ESCUELA TÉCNICA SUPERIOR DE INGENIEROS INDUSTRIALES Y DE TELECOMUNICACIÓN

UNIVERSIDAD DE CANTABRIA

Trabajo Fin de Grado

Despliegue Y Evaluación de una Red On-Site LoRaWAN basada en The Things Network Stack versión 3

(Deployment and Evaluation of an On-Site LoRaWAN Network based on The Things Stack version 3)

Para acceder al Título de

Graduado en

Ingeniería de Tecnologías de Telecomunicación

Autor: Juan Carlos Merino Polidura

Octubre - 2019

GRADUADO EN INGENIERÍA DE TECNOLOGÍAS DE TELECOMUNICACIÓN

CALIFICACIÓN DEL TRABAJO FIN DE GRADO

Realizado por: Juan Carlos Merino Polidura

Director del TFG: Luis Sánchez González

Título: "Despliegue Y Evaluación de una Red On-Site LoRaWAN basada en The Things Network Stack versión 3 "

Title: "Deployment and Evaluation of an On-Site LoRaWAN Network based on The Things Stack version 3 **"**

Presentado a examen el día:

para acceder al Título de

GRADUADO EN INGENIERÍA DE TECNOLOGÍAS DE TELECOMUNICACIÓN

Composición del Tribunal:

Presidente (Apellidos, Nombre): Eduardo Artal Latorre

Secretario (Apellidos, Nombre): Jorge Lanza Calderón

Vocal (Apellidos, Nombre): Luis Sánchez González

Este Tribunal ha resuelto otorgar la calificación de: .....

Fdo.: El Presidente

Fdo.: El Secretario

Fdo.: El Vocal

Fdo.: El Director del TFG

(sólo si es distinto del Secretario)

Vº Bº del Subdirector

Trabajo Fin de Grado Nº
(a asignar por Secretaría)

RESUMEN

Hoy en día, cualquier persona puede acceder a Internet con facilidad, por lo que estamos cada vez más interconectados tanto entre personas, como con los objetos que nos rodean.

De esto último surge el concepto de la Internet de las Cosas, cuyo objetivo es crear una infraestructura en la que podamos acceder con facilidad a los dispositivos que deseemos en cada momento, e interconectar dichos dispositivos entre sí para que, sin que exista una intervención humana, puedan intercambiar información entre ellos.

Sin embargo, para la realización de este paradigma es necesario seleccionar cuidadosamente la tecnología de comunicaciones que emplear puesto que es imprescindible optimizar el uso de los recursos de los que disponen los objetos. En nuestro caso, buscamos una tecnología que permita el envío de datos entre nodos muy distanciados y además necesitamos que el consumo de energía de estos nodos sea el mínimo posible, para que las baterías con las que habitualmente están equipadas tengan una vida del orden de años.

En este punto entran las tecnologías LoRa y LoRaWAN, cuyas características se adecúan a nuestros requisitos ofreciendo tanto grandes áreas de cobertura como consumos reducidos.

En este proyecto se aborda el despliegue y configuración de una red LoRaWAN que utilice el stack de código abierto de The Things Network. Además de analizar las funcionalidades de este stack y comparándolas con las descripciones teóricas de la especificación de LoRaWAN, el proyecto pretende definir una serie de casos de uso que permitan trasladar el análisis realizado en este trabajo a un ámbito más didáctico que se pueda emplear en laboratorios docentes en los que los alumnos puedan tener un contacto más directo con estas tecnologías.

Para terminar se extraerán unas conclusiones de los resultados obtenidos en el análisis y se darán unas ideas en lo referido a las líneas futuras de esta tecnología.

ABSTRACT

Nowadays, anyone can access the Internet easily, so we are increasingly interconnected between people as well as with objects around us.

From the latter arises the concept of the Internet of Things, whose objective is to creat e an infrastructure in which we can easily access to the devices we want at all times, and interconnecting these devices with each other so that, without human intervention, they can exchange information between them.

However, in order to realize this paradigm it is necessary to carefully select the communications technology to use since it is essential to optimize the use of the resources available to objects. In our case, we are looking for a technology that allows connecting far-off nodes while keeping power consumption of these nodes to the minimum possible, so that the batteries with which they are usually equipped have a life in the order of years.

LoRa and LoRaWAN, technologies adapt to our requirements offering both large areas of coverage and reduced consumptions.

This project describes the deployment and configuration of a LoRaWAN network that uses The Things Network's open source stack. In addition to analyzing the functionalities of this stack and comparing them with the theoretical descriptions of the LoRaWAN specification, the project aims to define a series of use cases that allow to move the analysis carried out in this work to a more didactic field that can be used in teaching laboratories where students can have more direct contact with these technologies.

Finally, conclusions will be drawn from the results obtained in the analysis and some future lines of work will be sketched.

Índice de Contenidos

Índice de figuras

Índice de tablas

Índice de acrónimos

ABP	Activation-by-Personalization
API	Application Programming Interface
AppSKey	Application Server Key
AS	Application Server
CA	Certification Authority
CLI	Command Line Interface
EUI	Extended Unique Identifier
HTTP	HyperText Transfer Protocol
IoT	Internet of Things
IP	Internet Protocol
ISM	Industrial, Scientific and Medical
LoRa	Long Range
LoRaWAN	Long Range Wide Area Network
LPWAN	Low-Power Wide-Area-Network
MAC	Media Access Control
MQTT	Message Queue Telemetry Transport
NB-IoT	Narrowband-IoT
SF	Spreading Factor
NS	Network Server
NTP	Network Time Protocol
NwkSKey	Network Server Key
OTAA	Over-The-Air-Activation
PHY	Física

RPMA	Random Phase Multiple Access
RSSI	Received Signal Strength Indicator
SNR	Signal Noise Ratio
TCP	Transmission Control Protocol
TFG	Trabajo de Fin de Grado
TTN	The Things Network
V.3	Version 3
WiFi	Wireless Ethernet Compatibility Alliance

1. INTRODUCCIÓN

La irrupción de la Internet de las Cosas (IoT, por sus siglas en inglés) está revolucionando el mundo de las tecnologías de la información y una buena prueba de ello es, que a día de hoy, hay aproximadamente 30 billones de dispositivos de IoT en funcionamiento [1], y se calcula que en unos 5 años este número se duplicará.

El internet de las Cosas es la interconexión de dispositivos y objetos a través de la red, donde pueden interactuar entre ellos.

La razón de increíble aumento es el abanico de posibilidades que nos ofrece este paradigma gracias a su capacidad para recolectar, almacenar, monitorizar datos en tiempo real y así, poder escoger la mejor alternativa en cada situación en el menor tiempo posible.

Actualmente, se está utilizando en ámbitos del hogar, en agricultura, en industria, en sanidad, supermercados, automóviles, etc. En realidad, en cualquier situación que nos imaginemos puede resultar de enorme utilidad.

Uno de los pilares del concepto de la IoT está en la ubicuidad de las nuevas tecnologías de comunicación inalámbricas. En especial aquellas que ofrecen esa ubicuidad con bajo coste y sobre todo bajo consumo energético.

Estas características las reúnen las tecnologías catalogadas como LPWAN. Entre ellas podemos encontrar ejemplos como Sigfox [2], RPMA [3], NB-IoT [4], LoRaWAN [5], etc.

Este proyecto se concentra en la tecnología LoRaWAN, que actualmente es la que más penetración tiene dentro de este tipo de redes.

En él, se expondrán sus principales características, su arquitectura de red, sus tipos de dispositivos y los modos de operación de estos. Además, se detallarán los dispositivos utilizados en el despliegue de una infraestructura LoRaWAN y los códigos (software) con los que estos se han programado.

1.1 Motivaciones

La red IoT está en auge en la actualidad y en continua evolución. A nivel de comunicaciones e infraestructura de red la tecnología que hoy en día concentra mayor interés son las denominadas redes LPWAN, que, como ya se ha comentado previamente, son muy eficientes desde el punto de vista energético, de coste y de cobertura, sustituyendo así a las redes multisalto basadas en el protocolo 802.15.4.[6]

Por ello, en este proyecto se ha propuesto instalar el núcleo de una red LoRaWAN haciendo uso del stack open source, que distribuye The Things Network (TTN), para posteriormente evaluar y analizar el comportamiento del sistema ante una serie de casos de uso. La selección y definición de estos casos de uso se fundamenta en poder estudiar las funcionalidades básicas de la red como son, formación de la red, registro de dispositivos y transmisión tanto uplink como downlink.

Además, tanto el despliegue como la definición y el análisis de estos casos se ha orientado de forma que se pueda utilizar como laboratorio docente para las titulaciones de Grado y Máster que actualmente se imparten.

1.2 Objetivos

El principal objetivo de este Trabajo de Fin de Grado (TFG) es el despliegue de una red LoRaWAN sobre la cual realizar el análisis de las características fundamentales de cada uno de los elementos que componen su arquitectura de red.

Para lograr este objetivo primario se han definido una serie de objetivos más específicos que de manera combinada, permitirán alcanzar la meta final:

1. Instalación del stack V.3 de TTN
   Breve resumen de las pautas seguidas durante el proceso de instalación y configuración de la red.
2. Definición de casos de uso fundamentales
3. Programación de sensores equipados con interfaces LoRa
   Para introducir nuestros dispositivos en la red es necesario implementar en ellos un software concreto para su correcto funcionamiento.
4. Análisis de funcionalidades de red
   Una vez esté configurada la red, se evaluará su comportamiento ante ciertas interacciones que ocurran en ella.

1.3 Resumen ejecutivo

La memoria de este Trabajo de Fin de Grado está organizada en cuatro capítulos.

En el primero de ellos, se orienta al lector hacia el propósito de este proyecto haciendo un breve resumen de las condiciones de contorno que motivan este TFG y presentando los objetivos del mismo.

En el Capítulo 2 se hace un repaso de las principales tecnologías que se han utilizado. En este sentido, se describe la tecnología LoRa, y más en profundidad, la tecnología LoRaWAN. Además, se presentan los protocolos y marcos de desarrollo software que se han empleado.

La descripción del despliegue de la red LoRaWAN que se han realizado, así como la definición de los casos de uso fundamentales que se han analizado y los resultados de dicho análisis se recogen en el Capítulo 3.

Por último, el Capítulo 4 presenta las conclusiones y líneas futuras, donde se contrastan los resultados obtenidos y se plantean posibilidades que se podrían realizar en una visión a largo plazo con la tecnología LoRAWAN.

2. LORA Y LORAWAN

En este capítulo se va a hacer una exposición teórica de LoRa y LoRaWAN, las tecnologías que son fundamentales para llegar a comprender el proyecto y que lideran el grupo de las LPWAN, gracias a algunas de sus características, como la comunicación bidireccional, su fácil despliegue y su bajo coste.

Además, se introducirán brevemente los protocolos y marcos de desarrollo software que se han empleado en la realización de este TFG.

2.1 LoRa:

LoRa (Long Range) es una tecnología inalámbrica que define únicamente la capa física de una red LPWAN. Desarrollada por Semtech en sus inicios en la actualidad está administrada por LoRa Alliance [7]. Se diseño con el objetivo poder establecer una conexión punto a punto con grandes áreas de cobertura y bajo consumo asumiendo que las tasas binarias serían reducidas.

La frecuencia de trabajo a la que transmite LoRa varía dependiendo de la localización geográfica en la que se opera, siendo las bandas ISM, 868 MHz en Europa, 915 MHz en América y 433 MHz en Asia.

Está basada en una modulación de espectro ensanchado, que aporta una gran robustez a interferencias y que permite ajustar la tasa de transferencia de datos a la que se envían paquetes (hasta 255 bytes por transmisión) aumentando o disminuyendo el spreading factor (SF) o factor de ensanchamiento, y manteniendo el ancho de banda constante.

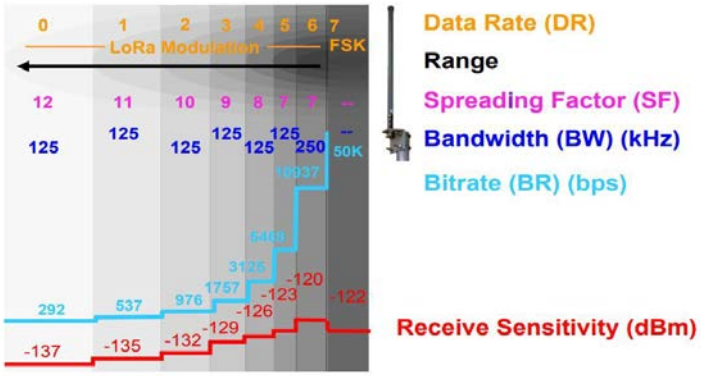

Figura 2.1 Spreading Factor sobre LoRa

El valor del spreading factor suele tomar el valor 7 por defecto, pero puede ascender hasta 12, donde se habría disminuido la velocidad de transmisión de datos para ganar en alcance y sensibilidad del receptor, tal como se muestra en la Figura 2.1.

2.2 LoraWAN:

LoRaWAN (Long Range Wide Area Network) es un protocolo de red que usa la tecnología LoRa como capa física. LoRaWAN define las capas de acceso al medio y de red para administrar y comunicar dispositivos LoRa permitiendo así establecer conexiones inalámbricas bidireccionales seguras, móviles y de bajo coste.

Este protocolo actúa como gestor de la red LPWAN, se encarga del enrutamiento de mensajes entre los componentes de la red, de que la red sea segura ante posibles ataques, de establecer velocidades de transmisión, frecuencias de trabajo, etc.

