

Proyecto 2ºASIR

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

Por Méndez y Cuenca.

# Índice

<b>Introducción</b>	3
<b>Hardware</b>	4
Wemos D1 mini pro	5
DHT22	5
MH-Z19B	6
Firmware	9
Instalación	10
Dispositivos	11
Conexión al servidor	16
Herramientas	17
<b>Software (Servidor)</b>	19
Creación	19
Mosquitto	20
Instalación	20
Configuración	20
Pruebas	21
InfluxDB	22
Instalación	22
Configuración	23
Telegraf	23
Datos	25
Grafana	26
Instalación	26
Configuración de gráficos	26
Alertas	28
<b>Bibliografía</b>	34

# Introducción

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 CO<sub>2</sub>. 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 a 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 CO<sub>2</sub> 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 CO<sub>2</sub> (dióxido de carbono) consta de una una placa con un microcontrolador ESP8266, a la que están conectados los sensores, el del CO<sub>2</sub>, 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 CO<sub>2</sub> en el aula, se pueda notificar y hacer el cambio de clase.

# Hardware

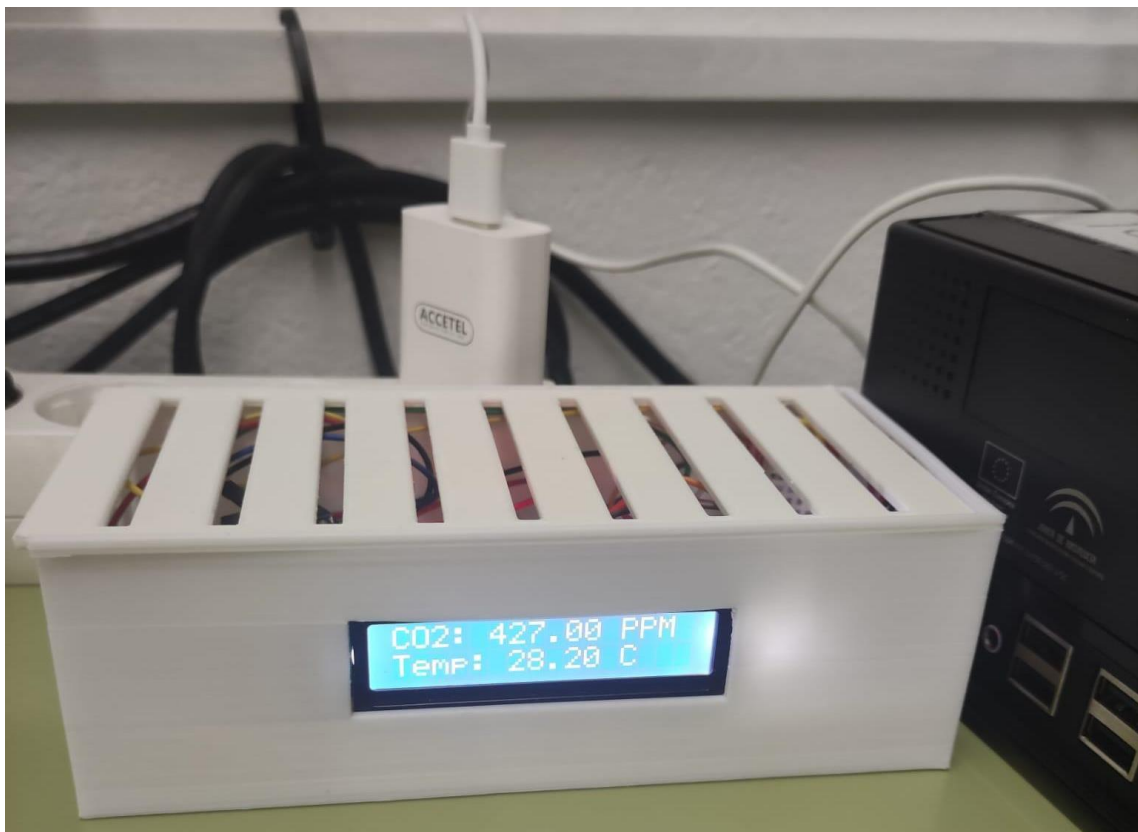
Necesitaremos una placa Wemos d1 mini pro, un sensor DHT22 (temperatura y humedad) y un sensor MHZ19B para CO2.

También añadiremos una pantalla LED para mostrar los valores a tiempo real

Estos componentes irán montados sobre una protoboard y conectado con jumpers a los pines.

Todo va dentro de una caja a medida hecha con impresora 3D cuyo código estará subido a github.

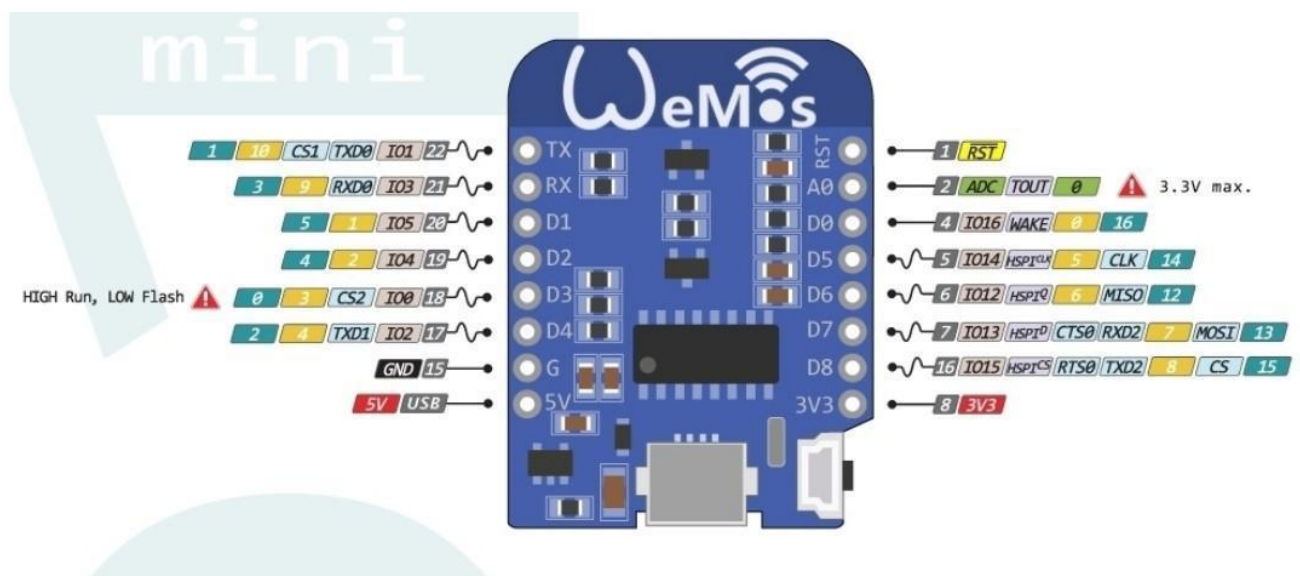
El modelo está hecho con tinkercad y pasado a código G que es el que interpreta la impresora con un programa llamado Slicer.



