Universidad de Alicante

Escuela Politecnica Superior

Entregable 2

Autor:
David Carbonell Pastor

2022

Índice

1. Uso de la plataforma IoT Ubidots

1.1. Evidencias del trabajo realizado

Se ha utilizado la plataforma IoT Ubidots a través de Node-Red como se puede ver en la figura 1 de esta manera se simplifica el apartado de la programación ya que de otra manera se ha de realizar mediante codigo python para lo que pretendemos implementar.

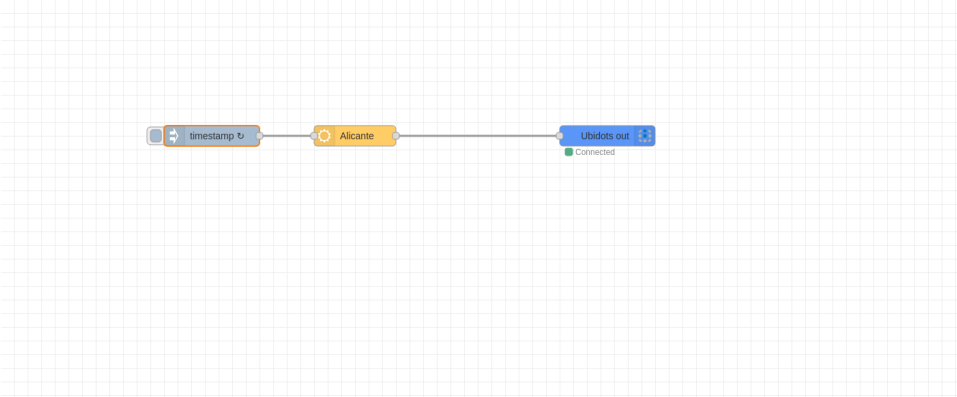

Figura 1: Conexion de ubidots con nodered

Enviamos la información del nodo de openweather al nodo de ubidots en la pagina de ubidots en el apartado devices nos aparecen los parametros que nos proporciona en formato json de una manera mas visual donde podemos ver los distintos datos que tenemos.

Con los datos proporcionados por el nodo de openweather se ha realizado un dashboard simple 2 el cual presenta una menor complejidad y mayores funcionalidades que los dashboard creados con node-red en el anterior entregable.

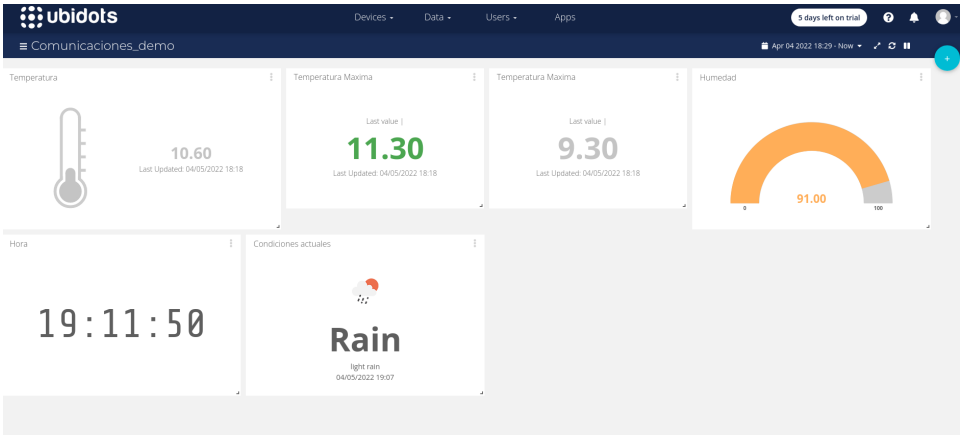

Figura 2: Dashboard simple con datos de openweather

Enlace al dashboard creado `https://industrial.ubidots.com/app/dashboards/public/dashboard/59mHN0d4o7UrQGhN3BC5KWEvK0hM2c7FXGd-t9ClxBk`

1.2. Ampliacion de conocimientos

Se va a realizar una lectura y escritura de datos obtenidos mediante el ESP-32 y la plataforma ubidots todo el codigo se encuentra adjunto en la practica y en el github para su consulta.

