UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

GRADO EN INGENERÍA ELECTRÓNICA INDUSTRIAL Y AUTOMÁTICA

DESARROLLO DE UNA ESTACIÓN METEOROLÓGICA USANDO UNA RASPBERRY-PI

AUTOR: OLMO MOYA SOLAZ

TUTOR: CARLOS DOMÍNGUEZ MONTAGUD

Curso Académico: 2015-2016

RESUMEN

Este proyecto consiste en la elaboración de una estación meteorológica para controlar y monitorizar las variables de cualquier sistema que requiera un entorno estable para su correcto funcionamiento.

Este proyecto funciona mediante un servidor, alojado en el microcontrolador. De esta manera cualquier usuario puede consultar todas las gráficas a través de un sitio web alojado en el servidor permitiendo un acceso rápido e intuitivo.

La información recogida por los diferentes sensores se almacena en una base de datos alojada también en el servidor. La base de datos sirve de enlace entre los sensores y el servidor web. Se trata de una base de datos en MySQL.

Para la elaboración de la estación se utiliza un microcontrolador denominado Raspberry-Pi.

Para el proyecto se ha escrito mediante varios lenguajes informáticos. Para la programación del servidor se ha utilizado PHP, HTML y Javascript. El programa principal, encargado de la comunicación de los sensores, se ha programado en Python.

OVERVIEW

The main goal of this project is to elaborate a weather station that is used to control and oversee the atmospheric variables that a system may need for the desired performance.

This project works with a server, which is placed in the microcontroller. Due to this, any user is able of see the different graphics and stats in devices with internet access in a fast and intuitive way.

The data measured by the sensors is placed in a database, which is located also in the server. The database is the link between the sensors and the web server. It is a MySQL database.

For designing and building the weather station, a microcontroller called Raspberry-Pi is used.

For the project, several programing languages have been used: PHP, HTML, Javascript. The main program has been coded with Python.

TABLA DE CONTENIDO

LISTA DE FIGURAS

Número *Página*

LISTA DE TABLAS

AGRADECIMIENTOS

Este proyecto no hubiera sido posible sin la ayuda de muchas personas.
Me gustaría agradecer a mi tutor, Carlos Domínguez, por su ayuda.
A todos mis compañeros y amigos que me han acompañado tanto en la carrera como en mi año de erasmus, así como, a mis compañeros de trabajo. También a Buse Chaglayan por su ayuda continua.
Por último, me gustaría dar las gracias a toda mi familia.

INTRODUCCIÓN

Existen numerosas razones para querer monitorizar las condiciones atmosféricas de un entorno. Para el cultivo de ciertos vegetales en invernaderos resulta imprescindible el estudio y análisis de temperatura, humedad, luminosidad etc. Los animales en terrarios o acuarios deben cumplir con ciertas condiciones atmosféricas para garantizar la supervivencia de estos. También es muy recomendable un estudio ambiental previo a la construcción de instalaciones fotovoltaicas, eólicas etc. En resumen, el monitoreo y análisis de diversos datos meteorológicos resulta útil para numerosas actividades, contribuyendo la rentabilidad de estas.

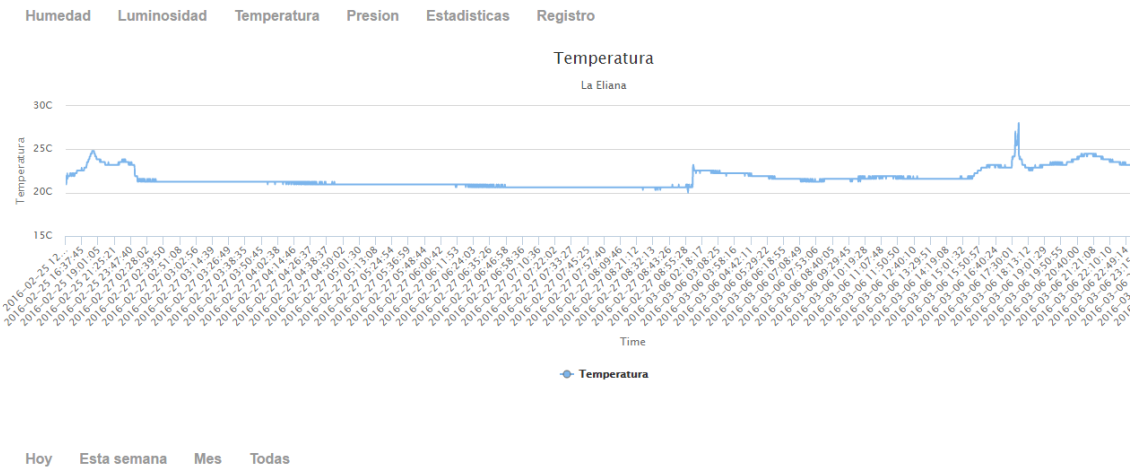
Con el auge de internet también ha crecido la tendencia a poder disponer de toda la información en cualquiera de nuestros dispositivos, en cualquier parte del mundo. Hoy en día, muchos de los objetos cotidianos disponen de acceso a internet. Estos muestran información en la red permitiendo también controlar actuadores. Un buen ejemplo de esta corriente, denominada el Internet de las Cosas, es la domótica. Cualquier usuario de una vivienda inteligente puede consultar la temperatura, en tiempo real, de su casa desde su teléfono y encender el termostato si así lo desea. Este proyecto seguirá el espíritu del internet de las cosas. Se busca que la estación esté conectada a internet de forma que esta pueda ser accesible desde cualquier lugar y de la misma forma esta pueda comunicar sus alertas en cualquier momento al usuario. Esto también otorga una utilidad muy importante al permitir realizar la consulta desde cualquier dispositivo con acceso a internet pudiendo monitorizar y controlar el entorno deseado en cualquier momento.

Existen numerosas estaciones meteorológicas en el mercado, pero el precio es más elevado. Además, normalmente estas no almacenan datos durante un periodo largo de tiempo, así como tampoco permiten configurar alertas para el usuario o consultar esta desde cualquier dispositivo.

Este proyecto va a tratar de cubrir todas estas necesidades. Se creará una estación meteorológica de bajo coste que supere muchas de las funcionalidades de las diferentes estaciones disponibles en el mercado. Esta analizará una serie de sensores y comunicará la información recogida a internet donde se podrá consultar fácilmente en forma de gráfica. Todo esto será almacenado en un servidor Apache. El servidor contiene la información a la que el

usuario podrá acceder mediante su navegador. De esta manera se podrá acceder y monitorizar la temperatura, humedad etc. desde cualquier dispositivo con acceso a internet.

Para este propósito se ha utilizado un ordenador de bajo coste denominado Raspberry-Pi, concretamente el modelo B. Esta tarjeta recibirá los datos de los sensores y los gestionará. El servidor, también instalado en el microcontrolador Raspberry-Pi almacenará la información en una base de datos y mostrará esta en forma de gráfica, pudiendo así consultar los valores de una forma visual.

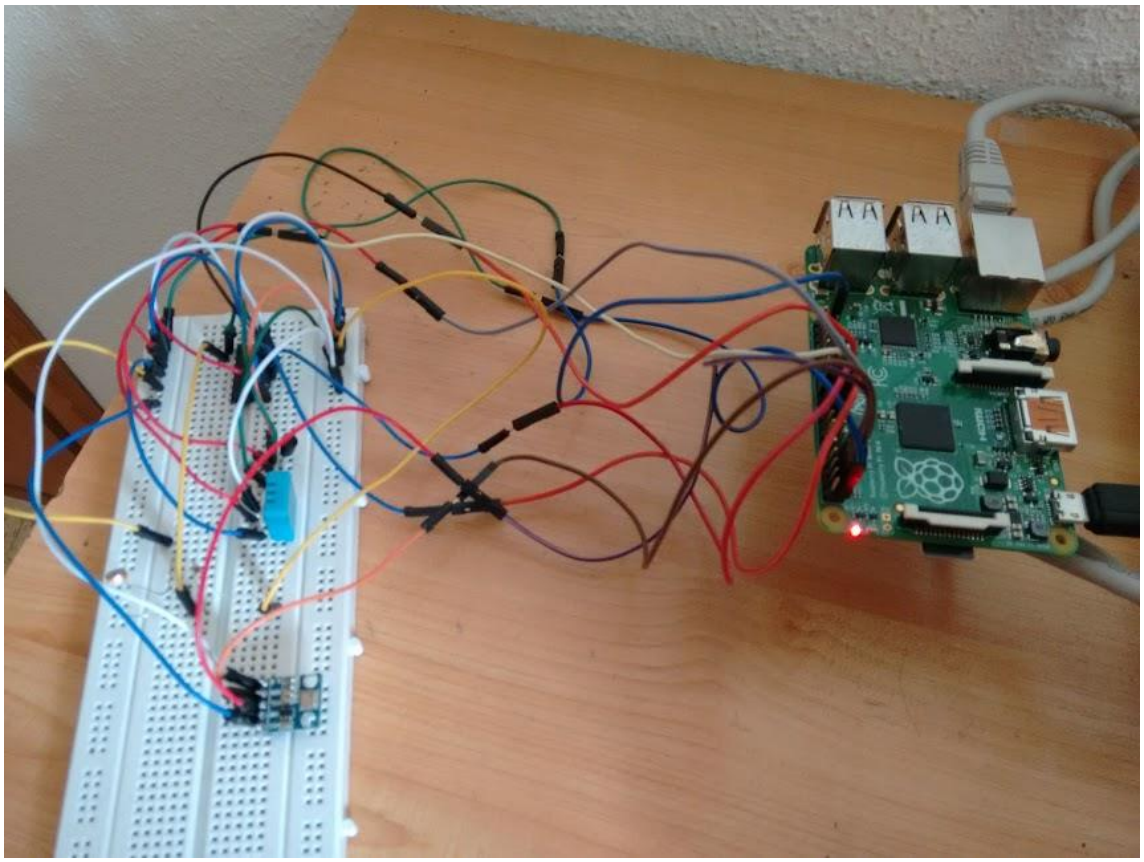


La estación estará equipada con varios sensores que comunicarán con el microcontrolador. Este recibirá la información y la escribirá en una base de datos ubicada en un servidor instalado en la tarjeta. Este servidor leerá la base de datos y mostrará los datos de forma gráfica.

También se dispone de un sistema de alarmas que enviará mensajes de error cuando se alcance uno de los valores críticos. Estos valores indicarán los valores máximos y mínimos, así como el tiempo de muestreo con el que el microcontrolador enviará los datos.

Una de las premisas de este proyecto es que el usuario pueda acceder y consultar la información manera simple y efectiva. La navegación a través de la página web de la estación será mediante un menú. El periodo de tiempo y la manera de visualizar los valores será configurable por el usuario desde la pantalla, así como el periodo de tiempo entre las diversas medidas que se desean realizar o los valores en los que se deberá alertar al usuario. Se ha tratado también de realizar las gráficas de la manera más visual y sencilla posible, pero sin perder ningún tipo de funcionalidad. El proyecto permitirá, por ejemplo, poder visualizar las estadísticas de los

valores recogidos: Máximos, mínimos, medios en diferentes periodos de tiempo. De esta manera el usuario tendrá acceso a una gran cantidad de información.



CAPÍTULO 1

OBJETIVOS Y MOTIVACIÓN PERSONAL

1.1 ESTADO DEL ARTE

Existen numerosas estaciones meteorológicas en internet de diferentes características cada una. Bien es cierto que, las estaciones meteorológicas pensadas para monitorizar las variables atmosféricas de un entorno suelen tener un precio superior en la mayoría de los casos. La mayor parte de estaciones disponibles tampoco disponen de una base de datos que permita mostrar la evolución de las medidas a lo largo de un periodo de tiempo o un servicio de alertas al usuario.