2.2.1 Pila de protocolos

En la Figura 2.2 [8] se puede observar cómo están repartidas estas tecnologías desde un punto de vista de un modelo de capas:

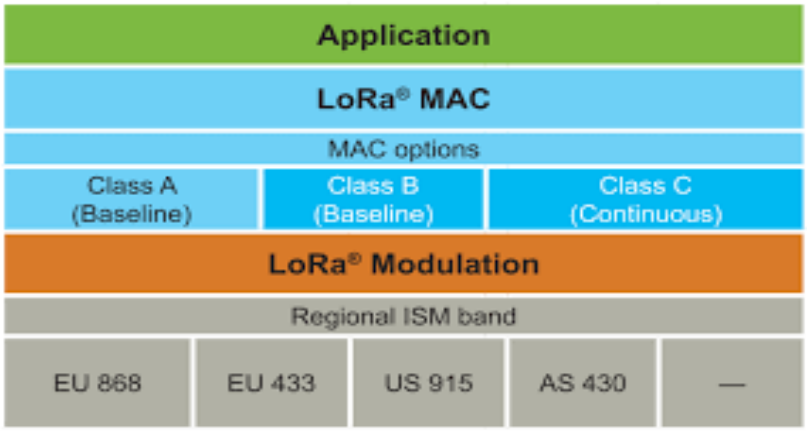

Figura 2.2 Pila de protocolos LoRaWAN

De abajo hacia arriba, LoRa ocupa la capa física operando en las bandas ISM y teniendo más peso en lo referido a las modulaciones necesarias para establecer las comunicaciones.

LoRaWAN se encuentra una capa por encima de LoRa, siendo responsable de la seguridad, el acceso a la red, gestión de velocidades de transmisión, potencia de dispositivos, enrutamiento entre gateways y nodos finales, etc.

A nivel MAC, LoRaWAN distingue tres tipos de dispositivos, clases A, B y C, que difieren en aspectos como periodos de escucha, transmisión y recepción de mensajes.

Por último, y ocupando la capa más alta, tenemos la capa de aplicación, que gestiona el modo de acceso de cada end-device.

Cabe destacar que las comunicaciones entre los nodos finales y los Gateways se realizan mediante conexiones LoRa y desde los Gateways hacia los servidores se utilizan conexiones IP.

2.2.2 Arquitectura de red

Las redes LoRaWAN clásicas, están formadas por un conjunto de nodos (típicamente sensores o actuadores de bajas capacidades), que recogen información del medio y se conectan a los gateways para transmitirles esos datos que poseen, y estos últimos reenvían los datos a los servidores de red, que, por medio de una interfaz de aplicación (API, por sus siglas en inglés), proveen a las aplicaciones de los usuarios los datos recogidos, tal como lo muestra la Figura 2.3 [9]:

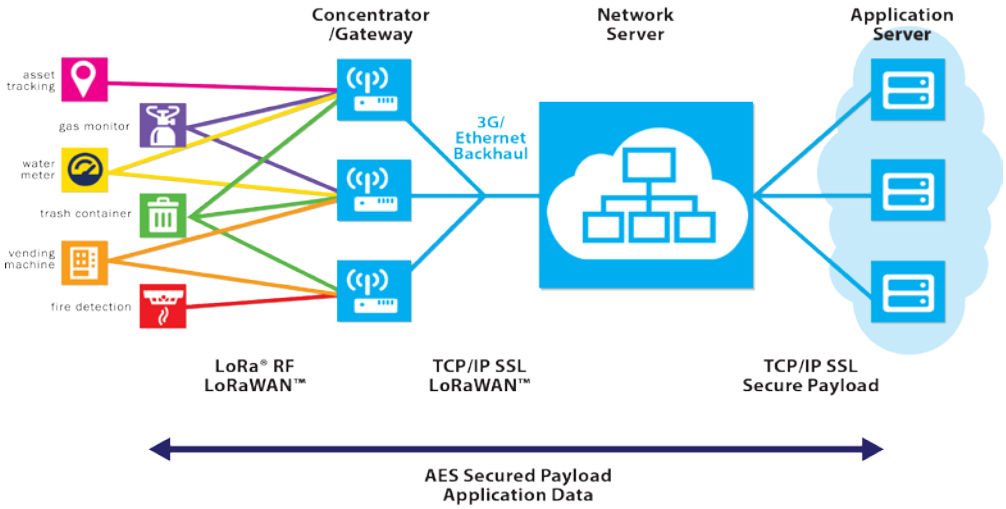

Figura 2.3 Esquema de red LoRaWAN

A continuación, se describen estos componentes de la red LoRaWAN de forma más detallada.

2.2.3 Nodos finales:

Son dispositivos hardware con una gran capacidad de detección, que pueden llegar a alcanzar una autonomía de varios años en el caso de que sean usados de forma eficiente, por ejemplo, encendiéndose únicamente en el momento en el que se vaya a hacer un envío de datos al Gateway.

Este tipo de envío al que se acaba de hacer referencia se denomina enlace ascendente o uplink; en cambio, cuando sucede al revés siendo el Gateway el que envía el mensaje al end-device, se denomina enlace descendente o downlink.

Existen 3 tipos de end-devices:

1. **Dispositivos de clase A**: Ideal para nodos que usan una batería, debido a que ofrece el menor consumo de energía. Tras enviar un mensaje uplink, abre dos ventanas de recepción, como se puede apreciar en la Figura 2.4. En ese período de escucha espera un mensaje de reconocimiento por parte del Gateway y algún posible mensaje proveniente de la aplicación.
   En el supuesto de que suceda el caso uno de la figura 2.4 [10], donde el servidor no utiliza ninguna de las dos ventanas de recepción del nodo, tendrá que esperar al siguiente mensaje de uplink por parte de éste. Si decide responder en una de las dos ventanas, no podrá utilizar la otra.
   Esta clase de dispositivos se suelen usar cuando no es habitual que estos reciban datos con mucha frecuencia.

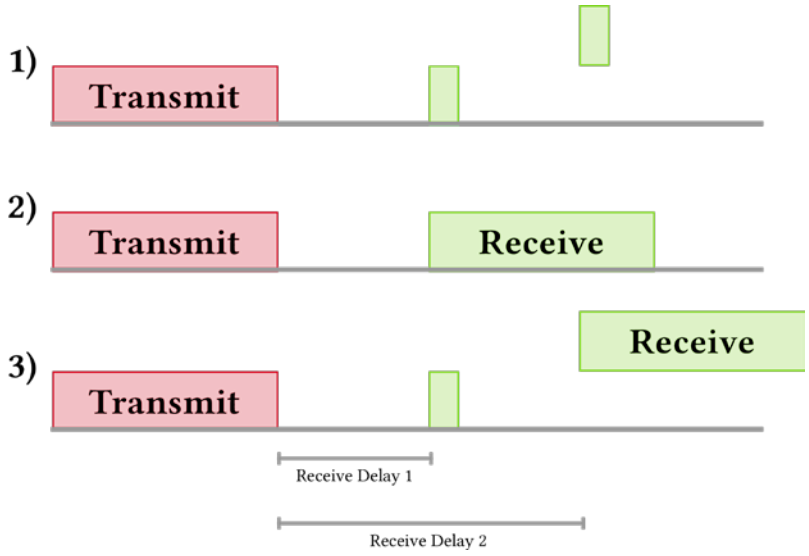

Figura 2.4 Modo de operación de nodo clase A

2. **Dispositivos de clase B:** Estos dispositivos si ofrecen la posibilidad de recibir mensajes downlink a pesar de no haber transmitido. Esto se consigue debido a que el Gateway envía beacons periódicamente, para conseguir que ambos estén sincronizados y así poder negociar los tiempos de recepción de paquetes downlink.
   Como consecuencia del envío de estos beacons, el consumo será mayor respecto a los dispositivos de clase A.

3. **Dispositivos de clase C:** Están continuamente escuchando, excepto cuando están enviando mensajes uplink, como se puede ver en la Figura 2.5 [11]. Esto

proporciona los mejores tiempos de respuesta entre nodo y servidor, a costa de un superior consumo energético. En la mayoría de los casos otra clase de dispositivos necesita una fuente de energía (continua o discontinua) para poder operar.

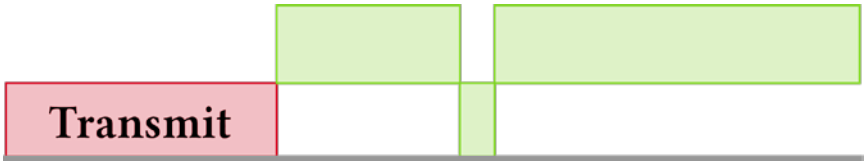

Figura 2.5 Modo de operación nodo clase C

2.2.4 Gateways:

También conocidos como puertas de enlace, módems, puntos de acceso o pasarelas. Se trata de un dispositivo que enlaza los nodos finales con el servidor reenviando los mensajes LoRaWAN que los nodos y aplicación intercambian.

Tal y como puede parecer, los Gateways no tienen un poder computacional muy potente, puesto que la parte compleja se ejecuta en el servidor. Estos reciben mensajes LoRa de los nodos finales y los convierten en mensajes IP que envían a los servidores. Sin embargo, sí deben tener una fuente de energía continua e ininterrumpida.

2.2.5 Network Server (NS)

Los servidores de red se encargan de recibir todos los mensajes procedentes de los Gateways y en él se realizan los procesos más complejos de todo el sistema LoRaWAN.

A continuación se enumerarán algunas de las funciones que tiene este servidor:

- **Enrutamiento de mensajes a la aplicación correspondiente:** Los brokers y handlers deben saber encaminar el tráfico a la aplicación que corresponda en cada caso.
- **Eliminación de mensajes duplicados**: El servidor de red puede recibir un mismo mensaje desde Gateways distintos y debe ser capaz de detectarlos y eliminar los duplicados.
- **Descifrado y encriptado de mensajes**: Con las claves de sesión correspondientes encripta o desencripta los mensajes.
- **Selección del mejor Gateway para envío de downlinks**: En función de la SNR (Signal Noise Ratio) y la RSSI (Received Signal Strength Indicator) debe decidir que Gateway tiene mejor calidad para el envío del mensaje descendente.

2.2.6 Application Server (AS)

En el servidor de aplicación se da utilidad a los datos recopilados por los nodos.

Se pueden ejecutar tanto en una nube pública como en una privada, la cual está en continua interacción con el servidor de red. Este gestiona la interfaz que da acceso al usuario a los datos ubicados en la aplicación.

2.2.7 Modos de acceder a la red LoRaWAN:

Existen dos maneras para que un nodo se registre en una red LoRaWAN:

Activation by Personalization (ABP): Es el modo más simple de conexión a la red, y los parámetros que se utilizan para ello son:

- **Network Session Key:** Clave de sesión empleada entre el servidor de red y el nodo.
- **Application Session Key:** Clave de sesión empleada para encriptar y desencriptar los mensajes.
- **DevAddress:** Dirección lógica que se utilizará durante toda la comunicación

Haciendo uso de estos parámetros, el nodo envía datos al Gateway, el cual, valida que esos datos correspondan a la sesión, y en el caso de que concuerde, se procesarían los datos, pero en caso contrario, se rechazarían.

La principal desventaja de este modo es que se pudiera extraer y clonar del dispositivo las claves de inicio de sesión, pero tiene muchas ventajas, como por ejemplo que no es necesario ningún protocolo de petición de conexión a la red para poder enviar datos, como si ocurre en el modo OTAA que se explicará a continuación.

Over-The-Air-Activation (OTAA): Es el modo más seguro de conexión a la red y utiliza los siguientes parámetros de configuración:

- **DevEUI**: Identificador que hace a cada dispositivo único.
- **AppEUI**: Identificador de aplicación único de 64 bits, utilizado para clasificar los dispositivos por la aplicación a la que pertenecen.
- **AppKey**: Clave AES de 128 bits compartida entre la red y el nodo, que se utiliza para determinar las claves de sesión.

Para comenzar el proceso de acceso a la red, el nodo envía un join request a la red con los datos de configuración y abre la ventana de recepción para poder recibir la respuesta por parte del Gateway. Cuando el Gateway recibe este join request, la envía al servidor, el cual verifica que el nodo que quiere acceder a la red y la llave de

encriptación sean válidos. En el caso de que así sea, otorga una sesión temporal y por medio del Gateway la envía nodo, que ya podría enviar datos a la red.

Como se comentó antes, la mayor ventaja que ofrece el modo de acceso OTAA es su seguridad, debido a que es muy complicado clonar el nodo al renovarse la sesión cada vez que se apaga, reinicia o simplemente se pierde la conexión.

2.2.8 Seguridad en redes LoRaWAN:

Para prevenir un posible ataque a nuestra red es fundamental dotarla de mecanismos de seguridad, y más aún cuando puede contener información sensible que debe ser protegida. Para ello, LoRaWAN aplica varias capas de cifrado utilizando el algoritmo criptográfico AES-128.

LoRaWAN es una de las pocas redes IoT que implementa encriptación de extremo a extremo y hace uso de tres tipos de claves para mantener la seguridad en su red:

- **Network Session Key:** Clave de 128 bits compartida entre el NS y el nodo para asegurar la comunicación segura entre ambos realizando tanto cifrado como autenticación a nivel de red.
- **Application Session Key:** Clave de 128 bits que garantiza la seguridad entre el nodo hasta la propia aplicación, es decir, de extremo a extremo, a nivel de aplicación. Incluso cuando se usan redes LoRaWAN públicas, esta clave permite la confidencialidad de los datos.
- **Application Key:** Clave de 128 que se utiliza con los nodos que usan el modo OTAA.

2.2.9 Pycom Lopy :

A lo largo de este proyecto, tanto para configurar nodos como gateways, se ha trabajado con el dispositivo Lopy de Pycom, puesto que es muy versátil, está disponible una extensa documentación para trabajar con ella y además está integrada en el entorno de The Things Network, haciendo más fácil trabajar con ella, debido a que hay una gran comunidad detrás resolviendo posibles problemas.