1.2.1. Enviar valores de del ESP-32 a Ubidots

Tanto para mandar como para recibir los valores desde el ESP-32 a Ubidots vamos a utlizar la librería `UbidotsEsp32Mqtt.h` desde el framework de arduino. El protocolo que utilizamos tanto para enviar como recibir los datos es MQTT.

Para mandar los datos a la plataforma Ubidots utlizamos los metodos `ubidots.add` y `ubidots.publish` donde introducimos el nombre del dispositivo y las variables que hemos leido previamente mediante las funciones como puede ser en nuestro caso `analogRead()` o `digitalRead()`.

Para saber si la conexion se ha realizado con exito debe aparecer el mensaje de la siguiente figura:

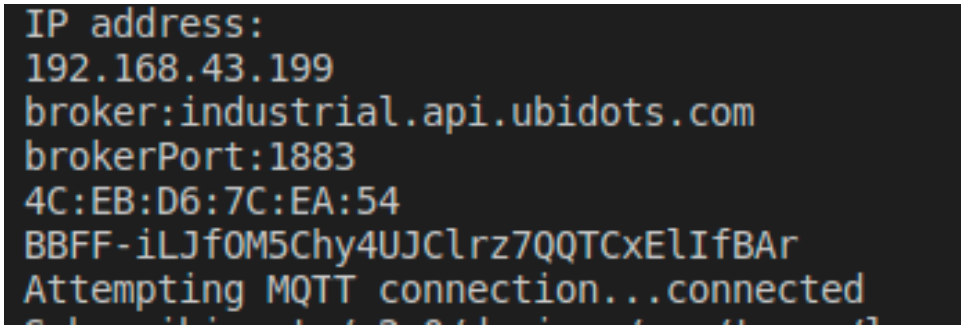

Figura 3: Mensaje debe aparecer en el monitor serie del ESP32

Los datos mandados a la Ubidots aparecen de la siguiente forma:

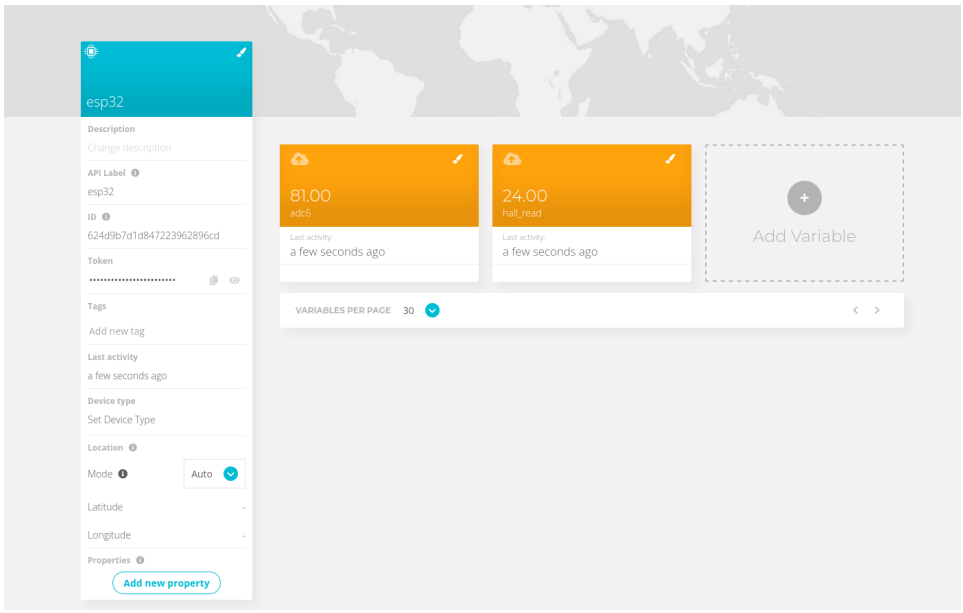

Figura 4: Datos del ADC6 y hall sensor en Ubidots

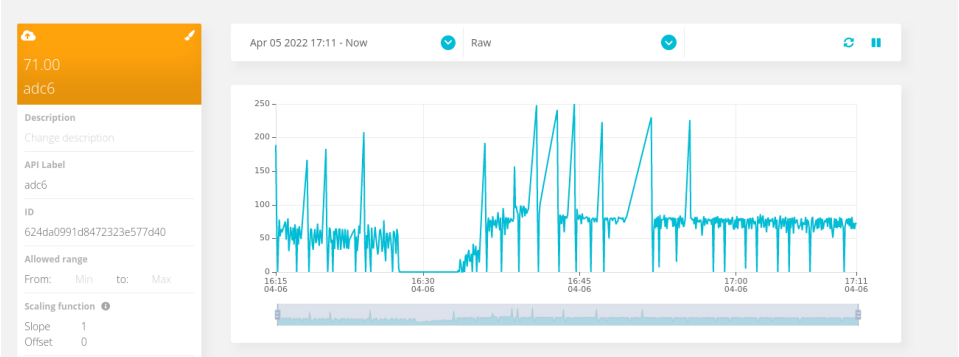

Figura 5: Datos proporcionados por el ADC6

1.2.2. Recibir valores de Ubidots a ESP-32

Para recibir los valores que mandamos a través de node-red a ubidots es un proceso muy similar al de mandar los valores en el codigo solamente añadimos la funcion `ubidots.subscribeLastValue` donde se indica el dispotivo al que nos subscribimos ya la variable que queremos leer. Esto no proporciona los valores de la variable por terminal de la siguiente forma:

```
Subscribing to/v2.0/devices/owo/tempc/lv
Message arrived [/v2.0/devices/owo/tempc/lv] 16.6
publishing to TOPIC:
```

Figura 6: Temperatura leida por el ESP32 a través de Ubidots

2. Uso de relés inteligentes

2.1. Configuración de Shelly 1

Shelly es uno de los dispositivos inteligentes que permiten el control por Wifi de la iluminación o los electrodomesticos. Shelly usa la red wifi para la interconexión de sus dispositivos, esto hace que su configuración sea muy fácil y rapida. A continuación 7 se muestra un diagrama de como realizar su conexionado a la red.

