



Universidad de
Oviedo



ESCUELA POLITÉCNICA DE INGENIERÍA DE GIJÓN.

**MÁSTER UNIVERSITARIO EN INGENIERÍA DE
TELECOMUNICACIÓN**

ÁREA DE TEORÍA DE LA SEÑAL Y COMUNICACIONES

TRABAJO FIN DE MÁSTER

**ESTUDIO Y COMPARATIVA DE REDES IoT DESPLEGADAS
SOBRE LoRA Y 5G: ESTUDIO DE REDES**

**D^a. CAÑIZO OTERO, PIARINA
TUTOR: D. RAFAEL GONZÁLEZ AYESTARÁN**

FECHA: JUNIO DE 2022



ÍNDICE GENERAL.

| | |
|--|-----|
| Índice general. | 1 |
| 1.- Introducción. | 2 |
| 1.1.- Datos del trabajo. | 3 |
| 1.2.- Objetivos. | 3 |
| 1.3.- Tareas. | 3 |
| 1.4.- Cronograma. | 5 |
| 2.- Red LoRaWAN. | 6 |
| 2.1.- Despliegue IoT sobre red LoRa. | 7 |
| 2.1.1.- Captación de datos con sensor LHT65. | 16 |
| 2.1.2.- Captación de datos con CubeCell. | 21 |
| 2.2.- Envío de datos a base de datos Influxdb. | 27 |
| 3.- Red 5g. | 37 |
| 3.1.- Despliegue IoT sobre varias redes. | 38 |
| 3.1.1.- Despliegue IoT sobre red Ethernet. | 38 |
| 3.1.2.- Despliegue IoT sobre red 5G. | 48 |
| 3.1.2.1.- Despliegue 5G utilizando router NETGEAR. | 53 |
| 3.1.2.2.- Despliegue de la red 5G utilizando un smartphone. | 59 |
| 3.1.2.3.- Despliegue de la red 5G utilizando un adaptador de módulo 5G a USB. | 70 |
| 3.1.3.- Ampliación de la implementación IoT sobre 5G. | 78 |
| 4.- Comparativa entre el despliegue LoRaWAN y 5G. | 87 |
| 5.- Conclusiones. | 91 |
| 6.- Referencias. | 92 |
| 7.- Índice de figuras. | 97 |
| 8.- Índice de tablas. | 101 |



1.- INTRODUCCIÓN.

Durante los últimos años se ha ido viendo un gran auge en lo denominado como Internet de las Cosas o IoT (*Internet of the Things*), lo cual implica una gran cantidad de dispositivos inteligentes conectados a internet, requiriendo estos dispositivos no solo de un gran número de direcciones IP sino también de unos requisitos de tasa de datos y conectividad especiales.

Se prevé que durante los consiguientes años este número de dispositivos conectados siga aumentando de manera masiva. Cisco considera que en el año 2030 el número de dispositivos conectados a internet será de 500 miles de millones, para soportar tal cantidad de dispositivos sin poner en riesgo la calidad de la red utilizada hay que estudiar que redes son las más adecuadas para ese propósito. [1]

Algunos de los desafíos a los que se enfrenta la tecnología IoT son la escalabilidad, el consumo de energía o la latencia, ya que en algunos casos estos dispositivos IoT serán usados, por ejemplo, en vehículos para controlar la seguridad vial donde unos pocos segundos pueden marcar la diferencia para prevenir accidentes en momentos de descuidos humanos.

A lo largo del presente documento se presentarán dos alternativas de redes sobre las que realizar el despliegue de sistemas IoT, la primera es la red LoRaWAN ampliamente utilizada actualmente para el despliegue de estos sistemas, y la segunda es la red 5G, la cual puede ser una alternativa interesante a las redes usadas comúnmente para IoT debido a sus cualidades y su esperada implantación como red móvil principal en todo el mundo.

Para poder comparar ambas, además de comentar sus características teóricas, obtenidas de diversos estudios y artículos sobre su implantación para despliegues IoT, se realizará un despliegue IoT de ambas en el laboratorio.

Como se verá a lo largo del documento, el despliegue sobre LoRaWAN es prácticamente inmediato teniendo todos los componentes necesarios y disponiendo de un servidor de red abierto, mientras que el despliegue sobre 5G supone un reto mayor al no haber actualmente una red abierta y desplegada a la que conectarnos. Por ello, para este despliegue se han hecho uso de dispositivos comerciales como son smartphones o routers portables 5G, así como el uso de un módulo 5G con convertor a USB 3.0 que a su vez consta de dos antenas a conectar. En cualquier caso, será necesario disponer de tarjetas SIM capaces de trabajar en 5G las cuales habrá que programar para trabajar en la red 5G desplegada en el laboratorio.



1.1.- Datos del trabajo.

Título: Estudio y comparativa de redes IoT desplegadas sobre LoRa y 5G: Estudio de redes.

Autor: Piarina Cañizo Otero.

Tutor: Rafael González Ayestarán.

Resumen: En este proyecto se va a llevar a cabo el estudio y despliegue de dos redes IoT sobre dos tecnologías distintas; las cuales son: LoRA y 5G. La finalidad es realizar la comparativa entre ambas tecnologías discutiendo sus ventajas e inconvenientes, esta comparación se realizará tras realizar la experimentación con ambas alternativas en entornos reales. Este trabajo es parte de un proyecto mayor, centrándose en el estudio de las redes de conexión LoRa y 5G, y el modo de utilizarlas para aplicaciones IoT.

Titulación: Máster Universitario en Ingeniería de Telecomunicación.

1.2.- Objetivos.

Entre los objetivos a conseguir con este trabajo se encuentran los siguientes: conocimiento y manejo de redes no utilizadas previamente, como son LoRaWAN y 5G; despliegue de redes IoT con varios sensores diferentes y el conocimiento de estos sensores y su programación; comparar los despliegues IoT realizados sobre diferentes redes y finalmente, comparar las dos redes utilizadas, en función de sus ventajas e inconvenientes generales para el despliegue de sistemas IoT.

1.3.- Tareas.

Entre las tareas a realizar durante la realización de este proyecto se encuentran las siguientes:

- Investigación teórica del funcionamiento de la red LoRaWAN previa a su despliegue.
- Despliegue de la red LoRaWAN.
- Despliegue IoT de varios sensores sobre la red LoRaWAN implementada.
- Investigación teórica del funcionamiento de la red 5G previa a su despliegue.
- Despliegue IoT de varios sensores sobre una red Ethernet o WiFi previo a realizarlo sobre la red 5G.



- Despliegue de la red 5G:
 - Investigación de alternativas comerciales disponibles para el despliegue.
 - Investigación teórica de la programación de tarjetas SIM 5G.
 - Programación de tarjetas SIM 5G.
 - Despliegue de la red 5G usando un dispositivo smartphone.
 - Despliegue de la red 5G usando un dispositivo enrutador.
 - Despliegue de la red 5G usando un módulo 5G:
 - Integración de módulo 5G a USB 3.0.
 - Conexión de antenas.
 - Instalación del módulo en una Raspberry Pi.
 - Desarrollo de la conectividad.
- Despliegue IoT de varios sensores sobre la red 5G.



1.4.- Cronograma

| TASK | START | END | feb 7, 2022 | | | | feb 14, 2022 | | | | feb 21, 2022 | | | | feb 28, 2022 | | | | mar 7, 2022 | | | | | | | | |
|---------------------------------------|---------|---------|-------------|---|---|----|--------------|----|----|----|--------------|----|----|----|--------------|---|---|---|-------------|---|---|----|---|---|---|---|--|
| | | | 7 | 8 | 9 | 10 | 14 | 15 | 16 | 17 | 21 | 22 | 23 | 24 | 28 | 1 | 2 | 3 | 7 | 8 | 9 | 10 | | | | | |
| | | | l | m | m | j | v | l | m | m | j | v | l | m | m | j | v | l | m | m | j | v | l | m | m | j | |
| Despliegue red LoRaWAN | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Busqueda de información | 2/7/22 | 2/11/22 | | | | | | | | | | | | | | | | | | | | | | | | | |
| Despliegue de gateway | 2/14/22 | 2/17/22 | | | | | | | | | | | | | | | | | | | | | | | | | |
| Conexión sensor LHT65 | 2/18/22 | 2/22/22 | | | | | | | | | | | | | | | | | | | | | | | | | |
| Conexión sensor Cubecell | 2/23/22 | 2/28/22 | | | | | | | | | | | | | | | | | | | | | | | | | |
| Creación y conexión de plataforma IoT | 3/1/22 | 3/8/22 | | | | | | | | | | | | | | | | | | | | | | | | | |

| TASK | START | END | mar 28, 2022 | | | | abr 4, 2022 | | | | abr 11, 2022 | | | | abr 18, 2022 | | | | abr 25, 2022 | | | | may 2, 2022 | | | | may 9, 2022 | | | | may 16, 2022 | | | | may 23, 2022 | | | | |
|--|---------|---------|--------------|----|----|----|-------------|---|---|---|--------------|----|----|----|--------------|----|----|----|--------------|----|----|----|-------------|---|---|---|-------------|----|----|----|--------------|----|----|----|--------------|----|----|----|--|
| | | | 28 | 29 | 30 | 31 | 4 | 5 | 6 | 7 | 11 | 12 | 13 | 14 | 18 | 19 | 20 | 21 | 25 | 26 | 27 | 28 | 2 | 3 | 4 | 5 | 9 | 10 | 11 | 12 | 16 | 17 | 18 | 19 | 23 | 24 | 25 | 26 | |
| | | | l | m | m | j | v | l | m | m | j | v | l | m | m | j | v | l | m | m | j | v | l | m | m | j | v | l | m | m | j | v | l | m | m | j | v | l | |
| Despliegue red 5G | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Busqueda de información | 3/29/22 | 4/5/22 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Despliegue IoT sobre Ethernet | 4/11/22 | 4/15/22 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Programación de tarjetas SIM 5G | 4/25/22 | 5/3/22 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Despliegue IoT utilizando router NETGEAR | 5/4/22 | 5/5/22 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Despliegue IoT utilizando smartphone | 5/4/22 | 5/5/22 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Despliegue IoT utilizando módulo Quectel | 5/9/22 | 5/17/22 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Ampliación de despliegue IoT | 5/18/22 | 5/25/22 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

| TASK | START | END | mar 7, 2022 | | | | mar 14, 2022 | | | | mar 21, 2022 | | | | mar 28, 2022 | | | | abr 4, 2022 | | | | abr 11, 2022 | | | | abr 18, 2022 | | | | abr 25, 2022 | | | | may 2, 2022 | | | | may 9, 2022 | | | | may 16, 2022 | | | | may 23, 2022 | | | | may 30, 2022 | | | |
|-------------------------------|---------|---------|-------------|---|---|----|--------------|----|----|----|--------------|----|----|----|--------------|----|----|----|-------------|---|---|---|--------------|----|----|----|--------------|----|----|----|--------------|----|----|----|-------------|---|---|---|-------------|----|----|----|--------------|----|----|----|--------------|----|----|----|--------------|----|---|---|
| | | | 7 | 8 | 9 | 10 | 14 | 15 | 16 | 17 | 21 | 22 | 23 | 24 | 28 | 29 | 30 | 31 | 4 | 5 | 6 | 7 | 11 | 12 | 13 | 14 | 18 | 19 | 20 | 21 | 25 | 26 | 27 | 28 | 2 | 3 | 4 | 5 | 9 | 10 | 11 | 12 | 16 | 17 | 18 | 19 | 23 | 24 | 25 | 26 | 30 | 31 | 1 | 2 |
| | | | l | m | m | j | v | l | m | m | j | v | l | m | m | j | v | l | m | m | j | v | l | m | m | j | v | l | m | m | j | v | l | m | m | j | v | l | m | m | j | v | l | m | m | j | v | l | m | m | | | | |
| Documentación | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Redacción de documentación I | 3/9/22 | 3/16/22 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Redacción de documentación II | 5/26/22 | 5/31/22 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |



2.- RED LORAWAN.

La red LoRa o LoRaWAN es una red de bajo consumo y amplia cobertura, siendo capaz de proveer un rango de comunicación de entre 10 y 40 Km en ámbitos rurales y entre 1 y 5 Km en zonas urbanas. La red *Low Power Wide Area* (LPWAN) ha sido diseñada para la interconexión inalámbrica de una gran variedad de dispositivos, o “cosas”, operativos con batería capaces de conectarse a internet, además cumple los requerimientos necesarios para dar soporte a IoT, como son la comunicación bidireccional, seguridad *end-to-end*, movilidad y servicios de localización.

Esta tecnología está administrada por *Lora Alliance*, que es una asociación encargada de asegurar la interoperabilidad de los diferentes dispositivos con la red LoRa mediante unos certificados creados por la propia organización. El objetivo principal de Lora Alliance es estandarizar LPWAN y así mismo posibilitar el despliegue de un gran volumen de dispositivos IoT utilizando esta tecnología [2].

LoRa es una tecnología de capa física que modula su señal en las bandas ISM, o bandas de radio industriales, científicas y médicas, sub-GHz usando técnicas de espectro ensanchado. Además, esta tecnología utiliza bandas no licenciadas de ISM, como son la banda de 868 MHz en Europa o la de 433 MHz en Asia. Para adaptarse a la tasa de datos requerida y al rango de *tradeoff* LoRa utiliza seis factores de propagación, SF7 a SF12. Un mayor factor de propagación permite mayores rangos de cobertura a expensas de menores tasas de datos. En LoRa la tasa de datos varía entre 300bps y 50Kbps, dependiendo del factor de propagación y del ancho de banda del canal, pudiendo ser este ancho de banda de 250KHz y 125 KHz. La máxima longitud de paquete que admite esta tecnología es de 243 bytes. [3]

El funcionamiento de la red LoRa se basa en una topología de estrella compuesta por múltiples *Gateway*, los cuales se encargan de recibir los mensajes de los dispositivos finales, también denominados nodos, y reenviarlos al servidor de red central. Los *Gateway* actúan como un puente transparente, esto es que simplemente convierten los paquetes RF que reciben de los nodos en paquetes IP para su posterior envío al servidor y viceversa. En la Figura 1 se muestra el esquema de la estructura de red de LoRa.

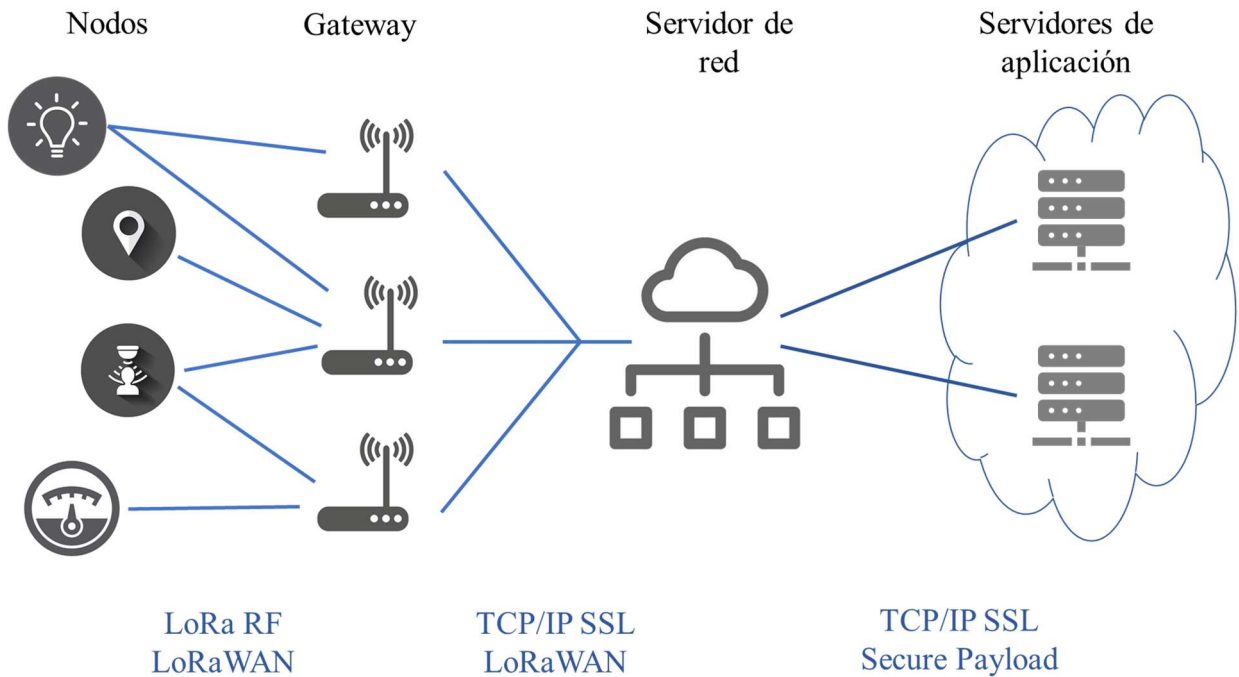


Figura 1. Estructura de red LoRaWAN.

En cuanto a la seguridad que ofrece esta red, LoRaWAN define dos capas de criptografía. La primera es una clave única de 128 bits, denominada *Network Session Key*, compartida entre los nodos y el servidor; y la segunda es una clave única de 128 bits, denominada *Application Session Key* o *AppSKey* compartida extremo a extremo en el nivel de aplicación. Además, utiliza algoritmos AES, *Advanced Encryption Standard*, para proveer autenticación e integridad a los paquetes.

2.1.- Despliegue IoT sobre red LoRa.

Para el despliegue de la red LoRa en el laboratorio se ha utilizado como servidor de red el dado por la plataforma *The Things Network*, ya que es el servidor LoRa de código abierto más ampliamente utilizado. Desde la página web de *The Things Network* es posible ver todos los Gateway disponibles a lo largo del mundo, tal como se muestra en la Figura 2. Buscando nuestra localización en el mapa observamos que existen varios Gateway cercanos que podríamos utilizar para la posterior conexión de dispositivos por LoRa, pero en este caso se ha optado por hacer todo el despliegue desde cero, lo cual implica implementar un Gateway propio. Se ha optado por realizar todo el despliegue con fines formativos de comprender mejor la tecnología y todos los componentes que son necesarios para la puesta en marcha de la misma.

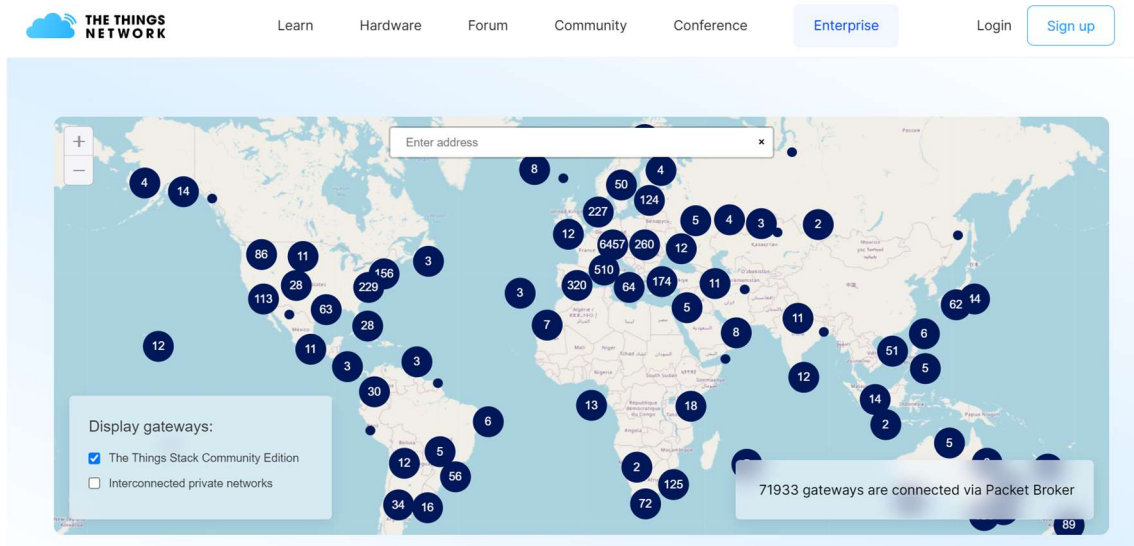


Figura 2. Mapa con Gateways de The Thing Network.

Para la realización del despliegue, el primer paso es registrarnos en la página de The Things Network o TTN y acceder a su consola o *Console* tal como aparece denominada en la página de TTN, donde posteriormente registraremos tanto el Gateway como los dispositivos finales o nodos que se van a utilizar en el despliegue, como se explicará en sucesivos apartados. En la Figura 3 se muestra la página inicial tras acceder a la Console de TTN.

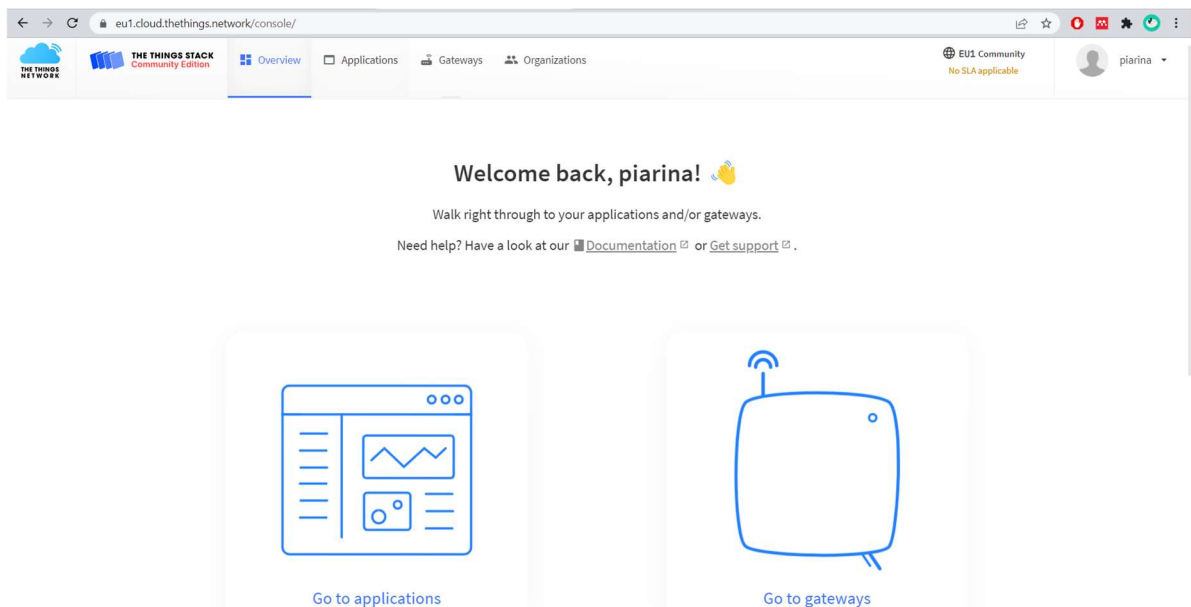


Figura 3. Console de TTN.



El Gateway que se va a utilizar es el *Mikrotik Routerboard LoRa8 kit* [4], al cual se le ha conectado una antena compatible [5], así mismo el Gateway se conecta a través de conexión ethernet a la red del laboratorio. Una vez conectado correctamente a la red, el dispositivo crea una red inalámbrica. Para poder configurar el Gateway debemos conectarnos a la red inalámbrica creada y acceder desde cualquier navegador a al url siguiente: <http://192.168.88.1>, tras esto nos aparece la pantalla de la Figura 4, el usuario inicial es admin y la contraseña quedará en blanco.

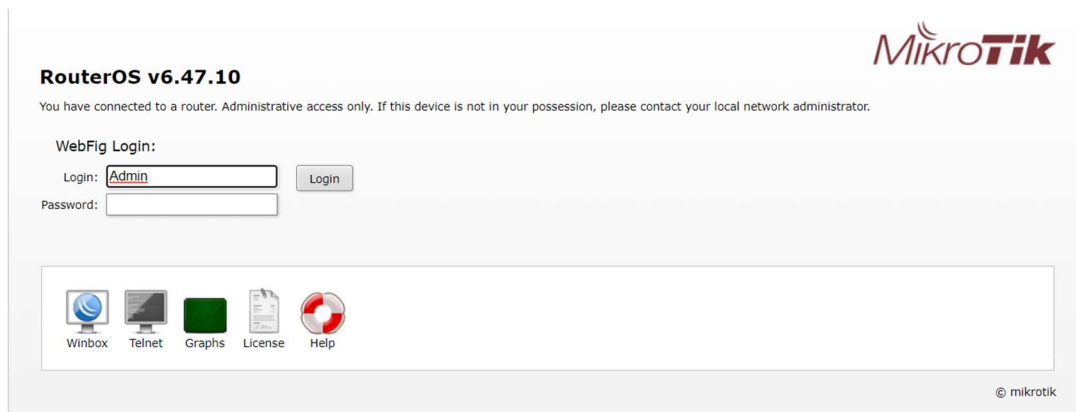


Figura 4. Configuración Gateway Mikrotik (1).

El primer paso para la configuración del Gateway es asegurarnos de que se conecta correctamente a la red de internet del laboratorio, para ello hay que darle una dirección IP, así como su correspondiente puerta de enlace y DNS. Esto se realiza modificando los datos correspondientes al apartado Internet que aparece a la derecha de la pantalla, tal como se muestra en la Figura 5 con los datos ya configurados. En nuestro caso particular tenemos una única dirección IP estática, que es la 156.35.117.16/24, cuya puerta de enlace corresponde con la IP 156.35.117.205 y su DNS es 156.35.14.2.

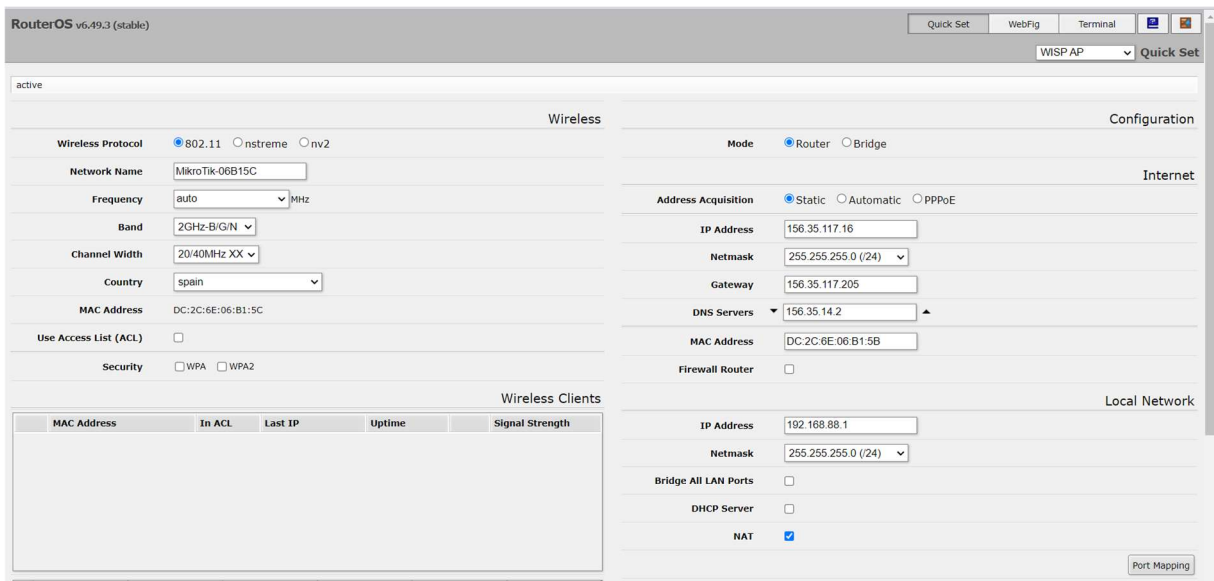


Figura 5. Configuración Gateway Mikrotik (2).

Una vez el dispositivo se conecte correctamente a la red, la dirección para seguir con su configuración cambiará a la dirección IP asignada, para acceder de nuevo a su configuración bastará con conectarnos en este caso a la url <http://156.35.117.16> desde cualquier otra red del laboratorio.

El siguiente paso es modificar la configuración de LoRa del dispositivo, para ello dentro de la pantalla WebFig de la configuración del Gateway hay que acceder a la opción LoRa que aparecerá en la columna de la izquierda, tal como se muestra en la Figura 6.

