UNIVERSIDAD DE VALLADOLID

ESCUELA TÉCNICA SUPERIOR

INGENIEROS DE TELECOMUNICACIÓN

TRABAJO DE FIN DE GRADO

GRADO EN TECNOLOGÍAS ESPECÍFICAS DE TELECOMUNICACIÓN,
MENCIÓN EN SISTEMAS DE TELECOMUNICACIÓN

Implementación de un radioenlace LPWAN con tecnología LoRa

Autor:
Dña. María Magdalena Valenciano López
Tutor:
D. Ramón de la Rosa Steinz

Valladolid, julio 2022

TÍTULO:	**Implementación de un radioenlace LPWAN con tecnología LoRa**
AUTOR:	**Dña. María Magdalena Valenciano López**
TUTOR:	**D. Ramón de la Rosa Steinz**
DEPARTAMENTO:	**Departamento de Teoría de la Señal y Comunicaciones e Ingeniería Telemática**

Tribunal

PRESIDENTE:	**D. Ramón de la Rosa Steinz**
VOCAL:	**D. Alonso Alonso Alonso**
SECRETARIO:	**D. Juan Carlos Aguado Manzano**
SUPLENTE 1:	**Dña. Patricia Férnandez del Reguero**
SUPLENTE 2:	**D. Ignacio de Miguel Jiménez**

FECHA:	**Julio 2022**
CALIFICACIÓN:	

Resumen

En este Trabajo de Fin de Grado se van a estudiar las principales tecnologías LPWAN existentes en la actualidad, para poder comprender mejor los apartados siguientes. El objetivo será implementar un radio enlace LPWAN empleando una puerta de enlace LoRaWAN, que permita establecer comunicaciones de largo alcance, y distintos dispositivos utilizados en la medición de temperatura y humedad. Posteriormente, se explicará de forma más detallada la tecnología que se va a emplear para el desarrollo del radio enlace, para poder comprender sus funcionalidades y también sus limitaciones. Se construirá e implementará una puerta de enlace a partir de un módulo comercial, y finalmente se realizarán distintos tipos de pruebas para la comprobación del correcto funcionamiento de la puerta de enlace desarrollada, así como la valoración de si los aspectos teóricos explicados se corresponden con lo que ocurre en la práctica. Además, se hará una valoración economía de la puerta de enlace construida.

Palabras Clave

LPWAN, LoRa, LoRaWAN, puerta de enlace.

Abstract

In this project, the main current LPWAN technologies will be studied. The main objective will be to implement an LPWAN radio link using a LoRaWAN gateway, which allows establishing long-range communications, and different devices used to measure temperature and humidity. Subsequently, the technology chosen for the development of the radio link will be explained in more detail, in order to understand its functionalities and its limitations. A gateway will be built and implemented from a commercial module, and finally, different types of tests will be carried out to verify that the developed gateway performs as expected, as well as the assessment of whether the theoretical aspects explained correspond with what happens in practice. In addition, an economic assessment of the developed gateway will be carried out.

Keywords

LPWAN, LoRa, LoRaWAN, gateway.

Índice general

Índice de figuras

Índice de tablas

1. Introducción

1.1. Motivación

En los últimos tiempos, las redes LPWAN, *Low Power Wide Area Network*, han cobrado una gran importancia, ya que estas redes se plantean como una alternativa de bajo consumo energético y bajo coste para conectar dispositivos finales en amplias áreas a las tecnologías anteriores, como *Bluetooth* o *ZigBee*. Estas tecnologías son especialmente útiles en zonas, especialmente rurales, donde el despliegue de otro tipo de redes pueda resultar complicado o muy costoso, ya sea por su gran extensión o lo abrupto del terreno

Dentro de este gran grupo, LoRa, *Long Range*, surge como una tecnología que permite comunicar dispositivos finales a grandes distancias con un consumo energético muy reducido por parte de estos dispositivos. Esto hace que esta tecnología sea ideal para el despliegue de redes empleadas para la monitorización, configuración y control de sensores en zonas remotas o de difícil acceso, lo cual se plantea como una ventaja en sectores como el agrícola.

Este trabajo surge con el fin de reducir los costes de despliegue que conllevan las redes LoRa.

1.2. Objetivos

El principal objetivo de este trabajo es el desarrollo de un *gateway* o puerta de enlace de bajo coste a partir de un módulo comercial que permita el despliegue de redes LPWA en distintos entornos, y con el que se puedan realizar comunicaciones con una baja tasa de datos, bajo consumo, largo alcance y un alto número de nodos. Además de obtener una puerta de enlace funcional, un segundo objetivo es poder incluir la herramienta *Node-Red*¸ y conseguir manejar los mensajes recibidos a través de ella.

Además, se hará una valoración de si el desarrollo merece la pena desde el punto de vista económico, comparándolo con algunas puertas de enlace comerciales.

1.3. Estructura del proyecto

Este proyecto se divide en varios apartados. Primero se definirán las redes LPWAN (*Low Power Wide Area Network*), sus características principales y algunas de las tecnologías más populares que existen actualmente dentro de este campo. A continuación, se definirá en más profundidad la modulación LoRa (*Long Range*), ya que será la tecnología que se emplee para el desarrollo de la puerta de enlace, así como sus principales características y aplicaciones, y el protocolo de comunicación LoRaWAN. Una vez se hayan definido estos conceptos, se

describirán los materiales y el software que se utilizarán para el desarrollo de la puerta de enlace. Después, se detallará el proceso de implementación del *gateway*, así como el proceso de instalación de la herramienta *Node-Red*. En el apartado de resultados, se describirán las pruebas realizadas, así como el montaje, y se realizará el cálculo del balance de enlace, lo cual ayudará a la comprensión de los resultados obtenidos. Finalmente, se discutirán estos resultados, y se realizará una valoración económica, comparando la puerta de enlace desarrollada con algunas de las alternativas comerciales más populares.

2. Revisión de tecnologías

Una red es un conjunto de sistemas digitales independientes conectados entre sí, de forma que se posibilita la transmisión de datos entre los mismos. Para que esto sea posible, se necesita que exista tanto una conexión lógica como física entre los sistemas. El objetivo principal de configurar redes es transmitir datos de un dispositivo a otro y poder disponer recursos en común, como bases de dato, servidores o impresoras. [1]

Las redes se pueden clasificar de distintas formas, por ejemplo, según la conexión física en la que se basen, de forma que tenemos redes que se conectan mediante cableado y redes que se conectan a través de tecnologías inalámbricas que, como su propio nombre indica, son redes que no requieren de cables lo cual permite que los dispositivos permanezcan conectados a la red y se puedan desplazar sin estar atados a ningún cable. [2]

Las redes también se pueden clasificar según su tamaño y alcance. Algunos de los tipos de redes más importantes son: PAN (*Personal Area Network*), LAN (*Local Area Network*), MAN (*Metropolitan Area Network*) o WAN (*Wide Area Network*). Esta clasificación se puede aplicar a las redes inalámbricas, obteniendo así: [3]

- *Wireless Personal Area Network* (WPAN) o redes de área personal: permiten establecer conexiones inalámbricas entre dispositivos en un área reducida, como pueden ser teléfonos móviles, ordenadores portátiles, etc. Algunas de las tecnologías que permiten el establecimiento de este tipo de redes son: *Bluetooth*, *ZigBee*, *RFID*.
- *Wireless Local Area Network* (WLAN) o redes de área local: permiten establecer conexiones inalámbricas dentro de un área local, como puede ser un edificio o un campus. Estas redes se pueden emplear para complementar a una red cableada ya existente para permitir que los usuarios se puedan mover libremente sin depender de cables. Una de las principales tecnologías dentro de las redes WLAN es *WiFi*.
- *Wireless Metropolitan Area Network* (WMAN) o redes de área metropolitana: permite el establecimiento de conexiones inalámbricas entre varios lugares dentro de un área metropolitana, por ejemplo, entre varios edificios dentro de una misma ciudad, de forma que se evita el coste de instalación del cableado. *WiMAX* es la tecnología más destacada dentro de este grupo.
- *Wireless Wide Area Network* (WWAN) o redes de área amplia: estas redes permiten a los usuarios establecer conexiones inalámbricas a través de redes remotas, las cuales se pueden mantener a lo largo de zonas extensas, como países o ciudades, empleando antenas en distintos lugares o sistemas satélite. Dentro de este grupo se encuentran las redes LPWAN.

2.1. Redes LPWAN

2.1.1. Introducción

Las redes LPWAN (*Low Power Wide Area Network*) son redes inalámbricas de baja potencia y consumo energético que, además, pueden cubrir grandes superficies de forma eficiente. Los dispositivos finales empleados en este tipo de redes suelen ser sensores y, gracias al bajo consumo, se pueden alimentar con baterías. Esto hace que se puedan desplegar redes que pueden dar respuesta a amplios territorios a pesar de no necesitar el despliegue de grandes infraestructuras.

Las características principales de estas redes son, como se ha mencionado anteriormente, bajo consumo energético y baja potencia, además de una amplia distancia entre los nodos y la antena de recepción de la puerta de enlace, aunque su capacidad de transmisión de datos es reducida. Podemos concluir en que se trata de redes con poco volumen de datos que pueden cubrir kilómetros de distancia, lo que se traduce en un bajo coste.

Estas redes se suelen emplear en aplicaciones de IoT (*Internet of Things*, o "internet de las cosas" en español) ya que estas no requieren un gran ancho de banda. Algunos ejemplos de este tipo de aplicaciones son: automatización de procesos industriales, seguimiento sanitario de personas o medición inteligente del consumo de energía.

Dentro de las redes LPWAN, hay distintas tecnologías que ofrecen diferentes servicios. Existen, por un lado, los sistemas de espectro licenciado y uso exclusivo, como las tecnologías NB-IoT y LTE-M. Y, por otro lado, hay tecnologías de espectro no licenciado, como son Sigfox y LoRaWAN. [4]

2.1.2. Comparación de redes LPWAN

2.1.2.1. Narrowband Internet of Things (NB-IoT)

Narrowband Internet of Things (NB-IoT) es un estándar de tecnología radio que fue desarrollado por 3GPP, diseñado para ampliar el futuro de la conectividad IoT de una manera más segura y fiable.

NB-IoT se desarrolló para dar cobertura en interiores, conectando a internet objetos cotidianos que requieren pequeñas cantidades de datos en periodos de tiempo largos, con una alta durabilidad de la batería y una alta densidad de conexión. NB-IoT utiliza la ya existente red móvil, pero limitando el ancho de banda a una única banda estrecha de 200 kHz. Utiliza la modulación OFDM (*Orthogonal Frecuency Division Multiplexing*) para comunicaciones de

downlink y SC-FDMA (*Single-Carrier Frecuency Division Multiple Access*) para comunicaciones de *uplink*.

Esta tecnología se utiliza en aplicaciones de IoT que requieren comunicaciones frecuentes pero que generan un tráfico de datos no muy alto. [5]

2.1.2.2. Sigfox

Sigfox es una tecnología que aporta soluciones al mundo de las M2M (*machine-to-machine*, en español "máquina-a-máquina") pensada para aplicaciones de IoT y las comunicaciones de baja velocidad y que requieren transmitir pocos datos. [6]

Sigfox emplea DBPSK (*Differential Binary Phase-Shift Keying*) y GFSK (*Gaussian Frecuency Shift Keying*) que permiten la comunicación empleando la banda ISM (*Industrial, Scientific and Medical*). En combinación con las modulaciones mencionadas anteriormente, utiliza una señal denominada *Ultra Narrowband* (banda ultra estrecha). Este tipo de señales dota a las comunicaciones de una alta capacidad, debido a que ocupa una gran parte del espectro, como podemos ver en la *Fig. 1*, y robustez frente a errores, ya que la potencia de la señal se concentra en una banda muy estrecha, evitando así colisiones. [7]

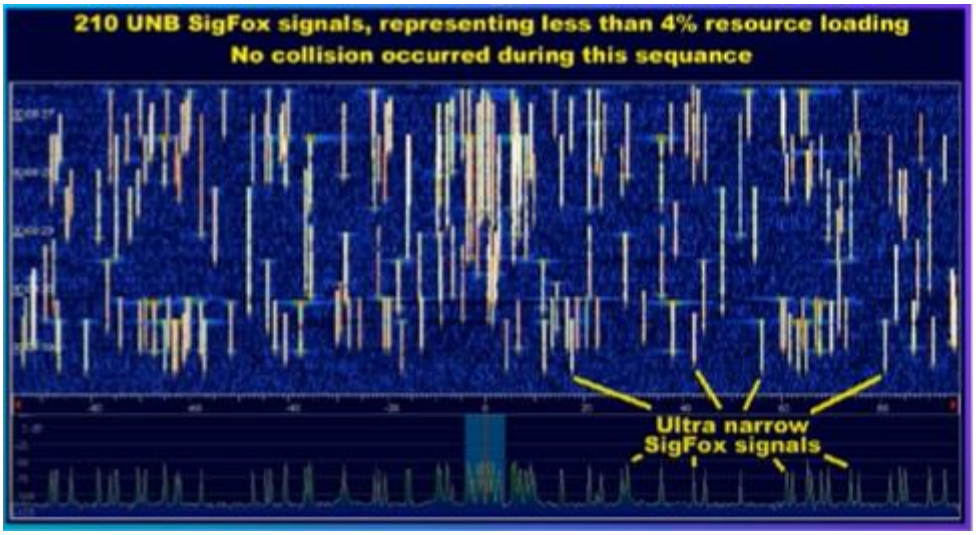

Fig. 1 Señales de banda ultra estrecha de Sigfox [7]

La red se basa en la topología de estrella en la que, como podemos ver en la *Fig. 2*, hay un nodo central que se encarga de organizar el tráfico de los dispositivos finales, pero que en el caso de Sigfox, en lugar de un nodo central, se requiere que haya un operador móvil que transporte el tráfico generado. Con esta topología los dispositivos no se conectan a una estación base específica, por lo que el mensaje transmitido lo reciben todas las estaciones base que se encuentren cerca.[7]

Esta tecnología tiene ventajas como su amplia cobertura y su accesibilidad, pero también tiene algunas desventajas, ya que solo permite enviar 140 mensajes de *uplink* y 4 mensajes de *downlink* al día. [8]

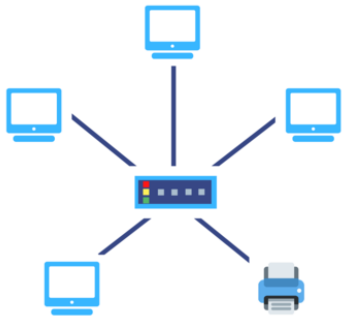

Fig. 2 Red de estrella [9]

2.1.2.3. Long Term Evolution - Machine (LTE-M)

LTE-M es el término simplificado que se usa para referirse al estándar LPWA LTE-MTC (*Long-Term Evolution Machine Type Communication*) desarrollado por 3GPP, concretamente LTE CatM1, para aplicaciones *machine-to-machine* (M2M) y de IoT.

LTE-M se caracteriza por permitir el envió de voz, movilidad, dar mayor cobertura, y disponer de un mayor ancho de banda, lo cual permite utilizar mayores tasas de datos, tener una latencia menor y un posicionamiento del dispositivo más preciso.