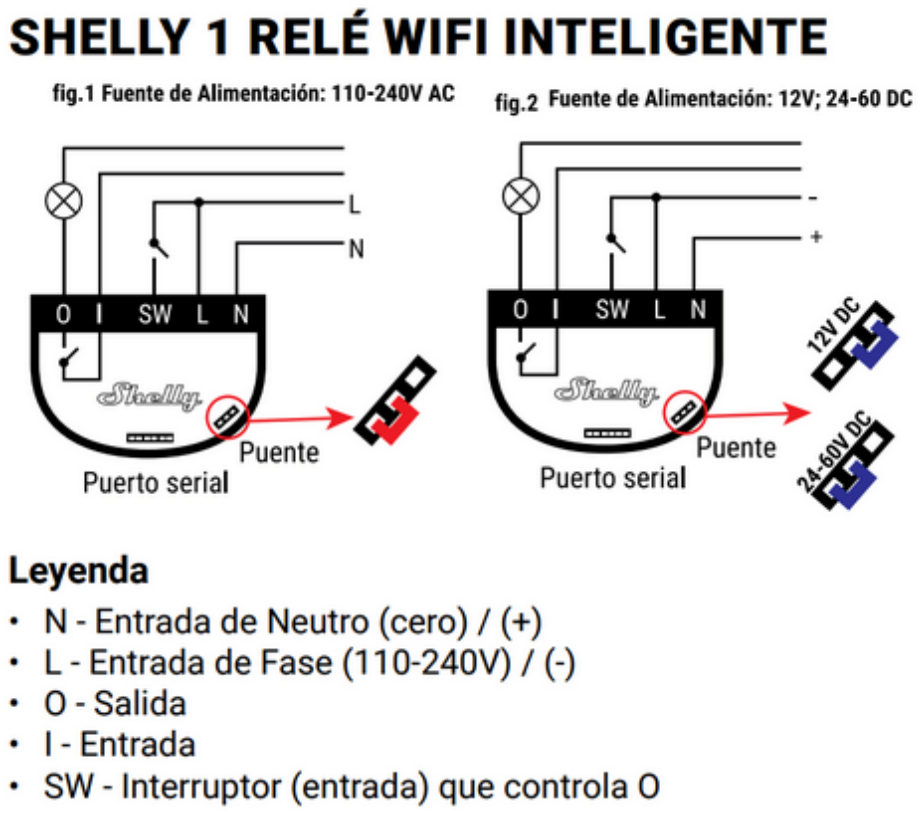

Figura 7: Esquema de conexión de Shelly 1 a la red electrica

Despues de haber realizado la instalación del dispositivo, el mismo crea una red wifi a la cual debes conectarte para realizar la configuración.

- Primero tienes que acceder la web de configuración del dispositivo introduciendo la ip `192.168.33.1` en el navegador.
- Dentro de la web se puede comprobar si es el shelly que queremos configurar apagando y encendiendo el dispositivo.
- En el apartado *Internet & Security* se selecciona *Wifi-Mode-Client* donde se introduce nuestra red wifi que queremos que se conecte el dispositivo.
- También se puede realizar la configuración desde la aplicación *Shelly Cloud*.

2.2. Relés con node-red mediante http request

A través de node-red se pueden controlar los diversos relés que se encuentran conectados a la red, a través del nodo `httprequest`.

Para ver toda la información 8 que nos proporciona el relé utilizamos el nodo `httprequest` con la siguiente direccion `http://192.168.10.110/emeter/0`

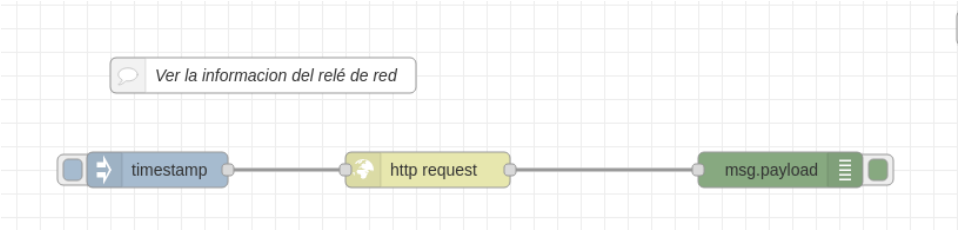

Figura 8: Esquema node-red para ver informacón del relé

Para cambiar el estado 9 al que se encuentra el relé utilizamos el nodo `httprequest` con la siguiente direccion `http://192.168.10.109/relay/0?turn=on` para cambiar el estado cambiamos el apartado turn a on o off.

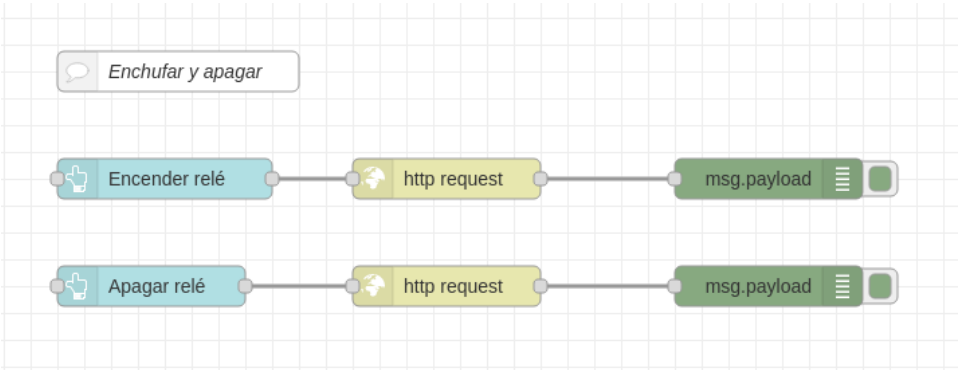

Figura 9: Esquema node-red para cambiar el estado del relé

Finalmente se realizó un dashboard 11 en el cual se obtienen los parametros proporcionados por el relé como son la potencia y el voltaje los cuales se leen emeter como se ha explicado previamente, para leer solamente los parametros que deseamos se ha utilizado un nodo function donde extraemos del *json* esos parametros. También se ha realizado mediante un nodo button que permite encender o apagar el relé de una manera mas sencilla.