Existe por tanto una necesidad. Una estación meteorológica de bajo coste a la que se pueda acceder desde internet pensada para monitorizar entornos. También se ofrece la posibilidad de una estación meteorológica configurable por el usuario. La idea de este proyecto es cubrir esta necesidad.

1.2 OBJETIVOS

La estación meteorológica desarrollada en este proyecto deberá estar dotada de las siguientes características para lograr el objetivo inicial de sencillez, conectividad y coste reducido que se había planteado al inicio del proyecto:

-Una interfaz simple e intuitiva.

-Un coste inferior a las estaciones similares disponibles en el mercado.

-La posibilidad de acceder desde cualquier dispositivo en cualquier parte del mundo.

-Poder analizar y filtrar la información recogida por fechas.

-Que la estación envié unas alertas automáticas al usuario según unos valores fijados por el mismo.

1.3 MOTIVACIÓN PERSONAL

Al empezar este proyecto se pensaron muchas formas para comunicar los sensores con un ordenador personal como, por ejemplo, usando el puerto serie RS232. Al final se optó por utilizar un servidor web debido a la mayor conectividad que ofrece internet. También se valoró la oportunidad de aprender nuevos lenguajes de programación que ofrece el proyecto.

Se han utilizado numerosos lenguajes para la realización de este proyecto (Pyton, HTML, PHP, Javascript, SQL) y se ha perfeccionado el manejo de estos.

En resumen, el proyecto ha ofrecido una gran cantidad de posibilidades para aprender sobre nuevos contenidos que además son muy demandados por la industria actual por lo que la realización de este resulta de enorme interés.

CAPÍTULO 2

LECTURA DE LOS SENSORES Y MANEJO DEL MICROCONTRALDOR

2.1 ELECIÓN DEL MICROCONTROLADOR

Durante los últimos años ha surgido una gran cantidad de nuevos tipos de microcontroladores. Los dos que se barajaron principalmente para el proyecto están pensados para enseñar a estudiantes conceptos de electrónica y programación, pero debido a su potencia y bajo precio, también resultan herramientas atractivas para el desarrollador.

Para la realización del proyecto se dudó, como se ha dicho, entre la utilización de dos microcontroladores de precio similar entre ellos. A continuación, se describirán y se explicará el porqué de la decisión final de utilizar la Raspberry-Pi.

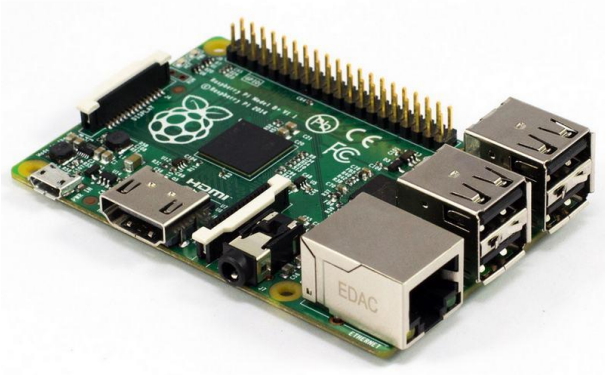
-Arduino uno: Una de sus principales ventajas es la de poder trabajar con entradas y salidas analógicas. Su programación y manejo es más sencillo que el de su competidor. También dispone de un gran número de pines de propósito general. Por el contrario, su potencia de cálculo no es tan potente como la del microcontrolador Raspberry-Pi.

-Raspberry Pi 2 B: Esta tarjeta es en esencia un ordenador de bajo coste. Funciona bajo un sistema operativo basado en Debian denominado Raspbian. Su principal ventaja es su potencia de cálculo y la facilidad de instalar un servidor en esta. Su mayor desventaja ha sido la ausencia de entradas analógicas, lo que ha obligado a tener que adquirir más hardware para poder trabajar con estas.

Como se ha mencionado al final se optó por la Raspberry-Pi por su mayor funcionalidad a la hora de montar un servidor web.

	RPI Model A	RPI Model A+	RPI Model B	RPI Model B+	RPI 2 Model B
SoC	Broadcom BCM2835	Broadcom BCM2835	Broadcom BCM2835	Broadcom BCM2835	Broadcom BCM2836
CPU	ARM11 ARMv6 700 MHz.	ARM11 ARMv6 700 MHz.	ARM11 ARMv6 700 MHz.	ARM11 ARMv6 700 MHz.	ARM11 ARMv7 ARM Cortex-A7 4 núcleos @ 900 MHz.
GPU	Broadcom VideoCore IV 250 MHz. OpenGL ES 2.0	Broadcom VideoCore IV 250 MHz. OpenGL ES 2.0	Broadcom VideoCore IV 250 MHz. OpenGL ES 2.0	Broadcom VideoCore IV 250 MHz. OpenGL ES 2.0	Broadcom VideoCore IV 250 MHz. OpenGL ES 2.0
RAM	256 MB LPDDR SDRAM 400 MHz.	256 MB LPDDR SDRAM 400 MHz.	512 MB LPDDR SDRAM 400 MHz.	512 MB LPDDR SDRAM 400 MHz.	1 GB LPDDR2 SDRAM 450 MHz.
USB 2.0	1	1	2	4	4
Salidas de vídeo	HDMI 1.4 @ 1920x1200 píxeles	HDMI 1.4 @ 1920x1200 píxeles	HDMI 1.4 @ 1920x1200 píxeles	HDMI 1.4 @ 1920x1200 píxeles	HDMI 1.4 @ 1920x1200 píxeles
Almacenamiento	SD/MMC	microSD	SD/MMC	microSD	microSD
Ethernet	No	No	Sí, 10/100 Mbps	Sí, 10/100 Mbps	Sí, 10/100 Mbps
Tamaño	85,60x56,5 mm	65x56,5 mm.	85,60x56,5 mm	85,60x56,5 mm	85,60x56,5 mm
Peso	45 g.	23 g.	45 g.	45 g.	45 g.
Precio	25 dólares	20 dólares	35 dólares	35 dólares	35 dólares

En la anterior imagen se puede observar las diferentes especificaciones de las tarjetas Raspberry-Pi. Se eligió de entre estas tarjetas el modelo B+ al tener mayor versatilidad para futuros proyectos.



2.2 MATERIAL EMPLEADO

Se han utilizado varios sensores y dispositivos para poner en funcionamiento la estación meteorológica.

-Sensor de temperatura LM35: Se trata de un sensor de muy bajo precio y fácil de encontrar. También asegura una precisión suficiente para el proyecto garantizando un error de aproximadamente 0.5ºC.

-Sensor de humedad DHT11: Es un sensor de humedad económico que dispone de mucha documentación en la red. Otra de sus ventajas es que no requiere de circuitos externos complicados para funcionar.

-Conversor Analógico/Digital MCP3008: Resulta útil para este proyecto al disponer de 8 canales de entrada analógica. De esta manera pueden conectarse 8 sensores analógicos usando solo una entrada de la tarjeta Raspberry-Pi. Para la comunicación usara también los pines de SPI.

-Sensor de presión BMP180: se trata de un sensor de presión con comunicación I2C, que mide temperatura, altitud y presión atmosférica con una exactitud adecuada al proyecto.

-LDR: Se trata de una resistencia cuyo valor variara según la intensidad lumínica que le llegue.

2.3 SERVIDOR WEB

Una parte muy importante de este proyecto es el servidor web. Se ha utilizado uno de los esquemas más comunes. Este es el denominado LAMP acrónimo para Linux, Apache, MySQL, PHP.

-GNU/Linux: Es el sistema operativo en el que el servidor funcionara. Es un sistema operativo de código abierto y software libre. El sistema operativo de la tarjeta Rasberry-Pi es Raspbian un sistema basado en Debian, una distribución GNU/Linux, más concretamente, en Debian Wheezy.

-Servidor HTTP Apache: Es un servidor web multi-plataforma de código abierto. Su principal ventaja es su gran popularidad, lo que garantiza que se pueda encontrar una gran documentación en internet.

-MySQL: Es un sistema de gestión de base de datos relacional, multihilo y multiusuario que desde el 2009 se desarrolla como software libre.

-PHP: Se trata de un lenguaje de programación del lado del servidor. Se trata de uno de los primeros lenguajes de estas características que se pueden incorporar directamente al documento HTML.

Se utiliza este esquema debido a la potencia que proporciona y la gran cantidad de opciones que otorga.

CAPÍTULO 3

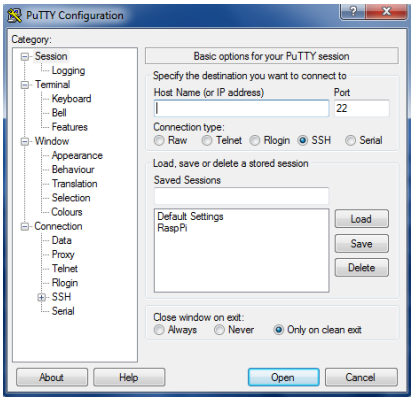
ENSAMBLAJE DE LA ESTACIÓN METEOROLÓGICA

3.1 CONFIGURACIÓN DEL MICROCONTROLADOR

La primera tarea a realizar se trata de configurar el microcontrolador Raspberry-Pi. La tarjeta realizará la mayor parte del trabajo en la estación meteorológica. Es la encargada de comunicarse con los sensores y de controlar el servidor ubicado en ella.

El primer paso es instalar el sistema operativo. El sistema operativo permitirá instalar todos los programas necesarios y permitirá manejar el microcontrolador. La tarjeta Raspberry-Pi cuenta con una ranura para tarjetas microsd. Por el contrario, no dispone de memoria interna. Por ello se debe de instalar, en la tarjeta sd, el sistema operativo. Se ha elegido Raspbian Jessie, se trata de una versión de Debian creada a propósito para la Raspberry-Pi. Este sistema operativo se puede encontrar en la página oficial del microcontrolador y es el más utilizado para esta tarjeta.

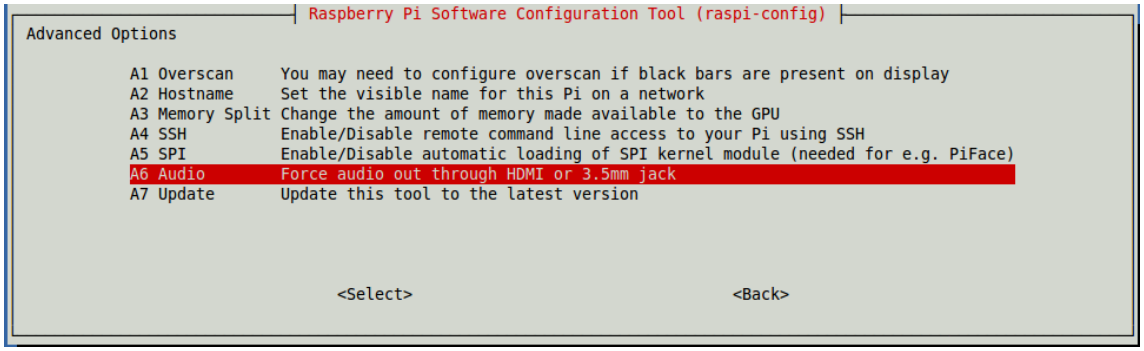
El siguiente paso es conectarse remotamente a la Raspberry-Pi para poder trabajar desde el ordenador sin la necesidad de tener un monitor extra para la tarjeta. Para ello se usara "PuTTY" un programa que permitirá conectarse mediante SSH (Secure SHell) al microcontrolador. Se eligió este sistema al ser el que más rápido permitía trabajar. Tan solo se necesita saber la dirección ip del microcontrolador para poder visualizar la consola del microcontrolador y actuar sobre ella.