Este protocolo se utiliza para conectar dispositivos sencillos con un bajo consumo de energía que necesitan transmitir pocos datos en largos intervalos de tiempo. [10]

2.1.2.4. Long Range (LoRa)

LoRa (*Long Range*, en español “largo alcance”) es la capa física o la modulación inalámbrica empleada para crear enlaces de comunicación de largo alcance. Muchos sistemas inalámbricos “heredados” usan modulación FSK, del inglés *Frecuency Shift Keying*, o modulación por desplazamiento de frecuencia como capa física ya que es una modulación muy eficiente para conseguir bajo consumo. LoRa está basado en la modulación CSS, *Chirp Spread Spectrum*, la cual mantiene el bajo consumo de la modulación FSK, pero aumenta de forma significativa el rango de comunicación. La modulación CSS se ha usado en comunicaciones militares y espaciales durante décadas debido a las largas distancias de comunicación que se

pueden conseguir y la robustez ante las interferencias, pero LoRa es la primera implementación *low cost* de uso comercial. [11]

Esta tecnología fue desarrollada por *Cycleo*, una empresa francesa, que después fue adquirida por Semtech, empresa líder en el mercado de los semiconductores y miembro fundador de *LoRa Alliance*, asociación creada para promover el uso global del estándar LoRaWAN. [12]

2.2. LoRa

2.2.1. Introducción

Como se ha mencionado en el apartado anterior, LoRa es la capa física o la modulación inalámbrica empleada para crear enlaces de comunicación de largo alcance. Algunas de las principales características de LoRa son su largo alcance, el bajo consumo, alta tolerancia a las interferencias, alta sensibilidad para recibir datos, frecuencias de trabajo en la banda ISM, basado en la modulación CSS.

En los siguientes apartados se van a explicar de forma más detenida algunas de las principales características.

2.2.2. Modulación LoRa

LoRa emplea una modulación de espectro ensanchado similar y derivada de la modulación CSS (*Chirp Spread Spectrum*). La modulación de espectro ensanchado de LoRa se realiza representando cada bit de información de la carga útil (*payload*) con varios *chirps*, *Fig.* 3, que denomina a un barrido lineal de frecuencia durante el tiempo de símbolo, de información. La tasa a la que se envía la información ensanchada se denomina tasa de símbolo, la relación entre la tasa de símbolo nominal y la tasa de *chirp* se denomina factor de ensanchado (SF, del inglés *Spreading Factor*) y representa el número de símbolos enviados por cada bit de información.

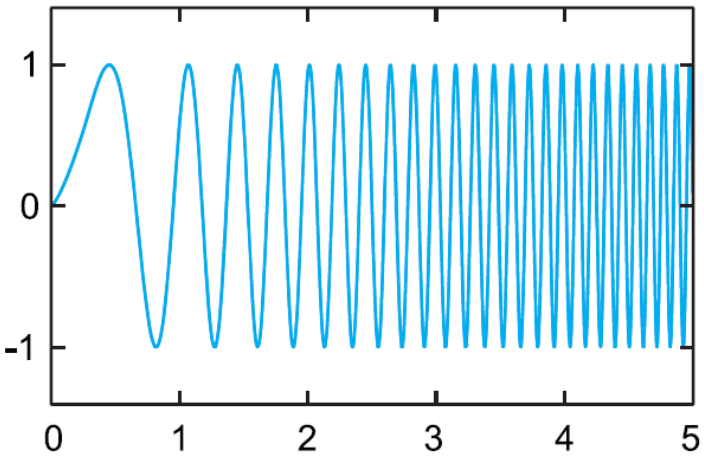

Fig. 3 LoRa Chirp Spread Spectrum [15]

Al emplear esta modulación, se puede cambiar tasa de datos por sensibilidad con un ancho de banda fijo seleccionando el factor de ensanchado (un parámetro de radio seleccionable entre 6 y 12). [13] Un SF bajo significa que se envían más *chirps* por segundo, de modo que se pueden codificar más datos por segundo. Un SF más alto implica que haya menos *chirps* por segundo por lo que hay menos datos que codificar por segundo. Si lo comparamos con SF más bajos, si enviamos la misma cantidad de datos con un SF mayor, se necesitará un mayor tiempo de transmisión, lo cual se traduce en un mayor consumo de energía. El beneficio de usar un factor de ensanchado más alto es que al haber un tiempo de transmisión más alto, le da al receptor más oportunidades para poder muestrear la potencia de la señal, con lo que se obtiene una mayor sensibilidad. [14]

El rango de alcance de LoRa puede alcanzar los 5 kilómetros en áreas urbanas, y 15 (o más) kilómetros en zonas más rurales. Además, LoRa utiliza la codificación FEC (*Forward Error Correction*) para mejorar la resiliencia ante las interferencias. [15]

2.2.3. Frecuencias utilizadas

LoRa utiliza bandas de frecuencias sin licencia por debajo de 1 gigahercio, principalmente las de la banda ISM (*Industrial, Scientific and Medical*), reservada originalmente por la UIT (Unión Internacional de Telecomunicaciones) para aplicaciones industriales, científicas y médicas. Esto significa que cualquiera puede transmitir en esas frecuencias sin necesidad de licencia, siempre que se respeten las regulaciones impuestas. [12]

Las bandas más comunes son 433 MHz, 868 MHz y 915 MHz, aunque las bandas empleadas dependen del país o la región. En la *Tabla 1* se muestran las bandas existentes, las frecuencias comprendidas por cada una, y algunas de las regiones en las que se emplea cada una.

Bandas de Frecuencia	Frecuencias comprendidas	Región(es)
EU968	863 - 870 MHz	Europa
US928	902 - 928 MHz	Norteamérica, Suramérica
CN779	779 - 787 MHz	China
EU433	433,05 – 434,79 MHz	Europa
AU915	915 - 928 MHz	Sudeste de Asia
CN470	470 - 510 MHz	China
AS923	915 - 928 MHz	Japón (AS1), Laos (AS2), Cuba (AS3)
KR920	920 - 923 MHz	Corea del Sur
IN865	865 - 867 MHz	India
RU864	864 - 870 MHz	Rusia

Tabla 1. Principales frecuencias empleadas en LoRa [16], [17]

2.2.4. Estudio de parámetros asociados

2.2.4.1. Factor de ensanchado

Como se ha mencionado anteriormente, el factor de ensanchado (SF, *Spreading Factor*) es la cantidad de código de ensanchamiento aplicado a la señal. El factor de ensanchado controla la tasa de *chirp*, y por lo tanto la velocidad de transmisión. Un factor de ensanchado bajo conlleva unos *chirps* más rápidos, y por lo tanto tasas de transmisión mayores. También se puede emplear el factor de ensanchado para controlar la congestión. Los factores de ensanchado son ortogonales entre ellos, por lo que, si dos señales se modulan con factores de ensanchado distintos, aunque se transmitan a la misma frecuencia y al mismo tiempo, las señales no interferirán entre ellas. [13]

Dentro de la modulación LoRa hay seis factores de ensanchado, desde SF7 hasta SF12. Los distintos factores de ensanchado afectan de diferentes formas a distintas características de la señal. Por ejemplo, si se emplea un factor de ensanchado más alto la tasa de transmisión será más baja que si se utiliza un factor de ensanchado bajo. Por el contrario, si se usan factores de ensanchado mayores, la ganancia de procesamiento también será mayor, por lo que la señal se recibirá con menos errores y podrá recorrer distancias mayores que si se emplearan factores de ensanchado más bajos. [14]

2.2.4.2. Tasa de codificación

La tasa de codificación (o tasa de información) es la proporción de información no redundante del flujo de datos. El valor de este parámetro debe elegirse entre 4/5, 4/6, 4/7 y 4/8, esto significa que, si tenemos una tasa 4/6, de los 6 bits totales, solo 4 van a ser útiles, y el resto

serán redundantes. [18] Cuanto menor sea la tasa, mayor será el tiempo en el aire y se tardará más en transmitir un paquete. Por una parte, esto hace más fácil la recepción de los datos ya que, al estar los símbolos más tiempo en el aire, el receptor puede demodular la información con menos potencia, por lo que el receptor tendrá una mejor sensibilidad. Pero, por otro lado, esto tiene un impacto negativo en la duración de la batería.

2.2.4.3. SNR

La relación señal-a-ruido (SNR, *Signal-to-Noise Ratio*) es la relación entre la potencia de la señal transmitida y el nivel de la potencia del ruido de fondo que corrompe a la señal, medido en decibelios.

El ruido de fondo es el área donde se encuentran todas las señales interferentes no deseadas que pueden corromper la señal transmitida. Si la SNR es mayor que 0, la señal recibida se encuentra por encima del ruido de fondo; si, por el contrario, la SNR es menor que 0, la señal recibida se encuentra por debajo del ruido de fondo.

LoRa puede trabajar con señales que se encuentren por debajo del ruido de fondo, aunque normalmente, este ruido de fondo es el límite físico de la sensibilidad. [19]

2.2.4.4. RSSI

El indicador de fuerza de la señal recibida o RSSI (del inglés *Received Signal Strength Indicator*) es una medida de la potencia de la señal de radio recibida.

El RSSI se mide en dBm y es un valor negativo. El valor de este parámetro indica que, cuanto más cercano se encuentre de 0, mejor será la señal recibida, es decir, más fuerte. [19]

2.2.4.5. Balance de enlace

El balance de enlace (*Link Budget*) es la suma de todas las ganancias y pérdidas desde el transmisor a través del espacio libre hasta el receptor. Esto es una forma de cuantificar el rendimiento del enlace y la calidad de la transmisión. [19]

Este balance se puede calcular de distintas formas, pero en este caso vamos a utilizar una fórmula simplificada, *Fig.4*, en la cual se tendrá en cuenta la potencia del transmisor - P_t, la ganancia de la antena transmisora - G_{ta}, las pérdidas de alimentación del transmisor - L_{tf}, las pérdidas del trayecto - L_{path}, la ganancia de la antena receptora - G_{ra}, y las pérdidas de alimentación del receptor - L_{rf}.

$$P_r(\text{dBm}) = P_t(\text{dBm}) + G_{ta}(\text{dB}) - L_{tf}(\text{dB}) - L_{path} + G_{ra}(\text{dB}) - L_{rf}(\text{dB})$$

Fig. 4 Fórmula simplificada para el cálculo del balance [20]

Todos estos datos se pueden saber o estimar a partir de las especificaciones de los distintos componentes, excepto las pérdidas del trayecto, para lo que se necesita emplear la fórmula que aparece en la *Fig. 5*, en la que se tiene en cuenta la frecuencia empleada, en megahercios, y la distancia, en kilómetros.

$$\text{Loss (dB)} = 32.45 + 20 \log_{10}(\text{frequency in MHz}) + 20 \log_{10}(\text{distance in km})$$

Fig. 5 Fórmula para el cálculo de las pérdidas del trayecto [20]

Estas fórmulas se emplearán en el apartado de resultados para ayudar a comprender los mismos.

2.2.5. Aplicaciones

LoRa puede ser utilizado en diversas aplicaciones gracias a su largo alcance, a su bajo coste y a su eficiencia. Entre otras, LoRa permite monitorizar, controlar y configurar sensores de forma remota y *low-cost*, lo cual se puede usar en aplicaciones más "complejas" como pueden ser la automatización de sistemas, gestión y facturación de consumo etc.

Estas aplicaciones se pueden emplear en distintas áreas en las que tienen distintas utilidades [21]:

- Agricultura: en el sector agrario es en uno de los que más se han notado las ventajas que conlleva la implementación de redes LoRa, ya que esto ha permitido reducir significativamente los costes derivados de la instalación de dispositivos, así como del mantenimiento. Además, la cobertura de LoRa puede alcanzar hasta 15 kilómetros, por lo que permite cubrir extensas áreas.
  El sector agrario es muy amplio, por lo que dentro de este hay distintas aplicaciones para cubrir las distintas y diversas necesidades: automatización de riegos y sistemas de ventilación, monitorización de la temperatura, humedad, pH etc. mediante sensores…

- Monitorización de agua y gas: al introducir la tecnología LoRa en este sector se ha maximizado la eficiencia a la par que, minimizado los costes, ya que, aunque previamente ya se tenían sensores capaces de ello, no se disponía de una tecnología de bajo coste para la comunicación con los mismos.

Algunos de los usos de la tecnología LoRa en este sector son: detección de fugas, monitorización de pozos, tuberías, riegos, etc.

- Edificios inteligentes: los edificios inteligentes cuentan con sensores que transmiten la información captada a una central, en la que se procesa dicha información, para después actuar enviando las órdenes oportunas a los distintos elementos para satisfacer las distintas demandas de mantenimiento, seguridad, confort, etc.
  En este campo, LoRa no se puede emplear en todas las aplicaciones necesarias, pero sí en algunas de ellas como control de iluminación y temperaturas, control de sistemas de aire acondicionados, monitorización de consumos, etc.

2.3. LoRaWAN

2.3.1. Introducción

LoRaWAN define el protocolo de comunicación y la arquitectura del sistema para la red mientras que la capa física de LoRa permite enlaces de comunicación de largo alcance. El protocolo y la arquitectura de la red tienen mayor influencia en la determinación de la vida útil de la batería del nodo, la capacidad de la red, la capacidad del servicio, la seguridad y la variedad de aplicaciones ofrecidas por la red. [11]

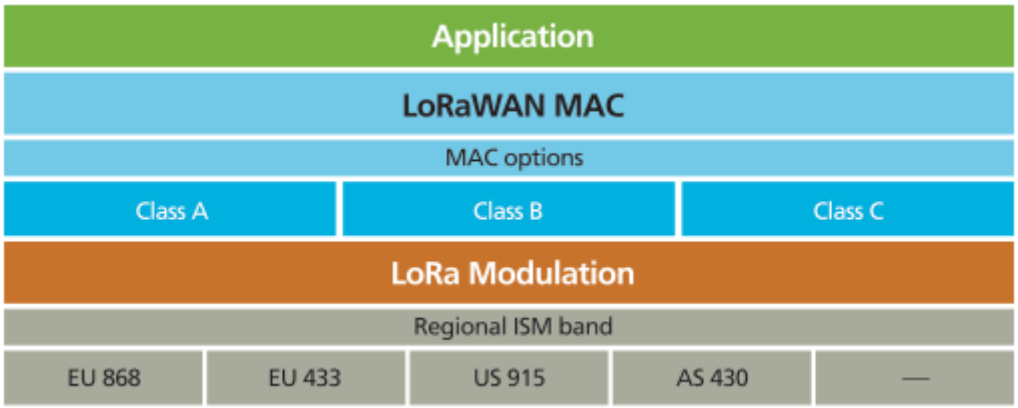

Fig. 6 Arquitectura LoRaWAN [11]

2.3.2. Arquitectura de la red

Actualmente, muchas de las redes existentes utilizan una arquitectura de red en malla, *Fig. 7*. En este tipo de redes, los nodos finales reenvían la información que reciben de otros nodos para incrementar el alcance de la comunicación y el tamaño de la celda de la red. Aunque es cierto que esto incrementa el alcance de la red, también hace que aumente la complejidad, se reduzca la capacidad de la red y la duración de la batería, ya que los nodos reciben y reenvían

información de otros nodos que, en muchos casos, les es irrelevante. Para preservar la duración de la batería de los dispositivos finales, se emplea una arquitectura de red en estrella de largo alcance.

