



UA

Sistemas Industriales

Daniel Asensi Roch DNI : **48776120C**

7 de noviembre de 2022

Índice

1. Sesión 1: Introducción Node-red y OpenWeather	2
1.1. OpenWeather API	2
1.2. Realizar Request OpenWeather con Node-red	4
1.3. Apartado Extra	5
1.4. Public APIs	5
2. Sesión 2 Dashboards, Randoms y Variables	6
2.1. Randoms	6
2.2. Dashboards	7
2.3. Extra	8
3. Sesión 3:	10
3.1. La intraned	10
3.2. Dispositivos atacables	10
3.3. Dashboard Shelly	11
3.4. Código node-red	12
4. Sesión 4	13
4.1. Apartado Extra: Nuestro nodo Mqtt	15
5. Sesión 5	15
5.1. Parte 1	15
5.2. Parte 2	16
6. Sesión 6:	17
6.1. Identificación de dispositivo	17
6.2. Configuración Script	18
6.3. Creación del servidor MQTT	18
6.4. Conexión de python a node-red	19
6.5. Cuenta y dashboard Ubidots	19

1. Sesión 1: Introducción Node-red y OpenWeather

En la presente sesión se ha realizado una introducción a la tecnología de **Node-Red**, la cual nos ofrece una forma sencilla de generar soluciones *No Code*, es decir que la programación se realiza mediante bloques de código los cuales implementan funciones escritas en JavaScript, las cuales debemos completar para realizar pequeñas funcionalidades las cuales se enlazarán de manera visual. La interfaz presentada por el **Node-Red** es la siguiente:

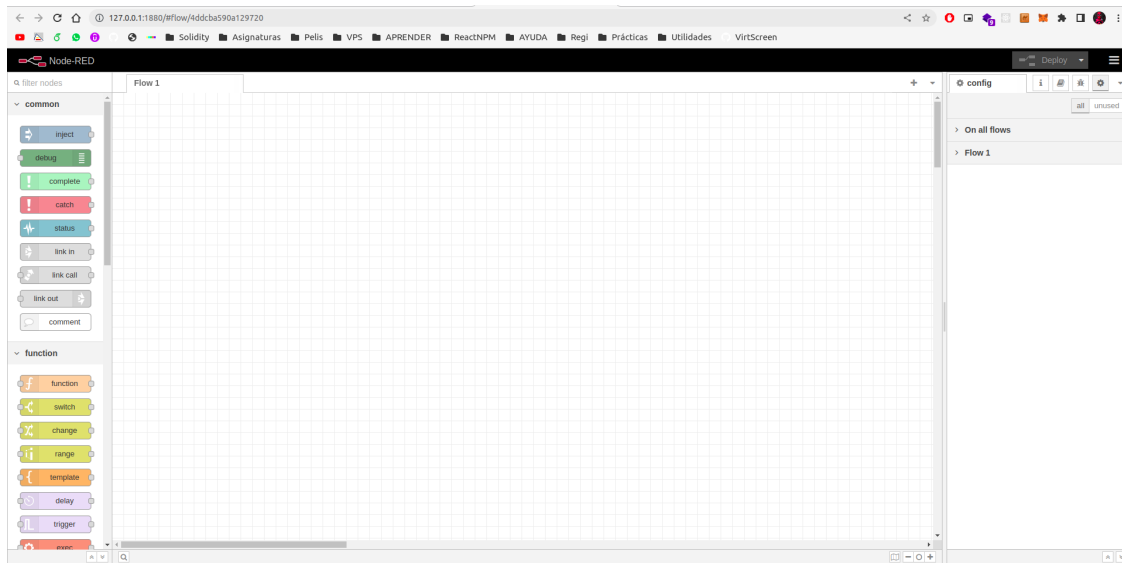


Figura 1: Interfaz gráfica Node-red

De tal manera dispondremos en el lateral de nuestro *IDE*, los bloques disponibles para formar nuestra solución.

A continuación se muestra un posible ejemplo de un *"Hello World"*:



Figura 2: Código de posible Hello World

1.1. OpenWeather API

En esta sesión también se nos ha presentado la API gratuita de OpenWeather, con la cual podemos acceder a los datos del clima en tiempo real de cualquier ciudad del mundo. Para acceder a dichos datos utilizaremos las URL proporcionadas por la documentación de la propia API, las cuales tienen una apariencia como la siguiente:

Obtención de datos por latitud y longitud:

<https://api.openweathermap.org/data/2.5/weather?lat=44.34&lon=10.99&appid={APIKEY}>

Obtención de datos por nombre de ciudad:

`https://api.openweathermap.org/data/2.5/weather?q=Alicante&appid={APIKEY}`

Los parámetros obligatorios para las anteriores URL son los siguientes:

- **lat, long**: Coordenadas geográficas (latitud, longitud)
- **APIKEY**: Api key del usuario para identificarlo

Los parámetros Opcionales para las anteriores URL son los siguientes:

- **exclude** : Para excluir algunas partes de los datos meteorológicos de la API.
- **units** : Unidades de medida (standard, metric e imperial).
- **lang** : Para obtener los datos en nuestro idioma.

Estas llamadas a la URL nos proporcionarán un objeto JSON, el cual contendrá los datos solicitados, organizados de la siguiente manera

```
1 {
2   "coord": {
3     "lon": -0.4815,
4     "lat": 38.3452
5   },
6   "weather": [
7     {
8       "id": 801,
9       "main": "Clouds",
10      "description": "few clouds",
11      "icon": "02d"
12    }
13  ],
14  "base": "stations",
15  "main": {
16    "temp": 297.72,
17    "feels_like": 297.98,
18    "temp_min": 294.21,
19    "temp_max": 299.39,
20    "pressure": 1017,
21    "humidity": 67
22  },
23  "visibility": 10000,
24  "wind": {
25    "speed": 0.51,
26    "deg": 0
27  },
28  "clouds": {
29    "all": 20
30  },
31  "dt": 1663660333,
32  "sys": {
33    "type": 1,
34    "id": 6391,
35    "country": "ES",
36    "sunrise": 1663652863,
37    "sunset": 1663696995
38  },
39  "timezone": 7200,
40  "id": 2521978,
41  "name": "Alicante",
42  "cod": 200
43 }
```

1.2. Realizar Request OpenWeather con Node-red

Para poder obtener la información nombrada en los apartados anteriores desde node-red tenemos dos maneras