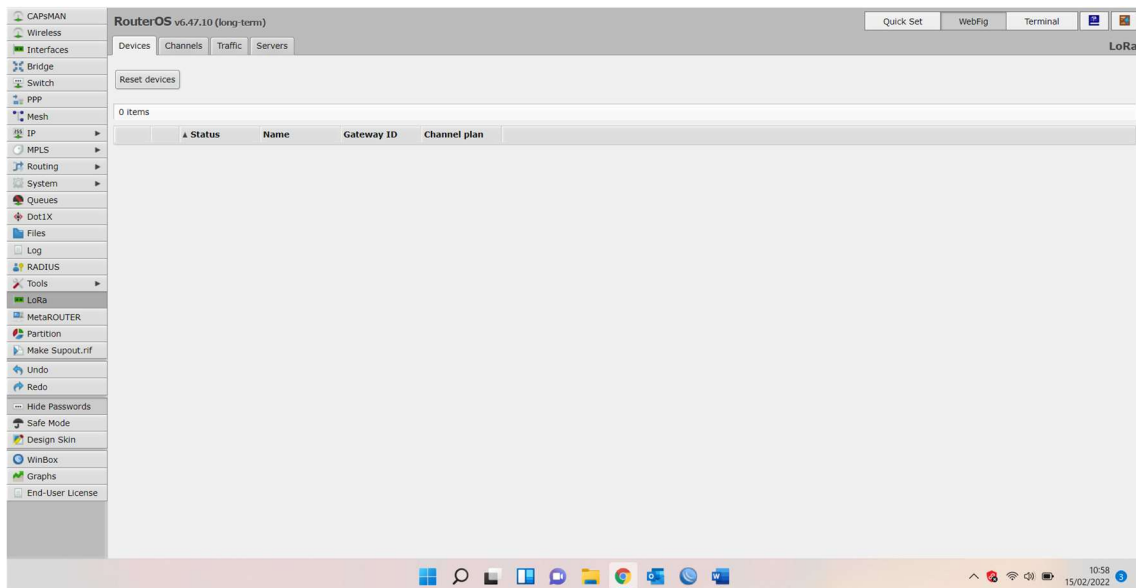


Figura 6. Configuración gateway Mikrotik (2).



Dentro de la configuración de LoRa lo primero es configurar el servidor, para ello en la pestaña *Servers* añadimos un nuevo servidor cuya dirección es la dada por TTN al acceder a la consola, en este caso al acceder a la consola europea la dirección es `eu1.cloud.thethings.network`, el nombre puede ser cualquiera identificativo, en este caso será `ttn`, en la Figura 7 se muestra esta parte de la configuración. Tras haber configurado el servidor el siguiente paso es configurar la pestaña de *Devices* con los datos correspondientes al dispositivo usado, los cuales se encuentran anotados en la parte de atrás del dispositivo, en la Figura 8 se muestra como viene esta información. Antes de acceder a modificar estos datos hay que deshabilitar el dispositivo, y volver a habilitarlo tras los cambios. La configuración de esta pestaña queda tal como se muestra en la Figura 9.

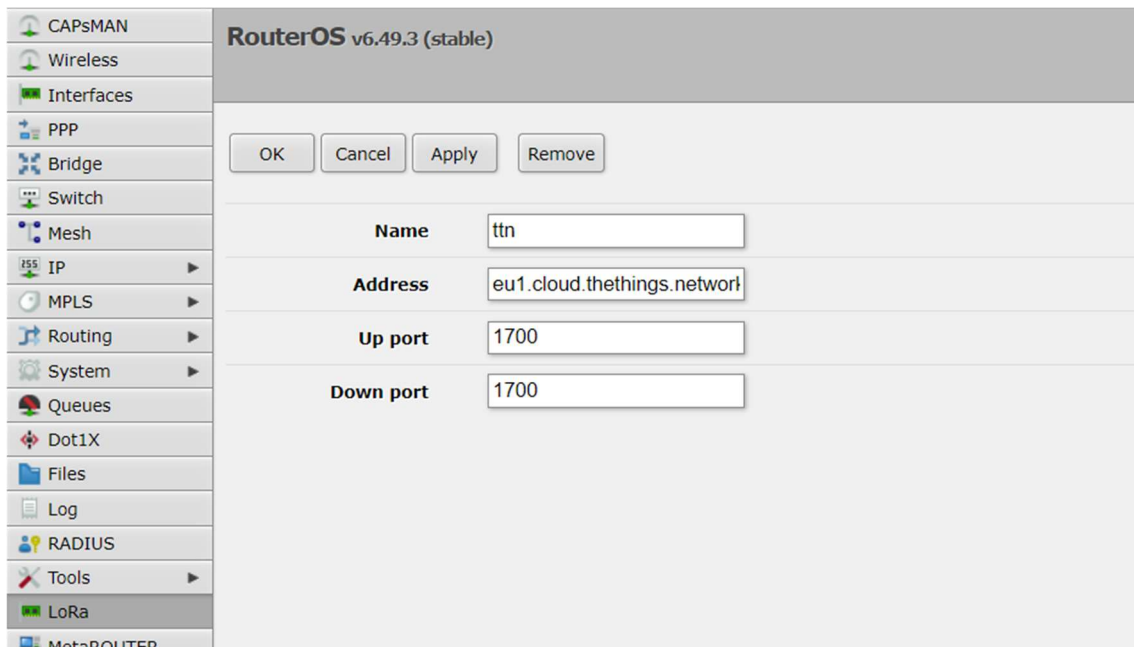


Figura 7. Configuración Gateway Mikrotik (3).

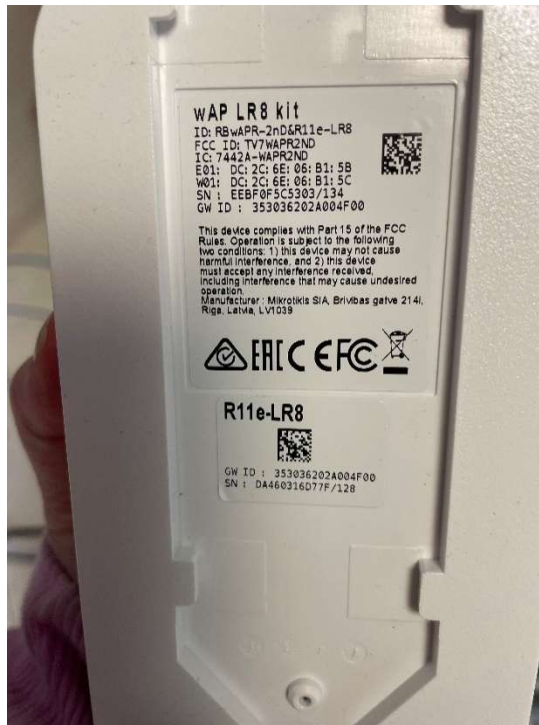


Figura 8. Información del dispositivo Mikrotik en su parte posterior.



RouterOS v6.49.3 (stable)

OK Cancel Apply

Enabled

Status Enabled

Name gateway-0

Hardware ID 353036202A004F00

Gateway ID 353036202A004F00

Firmware ID 6aaba37

Network Servers ttn

Channel plan EU 868

Antenna Gain 0 dB

Forward Valid Error Disabled

Network Public Private

Figura 9. Configuración gateway Mikrotik (4).

Con todos los pasos para la configuración del Gateway realizamos toca volver a la consola de TTN para realizar el registro del Gateway. Esto se hace simplemente accediendo al apartado Gateways de la consola y haciendo clic en +Add Gateway. Tras esto aparece la pantalla mostrada en la Figura 10, donde debemos introducir el identificador o id único que queremos darle al Gateway en este caso se llamará mk-gateway-p, también es posible darle un nombre y descripción si se viera necesario, en el campo Gateway EUI hay que introducir lo que el fabricante de Mikrotik denomina GW ID y que se encuentra en la parte posterior del dispositivo. Y el ultimo campo que en mi caso he completado es el correspondiente al plan de frecuencias, seleccionando la opción Europe 863-870 MHz (SF9 for RX2 -recommended) ya que vamos a trabajar en la banda de 868 MHz.



Add gateway

General settings

Gateway ID ^{*}

mk-gateway-p

Gateway EUI

35 30 36 20 1A 00 12 00

Gateway name

My new gateway

Gateway description

Description for my new gateway

Optional gateway description; can also be used to save notes about the gateway

Gateway Server address

eu1.cloud.thethings.network

The address of the Gateway Server to connect to

Require authenticated connection

Enabled

Controls whether this gateway may only connect if it uses an authenticated Basic Station or MQTT connection

Gateway status

Gateway status

Make status public

The status of this gateway may be visible to other users

Gateway location

Make location public

When set to public, the gateway location may be visible to other users of the network

Attributes

+ Add attributes

Attributes can be used to set arbitrary information about the entity, to be used by scripts, or simply for your own organization

LoRaWAN options

Frequency plan ^{*}

Europe 863-870 MHz (SF9 for RX2 - recommended)

Schedule downlink late

Enabled

Enable server-side buffer of downlink messages

Enforce duty cycle

Enabled

Recommended for all gateways in order to respect spectrum regulations

Schedule any time delay ^{*}

530 milliseconds

Configure gateway delay (minimum: 130ms, default: 530ms)

Figura 10. Configuración Gateway en TTN (1).

Figura 11. Configuración Gateway en TTN (2).



Si toda la configuración se ha realizado correctamente y todo ha ido bien, entonces al acceder al Gateway creado en TTN nos aparecerá algo parecido a lo mostrado en la Figura 12, donde se muestra tanto la información de creación del Gateway como la información relacionada con su conexión, justo debajo del ID, y los datos que le llegan en cada momento en la parte de *Live data*, a partir de aquí ya podemos conectar dispositivos a TTN a este Gateway. En el caso de haber ocurrido algún error durante la configuración que impidiera la correcta conexión del Gateway, aparecería un mensaje de *Failed to connected* debajo del ID.

La imagen muestra la interfaz de usuario de TTN (The Things Network) para un Gateway llamado MK-GATEWAY-P. El panel de configuración muestra:

- General information:** Gateway ID: mk-gateway-p, Gateway EUI: 35 39 36 29 2A 09 4F 09, Gateway description: None, Created at: Feb 15, 2022 11:59:45, Last updated at: Feb 16, 2022 13:38:39, Gateway Server address: eu1.cloud.thethings.network
- LoRaWAN information:** Frequency plan: EU_863_870_TTN, Global configuration: Download global_conf.json

El panel de **Live data** muestra un historial de actividad:

- 16:49:55: Receive uplink message DevAddr: 26 08 D9 0E, FCnt: 1
- 16:47:33: Receive gateway status Metrics: { ackr: 100, rxfr: 1, rxin: 1 }
- 16:47:13: Receive uplink message DevAddr: 26 08 23 E2, FCnt: 1
- 16:47:03: Receive gateway status Metrics: { ackr: 0, rxfr: 0, rxin: 0, }
- 16:46:33: Receive gateway status Metrics: { ackr: 0, rxfr: 0, rxin: 0, }
- 16:46:03: Receive gateway status Metrics: { ackr: 100, rxfr: 2, rxin: 2 }

Figura 12. Visualización del Gateway en TTN.

Para el envío de datos a TTN se ha optado por dos alternativas, una de ellas consiste en el envío de datos directamente de un sensor LoRa, el sensor LHT65 encargado de medir temperatura y humedad, y la otra alternativa es el envío de datos utilizando CubeCell, un dispositivo compatible con Arduino que permite el envío de información de diferentes sensores programables con Arduino. A continuación, se detalla la implementación de ambas alternativas.



2.1.1.- Captación de datos con sensor LHT65.

El sensor LHT65 [6], es un sensor de alto alcance LoRaWAN del fabricante Dragino que se encarga de recoger datos de temperatura y humedad, además en el caso de la temperatura envía dos datos de temperatura diferentes, uno correspondiente a la temperatura recogida del exterior, a la que denomina temp_SHT y otra correspondiente a la temperatura interna del sensor, a la que denomina temp_ds, también envía datos correspondientes a la tensión actual de su batería. Este sensor permite enviar datos a largas distancias y presenta alta inmunidad a interferencias mientras mantiene un bajo consumo de corriente. El LHT65 tiene incluido una batería no recargable de 2400mAh, con una estimación de uso de 10 años, debido a su bajo consumo, ya que el tiempo entre envíos de datos el dispositivo se mantiene en estado *Sleep*, es decir, prácticamente apagado. Una de sus principales ventajas es su compatibilidad con el estándar LoRa además de su capacidad de funcionamiento con cualquier Gateway LoRa estándar, por este momento se ha elegido como la primera alternativa a implementar dentro de la red LoRa creada [7].

Los primeros pasos para la captación de datos con este sensor se realizan en TTN. Antes de encenderlo y recoger los datos debemos crear una aplicación en TTN que incluya el sensor en sus *end devices*. El primer paso es abrir la consola de TTN y situarnos en la pantalla *Applications* de la misma, dentro de esta crearemos una nueva aplicación, ya que es nuestro primer dispositivo a conectar, de similar forma que creamos un nuevo Gateway, pero en este caso a la aplicación solo le debemos dar ID, y en caso de desearlo nombre y descripción, tal como se muestra en la Figura 13.

La imagen muestra la interfaz de usuario de TTN (The Things Network) en la sección de 'Applications'. El título de la página es 'Add application'. Hay un menú de navegación superior con 'Overview', 'Applications' (seleccionado), 'Gateways' y 'Organizations'. En la parte superior derecha, se muestra 'EU1 Community' con un enlace a 'No support plan' y un perfil de usuario 'piarina'. El formulario principal tiene los siguientes campos:

- Application ID***: un campo de texto con el valor 'my-new-application'.
- Application name**: un campo de texto con el valor 'My new application'.
- Description**: un campo de texto con el valor 'Description for my new application'. Debajo del campo hay un texto que dice: 'Optional application description; can also be used to save notes about the application'.

En la parte inferior del formulario hay un botón azul que dice 'Create application'.

Figura 13. Creación de aplicación en TTN.



El siguiente paso tras crear la aplicación, donde observaremos posteriormente los datos recibidos por TTN de nuestro dispositivo, es crear el dispositivo dentro de la aplicación, para ello debemos situarnos en la pestaña *End Devices*, que se muestra en el menú de la izquierda, Figura 14. Y dentro de esta pantalla añadiremos el dispositivo.

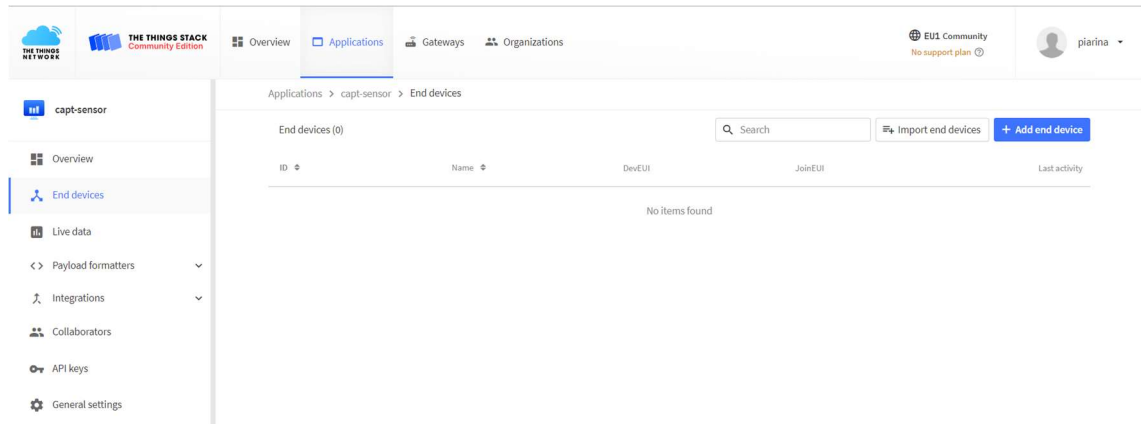


Figura 14. Creación de End device dentro de la aplicación de TTN.

Existen dos opciones para registrar el dispositivo dependiendo de si el dispositivo se encuentra registrado en el repositorio de dispositivos de LoRaWAN, Figura 15, o si hay que introducir sus datos manualmente, Figura 16. En este caso, aunque el sensor LHT65 si se encuentra dentro de este repositorio de dispositivos de LoRaWAN se ha decidido realizar su registro utilizando la alternativa manual, ya que todos los datos pedidos en ese apartado se encuentran en la caja del propio sensor como se puede observar en la Figura 17, por lo que no supone un esfuerzo extra realizarlo de esta forma. Los campos de datos de la caja y los de la configuración manual se corresponden tal cual, en nombre, los únicos datos extras a añadir es la banda de frecuencia, que es la misma que la del Gateway, y la versión LoRaWAN que se ha elegido la versión 1.0.2.

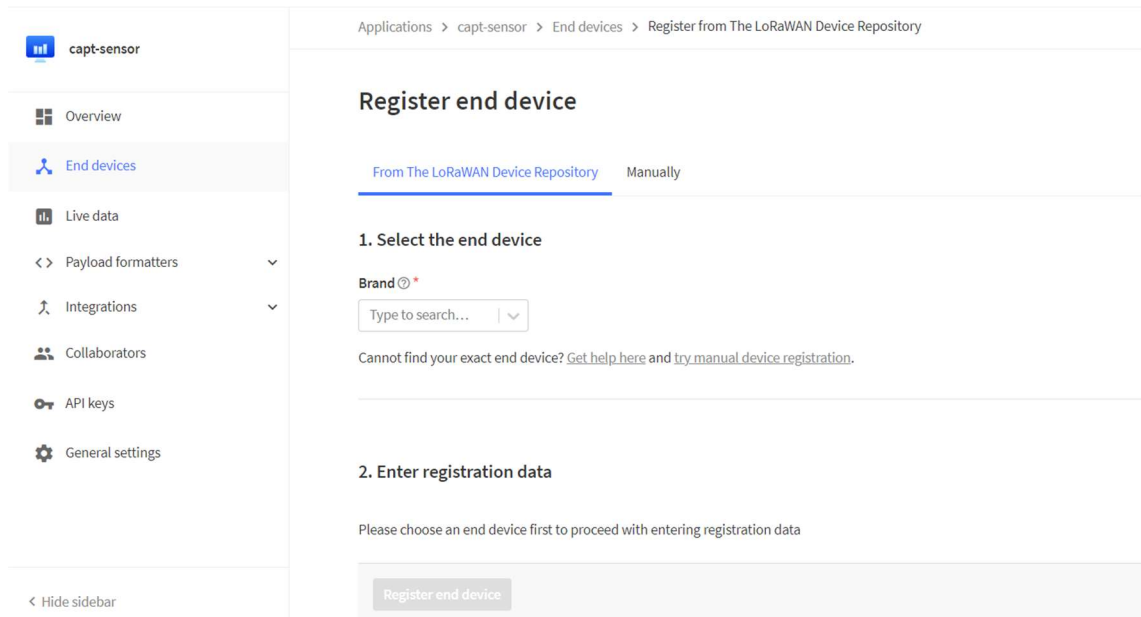


Figura 15. Registro de dispositivo del repositorio LoRaWAN.

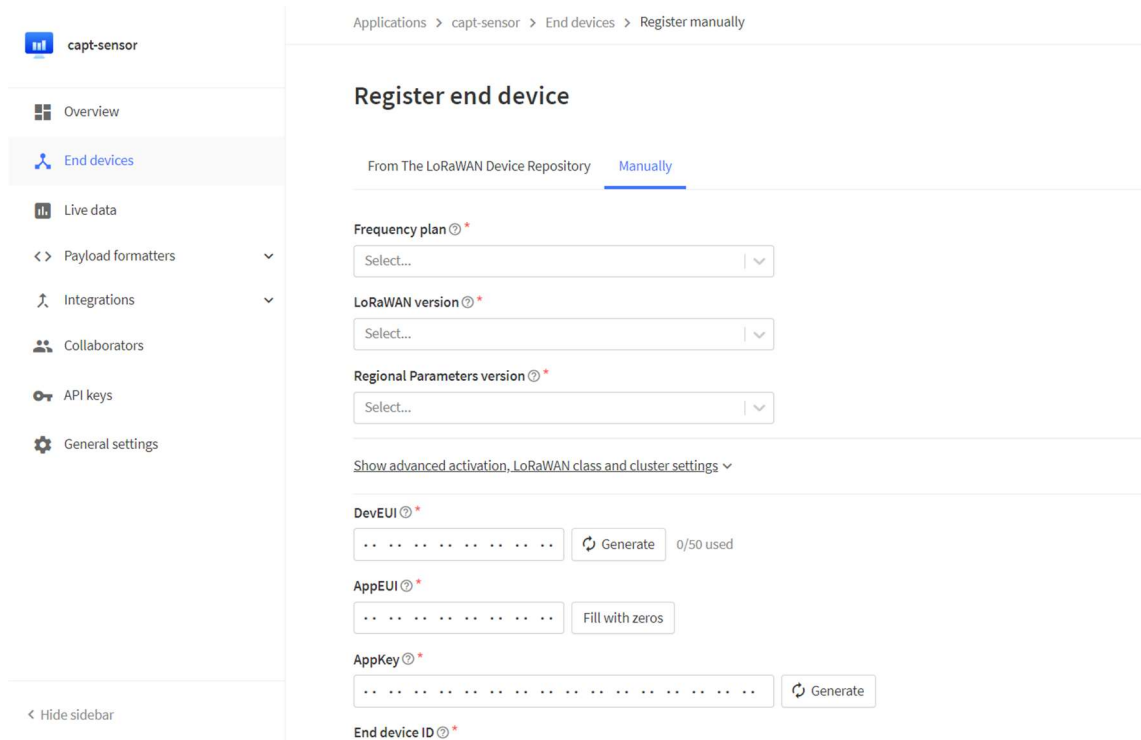


Figura 16. Registro de dispositivo de forma manual.



Figura 17. Datos dados por el fabricante para el registro del LHT65.

Una vez registrado el dispositivo el último paso sería encenderlo para comprobar que se ha configurado todo correctamente, este sensor no precisa de ningún otro paso añadido. El encendido del mismo se realiza, tal como se explica en su manual de instrucciones, pulsando el botón ACT, situado en la parte inferior, durante un tiempo superior a 3s, tras lo cual comenzará a parpadear un led verde, indicando que está intentando unirse a la red LoRaWAN, si todo va bien el led verde se mantendrá fijo durante 5s indicando que el sensor se ha unido correctamente a la red.

También podemos comprobar que su conexión se ha realizado correctamente desde TTN, dentro del dispositivo creado, en la pestaña Live Data se mostrarán los mensajes enviados por el dispositivo, así como se mostrará de manera general si el dispositivo sigue activo y cuál fue su último momento de actividad.

En este momento los datos de temperatura y humedad visibles en TTN serán simplemente una ristra de bytes, para poder sacar la información de esos bytes hace falta otro paso más. Este paso consiste en realizar un formateo de la carga útil, y se realiza en la pestaña Payload formatters dentro del menú del dispositivo, en la Figura 18 se muestra esta pestaña. Ya con el código en formato JSON necesario para la correcta decodificación de los bytes, en este caso el código venía dado también por el fabricante en su manual de instrucciones. En el caso de querer comprobar que el código es el correcto para los bytes recogidos, en la parte de Test, se introducen los bytes recibidos directamente en el apartado de Byte payload y se puede testear el resultado del código introducido.



Figura 18. Pestaña Payload formatters para la decodificación de la información.

Finalmente, ya tendríamos el dispositivo conectado a nuestra red LoRaWAN y podemos observar los datos de temperatura y humedad que envía desde la pantalla inicial del dispositivo creado en TTN, Figura 19. Durante su continuo funcionamiento se ha visto que el sensor LHT65 envía sus datos cada 20 minutos, debido a su característica principal de bajo consumo, esto se podría cambiar accediendo con un Arduino al dispositivo y programándolo, en caso de que se necesitara que se enviara la información cada menos tiempo.

Figura 19. Sensor LHT65 en funcionamiento en TTN.



2.1.2.- Captación de datos con CubeCell.

La segunda alternativa implementada para enviar datos a la red LoRaWAN es utilizar el dispositivo de Heltec CubeCell [8], al cual se le ha conectado el sensor BME/BMP280 [9]. En la Figura 20 se muestra una foto del CubeCell utilizado con el sensor BMP280 soldado directamente a las patillas correspondientes del dispositivo.



Figura 20. Imagen del CubeCell utilizado con el sensor BME/BMP280.

El dispositivo CubeCell nos permite crear un nodo LoRa al cual le podemos conectar los sensores que deseemos que sean compatibles con Arduino, debido esto a que una de las ventajas de CubeCell es que se puede programar utilizando Arduino. Igual que pasaba en el caso del sensor LHT65, CubeCell es un dispositivo de bajo consumo [10].

El sensor BMP280 utilizado en este caso se encarga de medir presión, temperatura y humedad [9]. Una de las ventajas de utilizar este sensor es que el código correspondiente a su uso para el envío de información a TTN se encuentra dentro de los ejemplos descargados tras añadir el dispositivo CubeCell al programa de Arduino, como se mostrará más adelante en este apartado.



Para poder empezar a trabajar con CubeCell, el primer paso es descargar el módulo correspondiente en Arduino, para ello se han seguido los pasos mostrados en la página oficial de Heltec [11].

Una vez se ha descargado correctamente el módulo necesario para trabajar en Arduino con la placa CubeCell, seleccionamos la placa y en el menú de herramientas debería salir los parámetros mostrados en la Figura 21. Tras la adición del módulo correspondiente a Arduino se observa que no solo se ha añadido la extensión para trabajar con la placa, sino que también se han añadido gran cantidad de códigos ejemplos que podemos ejecutar en nuestra placa CubeCell, Figura 22.

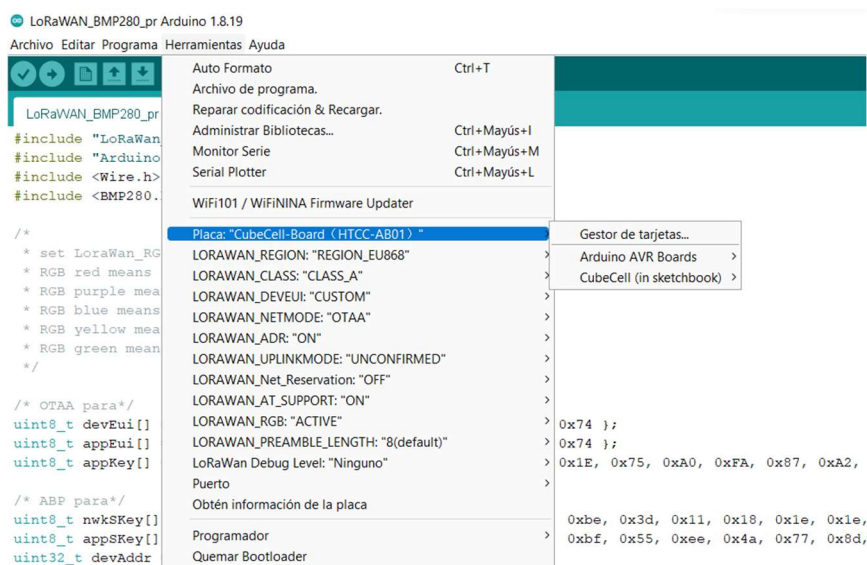


Figura 21. Placa CubeCell en Arduino.

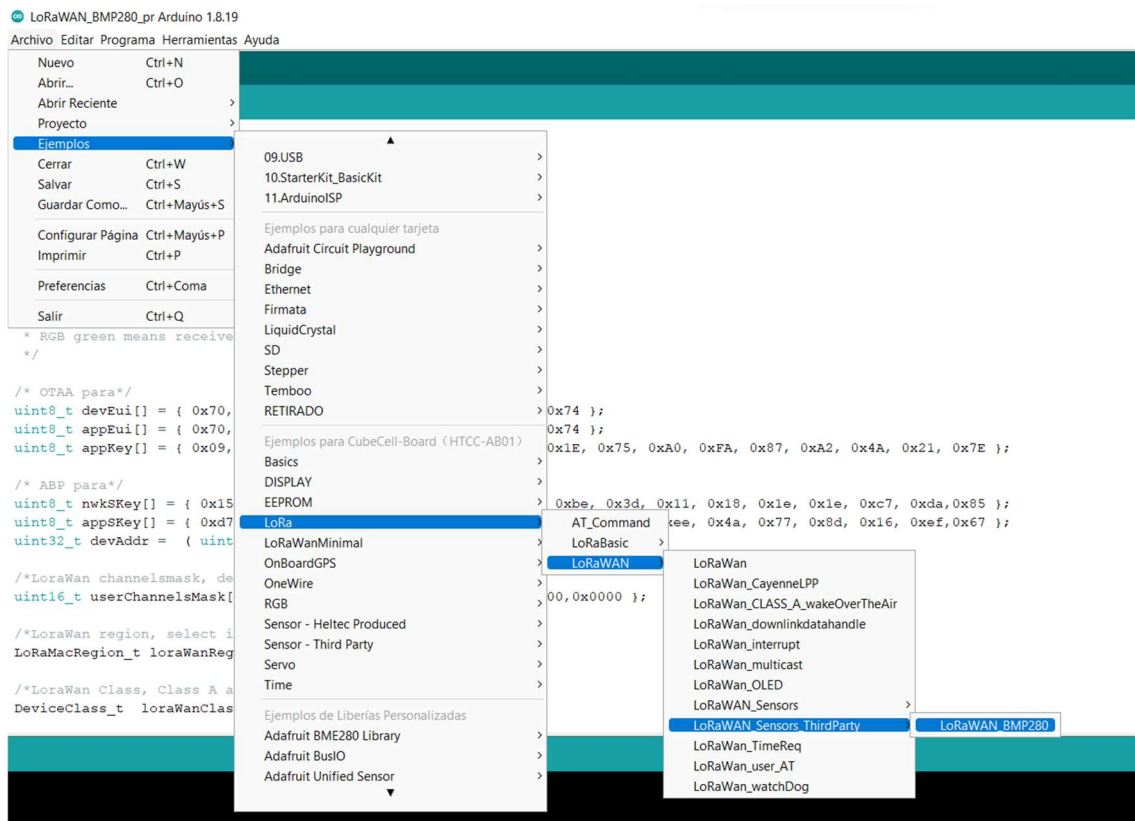


Figura 22. Códigos ejemplos para ejecutar en CubeCell.

