

A dark blue vertical bar on the left side of the page. A blue arrow points to the right from this bar, containing the date.

19-3-2021

# Sistema IoT Cloud para la monitorización y visualización de parámetros ambientales

Several thin, curved lines in dark blue and light grey that sweep upwards from the bottom left corner of the page.

Emilian Scurtu

IES GRAN CAPITAN  
ADMINISTRACION SISTEMAS INFORMATICOS EN RED

## Contenido

1. Descripción del proyecto.....	2
Sensores .....	4
Wemos D1 mini pro .....	4
DHT22 .....	4
MH-Z19B.....	5
2. Conexiones de los sensores.....	6
Conexión sensor CO <sub>2</sub> .....	6
Conexión sensor temperatura y humedad .....	7
Conexión real .....	9
3. Despliegue en la nube de AWS .....	10
4. Instalación de los servicios .....	10
InfluxDB .....	10
Telegraf .....	11
Grafana.....	11
MQTT .....	12
5. Configuración de los servicios .....	13
MQTT .....	13
InfluxDB .....	13
Sistema Alertas.....	14
6. Librerías necesarias .....	14
Sensor de temperatura .....	15
Sensor de CO <sub>2</sub> .....	15
7. Código necesario .....	17
Sensores .....	17
8. Driver necesario .....	19
9. Puertos abiertos .....	20

## 1. Descripción del proyecto

El objetivo del proyecto es implementar una red IoT (Internet of Things) en diferentes aulas del instituto para recoger valores medioambientales como serían temperatura, humedad y nivel de CO2. El despliegue necesario para este proyecto se basa básicamente en el microcontrolador ESP8266 y varios sensores que recogerían valores medioambientales y que enviarán a través de Wifi. Estos datos son procesados por un broker MQTT y enviados para su almacenamiento una base de datos a través de una aplicación llamada Telegraf. Para acceder a la información se usará Grafana, una herramienta que monitoriza y visualiza datos mediante representación con gráficas y tablas, con la finalidad de que la información representada sea más fácil de interpretar.

Un uso apropiado y útil de este proyecto de IoT, por la situación de pandemia actual, es el de medir los niveles de CO2 en las diferentes aulas, y si los niveles se superan en base a unos valores predefinidos, enviar alertas para que se actúe de forma inmediata.

La implementación de un medidor de CO2 (dióxido de carbono) consta de una placa con un microcontrolador ESP8266, a la que están conectados los sensores, el del CO2, temperatura y humedad. Mediante esa placa lo que haremos es configurarla para que recoja los datos de los sensores de una forma periódica, cada minuto. Los sensores son clientes publicadores en un Broker MQTT, que recoge los datos para posteriormente enviarlos al agente Telegraf, y que a su vez inserta los valores recogidos en una base de datos temporal InfluxDB. Por último y mediante los datos de la base de datos temporal lo que se hará es generar unos gráficos mediante Grafana.

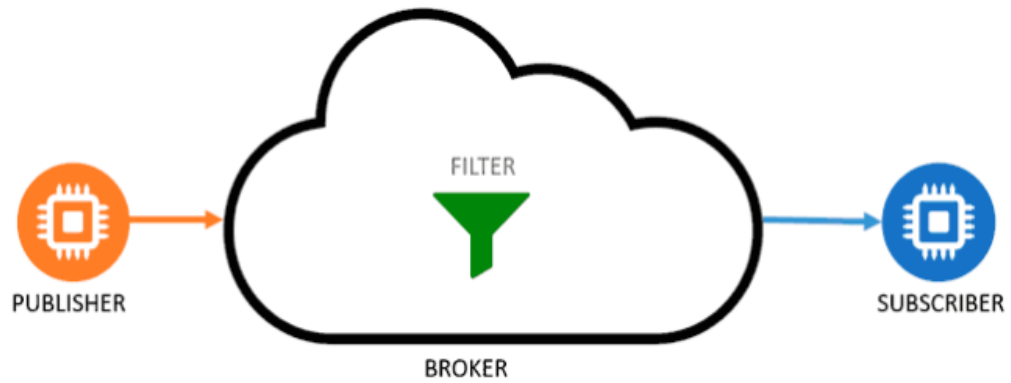
Una vez hecho todo el despliegue y visto que funciona correctamente le añadiremos un sistema de avisos mediante correo electrónico, esto es por ejemplo si se llega a una situación en la que hay demasiado CO2 en el aula, se pueda notificar y hacer el cambio de clase. El cuadro que tenemos en cuenta para posteriormente ventilar o no ventilar la clase sería este.

Niveles de CO2 ppm	400-600	600-800	800-1000	1000-
Ventilación	EXCELENTE	MUY BUENA	ACEPTABLE	MALA VENTILACIÓN (*)

Este cuadro lo que indica es el rango que hay entre los niveles de CO2 que se miden en partes por millón (ppm) y dependiendo en que rango estén los niveles de CO2 se enviara un alerta para que se diga que ese aula no es seguro y que se tiene que ventilar.

Vamos a explicar en qué consiste cada servicio que hemos mencionado anteriormente.

- MQTT: Proviene de las siglas MQ Telemetry Transport, es un protocolo de comunicación M2M (Machine to machine), en lo que se basa este servicio es en un bróker en el que se ponen todos los datos y hay una aplicación que trabaja como suscriptor de ese bróker y lo que hace es recoger esa información, en nuestro caso el suscriptor es Telegraf.



