Pract_2: Conexión con servicios remotos

Alberto Zafra Navarro

Cabe destacar que todos los archivos comentados durante la memoria se encuentran en la carpeta **pract_2** del repositorio **comm_pract** del usuario de GitHub denominado **albertozafra7**. El enlace a la carpeta es: https://github.com/albertozafra7/comm_pract/tree/master/pract_2.

Tarea 1 - Conexión de Ubidots con NodeRed

Ubidots es una plataforma que permite realizar la conexión IoT con los distintos servicios de terceros, mediante un sistema de desarrollo que permite realizar una aplicación para interactuar con dichos servicios de forma remota e interactiva.

Durante la primera tarea del entregable 2 se realizaron diversas pruebas con la interfaz de Ubidots, la cual es una plataforma de IoT que empodera a innovadores e industrias enfocándose en facilitar el desarrollo y la escalabilidad de distintos prototipos para la producción. Es decir, Ubidots facilita el envío de datos a la nube desde cualquier dispositivo conectado a internet, permitiendo visualizar y procesar los datos enviados en tiempo real, siendo posible de esta forma configurar alertas y acciones con dichos datos.

Para el uso de esta aplicación fue necesario introducir los nodos de Ubidots en la aplicación de Node-Red esto se realizó de una forma similar a la realizada en la práctica anterior, seleccionando el paquete mostrado en la Figura 1 mediante la opción de "Manage palets".

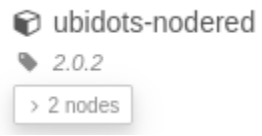


Figura 1: Paquete de nodos de Ubidots.

Cabe destacar que a lo largo de esta tarea se han utilizado nuevamente los servicios web de Openweather para obtener los datos del tiempo de Albatera. Durante este proceso al igual que en la práctica anterior la API key utilizada durante toda la práctica ha sido `"69019a20ac0d9cf59895844430362f0b"`.

Una vez realizada la configuración inicial de Node-Red se generó una estructura de bloques de prueba que recibía todos los datos del tiempo de la localidad de Albatera, utilizando un bloque de OpenWeather correctamente configurado con la API key mencionada anteriormente, para más tarde realizar el envio de estos datos a la plataforma de Ubidots. Siendo dicha estructura la representada en la Figura 2.

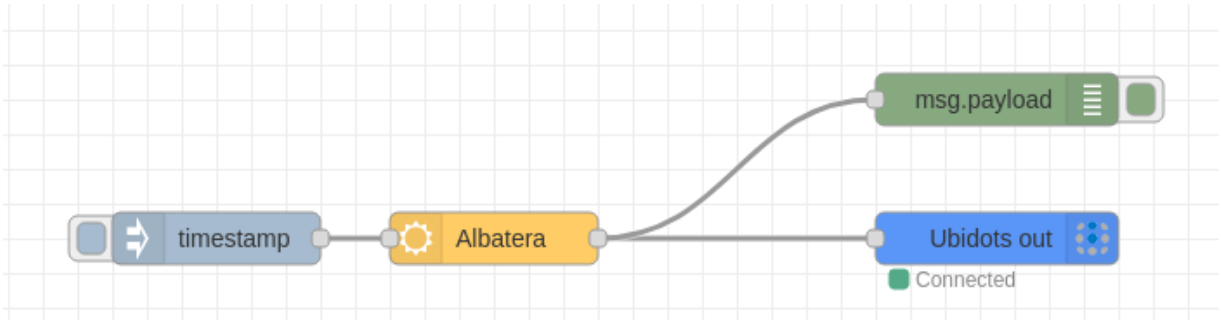


Figura 2: Estructura inicial de conexión entre Node-Red y Ubidots.

No obstante, dicha conexión no hubiese sido posible sin habernos registrado previamente en la página web de Ubidots y habiendo creado un dispositivo en la misma plataforma, en este caso se ha generado un dispositivo denominado como ”Comunicaciones”, con el id `"6229f5fe1d84721ee6f3c688"` y el token `"BBFF-m0oofCkWi0OdddpqgDss3JEyb951h3`. Puesto que como se puede visualizar en la Figura 3 dentro del bloque denominado como ”Ubidots out.es preciso realizar una configuración inicial en la que se debe de indicar el tipo de cuenta que se está utilizando en la web de Ubidots y el token del dispositivo al que se desea conectar.

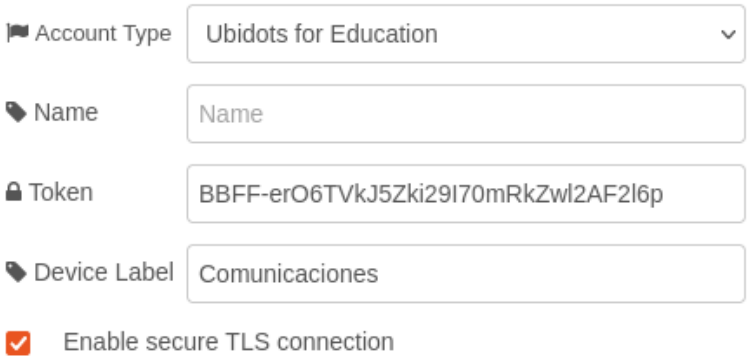


Figura 3: Configuración utilizada en el bloque de Ubidots.

Realizada esta configuración inicial, se puede observar que en el momento que se realice una llamada desde Node-Red a Ubidots mediante la estructura de conexión representada en la anterior figura, la Figura 2, se generará una serie de variables iniciales, las cuales contienen todos los datos obtenidos desde la API de OpenWeather y que se han enviado a nuestro dispositivo en Ubidots. Un ejemplo de estos datos se puede visualizar en la Figura 4.

No obstante, es posible que Ubidots no reciba ningún dato, manteniéndose la ventana vacía, como en la Figura 5, esto es debido a que diversos puertos de la red a la que se encuentra conectado el dispositivo que se encuentra ejecutando Node-Red se encuentren cerrados, como es el caso de la red de ”eduroam”, pues durante la primera sesión de prácticas fue imposible realizar la conexión entre Node-Red y Ubidots por este motivo.

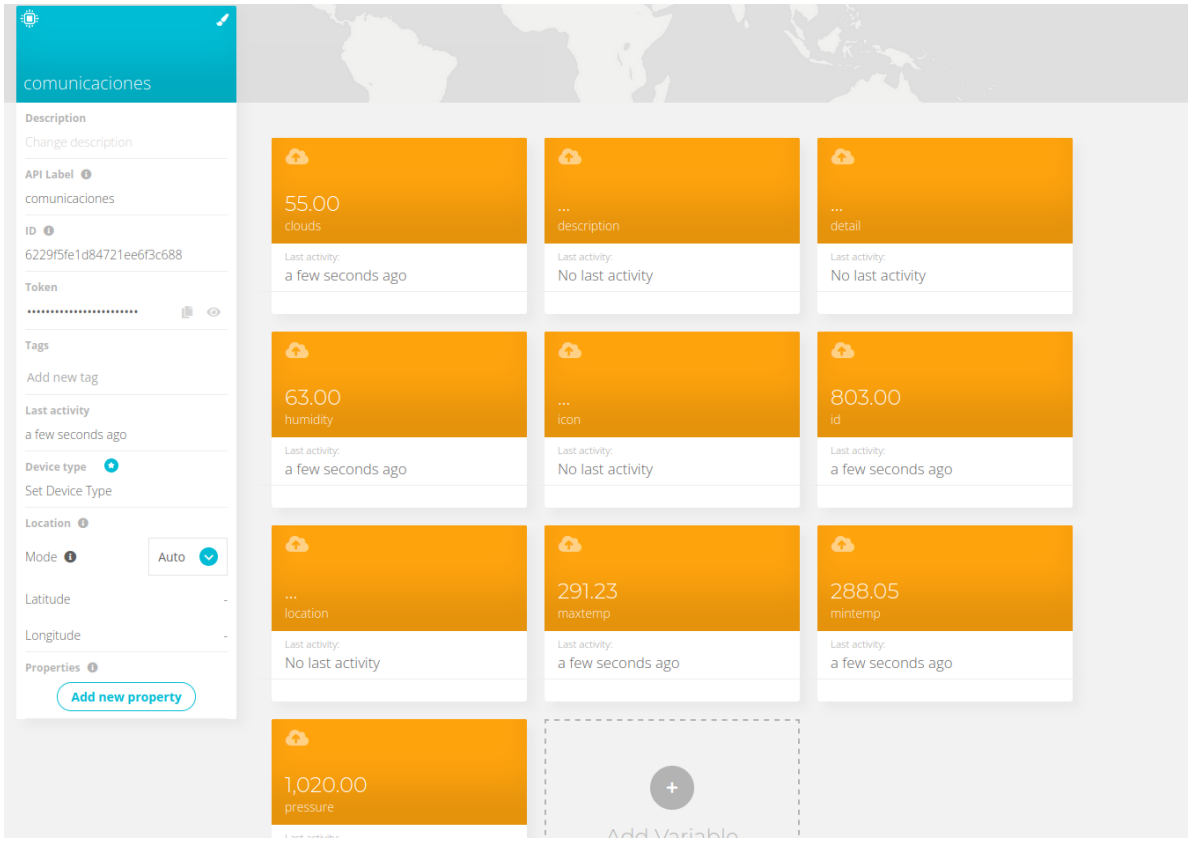


Figura 4: Datos iniciales recogidos por Ubidots.