## 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.

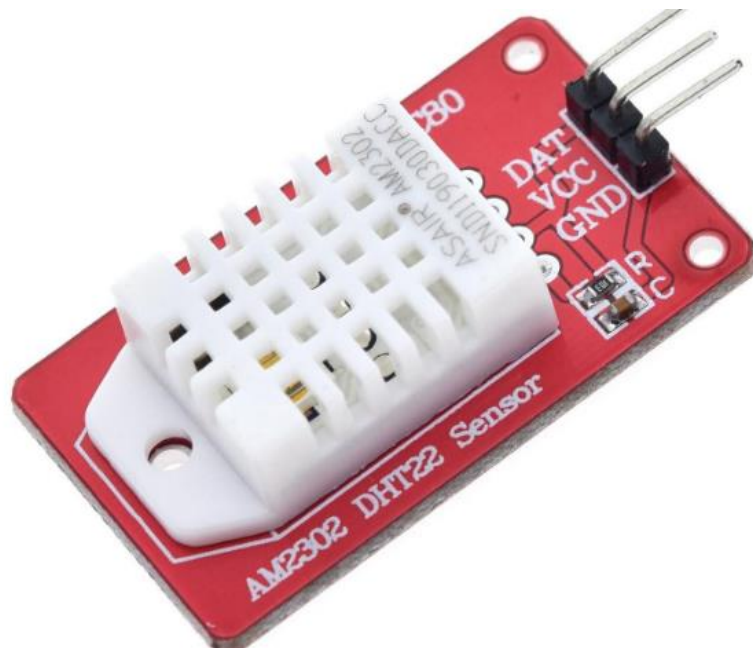
En cuanto a las ventajas que tiene es que posee un puerto USB mediante el cual nos conectamos a ella para introducirle las librerías y demás información, tiene un botón de reset y la más importante de todas tiene 16 mb de memoria.



## DHT22

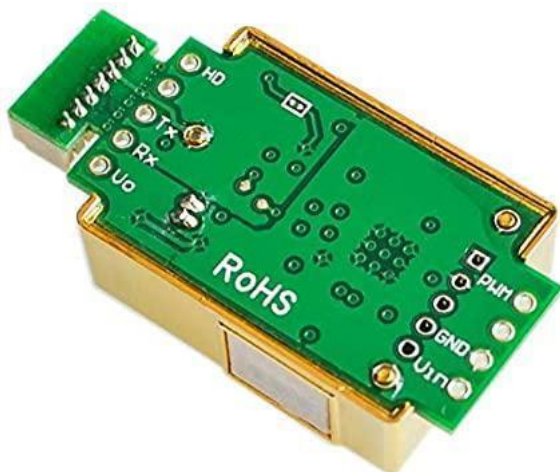
El DHT22 es un sensor que se utiliza para hacer mediciones de temperatura y de humedad, este sensor es del fabricante Adafruit. La ventaja más clara que he tenido en cuenta para decantarme por este y no por otro sensor es que este es digital, a diferencia de otros sensores

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.



## Definimos los pines

### Comunes:

Tanto los sensores como la placa llevan 2 jumpers que van del pin GND a tierra , y del pin de 5V a corriente.

### MHZ19B

Conectamos el pin RX a TX y viceversa

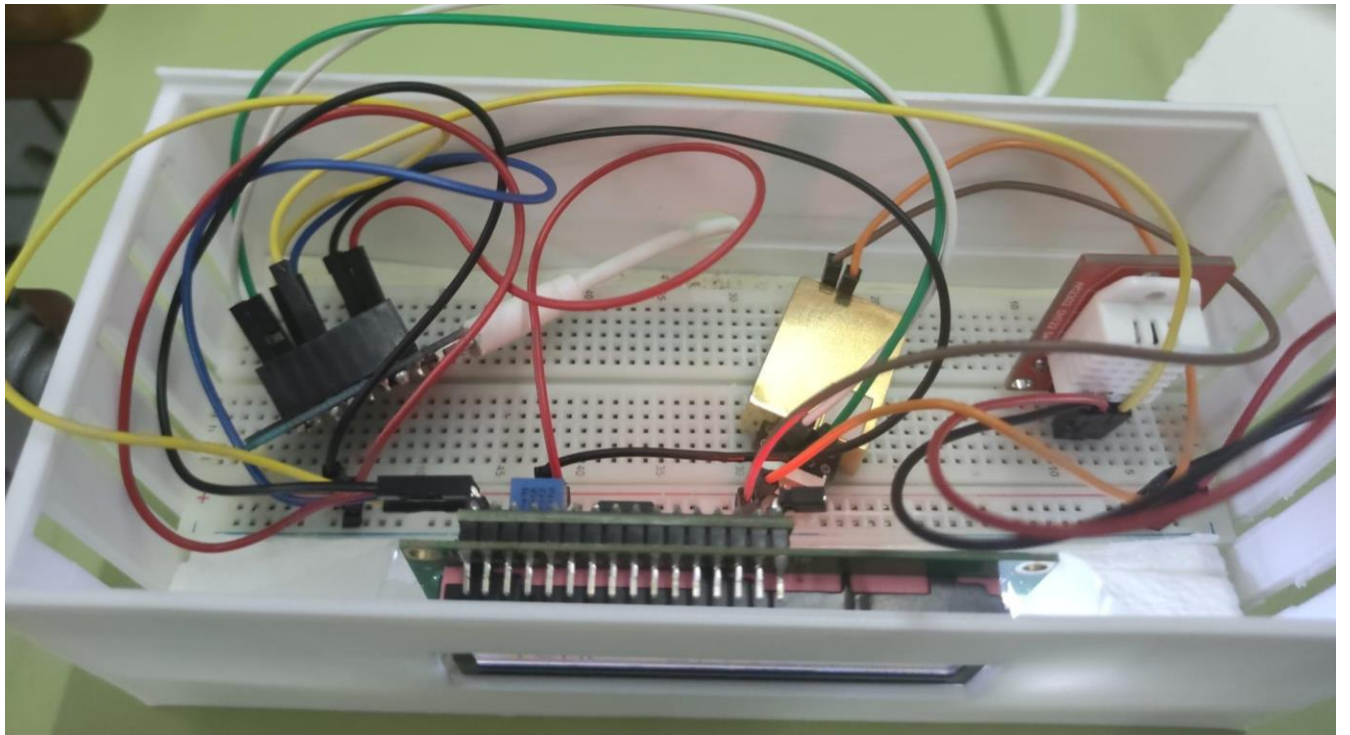
### DHT22

Conectaremos el pin de Datos (dat) al pin D5.

### Pantalla LED:

Lleva el pin SDA conectado a D2 y el pin SCL a D1

\*Excepto los del sensor de CO2 (MHZ19B), es indiferente en cual D estén conectados mientras se especifique en la configuración del firmware.\*





# Firmware

Utilizaremos el firmware ESP-Easy proporcionado por

<https://emariete.com/medidor-casero-co2/>

Lo primero que tienes que hacer es descargar al ordenador el programa que luego cargarás en el Wemos D1 Mini y que administrará todo el funcionamiento.

Hay que conectar la Wemos por USB al ordenador y ejecutar el programa. Esto creará un Punto de acceso Wifi temporal llamado ESP-EASY con la contraseña “configesp”. Hay que conectarse al mismo y nos pedirá una acción en el navegador, en este punto tendremos que configurar el WIFI al que se conectará la Wemos.

Es muy importante desconectar los pines TX t RX del sensor de CO2 al instalar el firmware ya que podría crear conflicto con el USB (desconéctalos temporalmente si ya los conectaste)

ESP Easy Flasher (0.04.007)

COM port: (COM5) USB-SERIAL CH340 (Port\_#0005.Hub\_#0001)

☒ Only active ports

Firmware (.bin): ESP\_Easy\_mega\_20201227\_normal\_ESP8266\_4M1M.bin

Baud rate: 115200

☐ Force -DOUT

☐ Post flash action

☐ Pause after flash (manual reboot)

UNIT: Name, Number, Admin password

WiFi: WiFi SSID (main), WiFi password (main), ☐ Fixed IP, 192, 168, 0, 123

WiFi SSID (fallback), WiFi password (fallback)

☐ Run custom serial commands (Settings\SerialCommands.txt)

Program only.

Rules1: EMPTY FOLDER, Open

Rules2: EMPTY FOLDER, Open

Rules3: EMPTY FOLDER, Open

Rules4: EMPTY FOLDER, Open

2 (USB) COM ports found.  
Creating "C:\Users\jakum\AppData\Local\Temp\Rar\$EXa12624.23187"  
Creating "C:\Users\jakum\AppData\Local\Temp\Rar\$EXa12624.23187"  
Creating "C:\Users\jakum\AppData\Local\Temp\Rar\$EXa12624.23187"

Save as default settings, Open serial monitor, Flash ESP Easy FW

## Instalación

Definimos el wifi y su contraseña y ya entramos en la interfaz del programa. En este punto ya podemos volver a conectar nuestro WIFI y con sólo introducir la IP que nos proporciona esp-easy en el navegador ya podemos acceder a la interfaz de configuración.