1. Desde Http Request de Node-red
2. Bloque de OpenWeather

Para realizar la llamada a la Api con el primer metodo deberemos realizar los siguientes pasos:

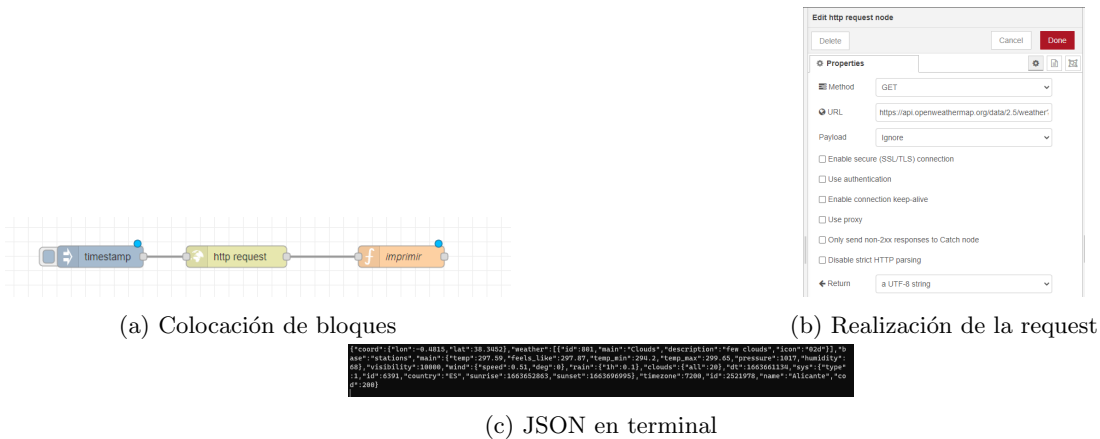


Figura 3: Código de Request OpenWeather

La otra manera en la cual podemos realizar nuestra petición a la API de OpenWeather, sería utilizando los bloques ofrecidos por la documentación, los cuales podemos instalar en node-red, el código de Request tendría el siguiente aspecto:



Figura 4: Código de Request OpenWeather blocks

La peculiaridad que presentan los bloques de OpenWeather, con respecto al Https Request proporcionado por node-red de manera nativa, es que los bloques de OpenWeather nos devuelven un objeto completo, en

vez de un json que luego deberíamos de parsear, de tal manera si deseamos devolver información concreta en un payload, deberemos realizar un destructuring de los datos que deseemos.

1.3. Apartado Extra

Como apartado extra se han explorado diferentes APIs las cuales nos podrían ser útiles en futuros desarrollos, por ejemplo tenemos la API **Cat-facts** con la cual desde el siguiente end-point podremos obtener datos curiosos sobre los gatos:

```
https://cat-fact.herokuapp.com
```

Añadiendo los siguientes parametros a la URL podremos obtener datos curiosos sobre gatos:

```
http://facts/random?animal_type=cat&amount=2
```

Obteniendo una respuesta como la siguiente:

```
[
  {
    "_id": "591f9894d369931519ce358f",
    "__v": 0,
    "text": "A female cat will be pregnant for approximately 9 weeks - between 62
and 65 days from conception to delivery.",
    "updatedAt": "2018-01-04T01:10:54.673Z",
    "deleted": false,
    "source": "api",
    "sentCount": 5
  },
  {
    "_id": "591f9854c5cbe314f7a7ad34",
    "__v": 0,
    "text": "It has been scientifically proven that stroking
a cat can lower one's blood pressure.",
    "updatedAt": "2018-01-04T01:10:54.673Z",
    "deleted": false,
    "source": "api",
    "sentCount": 3
  }
]
```

La cual como hemos visto en los apartados anteriores podremos mostrar de manera sencilla con Node-red.

1.4. Public APIs

A lo largo de nuestra vida como desarrolladores y Ingenieros informáticos deberos buscar maneras de sacar información de un campo de manera rápida, para ello se han creado repositorios de Github los cuales contienen los URLs de cientos de miles de APIs ordenadas, ordenadas por los campos de nuestro interés, desde datos curiosos hasta la información de la bolsa o Blockchain. Adjunto el repositorio:

```
https://github.com/public-apis/public-apis
```

2. Sesión 2 Dashboards, Randoms y Variables

En la presente sesión se han realizado introducciones y ejemplos de nuevos bloques de código, los cuales nos ayudarán en nuestro desarrollo.

2.1. Randoms

Los nodos random nos permitirán generar elementos aleatorios de una manera rápida y sencilla, los cuales podremos inyectar en objetos y variables.

Para realizar la instalación de dichos nodos deberemos realizar los siguiente pasos:

1. Dirigirnos al menú contextual derecho
2. Ingresar en Manage Palette
3. Instalar **random-generator_node-red-contrib**

De esta manera aparecerá el siguiente paquete y sus consecuentes nodos:

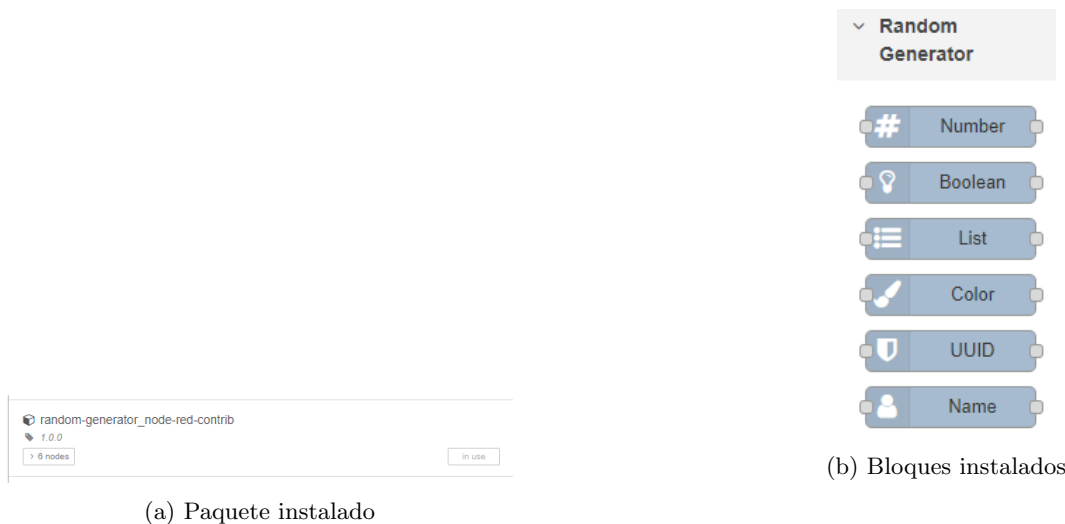


Figura 5: Bloques instalados

Mediante la interfaz de estos bloques podremos generar los valores aleatorios correspondientes a cada bloque, pondré el ejemplo de los números aleatorios ya que estos no serán útiles a lo largo de la práctica.