Dentro del código de ejemplo disponible para el sensor BMP280 solo hace falta cambiar un par de líneas para su correcto funcionamiento. Lo primero que debemos modificar son los datos correspondientes a los parámetros OTAA, donde introduciremos los datos de `devEui`, `appEui` y `appKey` que deseemos, siguiendo el modelo inicial dado, estos tres datos serán utilizados posteriormente para el registro del nuevo End device en TTN, por lo que será necesario tenerlos a mano.

Cambiando solo esos parámetros sería suficiente para realizar una correcta conexión LoRa, pero revisando el código nos damos cuenta de que se introduce un parámetro de humedad que no se llega a leer en ningún punto del código, por lo que se enviarían un valor 0, pasa lo mismo con otros tres parámetros más incluidos en el código a los que denomina `lux`, `co2` y `tvoc`. Por lo tanto, eliminamos todo el código relacionado con el envío de esta información vacía, y en este punto ya podríamos subir el código a nuestro CubeCell para comenzar su ejecución, pero antes de vamos a registrar el dispositivo en TTN, dentro de la aplicación creada previamente.

Para crear el nuevo dispositivo en TTN hay que seguir los mismos pasos que en el caso del sensor LHT65, presentado anteriormente, en este caso el registro debe ser manual, y los datos a introducir en los apartados de `DevEui`, `AppEui` y `AppKey` son



los mismos que habíamos incluido en el código de Arduino con el mismo nombre, el plan de frecuencia y la versión de LoRaWAN se escogen igual que en el caso del sensor LHT65, quedando la configuración del dispositivo tal como se muestra en la Figura 23.

Prueba de conexión de sensores

- Overview
- End devices
- Live data
- Payload formatters
- Integrations
- Collaborators
- API keys
- General settings

From The LoRaWAN Device Repository [Manually](#)

Frequency plan [ⓘ]*

Europe 863-870 MHz (SF9 for RX2 - recommended)

LoRaWAN version [ⓘ]*

LoRaWAN Specification 1.0.2

Regional Parameters version [ⓘ]*

RP001 Regional Parameters 1.0.2

Show advanced activation, LoRaWAN class and cluster settings

DevEUI [ⓘ]*

70 B3 D5 7E D0 04 D6 74 0/50 used

AppEUI [ⓘ]*

70 B3 D5 7E D4 44 D6 74

AppKey [ⓘ]*

09 E8 E0 E1 FD 67 FC 1E 75 A0 FA 87 A2 4A 21 7E

End device ID [ⓘ]*

cubecell-1

This value is automatically prefilled using the DevEUI

After registration

View registered end device

Figura 23. Registro de dispositivo CubeCell en TTN.

Tras realizar el registro del dispositivo, el siguiente paso sería subir el código de Arduino al CubeCell y comprobar que se envía la información correctamente a TTN. Al igual que se mostró en el caso del sensor LHT65, es necesario decodificar la información recibida, ya que esta llega en formato byte. Para ello en el apartado de `Payload formatters` se ha introducido el código mostrado en la Figura 24.



Uplink
Downlink

Prueba de conexión de sensores

- Overview
- End devices
- Live data
- Payload formatters
- Integrations
- Collaborators
- API keys
- General settings

Setup

Formatter type*
Custom Javascript formatter

Formatter code*

```

1 function bytesToFloat(by) {
2   var bits = by[3] << 24 | by[2] << 16 | by[1] << 8 | by[0];
3   var sign = (bits >>> 31 === 0) ? 1.0 : -1.0;
4   var e = bits >>> 23 & 0xff;
5   var m = (e === 0) ? (bits & 0x7ffff) << 1 : (bits & 0x7ffff) | 0x1;
6   var f = sign * m * Math.pow(2, e - 150);
7   return f;
8 }
9
10 function Decoder(bytes, port) {
11
12   // Decode an uplink message from a buffer
13   // (array) of bytes to an object of fields.
14   var decoded = {};
15
16   if (port === 2) {
17     var i = 0;
18     decoded.temperature = Number(bytesToFloat(bytes.slice(i, i + 4)));
19     decoded.altitude = Number(bytesToFloat(bytes.slice(i, i + 4)).toFixed(2));
20     decoded.pressure = Number(bytesToFloat(bytes.slice(i, i + 4)).toFixed(2));
21     decoded.battery = Number(((bytes[12] << 8) | bytes[13]).toFixed(0));
22   }
23
24   return decoded;
25 }

```

Test

Byte payload FPort

1F 85 CD 41 9D 4D 2D C0 D2 64 7D 44 0E B 2 Test decoder

Decoded test payload

```

{
  "altitude": -2.71,
  "battery": 3.774,
  "pressure": 1013.58,
  "temperature": 25.69
}

```

Complete uplink data

```

{
  "f_port": 2,
  "rx_payload": "H4XNQZ1NLCDSZHIEdz4=",
  "decoded_payload": {
    "altitude": -2.71,
    "battery": 3.774,
    "pressure": 1013.58,
    "temperature": 25.69
  },
  "rx_metadata": [
  ]
}

```

✔ Payload is valid

Figura 24. Código de decodificación para el sensor BMP280.

Una vez con todo en correcto funcionamiento, se observaba desde Live Data que el envío de información se hacía cada pocos segundos, lo cual parecía innecesario en principio al ser datos que no van a variar tan rápidamente, además de suponer este envío continuo de información un consumo más elevado del deseado en este tipo de despliegues. Por tanto, dentro del código se realizó la siguiente modificación, Figura 25, para incluir un retardo, con el comando *delay*, de 5 minutos tras enviar la información. De esta forma el CubeCell enviaría los datos del sensor cada 5 minutos en lugar de hacerlo casi continuamente.

Estudio y comparativa de redes IoT desplegadas sobre LoRA y 5G: estudio de redes

Piarina Cañizo Otero

Página 25 de 101



LoRaWAN_BMP280_pr

```
case DEVICE_STATE_JOIN:
{
  LoRaWAN.join();
  break;
}
case DEVICE_STATE_SEND:
{
  prepareTxFrame( appPort );
  LoRaWAN.send();
  delay(300000); /*envio de informacion cada 5 minutos*/
  deviceState = DEVICE_STATE_CYCLE;
  break;
}
case DEVICE_STATE_CYCLE:
{
  // Schedule next packet transmission
  txDutyCycleTime = appTxDutyCycle + randr( 0, APP_TX_DUTYCYCLE_RND );
  LoRaWAN.cycle(txDutyCycleTime);
  deviceState = DEVICE_STATE_SLEEP;
  break;
}
case DEVICE_STATE_SLEEP:
{
  LoRaWAN.sleep();
  break;
}
default:
{
  deviceState = DEVICE_STATE_INIT;
  break;
}
}
}
```

Figura 25. Modificación del código de Arduino para retrasar el envío de datos.



2.2.- Envío de datos a base de datos Influxdb.

El siguiente paso tras tener los dispositivos deseados funcionando en la red LoRaWAN con conexión a TTN es reenviar los valores recogidos por TTN a una tercera aplicación que los almacene y muestre, para realizar un posterior análisis si fuera necesario.

Para esta tarea se ha utilizado la base de datos *Influxdb*, en su versión *cloud* a la cual es posible acceder en cualquier momento desde un navegador y no es necesario descargar nada más en ninguna máquina, sin embargo, como se verá más adelante si es necesario utilizar otro programa, denominado *Telegraf*, encargado de redirigir los datos de TTN a la cuenta de Influxdb.

Todo esto se ha implementado sobre una Raspberry Pi en continuo funcionamiento, ya que en el momento en el que el programa Telegraf deja de estar corriendo los datos de TTN dejan de enviarse a Influxdb, y nos interesará recibir los datos de manera continua. La Raspberry Pi que se va a utilizar en este caso es la Raspberry Pi 4 Computer Model B, con dicha RPi conectada a internet a través de cable de tipo Ethernet conectado a un router.

El primer paso es registrarnos en la plataforma cloud de Influxdb, accediendo a la siguiente url <https://cloud2.influxdata.com/signup>, donde nos permite registrarnos de forma gratuita. Una vez hemos iniciado correctamente nos aparece la pantalla mostrada en la Figura 26. Dentro de Influxdb empezaremos creando el *bucket* donde se almacenarán los datos, para ello en la pantalla de la Figura 26 accedemos a Load Your Data, y con esto se abre la pantalla de la Figura 27, donde seleccionaremos del menú el tipo de fuente de donde recogeremos los datos, en nuestro caso será MQTT Consumer, ya que vamos a utilizar la integración MQTT de TTN para acceder a los datos de TTN.

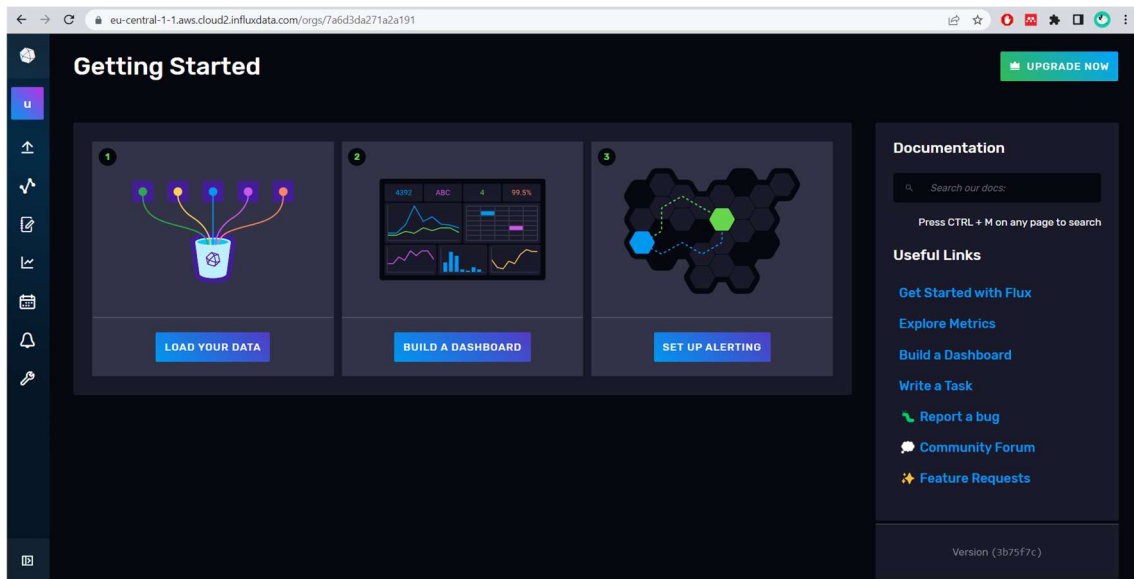


Figura 26. Página de inicio de Influxdb.

El protocolo MQTT, *Message Queuing Telemetry Transport*, es un protocolo máquina a máquina o M2M del tipo publicador suscriptor, esto es que uno de los extremos de la comunicación se encarga de suministrar información sobre un tema sin necesidad de que nadie le pida dicha información, este recibe el nombre de publicador, y en el otro extremo está el suscriptor, el cual se encarga de consumir la información publicada, decidiendo que información de la ofrecida por el suscriptor le interesa. [12] MQTT es uno de los protocolos más utilizado en comunicaciones IoT. [13]

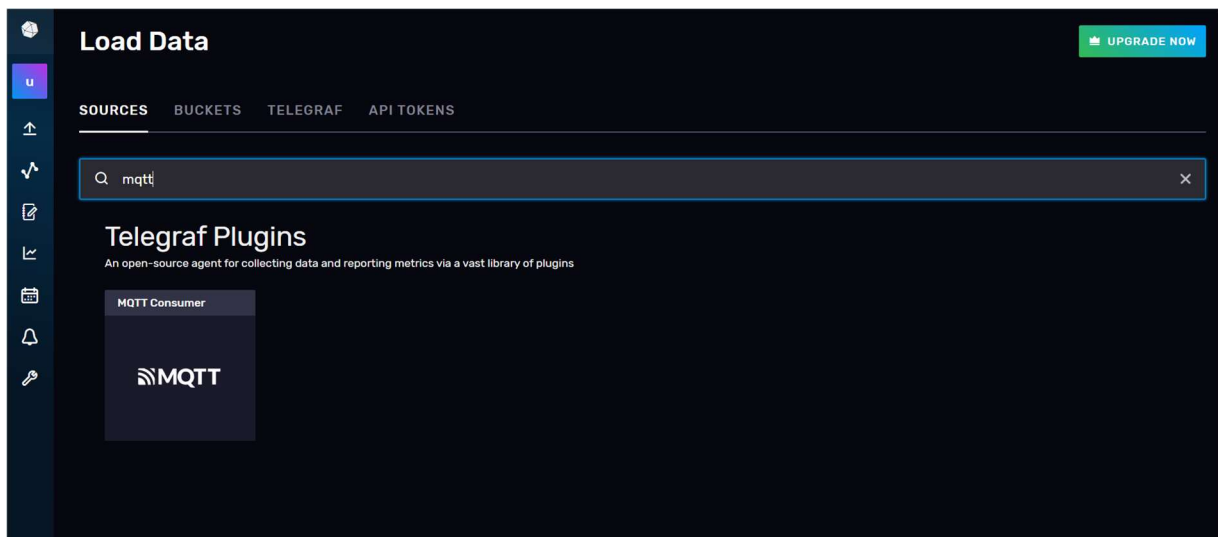


Figura 27. Fuente de datos en Influxdb.



Tras clicar en la fuente que hemos seleccionado, aparecerá una nueva pantalla, Figura 28, donde debemos seleccionar `Use this plugin>Create a new configuration`, lo cual abre un nuevo menú, Figura 29, donde daremos nombre al plugin y seleccionaremos el bucket al cual va a afectar, es decir, el bucket donde se van a almacenar los datos recibidos de MQTT, en el caso de no haber creado el bucket previamente, como se está describiendo, debemos seleccionar `Create a bucket` y podremos crearlo al momento, Figura 30. Para crearlo basta con darle un nombre e indicar cuanto tiempo va a almacenar los datos, en este caso se deja el valor por defecto de 30 días, al ser el mayor número de días dado como opción para las cuentas gratuitas.

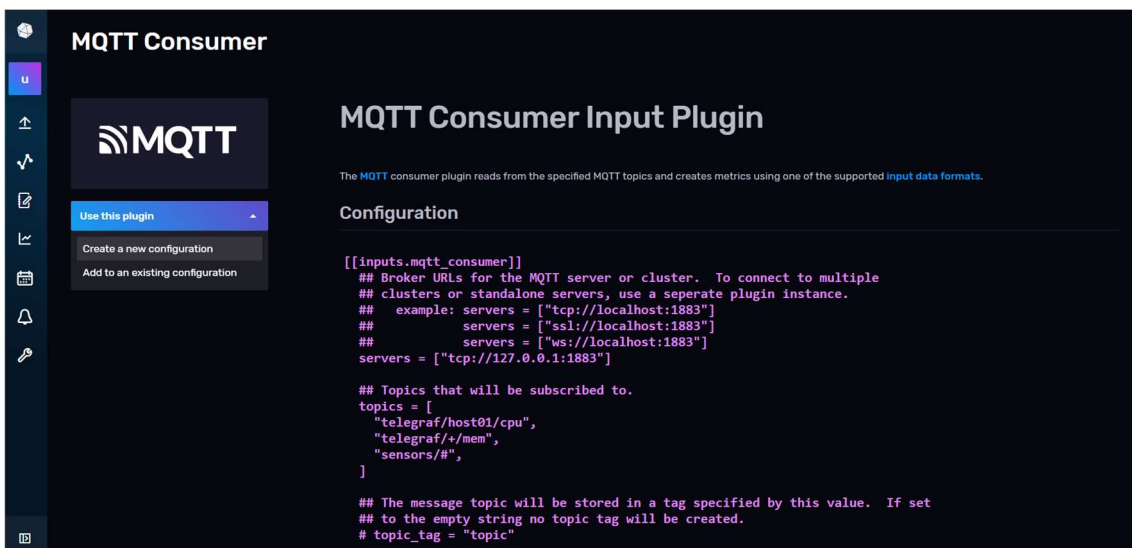


Figura 28. Menú MQTT Consumer.

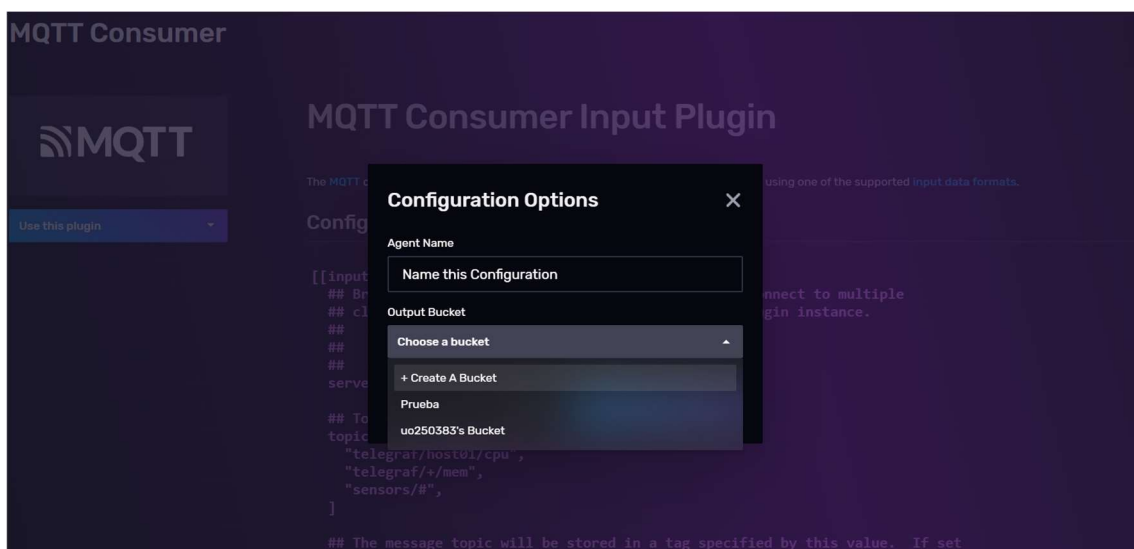


Figura 29. Configuración plugin MQTT Consumer.

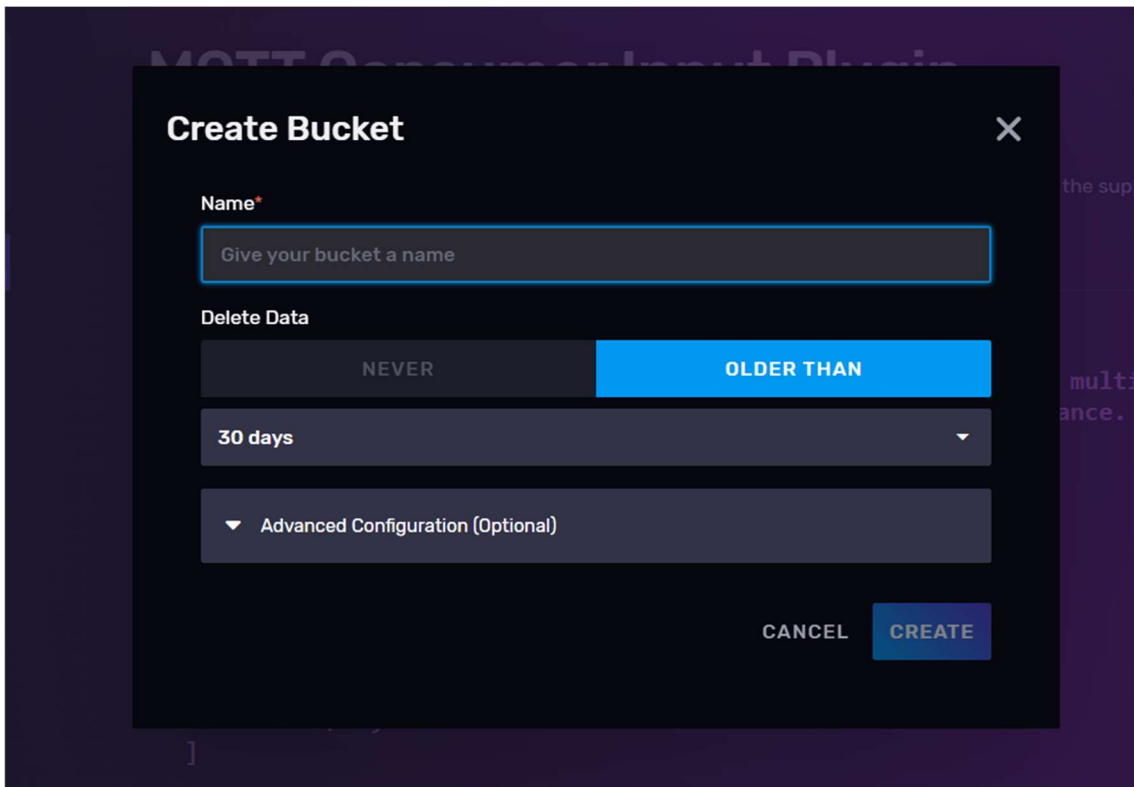


Figura 30. Creación del bucket.

Dentro de Influxdb el siguiente paso es crear el API Token, en la pestaña de API Tokens, Figura 31, para permitir accesos de escritura y lectura usando diferentes tokens.

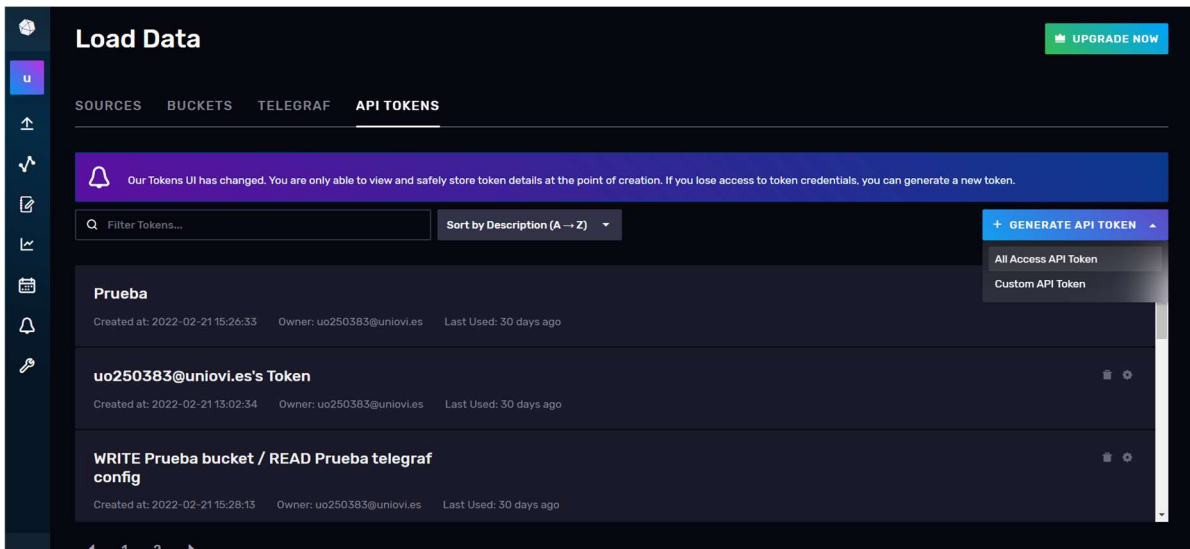


Figura 31. Creación de API Token.



Finalmente, el último paso de configuración a realizar dentro de Influxdb es crear una configuración Telegraf, esto se hace dentro de la pestaña Telegraf, Figura 32, en Create Configuration. Tras hacer clic en esta opción se abre la ventana de la Figura 33, donde en las opciones dadas seleccionamos nuestro bucket y como fuente MQTT Consumer de igual forma que hicimos antes. Esto nos creará una configuración que utilizaremos más adelante al descargar Telegraf en la RPi, Figura 34, y guardaremos la configuración.

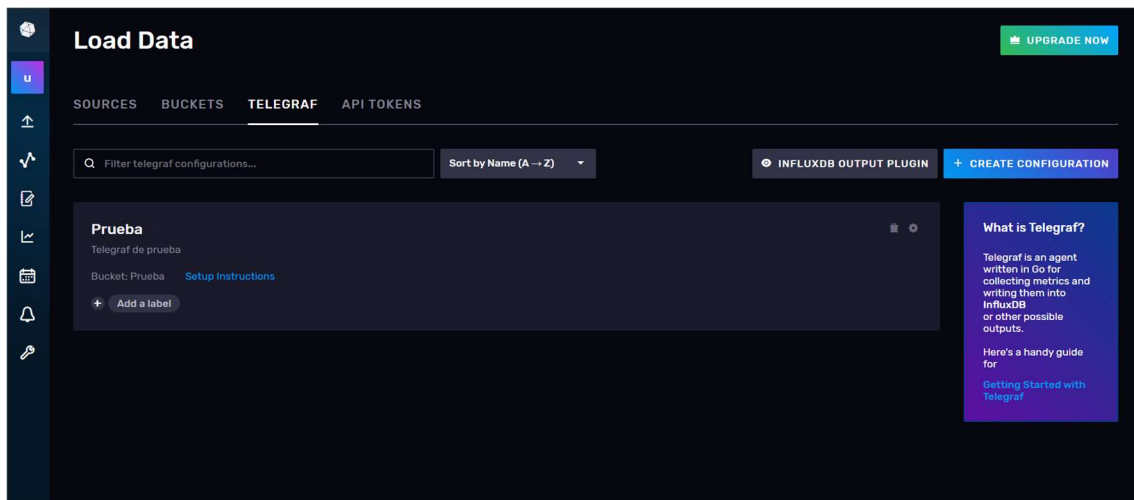


Figura 32. Creación de configuración Telegraf (1).

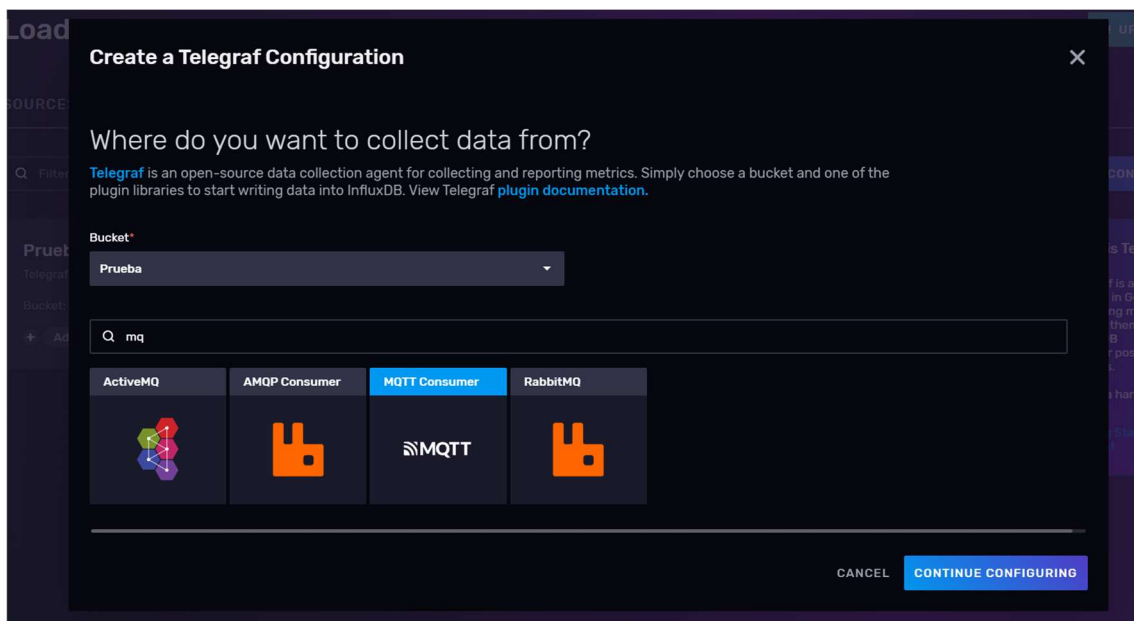


Figura 33. Creación de configuración Telegraf (2).