**ESP Easy Mega: ESP\_Easy**

[Main](#) [Config](#) [Controllers](#) [Hardware](#) [Devices](#) [Rules](#) [Notifications](#) [Tools](#)

**System Info**

Unit Number:	0
Git Build:	
Local Time:	2022-04-07 14:06:18
Uptime:	0 days 1 hours 8 minutes
Load:	15.65% (LC=3862)
Free RAM:	13296 (5104 - sendContentBlocking)
Free Stack:	3648 (752 - sendContentBlocking)
IP Address:	192.168.112.232
RSSI:	-65 dB (iesgc112)

More info

**Node List** **Name** **Build**

Powered by [Let's Control It community](#)

En la pestaña “Tools” que analizaremos más adelante podemos ver más información de la red.

**ESP Easy Mega: ESP\_Easy**

[Main](#) [Config](#) [Controllers](#) [Hardware](#) [Devices](#) [Notifications](#) [Tools](#)

**WiFi Setup Complete**

**Network** ?

Wifi:	802.11N (RSSI -37 dB)
IP Config:	DHCP
IP / Subnet:	192.168.112.232 / 255.255.255.0
Gateway:	192.168.112.252
Client IP:	192.168.4.100
DNS:	192.168.112.252 / (IP unset)
Allowed IP Range:	(IP unset) - (IP unset)
STA MAC:	2C:F4:32:13:23:83
AP MAC:	2E:F4:32:13:23:83
SSID:	iesgc112 (AC:9E:17:EC:27:C4)
Channel:	1
Connected:	15 s
Last Disconnect Reason:	(1) Unspecified
Number Reconnects:	0

192.168.112.232

Powered by [Let's Control It community](#)

En “Config” podemos cambiar el Wifi, el nombre del dispositivo y la contraseña para acceder al mismo.

The screenshot shows the 'ESP Easy Mega: ESP\_Easy' configuration interface. At the top, there is a navigation bar with tabs: 'Main', 'Config' (selected), 'Controllers', 'Hardware', 'Devices', 'Notifications', and 'Tools'. Below the navigation bar, the 'Main Settings' section is visible, containing fields for 'Unit Name' (ESP\_Easy), 'Unit Number' (0), 'Append Unit Number to hostname' (checked), and 'Admin Password' (masked with dots). Below this, the 'Wifi Settings' section is visible, containing fields for 'SSID' (iesgc112), 'WPA Key' (masked with dots), 'Fallback SSID', 'Fallback WPA Key', and 'WPA AP Mode Key' (masked with dots).

ESP Easy Mega: ESP\_Easy

△Main   ◉Config   ○ Controllers   🚦 Hardware   📶 Devices   📧 Notifications   🛠 Tools

**Main Settings**

Unit Name: ESP\_Easy

Unit Number: 0

Append Unit Number to hostname: ☒

Admin Password: .....

**Wifi Settings**

SSID: iesgc112

WPA Key: .....

Fallback SSID:

Fallback WPA Key:

WPA AP Mode Key: .....

## Dispositivos

Lo primero es indicarle los dispositivos que están conectados, esto lo haremos desde la pestaña “Devices”.

ESP Easy Mega: ESP_Easy								
<div> Main Config Controllers Hardware Devices Notifications Tools </div>								
	Task	Enabled	Device	Name	Port	Ctr (IDX)	GPIO	Values
<a href="#">Edit</a>	1	✓	Gases - CO2 MH-Z19	co2	HW Serial0	❶	RX: GPIO-3 (D9) TX: GPIO-1 (D10)	PPM: 567.00 T: 18.00 U: 0.00
<a href="#">Edit</a>	2	✓	Environment - DHT11/12/22 SONOFF2301/7021	dht		❶	GPIO-14 (D5)	Temperature: 18.20 Humidity: 66.00
<a href="#">Edit</a>	3	✓	Display - LCD2004	pantalla	I2C		SDA: GPIO-4 (D2) SCL: GPIO-5 (D1)	
<a href="#">Add</a>	4							
<a href="#">Add</a>	5							
<a href="#">Add</a>	6							
<a href="#">Add</a>	7							
<a href="#">Add</a>	8							
<a href="#">Add</a>	9							
<a href="#">Add</a>	10							
<a href="#">Add</a>	11							
<a href="#">Add</a>	12							

Powered by [Let's Control It](#) community

Esta es la ventana principal en la que nos muestra los dispositivos que tenemos conectados, podemos editar sus configuraciones o añadir uno nuevo.

Estas son las configuraciones de los 2 sensores.  
Aquí les indicamos los pines a los que se conectan, el nombre visible, que estén habilitados y que manden los datos al controlador(estos últimos los haremos a continuación)

ESP Easy Mega: ESP\_Easy

Main

Config

Controllers

Hardware

Devices

Notifications

Tools

Task Settings

Device:

Gases - CO2 MH-Z19 ? i

Name:

co2

Enabled:

☒

Sensor

Serial Port:

HW Serial0: GPIO-3 (D9) ← TX / GPIO-1 (D10) → RX

Auto Base Calibration:

Normal

Filter:

Skip Unstable

Checksum (pass/fail/reset):

221/0/3

Detected:

MH-Z19B

Data Acquisition

Send to Controller

☒

Interval:

10

[sec]

**ESP Easy Mega: ESP\_Easy**

△Main

⚙️Config

💬Controllers

🔧Hardware

📡Devices

📧Notifications

🔧Tools

**Task Settings**

Device: Environment - DHT11/12/22 SONOFF2301/7021 ? ⓘ

Name: dht

Enabled: ☒

**Sensor**

GPIO ⇄ Data: GPIO-14 (D5) ▾

Sensor model: DHT 22 ▾

**Data Acquisition**

Send to Controller ☒ ⓘ

Interval: 10 [sec]

Para la pantalla indicamos el nombre del dispositivo, el tamaño y el texto que queremos que muestre que almacena en variables.

También hay que indicarle los pines en la pestaña “hardware” en I2C interface.

Es posible que no detecte la pantalla por lo que habría que buscarla desde “Tools” aquí:

**Interfaces**

I2C Scan

Scan for I2C devices

## ESP Easy Mega: ESP\_Easy

△Main   ⊗Config   ◯Controllers   🔴Hardware   📌Devices   📧Notifications   🔧Tools

### Task Settings

Device: Display - LCD2004 ? i

Name: pantalla

Enabled: ☒

### I2C options

I2C Address: 0x27 (39) ▼

Force Slow I2C speed: ☐

### Device settings

Display Size: 2 x 16 ▼

Line 1: CO2: [co2#PPM] PPM

Line 2: Temp: [dht#Temperature] C

Line 3:

Line 4:

Display button: - None - ▼

Display Timeout: 0

## ESP Easy Mega: ESP\_Easy

△Main   ⊗Config   ◯Controllers   🔴Hardware   📌Devices   📧Notifications   🔧Tools

### Wifi Status LED

GPIO → LED: - None - ▼

Inversed LED: ☒  
*Note: Use 'GPIO-2 (D4)' with 'Inversed' checked for onboard LED*

### Reset Pin

GPIO ← Switch: - None - ▼

*Note: Press about 10s for factory reset*

### I2C Interface

GPIO ⇄ SDA: GPIO-4 (D2) ▼

GPIO → SCL: GPIO-5 (D1) ▼

Clock Speed: 400000 [Hz]

*Note: Use 100 kHz for old I2C devices, 400 kHz is max for most.*

Slow device Clock Speed: 100000 [Hz]

Hay que ajustar el Baud rate en advanced settings desde “Tools”

Advanced

Open advanced settings

**ESP Easy Mega: ESP\_Easy**

[Main](#)
[Config](#)
[Controllers](#)
[Hardware](#)
[Devices](#)
[Notifications](#)
[Tools](#)

Syslog Log Level:

Syslog Facility:

Serial Log Level:

Web Log Level:

**Serial Settings**

Enable Serial port: ☐

Baud Rate:

**Inter-ESPEasy Network**

UDP port:

**Special and Experimental Settings**

Webserver port:

*Note: Requires reboot to activate*

Fixed IP Octet:

WD I2C Address:  (decimal)

Ya que estamos aquí aprovechamos para ajustar la zona horaria del dispositivo con la siguiente configuración.

