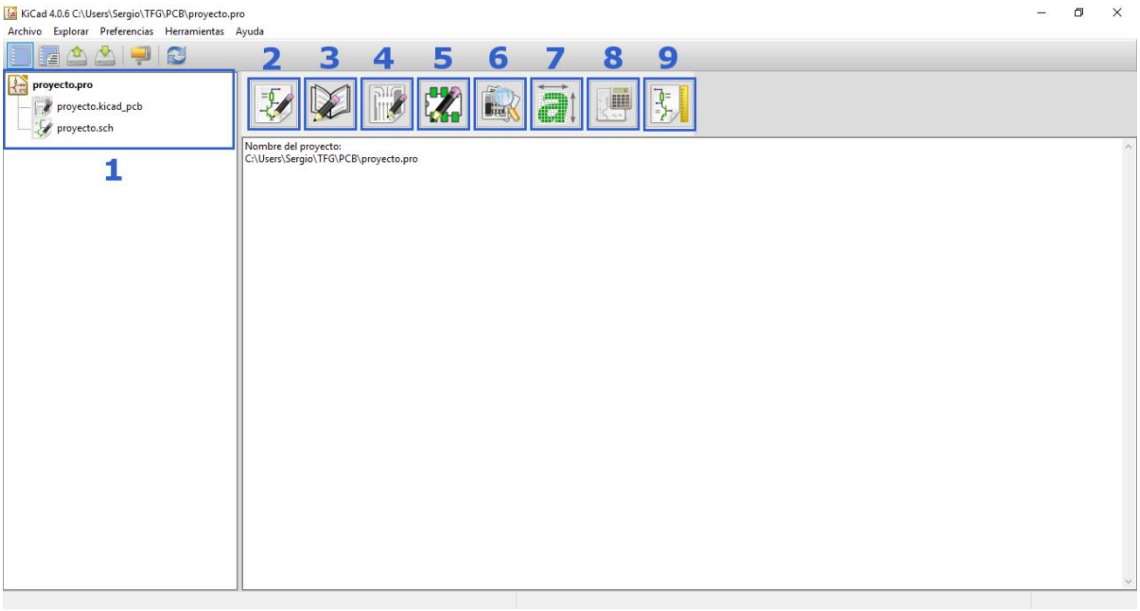

Figura 79. Entorno de trabajo en KiCad.

En ella se han remarcado de color azul las opciones más representativas y asignado un número a cada una para proceder a explicar su funcionalidad:

- Número 1: en esta sección se muestra los archivos presentes en el directorio de trabajo. Se puede ver que al crear un nuevo proyecto se han añadido dos archivos, uno con extensión “.kicad_pcb” y otro con extensión “.sch”.
- Número 2: es la herramienta Eeschema, sirve como editor de esquemas electrónicos, es decir, es donde se crean los esquemáticos del proyecto.
- Número 3: es el editor de librerías. Con él se realizan todas las gestiones relativas a las librerías de símbolos, ya sea para editarlos, crearlos, etc.
- Número 4: es la herramienta Pcbnew. Herramienta para diseñar el PCB, donde se emplazan los componentes y se realiza el rutado entre ellos.
- Numero 5: es el editor de huellas. Utilizada para la gestión de las huellas de los componentes a utilizar en el proyecto, para crearlas, modificarlas o visualizarlas.
- Número 6: es la herramienta GerbView. Se utiliza como visor de archivos Gerber, con la que se puede comprobar visualmente si los archivos de fabricación generados son correctos.
- Número 7: es la herramienta Bitmap2Component. Sirve para convertir imágenes en mapa de bits a elementos de Eeschema o PcbNew, de forma que aparezcan como serigrafía en la PCB.
- Número 8: es una calculadora de PCB. Tiene como utilidad realizar cálculos electrónicos, como por ejemplo la anchura de las pistas.
- Número 9: es el editor PI. Se utiliza como editor de la hoja de trabajo.

Para comenzar con el diseño de la placa, hay que abrir el editor de esquemas Eeschema. Se procede a crear el esquemático del diseño mediante la colocación de componentes e interconexión de ellos.