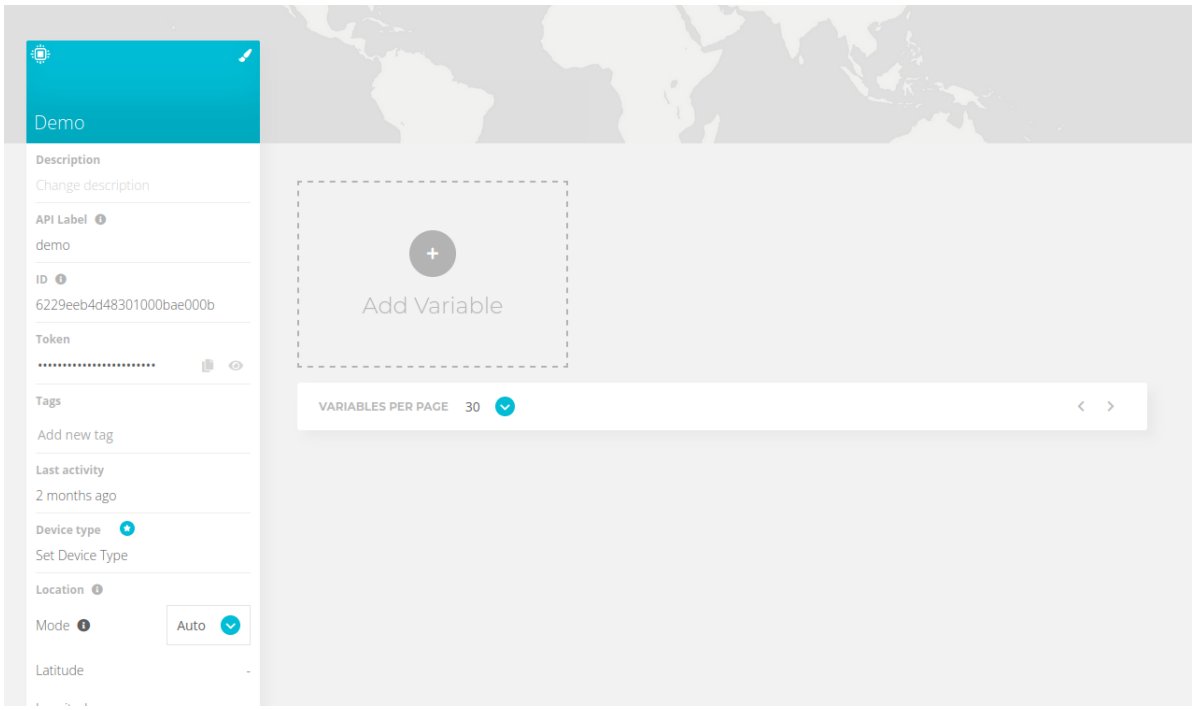
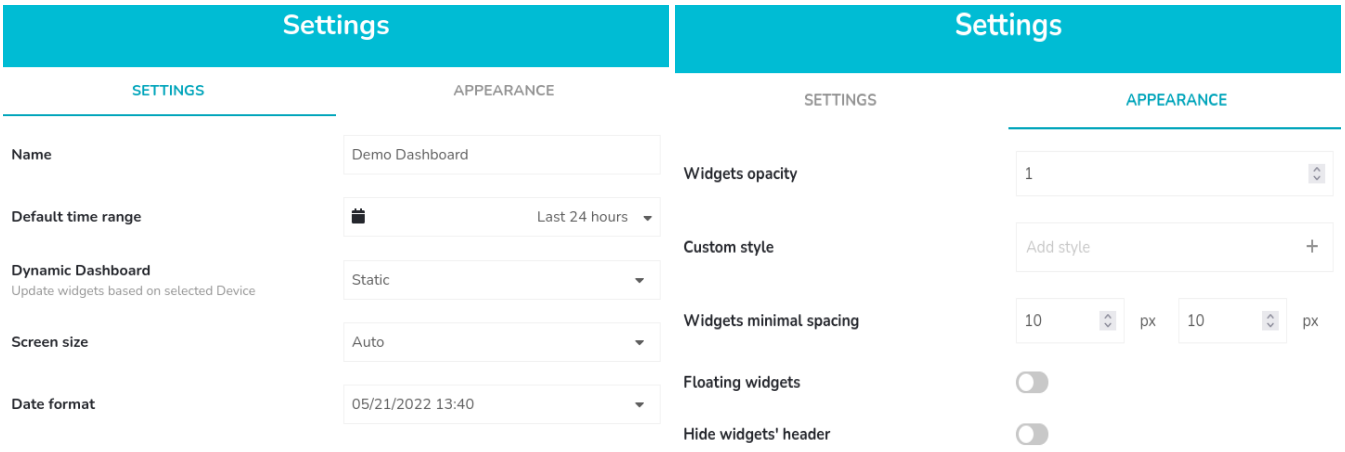


Figura 5: Ventana del dispositivo en Ubidots sin recibir datos de Node-Red.

Por otro lado, el número de variables que puede manejar un dispositivo en la versión gratuita de Ubidots se encuentra limitado a 10 variables, este es un dato que hay que tener en cuenta durante la configuración y la creación de los dispositivos, ya que el proyecto que se desee realizar puede encontrarse limitado por esta característica.

Finalmente, tras obtener las variables de Node-Red es posible generar un Dashboard en el cual se puedan analizar, mostrar y/o configurar acciones dependiendo de los valores de las variables de una forma más sencilla e intuitiva. No obstante, para cuentas gratuitas el uso de Dashboards se encuentra limitado a 3 Dashboards, por lo que se debe de tener cuidado con el número de proyectos que se encuentran ejecutándose simultáneamente en una misma cuenta.

Tras realizar la creación del Dashboard es posible modificar diversos parámetros de este para adaptarlo a la necesidad del usuario tal y como se muestra en la figura 6, sin embargo, en nuestro caso se ha mantenido la configuración inicial.



(a) Normal Settings (b) Appearance Settings

Figura 6: Pestaña de configuración del Dashboard.

Por otro lado, tras realizar la configuración del Dashboard se introdujeron distintos elementos en el mismo, los cuales recogían la información del porcentaje de humedad en Alicante y mostraban de formas distintas los valores de dicha variable, siendo esta representada por un slider, un gauge, una gráfica o una tabla de valores, tal y como se puede visualizar en la Figura 7.

