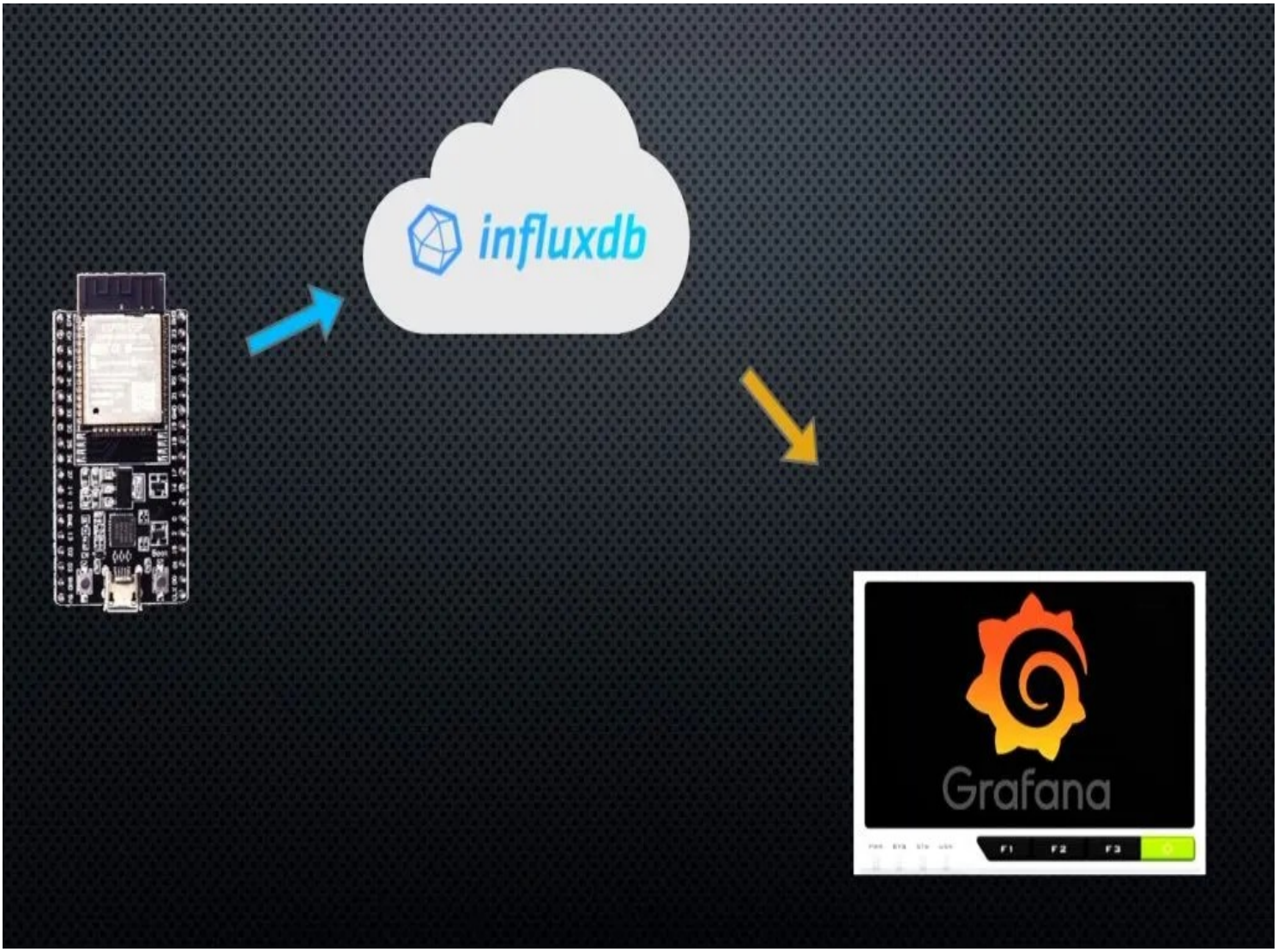
ESP32-INFLUXDB-GRAFANA

Autor: Ricardo Saravia Armaulia
Asignatura: Sistemas Digitales



INDICE

MOTIVACION....3
ESTADO DEL ARTE....4
TECNICAS Y MATERIALES UTILIZADOS....5
HARDWARE....5
SENSORES INTERNOS....5
SENSOR HALL:....6
SENSOR DE TEMPERATURA INTERNA:....6
PROGRAMAS....6
ARDUINO IDE 1.8.19....6
INFLUXDB:....6
GRAFANA:....6
PROCEDIMIENTO....7
SENSOR HALL....7
TEMPERATURA INTERNA....12
FUTUROS DESARROLLOS POSIBLES....13
CONCLUSIONES....13

MOTIVACION

Este proyecto tiene como objetivo aprender como poder enviar cualquier dato de medida y/o monitorización usando placas y códigos de Arduino a la red, usando un intermediario funcionando como una base de datos, y un graficador que procesara los datos de forma casi simultanea y nos hará un grafico, el cual podremos elegir el tipo según nos convenga.