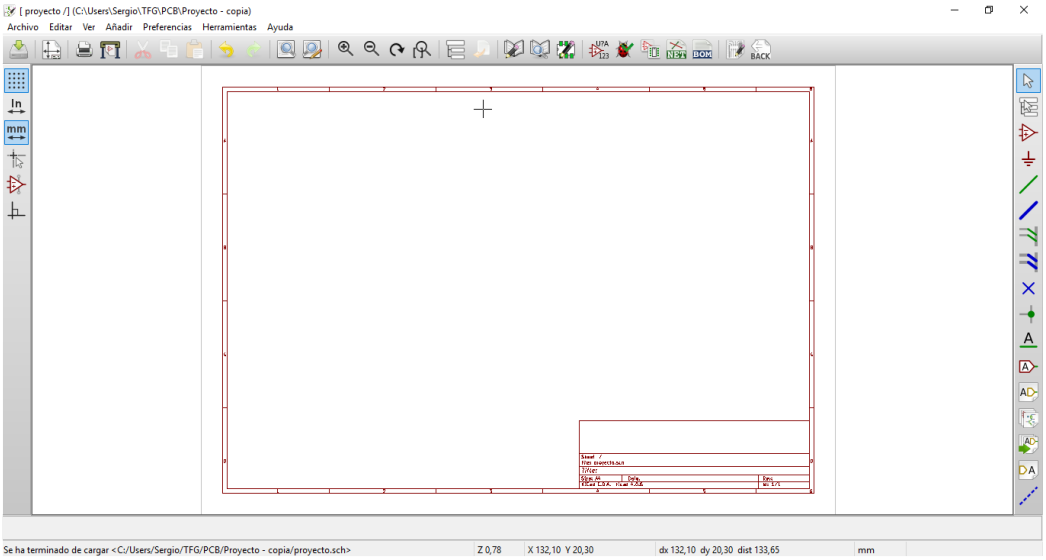

Figura 80. Editor de esquemas de KiCad.

A continuación, se buscan los componentes necesarios para formar el esquema. Es un esquema simple que contiene resistencias, condensadores, bobinas, pines de entrada/salida, dos osciladores, un regulador LM1117-3.3 y el microcontrolador LPC1768.

Todos los elementos excepto el microcontrolador LPC1768 se encuentran en las librerías de KiCad, por lo que se prestan dos opciones, crear un nuevo componente en KiCad o buscar una librería para incorporarla al diseño, optando por la segunda. La librería encontrada contiene el esquemático y el footprint del LPC1769, que es idéntico al microcontrolador LPC1768. Para poder utilizarla en el diseño es necesario añadirla al proyecto mediante la opción “Librerías de componentes” y ya estaría lista para usar.

Una vez que se tienen presentes todos los componentes a utilizar en el diseño se van añadiendo al esquemático mediante la opción “Añadir -> Componente”:

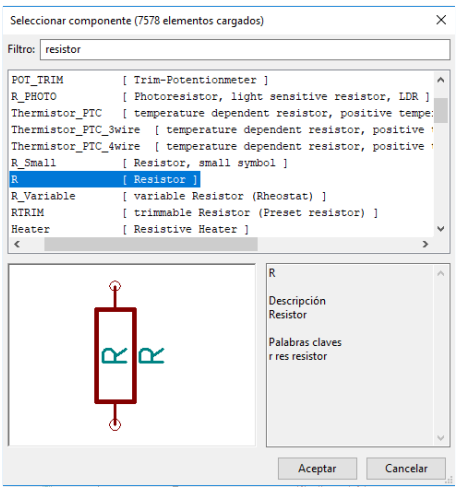

Figura 81. Añadir componentes al esquema.

Una vez añadidos todos los componentes se procede a su interconexión con la herramienta “Añadir línea” o mediante el uso de etiquetas locales con la herramienta “Añadir nombre de red – etiqueta local” para evitar dibujar líneas de conexión muy largas y de esta forma queda el esquemático más visual y limpio.

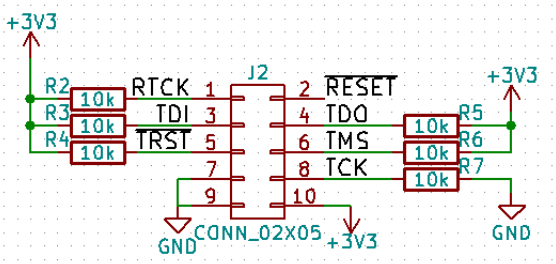

Figura 82. Ejemplo de uso de etiquetas.

