Universidad Zaragoza
1542

Trabajo Fin de Grado

Sistema Datalogger IoT y plataforma web PHP/CSS para monitorización de parámetros industriales.

IoT Datalogger System and PHP/CSS web platform for the monitoring of industrial parameters.

Autor:

Miguel Felipe Pardos

Directores:

Pedro Abad Martín
Juan Antonio Tejero Gómez

Titulación:

Grado en Ingeniería de Tecnologías Industriales

Escuela de Ingeniería y Arquitectura
2021

Resumen

Conocer los datos es la base fundamental para poder analizar, de forma objetiva, una situación actual y poder tomar decisiones fundamentadas. En la actualidad, gracias al avance tecnológico, es posible recopilar una gran cantidad de datos en tiempo real y tratarlos para quedarnos con aquellos que realmente nos aportan valor, así como enviarlos y recibirlos en cualquier tipo de dispositivo con conexión a la red.

En este trabajo se desarrolla una estructura capaz de tomar datos, procesarlos y mostrarlos en un entorno web. Para ello se utilizan los siguientes medios materiales prestados por la Universidad de Zaragoza: Raspberry Pi, dispositivos nodemcu ESP8266, sensores de temperatura y placas de conexiones. Por otro lado, para la programación se utiliza PHP/CSS, Arduino y Python.

Contenido

Índice Tablas

Índice Imágenes

1. Introducción.

El Internet de las cosas, *Internet of Things (IoT*) en inglés, es una de las nueve tecnologías que engloban la Industria 4.0, la cuarta revolución industrial. Esta tecnología en concreto busca la conexión de cualquier objeto a internet para permitir su control en tiempo real y desde cualquier lugar.

Imagen 1:Tecnologías 4.0

Aunque el término *IoT* está de moda y puede resultar muy novedoso porque es ahora cuando más se empieza a implantar, realmente el concepto ya existía desde hace años en círculos de investigación. Es más, fue en 1990 cuando John Romkey creó el primer objeto conectado a internet, una tostadora que se encendía y apagaba en remoto [1]. Pese a lo revolucionario de esta hazaña, la baja conectividad inalámbrica y el coste del hardware no hicieron posible apostar por estas ideas.

La base del proyecto es dotar a un objeto, el sensor de temperatura y humedad, de conexión a internet (Internet of Things). Esto se conseguirá al cablearlo a un dispositivo que tiene antena para conectarse a la red. Este dispositivo se programará para que lea cada cierto espacio de tiempo un dato y lo envíe mediante un canal a una placa de ordenador (Raspberry). La Raspberry será el elemento principal ya que es el que más funciones debe realizar. Por un lado, se programará para que almacene los datos

enviados por el dispositivo *IoT* en una base de datos y por otro lado, actuará como servidor web, por lo que habrá que diseñar una página web con la que pueda interactuar el usuario para mostrar y graficar los datos pertinentes.

En las siguientes imágenes se muestran un esquema conceptual de la conexión entre dispositivos y un diagrama del flujo que siguen los datos desde que se miden hasta que se visualizan y cuyas etapas que se irán detallando en esta memoria.

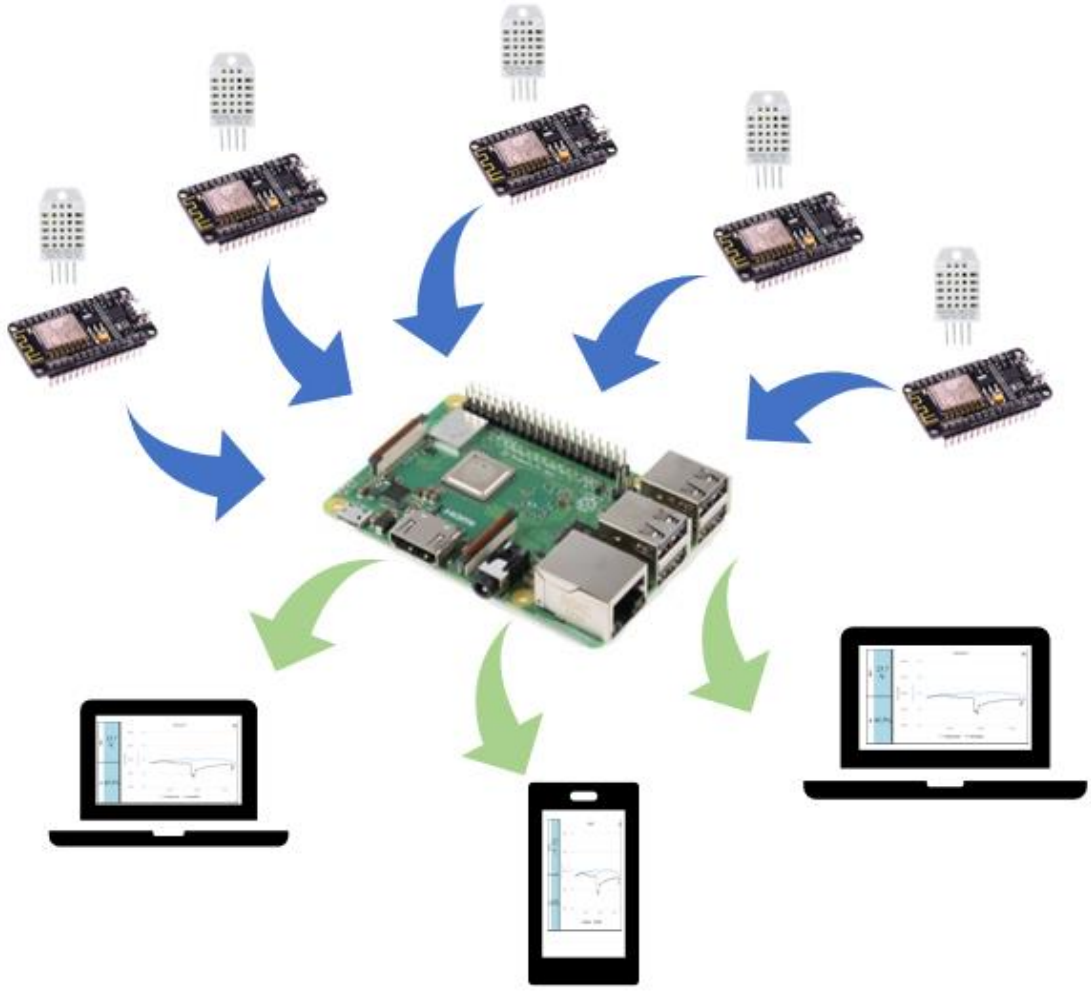


Imagen 2:Esquema conceptual interconexión entre dispositivos

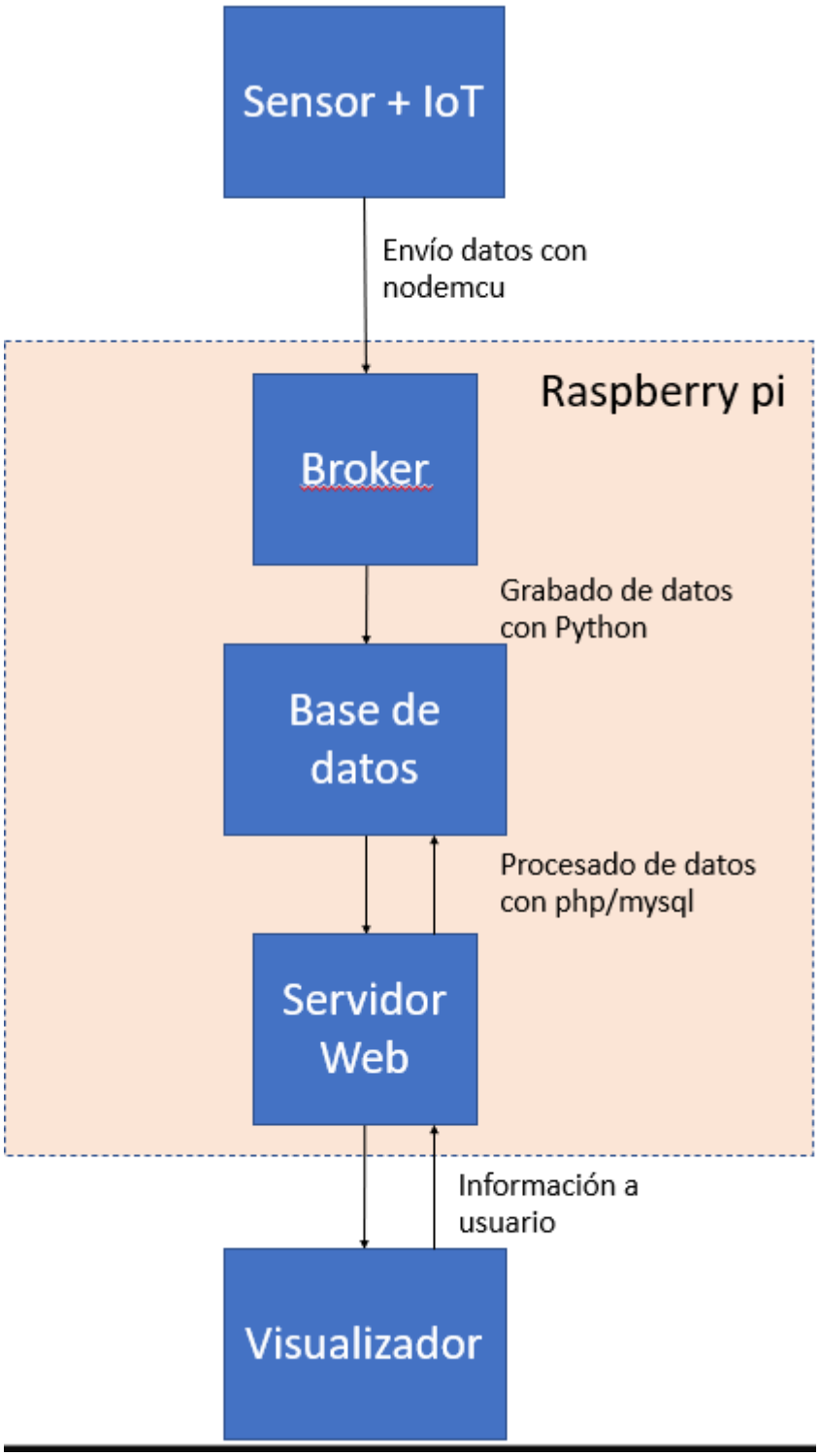


Imagen 3:Diagrama flujo de datos

En este trabajo se describen las fases que se han seguido para completar el proyecto propuesto. No obstante, aunque el orden de las fases se muestra tal y como se ha ido avanzando en el proyecto, evidentemente se han ido haciendo cambios en la programación según se detectaban errores, mejoras o simplemente cambios visuales hasta obtener el resultado definitivo.

2. Estado del Arte.

A continuación, se exponen distintas opciones existentes con las que se podría haber abordado este proyecto, además de la opción elegida y su justificación.

Placa de ordenador: Debe ser capaz de mantener buena conexión a internet, almacenar una base de datos y un servidor web. Además, se valora que la configuración y la instalación de programas sea sencilla o que exista una amplia documentación, y por supuesto que no tenga un precio inasumible. Puesto que la gran mayoría de las placas cumplen los requisitos básicos y tienen prácticamente las mismas conexiones (HDMI, 4 puertos USB, Ethernet…) los factores más importantes serán la cantidad de información que haya disponible y especialmente el precio.

Placa	Foto	Facilidad de uso	Precio (€)
Raspberry pi 3 b+		5	54,00
Asus Tinker Board S		3	138,63
Orange Pi Plus2		3	47,98
Le potato AML S905X-CC		3	106,90

Tabla 1: Comparación placas de ordenador

Dispositivo *IoT*: Ha de tener un tamaño reducido, suficientes pines de entrada para conectar como mínimo un sensor y una tensión de alimentación que sea suficiente para el sensor que se vaya a conectar.

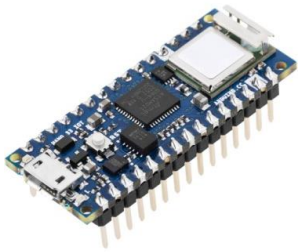
Dispositivo	Foto	Batería	Dimensiones (mm)	Programación	Precio (€)
M5StickC		SÍ	48,2x25,5x13,7	Programación por bloques	33,48
Nodemcu v2		NO	49x26	Arduino, micropython, Lua	7,99
Arduino Nano 33		NO	45x18	Arduino,IoT cloud	13,99
Arduino MKR1000		SÍ	61,5x25	Arduino, IoT cloud	33,76
Wemos D1 Mini		NO	34,2x25,6	Arduino, micropython, nodemcu	12,79

Tabla 2: Comparación Dispositivos IoT

Sensor de temperatura y humedad: Se quiere monitorizar ambos parámetros, pero esto se puede conseguir con un único sensor o con dos, uno para cada parámetro. Tiene que poder cubrir un amplio rango de temperaturas y tener una precisión y sensibilidad suficiente para tener unos datos fiables.

