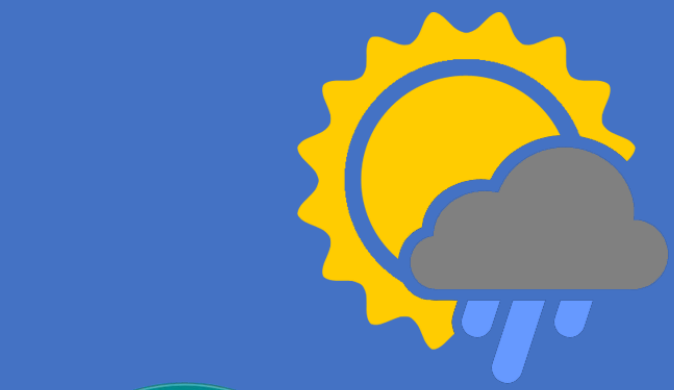
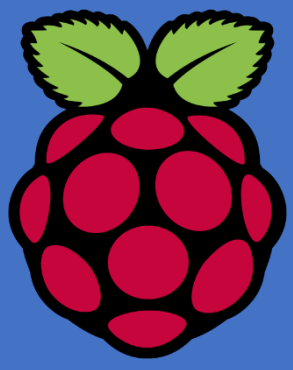
ESTACIÓN METEOROLÓGICA CON ARDUINO Y RASPBERRY PI





Francisco Miguel Jiménez Fernández

IES MAR DE CÁDIZ ASIR

Índice

Introducción

En este proyecto voy a crear una estación meteorológica con Arduino y varios sensores, como de presión, temperatura, humedad, sensor de lluvia, y pasaré los datos a Raspberry Pi mediante red.

Los datos recogidos los introduciré en una base de datos en MySql y los visualizaré con PHP y HTML y CSS, con el framework Bootstrap.

Será accesible desde cualquier parte del mundo ya que se accederá al servidor a través de mi ip externa mediante una cuenta con el servicio NO-IP.

Estos datos los usaré para crear un histórico sobre la meteorología de la zona, para visualizar las condiciones a tiempo real e incluso para hacer alguna predicción meteorológica con los datos obtenidos.

También crearé un tutorial con las nociones básicas y la sintaxis necesaria para entender el lenguaje de programación de Arduino, C++ con algunas modificaciones.

Motivación

Mi motivación principal para este proyecto han sido las ganas de poder elaborar nuevos aparatos, antes inexistentes gracias a mis conocimientos en programación web y Arduino.

Siempre me ha gustado el saber "cómo funcionan" las cosas por dentro, y gracias a la elaboración de este proyecto, he podido aprender bastante.

También, gracias al haber cursado este ciclo formativo, he visto que me encanta el mundo de la programación web, por lo que he querido hacer algo más serio en este ámbito cómo ha sido este proyecto.

El elaborar algo así conlleva un largo proceso, desde documentarse y aprender el lenguaje y el entorno en los que se va a programar, hasta el diseño web pasando por el diseño las bases de datos, diseño 3D de la estructura y la programación de la placa Arduino.

Tutorial de Arduino

Para entender el lenguaje de programación que usaré, voy a hacer una pequeña guía con lo esencial para entender el lenguaje. Mostraré cómo funcionan los bucles, arrays y la sintaxis en general.

¿Qué es Arduino?

Arduino es una plataforma de prototipos electrónica de código abierto (open-source) basada en hardware y software flexible y fácil de usar. Esto quiere decir que podemos hacer cualquier cosa para adaptarlo a nuestras necesidades. Existen multitud de módulos y sensores en el mercado muy asequibles y con gran utilidad. Las placas de Arduino también son bastante baratas.

¿Cómo es el lenguaje?

El lenguaje de programación de Arduino es C++ con algunas modificaciones. Estas modificaciones son librerías que el propio IDE de Arduino ya trae integradas.

La estructura de los programas Arduino es la siguiente:

Declaración de variables

Void setup()

{

}

Void loop()

{

}

Setup()

En la función setup() se inicializan y se preparan las variables y todos los valores que vamos a usar durante la ejecución del programa.

Lo interesante es que esta función sólo se ejecuta una única vez al iniciar el programa o nuestra placa de Arduino.

Loop()

En la función loop() situamos todo el conjunto de instrucciones que nuestro Arduino se encargará de ejecutar en bucle una y otra vez.

El programa responderá según los datos obtenidos en cada una de las ejecuciones de loop().

Aquí pongo un ejemplo de cómo sería el “Hola Mundo” de Arduino. Cabe decir que, en esta plataforma, por costumbre, el “Hola Mundo” se suele hacer haciendo parpadear un led.

```
void setup()                    // Se ejecuta cada vez que Arduino se inicia
{
pinMode(13,OUTPUT);             // Arranca pin 13
}
//Led en pin 13
void loop()                     // Esta función se mantiene ejecutando
{
digitalWrite(13,HIGH);          // Enciende el LED
delay(1000);                    // Espera un segundo (1s = 1000ms)
digitalWrite(13,LOW);           // Apaga el LED
delay(1000);                    // Espera un segundo (1s = 1000ms)
}
```

Comentarios

A la hora de hacer aclaraciones en nuestro código es esencial usar comentarios. Estas líneas son ignoradas por el compilador, por tanto no se transmiten a nuestro programa,

Arduino tiene dos formas de hacer comentarios:

- Comentarios simples: Se señalan con **//** delante y comentan todo lo que va detrás hasta el siguiente salto de línea. Sirve para hacer comentarios breves o comentar alguna línea de código.
- Comentarios multilíneas: Se abren con **/*** y se cierran con ***/**. Sirven para comentar con grandes párrafos o comentar varias líneas de código a la vez.

Un comentario simple puede ir dentro de uno multilínea, pero no al revés ni dentro de otro comentario multilínea.

Sintaxis

- Uso del **;** : Se usa detrás de cualquier declaración al igual que en otros lenguajes.
- Llaves {} : Se usan para delimitar bucles y funciones.

  Funciones

  void funcion(tipodato argumento){
  sentencias
  }

  Bucles

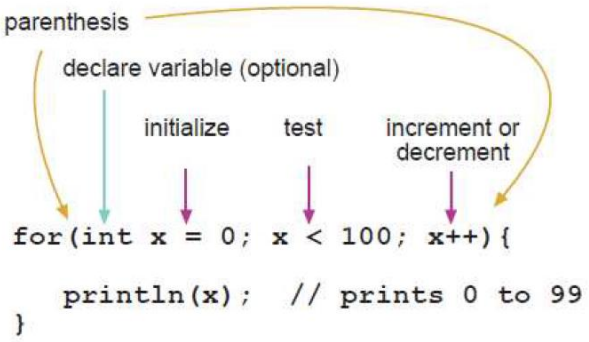
  while (expresión)
  {
  sentencias
  }

Estructuras de control

Las estructuras de control en Arduino son muy similares a las de otros lenguajes de programación.

Bucle FOR

El bucle FOR sirve para repetir un conjunto de instrucciones dentro de llaves. La sintaxis es la siguiente:



La declaración de la variable es opcional, a diferencia de otros lenguajes.

Estructura Switch Case

Esta estructura sirve para evaluar una variable y ejecutar diferente código en función del valor de la variable especificado en las sentencias case.

switch (var) {

case valor:

// sentencias

break;

case valor:

// sentencias

break;

default:

// sentencias

}

Dicha estructura tiene una sentencia “Break” que hace que salga del bucle cuando ya se ha ejecutado la sentencia que queríamos.

El caso “default” se ejecuta cuando el valor de la variable (var) no corresponde con ninguna sentencia “case valor”.

Bucle While

El Bucle While está siempre ejecutándose mientras la condición especificada se esté cumpliendo. La forma de salir este bucle puede ser una condición interna al bucle o una condición externa, cómo que un sensor dé un determinado valor.

La sintaxis es la siguiente:

```
while(expresion){

// sentencias

}
```

Bucle Do While

Este bucle funciona igual que el bucle while, solamente que siempre se va ejecutar al menos una vez, ya que evalúa la condición al final del bucle.