La posibilidad que nos ofrece esta forma de procesar datos es bastante amplia, ya que por ejemplo al tener los datos de un sensor en la red, podemos consultarla en cualquier momento y lugar con solo tener el Arduino conectado, no necesariamente estando en el mismo momento y lugar que la placa de Arduino. Es decir una vez este funcionando podremos hacer una consulta, simplemente manteniendo conectado la subida de datos.

ESTADO DEL ARTE

Se incluirán las consultas realizadas a diferentes paginas web que han servido como guía y con las cuales se podría replicar el proyecto.

Un comentario que seria bueno hacer, es que el proyecto en su inicio iba a ser un simple envío de datos a través de proceso MQTT, pero se amplió para poder usar un intermediario entre los datos y un graficador, con la ventaja de que es en tiempo real y se puede realizar una consulta de esos datos tiempo después de haber realizado las mediciones.

https://europe-west1-1.gcp.cloud2.influxdata.com/orgs/867f022a144e29ea/data-explorer INFLUXDB

http://localhost:3000/?orgId=1 GRAFANA

https://www.the-diy-life.com/grafana-weather-dashboard-using-influxdb-and-an-esp32-in-depth-tutorial/ PRINCIPAL GUIA USADA

https://tecnotizate.es/esp32-mapeo-de-pines-y-sensores-internos/ INFORMACION SOBRE LOS SENSORES DEL ESP32

http://kio4.com/arduino/219_Wemos_Temperatura_Hall.htm CODIGOS PARA LOS SENSORES (ORIGINAL, LUEGO SE TIENE QUE HACER UNA ADAPTACION PARA EL PROYECTO)

https://aprendiendoarduino.wordpress.com/2016/11/16/iot-con-arduino/ INFORMACION SOBRE IoT

TECNICAS Y MATERIALES UTILIZADOS

Empezaremos por describir los materiales utilizados en este proyecto, partiendo por el hardware y luego por los programas usados para su realización.

HARDWARE

Placa de Arduino ESP32

El ESP32 es el hermano mayor del ESP8266. Para muchos, el ESP32 es el primer chip realmente IoT. No está diseñado para reemplazar al ESP8266 pero sí lo mejora en todos los aspectos y añade nuevas características. Está basado en el procesador de 32 bits Xtensa LX6 dual core hasta 240MHz, incluye soporte WIFI encriptado, Bluetooth 4.2, amplificador de audio, cargador de baterías, interfaz para pulsadores touch, etc.

SENSORES INTERNOS

El codigo del proyecto trabaja sobre estos sensores ya incorporados en la placa de Arduino ESP32.

SENSOR HALL:

Este sensor resulta útil puesto que cualquier campo magnético cercano a la placa puede afectar al comportamiento de esta.

Este sensor hall interno, está puesto con la idea de que si el chip atraviesa un campo magnético, tome una acción, es decir, se desconecte, emita un aviso... pero no está puesto con la idea de ser utilizado para medir y mostrar magnetismo externo, aunque también puede servir para esto.

SENSOR DE TEMPERATURA INTERNA:

En realidad es un sensor de temperatura del núcleo. Ese sensor tiene como objetivo medir la temperatura de los núcleos del chip, no es un sensor destinado a medir temperatura externa.

PROGRAMAS

En este proyecto se han usado varios programas, todos usados de manera online(menos Arduino IDE) aunque se ha necesitado previa descarga en uno de ellos.

ARDUINO IDE 1.8.19

Un IDE es un entorno de programación que ha sido empaquetado como un programa de aplicación; es decir, que consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica (GUI).

INFLUXDB:

Es una base de datos de series temporales diseñada para el almacenamiento rápido y de alta disponibilidad y la recuperación de datos de series temporales. Puede funcionar como una solución independiente, o se puede utilizar para procesar datos de Graphite.
Tambien se puede usar como exportador de datos.