*Imagen 1.1*

- **Telegraf:** Telegraf es un agente de servicio que funciona como suscriptor del bróker que creamos, su función es recoger los datos y guardarlos en InfluxDB.
- **InfluxDB:** InfluxDB es una base de datos temporal en la cual recibirá los datos de Telegraf y los almacenará como máximo una hora, si hemos dicho que cada minuto se recogerá información del sensor lo que almacenará serán unos 60 datos de temperatura, humedad y CO<sub>2</sub>.
- **Grafana:** esta herramienta se encarga de crear gráficos a partir de los datos que estarán introducidos en InfluxDB, las gráficas estarán en constante actualización, ya que cada minuto recibirán un nuevo dato para que así se puedan actualizar y toda la información que este en las gráficas sea a tiempo real.

El esquema que queremos seguir está representado en la foto de abajo.

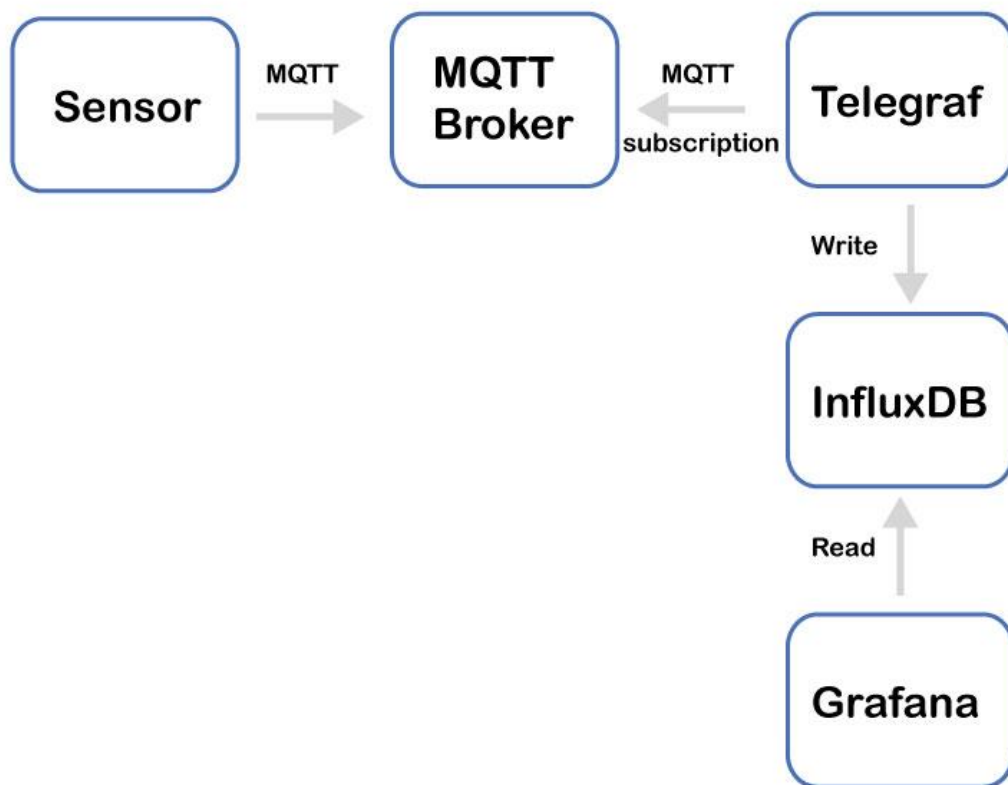


Imagen 1.2

### Sensores

En cuanto a los sensores vamos a necesitar 2 modelos de sensores y una placa que funciona de una forma similar a Arduino.

#### Wemos D1 mini pro

La placa que necesitamos para el proyecto es la Wemos D1 mini pro de 16 mb de memoria con el chip ESP8266, esta placa es una de las mejores que hay por ese rango de precios, entre 4 y 5 euros. Además con esta placa han trabajado unos compañeros míos realizando un proyecto y ellos también se decantaron por ella. Hemos elegido esta placa del fabricante Wemos frente a otros fabricantes como sería Olimex u Adafruit, en el caso de Olimex no posee ningún puerto USB y en cuanto a Adafruit sería una placa con más características de las que necesitamos.

En cuanto a las ventajas que tiene es que posee un puerto USB mediante el cual nos conectamos a ella para introducirle las librerías y demás información, tiene un botón de reset y la más importante de todas tiene 16 mb de memoria.

#### DHT22

El DHT22 es un sensor que se utiliza para hacer mediciones de temperatura y de humedad, este sensor es del fabricante Adafruit. La ventaja más clara que he tenido en cuenta para decantarme por este y no por otro sensor es que este es digital, a diferencia de otros sensores como el LM35, este utiliza un pin digital mediante el cual se envía la información, esto hace que haya menos ruido, es decir que la información que recibamos sea más acorde con la realidad.