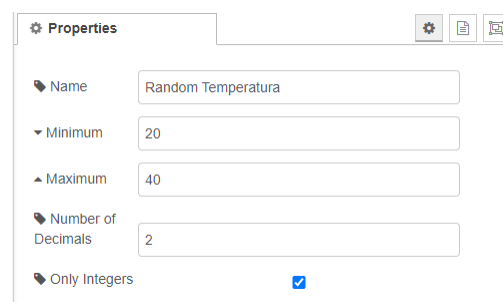


Figura 6: Interfaz gráfica Node-red Randoms

Como se muestra en la imagen anterior, podemos insertar parámetros en nuestro número aleatorio.

En la siguiente imagen podemos ver un ejemplo de una generación de un número aleatorio que inyectaremos en una variable y después sacaremos mediante nuestro debugger.

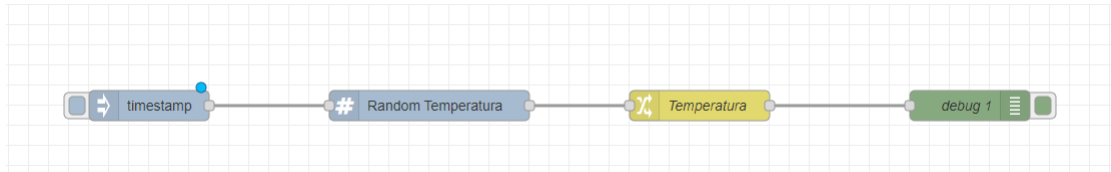


Figura 7: Ejemplo

2.2. Dashboards

Los nodos random nos permitirán generar elementos aleatorios de una manera rápida y sencilla, los cuales podremos inyectar en objetos y variables.

Para realizar la instalación de dichos nodos deberemos realizar los siguientes pasos:

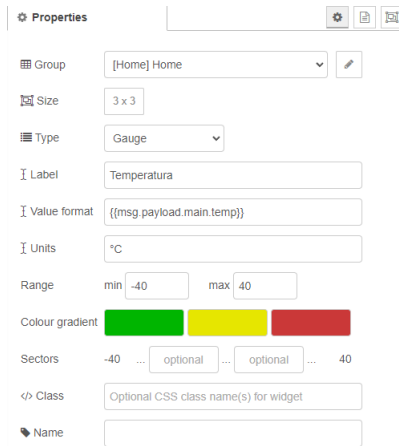
1. Dirigirnos al menú contextual derecho
2. Ingresar en Manage Palette
3. Instalar **node-red-dashboard**

De esta manera aparecerá el siguiente paquete y sus consecuentes nodos:

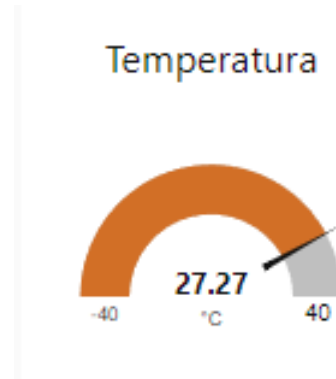


Figura 8: Ejemplo

Para configurar algunos de los nodos del Dashboard contamos con una interfaz gráfica capaz de recibir parámetros de manera sencilla.



(a) Configuración Dashboard



(b) Resultado

Figura 9: Dashboard

Como podemos observar en las imágenes, en la configuración donde introduciríamos el valor de manera directa, obtenemos los valores desde el payload, lo que nos permitirá realizar una personalización de los nodos.

2.3. Extra

Como apartado extra de esta sesión he realizado un Dashboard más complejo que el propuesto durante la sesión de trabajo en concreto el siguiente:

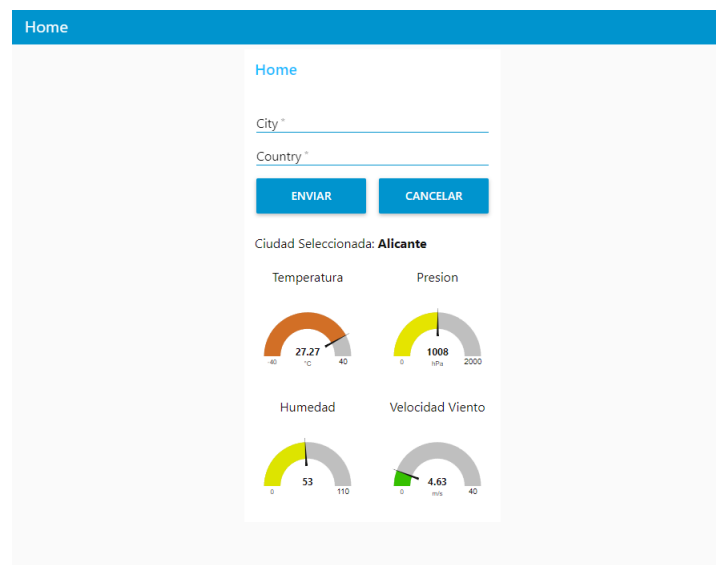


Figura 10: Dashboard Complejo

En la dashboard que he configurado el usuario deberá ingresar los datos de una ciudad y un país concretos para que estos sean inyectados en el nodo de la API de Openweather.

De tal manera los nodos utilizados y enlazados se ven en la figura 11.

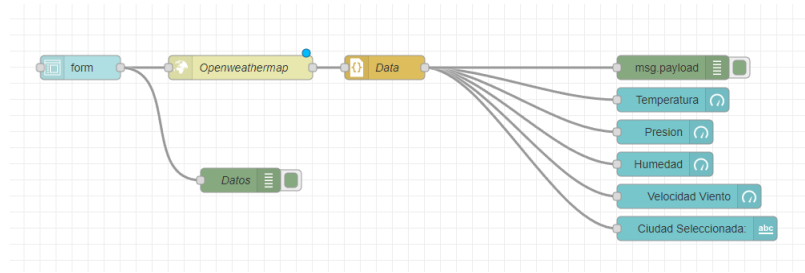


Figura 11: Dashboard Complejo Configuración

He optado por utilizar el nodo del formulario para poder especificar los datos de manera más sencilla y evitando la redundancia de nodos, este nodo ha sido configurado como se ve en la imagen 12.