La sintaxis es la siguiente:

```
do{

// sentencias

} while (expresión);
```

Condicional

If compara una condición con un valor determinado y ejecuta una acción u otra, dependiendo de este valor. La sintaxis es así:

```
if (variable > valor){

// sentencias

}

else if{

// sentencias

}

else{

// sentencias

}
```

La sentencia else se ejecuta cuando no se cumple la condición especificada en if.

También tenemos las sentencias else if, que evalúan otras condiciones para poder realizar múltiples comprobaciones en una misma estructura de condiciones.

En la estructura condicional es imprescindible usar comparadores. Los comparadores del lenguaje Arduino son los siguientes:

x == y (x es igual a y)

x != y (x no es igual a y)

x < y (x es menor a y)

x > y (x es mayor a y)

x <= y (x es menor o igual a y)

x >= y (x es mayor o igual a y)

Variables y constantes

Las constantes son variables predefinidas del lenguaje de Arduino.

Hay de varios tipos: las que definen niveles lógicos, las que definen el nivel de los pines y las que definen los pines digitales

- **Valores lógicos:** Los operadores que tenemos en este grupo son **TRUE** y **FALSE**. Estos operadores para representar si algo es cierto o falso o en sentido booleano, si es no cero, o cero.
- **Nivel de los pines:** Cuando leemos o escribimos en un pin digital, existen sólo dos valores que podemos leer o asignar, LOW y HIGH. Estos significan 0 o 1.
  Un pin está en valor **HIGH** si da más de 3V en INPUT y +5V configurado en modo OUTPUT.
  Un pin está en valor **LOW** si da menos de 2V en INPUT y 0V configurado en modo OUTPUT.
- **Pines digitales:** Los pines digitales pueden configurarse para funcionar como OUTPUT (salida) o INPUT (entrada).

Funciones

Hay varias funciones predefinidas en el lenguaje de programación. Mostraré las más importantes y las más usadas en el lenguaje.

Digital

- **PinMode():** Sirve para establecer si un pin digital del Arduino será de entrada o de salida. La sintaxis es la siguiente:
  pinMode(pin, modo)
  Donde pin es el número del pin y el modo si es HIGH o LOW.
- **DigitalRead():** Seleccionamos el pin del que queremos leer información. La sintaxis es la siguiente:
  DigitalRead(pin)
  Esta función devolverá **LOW o HIGH,** dependiendo del valor que reciba por el pin.
- **DigitalWrite():** Escribe un valor **HIGH o LOW** hacia un pin digital. La sintaxis es la siguiente:
  DigitalWrite(pin, valor)
  Donde pin es el pin por el que queremos escribir y valor HIGH o LOW para escribir ese valor.

Analógico

- **AnalogRead():** Sirve para leer un valor de tensión en el pin especificado. Convertirá tensiones de 0 a 5v en valores de 0 a 1023. La sintaxis es la siguiente:
  analogRead(pin)
  Pin indica el número del pin de la entrada analógica que deseamos leer.
- **AnalogWrite():** Escribe un valor analógico en un pin. Esto sirve para poder controlar nuestros sensores y elementos con una mayor precisión. La sintaxis es la siguiente:
  analogWrite(pin, valor)
  Parámetros
  Pin indica el pin en el cual se quiere generar la señal. Valor indica el ciclo de trabajo deseado comprendido entre 0 (siempre apagado) y 255 (siempre encendido).

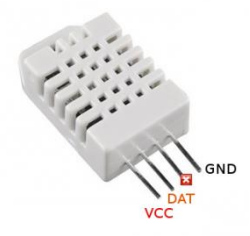
Sensores a utilizar

Ahora detallaré qué sensores voy a utilizar en mi proyecto. Estos sensores son baratos y muy versátiles. Algunos tendrán que usar sus propias librerías mientras que a otros no les hará falta. Esto depende del fabricante del sensor.

Sensor de temperatura y humedad

Para medir la temperatura y la humedad he optado por un sensor **DHT22**. Este sensor combina mediciones de temperatura y humedad relativa, con un alto grado de sensibilidad.

- Entre 0 - 100 % de humedad relativa.
- Rango de temperatura -40 a 125 ºC
- Precisión de +- 2% en humedad relativa.
- Precisión de +- 0.5 ºC
- Hasta 2 muestras por segundo (0.5 Hz)
- Usa sus propias librerías.



Sensor de presión atmosférica

Para medir la presión atmosférica, usaré el sensor barométrico BMP 180. Este sensor es capaz de medir presión atmosférica, altura, altura entre dos puntos e incluso temperatura.

- Rango de medida entre 300 y 1100 hPa (Hecto Pascal)
- Precisión de +- 0.03 hPa
- Bajo consumo
- Usa sus propias librerías

Sensor de lluvia

Para ver si llueve y cuánto lo hace, usaré un sensor de gotas de lluvia **YL-83**. Este sensor puede enviar valores digitales LOW (no llueve), HIGH (llueve) o valores analógicos para indicar cuánto llueve.

Este módulo crea un corto circuito cada vez que las pistas se mojan.

Reloj

Este módulo sirve para obtener la fecha y hora a la que se tomaron las medidas. Debido a que en la inserción en la base de datos podemos incluir la fecha y la hora a la que se insertaron los datos, no lo usaré, pero voy a detallar su funcionamiento.

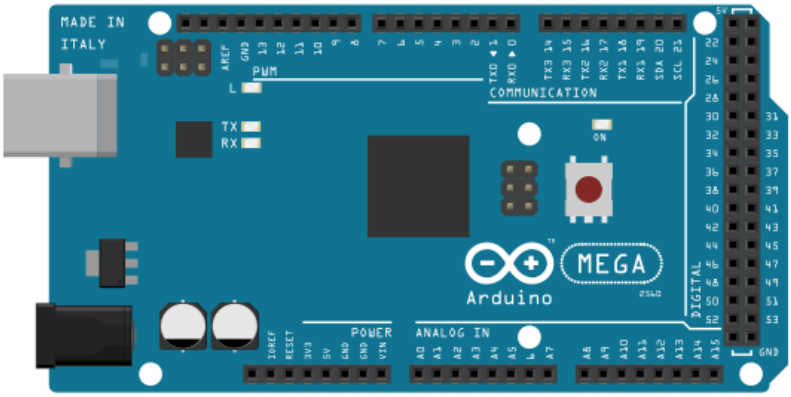
Usa una pila CR2032 para poder almacenar la hora. Incluye **su propia librería** con funciones interesantes sobre la fecha y la hora.

Hardware a utilizar

Arduino MEGA

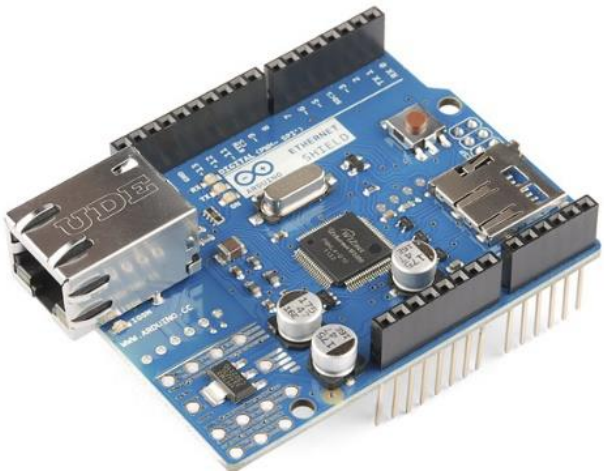
Dada su potencia y versatilidad, usaré para el proyecto una placa controladora Arduino MEGA 2560. Esta placa posee una gran cantidad de pines digitales y analógicos.