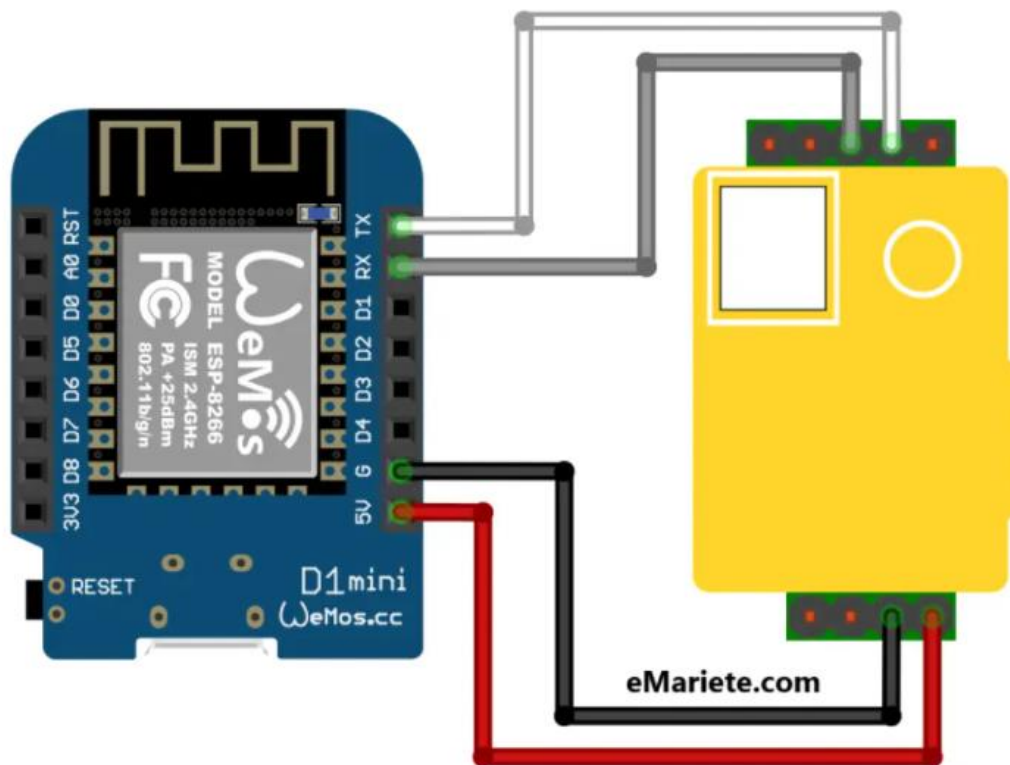
#### MH-Z19B

El sensor de vamos a utilizar es el MH-Z19B, este sensor es de los mejores que hay calidad precio, hay uno que es un poco más caro el Senseair s8, este es un poco mejor que este sensor, pero por falta de tiempo a la hora de pedir los componentes, nos hemos decantando por el MH-Z19B que también es uno de los mejores sensores que hay actualmente en el mercado.

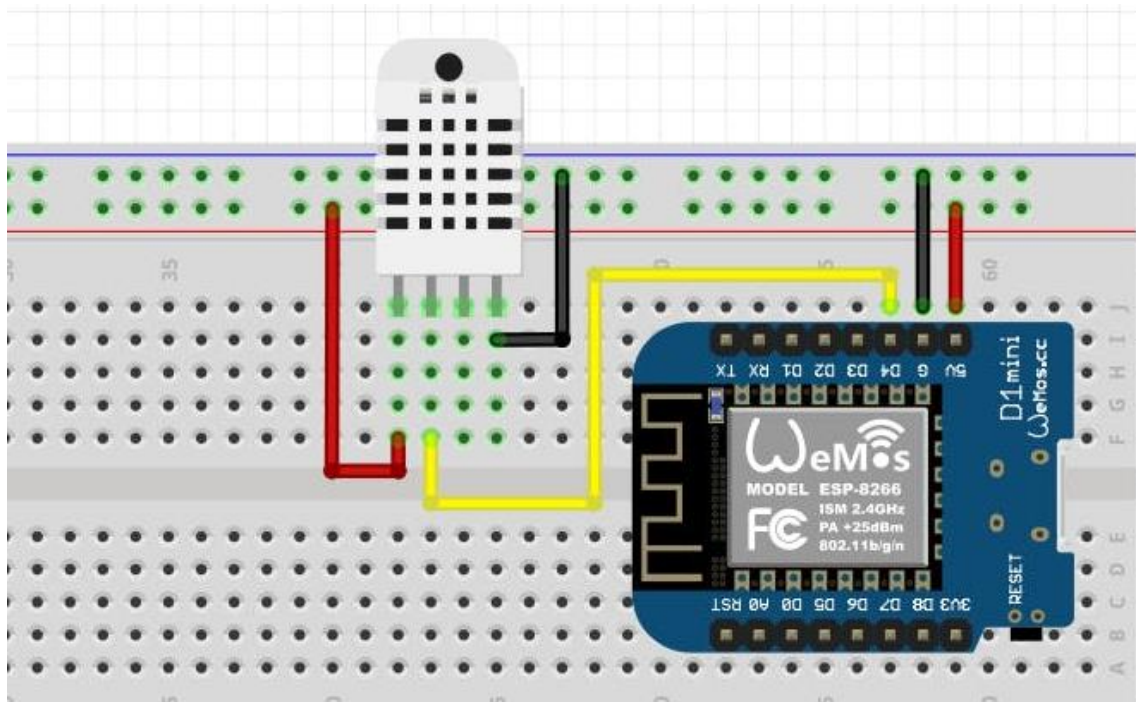
## 2. Conexiones de los sensores

Para comenzar el proyecto tenemos que conectar ambos sensores con el correspondiente patillaje para así se puedan transmitir los datos y puedan funcionar. En cuanto a los sensores, ambos tienen una patilla conectada al negativo que sería el jumper negro y otra a los 5V que sería los positivos y otra que sería el de los datos.