GRAFANA:

Grafana es una herramienta de código abierto para el análisis y visualización de métricas. Se utiliza frecuentemente para visualizar de una forma elegante series de datos en el análisis de infraestructuras y aplicaciones.

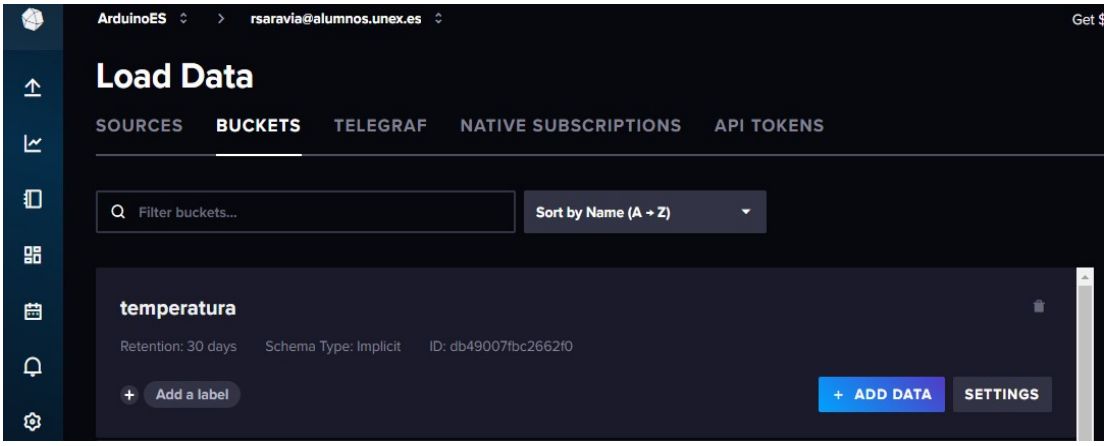
PROCEDIMIENTO

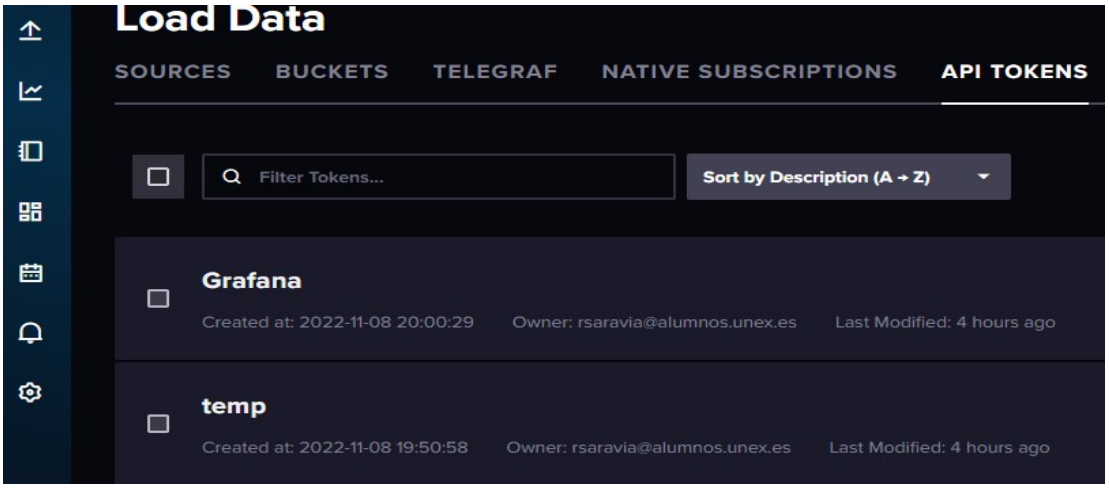
SENSOR HALL

Una breve descripción de como funciona o lo que hace el proyecto , es que, a partir de unas mediciones en el sensor interno del ESP32 en tiempo real, enviamos estos datos a un tipo de almacenaje (INFLUXDB), para posteriormente enviarlos a un graficador (GRAFANA).