- Microcontrolador ATmega2560.
- Voltaje de entrada de – 7-12V.
- 54 pines digitales de Entrada/Salida (14 de ellos son salidas PWM).
- 16 entradas análogas.
- 256k de memoria flash.
- Velocidad del reloj de 16Mhz



Ethernet Shield

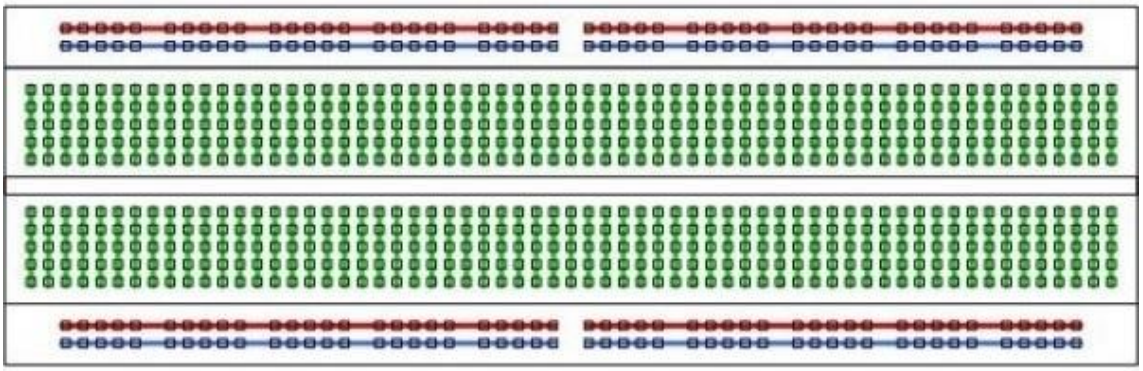
Para dotar a Arduino de conexión a internet, voy a usar el módulo Ethernet Shield. Este módulo posee sus librerías para poder configurar los parámetros de la conexión.



BreadBoard

A la hora de realizar los prototipos en Arduino, es necesario conectar y desconectar módulos, cables… Para hacer esto habría que soldar, desoldar, lo cual sería muy engorroso. Para evitar este problema usaré una BreadBoard o ProtoBoard. Consiste en una plataforma donde pinchar nuestros módulos de forma fácil haciendo que todo quede ordenado.

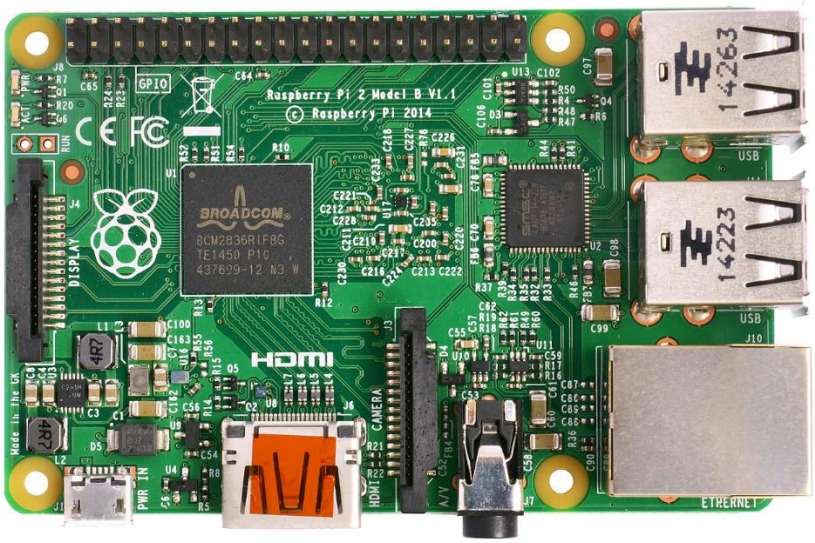
Este es el esquema de conexión. Posee 4 columnas verticales las cuales se usan para dotar de corriente a los módulos. En las filas internas haríamos las conexiones de los módulos.



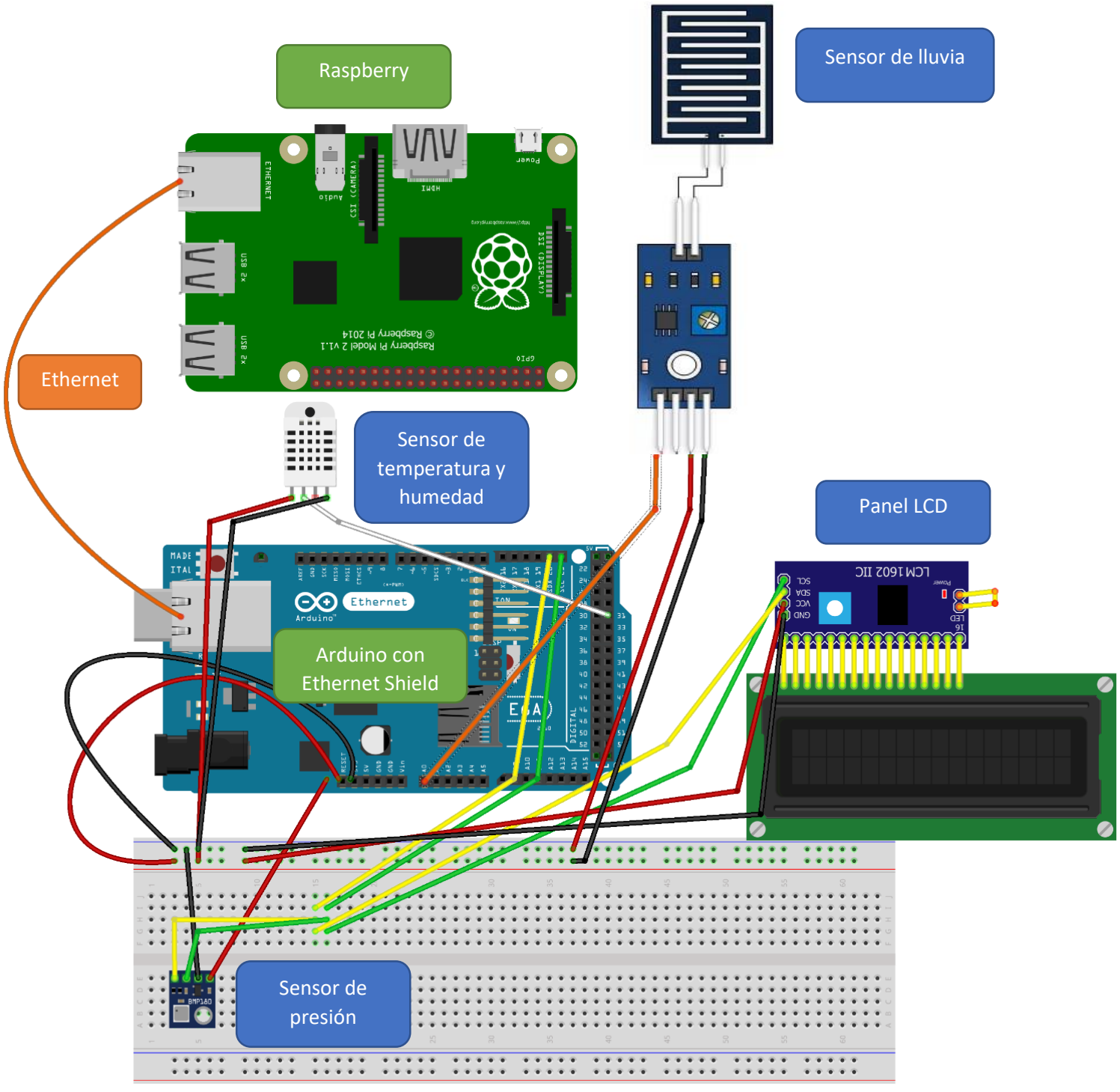
Raspberry Pi

Para recoger las medidas hechas por Arduino y almacenarlas en una base de datos, la solución más sencilla, versátil y barata es usar una Raspberry Pi. Este dispositivo debido a su bajo consumo puede estar siempre en funcionamiento recogiendo datos, lo cual es ideal para nuestro proyecto. También por otro lado, será la encargada de hacer de servidor web Apache para mostrar los datos, debido al sistema Linux basado en Debian que es capaz de ejecutar.