### Conexión sensor CO<sub>2</sub>

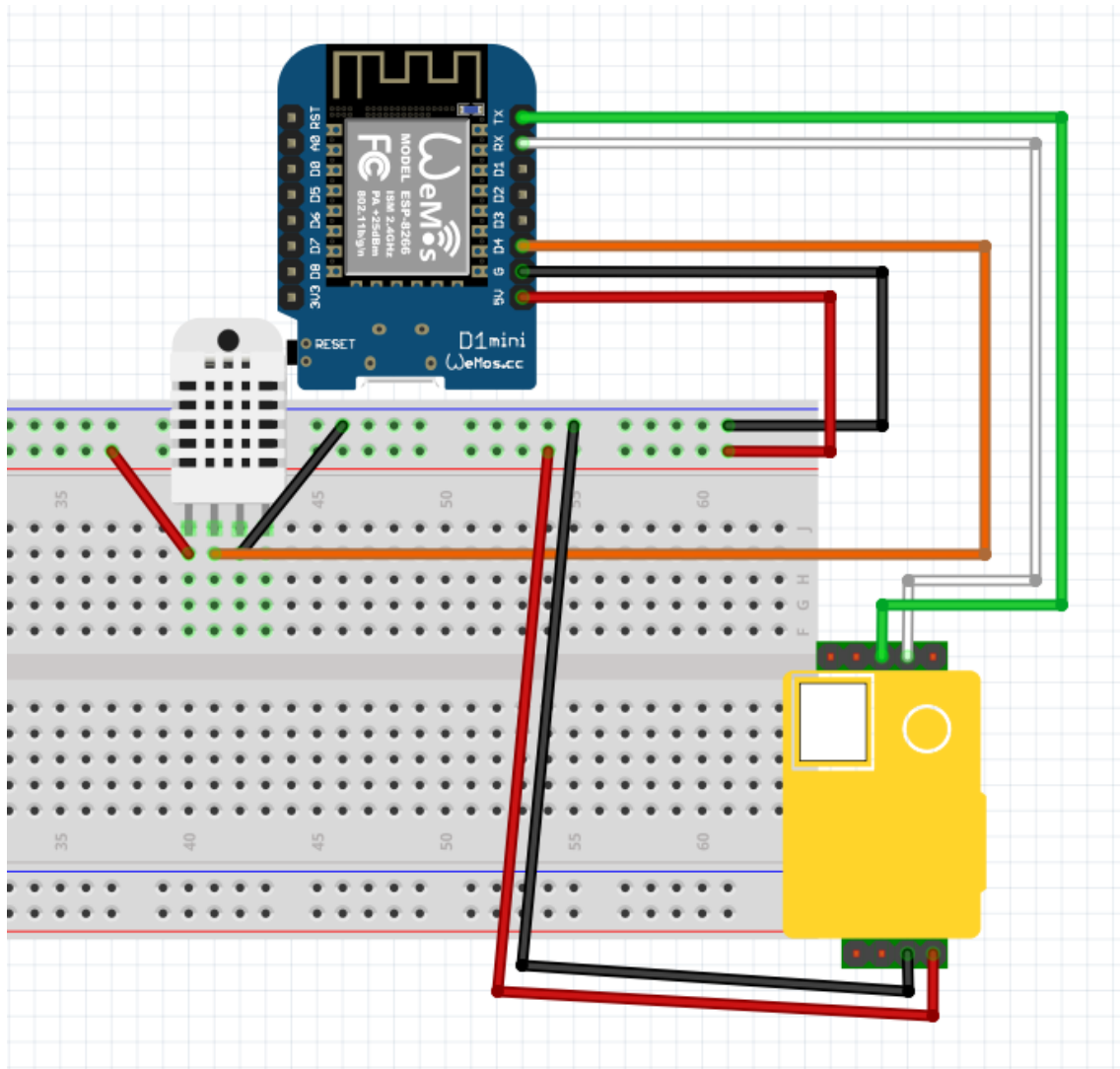


## Conexión sensor temperatura y humedad



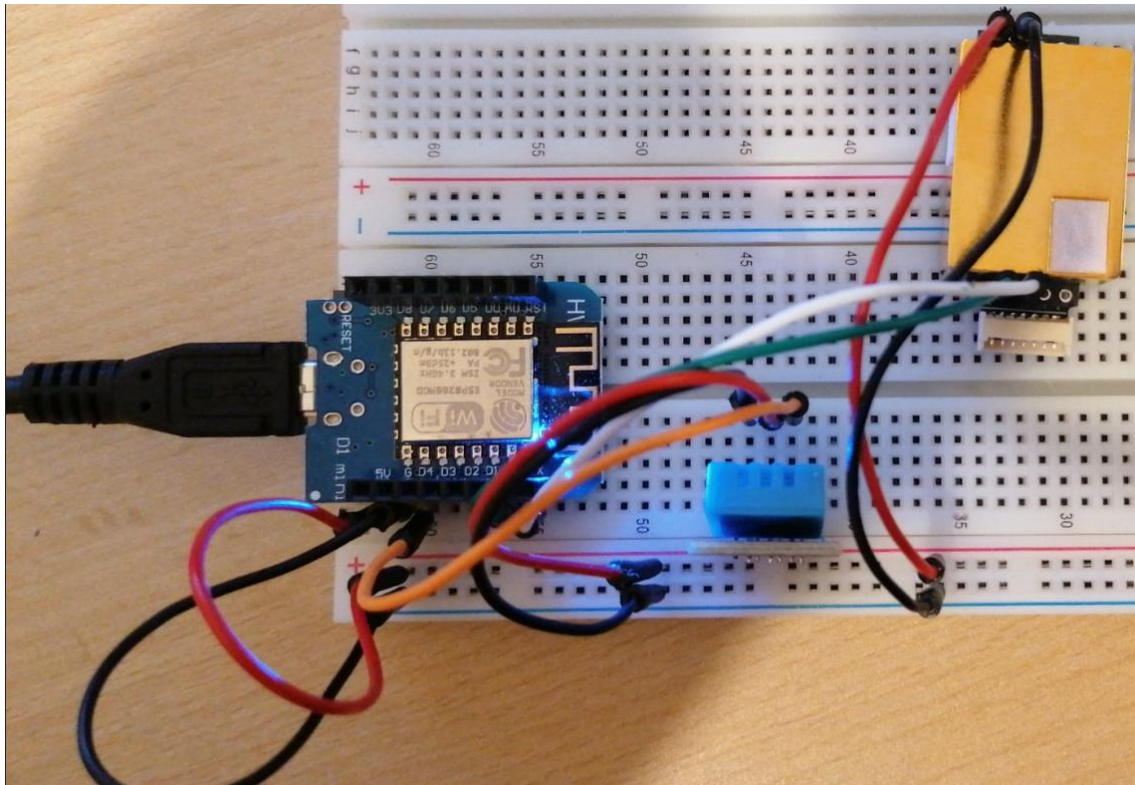
Esta es la conexión de datos para el sensor de la humedad, mi sensor tenía tres patillas pero al buscar el sensor en concreto se puede ver cuál es la patilla negativa, la positiva y la de control de datos.