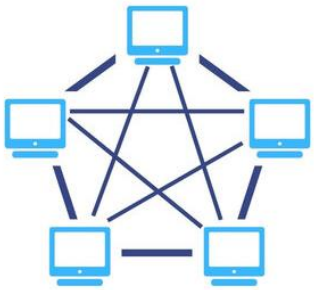

Fig. 7 Red en malla [9]

Las redes LoRaWAN utilizan el método *ALOHA* para la comunicación entre los dispositivos finales y sus servidores de red asociados. El método *ALOHA* es una posible solución al problema del medio compartido dentro de un sistema de telecomunicaciones, y su funcionamiento es el siguiente: cuando un dispositivo final tiene datos que transmitir, los transmite y espera a recibir el asentimiento o *ACK*, si no lo recibe tras un tiempo determinado, retransmitirá los datos. [22] Al usar este método, los dispositivos finales envían datos a través del *gateway* solo cuando uno o varios de sus sensores detectan un cambio, o bien si ocurre algún evento, como el fin de un temporizador. Cuando el dispositivo termina de enviar su *uplink*, se queda "escuchando" al servidor por si este envía algún mensaje uno o dos segundos después del *uplink*, y después vuelve a modo espera.

Los dispositivos finales pasan la mayor parte del tiempo inactivos. Durante este tiempo, consumen muy poca energía. Esta forma de ahorro de batería asegura que las aplicaciones pueden alcanzar una esperanza de vida de 10 a 15 años con una batería pequeña. En cambio, si utilizamos *Time Division Multiple Access* (TDMA) se requiere sincronización entre los dispositivos, lo cual conlleva un coste energético adicional en los dispositivos finales ya que tienen que estar activos y transmitiendo para asegurar que siguen sincronizados. [23]

Las redes LoRaWAN, *Fig. 8*, están compuestas por cuatro elementos principales: dispositivos finales, *gateways*, servidor de red y servidor de aplicación. Adicionalmente, en el caso de que sea una comunicación OTAA, también se necesita el servidor de unión.

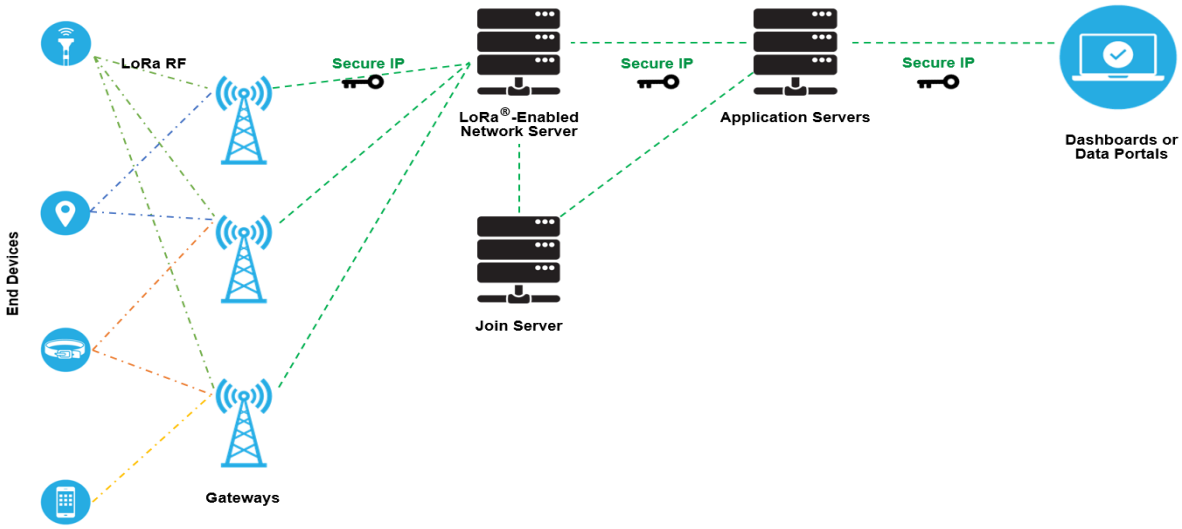

Fig. 8 Estructura red LoRaWAN con activación OTAA [15]

A continuación, se van a explicar, de forma resumida, los distintos componentes, así como sus diferentes funciones dentro de la red y sus características principales.

2.3.2.1. Dispositivos finales

Los dispositivos finales LoRaWAN, *Fig. 9*, suelen ser sensores que se conectan de forma inalámbrica a una red LoRaWAN a través de las puertas de enlace.

En la mayoría de las aplicaciones, los dispositivos finales son sensores que digitalizan condiciones físicas y sucesos ambientales, autónomos y normalmente alimentados por baterías.

Cuando se fabrican, a los dispositivos basados en LoRa se les asignan varios identificadores únicos, los cuales dependen del modo de activación, que se utilizan para activar y administrar de forma segura el dispositivo, asegurar el transporte seguro de paquetes a través de redes públicas o privadas y llevar los datos encriptados a la nube. [15]

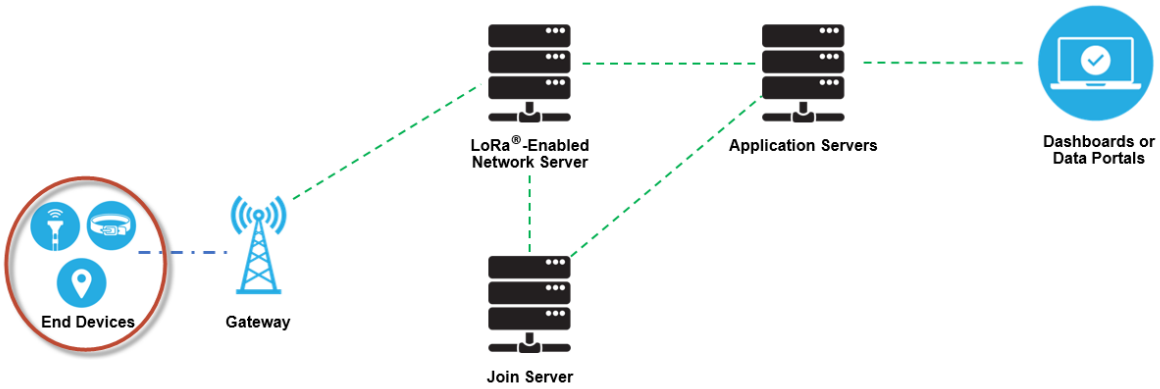

Fig. 9 Dispositivos finales dentro de una red LoRaWAN [15]

2.3.2.2. Gateways

Un *gateway* o puerta de enlace LoRaWAN, *Fig. 10*, recibe mensajes de cualquier dispositivo que se encuentre dentro del radio de cobertura del *gateway* y reenvía estos mensajes al servidor de red. No hay una asociación fija entre un dispositivo final y un *gateway* en concreto. En cambio, cada mensaje de *uplink* enviado por el dispositivo final lo recibirán todos los *gateways* a su alcance. *Fig. 11*. De este modo, se reduce de forma significativa la tasa de error, ya que es muy probable que haya al menos un *gateway* que reciba el mensaje, y la sobrecarga de la batería para dispositivos móviles, y permite geolocalización *low-cost*, en el caso de que los *gateways* tengan capacidad para ello.

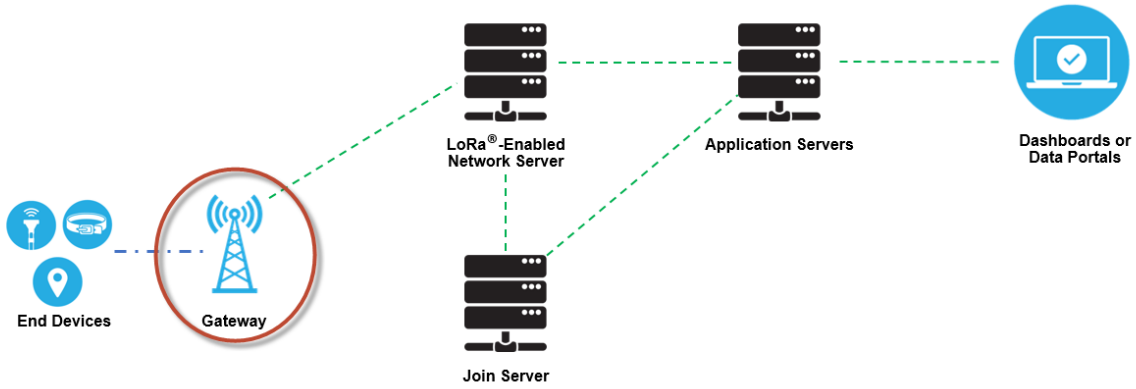

Fig. 10 Gateway dentro de una red LoRaWAN [15]

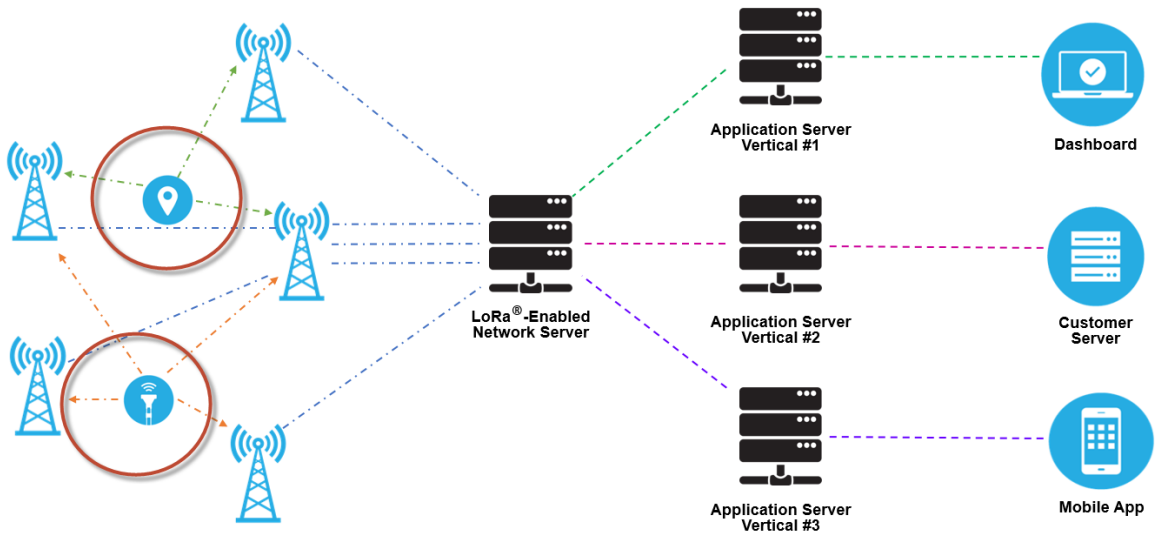

Fig. 11 Gateways recibiendo y enviando mensajes de los dispositivos finales [15]

El tráfico del *gateway* se puede enviar al servidor de red a través de Wi-Fi, Ethernet, etc. Las puertas de enlace LoRaWAN solo trabajan en la capa física, y solo verifican la integridad de los datos de los mensajes recibidos. Si el CRC (*Cyclic Redundancy Check*) es incorrecto, el mensaje se eliminará. Si, por el contrario, el CRC es correcto, el *gateway* reenviará el mensaje al servidor de red, junto con otros datos, como el nivel de RSSI (*Received Signal Strength Indicator*) recibido. En el caso de mensajes de *downlink*, el *gateway* realiza las transmisiones solicitadas por el servidor de red sin hacer ninguna interpretación de la carga útil. Como varios *gateways* distintos pueden recibir el mismo mensaje de un dispositivo final, el servidor de red elimina todas las copias. Esto lo realiza basándose en el nivel de RSSI de los mensajes que coinciden y, normalmente, el servidor de red selecciona el *gateway* que ha recibido el mensaje con mejor RSSI cuando se transmiten mensajes de *downlink*, ya que esto significa que ese *gateway* es el más cercano al dispositivo final. [15]

2.3.2.3. Servidor de red

El servidor de red LoRaWAN, *Fig. 12*, administra toda la red. Se encarga establecer conexiones seguras para el transporte de datos de extremo a extremo, desde el dispositivo final a la aplicación de los usuarios finales en la nube, y de controlar el tráfico que fluye desde el dispositivo final al servidor de red, y viceversa, entre otras cosas. El servidor de red garantiza la autenticidad de todos los sensores de la red y la integridad de todos los mensajes. Al mismo tiempo, el servidor de red no puede ver ni acceder a los datos de la aplicación.

En general, todos los servidores de red LoRaWAN comparten las siguientes características: comprueban la dirección del dispositivo, autenticación de tramas y gestión de contadores, "asentimientos" de mensajes recibidos, adaptación de tasas de datos, … [15]

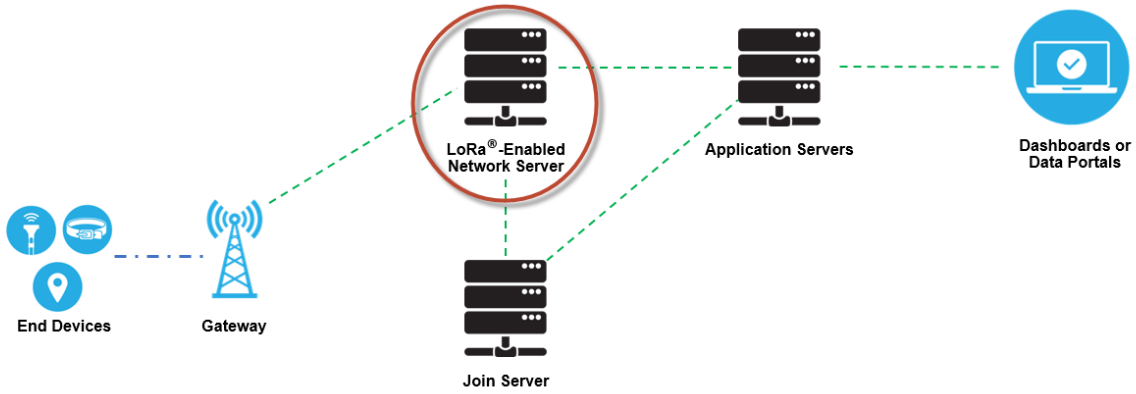

Fig. 12 Servidor de Red dentro de una red LoRaWAN [15]

2.3.2.4. Servidor de aplicación

Los servidores de aplicación, *Fig. 13*, son los responsables de gestionar, administrar e interpretar de forma segura los datos de aplicaciones de los dispositivos finales. También se encargan de generar las cargas útiles de *downlink* de la capa de aplicación a los dispositivos finales. [15]

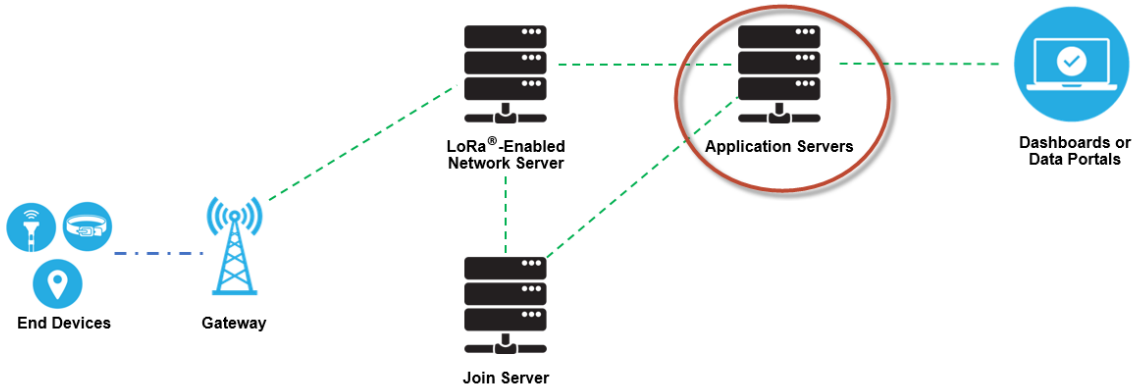

Fig. 13 Servidor de Aplicación dentro de una red LoRaWAN [15]

2.3.2.5. Servidor de unión

El servidor de unión, *Fig. 14*, gestiona la activación *Over-The-Air* para que los dispositivos finales se unan a la red, por lo que en el caso de que se utilice activación ABP no se utiliza.