Una vez introducida la dirección ip del microcontrolador en la interfaz de PuTTy la consola de Raspbian ya aparecerá en el ordenador, gracias a la conexión SSH. De esta manera ahora se puede configurar la tarjeta para la estación meteorológica.

Se debe activar las interfaces de comunicación SPI e I2C. Estas sirven para comunicar algunos de los sensores que se utilizaran con el microcontrolador como se explicará más adelante.

Ahora ya se puede acceder al menú de configuración de la tarjeta. Al introducir el comando *raspi-config* en el terminal se visualizará el menú que aparece en la siguiente foto. Este permite configurar una serie de características del microcontrolador.



Por ultimo queda la instalación del servidor. Como ya se ha comentado se eligió el esquema LAMP. Por ello se debe instalar Apache y MySQL. Se puede instalar fácilmente desde la consola si la tarjeta Raspberry-Pi tiene conexión a internet. También se decidió instalar Phpmyadmin para un manejo sencillo de la base de datos ya que permite gestionar las diferentes tablas mediante su interfaz sin necesidad de trabajar siempre con código.

3.2 INTERFACES DE COMUNICACIÓN

Se denomina interfaz a la conexión física que permite la comunicación entre dos dispositivos electrónicos de cualquier tipo. Para este proyecto se han utilizado las dos únicas interfaces que existen en el microcontrolador (SPI e I2C) que serán descritas a continuación.

Las interfaces pueden ser síncronas si la transmisión de información depende de los pulsos de un reloj o asíncrona si son independientes del reloj.

Otra de sus características es la comunicación *master/slave*. Uno de los dispositivos, denominado *master* tendrá el control sobre el otro denominado *slave*. Esto se puede elegir en la configuración

de la interfaz ya sea SPI o I2C pasando uno de los dispositivos a comportarse como *master* y el otro como *slave*.

3.2.1 SPI

Siglas en inglés para *Serial Peripheral Interface*. Es el sistema de comunicación más común para comunicar entre dispositivos electrónicos. Su gran ventaja es la sencillez frente a otras interfaces de comunicación. Se trata de una comunicación síncrona, en cada pulso del reloj comunicará o leerá un bit. Para su conexión se utilizan cuatro pines.

-SCLK (*Clock*): Es el pin del reloj. Este mandara los pulsos necesarios para una comunicación síncrona.

-MOSI (*Master Output Slave Input*): Salida de datos del *master* y entrada del *slave*.

-MISO (*Master Input Slave Output*): Entrada de datos del *master* y salida del *slave*.

-SS/Select: Para seleccionar un *slave* o para activar el *slave* desde el master.

3.2.2 I2C

Siglas en inglés para *Inter-Integrated Circuit*. Se trata de una interfaz síncrona. Su principal característica y diferencia con el modo SPI es que la comunicación es en serie en lugar de en paralelo. En la comunicación en serie solo se puede transmitir información en un canal, la comunicación en paralelo permite transmitir información por varios canales de forma simultánea. Esto implica que la comunicación I2C se realizará por un solo cable. Para su utilización se utilizarán tres pines:

-SDA: Canal que transmitirá los datos que se desean comunicar.

-SCL: Se trata del pin de reloj, que funciona de una manera similar al reloj del que dispone la transmisión SPI.

-GND: El pin de conexión a tierra.

Los dos primeros cables necesitaran una resistencia de *pull-up* para funcionar correctamente. Esta se usa para evitar ruido cuando se encuentran a un nivel lógico bajo.

Ambas interfaces deben ser configuradas en el microcontrolador pues vienen desactivadas por defecto. La instalación es sencilla y solo requiere de unos pocos comandos y la instalación de varios módulos extras. Se puede instalar desde la terminal de configuración de la propia tarjeta a la que se accede mediante el comando *raspi-config*. Dentro del menú de configuración se pueden activar las dos interfaces. Posteriormente se deben instalar las librerías de ambas herramientas.

3.3 DISEÑO Y COMUNICACIÓN CON LOS SENSORES

En este apartado se va a hablar sobre el diseño de los circuitos necesarios para el correcto funcionamiento de los sensores. Los valores ambientales elegidos para monitorizar son Humedad, luminosidad, temperatura y presión atmosférica.

3.3.1 MCP3008

Como ya se ha dicho la tarjeta Raspberry-Pi no dispone de entradas analógicas. Por ello se usa un conversor analógico/digital. Se decidió utilizar el MCP3008. Este conversor resulta útil al disponer de comunicación SPI. Su documentación se puede encontrar en : https://cdn-shop.adafruit.com/datasheets/MCP3008.pdf

De esta manera utilizando los 4 pines de la comunicación SPI existentes en el microcontrolador, se puede leer hasta 8 sensores analógicos diferentes.

Configurar el canal SPI resulta sencillo en la Raspberry-Pi. Primero se debe activar la opción en el menú de configuración. Una vez el canal esté disponible se introduce el código en el programa para establecer la comunicación.

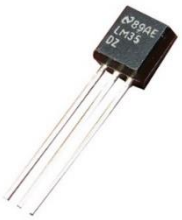
```
def getADC (channel):
        r = spi.xfer2([1, (8+channel) << 4,0])
        adcOut = ((r[1] & 3) << 8) + r[2]
        return adcOut
```

El conversor transformara el valor analógico a un valor digital. Este valor será leído en el programa. En la siguiente imagen vemos el código usado para leer el valor digital y transformarlo otra vez en analógico.

```
        value=getADC(0)
        volts = (value* 3.3)/1024
        temperatura = volts / (10.0/1000)
        print 'temperatura =' ,temperatura
```

3.3.2 LM35

Para medir la temperatura se ha utilizado un sensor LM35. Se trata de un sensor muy común y económico. Tiene una salida analógica de 10mv/ºC y puede medir hasta 150ºC.



Se trata de un sensor analógico, la tarjeta Raspberry-Pi no puede leer valores analógicos. Por ello se debe usar un conversor analógico digital, el MCP3008.

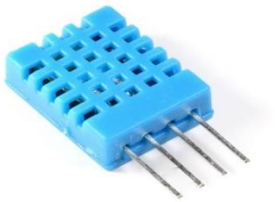
La documentación del sensor se encuentra en: http://www.ti.com/lit/ds/symlink/lm35.pdf

3.3.3 DHT11

Se trata de un sensor de humedad de bajo coste. Tiene un error de ± 5%. El bajo precio de este sensor lo hace uno de los más usados para proyectos. Esto permite encontrar una gran cantidad de documentación y librerías en internet. Se utilizó una suministrada por *Adafruit*.

Se ha usado también, como se puede observar, una resistencia de 10kΩ como pull up necesaria para evitar ruido cuando el nivel lógico es bajo.

El datasheet aparece en la dirección web: http://www.micropik.com/PDF/dht11.pdf



3.3.4 LDR

Para medir la intensidad lumínica se ha utilizado una LDR, una resistencia dependiente de la luz. Se ha realizado un divisor de tensión para obtener un voltaje variable respecto a la cantidad de luz que le llega. Para calibrarla se ha medido el voltaje en oscuridad total y en luminosidad total que se ha simulado con una linterna.

Se ha utilizado el montaje del MCP3008 ya mencionado anteriormente para poder medir el voltaje analógico de salida respecto de la luz.

La documentación se encuentra fácilmente en internet (http://www.gotronic.fr/pj-1284.pdf)

3.3.5 BMP 180

Para medir la presión atmosférica se ha usado un sensor BMP 180. Tiene un rango de presiones desde 300 a 1100hPa. Para comunicarse con la tarjeta utiliza una conexión I2C que ha de ser activada previamente en la tarjeta Raspberry-Pi. Se ha utilizado una biblioteca disponible en la red para obtener las medidas que se almacenaran en la base de datos.

Se puede encontrar información sobre este sensor en la dirección web: https://cdn-shop.adafruit.com/datasheets/BST-BMP180-DS000-09.pdf

3.4 PROGRAMA PRINCIPAL

El lenguaje elegido para la creación del programa principal ha sido Python. Se trata de un lenguaje interpretado, multiplataforma, multiparadigma y de código abierto creado por Guido van Rossum en 1991.

En la tarjeta Raspberry-Pi se encuentra el programa principal encargado, como se ha visto, de comunicar con los sensores. Este código tiene varias tareas.

En primer lugar, se configura la base de datos y el servicio de correo que enviará las alertas a los usuarios. Una vez configurada la base de datos el programa escribirá en ella los valores registrados por los sensores.

También debe leer, únicamente, el último valor registrado en la tabla registro. Para tener en cuenta solo el ultimo valor que el usuario desea configurar como crítico. Esta tabla está formada por los valores introducidos por el usuario en la correspondiente página del servidor.

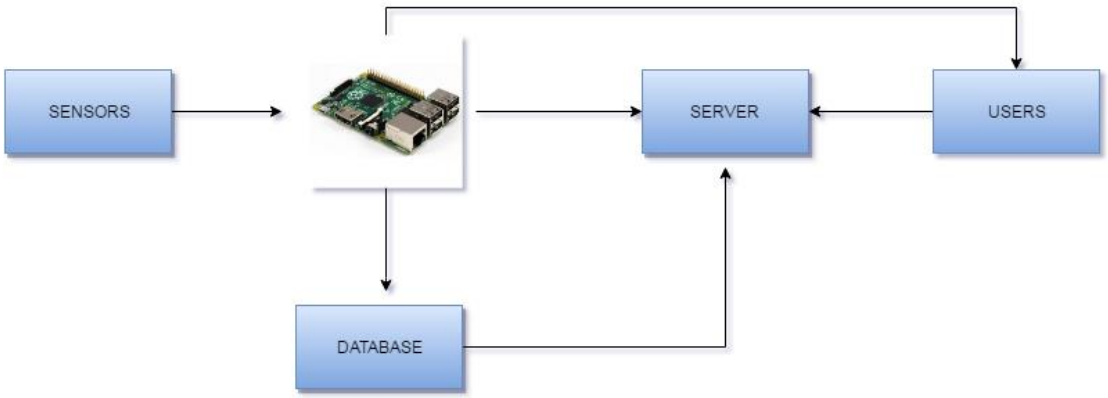
Por último, el programa enviará un correo al usuario en caso de superar cualquiera de los valores críticos introducidos y permanecerá en espera hasta que se cumpla el periodo también introducido por el usuario.

CAPÍTULO 4

PROGRAMACIÓN Y DISEÑO DE LA ESTACIÓN METEOROLÓGICA

4.1 ESQUEMA DE FUNCIONAMIENTO

Como se puede ver en la siguiente imagen el esquema de la estación es sencillo.

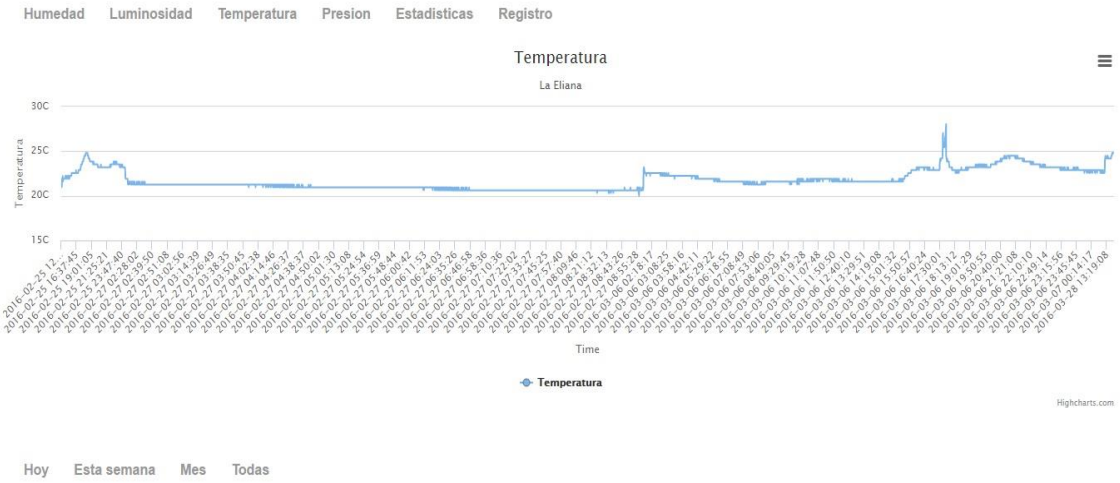


Todo el proyecto está construido alrededor del microcontrolador y el servidor. Los sensores comunican los valores recibidos a la tarjeta Raspberry-Pi, donde son guardados en la base de datos. El servidor se encuentra ubicado en el microcontrolador. Las diferentes páginas web almacenadas en el servidor leen la información de la base de datos creando las gráficas que el usuario puede consultar. Por último, el microcontrolador envía mensajes de alerta a los usuarios

4.2 INTERFAZ

Se ha decidido mostrar la información en forma de gráfica para que resulte intuitiva para el usuario.