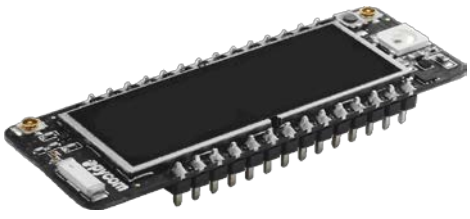

Figura 2.6 Placa Lopy de Pycom [12]

En lo que se refiere a la configuración y programación de la placa, se ha utilizado la plataforma Visual Studio Code, que es un editor que nos permite conectarnos a los puertos de la Pycom Lopy e implementar los programas pertinentes.

2.3 The Things Network y operadores LoRaWAN

A continuación, se explicará qué es y en qué consiste The Things Network y como se está utilizando para realizar despliegues de redes IoT. Además, se nombrarán algunos de los operadores LoRaWAN que actualmente están haciendo uso de esta tecnología.

2.3.1 The Things Network

The Things Network (TTN), es una red de IoT global, descentralizada, gratuita, abierta y en la que cualquier persona puede aportar sus conocimientos acerca de IoT con la finalidad de mejorar cada vez más la plataforma.

TTN fue creado en el verano de 2015 en Amsterdam , por sus dos fundadores Wienke Giezeman y Johan Stokking, entre otros y cuyo objetivo era facilitar el diseño y la creación de redes, para lo cual daba soporte a los gateways distribuidos por el mundo y además ofrecía la infraestructura de back-end a sus usuarios, es decir, una implementación de las funcionalidades del servidor de red y del servidor de aplicación.

Este backend se enfrenta a duplicidades de mensajes, gestiona la transmisión de mensajes descendentes, gestiona integraciones con plataformas, como HTTP o MQTT y ofrece una serie de APIs en diferentes lenguajes de programación como Java, Node.js o Node-Red.

Como se trata de una plataforma opensource, los códigos con los que se ha construido el backend son públicos y se puede acceder a ellos a través de un proyecto en GitHub [13].

Actualmente TTN tiene registradas alrededor de 9000 gateways distribuidos por todo el mundo y están registrados más de 80000 miembros. Teniendo en cuenta que hace tan solo un año había registradas 3000 gateways, nos podemos hacer a la idea de la evolución tan rápida que está teniendo la tecnología LoRaWAN y a su vez la red de TTN. En la Figura 2.7 [14] se puede observar la distribución de los gateways en el mundo:

Este continuo crecimiento de la comunidad permite a muchas personas acceder de manera muy sencilla y gratuita a un entorno IoT donde poder realizar proyectos o implementar sus ideas.

Figura 2.7 Distribución de los Gateways a nivel global

Además, TTN ha desarrollado una gran cantidad de comunidades en las que los usuarios, generalmente ubicados en las mismas zonas geográficas puedan intercambiar información o conocimientos que puedan ser útiles para otros usuarios y así, de manera recíproca pueden servirse de ayuda unos con otros. Por ejemplo, en España se han creado recientemente varias comunidades de TTN en Madrid, Bilbao, Barcelona, etc., lo cual indica que estamos notando el auge de LoRaWAN de forma más cercana.

2.3.2 The Things Stack para LoRaWAN

The Things Stack es una pila de red de código abierto disponible para un público global, amplio y distribuido destinada a desplegar redes tanto privadas como públicas.

Desde su creación, año tras año se han ido publicando las versiones V.0, V.1, V.2 y V.3 cuya evolución se puede ver en la Figura 2.8 [15].

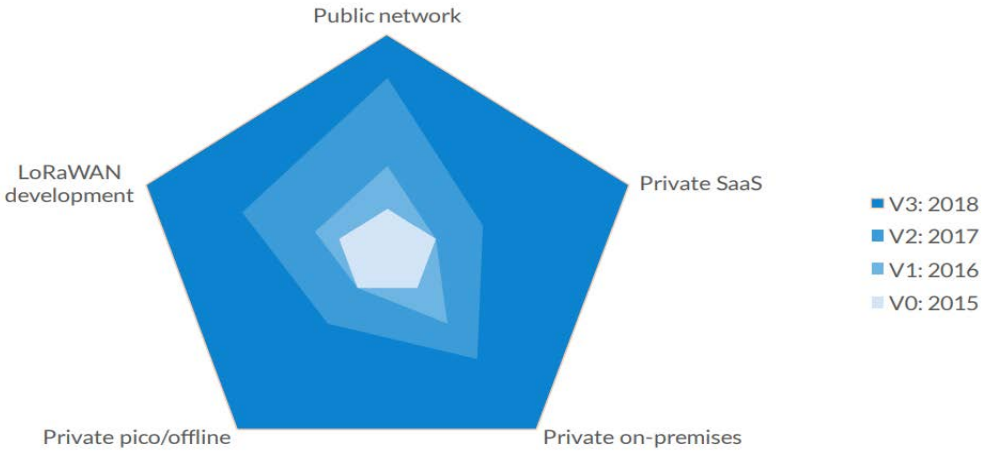

Figura 2.8 Evolución de las versiones del stack LoRaWAN

Al observar la Figura 2.8, se puede apreciar un gran desarrollo realizado desde la V.2 hasta la V.3 ya que en la V.2 destacaba por encima de las demás el ámbito de la red pública de TTN, pero ha habido un gran avance no sólo en ese aspecto, sino que todos los bloques se han optimizado dando lugar a un stack muy potente.

En el caso de la versión que se va a emplear en este proyecto, la versión 3, el stack de LoRaWAN de TTN, se instalará utilizando Docker y Docker-compose, debido a que son unas plataformas muy útiles a la hora de arrancar servidores.

Una vez configurada la parte de Docker y habiendo ejecutado el comando docker-compose up, TTN nos ofrece la aplicación TTN Console, desde donde será mucho más cómodo configurar la red, tanto registrando gateways, nodos y creando aplicaciones y además controlar el tráfico que entra o sale de cada componente de la red.

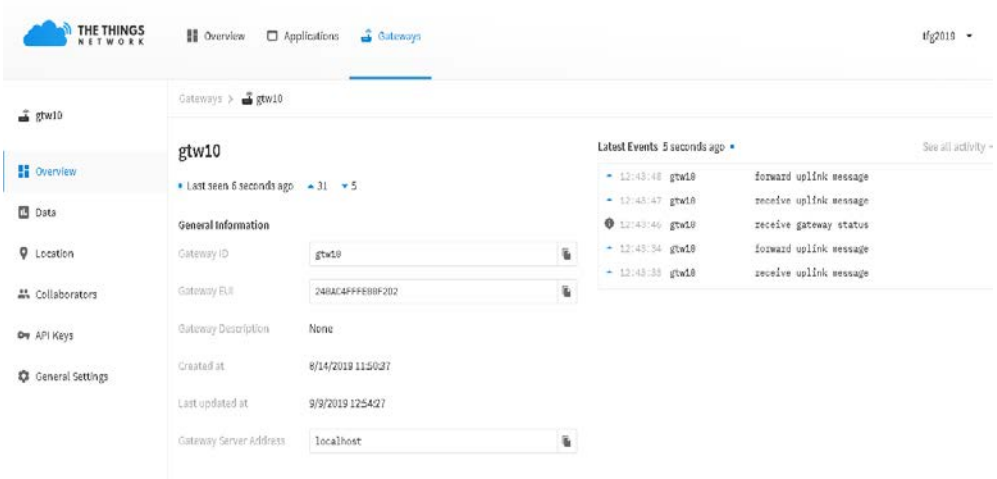

Figura 2.9 Consola de The Things Network

2.3.3 Operadores LoRaWAN

A parte de TTN, existen otros operadores de redes LoRaWAN que están desplegando sus servicios a nivel mundial.

- **Loriot**:
  Es una plataforma que provee una infraestructura para la IoT basado en la tecnología LoRaWAN, introduciendo los gateways y nodos finales de los usuarios dentro de la nube. Desde 2015, ha sido responsable de proyectos en más de 70 países.
  Un usuario que posea un Gateway propio, puede utilizar el software que ofrece Loriot para conectarlo a su nube, donde el flujo de mensajes que ocurra en ella será mostrado al usuario a través de APIs u otros servicios.
  Los clientes que generalmente hacen uso de esta plataforma son empresas pequeñas o medianas en las que su negocio se basa fundamentalmente en IoT.

- **Senet**:
  Esta plataforma posee la red LoRaWAN más grande de toda América del Norte y está destinada impulsar redes comerciales y empresariales por todo el mundo. En la Figura 2.10 [16] se observa en color verde los lugares donde Senet tiene desplegadas infraestructuras LoRaWAN, en azul oscuro las zonas donde tiene servicio e infraestructura adecuada para ofrecer cobertura a usuarios, en azul claro las zonas que próximamente estarán operativas y por último, en gris los dominios en que LoRa Alliance no tiene por ahora especificado un posible soporte de servicios.

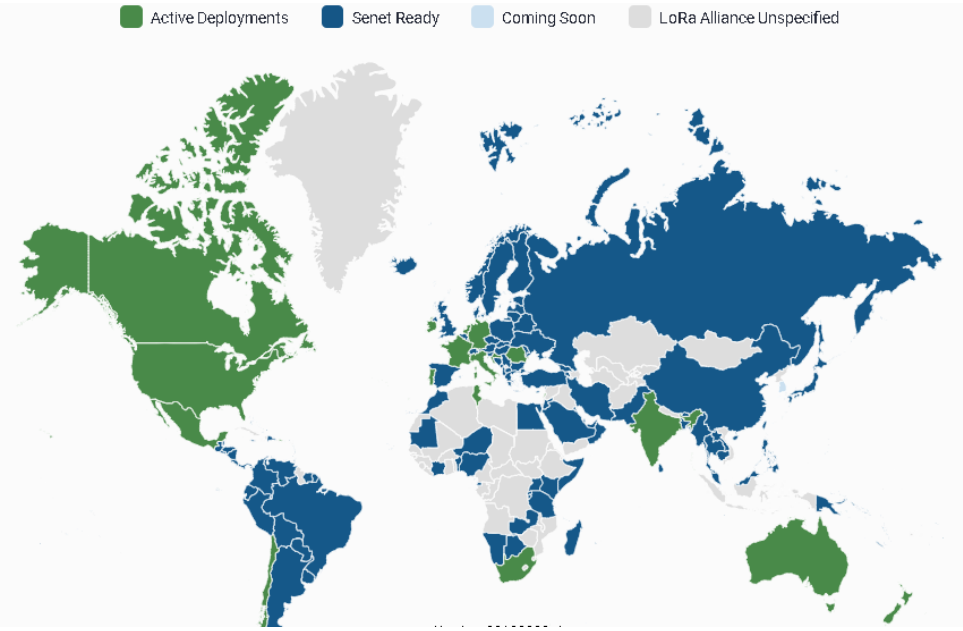

Figura 2.10 Despliegues globales de Senet

- **Redexia**:
  Se trata de la primera red nacional LoRaWAN en España dedicada a la IoT, fundada por Alex Bryszkowski y Juan Ederra recientemente, se prevé que en 3 años consigan tener una cobertura nacional.
  Esta plataforma es miembro de LoRa Alliance, la asociación sin ánimo de lucro que investiga la tecnología LoRaWAN.

2.4 Entorno de desarrollo software

A continuación, se describirán algunas de las plataformas y protocolos de transporte que se han empleado para complementar la instalación del core y posteriormente analizar los resultados:

2.4.1 Docker

Es una plataforma que empaqueta el software en unidades llamadas contenedores, que incluyen todo lo imprescindible para que el software se ejecute, incluyendo herramientas del sistema, código, librerías, etc, en cualquier máquina con Docker instalado.

Docker se instala en cada servidor y con una serie de comandos es capaz de crear, iniciar y detener contenedores. Ofrece un modelo de implementación basado en imágenes, que son plantillas que construyen y ejecutan los contenedores, y que permiten compartir un servicio o una aplicación en diferentes entornos.

Las imágenes se crean a partir de los Dockerfiles, que son ficheros de configuración en los que se describen todo el software que encapsularán los contenedores.

Esta plataforma permite introducir en un contenedor una aplicación y todos los recursos que ésta utilice, como puede ser Java, Tomcat…, facilitando así el desarrollo de la aplicación en la que se esté trabajando en cualquier entorno donde se trabaje sin necesidad de ajustar posibles versiones antiguas de programas que pueda haber entre diferentes máquinas, al estar instaladas en el contenedor.

A simple vista, Docker puede asemejarse a una máquina virtual, al poder ejecutar varios contenedores en el mismo PC, pero a diferencia de esta última, Docker no necesita instalar el sistema operativo para trabajar, al utilizar el que tiene instalado de serie la máquina donde estas funcionando y esto lo convierte en una herramienta más ligera pudiendo llevar las aplicaciones de un sitio a otro con mayor facilidad.

En la Figura 2.11 [17] podemos observar una comparación entre las dos plataformas.

Figura 2.11 Diferencia entre Docker y Máquinas virtuales

Como se puede apreciar, es necesario que las máquinas virtuales instalen todo el sistema operativo, que a través del Hypervisor, se comunica con el servidor. Sin embargo, los contenedores de Docker se procesan dentro del sistema operativo de la

máquina, teniendo como ventajas el menor tiempo de arranque del sistema y que los contenedores son mucho más manejables que las máquinas virtuales.

2.4.2 MQTT

Message Queue Telemetry Transport (MQTT), es un protocolo utilizado para la comunicación Máquina-a-Máquina (M2M, por sus siglas en inglés), en entornos de IoT, que está basado en TCP/IP durante el proceso de comunicación.