Una vez descrita las herramientas usadas, el desarrollo del proyecto seria el siguiente:

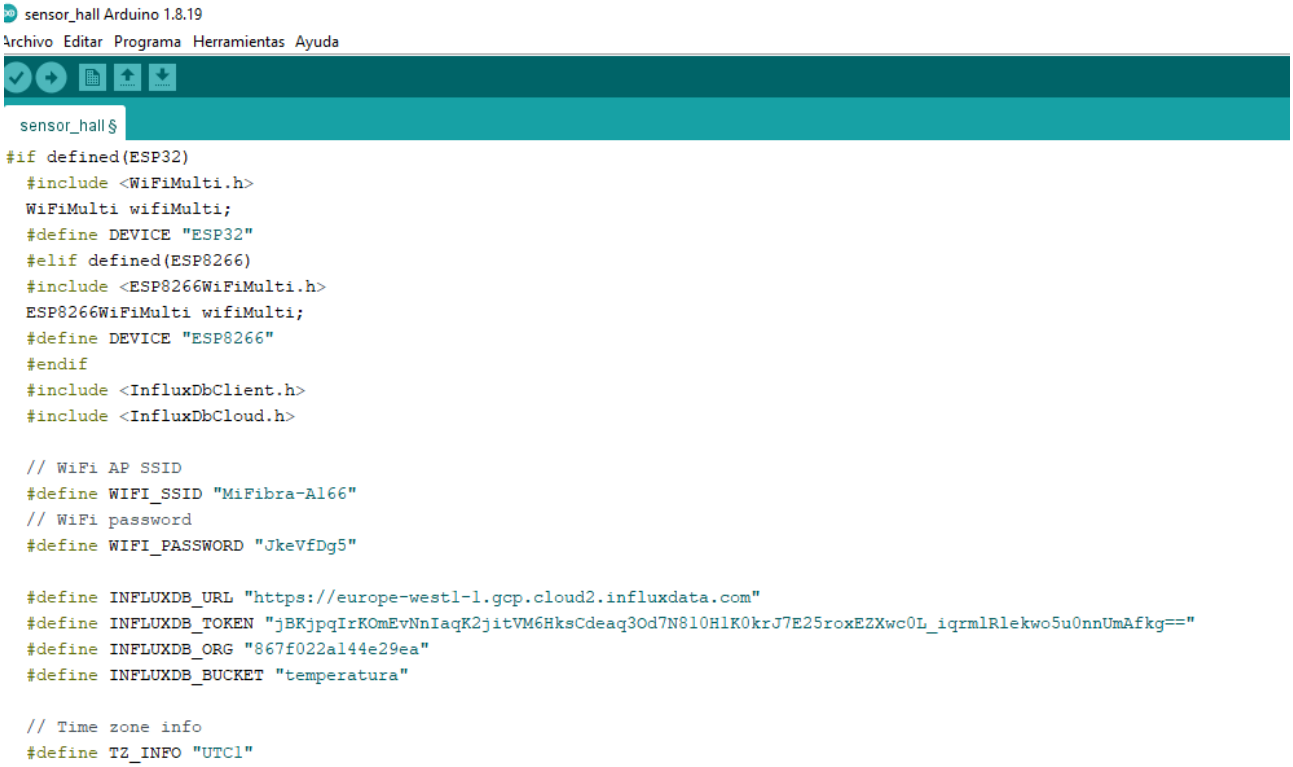
1º Creamos un BUCKET en INFLUXDB, esto funciona como si creásemos una carpeta en la que meteremos nuestros datos, posteriormente creamos un API TOKEN, el cual podríamos decir que es como una forma de direccionar los datos hacia el BUCKET creado previamente.





2º Se ha creado un código acorde al objetivo del proyecto, en este caso usar los sensores, se han diseñado dos códigos iguales en la parte de “void setup()” para ambos sensores , pero distintas en la parte de “void loop()”.

El primer código es aquel que usa el sensor hall de la placa:




```
#if defined(ESP32)
  #include <WiFiMulti.h>
  WiFiMulti wifiMulti;
  #define DEVICE "ESP32"
  #elif defined(ESP8266)
  #include <ESP8266WiFiMulti.h>
  ESP8266WiFiMulti wifiMulti;
  #define DEVICE "ESP8266"
  #endif
  #include <InfluxDbClient.h>
  #include <InfluxDbCloud.h>

  // WiFi AP SSID
  #define WIFI_SSID "MiFibra-A166"
  // WiFi password
  #define WIFI_PASSWORD "JkeVfDg5"

  #define INFLUXDB_URL "https://europe-westl-1.gcp.cloud2.influxdata.com"
  #define INFLUXDB_TOKEN "jBKjpqIrKOmEvNnIaqK2jitVM6HksCdeaq3Od7N8l0HlK0krJ7E25roxEZXwc0L_iqrmlRlekwo5u0nnUmAfkg=="
  #define INFLUXDB_ORG "867f022a144e29ea"
  #define INFLUXDB_BUCKET "temperatura"

  // Time zone info
  #define TZ_INFO "UTCl"
```