- GPU Broadcom VideoCore IV 250 MHz. OpenGL ES 2.0
- RAM 1 GB LPDDR2 SDRAM 450 MHz
- Salida de video HDMI
- 4 x Puertos USB 2.0
- Puerto Ethernet



Esquema de conexión



Estructura

He diseñado con el software CATIA una carcasa compuesta por dos piezas para proteger el Arduino de las inclemencias del tiempo. Esta carcasa se imprimirá con una impresora 3D.

Esta estructura soporta los sensores y se puede colocar en cualquier lugar para que empiece a transmitir datos.

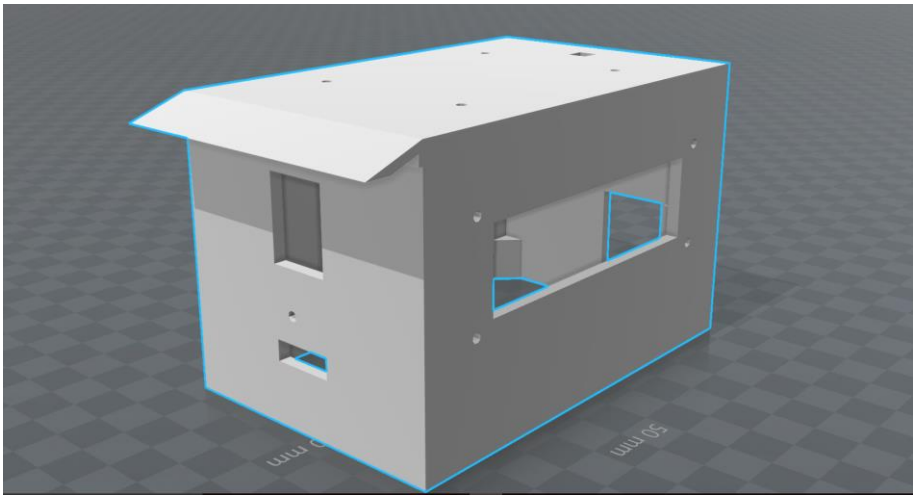
Primera pieza

Consta de la parte de arriba de la carcasa.

En la parte alta, irá el sensor de lluvia.

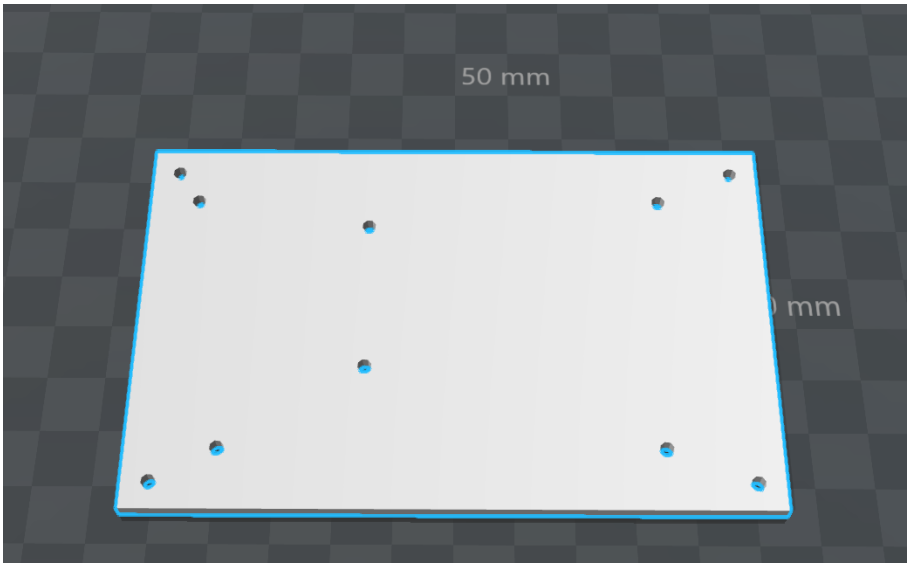
En un lateral, irá el LCD que mostrará los datos *In-Situ.*

En el frontal irá el sensor DHT22 de temperatura y humedad, y bajo este, el sensor de presión atmosférica.



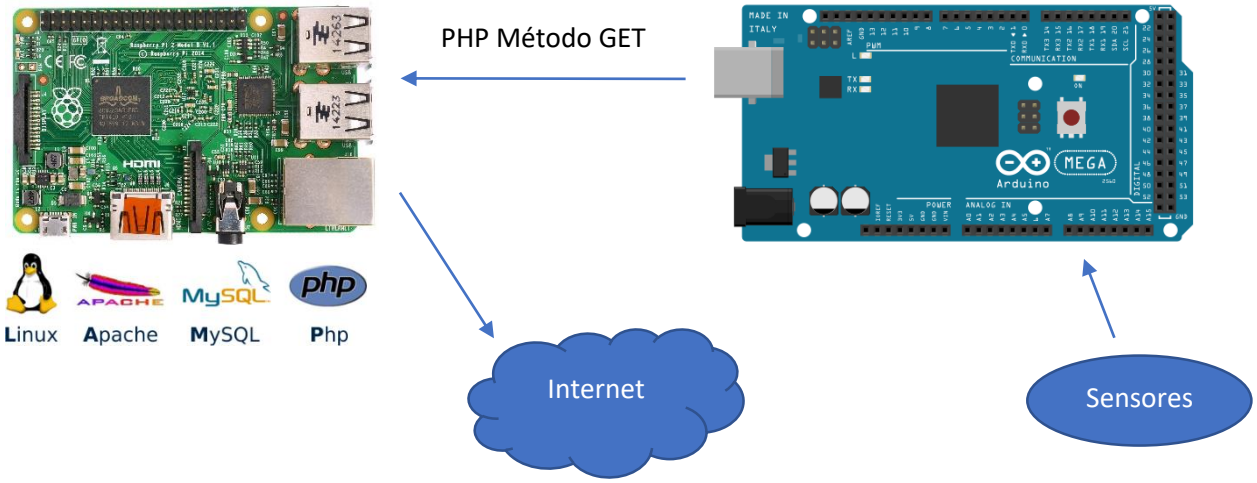
Segunda pieza

Esta pieza incluye la parte de abajo del conjunto, en la cual va atornillado el controlador Arduino. Esta pieza se une a la anterior mediante cuatro tornillos.



Conexión

La conexión requiere una red Ethernet. Los sensores están conectados a Arduino. Arduino envía los datos a un servidor **LAMP (Linux Apache MySql Php)** instalado en una Raspberry Pi para mayor versatilidad en el tratamiento de los datos.



Arduino enviará datos por método GET y se recogerán con un script en la Raspberry que los insertará en la base de datos.

Este es un fragmento de código en el que desde Arduino enviamos datos de temperatura y humedad al servidor de bases de datos. Donde "server" es la ip de nuestro servidor e insertaMedidas.php es el script que inserta los datos en la base de datos.

```
1.  if (client.connect(server, 80)>0) {  // Conexion con el
    servidor
2.        client.print("GET
    /EstacionMeteo/insertaMedidas.php?temperatura="); // Enviamos la
    temperatura por GET
3.        client.print(t);
4.        client.print("&humedad="); // Enviamos la humedad por GET
5.        client.print(h);
6.        client.print("&lluvia="); // Enviamos si llueve o no por
    GET
7.        switch (rango) {
8.            case 0:    // Sensor muy mojado
9.                    client.print("Lluvia%20fuerte");
10.           break;
11.           case 1:    // Sensor mojándose
12.                   client.print("Lluvia%20debil");
13.           break;
14.           case 2:    // Sensor seco.
15.                   client.print("Seco");
16.           break;
17.   }
18.               client.print("&presion="); //Enviamos la presión
    por GET
19.               client.print(p);
20.               client.print("&ubicación="); //Enviamos el
    idubicacion por GET (CAMBIANDO ESTE VALOR SIMULO LAS DISTINTAS
    UBICACIONES)
```

```
21.             //LO CAMBIARE CON UN BUCLE FOR PARA QUE CADA VEZ
   ENVIE DATOS A UNA UBICACION DISTINTA
22.             client.print(ubicacion);
23.             client.println(" HTTP/1.0");
24.             client.println("User-Agent: Arduino 1.0");
25.             client.println();
26.             Serial.println("Conectado");
27.           } else {
28.             Serial.println("Fallo en la conexion");
29.           }
```