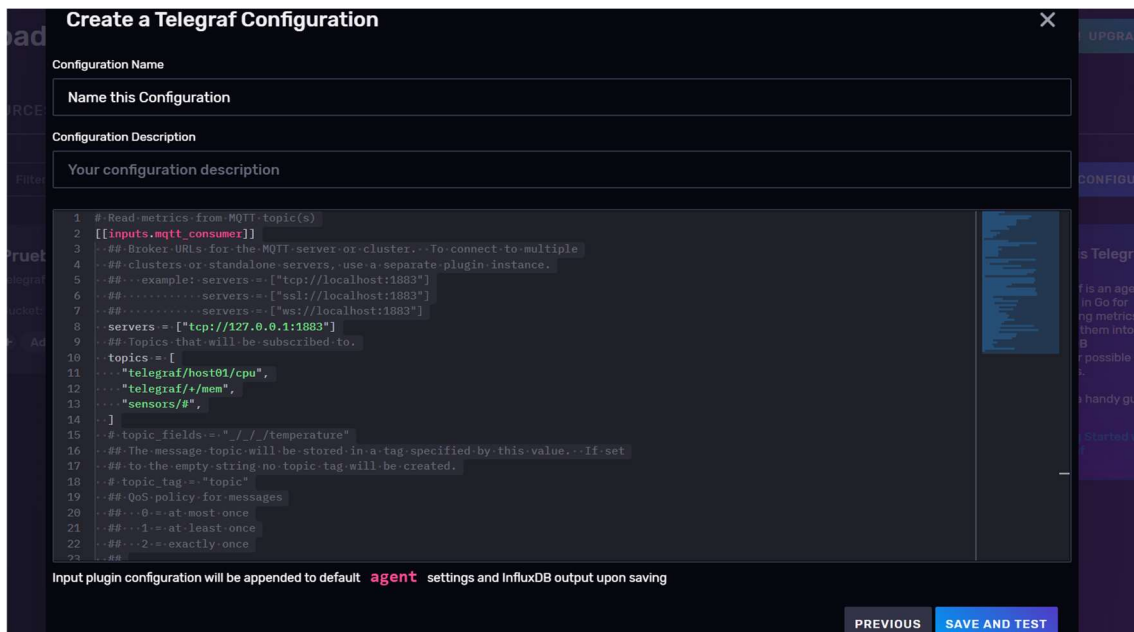


Figura 34. Creación de configuración Telegraf (3).

Con esto concluirían los pasos de configuración de Influxdb para poder recibir los datos tras el arranque del programa en la RPi. El siguiente paso es la instalación en Telegraf sobre la RPi, para ello introduciremos los siguientes comandos en su consola:

```
$ curl -sL https://repos.influxdata.com/influxdb.key | sudo
apt-key add -
```

```
$ echo "deb https://repos.influxdata.com/debian stretch
stable" | sudo tee /etc/apt/sources.list.d/influxdb.list
```

```
$ sudo apt-get update
$ sudo apt-get install telegraf
```

Tras instalar Telegraf, debemos proceder a su configuración para su correcta conexión a TTN, de donde obtendrá los datos. Para esta configuración procedemos a modificar el fichero `telegraf.conf`, al cual accedemos en la siguiente ruta: `/etc/telegraf/telegraf.conf`. Necesitaremos permisos de administrador para poder modificar dicho fichero, por ello el comando a utilizar será `sudo nano /etc/telegraf/telegraf.conf`. Dentro de este fichero vamos a introducir los datos de configuración que se encuentran en la página de Influxdb, en la pestaña de Telegraf. Debemos modificar dos apartados, el primero es el apartado `[[inputs.mqtt_consumer]]`, con la información mostrada durante la



configuración de Telegraf, Figura 34, también se puede copiar después de su creación al hacer clic sobre el nombre de la configuración de Telegraf creada.

El otro apartado a modificar en el fichero de configuración comienza con `[[outputs.influxdb_v2]]`, y la información a introducir en este apartado se obtiene de la ventana que se despliega al hacer clic en `Influxdb Output Plugin` en la misma ventana de Telegraf dentro de Influxdb, como se puede ver en la Figura 32.

Tras realizar esta configuración el siguiente paso sería ya la puesta en marcha del servicio de Telegraf, para ello basta con seguir los pasos mostrados en la página de Influxdb, dentro de la ventana correspondiente a Telegraf, al hacer clic a la opción `Setup Instructions` dentro de la configuración de Telegraf correspondiente, con esto se abre la ventana, mostrada en la Figura 35, con los comandos necesarios a introducir en la RPi para arrancar el servicio. Habría que sustituir `<INFLUX_TOKEN>` por el API Token creado o generar uno nuevo que se puede copiar directamente desde esa ventana.

Con la realización de estos pasos ya tendríamos a Influxdb recibiendo los datos de TTN y almacenándolos en el bucket creado.

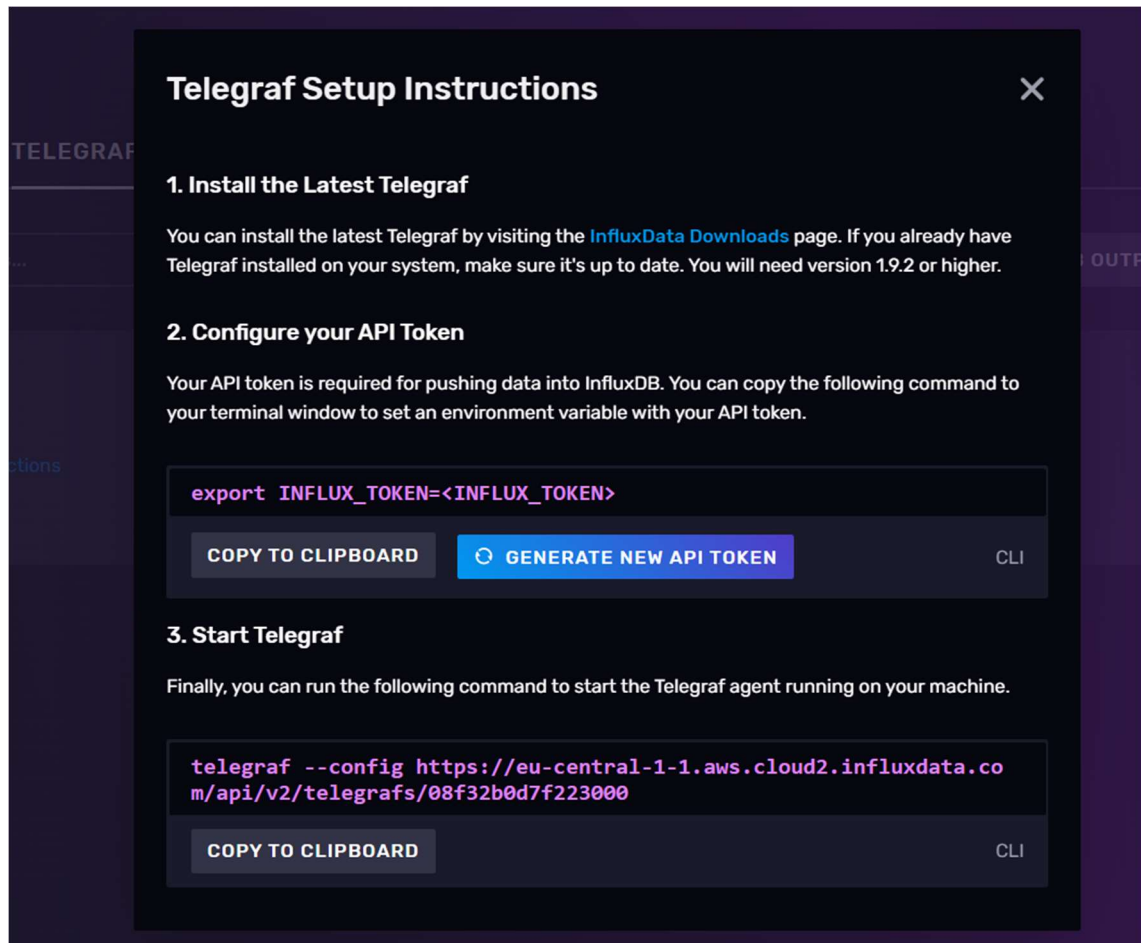


Figura 35. Instrucciones para arrancar el servicio Telegraf.

Influxdb además de hacer de base de datos almacenando los datos recibidos en un bucket, también nos permite visualizar gráficamente los datos recibidos creando un *dashboard*. Para ello simplemente nos desplazamos en el menú de la izquierda al apartado de Dashboard, donde podremos acceder a las dashboards ya creadas o crear una nueva, Figura 36. En este caso se han creado dos dashboards independientes, una para cada sensor utilizado. De igual forma que en cualquier dashboard se pueden crear diferentes gráficas para mostrar los valores deseados, en el caso de la dashboard Cubecell+BMP280 se han implementado tres gráficas para presentar los datos de temperatura, presión atmosférica y nivel de tensión de la batería del Cubecell, Figura 37. De igual forma, en el dashboard Sensor LHT65 se han implementado tres gráficas donde se presentan el nivel de tensión de la batería del sensor LHT65, y la temperatura y humedad recogida, Figura 38.

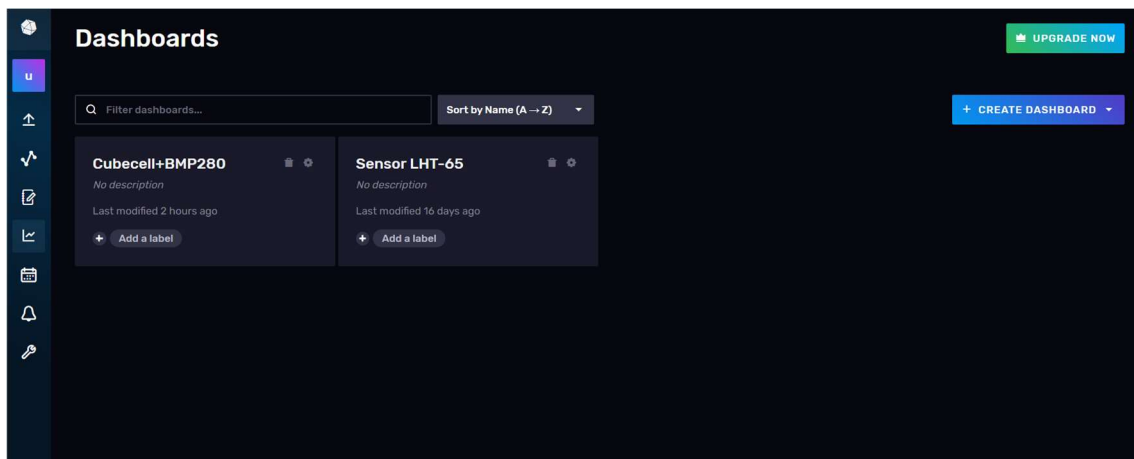


Figura 36. Pantalla de Dashboard de Influxdb.



Figura 37. Dashboard Cubecell+BMP280.



Figura 38. Dashboard Sensor LHT-65.



3.- RED 5G.

El Sector de Radiocomunicaciones de la Unión Internacional de Telecomunicaciones, ITU-R, definió en 2020 los requerimientos de 5G en base a tres diferentes casos de uso: banda ancha móvil mejorada (eMBB), comunicaciones masivas tipo máquina (mMTC) y comunicaciones ultra confiables de baja latencia (URLLC).

La banda ancha móvil mejorada o eMBB es el servicio por el cual se provee acceso inalámbrico a internet con tasas de datos más elevadas a la existente en tecnologías anteriores. En general, eMBB provisiona gran capacidad, alta movilidad y una conexión permanente a la red 5G. Las comunicaciones masivas tipo máquina o mMTC tienen como funcionalidad principal dar soporte a la tecnología IoT, ya que presenta gran capacidad de conexiones simultáneas, alrededor de 1 millón de dispositivos por Km². Finalmente, las comunicaciones ultra confiables de baja latencia o URLLC tienen como características principales la confiabilidad en la transmisión y el reducido retardo en transmisiones extremo a extremo. [14]

A continuación, se presentan los requerimientos definidos para 5G en relación con los diferentes aspectos a tener en cuenta:

- Tasa de datos. La tasa de datos, para poder dar soporte a eMBB, para el enlace de subida es de hasta 10 Gb/s, y para el enlace de bajada de hasta 20 Gb/s, experimentando el usuario unas tasas de datos de hasta 50 Mb/s y 100 Mb/s para los enlaces de subida y de bajada respectivamente.
- Latencia. La latencia de usuario, para poder dar soporte a eMBB debe ser de 4 ms o menos para el envío de un paquete de datos PDU, Unidad de datos de protocolo, a través de la capa física. [15] En el caso de dar soporte a URLLC esta latencia debe ser de 1ms. Mientras que en el caso de ambos la latencia de control, definida como el tiempo necesario para cambiar el estado del equipo.
- Densidad. El requerimiento de densidad de conexión de dispositivos, para poder dar soporte a las especificaciones de mMTC es de mínimo 1.000.000 dispositivos/Km².

Toda la información previa se ha obtenido del artículo con la siguiente referencia [16].

Además, la red 5G se encuentra desplegada sobre la banda de frecuencia de 700 MHz y de 3.5 GHz, así mismo se prevé desplegar esta tecnología sobre las frecuencias milimétricas o bandas mmWave de 26 GHz y 28 GHz [17].



Debido a la dificultad inicial que supone desplegar la red 5G desde cero se ha optado por desplegar primero el sistema IoT en otra red diferente, en este caso como se verá en siguientes apartados se optado por utilizar la red Ethernet, al tener previamente la RPi conectada a esta red. Esto ha sido realizado para asegurarnos inicialmente del correcto funcionamiento del despliegue IoT planteado en otra red antes de sustituir dicha red por la red 5G desplegada posteriormente, y así cerciorarnos que en caso de surgir algún problema en el despliegue IoT sobre 5G sería debido a problemas en la red y no problemas de los propios sensores o de su conexión con la plataforma IoT utilizada en este caso.

3.1.- Despliegue IoT sobre varias redes.

Durante este apartado describiremos los pasos realizados en el laboratorio para el despliegue de un entorno IoT implementado sobre diferentes redes disponibles en el laboratorio de la cátedra Thin5G de la Universidad de Oviedo.

3.1.1.- Despliegue IoT sobre red Ethernet.

Antes de comenzar el despliegue IoT sobre la red 5G se ha realizado el mismo despliegue sobre una red Ethernet, para posteriormente sustituir esta red por la red 5G.

Para este despliegue IoT se han utilizado dos sensores, un sensor de temperatura y humedad, del tipo DHT11 [18], y un sensor de intensidad lumínica o LDR [19], así como una RPi a la cual conectaremos los sensores para enviar sus datos a una plataforma IoT. La plataforma IoT elegida para este despliegue ha sido *Thingsboard IoT Gateway*. Estos sensores se conectan a la plataforma IoT mediante la red Ethernet, como ya se ha mencionado, utilizando el protocolo MQTT para el envío de la información, protocolo presentado anteriormente.

El primer paso ha sido la puesta en marcha de la plataforma IoT que va a recibir los datos. Thingsboard IoT Gateway es una plataforma IoT de código abierto para la recolección de datos, así como su procesado y visualización. Para su puesta en marcha hay que lanzar el servidor Thingsboard, es posible utilizar una demo en la nube y utilizarlo directamente, pero en nuestro caso, al disponer de una RPi donde podemos lanzar el servidor hemos optado por esta opción. Para poder lanzar el servidor de Thingsboard en nuestra RPI seguimos el tutorial dado por la propia plataforma, que se encuentra en la siguiente referencia [20].

Tras tener el servidor de Thingsboard corriendo podremos acceder a la plataforma utilizando un navegador con la siguiente url: <http://156.35.117.17:8080/>, siendo 156.35.117.17 la IP de acceso a internet de la RPi. Al acceder a esta url se nos muestra la



pantalla de la Figura 39, donde nos pide unos datos de inicio de sesión, en este caso estos datos son los siguientes, el usuario es tenant@thingsboard.org y la contraseña es *tenant*, lo cual viene también recogido en el tutorial [20]. Con este usuario accedemos a la plataforma como usuario administrador, existen otros dos usuarios más ya creados, uno de ellos tiene el papel de administrador de sistema, con usuario sysadmin@thingsboard.org y contraseña *sysadmin*, con el cual podríamos crear otras cuentas de tipo tenant o propietarias, y el último rol es el de simplemente cliente con el que no se pueden realizar modificaciones en la plataforma, simplemente ver lo creado, con usuario customer@thingsboard.org y contraseña *customer*. Una vez iniciada sesión en Thingsboard aparece la pantalla principal mostrada en la Figura 40.

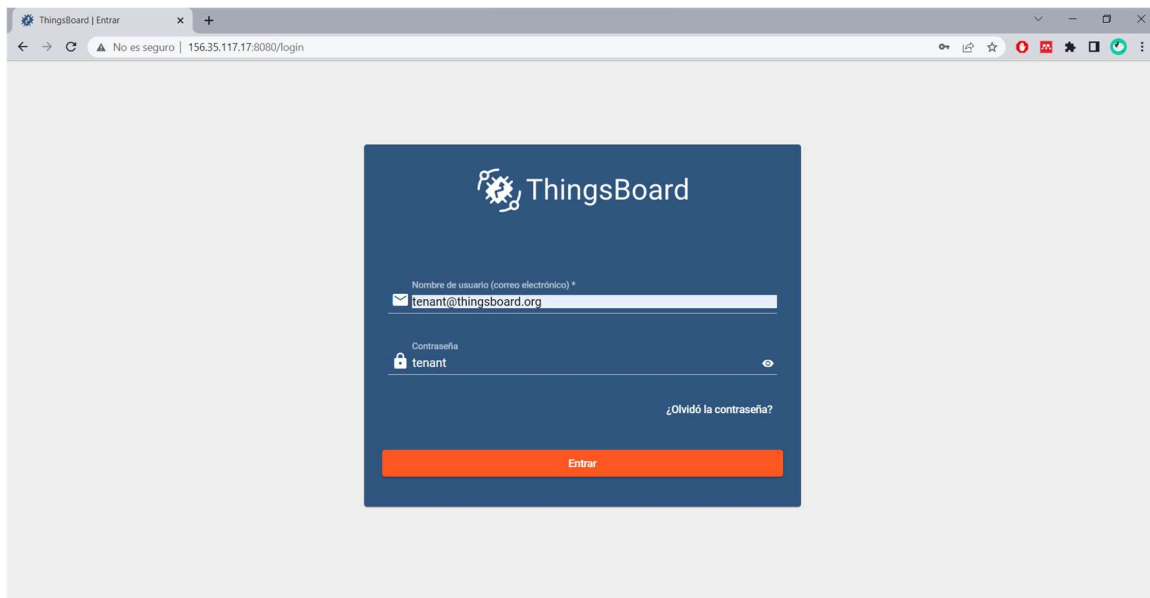


Figura 39. Pantalla inicio de sesión en Thingsboard.

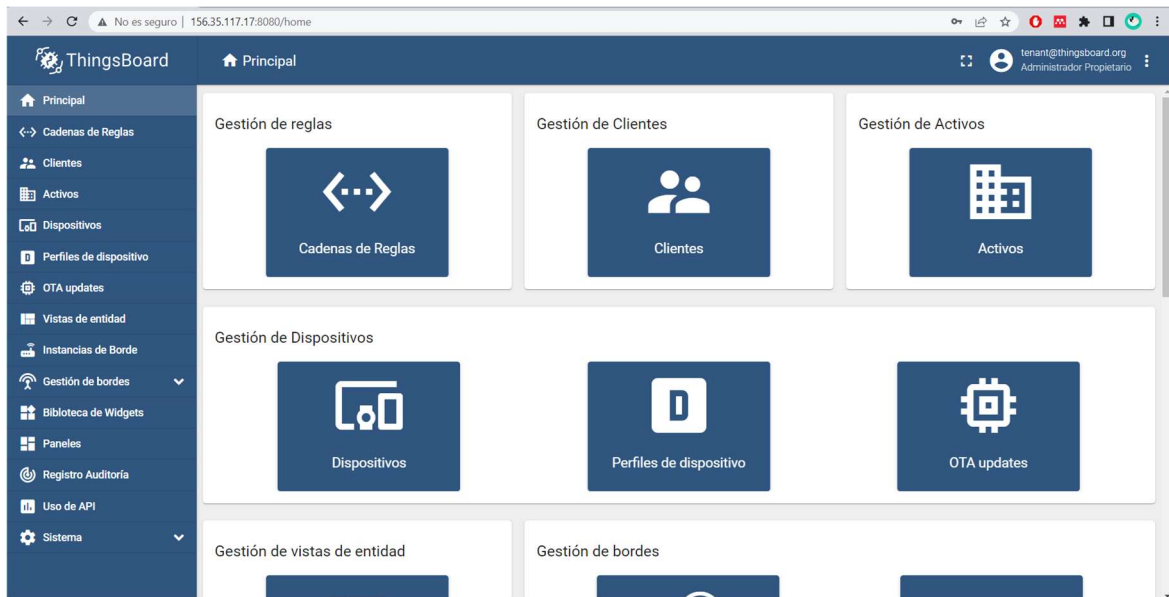


Figura 40. Pantalla inicial de Thingsboard.

El siguiente paso a realizar dentro de la plataforma es la creación de un nuevo dispositivo, el cual se corresponderá con la RPi, y los valores de los sensores se almacenarán bajo el nombre de este dispositivo. Para esto, en la pestaña **Dispositivos** hacemos clic en +, Figura 41, y creamos el dispositivo con el nombre deseado, tal como se muestra en la Figura 42.

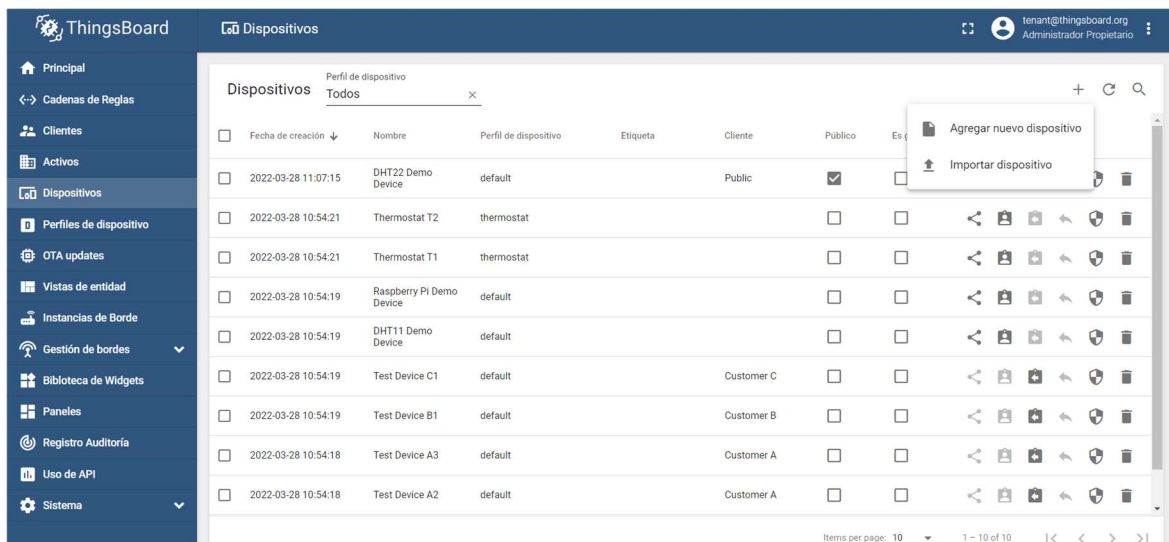


Figura 41. Pantalla Dispositivos en Thingsboard.



Di:

Agregar nuevo dispositivo ? X

1 Detalles del dispositivo 2 Credenciales Optional 3 Cliente Optional

Nombre *
Despliegue IoT

Etiqueta

Perfil de dispositivo *
 Seleccionar un perfil existente default X
 Crear un nuevo perfil de dispositivo

Es gateway

Descripción

Siguiete: Credenciales

Cancelar Agregar

Figura 42. Creación del dispositivo.

Una vez dado el nombre al dispositivo, pasamos a la siguiente pestaña de su configuración llamada *Credenciales* y le damos una credencial del tipo *Access Token*, Figura 43, copiaremos el valor del campo *Access Token* dado, ya que lo necesitaremos a la hora de implementar el programa en *Python*. Con esto ya podemos crear el nuevo dispositivo en la plataforma sin más que hacer clic en *Agregar*. Además, también necesitaremos conocer el valor del campo *ID* del dispositivo, el cual se puede obtener haciendo clic en la opción *Copiar ID*, Figura 44.



Agregar nuevo dispositivo

1 Detalles del dispositivo | 2 **Credenciales** (Optional) | 3 Cliente (Optional)

Añadir credencial

Tipo de credencial
Access token

Tóken de acceso *
DESPL_IOT_1|

Atrás | Sigiente: Cliente | Cancelar | **Agregar**

Figura 43. Creación de las credenciales del dispositivo.

Dispositivos | Perfil de dispositivo: Todos

| Fecha de creación | Nombre | Perfil de dispositivo |
|---------------------|--------------------------|-----------------------|
| 2022-04-20 10:43:49 | Despliegue IoT | default |
| 2022-03-28 11:07:15 | DHT22 Demo Device | default |
| 2022-03-28 10:54:21 | Thermostat T2 | thermostat |
| 2022-03-28 10:54:21 | Thermostat T1 | thermostat |
| 2022-03-28 10:54:19 | Raspberry Pi Demo Device | default |
| 2022-03-28 10:54:19 | DHT11 Demo Device | default |
| 2022-03-28 10:54:19 | Test Device C1 | default |
| 2022-03-28 10:54:19 | Test Device B1 | default |
| 2022-03-28 10:54:18 | Test Device A3 | default |

Despliegue IoT | Detalles del dispositivo

Detalles | Atributos | Última telemetría | Alarmas | Eventos | Relaciones | Registro de actividad

Open details page | Hacer dispositivo público | Asignar a cliente | Gestionar credenciales

Eliminar dispositivo

Copiar ID | Copiar access token

Nombre: Despliegue IoT

Perfil de dispositivo: default

Etiqueta:

Assigned firmware:

Assigned software:

Figura 44. Detalles del dispositivo creado.



Con el servidor de nuestra plataforma corriendo procedemos al conexionado de los sensores a la RPi, el conexionado utilizado se muestra en la Figura 45. El esquema del conexionado se ha realizado utilizando la herramienta *Fritzing* [21].

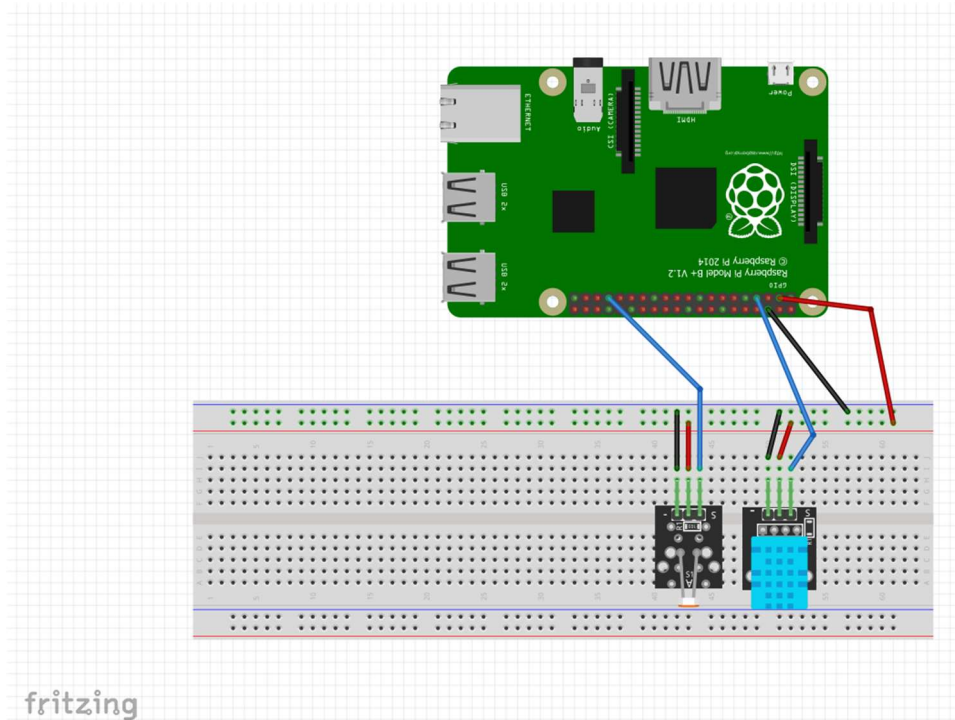


Figura 45. Esquema del conexionado de los sensores a la RPi.