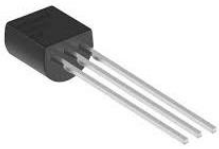
Dispositivo	Foto	Tem	Hum	Precio (€)
DHT11		Rango: 0-50ºC; Precisión: +-2ºC Sensibilidad: 1ºC	Rango: 20-90% Precisión: +-2% Sensibilidad:1%	5,99
DHT22		Rango: -40-80ºC Precisión: +-0,5ºC Sensibilidad: 0,1ºC	Rango: 0-100% Precisión: +-2% Sensibilidad: 0,1%	9,99
Ds18B20		Rango: -55ºC-125ºC Precisión +-0,5% (de -10ºC a 85ºC) Sensibilidad: 1ºC	No mide humedad	4,90 (5ud)
Tmp36		Rango: -40-125ºC Precisión: +-2ºC	No mide humedad	3,50
HR202		No mide temperatura	Rango: 20-95% Precisión: +- 5% Sensibilidad: 1%	19,42

Tabla 3: Comparación Sensores

Por las características del proyecto no se requiere ni gran capacidad de la placa de ordenador, ni buenos gráficos ni buena calidad de audio. Es por eso que los factores más determinantes son el precio y la cantidad de documentación y facilidad en su configuración. En ese sentido la opción más adecuada es la Raspberry pi 3 b+.

Respecto al dispositivo IoT, aunque todos podrían desempeñar la misma función, por un lado, se descarta el m5stickC porque pese a tener el sensor incorporado y ser fácil de programar eso también le resta flexibilidad y personalización. Por otro lado, aunque incorporar batería supone un plus, el precio se dispara en el caso del Arduino MKR1000. De los otros tres presentados el factor diferencial es el precio, por eso se elige el nodemcu v2.

Finalmente, el sensor, considerando la comodidad de utilizar uno sólo para leer la temperatura y la humedad deja como opciones el DHT11 y el DHT22. Las prestaciones técnicas del primero no se consideran suficientemente buenas por lo que la mejor opción es el DHT22.

Además de esos elementos principales se requiere de una caja con las dimensiones suficientes para el montaje del dispositivo IoT con el sensor y la fuente de alimentación de tipo micro USB.
Se ha elegido una caja de dimensiones 100x70x50 (mm). Aunque se han contemplado tamaños más reducidos, utilizando también placas de conexiones más cortas, finalmente se ha seleccionado la indicada anteriormente ya que también se da opción a introducir otros sensores posteriormente.

Respecto a el cargador directamente se ha optado por la opción más económica sin importar otros aspectos como la fecha de envío.

Imagen 4:Caja y cargador seleccionados

Con todos los componentes finalmente seleccionados se realiza el presupuesto necesario para abordar el proyecto.

Equipo	Precio ud. (€)	Cantidad	Precio Total (€)
Raspberry Pi 3 b+	54,00	1	54,00
Nodemcu v2	7,99	5	39,95
Sensor DHT22	9,99	5	49,95
Cables y Protoboard	2,20	5	11,00
Tarjeta SD	8,47	1	8,47
Caja	4.16	5	20,80
Cargador+USB	3,08	5	15,40
		Total:	**199,57 €**

Tabla 4: Presupuesto

3. Configuración de la Raspberry.

Para iniciar la *Raspberry* por primera vez se ha de disponer de otros equipos, algunos de los cuales, no serán necesarios después. Éstos son: un teclado, un ratón, un monitor, un cable HDMI, una tarjeta microSD y por supuesto la fuente de alimentación.

En primer lugar, puesto que la *Raspberry* viene "vacía" se debe cargar el sistema operativo en la tarjeta microSD, para ello simplemente hay que descargar desde la página oficial de *Raspberry* [2] el sistema operativo gratuito *Raspberry PI OS*.

A continuación, tras insertar la tarjeta microSD y conectar el resto de los dispositivos se inicia la *Raspberry* y se establece la conexión por Wifi. También puede conectarse por cable Ethernet, lo cual es más conveniente, siempre que sea posible ya que la conexión a Internet es más rápida y fiable.

Para trabajar con más comodidad se habilita el escritorio remoto mediante conexión VNC local [3]. Esto permite acceder a la *Raspberry* desde el ordenador habitual (como ejecutando un programa cualquiera) y dejar de necesitar el teclado, el monitor y el cable HDMI.

El siguiente paso es habilitar la *Raspberry* como servidor web para poder iniciar desde ahí la página web. Así pues, se instala el servidor web Apache y adicionalmente el software PHP que se usará para programar la página web [4].

Llegados a este punto ya es posible comenzar a trabajar y hacer pruebas con el diseño de la página web. No obstante, todavía es necesario instalar algunos programas para usar datos procedentes de la base de datos, este proceso se indica en el apartado 5. Finalmente, también se instalará el bróker '*mosquitto'* para recibir las mediciones del sensor como se describe en el apartado 6.

4. Programación Página Web.

La programación de la página web se realiza con ficheros PHP y CSS, no obstante intervienen dentro de los mismos otros lenguajes como HTML y Javascript.

PHP es un lenguaje de programación de código abierto que se utiliza comúnmente en los servidores web por su simple integración a HTML y su compatibilidad con numerosas bases de datos. Por su parte HTML es un lenguaje basado en etiquetas que define los elementos que se muestran en una página web (textos, imágenes, vídeos, etc), estos elementos se personalizan y se les da formato con el lenguaje css (Cascading style sheet) que se creó justamente para este fin de tener un abanico más grande de personalización y con mayor preecisión.

En primera instancia al poner la dirección web se abre por defecto el fichero "index.php" y automáticamente llama a otro fichero "password.php" que solicita al usuario ingresar usuario y contraseña con el fin de limitar el acceso únicamente a quien se desee.

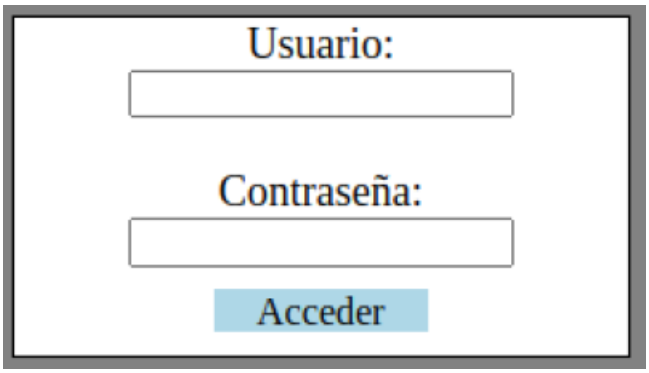


Imagen 5:Cuadro de diálogo petición de contraseña

En segunda instancia, tras haber introducido correctamente el usuario y la contraseña, en el fichero "index.php", que es el que se muestra por defecto cuando el usuario accede a la web, se crea un formulario basado en botones y selectores para que pueda elegir con rapidez y comodidad qué parámetros visualizar.

```
echo "<FORM ACTION=$_SERVER[PHP_SELF] METHOD=GET>";
echo "<rec><INPUT type='button' value='Inicio' onClick=location.href='index.php'>";

//Se crea el selector de sensor.
echo "<select name='Sel.sensor'>
            <option>sensor1</option>
            <option>sensor2</option>
            <option>sensor3</option>
            <option>sensor4</option>
            <option>sensor5</option>
    </select>";
//Se crean 2 botones rápidos para ver la última hora o las últimas 24h.
echo "<diahora><TD>Día</TD><INPUT TYPE='checkbox' name='24h' VALUE='24h'>";
echo "<TD>Hora</TD><INPUT TYPE='checkbox' name='1h' VALUE='1h'></diahora>";

//Se crean los formularios provisionales para consultar un periodo concreto.

echo "<fecha><TD>Fecha inicio</TD><INPUT TYPE='text' name='fechai' value='' >";
echo "<TD>Hora inicio</TD><INPUT TYPE='text' name='horai' value='' >";
echo "<TD>Fecha final</TD><INPUT TYPE='text' name='fechaf' value='' >";
echo "<TD>Hora final</TD><INPUT TYPE='text' name='horaf' value='' ></fecha>";

//Se crea el botón para acceder a  la gráfica.
```

Imagen 6:Código php creación de formulario

Imagen 7: Formulario selección datos

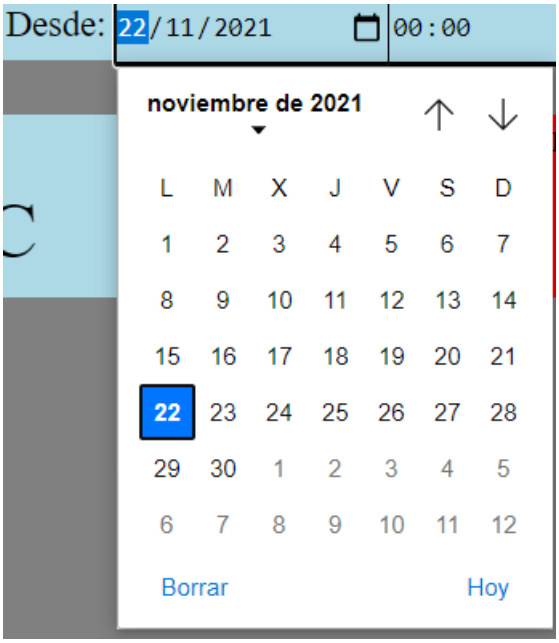


Imagen 8: Selector de fecha (Datepicker)

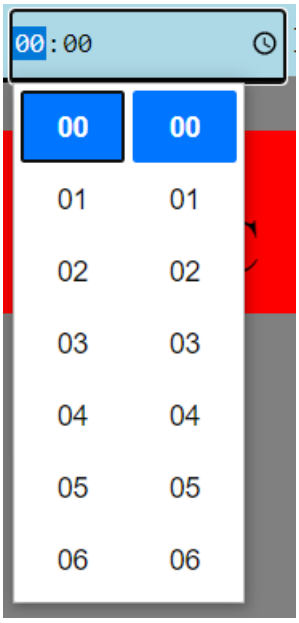


Imagen 9: Selector de hora (Hourpicker)

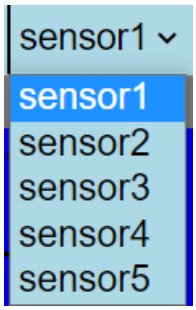


Imagen 10:Selector de la estancia

Con el objetivo de tener el código más organizado, en un fichero auxiliar se escriben las funciones más relevantes que se utilizan para hacer consultas SQL a la base de datos.

Tras introducir los parámetros que se desea visualizar, se accede a un otro fichero "grafica.php" en el cual, acorde con los datos recogidos en el formulario, se definen las consultas que se van a realizar, haciendo uso del fichero auxiliar "funciones.php".

```php
//Conectarse a la Base de datos
function conexion_db(){
        $db_host="localhost";
        $db_nombre="base_tfg";
        $db_usuario="root";
        $db_contra="password";

        $conexion=mysqli_connect($db_host,$db_usuario,$db_contra,$db_nombre);
        if(!$conexion){
                echo ("Fallo de conexion");
        }
        return $conexion;
}

//Desconectarse de la DB
function desconexion_db($conexion){
        $close = mysqli_close($conexion);

    if(!$close){
        echo "aqui";
    echo ("Error en base de datos");
    }
    return $close;
}

//Realizar consulta y almacenar resultados
function leer_db($consulta){
        $conexion=conexion_db();
        $datos=array();
        $i=0;
        $resultado=mysqli_query($conexion,$consulta);
        while($fila=mysqli_fetch_array($resultado)){
                $datos[$i]=$fila;
                $i++;

        }
        desconexion_db($conexion);
        return $datos;
```

Imagen 11:Código funciones de conexión con base de datos

Se utiliza la librería de *highcharts* para construir la gráfica. Es una librería gratuita para usos no comerciales que permite dibujar una gran variedad de gráficos con un alto grado de personalización. Los ficheros necesarios para utilizar esta librería pueden descargarse o importarse directamente desde su página web que es como se ha decidido hacer.

```
series: [{

    name: 'Temperatura',
    yAxis: 0,
    data: (function() {
       var data = [];
        <?php
            for($i = 0 ;$i<count($datos);$i++){
        ?>
        data.push([<?php echo $datos[$i]["date"];?>,<?php echo $datos[$i]["t"];?>]);
        <?php }
        ?>

    return data;
    })()
            },

            {

    name: 'Humedad',
    yAxis:1,
    data: (function() {
       var data = [];
        <?php
            for($i = 0 ;$i<count($datos);$i++){
        ?>
        data.push([<?php echo $datos[$i]["date"];?>,<?php echo $datos[$i]["h"];?>]);
        <?php }
        ?>
    return data;
    })()

}]
```

Imagen 12:Código para representación de series de temperatura y humedad

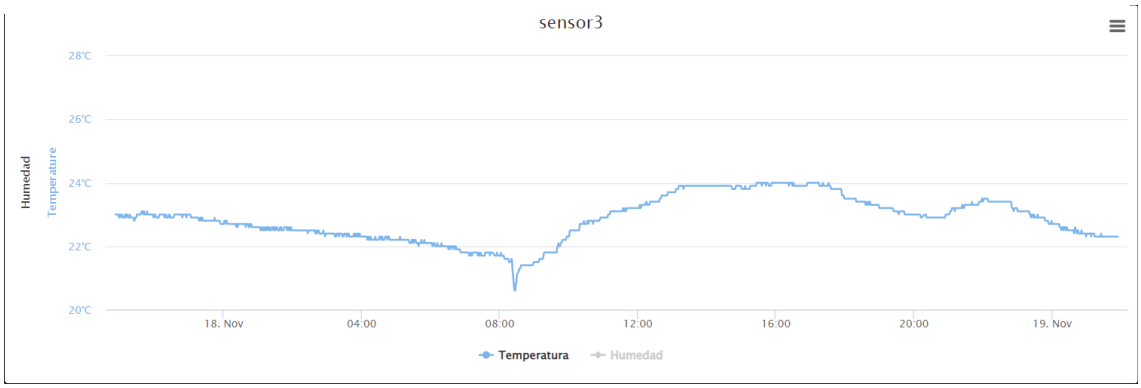


Imagen 13:Ejemplo Gráfica de la temperatura en las últimas 24h.