Este sería el esquema de ambos componentes conectadas, tanto el sensor de temperatura, como el sensor de CO<sub>2</sub>.

### Conexión real

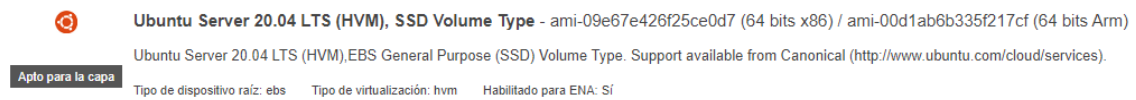


Esta sería la conexión real con los jumpers (los jumpers son los cables que unen los pines de la protoboard con los sensores y la placa).

### 3. Despliegue en la nube de AWS

La finalidad de este proyecto se hace en base a una prueba de concepto, con la intención de proponer una idea que pueda resultar aplicable en base a la situación de pandemia actual, y que si se considera de utilidad se ponga en producción.

Para desarrollar hemos usado los servicios de AWS Educate y hemos realizado el despliegue de las tecnologías: MQTT, INFLUXDB, TELEGRAF y GRAFANA en una máquina EC2 con Ubuntu 20.04 en la nube (Cloud).



Esta es la Ubuntu que hemos utilizado, su instalación no es muy compleja, ya que únicamente lo que hemos hecho ha sido darle los parámetros por defecto, y hemos descargado el par de claves. Una vez instalada lo que hemos hecho ha sido asignarle una IP elástica para que así siga siempre con la misma IP y no se quite cada vez que apaguemos la máquina. En cuanto a los puertos necesarios, los comentare más abajo, hay un solo apartado dedicado a eso.

### 4. Instalación de los servicios

#### InfluxDB

Descargamos el repositorio de InfluxDB

```
emilian@emilian-VirtualBox:~/Escritorio$ wget -qO- https://repos.influxdata.com/influxdb.key | sudo apt-key add -  
[sudo] contraseña para emilian:  
OK
```

Escribimos una línea de comando en un archivo y hacemos un update.

```
emilian@emilian-VirtualBox:~/Escritorio$ source /etc/lsb-release  
emilian@emilian-VirtualBox:~/Escritorio$ echo "deb https://repos.influxdata.com/${DISTRIB_ID,,}${DISTRIB_CODENAME} stable" | sudo tee /etc/apt/sources.list.d/influxdb.list  
deb https://repos.influxdata.com/ubuntu focal stable  
emilian@emilian-VirtualBox:~/Escritorio$ sudo apt-get update
```

Tras eso ya podemos descargar InfluxDB, iniciarlo y ver si está ejecutado.

```
emilian@emilian-VirtualBox:~/Escritorio$ sudo apt-get install influxdb
```

Una vez instalado voy a iniciarlo y comprobar el estado.

```
emilian@emilian-VirtualBox:~/Escritorio$ sudo service influxdb start  
emilian@emilian-VirtualBox:~/Escritorio$ sudo systemctl status influxdb  
● influxdb.service - InfluxDB is an open-source, distributed, time series database  
   Loaded: loaded (/lib/systemd/system/influxdb.service; enabled; vendor preset: enabled)  
   Active: active (running) since Thu 2021-04-15 13:29:04 CEST; 7s ago  
     Docs: https://docs.influxdata.com/influxdb/  
   Main PID: 5889 (influxd)  
     Tasks: 8 (limit: 1108)  
    Memory: 39.6M  
   CGroup: /system.slice/influxdb.service  
           └─5889 /usr/bin/influxd -config /etc/influxdb/influxdb.conf
```

Una vez hecho esto vamos a crear un usuario para más tarde. Pero antes de hacer el próximo comando tenemos que instalar curl con un simple apt install curl.

```
emilian@emilian-VirtualBox:~/Escritorio$ curl -XPOST "http://localhost:8086/query" --data-urlencode "q=CREATE USER iesgc WITH PASSWORD 'patata' WITH ALL PRIVILEGES" {"results":[{"statement_id":0}]}
```

## Telegraf

Para instalar Telegraf también tenemos que ejecutar solo un comando.

```
emilian@emilian-VirtualBox:~/Escritorio$ sudo apt-get install telegraf
```

Y luego vemos si el estado que tiene es activo.

```
emilian@emilian-VirtualBox:~/Escritorio$ sudo systemctl status telegraf
● telegraf.service - The plugin-driven server agent for reporting metrics into InfluxDB
   Loaded: loaded (/lib/systemd/system/telegraf.service; enabled; vendor preset: enabled)
   Active: active (running) since Thu 2021-04-15 14:02:03 CEST; 1min 8s ago
     Docs: https://github.com/influxdata/telegraf
    Main PID: 7577 (telegraf)
      Tasks: 7 (limit: 1108)
    Memory: 27.9M
    CGroup: /system.slice/telegraf.service
            └─7577 /usr/bin/telegraf -config /etc/telegraf/telegraf.conf -config-directory /etc>
```

## Grafana

Para instalar grafana inicialmente tenemos que ejecutar estos comandos.

```
emilian@emilian-VirtualBox:~/Escritorio$ sudo apt-get install -y apt-transport-https
```

Luego este.

```
emilian@emilian-VirtualBox:~/Escritorio$ sudo apt-get install -y software-properties-common wget
```

Y por último estos.