Este es un ejemplo de cómo sería el script para realizar la inserción de datos en la base de datos.

```
1. <?php
2.     // insertaMedidas.php
3.     // Importamos la configuración
4.     require("comun.php");
5.     $mysqli = conecta();
6.     // Leemos la temperatura por GET
7.     $temperatura = $_GET['temperatura'];
8.     // Leemos la humedad por GET
9.     $humedad = $ GET['humedad'];
10.          // Leemos la lluvia por GET
11.          $lluvia = $_GET['lluvia'];
12.          // Leemos la presión por GET
13.          $presion=$_GET['presion'];
14.          // Leemos la ubicación de la estación por GET (ID
   UBICACION)
15.          $ubicacion=$_GET['ubicacion'];
16.
17.          // Esta es la instrucción para insertar los valores
18.          $query = "INSERT INTO
   medida(temperatura,presion,humedad,llueve,idubicacion)
   VALUES('$temperatura','$presion','$humedad','$lluvia','$ubicacio
   n')";
19.          // Ejecutamos la instrucción
20.          mysqli_query($mysqli, $query);
21.          mysqli_close($mysqli);
22.     ?>
```

Este método para otro propósito no es demasiado seguro ya que cualquiera podría falsear los datos al ir visibles en la URL, pero para nuestro proyecto es suficiente, ya que haría falta saber el nombre de todos los valores que van en la URL.

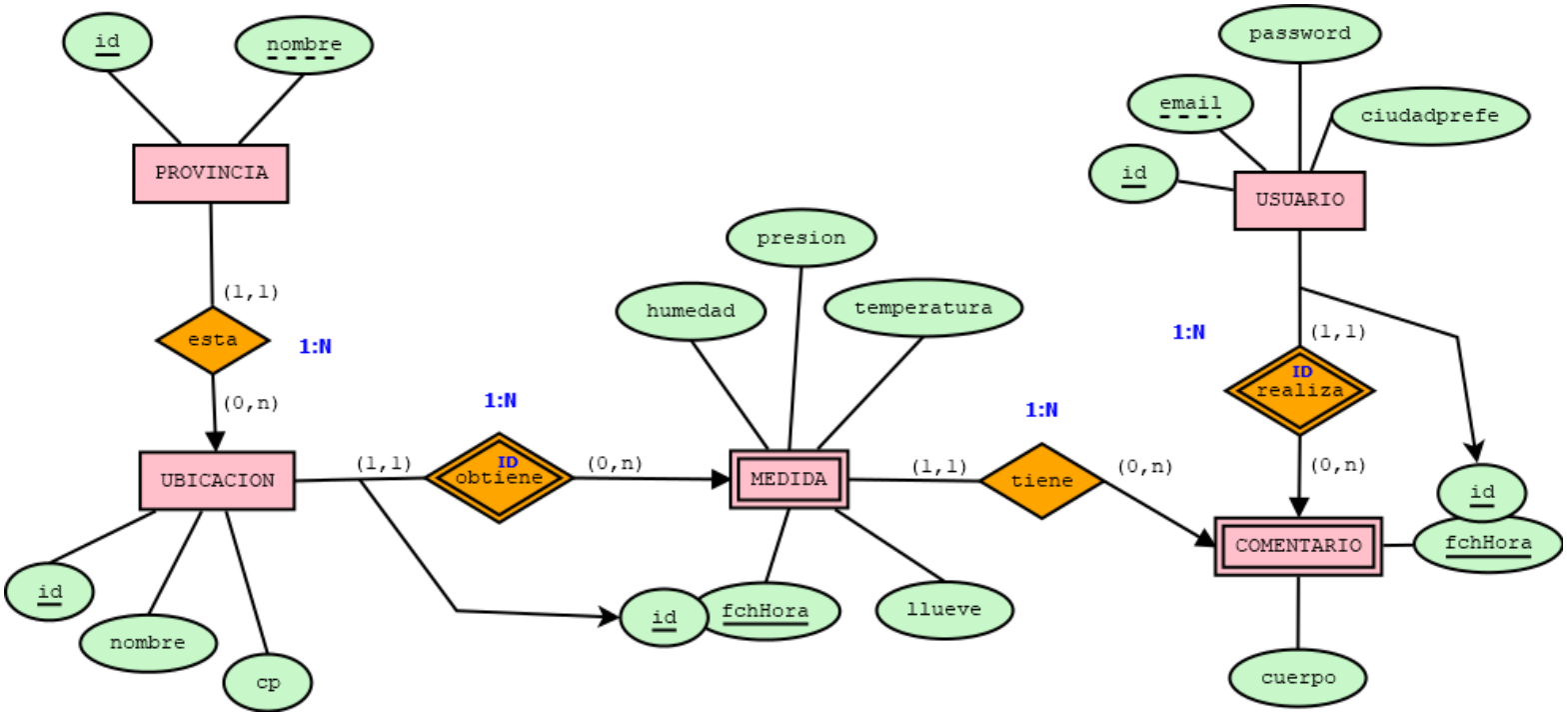
Base de datos

Diagrama ER

Vamos a simular que tenemos varios dispositivos por toda Andalucía recogiendo medidas para poder mostrarlas desde nuestra página web.

Los usuarios podrán hacer comentarios sobre las medidas como pequeñas predicciones según los datos obtenidos.

Para ello, he elaborado el siguiente diagrama entidad relación:



Entidades e interrelaciones

1. **ENTIDAD PROVINCIA:**

- ATRIBUTOS:
  - Id: Número Identificador de la provincia.
  - Nombre: Nombre de la provincia.

ATRIBUTO	TIPO	RANGO	CLAVE	OPCIONAL
ID	Campo Autoincrementado	0 a 50	PRIMARIA	NO
Nombre	Caracteres	MAXIMO 50 (VARIABLE)	ALTERNATIVA	NO

2. **ENTIDAD UBICACIÓN:**

- ATRIBUTOS:
  - Id: Número Identificador de la ubicación.
  - Nombre: Nombre de la ubicación.
  - CP: Código Postal de la ubicación.
  - IdProvincia: Número Identificador de la provincia a la que pertenece.

ATRIBUTO	TIPO	RANGO	CLAVE	OPCIONAL
ID	Campo Autoincrementado	0 a 50	PRIMARIA	NO
Nombre	Caracteres	MAXIMO 50 (VARIABLE)	NO	NO
CP	Caracteres	CARÁCTER 5	NO	NO

3. **ENTIDAD MEDIDA:**

- ATRIBUTOS:
  - Fchhora: Fecha y hora de la medida.
  - Temperatura: Temperatura tomada.
  - Presión: Presión tomada.
  - Humedad: Humedad tomada.
  - Llueve: Nos indica si no llueve, o lo hace de forma leve o moderada.
  - Id ubicación: Número Identificador de la ubicación en la que se toma la medida.

ATRIBUTO	TIPO	RANGO	CLAVE	OPCIONAL
fchHora	TIMESTAMP		NO	NO
Temperatura	DECIMAL 4,2	-40 a 125	NO	NO
Presión	DECIMAL 6,2	850 a 1300	NO	NO
Humedad	DECIMAL 4,2	0 a 100	NO	NO

Llueve	Caracter	Carácter Fijo (15)	NO	NO

4. ENTIDAD COMENTARIO:

- ATRIBUTOS:
  - Cuerpo: Cuerpo del mensaje.
  - Fchhora: Fecha y hora del mensaje.

ATRIBUTO	TIPO	RANGO	CLAVE	OPCIONAL
cuerpo	Caracter	CARÁCTER VARIABLE LÍMITE 300	NO	NO
fchHora	TIMESTAMP		NO	NO

5. ENTIDAD USUARIO:

- ATRIBUTOS:
  - Id: Número Identificador del usuario.
  - Email: Email del usuario.
  - Ciudadprefe: Ciudad favorita del usuario, se elige al principio con las ciudades disponibles y gracias a esta, se mostrará información personalizada para el usuario.