A continuación, se presenta el código implementado en Python para la conexión de estos sensores a la plataforma IoT a través de Ethernet, el cual se ha obtenido de modificar los códigos mostrados en las siguientes referencias [22] [23]. Además, como el sensor DHT11 utilizado no es del fabricante *Adafruit*, tal como se asume en la referencia, hay que hacer una modificación a la forma de obtener los datos del sensor, usando la librería *pigpio_dht*. En este caso será necesario tener descargadas las librerías mostradas en los *import*, para ello se ha utilizado el comando `sudo pip3 install <nombre librería>`.

```
import os
import time
import sys
import paho.mqtt.client as mqtt
import json
from pigpio_dht import DHT11, DHT22
import RPi.GPIO as GPIO
```



```
#Sensor LDR de luminosidad
GPIO.setmode(GPIO.BOARD)

delayt = .1
value = 0 # Variable utilizada para almacenar el valor del ldr
ldr = 33 #ldr en el pin 33

#Función para obtener el valor de luminosidad
def rc_time (ldr):
    count = 0
    GPIO.setup(ldr, GPIO.OUT)
    GPIO.output(ldr, False)
    time.sleep(delayt)
    GPIO.setup(ldr, GPIO.IN)
    #Contar hasta que el pin se pone en valor alto
    while (GPIO.input(ldr) == 0):
        count += 1
    return count

#Sensor DHT11
gpio=4 #Pin BCM
sensor=DHT11(gpio)

#Datos para la conexión a Thingsboard
THINGSBOARD_HOST = '156.35.117.17' #Introducir aquí la IP de acceso a internet de
la RPi la cual cambia al cambiar la red de acceso utilizada.
ACCESS_TOKEN = 'DESPL_IOT_1'

# Intervalo de espera de datos del DHT11
INTERVAL=10

#Datos del sensor a enviar a Thingsboard
sensor_data = {'temperature': 0, 'humidity': 0, 'luminty':0, 'luz': "ON"}
next_reading = time.time()
client = mqtt.Client()
```



```
# Acceso al Token
client.username_pw_set(ACCESS_TOKEN)

# Conexión a Thingsboard usando MQTT
client.connect(THINGSBOARD_HOST, 1883, 60)
client.loop_start()
#Captura de datos de sensores y envío
try:
while True:
#Sensor DHT11
data=sensor.read()
temperature=data["temp_c"]
humidity=data["humidity"]
humidity = round(humidity, 2)
temperature = round(temperature, 2)
print(u"Temperature: {:g}\u00b0C, Humidity: {:g}%".format(temperature, humidity))
#Sensor luminosidad
luminosidad = rc_time(ldr)
print(u"Luminosidad: {:g}".format(luminosidad))
if (luminosidad <= 6000):
sensor_data['luz'] = "ON"
if (luminosidad > 6000):
sensor_data['luz'] = "OFF"

#Datos a enviar a Thingsboard
sensor_data['temperature'] = temperature
sensor_data['humidity'] = humidity
sensor_data['luminty'] =luminosidad
# Envío de datos a Thingsboard
client.publish('v1/devices/me/telemetry', json.dumps(sensor_data), 1)
```



```
next_reading += INTERVAL
sleep_time = next_reading-time.time()
if sleep_time > 0:
time.sleep(sleep_time)
except KeyboardInterrupt:
pass
client.loop_stop()
client.disconnect()
```

Tras el correcto funcionamiento del código implementado, volveremos a la plataforma Thingsboard, donde observaremos desde Dispositivos en la pestaña Última telemetría que los datos de los sensores se obtienen correctamente. Posteriormente tras comprobar que los datos se reciben correctamente, crearemos un panel de control o dashboard para visualizar gráficamente los datos. Para ello accedemos al apartado Paneles, Figura 46, y crearemos un nuevo panel de control, ya sea de cero o utilizando una referencia descargada previamente de algún tutorial. En nuestro caso el panel de control se denomina Despliegue IoT sobre Ethernet: Sensor DHT11 y LDR, Figura 47. Al acceder al mismo, en Abrir Panel, se observa que se han creado dos gráficas separadas para mostrar los valores de temperatura y humedad en el tiempo, así como dos válvulas para mostrar los últimos datos obtenidos para ambos registros, de igual forma para mostrar el nivel lumínico en cada momento se ha utilizado otra válvula y un panel donde se muestra si la luz en el lugar del sensor se encuentra encendida o apagada en ese momento, ON/OFF, tal como se muestra en la Figura 48.

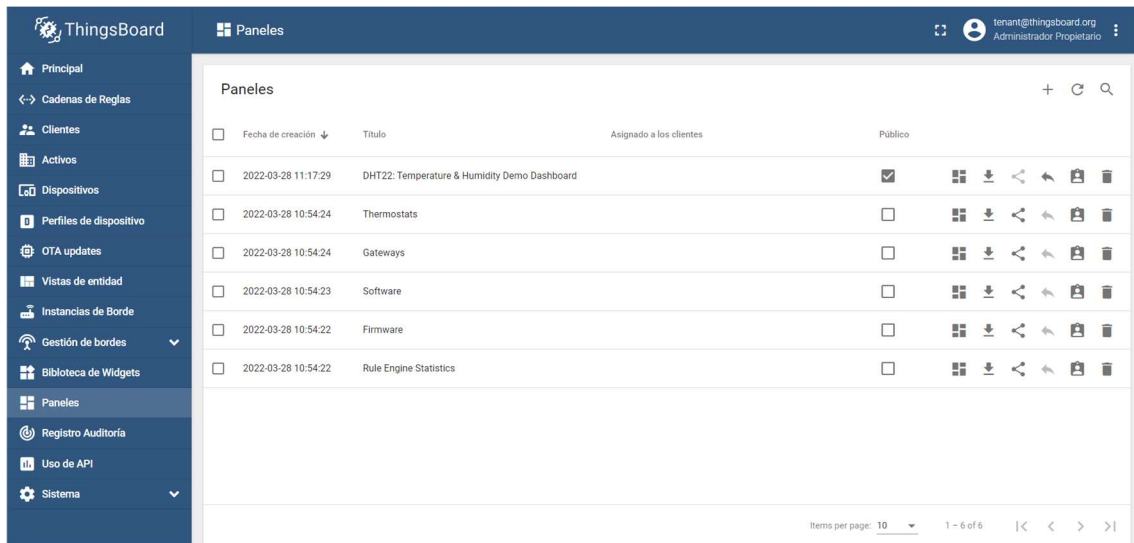


Figura 46. Pantalla Paneles de Thingsboard.

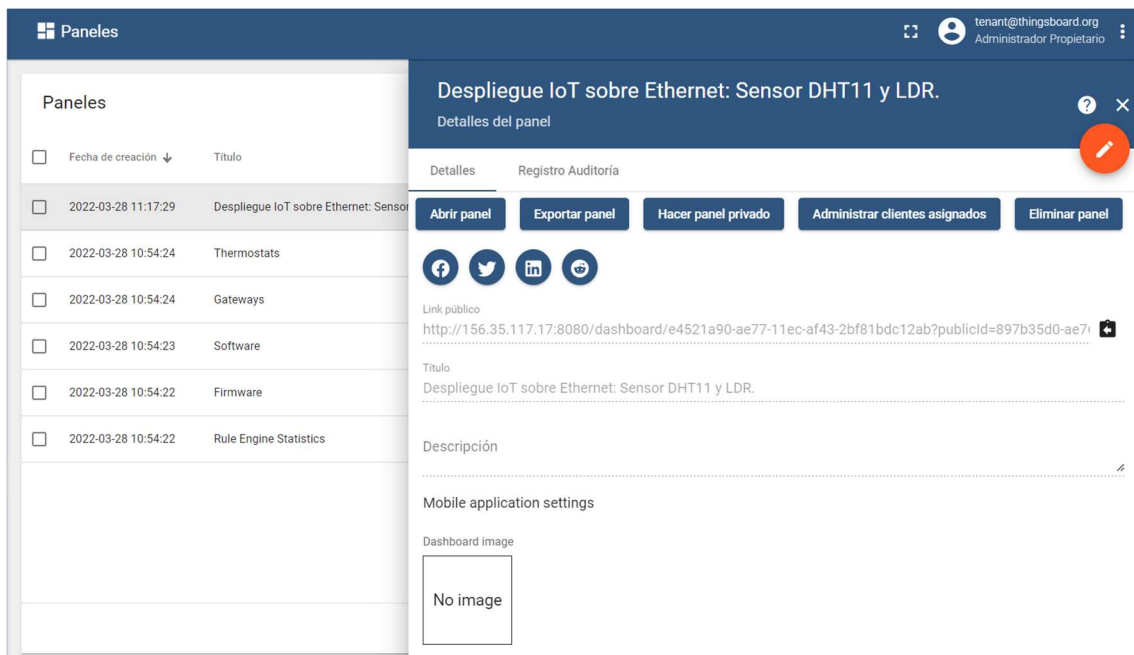


Figura 47. Información del panel creado.

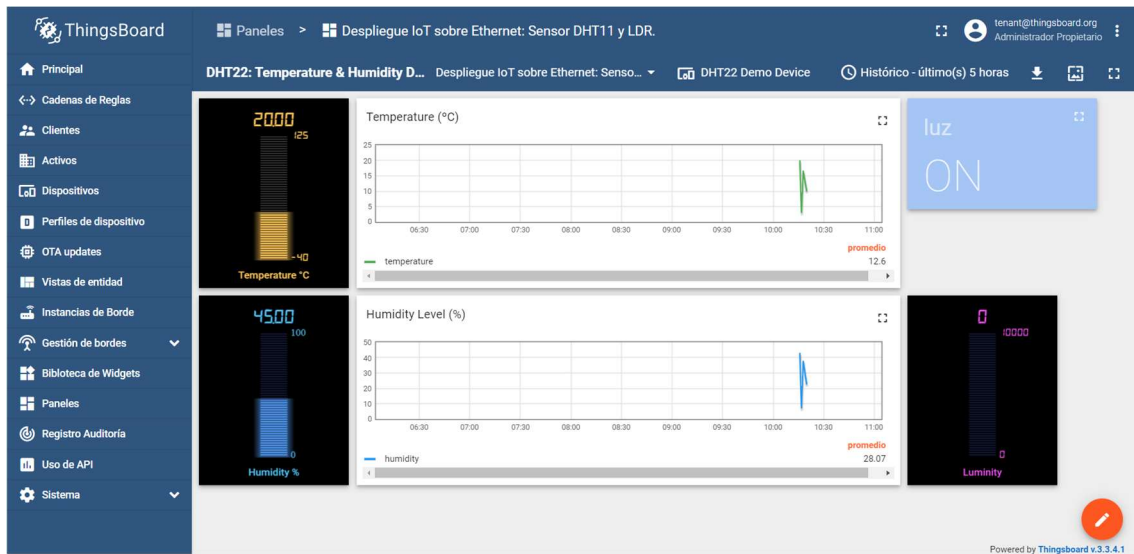


Figura 48. Visualización del panel de control creado.

Con la visualización de los datos obtenidos por los sensores en Thingsboard termina el despliegue sobre Ethernet, el siguiente paso es sustituir esta red por la red 5G.

3.1.2.- Despliegue IoT sobre red 5G.

Como ya se ha mencionado, los sensores a desplegar en este caso son los mismos que los utilizados en el despliegue sobre red Ethernet, de igual forma se va a utilizar la misma plataforma IoT, por lo tanto, la única diferencia es la red utilizada para el envío de los datos de los sensores a la plataforma IoT.

El despliegue de la red 5G ha conllevado más tiempo y más esfuerzo que el despliegue sobre LoRaWAN, esto ha sido debido a que el despliegue de 5G no es tan sencillo como desplegar un dispositivo y conectarnos a él, hay que primero investigar que dispositivos comerciales pueden proveer de red 5G al despliegue IoT a implementar, así como investigar el funcionamiento y puesta en marcha del dispositivo. Así mismo, es necesario disponer de tarjetas SIM 5G configuradas correctamente para poder funcionar dentro de la red existente en el laboratorio, lo cual no ha sido inmediato debido también a que estas tarjetas se deben registrar a su vez en la red.

Tras la investigación de las posibles alternativas a utilizar para el despliegue se ha elegido utilizar los siguientes tres enfoques: utilizando el router móvil NETGEAR NIGHTHAWK M5 5G [24], utilizando un teléfono móvil que permite compartir la red 5G a la RPi utilizando un cable USB y por último se va a utilizar un módulo 5G con su correspondiente adaptador 5G a USB 3.0 y las antenas necesarias para el correcto funcionamiento del módulo.



Como ya se han mencionado, en todos los despliegues se utilizará una tarjeta SIM programada para funcionar en 5G, la tarjeta SIM a utilizar en este caso es del fabricante sysmocom siendo el modelo de tarjeta en cuestión sysmoISIM-SJA2 [25], en la Figura 49 se muestra un ejemplo de cómo son estas tarjetas.



Figura 49. Modelo de tarjeta SIM a utilizar para el despliegue 5G.

Para la configuración de la SIM se han seguidos las instrucciones dadas en el manual de usuario proporcionado por la empresa encargada de la instalación de la red 5G en el laboratorio [26]. A continuación, se presentan detalladamente los pasos seguidos para dicha configuración, esta configuración se va a realizar utilizando una RPI a la cual conectaremos la tarjeta utilizando un lector USB CCID (*Circuit Card Identifier*). Antes de empezar con la configuración de la SIM propiamente dicha es necesario realizar la instalación de varios programas que permiten realizar esta configuración, para ello introducimos los siguientes comandos en el terminal de la RPI:

```
$ cd
$ git clone https://gitea.sysmocom.de/sysmocom/sysmo-
usim-tool.git
```

Para la instalación del siguiente programa necesario hay que acceder a la siguiente página web: <https://sourceforge.net/projects/pyscard/> y descargar la carpeta que contiene el programa. Una vez descargada y descomprimida la carpeta continuamos con la ejecución de los siguientes comandos:

```
$ cd Downloads/pyscard-1.9.5
$ sudo apt update
$ sudo apt install libpcsclite-dev
$ sudo /usr/bin/python setup.py build_ext install
```

Al intentar ejecutar ese último comando aparece un error de ejecución debido a la falta de una librería, que a su vez necesita de otra librería para su correcta descarga y configuración, por tanto, antes de poder seguir con los siguientes comandos dados en el manual de instrucciones hay que descargar e instalar estas dos nuevas librerías. Para ello



el primer paso es que acceder a la siguiente página <https://sourceforge.net/projects/pcre/> y descargar la librería correspondiente, una vez descargada y descomprimida seguiremos con los siguientes comandos:

```
$ cd Downloads/pcre-8.45
$ ./configure
$ sudo make
$ sudo make install
```

La segunda librería que necesitamos descargar es la librería `swig`, lo cual haremos desde la siguiente página <https://sourceforge.net/p/swig/news/2020/06/swig-402-released/>, de igual forma que en los casos anteriores debemos introducir los siguientes comandos tras la descompresión del archivo descargado.

```
$ cd Downloads/ swig-4.0.2
$ sudo make
$ sudo make install
```

Por último, para solucionar el problema de compilación obtenido debemos ejecutar los siguientes comandos:

```
$ cd /pcre-8.42/.libs
$ sudo mv -v libpcre.so.* /usr/lib/
```

Una vez hecho lo anterior volvemos a ejecutar el comando que nos había dado error, `$ sudo /usr/bin/python setup.py build_ext install` y observamos que ahora sí se realiza la ejecución correctamente, y seguimos con la instalación con los siguientes comandos:

```
$ sudo apt-get install python-pyscard pcscd
$ systemctl start pcscd
$ pip install pytlv
```

El último programa que debemos instalar para realizar la configuración de la tarjeta SIM es `pysim`, el cual es un programa Python que permite la lectura y escritura en tarjetas SIM programables. Para instalarlo ejecutaremos lo siguiente:

```
$ cd
$ git clone git://git.osmocom.org/pysim.git
$ cd pysim
```



```
$ sudo apt-get install python3-pyscard python3-serial  
python3-pip python3-yaml
```

```
$ pip3 install -r requirements.txt
```

Tras esto pasamos a la configuración propiamente dicha de la tarjeta, para ello con la tarjeta conectada a la RPi mediante el conector USB mencionado previamente, ejecutamos los siguientes comandos:

```
$ cd
```

```
$ cd sysmo-usim-tool
```

```
$ ./sysmo-isim-tool.sja2.py --adm1 {ADM1} -o -k
```

Donde ADM1 es un número dado por el fabricante y es diferente para cada tarjeta SIM, tras ejecutar este comando observamos por pantalla información relativa a la tarjeta SIM, al añadir al comando los parámetros -o y -k nos mostrará solo la información relativa al OPc y al KI, además de mostrar el valor IMSI, tal como se observa en la Figura 50.

A continuación, se presentan los comandos para completar la configuración de la tarjeta, los cuales son utilizados para la generación de un nuevo valor de OPc diferente del dado por el fabricante. Este valor nuevo de OPc depende del valor -op introducido, el cual es un valor fijo dado por el operador de la red y necesario para poder configurar la tarjeta de acuerdo a la red en la que va a operar:

```
$ cd
```

```
$ cd pysim
```

```
$ ./pySim-prog.py -p 0 -t sysmoISIM-SJA2 -a {ADM1} -x  
222 -y 01 -i
```

```
{IMSI} -s {ICCID} --op=1006020f0a478bf6b699f15c062e42b3  
-k
```

```
{KI}
```

En la Figura 51 se muestra la salida obtenida de ejecutar estos comandos, donde se observa que el OPc ha cambiado respecto al mostrado en la Figura 50.



```

pi@raspberrypi: ~/sysmo-usim-tool
Archivo Editar Pestañas Ayuda
pi@raspberrypi:~$ ./sysmo-isim-tool.sja2.py --adm1 41499998 -o -k
bash: ./sysmo-isim-tool.sja2.py: No existe el fichero o el directorio
pi@raspberrypi:~$ cd sysmo-usim-tool/
pi@raspberrypi:~/sysmo-usim-tool$ ./sysmo-isim-tool.sja2.py --adm1 13585255 -o -k
sysmoISIM-SJA2 parameterization tool
Copyright (c)2019 Sysmocom s.f.m.c. GmbH

Trying to find card with ATR: 3B 9F 96 80 1F 87 80 31 E0 73 FE 21 18 67 4A 4C 75 30 34 05 4B A9
Initializing smartcard terminal...
* Detected card IMSI: 999700000057581
  ISIM Application installed
  USIM Application installed

Authenticating...
* Remaining attempts: 3
* Authenticating...
* Authentication successful
* Remaining attempts: 3

Reading KI value...
* Initializing...
* Reading...
* Current KI setting:
  KI: 18787ca1bc49e38da3ead743a17cabb5

Reading OP/c value...
* Initializing...
* Reading...
* Current OP/OPc setting:
  OPc: e07f19ff5fb71fa607f3a9f3a3c24c4a

Done!
pi@raspberrypi:~/sysmo-usim-tool$ █

```

Figura 50. Salida del terminal tras ejecutar el comando `./sysmo-isim-tool.sja2.py --adm1 {ADM1} -o -k`.

```

pi@raspberrypi: ~/pysim
Archivo Editar Pestañas Ayuda
* Authenticating...
* Authentication successful
* Remaining attempts: 3

Reading KI value...
* Initializing...
* Reading...
* Current KI setting:
  KI: 18787ca1bc49e38da3ead743a17cabb5

Reading OP/c value...
* Initializing...
* Reading...
* Current OP/OPc setting:
  OPc: e07f19ff5fb71fa607f3a9f3a3c24c4a

Done!
pi@raspberrypi:~/sysmo-usim-tool$ cd
pi@raspberrypi:~$ cd pysim/
pi@raspberrypi:~/pysim$ ./pySim-prog.py -p 0 -t sysmoISIM-SJA2 -a 13585255 -x 999 -y 70 -i 999700000057581 -s 8988211000000575814 --op=1006020f0a478bf6b699f15c062e42b3 -k 18787CA1BC49E38DA3EAD743A17CABB5
Using PC/SC reader interface
Ready for Programming: Insert card now (or CTRL-C to cancel)
Generated card parameters :
> Name      : Magic
> SMSP     : e1ffffffffffffffffffff0581005155f5ffffffffffff000000
> ICCID    : 8988211000000575814
> MCC/MNC  : 999/70
> IMSI     : 999700000057581
> KI       : 18787CA1BC49E38DA3EAD743A17CABB5
> OPC      : 3aaF13b065d7e246ccc507ba5be56a5f
> ACC      : None
> ADM1(hex): 3133353835323535
> OPMODE   : None
Programming ...
Warning: Programming of the ICCID is not implemented for this type of card.
Programming successful: Remove card from reader
pi@raspberrypi:~/pysim$ █

```

Figura 51. Salida del terminal tras ejecutar los comandos CAMBIAR CAPTURA.



3.1.2.1.- Despliegue 5G utilizando router NETGEAR.

El primer despliegue de red 5G que se va a presentar es el implementado utilizando el dispositivo NETGEAR Nighthawk M5 5G, el cual es un enrutador móvil que permite crear una red 5G en cualquier sitio, al disponer de una batería recargable para su funcionamiento, insertando una tarjeta SIM que permita la conexión a la red 5G. En la Figura 52 se muestra el dispositivo NETGEAR a utilizar. De lo que se encarga este enrutador es de conectarse a la red 5G utilizando la tarjeta SIM programada previamente para utilizar la red 5G y crear una red WiFi a partir de dicha red, a la cual nos conectaremos con los dispositivos deseados para acceder a la misma, también es posible conectarnos directamente al enrutador por medio de un cable ethernet y conectarnos a la red 5G de esta forma. [24]



Figura 52. Dispositivo Netgear Nighthawk M5 Mobile Router.

La configuración del propio dispositivo NETGEAR se realiza utilizando la pantalla táctil del mismo. A continuación, se presentan todos los pasos necesarios para poner en marcha el dispositivo desde cero, tras haber introducido la tarjeta SIM en el mismo.

Nada más encender el dispositivo nos pedirá seleccionar el idioma en el que queremos que funcione el dispositivo, Figura 53, de la lista dada seleccionamos Español (Spanish), tras seleccionarlo continuará el arranque del dispositivo, Figura 54. Una vez el dispositivo ha acabado su arranque nos aparece la pantalla de la Figura 55, donde pulsaremos en Introducción y pasaremos directamente a la pantalla de la Figura 56. En esta nueva pantalla se nos pide crear una contraseña de administración para el dispositivo, la cual se utilizará en el caso de querer administrar el dispositivo de forma remota desde otro dispositivo conectado a la red creada por el propio dispositivo, para lo



cual hay que acceder a la url <http://mywebui.net> o <http://192.168.1.1>, estas urls se obtiene de la pantalla de información del dispositivo del propio dispositivo o de su manual de instrucciones. En este caso se ha decidido no crear una nueva contraseña de configuración por lo que en la pantalla de la Figura 56 se selecciona la opción de Omitir y pasaríamos a la pantalla de la Figura 57 que nos permite configurar la WiFi creada según preferencias, aquí se ha optado por la opción de No personalizar. En la siguiente pantalla, Figura 58, nos aparecen las credenciales de la red WiFi que se va a crear, los cuales también aparecen en todo momento en la pantalla principal del dispositivo. El último paso es la creación del APN o nombre del punto de acceso de la tarjeta SIM, para lo cual en la pantalla de la Figura 59, a la cual accedemos desde Configuración>Router>APN, hacemos clic en +Añadir. Al hacer esto pasamos a la pantalla de la Figura 60 donde configuramos los datos necesarios, que en este caso son el nombre, para el cual hemos elegido firecell que es el nombre genérico de la red 5G del laboratorio, y el APN propiamente dicho cuyo valor es oai.ipv4, no es necesario modificar el resto de campos. Con esto quedaría configurado el dispositivo NETGEAR y debería crear la red 5G, lo cual se comprueba que es así al ver en la pantalla inicial el icono 5G que lo indica en la parte superior izquierda, tal como se observa en la Figura 61.

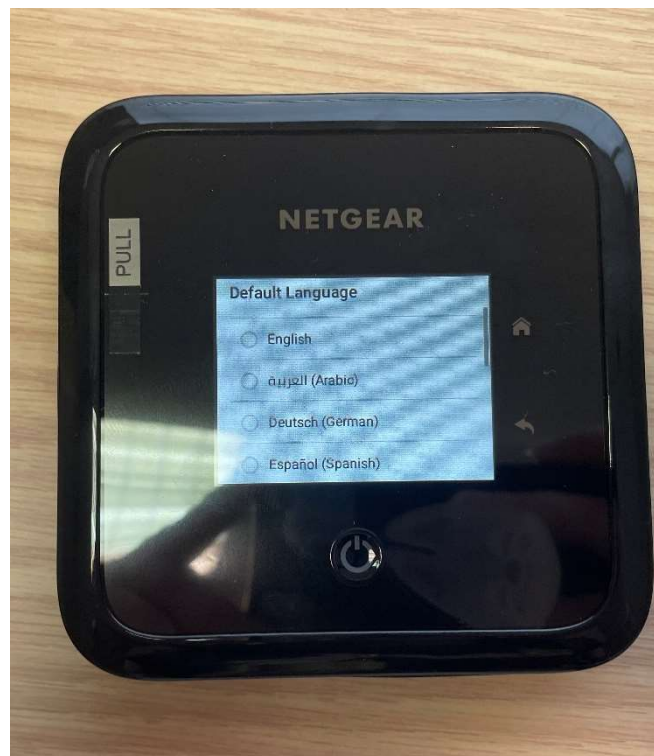


Figura 53. Pantalla de arranque del dispositivo Netgear donde seleccionar idioma.



Figura 54. Pantalla de encendido del dispositivo Netgear.



Figura 55. Proceso de configuración inicial del dispositivo Netgear (1).



Figura 56. Proceso de configuración inicial del dispositivo Netgear (2).



Figura 57. Proceso de configuración inicial del dispositivo Netgear (3).



Figura 58. Proceso de configuración inicial del dispositivo Netgear (4)

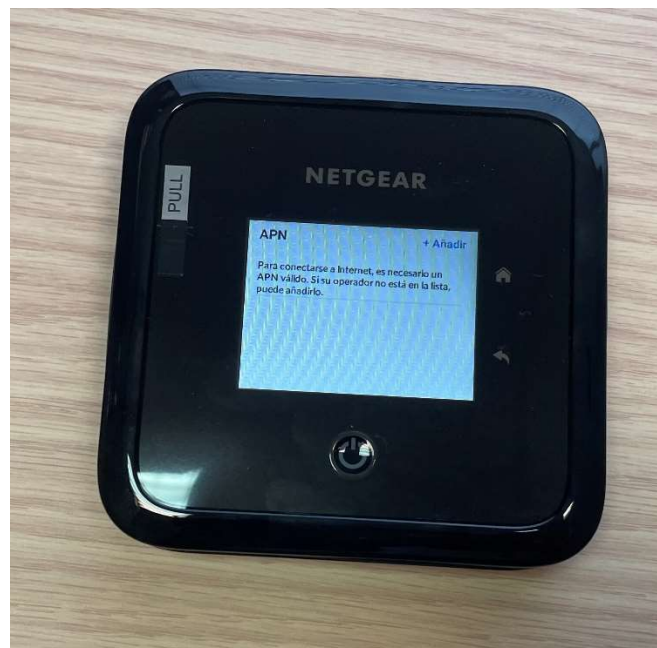


Figura 59. Configuración del APN en el dispositivo Netgear (1).



Figura 60. Configuración del APN en el dispositivo Netgear (2).



Figura 61. Dispositivo Netgear conectado a la red 5G y en funcionamiento.

Tras tener el router Netgear funcionando conectaremos la RPi a éste, se puede conectar tanto accediendo a la red WiFi creada por el dispositivo o usando un cable ethernet, en este caso se eligió utilizar el cable ethernet para conectar la RPi a la red. El último paso para comprobar el correcto despliegue IoT es ejecutar el mismo código del



despliegue sobre red Ethernet y comprobar que con la nueva dirección de red los datos lleguen a la plataforma Thingsboard.

3.1.2.2.- Despliegue de la red 5G utilizando un smartphone.

El siguiente despliegue que se va a describir es el basado en utilizar un smartphone para crear una red 5G a la cual poder conectar la RPi a la cual tenemos conectada el sistema IoT.