Para mejorar la visualización en tiempo real se añade otro fichero más "resumen.php" que muestra la temperatura actual medida por cada sensor. Además, los recuadros se comportan de manera dinámica cambiando de color en función de si se superan los límites superiores o inferiores recomendados por el RITE [5], es decir, más de 27ºC o menos de 21ºC. Como se ve en la siguiente imagen cuando la temperatura supera el límite máximo el recuadro se pone rojo y cuando se supera el límite inferior se vuelve azul.

Imagen 14: Recuadros Temperatura

La personalización del estilo de la gráfica y el resto de los elementos que se muestran por pantalla se detallan en el fichero estilos.css [6] pero también, para la gráfica, dentro del propio código de javascript según la librería de highcharts [7]. En la siguiente imagen se muestra un ejemplo del estilo que se le aplica al selector del sensor.

```
select{
    background-color:#A7D0D9;
    height:100%;
    width:9%;
    font-size:18px;
    border-style:solid;
    border-right:0px;
    border-bottom:0px;
    border-top:0px;
    border-left:0px;
    border-color:black;
}
```

Imagen 15: Código CSS aplicado al selector de estancia

Aquí se puede ver una imagen general con todos los elementos.

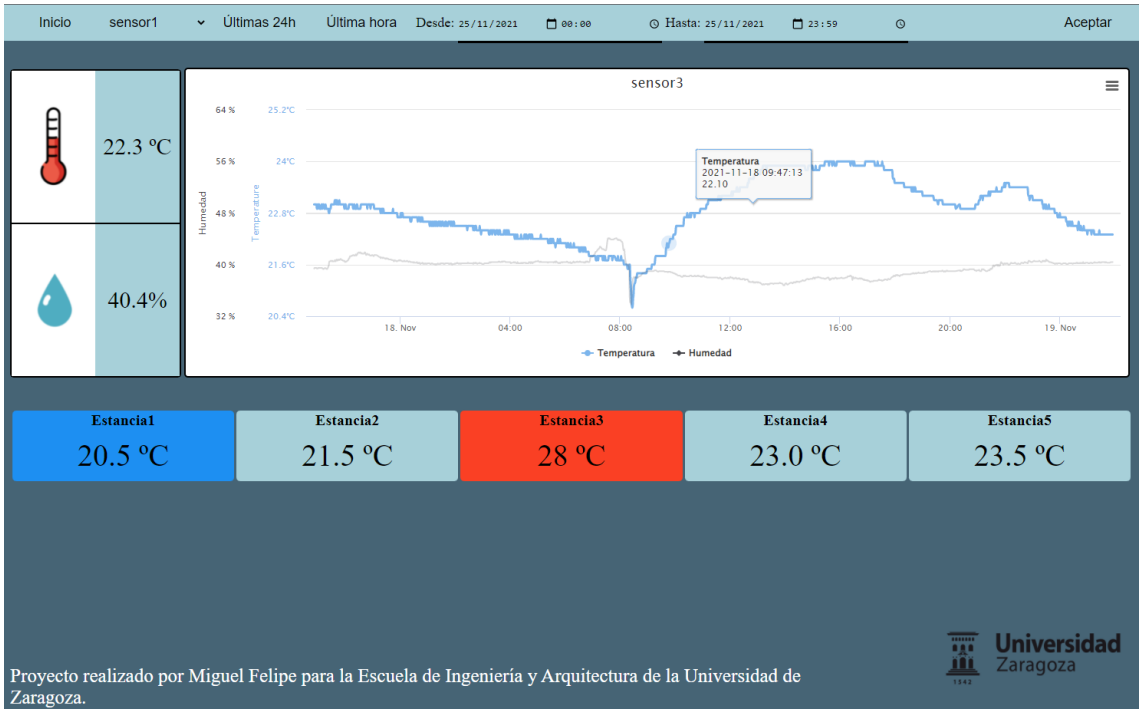


Imagen 16:Pantalla completa de la web

5. Creación Base de datos.

Para comenzar a crear base de datos primero se debe instalar mysql y mariadb server [8], [9]. MySQL es un sistema de gestión de bases de datos relacionales de (originalmente) código abierto con un modelo cliente-servidor. MariaDB es exactamente lo mismo (de hecho, se creó para reemplazar mysql manteniendo su compatibilidad) pero con nuevas funcionalidades.

Las bases de datos SQL son aquellas que están escrita en un lenguaje SQL (Structured Query Language) [10], considerado el lenguaje estándar en bases de datos

El SQL tiene 5 tipos de comandos:

- DDL (Data Definition Language): Define la estructura.
- DQL (Data Query Language): Sirve para hacer consultas.
- DML (Data manipulation language: Trata la manipulación o edición de los datos
- DCL (Data control language): se encarga de los permisos, derechos y otros controles

- TCL (Transaction Control Language): sirve para gestionar las transacciones

Adicionalmente se instala el programa phpmyadmin [11] que es una herramienta a la que se accede vía web, y gracias a su potente interfaz facilita la visualización y la gestión y creación de las tablas y las bases de datos

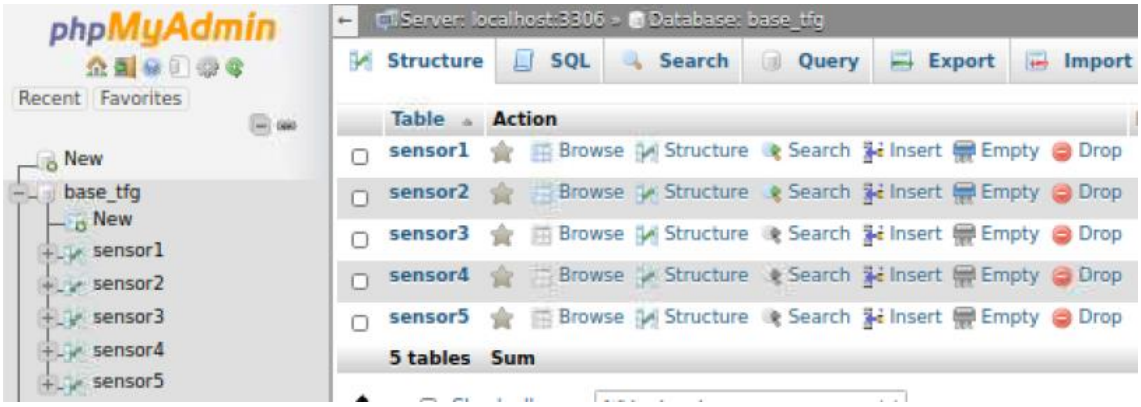


Imagen 17: Tablas en la base de datos

Puesto que el objetivo es medir parámetros de cinco estancias se decide crear una sola base de datos con cinco tablas (una para cada sensor). Se usa la misma estructura de tabla para todas ellas que consta de cuatro columnas

- Columna 1: ID. Número de identificación que permite que cada registro sea único y se pueda ordenar en el tiempo. Es de tipo *integer* y su valor se autoincremente cada vez que se escribe un nuevo registro.
- Columna 2: t. Es de tipo *varchar*, contiene los valores de temperatura.
- Columna 3: h. Es de tipo *varchar*, contiene los valores de humedad.
- Columna 4: date. Es de tipo *timestamp*, cada vez que se escribe un nuevo registro se graba la fecha y hora correspondiente.

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra
1	ID	int(11)			No	*None*		AUTO_INCREMENT
2	t	varchar(10)	utf8mb4_general_ci		No	*None*		
3	h	varchar(10)	utf8mb4_general_ci		No	*None*		
4	date	timestamp			No	current_timestamp()		

Imagen 18:Estructura de las tablas en base de datos

Tras ser creada, ésta queda totalmente activa para recibir nuevos registros y grabarlos como se explica en el siguiente apartado.

6. Lectura, envío y registro de datos.

En primer lugar, se instala Arduino IDE (en el ordenador de uso habitual, no en la *raspberry*) [12] para poder programar el dispositivo nodemcu. Arduino es una plataforma de electrónica de código abierto que permite programar una gran cantidad de microcontroladores.
Antes de empezar a escribir el código, para trabajar con el microcontrolador que se posee, hay que descargar la librería correspondiente al microcontrolador esp8266 [13].

Por otro lado, puesto que se ha decidido que el envío de datos se realice mediante protocolo mqtt hay que instalar en la *Raspberry* un *broker*, *mosquitto* [14] en este caso. MQTT (*Message Queue Telemetry Transport*) es un protocolo de comunicación específico para IoT que necesita muy poco ancho de banda y su red de tipo estrella está compuesta por un bróker y los clientes o nodos. Un *broker* es un servidor cuya función es gestionar la red en la que los clientes publican o reciben datos en un *topic* determinado. Los clientes pueden ser *Publisher* o *Subscriber* en función de si son los que envían los datos o los que los reciben.

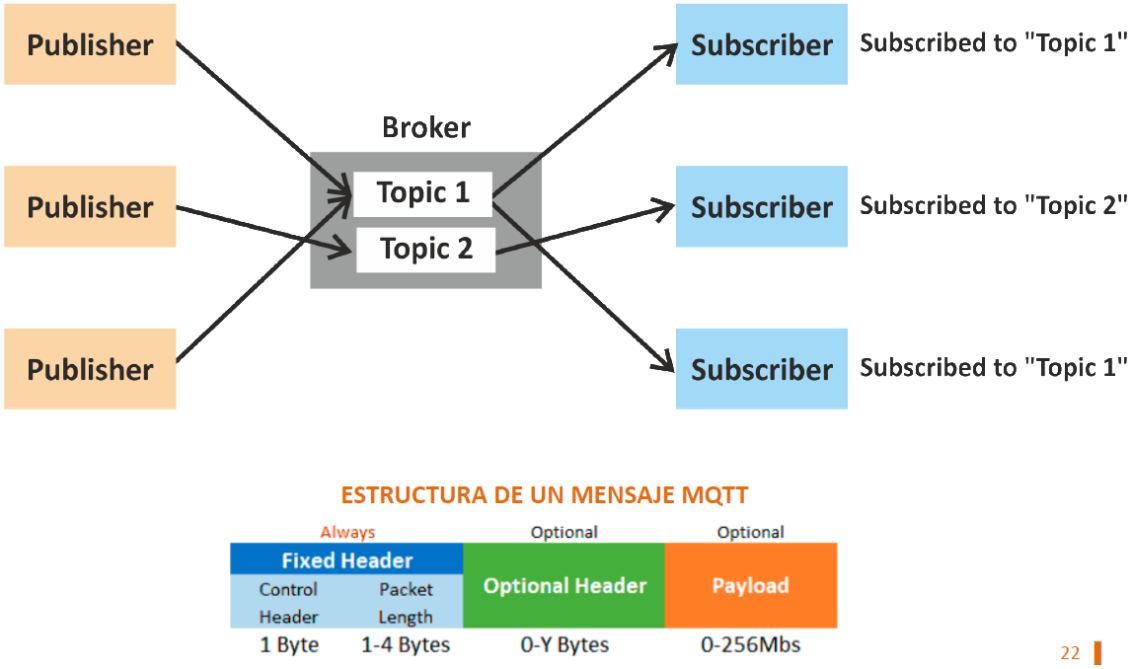


Imagen 19:Esquema conceptual funcionamiento de un broker

El código Arduino debe realizar dos funciones:

1) Leer en bucle cada cierto tiempo los valores de temperatura y humedad.
2) Enviar al bróker los datos leídos.