El servidor dispondrá de un sencillo menú para navegar entre las distintas gráficas. En cada una de estas se podrá solicitar ver solo los datos del día, semana, mes o año. También se podrá visualizar las mínimas, máximas y valor medio en un periodo de tiempo elegido por el usuario.



En la imagen se puede ver el aspecto final de una de las páginas web del servidor. La gráfica se sitúa en el medio. Esta muestra los valores recogidos y su evolución respecto al tiempo. Arriba se sitúa un menú que permite navegar entre los diferentes datos. Como también, ver las estadísticas de estos o introducir los valores de las alarmas. Por último, abajo se encuentra un menú para poder filtrar los datos respecto al periodo de tiempo que se desea consultar.

Por último, en una de las pestañas se incluirá un registro donde el usuario podrá insertar una serie de valores de utilidad para el funcionamiento de la estación.

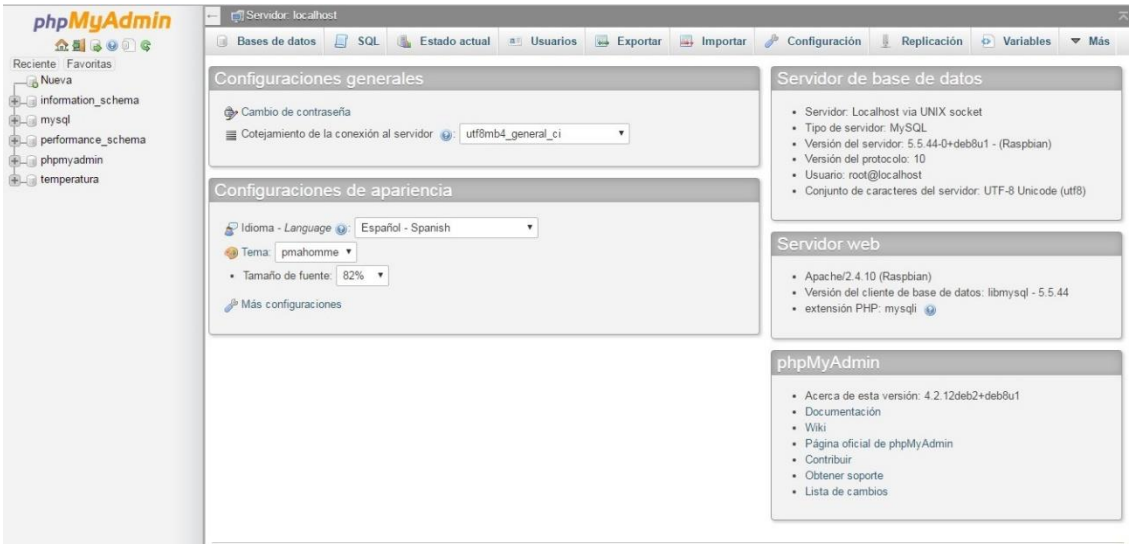
4.3 BASES DE DATOS

El proyecto consta de dos partes bien diferenciadas. Una de ellas se encarga de leer la información de los sensores y almacenarla en una base de datos. La otra se encarga de la lectura de los datos y representar estos en una página web

Para la base de datos se ha utilizado MySQL un sistema de gestión de bases de datos con código libre cada vez más utilizado por los programadores. También se ha utilizado un programa conocido como Phpmyadmin que permite manejar las bases de datos con interfaz. Para poder usar todas las herramientas requeridas se ha tenido que instalarlas previamente en el sistema. Todo esto se ha realizado de forma rápida mediante la consola necesitando tan solo un comando (*sudo apt-get install mysql-server mysql-client php5-mysql phpmyadmin*).

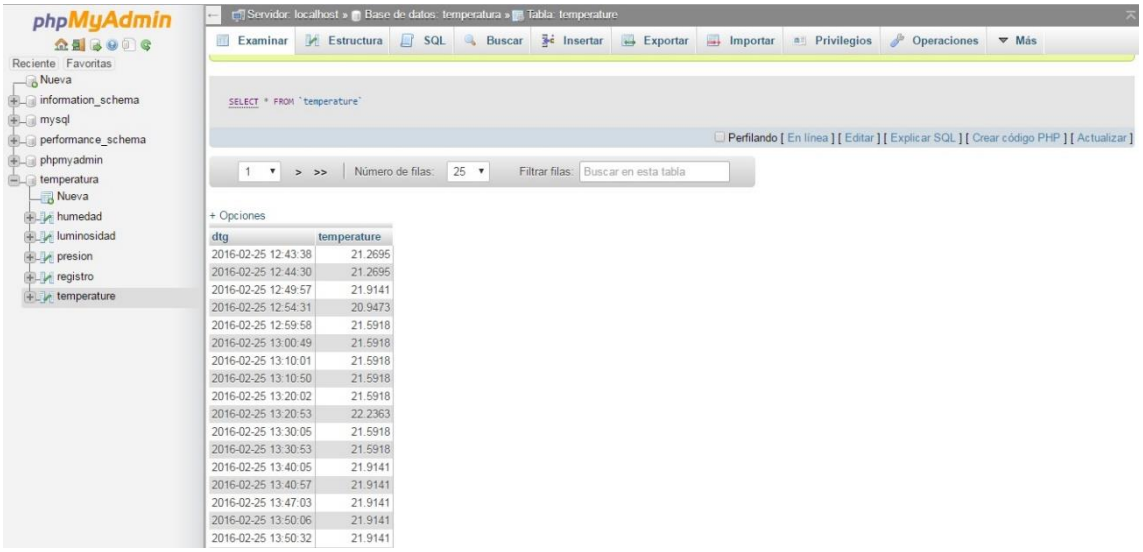
4.3.1 PHPMYADMIN

Phpmyadmin permite, como se ha dicho, manejar la base de datos y sus diferentes tablas de una manera más sencilla. Esto es posible gracias a su interfaz gráfica que evita realizar todas las operaciones con código.



En la imagen superior se puede observar la pantalla de inicio de Phpmyadmin, A la izquierda se encuentran las diferentes bases de datos con sus respectivas tablas. En el menú superior aparecen diferentes pestañas con todas las acciones que se pueden realizar.

Phpmyadmin permite de esta manera, modificar las diferentes bases de datos permitiendo añadir tablas, columnas etc. En la siguiente imagen se puede observar una tabla de la base de datos y como esta puede ser modificada.



4.3.2 MANEJO DE LA BASE DE DATOS

La base de datos que utilizará el servidor se denomina *temperatura*. Esta dispone de diferentes tablas donde recoge la información (Humedad, Luminosidad etc.) Cada una de estas tablas dispone de dos columnas. Una en la que almacenará el valor del sensor y otra en la que guardará la hora y fecha de la medición, que será obtenida de internet.

MySQL permite ejecutar consultas a la base de datos. Estas consultas son muy variadas. Se puede consultar el valor mínimo o máximo de una columna, calcular el valor medio o filtrar la información para obtener solo los valores necesarios.

Esto ha resultado útil, ya que ha permitido obtener las estadísticas de los diferentes valores almacenados y filtrarlos por año, mes, día etc… También ha resultado útil para el registro de los valores máximos y mínimos que el usuario seleccionará para recibir el correo con las alertas.

La estructura de las frases usadas para realizar consultas en SQL (Structured Query Lenguage) siguen siempre el mismo orden. Empiezan con *SELECT* y aparece *WHERE* al final de la consulta. *SELECT* marca que valores se desean mostrar. Se puede pedir una sola columna

poniendo el nombre de esta después del *SELECT* o todos los valores introduciendo un asterisco.

WHERE marca las condiciones que tienen que cumplir los valores mostrados. Por ejemplo, tener todas las mismas fechas. Se pueden poner más condiciones usando *AND* o *OR*.

En la siguiente imagen se puede observar el código de una consulta realizada en PHP. En ella se piden solo los datos del día actual. También se observa cómo se realiza la conexión a la base de datos en PHP.

```php
<?php
$con = mysql_connect("localhost", "root", "raspberry");
if(!$con){
die('Could not connect: ' .mysql_error());
}
mysql_select_db("temperatura", $con);
$diaactual = date("d");
$mesactual = date("m");
$anoactual = date("y");
$result = mysql_query("SELECT * FROM `temperature` WHERE day(dtg) = '$diaactual' AND month(dtg) = '$mesactual'");
while($row = mysql_fetch_array($result)) {
echo $row['dtg'] . "/" . $row['temperature']. "/" ;
}
mysql_close($con);
?>
```

Por último, se debe establecer contacto con la base de datos desde el programa principal. Desde este se guardará la información recogida por cada uno de los sensores. En la siguiente imagen se ve como se realiza la conexión en Python a la base de datos y como se introduce un valor en ella.

```python
pi_hum = humidity
con = mdb.connect('localhost', \
                  'pi_insert', \
                 'raspberry', \
                  'temperatura');
cur = con.cursor()
cur.execute("""INSERT INTO humedad(humedad) \
               VALUES(%s)""", (pi_hum))
```

Como se puede ver en la imagen anterior, para conectarse e introducir valores en la base de datos, se necesita crear un cursor si se usa SQL desde Python. Para insertar valores se usa el comando *INSERT INTO* seguido de la columna a la que se introduce. Hay que poner también el termino *VALUES* para indicar el valor a introducir.

Se ha comentado anteriormente la utilidad de MySQL en este proyecto. Se ha tenido que realizar numerosas consultas para poder obtener los datos deseados. A continuación, se verán algunas de las consultas realizadas.

```
$mesactual = date("m");
$anoactual = date("y");
$result = mysql_query("SELECT * FROM `humedad` WHERE day(dtg) = '$diaactual' AND month(dtg) = '$mesactual'") or die ("Connection error");
```

En la imagen anterior, se puede ver la consulta usada para obtener los valores del día actual. Se filtran los valores por el día actual y el mes actual.

En la siguiente imagen aparece el código necesario para obtener los valores máximos, medios y mínimos. Con esto se construyen las estadísticas mostradas por la estación meteorológica.

```
mysql_select_db("temperatura", $con);
$query = "SELECT AVG(temperature),MAX(temperature), MIN(temperature) FROM temperature";
```

4.4 MENSAJES DE ALERTA

Se deseaba implementar un sistema de alerta, que pudiera ser configurado por el usuario para avisar a este de cuando alguno de los sensores ha recogido algún valor anormal o peligroso para el entorno estudiado.

Se deicidio usar el correo electrónico, al ser un servicio muy extendido entre la mayor parte del público. De esta manera el usuario podría recibir notificaciones en su teléfono consultándolas rápidamente.