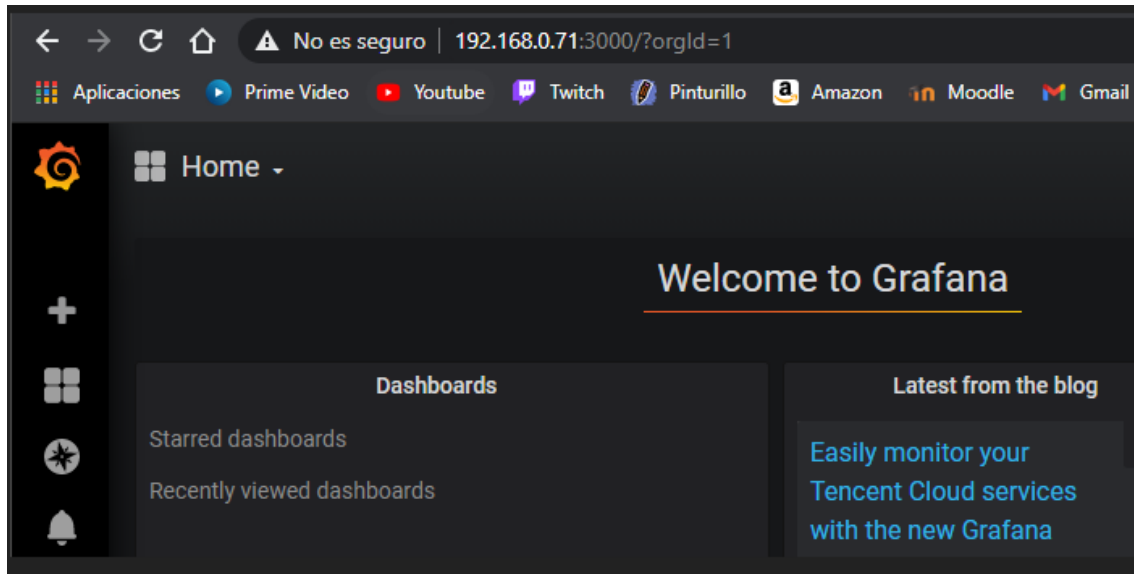
```
emilian@emilian-VirtualBox:~/Escritorio$ wget -q -O - https://packages.grafana.com/gpg.key | sudo apt-key add -
```

```
emilian@emilian-VirtualBox:~/Escritorio$ echo "deb https://packages.grafana.com/oss/deb stable main" | sudo tee -a /etc/apt/sources.list.d/grafana.list
deb https://packages.grafana.com/oss/deb stable main
```

Tras esto ya podemos instalar grafana pero se hace mediante snap.

```
emilian@emilian-VirtualBox:~/Escritorio$ snap install grafana
Se ha instalado grafana 6.7.4 por Alvaro Uría (aluria)
```

Una vez instalado accedemos vía web para ver si funciona.



## MQTT

Para instalar MQTT solo tendremos que ejecutar una serie de comandos.

```
emilian@emilian-VirtualBox:~/Escritorio$ sudo apt-get install mosquitto mosquitto-clients
```

Este es el único comando que tenemos que poner para instalarlo, tras haberse ejecutado el comando habilitamos el servicio.

```
emilian@emilian-VirtualBox:~/Escritorio$ sudo systemctl enable mosquitto.service
```

Y tras esto comprobamos que este ejecutándose.

```
emilian@emilian-VirtualBox:~/Escritorio$ sudo systemctl status mosquitto.service
● mosquitto.service - Mosquitto MQTT v3.1/v3.1.1 Broker
   Loaded: loaded (/lib/systemd/system/mosquitto.service; enabled; vendor preset: enabled)
   Active: active (running) since Thu 2021-04-15 14:15:02 CEST; 3min 1s ago
     Docs: man:mosquitto.conf(5)
           man:mosquitto(8)
   Main PID: 11061 (mosquitto)
      Tasks: 3 (limit: 1108)
     Memory: 1.1M
    CGroup: /system.slice/mosquitto.service
            └─11061 /usr/sbin/mosquitto -c /etc/mosquitto/mosquitto.conf
```

## 5. Configuración de los servicios

### MQTT

```
GNU nano 4.8 /etc/mosquitto/mosquitto.conf
# Place your local configuration in /etc/mosquitto/conf.d/
#
# A full description of the configuration file is at
# /usr/share/doc/mosquitto/examples/mosquitto.conf.example

pid_file /var/run/mosquitto.pid

persistence true
persistence_location /var/lib/mosquitto/




log_dest file /var/log/mosquitto/mosquitto.log

include_dir /etc/mosquitto/conf.d

listener 1883
allow_anonymous true
```

Este es el archivo de configuración de Mosquitto el mosquitto.conf tenemos que poner que escuche mediante el 1883 y allow\_anonymous en true.

Ahora podemos probar para poder suscribirnos.

1883	192.168.0.70 TCP	1883	1883				
------	------------------	------	------	---	---	---	---

Este es el puerto que hemos abierto en el PC, el 1883.

### InfluxDB

InfluxDB no necesita ninguna configuración adicional.

Solo tenemos que crear un archivo telegraf.conf, el antiguo no lo borreís, solo cambiadlo de nombre.

```
ubuntu@ip-172-31-26-155: ~
GNU nano 4.8
[[inputs.mqtt_consumer]]
  servers = ["tcp://localhost:1883"]

  topics = [
    "iesgrancapitan/#"
  ]

  data_format = "value"
  data_type = "float"
[[outputs.influxdb]]
  urls = ["http://localhost:8086"]

  database = "sensores"

  skip_database_creation = true

  username = "PROYECTO"
  password = "1234"
```

Y sustituir algunos parámetros.