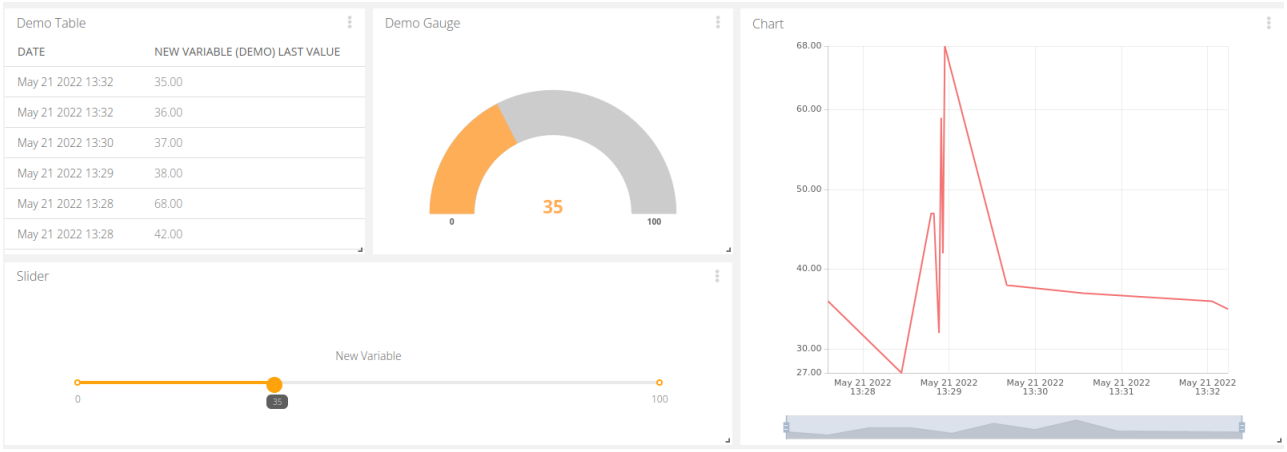


Figura 7: Dashboard con el porcentaje de humedad Alicante.

Para ello, además de introducir los elementos en el Dashboard fue preciso configurarlos para que recibiesen los valores de dicha variable, esto es realizado seleccionando en la pestaña de ”Data” de cada elemento la variable que se desea representar. Por otro lado, si se desease modificar la posición de cada elemento en el Dashboard esto sería posible mediante la conocida técnica de pinchar y arrastrar alrededor del Dashboard que se está modificando.

Creación de un Dashboard con los datos del tiempo de Albatera

Tras experimentar las diversas conexiones que se pueden realizar con Ubidots y con los diversos elementos de los Dashboards de la plataforma, se pensó en expandir esta tarea mediante la implementación de un Dashboard, en el cual se mostrasen los datos de humedad, temperatura, presión atmosférica y nubosidad del municipio de Albatera. Para ello fue preciso modificar los nodos de la estructura previamente implementada en Node-Red para realizar un filtrado de los datos que se deseaban transmitir a la plataforma de Ubidots, añadiendo un bloque de función a la estructura de Node-Red previamente implementada. De esta forma la nueva estructura de Node-Red se quedaría de forma similar a la mostrada en la Figura 8.

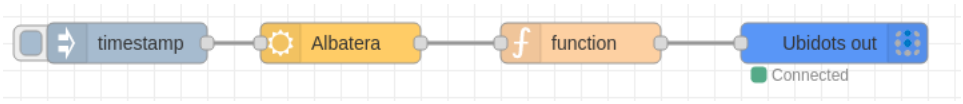


Figura 8: Estructura de Node-Red con filtrado.

Una vez implementada dicha estructura, al iniciar la ejecución del programa es posible de visualizar como en el dispositivo previamente creado en Ubidots se han creado las nuevas variables a utilizar, de forma similar a la de la Figura 9.

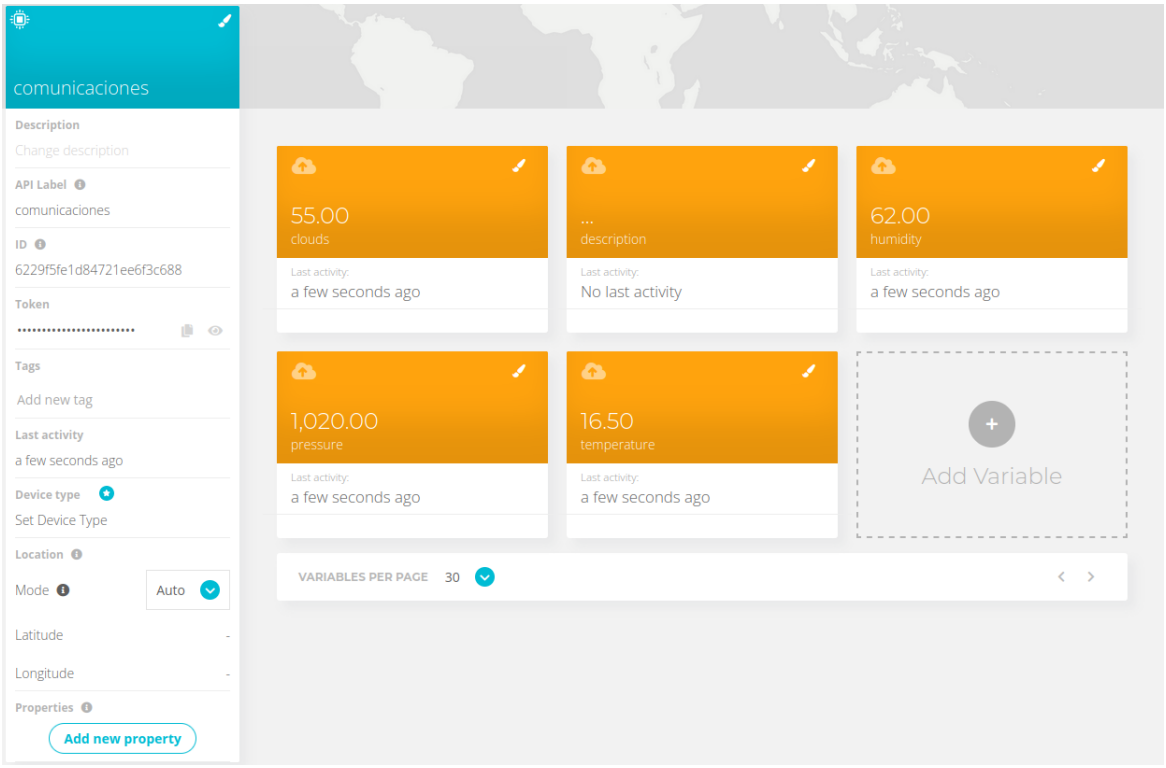


Figura 9: Variables filtradas en el dispositivo de Ubidots.

Por lo que ya solamente restaría la creación de nuestro Dashboard que permita la visualización de dichas variables,

en este caso se ha añadido una foto del municipio de donde se obtiene la información para darle un toque más profesional, quedando el Dashboard de una forma similar a la mostrada en la Figura 10.

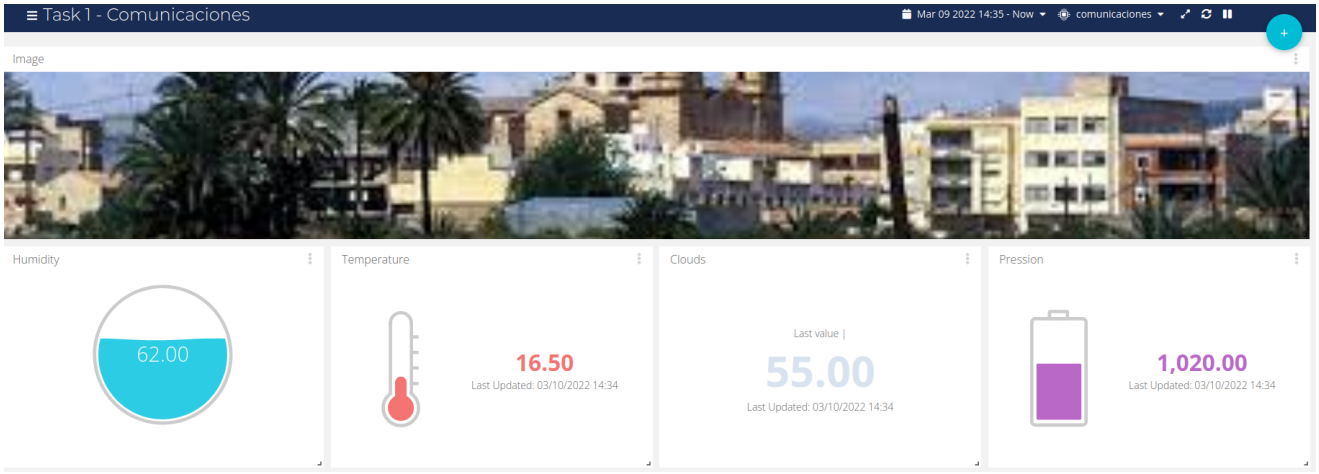


Figura 10: Dashboard final Ubidots.