[Main](#)
[Config](#)
[Controllers](#)
[Hardware](#)
[Devices](#)
[Notifications](#)
[Tools](#)

**Advanced Settings**

**Rules Settings**

Rules: ☒

Old Engine: ☒

Tolerant last parameter: ☐

*Note: Perform less strict parsing on last argument of some commands (e.g. publish and sendToHttp)*

SendToHTTP wait for ack: ☐

**NTP Settings**

Use NTP: ☒

NTP Hostname:

**DST Settings**

Last:

Start (week, dow, month):

Mar:

Start (localtime, e.g. 2h→3h):  [hour →]

Last:

End (localtime, e.g. 3h→2h):  [hour →]

DST: ☒

**Location Settings**

Timezone Offset (UTC +):  [minutes]

Latitude:  [°]

Longitude:  [°]

*Note: Longitude and Latitude are used to compute sunrise and sunset*

**Log Settings**

Syslog IP:

Syslog UDP port:

Syslog Log Level:

Syslog Facility:

Serial Log Level:

Web Log Level:

## Conexión al servidor

Ya están los dispositivos funcionando, sólo queda que envíen la información a nuestro servidor.

Vamos a la pestaña de “Controllers” y añadimos uno nuevo.

Tenemos que indicarle el protocolo, en este caso MQTT, la IP en la que está alojado el servidor y el puerto además de las otras opciones que necesitemos.

**ESP Easy Mega: ESP\_Easy**

△Main

⊗Config

○Controllers

✚Hardware

▼Devices

→Rules

☒Notifications

**Controller Settings**

Protocol:

Home Assistant (openHAB) MQTT

?

Locate Controller:

Use IP address

Controller IP:

192.168.12.207

Controller Port:

1883

**Controller Queue**

Minimum Send Interval:

100

[ms]

Max Queue Depth:

10

Max Retries:

10

Full Queue Action:

Ignore New

Check Reply:

Ignore Acknowledgement

Client Timeout:

100

[ms]

**MQTT**

Controller Client ID:

%sysname%\_%unit%

Unique Client ID on Reconnect:

☐

Current Client ID:

ESP\_Easy\_0

Note: Updated on load of this page

Publish Retain Flag:

☐

Controller Subscribe:

%sysname%/#

Controller Publish:

%sysname%/%tskname%/%valname%

Controller LWT Topic:

LWT Connect Message:

LWT Disconnect Message:

Send LWT to broker:

☒

Will Retain:

☒

Clean Session:

☐

Enabled:

☒

Close

Submit

**ESP Easy Mega: ESP\_Easy**

△Main

⊗Config

○Controllers

✚Hardware

▼Devices

→Rules

☒Notifications

🔧Tools

	Nr	Enabled	Protocol	Host	Port
Edit	1	✓	Home Assistant (openHAB) MQTT	192.168.12.207	1883
Add	2				
Add	3				



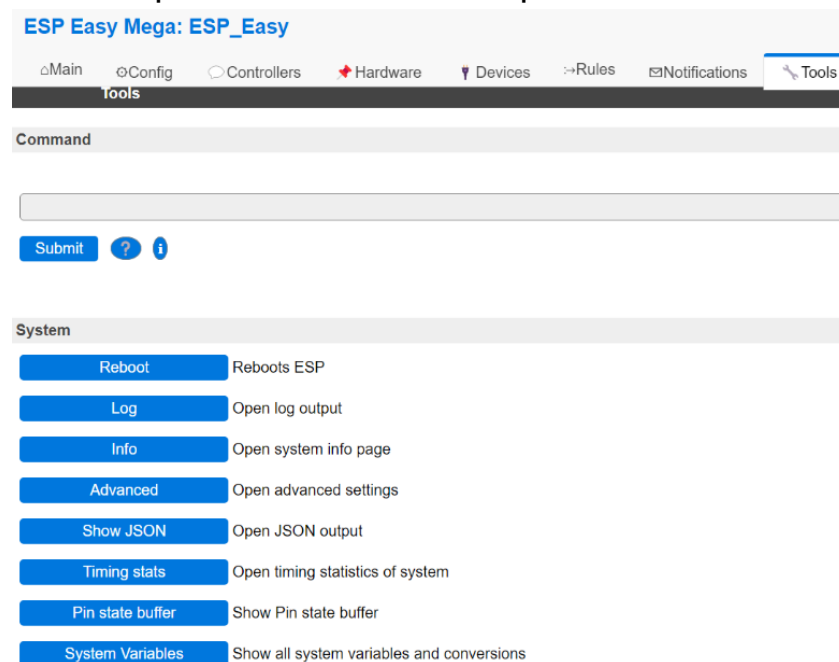
Ya lo tenemos configurado, sólo habría que marcar esta opción en cada uno de los dispositivos que queremos que envíen la información a nuestro servidor.



## Herramientas

Sólo nos queda por ver la pestaña de “Tools”, ésta tiene múltiples opciones:

Entre ellas reiniciar el firmware, ver logs, obtener información, escanear wifi o dispositivos I2C como la pantalla etc.



Wifi

Connect

Connects to known Wifi network

Disconnect

Disconnect from wifi network

Scan

Scan for wifi networks

Interfaces

I2C Scan

Scan for I2C devices

Settings

Load

Loads a settings file

Note: (File MUST be renamed to "config.dat" before upload!)

Save

Saves a settings file

Firmware

Update Firmware

Load a new firmware Max sketch size: 1019 kB (1044464 bytes)

Filesystem

File browser

Show files on internal flash file system

También podemos introducir comandos para administrar nuestro entorno como por ejemplo éste para calibrar el co2.

ESP Easy Mega: [ESP\\_Easy](#)

◐Main◐Config◐Controllers◐Hardware◐Devices◐Rules◐NotificationsTools

Tools

Command

mhzcaltibratezero

Submit

?

i

Toda esta información podemos guardarla y cargarla con las opciones de Load y Save file.

Esto es muy útil para transportar nuestra configuración a otras instalaciones o para replicar la configuración en otro entorno.

# Software (Servidor)

## Creación

Una vez la placa está completada toca la parte del servidor, este entorno está montado sobre el servidor del instituto en una máquina virtual de vmware, es un ubuntu server 20.04 con 2 GB de ram y dos núcleos y 20 GB de almacenamiento, no hace falta más ya que en sí el sistema de monitorización y de gráficos no consume muchos recursos.

Los puertos usados son los siguientes:

- 22. Este puerto es el usado por defecto para conectarte mediante el protocolo ssh.
- 25. Puerto usado por SMTP, usado para enviar emails en grafana.
- 80. Puerto usado por el protocolo HTTP, usado por la página que muestra en tiempo real los datos recogidos de la placa.
- 443. Puerto usado por HTTPS.
- 587. Puerto usado por SMTP, usado para enviar emails en grafana.
- 1883. Puerto usado para recibir datos a través de mosquito.
- 3000. Puerto usado por grafana.
- 3306. Puerto usado para la base de datos.
- 8086. Puerto usado por influxDB.

Una vez terminado esto lanzamos la instancia, luego nos conectamos a ella y lanzamos un “apt update” para actualizar los repositorios.