Para ello se instaló un programa llamado sSMTP (simpe SMTP). Se trata de un programa sencillo que solo permite enviar mensajes a un correo.

El uso de este servicio es rápido, tan solo se necesita escribir la cuenta del correo y el mensaje que se desea enviar.

```
fromaddr = 'olmomoya@gmail.com'
toaddrs = 'olmomoya@gmail.com'

msg1 = "\r\n".join([
  "From: olmomoya@gmail.com",
  "To: olmomoya@gmail.com",
  "Subject: Alarm",
  "",
  "La temperatura es demasiado alta"
  ])
```

Se necesita también leer el ultimo valor introducido por el usuario en la base de datos ya que este será la última restricción que el usuario desea.

```
cur.execute("SELECT temperaturamaxima FROM registro ORDER BY id DESC LIMIT 1")
temperaturamaxima = cur.fetchall()
temperaturamax = temperaturamaxima[0][0]
```

Por último, se envía el mensaje si se cumplen las condiciones preestablecidas.

```
    if temperatura > temperaturamax:
            server.sendmail(fromaddr, toaddrs, msg1)
```

4.5 SERVIDOR WEB

Para el proyecto se ha necesitado implementar un servidor web. Como se ha comentado anteriormente, se ha elegido un servidor Apache ubicado en la Raspberry-Pi. De esta manera las gráficas podrán ser consultadas de forma cómoda desde cualquier dispositivo.

Sin embargo, el router del servidor solo dispone de una ip móvil. Esto provoca que la dirección de la estación meteorológica cambie constantemente.

Por ello, se ha tenido que utilizar los servicios de una página web de internet *www.noip.com*. Esta nos permite tener un dominio que redirigirá automáticamente a la ip del router encontrado en el servidor y actualizará esta ip cuando cambie. De esta manera se puede acceder a la estación también desde un lugar fuera de la red local.

Se ha tenido también que abrir el puerto del router en el que está conectado el microcontrolador para permitir cualquier acceso desde fuera de la red local. Se ha configurado para que cualquier acceso sea redireccionado al servidor Apache.

Para mostrar la información recogida en la base de datos se decidió representarla en forma de gráfica, para que resultará sencillo consultar y analizar la información recibida. También se decidió diseñar un sencillo menú para poder seleccionar cualquiera de las gráficas disponible en la estación meteorológica.

La mayor parte de las páginas del servidor siguen el mismo esquema. Disponen normalmente de tres archivos con tres extensiones diferentes:

-Archivo HTML: Esta extensión marcara la disposición de los elementos que visualizara el usuario (menú, grafica etc…)

-Archivo PHP: Se ha utilizado PHP para establecer conexión con la base de datos y realizar las diferentes consultas con la base de datos.

-Archivo JAVASCRIPT: Para la visualización los datos se ha utilizado la ayuda de una página, *highcharts,* esta contiene numerosas graficas que pueden ser utilizadas en cualquier proyecto de modo gratuito.

A continuación, se va a describir en mayor profundidad estos archivos.

4.5.1 HTML

Para definir la estructura y la organización de una página web se utiliza habitualmente HTML (HyperText Markup Language). En HTML se utilizan sencillas estructuras semánticas para definir la ubicación de los elementos que se visualizaran.

También dispone de potentes herramientas para definir la presentación y el aspecto de los elementos y como serán vistos estos por el usuario final. Una de estas herramientas, utilizada en este proyecto, se denomina CSS (Cascade style sheets). De esta manera con HTML se decide la estructura de un documento y con CSS se le otorga un aspecto atractivo para el usuario.

La mayoría de las páginas de este proyecto dispone de un archivo con extensión HTML en el cual se estructura la visualización de los diferentes elementos. También en este documento se diseña, con CSS, un sencillo menú para permitir la navegación entre las distintas páginas de la estación.

De esta manera el archivo HTML estaría formado por dos partes. La primera decidirá el orden de la página web y la dirección de los enlaces. También se incluye aquí las referencias Javascript dentro del elemento script como se ve en la parte superior de la siguiente imagen.

```
<html>
<head>
<title>Temperatura</title>
<script src="http://ajax.googleapis.com/ajax/libs/jquery/1.9.1/jquery.min.js" type="text/javascri
<script src="http://code.highcharts.com/highcharts.js"></script>
<script src="http://code.highcharts.com/modules/exporting.js"></script>
<script type="text/javascript" src="data.js" ></script>
</head>
<body>
<nav>
<ul>
                <li><a href="http://olmoso.ddns.net/humedad">Humedad</a></li>
                <li><a href="http://olmoso.ddns.net/lum">Luminosidad</a></li>
                <li><a href="http://olmoso.ddns.net/temp">Temperatura</a></li>
                <li><a href="http://olmoso.ddns.net/presion">Presion</a></li>
                <li><a href="http://olmoso.ddns.net/statstemperatura">Estadisticas</a></li>
                <li><a href="http://olmoso.ddns.net/registro/registro.php">Registro</a></li>

</ul>
</nav>
<div id="chart" style="height: 400px; margin: 0px auto"></div>
<br/>
<br/>
<nav>
<ul>
                <li><a href="http://olmoso.ddns.net/temperaturedaystats">Hoy</a></li>
                <li><a href="http://olmoso.ddns.net/temperatureweekstats">Esta semana</a></li>
                <li><a href="http://olmoso.ddns.net/temperaturemonthstats">Mes</a></li>
                <li><a href="http://olmoso.ddns.net/temp">Todas</a></li>

</ul>
</nav>


</body>
</html>
```

Como se observa la mayor parte del código se encarga de definir las pestañas que formaran el menú. En la siguiente imagen se puede ver en detalle cómo se hace esto. La primera parte contiene la dirección a la que dirigirá el enlace. Después aparece el texto que aparecerá como pestaña.

```
<li><a href="http://olmoso.ddns.net/humedad">Humedad</a></li>
<li><a href="http://olmoso.ddns.net/lum">Luminosidad</a></li>
<li><a href="http://olmoso.ddns.net/temp">Temperatura</a></li>
<li><a href="http://olmoso.ddns.net/presion">Presion</a></li>
<li><a href="http://olmoso.ddns.net/statstemperatura">Estadisticas</a></li>
<li><a href="http://olmoso.ddns.net/registro/registro.php">Registro</a></li>
```

La segunda parte del archivo HTML, realiza el diseño de los diferentes elementos del menú. Se trata de la parte escrita en CSS. En la siguiente imagen se puede ver un fragmento del código.

```
<style type = "text/css">

nav{
        /*Bordes redondeados*/

        overflow:hidden;
        padding:10px;
        width:950px;
}
nav ul{
        list-style:none;
        margin:0 10px 0 10px;
        padding:0;
}
nav ul li{
        /*Bordes redondeados*/
        -webkit-border-radius:5px;/*Chrome y Safari*/
        -moz-border-radius:5px;/*Firefox*/
        -o-border-radius:5px;/*Opera*/
        border-radius:5px;/*Estandar por defecto*/
        float:left;
        font-family:Arial, Helvetica, sans-serif;
        font-size:16px;
        font-weight:bold;
        margin-right:10px;
        text-align:center;
        /*Sombras para texto, los mismos parametros que box-shadow*/
        text-shadow: 0px 1px 0px #FFF;
}
```

En la imagen anterior se puede ver como se modifican los aspectos visuales del menú: el tipo de letra elegido para el texto, el color del menú incluso aspectos como el radio de los diferentes botones que necesarios en el proyecto. También se pueden definir las acciones que se realizan cuando el cursor pase sobre ellos, como se ha hecho para el proyecto donde aparece un sombreado.

4.5.2 PHP

Se trata de un lenguaje de programación diseñado para ser utilizado junto a HTML. De esta manera se puede obtener una página web dinámica. Por ello con PHP (Hypertext Pre-Processor) no se necesita llamar a ningún archivo externo para procesar los datos.

En el proyecto, se ha utilizado PHP para realizar las consultas con la base de datos debido a la sencillez del manejo de esta en este lenguaje.

En el anterior apartado donde se describían las consultas realizadas con MySQL se podía observar cómo se realiza la conexión a la base de datos y las consultas necesarias en PHP.

4.5.3 JAVASCRIPT

Se trata como PHP de un lenguaje de programación para la obtención de páginas web dinámicas. Todas las gráficas de la estación meteorológica han sido implementadas con Javascript con la ayuda de Highcharts. Highcharts es una empresa que suministra gráficas interactivas de manera gratuita escritas en Javascript. Tiene numerosos modelos (Columnas, Área etc.) Para este proyecto se han utilizado las que se han creído más intuitivas para el usuario final.

El archivo Javascript contiene el diseño de la gráfica. Determina el modelo de ella, así como todas sus características: color, leyenda etc…

```
$('#chart').highcharts({
    chart : {
        type : 'spline',
        backgroundColor: 'transparent',

    },
```

En la imagen anterior se puede ver como se selecciona el tipo de grafica que se ha elegido (barras, línea, circulo etc…).

También se puede especificar la leyenda de la gráfica, las unidades de los valores etc. En la siguiente imagen se puede ver como se han configurado los diferentes títulos y subtítulos que aparecerán. Se ha programado para que también muestre el valor recogido al pasar el ratón.

```
    title : {
        text : 'Temperatura'
    },
    subtitle : {
        text : 'La Eliana'
    },
    xAxis : {
        type: 'datetime',
       title : {
            text : 'Time'
        },
        categories : x_values
    },
    yAxis : {
        title : {
            text : 'Temperatura'
        },
        labels : {
            formatter : function() {
                return this.value + 'C'
            }
        }
    },
```

Para poder mostrar la información en las gráficas se ha debido programar una función que leerá toda la información que suministra el archivo PHP como se ha comentado anteriormente, este es el archivo encargado de leer y realizar las consultas a la base de datos.

```
$(function() {

    var x_values = [];
    var y_values = [];
    var switch1 = true;
    $.get('values.php', function(data) {

        data = data.split('/');
        for (var i in data)
        {
            if (switch1 == true)
            {
                x_values.push(data [i]);
                switch1 = false;
            }
            else
            {
                y_values.push(parseFloat(data[i]));
                switch1 = true;
            }

        }
```

En la imagen anterior se puede ver como se leen los valores recogidos por los sensores. Se llama a *values.php*, donde se recogen los valores mediante una consulta en la base de datos. Hay un bucle que recoge el primer valor del eje x, cuando encuentra la primera barra ("/") modifica el valor de *switch1* para recoger el valor del eje y cambiando otra vez el valor de *switch1* al encontrar la siguiente barra. De esta manera se pueden recogen todos los valores automáticamente.

4.6 REGISTRO

Dentro de la estación meteorológica se encuentra una página diferente al resto de las otras. Se trata del registro. Esta se encarga de recoger los valores introducidos por el usuario para configurar las alertas.

Este archivo tiene partes en HTML y PHP. Pero el contenido de estas varia comparado con las otras.

```
<style>
        *{
            font-size: 14px;
            font-family: sans-serif;
        }
        form.registro{
            background: none repeat scroll 0 0 #F1F1F1;
            border: 1px solid #DDDDDD;
            margin: 0 auto;
            padding: 20px;
            width: 278px;
            box-shadow:0px 0px 20px black;
            border-radius:10px;
            position:relative;
            top:30px;
        }
        form.registro div {
            margin-bottom: 15px;
            overflow: hidden;
        }
        form.registro div label {
            display: block;
            float: left;
            line-height: 25px;
        }
        form.registro div input[type="text"], form.registro div input[type="password"] {
            border: 1px solid #DCDCDC;
            float: right;
            padding: 4px
        }
        form.registro div input[type="submit"] {
            background: none repeat scroll 0 0 #DEDEDE;
            border: 1px solid #C6C6C6;
            float: right;
            font-weight: bold;
            padding: 4px 20px;
        }
```

En el fragmento anterior se puede observar el archivo HTML. Como se ha dicho anteriormente este se encarga del aspecto visual. Se puede modificar cosas como el sombreado de los objetos, tamaño, color etc.

