



# DOCUMENTACIÓN SISTEMAS EMPOTRADOS

---

Alfonso Alarcón Tamayo

[Alfonsoalarcontamayo27@gmail.com](mailto:Alfonsoalarcontamayo27@gmail.com)

## Contenido

Introducción .....	3
Presentación del trabajo .....	3
<b>Tabla de componentes</b> .....	4
Nivel de sensado y actuación.....	6
Pasarela IoT .....	11
Servicio en la nube .....	15

# Introducción

Este trabajo ha sido realizado por Alfonso Alarcón Tamayo y en este documento detallaré todo lo relacionado con el trabajo de la asignatura de Sistemas Empotrados, Distribuidos y Ubicuos.

La estructura de la documentación estará dividida en los siguientes bloques:

- **Presentación del trabajo:** breve resumen del trabajo que he realizado
- **Tabla de componentes:** Aquellos que he utilizado para el proyecto.
- **Nivel de sensado y actuación**
- **Pasarela IoT**
- **Servicio en la nube**

## Presentación del trabajo

Para este trabajo he realizado el trabajo que proponen en la asignatura, es decir, una estación meteorológica. La estación estará compuesta de un apartado de sensado y actuación donde veremos el microcontrolador, los sensores y actuadores que controlaremos desde el propio microcontrolador y un apartado de sistema en tiempo real compuesto con semáforos y colas para reducir controlar y reducir el consumo del sistema.

Después veremos la comunicación entre el microcontrolador y la pasarela de comunicación usando máquinas de estado.

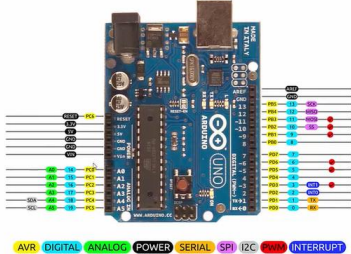


Entrando en el segundo bloque, veremos la pasarela donde recopilaremos la información de los microcontroladores y que usaremos como maestro y usaremos esta pasarela para mandar la información a la nueva



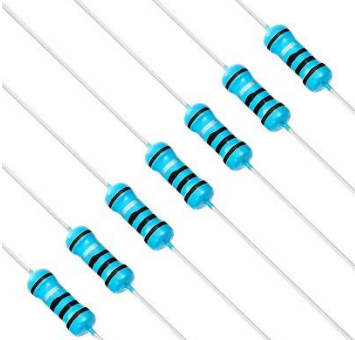
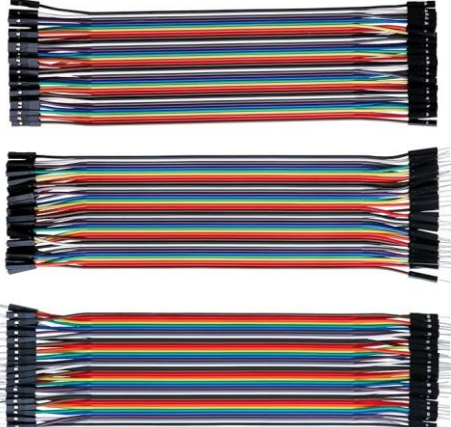

Por último, tenemos el bloque de Servicios en la nube (envío, almacenamiento e interpretación), donde como su nombre indica realizaremos el envío a una plataforma online de los datos obtenidos de los sensores y que tenemos almacenados en la pasarela para su interpretación

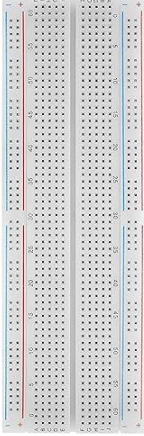
## Tabla de componentes

Este apartado es un listado inicial de los componentes que he utilizado en mi estación meteorológica ya que no he usado todos los que se piden en la cesta de la compra, la tabla está compuesta de una imagen del componente (tipo de actuadores o de microcontrolador), el propio nombre de esta a título informativo y una breve descripción de estos.