Este servidor tiene la información requerida para procesar las tramas de *join-request* de *uplink* y generar las tramas de *join-accept* de *downlink*. Esto le indica al servidor de red a qué servidor de aplicación se debe conectar al dispositivo final. [15]

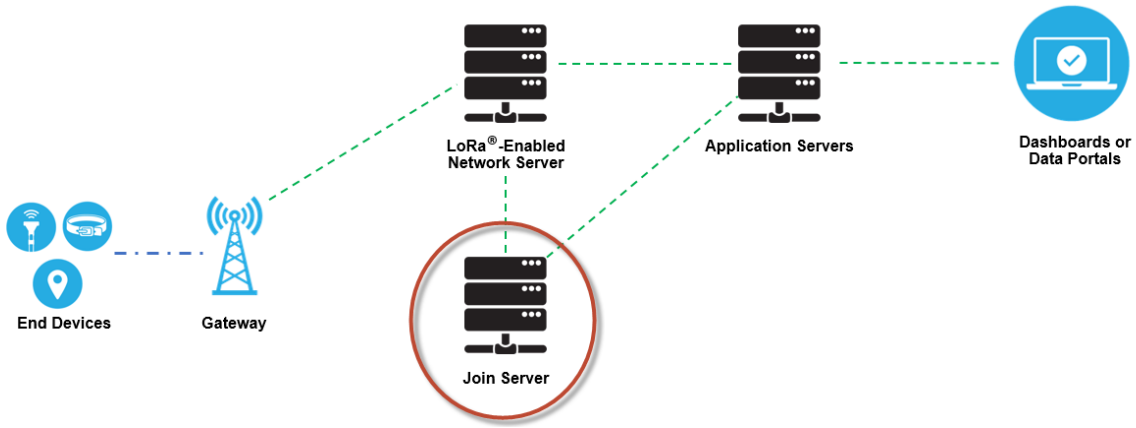

Fig. 14 Servidor de Unión dentro de una red LoRaWAN [15]

2.3.3. Clases de dispositivos

Los dispositivos finales sirven para diferentes aplicaciones y tienen requisitos distintos. Para optimizar la variedad de perfiles de aplicaciones finales, LoRaWAN utiliza clases de dispositivos diferentes. Las clases de dispositivos compensan la latencia de comunicación del *downlink* de la red frente a la vida útil de la batería. En una aplicación de tipo control, la latencia de comunicación de *downlink* es un factor importante. [11]

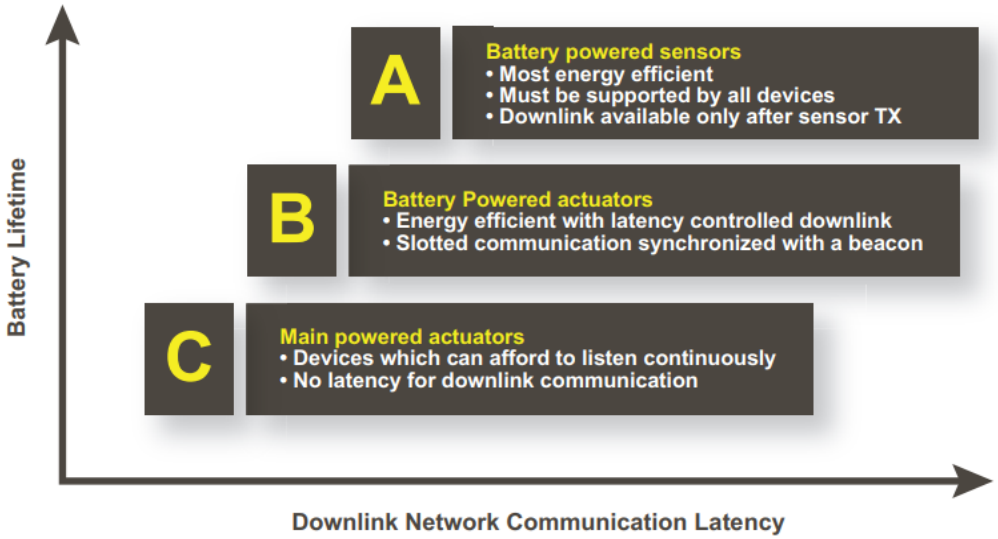

Fig. 15 Clases de dispositivos [11]

Dentro de la especificación de LoRaWAN se definen tres tipos de dispositivos (*Fig. 15*): Clase A, Clase B y Clase C. A continuación, se van a explicar las principales características de estos tipos de dispositivos.

- Clase A: clase por defecto que todos los dispositivos finales LoRaWAN deben de soportar. La comunicación de clase A siempre la empieza el dispositivo final y es completamente asíncrona. Cada transmisión de *uplink* se puede producir en cualquier momento, ya que se trata de un protocolo de tipo *ALOHA*, y va seguida de dos pequeñas ventanas de transmisión *downlink*, con un pequeño retardo entre el final de la transmisión de *uplink* y el comienzo de las ventanas de recepción, como se puede ver en la *Fig. 16*, haciendo posible la comunicación bidireccional. El dispositivo final puede entrar en el modo de bajo consumo tanto tiempo como se haya definido en su aplicación: la red no requiere que haya activaciones periódicas. Esto hace que la Clase A sea el modo de operación de consumo más bajo, mientras que también permite comunicaciones *uplink* en cualquier momento. Como la comunicación de *downlink* siempre tiene que venir después de una transmisión de *uplink* con un cronograma definido por la aplicación del dispositivo final, si el servidor de red no responde durante estas dos ventanas de recepción, esta información se almacenará en el servidor de red hasta la siguiente transmisión de *uplink*. [25] El servidor de red puede responder durante la primera o la segunda ventana de recepción, pero no usa las dos.

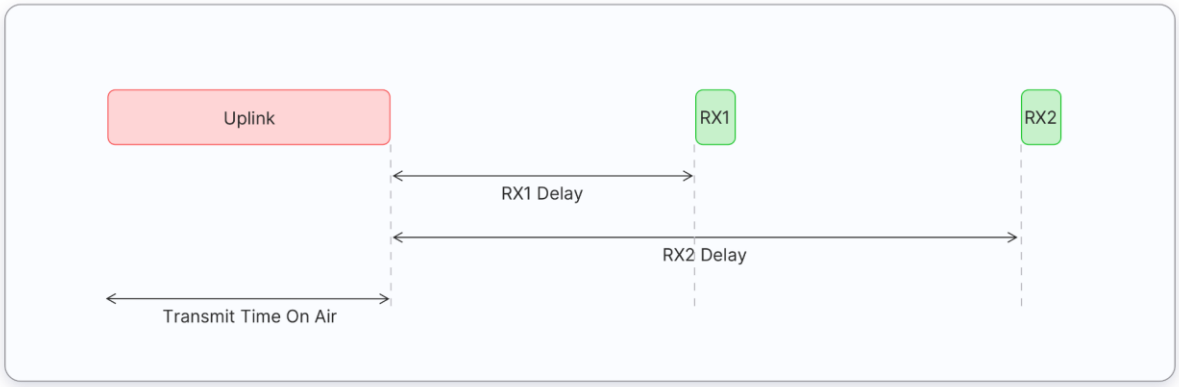

Fig. 16 Ventanas de recepción en dispositivos de Clase A [24]

Los dispositivos de clase A tienen un menor consumo de energía, por lo que normalmente se alimentan con baterías, pasan la mayor parte del tiempo en inactivos, normalmente esperan largos intervalos entre transmisiones de *uplink*, tienen alta latencia en el *downlink*, ya que para recibir un *downlink* antes se tiene que transmitir un *uplink*.

Algunos usos de los dispositivos de clase A son: detección de incendios, detección de fugas de agua, detección temprana de terremotos, seguimiento de animales, seguimiento de ubicación, monitoreo ambiental. [24]

- Clase B: además de las ventanas de la recepción de la Clase A, los dispositivos de Clase B se sincronizan con la red usando balizas periódicas, y abren unas ranuras de tiempo llamadas 'ping slots' en intervalos de tiempo programados para recibir mensajes de *downlink* del servidor de red, *Fig. 17*. El tiempo entre dos balizas se conoce como período de baliza, *beacon period* en inglés. Esto le da a la red la capacidad de enviar comunicaciones de *downlink* con una latencia determinista, a costa de un consumo de potencia adicional en el dispositivo final. La latencia es programable hasta 128 segundos para adaptarse a diferentes aplicaciones, y el consumo de potencia adicional es lo suficientemente bajo como para seguir siendo válido para aplicaciones alimentadas con baterías. [25]

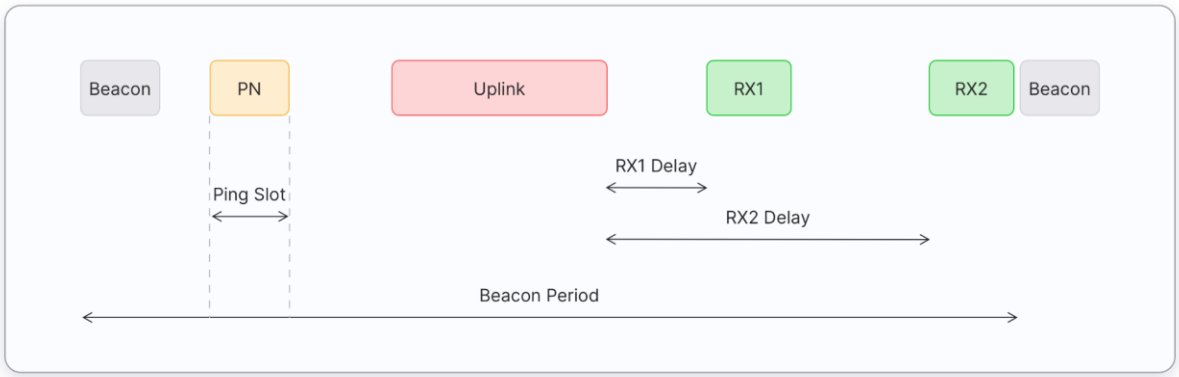

Fig. 17 Ventanas de recepción en dispositivos de Clase B [24]

Los dispositivos de clase B tienen una menor latencia que los de clase, ya que son accesibles en instantes de tiempo preconfigurados y no necesitan enviar un *uplink* para poder recibir un *downlink*. La vida útil de la batería es más corta que en los dispositivos de clase A, ya que los dispositivos están más tiempo activos durante las balizas y los "ping slots".

Usos comunes de dispositivos de clase B: contadores de servicios públicos, informes de temperatura. [24]

- Clase C: además de la estructura de la Clase A de *uplink* seguido de dos ventanas de *downlink*, la clase C reduce aún más la latencia en el *downlink* manteniendo las ventanas de recepción del dispositivo final abiertas siempre que el dispositivo no esté transmitiendo (*half duplex*), *Fig. 18*. En base a esto, el servidor de red puede comenzar una transmisión de *downlink* en cualquier momento asumiendo que el receptor del dispositivo final está abierto, por lo que no hay latencia. El compromiso es el consumo de energía del receptor (hasta 50mW) por lo que la Clase C es adecuada para aplicaciones donde hay energía disponible continuamente. Para dispositivos alimentados por baterías, es posible hacer cambios temporales entre clase A y C, y es útil para tareas intermitentes como actualizaciones de *firmware* "por el aire". [25]

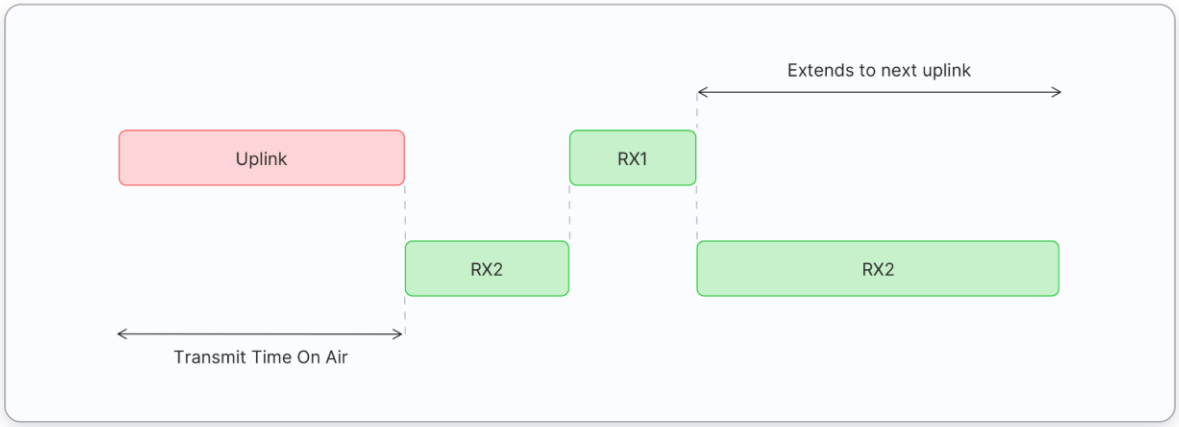

Fig. 18 Ventanas de recepción en dispositivos de Clase C [24]

Los dispositivos de clase C normalmente están alimentados por la red, no tienen latencia de *downlink*.

Algunas aplicaciones en las que se usan dispositivos de clase C: contadores de servicios públicos con válvulas de corte/interruptores, control de farolas. [24]

2.3.4. Modos de activación

Para poder participar en una red LoRaWAN, los dispositivos finales tienen que ser activados y personalizados. La activación de los dispositivos finales se puede realizar de dos formas: *Over-The-Air-Activation* (OTAA) o *Activation By Personalization* (ABP).

Cabe resaltar que los modos de activación y parámetros que se van a explicar a continuación son los indicados por *LoRa Alliance* en la especificación de LoRaWAN v1.1 [27], por lo que varían ligeramente con respecto a los definidos en anteriores especificaciones y más extendidos.