El código también necesita conectar con la base de datos.

```
<?php
$con = mysql_connect("localhost","root","raspberry");
if (!$con) {
die('Could not connect: ' . mysql_error());
}
mysql_select_db("temperatura", $con);
```

Se ha programado para que no se pueda enviar los valores del usuario a menos que este rellene todos los campos. Como se puede ver en la siguiente foto.

```
if(isset($_POST['enviar']))
{
   if($_POST['muestreo'] == '' or $_POST['tmaxima'] == '' or $_POST['tminima'] == '' or $_POST['hmaxima'] == '' or $_POST['hminima'] == '' or $_POST['pminima'] == '' )
   {
      echo 'Por favor llene todos los campos.';
   }
```

Por último, se deben introducir los valores deseados por el usuario en la base de datos para que puedan ser utilizados por el programa principal, que enviará los mensajes de alerta.

```
else
{
            $muestreo = $_POST['muestreo'];
            $tmaxima = $_POST['tmaxima'];
            $tminima = $_POST['tminima'];
            $hmaxima = $_POST['hmaxima'];
            $hminima = $_POST['hminima'];
            $pminima = $_POST['pminima'];
            $query = "INSERT INTO registro (temporizador, temperaturamaxima, temperaturaminima, humedadmaxima, humedadminima, presionminima) VALUES ($muestreo,
            mysql_query($query);
            echo 'Usted se ha registrado correctamente.';
}
```

En la foto anterior se muestra como se registran los valores introducidos por el usuario. Se guarda el valor introducido por el usuario en varias variables que son insertadas después en la base de datos. El código principal, encargado de enviar los mensajes de alerta, solo leerá el último valor introducido en la base de datos pues será la última restricción introducida por el usuario.

CAPITULO 5

COSTE Y PLAN DE TRABAJO

5.1 PLAN DE TRABAJO

Tipo de Tarea	Descripción	Tiempo en horas
Contextualización	Configuración inicial de la Raspberry-Pi e instalación de los módulos necesarios.	12
Contextualización	Lectura de manuales y tutoriales sobre los diferentes lenguajes de programación utilizados	40
Contextualización	Búsqueda en internet de los diferentes sensores a emplear	3
Contextualización	Creación de la base de datos	1
Electrónica	Lectura de los diferentes datasheets y montaje de los sensores	5
Electrónica	Creación del programa para leer los sensores, comunicar con la base de datos y enviar alertas	40
Programación	Primer diseño de las páginas	40

Programación	Primer diseño de las gráficas y manejo de Highcharts	20
Programación	Diseño del menú y algunos aspectos gráficos	30
Programación	Comunicación y lectura de información de la base de datos	30
Programación	Creación del resto de las páginas del servidor	30
Programación	Revisión y retoques finales	50
Documentación	Redacción de la memoria	50
TOTAL		351

5.2 COSTE DEL PROYECTO

COMPONENTE	PRECIO
LDR	0.12 €
RASPBERRY PI 2	30€
DHT11	2€
BMP180	1.89€
LM35	2€
MCP3008	5€
TOTAL	**39.01€**

TRABAJADOR	PRECIO HORA	PRECIO
Programador	15€/h	5265€

CAPÍTULO 6

CONCLUSIONES

6.1 RESUMEN

La estación meteorológica conseguida en este trabajo cumple los objetivos propuestos durante la planificación del proyecto.

Se ha conseguido realizar una estación meteorológica competitiva con aquellas que están en el mercado. El microcontrolador ha sido configurado correctamente. Este recibe la información de los sensores y la escribe en tiempo real dentro de la base de datos. A la vez lee los últimos valores introducidos por el usuario, para poder modificar los tiempos de consulta y los valores en que las alarmas enviaran un mensaje de advertencia al usuario.

El servidor Apache ha sido implementado correctamente en la tarjeta Raspberry-Pi. En el servidor está ubicada la base de datos y todo el código que conforma los diferentes apartados de la estación.

La información se muestra de forma intuitiva. Se permite un análisis completo de los valores recogidos gracias a la base de datos.

Se ha tenido que recurrir a una página web externa (www.noip.com) para poder tener un dominio fijo que permita acceder desde dispositivos que no estén conectados a la red local.

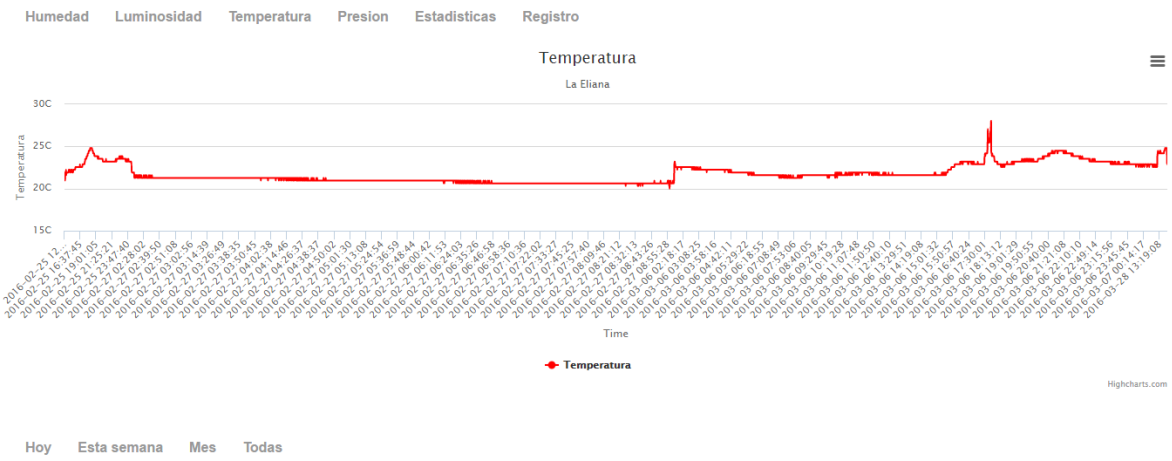
En resumen, la estación cumple con los objetivos fijados en el inicio. Ha mantenido un bajo coste en comparación con las estaciones meteorológicas disponibles en el mercado. Se puede acceder con cualquier dispositivo que disponga de conexión a internet independientemente de la red que se utilice.

La interfaz resulta funcional, dispone de un sencillo menú para poder navegar por el servidor.

Por último, el usuario puede configurar fácilmente el sistema de alertas, así como, el periodo en el que desea que la estación realice las medidas.

6.2 MANUAL DE USO

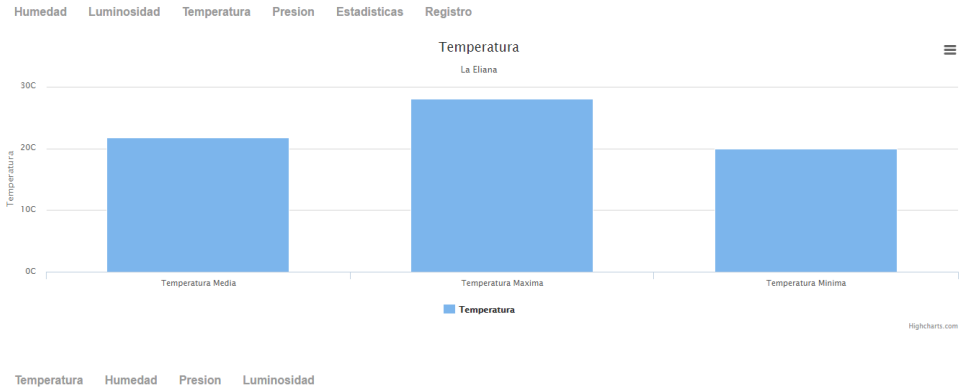
El diseño final de la estación se puede observar en la siguiente fotografía, cambiando el color de la gráfica según la variable observada.



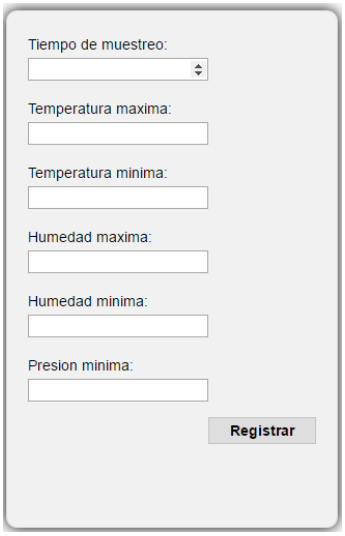
En la parte inferior se encuentra un menú, donde se puede cambiar los datos visualizados según el periodo de tiempo.

Para poder navegar entre las diferentes secciones del servidor se dispone de un menú para las diferentes variables.

Como se puede observar, hay dos pestañas con nombres diferentes: Estadísticas y Registro. La primera de ellas es la encargada de mostrar los valores mínimos, medios y máximos.



Por último, estaría el registro donde el usuario puede introducir los valores deseados para las alertas y el tiempo de muestreo. Bastaría con introducir las variables en el recuadro correspondiente y estas se guardarán de manera inmediata en la base de datos.



6.3 POSIBLES MEJORAS

Conforme se realizaba el proyecto se han observado algunas mejoras que podrían incluirse en la estación meteorológica. Estas serán descritas en los siguientes puntos.

6.3.1 SENSORES

Existen sensores de precio superior con los que mejorar la estación meteorológica. Existen varios modelos que disponen de una resolución superior (BMP280, DHT22 etc…). Hay una

gama muy amplia donde poder elegir dependiendo de los parámetros y la exactitud con la que se desea medir.

Por otra parte, se puede comprar sensores para recoger otro tipo de valores. Se valoró añadir sensores de gases y un anemómetro, pero su precio era superior al presupuesto disponible.

Por último, uno de los problemas de la estación meteorológica es la imposibilidad de estar al exterior debido a que los sensores disponibles no pueden resistir el agua. Para ello existen sensores resistentes al agua por un precio superior como el TE224, por ejemplo, para temperatura.

6.3.2 OTRAS FUNCIONALIDADES

También existen otras mejoras para ampliar las posibles funcionalidades del proyecto.

En primer lugar, el microcontrolador no puede alejarse de ninguna toma de red ya que solo dispone de conexión mediante cable de red. Para ello existen módulos que le otorgaran conexión wifi a la Raspberry-Pi, por ejemplo, el Belkin F7D1102az.

Por último, al medir valores como la temperatura, humedad etc. Se podría considerar la instalación de actuadores para regular los cambios bruscos de temperatura, luminosidad etc… obteniendo de esta forma un completo sistema domótico.