*He omitido cargadores, cables, midroSD HDMI, alargador USB y demás componentes relevantes*

Componente	Nombre	Descripción
	Arduino Mega de 2560	Es el microcontrolador que usaremos
	Raspperri PI 3B+	En nuestro proyecto será la pasarela que conectará el nivel de sensado y actuación con la nube, usa el S.O Raspbian
	Servomotor SG-9	Es un actuador que controlaremos en función del comando mandado

	Sensor DHT22	Sensor digital para obtener la temperatura y la humedad
	Sensor LDR	Sensor analógico de la intensidad de la luz
	Resistencias	Como su nombre indica para limitar la corriente
	Cables monohilo	Para la interconexión de los componentes a través de la tabla de prototipado, los sensores, actuadores y el microcontrolador
	Placa con motor	Otro actuador, el cual controlaremos la velocidad y el encendido y apagado a través del serial

	Placa de prototipado	Para el montaje del circuito
---	----------------------	------------------------------

## Nivel de sensado y actuación

Este primer nivel está compuesto por:

- Microntrolador: Arduino mega 2560
- Software: Arduino IDE

El entorno de desarrollo que he empleado es Arduino IDE que es un software utilizado para programar placas basadas en Arduino





La aplicación tiene gran cantidad de ejemplos por tipo de ejercicio, da la opción de descargar e instalar gran cantidad de librerías, permite verificar el código antes de compilarlo/subirlo, posee un monitor para el serial y el muy intuitivo entre sus principales cualidades

- **Librerías**

Las librerías que he utilizado para mi proyecto son:

**Arduino\_FreeRTOS.h**: Biblioteca para administrar tareas en tiempo real en placas Arduino utilizando FreeRTOS.

**Wire.h** - Biblioteca para la comunicación I2C en Arduino.

**DHTesp.h** - Biblioteca para interactuar con sensores de humedad y temperatura de la serie DHT.

**Servo.h** - Biblioteca para controlar los servomotores en el proyecto (servomotor y la helice)

- **Máquina de estado**

Para comprobar las tramas recibidas y en caso de fallo mandar un error [E].

Las tramas con la información de los sensores seguirán un formato:

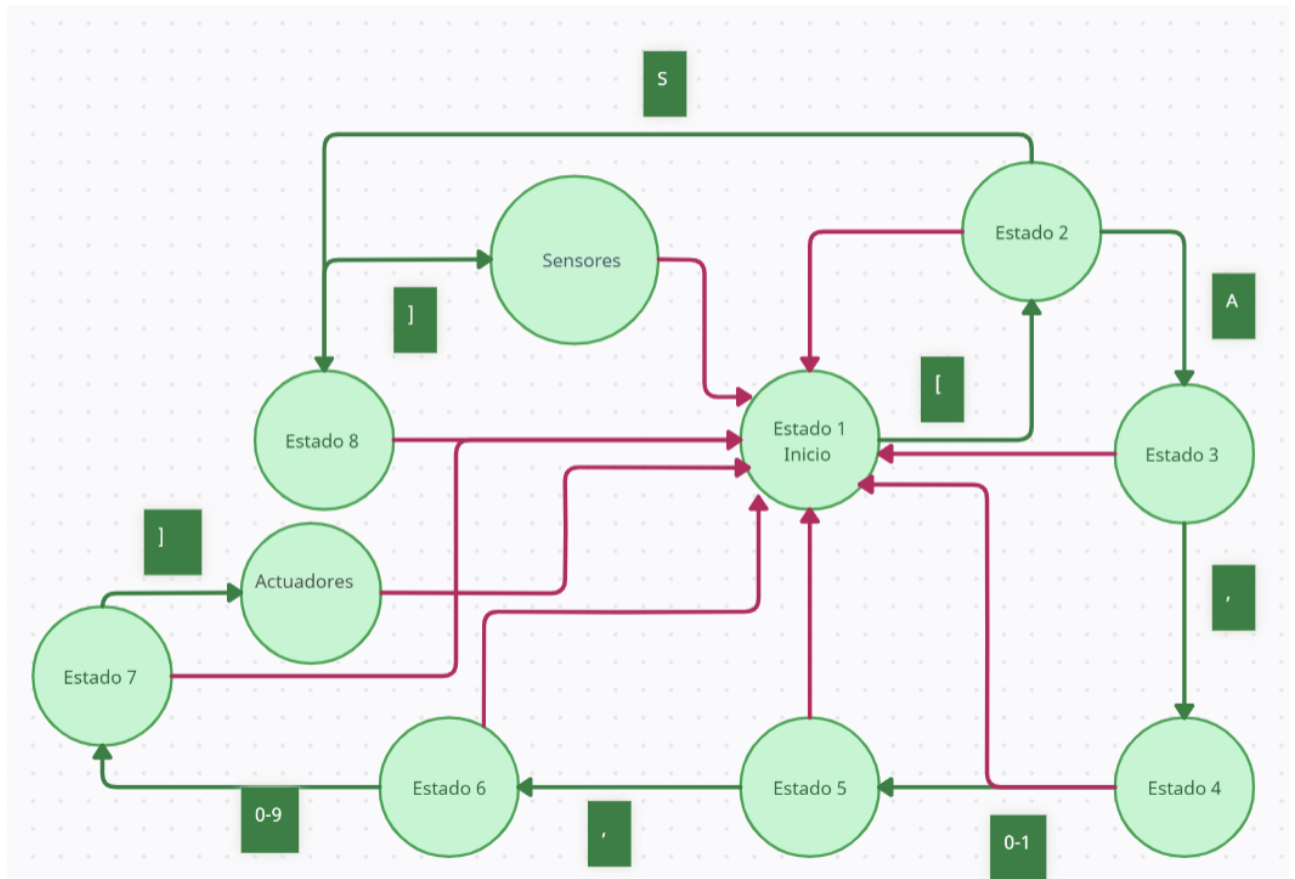
**“["+ "0" + "LDR" + "Humedad" + " Temperatura" + " Checksum" +"]”**