También se usan los componentes de alimentación de 3.3V y GND, los cuales deben ir conectados a etiquetas de “PWR_FLAG” para que en la comprobación de errores no se produzca ningún fallo.

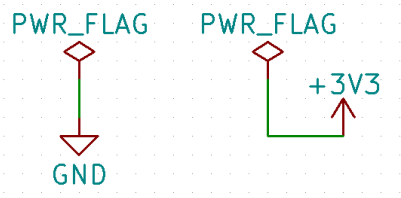

Figura 83. Componentes de alimentación.

Los pines que no se utilizan se deben señalizar con la herramienta de “Añadir marca de no conectado”:

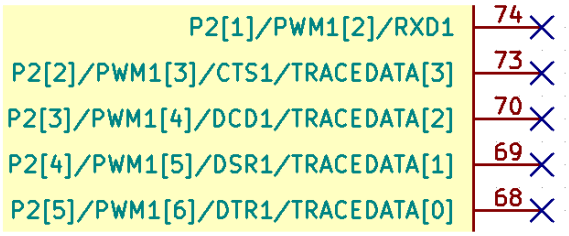

Figura 84. Ejemplo de pines no conectados.

Una vez realizadas todas las conexiones se procede a referenciar todos los componentes mediante la asignación de un nombre único y un valor, para ello se hecho con un orden de izquierda a derecha y de arriba abajo.

Después de realizar todos los pasos anteriores se procede a hacer un test eléctrico al esquemático mediante la herramienta “Control de las reglas eléctricas (ERC)” con las siguientes características:

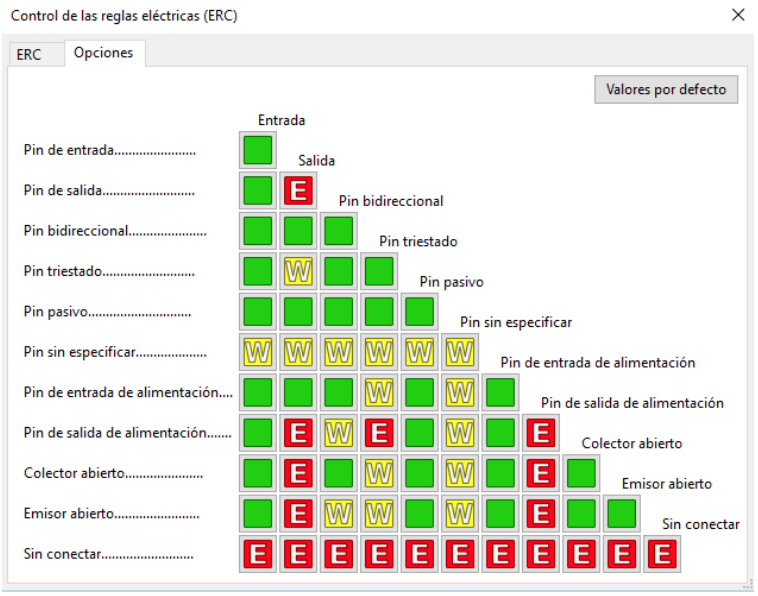

Figura 85. Control de las reglas eléctricas.

Tras verificar que el esquemático no tiene ningún error se usa la herramienta Cvpcb que sirve para asignar a cada componente su huella.

Las huellas de los componentes o también conocidas como footprints especifican las características físicas del componente para que a la hora de soldar quede en perfectas condiciones, dependen del componente y se deben consultar en las hojas de características de los componentes que se vayan a utilizar.

Para los condensadores, resistencias y bobinas se usan huellas de 0603, excepto para dos condensadores que se usan de 1812, se selecciona la versión de soldado a mano ya que las huellas son un poco más grandes de forma que se facilita la soldadura a mano.