Conceptualmente debe seguir un esquema de este estilo:

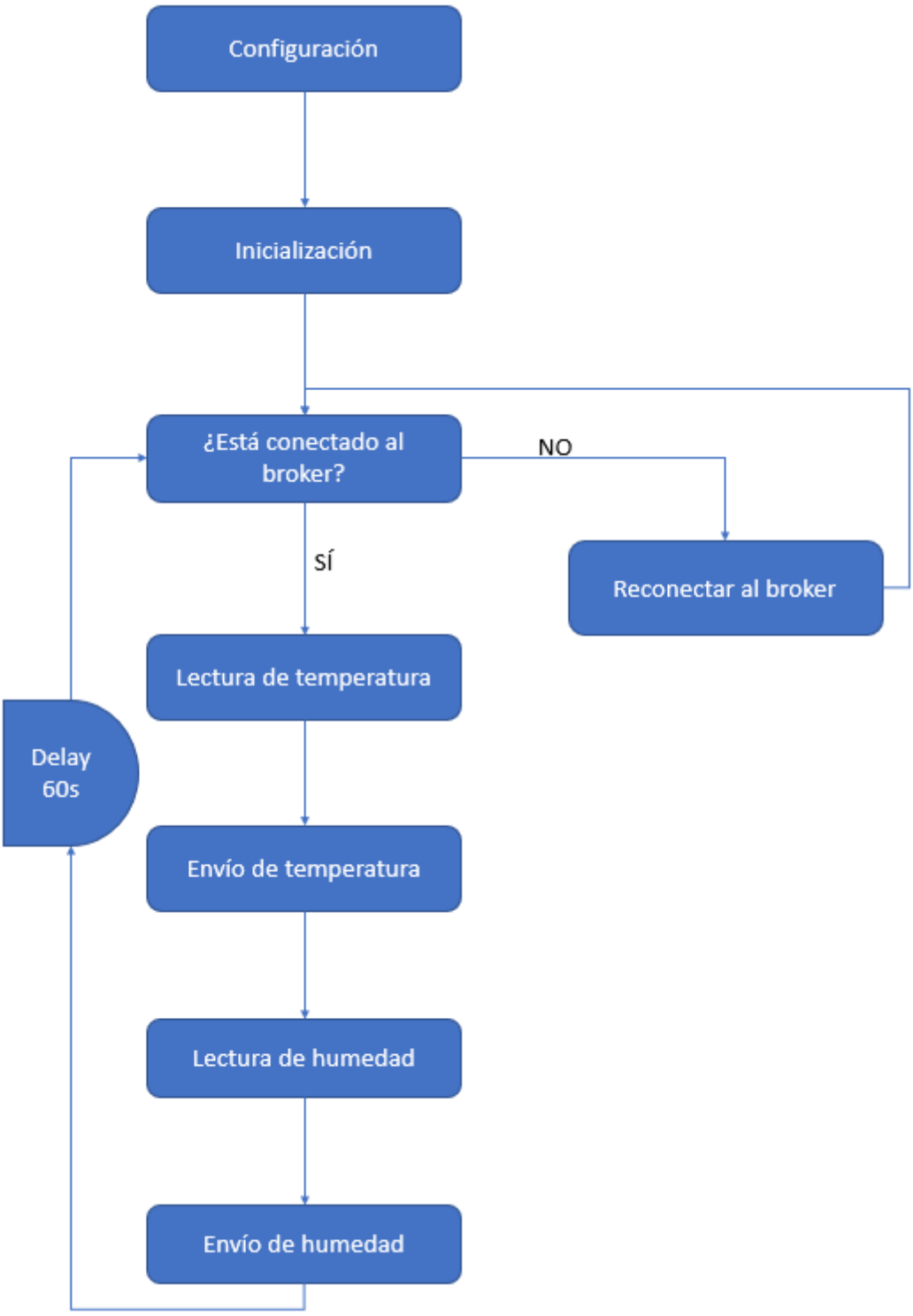


Imagen 20: Diagrama código Arduino para los IoT

En la cabecera del programa se definen las librerías que se utilizan, los datos de la WiFi a la cual conectarse, los datos del bróker al cual enviar los datos y el pin de lectura.

```
//Librerías para conectarse a la red WiFi y al broker. También la del sensor.
#include <ESP8266WiFi.h>
#include <PubSubClient.h>
#include "DHT.h"

//Datos de la red WiFi
const char* ssid = [redacted];
const char* password = [redacted];

//Datos del servidor broker mqtt (mosquitto)
const char* mqtt_server = "192.168.1.56"; //IP de la raspberry donde está el broker
const char* topic="room2/temperature"; //Topic al cual se envían los datos de temperatura
const char* topic2="room2/humidity";

//El cliente que se conectará a la IP y utilizará la librería PubSubClient
WiFiClient espCliente;
PubSubClient client(espCliente);

//Para el sensor2

//Defino los pines que se van a usar del dispositivo nodemcu
//En el pin 4 se reciben los datos
#define DHTTYPE DHT22
uint8_t DHTPin = D4;
```

Imagen 21:Cabecera código Arduino para los IoT

```
void setup() {
  pinMode(DHTPin, INPUT); //Se indica que el pin 4 es de entrada
  Serial.begin(115200);   //Se define la velocidad de comunicacion bits/s

  dht.begin();

  setup_wifi(); //Llamada a la función para conectarse a WiFi

  client.setServer(mqtt_server,1883); //Se define la dirección y puerto del broker

  const byte pinPowerDHT22 = 10;

}

void setup_wifi(){
  WiFi.begin(ssid,password);  //se indica a qué WiFi conectarse
  while (WiFi.status() != WL_CONNECTED){    //Hasta que no se conecte no se abandona esta función
    delay(500);
  }
}
```

Imagen 22:Código del Setup del Arduino para los IoT

A continuación, se define la inicialización del microcontrolador donde se índica conectarse a la WiFi y otros parámetros.

Este es el código que el microcontrolador repetirá en bucle en el que accederá cada 60 segundos a la función '*publish'* a no ser que no haya conexión con el bróker que entonces tratará de volver a conectarse.

```
void publish(){ //Función para publicar los datos.
  Serial.println("hey");
  float hum = dht.readHumidity(); //"Mide la humedad"
  char* humedad="";
  dtostrf(hum,10,3, humedad); //Se convierte el valor de la humedad en una cadena de ancho 10 y 3 dígitos decimales.
  client.publish(topic2, humedad); //Publica el valor de la humedad
  //Serial.println(humedad);
  float temp = dht.readTemperature(); //Mide la temperatura
  //Serial.println(temp);
  char* temperatura="";
  dtostrf(temp,10,3, temperatura);  //Se convierte el valor de la temperatura en una cadena de ancho 10 y 3 dígitos decimales.
  client.publish(topic, temperatura);  //Publica el valor de la temperatura

}

void loop() {
  if (!client.connected()){ //Se ejecuta si no hay conexión con el broker
    reconnect();
  }
  else {
    publish();
    delay(20000);   //Se van a tomar datos de la temperatura cada 20 segundos
  }

}
```

Imagen 23:Bucle del código Arduino para los IoT

La función principal *publish* lee los datos de temperatura y humedad como tipo *float* pero antes de publicarlos los convierte en una cadena para transmitirlos por protocolo mqtt.

La ventaja de usar el mismo modelo de *IoT* para todas las ubicaciones es que se puede replicar y grabar este mismo programa en cada dispositivo cambiando únicamente los k a los cuales enviar los datos.

Tras enviar los datos ahora toca recogerlos e introducirlos en la base de datos. Para esta operación se utiliza el lenguaje python que utilizando la librería adecuada facilitará esta tarea. Importando la librería *paho* sólo es necesario llamar a una función que se activará cuando el bróker reciba dos mensajes en los *topics* pertinentes.
Cuando se activa ejecuta las operaciones SQL para grabar los datos en la base de datos.

```
while (1):
    msg= mqtt.simple(["room2/temperature","room2/humidity"], qos=0, msg_count=2, retained=False, hostname="192.168.1.56",
        port=1883, client_id="", keepalive=60, will=None, auth=None, tls=None)

    hum=float(msg[0].payload)
    temp=float(msg[1].payload)
    write_db(temp, hum)
```

Imagen 24:Bucle de Python para recibir y escribir mensajes

Puesto que se utiliza un programa python para cada sensor hay que ejecutar los cinco simultáneamente.

ID	t	h	date
1	25.0	45.9	2021-09-24 20:49:47
2	25.0	45.9	2021-09-24 20:50:12
3	24.9	45.9	2021-09-24 20:50:37
4	25.0	45.9	2021-09-24 20:51:02
5	23.1	58.7	2021-09-25 09:46:50
6	23.0	58.9	2021-09-25 09:47:16
7	22.9	59.1	2021-09-25 09:47:41
8	23.0	59.1	2021-09-25 09:48:06
9	23.2	59.0	2021-09-25 09:48:31
10	23.0	58.9	2021-09-25 09:48:56
11	23.0	58.8	2021-09-25 09:49:21
12	22.7	58.7	2021-09-25 09:49:46
13	23.1	58.8	2021-09-25 09:50:11

Imagen 25:Ejemplo escritura en BBDD

7. Montaje y cableado.

El conjunto consta de la placa de conexiones, el dispositivo nodemcu, el sensor de temperatura, una resistencia de 10k, y cableado.

Se colocan el sensor y el nodemcu y se conecta siguiendo el diagrama proporcionado en el *datasheet* del sensor DHT22 (anexo I) [15]. De tal manera que el pin 1 es la alimentación, el pin 3 la tierra, y el pin 2 los datos, que se conectan a uno de los pines del nodemcu, en este caso el pin 4.

También debe conectarse el dispositivo nodemcu a la alimentación puesto que no dispone de batería.

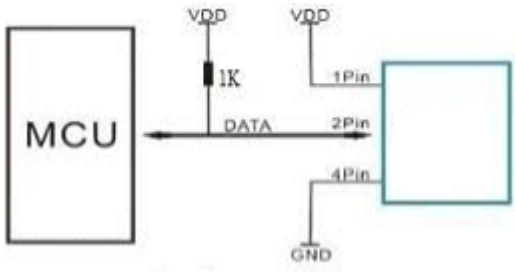


Imagen 27:Esquema de conexiones

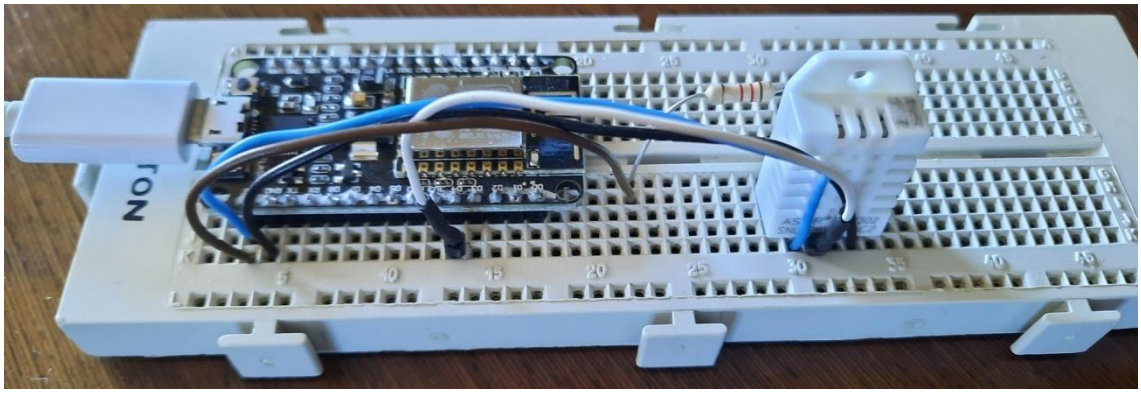


Imagen 26: Montaje y cableado del IoT con el sensor

Finalmente, se hará el montaje definitivo en una placa más pequeña de uniones soldadas y se introducirá en la caja (Imagen 28) y se distribuirán en las zonas deseadas, según indican las flechas verdes en el croquis de la imagen 29 procurando no ubicarlas junto a fuentes de calor o de frío que distorsionen la realidad.