Label	Name	Type	Required	UIRows	Remove
City	city	Text	<input checked="" type="checkbox"/>		
Country	country	Text	<input checked="" type="checkbox"/>		

Sección de botones: Enviar, Cancelar

Sección de configuración: Place the form elements in two columns (desactivado), Topic (seleccionado: topic), Class (Optional CSS class name(s) for widget)

Figura 12: Dashboard Complejo Configuración

Estos datos que se capturarán del formulario, son inyectados dentro de la URL de la API de openWeather utilizando las llaves para obtener los datos del payload como se ve en la figura 40.

Method: GET

URL: `http://api.openweathermap.org/data/2.5/weather?q={{(payload.city)}}&units=metric&appid=b5b3a3a027882bfcc2fb12c033de7d5d`

Payload: Append to query-string parameters

Figura 13: Dashboard Complejo Configuración

Una vez realizada la consulta a la API los datos son inyectados en un nodo JSON, el cual los transformará a objeto para que puedan ser leídos de manera sencilla por los nodos Dashboards.

3. Sesión 3:

En esta sesión de trabajo se nos ha planteado la interacción con paneles físicos utilizando el protocolo de http, de esta manera podríamos interactuar con los mencionados paneles utilizando APIS y lanzando request para realizar modificaciones en el comportamiento de los mismos.

3.1. La intranet

Para poder interactuar con los mencionados dispositivos, debíamos acceder a una intranet creada por nuestro tutor, accediendo de manera muy similar a la manera que accederíamos a una wifi privada.

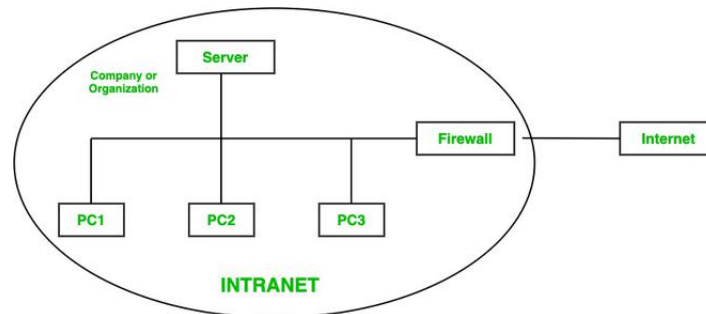


Figura 14: Esquema Intranet

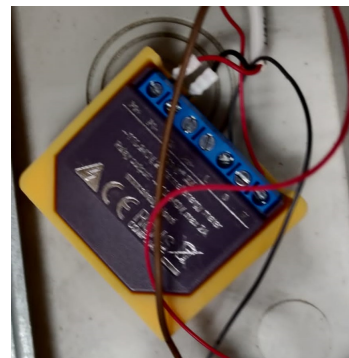
Una vez hubiéramos accedido a dicha intranet, podíamos atacar a determinadas ip que contenían una interfaz gráfica, la cual nos permitía interactuar de manera cómoda con dichos paneles.

3.2. Dispositivos atacables

Antes de comenzar a hablar de la mencionada anteriormente interfaz gráfica, debemos hablar de algunos dispositivos atacables:



(a) Dispositivo 1



(b) Dispositivo 2

Figura 15: Dispositivos

Mediante el uso de los anteriores dispositivos podemos controlar el flujo de corriente que llega a los enchufes de nuestro hogar y puesto de trabajo de manera remota, además de ver el voltaje consumido y hacer el encendido y apagado de dichos dispositivos.



Figura 16: Panel Completo

Todos estos dispositivos inteligentes que nos ayudarán en la gestión de nuestra smarthome o smartindustry son manejados desde el dashboard de Shelly.

3.3. Dashboard Shelly

Para tener un control amigable con el usuario, este dispone de una aplicación de acceso web privatizada por la empresa Shelly, la cual se presenta de la siguiente manera:

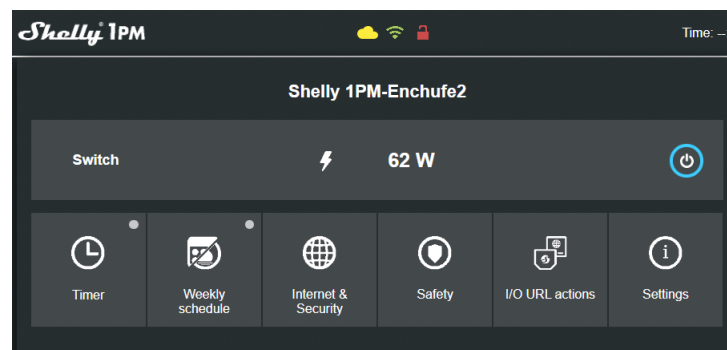


Figura 17: Panel de Shelly

Como podemos observar en dicho panel contamos, con un botón de power, el cual nos permitirá manipular el flujo de corriente, además podemos observar la cantidad Watios que nos encontramos consumiendo actualmente, por último contamos con una barra de configuración con la cual podremos configurar horarios y seguridad.

3.4. Código node-red

En esta sesión también hemos podido probar y configurar un código node-red creado por el profesor, el cual de igual manera nos permite interactuar con dichos dispositivos utilizando APIs, el código en cuestión es el siguiente:

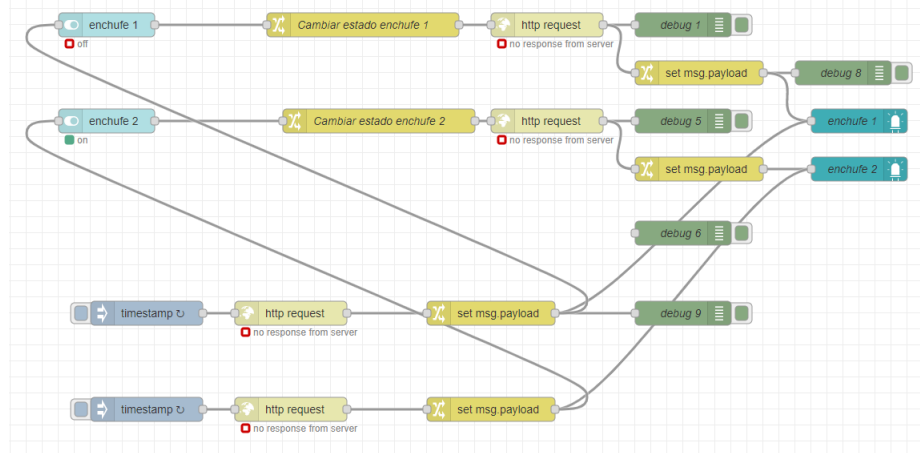


Figura 18: Código principal

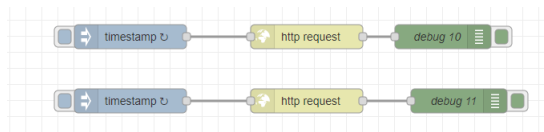


Figura 19: Código temporizadores

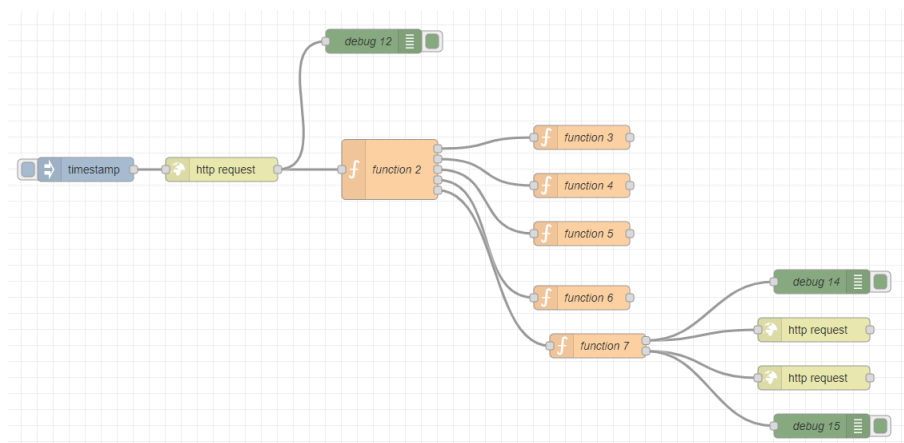


Figura 20: Código condicional temporizador

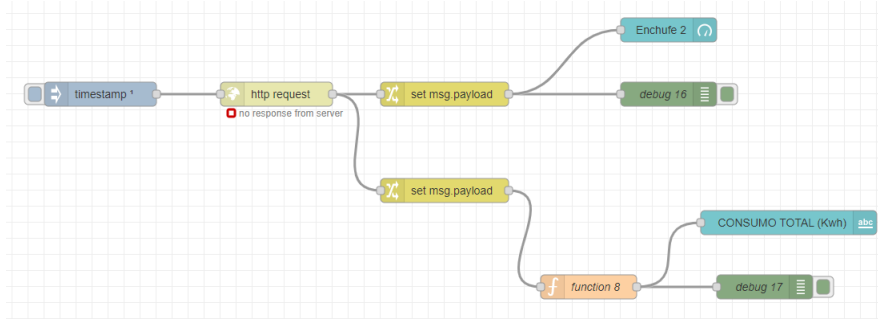


Figura 21: Código visualización voltaje

De tal forma presentará la siguiente interfaz para que el usuario pueda encender y apagar los enchufes así como ver su estado y voltaje de un solo vistazo.

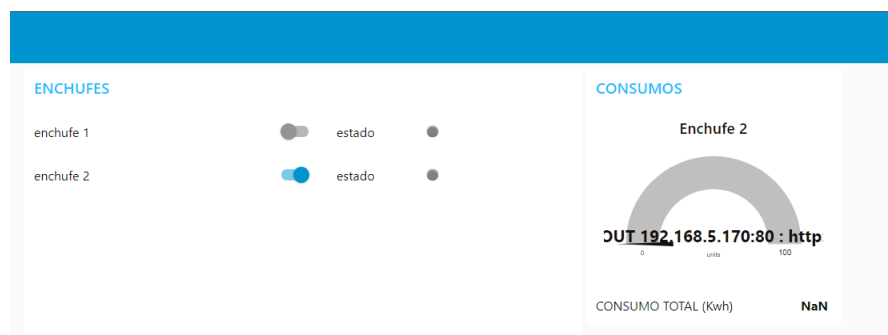


Figura 22: Interfaz de usuario

4. Sesión 4

En esta sesión se ha realizado el mismo procedimiento que en la anterior, pero esta vez utilizando el protocolo MQTT, este protocolo al igual que el HTTP nos permite recibir y enviar datos a un sistema inteligente para controlar su funcionamiento.

Para comprobarlo se utilizaron los dispositivos de la práctica anterior, por lo que en esta sesión y ya que se encuentra anteriormente explicado no indagaré en su funcionamiento.

Para interactuar con el protocolo **MQTT** en nuestra aplicación **Node-red** deberemos utilizar los siguientes nodos:

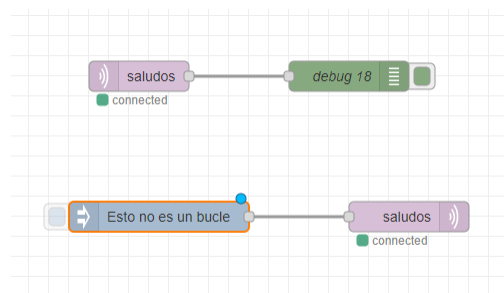


Figura 23: Nodos-Red

El nodo superior de la imagen será nuestro suscriptor, el termino suscriptor se acuña ya que este nodo recibe una determinada información de un determinado tema, este tema se le denomina **Topic**, los suscriptores

pueden acceder a la información que se envía a través de ellos.