Para poder explicar los modos de activación hay que conocer previamente algunos parámetros que intervienen en estos procesos de activación [27]:

- *DevAddr*: se encarga de identificar al dispositivo dentro de la red actual, de forma similar a una dirección IP.
- *JoinEUI*: se encarga de identificar al servidor de unión. En dispositivos OTAA debe estar almacenado en el dispositivo antes de comenzar el proceso de unión, pero en dispositivos que sólo permiten ABP no es necesario. En anteriores especificaciones se conocía como *AppEUI*. [28]
- *DevEUI*: se encarga de identificar al dispositivo de forma única, independiente del modo de activación empleado, similar a una dirección MAC. En dispositivos

OTAA debe estar almacenado en el dispositivo antes del proceso de unión, y en dispositivos ABP, aunque no es obligatorio, se recomienda que también se haga.

- *AppKey* y *NwkKey*: claves raíz que se asignan a cada dispositivo en la fabricación. Estas claves se utilizan para obtener, en el caso de activación *over-the-air*, las claves de sesión de aplicación y de red.
- *AppSkey*: es una clave de sesión de aplicación específica para el dispositivo final que se emplea en la encriptación y desencriptación. Esta clave debe de almacenarse de forma que se prevenga su extracción y reutilización por parte de actores externos.
- *FnwkSIntKey*, *SnwkSIntKey* y *NwkSEncKey*: claves de sesión de red específicas para el dispositivo final. Estas claves deben de almacenarse de forma que se prevenga su extracción y reutilización por parte de actores externos. En anteriores especificaciones estas tres estaban englobadas dentro de *NwkSkey*. [28]
- *DevNonce*: contador que comienza en 0 al encender el dispositivo y se incrementa con cada *join-request* enviado.

En el *Anexo 1* se encuentran estos parámetros definidos de forma más completa.

- *Over-The-Air-Activation* (OTAA):

En la activación OTAA, el dispositivo final realiza un proceso de unión a una red LoRaWAN, durante el cual se le asigna dinámicamente el *DevAddr*, y las claves raíz (*AppKey* y *NwkKey*), las cuales se asignan al dispositivo durante su fabricación, se utilizan para obtener las claves de sesión. De esta forma, tanto el *DevAddr* como las claves de sesión cambian con cada nueva sesión establecida. [26]

En la *Fig. 19* aparecen los parámetros empleados en el proceso de unión de OTAA en una versión anterior a la especificación LoRaWAN v1.1, por lo que, tanto el *AppEUI* como el *NwkSKey* se sustituirían por *JoinEUI y FnwkSIntKey*, *SnwkSIntKey* y *NwkSEncKey*, respectivamente. [27], [29]

El proceso de unión requiere dos mensajes entre el dispositivo y el servidor de unión. *join-request* (del dispositivo al servidor de unión) y *join-accept* (del servidor de unión al dispositivo).Antes de la activación, el *DevEUI*, *JoinEUI*, *AppKey* y *NwkKey* deben estar almacenados en el dispositivo final.

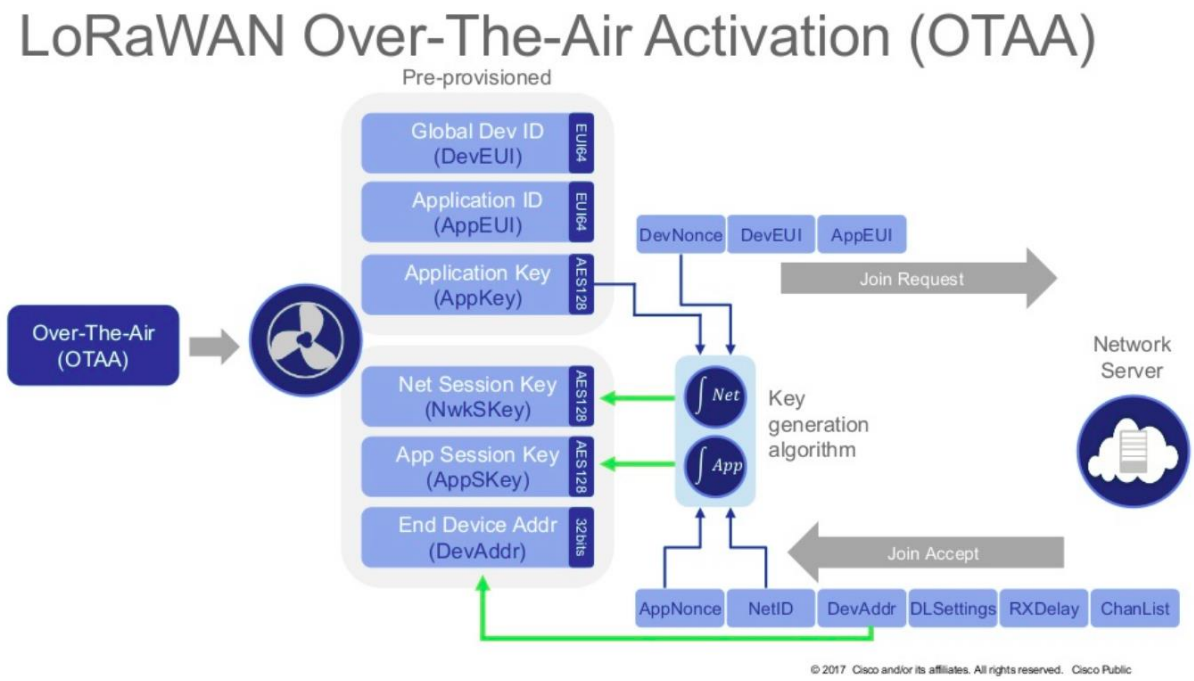

Fig. 19 Esquema de los parámetros empleados en activación OTAA

El proceso de unión siempre lo inicia el dispositivo final, que envía el mensaje de *join-request* a la red a la que se quiere unir. Este mensaje está compuesto por: *JoinEUI*, *DevEUI* y *DevNonce*. Cuando recibe este mensaje, el servidor de red manda el *join-request* al servidor de unión correspondiente. El servidor de red procesa el mensaje y, si se permite al dispositivo unirse a la red, genera las claves de sesión: *AppSKey*, *FNwkSIntKey*, *SNwkSIntKey* y *NwkSEncKey*. Si los pasos anteriores se han realizado con éxito, el servidor de red se encarga de generar el mensaje de *join-accept*. Después, el mensaje de *join-accept* se encripta y se envía como un mensaje de *downlink* normal. El servidor de unión envía al servidor de aplicación la clave *AppSKey* y al servidor de red las claves de sesión de red (*FNwkSIntKey*, *SNwkSIntKey* y *NwkSEncKey*). [27]

- *Activation By Personalization* (ABP):

Para este método de activación, durante la fabricación del dispositivo se le asignan las claves de sesión y el *DevAddr* para una red en concreto, y no cambian durante la vida del dispositivo ABP. Así, el dispositivo final se salta el proceso de unión, lo cual es más sencillo que el método anterior, pero esto conlleva que también sea menos seguro. [26] Este método tiene también la desventaja de que, al utilizar claves y parámetros de red fijos, no pueden conectarse a cualquier red sin antes cambiar de forma manual las claves del dispositivo. [28]

En la *Fig. 20* aparecen los parámetros empleados en el proceso de unión de ABP en una versión anterior a la especificación LoRaWAN v1.1, por lo que la clave *NwkSKey* se sustituiría por *FnwkSIntKey*, *SnwkSIntKey* y *NwkSEncKey*. [27], [29]

El dispositivo almacena el *DevAddr*, las claves de sesión de red (*FNwkSIntKey*, *SNwkSIntKey* y *NwkSEncKey*) y de aplicación (*AppSKey*), en vez de guardar el *DevEUI*, *JoinEUI*, *AppKey* y *NwkKey*. El servidor de red tiene almacenados las claves de sesión de red y el *DevAddr*, y el servidor de aplicación almacena la clave *AppSKey*. De esta forma, al encender el dispositivo este comenzará a transmitir al encenderlo, sin tener que realizar un proceso de unión. [27]

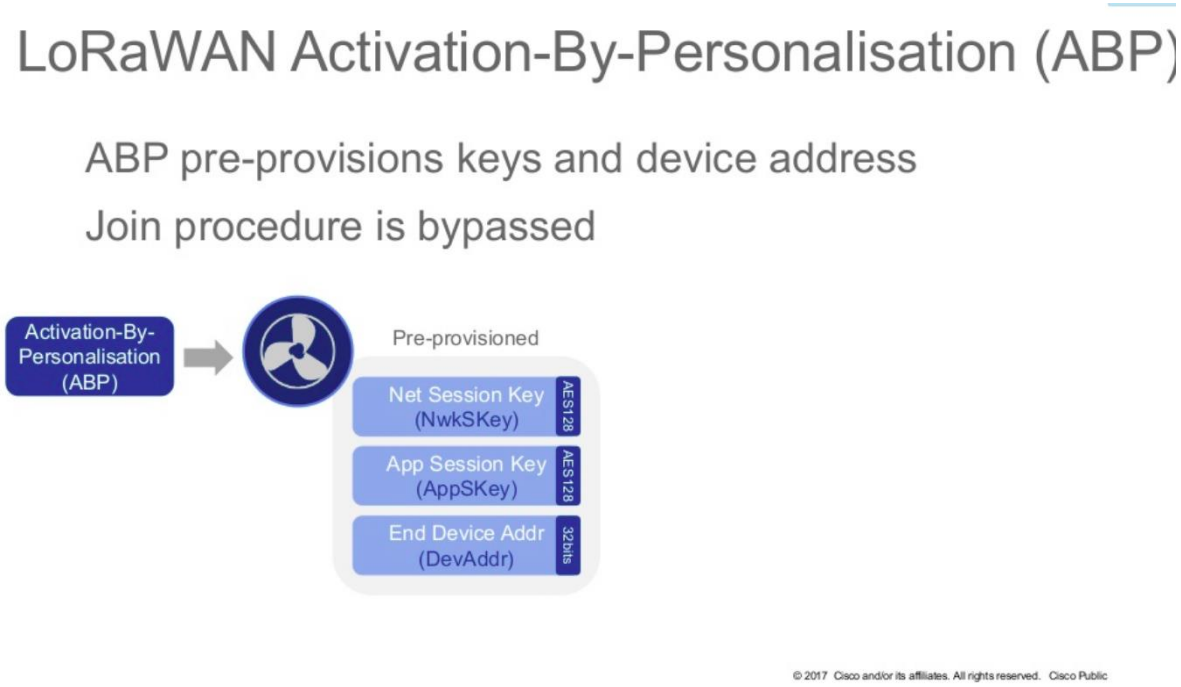

Fig. 20 Esquema parámetros empleados en la activación ABP

2.3.5. Vida útil de la batería

Como se ha comentado antes, los nodos de una red LoRaWAN son asíncronos y solo envían cuando tienen datos listos para enviar, ya sea porque se ha producido algún evento o porque estaba programado. Este protocolo es de tipo *ALOHA*, como se ha mencionado anteriormente.

Hay muchos factores que afectan a la batería, como el factor de ensanchado o la tasa de codificación, pero el principal causante de la reducción de la vida útil de la batería es la sincronización. Al tratarse de una red de malla, los dispositivos normalmente se tienen que "despertar" para sincronizarse con la red y comprobar si hay mensajes, ya que la mayoría del tiempo estos dispositivos se encuentran en "dormidos" o en un estado de aletargo. [11]

2.3.6. Capacidad de la red

Para poder hacer una red de estrella de largo alcance viable, el *gateway* debe tener una alta capacidad para recibir mensajes de una gran cantidad de nodos. Una capacidad de red alta en una red LoRaWAN se puede conseguir utilizando una velocidad de datos adaptativa o ADR, *Adaptative Data Rate*, y usando *gateway* multicanal para que se puedan recibir mensajes simultáneos de distintos canales. Algunos factores críticos que afectan a la capacidad son el número de canales simultáneos, la tasa de datos (tiempo en el aire), el tamaño de la carga útil y la frecuencia con la que los nodos transmiten. Como LoRa es una modulación que se basa en el espectro ensanchado, las señales son prácticamente ortogonales entre sí cuando se utilizan factores de ensanchado distintos. Al cambiar el factor de ensanchado, la tasa de datos efectiva se ve afectada. El *gateway* se aprovecha de esta propiedad al ser capaz de recibir varias tasas de datos diferentes sobre el mismo canal a la vez. Si un nodo tiene un buen enlace y está cerca del *gateway*, no hay ninguna razón por la que tenga que utilizar siempre la tasa de datos más baja y rellenar el resto del espectro disponible más tiempo del necesario. Al aumentar tasa de datos, el tiempo en el aire se acorta, y se abre más espacio para que otros nodos puedan transmitir. La tasa de datos adaptativa también optimiza la duración de la batería de los nodos. Para hacer que esto funcione, se requiere que los enlaces ascendente y descendente sean simétricos con suficiente capacidad de transmisión. Estas características hacen posible que una red LoRaWAN tenga una gran capacidad y que la red sea escalable. Una red se puede desplegar con una cantidad mínima de infraestructura, y como se necesita capacidad, se pueden añadir más *gateways*, aumentando las tasas de datos, reduciendo la cantidad de tiempo de escucha de otros *gateways* y escalando la capacidad por 6-8x. Otras alternativas LPWAN no tienen la escalabilidad de LoRaWAN debido a compensaciones tecnológicas, las cuales limitan la capacidad de transmisión (*downlink capacity*) o hacen que el enlace descendente sea asimétrico al ascendente. [11]

2.3.7. Seguridad

LoRaWAN, como toda red LPWAN, tiene que incorporar seguridad. La seguridad dentro de LoRaWAN se proporciona de distintas maneras. Por una parte, se utilizan dos capas de seguridad, las capas de red y de aplicación. La capa de red asegura la autenticidad del nodo en la red mientras que la capa de aplicación asegura que el operador de red no tiene acceso a los datos de la aplicación del usuario final. Se utiliza encriptación AES con intercambio de claves, *AppSkey*, *FnwkSIntKey*, *SnwkSIntKey* y *NwkSEncKey*, utilizando un identificador IEEE EUI64. [11]

Además del intercambio de claves, también existen contadores, como el *DevNonce*. El servidor de red se encarga de supervisar los últimos valores del *DevNonce* y, si este contador no ha incrementado, el servidor descarta el *join-request*. [27]

La modulación empleada también juega un papel importante en la seguridad frente a las condiciones de funcionamiento, aunque no tanto frente al espionaje. [30]

3. Métodos/ descripción

3.1. ChirpStack

ChirpStack es un proyecto de código abierto, desarrollado por Orne Brocaar, que proporciona los elementos necesarios para crear redes LoRaWAN. Además, incluye una interfaz web fácil de usar para administrar los dispositivos y *gateways* y APIs para la integración. *ChirpStack* cuenta con una licencia MIT, por lo que puede usarse con fines comerciales. [30]

ChirpStack soporta dispositivos de clase A, B y C, tasa de datos adaptativa, es compatible con las versiones 1.0 y 1.1 de LoRaWAN, reconfiguración de canales, y registro de tramas en tiempo real tanto de dispositivos como de *gateways*. Además, cuenta con una interfaz web que permite administrar y configurar los dispositivos y *gateways*. *ChirpStack* proporciona los siguientes componentes:

- *ChirpStack Gateway Bridge*: se encarga de la comunicación con los *gateways* LoRaWAN. Convierte los protocolos de *LoRa Packet Forwarder* en un formato de datos común de *ChirpStack Network Server* (JSON y Protobuf)
- *ChirpStack Network Server*: es la implementación del servidor de red LoRaWAN. Este servidor se encarga de la de-duplicación de las tramas recibidas por los *gateways*, y una vez se tienen las tramas se encarga de la autentificación y de planificar las tramas de *downlink.*
- *ChirpStack Application Server*: es la implementación del servidor de aplicación LoRaWAN. Es el responsable del "inventario" del dispositivo, encargándose de las solicitudes de unión y encriptar las cargas útiles.
- *ChirpStack Gateway OS*: sistema operativo basado en Linux para ejecutar *ChirpStack* en un *gateway* LoRa basado en Raspberry Pi.