Está orientado a comunicación de sensores debido a que utiliza muy poco ancho de banda. Su arquitectura sigue una topología en estrella, llamándose "broker" el nodo central que cumple la función de un servidor, gestionando y enviando los mensajes.

Cuando un cliente quiere enviar un mensaje publica este mensaje y lo asocia a un topic o tema, al que cualquier otro cliente se puede suscribir para recoger esa información. El bróker es el encargado de filtrar y enviar la publicación de un cliente a todos los demás clientes que se hayan suscrito a ese topic.

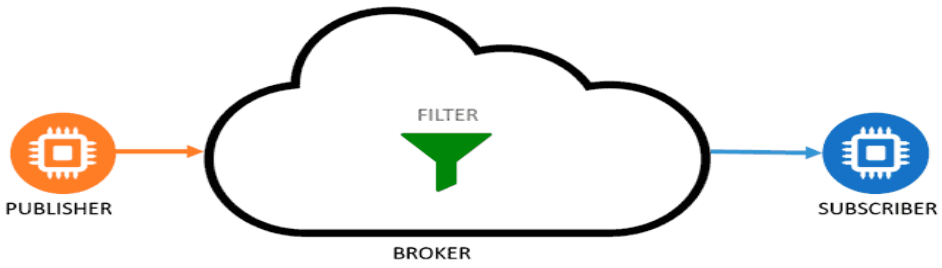

Figura 2.12 Estructura de un servicio MQTT [18]

Al iniciar un cliente una conexión TCP/IP con el bróker, esta se mantiene abierta, por defecto en el puerto 1883 o en caso de utilizar TLS en el 8883, hasta que el cliente la finaliza, enviando un mensaje de tipo DISCONNECT.

Para comenzar la comunicación, envía un mensaje de tipo CONNECT al bróker, que contiene datos propios del cliente para identificarse, y este le responde con un mensaje de tipo CONNACK, que puede aceptar o rechazar a conexión.

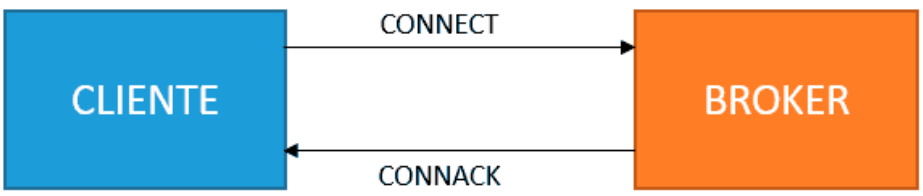

Figura 2.13 Inicio de la comunicación entre cliente y bróker [19]

En el caso de que el cliente quiera publicar un topic, enviará un mensaje de tipo PUBLISH:

Figura 2.14 El cliente envía un Publish al broker [20]

Por último, en el caso de que quiera suscribirse o desuscribirse a un topic transmitirá un mensaje de tipo SUBSCRIBE o UNSUBSCRIBE, respectivamente, a lo que el bróker responderá con un mensaje de tipo SUBACK o UNSUBACK.

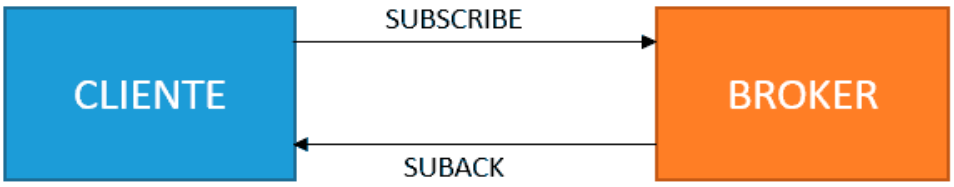

Figura 2.15 Proceso de suscripción del cliente [21]

Las ventajas que tiene el protocolo MQTT frente a otros que lo hacen apropiado para redes IoT es que es muy ligero, por lo que es óptimo para equipos que poseen baja potencia y para equipos que están continuamente en funcionamiento, ya que al utilizar menos recursos la energía que se emplea es mucho menor.

2.4.3 Datadog

Datadog es una plataforma destinada a monitorizar aplicaciones en la nube que ofrece la posibilidad de integrar servicios y obtener las métricas a partir de ellos.

En este proyecto se van a monitorizar con esta plataforma los servicios de Docker, para lo cual es necesario instalar el Datadog Agent en el host, donde puede acceder a las características de los contenedores.

El objetivo que queremos conseguir en este proyecto utilizando esta plataforma es monitorizar los archivos de log que se producen cuando hay un intercambio de mensajes entre los componentes de la red LoRaWAN o cuando se produce algún registro de un Gateway, una aplicación o un nodo y una vez los hemos obtenido, analizar el comportamiento de la red.

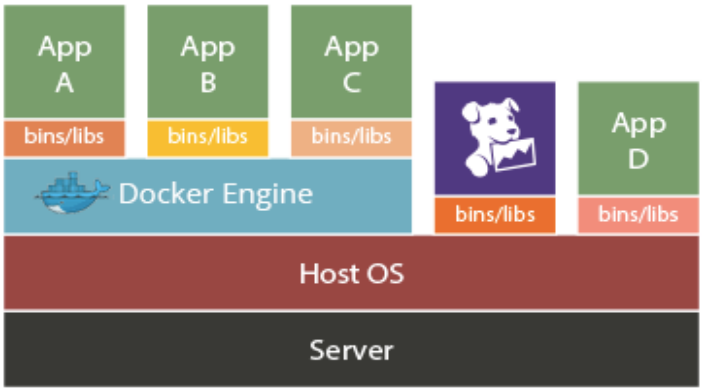

Figura 2.16 Arquitectura de Datadog [22]

Una vez hemos instalado y configurado perfectamente la plataforma Datadog para extraer todos los logs de nuestra red, es necesario hacer un filtro para visualizar solamente los mensajes que nos interesan y no perder el tiempo con mensajes sin importancia. Para ello, hacemos uso de las pipelines, que nos permiten filtrar los logs por varios atributos, como el estado, la fuente, la fecha de emisión, etc.

3. INSTALACIÓN Y ANÁLISIS

3.1 Instalación del core

Como se ha comentado en apartados anteriores, se ha realizado la instalación de un servidor de red privado de LoRaWAN desde la línea de comandos haciendo uso de The Things Stack. Para ello, se ha utilizado la plataforma Docker y Docker-compose.yml que ofrecen ventajas de gran utilidad.

Por ejemplo, una de las ventajas que nos proporciona Docker es que la instalación está almacenada en contenedores y es fácil transportarla al no ser necesario volver a hacerla en cada sistema operativo que trabajemos y sólo haría falta ejecutar el contenedor.

Otra ventaja de operar con Docker es la modularidad que nos ofrece, porque es mucho más cómodo para el desarrollador poder separar la totalidad del trabajo en partes más pequeñas y sencillas.

En el caso de la instalación del core de LoRaWAN, se dispone de los siguientes contenedores:

- **Cockroach**: Es una base de datos SQL distribuida que se utiliza en el Identity Server.
- **Redis**: Es un almacén de datos en memoria que utilizamos como base de datos para datos “activos”.
- **Docker Agent**: Es la versión de Datadog en forma de contenedor de Docker.

Para que estos contenedores se ejecuten correctamente es necesario encontrar una etiqueta reciente de la imagen de cada contenedor y actualizarlas en el fichero docker-compose.yml, al igual que con la imagen thethingsnetwork/lorawan-stack [23].

A continuación, arrancamos el Docker Compose con el comando “ttn-lw-stack start” y le indicamos que depende de Cockroach y Redis.

Para garantizar la seguridad utilizando el protocolo Transport Layer Security (TLS, por sus siglas en inglés), es necesario indicar en el archivo docker-compose.yml los certificados que están siendo utilizados. Por ello, aplicando conceptos aprendidos en

alguna asignatura de seguridad en redes se ha creado una Autoridad de Certificación (CA) autofirmada y posteriormente los certificados requeridos se han creado y firmado por dicha CA.

Para que el stack reconozca la CA que va a garantizar la seguridad de nuestro sistema es necesario añadir al final de cada comando que introduzcamos, para configurar la red desde la línea de comandos, la etiqueta *--ca "nombre del certificado de la CA"* siendo necesario que al ejecutar el comando estemos en la ubicación de la carpeta con los certificados.

En este punto, ya es posible arrancar The Things Stack, para lo cual se introduce el comando *docker-compose pull*, con el que se llama a las imágenes del docker-compose. Este comando solo de debe utilizar la primera vez que se arranca The Things Stack.

El siguiente paso es inicializar la base de datos del Identity Server y después crear un usuario junto con su contraseña para poder acceder al stack. A continuación se muestran los comandos ejecutados respectivamente para cada configuración.

```
$ "docker-compose run --rm stack is-db init")
```

```
$ "docker-compose run –rm stack is-db create-admin-user

--id admin --email your@email.com"
```

Por último, se deben registrar tanto la línea de comandos como la consola, como clientes Oauth, para poder acceder al servicio y posteriormente capaces de interactuar en la red. A continuación se indican los comandos utilizados respectivamente para cada registro.

```
$ "docker-compose run --rm stack is-db create-oauth-
client --id cli --name "Command Line Interface" --owner
admin  --no secret --redirect-uri 'local-callback' --
redirect uri 'code' "
```

```
$ "docker-compose run --rm stack is-db create-oauth-
client --id console --name "Console" --owner admin --
secret 'the secret you generated before' --redirect-uri
'https://thethings.example.com/console/oauth/callback'
--redirect uri '/console/oauth/callback' "
```

A continuación, al ejecutar el comando *"docker-compose up"*, se arrancará el stack y aparecerán los primeros logs en la terminal.

En el caso de que se quiera que el stack se mantenga activo cuando no estemos trabajando con ello, bastará con añadir la opción –d. Los logs se podrán ver posteriormente con un *"docker-compose logs".*

Una vez registradas la consola y la CLI, solo queda hacer un login para comenzar a interactuar con la red LoRaWAN (p.ej. añadir o eliminar gateways, nodos, aplicaciones, etc.)

3.2 Casos de uso fundamentales

A continuación, se expondrá una lista de funcionalidades que, tras instalar el core de LoRaWAN, se describirán los escenarios de cada uso, y posteriormente se analizará el comportamiento de la red.

Para el análisis de las funcionalidades de la red LoRaWAN se han definido una serie de casos de uso a través de los cuáles se podrá validar y evaluar dichas funcionalidades.

En los siguientes apartados se presentarán los escenarios de cada caso de uso, se describirán las acciones que se llevan a cabo en ellos y por último se analizará el comportamiento de cada elemento de la red que interviene en dicho escenario.

3.2.1 Registro de un Gateway en TTN

Dentro de una red LoRaWAN, el Gateway es el único componente al que se conectan todos los dispositivos y hace la función de puente entre los nodos finales y el core de la red LoRaWAN.
Un Gateway se puede registrar tanto por consola como por línea de comandos, pero como la consola de The Things Network es más cómoda vamos a registrarlo por esa vía.
Para ello, una vez introducidas las credenciales correspondientes para acceder a la consola se mostrará la página de *Overview*, desde la cual se seleccionará el menú de Gateways. Una vez allí, haciendo un click en la pestaña *"+Add Gateway"* se abrirá una ventana con un formulario que habrá que rellenar para registrar el nuevo Gateway.
Alli se hará click en *"Create Gateway"* tras introducir los parámetros de configuración: Gateway ID, Gateway EUI, frequency plan, etc. para finalizar el proceso de registro y se retornará a la ventana de *overview* del Gateway que acabamos de crear.
El parámetro Gateway ID, es un identificador único del Gateway que puede tomar cualquier valor. El valor que se ha otorgado en este campo es "gtw10", accesible para el usuario. En redes más extensas este campo tomará nombres más complejos.

Si el Gateway no viene con un EUI integrado, se puede usar uno propio. En nuestro se ha seleccionado la cadena de caracteres (`240AC4FFFE00F2D2`) en función de la dirección MAC del dispositivo.
El Gateway Name es un campo opcional utilizado para nombrar al Gateway de una manera alternativa. Este campo lo hemos dejado en blanco.
La Gateway Description, es una breve descripción opcional del Gateway que estamos registrando y tampoco se ha rellenado.
El campo Gateway Server Address debe contener la dirección del servidor del Gateway al que se conecta; en nuestro caso el localhost.
Por último, el Frequency Plan se corresponde como su nombre indica al plan de frecuencias en las que operará nuestro Gateway. En este caso hemos escogido la opción "Europe 868 Mhz". El campo Duty Cycle le marcaremos con el tic para que alcance los límites permitidos para el envío de mensajes, debido a que existen una serie de restricciones.