Para ello de en Node-red debemos realizar las siguientes configuraciones:

(a) Configuración Topic

(b) Configuración suscripción y seguridad

Figura 24: Configuraciones de nodo In

De esta manera podremos recibir mensajes como los siguientes:

```
10/10/2022, 16:49:23 node: debug 18
saludos : msg.payload : number
1665413359197

10/10/2022, 16:49:24 node: debug 18
saludos : msg.payload : number
1665413360211

10/10/2022, 16:49:25 node: debug 18
saludos : msg.payload : number
1665413361222

10/10/2022, 16:49:25 node: debug 18
saludos : msg.payload : string[23]
"AAAAAAAAAAAAAAAAAAAAAAAAA"
```

Figura 25: Mensajes de compañeros

Si realizamos la suscripción a uno de los dispositivos Shelly proporcionados por el profesor para la realización de la práctica los mensajes tendrán el siguiente formato:

```
10/10/2022, 17:42:35 node: debug 18
shellyshelly(pmrelay0 : msg.payload : string[3]
"off"

10/10/2022, 17:42:35 node: debug 18
shellyshelly(pmrelay0 : msg.payload : number
0

10/10/2022, 17:42:35 node: debug 18
shellyshelly(pmrelay0 : msg.payload : number
0

10/10/2022, 17:42:35 node: debug 18
shellyshelly(pmrelay0 : msg.payload : number
0

10/10/2022, 17:42:35 node: debug 18
shellyshelly(pmrelay0 : msg.payload : number
42.31

10/10/2022, 17:42:35 node: debug 18
shellyshelly(pmrelay0 : msg.payload : number
100.16

10/10/2022, 17:42:35 node: debug 18
shellyshelly(pmrelay0 : msg.payload : number
0

10/10/2022, 17:42:35 node: debug 18
shellyshelly(pmrelay0 : msg.payload : string[5]
"normal"
```

Figura 26: Mensajes de Shelly

Estos dispositivos como podemos observar nos proporcionan la temperatura, voltaje, humedad del ambiente etc, cada uno de ellos separado en un topic al que podemos suscribirnos.

4.1. Apartado Extra: Nuestro nodo Mqtt



Para configurar nuestro nodo Mqtt utilizaremos docker, mediante el cual podemos realizar una configuración aislada de nuestra maquina, además usaremos la imagen de **Mosquitto** un creador de nodos Mqtt con el cual podremos controlar el flujo de los mensajes.

Para ello deberemos escribir los siguientes comandos en nuestra terminal:

```
$ docker run -it -p 1883:1883 -p 9001:9001  
-v mosquitto.conf:/mosquitto/config/mosquitto.conf  
-v /mosquitto/data -v /mosquitto/log eclipse-mosquitto
```

Desde este momento y cuando terminen de descargarse las imagenes de Docker que contendrán las "maquinas-virtuales" configuraciones, tendremos corriendo en nuestro puerto 1883 un nodo Mosquitto para el envio de mensajes MQTT



5. Sesión 5

5.1. Parte 1

En esta sesión se ha realizado el mismo procedimiento que en la anterior, pero esta vez profundizando en gran medida en los sistemas proporcionados por la gama Shelly para el control de nuestras instalaciones, utilizando todos sus sensores disponibles a través de diferentes protocolos.

Para investigar dichos dispositivos de la gama Shelly el profesor nos ha dispuesto diferentes IPs en una red local a las que podemos atacar para ver su interfaz gráfica con la información brindada por los dispositivos, las interfaces son las siguientes:

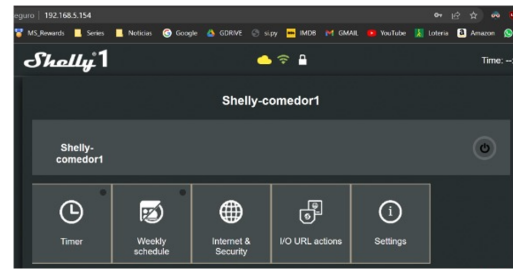
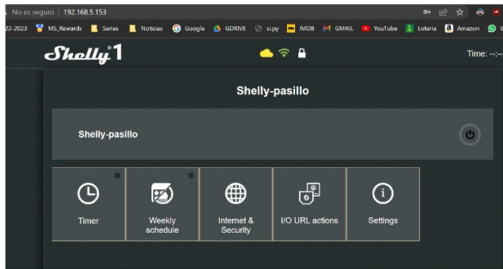


Figura 27: Interfaces Shelly

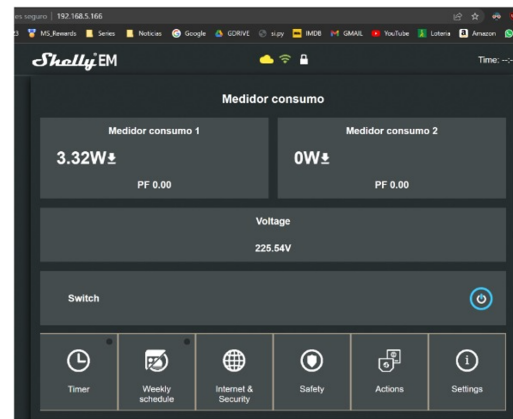
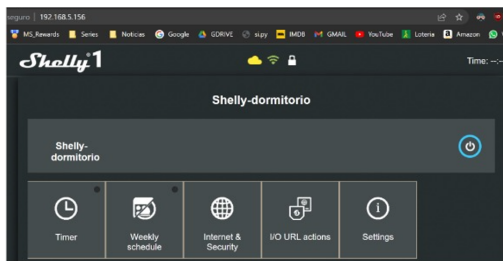


Figura 28: Interfaces Shelly

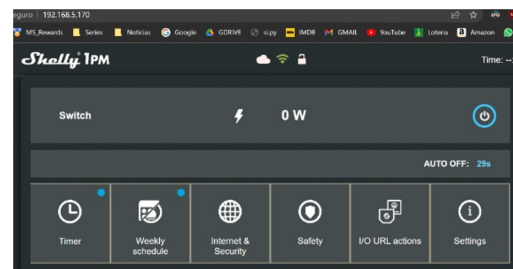
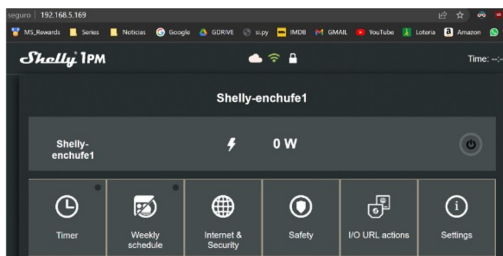


Figura 29: Interfaces Shelly

5.2. Parte 2

En la segunda parte de esta sesión hemos interactuado con los dispositivos de temperatura y humedad de Xiaomi, en concreto con los Xiaomi Mijia.