El dispositivo smartphone que se va a utilizar para el despliegue de la red 5G es el modelo Nord 2 del fabricante OnePlus [27] y para crear la red 5G a la que conectarnos se utiliza *tethering* USB o anclaje de red con cable USB. El tethering es una tecnología que permite que un dispositivo móvil con acceso a internet, generalmente mediante red LTE o 5G como es nuestro caso, se convierta en punto de acceso. A este punto de acceso es posible conectarse mediante Bluetooth, WiFi o utilizando un cable USB, como será nuestro caso. [28]

Para implementar este despliegue en el smartphone hay que introducir una de las tarjetas SIM configuradas previamente como SIM 5G, y configurar su APN de forma similar que en el caso del dispositivo router anterior. Para configurar este parámetro hay que acceder a la configuración del smartphone y dentro de esta a Tarjeta SIM y datos móviles > SIM1 > Nombre del punto de acceso, pantalla que se muestra en la Figura 62. Dentro de esta pantalla aparecerán inicialmente varios APN por defecto con valores diferentes al nuestro, por tanto, debemos crear uno nuevo, esto se hace haciendo clic en el botón +, esta pantalla de creación del APN se muestra en la Figura 63. Los datos de APN a introducir serán los mismos que en el caso del dispositivo router, la configuración de los datos se observa en la Figura 64. Tras configurar el APN deberíamos obtener señal de la red 5G en el dispositivo, tras como se muestra en la Figura 65 en la parte superior izquierda de la pantalla, con el icono mostrándonos que está conectado a la red 5G y la cobertura que tiene.

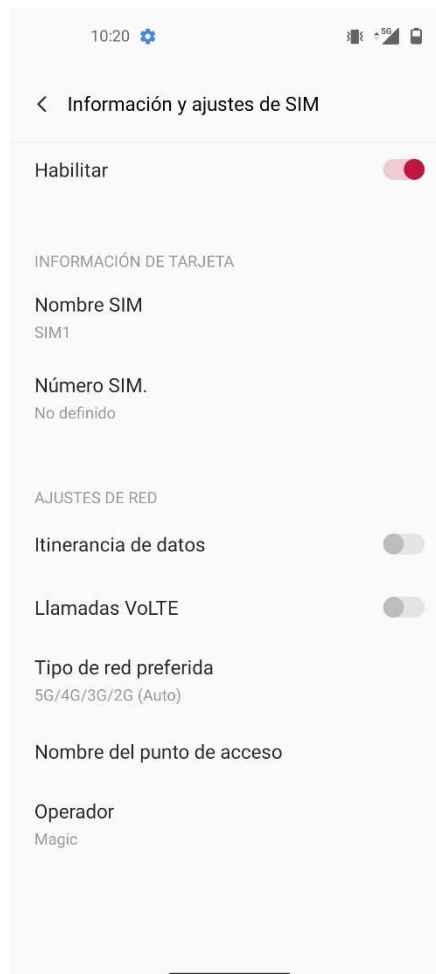


Figura 62. Configuración del APN en smartphone (1).

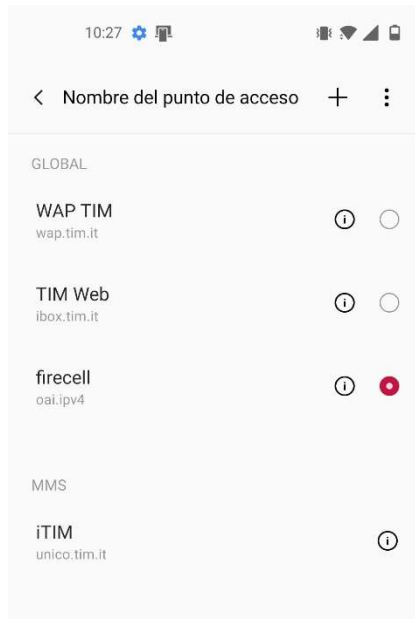


Figura 63. Configuración del APN en smartphone (2).

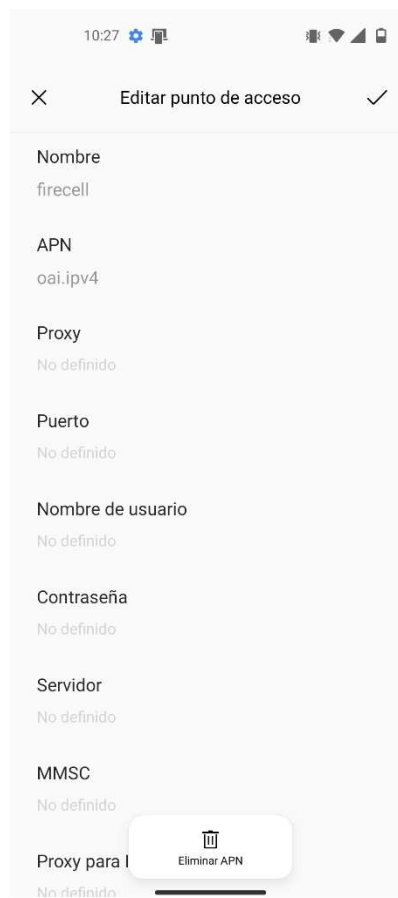


Figura 64. Configuración del APN en smartphone (3).

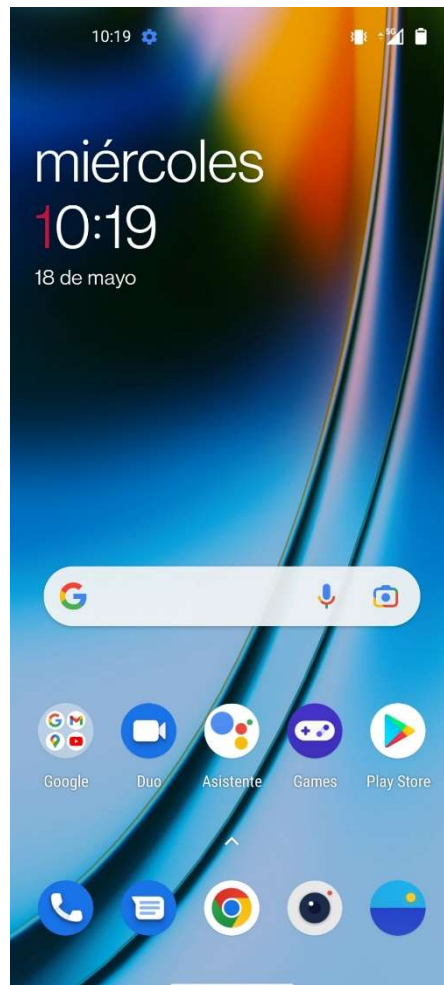


Figura 65. Smartphone funcionando con la tarjeta SIM 5G.

Con la señal 5G activa pasamos a implementar los pasos para poner en funcionamiento el tethering, para lo cual el primer paso es conectar mediante un cable USB al dispositivo al cual queremos dar conectividad 5G, que en este caso es la RPi, el cable USB utilizado es el incluido en la caja del dispositivo, Una vez conectado el smartphone a la RPi mediante el cable USB procedemos a activar el tethering, para ello accedemos al menú Configuración>Conexión y compartir. y pulsamos en la opción Compartir datos por USB, tal como se muestra en la Figura 66. Tras esto podemos observar en la RPi que se ha conectado a una nueva red a la que denomina `usb0`, y al ejecutar en la terminal el comando `$ifconfig` aparece lo mostrado en la Figura 67, que es la información relativa a la nueva red creada con sus direcciones correspondientes.

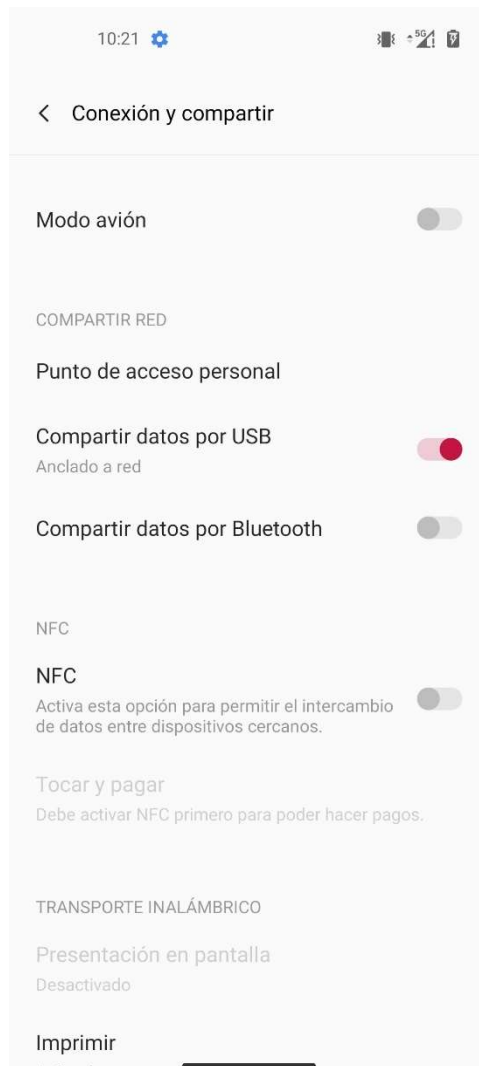


Figura 66. Configuración de tethering mediante USB (1).

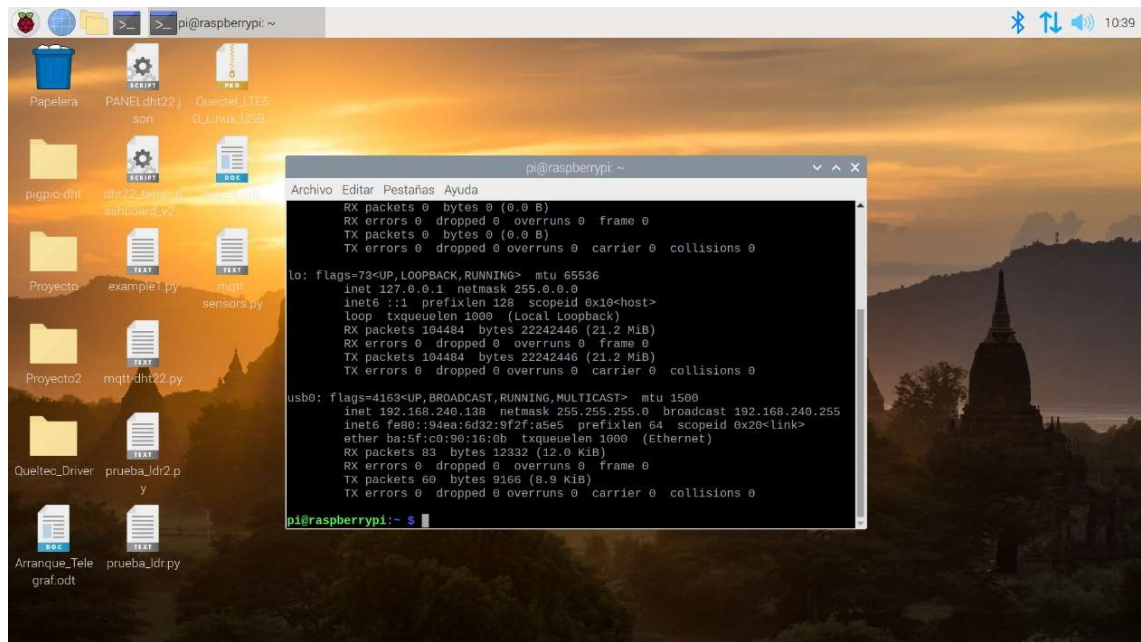


Figura 67. Comprobación en la RPi de la correcta conexión a 5G mediante smartphone.

En principio, con esto acabaría la configuración del tethering y seríamos capaces de conectarnos a internet desde la RPi usando la red 5G, en el caso de que no fuera posible acceder a internet a pesar de aparecer la red creada al ejecutar `$ifconfig` sería necesario modificar otro par de parámetros en el smartphone, para esto es necesario que el dispositivo móvil se encuentre en modo desarrollador. Para entrar en este modo desde este modelo en particular hay que acceder al menú Configuración>Información del teléfono > Versión, y hacer clic varias veces seguidas sobre la opción Número de versión, en la Figura 68 se muestra dicha pantalla. Las modificaciones necesarias se realizan desde el menú Configuración>Configuración adicional>Opciones de desarrollador, la pantalla que se verá en el dispositivo es la mostrada en la Figura 69. En esta pantalla debemos modificar lo siguiente, en el apartador de DEPURAR debemos activar la opción Depuración por USB, Figura 70, además dentro del mismo apartado, pero desplazando la pantalla hacia abajo debemos activar la opción de Verificar aplicaciones por USB si no lo estuviera, Figura 71. Por último, dentro del apartado RED de esta pantalla, debemos comprobar que dentro de la opción Configuración de USB predeterminada, Figura 72, se encuentra seleccionada la opción Compartir conexión por USB tal como se muestra en la Figura 73.

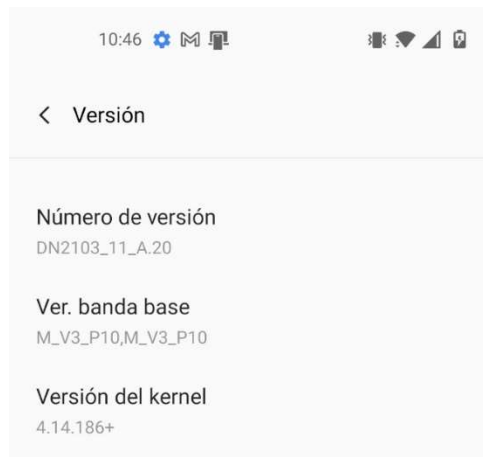


Figura 68. Pantalla para entrar en modo desarrollador en el smartphone OnePlus Nord 2 5G.

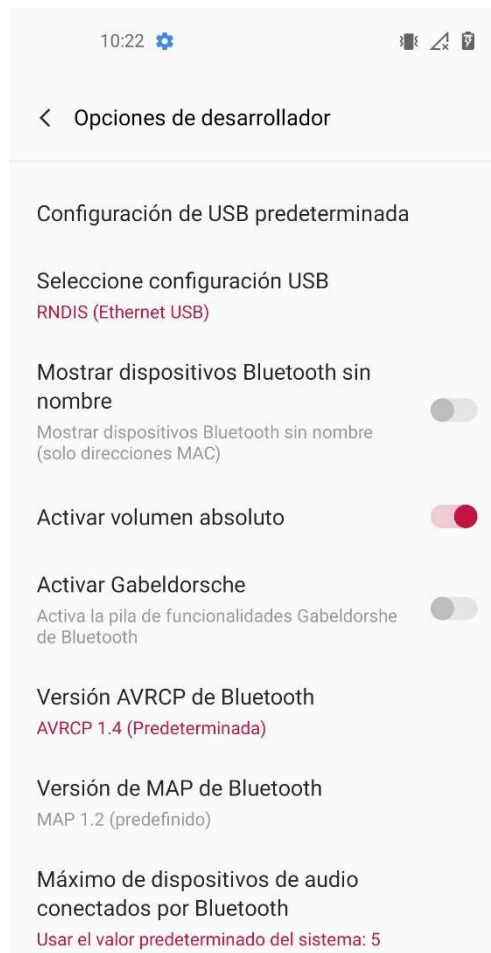


Figura 69. Pantalla de Opciones de desarrollador.

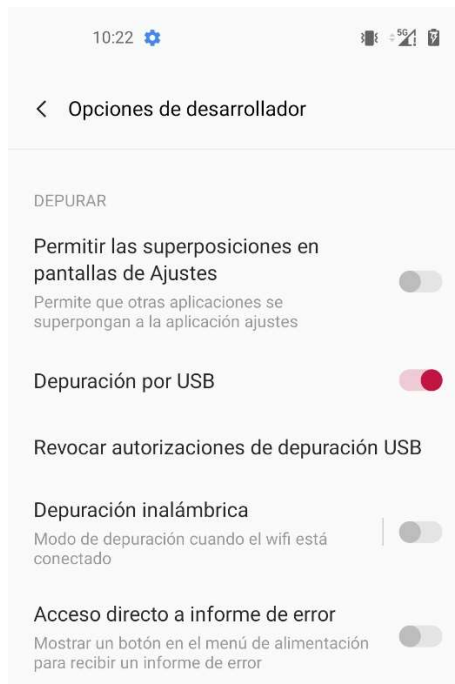


Figura 70. Modificaciones en Opciones de desarrollador para el funcionamiento del tethering por USB (1).

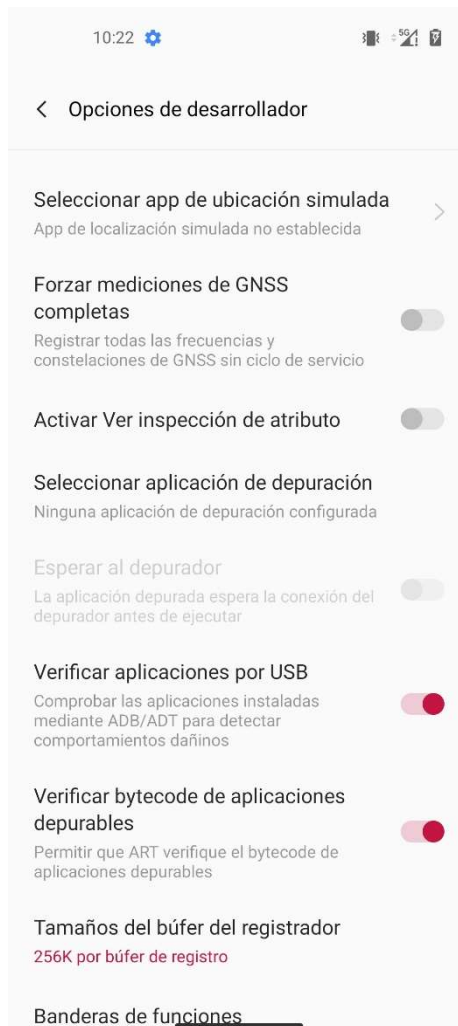


Figura 71. Modificaciones en Opciones de desarrollador para el funcionamiento del tethering por USB (2).

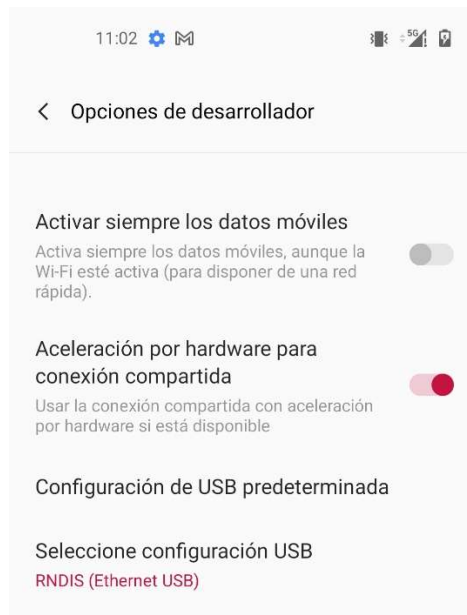


Figura 72. Modificaciones en Opciones de desarrollador para el funcionamiento del tethering por USB (3).

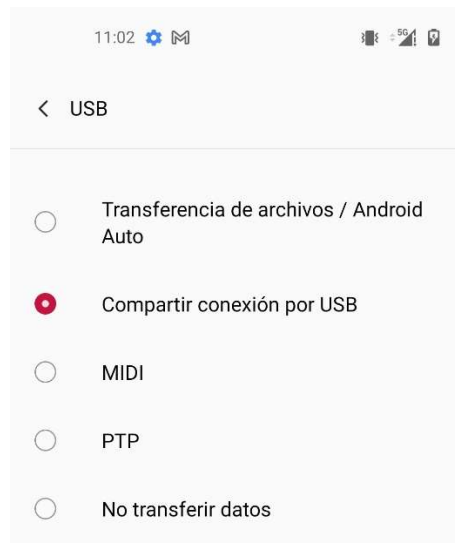


Figura 73. Modificaciones en Opciones de desarrollador para el funcionamiento del tethering por USB (4).

Tras estas modificaciones al seleccionar la opción de Compartir mediante USB ahora ya si la RPi está conectada a la red 5G a la que está conectada el smartphone y con conexión a internet.

Con este despliegue de la red 5G se puede observar de forma totalmente gráfica que en este caso no es necesario el uso de ninguna red adicional, ya que con el smartphone conectado a la RPi proveemos de red 5G con acceso a internet a la RPi y a su vez podemos ver los datos subidos a nuestro panel de datos implementado en Thingsboard en el propio smartphone. En la Figura 74 se muestra una captura de cómo se visualiza el panel de



control de Thingsboard desde el smartphone estando conectado este a la red 5G, tal como se ve en la parte superior derecha de la pantalla, y además se observa que en ese momento el despliegue IoT está funcionando correctamente al verse los datos en las gráficas correspondientes.

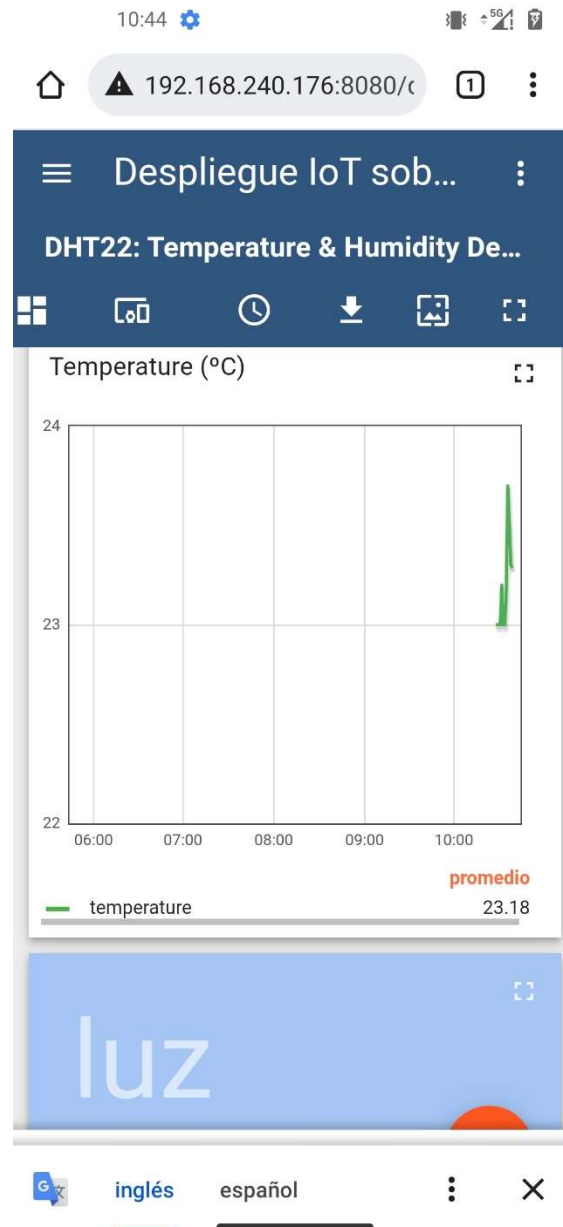


Figura 74. Acceso al panel de datos de Thingsboard desde el smartphone suministrador de red 5G.



3.1.2.3.- Despliegue de la red 5G utilizando un adaptador de módulo 5G a USB.

La última alternativa utilizada para el despliegue de la red 5G a la que conectaremos la RPi es utilizando un adaptador de módulo 5G a USB 3.0, con un módulo 5G del fabricante Quectel [29]. En la Figura 75 se muestran todas las partes necesarias a montar para el funcionamiento del módulo 5G con su adaptador a USB 3.0, y a continuación en la Figura 76 se muestra el módulo montado con su adaptador, en este módulo también introducimos una tarjeta SIM 5G.



Figura 75. Elementos necesarios para el montaje del adaptador de módulo 5G a USB 3.0.



Figura 76. Adaptador 5G a USB con módulo de Quectec montado.

Al conectar este módulo a la RPi se observa en la misma que la propia RPi reconoce este módulo sin necesidad de instalar drivers y le asigna una dirección IP bajo el tipo de `wwan0`, tal como se muestra en la Figura 77 tras hacer ejecutar el comando `$ifconfig`. Pero a pesar de asignarle una dirección IP y parecer aparentemente que ya estaría todo configurado para poder acceder a internet utilizando esta nueva red 5G creada, al intentar realizar un ping a la dirección de www.google.com se observa que no llega al destino, lo cual se muestra en la Figura 78, lo que indica que aún no tenemos conexión a internet utilizando el módulo.



```

pi@raspberrypi:~$ ifconfig
docker0: flags=4099<UP, BROADCAST, MULTICAST> mtu 1500
inet 172.17.0.1 netmask 255.255.0.0 broadcast 172.17.255.255
ether 02:42:09:3b:7d:54 txqueuelen 0 (Ethernet)
RX packets 0 bytes 0 (0.0 B)
RX errors 0 dropped 0 overruns 0 frame 0
TX packets 0 bytes 0 (0.0 B)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

eth0: flags=4099<UP, BROADCAST, MULTICAST> mtu 1500
ether e4:5f:01:31:29:da txqueuelen 1000 (Ethernet)
RX packets 0 bytes 0 (0.0 B)
RX errors 0 dropped 0 overruns 0 frame 0
TX packets 0 bytes 0 (0.0 B)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP, LOOPBACK, RUNNING> mtu 65536
inet 127.0.0.1 netmask 255.0.0.0
inet6 ::1 prefixlen 128 scopeid 0x10<host>
loop txqueuelen 1000 (Local Loopback)
RX packets 110741 bytes 23617688 (22.5 MiB)
RX errors 0 dropped 0 overruns 0 frame 0
TX packets 110741 bytes 23617688 (22.5 MiB)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

wwan0: flags=4163<UP, BROADCAST, RUNNING, MULTICAST> mtu 1500
inet 169.254.238.146 netmask 255.255.0.0 broadcast 169.254.255.255
inet6 fe80::a6b1:e6e5:b2dc:4cd8 prefixlen 64 scopeid 0x20<link>
ether da:6f:f4:c0:dc:f5 txqueuelen 1000 (Ethernet)
RX packets 0 bytes 0 (0.0 B)
RX errors 0 dropped 0 overruns 0 frame 0
TX packets 0 bytes 0 (0.0 B)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

```

Figura 77. Salida de la ejecución del comando `ifconfig` con el módulo conectado inicialmente.

```

lo: flags=73<UP, LOOPBACK, RUNNING> mtu 65536
inet 127.0.0.1 netmask 255.0.0.0
inet6 ::1 prefixlen 128 scopeid 0x10<host>
loop txqueuelen 1000 (Local Loopback)
RX packets 110741 bytes 23617688 (22.5 MiB)
RX errors 0 dropped 0 overruns 0 frame 0
TX packets 110741 bytes 23617688 (22.5 MiB)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

wwan0: flags=4163<UP, BROADCAST, RUNNING, MULTICAST> mtu 1500
inet 169.254.238.146 netmask 255.255.0.0 broadcast 169.254.255.255
inet6 fe80::a6b1:e6e5:b2dc:4cd8 prefixlen 64 scopeid 0x20<link>
ether da:6f:f4:c0:dc:f5 txqueuelen 1000 (Ethernet)
RX packets 0 bytes 0 (0.0 B)
RX errors 0 dropped 0 overruns 0 frame 0
TX packets 0 bytes 0 (0.0 B)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

pi@raspberrypi:~$ ping -I wwan0 www.google.com -c 5
ping: www.google.com: Fallo temporal en la resolución del nombre
pi@raspberrypi:~$

```

Figura 78. Comprobación de conexión a internet de la nueva red 5G.

Para poder conectarnos realmente a la red 5G de la tarjeta SIM y acceder a internet debemos de seguir los pasos descritos a continuación, los cuales se han obtenido de la siguiente página web [30].



El primer paso es comprobar que la RPi detecta correctamente el módulo Quectec, para lo cual se ejecuta el comando `$lsusb`, y se observa que si lo reconoce tal como se muestra en la Figura 79, lo siguiente a comprobar sería asegurarnos de que el dispositivo aparece en la interfaz `wwan0` y que tiene una dirección IP asignada, como ya se hizo previamente y se mostró en la Figura 77.

```

pi@raspberrypi: ~
Archivo Editar Pestañas Ayuda
TX packets 110741 bytes 23617688 (22.5 MiB)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

wwan0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
inet 169.254.238.146 netmask 255.255.0.0 broadcast 169.254.255.255
inet6 fe80::a6b1:e6e5:b2dc:4cd8 prefixlen 64 scopeid 0x20<link>
ether da:6f:f4:c0:dc:f5 txqueuelen 1000 (Ethernet)
RX packets 0 bytes 0 (0.0 B)
RX errors 0 dropped 0 overruns 0 frame 0
TX packets 0 bytes 0 (0.0 B)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

pi@raspberrypi:~$ ping -I wwan0 www.google.com -c 5
ping: www.google.com: Fallo temporal en la resolución del nombre
pi@raspberrypi:~$ lsusb
Bus 002 Device 001: ID 1d6b:0003 Linux Foundation 3.0 root hub
Bus 001 Device 016: ID 046d:c077 Logitech, Inc. M105 Optical Mouse
Bus 001 Device 024: ID 2c7c:0800 Quectel Wireless Solutions Co., Ltd.
Bus 001 Device 003: ID 046d:c34b Logitech, Inc.
Bus 001 Device 002: ID 2109:3431 VIA Labs, Inc. Hub
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
pi@raspberrypi:~$

```

Figura 79. Ejecución del comando `$lsusb`.