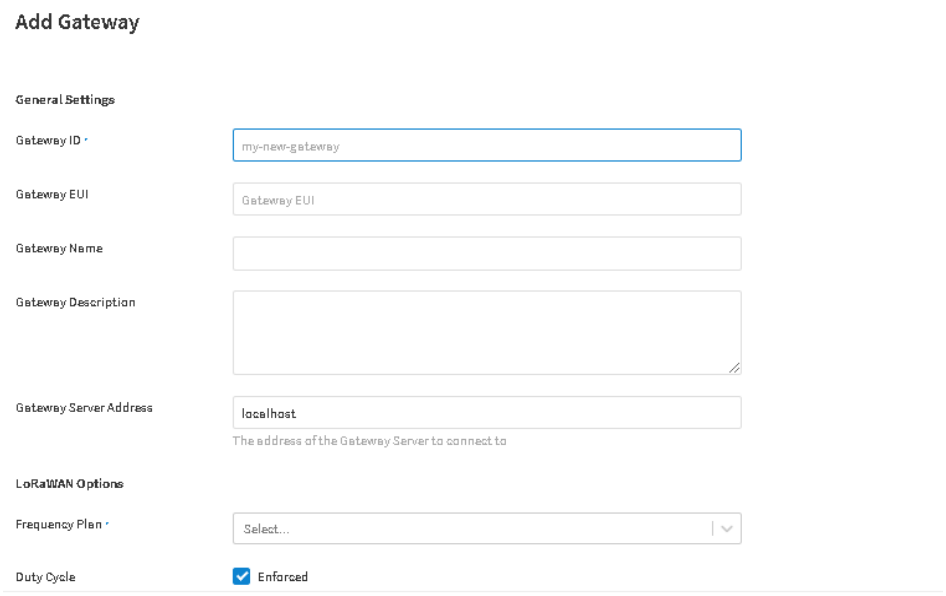

Figura 3.1 Registro de un Gateway

A partir de este momento ya se puede conectar el dispositivo físico que actuará como Gateway de la red LoRaWAN. Para ello, será necesario implementar en dicho dispositivo (en el caso de este proyecto hemos usado la Pycom Lopy) los siguientes ficheros programados en Python [24].

- **config.py**: Fichero que configura las variables constantes que se emplean durante el establecimiento de la comunicación, como pueden ser los parámetros para una conexión correcta a Internet, la sincronización del reloj usando el protocolo NTP, la dirección del servidor TTN junto con su puerto correspondiente, etc.

En la Tabla 1 se recopilan las principales variables que se encuentran en el fichero:

- WIFI_MAC: Extrae la dirección MAC del dispositivo con una función interna de código.
- GATEWAY_EUI: Valor seleccionable por el usuario, en este caso se utiliza la dirección MAC del dispositivo, introduciéndole una serie de caracteres en ella.
- SERVER: Servidor al que se conecta el Gateway. En este caso se ha utilizado el disponible en el áula de laboratorio del departamento de Telemática de la Universidad de Cantabria.
- PORT: Puerto al que se conecta el Gateway. En nuestro caso, el 31700, que estaba disponible para su uso.
- NTP: Parámetro que permite configurar los tiempos de sincronización del dispositivo con la hora oficial en España.
- NTP_PERIOD_S: Periodo del protocolo NTP establecido por defecto en 3600.
- WIFI_SSID: Identificador de WiFi del laboratorio.
- WIFI_PASS: Clave de WiFi del laboratorio.
- LORA_FREQUENCY: Frecuencia a la que opera el Gateway. En este caso tomará el valor 868100000 porque este se encuentra en Europa.
- LORA_GW_DR: Campo que indica el factor de ensanchamiento y el ancho de banda utilizado para establecer las comunicaciones. Toma valores por defecto.
- LORA_NODE_DR: Valor opcional que indica la velocidad de transmisión inicial para enviar un Join Request. Puede tomar valores de 0 a 5 en Europa. Se ha seleccionado el 5.

Tabla 1 Parámetros de configuración del Gateway

WIFI_MAC	ubinascii.hexlify(machine.unique_id.()).upper()
GATEWAY_EUI	WIFI_MAC[:6] + “FFFE” + WIFI_MAC[6:12]
SERVER	‘mu.tlmat.unican.es’
PORT	31700
NTP	“hora.roa.es”
NTP_PERIOD_S	3600
WIFI_SSID	‘TLMAT_LAB_24’
WIFI_PASS	*
LORA_FREQUENCY	868100000
LORA_GW_DR	SF7BW125
LORA_NODE_DR	5

- **nanogateway.py:** Controla toda la generación de los paquetes y los reenvíos de datos LoRa. Es inicializado por el programa main.py.

- **main.py**: Es el programa principal desde donde se hace la llamada al fichero nanogateway.py y se le pasan a este fichero los parámetros de configuración procedentes del config.py.

  Una vez descrito el escenario y procedimiento de registro de un Gateway, se va a analizar el comportamiento de la red observando el intercambio de mensajes entre cada componente apoyándonos en la plataforma Datadog.

  Situándonos en el punto anterior en el que se ha hecho un click en *"Create Gateway"* y se captura el tráfico que ocurre en la red se puede observar lo que muestra la Figura 3.2.

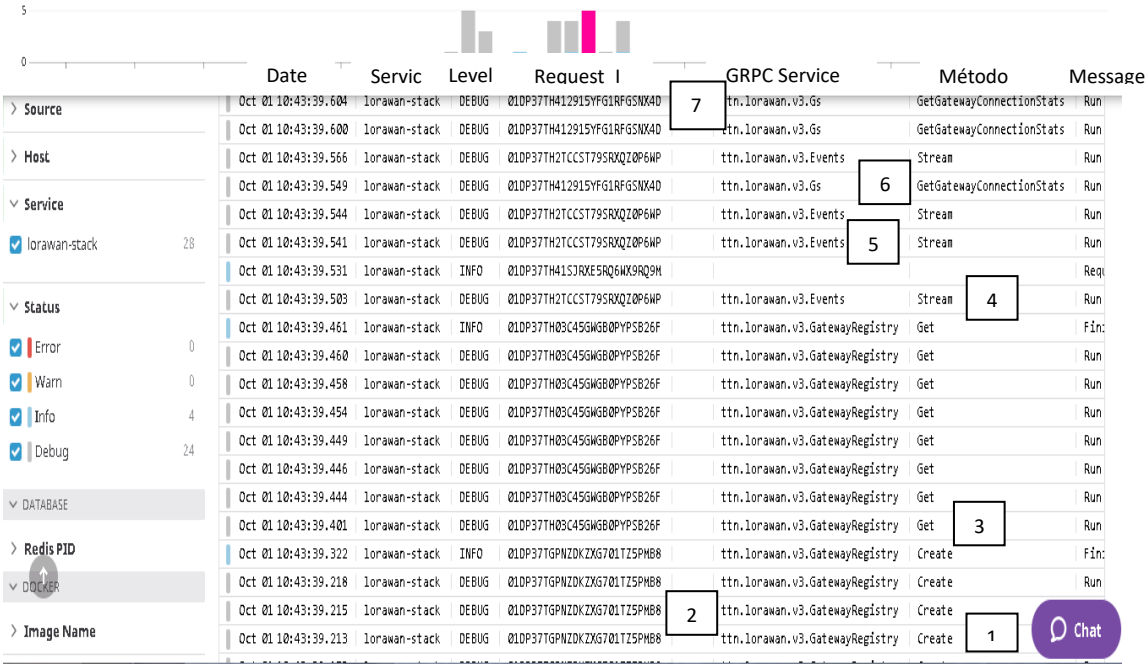

Figura 3.2 Flujo de mensajes tras crear un Gateway

El orden de llegada de los mensajes es en sentido ascendente, tal como indica la fecha temporal de llegada de los mensajes en la primera columna de datos. Las columnas las enumeraremos en orden ascendente de izquierda a derecha, habiendo por tanto siete columnas de datos.

Al observar la columna Métodos GRPC se pueden apreciar varios tipos de mensajes, los cuales están relacionados con el entorno de la base de datos del sistema LoRaWAN.

A lo largo del análisis, se van a atribuir a cada método GRPC su servicio GRPC, que proporciona características como autenticación, transmisión bidireccional y control de flujo, enlaces de bloqueo o no bloqueo, y cancelación y tiempos de espera.

Para empezar, se manda un mensaje **Create** (recuadro nº 1), cuyo servicio GRPC es "ttn.lorawan.v3.GatewayRegistry" (recuadro nº2) y su petición es seleccionar parámetros de la base de datos, ya sean de seguridad, del usuario, del cliente, etc.

Una vez que ha confirmado las credenciales de autenticación, inserta en la base de datos con etiqueta "Gateways", datos propios del Gateway que se ha creado, como pueden ser su fecha de creación, el Gateway EUI, el Gateway ID, el plan de frecuencias, etc.

A continuación, el siguiente mensaje es un **Get** (recuadro nº3), cuyo servicio GRPC sigue siendo "ttn.lorawan.v3.GatewayRegistry", y que al igual que con el mensaje create, hace una petición de parámetros para garantizar la seguridad, tanto del cliente como del usuario.

Además de eso, también pide información acerca del gateway que acabamos de crear. Para referirse a este Gateway en concreto y enviar la petición correctamente, especifica el Gateway ID, para que la base de datos extraiga la información del Gateway que nos interesa.

El siguiente mensaje es un **Stream** (recuadro nº 4), cuyo servicio GRPC en este caso es un "ttn.lorawan.v3.GatewayEvents" (recuadro nº5) y que no realiza ninguna función especial en referencia a mensajes anteriores, debido a que como los demás, realiza peticiones de parámetros de seguridad, de usuario y de cliente.

Por último, tenemos el mensaje **GetGatewayConnectionStats**, (recuadro nº 6) cuyo servicio GRPC es "ttn.lorawan.v3.Gs" (recuadro nº 7), y como su propio nombre indica nos muestra el estado el Gateway en cada momento. Por ejemplo, si no está conectado, nos muestra el siguiente mensaje: "`error: pkg/gatewayserver:not_connected`".

Como hace en todos sus mensajes, previamente a esto ha hecho peticiones de parámetros de seguridad, de usuario y de cliente para confirmar que el usuario no está siendo suplantado.

Otro caso que puede resultar relevante, es que se intente registrar un Gateway con un Gateway EUI que ya existe para otro Gateway. Entonces la base de datos del Identity Server lo detecta y nos muestra un mensaje de error, tal y como se ve en la Figura 3.3.

Figura 3.3 Error al registrar un Gateway

3.2.2 Registro de una Aplicación en TTN

En este punto, ya tenemos instalado el Gateway de nuestra red LoRaWAN y el siguiente paso para seguir operando con nuestra red es registrando una aplicación que esté comunicada con los nodos finales a través del Gateway.

Dicha aplicación nos permite crear nuestros nodos y gestionarlos en nuestros servidores. Todos los datos que envían los nodos se reciben en la aplicación por medio de una API y aquí el usuario puede acceder a ellos.

Para registrar una aplicación desde la consola de TTN, es necesario acceder a la ventana de aplicaciones desde la página de *Overview* y una vez allí, hacer click en *"+Add Application"*. A continuación, en la ventana de registro de aplicaciones será obligatorio rellenar el campo Application ID, puesto que será tomado como referencia para posteriormente hacer un link entre el Application Server y el Network Server. Al hacer click en *"Create Application"* se registrará la aplicación en nuestra red LoRaWAN.
El Application ID es un identificador de la aplicación en la red y puede tomar cualquier valor que el usuario quiera. Por ello, en una red pequeña como la que tenemos, es recomendable escoger un identificador sencillo. En redes más extensas los identificadores suelen ser más complejos. En este caso se ha escogido "app50" como Application ID.
El parámetro Application Name es un alias que se le puede atribuir de forma opcional a la aplicación y puede tomar cualquier valor. En este caso, se le ha llamado "myapp".

El campo Descripción es similar al campo que tenía el Gateway de Descripción, puesto que es opcional y sirve para describir brevemente la aplicación. En este caso no se ha rellenado.

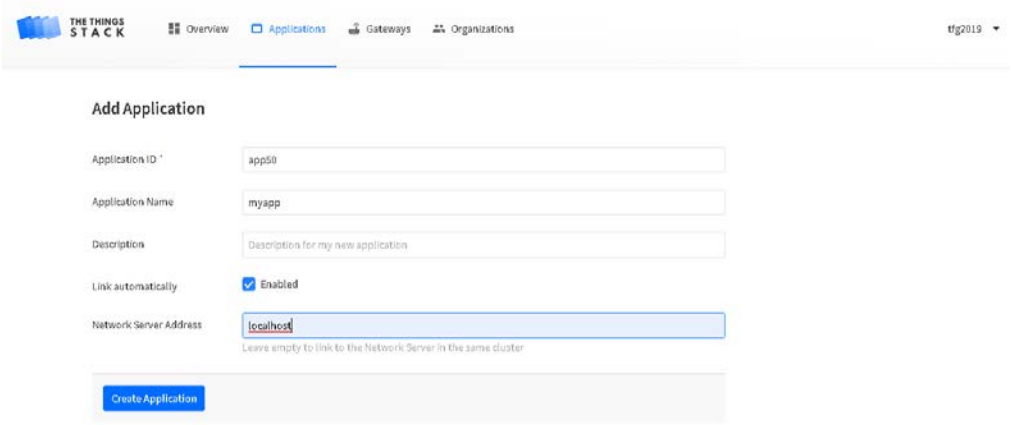

Figura 3.4 Creación de aplicación en consola

La opción Link automatically se pone como habilitada (Enabled), por defecto, para que las vinculaciones de la aplicación con elementos de la red sean automáticas.
Por último, en el campo Network Server Address se debe indicar cuál es la dirección del servidor de red al que se va a registrar nuestra application. En nuestro caso, la dirección del Network Server será localhost.
Como el objetivo es envíar y recibir mensajes hacia y desde los nodos respectivamente, es necesario vincular el servidor de aplicación con el servidor de red, para lo cual se creará un API key dentro del menú *"API keys"* haciendo click en *"+Add API Key"*.
A continuación, se le adjudica un nombre, dentro de la lista de derechos de la API Key, se le atribuye el de vincular la aplicación con el Network Server y por último haciendo click en *"Create API Key"* la creamos.
Para acabar, accediendo al menú de *"Link"* se debe enlazar la API Key que se ha creado previamente y guardar los cambios. En este momento la aplicación está en perfecto estado y expone un servidor MQTT para trabajar con eventos de streaming, es decir, se puede enviar mensajes hacia los nodos y recibir los que envían los nodos.
Para hacer uso de este servidor MQTT, es necesario crear una API key para autenticarse. Una vez creado, se hace un login con el Application ID como nombre de usuario y la nueva API key como contraseña. En el caso de que se quiera enviar un mensaje se hará uso del comando "mosquitto_pub" utilizando una serie de topics. Por ejemplo, para transmitir mensajes descendentes, se realizará un mensaje de tipo PUBLISH en el topic "v3/{application id}/ devices/{device id}/down/push". Por otro lado, si el objetivo es suscribirse, es decir, observar los mensajes depositados en un topic, se podrá conseguir haciendo uso del comando "mosquitto_sub" utilizando una lista de topics con el formato "v3/{application id}/ devices/{device id}". A continuación del device id es posible añadir "/join","/up","/down", etc.