Tarea 2 - Conexión con dispositivos físicos

Tras trabajar con los distintos tipos de interfaces mediante simulación es posible dar el siguiente paso para trabajar con distintos dispositivos físicos, los cuales muestren un comportamiento real con el que se trabajará en un futuro. En nuestro caso se va a proceder a trabajar con dispositivos conectados a un SHELLY, el cual es un interruptor WiFi que permite el control de dispositivos de forma remota.

Conexión y Configuración de un dispositivo SHELLY

Antes de comenzar a trabajar con el dispositivo es preciso configurar el dispositivo *SHELLY*, para ello debemos de conectarnos al dispositivo mediante la conexión a una red WiFi en la que se encuentre conectado el *SHELLY*, esto puede hacerse conectándose a la red generada por el mismo router, denominada en este caso como se muestra en la Figura 11.

Figura 11: Dashboard final Ubidots.

Una vez conectado a dicho router es posible configurar el dispositivo accediendo a la dirección del dispositivo desde el navegador, accediendo a `192.168.33.1`. Una vez conectado al dispositivo es posible configurar el dispositivo con una red existente en nuestro espacio de trabajo configurando en la sección de *Internet y Seguridad* el nombre de la red a la que deseamos conectar el dispositivo Shelly y la contraseña del mismo, cabe destacar que es preciso establecer una IP estática, puesto que, en caso de reiniciar la red a la que se conecta el dispositivo es posible que la IP varíe en caso de que esta no fuese estática y necesitar configurar nuevamente todos los servicios conectados a dicho dispositivo.

Durante la realización de las prácticas se conectó el dispositivo a la red del laboratorio denominada como `Cudy-081C` y se estableció la dirección IP estática `192.168.10.100`, puesto que la red local del router se encuentra entre las direcciones `192.168.10.1` y `192.168.10.255`. Realizada esta configuración el dispositivo *SHELLY* se conectó

automáticamente a la red local, el siguiente paso es conectar el ordenador mediante el cual se desea operar a la misma red. Para ello se debe de acceder a la red `Cudy-081C` mediante el uso de la contraseña `Comunica2022`.

Trabajo con el SHELLY mediante navegador

Una vez conectado el ordenador con el que se desea operar a la misma red que se encuentra conectado el *SHELLY* es posible acceder a la web del mismo dispositivo poniendo la dirección `192.168.10.110` en el navegado. Cabe destacar que el ancho de banda del dispositivo no es demasiado grande, por lo que si se conectan un número considerable de dispositivos a la vez a dicha dirección se puede producir una denegación de servicios del dispositivo, algo que sucedió durante la sesión de prácticas. Por otro lado, si se accede correctamente a dicha dirección, el navegador deberá de cargar una web similar a la mostrada en la Figura 12. Desde el cual es posible de visualizar e interactuar con los distintos dispositivos que se encuentran conectados al *SHELLY*.

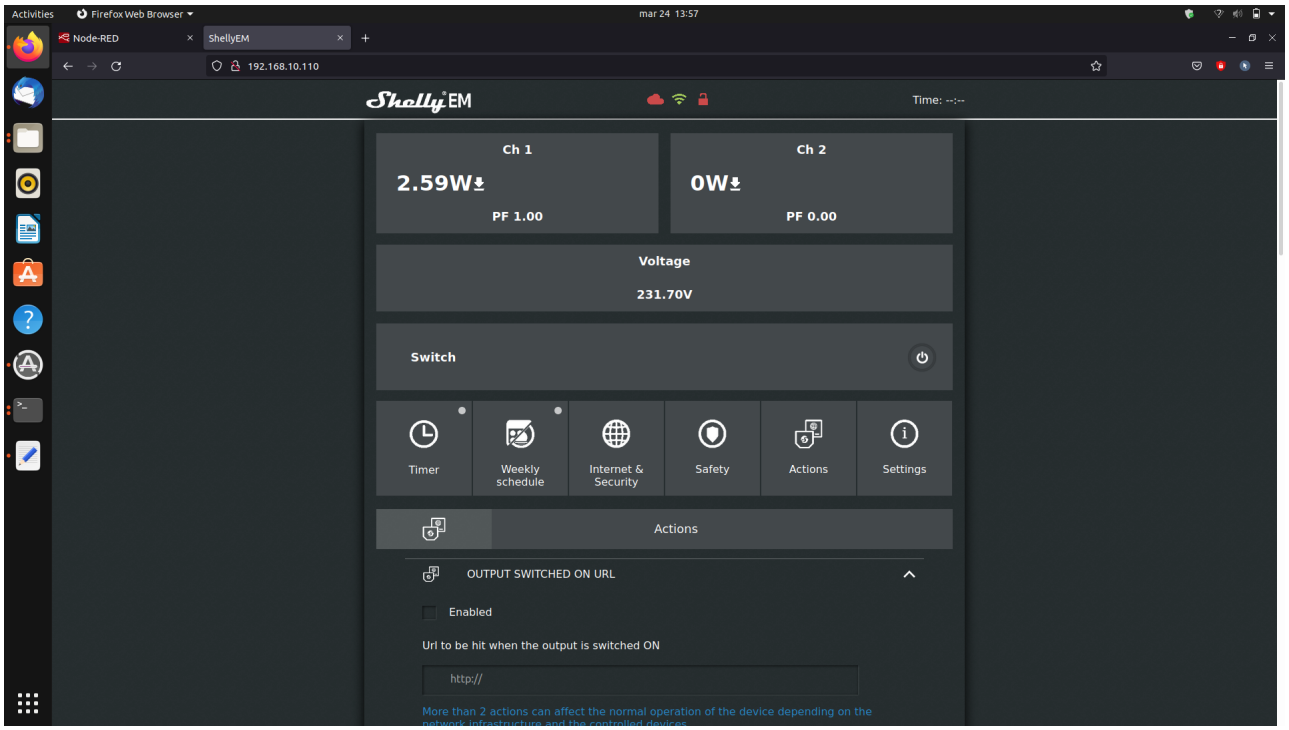


Figura 12: Shelly Webpage.

En este caso se está trabajando con un relé, un medidor de potencia y medidor de voltaje, tal y como se encuentra reflejado en la Figura 13. Adicionalmente, en el laboratorio se dispone de un motor el cual simula el comportamiento de una persiana automática, no obstante, debido a los problemas surgidos durante el trabajo con el relé, puesto que al recibir diversas peticiones a la vez este se volvía loco. Por lo que se decidió no trabajar con el motor.

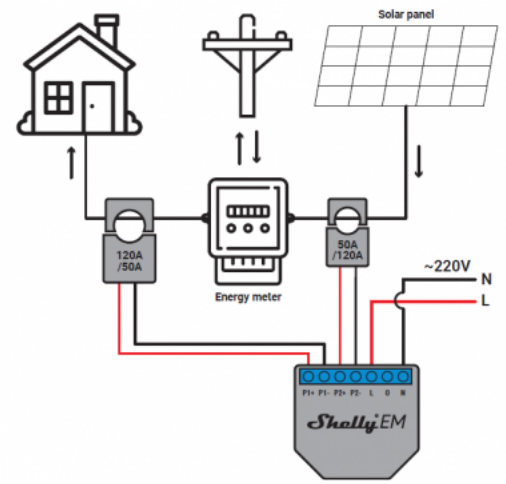


Figura 13: Esquema de conexiones actuales del SHELLY.

Por otro lado, la interacción con los dispositivos conectados al *SHELLY* se puede realizar a través de la web del dispositivo, previamente mencionada o se puede realizar mediante peticiones `http`, las cuales son peticiones más lentas y que producen una mayor sobrecarga de la red. Un ejemplo de estas peticiones puede ser la petición para obtener el valor de tensión medido por el medidor de voltaje, esto es realizado mediante la introducción de la dirección `192.168.10.110/emeter/0`, el cual nos devuelve un valor de 2.71 V. Por otro lado, también es posible controlar el relé conectado al *SHELLY* mediante el navegador, para ello existen tres peticiones de tipo `set` distintas:

- `192.168.10.110/relay/0?turn=on` : Esta petición posiciona el relé en modo ”On”, es decir, activa el relé.
- `192.168.10.110/relay/0?turn=off` : Esta petición posiciona el relé en modo ”Off”, es decir, desactiva el relé.
- `192.168.0.40/relay/0?turn=toggle` : Esta petición posiciona el relé en el modo opuesto al que este se encuentre actualmente, es decir, si el relé se encuentra activo este se desactiva y viceversa.