Tras realizar estas comprobaciones procederemos a realizar la configuración en sí misma, para esto deberemos instalar un par de librerías usando el siguiente comando:

```
$ sudo apt update && sudo apt install libqmi-utils udhcpc
```

Una vez instaladas correctamente las librerías, comprobaremos el modo en el que está operando el módulo con el siguiente comando:

```
$ sudo qmicli -d /dev/cdc-wdm0 --dms-get-operating-mode
```

En la Figura 80 se muestra la salida obtenida a este comando, en el caso en el que el modo mostrado no fuera `online` habría que modificarlo a modo `online` ejecutando el siguiente comando:

```
$ sudo qmicli -d /dev/cdc-wdm0 --dms-set-operating-mode='online'
```



```
pi@raspberrypi:~ $ sudo qmicli -d /dev/cdc-wdm0 --dms-get-operating-mode
[/dev/cdc-wdm0] Operating mode retrieved:
  Mode: 'online'
  HW restricted: 'no'
pi@raspberrypi:~ $
```

Figura 80. Configuración de módulo Quectec 5G (1).

El siguiente paso a realizar es establecer la interfaz en modo `raw_ip`, para lo cual ejecutamos los siguientes comandos:

```
$ sudo ip link set wwan0 down
$ echo 'Y' | sudo tee /sys/class/net/wwan0/qmi/raw_ip
$ sudo ip link set wwan0 up
```

Para comprobar que este modo se ha establecido correctamente ejecutamos el comando `$ sudo qmicli -d /dev/cdc-wdm0 --wda-get-data-format`, el cual nos devuelve lo mostrado en la Figura 81, donde se observa que si se ha establecido correctamente al aparecer `raw-ip` como `Link layer protocol`.



```

pi@raspberrypi: ~
Archivo Editar Pestañas Ayuda
pi@raspberrypi:~$ sudo qmicli -d /dev/cdc-wdm0 --dms-get-operating-mode
[/dev/cdc-wdm0] operating mode retrieved:
  Mode: 'online'
  HW restricted: 'no'
pi@raspberrypi:~$ sudo ip link set wwan0 down
pi@raspberrypi:~$ echo 'Y' | sudo tee /sys/class/net/wwan0/qmi/raw_ip
Y
pi@raspberrypi:~$ sudo ip link set wwan0 up
pi@raspberrypi:~$ sudo qmicli -d /dev/cdc-wdm0 --wda-get-data-format
[/dev/cdc-wdm0] Successfully got data format
  QoS flow header: no
  Link layer protocol: 'raw-ip'
  Uplink data aggregation protocol: 'disabled'
  Downlink data aggregation protocol: 'disabled'
  NDP signature: '0'
  Uplink data aggregation max size: '0'
  Downlink data aggregation max size: '0'
pi@raspberrypi:~$

```

Figura 81. Configuración de módulo Quectel 5G (2).

Ahora es el momento de configurar el APN de la tarjeta SIM, para ello necesitamos ejecutar el siguiente comando, donde introducimos el nombre de nuestro APN:

```

$ sudo qmicli -p -d /dev/cdc-wdm0 --device-open-
net='net-raw-ip|net-no-qos-header' --wds-start-
network="apn='oai.ipv4',ip-type=4" --client-no-release-cid

```

La salida del comando nos indicara si la red se ha iniciado correctamente, como es el caso tal como se muestra en la Figura 82, con el mensaje Network started. El último paso de esta configuración es utilizar la librería udhcpc para asignar una dirección IP y ruta por defecto, para ello el comando a utilizar es `$ sudo udhcpc -q -f -i wwan0`, Figura 83.



```

pi@raspberrypi: ~
Archivo Editar Pestañas Ayuda
Mode: 'online'
HW restricted: 'no'
pi@raspberrypi:~ $ sudo ip link set wwan0 down
pi@raspberrypi:~ $ echo 'Y' | sudo tee /sys/class/net/wwan0/qmi/raw_ip
Y
pi@raspberrypi:~ $ sudo ip link set wwan0 up
pi@raspberrypi:~ $ sudo qmicli -d /dev/cdc-wdm0 --wda-get-data-format
[/dev/cdc-wdm0] Successfully got data format
QoS flow header: no
Link layer protocol: 'raw-ip'
Uplink data aggregation protocol: 'disabled'
Downlink data aggregation protocol: 'disabled'
NDP signature: '0'
Uplink data aggregation max size: '0'
Downlink data aggregation max size: '0'
pi@raspberrypi:~ $ sudo qmicli -p -d /dev/cdc-wdm0 --device-open-net='net-raw-ip
|net-no-qos-header' --wds-start-network="apn='oai.ipv4', ip-type=4" --client-no-
release-cid
[/dev/cdc-wdm0] Network started
Packet data handle: '4003082416'
[/dev/cdc-wdm0] Client ID not released:
Service: 'wds'
CID: '15'
pi@raspberrypi:~ $

```

Figura 82. Configuración de módulo Quectec 5G (3).

```

pi@raspberrypi: ~
Archivo Editar Pestañas Ayuda
pi@raspberrypi:~ $ echo 'Y' | sudo tee /sys/class/net/wwan0/qmi/raw_ip
Y
pi@raspberrypi:~ $ sudo ip link set wwan0 up
pi@raspberrypi:~ $ sudo qmicli -p -d /dev/cdc-wdm0 --device-open-net='net-raw-ip
|net-no-qos-header' --wds-start-network="apn='oai.ipv4', ip-type=4" --client-no-
release-cid
error: couldn't open the QmiDevice: Transaction timed out
pi@raspberrypi:~ $ sudo qmicli -p -d /dev/cdc-wdm0 --device-open-net='net-raw-ip
|net-no-qos-header' --wds-start-network="apn='oai.ipv4', ip-type=4" --client-no-
release-cid
[/dev/cdc-wdm0] Network started
Packet data handle: '3818752784'
[/dev/cdc-wdm0] Client ID not released:
Service: 'wds'
CID: '15'
pi@raspberrypi:~ $ sudo udhcpc -q -f -i wwan0
udhcpc: started, v1.30.1
No resolv.conf for interface wwan0.udhcpc
udhcpc: sending discover
udhcpc: sending select for 12.1.1.3
udhcpc: lease of 12.1.1.3 obtained, lease time 7200
Too few arguments.
Too few arguments.
pi@raspberrypi:~ $

```

Figura 83. Configuración de módulo Quectec 5G (4).

Si todo ha salido bien, ya deberíamos poder acceder a internet desde nuestra RPi utilizando el módulo Quectec, para comprobar esto se vuelve a enviar un ping a www.google.com y el resultado es el mostrado en la Figura 84. Se observa que el ping es recibido correctamente y ya tenemos la red 5G funcionando correctamente.



```

pi@raspberrypi: ~
Archivo Editar Pestañas Ayuda

wwan0: flags=4305<UP,POINTOPOINT,RUNNING,NOARP,MULTICAST> mtu 1500
inet 12.1.1.4 netmask 255.255.255.248 destination 12.1.1.4
unspec 00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00 txqueuelen 1000
(UNSPEC)
RX packets 30 bytes 9096 (8.8 KiB)
RX errors 0 dropped 0 overruns 0 frame 0
TX packets 48 bytes 5048 (4.9 KiB)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

pi@raspberrypi:~ $ ping -I wwan0 www.google.com -c 5
PING www.google.com (142.250.200.68) from 12.1.1.4 wwan0: 56(84) bytes of data.
64 bytes from mad07s24-in-f4.1e100.net (142.250.200.68): icmp_seq=1 ttl=114 time
=25.1 ms
64 bytes from mad07s24-in-f4.1e100.net (142.250.200.68): icmp_seq=2 ttl=114 time
=41.6 ms
64 bytes from mad07s24-in-f4.1e100.net (142.250.200.68): icmp_seq=3 ttl=114 time
=39.9 ms
64 bytes from mad07s24-in-f4.1e100.net (142.250.200.68): icmp_seq=4 ttl=114 time
=38.7 ms
64 bytes from mad07s24-in-f4.1e100.net (142.250.200.68): icmp_seq=5 ttl=114 time
=27.8 ms

--- www.google.com ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 10ms
rtt min/avg/max/mdev = 25.108/34.620/41.623/6.792 ms
pi@raspberrypi:~ $

```

Figura 84. Configuración de modulo Quectec 5G (5).

La principal desventaja de este dispositivo es que al reiniciar la RPi o al desconectar y volver a conectar el dispositivo de la misma hay que volver a ejecutar todos los comandos de configuración para su conexión. En el caso del reinicio la solución es crear un archivo para la interfaz wwan0 que se ejecute al iniciar la RPi con todos los comandos necesarios, el archivo se debe crear en la ruta /etc/network/interfaces.d/wwan0 y debe de contener lo siguiente:

```

auto wwan0

iface wwan0 inet manual

pre-up ifconfig wwan0 down

pre-up echo Y > /sys/class/net/wwan0/qmi/raw_ip

pre-up for _ in $(seq 1 10); do /usr/bin/test -c
/dev/cdc-wdm0 && break; /bin/sleep 1; done

pre-up for _ in $(seq 1 10); do /usr/bin/qmicli -d
/dev/cdc-wdm0 --nas-get-signal-strength && break; /bin/sleep
1; done

pre-up sudo qmicli -p -d /dev/cdc-wdm0 --device-
open-net='net-raw-ip|net-no-qos-header' --wds-start-
network="apn='oai.ipv4',ip-type=4" --client-no-release-cid

```



```
pre-up udhcpc -i wwan0
post-down /usr/bin/qmi-network /dev/cdc-wdm0 stop
```

Con este fichero tendríamos solucionada la configuración automática después del apagado de la RPi para el caso de desconectar el módulo mientras la RPi este encendida y volver a conectarlo a ella aún encendida la solución sería reiniciarla o volver a ejecutar los comandos de configuración anteriores.

Habiendo realizado todos los pasos anteriores ya podríamos ejecutar nuestro código de la implementación IoT sobre la red 5G desplegada con el módulo Quectel de igual forma que se hizo en los dos anteriores despliegues y comprobar que los datos de los sensores llegan correctamente a la plataforma Thingsboard.

3.1.3.- Ampliación de la implementación IoT sobre 5G.

En la implementación inicial IoT probada tanto sobre Ethernet como sobre 5G se utilizaban únicamente dos sensores, por lo que para poder mostrar en muy pequeña escala que la red 5G puede soportar multitud de sensores se han decidido incorporar nuevos sensores a esta implementación. Estos nuevos sensores son los siguientes, un sensor avoid de detección de obstáculos [31], un sensor que detecta llamas o fuego [32], un sensor de shock [33] que detecta vibraciones o movimientos bruscos y un LED RGB [34]. A pesar de todo obviamente siguen siendo una cantidad ínfima de sensores comparado al número teórico que la red puede manejar, pero también es una forma de mostrar un despliegue más aproximado a un despliegue real donde se utilizan generalmente más de dos sensores.

En la Figura 85 se muestra el conexionado de la nueva implementación, utilizando la herramienta Fritzing para la realización de este esquema.

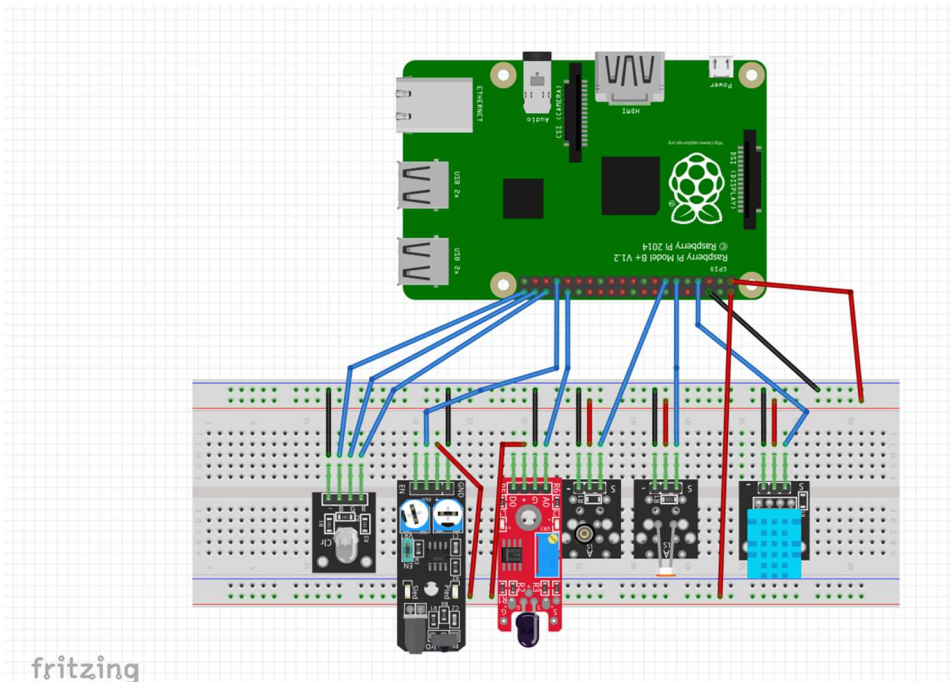


Figura 85. Esquema de conexión de la ampliación IoT.

A continuación, se presenta el nuevo código implementado, el cual es una modificación del código anteriormente presentado donde solo estaban conectados los sensores DHT11 de temperatura y humedad y el sensor de luminosidad. La nueva parte del código se ha obtenido de modificar los códigos mostrados en las siguientes referencias [35] [36] [33] [37].

```
import os
import time
import sys
import paho.mqtt.client as mqtt
import json
from pigpio_dht import DHT11
import RPi.GPIO as GPIO

#Sensor DHT11
gpio=4 #BCM Numbering
sensor=DHT11(4)
```




```
#Sensor LDR de luminosidad
GPIO.setmode(GPIO.BOARD)
delayt = .1
value = 0 # Variable usada para almacenar el valor de LDR
ldr = 11 #Pin al que está conectado
#Sensor avoid (obstáculo)
Pavoid=33
cont_avoid=0 #Variable utilizada para contar cuantos obstáculos se detectan
GPIO.setup(Pavoid, GPIO.IN, pull_up_down=GPIO.PUD_UP)
Pshock=13 #Pin de conexión del sensor shock
# LED RGB
redPin=36
greenPin=38
bluePin=40
GPIO.setup(redPin, GPIO.OUT) #luz roja
GPIO.setup(greenPin, GPIO.OUT) #luz verde
GPIO.setup(bluePin, GPIO.OUT) #luz azul

#Sensor flame (detección de fuego)
flame=32
GPIO.setup(flame, GPIO.IN)
alarma="OFF"

def rc_time (ldr):
    count = 0
    GPIO.setup(ldr, GPIO.OUT)
    GPIO.output(ldr, False)
    time.sleep(delayt)
    GPIO.setup(ldr, GPIO.IN)
    while (GPIO.input(ldr) == 0):
```



```
    count += 1
    return count

def shock(null):
    #Función que cuando detecta una vibración en el sensor ilumina el LED de color morado
    print("Sensor de shock activado")
    mov_brusco="Shock activado"
    purple()
    time.sleep(2)
#Funciones con definiciones de colores posibles para encender el LED
def white():
    GPIO.output(redPin,GPIO.HIGH)
    GPIO.output(greenPin,GPIO.HIGH)
    GPIO.output(bluePin,GPIO.HIGH)

def turnOff():
    GPIO.output(redPin,GPIO.LOW)
    GPIO.output(greenPin,GPIO.LOW)
    GPIO.output(bluePin,GPIO.LOW)

def red():
    GPIO.output(redPin,GPIO.HIGH)
    GPIO.output(greenPin,GPIO.LOW)
    GPIO.output(bluePin,GPIO.LOW)

def green():
    GPIO.output(redPin,GPIO.LOW)
    GPIO.output(greenPin,GPIO.HIGH)
    GPIO.output(bluePin,GPIO.LOW)
```



```
def blue():
```

```
    GPIO.output(redPin,GPIO.LOW)
    GPIO.output(greenPin,GPIO.LOW)
    GPIO.output(bluePin,GPIO.HIGH)
```

```
def yellow():
```

```
    GPIO.output(redPin,GPIO.HIGH)
    GPIO.output(greenPin,GPIO.HIGH)
    GPIO.output(bluePin,GPIO.LOW)
```

```
def purple():
```

```
    GPIO.output(redPin,GPIO.HIGH)
    GPIO.output(greenPin,GPIO.LOW)
    GPIO.output(bluePin,GPIO.HIGH)
```

```
def lightBlue():
```

```
    GPIO.output(redPin,GPIO.LOW)
    GPIO.output(greenPin,GPIO.HIGH)
    GPIO.output(bluePin,GPIO.HIGH)
```

```
def dec_fuego(flame):
```

```
#Función que se ejecuta cuando se detecta fuego por el sensor flame.
```

```
    global alarma
    print("Fuego detectado")
    alarma="ON"
```

```
#Sensor de shock (movimiento)
```

```
GPIO.setup(Pshock, GPIO.IN, pull_up_down=GPIO.PUD_UP)
```

```
#Deteccion de shock
```

```
GPIO.add_event_detect(Pshock, GPIO.FALLING, callback=shock) #El sensor pasa a estado bajo al detectar movimiento
```



```

#Detección de fuego

GPIO.add_event_detect(flame, GPIO.RISING, callback=dec_fuego) #El sensor pasa a
estado alto al detector fuego

#Configuración de parámetros para el envío de información a Thingsboard

THINGSBOARD_HOST = '12.1.1.3' #Dirección IP de la RPi
ACCESS_TOKEN = 'DESPL_IOT_1'

INTERVAL=6

sensor_data = {'temperature': 0, 'humidity': 0, 'luminty':0, 'luz': "ON", 'obstaculo': "NO",
'cont_avoid': 0, 'mov_brusco':"Shock desactivado", 'alarma':"OFF", }

next_reading = time.time()

client = mqtt.Client()

client.username_pw_set(ACCESS_TOKEN)

client.connect(THINGSBOARD_HOST, 1883, 60)

client.loop_start()

#Ejecución de bucle

try:

while True:

    turnOff()

    #Sensor de obstaculos avoid
    sens_avoid=GPIO.input(11)

    #Sensor DHT11
    data=sensor.read(25)

    temperature=data["temp_c"]
    humidity=data["humidity"]

    humidity = round(humidity, 2)

    temperature = round(temperature, 2)

    print(u"Temperature:  {:g}\u00b0C,  Humidity:  {:g}%".format(temperature,
humidity))

    #Sensor luminosidad

```



```
luminosidad = rc_time(ldr)
print(u"Luminosidad: {:g}".format(luminosidad))
if ( luminosidad <= 6000 ):
    sensor_data['luz']= "ON"
if (luminosidad > 6000):
    sensor_data['luz']= "OFF"
#Sensor avoid
if GPIO.input(Pavoid)==0:
    sensor_data['obstaculo']= "SI"
    obstaculo="SI"
    cont_avoid = cont_avoid +1
else:
    sensor_data['obstaculo']= "NO"
    obstaculo="NO"
print("Obstaculo: ",(obstaculo),"Contador", cont_avoid)
#Datos a enviar a thingsboard
sensor_data['temperature'] = temperature
sensor_data['humidity'] = humidity
sensor_data['luminity']=luminosidad
sensor_data[' cont_avoid ']= cont_avoid
sensor_data['mov_brusco']=mov_brusco
sensor_data['alarma']=alarma
# Envio de datos
client.publish('v1/devices/me/telemetry', json.dumps(sensor_data), 1)
next_reading += INTERVAL
sleep_time = next_reading-time.time()
if sleep_time > 0:
    time.sleep(sleep_time)
except KeyboardInterrupt:
    pass
```



```
client.loop_stop()  
client.disconnect()
```

Además de modificar el código también se ha modificado el panel de control dentro de Thingsboard para mostrar la nueva información obtenida de estos sensores, para ello se ha creado un nuevo panel independiente al que se le ha dado el título Despliegue IoT sobre red 5G, como se muestra en la Figura 86. El nuevo panel de datos se muestra en las Figuras 87, 88 y 89, al ser los nuevos datos enviados valores de tipo cadena de caracteres se han utilizado *widgets* de tipo nota.

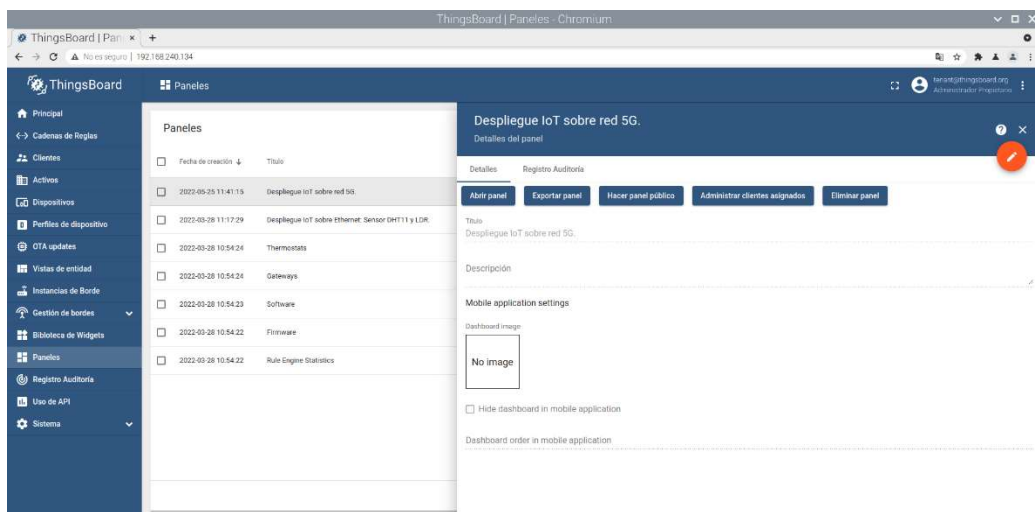


Figura 86. Panel de datos para la ampliación de la implementación IoT.

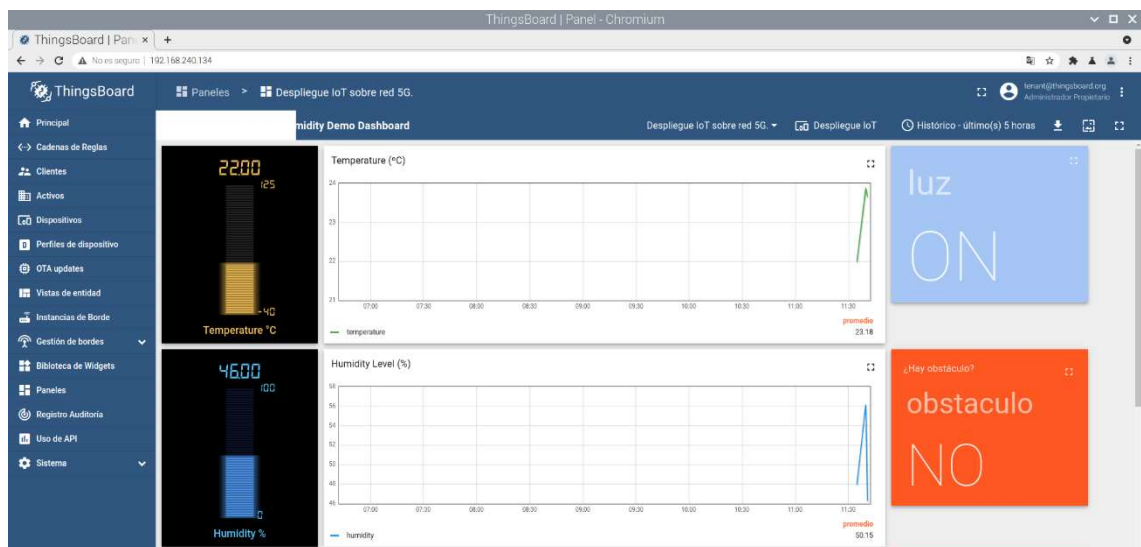


Figura 87. Panel de datos del despliegue IoT ampliado sobre red 5G (1).

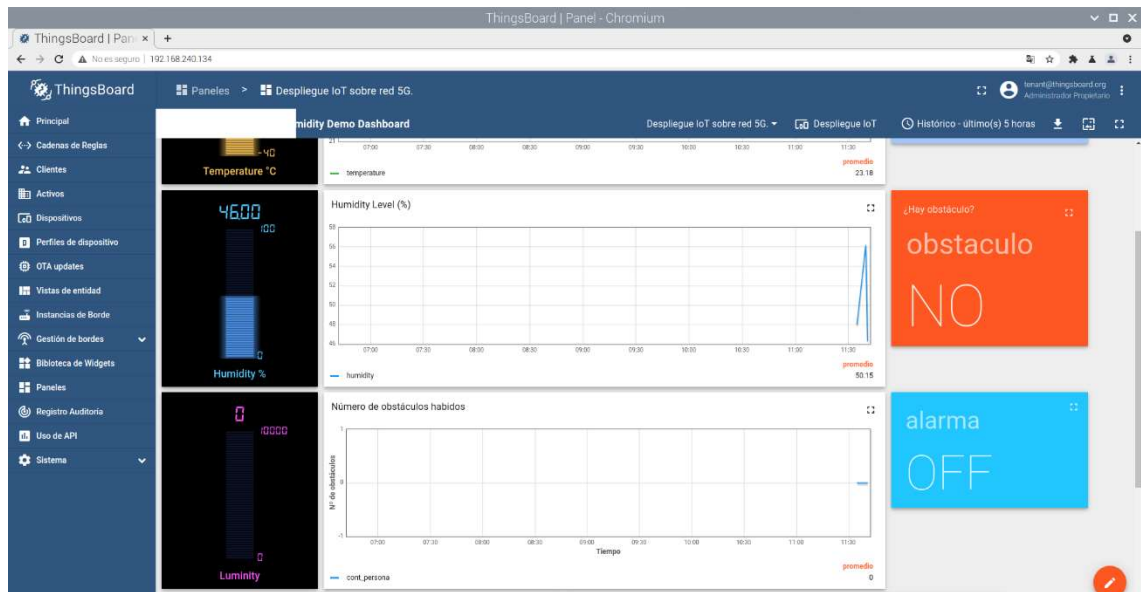


Figura 88. Panel de datos del despliegue IoT ampliado sobre red 5G (2).

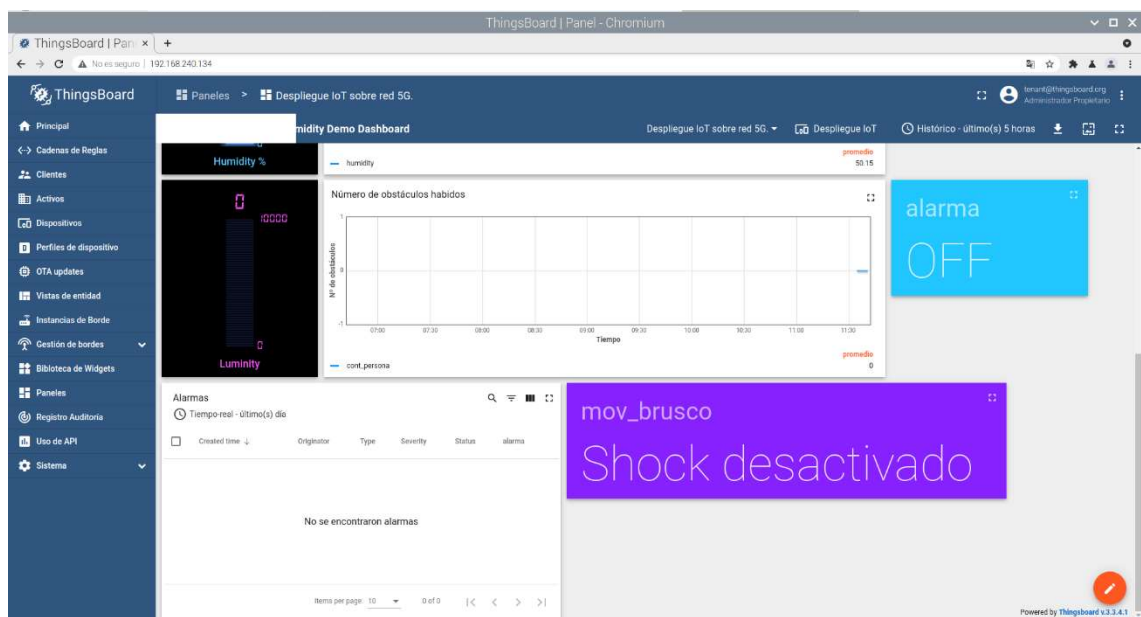


Figura 89. Panel de datos del despliegue IoT ampliado sobre red 5G (3).