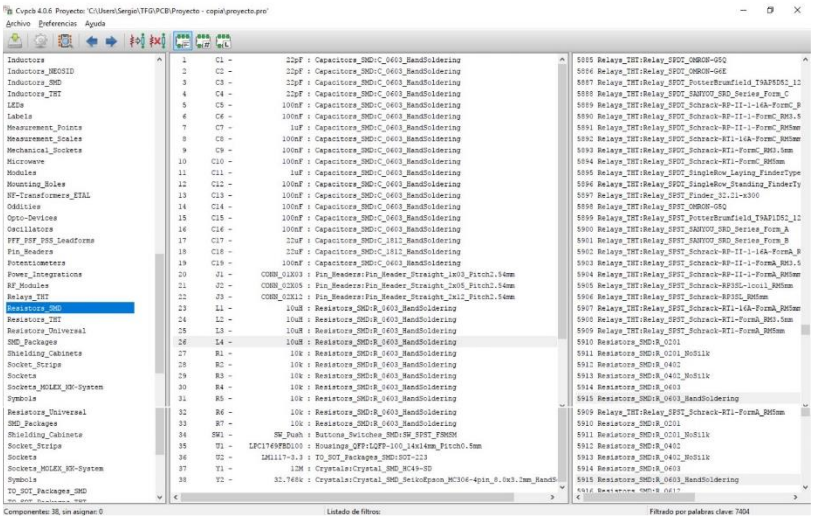

Figura 86. Asignación de componentes a huellas.

Tras la asignación de componentes con sus respectivas huellas se procede a generar la netlist con la herramienta “Generar listado de redes”:

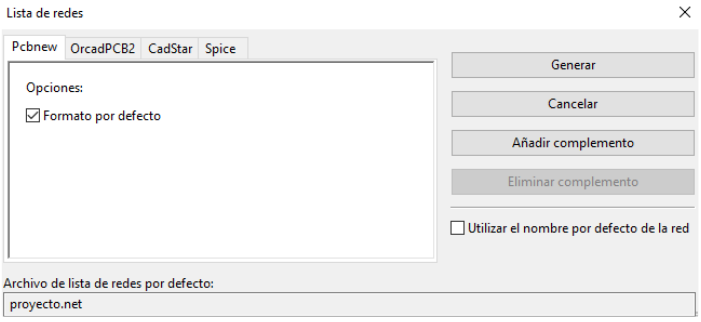

Figura 87. Generar netlist.

Se genera un archivo con extensión .net que contiene toda la lista de conexiones.

Una vez realizado todo lo anterior es el momento de comenzar con el rutado de la PCB. Para ello se ejecuta la herramienta “PcbNew” en la que aparece una hoja de trabajo vacía. El siguiente paso es añadir las huellas con sus respectivas conexiones creadas anteriormente, para ello se lee la netlist generada anteriormente:

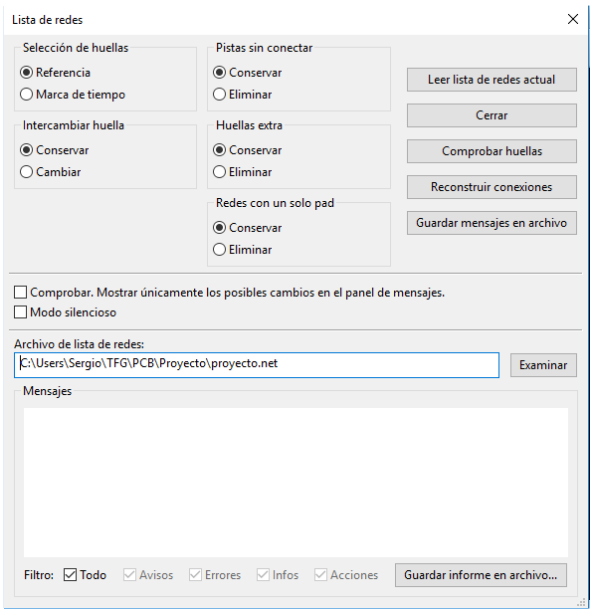

Figura 88. Apertura de la netlist.

Ahora en la hoja de trabajo aparecen todas las huellas de los componentes.

Primero se definen los límites de la PCB, en este caso un cuadrado de 45 mm de lado. Luego se van emplazando las huellas dentro de los límites de la PCB y se va realizando su conexionado. El ruteo de las conexiones se realiza a mano debido a que no son muchas y a que el ruteador que incorpora por defecto KiCad no es muy eficiente.

Se ha optado por un diseño de dos capas con montaje SMD de los componentes en la parte superior de la placa, excepto los conectores que son SMT, por los que habrá que soldarlo en la parte inferior.