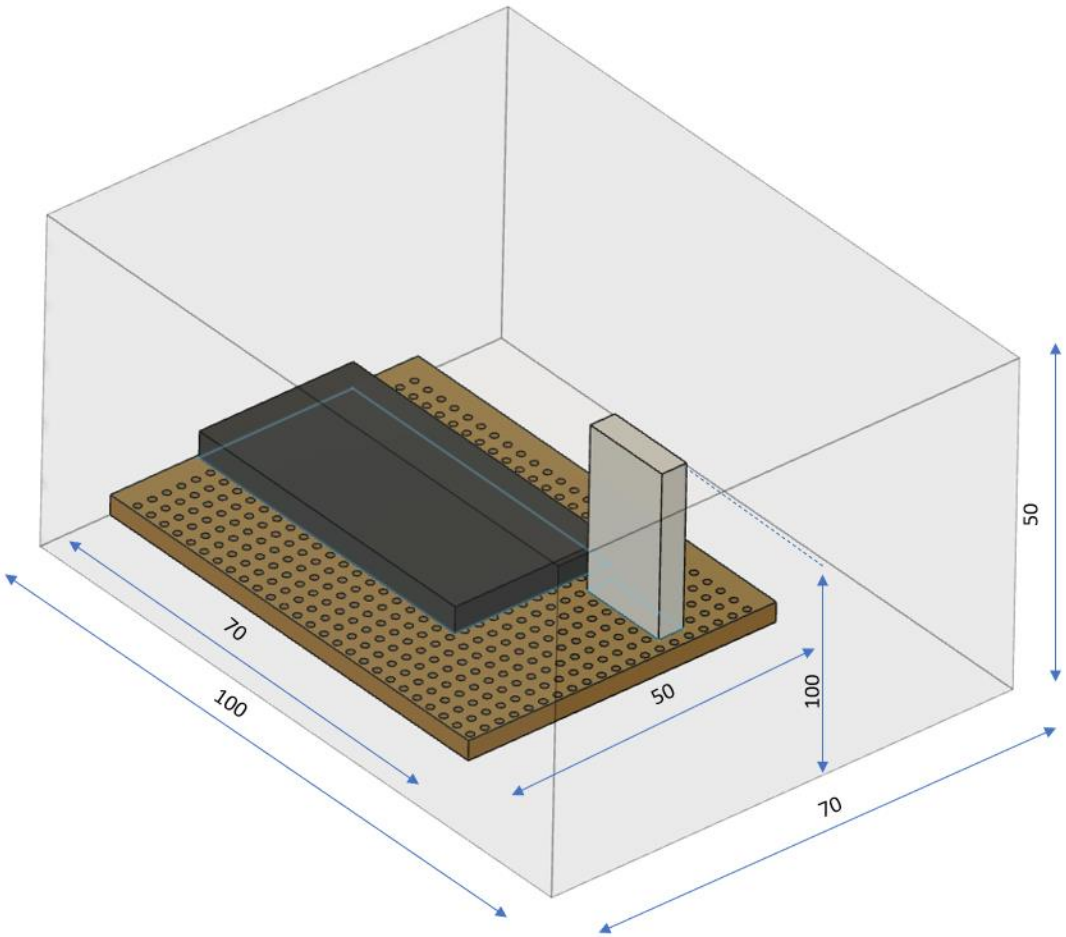


Imagen 28: Croquis montaje dentro de la caja

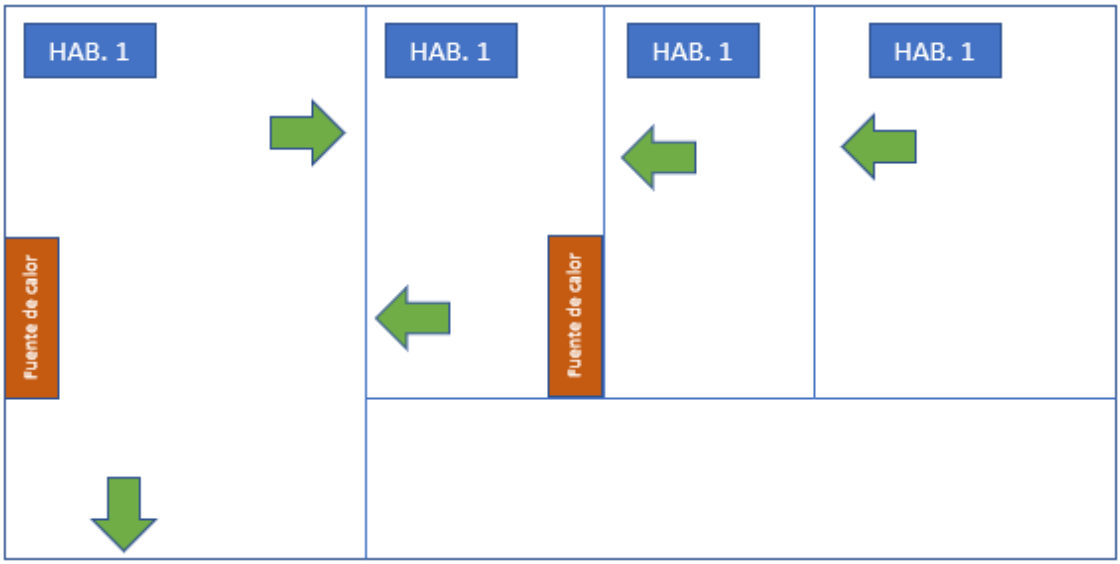


Imagen 29:Croquis ubicación componentes

8. Acciones de mejora

Tras haber acabado el proyecto cumpliendo los objetivos previamente planteados se describen tres ejes de mejora:

En primer lugar, reducir el tamaño del montaje en la placa de conexiones, bien utilizando una placa más pequeña o bien utilizando uniones soldadas.

Por otro lado, el diseño de la página web siempre es mejorable y se pueden añadir funcionalidades como por ejemplo dejar registrarse a usuarios nuevos con su propia contraseña.

Finalmente, en la misma línea que el primer punto, reducir el tamaño de la caja en la que se integra el montaje. Para ello, la mejor solución sería diseñarla personalmente para imprimirla en 3D.

9. Conclusiones

El objetivo de este TFG era lograr visualizar en tiempo real los datos de temperatura y humedad en cinco estancias. Para ello era necesario programar varios dispositivos, crear una base de datos y programar una página web. El desconocimiento de estos conceptos al inicio del TFG es lo que ha supuesto la mayor dificultad a pesar de la cantidad de información disponible en internet.

Finalmente, comentar que para la toma y visualización de datos existen múltiples y muy diferentes maneras de hacerlo y corresponde decidirse por una u otra en función de la simplicidad que se desee, la rapidez o el grado de personalización, ya que según para qué, habrá una opción mejor que otra. Para este proyecto en particular, la parte que probablemente sería conveniente plantearla de otra manera sería la programación en web ya que en la actualidad existen múltiples opciones que permiten una diseñar una

web de manera muy sencilla y ahorraría mucho tiempo en una tarea que no aporta tanto valor al proyecto como tiempo hay que dedicarle.

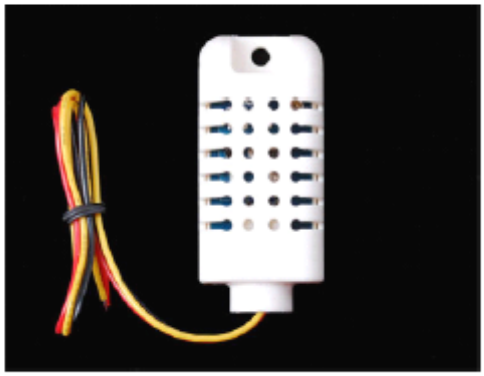
Bibliografía

[1] [En línea]. Available: https://www.iotworldonline.es/.
[2] [En línea]. Available: https://www.raspberrypi.org.
[3] [En línea]. Available: https://www.raspberrypi.org/documentation/remote-access/vnc/README.md.
[4] [En línea]. Available: https://www.raspberrypi.org/documentation/remote-access/web-server/apache.md.
[5] [En línea]. Available: https://www.boe.es/buscar/doc.php?id=BOE-A-2007-15820.
[6] [En línea]. Available: https://www.w3schools.com/css/default.asp.
[7] [En línea]. Available: https://api.highcharts.com/highcharts/.
[8] [En línea]. Available: https://www.raspberrypi.org/documentation/remote-access/web-server/apache.md.
[9] [En línea]. Available: https://projects.raspberrypi.org/en/projects/lamp-web-server-with-wordpress/4.
[10] [En línea]. Available: https://www.ticportal.es/glosario-tic/base-datos-sql.
[11] [En línea]. Available: https://projects.raspberrypi.org/en/projects/lamp-web-server-with-wordpress/4.
[12] [En línea]. Available: https://www.arduino.cc/en/software.
[13] [En línea]. Available: https://github.com/esp8266/arduino.
[14] [En línea]. Available:] https://programarfacil.com/esp8266/mqtt-esp8266-raspberry-pi/.
[15] [En línea]. Available: https://cdn-shop.adafruit.com/datasheets/Digital+humidity+and+temperature+sensor+AM2302.pdf.

ANEXO I: Especificaciones Sensor DHT22

Your specialist in innovating humidity & temperature sensors

Standard AM2302/DHT22



AM2302/DHT22 with big case and wires

Digital relative humidity & temperature sensor AM2302/DHT22

1. Feature & Application:

*High precision
*Capacitive type
*Full range temperature compensated
*Relative humidity and temperature measurement
*Calibrated digital signal
*Outstanding long-term stability
*Extra components not needed
*Long transmission distance, up to 100 meters
*Low power consumption
*4 pins packaged and fully interchangeable

2. Description:

AM2302 output calibrated digital signal. It applys exclusive digital-signal-collecting-technique and humidity sensing technology, assuring its reliability and stability. Its sensing elements is connected with 8-bit single-chip computer.

Every sensor of this model is temperature compensated and calibrated in accurate calibration chamber and the calibration-coefficient is saved in type of programme in OTP memory, when the sensor is detecting, it will cite coefficient from memory.

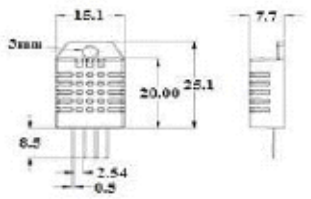
Small size & low consumption & long transmission distance(100m) enable AM2302 to be suited in all kinds of harsh application occasions. Single-row packaged with four pins, making the connection very convenient.

3. Technical Specification:

Model	AM2302	
Power supply	3.3-5.5V DC	
Output signal	digital signal via 1-wire bus	
Sensing element	Polymer humidity capacitor	
Operating range	humidity 0-100%RH;	temperature -40~80Celsius
Accuracy	**humidity +-2%RH**(Max +-5%RH);	temperature +-0.5Celsius
Resolution or sensitivity	humidity 0.1%RH;	temperature 0.1Celsius
Repeatability	humidity +-1%RH;	temperature +-0.2Celsius
Humidity hysteresis	+-0.3%RH	
Long-term Stability	+-0.5%RH/year	
Interchangeability	fully interchangeable	

Your specialist in innovating humidity & temperature sensors

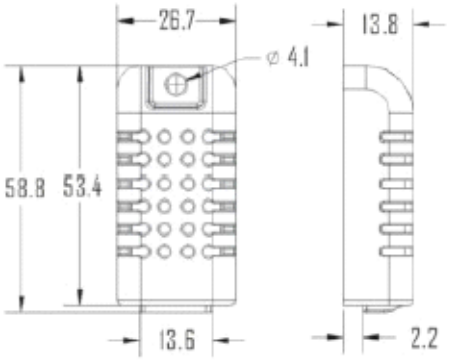
4. Dimensions: (unit----mm)



Pin sequence number: 1 2 3 4 (from left to right direction).

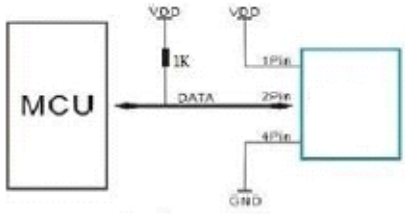
Pin	Function
1	VDD---power supply
2	DATA--signal
3	GND
4	GND

Standard AM2302's dimensions as above



Big case's dimensions as above
Red wire--power supply, Black wire--GND
Yellow wire--Data output

5. Electrical connection diagram:



6. Operating specifications:

(1) Power and Pins

Power's voltage should be 3.3-5.5V DC. When power is supplied to sensor, don't send any instruction to the sensor within one second to pass unstable status. One capacitor valued 100nF can be added between VDD and GND for wave filtering.

(2) Communication and signal

1-wire bus is used for communication between MCU and AM2302. (Our 1-wire bus is specially designed, it's different from Maxim/Dallas 1-wire bus, so it's incompatible with Dallas 1-wire bus.)