Por otro lado, también es posible consultar la posición del relé mediante una petición `http` de tipo `get`, esta petición se realiza a la dirección `192.168.10.110/relay/0`.

Adicionalmente, para el relé existe un modificador adicional al controlador de estado, este modificador hace que se establezca el estado deseado durante un determinado número de segundos, este modificador es el modificador `timer=X` y un ejemplo de uso es el siguiente: `http://192.168.0.40/relay/0?turn=on&timer=10` esta petición de tipo `set` activará el relé durante 10 segundos y más tarde volverá a desactivarlo. Este tipo de modificadores es bastante útil cuando se desea establecer patrones de funcionamiento o comportamientos similares, por ejemplo, cuando se desee encender una luz durante un tiempo determinado de forma remota o cuando se desee abrir y cerrar una puerta de garaje.

Como ampliación de este apartado se ha buscado en internet los distintos tipos de peticiones `http`, tanto de tipo `set` como de tipo `get` de las que dispone el *SHELLY* y como resultado se obtuvo una serie de peticiones dependiendo del dispositivo que se desee utilizar en cada instante. Toda la información del dispositivo se encuentra disponible en la documentación del mismo en https://shelly.cloud/documents/developers/ddd_communication.pdf

Un ejemplo de este son los comandos relacionados con el control del motor de una persiana automática, estos comandos son los siguientes:

- **go=open** : Este comando abrirá la persiana, es decir, hará que gire el motor de modo que se suba la persiana, un ejemplo de uso sería utilizar la siguiente petición `http://192.168.0.40/roller/0?go=open`.
- **go=close** : Este comando cerrará la persiana, es decir, hará que gire el motor de modo que la persiana baje, un ejemplo de uso sería utilizar la siguiente petición `http://192.168.0.40/roller/0?go=close`.
- **go=stop** : Este comando hará que se detenga el motor, por ejemplo, si la persiana se está cerrando, podemos hacer que esta se quede a media altura mediante el uso de la siguiente petición `http://192.168.0.40/roller/0?go=stop`.
- **go=to_pos&roller_pos=0-100** : Este comando hará que la persiana suba o baje hasta una determinada posición determinada como un porcentaje de 0 a 100. Por ejemplo si se desea que se encuentre abierta la persiana un 30 % se deberá de realizar la siguiente petición `http://192.168.0.40/roller/0?go=to_pos&roller_pos=30`.

Por otro lado, el controlador del motor también se ve afectado por el modificador de tiempo utilizado para el relé, no obstante, en este caso en lugar de utilizar el modificador `timer` se utiliza un nuevo modificador determinado como `duration`, por lo que, si por ejemplo, se desea echar un vistazo rápido al vecino, sin que este se de cuenta se puede realizar la siguiente petición `http://192.168.0.40/roller/0?turn=open&duration=5`, la cual abrirá la persiana durante 5 segundos para luego cerrarla.

También, si el *SHELLY* se encontrase conectado a una bombilla que pudiese iluminarse de distintos colores sería posible controlar la intensidad y el color de esta mediante los siguientes comandos.

- **turn=on** : Este comando enciende la bombilla conectada al dispositivo.
- **turn=off** : Este comando apaga la bombilla conectada al dispositivo.
- **turn=toggle** : Este comando invierte el estado de la bombilla, es decir, si esta se encuentra encendida, la apaga, y si esta se encuentra apagada, la enciende.
- **red = 0 - 255** : Este comando determina la intensidad de color rojo que emite la bombilla, estos valores permiten generar colores distintos mediante combinación de intensidades. Cabe destacar que este valor se encuentra entre 0 y 255.
- **blue = 0 - 255** : Este comando determina la intensidad de color azul que emite la bombilla, este valor se encuentra entre 0 y 255.
- **green = 0 - 255** : Este comando determina la intensidad de color verde que emite la bombilla, este valor se encuentra entre 0 y 255.
- **white = 0 - 255** : Este comando determina la intensidad de color blanco que emite la bombilla, este valor se encuentra entre 0 y 255.
- **gain = 0 - 100** : Este comando determina la intensidad global de la bombilla en porcentaje, al margen del valor de intensidad de los distintos colores.

Un ejemplo de funcionamiento de este dispositivo puede ser realizar una petición de tipo `set` que se encienda la bombilla y se determinen los valores de intensidad de los colores RGB.
`http://192.168.0.50/color/0?turn=on&red=255&green=86&blue=112&white=0`. Por otro lado también

se podría realizar una petición de tipo `set` que únicamente estableciese el valor del color blanco de la bombilla, iluminándose esta en un tono grisáceo. `http://192.168.0.50/color/0?turn=on&white=20`. También es posible encender la bombilla sin determinar sus valores de intensidad mediante una única petición `set` similar a las anteriores. `http://192.168.0.40/color/0?turn=on`.

Finalmente, a este dispositivo también es posible aplicar el modificador de tiempo aplicado en el relé, por lo que se podrían generar efectos lumínicos similares a los de una discoteca mediante un simple algoritmo que se conecte al dispositivo *SHELLY*.

Peticiones http desde Node-Red

Como se ha mencionado anteriormente, es posible controlar los dispositivos conectados al *SHELLY* mediante peticiones `http`, sin embargo, dichas peticiones no tienen por qué ser necesariamente realizadas mediante el navegador, Node-Red nos permite realizar dichas peticiones mediante un bloque de peticiones `http`, su funcionamiento es el mismo que con el navegador, no obstante para realizar la petición, en lugar de cargar la página web hay que inyectar un valor positivo al bloque de peticiones. Además, dicho bloque nos permite recibir el resultado obtenido de peticiones de tipo `get` por lo que, en caso de ser necesario, sería posible generar una interfaz de usuario en la que se muestren los datos recibidos del *SHELLY*.

Para comprobar su funcionamiento se experimentó con una serie de bloques que contenía gran parte de los comandos mencionados en la ampliación de la sección anterior. Siendo estos bloques utilizados para controlar el relé conectado a la maqueta del laboratorio, el motor y el medidor de tensión, tal y como se muestra en la Figura 14. No obstante, como se ha mencionado anteriormente, únicamente fueron utilizados el medidor de tensión y el relé del dispositivo.

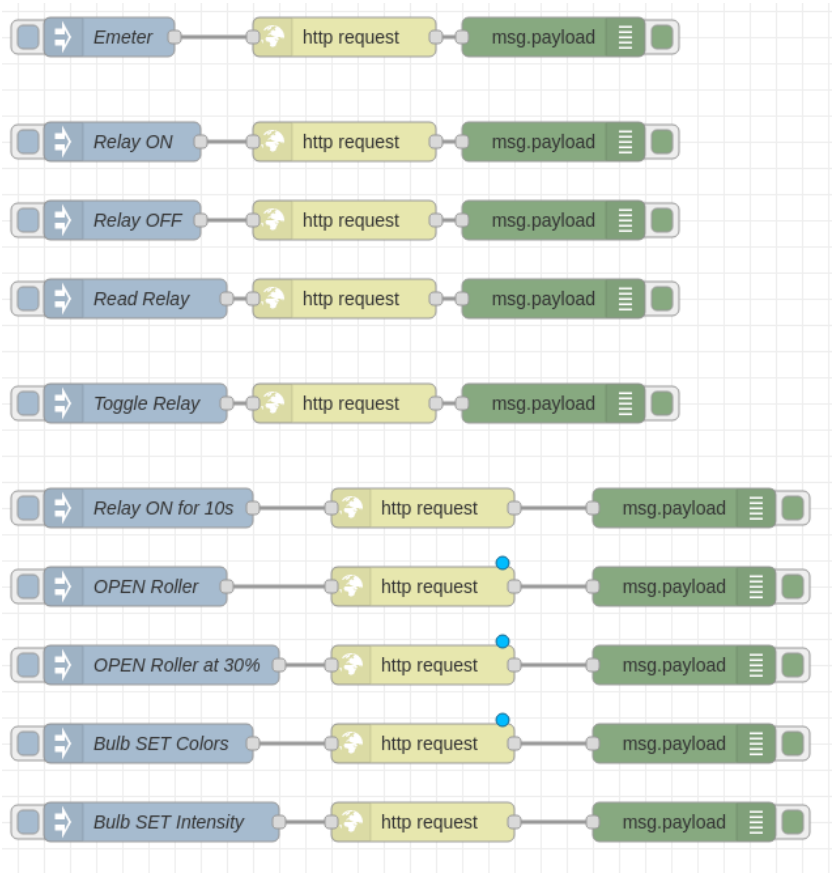
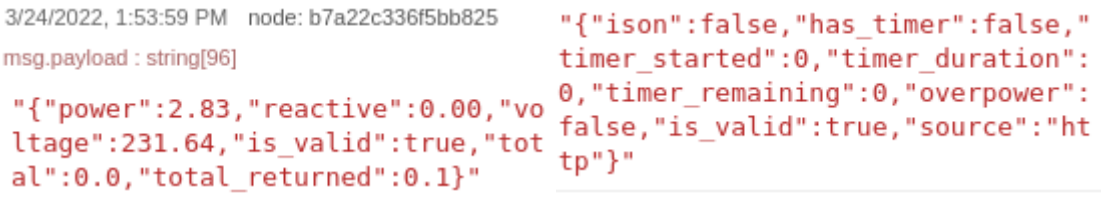