3.2. Componentes del *gateway*

El gateway creado para este proyecto está constituido por una Raspberry Pi (*Fig. 21*), un módulo "Seeed Studio 114992549" (*Fig. 22*), un adaptador WM1302 Pi Hat (*Fig. 23*), y una antena de monopolo de cuarto de onda.

Los módulos de puerta de enlace LoRaWAN WM1302 de Seeed Studio son una nueva generación de módulos de *gateway* LoRaWAN en formato mini-PCIe basados en Semtech® SX1302. Estos módulos cuentan con un consumo de energía extremadamente bajo y un rendimiento excepcional con certificación CE, FCC y Telec. Los módulos WM1302 tienen un factor de forma mini-PCIe que ayuda a los módulos a integrarse fácilmente con varios

dispositivos de *gateway* con 52 pines. Estos módulos funcionan a temperaturas ultra-bajas sin disipación de calor adicional que reduce el tamaño de la *gateway* LoRaWAN. Los módulos de *gateway* WM1302 LoRaWAN están diseñados para aplicaciones M2M e IoT y se pueden aplicar ampliamente en escenarios compatibles con *gateway* LPWAN. Las aplicaciones típicas incluyen el desarrollo de dispositivos de *gateway* LPWAN, el desarrollo de aplicaciones de comunicación inalámbrica de larga distancia y el aprendizaje y la investigación de aplicaciones LoRa y LoRaWAN. [31]

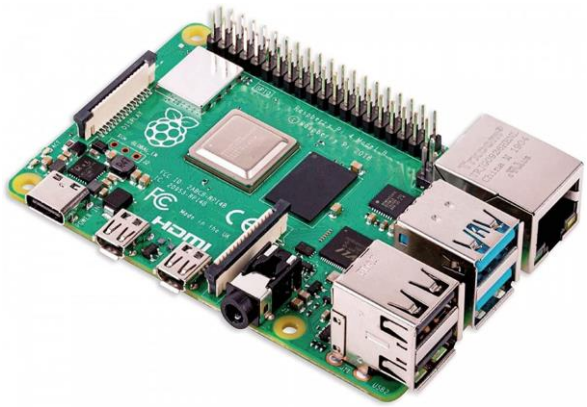

Fig. 21 Raspberry Pi 3 modelo B

Fig. 22 Módulo WM1302 [31]

Fig. 23 Adaptador para el módulo WM1302 [32]

3.3. Implementación del gateway

En este apartado se va a explicar cómo se ha realizado la implementación del *gateway*, y en el Anexo 2 se incluye una explicación más extensa en la que se incluyen todos los comandos empleados a lo largo del proceso.

Para empezar, hay que instalar el sistema operativo "*Raspberry Pi OS with desktop and recommended software*" en la tarjeta SD de la *Raspberry*. Para poder conectarse a la *Raspberry* a través de SSH primero hay que habilitarlo de forma manual, ya que este servicio está deshabilitado por defecto, añadiendo un archivo SSH sin extensión en el directorio de arranque. Una vez hecho esto, podemos conectarnos desde nuestro ordenador a la *Raspberry* con la dirección IP de esta.

Cuando se haya establecido la conexión entre el ordenador y la *Raspberry*, hay que seguir los pasos indicados por *ChripStack* en su página. [34]

Para empezar, hay que instalar los siguientes servicios para que puedan funcionar los servicios específicos de *ChirpStack*:

-*MQTT broker* - Un protocolo de publicación / suscripción que permite a los usuarios publicar información sobre temas a los que otros pueden suscribirse. Una implementación popular del protocolo MQTT es *Mosquitto*.
-*Redis* - motor de base de datos en memoria, basado en el almacenamiento en tablas de hashes pero que opcionalmente puede ser usada como una base de datos durable o persistente.
-*PostgreSQL* - sistema de gestión de bases de datos relacional orientado a objetos y de código abierto.

Una vez tenemos esto instalado, hay que configurarlo. Empezaremos por configurar las bases de datos y usuarios de PostgreSQL, para lo cual hay que crear los usuarios, uno para el *Application Server* y otro para *Network Server*, con sus contraseñas, y luego crear las bases de datos correspondientes a los usuarios anteriormente creados. Una vez creados hay que habilitar las extensiones *pg_trgm* y *hstore*. La extensión *pg_trgm* proporciona funciones y operadores para determinar la similitud de texto alfanumérico en función de la coincidencia de trigramas, así como clases de operadores de índice que admiten la búsqueda rápida de cadenas similares. *Hstore* es una extensión de *PostgreSQL* que implementa el tipo de datos *hstore*, tipo de datos para almacenar conjuntos de parejas clave/valor dentro de un único valor de *PostgreSQL*. En la *Fig. 24* podemos ver los comandos empleados en este proceso.

```
-- set up the users and the passwords
-- (note that it is important to use single quotes and a semicolon at the end!)
create role chirpstack_as with login password 'dbpassword';
create role chirpstack_ns with login password 'dbpassword';

-- create the database for the servers
create database chirpstack_as with owner chirpstack_as;
create database chirpstack_ns with owner chirpstack_ns;

-- change to the ChirpStack Application Server database
\c chirpstack_as

-- enable the pq_trgm and hstore extensions
-- (this is needed to facilitate the search feature)
create extension pg_trgm;
-- (this is needed to store additional k/v meta-data)
create extension hstore;

-- exit psql
\q
```

Fig. 24 Configuración de las bases de datos de PostgrSQL [34]

Después hay que configurar el repositorio del software de *ChirpStack*. *ChirpStack* proporciona un repositorio que es compatible con el sistema de paquetes *apt* de *Ubuntu*. Primero, hay que asegurarse de que *dirmngr* y *apt-transport-https* estén instalados, configurar la clave para este nuevo repositorio, y agregar el repositorio a la lista de repositorios creando un nuevo archivo. Por último, se actualiza la caché del paquete *apt*.

Ahora tenemos que instalar *ChirpStack Gateway Bridge*, *ChirpStack Network Server* y, por último, *ChirpStack Application Server*. Una vez instalados, hay que hacer algunos cambios en el archivo de configuración de *ChirpStack Application Server*, reemplazando el *jwt_secret* por uno seguro, como se indica en la página de *ChirpStack*, generando un *jwt_secret* aleatorio, como se puede ver en la *Fig. 25*.

```
# JWT secret used for api authentication / authorization
# You could generate this by executing 'openssl rand -base64 32' for example
jwt_secret="oQcCYJXWnZo69v+Sm4/CaB4oM0L2prqczJ+Y7mGJPxU="
```

Fig. 25 Ejemplo de jwt_secret

Dentro de los archivos de configuración de *Network Server* y *Application Server* hay que hacer unos cambios para incluir los usuarios y contraseñas creados al configurar *PostgreSQL*

Ahora que ya están instalados todos los servicios, hay que iniciarlos y habilitarlos. En la *Fig. 26* se indica como hacer esto. Hay que tener en cuenta que hay que sustituir en cada caso el servicio a iniciar y habilitar. Una vez hecho esto, hay que comprobar que no haya errores en los archivos de configuración, usando el comando que se muestra en la *Fig. 27*, sustituyendo en cada caso el servicio que se necesite.

```
# start chirpstack-network-server
sudo systemctl start chirpstack-network-server

# start chirpstack-network-server on boot
sudo systemctl enable chirpstack-network-server
```

Fig. 26 Comandos para el inicio y la habilitación de Network Server

```
sudo journalctl -f -n 100 -u chirpstack-network-server
```

Fig. 27 Comando para comprobar errores en el archivo de configuración de Network Server

Ahora que estos servicios están funcionando correctamente, hay que instalar el módulo WM1302 siguiendo la guía que ofrecen en la página de *Seeed* [32]. Primero, hay que comprobar que en el archivo de configuración del *Gateway bridge*, este bien configurado el *backend*. Para que todo funcione bien tiene que estar configurado como en la *Fig. 28*.

```
# Gateway backend configuration.
[backend]
# Backend type.
type="semtech_udp"
```

Fig. 28 Configuración del backend

El módulo WM1302 se comunica con Raspberry Pi a través de SPI e I2C, pero estas dos interfaces no están habilitadas de forma predeterminada en Raspbian, por lo que hay que habilitarlas antes de usar el módulo WM1302. Para configurarlo, hay que acceder a la configuración de la *Raspberry*, en concreto a la sección de *Interface Options*. Una vez hecho esto, se reinicia la *Raspberry* para asegurar que la configuración funcione.

Ahora, hay que descargar la biblioteca y programas específicos de *SX1302 LoRa Gateway*, y compilarlo. A continuación, hay que realizar unas pequeñas modificaciones que se

detallan en el Anexo 2. Una vez hechos estos cambios, podemos ejecutar el *packet forwarder*, para lo cual hay que seleccionar un archivo de configuración de los que aparecen en la *Fig. 29*.

```
pi@raspberrypi:~/sx1302_hal/packet_forwarder $ ls
global_conf.json.sx1250.AS923.USB          inc
global_conf.json.sx1250.CN490              lora_pkt_fwd
global_conf.json.sx1250.CN490.USB          Makefile
global_conf.json.sx1250.EU868              obj
global_conf.json.sx1250.EU868.USB          PROTOCOL.md
global_conf.json.sx1250.US915              readme.md
global_conf.json.sx1250.US915.USB          reset_lgw.sh
global_conf.json.sx1255.CN490.full-duplex  src
global_conf.json.sx1257.EU868
```

Fig. 29 Posibles archivos de configuración para ejecutar el packet forwarder

En este caso el archivo a utilizar es el que acaba por EU868, dentro de este hay que modificar los parámetros *gateway_ID*, *server_address*, *serv_port_up* y *serv_port_down* para que se reenvíen los paquetes al servidor de manera correcta, como se muestra en la *Fig. 30*. Así, podemos ejecutar el *packet forwarder* y ver que funciona de forma correcta.

```
"gateway_conf": {
    "gateway_ID": "AA555A0000000000",
    /* change with default server address/ports */
    "server_address": "localhost",
    "serv_port_up": 1700,
    "serv_port_down": 1700,
```

Fig. 30 Parámetros modificados

Una vez completados estos pasos, podemos pasar a la configuración del *gateway* y, posteriormente, a la configuración de un dispositivo. Todo esto se hace desde la interfaz web del *Application Server* de *ChirpStack* que, una vez instalado y funcionando, se puede acceder a la interfaz web: http://localhost:8080/.

Inicialmente, al abrir la página, se mostrarán todos los elementos vacíos. Primero, hay que añadir un *Network Server* o Servidor de Red (*Fig. 31*)

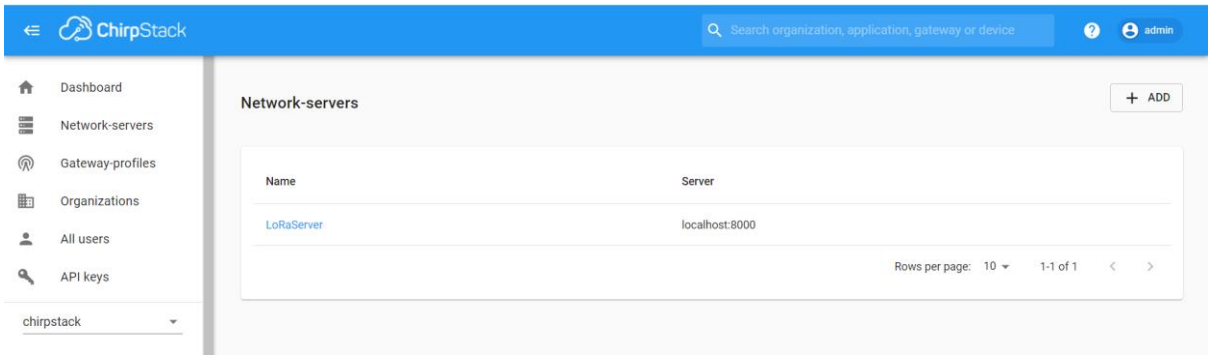

Fig. 31 Servidores de red en la interfaz web de ChirpStack

Se rellenan los campos requeridos, *Fig. 32*, con los parámetros usados en la instalación.

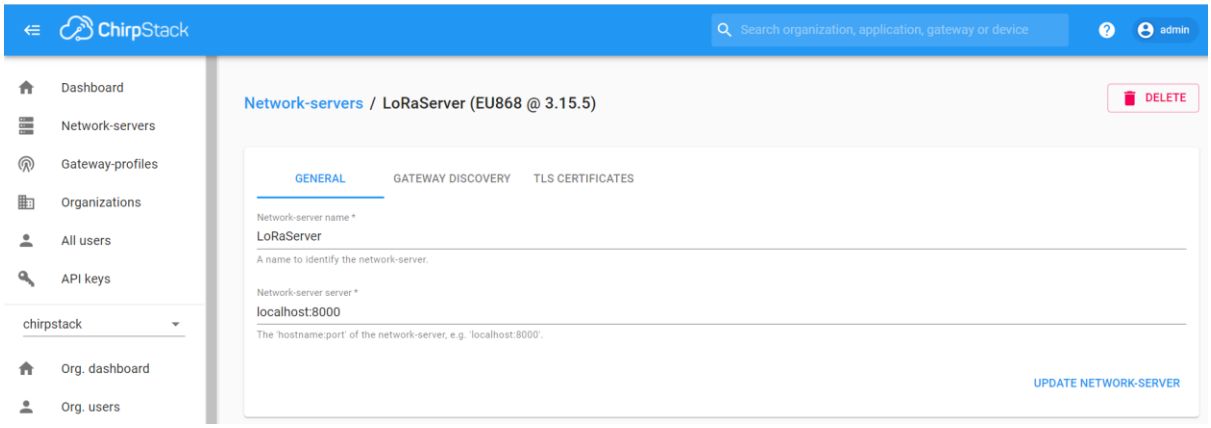

Fig. 32 Campos requeridos para añadir un Servidor de Red

Se requiere disponer de un *gateway profile*, o perfil de *gateway*, para después poder añadir un *gateway*, *Fig. 33*.