### Sistema Alertas

Para el sistema de alertas que integra Grafana solo tenemos que abrir un puerto y configurar el sistema de alertas en el archivo de configuración de grafana.ini, en password hay que poner la contraseña.

```
##### SMTP / Emailing #####
[smtp]
enabled = true
host = smtp.gmail.com:587
user = a19scemem@iesgrancapitan.org
# If the password contains # or ; you have to wrap it with triple quotes. Ex ""#password;""
password =
;cert_file =
;key_file =
;skip_verify = false
from_address = admin@grafana.localhost
from_name = Grafana
# EHLO identity in SMTP dialog (defaults to instance_name)
;ehlo_identity = dashboard.example.com
# SMTP startTLS policy (defaults to 'OpportunisticStartTLS')
;starttls_policy = NoStartTLS
```

## 6. Librerías necesarias

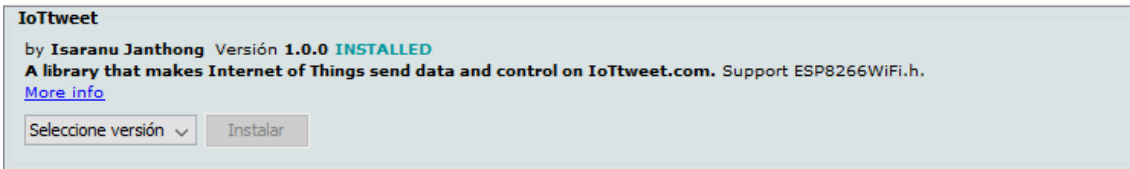
Para poder poner el código necesitaremos las siguientes librerías, las voy a enumerar y adjuntare captura de la aplicación de Arduino para que podáis saber cuál es.

Antes de ver las librerías de los sensores vamos a ver los que están en conjunto en ambos códigos. El primero de todos es el de la propia placa el de ESP8226WiFi y el de MQTT, tanto el client como el que envía.

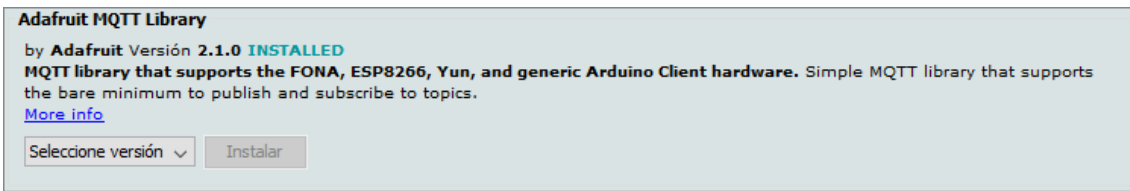


```
#include <DHT.h>
#include <ESP8266WiFi.h>
#include <Adafruit_MQTT.h>
#include <Adafruit_MQTT_Client.h>
```

Los que están en amarillo son los que son comunes en ambos códigos.



El de arriba es la librería que se utiliza para la parte de la placa

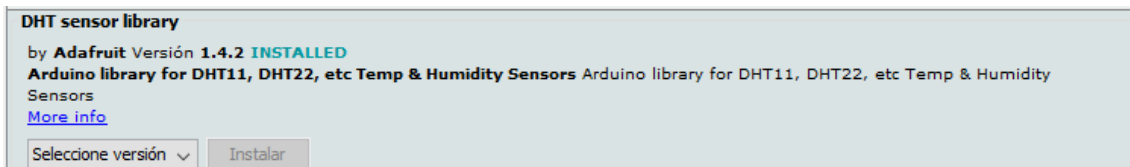


El de abajo es el utilizado para la parte de MQTT.

### Sensor de temperatura

```
#include <DHT.h>
#include <ESP8266WiFi.h>
#include <Adafruit_MQTT.h>
#include <Adafruit_MQTT_Client.h>
```

Esta parte es para la del sensor de temperatura, en cuánto a este solo necesitaremos una librería extra, la de DHT.



Esta es la librería que utilizaremos para el DHT.

### Sensor de CO<sub>2</sub>

```
#include <MHZ.h>
#include <ESP8266WiFi.h>
#include <Adafruit_MQTT.h>
#include <Adafruit_MQTT_Client.h>
// #include <Adafruit_Sensor.h>
#include <SoftwareSerial.h>
```

Para el sensor de CO<sub>2</sub> necesitaremos dos librerías, el del MHZ y el SoftwareSerial.



#### MH-Z CO2 Sensors

by Tobias Schürg Versión 1.2.0 **INSTALLED**

**Ready to use implementation for CO2 sensors of the MHZ series (Intelligent Infrared CO2 Module)** Carbon Dioxide modules such as MH-Z14A, MH-Z19B (and maybe some more) are supported on Arduino / ESP8266. Both output signal modes UART via Serial Port and PWM are supported. See example and/or manual for wiring the mhz14a or mhz19b.

[More info](#)

Esta es la librería del sensor que tenemos, el MH-Z19B.

Y el SoftwareSerial viene incluido también en la librería anterior.

## 7. Código necesario

El código que tendremos que tener es el siguiente.