Figura 14: Pruebas de peticiones http desde Node-Red.

Los resultados obtenidos de estas peticiones fueron similares a las obtenidas cuando se trabajaba con el navegador, de hecho, resultaba mucho más fácil trabajar mediante peticiones introducidas en Node-Red ya que se puede tener todo un rango de peticiones en la misma pestaña e ir activando cada una individualmente pulsando un único botón. Además, los resultados de las peticiones de tipo `get` son recogidas en formato `json` por Node-Red, como se muestra en la figura 15, lo que facilita en gran medida el trabajo con estos datos.



(a) Medición de Voltaje (b) Estado del Relé

Figura 15: Resultados de las peticiones `get` mediante Node-Red.

Como ampliación de estas peticiones `http` con Node-Red se intentó desarrollar una interfaz en Ubidots que permitiese controlar el relé y visualizar el voltaje medido por el sensor conectado al dispositivo *SHELLY*, no obstante, debido a que no se disponía de conexión a internet desde el router al que nos encontrábamos conectados, se decidió realizar el mismo Dashboard en Node-Red. Para ello se generó inicialmente la estructura presentada en la Figura 16, donde se puede visualizar que se han implementado 3 ramas distintas, la rama superior consta de un botón el cual cambia el estado del relé a activo/desactivo cuando este es pulsado; la rama intermedia obtiene el estado del relé cada segundo y modifica el estado del interruptor situado en el Dashboard concorde al estado de este, adicionalmente, si este es modificado manualmente, se envía una petición de tipo `set` al propio relé, el cual cambia de estado concorde al cambio manual producido en el interruptor. Finalmente, se obtiene el valor de voltaje medido en el sensor de tensión conectado al *SHELLY* y se muestra gráficamente en un gauge.

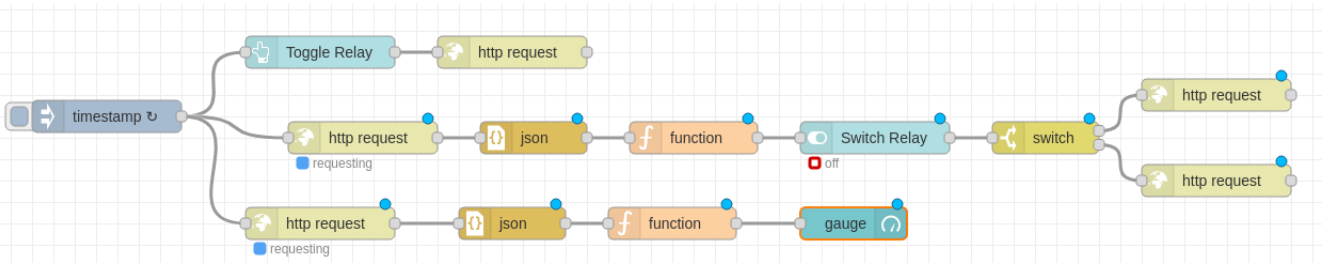


Figura 16: Estructura del Dashboard inicial.

No obstante, como esta interfaz no era lo suficientemente intuitiva se decidió incorporar un led el cual se iluminase de color verde cuando el relé se encontrase activo y de color rojo cuando este se encontrase desactivado. Obteniendo de esta forma un Dashboard similar al mostrado en la Figura 17.

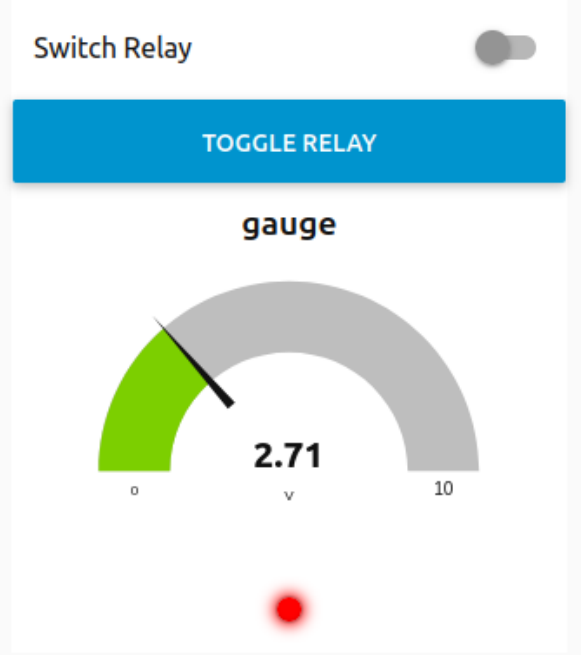


Figura 17: Dashboard http final.

Para ello se tuvieron que realizar modificaciones en la estructura previamente implementada, siendo el cambio más importante la adición de un nodo tipo led a esta, quedando dicha estructura de una forma similar a la mostrada en la Figura 18.

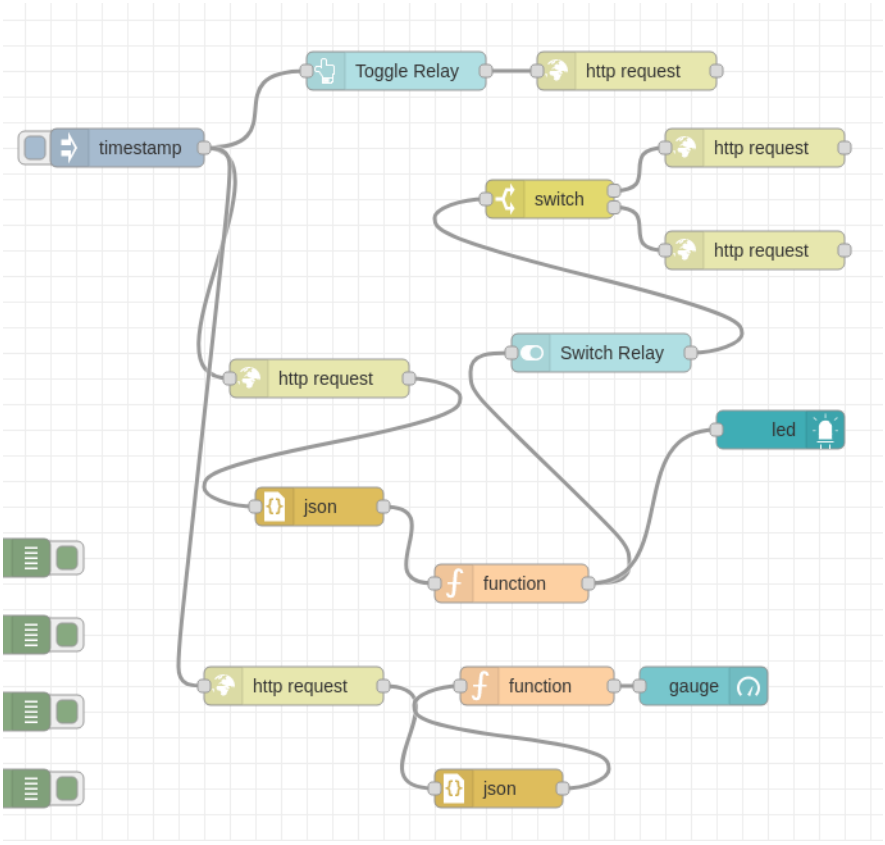


Figura 18: Estructura del Dashboard http final.