Una vez descrito el proceso de registro de la aplicación, se va a proceder a analizar cómo se comporta la red durante el propio registro con el apoyo de la plataforma Datadog.

Como se indicó anteriormente, para crear la aplicación es necesario haber rellenado el campo Application ID entre otros, y posteriormente hacer click en *"Create application"*. Al ejecutarlo se puede observar el tráfico de datos capturado, tal y como se muestra en la Figura 3.5.

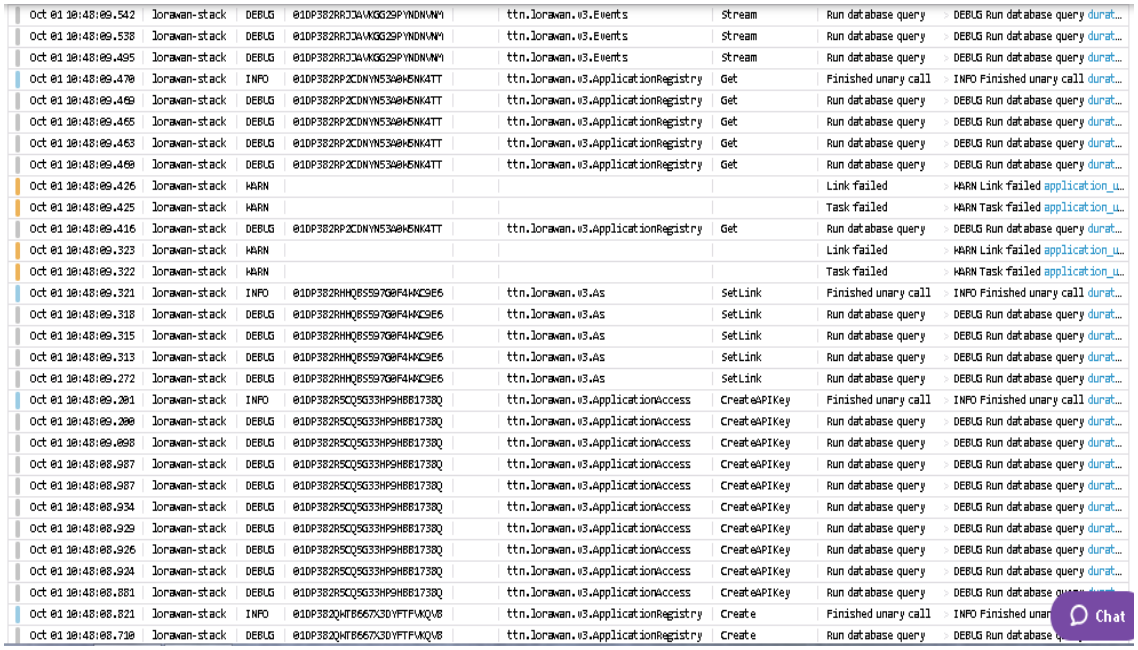

Figura 3.5 Flujo de mensajes tras crear una aplicación

Como es lógico el entorno sobre el que giran los mensajes emitidos es la base de datos de nuestra red LoRaWAN. Los 5 tipos de mensajes que aparecen son Create, CreateAPIKey, SetLink, Get y Stream.

Para empezar, se manda el mensaje **Create**, cuyo servicio GRPC es "ttn.lorawan.v3.ApplicationRegistry", y al igual que el mensaje Create que se ha visto en la creación de un Gateway, hace una petición de parámetros a la base de datos, con el objetivo de mantener la seguridad, tener información tanto del usuario como del cliente, etc.

Una vez ha confirmado que los datos son correctos, hace un INSERT, es decir, inserta en la base de datos algunos datos de la aplicación que estamos registrando, como puede ser la fecha de creación y el application ID.

A continuación, se quiere vincular el servidor de aplicación con el servidor de red para ser capaces de enviar tráfico en ambos sentidos. Por ello, se manda el mensaje

CreateAPIKey, cuyo servicio GRPC es “ttn.lorawan.v3.ApplicationAccess”, y que tras enviar una petición de parámetros para asegurar y verificar que la conexión es segura, inserta en la base de datos, concretamente en “api_keys”, los datos referentes a la API key que se está creando, como puede ser la fecha de creación, el ID del API Key, la propia clave, etc.

El siguiente mensaje es el **SetLink**, cuyo servicio GRPC es “ttn.lorawan.v3.As”, y su función es comprobar mediante peticiones de parámetros a la base de datos que el proceso de autenticación, el usuario y el cliente son correctos, mientras se produce la unión del Application Server con el Network Server.

A continuación, el mensaje **Get**, cuyo servicio GRPC es “ttn.lorawan.v3.ApplicationRegistry”, tiene como función, aparte de verificar la seguridad a nivel de usuario y cliente, comprobar que se trata de la aplicación correcta haciendo una petición del Application_ID para verificar su identidad.

Por último, el mensaje **Stream**, con servicio GRPC “ttn.lorawan.v3.Events”, simplemente tiene como objetivo mantener una seguridad verificando los procesos de autentificación, y corroborando los identificadores de usuario y cliente.

3.2.3 Registro de un nodo en modo OTAA

Hasta ahora se ha creado tanto el Gateway como la aplicación, por lo tanto, sólo queda registrar un nodo que se pueda emplear como sensor o actuador en nuestra aplicación. Sin la existencia de una aplicación, no es posible registrar un nodo, debido a que este debe pertenecer siempre a una aplicación.

En este caso, se va a proceder a registrar un nodo en modo OTAA. Como se describió en el Capítulo 2 este modo de registro nos permite aumentar la seguridad de nuestra comunicación, ya que cada vez que el nodo se conecta inicia una sesión nueva con claves distintas y es más difícil para los atacantes descifrar esas claves en tan poco tiempo.

Para registrar un nodo como OTAA desde la consola es necesario entrar en el menú de nuestra aplicación, entrar en el apartado de *Devices*, y hacer click en *“+Add Device”.* Esto nos llevará a una ventana de registro de nodos, donde se deben introducir el Device_ID, las versiones de las MAC y PHY de LoRaWAN y el plan de frecuencia que se va a utilizar.

El Device_ID es un identificador del nodo que puede tomar el valor que desee el usuario. Una vez registrado el nodo este parámetro es fijo y no es posible cambiarlo a posteriori. En nuestro caso, se le ha dado el nombre dev11. En redes más extensas este tipo de identificadores toman valores más complejos.

A continuación, el campo Device Description nos permite hacer una breve descripción del nodo que se está registrando y no es obligatorio rellenarlo.

Los campos MAC Version y PHY Version se rellenan por defecto con los valores MAC V1.0.2 y PHY V1.0.2 REV B, respectivamente.

El Frequency Plan establece el rango de frecuencias en las que el nodo va a operar. Como en este caso el dispositivo está ubicado en Europa, se rellenará el campo con Europe 863-870 MHz (TTN).

La opción de Supports Class C se mantendrá activada para que el nodo sea capaz de trabajar en modo clase C.

Por último, los campos Network Server Address y Application Server Address, deberán contener las direcciones tanto del servidor de red como del servidor de aplicación; en nuestro caso localhost.

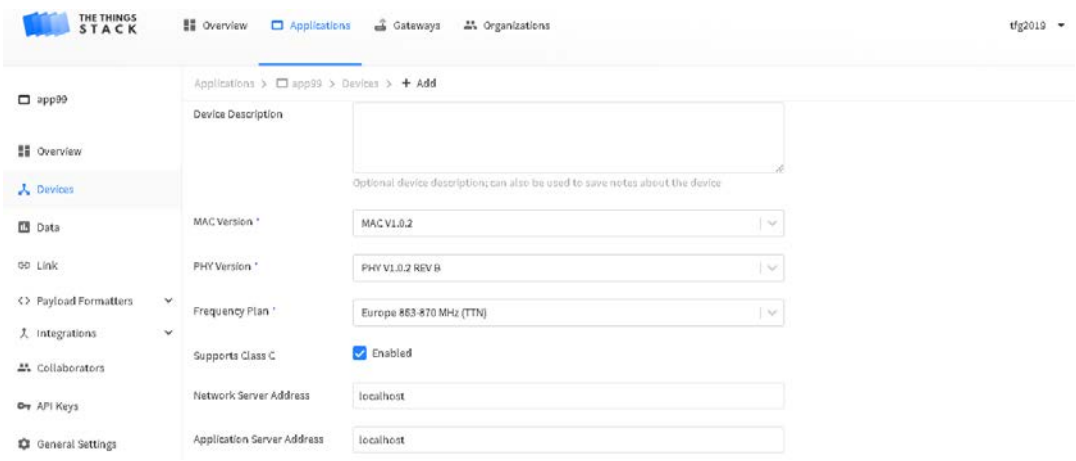

Figura 3.6 Registro de nodo en TTN

Una vez estén rellenados esos datos, nos pedirán unos datos u otros en función de si escogemos que el modo sea OTAA o ABP.

Al escoger OTAA nos pedirá el Join EUI, el Dev EUI y el Join Server Address, pero las demás claves las genera aleatoriamente. A continuación, al pulsar *"Create Device"* se crea el nodo y se abre la ventana de control del nodo creado.

El Join EUI (App EUI en versiones anteriores de LoRaWAN a la 1.1) sólo es relevante cuando se está utilizando un Join Server externo. Se trata de un ID que identifica de forma única a la aplicación capaz de procesar la trama Join Request. El usuario puede asignarle cualquier valor; en el ejemplo está asignada la cadena "12 34 56 78 90 AB CD EF".

El DevEUI es el identificador único del nodo en la red y es seleccionable por el usuario. En este caso se le ha atribuido el valor “23 23 23 23 23 23 23 23”.

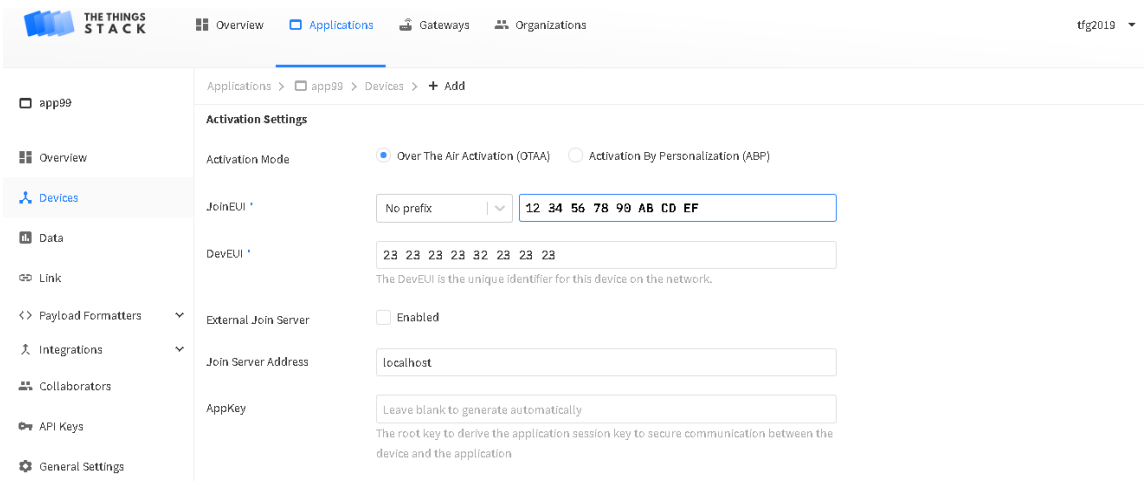

Figura 3.7 Registro de nodo en modo OTAA

Como la dirección del Join Server que se ha configurado es localhost, se debe deshabilitar la opción de External Join Server.

Por último, la App Key es la clave que genera las claves de sesión NwkSkey y AppSKey específicas para ese nodo. Con estas claves se encriptan las comunicaciones entre el nodo y la aplicación. Es posible para el usuario establecer manualmente esta clave, pero si dejamos vacío este campo se asignará automáticamente una clave aleatoria.

Para este proyecto, hemos utilizado 2 Pycom Lopy, una para configurar el Gateway y la otra para configurar los nodos. Si se quiere conectar al Stack de TTN nuestros nodos en modo OTAA se debe implementar en ellos los siguientes programas [25]:

- **Otaa_node.py:** Se trata del programa principal que incluye el código relativo a las funcionalidades principales del nodo. Cabe destacar entre éstas, los parámetros de autenticación, en este caso relativos al registro OTAA, y toda la lógica de aplicación encargada de la recepción y transmisión de mensajes desde y hacia la red.
- **Boot.py:** Programa desde donde se hace la llamada a otaa_node.py. Para cambiar de configuración de OTAA a ABP, solo haría falta cambiar la línea en la que llama a otaa_node.py por abp_node.py y hacer un update del dispositivo, es decir, actualizarlo con la nueva configuración.