Esta seria la primera parte del codigo, en la que cabe aclarar que previamente, se ha tenido que realizar una instalación de la placa ESP32. En esta parte vemos como lo primero seria reconocer la placa usada, la librería “wifimulti.h” ya venia incluida, sin embargo se ha añadido la librería “influxdb” desde el buscador de Arduino. Esta ultima librería nos permite realizar la conexión vía Wi-Fi con INFLUXDB.

Para que la conexión vía Wi-Fi sea posible definimos nuestra “WIFI-SSID”(nombre de la red wi-fi) y la contraseña de la misma (WIFI-PASSWORD)

En cuanto a la parte de las definiciones de “INFLUXDB_XXX”, el primero seria la URL la cual varia según en que región del mundo te encuentres y que es generado de forma automática. El TOKEN es una forma de direccionamiento hacia un “hueco” previamente creado dentro de BUCKET, que en este caso recibe el nombre de temperatura. La parte de ORG es generada automáticamente cuando generamos un BUCKET en INFLUXDB.

Siguiendo con la parte de “void setup()”. Queda aclarar que esta parte es común tanto para el codigo de sensor hall como el de temperatura interna.

```
InfluxDBClient client(INFLUXDB_URL, INFLUXDB_ORG, INFLUXDB_BUCKET, INFLUXDB_TOKEN, InfluxDbCloud2CACert);
Point sensor("wifi_status");
void setup() {
  Serial.begin(115200);

  // Setup wifi
  WiFi.mode(WIFI_STA);
  wifiMulti.addAP(WIFI_SSID, WIFI_PASSWORD);

  Serial.print("Connecting to wifi");
  while (wifiMulti.run() != WL_CONNECTED) {
    Serial.print(".");
    delay(100);
  }
  Serial.println();
  timeSync(TZ_INFO, "pool.ntp.org", "time.nis.gov");


  // Check server connection
  if (client.validateConnection()) {
    Serial.print("Connected to InfluxDB: ");
    Serial.println(client.getServerUrl());
  } else {
    Serial.print("InfluxDB connection failed: ");
    Serial.println(client.getLastErrorMessage());
  }
}
```

La primera linea de codigo que vemos en esta parte, usa los parámetros de INFLUXDB que definimos previamente, para así poder establecer el vinculo que nos permite enviar datos

La parte del codigo de “void setup” se encarga de realizar la conexión, tanto con nuestro Wi-Fi en primer lugar, como con INFLUXDB. En caso de no ser posible nos notificara un error.

Parte de código “void loop()”:

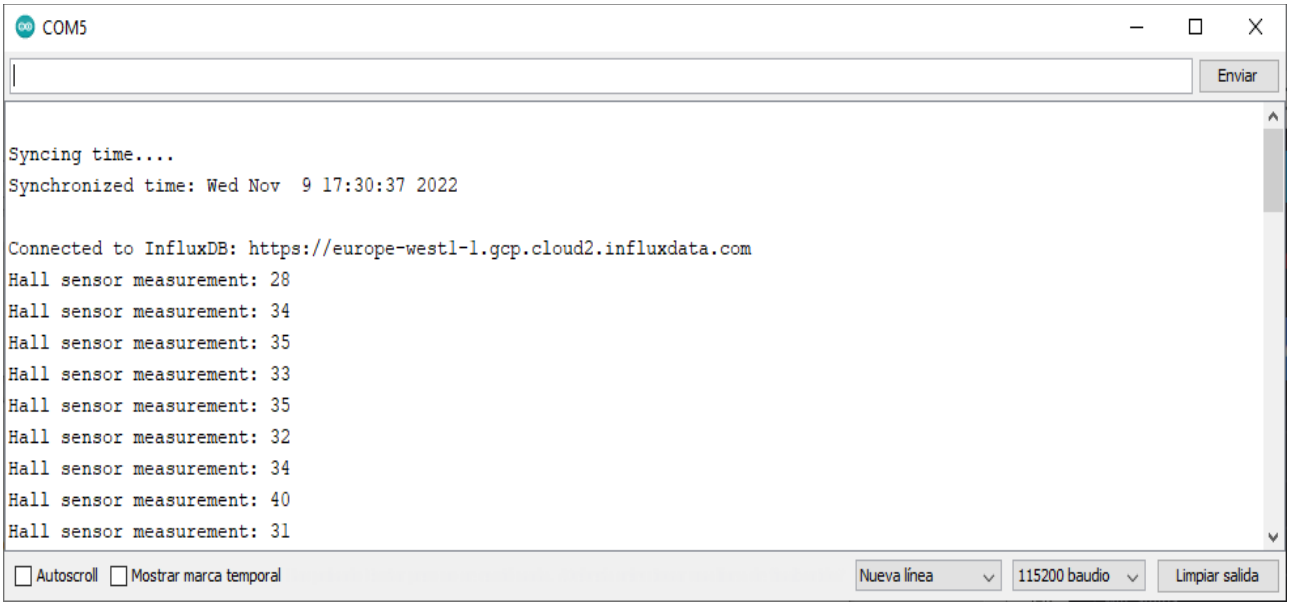
```
void loop() {

  int measurement = 0;

  measurement = hallRead();

 sensor.clearFields();
 sensor.addField("sensor_hall",measurement);

 if(wifiMulti.run() !=WL_CONNECTED)
    Serial.print("Conexion wifi perdida");

 if(!client.writePoint(sensor))
 {
  Serial.print("Error influxdb escritura ");
  Serial.println(client.getLastErrorMessage());
 }

  Serial.print("Hall sensor measurement: ");
  Serial.println(measurement);
  delay(1000);
}
```

Primero declaramos un variable(measurement) como entera, en esta variable guardaremos el valor de hallRead(), que es la función que nos proporciona la medición del campo magnético, cabe destacar que es necesario que la variable sea de tipo “int” o “float” ya que INFLUXDB solo acepta este formato para realizar el guardado de datos.

Con las dos lineas siguientes creamos dentro de BUCKET un nuevo “field” , es decir, que esta es la parte que nos permite enviar los datos.

A continuación comprobamos si la conexión Wi-Fi es correcta para enviar los datos, y después si estos datos son del formato correcto para INFLUXDB.

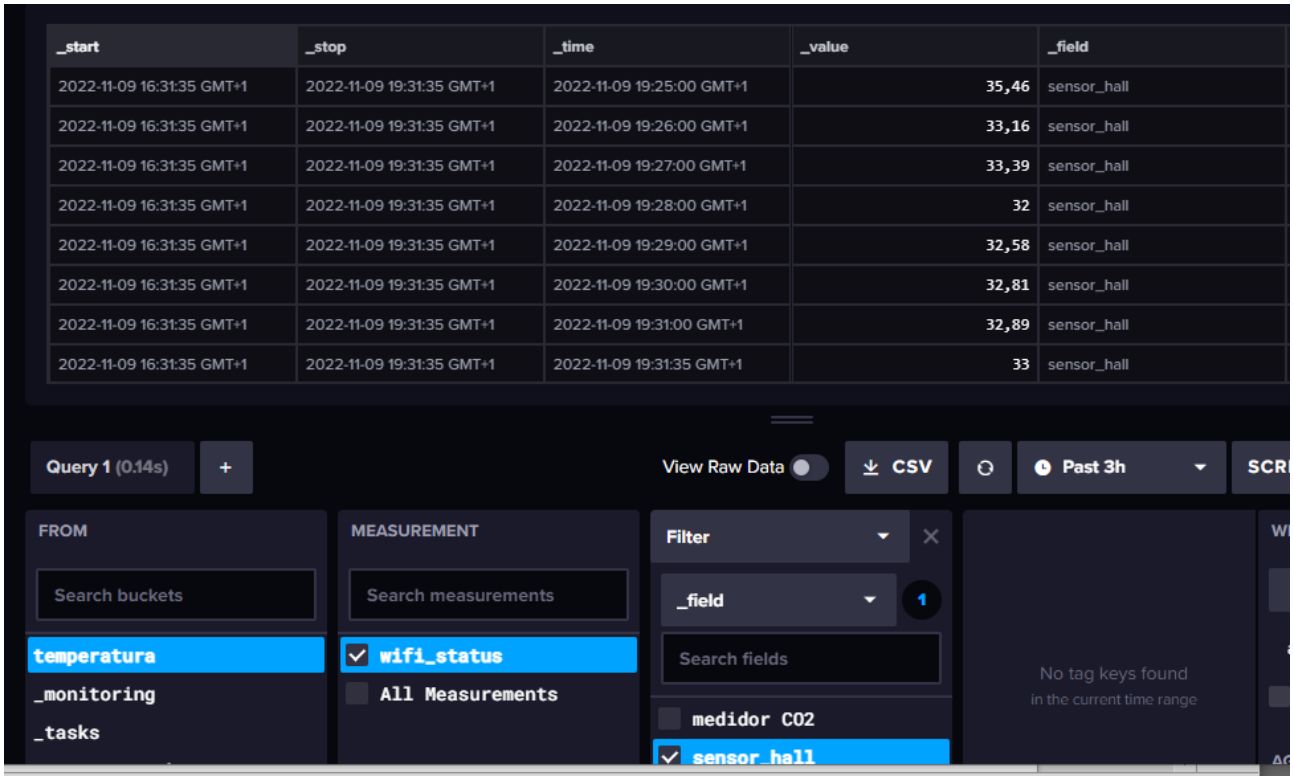
Una vez realizado esto, deberíamos poder ver esto en el puerto COM de Arduino:



Y hasta aquí termina la parte de codigo de Arduino del sensor hall, el resto es el procesado de datos.

3º PROCESAMIENTO DE DATOS

En INFLUXDB deberíamos ir recibiendo los datos del sensor hall, en esta parte elegiremos la forma de tabla simple para ver los datos, nos da información del tiempo y la medida registrada, ademas de otros datos.



_start	_stop	_time	_value	_field
2022-11-09 16:31:35 GMT+1	2022-11-09 19:31:35 GMT+1	2022-11-09 19:25:00 GMT+1	35,46	sensor_hall
2022-11-09 16:31:35 GMT+1	2022-11-09 19:31:35 GMT+1	2022-11-09 19:26:00 GMT+1	33,16	sensor_hall
2022-11-09 16:31:35 GMT+1	2022-11-09 19:31:35 GMT+1	2022-11-09 19:27:00 GMT+1	33,39	sensor_hall
2022-11-09 16:31:35 GMT+1	2022-11-09 19:31:35 GMT+1	2022-11-09 19:28:00 GMT+1	32	sensor_hall
2022-11-09 16:31:35 GMT+1	2022-11-09 19:31:35 GMT+1	2022-11-09 19:29:00 GMT+1	32,58	sensor_hall
2022-11-09 16:31:35 GMT+1	2022-11-09 19:31:35 GMT+1	2022-11-09 19:30:00 GMT+1	32,81	sensor_hall
2022-11-09 16:31:35 GMT+1	2022-11-09 19:31:35 GMT+1	2022-11-09 19:31:00 GMT+1	32,89	sensor_hall
2022-11-09 16:31:35 GMT+1	2022-11-09 19:31:35 GMT+1	2022-11-09 19:31:35 GMT+1	33	sensor_hall

Por ultimo nos queda enviar el archivo a GRAFANA, para este paso también es necesario crear un API TOKEN para poder enviar los datos, paso que se puede apreciar en el punto 1º, y que tiene el nombre de “Grafana”. Una vez realizado estos pasos hacemos login en GRAFANA y veríamos lo siguiente:

Las lineas de codigo que se ven en la parte de abajo se introducen primero, ya que son necesarias para importarlas desde INFLUXDB.

Un buen ejemplo seria ver una grafica primero en a lo largo de un amplio periodo de tiempo y luego una que ocupe un corto espacio de tiempo.