ANEXO

FRAGMENTO DEL CÓDIGO DE LA ESTACIÓN METEOROLÓGICA

PROGRAMA PRINCIPAL ESTACIÓN.PY

```
import smtplib
import time
import os
import RPi.GPIO as GPIO
import Adafruit_BMP.BMP085 as BMP085
import glob
import Adafruit_DHT
import MySQLdb as mdb
GPIO.setmode(GPIO.BCM)
import spidev
fromaddr = 'olmomoya@gmail.com'
toaddrs = 'olmomoya@gmail.com'

msg1 = "\r\n".join([
  "From: olmomoya@gmail.com",
  "To: olmomoya@gmail.com",
  "Subject: Alarm",
  "",
  "La temperatura es demasiado alta"
  ])

msg2 = "\r\n".join([
  "From: olmomoya@gmail.com",
  "To: olmomoya@gmail.com",
  "Subject: Alarm",
  "",
  "La humedad es demasiado alta"
  ])

msg3 = "\r\n".join([
  "From: olmomoya@gmail.com",
  "To: olmomoya@gmail.com",
  "Subject: Alarm",
  "",
  "La presion es demasiado baja"
  ""
  ])

msg4 = "\r\n".join([
  "From: olmomoya@gmail.com",
  "To: olmomoya@gmail.com",
  "Subject: Alarm",
  "",
  "La temperatura es demasiado baja"
  ""
  ])
msg5 = "\r\n".join([
  "From: olmomoya@gmail.com",
  "To: olmomoya@gmail.com",
  "Subject: Alarm",
  "",
  "La humedad es demasiado baja"
  ""
  ])



username = 'olmomoya@gmail.com'
password = '***********'
```

```
server = smtplib.SMTP('smtp.gmail.com:587')
server.ehlo()
server.starttls()
server.login(username,password)
sensor1 = BMP085.BMP085()
sensor = Adafruit_DHT.DHT11
pin_DHT = 4
spi = spidev.SpiDev()
spi.open(0,0)
def getADC (channel):
 r = spi.xfer2([1, (8+channel) << 4,0])
 adcOut = ((r[1] & 3) << 8) + r[2]
 return adcOut
while True:
pressure = sensor1.read_sealevel_pressure()
altitude = sensor1.read_altitude()
altitude = altitude * -1
pressure = pressure * 0.0075006
humidity, medicion = Adafruit_DHT.read_retry(sensor, pin_DHT)
value=getADC(0)
value1=getADC(1)
volts = (value * 3.3) / 1024
volts1 = (value1 * 3.3) / 1024
volts1 = volts1 - 1.6
lumin= volts1 / 1.7
lumin = 1 - lumin
lumin = lumin * 100
if lumin > 100:
lumin = 100
print 'luminosidad =' ,lumin
temperatura = volts / (10.0/1000)
print 'temperatura =' ,temperatura
print 'presion =' ,pressure
print 'altitud =' ,altitude
if humidity is not None:
print 'Humedad =' , humidity
pi_temp = temperatura
pi_hum = humidity
con = mdb.connect('localhost', \
                  'pi_insert', \
                  'raspberry', \
                 'temperatura');
cur = con.cursor()
cur.execute("""INSERT INTO humedad(humedad) \
                       VALUES(%s)""", (pi_hum))
cur.execute("""INSERT INTO temperature(temperature) \
                       VALUES(%s)""", (pi_temp))
cur.execute("""INSERT INTO luminosidad(luminosidad) \
                       VALUES(%s)""", (lumin))
cur.execute("""INSERT INTO presion(presion) \
                      # VALUES(%s)""", (pressure))
   cur.execute("SELECT temporizador FROM registro ORDER BY id DESC LIMIT
1")
muestreo = cur.fetchall()
tiempo = muestreo[0][0]
print tiempo
    cur.execute("SELECT temperaturamaxima FROM registro ORDER BY id DESC
LIMIT 1")
temperaturamaxima = cur.fetchall()
temperaturamax = temperaturamaxima[0][0]
print temperaturamax
```

```
    cur.execute("SELECT temperaturaminima FROM registro ORDER BY id DESC
LIMIT 1")
temperaturaminima = cur.fetchall()
temperaturamin = temperaturaminima[0][0]
print temperaturamin
  cur.execute("SELECT humedadmaxima FROM registro ORDER BY id DESC LIMIT
1")
humedadmaxima = cur.fetchall()
humedadmax = humedadmaxima[0][0]
print humedadmax
    cur.execute("SELECT temperaturaminima FROM registro ORDER BY id DESC
LIMIT 1")
humedadminima = cur.fetchall()
humedadmin = humedadminima[0][0]
print humedadmin
  cur.execute("SELECT presionminima FROM registro ORDER BY id DESC LIMIT
1")
presionminima = cur.fetchall()
presionmin = presionminima[0][0]
print presionmin
if temperatura > temperaturamax:
server.sendmail(fromaddr, toaddrs, msg1)
if humidity > humedadmax:
server.sendmail(fromaddr, toaddrs, msg2)
if pressure < presionmin:
server.sendmail(fromaddr, toaddrs, msg3)
if temperatura < temperaturamin:
server.sendmail(fromaddr, toaddrs, msg4)
if humidity < humedadmin:
server.sendmail(fromaddr, toaddrs, msg5)
con.commit()



time.sleep(tiempo)
```

TEMPERATURA DATA.JS

```
$(function() {

    var x_values = [];
    var y_values = [];
    var switch1 = true;
    $.get('values.php', function(data) {

        data = data.split('/');
        for (var i in data)
        {
            if (switch1 == true)
            {
                x_values.push(data [i]);
                switch1 = false;
            }
            else
            {
                y_values.push(parseFloat(data[i]));
                switch1 = true;
            }

        }


        $('#chart').highcharts({
            chart : {
                type : 'spline',
                backgroundColor: 'transparent',


            },
            title : {
                text : 'Temperatura'
            },
            subtitle : {
                text : 'La Eliana'
            },
            xAxis : {
                 type: 'datetime',
               title : {
                    text : 'Time'
                },
                categories : x_values
            },
            yAxis : {
                title : {
                    text : 'Temperatura'
                },
                labels : {
                    formatter : function() {
                        return this.value + 'C'
                    }
                }
            },
            tooltip : {
                crosshairs : true,
                shared : true,
```

```
                valueSuffix : ''
            },
            plotOptions : {


                spline : {
                    color: 'red',
                    marker : {
                        radius : 4,
                        lineColor : 'red',
                        lineWidth : 1
                    }
                }
            },
            series : [{

                name :'Temperatura',
                data : y_values
            }]
        });
    });
});
```

TEMPERATURA INDEX.HTML

```
<html>
<head>
<title>Temperatura</title>
<script
src="http://ajax.googleapis.com/ajax/libs/jquery/1.9.1/jquery.min.js"
type="text/javascript"></script>
<script src="http://code.highcharts.com/highcharts.js"></script>
<script src="http://code.highcharts.com/modules/exporting.js"></script>
<script type="text/javascript" src="data.js" ></script>
</head>
<body>
<nav>
<ul>
          <li><a href="http://olmoso.ddns.net/humedad">Humedad</a></li>
          <li><a href="http://olmoso.ddns.net/lum">Luminosidad</a></li>
          <li><a href="http://olmoso.ddns.net/temp">Temperatura</a></li>
          <li><a href="http://olmoso.ddns.net/presion">Presion</a></li>
                                                                  <li><a
href="http://olmoso.ddns.net/statstemperatura">Estadisticas</a></li>

<li><ahref="http://olmoso.ddns.net/registro/registro.php">Registro</a></
li>

</ul>
</nav>
<div id="chart" style="height: 400px; margin: 0px auto"></div>
<br/>
<br/>
<nav>
<ul>
                <li><a
href="http://olmoso.ddns.net/temperaturedaystats">Hoy</a></li>
                <li><a
href="http://olmoso.ddns.net/temperatureweekstats">Esta semana</a></li>
                <li><a
href="http://olmoso.ddns.net/temperaturemonthstats">Mes</a></li>
                <li><a href="http://olmoso.ddns.net/temp">Todas</a></li>

</ul>
</nav>


</body>
</html>


<style type = "text/css">

nav{


overflow:hidden;
padding:10px;
width:950px;
}
nav ul{
list-style:none;
margin:0 10px 0 10px;
padding:0;
```

```
}
nav ul li{

border-radius:5px;/*Estandar por defecto*/
float:left;
font-family:Arial, Helvetica, sans-serif;
font-size:16px;
font-weight:bold;
margin-right:10px;
text-align:center;
text-shadow: 0px 1px 0px #FFF;
}
nav ul li:hover{
background-image:  -webkit-gradient(linear,  left  top,  left  bottom,
from(#FFF),
to( #E3E3E3));
background-image:      -moz-linear-gradient(top      center,      #FFF,
#E3E3E3);/*Firefox*/
background-image: -o-linear-gradient(top, #FFF, #E3E3E3);
background-image: linear-gradient(top, #FFF, #E3E3E3);
-webkit-box-shadow:  1px -1px 0px #999;
-moz-box-shadow:  1px -1px 0px #999;/*Firefox*/
-o-box-shadow:  1px -1px 0px #999;/*Opera*/
box-shadow:  1px -1px 0px #999;/*Estandar por defecto*/
border:1px solid #E3E3E3;
}
nav ul li a{
color:#999;
display:block;
padding:10px;
text-decoration:none;
/*Transiciones: Tiempo, Efecto y Propiedades aplicadas
-webkit-transition: 0.4s linear all;
 -moz-transition: 0.4s linear all;
 -o-transition: 0.4s linear all;
 transition: 0.4s linear all;
}
nav ul li a:hover {
color:#000;
}
```

TEMPERATURA VALUES.PHP

```php
<?php
$con = mysql_connect("localhost", "root", "raspberry");
if(!$con){
die('Could not connect: ' .mysql_error());
}
mysql_select_db("temperatura", $con);
$result = mysql_query("SELECT * FROM `temperature`") or die ("Connection
error");
while($row = mysql_fetch_array($result)) {
echo $row['dtg'] . "/" . $row['temperature']. "/" ;
}
mysql_close($con);
?>
```

TEMPERATURA DIA VALUES.PHP

```
<?php
$con = mysql_connect("localhost", "root", "raspberry");
if(!$con){
die('Could not connect: ' .mysql_error());
}
mysql_select_db("temperatura", $con);
$diaactual = date("d");
$mesactual = date("m");
$anoactual = date("y");
$result = mysql_query("SELECT * FROM `temperature` WHERE day(dtg) =
'$diaactual' AND month(dtg) = '$mesactual'") or die ("Connection error");
while($row = mysql_fetch_array($result)) {
echo $row['dtg'] . "/" . $row['temperature']. "/" ;
}
mysql_close($con);
?>
```

TEMPERATURA SEMANA VALUES.PHP

```
<?php
$con = mysql_connect("localhost", "root", "raspberry");
if(!$con){
die('Could not connect: ' .mysql_error());
}

mysql_select_db("temperatura", $con);

$diaactual = date("d");
$mesactual = date("m");
$anoactual = date("y");
$semanaactual = date("W");
$result = mysql_query("SELECT * FROM `temperature`    WHERE week(dtg) =
'$semanaactual'") or die ("Connection error");
while($row = mysql_fetch_array($result)) {
echo $row['dtg'] . "/" . $row['temperature']. "/" ;
}
mysql_close($con);
?>
```

TEMPERATURA MES VALUES.PHP

```
<?php
$con = mysql_connect("localhost", "root", "raspberry");
if(!$con){
die('Could not connect: ' .mysql_error());
}

mysql_select_db("temperatura", $con);
$diaactual = date("d");
$mesactual = date("m");
$anoactual = date("y");
$result = mysql_query("SELECT * FROM `temperature` WHERE month(dtg) =
'$mesactual'") or die ("Connection error");
while($row = mysql_fetch_array($result)) {
echo $row['dtg'] . "/" . $row['temperature']. "/" ;
}
mysql_close($con);
?>
```

HUMEDAD DATA.JS

```
$(function() {

    var x_values = [];
    var y_values = [];
    var switch1 = true;
    $.get('values.php', function(data) {

        data = data.split('/');
        for (var i in data)
        {
            if (switch1 == true)
            {

               x_values.push(data [i]);
                switch1 = false;
            }
            else
            {
                y_values.push(parseFloat(data[i]));
                switch1 = true;
            }

        }
        x_values.pop();

        $('#chart').highcharts({
            chart : {
                type : 'spline'
            },
            title : {
                text : 'Humedad'


            },
            subtitle : {
                text : 'La Eliana'
            },
            xAxis : {
                title : {
                    text : 'Time'
                },
                categories : x_values
            },
            yAxis : {
                title : {
                    text : 'Humedad'
                },
                labels : {
                    formatter : function() {
                        return this.value + '%'
                    }
                }
            },
            tooltip : {
                crosshairs : true,
                shared : true,
                valueSuffix : ''
            },
            plotOptions : {
```

```
                spline : {
                    marker : {
                        radius : 4,
                        lineColor : '#666666',
                        lineWidth : 1
                    }
                }
            },
            series : [{

                name : 'Humedad',
                data : y_values
            }]
        });
    });
});
```