En la Tabla 2 se recopilan los principales parámetros de este programa:

Tabla 2 Parámetros de configuración del nodo OTAA

Region	LoRa.EU868
DEV_EUI	AABBCCDDEEFF7778
APP_EUI	AB3589FFCC21BE45
APP_KEY	DE6D3FBD444A7E05A431AB7263DCB539

La Figura 3.8 muestra la secuencia de eventos en el log del stack TTN que se desencadena cuando, una vez se han cargado los códigos en el nodo y este se ha activado, se hace click sobre *"Create Device"*.

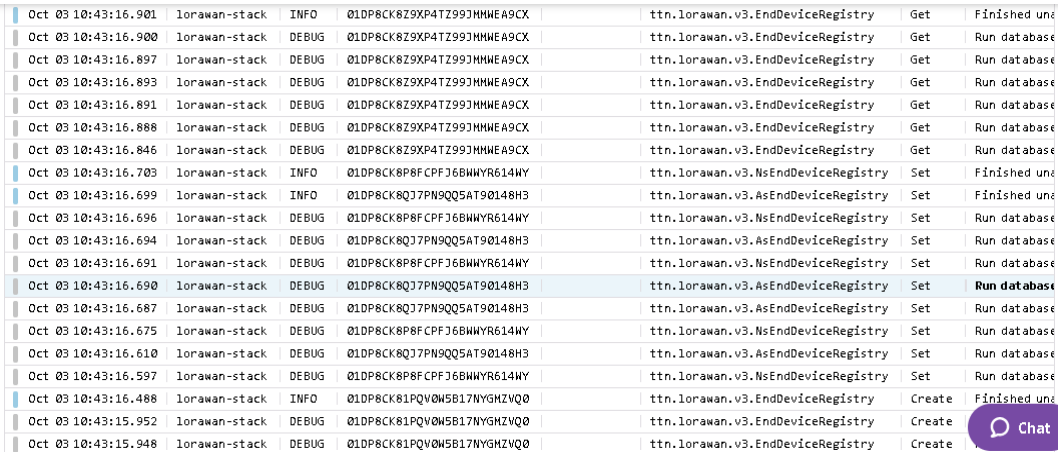

Figura 3.8 Flujo de datos tras crear el nodo OTAA

Se pueden observar 3 tipos de mensajes: Create, Set y Get.

Para empezar, el mensaje **Create**, cuyo servicio GRPC es "ttn.lorawan.v3.EndDeviceRegistry", después de ejecutar el proceso de selección de datos con el objetivo de garantizar la seguridad de la comunicación, inserta en la base de datos los parámetros propios del nodo que se está registrando, como pueden ser su device_id, su fecha de creación, la aplicación a la que está asociado, las direcciones de los servidores en los que opera, etc.

A continuación, el mensaje **Set** utiliza 2 tipos de servicios GRPC, tanto el "ttn.lorawan.v3.NsEndDeviceRegistry", como el "ttn.lorawan.v3.AsEndDeviceRegistry", refiriendo a los entornos del Network Server y del Application Server respectivamente.

Tiene como función la autenticación dentro de estas entidades y conocer los datos del usuario y del cliente que van a operar en ellas.

Por último, el mensaje **Get**, utiliza el servicio GRPC “ttn.lorawan.v3.EndDeviceRegistry” para realizar la autenticación y hacer la petición de los datos de usuario, de cliente y del nodo que se ha registrado. También utiliza los servicios “ttn.lorawan.v3.NsEndDeviceRegistry”, “ttn.lorawan.v3.AsEndDeviceRegistry” y “ttn.lorawan.v3.JsEndDeviceRegistry”; este último en referencia al entorno del Join Server, y realiza al igual que en los mensajes anteriores una autenticación y petición de datos del usuario y cliente.

En el caso del Join Server, muestra el error “device not found”, debido a que el nodo que se ha creado no ha sido programado todavía y lógicamente no ha hecho ningún intercambio de claves, por lo que el Join Server no ha intervenido aún.

3.2.4 Registro de un nodo como ABP:

En este caso, se va a registrar un nodo en modo ABP, que puede no tener la seguridad del modo OTAA, pero para entornos en los que no es necesaria tanta protección de los datos es más eficiente.
El modo de registro de un nodo ABP es idéntico que el de un OTAA hasta llegar al punto en el que se selecciona si se quiere configurar el nodo en modo ABP u OTAA.
En este caso, se selecciona el modo ABP, desplegándose así los parámetros que debemos configurar para completar el registro, como se puede ver en la Figura 3.9.

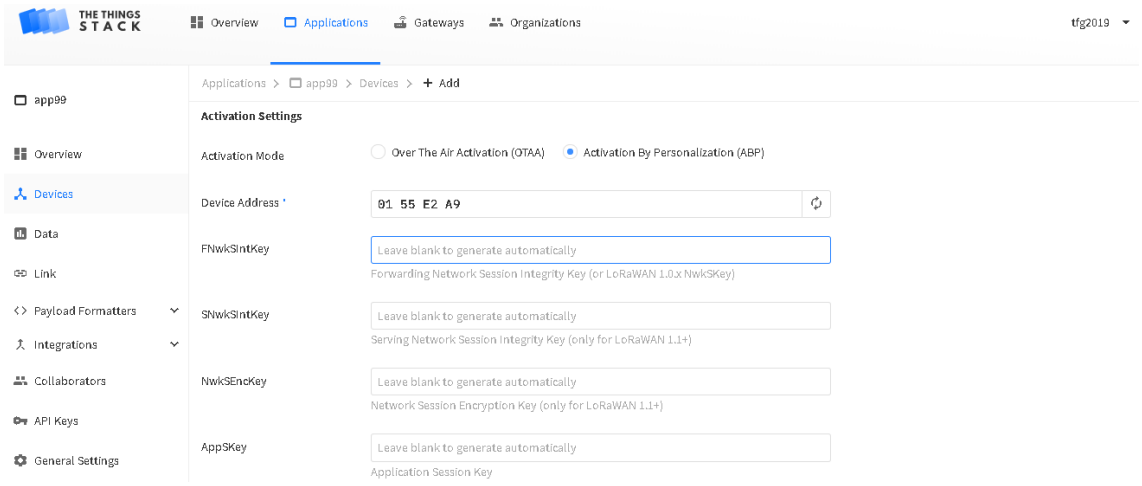

Figura 3.9 Registro de un nodo en modo ABP

Al seleccionar la opción ABP, debemos introducir solamente el Device Address, un identificador del nodo dentro de la red que puede ser asignado por el usuario (p.ej. “01 55 E2 A9”), puesto que las claves de sesión se generan automáticamente al hacer el registro, aunque también es posible rellenarlas para tener claves creadas por el

usuario. Si se elige la opción de generación automática será necesario copiar estas claves para poder utilizarlas en el código que se carga en dicho nodo.

Para finalizar, pulsamos el botón *"Create Device"* para que se complete el registro del nodo en modo ABP.

Al igual que en el modo OTAA, para conectar el dispositivo Pycom Lopy como nodo ABP al stack de TTN, es necesario configurarlo con una serie de programas [26]:

- **Abp_node.py:** Programa principal que es el equivalente al otaa_node.py del modo OTAA, debido a que realiza diversas funciones, como establecer la región en la que se utiliza el dispositivo, configurar los parámetros de autenticación (aquí deberán copiarse las claves que se hayan generado o introducido al crear el nodo en la consola de TTN), unirse a la red usando ABP, lógica de la aplicación, etc.

En la Tabla 3 se recopilan los principales parámetros de este programa:

Tabla 3 Parámetros de configuración del nodo ABP

Region	LoRa.EU868
DEV_ADDR	2601147D
NWK_SWKEY	3C74F4F40CAEA021303BC24284FCF3AF
APP_SWKEY	86404D6B24DF41B574EC44440D1E45EB

El programa Boot.py es análogo al ya descrito en la sección anterior.

A continuación, se va a proceder a analizar el flujo de datos que transcurre en la red en el momento en el que se registra el nodo en modo ABP, utilizando la herramienta Datadog.

Al igual que con el modo OTAA, al hacer click en *"Create Device"* observamos el flujo de datos que nos muestra la Figura 3.10:

Como se puede apreciar, es prácticamente idéntico al flujo de mensajes al registrar un nodo OTAA. La diferencia más notable es que los parámetros que se guardan en la base de datos cambian, porque cada tipo de modo tiene diferentes, pero todo lo referido a autenticaciones, petición de datos de usuario, de cliente, de aplicación… es similar.

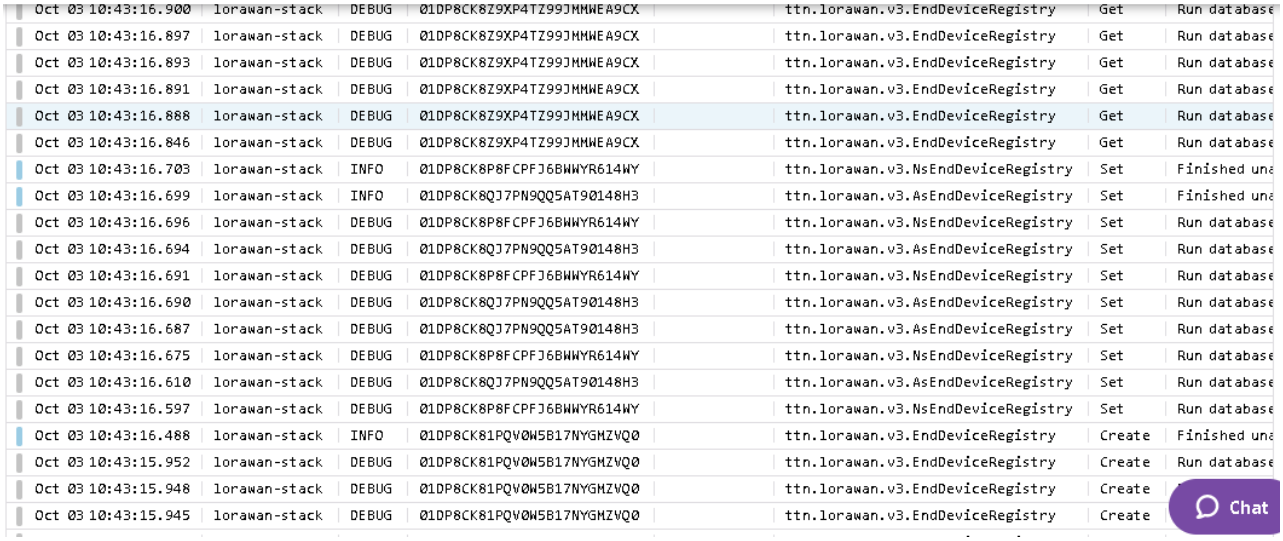

Figura 3.10 Flujo de datos tras registro de nodo ABP

3.2.5 Visualización de claves de seguridad en Identity Server

Otro objetivo del proyecto que se está llevando a cabo, es analizar cómo gestiona la red LoRaWAN las claves de seguridad que son utilizadas para que distintas entidades se puedan autenticar correctamente y hacer más complicado que posibles atacantes puedan entrar en nuestro sistema.

Para ello, debemos localizar dónde se encuentra el almacén de las claves y así, poder observarlas. La base de datos que guarda todo tipo de claves está instalada en CockRoach, y con la plataforma Dbeaver, es posible acceder a ella a través del puerto 26257 del servidor en el que está instalado el stack TTN.

Una vez se ha accedido a la base de datos en Dbeaver, es posible observar las claves que se han ido empleando a lo largo de la configuración y uso de la red LoRaWAN. A modo de ejemplo, se ha hecho una consulta de las API Keys que se utilizan al vincular el Network Server con el Application Server, y nos permite visualizar distintos atributos de cada API Key, como la clave de sesión o la entidad que la almacena junto con su identificador, véase en la Figura 3.11.

En la parte izquierda de la Figura 3.11, se pueden ver algunas de las tablas de las que se compone la base de datos, como la de access_tokens, propietaria de las claves de autenticación del usuario, o la de end-devices, que contiene atributos de cada nodo, como pueden ser su Device_ID, su fecha de creación, la aplicación a la que pertenece, etc.

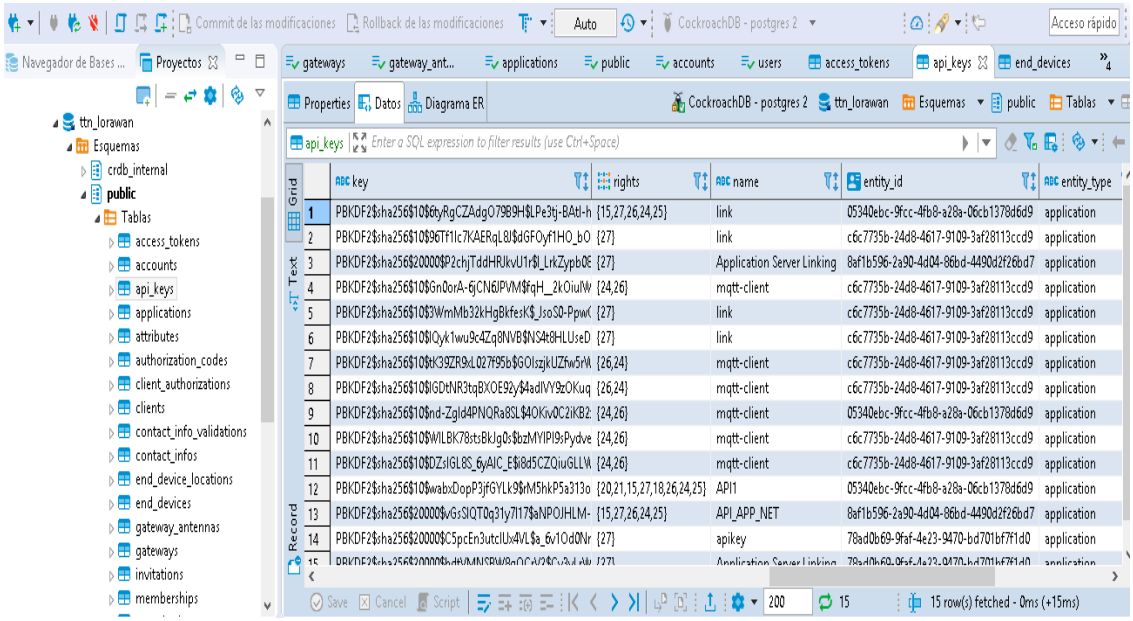

Figura 3.11 Visualización de claves en Dbeaver

3.2.6 Configuración de Pycom como nodo clase C

La característica principal de los nodos de clase C es que están escuchando en todo momento, salvo cuando están enviando mensajes ascendentes. A costa de tener un alto consumo energético, se consiguen buenos tiempos de respuesta entre nodo y servidor.