Illustration of our 1-wire bus:

Your specialist in innovating humidity & temperature sensors

DATA=16 bits RH data+16 bits Temperature data+8 bits check-sum

Example: MCU has received 40 bits data from AM2302 as

0000 0010 1000 1100 0000 0001 0101 1111 1110 1110

16 bits RH data | 16 bits T data | check sum

Here we convert 16 bits RH data from binary system to decimal system,

0000 0010 1000 1100 → 652

Binary system | Decimal system

RH=652/10=65.2%RH

Here we convert 16 bits T data from binary system to decimal system,

0000 0001 0101 1111 → 351

Binary system | Decimal system

T=351/10=35.1℃

When highest bit of temperature is 1, it means the temperature is below 0 degree Celsius.

Example: 1000 0000 0110 0101, T= minus 10.1℃

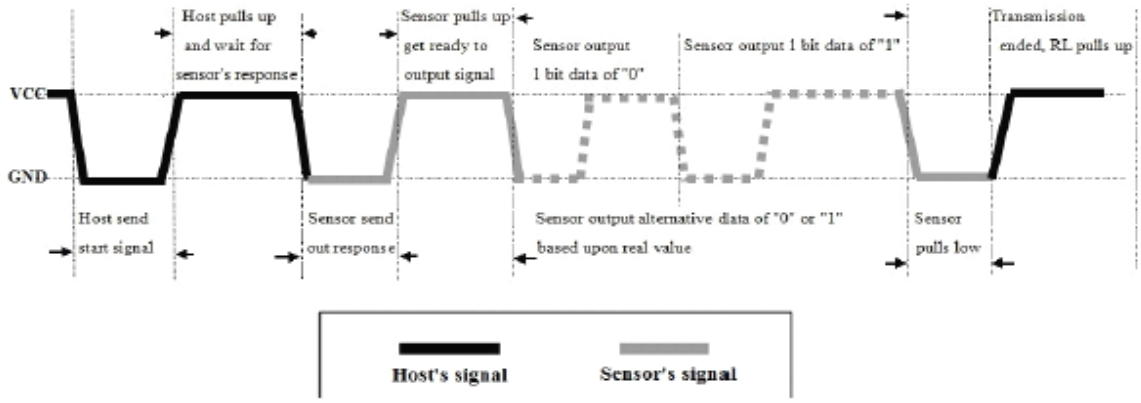
16 bits T data

Sum=0000 0010+1000 1100+0000 0001+0101 1111=1110 1110

Check-sum=the last 8 bits of Sum=1110 1110

When MCU send start signal, AM2302 change from standby-status to running-status. When MCU finishs sending the start signal, AM2302 will send response signal of 40-bit data that reflect the relative humidity and temperature to MCU. Without start signal from MCU, AM2302 will not give response signal to MCU. One start signal for one response data from AM2302 that reflect the relative humidity and temperature. AM2302 will change to standby status when data collecting finished if it don't receive start signal from MCU again.

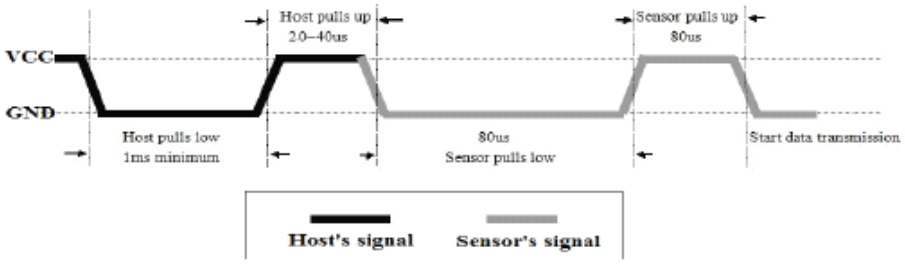
See below figure for overall communication process, **the interval of whole process must beyond 2 seconds.**



1) Step 1: MCU send out start signal to AM2302 and AM2302 send response signal to MCU

Data-bus's free status is high voltage level. When communication between MCU and AM2302 begins, MCU will pull low data-bus and this process must beyond at least 1~10ms to ensure AM2302 could detect MCU's signal, then MCU will pulls up and wait 20-40us for AM2302's response.

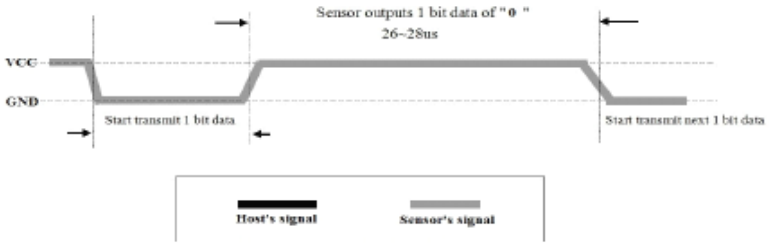
When AM2302 detect the start signal, AM2302 will pull low the bus 80us as response signal, then AM2302 pulls up 80us for preparation to send data. See below figure:

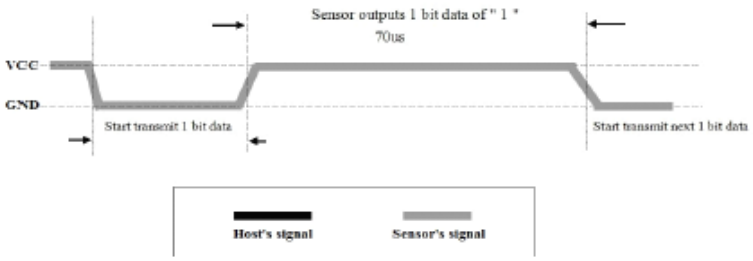


---

2). Step 2: AM2302 send data to MCU

When AM2302 is sending data to MCU, every bit's transmission begin with low-voltage-level that last 50us, the following high-voltage-level signal's length decide the bit is "1" or "0". See below figures:





Attention:

If signal from AM2302 is always high-voltage-level, it means AM2302 is not working properly, please check the electrical connection status.

7. Electrical Characteristics:

Items	Condition	Min	Typical	Max	Unit
Power supply	DC	3.3	5	6	V
Current supply	Measuring	1		1.5	mA
	Stand-by	40	Null	50	uA
Collecting period	Second		2		Second

8. Attentions of application:

(1) Operating and storage conditions

We don't recommend the applying RH-range beyond the range stated in this specification. The AM2302 sensor can recover after working in abnormal operating condition to calibrated status, but will accelerate sensors' aging.

(2) Attentions to chemical materials

Vapor from chemical materials may interfere AM2302's sensitive-elements and debase AM2302's sensitivity.

(3) Disposal when (1) & (2) happens

Step one: Keep the AM2302 sensor at condition of Temperature 50~60Celsius, humidity <10%RH for 2 hours;

Step two: After step one, keep the AM2302 sensor at condition of Temperature 20~30Celsius, humidity >70%RH for 5 hours.

(4) Attention to temperature's affection

Relative humidity strongly depend on temperature, that is why we use temperature compensation technology to ensure accurate measurement of RH. But it's still be much better to keep the sensor at same temperature when sensing.

AM2302 should be mounted at the place as far as possible from parts that may cause change to temperature.

(5) Attentions to light

Long time exposure to strong light and ultraviolet may debase AM2302's performance.

(6) Attentions to connection wires

The connection wires' quality will effect communication's quality and distance, high quality shielding-wire is recommended.

(7) Other attentions

* Welding temperature should be bellow 260Celsius.

* Avoid using the sensor under dew condition.

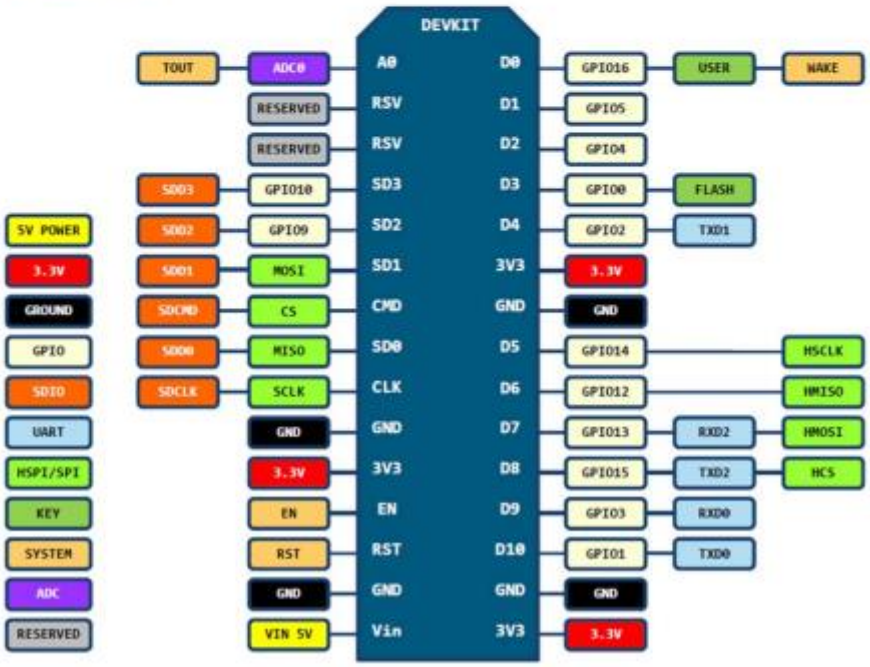
* Don't use this product in safety or emergency stop devices or any other occasion that failure of AM2302 may cause personal injury.

ANEXO II: Especificaciones nodemcu

1. Specification:

- Voltage:3.3V.
- Wi-Fi Direct (P2P), soft-AP.
- Current consumption: 10uA~170mA.
- Flash memory attachable: 16MB max (512K normal).
- Integrated TCP/IP protocol stack.
- Processor: Tensilica L106 32-bit.
- Processor speed: 80~160MHz.
- RAM: 32K + 80K.
- GPIOs: 17 (multiplexed with other functions).
- Analog to Digital: 1 input with 1024 step resolution.
- +19.5dBm output power in 802.11b mode
- 802.11 support: b/g/n.
- Maximum concurrent TCP connections: 5.

2. Pin Definition:



D0(GPIO16) can only be used as gpio read/write, no interrupt supported, no pwm/i2c/ow supported.

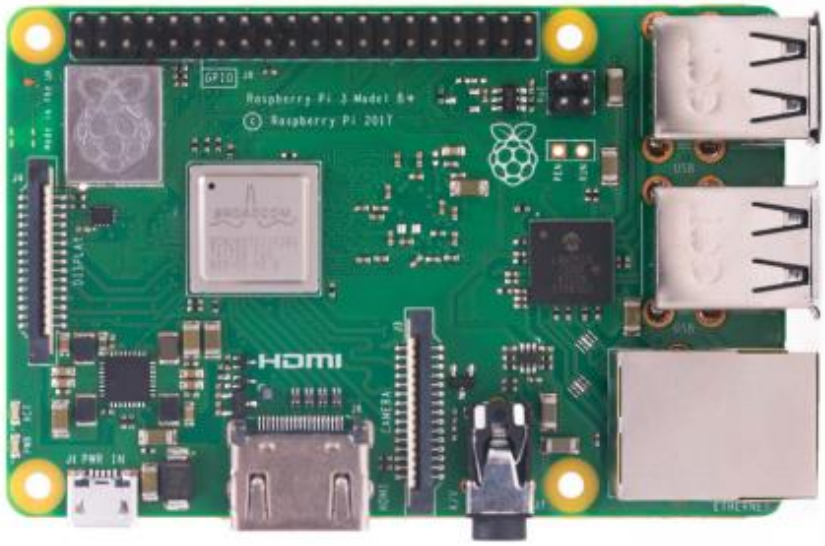
ANEXO III: Especificaciones Raspberry Pi 3+

Raspberry Pi 3 Model B+

Raspberry Pi 3 Model B+

1

Overview



The Raspberry Pi 3 Model B+ is the latest product in the Raspberry Pi 3 range, boasting a 64-bit quad core processor running at 1.4GHz, dual-band 2.4GHz and 5GHz wireless LAN, Bluetooth 4.2/BLE, faster Ethernet, and PoE capability via a separate PoE HAT

The dual-band wireless LAN comes with modular compliance certification, allowing the board to be designed into end products with significantly reduced wireless LAN compliance testing, improving both cost and time to market.