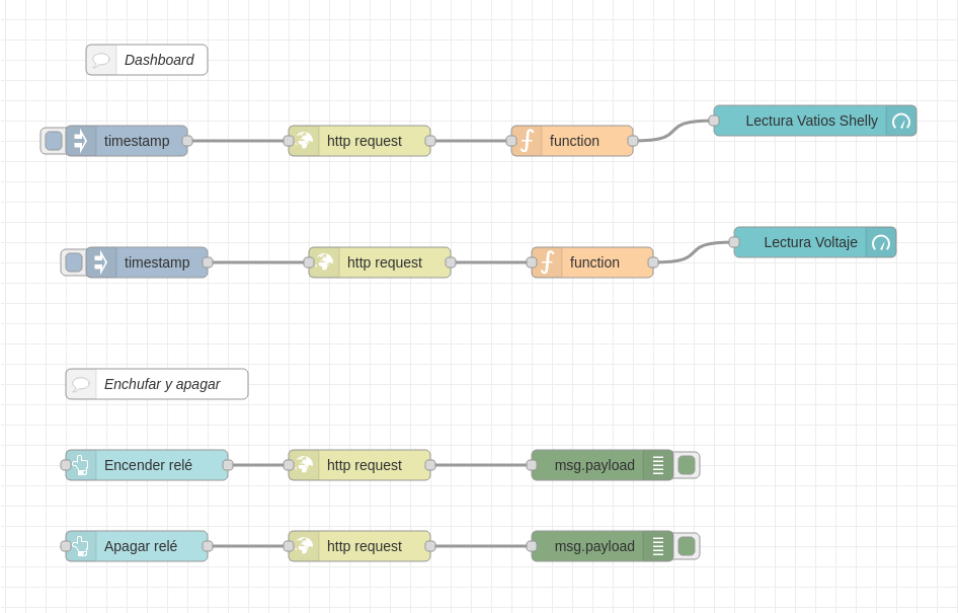

Figura 10: Nodos para realizar el dashboard

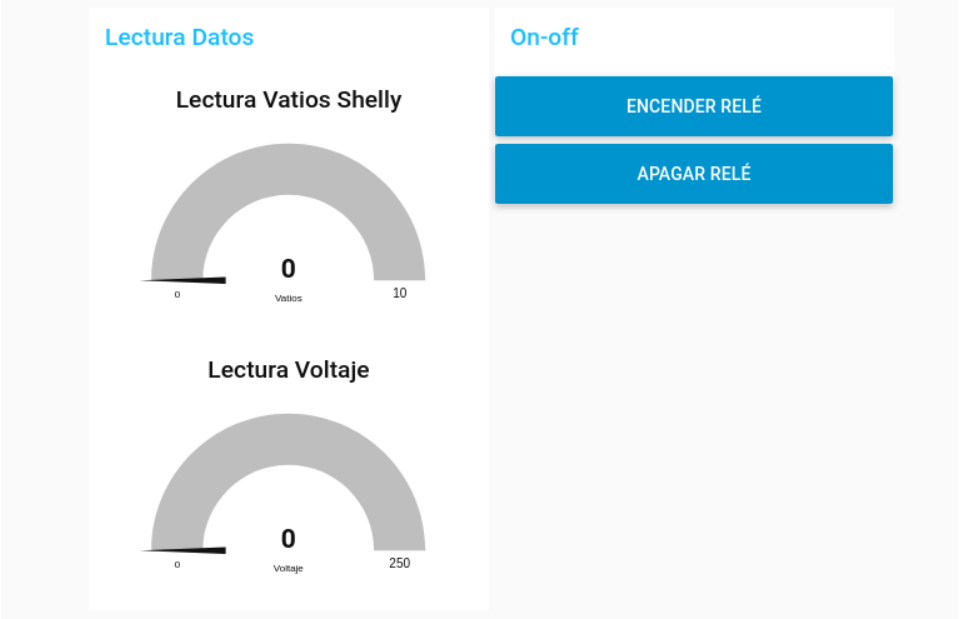

Figura 11: Dashboard de datos y encender el relé

De manera extra se probaron también otras funciones de los relés que se encontraban en el laboratorio. Primero se probó el encender el relé durante 10 segundos 12 mediante la siguiente dirección `http://192.168.10.110/relay/0?turn=on&timer=10` donde en el apartado timer se especifica el tiempo en segundos que va permanecer encendido el relé.También se probó con el relé que controla la persiana 13 donde se probó a abrirla completamente con la dirección `http://192.168.10.110/roller/0?go=open`.Y finalmente a abrir la persiana en un 30 porciento 14 mediante la siguiente dirección `http://192.168.10.110/roller/0?go=to_pos&roller_pos=30`

Figura 12: Encender relé durante 10 segundos

Figura 13: Encender el relé de la persiana

Figura 14: Encendemos el relé de la persiana en 30 porciento

2.3. Relés con node-red mediante MQTT