Ya podemos iniciar la instalación de los programas necesarios.

## Mosquitto

Mosquitto es la herramienta que se encarga de recibir los datos de la placa. Este servicio utiliza el puerto 1883, que lo abrimos anteriormente.

### Instalación

Lo primero es lo primero, por lo que hay que instalarlo con el comando:

```
sudo apt install mosquitto
```

Hará que funcione la parte de recoger datos, también hay que instalar el cliente de mosquitto para mostrar los datos en la página web.

```
sudo apt install mosquitto-clients
```

Ahora hay que configurar para que mosquitto se inicie automáticamente:

```
sudo systemctl enable mosquitto.service
```

Hecho todo esto, el servicio ya estará funcionando, pero hay que tocar un archivo de la configuración para que pueda recibir datos.

### Configuración

El archivo de configuración se encuentra en la carpeta /etc/mosquitto.

Dentro de esta carpeta está el archivo de configuración principal y los certificados que permiten utilizar una conexión segura.

Modificamos el archivo mosquitto.conf.

Hay que añadir las siguientes líneas.

- |                        |  |
|------------------------|--|
| - listener 1883        | Con esto se indica el puerto que utiliza.  |
| - allow_anonymous true | Indicamos que puede recibir datos de cualquiera.                                 |
| - protocol mqtt        | Indicamos el protocolo que usará el puerto. MQTT es el usado para recibir datos. |

Ahora tenemos que reiniciar el servicio con el comando:

```
sudo systemctl restart mosquitto.service
```

## Pruebas

Una vez terminada la configuración podemos comprobar que recibe mensajes.

El comando `mosquitto_sub` podemos suscribirnos a un “topic” el cual mostrará por la consola todos los mensajes recibidos.

`sudo mosquitto_sub -t "iesgrancapitan/112/#"` es el comando que usaremos para mostrar los datos.

```
ubuntu@ip-172-31-12-147:~$ sudo mosquitto_sub -t "iesgrancapitan/112/#"
23.40
43.00
0
23.40
43.60
0
23.40
43.70
0
23.40
44.10
0
23.40
44.40
0
23.40
45.00
0
23.40
45.30
0
23.40
45.40
0
```

También podemos publicar datos manualmente con el comando

`sudo mosquitto_pub -h localhost -t "iesgrancapitan/112/temperatura" -m "20"`

Sería algo así.

Enviamos el mensaje

```
ubuntu@ip-172-31-12-147:~$ sudo mosquitto_pub -h localhost -t "iesgrancapitan/112/temperatura" -m "20"
```

Y con la suscripción vemos el mensaje.

```
ubuntu@ip-172-31-12-147:/etc/mosquitto$ sudo mosquitto_sub -t "iesgrancapitan/112/#"
20
|
```

# InfluxDB

Esta es la base de datos donde se albergarán los datos recogidos de mosquito y luego grafana accederá a estos para el sistema de alertas.

## Instalación

Hay que tener en cuenta una cosa antes de instalar y es instalar la última versión ya que versiones antiguas no disponen de su propia lenguaje de consultas, en versiones anteriores usaba un lenguaje similar a SQL mientras en las últimas versiones puedes usar este lenguaje o su propio lenguaje, que es mucho mejor e intuitivo ya que puedes usar una herramienta similar al phpmyadmin.

Dicho lo anterior procedamos a la instalación.

La instalación se tiene que realizar sobre docker, por lo que hay que instalar docker con los siguientes comandos.

```
sudo apt install docker
```

Y también hay que instalar docker-compose para lanzar imágenes.

```
sudo apt install docker-compose
```

Una vez instalado tenemos que crear un archivo que se llamará docker-compose.yml, dentro de él añadiremos los parámetros necesarios para la instalación de influxdb. Esto es lo que hay que añadir.

```
influxdb:
  image: influxdb:2.1.1
  ports:
    - 8086:8086
  volumes:
    - influxdb_data:/var/lib/influxdb
  environment:
    - INFLUXDB_DB=sensores
    - INFLUXDB_ADMIN_USER=iesgc
    - INFLUXDB_ADMIN_PASSWORD=proyecto
    - INFLUXDB_HTTP_AUTH_ENABLED=true
```

Ahora usamos el comando `sudo docker-compose up -d` que leerá el archivo y procederá a descargar la imagen 2.1.1 de influxdb y utilizará el puerto 8086, además de crear un usuario administrador.

Una vez terminado este proceso tenemos que usar el comando `sudo docker ps` para mostrar todas las imágenes docker y así poder saber el ID de la de influxdb y así poder lanzarla. En mi caso la ID era “37b0f78a6b25”, por lo que para iniciarla hay que usar el siguiente comando:

```
sudo docker-compose up -d
```

Ahora podemos usar este comando para usar la imagen:

```
sudo docker exec -it 37b0f78a6b25 /bin/bash
```

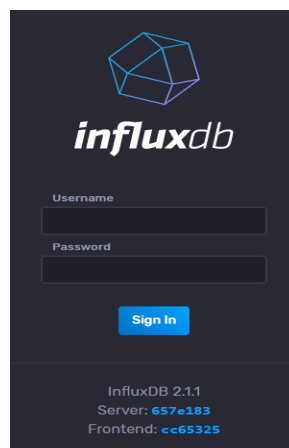
Esto abrirá una consola tal que así:

```
ubuntu@ip-172-31-12-147:~$ sudo docker exec -it 37b0f78a6b25 /bin/bash
root@37b0f78a6b25:/# |
```

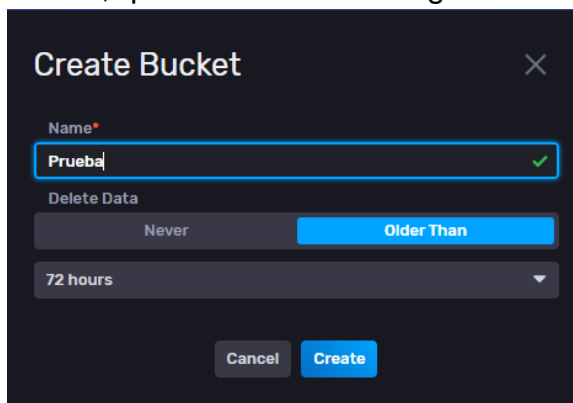
Donde podremos acceder a la configuración y administración por comandos, aunque lo recomendable es por su interfaz web.

## Configuración

Hecho lo anterior nos vamos a un navegador web y ponemos ip\_servidor:8086 para acceder a la interfaz web, nos saldrá una página donde nos pedirá la información de logeo de la cuenta que se creó cuando se lanzó el compose up.



Entramos y en la parte izquierda vamos a data y lo primero de todo es crear un bucket, que es donde se recogen los datos.



Es recomendable habilitar el borrado de datos tras unos días para que no se llene de el almacenamiento. Ahora hay que instalar telegraf.

## Telegraf

Telegraf es la herramienta que se utiliza para conectar los datos obtenidos a través de mosquito a influxdb y guardarlos ahí.