El objetivo de este caso de uso es conseguir que el Lopy de Pycom sea capaz de recibir un mensaje codificado en JSON enviado desde un cliente (se ha utilizado Mosquitto) y encienda inmediatamente un LED de la placa de expansión.

Para ello se ha desarrollado un código en Python basando la programación fundamentalmente en eventos, optimizando el uso de nuestro sistema, debido a que no es necesario utilizar un comando “WHILE TRUE”, que se mantiene constantemente activo. Con el uso de los eventos, cuando la placa recibe o envía un mensaje, automáticamente llama a una función por medio de un callback, y esta se encarga de encender el LED. En la Figura 3.12 se indica el pseudocódigo del programa que se ha desarrollado en Python. En el paso en el que se definen los atributos o claves del nodo, si la configuración es ABP, se definirán el Device_Address y las claves de sesión; en cambio, si el nodo se configura como OTAA, se definirán el dev_eui, el app_eui y la clave de sesión.

Una vez el código ha sido cargado en la placa Lopy, es posible abrir un cliente MQTT para enviar un mensaje hacia nuestro nodo. Para ello es necesario enviar un mensaje

MQTT de tipo PUBLISH utilizando como topic para ese envío el asociado a la aplicación que previamente hemos creado.

```
PSEUDOCÓDIGO

1. LLAMADA A LIBRERÍAS
2. DEFINIR FUNCIÓN DE EVENTO
3. DEFINIR ATRIBUTOS O CLAVES DE NODO EN CASO ABP U OTAA
4. ESTABLECIMIENTO DE CANALES LORA
5. ACCESO A LA RED
6. CREACIÓN DE SOCKETS
7. CALLBACK A EVENTOS EN CASO DE ENVÍO O RECEPCION DE MENSAJE
```

Figura 3.12 Pseudocódigo de nodo clase C

En la Figura 3.13 (en la esquina superior derecha) se ve tanto la ejecución del comando para el envío del mensaje: "*mosquitto_pub –h localhost –t 'v3/app10/devices/test-class-a-01/down/replace –u app10 –P '***' –m '{"downlinks":[{"f_port": 15, "frm_payload": "cHJ11ZUJh","priority": " NORMAL"}]}' "*, como (en la parte inferior) el mensaje que muestra el nodo en la terminal de desarrollo cuando recibe un mensaje.

El hecho de que tras enviar un mensaje de tipo Publish inmediatamente es recibido por el nodo, se debe a que la ventana de recepción de mensajes está permanentemente escuchando. Esto permite ver con claridad que el nodo se está comportando como modo clase C.

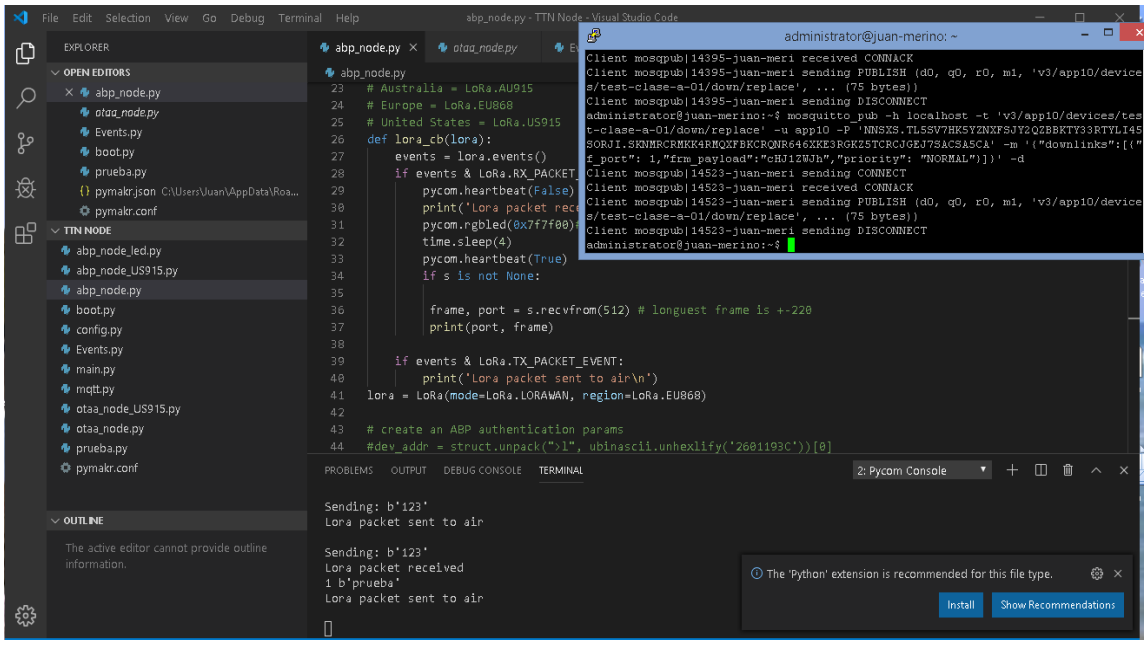

Figura 3.13 Mensaje recibido por el nodo

3.2.7 Configuración de Pycom como nodo clase A:

La característica fundamental de los nodos de clase A es su bajo consumo, que lo logran manteniéndose activos el tiempo estrictamente necesario. Por ello, solo abren dos ventanas de recepción para recibir mensajes descendentes cuando previamente hayan enviado un mensaje ascendente. En ese período de escucha espera un mensaje de reconocimiento por parte del Gateway y algún posible mensaje proveniente de la aplicación.

El objetivo planteado en este caso de uso era implementar en un nodo un programa que reciba un mensaje JSON enviado desde un cliente MQTT y encienda el LED de la placa de expansión cuando se pulse el botón de la placa de expansión. Al pulsar el botón el nodo enviará un mensaje y, por consiguiente, abrirá sendos periodos de recepción en los cuales podrán recibir los comandos que le hayan enviado.

Para ello, al igual que en el anterior apartado, se ha desarrollado un código en python haciendo uso de los eventos para optimizar el programa. Además se ha importado la librería Pin, permitiendo interactuar con los pines de la placa Pytrack siendo fundamental en este caso de uso. En la Figura 3.14 se muestra el pseudocódigo del programa implementado en el nodo:

```
PSEUDOCÓDIGO

1. LLAMADA A LIBRERÍAS
2. PIN 14 (BOTÓN) COMO ENTRADA
3. DEFINIR FUNCIÓN DE BOTÓN
  3.1 TRAS ENVIAR ABRIR RECEPCION
2. DEFINIR FUNCIÓN DE EVENTO
3. DEFINIR ATRIBUTOS O CLAVES DE NODO EN CASO ABP U OTAA
4. ESTABLECIMIENTO DE CANALES LORA
5. ACCESO A LA RED
6. CREACIÓN DE SOCKETS
7. CALLBACK A EVENTOS EN CASO DE ENVÍO O RECEPCION DE MENSAJE
```

Figura 3.14 Pseudocódigo de nodo clase A

Como se puede ver en la Figura 3.14, el pseudocódigo es muy similar al utilizado para evaluar el comportamiento de nodos clase C. La diferencia más notable es que el periodo de recepción se abre en el momento en el que se realiza el envío del mensaje, que en este caso se hace de manera forzada.

A diferencia del nodo clase C, este nodo sólo es capaz de recibir un mensaje descendente en el momento en el que finaliza la transmisión de un mensaje ascendente. En ese instante, abre su ventana de recepción para recibir posibles

mensajes externos. Hacer uso del Pin 14 para emitir un mensaje en el momento deseado ha permitido observar el proceso que se acaba de describir con mucha claridad.

4. CONCLUSIONES Y LÍNEAS FUTURAS

Con este último capítulo se finaliza la memoria, reflejando las conclusiones a las que se ha llegado y resumiendo los distintos casos de uso fundamentales que se han desarrollado en el capítulo 3. Además, se expondrán brevemente las posibles líneas futuras que se podrían llevar a cabo para darle continuidad a los resultados obtenidos.

4.1 Conclusiones

Una vez dado por terminado este proyecto se puede concluir que se ha cumplido el objetivo de este, el cual era desplegar una red LoRaWAN sobre la cual realizar el análisis de las características fundamentales de cada uno de los elementos que componen su arquitectura de red.

La instalación del stack V.3 de TTN, nos ha permitido analizar el comportamiento interno de una red LoRaWAN frente a diversas interacciones, como pueden ser registro de gateways, de aplicaciones, de nodos, intercambio de mensajes entre los componentes de la red o los procesos de autenticación para garantizar la seguridad de la red.

Tras haber finalizado la instalación del stack, se comprobó como un nodo configurado como OTAA es más seguro que uno configurado como ABP, debido a que el primero no tiene fijas sus claves de sesión. Cabe destacar que en entornos privados que no requieran excesiva privacidad utilizar un nodo ABP es más eficiente.

Además, en el ámbito de seguridad, se ha accedido a la base de datos que contiene las claves que se usan en la red y se ha analizado el comportamiento de esta en el momento que se asignan y almacenan dichas claves.

Por último, tras desarrollar la configuración de la LoPy como nodo clase C, cuya función era encender un LED de la placa de expansión inmediatamente después de recibir un mensaje enviado desde un cliente MQTT, concluimos que solo es eficiente utilizarlo en el supuesto de que esté constantemente conectado a una fuente de alimentación (continua o discontinua) debido a que tiene un alto consumo de energía. En cambio, tras implementar la configuración de la LoPy como nodo clase A, se llegó a la conclusión de que este tipo de nodo es recomendable hacer uso de él si el dispositivo dispone de una batería, porque tiene un consumo menor al no estar constantemente en periodo de escucha, ya que solo entra en este periodo al transmitir un mensaje ascendente. Por ese motivo, los nodos clase A son los más utilizados en la actualidad.

4.2 Líneas futuras

En este apartado se exponen algunas de las posibles líneas futuras que se pueden derivar tras analizar los resultados de este proyecto.

La principal línea futura que se propone llevar a cabo tras finalizar este proyecto es conseguir utilizar este trabajo para establecer un laboratorio docente dedicado a las tecnologías LoRa y LoRaWAN, debido a que hasta la fecha, sólo ha sido posible introducir a los alumnos estas tecnologías de manera estrictamente teórica.De este modo, los alumnos podrían interactuar en la red LoRaWAN y relacionar los conceptos teóricos aprendidos con la práctica.

Además de aplicar este proyecto a la docencia, sería interesante evaluar las posibles vulnerabilidades en seguridad que podría tener la red desplegada en un entorno urbano.

Otra posible línea a investigar es el análisis de la cobertura que ofrece LoRaWAN, observando la influencia de los obstáculos que puedan influir en la transmisión de la señal y obtener resultados de relaciones señal a ruido (SNR, por sus siglas en inglés) y RSSI (Received Signal Strength Indicator, por sus siglas en inglés) variando el spreading factor con el objetivo de optimizar estos resultados.

5. REFERENCIAS

[1]. https://www.hiberus.com/crecemos-contigo/introduccion-al-iot-internet-of-things/

[2]. https://www.sigfox.com/en

[3]. https://www.iotforall.com/rpma%E2%80%8A-%E2%80%8Aoverview-ingenus-lpwan-technology/

[4]. http://informatica.blogs.uoc.edu/2018/11/22/que-es-nb-iot/

[5]. https://lora-alliance.org/about-lorawan

[6]. https://cicese.repositorioinstitucional.mx/jspui/handle/1007/1274

[7]. https://lora-alliance.org/

[8]. https://medium.com/beelan/haciendo-iot-con-lora-cap%C3%ADtulo-1-qu%C3%A9-es-lora-y-lorawan-8c08d44208e8

[9]. https://www.digikey.es/es/articles/techzone/2017/jun/develop-lora-for-low-rate-long-range-iot-applications

[10] y [11]. https://www.thethingsnetwork.org/docs/lorawan/

[12]. https://docs.pycom.io/datasheets/development/lopy/

[13]. https://github.com/TheThingsNetwork/lorawan-stack

[14]. https://www.thethingsnetwork.org/community/barranquilla/post/the-things-network

[15]. https://www.youtube.com/watch?v=dvYREtWc1lA

[16]. https://www.iotworldtoday.com/2019/04/26/lorawan-pioneer-senet-expands-business-with-new-focus/

[17]. https://ualmtorres.github.io/SeminarioDockerPresentacion/

[18], [19], [20] y [21]. https://www.luisllamas.es/que-es-mqtt-su-importancia-como-protocolo-iot/

[22]. https://www.datadoghq.com/blog/monitor-docker-datadog/

[23]. https://thethingsstack.io/v3.2.0/

[24]. https://docs.pycom.io/tutorials/lora/lorawan-nano-gateway/

[25]. https://docs.pycom.io/tutorials/lora/lorawan-otaa/

[26]. https://docs.pycom.io/tutorials/lora/lorawan-abp/