ATRIBUTO	TIPO	RANGO	CLAVE	OPCIONAL
ID	Campo Autoincrementado	0 a 50	PRIMARIA	NO
Email	Caracteres	MAXIMO 50 (VARIABLE)	ALTERNATIVA	NO
Password	Caracteres	MAXIMO 50 (VARIABLE)	NO	NO
CiudadPrefe	Caracteres	MAXIMO 50 (VARIABLE)	NO	NO

Diagrama relacional

- **R1=PROVINCIA** (**id**, nombre)
  CLAVE ALTERNATIVA = nombre
  Borrado Restringido, modificación cascada.

- **R2=UBICACION** (**id**, cp, idprovincia(FK1), nombre)
  CLAVE ALTERNATIVA = cp
  Clave ajena: idprovincia.
  idprovincia referencia al atributo id de la tabla PROVINCIA.
  Borrado Restringido, modificación cascada.

- **R3=MEDIDA** ((**idubicacion(FK2), fchhora)** temperatura, humedad, llueve, presión)
  Clave ajena: idubicacion.
  idubicacion referencia al atributo id de la tabla UBICACION.
  Borrado Restringido, modificación cascada.

- **R4=USUARIO** (**id**, email, password, ciudadprefe)
  CLAVE ALTERNATIVA = nombre.
  Borrado Restringido, modificación cascada.

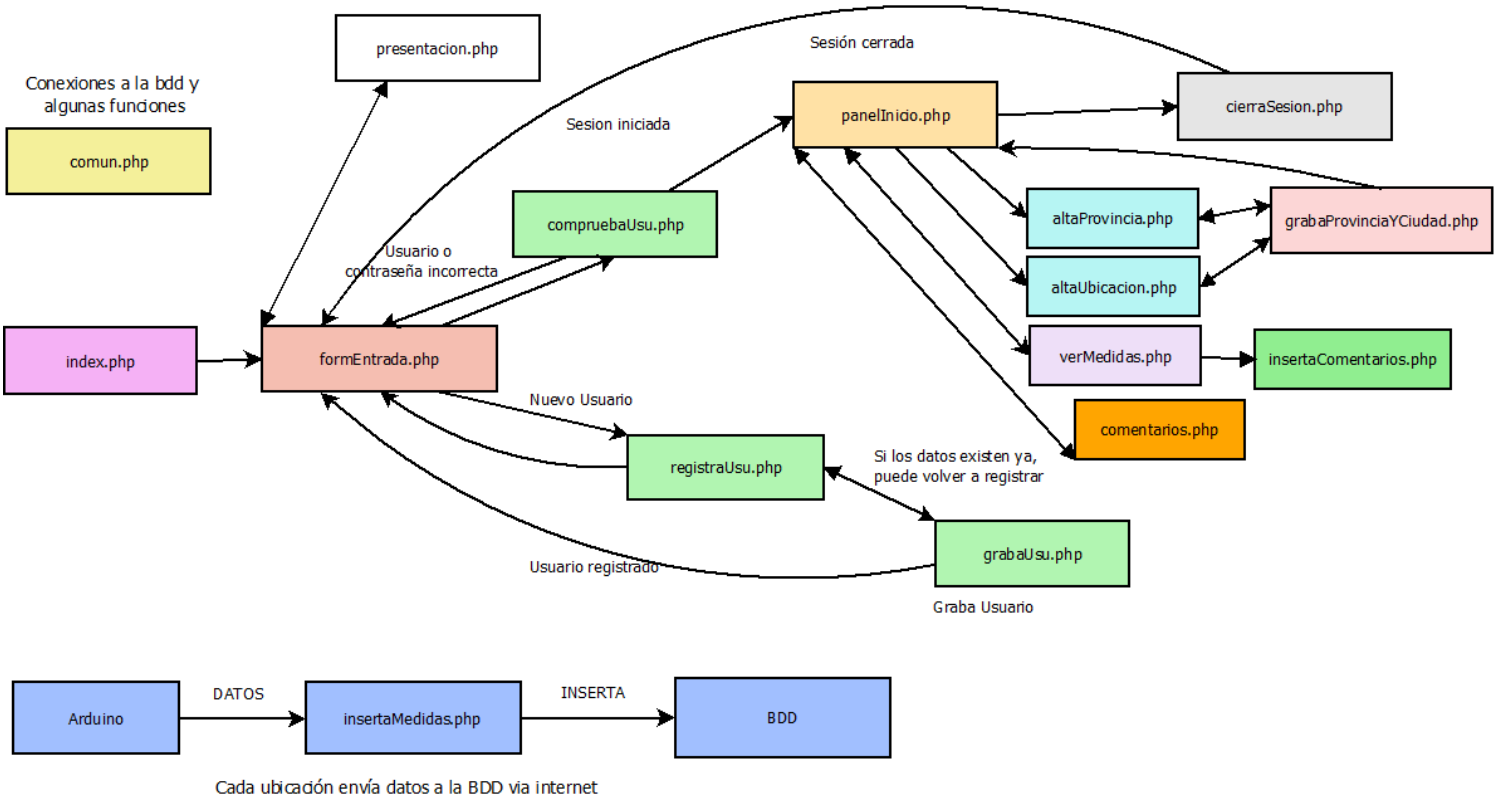
- **R5=COMENTARIO** ((**idusuario(FK4), fchhora,)** fchhoraMedida(FK3), idubicaciónMedida(FK3), cuerpo).
  Clave ajena: idubicacionMedida, fchhoraMedida, idusuario.
  idusuario referencia al atributo id de la tabla USUARIO.
  idubicaciónMedida referencia al atributo idubicacion de la tabla MEDIDA.
  fchhoraMedida referencia al atributo fchhora de la tabla MEDIDA

  Borrado Restringido, modificación cascada.

Diagrama de flujo

Este es el diagrama de flujo de la web.

Detallo el proceso de dar de alta a un usuario y su comprobación, cómo visualizar los datos y cómo dar de alta ciudades y provincias.



Cada ubicación envía datos a la BDD via internet

Mostrar datos

He creado una interfaz que muestra los datos de forma ordenada y agradable a la vista.

Esta interfaz está basada en PHP, HTML, CSS y MySql. Hago uso para el diseño, del framework Bootstrap que facilita mucho el proceso, y hace que consigamos resultados mucho más profesionales.

La página web estará protegida mediante un login para que sólo puedan acceder usuarios registrados. También tendrá la opción de registrarse en la página.

Al registrarse se mostrará una lista desplegable para seleccionar la ciudad de la cual quiere que se le muestre el tiempo.

También muestro una página donde me presento y doy algo de información sobre el proyecto.

Al iniciar sesión, se mostrará el tiempo actual de la ciudad elegida, también varias opciones de visualización de datos e incluso nos podrá llevar a un formulario mediante el que dar de alta ciudades y provincias, y un sistema de comentarios con el que compartir con otros usuarios información sobre los registros meteorológicos.

Mostraré máximas y mínimas del mes, su media y los últimos 48 registros.

La dirección para acceder es la siguiente: http://eltiempo-asir.ddns.net.