Conexión mqtt desde Node-Red

Puesto que las conexiones realizadas mediante `http` son lentas y producen una sobrecarga en el servidor, generando denegación de servicios diversas veces a lo largo de las sesiones de prácticas, se introdujo una conexión mediante el protocolo `mqtt` el cual permite crear topics a los que subscribirse para recibir datos de estos y/o publicar datos sobre los mismos. Antes de conectarse directamente a los topics generados por el *SHELLY* se realizaron diversas pruebas con diversos topics simulados, en los cuales el usuario era capaz de publicar o recibir datos, usando como topic base el topic `clase` generado por el profesor utilizando una Raspberry como broker, el cual fue utilizado mayoritariamente como sala de chat. Tras estas pruebas, se introdujeron dos sensores físicos a dicho topic, generando dos subtopics, "`/clase/humedad`" y "`/clase/temperatura`". Por lo que para interactuar con dichos topics se generó una estructura en Node-Red, similar a la de la Figura 19, la cual recibía los datos de dichos sensores.

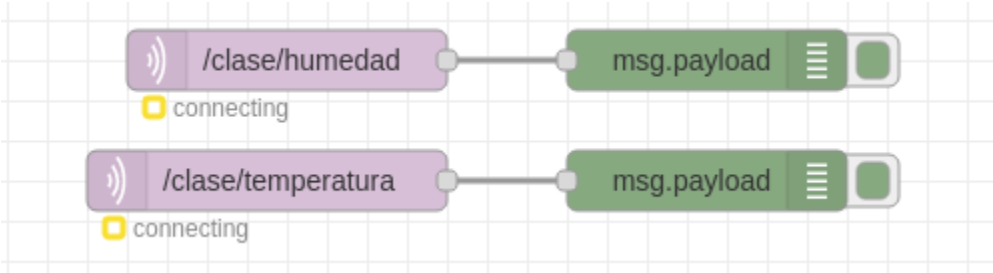


Figura 19: Estructura de subscripción a los sensores de humedad y temperatura.

No obstante, para poder visualizar de una forma más intuitiva los datos recibidos de dichos topics se generó una nueva estructura, similar a la de la Figura 20, en la cual se generaba un Dashboard que contenía una gráfica de humedad y un gauge con los valores de temperatura recibidos.

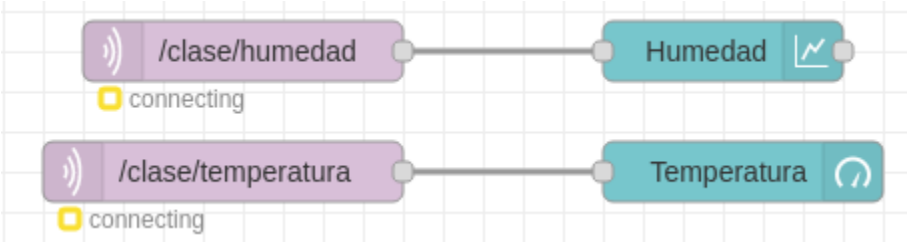


Figura 20: Estructura del Dashboard de humedad y temperatura.

Cabe destacar que existen dos tipos de bloques utilizados en la librería de mqtt, dichos bloques son los bloques "mqtt_in" y "mqtt_out", donde el bloque "mqtt_in" es usado para subscribirse a un topic en modo receptor, dichos bloques son los que se encuentran siendo utilizados en las figuras 19 y 20; mientras que el bloque "mqtt_out" es usado para subscribirse a un topic en modo publicador, el cual se usará más adelante para controlar el estado del relé conectado al dispositivo *SHELLY*.

Sin embargo, antes de poder utilizar dichos bloques es preciso configurarlos, para ello inicialmente es preciso de establecer una conexión entre el bloque y el broker. Para ello es preciso introducir el bloque mqtt deseado y configurar el server con la ip `192.168.10.144`, dicho broker es el que contiene las conexiones entre los dispositivos y el *SHELLY*. Estos valores son introducidos en la pestaña de "conexión", por otra parte en la pestaña de "seguridad" es preciso introducir el usuario y la contraseña que se debe de utilizar para el acceso al broker, este caso se utilizó el usuario "comunicaciones" y la contraseña "comunicacionespass". Una vez realizada dicha configuración es posible introducir el topic al que se desea subscribir dicho bloque.

Tras realizar las pruebas necesarias con los sensores físicos y los distintos topics simulados se prosiguió a realizar

pruebas mediante la conexión al dispositivo *SHELLY*. No obstante, antes de poder conectarnos a un topic en concreto del *SHELLY* es preciso saber qué topics se encuentran disponibles, para ello es preciso subscribirse al topic "`shellies/shellyem1/#`", donde "shellies" indica la intención de conectarse a un *SHELLY*, "/shellyem1" indica el id del *SHELLY* al que me quiero conectar y lo que va después indica el servicio al cual me quiero conectar. Por otro lado, si quiero recibir lo que publiquen todos los *SHELLYS* conectados a la red es posible hacerlo mediante una subscripción al topic "shellies/#". Una vez recibida la información de todos los topics disponible se puede decir con certeza que existe un topic que nos indica el estado del relé (On/Off), el cual es el topic "shellies/shellyem1/relay/0", un topic que nos permite controlar dicho estado, conocido como "shellies/shellyem1/relay/0/command" y un topic que nos muestra el voltaje medido por el sensor, con la dirección "shellies/shellyem1/emeter/0/total".

Conocidos dichos topics se decidió generar una estructura en Node-Red para analizar la información obtenida de estos y comprobar su funcionamiento. Para ello se realizó la estructura mostrada en la Figura 21.

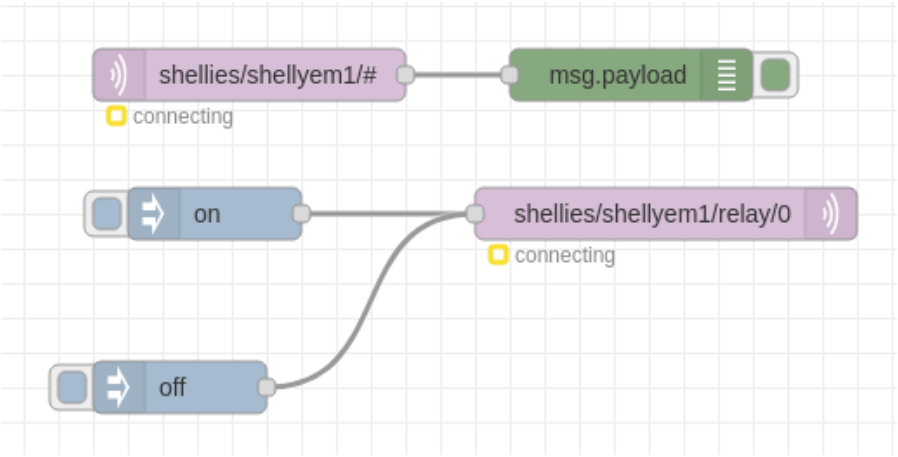


Figura 21: Estructura para probar mqtt con el SHELLY.

Una vez comprendido el funcionamiento de dichos topics se decidió ampliar la tarea creando un Dashboard similar al implementado mediante la conexión `http`, por lo que se implementó una estructura similar a la mostrada en la Figura 22.