LUMINOSIDAD DATA.JS

```
$(function() {

    var x_values = [];
    var y_values = [];
    var switch1 = true;
    $.get('values.php', function(data) {

        data = data.split('/');
        for (var i in data)
        {
            if (switch1 == true)
            {

               x_values.push(data [i]);
                switch1 = false;
            }
            else
            {
                y_values.push(parseFloat(data[i]));
                switch1 = true;
            }

        }


        $('#chart').highcharts({
            chart : {
                type : 'spline',
                backgroundColor: 'transparent',
            },
            title : {
                text : 'Luminosidad'
            },
            subtitle : {
                text : 'La Eliana'
            },

            xAxis : {
                type: 'datetime',
                title : {
                    text : 'Time'
                },
                categories : x_values
            },
            yAxis : {
                title : {
                    text : 'Luminosidad'
                },
                labels : {
                    formatter : function() {
                        return this.value + '%'
                    }
                }
            },
            tooltip : {
                crosshairs : true,
                shared : true,
                valueSuffix : ''
```

```
            },
            plotOptions : {
                spline : {
                    marker : {
                color: 'yellow',
                            radius : 4,
                            lineColor : 'yellow',
                            lineWidth : 1
                        }
                }
            },


             series : [{

                name : 'Luminosidad',
                data : y_values
            }]
        });
    });
});
```

PRESIÓN DATA.JS

```
$(function() {

    var x_values = [];
    var y_values = [];
    var switch1 = true;
    $.get('values.php', function(data) {

        data = data.split('/');
        for (var i in data)
        {
            if (switch1 == true)
            {
                x_values.push(data [i]);
                switch1 = false;
            }
            else
            {
                y_values.push(parseFloat(data[i]));
                switch1 = true;
            }

        }


        $('#chart').highcharts({
            chart : {
                type : 'spline',
                backgroundColor: 'transparent',


            },
            title : {
                text : 'Presion Atmosferica'
            },
            subtitle : {
                text : 'La Eliana'
            },
            xAxis : {
               type: 'datetime',
               title : {
                    text : 'Time'
                },
                categories : x_values
            },
            yAxis : {
                title : {
                    text : 'Presion'
                },
                labels : {
                    formatter : function() {
                        return this.value + 'mmHg'
                    }
                }
            },
            tooltip : {
                crosshairs : true,
                shared : true,
                valueSuffix : ''
            },
```

```
            plotOptions : {
                color: 'gray',
                spline : {
                    marker : {


                        radius : 4,
                        lineColor : 'gray',
                        lineWidth : 1
                    }
                }
            },
            series : [{

                name :'Presion',
                data : y_values
            }]
        });
    });
});
```

TEMPERATURA ESTADÍSTICAS DATA.JS

```
$(function() {

    var y_values = [];

    $.get('values.php', function(data) {

        data = data.split('/');
        for (var i in data)
        {

                y_values.push(parseFloat(data[i]));
        }


        $('#chart').highcharts({
            chart : {
                type : 'column',
                backgroundColor: 'transparent',


            },
            title : {
                text : 'Temperatura'
            },
            subtitle : {
                text : 'La Eliana'
            },
            xAxis : {
                categories: [
                'Temperatura Media',
                'Temperatura Maxima',
                'Temperatura Minima'
            ],
                crosshair: false,


            },
            yAxis : {
                title : {
                    text : 'Temperatura'
                },
                labels : {
                    formatter : function() {
                        return this.value + 'C'
                    }
                }
            },
            tooltip : {
            pointFormat:'<tr><td
style="color:{series.color};padding:0">{series.name}: </td>' +
                '<tdstyle="padding:0"><b>{point.y:.1f} C</b></td></tr>',
            footerFormat: '</table>',
            shared: true,
            useHTML: true
            },
            plotOptions : {
                column : {
```

```
            pointPadding: 0



                }
            },
            series : [{

                name :'Temperatura',
                data : y_values
            }]
        });
    });
});
```

ESTADÍSTICAS TEMPERATURA INDEX.HTML

```
<html>
<head>
<title>Estadisticas</title>
<script
src="http://ajax.googleapis.com/ajax/libs/jquery/1.9.1/jquery.min.js"
type="text/javascript"></script>
<script src="http://code.highcharts.com/highcharts.js"></script>
<script src="http://code.highcharts.com/modules/exporting.js"></script>
<script type="text/javascript" src="data.js" ></script>
</head>
<body>
<nav>
<ul>
<li><a href="http://olmoso.ddns.net/humedad">Humedad</a></li>
<li><a href="http://olmoso.ddns.net/lum">Luminosidad</a></li>
<li><a href="http://olmoso.ddns.net/temp">Temperatura</a></li>
<li><a href="http://olmoso.ddns.net/presion">Presion</a></li>
                                                                    <li><a
href="http://olmoso.ddns.net/statstemperatura">Estadisticas</a></li>
                <li><a
href="http://olmoso.ddns.net/registro/registro.php">Registro</a></li>
</ul>
</nav>
<div id="chart" style="height: 400px; margin: 0px auto"></div>
<br/>
<br/>
<nav>
<ul>
<li><ahref="http://olmoso.ddns.net/statstemperatura">Temperatura</a></li
>
<li><a href="http://olmoso.ddns.net/statshumedad">Humedad</a></li>
<li><a href="http://olmoso.ddns.net/stats">Presion</a></li>
<li><a href="http://olmoso.ddns.net/statslum">Luminosidad</a></li>
</ul>
</nav>
</body>
</html>


<style type = "text/css">

nav{


overflow:hidden;
padding:10px;
width:950px;
}
nav ul{
list-style:none;
margin:0 10px 0 10px;
padding:0;
}
nav ul li{

-webkit-border-radius:5px;
-moz-border-radius:5px;
```

```
-o-border-radius:5px;
border-radius:5px;
float:left;
font-family:Arial, Helvetica, sans-serif;
font-size:16px;
font-weight:bold;
margin-right:10px;
text-align:center;
text-shadow: 0px 1px 0px #FFF;
}
nav ul li:hover{

background-image:  -webkit-gradient(linear,  left  top,  left  bottom,
from(#FFF),
to( #E3E3E3));
background-image: -moz-linear-gradient(top center, #FFF, #E3E3E3);
background-image: -o-linear-gradient(top, #FFF, #E3E3E3);
background-image: linear-gradient(top, #FFF, #E3E3E3);
-webkit-box-shadow:  1px -1px 0px #999;
-moz-box-shadow:  1px -1px 0px #999;
-o-box-shadow:  1px -1px 0px #999;
box-shadow:  1px -1px 0px #999;
border:1px solid #E3E3E3;
}
nav ul li a{
color:#999;
display:block;
padding:10px;
text-decoration:none;
-webkit-transition: 0.4s linear all;
-moz-transition: 0.4s linear all;
-o-transition: 0.4s linear all;
transition: 0.4s linear all;
}
nav ul li a:hover {
color:#000;
}
```

TEMPERATURA ESTADÍSTICAS VALUES.PHP

```
<?php
$con = mysql_connect("localhost", "root", "raspberry");
if(!$con){
die('Could not connect: ' .mysql_error());
}
mysql_select_db("temperatura", $con);
$query = "SELECT AVG(temperature),MAX(temperature), MIN(temperature) FROM
temperature";
$result = mysql_query("$query") or die ("Connection error");
while($row = mysql_fetch_array($result)) {
echo $row['AVG(temperature)'] . "/" . $row['MAX(temperature)'] . "/" .
$row['MIN(temperature)']. " a" ;
}
mysql_close($con);
?>
```

REGISTRO REGISTRO.PHP

```
<style>
        *{
            font-size: 14px;
            font-family: sans-serif;
        }
        form.registro{
            background: none repeat scroll 0 0 #F1F1F1;
            border: 1px solid #DDDDDD;
            margin: 0 auto;
            padding: 20px;
            width: 278px;
            box-shadow:0px 0px 20px black;
            border-radius:10px;
            position:relative;
            top:30px;
        }
        form.registro div {
            margin-bottom: 15px;
            overflow: hidden;
        }
        form.registro div label {
            display: block;
            float: left;
            line-height: 25px;
        }
        form.registro   div   input[type="text"],   form.registro   div
input[type="password"] {
            border: 1px solid #DCDCDC;
            float: right;
            padding: 4px
        }
        form.registro div input[type="submit"] {
            background: none repeat scroll 0 0 #DEDEDE;
            border: 1px solid #C6C6C6;
            float: right;
            font-weight: bold;
            padding: 4px 20px;
        }
        .error{
            color: red;
            font-weight: bold;
            margin: 10px;
            text-align: center;
        }
</style>

<form action="" method="post" class="registro">
<div><label>Tiempo de muestreo: </label>
<input type="number" name="muestreo"></div>
<div><label>Temperatura maxima: </label>
<input type="number" name="tmaxima"></div>
<div><label>Temperatura minima:  </label>
<input type="number" name="tminima"></div>
<div>
<div><label>Humedad maxima:  </label>
<input type="number" name="hmaxima"></div>
<div>
<div><label>Humedad minima:  </label>
<input type="number" name="hminima"></div>
```

```
<div>
<div><label>Presion minima: <br></label>
<input type="number" name="pminima"></div>
<div>
<input type="submit" name="enviar"  value="Registrar"></div>

</form>

<?php
$con = mysql_connect("localhost","root","raspberry");
if (!$con) {
die('Could not connect: ' . mysql_error());
}
mysql_select_db("temperatura", $con);

if(isset($_POST['enviar']))
{
    if($_POST['muestreo']  ==  ''  or  $_POST['tmaxima']  ==  ''  or
$_POST['tminima'] == '' or $_POST['hmaxima'] == '' or $_POST['hminima']
== '' or $_POST['pminima'] == '' )
    {
        echo 'Por favor llene todos los campos.';
    }
    else
   {
       $muestreo = $_POST['muestreo'];
       $tmaxima = $_POST['tmaxima'];
       $tminima = $_POST['tminima'];
       $hmaxima = $_POST['hmaxima'];
       $hminima = $_POST['hminima'];
       $pminima = $_POST['pminima'];
        $query = "INSERT INTO registro (temporizador, temperaturamaxima,
       Temperaturaminima,
       humedadmaxima,humedadminima,presionminima)  VALUES  ($muestreo,
$tmaxima, $tminima, $hmaxima,  $hminima, $pminima)";
mysql_query($query);
echo 'Usted se ha registrado correctamente.';
   }
}
?>
```

BIBLIOGRAFÍA

-http://dplinux.net/guia-raspberry-pi

-http://www.frambuesapi.co/

-http://www.instructables.com/id/Wiring-up-a-MCP3008-ADC-to-a-Raspberry-Pi-model-B-/?ALLSTEPS

-http://jeremyblythe.blogspot.com.es/2012/09/raspberry-pi-hardware-spi-analog-inputs.html

-https://www.raspberrypi.org

-https://leanpub.com/RPiMRE/read#leanpub-auto-measure

-https://blueflame-software.com/using-highcharts-with-php-and-mysql

-http://www.w3schools.com/sql/sql_like.asp

-https://rpi.tnet.com/project/faqs/smtp

-http://www.tizag.com/mysqlTutorial/mysqlavg.php

-https://learn.adafruit.com/using-the-bmp085-with-raspberry-pi/overview