4.- COMPARATIVA ENTRE EL DESPLIEGUE LoRaWAN Y 5G.

Durante el presente apartado se presenta una comparativa sobre las dos redes utilizadas, la primera característica a comparar es la tasa de datos a nivel teórico. En el caso de la red LoRaWAN se habla de una tasa de datos teórica de entre 300 bps y 50 Kbps, la cual es bastante inferior a la tasa de datos teórica de la red 5G que puede alcanzar hasta 20 Gbps en el enlace de bajada. La baja tasa de datos en LoRaWAN permite trabajar con bajos consumos, primando el concepto de tecnología de bajo consumo a expensas de la velocidad ya que en términos de las aplicaciones que se despliegan sobre esta red es más prioritario un bajo consumo con dispositivos con baterías en funcionamiento durante largos periodos de tiempo sin necesidad de cambiar la batería o el dispositivo frente a una alta velocidad en el envío de datos del tipo monitorización de la humedad de una zona.

En términos de cobertura en la actualidad ambas redes necesitarían probablemente de un despliegue desde cero ya que, aunque ambas redes parecen tener presencia en gran abundancia de núcleos urbanos, teniendo que revisar en el caso de LoRaWAN la distancia al Gateway más cercano al tener una cobertura inferior a 5 Km en zonas urbanas, no sería en estos momentos común que estas redes estuvieran desplegadas en zonas rurales. En caso de tener que implementar un nuevo despliegue la red LoRaWAN sería más asequible de implementar ya que simplemente sería necesario tener un dispositivo Gateway LoRaWAN y un router con salida ethernet conectado por ejemplo a una red Ethernet, mientras que en el caso de la red 5G sería necesario contratar a una empresa ajena para realizar el despliegue de la misma, lo cual aumentaría sobre todo gastos, así como escoger el dispositivo más adecuado para el despliegue 5G deseado y configurarlo, además se debe realizar la programación de la tarjeta SIM 5G que utilice el dispositivo encargado de dar soporte 5G al despliegue, tal como se ha presentado anteriormente. La cantidad de pasos necesarios para el despliegue de la red LoRaWAN es bastante inferior a la de la red 5G, así como su complejidad es menor. Además, la cobertura actual de la red 5G situando la estación base en una habitación es de un rango apenas superior al de la propia habitación, como se ha comprobado en el laboratorio que la cobertura de la red apenas llegaba fuera de la propia sala y en cuanto te alejabas un par de pasos más de la puerta de salida la cobertura se perdía, a diferencia de los hasta 5 Km soportados por la red LoRaWAN es una distancia bastante inferior.

Una de las desventajas de la red LoRaWAN frente a la red 5G es la necesidad que tiene LoRa de utilizar otras redes para su funcionamiento, ya que incluso su Gateway necesita estar conectado a otra red conectada a internet para su funcionamiento, así como



su estructura de red, en la cual se ha visto que es necesario un servidor de red, como es The Things Network como elemento intermediario al que el Gateway envía los datos recogidos y estos datos deberán ser recogidos por la aplicación final de este servidor de red, necesitando esta aplicación final de otra red conectada a internet para acceder, utilizando el protocolo MQTT, a los datos recibidos en The Things Network y posteriormente almacenar y mostrar estos datos en dicha aplicación. Mientras en el caso de la red 5G no es necesaria más que la propia red 5G durante todo el proceso, ya que no solo con la red 5G es posible enviar directamente los datos recogidos de los sensores a una aplicación final donde almacenarlos y representarlos, sino que además es posible visualizar estos datos usando una conexión 5G sin necesidad de depender de la existencia de otras redes.

Ambas redes están pensadas para soportar un alto número de dispositivos conectados a ellas, siendo esta característica una de las principales ventajas de ambas, para la red 5G se estima una capacidad teórica de esta 1.000.000 de dispositivos por Km² mientras que en el caso de LoRaWAN no se habla de ninguna estimación teórica de número de dispositivos, pero se habla de una elevada cantidad, aunque parece lógico pensar que en ningún caso llegará a igual el número de dispositivos posibles conectados a la vez a 5G, debido a que los datos enviados por cada dispositivo a The Things Network deben ser enviados en distintos instantes de tiempo para su correcto recibo, lo cual se complica al ir elevando el número de dispositivos conectados. Para poder comparar de forma práctica si hay gran diferencia entre la cantidad de dispositivos que se pueden conectar a ambas redes sin degradar la velocidad de transmisión de información a tasas notablemente inferiores sería necesario conectar un gran número de dispositivos, probablemente cercano a los millares o al menos a los cientos, lo cual no ha sido posible realizar físicamente en el laboratorio, a pesar de haber intentado conectar más dispositivos en el segundo despliegue sobre la red 5G, 6 dispositivos conectados es una cantidad ínfima, de igual forma ocurre con los solo dos dispositivos conectados a la red LoRaWAN.

No solo se debe tener en cuenta la cantidad de dispositivos soportados por la red sino la existencia del dispositivo deseado para trabajar sobre esa red, en el caso de los sensores no se ha visto ningún tipo de problema en ninguna de las dos debido a que a ambas se les pueden conectar controladores de tipo Arduino o Raspberry Pi los cuales soportan una gran variedad de sensores diferentes. Pero en el caso de dispositivos capaces de realizar el despliegue de las redes no existe tanta variedad actual para la red 5G como lo hay para la red LoRaWAN. Se ha visto que es bastante sencillo encontrar Gateways de LoRaWAN para el despliegue de la red, sin embargo actualmente a pesar de haber variedad de dispositivos capaces de trabajar con la red 5G o de desplegarla no son demasiados, además de que la mayoría de los dispositivos actuales compatibles lo son



solo con la tecnología de 5G que utiliza un núcleo 4G para su funcionamiento, lo cual se denomina 5G NSA o 5G NonStandAlone, como son el caso del router y el smartphone utilizado, la tecnología 5G que utiliza núcleo 5G propiamente dicho y que se denomina 5G SA o 5G StandAlone [38] no se encuentra implementada en una alta variedad de dispositivos, en este caso el único dispositivo capaz de trabajar así es el módulo 5G. Actualmente, no parece un gran problema debido a que muchas de las redes desplegadas siguen un modelo 5G NSA, este es el caso de la utilizada en el laboratorio. En la Figura 90 se muestra de forma gráficamente los escenarios de 5G NSA, 5G SA y el escenario inicial de 4G para poder comprender mejor de forma visual este concepto, la imagen se ha obtenido de la referencia [38].

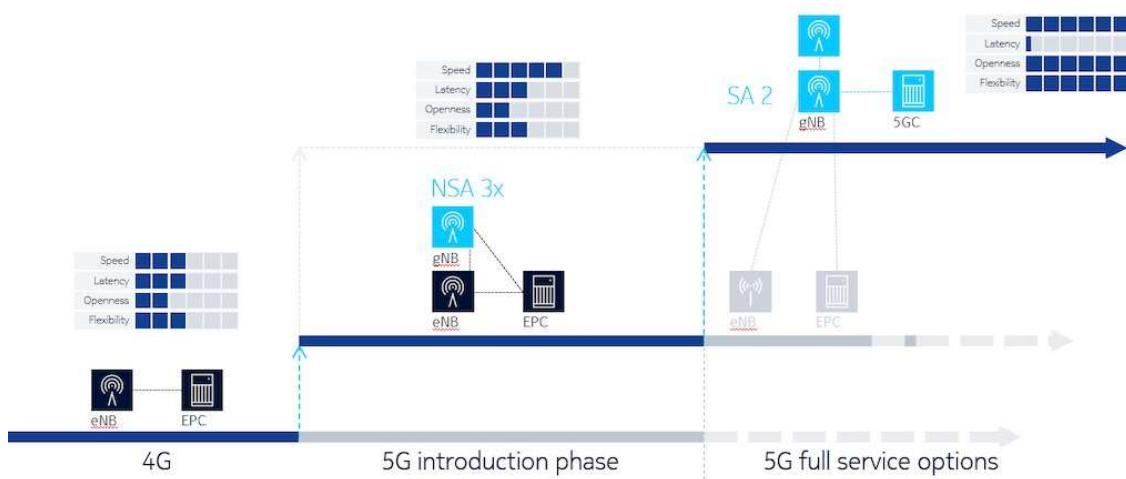


Figura 90. Comparativa de los diferentes escenarios de funcionamiento de 5G [38].

A continuación de forma gráfica en la Tabla 1 se presenta una comparativa de las principales características teóricas de ambas redes, las cuales se han mencionado en sus respectivos apartados teóricos.

| | Características | | | | | |
|-------------|---|--|-------------------|--|---------------------------|--------------------------------------|
| | Rango de cobertura | Banda de frecuencia | Tasa de datos | Nº de dispositivos conectables | Complejidad de despliegue | Dispositivos comerciales disponibles |
| Red LoRaWAN | 10-40 Km (zona rural) 1-5 Km (zona urbana) | Bandas no licenciadas ISM (868 MHz Europa) | 300 bps - 50 Kbps | Elevado | Reducida | Gran variedad |
| Red 5G | >> 1 Km | 700 MHz y 3,5 GHz | 10 Gbps - 20 Gbps | Muy elevado (~1.000.000 disp./Km ²) | Muy elevada | Variedad más limitada |

Tabla 1. Comparativa teórica de las redes LoRaWAN y 5G.





5.- CONCLUSIONES.

Con la situación en la que se encuentra actualmente el despliegue de red 5G parece más cómodo y rápido optar por utilizar la red LoRaWAN para un despliegue IoT debido a su fácil instalación y funcionamiento tal como se ha presentado, pero en este caso se ha de tener en cuenta que es necesario estar en un entorno con acceso a otra red distinta, mientras que en el momento en el que la red 5G esté lo suficientemente desplegada solo se necesitaría esta red para realizar un despliegue de IoT completo. Además del dispositivo proveedor de la red, en este caso para el despliegue sobre 5G solo han sido necesarios los sensores y una RPi donde ejecutar el programa de envío de la información de los sensores a la aplicación así como la ejecución de la aplicación a la cual se accede desde cualquier dispositivo conectado a la propia red 5G, mientras que para el despliegue sobre LoRaWAN además de los dispositivos propios de LoRaWAN como son los nodos y el Gateway ha sido necesario también el uso de una RPi conectada a para realizar la conexión del servidor de red The Things Network con la aplicación InfluxDB utilizada.

Las principales ventajas que supone utilizar la red 5G frente a la red LoRaWAN son las siguientes, no necesita más que la propia red 5G para realizar todas las conexiones necesarias durante el despliegue, la tasa de datos es superior en caso de necesitarlo, también es posible controlar el consumo a partir de la modificación del programa de envío de datos de los sensores para que este envío se realice de forma periódica en un tiempo por ejemplo de 10 minutos de separación entre envío de datos y a la vez es posible enviar datos de alarmas, por ejemplo del sensor de detección de incendios, en el momento en el que se detecte sin afectar al resto de tiempos. Frente a estas ventajas la mayor desventaja que supondría utilizar la red 5G sería su rango de cobertura, pero en el supuesto de tener un despliegue total de esta red en la zona a desplegar el sistema IoT este problema sería probablemente solucionado por el operador instalador de la red 5G en cuestión.

También es necesario tener en cuenta la facilidad del despliegue de la red LoRaWAN frente al despliegue de la red 5G, donde el despliegue de la red no es tan inmediato y lleva un periodo de tiempo de instalación y puesta en marcha más largo.

En cualquier caso, la decisión final de que despliegue IoT se adapta mejor a lo que se desea desplegar dependerá tanto las características del propio despliegue, ya sea de los dispositivos que se van a conectar o el tiempo de vida que tienen estos dispositivos, de lo cual dependerá que consumo tiene que haber en los mismos, como del lugar donde se quiere realizar el despliegue y de las redes a las que se tiene acceso en ese lugar.



6.- REFERENCIAS

- [1] K. Shafique, B. A. Khawaja, F. Sabir, S. Qazi y M. Mustaqim, «Internet of things (IoT) for next-generation smart systems: A review of current challenges, future trends and prospects for emerging 5G-IoT Scenarios,» *IEEE Access*, vol. 8, pp. 23022-23040, 2020.
- [2] «Homepage - LoRa Alliance®,» [En línea]. Available: <https://loralliance.org/>. [Último acceso: 22 Marzo 2022].
- [3] K. Mekki, E. Bajic, F. Chaxel y F. Meyer, «A comparative study of LPWAN technologies for large-scale IoT deployment,» *ICT Express*, vol. 5, nº 1, pp. 1-7, 2019.
- [4] «Mikrotik wAP LR8 kit,» Mikrotik, [En línea]. Available: https://mikrotik.com/product/wap_lr8_kit. [Último acceso: 08 Junio 2022].
- [5] MikroTik, «MikroTik Routers and Wireless - Products: 868 Omni antenna,» MikroTik, [En línea]. Available: https://mikrotik.com/product/868_omni_antenna. [Último acceso: 13 Junio 2022].
- [6] «LHT65 LoRaWAN Temperature & Humidity Sensor,» Dragino, [En línea]. Available: <https://www.dragino.com/products/temperature-humidity-sensor/item/151-lht65.html>. [Último acceso: 08 Junio 2022].
- [7] LHT65 Temperature & Humidity sensor LHT65 Temperature & Humidity Sensor User Manual Document Version: 1.3 Image Version: v1.4.
- [8] «Heltec CubeCell – Dev-Board,» Heltec, [En línea]. Available: <https://heltec.org/project/htcc-ab01/>. [Último acceso: 08 Junio 2022].
- [9] «Sensor BME280 Temp Presión Humedad - Tienda Prometec,» [En línea]. Available: <https://store.prometec.net/producto/sensor-bme280/>. [Último acceso: 10 Marzo 2022].



- [10] «Contents — Heltec Automation Docs V0.0.1 documentation,» [En línea]. Available: <https://heltec-automation-docs.readthedocs.io/en/latest/cubecell/index.html>. [Último acceso: 16 Marzo 2022].
- [11] «CubeCell Series Quick Start — Heltec Automation Docs V0.0.1 documentation,» [En línea]. Available: https://heltec-automation-docs.readthedocs.io/en/latest/cubecell/quick_start.html. [Último acceso: 16 Marzo 2022].
- [12] «Publicadores y suscriptores - Documentación de IBM,» [En línea]. Available: <https://www.ibm.com/docs/es/ibm-mq/8.0?topic=ssfksj-8-0-0-com-ibm-mq-explorer-doc-p-pubsubers-htm>. [Último acceso: 18 Abril 2022].
- [13] A. S y M. M, *Proceedings, 2017 International Conference on Engineering & MIS (ICEMIS), (ICEMIS'2017) : University of Monastir, Monastir, Tunisia, 08-10 May, 2017*.
- [14] «La pirámide de 5G: Servicios eMBB,URLLC y mMTC - Comunidad Huawei Enterprise,» [En línea]. Available: <https://forum.huawei.com/enterprise/es/la-pir%C3%A1mide-de-5g-servicios-embb-urllc-y-mmtc/thread/743089-100763>. [Último acceso: 18 Abril 2022].
- [15] «Plane Latency - an overview | ScienceDirect Topics,» [En línea]. Available: <https://www.sciencedirect.com/topics/engineering/plane-latency>. [Último acceso: 18 Abril 2022].
- [16] J. N, S. A, N. H y G. N, «Intelligence in IoT-Based 5G Networks: Opportunities and Challenges,» *IEEE Communications Magazine*, vol. 56, n° 10, pp. 94-100, 2018.
- [17] I. o. E. a. E. Engineers, 2018 Global Information Infrastructure and Networking Symposium (GIIS) : 23-25 Oct. 2018, 2018.
- [18] «KY-015 Temperature and Humidity Sensor Module,» Arduino Modules, [En línea]. Available: <https://arduinomodules.info/ky-015-temperature-humidity-sensor-module/>. [Último acceso: 08 Junio 2022].



- [19] «KY-018 Photoresistor module.» Arduino Modules, [En línea]. Available: <https://arduinomodules.info/ky-018-photoresistor-module/>. [Último acceso: 08 Junio 2022].
- [20] «Installing ThingsBoard on Raspberry Pi 3 Model B | ThingsBoard Community Edition,» [En línea]. Available: <https://thingsboard.io/docs/user-guide/install/rpi/>. [Último acceso: 6 Abril 2022].
- [21] «Fritzing,» [En línea]. Available: <https://fritzing.org/>. [Último acceso: 13 Junio 2022].
- [22] «Temperature upload over MQTT using Raspberry Pi and DHT22 sensor | ThingsBoard Community Edition,» [En línea]. Available: <https://thingsboard.io/docs/samples/raspberry/temperature/>. [Último acceso: 6 Abril 2022].
- [23] «Raspberry pi 3 LDR Sensor, circuit and python programming,» [En línea]. Available: <https://www.electronicclinic.com/raspberry-pi-3-ldr-sensor-circuit-and-python-programming/>. [Último acceso: 6 Abril 2022].
- [24] «Router móvil WiFi 6 M5 5G Nighthawk - MR5200 | NETGEAR,» Netgear, [En línea]. Available: <https://www.netgear.com/es/home/mobile-wifi/hotspots/mr5200/>. [Último acceso: 11 Mayo 2022].
- [25] «sysmoISIM-SJA2 SIM + USIM + ISIM Card (10-pack) with ADM keys,» sysmocom, [En línea]. Available: <http://shop.sysmocom.de/products/sysmoISIM-SJA2>. [Último acceso: 08 Junio 2022].
- [26] firecell, *LabKit - User Guide*, 2022.
- [27] «OnePlus Nord2 5G,» OnePlus, [En línea]. Available: <https://www.oneplus.com/es/nord-2-5g>. [Último acceso: 08 Junio 2022].
- [28] C. M, O. E, D. Y, N. S, d. R. M, B. H, D. M y G. P, «Mobile tethering: Overview, perspectives and challengess,» vol. 16, nº 3, pp. 40-53, 2014.



- [29] «Quectel 5G RM50xQ series,» Quectel, [En línea]. Available: <https://www.quectel.com/product/5g-rm50xq-series>. [Último acceso: 08 Junio 2022].
- [30] J. Geerling, «Using 4G LTE wireless modems on a Raspberry Pi,» 20 Enero 2022. [En línea]. Available: <https://www.jeffgeerling.com/blog/2022/using-4g-lte-wireless-modems-on-raspberry-pi>. [Último acceso: 23 Mayo 2022].
- [31] «KY-032 Infrared Obstacle Avoidance Sensor Module,» Arduino Modules, [En línea]. Available: <https://arduinomodules.info/ky-032-infrared-obstacle-avoidance-sensor-module/>. [Último acceso: 08 Junio 2022].
- [32] «KY-026 Flame Sensor Module,» Arduino Modules, [En línea]. Available: <https://arduinomodules.info/ky-026-flame-sensor-module/>. [Último acceso: 08 Junio 2022].
- [33] «KY-031 Vibration Sensor Module,» Arduino Modules, [En línea]. Available: <https://arduinomodules.info/ky-002-vibration-switch-module/>. [Último acceso: 25 Mayo 2022].
- [34] «KY-016 RGB Full color LED Module,» Arduino Modules, [En línea]. Available: <https://arduinomodules.info/ky-016-rgb-full-color-led-module/>. [Último acceso: 08 Junio 2022].
- [35] «BUILD YOUR OWN RASPBERRY-PI COLLISION AVOIDANCE ROBOT FOR BEGINNERS IN PYTHON | by afroRobotist | Medium,» [En línea]. Available: <https://medium.com/@baobo.obi/build-your-own-raspberry-pi-collision-avoidance-robot-for-beginners-in-python-bfaa57f9fce6>. [Último acceso: 25 Mayo 2022].
- [36] «Flame Sensor (Raspberry Pi) : 4 Steps (with Pictures) - Instructables,» [En línea]. Available: <https://www.instructables.com/Flame-Sensor-Raspberry-Pi/>. [Último acceso: 25 Mayo 2022].
- [37] «Raspberry Pi Tutorial: How to Use a RGB LED : 4 Steps - Instructables,» [En línea]. Available:



<https://www.instructables.com/Raspberry-Pi-Tutorial-How-to-Use-a-RGB-LED/>. [Último acceso: 25 Mayo 2022].

- [38] J. Harvey, «How to tackle the burning issue of moving from 5G NSA to SA,» Nokia, [En línea]. Available: https://www.nokia.com/blog/how-to-tackle-the-burning-issue-of-moving-from-5g-nsa-to-sa/?did=d00000000608&gclid=CjwKCAjwTlaVBhBkEiwAsr7-c7FOmW158V4YQUGaElgc4ww3iA2Z3MEb6XWYIYZNGKaJ-SyMTWO4JBoCOXUQAvD_BwE. [Último acceso: 09 Junio 2022].



7.- ÍNDICE DE FIGURAS.

| | |
|--|----|
| Figura 1.Estructura de red LoRaWAN..... | 7 |
| Figura 2. Mapa con Gateways de The Thing Network. | 8 |
| Figura 3. Console de TTN..... | 8 |
| Figura 4. Configuración Gateway Mikrotik (1). | 9 |
| Figura 5. Configuración Gateway Mikrotik (2). | 10 |
| Figura 6.Configuración gateway Mikrotik (2). | 10 |
| Figura 7. Configuración Gateway Mikrotik (3). | 11 |
| Figura 8.Información del dispositivo Mikrotik en su parte posterior. | 12 |
| Figura 9.Configuración gateway Mikrotik (4). | 13 |
| Figura 10. Configuración Gateway en TTN (1). | 14 |
| Figura 11.Configuración Gateway en TTN (2). | 14 |
| Figura 12. Visualización del Gateway en TTN. | 15 |
| Figura 13. Creación de aplicación en TTN. | 16 |
| Figura 14. Creación de End device dentro de la aplicación de TTN. | 17 |
| Figura 15. Registro de dispositivo del repositorio LoRaWAN. | 18 |
| Figura 16. Registro de dispositivo de forma manual. | 18 |
| Figura 17. Datos dados por el fabricante para el registro del LHT65. | 19 |
| Figura 18. Pestaña Payload formatters para la decodificación de la información. | 20 |
| Figura 19. Sensor LHT65 en funcionamiento en TTN. | 20 |
| Figura 20. Imagen del CubeCell utilizado con el sensor BME/BMP280. | 21 |
| Figura 21. Placa CubeCell en Arduino. | 22 |
| Figura 22. Códigos ejemplos para ejecutar en CubeCell. | 23 |
| Figura 23. Registro de dispositivo CubeCell en TTN. | 24 |
| Figura 24. Código de decodificación para el sensor BMP280. | 25 |
| Figura 25. Modificación del código de Arduino para retrasar el envío de datos. . | 26 |
| Figura 26. Página de inicio de Influxdb. | 28 |



| | |
|--|----|
| Figura 27. Fuente de datos en Influxdb..... | 28 |
| Figura 28. Menú MQTT Consumer. | 29 |
| Figura 29. Configuración plugin MQTT Consumer. | 29 |
| Figura 30. Creación del bucket. | 30 |
| Figura 31. Creación de API Token. | 30 |
| Figura 32. Creación de configuración Telegraf (1)..... | 31 |
| Figura 33. Creación de configuración Telegraf (2)..... | 31 |
| Figura 34. Creación de configuración Telegraf (3)..... | 32 |
| Figura 35. Instrucciones para arrancar el servicio Telegraf. | 34 |
| Figura 36. Pantalla de Dashboard de Influxdb..... | 35 |
| Figura 37. Dashboard Cubecell+BMP280. | 35 |
| Figura 38. Dashboard Sensor LHT-65. | 36 |
| Figura 39. Pantalla inicio de sesión en Thingsboard..... | 39 |
| Figura 40. Pantalla inicial de Thingsboard. | 40 |
| Figura 41. Pantalla Dispositivos en Thingsboard. | 40 |
| Figura 42. Creación del dispositivo. | 41 |
| Figura 43. Creación de las credenciales del dispositivo. | 42 |
| Figura 44. Detalles del dispositivo creado. | 42 |
| Figura 45. Esquema del conexionado de los sensores a la RPi..... | 43 |
| Figura 46. Pantalla Paneles de Thingsboard. | 47 |
| Figura 47. Información del panel creado. | 47 |
| Figura 48. Visualización del panel de control creado. | 48 |
| Figura 49. Modelo de tarjeta SIM a utilizar para el despliegue 5G..... | 49 |
| Figura 50. Salida del terminal tras ejecutar el comando \$./sysmo-isim-tool.sja2.py --adm1 {ADM1} -o -k..... | 52 |
| Figura 51. Salida del terminal tras ejecutar los comandos CAMBIAR CAPTURA. | 52 |
| Figura 52. Dispositivo Netgear Nighthawk M5 Mobile Router. | 53 |



| | |
|--|----|
| Figura 53. Pantalla de arranque del dispositivo Netgear donde seleccionar idioma. | 54 |
| Figura 54. Pantalla de encendido del dispositivo Netgear. | 55 |
| Figura 55. Proceso de configuración inicial del dispositivo Netgear (1)..... | 55 |
| Figura 56. Proceso de configuración inicial del dispositivo Netgear (2)..... | 56 |
| Figura 57. Proceso de configuración inicial del dispositivo Netgear (3)..... | 56 |
| Figura 58. Proceso de configuración inicial del dispositivo Netgear (4)..... | 57 |
| Figura 59. Configuración del APN en el dispositivo Netgear (1)..... | 57 |
| Figura 60. Configuración del APN en el dispositivo Netgear (2)..... | 58 |
| Figura 61. Dispositivo Netgear conectado a la red 5G y en funcionamiento. | 58 |
| Figura 62. Configuración del APN en smartphone (1). | 60 |
| Figura 63. Configuración del APN en smartphone (2). | 61 |
| Figura 64. Configuración del APN en smartphone (3). | 61 |
| Figura 65. Smartphone funcionando con la tarjeta SIM 5G. | 62 |
| Figura 66. Configuración de tethering mediante USB (1). | 63 |
| Figura 67. Comprobación en la RPi de la correcta conexión a 5G mediante smartphone. | 64 |
| Figura 68. Pantalla para entrar en modo desarrollador en el smartphone OnePlus Nord 2 5G. | 65 |
| Figura 69. Pantalla de Opciones de desarrollador..... | 65 |
| Figura 70. Modificaciones en Opciones de desarrollador para el funcionamiento del tethering por USB (1). | 66 |
| Figura 71. Modificaciones en Opciones de desarrollador para el funcionamiento del tethering por USB (2). | 67 |
| Figura 72. Modificaciones en Opciones de desarrollador para el funcionamiento del tethering por USB (3). | 68 |
| Figura 73. Modificaciones en Opciones de desarrollador para el funcionamiento del tethering por USB (4). | 68 |
| Figura 74. Acceso al panel de datos de Thingsboard desde el smartphone suministrador de red 5G. | 69 |



| | |
|---|----|
| Figura 75. Elementos necesarios para el montaje del adaptador de módulo 5G a USB 3.0. | 70 |
| Figura 76. Adaptador 5G a USB con módulo de Queltec montado..... | 71 |
| Figura 77. Salida de la ejecución del comando ifconfig con el módulo conectado inicialmente. | 72 |
| Figura 78. Comprobación de conexión a internet de la nueva red 5G..... | 72 |
| Figura 79. Ejecución del comando \$lsub..... | 73 |
| Figura 80. Configuración de modulo Queltec 5G (1). | 74 |
| Figura 81. Configuración de modulo Queltec 5G (2). | 75 |
| Figura 82. Configuración de modulo Queltec 5G (3). | 76 |
| Figura 83. Configuración de modulo Queltec 5G (4). | 76 |
| Figura 84. Configuración de modulo Queltec 5G (5). | 77 |
| Figura 85. Esquema de conexionado de la ampliación IoT. | 79 |
| Figura 86. Panel de datos para la ampliación de la implementación IoT. | 85 |
| Figura 87. Panel de datos del despliegue IoT ampliado sobre red 5G (1). | 85 |
| Figura 88. Panel de datos del despliegue IoT ampliado sobre red 5G (2). | 86 |
| Figura 89. Panel de datos del despliegue IoT ampliado sobre red 5G (3). | 86 |
| Figura 90. Comparativa de los diferentes escenarios de funcionamiento de 5G [38]. | 89 |



8.- ÍNDICE DE TABLAS.

Tabla 1. Comparativa teórica de las redes LoRaWAN y 5G. 89