Es necesario marcar las capas de trabajo con las que se han trabajado en el diseño de la PCB.

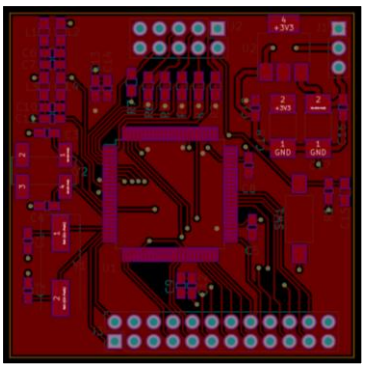

Figura 89. Capa F.Cu.

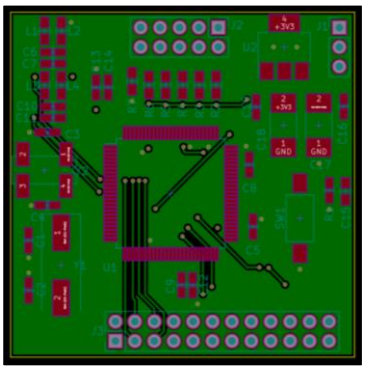

Figura 90. Capa B.Cu.

Al igual que se hizo el test DRC en el esquemático, también hay que realizarlo en el diseño de las placas para comprobar que no se ha cometido ningún fallo.

Una vez comprobado que el proyecto no tiene ningún error de diseño, se procede a generar los archivos gerbers. Para ello hay que tener en cuenta el formato que acepta el fabricante al que se enviarán para su fabricación. En este caso la empresa a la que se enviarán es SEEDSTUDIO, que claramente indica el formato de los gerbers.

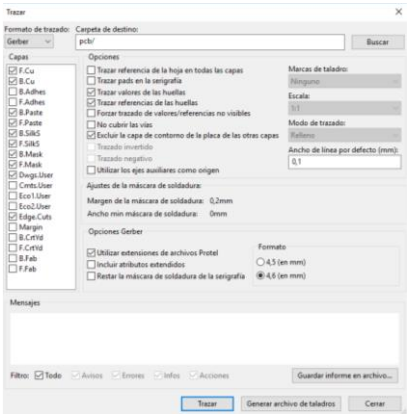

Figura 91. Generación de Gerbers.

El resultado final de la PCB es:

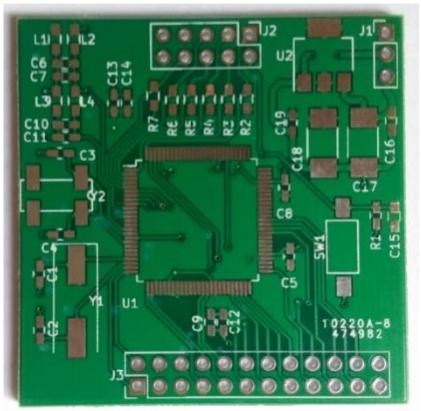

Figura 92. Parte superior de la PCB.

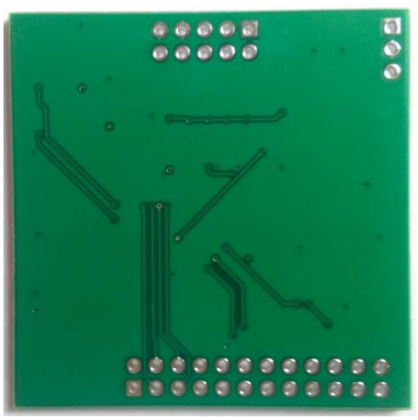

Figura 93. Parte inferior de la PCB.