Como http request presenta limitaciones a la hora de controlar nodos ya que este protocolo al ser una petición http tiende mas a caerse o a fallar.Utilizamos el protocolo mqtt que acepta mayor numero de peticiones a la vez y es mas fiable. Para ello al igual que en http request nos conectamos a la misma red wifi en la que se encuentran los relés pero a diferencia de http request ahora nos conectamos al servidor mqtt que se ha creado en dicha red. A continuación se muestra un ejemplo de mandar un mensaje y recibirlo mediante mqtt:

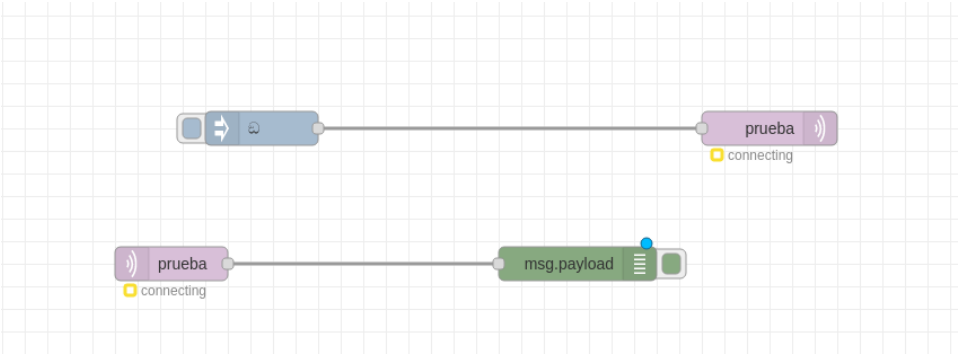

Figura 15: Enviar y recibir mensajes mediante mqtt

Para obtener la información de los relés mediante *MQTT* se ha utilizado un nodo `mqtt` que nos permite recibir los mensajes de los relés con el topic `shellies/#`

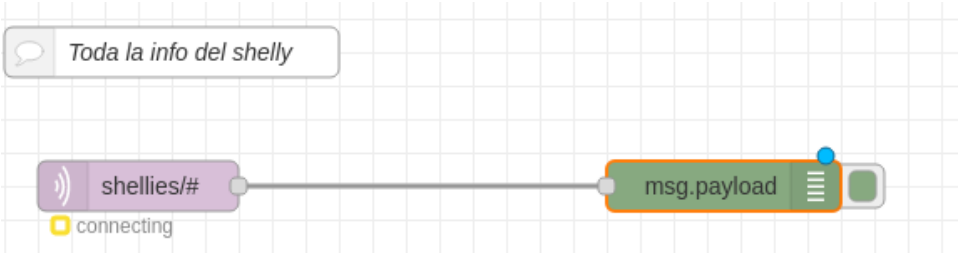

Figura 16: Obtener toda la información del relé mediante mqtt

Para obtener otro parametros 17 como pueden ser el voltaje o la potencia, se cambia el topic del nodo mqtt. Por ejemplo en el caso de la potencia y el voltaje los topics son los siguientes `shellies/shellyem1/emeter/0/voltage` `shellies/shellyem1/emeter/0/power`