Como siempre lo primero es instalar el programa.

```
wget -qO- https://repos.influxdata.com/influxdb.key | sudo tee
/etc/apt/trusted.gpg.d/influxdb.asc >/dev/null
```

```
source /etc/os-release
```

```
echo "deb https://repos.influxdata.com/${ID} ${VERSION_CODENAME} stable" | sudo tee
/etc/apt/sources.list.d/influxdb.list
```

```
sudo apt-get update && sudo apt-get install telegraf
```

Una vez instalado nos vamos al archivo de configuración /etc/telegraf/telegraf.conf y lo dejamos tal que así.

```
[[outputs.influxdb_v2]]
# The URLs of the InfluxDB cluster nodes.
#
# Multiple URLs can be specified for a single cluster, only ONE of the
# urls will be written to each interval.
# ex: urls = ["https://us-west-2-1.aws.cloud2.influxdata.com"]
urls = ["http://192.168.12.207:8086"]
#
# Token for authentication.
token = "qc8ctkj8Wyzf09EYAyEFO0uAqPdc9SHUaAv8yqcD2EZ19pt3KGSiZ9nk6dUyAhXnlocHBibuUuTne64q5xS9A=="
#
# Organization is the name of the organization you wish to write to; must exist.
organization = "iesgc"
#
# Destination bucket to write into.
bucket = "iesgc"
```

Tenemos que indicar la url de influxdb junto a su puerto, poner el token que se obtiene en la parte de data de influxdb, api tokens y seleccionas el nombre del usuario que se creó ahí te saldrá una cadena de valores y la pegamos aquí. Luego ponemos en organization y bucket el nombre del bucket que creamos. Luego al fondo del archivo añadimos lo siguiente.

```
[[inputs.mqtt_consumer]]
    servers = ["tcp://localhost:1883"]
    topics = ["#"]

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

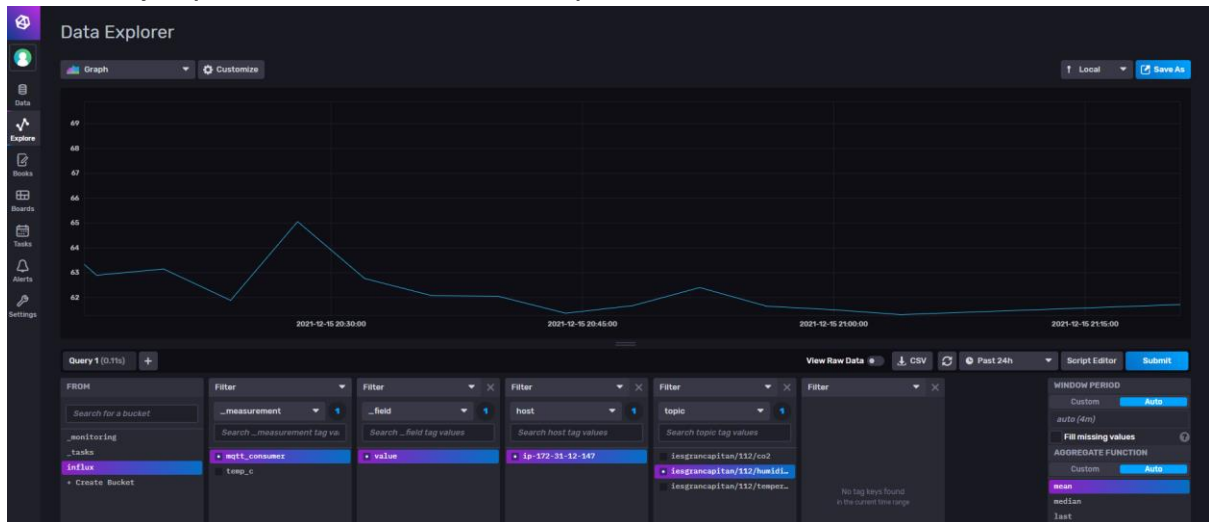
database = "sensores"
skip_database_creation = true
username = "iesgc"
password = "proyecto"
```

Con todo esto añadido recibirá los datos y luego los pasará a influxdb donde serán guardados. Reiniciamos el servicio y la parte de telegraf está terminada.



## Datos

Si todo está configurado como lo dicho anteriormente debería de guardarse los datos ya en influxdb. Para acceder a ellos, en la parte izquierda nos vamos a explore y ahí seleccionamos el bucket, que en mi caso es influx, y podemos elegir que ver, si el co2, temperatura, humedad o cualquier combinación de las tres. En este ejemplo enseñaré la humedad que había el día 15 de las 20:15 a las 21:20.



También podemos ver los valores en una tabla.

table	_measurement	_field	_value	_start	_stop	_time	host	topic
0	mqtt_consumer	value	69.85	2021-12-15T18:26:31.919Z	2021-12-16T18:26:31.919Z	2021-12-15T19:00:00.000Z	ip-172-31-12-147	iesgrancapitan/112/humidity
0	mqtt_consumer	value	68.4093823255814	2021-12-15T18:26:31.919Z	2021-12-16T18:26:31.919Z	2021-12-15T19:04:00.000Z	ip-172-31-12-147	iesgrancapitan/112/humidity
0	mqtt_consumer	value	65.14629629629628	2021-12-15T18:26:31.919Z	2021-12-16T18:26:31.919Z	2021-12-15T19:08:00.000Z	ip-172-31-12-147	iesgrancapitan/112/humidity
0	mqtt_consumer	value	65.2792452381887	2021-12-15T18:26:31.919Z	2021-12-16T18:26:31.919Z	2021-12-15T19:12:00.000Z	ip-172-31-12-147	iesgrancapitan/112/humidity
0	mqtt_consumer	value	62.913297547169894	2021-12-15T18:26:31.919Z	2021-12-16T18:26:31.919Z	2021-12-15T19:16:00.000Z	ip-172-31-12-147	iesgrancapitan/112/humidity
0	mqtt_consumer	value	63.16415894339623	2021-12-15T18:26:31.919Z	2021-12-16T18:26:31.919Z	2021-12-15T19:20:00.000Z	ip-172-31-12-147	iesgrancapitan/112/humidity
0	mqtt_consumer	value	61.90188679245281	2021-12-15T18:26:31.919Z	2021-12-16T18:26:31.919Z	2021-12-15T19:24:00.000Z	ip-172-31-12-147	iesgrancapitan/112/humidity
0	mqtt_consumer	value	65.87187499999999	2021-12-15T18:26:31.919Z	2021-12-16T18:26:31.919Z	2021-12-15T19:28:00.000Z	ip-172-31-12-147	iesgrancapitan/112/humidity
0	mqtt_consumer	value	62.70145145145148	2021-12-15T18:26:31.919Z	2021-12-16T18:26:31.919Z	2021-12-15T19:32:00.000Z	ip-172-31-12-147	iesgrancapitan/112/humidity

Y también de muchas otras formas.

# Grafana

Esta es la herramienta encargada de leer los datos de influxdb y crear gráficos donde puedes configurar alertas para que envíe emails a las direcciones que uno desee.

## Instalación

Lo primero de todo es instalar grafana. Para esto hay que instalar unos paquetes básicos antes.

```
sudo apt-get install -y apt-transport-https
```

```
sudo apt-get install -y software-properties-common wget
```

Ahora obtenemos el repositorio de grafana.

```
wget -q -O - https://packages.grafana.com/gpg.key | sudo apt-key add
```

```
echo "deb https://packages.grafana.com/enterprise/deb stable main" | sudo tee -a /etc/apt/sources.list.d/grafana.list
```

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

Ahora actualizamos los repositorios y ya podemos instalar grafana.

```
sudo apt-get update
```

```
sudo apt-get install grafana
```

Hecho esto ya se instalará e iniciará automáticamente grafana, una cosa a tener en cuenta es que para reiniciar, iniciar, parar etc... hay que usar `/etc/init.d/grafana-server`

## Configuración de gráficos

Una vez instalado nos vamos a un navegador y escribimos la siguiente URL `ip-servidor:3000`.

Una vez dentro nos pedirá que ingresemos, el usuario y contraseña inicial es admin admin, una vez entremos nos pedirá una contraseña nueva.

Ya dentro por fin, lo primero que tenemos que hacer es añadir un origen de datos.

Le damos a añadir data source y seleccionamos influxDB.

Le damos un nombre y MUY importante, seleccionamos en query language "flux" ya que es como estamos haciendo las consultas y está configurado.

Name ⓘ InfluxDB

Query Language

Flux

Ahora ponemos la ip ( sirve con <http://localhost:8086> )

URL ⓘ <http://localhost:8086>

Y bajamos al final y donde pone InfluxDB Details ponemos la organización, el token que se obtiene de la misma forma que con telegraf y el bucket.