Figura 30: Xiaomi Mijia

Para poder utilizarlos y leer datos de los mismos debemos de conocer la MAC de cada uno de los dispositivos, para ello hemos utilizado la herramienta de lectura de IPs y MACs, tal y como se muestra en la siguiente imagen:

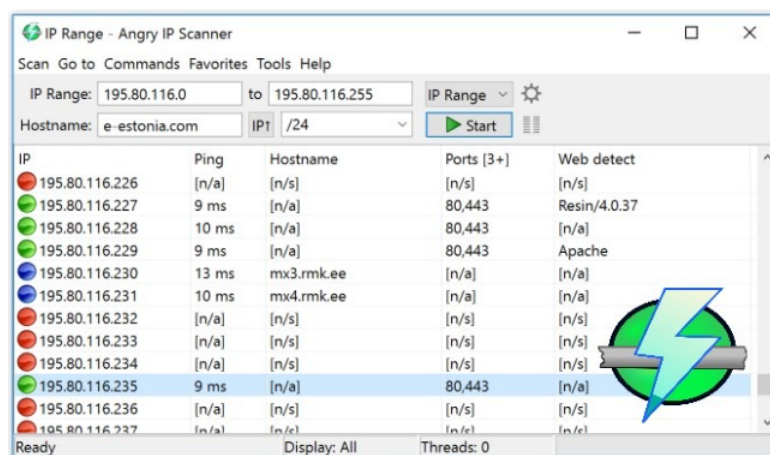


Figura 31: Lectura de Ips y macs

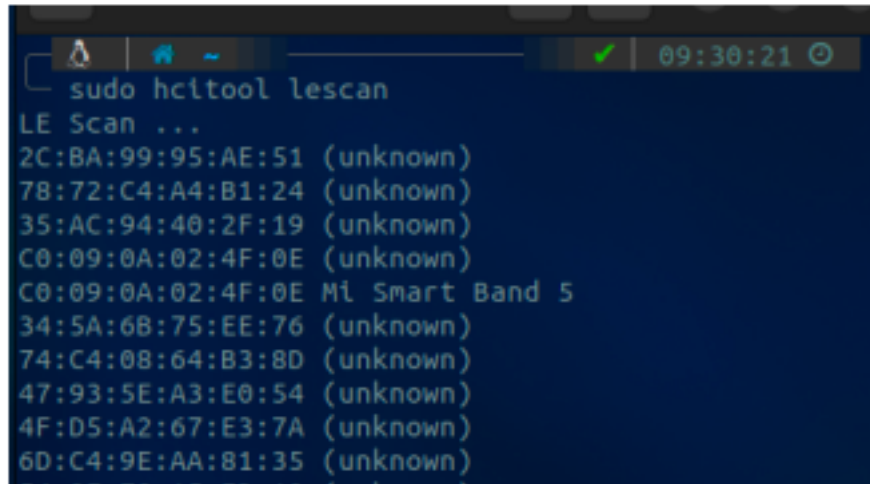
Para poder interactuar y leer los datos, se ha utilizado un código ya hecho que el profesor nos ha proporcionado desde github, sobre el cual debemos inyectar la MAC del dispositivo a atacar y instalar un par de librerías para el control Bluetooth del mismo.

6. Sesión 6:

6.1. Identificación de dispositivo

Otra de las formas para leer la MAC del dispositivo de Xiaomi ha sido utilizar la herramienta **hcitool** para detectar señales de Bluetooth Low Energy.

Este se ejecutaría de la siguiente manera:



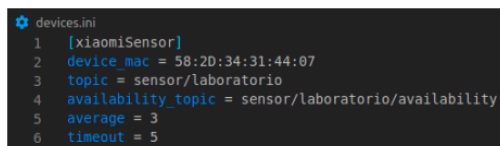
```
sudo hcitool lescan
LE Scan ...
2C:BA:99:95:AE:51 (unknown)
78:72:C4:A4:B1:24 (unknown)
35:AC:94:40:2F:19 (unknown)
C0:09:0A:02:4F:0E (unknown)
C0:09:0A:02:4F:0E Mi Smart Band 5
34:5A:6B:75:EE:76 (unknown)
74:C4:08:64:B3:8D (unknown)
47:93:5E:A3:E0:54 (unknown)
4F:D5:A2:67:E3:7A (unknown)
6D:C4:9E:AA:81:35 (unknown)
```

Figura 32: Uso de la función de hcitool

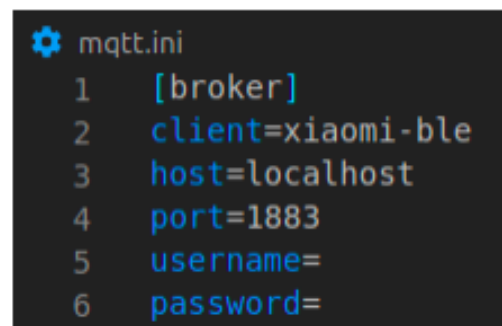
6.2. Configuración Script

Una vez identificada la MAC a la que queremos atacar deberemos configurar el script de python que se nos facilitó en la sesión anterior, el cual leer los datos del sensor y los emite a un servidor **MQTT**.

Para la configuración del script editaremos los siguientes archivos añadiendo los parámetros pertinentes:



```
devices.ini
1 [xiaomiSensor]
2 device_mac = 58:2D:34:31:44:07
3 topic = sensor/laboratorio
4 availability_topic = sensor/laboratorio/availability
5 average = 3
6 timeout = 5
```

(a) configuración de **devices.ini**

```
mqtt.ini
1 [broker]
2 client=xiaomi-ble
3 host=localhost
4 port=1883
5 username=
6 password=
```

(b) configuración de **mqtt.ini**

Figura 33: Configuración de Script python

Una vez realizada dicha configuración podremos comenzar a emitir los datos al servidor

6.3. Creación del servidor MQTT

Para configurar nuestro nodo Mqtt utilizaremos docker, mediante el cual podemos realizar una configuración aislada de nuestra máquina, además usaremos la imagen de **Mosquitto** un creador de nodos Mqtt con el cual podremos controlar el flujo de los mensajes.