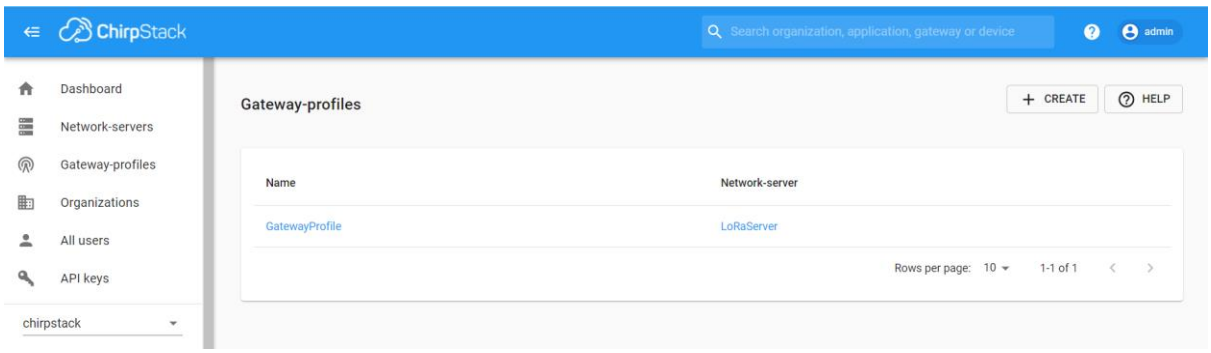

Fig. 33 Perfiles de gateway en la interfaz web de ChirpStack

Dentro de este, se indican los canales que usa el *gateway*, el intervalo de tiempo en el que el *gateway* informa de sus estadísticas, y el *network server* creado anteriormente, *Fig. 34*.

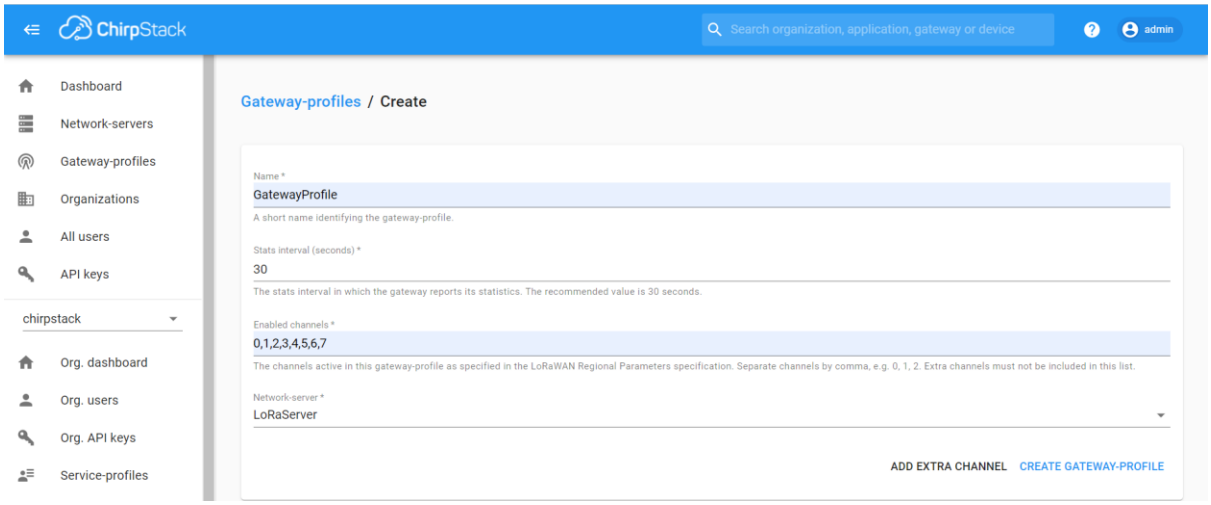

Fig. 34 Campos requeridos para añadir un perfil de gateway

A continuación, es necesario crear un *Service Profile*, o perfil de servicio, para los *gateways*, *Fig. 35*.

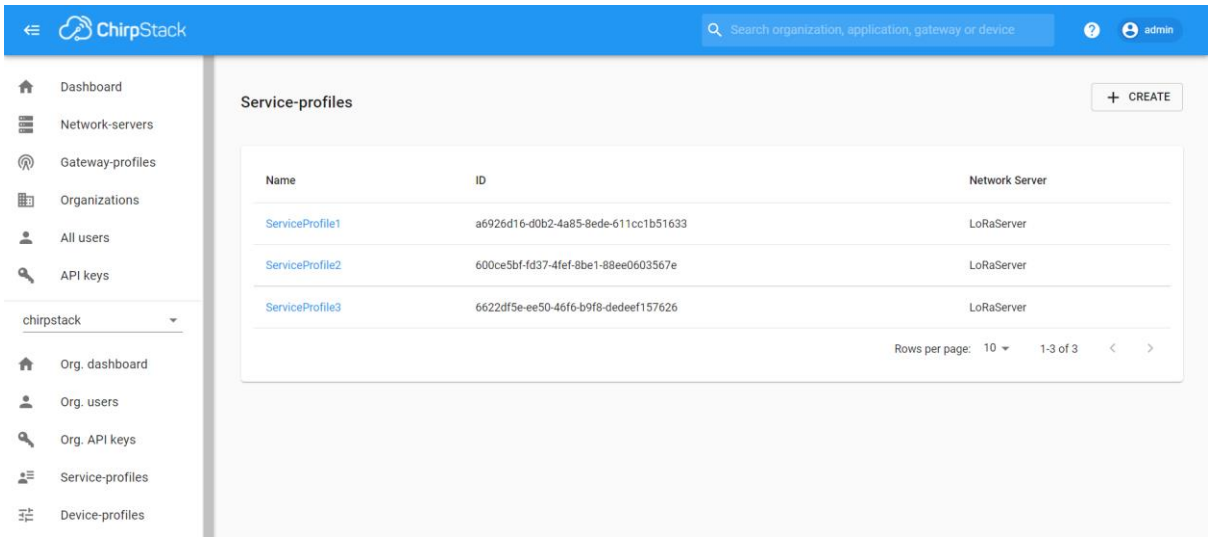

Fig. 35 Perfiles de servicios en la interfaz web de ChirpStack

Dentro de este perfil se indica a que *Network Server* se conectará nuestro *gateway* y parámetros relacionados con la señal, *Fig. 36*.

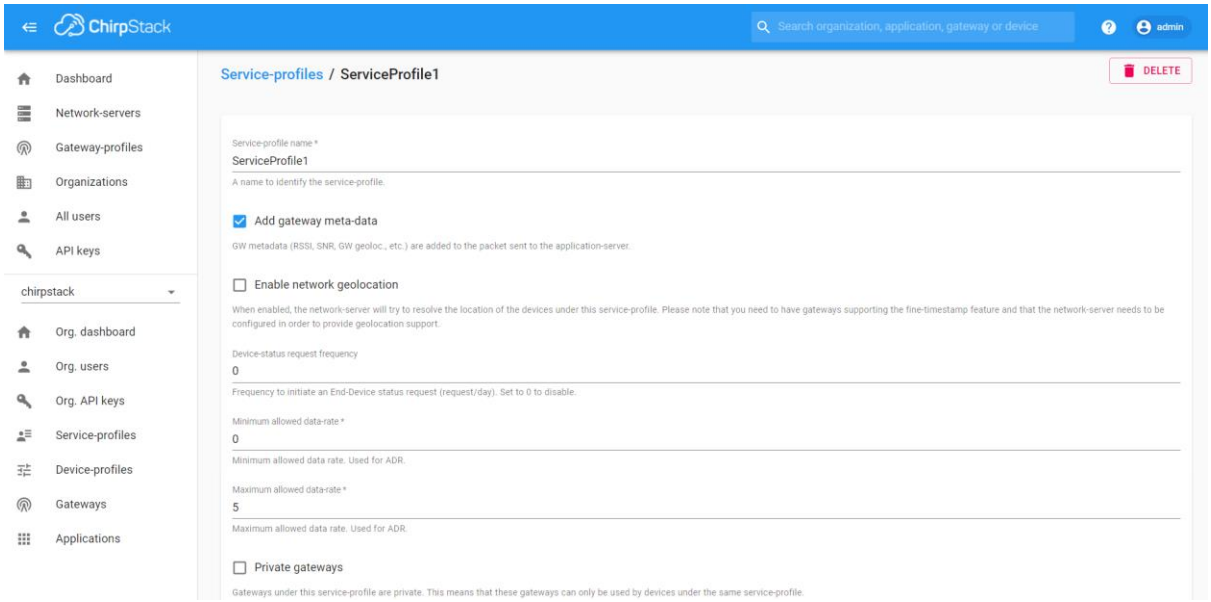

Fig. 36 Campos requeridos para añadir un perfil de servicio

Una vez completados estos pasos, se puede añadir un *gateway*, *Fig. 37*.

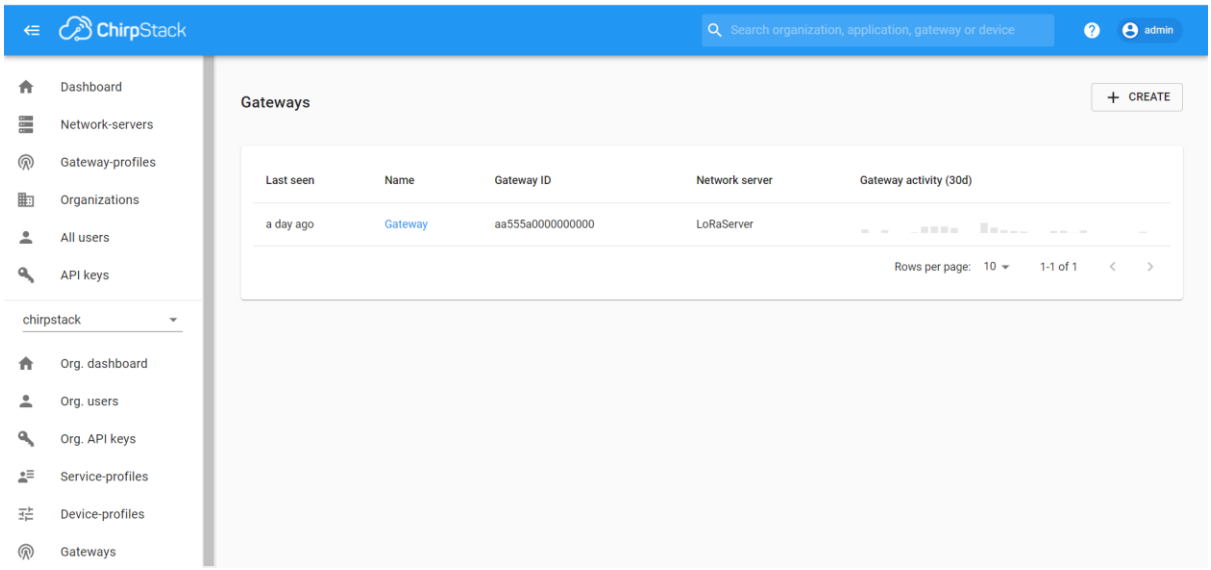

Fig. 37 Gateways en la interfaz web de ChirpStack

Para poder añadir el *gateway* necesitamos el parámetro *gateway_ID*, *Fig. 38*,configurado anteriormente en el archivo de configuración del *packet forwarder.*

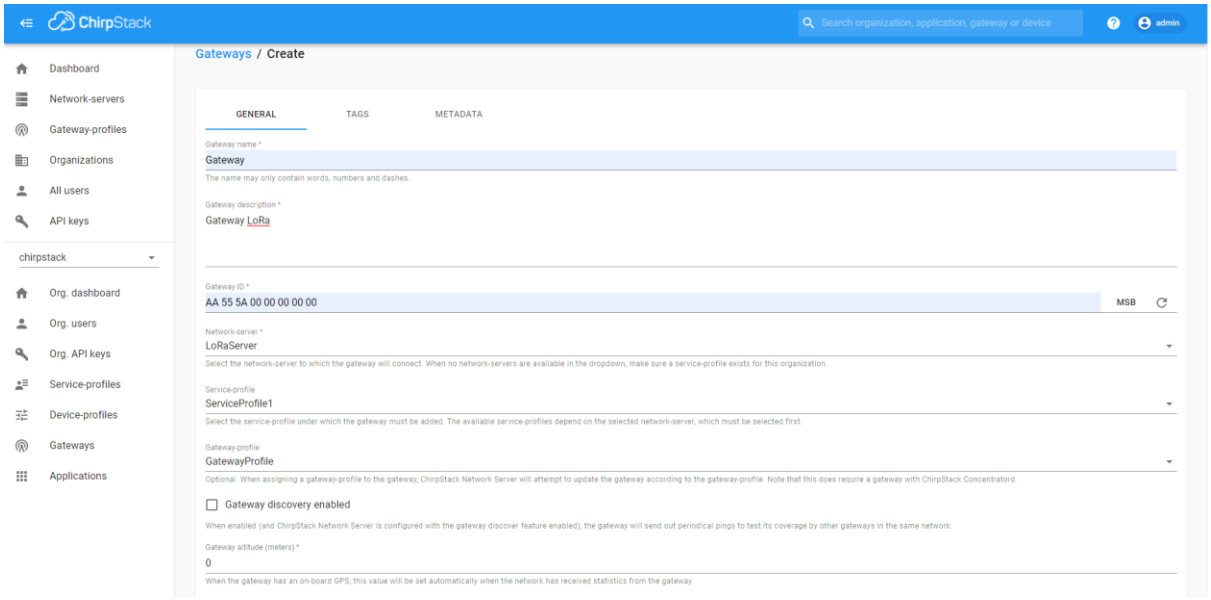

Fig. 38 Campos requeridos para añadir un gateway

Una vez completados estos pasos, hay que comprobar que el *gateway* funciona correctamente. La primera forma es comprobar que en el apartado *Last seen* del menú de *gateways* ponga *a few seconds ago*, ya que, aunque el *gateway* no esté recibiendo datos, periódicamente envía sus estadísticas, en este caso cada 30 segundos, y cuando recibe estas estadísticas el *Application Server* actualiza el *Last seen*. La segunda es ir al apartado *Dashboard* y comprobar que el apartado de *Active gateways* aparezca en verde, *Fig. 39*.

Fig. 39 Dashboard de la interfaz web de ChirpStack

Otra forma de comprobarlo es entrar en la página del *gateway*, y una vez ahí abrir la pestaña *Live LoRaWAN Frames*, esto mostrará todas las tramas *LoRaWAN* que recibe y envía el *gateway*. En el caso de tramas recibidas, aquí se pueden ver tramas recibidas de dispositivo, aunque estos estén configurados todavía. Por lo tanto, esta pantalla es útil para validar si el *gateway* puede recibir tramas LoRaWAN y reenviarlas a *ChirpStack*.

Una vez configurado el *gateway*, podemos configurar un dispositivo, para lo cual, igual que al añadir un *gateway*, hay que realizar unos pasos previos. Primero, hay que crear un nuevo *Service Profile*, de forma análoga a como se ha realizado con el *gateway*.

A continuación, hay que crear un *Device Profile*, o perfil de dispositivo, que describe la forma de conexión y autenticación de un grupo de dispositivos, *Fig. 40*. Aquí es donde se indicará si los dispositivos autentican por medio de OTAA/APB, si se incluirán dispositivos clase B, clase C y las instrucciones para codificar o decodificar los datos del dispositivo (CODEC).