La máquina ira procesando los comandos que reciba por el serial y comprobando carácter a carácter y si cumple los requisitos de la máquina de estados.

Si la trama es [S] devolverá el formato de trama anunciado anteriormente y si tiene la estructura [A,y,x].

siendo  $y$  un valor entre 0 y 1, siendo 0 para mover el servomotor y 1 la hélice y la  $x$  un valor entre 0 y 9 indicando el movimiento/velocidad del actuador correspondiente.

El diagrama de la máquina de estados quedaría:



- **Métodos, tareas y semáforos**

Para empezar, tendremos que definir los baudios ya que deberán ser los mismos en todas las partes del proyecto (9600 en mi caso) pero puede ir hasta los 115200

En mi trabajo leo de dos sensores distintos, por una parte, tenemos el sensor de temperatura LDR y por otra el DHT22 de los que obtendremos la humedad y la temperatura



Defino la configuración inicial en el **Setup** (baudios, inicializo, configuro los sensores, inicio y defino los pines que son los que van a controlar los dispositivos). También creo los dos semáforos binarios que voy a utilizar con el método **xSemaphoreCreateBinary** los cuales necesitaré para gestionar y sincronizar el acceso a los recursos.

También creare las cuatro tareas con **xTaskCreate** con algunos atributos como el nombre y la prioridad, quedando así las 4 tareas en:

- **SendSerial:** Envía los datos desde la cola
- **MoveServo:** Mueve los actuadores
- **ReadSensors:** Lectura de los sensores
- **ReadSerial:** Gestiona y procesa los comandos entrantes

También creo una cola, la cual permitirá contener hasta 30 elementos de tipo char ya que añadiré los valores carácter a carácter.

Para la lectura de sensores llamo a los sensores en tres métodos distintos pero la estructura es la misma:

Leo el valor del pin que corresponda, defino un buffer para añadir los datos, ya que convierto los valores en una cadena de caracteres mediante **dtostrf**, itero en un **for**, compruebo que no es un espacio en blanco y lo mando a la **cola**, finalmente añado un pequeño delay

Para el **checksum** lo que he hecho ha sido comprobar los valores de los sensores, es decir, del LDR y el DHT(temperatura y humedad) y sin son valores validos(LDR  $\geq 0$  y la temperatura y humedad  $> 0$  ) devuelve el checksum un **1** y si no un **0**.