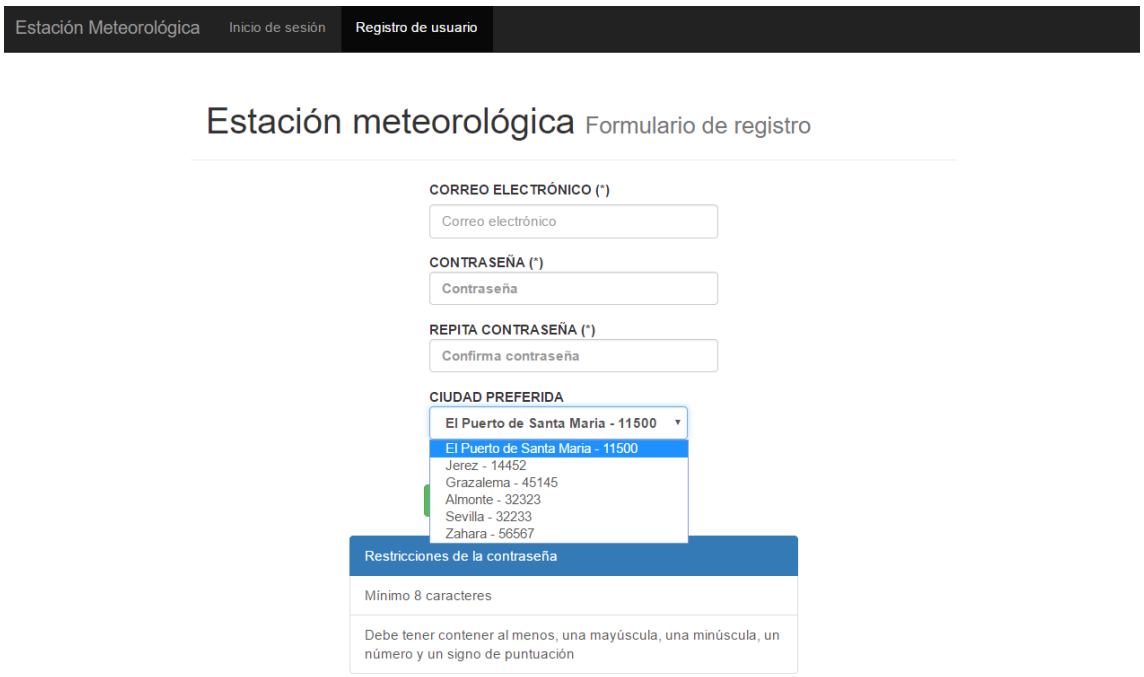
Página de inicio

Al entrar en la aplicación se nos muestra la página de inicio con un login.

Desde esta página podemos acceder a “Registro de Usuario” para dar de alta a un usuario o a “¿Quién soy?” para ver más información sobre el proyecto.

Registro de usuario

En el registro de usuario podemos dar de alta a nuevos usuarios indicando su email, la contraseña por partida doble y su ciudad favorita de entre todas las de la lista.



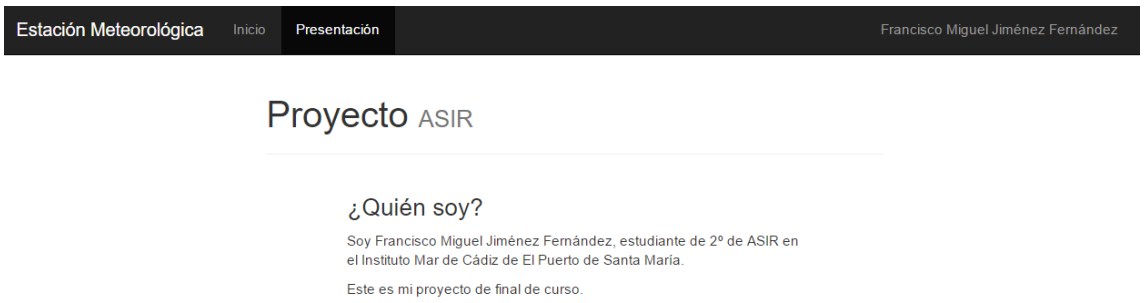
Cómo puede haber dos estaciones en la misma localidad, se indica al lado el código postal de la localización.

Las restricciones de la contraseña se especifican en el cuadro de la parte inferior de la página.

¿Quién soy?

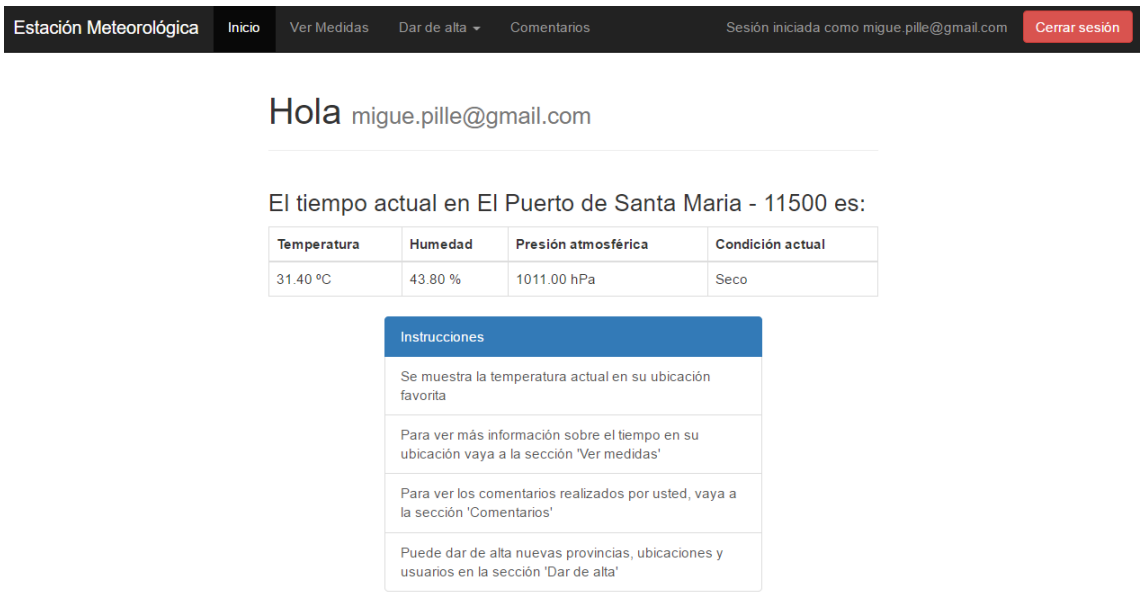
En esta parte de la página hago una breve presentación y explicación de mi proyecto y de las partes que lo componen.

Detallo cómo funciona el proyecto, quién soy, cómo se ha realizado el diseño de la web y mi motivación principal.



Panel de inicio

Una vez hemos iniciado sesión, la página nos dirige al panel de inicio. Desde aquí podemos acceder a las diferentes partes de la página.

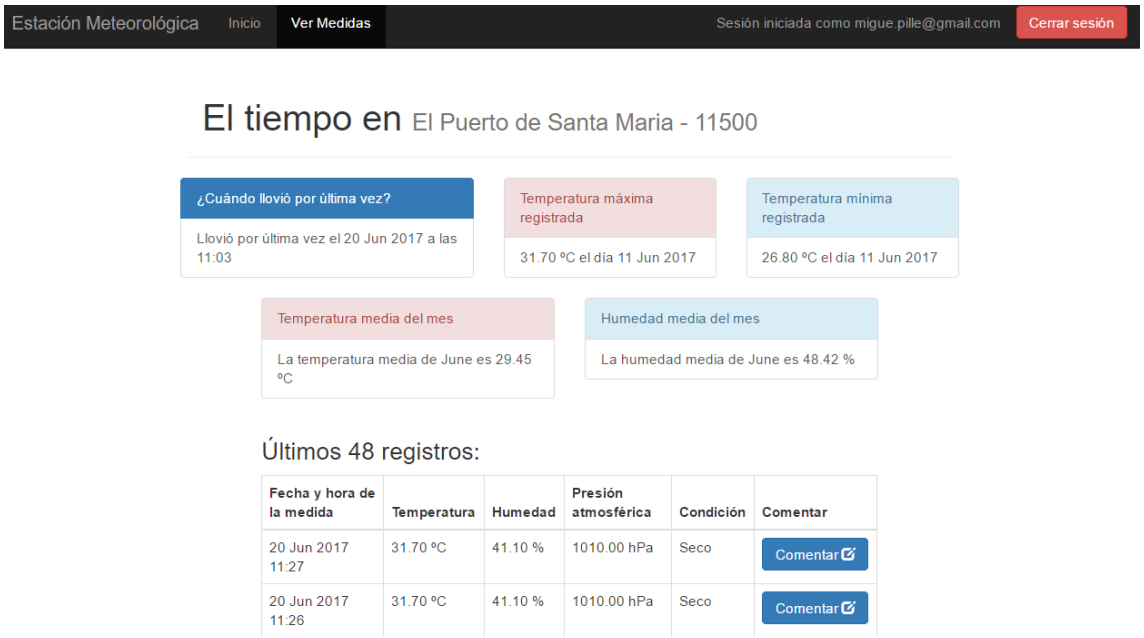


- Panel de inicio:
  - Tenemos la temperatura actual (Último registro recibido en la base de datos de la localidad).
  - Instrucciones para el uso de la aplicación.
- Ver medidas:
  - Para ver un registro detallado de las medidas.
- Dar de alta:
  - Para dar de alta a usuarios.
  - Para dar de alta provincias.
  - Para dar de alta nuevas ubicaciones dónde se instale una estación.
- Comentarios:
  - Ver los comentarios realizados en las medidas.
- Cerrar sesión:
  - Cierra la sesión del usuario.