10.3. Anexo 3. Esquemático de la PCB diseñada.

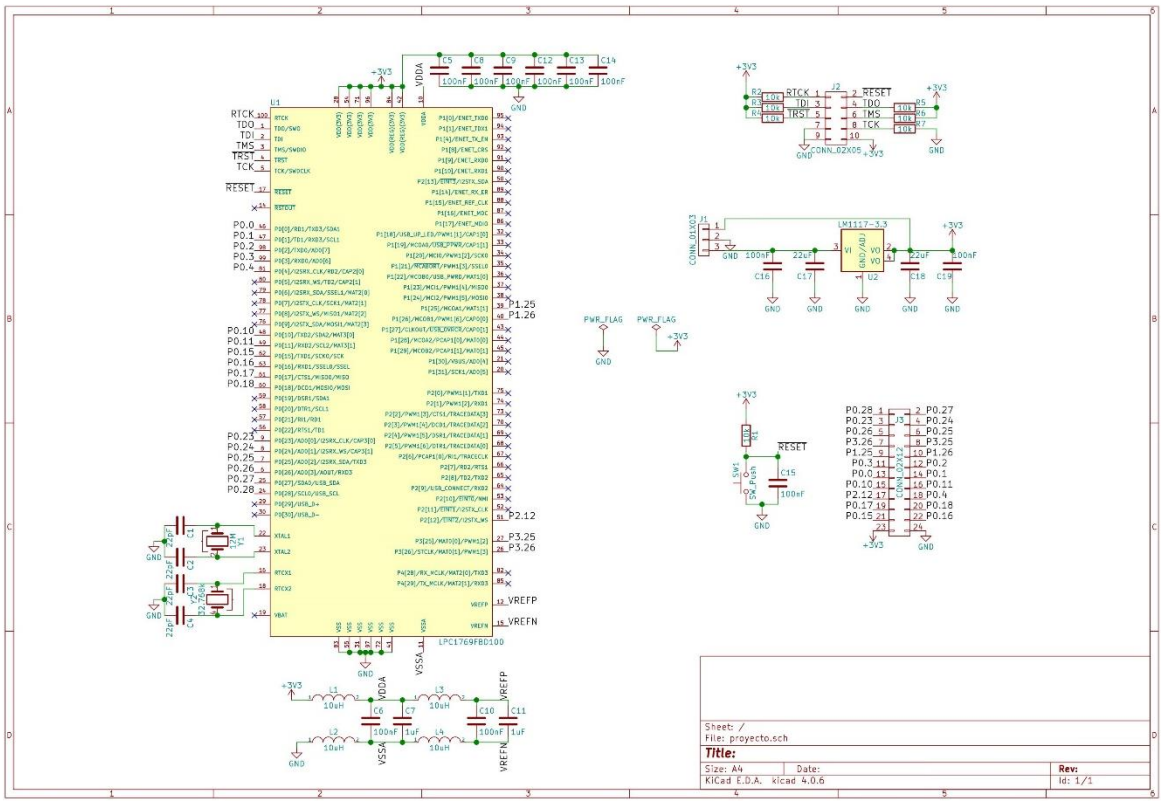

Figura 94. Esquemático de la PCB diseñada.