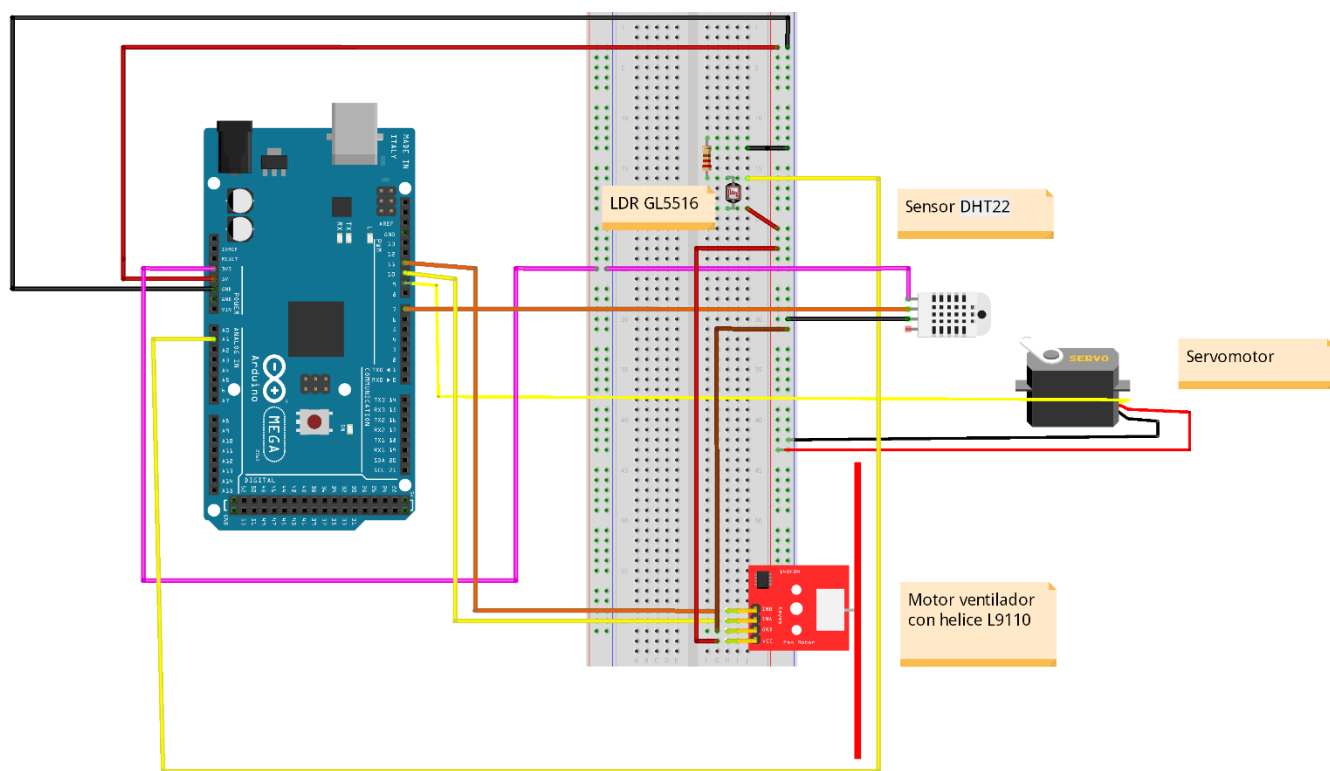
Para la lectura de sensores, solicito el semáforo con **xSemaphoreTake** y empiezo a mandarle datos a la cola siguiendo el formato comentado anteriormente:

**“[“+ “0” + “LDR” + “Humedad” +” Temperatura” +” Checksum” +”]”**

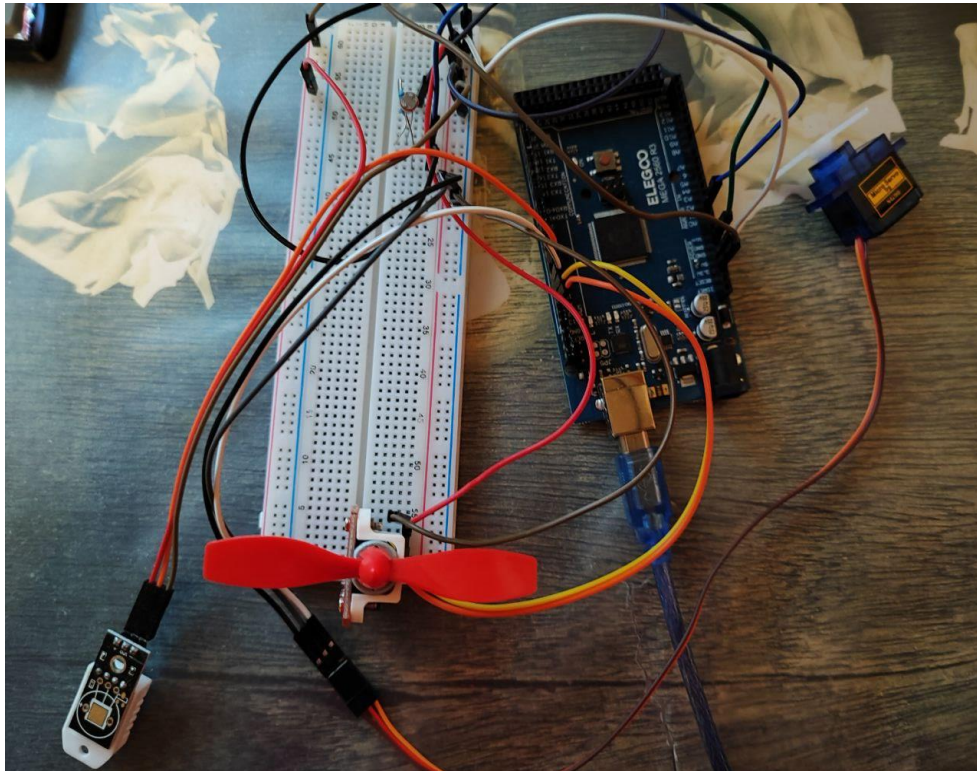
Añadiendo un pequeño delay para evitar conflictos.

Para **MoveServo** llamo solicito el semáforo y compruebo si el valor recibido es un 0 o 1 en la trama, ya que el 0 mueve el servomotor y el 1 la hélice, y comprobamos el segundo valor que indica el movimiento/velocidad que lo calculo en el caso del movimiento con una operación matemática y en el de la velocidad un mapeo

## Diseño



## Trabajo



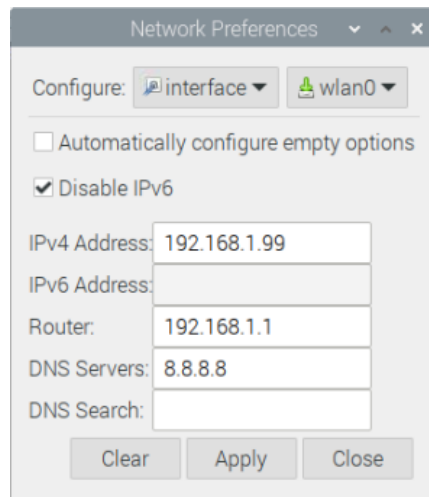
## Pasarela IoT

Para la pasarela IoT usaré la raspberry 3B+ con una tarjeta microSD de 128 gbs, le he instalado el sistema operativo Raspbian junto el paquete que viene en la página que incluye multitud de programas, además de otros como MariaDB. Node..