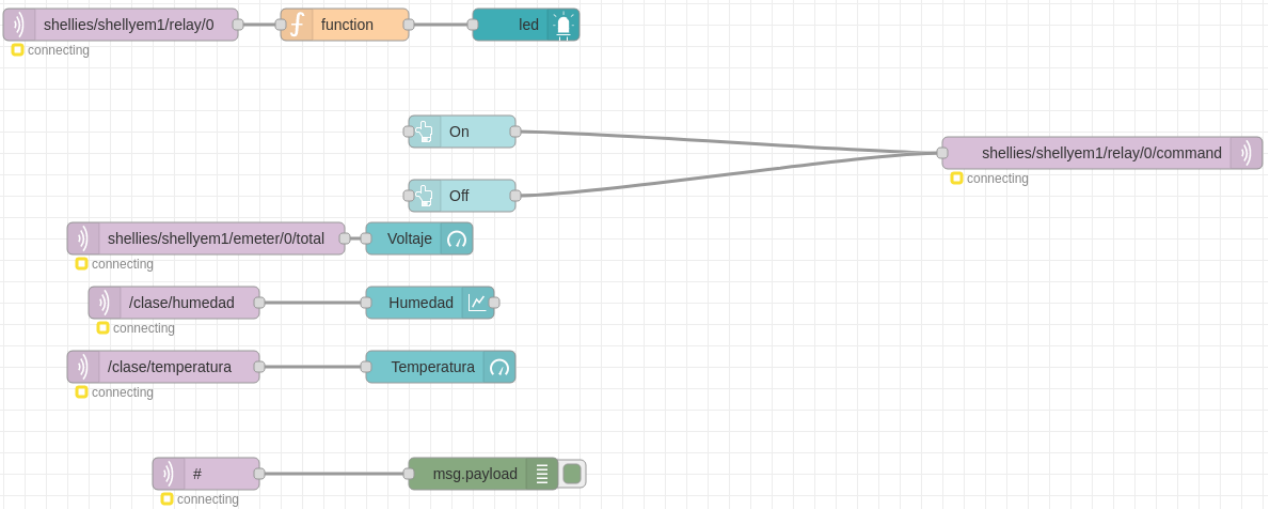


Figura 22: Estructura del Dashboard con SHELLY.

Dicho Dashboard tiene como misión principal mostrar el estado del relé mediante un led a la vez que permite

su interacción con el mismo a través de botones, puesto que, debido a la baja latencia del protocolo `mqtt`, si se utilizaba la misma estructura de bloques para controlar un interruptor y posicionarlo concorde al estado del relé, este producía un bucle infinito que publicaba valores opuestos en el topic del relé, produciendo que este se volviese loco. Por otro lado, en el Dashboard creado es posible de visualizar la gráfica de humedad ambiente a la vez de ser representados por un elemento de tipo gauge la temperatura y el voltaje medido. Finalmente, el Dashboard debería de poseer una apariencia similar a la de las Figuras 23 y 24.

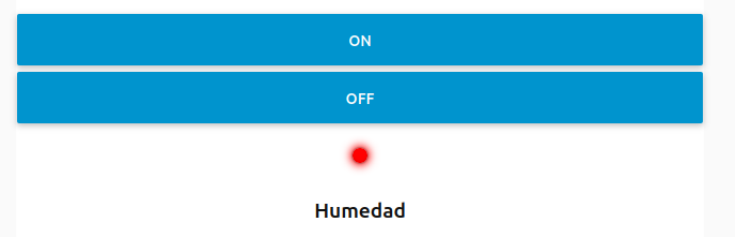


Figura 23: Representación del relé en el Dashboard.

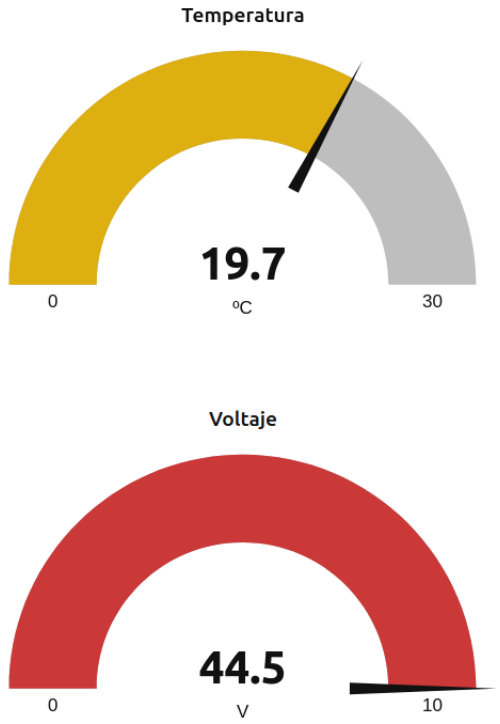


Figura 24: Representación de la temperatura y el voltaje en el Dashboard.

Ampliación

Para finalizar, se deseó indagar en el proceso de programación de un protocolo mqtt, por lo que se intentó implementar dicho protocolo utilizando una ESP32. Para ello se investigó al respecto y se generó el siguiente código en Arduino para poder utilizar la ESP32 como broker, basado en el código de luisllamas.

”Code.ino”

```
#include <WiFi.h>
#include <SPIFFS.h>
#include <PubSubClient.h>
#include "config.h"  // Sustituir con datos de vuestra red
#include "MQTT.hpp"
#include "ESP32_Utils.hpp"
#include "ESP32_Utils_MQTT.hpp"

void setup(void)
{
   Serial.begin(115200);
   SPIFFS.begin();

   ConnectWiFi_STA(true);

   InitMqtt();
}

void loop()
{
   HandleMqtt();

   PublisMqtt(millis());

   delay(1000);
}
```

Como es posible de visualizar el código se basa en el uso de distintas funciones previamente implementadas en las librerías incluidas en el código, para publicar sobre el topic definido en la función `PublisMqtt()` el valor del tiempo de ejecución. En nuestro caso el topic utilizado se denomina como ”Comunicaciones”, esto es algo que puede ser editado en la librería ”MQTT.hpp”. Que en este instante se ha modificado para que tenga la siguiente estructura.

”MQTT.hpp”

```
const char* MQTT_BROKER_ADRESS = "192.168.1.150";
const uint16_t MQTT_PORT = 1883;
const char* MQTT_CLIENT_NAME = "ESP8266Client_1";

WiFiClient espClient;
PubSubClient mqttClient(espClient);

void SuscribeMqtt()
{
   mqttClient.subscribe("Comunicaciones");
}

String payload;
void PublisMqtt(unsigned long data)
{
   payload = "";
   payload = String(data);
   mqttClient.publish("Comunicaciones", (char*)payload.c_str());
}

String content = "";
void OnMqttReceived(char* topic, byte* payload, unsigned int length)
{
```

```
    Serial.print("Received on ");
    Serial.print(topic);
    Serial.print(": ");

    content = "";
    for (size_t i = 0; i < length; i++) {
        content.concat((char)payload[i]);
    }
    Serial.print(content);
    Serial.println();
}
```

Esta es una aplicación sencilla que muestra en el puerto serie el tiempo de ejecución del programa, no obstante, este programa puede ser fácilmente escalado para que se pueda seleccionar en qué topic escribir y cómo proceder dependiendo de los datos recibidos.

”Code.ino”

```
#include <WiFi.h>
#include <SPIFFS.h>
#include <PubSubClient.h>
#include "config.h"  // Sustituir con datos de vuestra red
#include "MQTT.hpp"
#include "ESP32_Utils.hpp"
#include "ESP32_Utils_MQTT.hpp"

void setup(void)
{
    Serial.begin(115200);
    SPIFFS.begin();

    ConnectWiFi_STA(true);

    InitMqtt();
}

void loop()
{
    HandleMqtt();

    PublisMqtt("time",millis());

    delay(1000);
}
```

”MQTT.hpp”

```
const char* MQTT_BROKER_ADRESS = "192.168.1.150";
const uint16_t MQTT_PORT = 1883;
const char* MQTT_CLIENT_NAME = "ESP8266Client_1";

WiFiClient espClient;
PubSubClient mqttClient(espClient);

void SuscribeMqtt(String topic)
{
   mqttClient.subscribe(topic);
}

String payload;
void PublisMqtt(String topic, unsigned long data)
{
   payload = "";
   payload = String(data);
   mqttClient.publish(topic, (char*)payload.c_str());
}

String content = "";
void OnMqttReceived(char* topic, byte* payload, unsigned int length)
{
   Serial.print("Received on ");
   Serial.print(topic);
   Serial.print(": ");

   content = "";
   for (size_t i = 0; i < length; i++) {
      content.concat((char)payload[i]);
   }
   Serial.print(content);
   Serial.println();
}
```