Organization	influx
Token	configured
Default Bucket	influx

Le damos a save and test y si todo va bien nos deberá de decir que se ha encontrado 3 buckets.

Ahora ya podemos crear los gráficos donde se crearán las alertas.

Para crear un gráfico solo hay que darle al más en la parte izquierda y luego a new panel, nos saldrá esto.

The screenshot shows the InfluxDB web interface. The main area displays 'No data'. The bottom section shows the 'Query' tab with 'Data source' set to 'InfluxDB'. The right sidebar is open, showing 'Search options' and 'Panel options'. Under 'Panel options', 'Title' is set to 'Panel Title' and 'Description' is empty. 'Transparent background' is checked. 'Panel links' and 'Repeat options' are expanded. 'Tooltip' is set to 'Single'. 'Legend mode' is set to 'List' and 'Legend placement' is set to 'Bottom'. 'Legend values' are set to 'Choose'.

Aquí podremos escribir la consulta que usará para leer los datos.

Para obtenerla nos vamos a la web de influxdb y le damos a explore, seleccionamos el parámetro que queremos ver, en este caso va a ser humedad y le damos a submit y luego a script editor para ver la consulta.

```
Query 1 (0.12s) +
1 from(bucket: "influx")
2   |> range(start: v.timeRangeStart, stop: v.timeRangeStop)
3   |> filter(fn: (r) => r["_measurement"] == "mqtt_consumer")
4   |> filter(fn: (r) => r["_field"] == "value")
5   |> filter(fn: (r) => r["host"] == "ip-172-31-12-147")
6   |> filter(fn: (r) => r["topic"] == "iesgrancapitan/112/humidity")
7   |> aggregateWindow(every: v.windowPeriod, fn: mean, createEmpty: false)
8   |> yield(name: "mean")
```

Esto tendrá que salir, lo copiamos y lo pegamos en el editor de grafana y se tendrá que ver un gráfico.



Una vez hecho esto lo repetimos con el co2 y la temperatura. Nos quedará una dashboard tal que así.



## Alertas

Las alertas están pensadas para que cada vez que el último valor se salga de un rango de valores o supere un valor máximo envíe un email a unos receptores

previamente introducidos. También envía emails cuando hay un error, es decir no ha recibido ningún dato y cuando vuelve al funcionamiento normal y los valores vuelven al rango seguro.

Lo primero de todo es instalar ssmtp en el servidor, que es el gestor que se encargará de enviar los correos.

Su instalación es la siguiente:

```
sudo apt install msmtp
```

Luego creamos el archivo de configuración de msmtp al que llamaremos msmtp.rc e introducimos las siguientes líneas de código que también incluyen los datos del correo.

```
# Default settings
defaults
auth      on
tls       on
tls_trust_file  /etc/ssl/certs/ca-certificates.crt
logfile   ~/.msmtp.log

account      outlook
host         smtp.office365.com
port         587
from         iesgcpruebagrafana@outlook.com
user         iesgcpruebagrafana@outlook.com
password     Proyecto2022.
```

Y ahora le damos permisos 600 con chmod para que funcione correctamente

Y ahora para comprobar que todo está bien podemos enviar un correo de prueba con el siguiente comando.

```
echo -e "\nHOLA." | msmtp -a outlook a19romefr@iesgrancapitan.org
```

iesgcpruebagrafana@outlook.com

para ▼

🌐 inglés ▼ > español ▼ [Traducir mensaje](#)

HOLA.

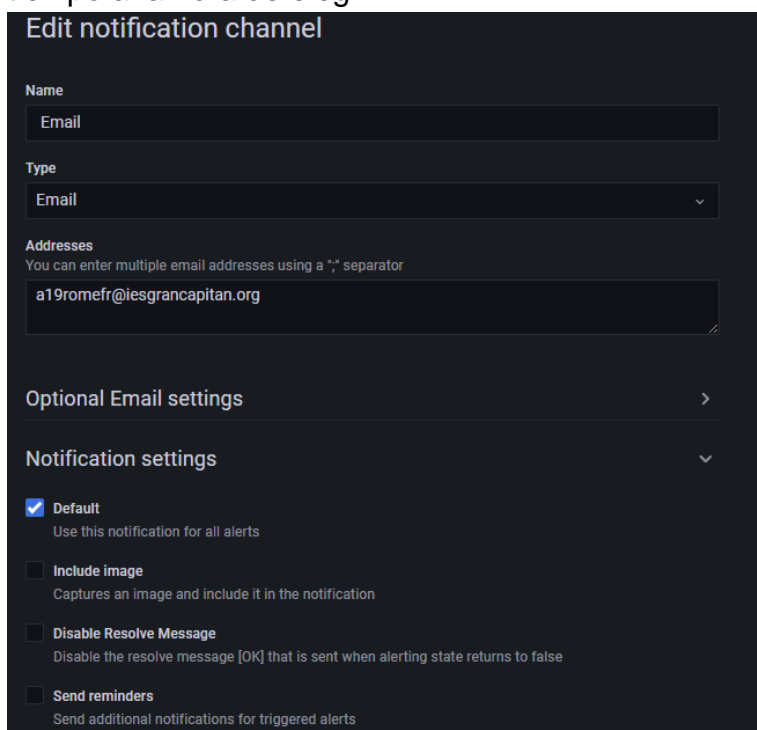
Hecho esto ya tenemos configurado el gestor, ahora tenemos que configurar grafana, nos vamos a su archivo de configuración /etc/grafana/grafana.ini

Una vez dentro nos vamos al apartado smtp e introducimos los datos del correo que usaremos.

```
[smtp]
enabled = true
host = smtp.office365.com:587
user = iesgcpruebagrafana@outlook.com
# If the password contains # or ; you have to wrap it in quotes
password = Proyecto2022.
;cert_file =
;key_file =
skip_verify = true
from_address = iesgcpruebagrafana@outlook.com
from_name = Grafana
```

A continuación reiniciamos grafana mediante la orden `/etc/init.d/grafana-server restart`

Con esto ya grafana puede enviar correos. Pero antes hay que ir a la web de grafana y en la parte izquierda le damos a la campana, dentro de ahí seleccionamos notification channels y añadimos uno nuevo, le damos un nombre y seleccionamos email, también se puede elegir muchos tipos más. Y ponemos las direcciones de correo a las que queremos que lleguen los emails. Además le damos a notification settings y seleccionamos default para que sea por defecto siempre y nos ahorre tiempo a la hora de elegir.



Edit notification channel

Name  
Email

Type  
Email

Addresses  
You can enter multiple email addresses using a ";" separator  
a19romefr@iesgrancapitan.org

Optional Email settings >

Notification settings ▾


- ☒ **Default**  
Use this notification for all alerts
- ☐ **Include image**  
Captures an image and include it in the notification
- ☐ **Disable Resolve Message**  
Disable the resolve message [OK] that is sent when alerting state returns to false
- ☐ **Send reminders**  
Send additional notifications for triggered alerts

Si le damos a test, nos enviará un correo de prueba, si todo está bien nos debería de llegar un correo a la direcciones especificadas.

[Alerting] Test notification Externo Recibidos x

Grafana <iesgcpruebagrafana@gmail.com>  
para mí ▾

🌐 inglés ▾ > español ▾ [Traducir mensaje](#)



**[Alerting] Test notification**

Someone is testing the alert notification within Grafana.

**Error message**

this is only a test

Metric name	Value
High value	100.000
Higher Value	200.000

[View your Alert rule](#)[Go to the Alerts page](#)

Ahora nos vamos al dashboard y creamos las alertas para cada gráfico.  
Por ejemplo seleccionamos temperatura y le damos a alert, hay dentro le damos a crear una, le damos un nombre y un periodo de revisión de datos, la placa envía datos cada 10 segundos, por lo que he puesto que lo revise cada 10 segundos y si en un 1 minuto no ha vuelto a un dato dentro del rango envía la alerta.