Ver medidas

En este panel se visualizan algunos datos históricos como la temperatura y humedad media del mes, la temperatura máxima y mínima y cuándo llovió por última vez.

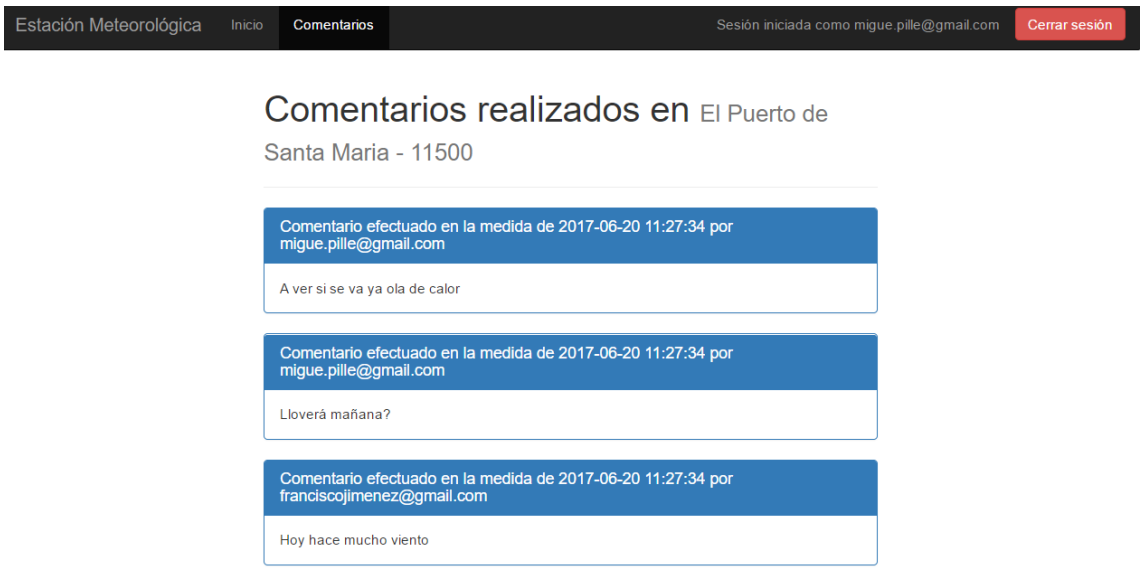
También se muestran los últimos 48 registros recogidos por la aplicación.



En cada registro, podemos hacer comentarios para posteriormente verlos desde la pestaña "Comentarios" del panel de inicio

Comentarios

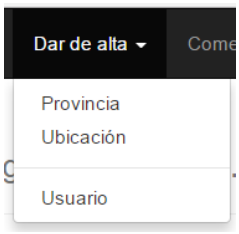
En la pestaña comentarios podemos ver todos los comentarios hechos en nuestra localidad favorita, tanto si los hemos hecho nosotros, como si no.



Aparece la fecha y hora de la medida, quién ha efectuado el comentario y el cuerpo del comentario.

Alta ubicación/provincia

También podemos dar de alta nuevas provincias, ubicaciones y usuarios desde la pestaña “Dar de alta”.



Si pulsamos en la opción “Provincia”, podemos dar de alta una nueva provincia con tan sólo indicar el nombre de ésta.

Al pulsar en “Ubicación”, podemos dar de alta una nueva ubicación dónde se haya instalado una estación.

Tenemos que seleccionar el código postal, el nombre y la provincia en la que se encuentra la ubicación. El nombre se concatenará con el código postal automáticamente para poder diferenciar dos ubicaciones con el mismo nombre.

Mejoras a realizar en el futuro

Con la realización de este proyecto, se me han ocurrido mejoras a implementar en un futuro, que ahora mismo, por falta de tiempo, presupuesto o porque es un prototipo en desarrollo, no he podido implementar.

Las mejoras a implementar son las siguientes:

- **Alimentación solar:** Una gran mejora sería eliminar al máximo los cables que componen nuestro dispositivo.
  Una buena forma y con la cual apoyamos el medio ambiente, es la utilización de una pequeña placa fotovoltaica que dote de energía y autonomía al aparato.
  La implementación es relativamente sencilla, ya que Arduino y los sensores que he utilizado destacan por su bajo consumo.

- **Transferencia de datos sin cables:** Otro avance, al igual que el de la autonomía eléctrica, sería eliminar el cable Ethernet de nuestro dispositivo.
  Esto lo podríamos hacer usando un **shield WIFI** que conectará nuestro dispositivo a internet sin cables.
  Una mejora de esto, sería usar un **shield 3G**, por lo cual tampoco dependeríamos de tener que conectarnos a una red WIFI, por lo que ya, nuestro montaje sería totalmente autónomo, ideal para colocar la estación en el campo o en lugares remotos, donde obtendríamos datos realmente interesantes.

- **Estanqueidad:** El agua es un enemigo natural de la electrónica. Una mejora del diseño de la carcasa y el uso de materiales estancos, incrementaría la durabilidad de nuestro dispositivo.

- **Anemómetro y veleta:** La obtención de datos sería mucho más interesante y completa si pudiésemos obtener datos de la velocidad y dirección del viento. Me he informado sobre el precio de estos dispositivos y su precio es demasiado elevado.

- **Cambios en el diseño:** Cuando la estación lleva bastante tiempo encendida, esta suele calentarse y puede falsear las medidas de temperatura. Habría que rediseñar el producto para poder disipar mejor el calor generado por la electrónica,

Conclusión

Gracias a la elaboración de este proyecto, he aprendido mucho sobre la programación en Arduino, el uso de diversos sensores y la programación web.

He aprendido a realizar algunos diseños en 3D en CATIA para realizar la estructura que soporta el proyecto.

Para mostrar los datos, me baso en distintos tipos de consultas a la base de datos y de sesiones y cookies en PHP, por lo que también he tenido que recordar el funcionamiento de éstas y usar funciones que hacía mucho que no usaba.

Gracias al uso y aprendizaje del framework "Bootstrap", podré darle mejor diseño a las aplicaciones web que haga a partir de ahora, ya que, con el uso de este framework, se simplifica muchísimo el proceso de maquetación de la web.

La creación de una interfaz amigable y sencilla hace que consultar los datos sea mucho más fácil que con los datos en bruto directamente. Gracias a esto, a partir de ahora, podré monitorizar el tiempo atmosférico de mi zona y podré compartirlo con el mundo, y quién sabe, si dentro de un tiempo, podré instalar nuevas estaciones meteorológicas en otras ubicaciones.

Bibliografía

Documentación de Bootstrap: http://getbootstrap.com/components/

W3 Schools para HTML, CSS, MySql y Bootstrap: https://www.w3schools.com/

Referencia de Arduino: https://www.arduino.cc/en/Reference/HomePage

Cómo medir con DHT22: https://www.luisllamas.es/arduino-dht11-dht22/

Cómo medir presión con BMP180: https://learn.sparkfun.com/tutorials/bmp180-barometric-pressure-sensor-hookup-

Pasar variables mediante la URL en PHP (Método GET): https://desarrolloweb.com/articulos/317.php

Documentos de https://fedoce.milaulas.com sobre PHP y MySql.

Aprendizaje del software CATIA: http://www.muchocatia.es/index.html

Para realizar el esquema de conexión de los componentes he usado Fritzing: http://fritzing.org/learning/