10.4. Anexo 4. Esquemático del módulo ESP8266-V01.

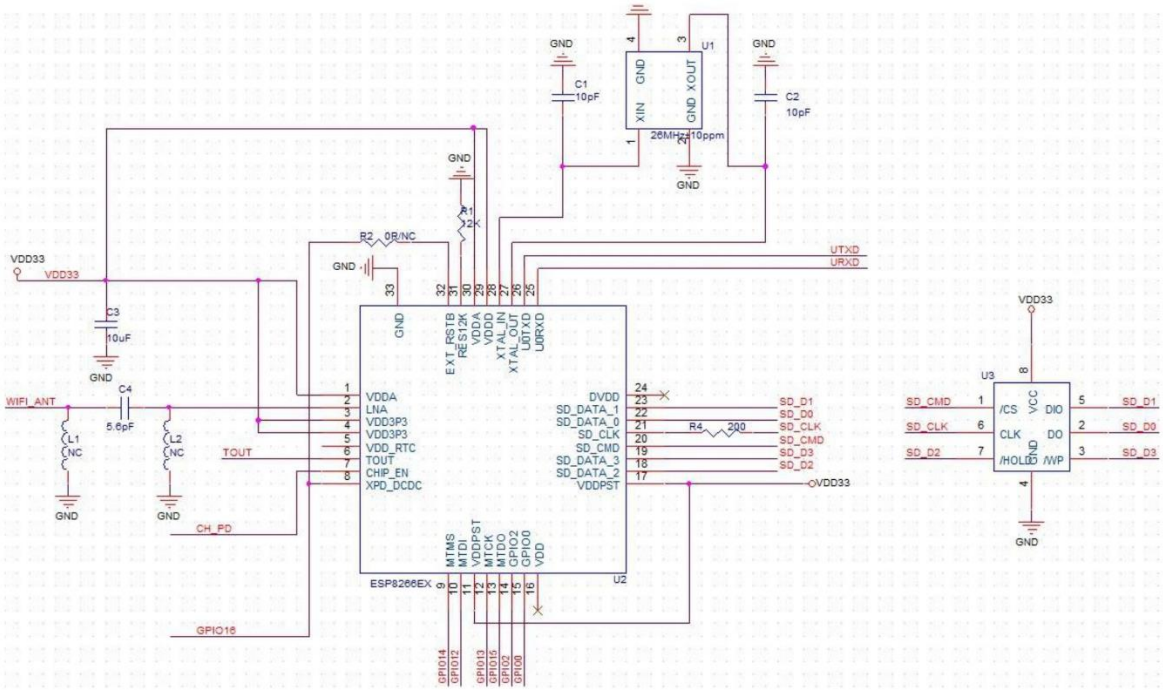

Figura 95. Esquemático del módulo ESP8266-V01.

10.5. Anexo 5. Esquemático del módulo SIM808.

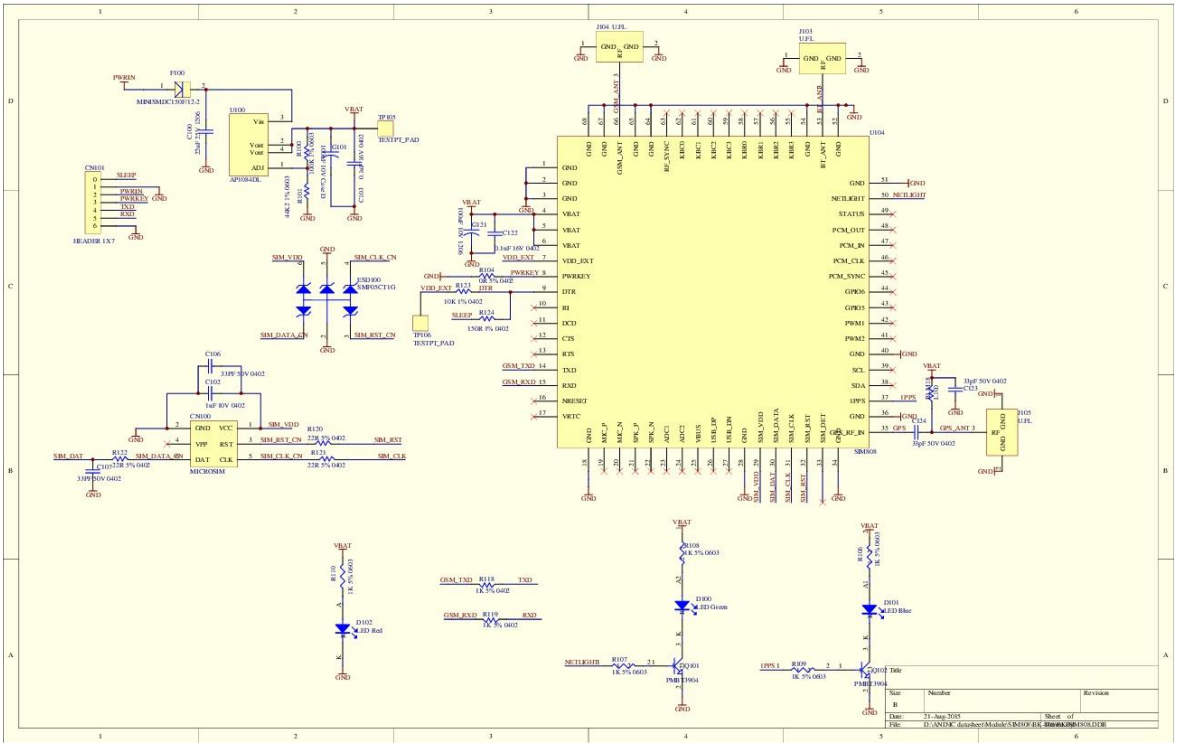

Figura 96. Esquemático del módulo SIM808.

Universidad de Alcalá
Escuela Politécnica Superior

ESCUELA POLITECNICA
SUPERIOR

Universidad
de Alcalá