TEMPERATURA INTERNA

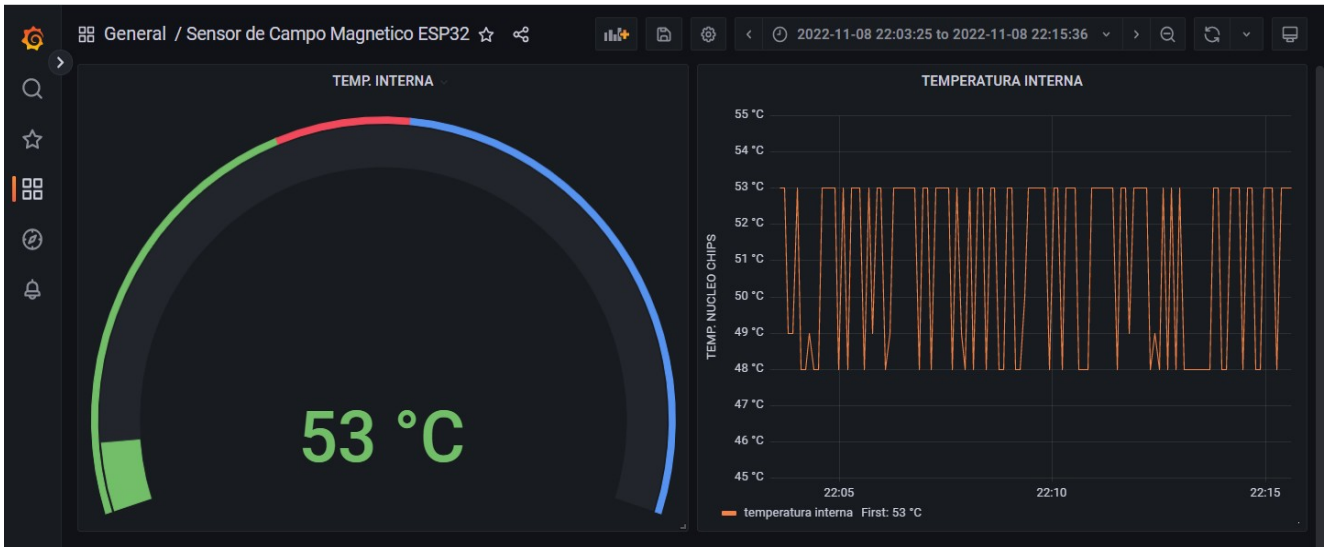
Para esta parte simplemente repetiremos los pasos descritos en la parte del codigo de sensor hall, con la variante que tendremos que generar un nuevo API TOKEN y sustituirlo en la parte de “INFLUXDB_TOKEN” del código. Respecto a la parte de código es todo igual excepto en la parte de “void loop”.

```
void loop() {

  int tempint=((temprature_sens_read()-32)/ 1.8);

  sensor.clearFields();
  sensor.addField("temperatura interna",tempint);

   if(wifiMulti.run() !=WL_CONNECTED)
      Serial.print("Conexion wifi perdida");

   if(!client.writePoint(sensor))
   {
    Serial.print("Error influxdb escritura ");
    Serial.println(client.getLastErrorMessage());
   }
  Serial.print("Temperature: ");
  // Convert raw temperature in F to Celsius degrees
  Serial.println(tempint);
  delay(5000);
}
```

Aquí volvemos a definir una variable tipo “int”, en la que posteriormente guardaremos el medidor de temperatura, que como se puede ver ya esta convertido a Celsius.

También se puede ver que hemos añadido un nuevo field, el cual estará dentro del BUCKET que creamos en la parte de código de sensor hall.

Y al igual que con el código de sensor hall, recibiremos los datos en INFLUXDB, para posteriormente enviarlos a GRAFANA, obteniendo como resultado final:



FUTUROS DESARROLLOS POSIBLES

Este proyecto esta muy relacionado con IoT(internet of things), es un concepto que se refiere a la interconexión digital de objetos cotidianos con Internet. Y puede servir como una introducción, ya que aunque en este caso solo usemos en nuestro proyecto un sensor interno de la placa, simplemente cambiando el contenido de codigo adaptándolo a otro tipo de sensores externos o internos y manteniendo el procedimiento que se ha explicado en el proyecto, podemos hacer y elaborar conexiones y envió de datos bastante complejos y que nos aportarían mucha información, aparte de poder visualizarlas.

CONCLUSIONES

Como aclaración seria relevante mencionar que, podríamos haber comprimido o unido ambos códigos en uno, pero se ha hecho de esta forma para ilustrar que simplemente con la dos primeras partes del codigo, las cuales solo variarían a la hora de declarar los parámetros de la red Wi-Fi, o añadiendo alguna librería que hiciera falta, podemos lograr enviar cualquier tipo de medida de cualquier sensor.

Este proyecto como ya se ha mencionado antes, puede servir como una introducción a IoT, faltando por desarrollar la parte de enviar los datos recibidos y graficados hacia otro dispositivo en tiempo real.

La cantidad de aplicaciones que se le puede dar a este tipo de procesado de datos es bastante amplio y variado, por lo que se usara seguramente en futuros proyectos.