El apartado de la pasarela lo he dividido en:

### Configuración:

Para poder acceder a la raspberry remotamente deberemos definir una ip especifica y un servidor dns para poder acceder desde nuestra red



También deberemos asegurarnos de tener habilitados el VNC y el SSH



## Scripts de autoarranque

Como uno de los requisitos se pide que hagamos un script de autoarranque, para ello necesitamos crear un archivo. service en lib/systemd/system/

Tendremos que definir algunos campos, algunos de los más relevantes son:

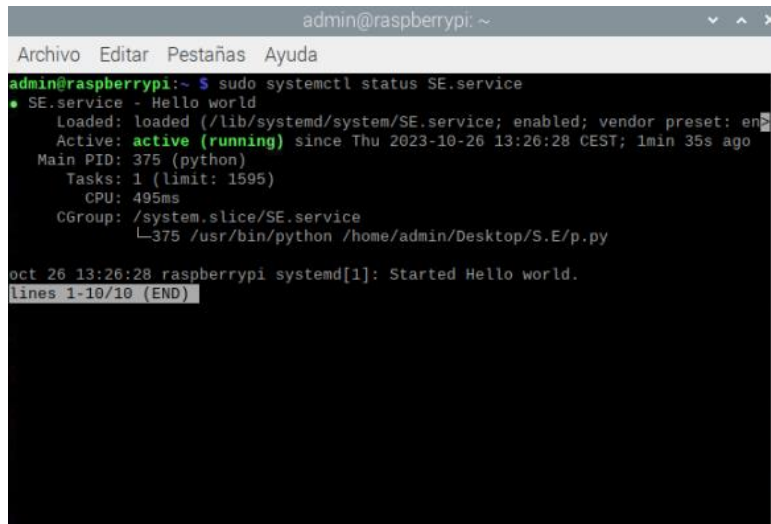
**After=multi user.target:** Que especifica que el servicio debería iniciarse después de "multi-user.target"

**ExecStart=/usr/bin/python /home/admin/Desktop/S.E/p.py** la orden/script que se ejecutará y donde

**Restart-on-abort** indicando que se reinicie en caso de aborto inesperado

**WantedBy=multi-user.target** Características que requiere el servicio

Pudiéndose comprobar con el comando: **sudo systemctl status SE.service** y obteniendo:



```

admin@raspberrypi: ~
Archivo Editar Pestañas Ayuda
admin@raspberrypi:~$ sudo systemctl status SE.service
● SE.service - Hello world
   Loaded: loaded (/lib/systemd/system/SE.service; enabled; vendor preset: en
   Active: active (running) since Thu 2023-10-26 13:26:28 CEST; 1min 35s ago
     Main PID: 375 (python)
       Tasks: 1 (limit: 1595)
          CPU: 495ms
      CGroup: /system.slice/SE.service
              └─375 /usr/bin/python /home/admin/Desktop/S.E/p.py

oct 26 13:26:28 raspberrypi systemd[1]: Started Hello world.
lines 1-10/10 (END)
  
```

## Base de datos

Como último apartado de la configuración y como se pide en el trabajo, deberemos almacenar la información en local en un base de datos, para ello utilizaremos MariaDB

A la cual podremos acceder con el comando **sudo mariadb**;

Tendremos que crear una tabla para almacenar los datos:

**create database db2;**

podremos observar si se ha creado con: **Show databases;**

y la usaremos con: **Use db2;**

Finalmente necesitaremos una tabla, la cual estará compuesta por:

Un ID que será un entero que se vaya incrementando.

La **fecha** de tipo date

Y tres tipos INT que serán respectivamente el **ldr**, **temperatura** y **humedad**

```
Database changed
MariaDB [db2]> describe tabla2;
```

Field	Type	Null	Key	Default	Extra
ID	int(11)	NO	PRI	NULL	auto_increment
Fecha	date	NO		curdate()	
ldr	int(11)	YES		NULL	
temperatura	int(11)	YES		NULL	
humedad	int(11)	YES		NULL	

### Comunicación Serie y almacenamiento de los datos:

Para la comunicación serie con nuestra estación usaremos un script programado en Python usando Thonny como entorno de programación

En el script haremos comunicación con el puerto serie a través de una variable que usa la función serial, donde especificaremos el **puerto** ('/dev/ttyACM0') , la **velocidad en baudios** (que debe ser la misma que la de la placa 9600 en nuestro caso) y un timeout.