**Rule type**

**Rule name**

Temperatura

**Folder** ⓘ  
Select a folder to store your rule.

112 ▾

**Group**  
Rules within the same group are evaluated after the same time interval.

112

**Define alert conditions**

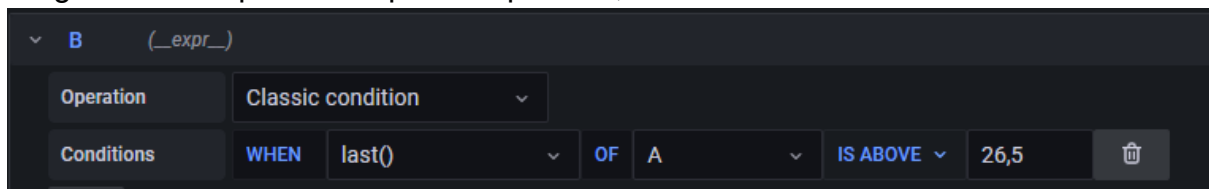
**Condition**  
The query or expression that will be alerted on

B ▾

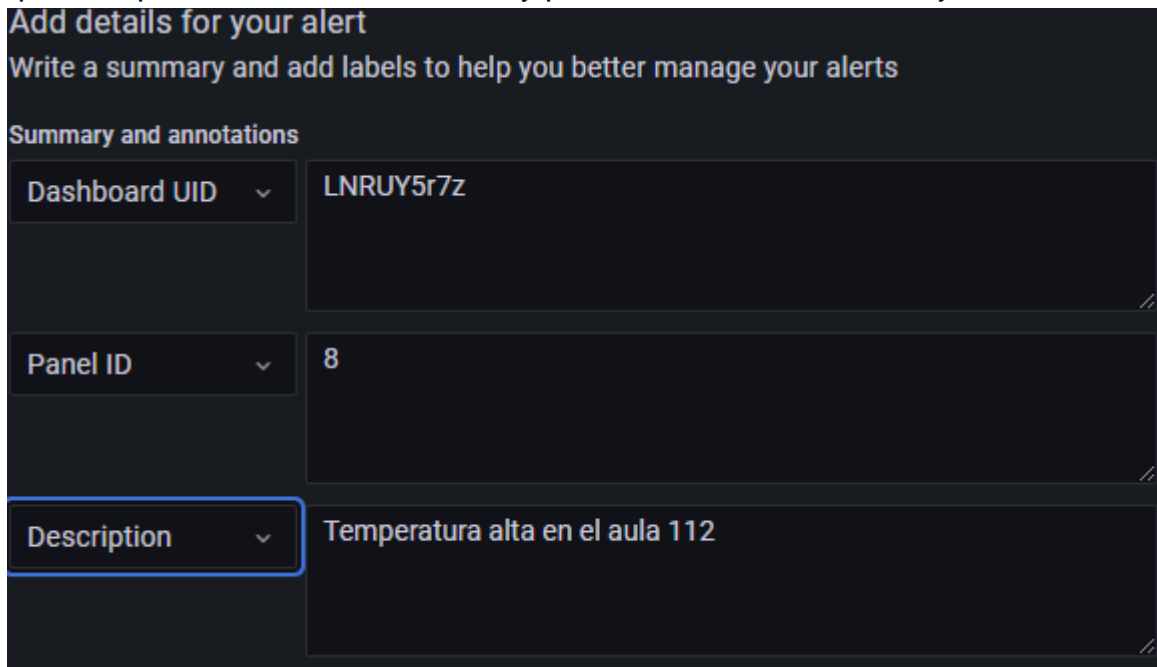
**Evaluate**  
Evaluation interval applies to every rule within a group. It can overwrite the interval of an existing alert rule.

Evaluate every ⓘ 10s for ⓘ 1m

Luego en conditions seleccionamos last() en el valor que selecciona y “is outside range” a esto le ponemos que si supera 26,5 salte la alerta



Y al final en la notificación nos selecciona automáticamente la forma de envío ya que lo especificamos anteriormente y podemos escribir un mensaje.

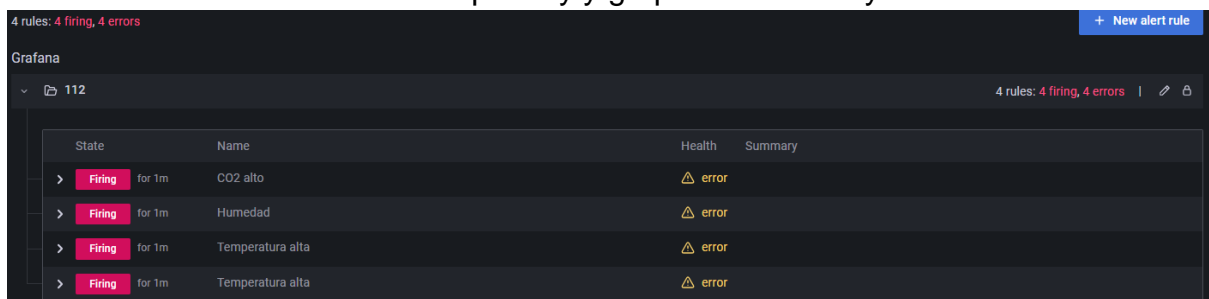


Una cosa a tener en cuenta es que las alertas son planas, es decir no puede identificar si la temperatura está por encima o por debajo, pero envía el valor por correo, así que la persona que lo lea puede saber si el valor está por debajo o por encima o si hay un error en la placa.

Ahora repetimos el mismo proceso en cada gráfico y ponemos en el de co2 que el valor sea superior a 1000 y en humedad que los valores estén fuera del rango de 29.5 y 70.5

Guardamos los cambios y ya estarían las alertas completadas

Para comprobar que todo ha ido bien nos podemos ir a la campana de la izquierda, donde saldrán todas las alertas que hay y grupos de alertas y el estado.



State	Name	Health	Summary
> Firing for 1m	CO2 alto	error	
> Firing for 1m	Humedad	error	
> Firing for 1m	Temperatura alta	error	
> Firing for 1m	Temperatura alta	error	



Si ahora en cualquier momento hay un error en la placa, es decir no envía, o se dan las condiciones para que se envíen alertas o vuelve a un estado normal nos llegará un correo diciéndonos si no hay datos, si está ok o si hay un aviso.

**Resolved: 4 alerts**

**Resolved** **CO2 alto**

**Value:**

**Description:** CO2 en el aula 112

**Labels:** • alertname: CO2 alto

Este es un correo que nos indica que se ha arreglado la incidencia.

**Firing** **DatasourceNoData**

**Value:**

**Description:** Humedad aula 112 alta

**Labels:** • alertname: DatasourceNoData  
• datasource\_uid: zHPDjp9nz  
• ref\_id: A  
• rulename: Humedad alta

Correo de que no hay datos.

**Firing** **CO2**

**Value:** [ var='B0' metric='mqtt\_consumer' labels={host=sensores, topic=ESP\_Easy/co2/PPM} value=2367 ]

**Description:** CO2 aula 112 alto.

**Labels:** • alertname: CO2

Correo de que los datos superan el límite establecido.

# Bibliografía

<https://docs.influxdata.com/influxdb/v2.1/>

<https://techexpert.tips/es/grafana-es/configuracion-de-la-notificacion-por-correo-electronico-de-grafana/>

<https://pypi.org/project/paho-mqtt/>

[https://www.youtube.com/watch?v=dk6895EwVbM&ab\\_channel=SteveCope](https://www.youtube.com/watch?v=dk6895EwVbM&ab_channel=SteveCope)

<https://emariete.com/medidor-casero-co2/>

[https://www.letscontrolit.com/wiki/index.php?title=ESP\\_Easy\\_web\\_interface](https://www.letscontrolit.com/wiki/index.php?title=ESP_Easy_web_interface)

<https://github.com/josejuansanchez/iot-demo>

<https://josejuansanchez.org/teaching/2021/02/18/iot-dashboard-sensores.html>