Al igual que realizamos en el apartado extra de la práctica 5 arrancaremos nuestro **Broker MQTT**:

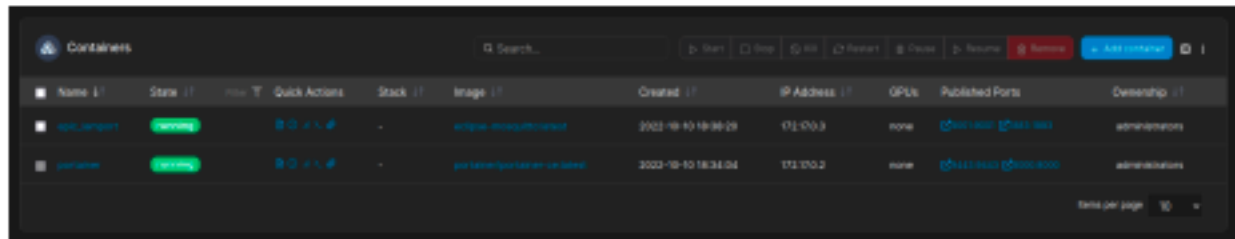


Figura 34: Arranque de nodo MQTT

6.4. Conexión de python a node-red

Para realizar la inyección de nuestro script en node-red utilizaremos enlace entre un nodo **trigger** y un nodo **exec**, en el nodo **exec**:



Figura 35: Nodo exec

De tal manera utilizando el nodo, nuestro *código* queda de la siguiente manera:

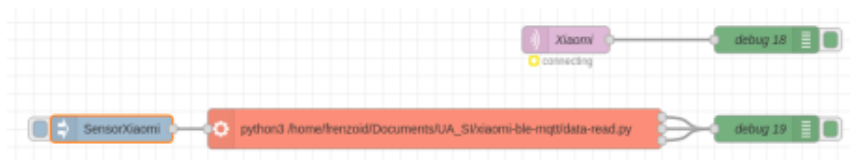


Figura 36: Código

De esta manera seremos capaces de ejecutar código en python desde nuestro node-red viendo desde consola, como nuestro código de python emite la información a nuestro broker, la cual se emite de la siguiente manera:

```
sensor/laboratorio : msg.payload : Object
  { temperature: 26.5, humidity:
    78.2, battery: 69, average: 3 }
```

Figura 37: Información enviada

6.5. Cuenta y dashboard Ubidots

Una vez hayamos creado nuestra cuenta institucional en Ubidots deberemos crear un nuevo **dashboard** para ello nos dirigiremos al menú y dashboards, allí configuraremos el mismo en una interfaz como la siguiente:

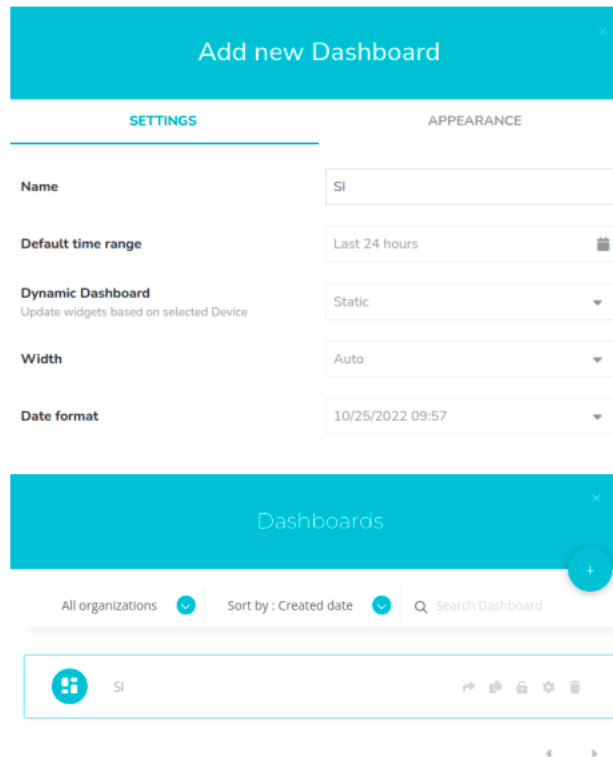


Figura 38: Configuración dashboard

Una vez configurado deberemos acceder al siguiente enlace donde encontraremos código python que deberemos copiar para terminar la configuración:

<https://help.ubidots.com/en/articles/569964-simulate-data-in-ubidots-using-python>

En el presente código podemos encontrarnos lo siguiente:

```
TOKEN = "... " # Put your TOKEN here
DEVICE_LABEL = "machine" # Put your device label here
VARIABLE_LABEL_1 = "temperature" # Put your first variable label here
VARIABLE_LABEL_2 = "humidity" # Put your second variable label here
VARIABLE_LABEL_3 = "position" # Put your second variable label here
```

Por lo que podemos ver en el código, este crea un nuevo dispositivo llamado “machine” con 3 variables: humedad, temperatura y ubicación. Para conectarlo con nuestro dashboard en ubidots copiaremos nuestro “Token” y lo pegaremos en la variable “TOKEN”

Ahora, en nuestro dashboard, podremos añadir widgets y enlazarlos automáticamente a las variables emitidas por el script. Para ello, volvemos a nuestro dashboard, y añadimos un par de widgets para mostrar la temperatura, la humedad, y un mapa para la ubicación.

Si añadimos dos termomentros obtendremos la siguiente salida:

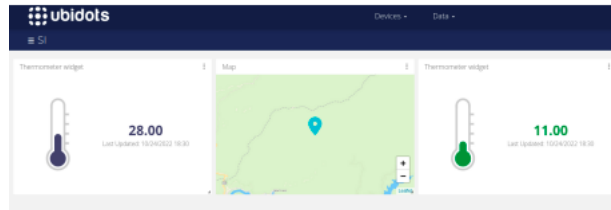


Figura 39: DashBoard completo

Para conectar node-red con Ubidots tendremos que importar su paquete de nodos, una vez hecho esto tendremos disponible un nuevo nodo, el cual podremos configurar de la siguiente manera:

The screenshot shows the 'Edit ubidots_out node' configuration window in Node-RED. At the top, there are three buttons: 'Delete', 'Cancel', and 'Done'. Below these is a 'Properties' section with a gear icon and three sub-panels. The first sub-panel is 'Account Type' with a dropdown menu set to 'Ubidots for Education'. The second sub-panel is 'Name' with a text input field containing 'Name'. The third sub-panel is 'Token' with a text input field containing 'BBFF-d2FF9n7D9SAqBsmDfnQGhzINkUZEVI'. Below these is a 'Device Label' section with a text input field containing 'XiaomiSensor'. At the bottom, there is a checkbox labeled 'Enable secure TLS connection' which is checked.

Figura 40: Node-red Ubidots