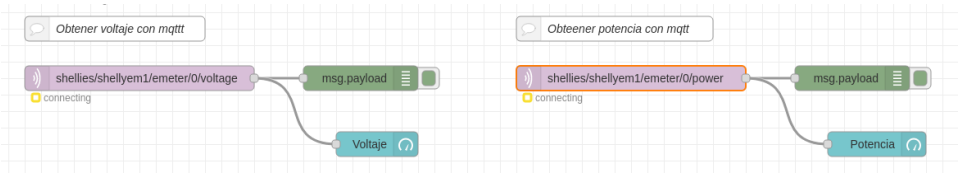

Figura 17: Obtener el voltaje y la potencia del relé mediante mqtt

También se pueden apagar y encender los relés mediante mqtt al igual que hemos hecho con http request, pero a la hora del mundo real mqtt es mucho mas interesante que http request ya que aguanta un mayor numero de llamadas y es mas fiable evitando que el relé falle o se caiga la conexión como pudimos ver de manera experimental en el laboratorio con mqtt y toda la gente realizando peticiones funcionaba de una manera correcta mientras que con http request presentaba alguna caida.

A continuación se muestra un ejemplo de como encender y apagar el relé mediante mqtt:

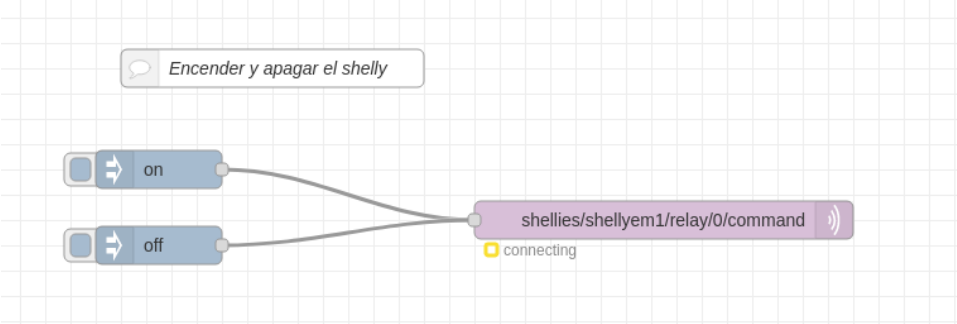

Figura 18: Encender y apagar relé mediante mqtt

3. Ampliación MQTT: Conectar ESP32 a broker MQTT

Para esta aplicacion vamos a utlizar un broker gratuito llamado *EMQX* que nos permite conectar nuestra aplicación a un broker mqtt. La propia web nos ofrece un interfaz 19 en la cual podemos ver los diversos mensajes que nos llega o enviamos a través de MQTT. Se ha realizado una prueba mediante node-red para probar las funcionalidades que presenta, en este caso se ha probado mandar mensajes como aparecen a continuación:

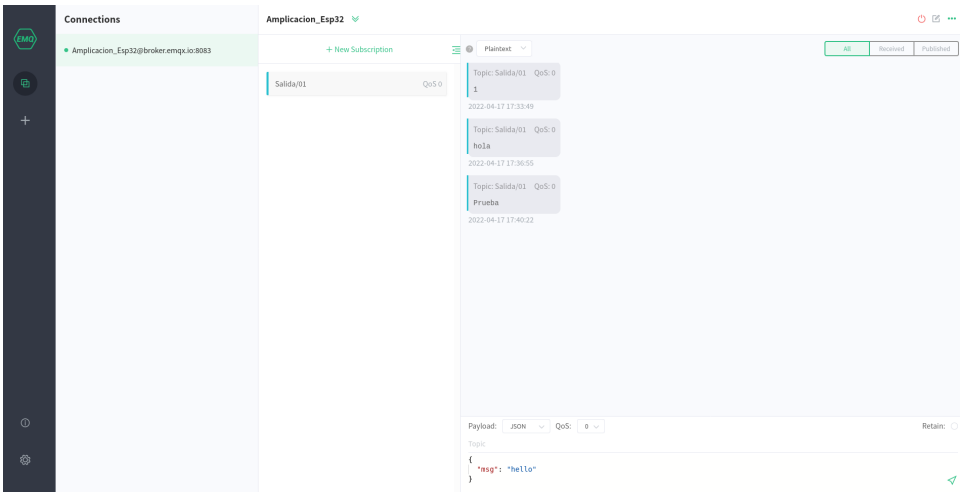

Figura 19: Prueba de la interfaz que nos ofrece el broker MQTT

La aplicacion que vamos a hacer es leer el valor de adc6 y mandarlo al broker mqtt, para ello hacemos uso de las librerías `Wifi.h` y `PubSubClient.h`

que nos permiten conectarnos con el ESP-32 a través de wifi al broker.

Finalmente obtenemos el siguiente resultado por puerto serie 20 y este resultado en la web del broker mqtt 21 no coincide el valor del adc al ser un broker publico le llegan los valores con cierto retraso y el que aparece es el de una prueba anterior.

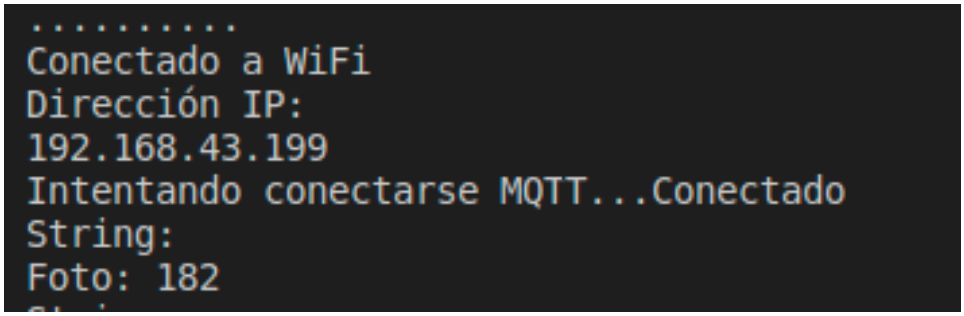

Figura 20: Informacion que nos muestra el esp32 en el puerto serie

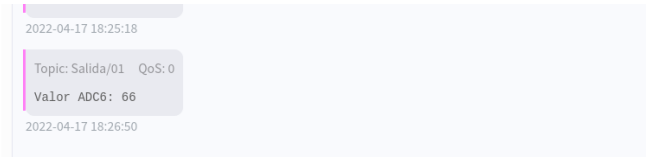

Figura 21: Valor del adc en el broker mqtt

4. Consulta del codigo de la practica

Para consultar el codigo de la practica se encuentra adjuntado a la misma y también en el github `https://github.com/carbonto/Comunicaciones-2`

5. Conclusiones

En el desarrollo de esta practica se ha conseguido aprender a utliziar plataformas IoT como en nuestro caso ubidots.Además de profundizar en el uso de relés inteligentes y sus diversas funcionalidad y protocolos de comunicacion. Finalmente se ha aprendido a utilizar el protocolo MQTT para comunicarnos con otros dispotivos y sensores de la red.

6. Bibliografia

https://help.ubidots.com/en/articles/748067-connect-an-esp32-devkitc-to-ubidots-ov
https://industrial.ubidots.com
https://www.hackster.io/shahizat005/building-an-esp32-based-iot-weather-station-with-
https://www.emqx.com/en/mqtt/public-mqtt5-broker