### Sensores

```
#include <DHT.h>
#include <ESP8266WiFi.h>
#include <Adafruit_MQTT.h>
#include <Adafruit_MQTT_Client.h>

#include <MHZ19.h>
#include <Arduino.h>
#include <SoftwareSerial.h>

#define MH_Z19_RX TX
#define MH_Z19_TX RX
#define DHTPin D2
#define DHTTYPE DHT22
DHT dht(DHTPin, DHTTYPE);

#define WLAN_SSID      "iesgc112"
#define WLAN_PASS      "1234567890"

#define MQTT_SERVER     "ec2-54-88-159-111.compute-1.amazonaws.com"
#define MQTT_SERVERPORT 1883
#define MQTT_USERNAME   ""
#define MQTT_KEY         ""
#define MQTT_FEED_TEMP  "iesgrancapitan/aulal/temperature"
#define MQTT_FEED_HUMI  "iesgrancapitan/aulal/humidity"
#define MQTT_FEED_CO2   "iesgrancapitan/aulal/co2"

MHZ19 myMHZ19;
SoftwareSerial mySerial(MH_Z19_RX, MH_Z19_TX);

WiFiClient client;

Adafruit_MQTT_Client mqtt(&client, MQTT_SERVER, MQTT_SERVERPORT, MQTT_USERNAME, MQTT_USERNAME, MQTT_KEY);

Adafruit_MQTT_Publish temperatureFeed = Adafruit_MQTT_Publish(&mqtt, MQTT_FEED_TEMP);

Adafruit_MQTT_Publish humidityFeed = Adafruit_MQTT_Publish(&mqtt, MQTT_FEED_HUMI);

// Feed to publish CO2
Adafruit_MQTT_Publish co2Feed = Adafruit_MQTT_Publish(&mqtt, MQTT_FEED_CO2);

//-----

void connectWiFi();
```

---

```

void setup() {
  Serial.begin(115200);
  Serial.println("IoT demo");
  dht.begin();
  connectWiFi();
  connectMQTT();

  Serial.println("");
  Serial.println("Init...");

  // Init MH-Z19
  mySerial.begin(9600);
  myMHZ19.begin(mySerial);

  // Show firmware
  char myVersion[4];
  myMHZ19.getVersion(myVersion);

  Serial.print("Firmware Version: ");
  for(byte i = 0; i < 4; i++)
  {
    Serial.print(myVersion[i]);
    if(i == 1)
      Serial.print(".");
  }
  Serial.println("");

  Serial.printf("Max range: %d ppm\n", myMHZ19.getRange());
  Serial.printf("ABC status: ");
  Serial.println(myMHZ19.getABC() ? "ON" : "OFF");
  Serial.println("Setup done.");
}

//-----

void loop() {
  delay(600);

  float h = dht.readHumidity();
  float t = dht.readTemperature();

  if (isnan(h) || isnan(t)) {

```

```

        Serial.println("Fallo en la lectura");
        return;
    }

    Serial.print("Humidity: ");
    Serial.print(h);
    Serial.print(" %\t");
    Serial.print("Temperature: ");
    Serial.print(t);
    Serial.println(" *C ");

    Serial.printf("CO2: %d ppm\n", myMHZ19.getBackgroundCO2());
    delay(5000);

    temperatureFeed.publish(t);
    humidityFeed.publish(h);
    co2Feed.publish(myMHZ19.getBackgroundCO2());
}

//-----

void connectWiFi() {
    WiFi.begin(WLAN_SSID, WLAN_PASS);
    while (WiFi.status() != WL_CONNECTED) {
        delay(500);
        Serial.print(".");
    }

    Serial.println("WiFi connected");
    Serial.println("IP address: ");
    Serial.println(WiFi.localIP());
}

//-----

void connectMQTT() {
    if (mqtt.connected())
        return;

    Serial.print("Connecting to MQTT... ");
    while (mqtt.connect() != 0) {
        Serial.println("Error. Retrying MQTT connection in 5 seconds...");
        mqtt.disconnect();

        delay(5000);
    }
}

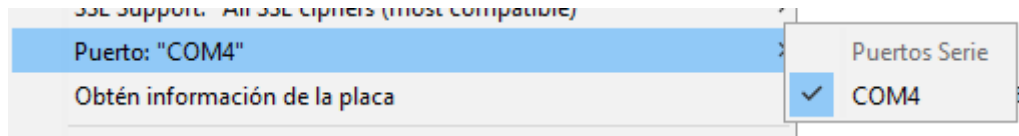
```

## 8. Driver necesario

Una vez tengamos el código tenemos que subir el código y para que se suba el código tenemos que tener una conexión entre la placa y el PC. Para eso, por defecto no viene nada instalado y

por ello tenemos que instalar un driver para que así pueda haber conexión entre ambos dispositivos y que se puedan conectar.

El driver que tenemos que instalar sería el siguiente, el ch340, yo me lo he descargado de un dropbox de una persona y lo he instalado, una vez instalado ya podemos conectarnos placa y PC.



Al terminar de instalar nos tiene que salir este puerto para así poder conectarnos.

## 9. Puertos abiertos

Los puertos que debemos abrir son los siguientes:

Tipo	Protocolo	Intervalo de puertos	Origen
HTTP	TCP	80	0.0.0.0/0
HTTP	TCP	80	::/0
SSH	TCP	22	0.0.0.0/0
SSH	TCP	22	::/0
SMTP	TCP	25	0.0.0.0/0
SMTP	TCP	25	::/0
TCP personalizado	TCP	8086	0.0.0.0/0
TCP personalizado	TCP	8086	::/0
MYSQL/Aurora	TCP	3306	0.0.0.0/0
MYSQL/Aurora	TCP	3306	::/0
TCP personalizado	TCP	3000	0.0.0.0/0
TCP personalizado	TCP	3000	::/0
TCP personalizado	TCP	1883	0.0.0.0/0
TCP personalizado	TCP	1883	::/0
RDP	TCP	3389	0.0.0.0/0
RDP	TCP	3389	::/0
HTTPS	TCP	443	0.0.0.0/0
HTTPS	TCP	443	::/0

Estos son todos los puertos que tenemos que abrir para que nos funcionen todos los servicios, en cuanto al origen lo tenemos para que pueda acceder cualquier PC, ya que para Ipv4 es 0.0.0.0 y para Ipv6 es ::/0.