Le metemos un tiempo de setup de 5 segundos con **time.sleep(5)**

Luego tendremos que definir una función para configurar la base de datos donde vamos a guardar la información: **host,user ,nombre de la base de datos..**

Cuando solicitamos la lectura de los sensores y recibimos la información, recibiremos un array de bytes por lo que tenemos que decodificarla con UTF-8

Finalmente cortaremos la cadena eliminando información irrelevante y mostrando por consola los valores obtenidos

```
print("El LDR "+ separadas[0]+" La temperatura "+separadas[1]+" La
humedad "+separadas[2]+" El checksum "+separadas[3]).
```



También como especifican es los ejercicios he definido un umbral del ldr para que si lo supera mande una petición para mover el servo.

Finalmente compruebo el checksum y si es correcto mediante una query añado los valores a la base de datos:

```
query="INSERT INTO tabla2 (ldr,temperatura,humedad) VALUES ('%s" %
separadas[0]
query += ", '%s'" % separadas[1]
query += ", '%s'" % separadas[2]
```

## Servicio en la nube

Para esta última parte utilizaremos el servicio en la nube para acceder a la información que tenemos en nuestra base de datos y como recomiendan en el enunciado borraré las entradas leídas para evitar duplicar datos.

Para controlar la raspberry desde nuestro ordenador configuraremos una IP específica y accederemos a ella con el programa **REALVCN Viewer**

Para acceder a la información almacenada en la base de datos, utilizaremos un script de node que estará compuesto por:

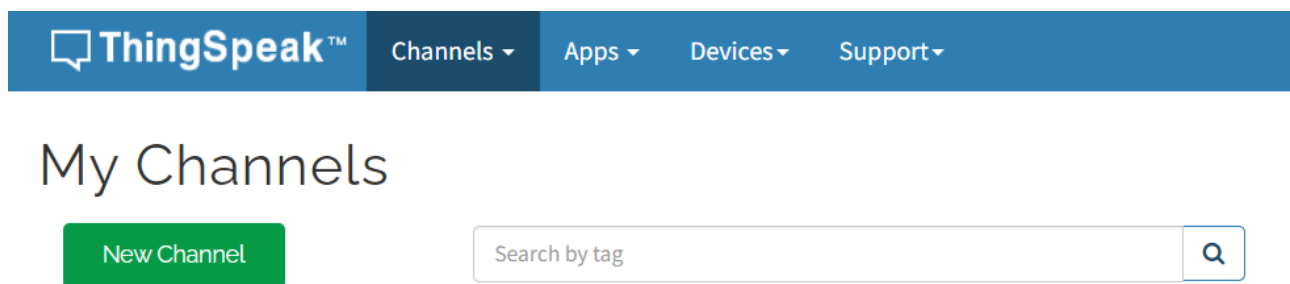
- Librerías que en este caso son requests y mariadb
- La información de la base de datos para acceder a ella:  
**host,user,password y database**
- Una conexión con la base de datos para extraer datos por ID usando una query

```
'SELECT ID, ldr, temperatura, humedad FROM tabla2 ORDER BY ID
ASC LIMIT 1';
```

- Un método para subir a thingSpeak ese dato
- Un método para borrar de la base de datos el valor subido a través de una query

```
`DELETE FROM tabla2 WHERE ID = ${ID}`;
```

El servicio que he escogido es ThingSpeak, muestras gráficas, funciona con una API y es gratuito hasta 3 canales con lo que puedo mostrar los valores de mi base datos sin problemas



Necesitaremos definir el nombre de las gráficas a mostrar, que en mi caso son LDR, Temperatura y Humedad y generará el grafico con los valores que vaya recibiendo de la pasarela a través del script, es importante configurar nuestro script con la API Key

Una vez hecho esto simplemente ejecutaremos en la pasarela el script y cada 30 segundos ira recibiendo información de la base de datos y los mostrará en las gráficas. El script está configurado para ir mostrando por pantalla la información y el error si ocurre