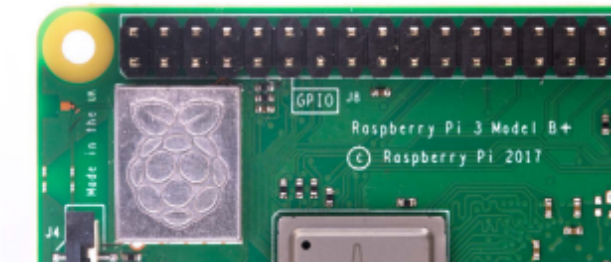
The Raspberry Pi 3 Model B+ maintains the same mechanical footprint as both the Raspberry Pi 2 Model B and the Raspberry Pi 3 Model B.

raspberrypi.org

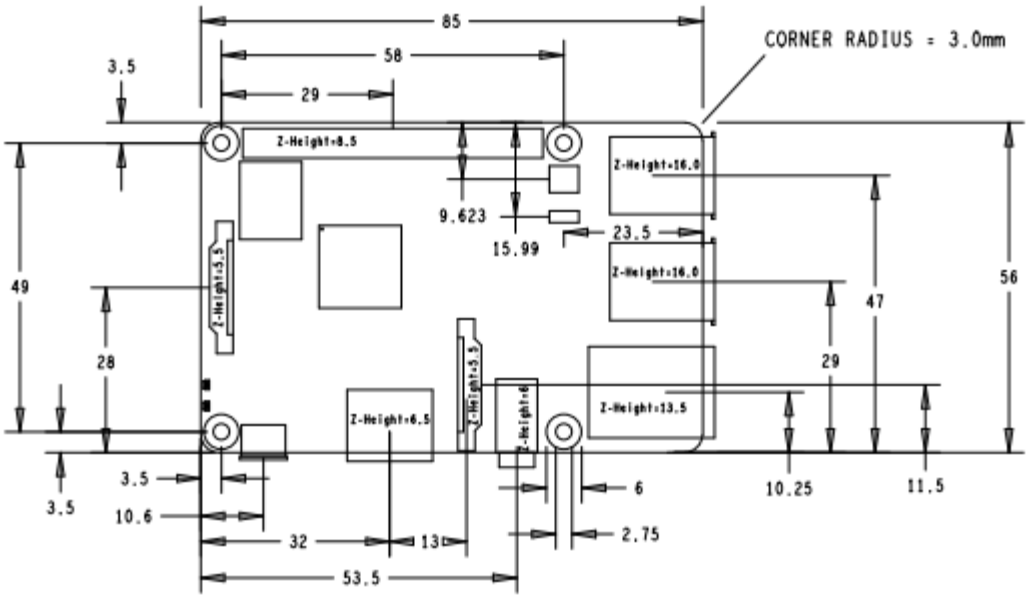
Specifications

Processor:	Broadcom BCM2837B0, Cortex-A53 64-bit SoC @ 1.4GHz
Memory:	1GB LPDDR2 SDRAM
Connectivity:	■ 2.4GHz and 5GHz IEEE 802.11.b/g/n/ac wireless LAN, Bluetooth 4.2, BLE ■ Gigabit Ethernet over USB 2.0 (maximum throughput 300Mbps) ■ 4 × USB 2.0 ports
Access:	Extended 40-pin GPIO header
Video & sound:	■ 1 × full size HDMI ■ MIPI DSI display port ■ MIPI CSI camera port ■ 4 pole stereo output and composite video port
Multimedia:	H.264, MPEG-4 decode (1080p30); H.264 encode (1080p30); OpenGL ES 1.1, 2.0 graphics
SD card support:	Micro SD format for loading operating system and data storage
Input power:	■ 5V/2.5A DC via micro USB connector ■ 5V DC via GPIO header ■ Power over Ethernet (PoE)–enabled (requires separate PoE HAT)
Environment:	Operating temperature, 0–50°C
Compliance:	For a full list of local and regional product approvals, please visit www.raspberrypi.org/products/raspberry-pi-3-model-b+
Production lifetime:	The Raspberry Pi 3 Model B+ will remain in production until at least January 2023.

raspberrypi.org



Physical specifications



Warnings

- This product should only be connected to an external power supply rated at 5V/2.5A DC. Any external power supply used with the Raspberry Pi 3 Model B+ shall comply with relevant regulations and standards applicable in the country of intended use.
- This product should be operated in a well-ventilated environment and, if used inside a case, the case should not be covered.
- Whilst in use, this product should be placed on a stable, flat, non-conductive surface and should not be contacted by conductive items.
- The connection of incompatible devices to the GPIO connection may affect compliance, result in damage to the unit, and invalidate the warranty.
- All peripherals used with this product should comply with relevant standards for the country of use and be marked accordingly to ensure that safety and performance requirements are met. These articles include but are not limited to keyboards, monitors, and mice when used in conjunction with the Raspberry Pi.
- The cables and connectors of all peripherals used with this product must have adequate insulation so that relevant safety requirements are met.

Safety instructions

To avoid malfunction of or damage to this product, please observe the following:

- Do not expose to water or moisture, or place on a conductive surface whilst in operation.
- Do not expose to heat from any source; the Raspberry Pi 3 Model B+ is designed for reliable operation at normal ambient temperatures.
- Do not expose the printed circuit board to high-intensity light sources (e.g. xenon flash or laser) whilst in operation.
- Take care whilst handling to avoid mechanical or electrical damage to the printed circuit board and connectors.
- Whilst it is powered, avoid handling the printed circuit board, or only handle it by the edges to minimise the risk of electrostatic discharge damage.

raspberrypi.org

ANEXO IV: Código Arduino

```
//Librerías para conectarse a la red WiFi y al broker. También la del sensor.
#include <ESP8266WiFi.h>
#include <PubSubClient.h>
#include "DHT.h"

//Datos de la red WiFi
const char* ssid = "MOVISTAR_1FAD";
const char* password = "B95288C2558E1C1E1EE5";

//Datos del servidor broker mqtt (mosquitto)
const char* mqtt_server = "192.168.1.56"; //IP de la raspberry donde está el broker
const char* topic="room5/temperature"; //Topic al cual se envían los datos de temperatura
const char* topic2="room5/humidity";


WiFiClient espCliente;
PubSubClient client(espCliente);

//Para el sensor2

//Defino los pines que se van a usar del dispositivo nodemcu
//En el pin 4 se reciben los datos
#define DHTTYPE DHT22
uint8_t DHTPin = D4;

DHT dht(DHTPin, DHTTYPE);


void setup() {
  pinMode(DHTPin, INPUT); //Se indica que el pin 4 es de entrada
  Serial.begin(115200);   //Se define la velocidad de comunicacion bits/s

  dht.begin();

  setup_wifi(); //Llamada a la función para conectarse a WiFi

  client.setServer(mqtt_server,1883); //Se define la dirección y puerto del broker

}

```

```
void setup_wifi(){
  WiFi.begin(ssid,password);  //se indica a qué WiFi conectarse
  while (WiFi.status() != WL_CONNECTED){    //Hasta que no se conecte no se abandona esta función
    delay(500);
  }
}

void reconnect() {  //Función para reconectarse al broker en caso de perder conexión.
  while (!client.connected()){
    Serial.println("Desconectado");
    if (client.connect("ESP8266Client")){
      Serial.println("Conectado");

    }
    delay(100);
  }
}

void publish(){ //Función para publicar los datos.

  float hum = dht.readHumidity(); //"Mide la humedad"
  char* humedad="";
  dtostrf(hum,10,3, humedad); //Se convierte el valor de la humedad en una cadena de ancho 10 y 3 dígitos decimales.
  client.publish(topic2, humedad); //Publica el valor de la humedad
  delay(200);
  float temp = dht.readTemperature(); //Mide la temperatura
  char* temperatura="";
  dtostrf(temp,10,3, temperatura);  //Se convierte el valor de la temperatura en una cadena de ancho 10 y 3 dígitos decimales.
  client.publish(topic, temperatura);  //Publica el valor de la temperatura
}

void loop() {
  if (!client.connected()){ //Se ejecuta si no hay conexión con el broker
    reconnect();
  }
  else {
    publish();
    delay(59600);   //Se van a tomar datos de la temperatura cada 20 segundos
  }

}
```

ANEXO V: Código fichero index.php

```
<HTML>
<HEAD>
	<TITLE>RASPBERRY</TITLE>

</HEAD>
<BODY>

<link rel="stylesheet" type="text/css" href="estilos.css">

<?php
session_start();
if (!isset($_SESSION['user_id'])){

	require("password.php");
	exit;
}
else{

//Se crea un botón de Inicio que al pulsar se vuelve al index.
echo "<FORM ACTION=$_SERVER[PHP_SELF] METHOD=GET>";
echo "<rec><INPUT type='button' value='Inicio' onClick=location.href='index.php'>";

//Se crea el selector de sensor.
echo "<select name='Sel.sensor'>
			<option>sensor1</option>
			<option>sensor2</option>
			<option>sensor3</option>
			<option>sensor4</option>
			<option>sensor5</option>
	</select>";
//Se crean 2 botones rápidos para ver la última hora o las últimas 24h.

echo "<INPUT TYPE=submit name='24h' VALUE='Últimas 24h'>";
echo "<INPUT TYPE=submit name='1h' VALUE='Última hora'>";

//Se crean los formularios provisionales para consultar un periodo concreto.
$fe= date("Y-m-d");

echo "<fec> Desde: </fec><INPUT TYPE='date' name='fechai' value='$fe'>";
echo "<INPUT TYPE='time' name='horai' value='00:00'>";
echo "<fec> Hasta: </fec><INPUT TYPE='date' name='fechaf' value='$fe'>";
echo "<INPUT TYPE='time' name='horaf' value='23:59'>";

//Se crea el botón para acceder a  la gráfica.
echo "<INPUT TYPE=submit id=acep name='ver' VALUE='Aceptar'></rec>";
include("funciones.php");
```

```
echo"<br>";

//Se almacenan en variables la información dada
$sensor=$_REQUEST['Sel_sensor'];
$fechai=$_REQUEST['fechai'];
$horai=$_REQUEST['horai'];
$fechaf=$_REQUEST['fechaf'];
$horaf=$_REQUEST['horaf'];

//Se comprueba el periodo de tiempo
if ($_REQUEST['24h']=='Últimas 24h'){
        $periodo=1440;
        $fecha_pred=True;
        require("grafica.php");
}

if ($_REQUEST['1h']=='Última hora'){
        $periodo=60;
        $fecha_pred=True;
        require("grafica.php");
}


if ($_REQUEST['ver']=='Aceptar'){
        require("grafica.php");
}

}

require("resumen.php");
?>

</BODY>
</HEAD>
<FOOTER>

Proyecto realizado por Miguel Felipe para la Escuela de Ingeniería y Arquitectura de la Universidad de Zaragoza.
</FOOTER>
<img id='logo' src='logounizar.png' alt='Problem'>
```

ANEXO VI: Código fichero grafica.php

```
<link rel="stylesheet" type="text/css" href="estilos.css">
<?php
echo "<br>";

if ($fecha_pred){
   $consulta="SELECT t,h,date from $sensor ORDER BY ID DESC LIMIT $periodo";
   //echo "<br>hola";
}
if (!$fecha_pred){
   $ini=$fechai ." ". $horai;
   $fin=$fechaf ." ". $horaf;
   $consulta="SELECT t,h,date from $sensor WHERE date BETWEEN '$ini' AND
'$fin'";
}

$datos= leer_db($consulta);

for($i=0;$i<count($datos);$i++){
        $time=$datos[$i]["date"];
        $date=new DateTime($time);
        $datos[$i]["date"]=$date->getTimestamp()*1000;
}

$last_t=$datos[0]["t"];
$last_h=$datos[0]["h"];


$cons="SELECT * from $sensor ORDER BY ID DESC LIMIT 1";
$res=mysqli_use_result($conexion,$cons);
$last_id=mysqli_fetch_row($res);
$k=$last_id[3];

echo "<br>";
$close = mysqli_close($con);

echo "<table id=actual>
<tr id=fila>
<td id=col1_actual><img src='termometro2.png' alt='Problem'></td>
<td id=col2_actual>$last_t ºC</td></tr>
<tr id=fila><td id=col1_actual><img src='humedad2.png' alt='Problem'></td>
<td id=col2_actual>$last_h%</td></tr>
</tr>
</table>";
?>

<HTML>
```

```
<BODY>

<meta charset="utf-8">

<script src="https://code.jquery.com/jquery.js"></script>
<script src="http://code.highcharts.com/stock/highstock.js"></script>
<script src="http://code.highcharts.com/modules/exporting.js"></script>
<script src="../../code/highcharts.js"></script>
<script src="../../code/modules/exporting.js"></script>
<script src="../../code/modules/export-data.js"></script>
<script src="../../code/modules/accessibility.js"></script>

<div id="container">
</div>

<script type='text/javascript'>
$(function () {
   $(document).ready(function() {
      Highcharts.setOptions({
         global: {
            useUTC: false
         }
      });

      var chart;
      $('#container').highcharts({
         chart: {
            type: 'spline',
            animation: Highcharts.svg,
            marginRight: 10,
         },
         title: {
            text: '<?php echo $sensor?>'
         },
         xAxis: {
            type: 'datetime',
            tickPixelInterval: 150
         },
         yAxis: [{ // Eje Y, 1
                  softMax:25,
                  softMin:21,
                  labels: {
                     format: '{value}°C',
                     style: {
                           color: Highcharts.getOptions().colors[0]
                     }
               },
               title: {
                     text: 'Temperature',
                     style: {
```

```
                color: Highcharts.getOptions().colors[0]
            }
        }
    }, { // EjeY, 2
            softMax:60,
            softMin:40,
            title: {
                text: 'Humedad',
                style: {
                    color: Highcharts.getOptions().colors[1]
            }
        },
                labels: {
                    format: '{value} %',
                    style: {
                        color: Highcharts.getOptions().colors[1]
            }
        },

    }],
        tooltip: {
            formatter: function() {
                return '<b>'+ this.series.name +'</b><br/>'+
                Highcharts.dateFormat('%Y-%m-%d %H:%M:%S', this.x) +'<br/>'+
                Highcharts.numberFormat(this.y, 2);
            }
        },
        legend: {
            enabled: true
        },
            credits: {
                enabled: false
            },
        exporting: {
            enabled: true
        },
        series: [{

            name: 'Temperatura',
                yAxis: 0,
            data: (function() {
              var data = [];
               <?php
                  for($i = 0 ;$i<count($datos);$i++){
               ?>
               data.push([<?php echo $datos[$i]["date"];?>,<?php echo
$datos[$i]["t"];?>]);
               <?php }
               ?>
```

```
            return data;
            })()
                        },

                        {

            name: 'Humedad',
                  yAxis:1,
            data: (function() {
              var data = [];
               <?php
                  for($i = 0 ;$i<count($datos);$i++){
               ?>
               data.push([<?php echo $datos[$i]["date"];?>,<?php echo
$datos[$i]["h"];?>]);
               <?php }
               ?>
            return data;
            })()

          }]
       });
    });

});

</script>
</html>
```

ANEXO VII: Código resumen.php

```
<link rel="stylesheet" type="text/css" href="estilos.css">
<?php

$consulta="SELECT t,h from sensor1 ORDER BY ID DESC LIMIT 1";
$conexion=conexion_db();
$resultado=mysqli_query($conexion,$consulta);
$last_id=mysqli_fetch_row($resultado);
$t1=$last_id[0];
$h1=$last_id[1];

$consulta="SELECT t,h from sensor2 ORDER BY ID DESC LIMIT 1";
$conexion=conexion_db();
$resultado=mysqli_query($conexion,$consulta);
$last_id=mysqli_fetch_row($resultado);
$t2=$last_id[0];
$h2=$last_id[1];

$consulta="SELECT t,h from sensor3 ORDER BY ID DESC LIMIT 1";
$conexion=conexion_db();
$resultado=mysqli_query($conexion,$consulta);
$last_id=mysqli_fetch_row($resultado);
$t3=$last_id[0];
$h3=$last_id[1];

$consulta="SELECT t,h from sensor4 ORDER BY ID DESC LIMIT 1";
$conexion=conexion_db();
$resultado=mysqli_query($conexion,$consulta);
$last_id=mysqli_fetch_row($resultado);
$t4=$last_id[0];
$h4=$last_id[1];

$consulta="SELECT t,h from sensor5 ORDER BY ID DESC LIMIT 1";
$conexion=conexion_db();
$resultado=mysqli_query($conexion,$consulta);
$last_id=mysqli_fetch_row($resultado);
$t5=$last_id[0];
$h5=$last_id[1];

$a=20.5;
$b=28;

desconexion_db($conexion);
echo "<table id=resumen>
        <tr>
        <th";?> <?php if ($a<21){ echo "id=cabf";} elseif ($t1>27){ echo "id=cabr";}
else {echo "id=cab";} ?> <?php echo">Estancia1</th>
```

```
        <th";?> <?php if ($t2<21){ echo "id=cabf";} elseif ($t2>27){ echo "id=cabr";}
else {echo "id=cab";} ?> <?php echo">Estancia2</th>
        <th";?> <?php if ($b<21){ echo "id=cabf";} elseif ($b>27){ echo "id=cabr";}
else {echo "id=cab";} ?> <?php echo">Estancia3</th>
        <th";?> <?php if ($t4<21){ echo "id=cabf";} elseif ($t4>27){ echo "id=cabr";}
else {echo "id=cab";} ?> <?php echo">Estancia4</th>
        <th";?> <?php if ($t5<21){ echo "id=cabf";} elseif ($t5>27){ echo "id=cabr";}
else {echo "id=cab";} ?> <?php echo">Estancia5</th>
        </tr>
        <tr>
        <td";?> <?php if ($a<21){ echo "id=frio";} elseif ($t1>27){ echo "id=rojo";}
else {echo "id=sen";} ?> <?php echo">$a ºC</td>
        <td";?> <?php if ($t2<21){ echo "id=frio";} elseif ($t2>27){ echo "id=rojo";}
else {echo "id=sen";} ?> <?php echo">$t2 ºC</td>
        <td";?> <?php if ($t3<21){ echo "id=frio";} elseif ($b>27){ echo "id=rojo";}
else {echo "id=sen";} ?> <?php echo">$b ºC</td>
        <td";?> <?php if ($t4<21){ echo "id=frio";} elseif ($t4>27){ echo "id=rojo";}
else {echo "id=sen";} ?> <?php echo">$t4 ºC</td>
        <td";?> <?php if ($t5<21){ echo "id=frio";} elseif ($t5>27){ echo "id=rojo";}
else {echo "id=sen";} ?> <?php echo">$t5 ºC</td>
        </tr>
</table>"
?>
```

ANEXO VIII: Código funciones.php

```
<?php

//Conectarse a la Base de datos
function conexion_db(){
	$db_host="localhost";
	$db_nombre="base_tfg";
	$db_usuario="root";
	$db_contra="password";

	$conexion=mysqli_connect($db_host,$db_usuario,$db_contra,$db_nombre);
	if(!$conexion){
		echo ("Fallo de conexion");
	}
	return $conexion;
}

//Desconectarse de la DB
function desconexion_db($conexion){
	$close = mysqli_close($conexion);

   if(!$close){
	echo "aqui";
   echo ("Error en base de datos");
   }
   return $close;
}

//Realizar consulta y almacenar resultados
function leer_db($consulta){
	$conexion=conexion_db();
	$datos=array();
	$i=0;
	$resultado=mysqli_query($conexion,$consulta);
	while($fila=mysqli_fetch_array($resultado)){
		$datos[$i]=$fila;
		$i++;

	}
	desconexion_db($conexion);
	return $datos;
}
?>
```

ANEXO IX: Código estilos.css

```
body{
	background-color:#466475;
}
input[type="button"]{
	background-color: #A7D0D9;
	margin: 0 auto;
	border:0px;
	position:relative;
	left:0px;
	top:0px;
	height:100%;
	width:9%;
	font-size:18;
	padding:0 px;
}
input[type="button"]:hover{
	background-color: gold;
}

input[type="submit"]{
	background-color: #A7D0D9;
	margin: 0 auto;
	border:0px;
	position:relative;
	left:0px;
	height:100%;
	width:9%;
	font-size:18;
	padding:0 px;

}
input[type="submit"]:hover{
	background-color: gold;
}

#acep{
	position:relative;
	float:right;
	right:0px;
	margin-right:0px;
}


input[type="date"]{
	width:9%;
	height:100%;
```

```
        background-color:#A7D0D9;
        border-bottom-style:groove;
        border-color:black;
        border-width:0px 0px 3px 0px;
        margin:0px 0px 3px 0px;
        padding:0px;
        position:relative;
        left:0px;
}

input[type="time"]{
        width:9%;
        height:100%;
        background-color:#A7D0D9;
        border-bottom-style:groove;
        border-color:black;
        border-width:0px 0px 3px 0px;
        margin:0px 0px 3px 0px;
        position:relative;
        left:0px;
}

fec{
        margin: 0px;
        border:0px;
        min-height:100%;
        min-width:9%;
        font-size:18;
}
#fec{
        margin: 0 auto;
        border:0px;
        position:relative;
        left:0px;
        height:100%;
        width:9%;
        font-size:18;
        padding:10px 0px 10px 0px;
}

select{
        background-color:#A7D0D9;
        height:100%;
        width:9%;
        font-size:18px;
        border-style:solid;
        border-right:0px;
        border-bottom:0px;
        border-top:0px;
        border-left:0px;
```

```
        border-color:black;
}

rec{
        position: fixed;
        z-index:1;
        left: 0px;
        right: 0px;
        top: 0px;
        height: 5%;
        padding: 0px;
        border-width: 0px;
        background-color: #A7D0D9;
        border-collapse:collapse;
}

#container{
        width:84%;
        border: 2px solid black;
        position:relative;
        left:0px;
        display:inline-block;
        margin:20px 0px 10px 0px;
        height:400px;
        border-radius:5px;
}
img{
        max-width:100%;
        max-height: 100%;

}
#actual{
        position:relative;
        left:0px;
        margin:20px 0px 10px 0px;
        background-color:white;
        display:inline-block;
        width:15.2%;
        font-size:30;
        border:1px solid black;
        border-spacing:0px 0px;
        border-radius:5px;
}
#col1_actual{
        padding:0px;
        border-width:1px 0px 1px 1px;
        border-style:solid;
        width:50%;
}
#col2_actual{
```

```
        padding:0px;
        background-color:#A7D0D9;
        border-width:1px 1px 1px 0px;
        border-style:solid;
        width:50%;
        font-size:100%;
        text-align:center;
}
#fila{
        height:200px;
        border-collapse:collapse;

}


#resumen{
        width:100%;
        border-spacing:3px 0px;
        margin-top:2%;
        position:relative;
}
#cab{
        font-size:20;
        width:20%;
        border:0px;
        background-color:#A7D0D9;
        border-radius:5px 5px 0px 0px;
        text-align:center;
}

#cabr{
        font-size:20;
        width:20%;
        border:0px;
        background-color:#FA4024;
        border-radius:5px 5px 0px 0px;
        text-align:center;
}

#cabf{
        font-size:20;
        width:20%;
        border:0px;
        background-color:#1D8FF2;
        border-radius:5px 5px 0px 0px;
        text-align:center;
}

#sen{
   position:relative;
```

```
        font-size:40;
        border:0px;
        background-color: #A7D0D9;
        border-radius:0px 0px 5px 5px;
        padding:10px;
        text-align:center;
}
#rojo{
        font-size:40;
        border:0px;
        background-color: #FA4024;
        border-radius:0px 0px 5px 5px;
        padding:10px;
        text-align:center;
}
#frio{
        font-size:40;
        border:0px;
        background-color: #1D8FF2;
        border-radius:0px 0px 5px 5px;
        padding:10px;
        text-align:center;
}
tex{

        left:40%;
        align-items:center;
        border:2px solid black;
        text-align:center;
        width:20%;
        font-size:20;
        background-color:white;


}
#acceso{
        height:20px;
        width:100px;
        margin-top:10px;
        margin-bottom: 10px;
}
footer{
        position:absolute;
        /*text-align:center;*/
        width:75%;
        bottom:0px;
        margin-top:15px;
        font-size:20;
        height:8%;
        /*background-color:green;*/
```

```
        color:white;
        font-size:25;
        padding-top:4%;
}
#logo{
        max-width:250px;
        height:150px;
        position:absolute;
        right:10px;
        bottom:10px;
        padding-bottom:0px;
}
fin{
        color:white;
        position:relative;
        left:0px;
        bottom:50%
}
```

ANEXO X: Código Python

```
import paho.mqtt.subscribe as mqtt
import time
import pymysql

global temp
global hum
global t
global h
hum= False
temp= False

def write_db(t,h):
    #print(t,h)
    miConexion=pymysql.connect(host="localhost", user="root", password="password", database="base_tfg")
    miCursor=miConexion.cursor()
    miCursor.execute("INSERT INTO sensor5(t,h) VALUES(%s,%s)" %(t,h))
    #miCursor.execute("INSERT INTO prueba(h) VALUES(%s)" %h)

    miConexion.commit()
    miConexion.close()

while (1):
    msg= mqtt.simple(["room5/humidity","room5/temperature"], qos=0, msg_count=2, retained=False, hostname="192.168.1.56",
        port=1883, client_id="", keepalive=60, will=None, auth=None, tls=None)
    if msg[0].topic=="room5/humidity":
        hum=float(msg[0].payload)
    else:
        temp=float(msg[0].payload)
    if msg[1].topic=="room5/temperature":
        temp=float(msg[1].payload)
    else:
        hum=float(msg[1].payload)
    write_db(temp, hum)
```