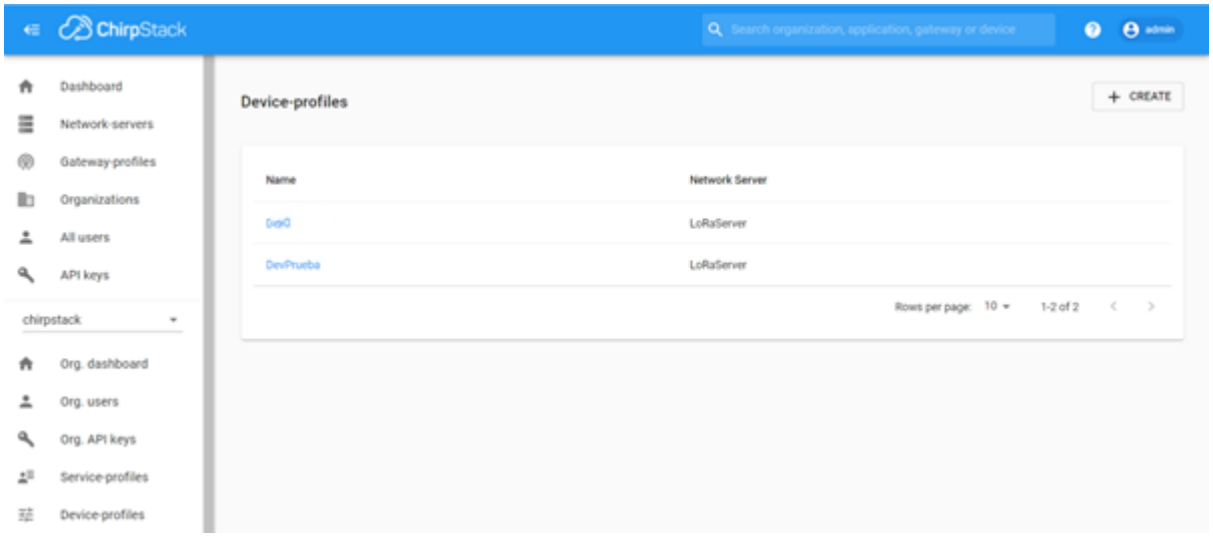

Fig. 40 Perfiles de dispositivo dentro de la interfaz web de ChirpStack

Ahora hay que crear una aplicación, *Fig. 41*, que agrupa dispositivos que realizan funciones semejantes, mediante el *Device Profile*. Aquí solo hay que rellenar el nombre, una pequeña descripción de la función que van a realizar los dispositivos y el *Service Profile*.

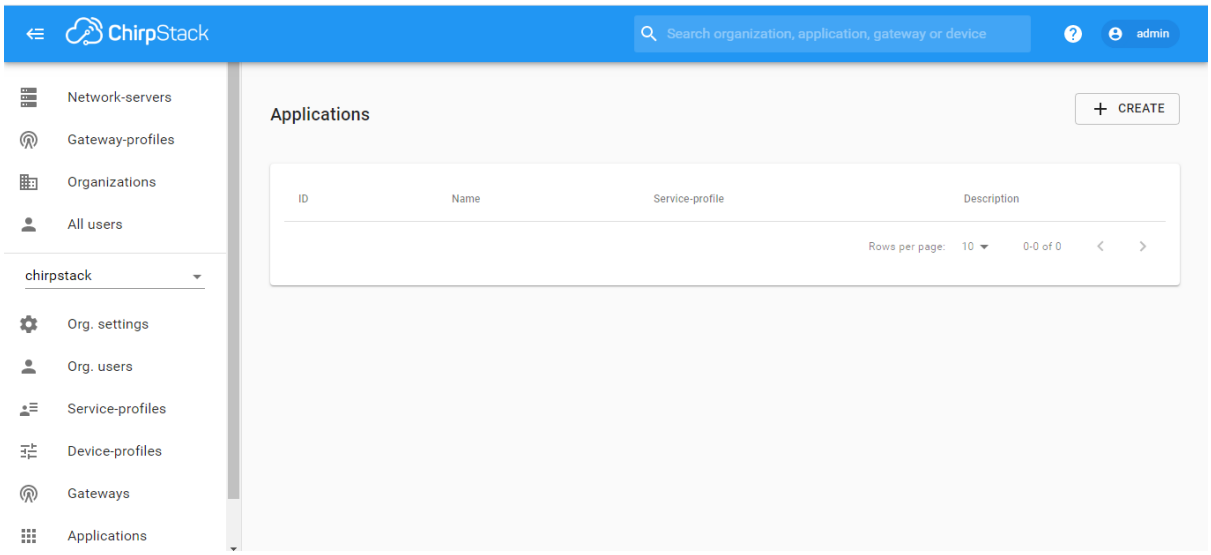

Fig. 41 Aplicaciones en la interfaz web de ChirpStack

Una vez creada la aplicación, hay que añadir el dispositivo que se quiere usar para esa aplicación, para lo que hay que completar el nombre y la descripción, *Fig. 42*. Lo siguiente es el *device EUI* o *DevEUI*. Al introducirlo hay que tener en cuenta el orden de los bits: MSB o LSB. Al usar OTAA hay que rellenar también el campo de *AppKey*,

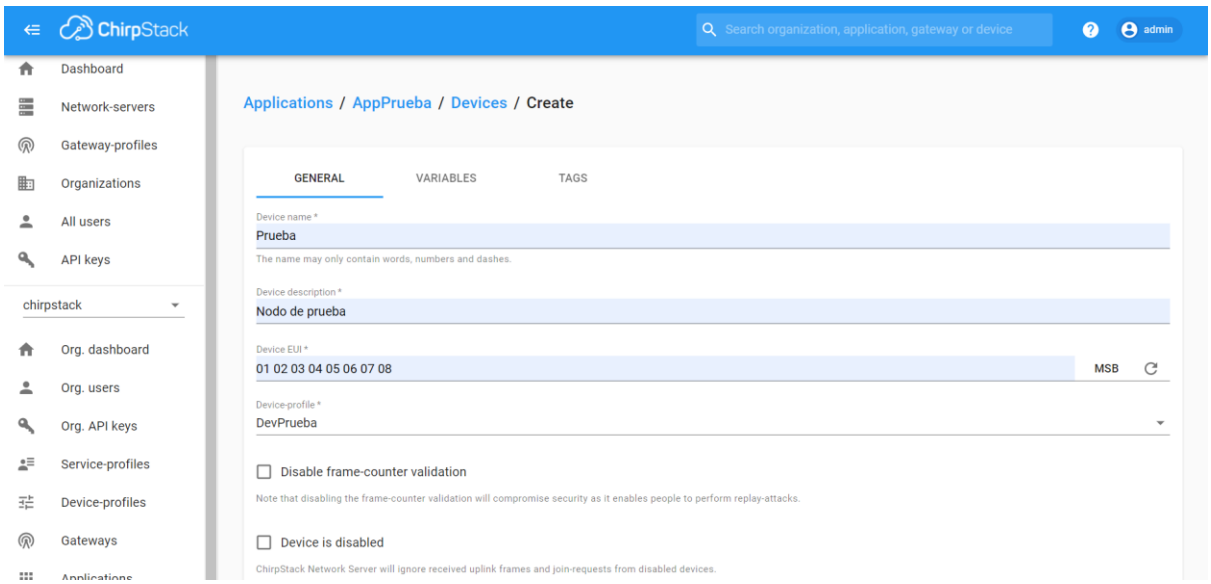

Fig. 42 Campos a rellenar para añadir un dispositivo

Se puede comprobar si el dispositivo se ha conectado correctamente entrando a la página del dispositivo, y ahí a la pestaña *Live LoRaWAN Frames*. Ahí, al ser un dispositivo OTAA, primero debería aparecer un *JoinRequest* seguido de un mensaje *JoinAccept*, *Fig. 43*.

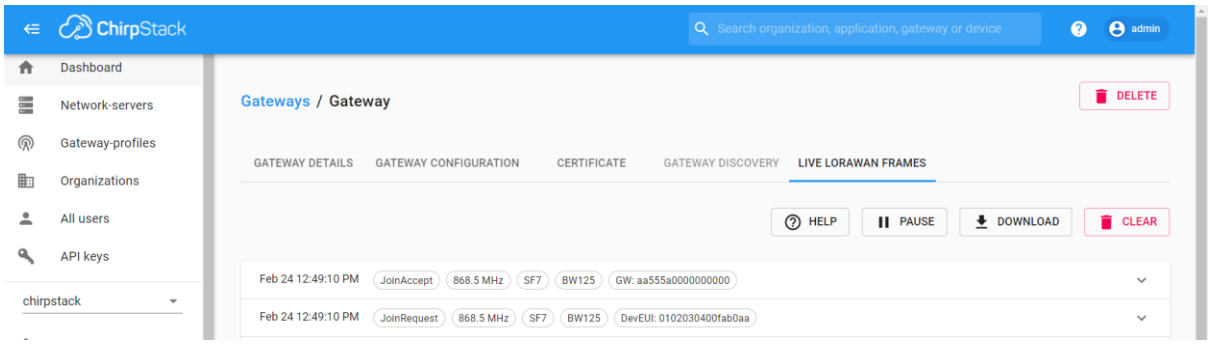

Fig. 43 Mensajes de join-request y join-accept

3.4. Instalación de Node-Red

Node-Red es una herramienta de programación creada para conectar dispositivos de hardware, API y servicios en línea a través de un editor basado en navegador que lo hace más sencillo. [35]

Al instalar *ChirpStack* sobre *Raspbian*, *Node-Red* no viene por defecto por lo que hay que instalarlo manualmente. Con los comandos `sudo npm install npm@latest -g` y `sudo npm install -g –unsafe-perm node-red` instalamos *npm* y *Node-Red*. Para poner en marcha *Node-red* hay que utilizar el comando `node-red-start`, si se quisiera parar el servicio habría que usar el comando `node-red-stop`. Una vez instalado, hay que ponerlo en marcha para poder acceder a su interfaz web, http://[IP_ADDRESS]:1880. *ChirpStack* tiene unos nodos propios, *Fig. 44*, dentro de *Node-Red*, que tienen que instalarse para poder hacer que ambos funcionen de forma conjunta.

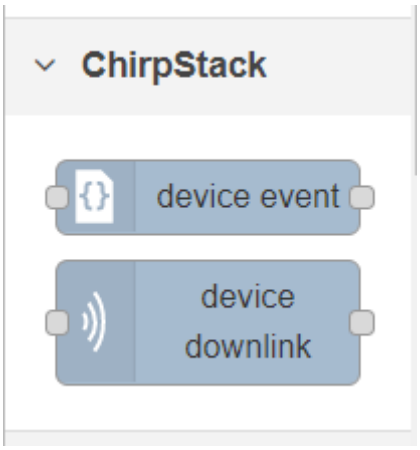

Fig. 44 Nodos propios de ChirpStack dentro de Node-Red

Para instalar estos nodos, hay que emplear los comandos `npm install @chirpstack/node-red-contrib-chirpstack`. Para que al acceder a la interfaz web de *Node-Red* aparezcan estos nodos hay que moverlos al directorio correcto con el comando `cp -r node_modules/@chirpstack/node-red-contrib- chirpstack/.node-red/node_modules/@chirpstack`, y reiniciar el servicio con el comando `node-red-restart`.

Ahora que *Node-Red* está instalado, y se han añadido los nodos necesarios, podemos empezar a crear un flujo de eco sencillo que muestre cómo recibir mensajes de enlace ascendente y procesarlos.

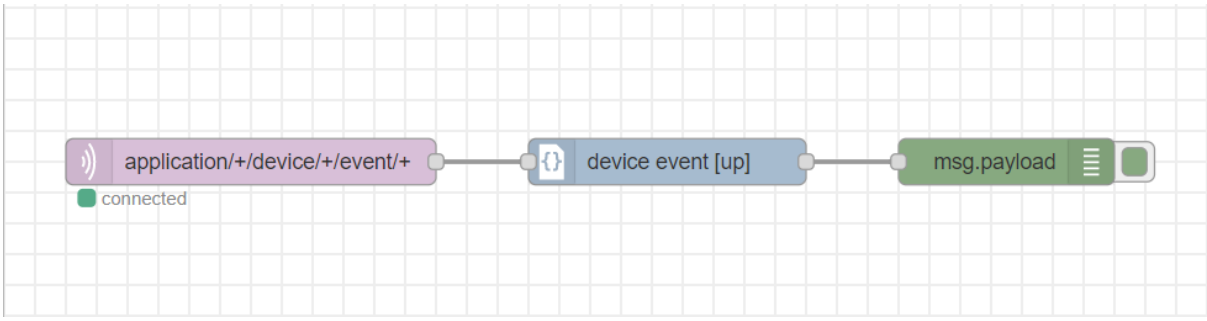

Fig. 45 Flujo de eco para recibir mensajes de enlace ascendente

Para crear el flujo de eco que aparece en la *Fig.* 45, primero añadimos un nodo "*mqtt in*" que se usará para recibir eventos de dispositivos que publica el agente MQTT que se ejecuta en el *gateway*, proporcionado por el sistema operativo *ChirpStack Gateway*). Se deben establecer las propiedades del nodo que aparecen en la *Fig. 46*. A continuación, se añaden los nodos de "*device event up*", que se encarga de filtrar los mensajes de enlace ascendente, y de "*debug*", que se encarga de mostrar los mensajes que recibe del nodo anterior.

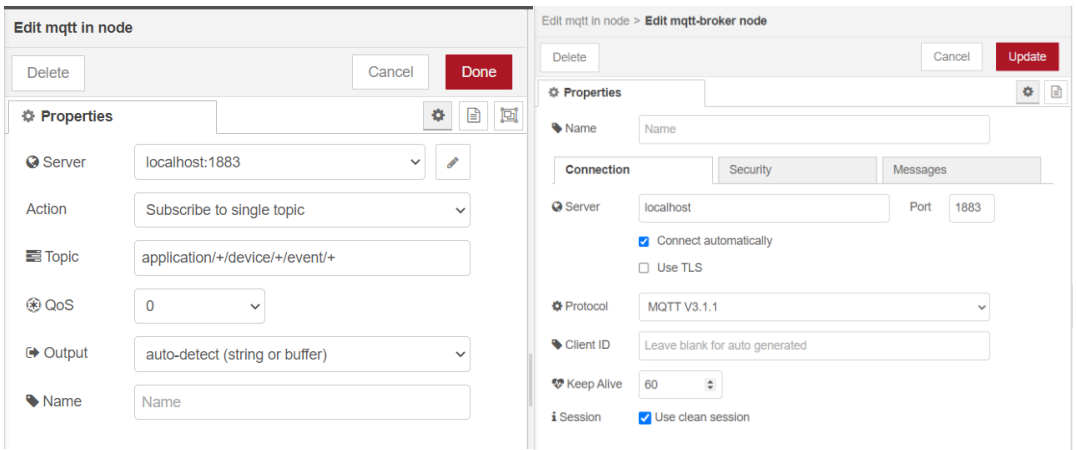

Fig. 46 Propiedades nodo "mqtt"

Antes de comprobar si este flujo de eco funciona, hay que cambiar la configuración "*marshaler*" de "*json_v3*" a "*json*", como se ve en la *Fig. 47*, del servidor de aplicación, en su archivo de configuración, "chirpstack-application-server.toml". Si no se realiza este cambio, saldrá un error como el que se muestra en la *Fig.48* al recibir mensajes del dispositivo.

```
[application_server.integration]
# Payload marshaler.
#
# This defines how the MQTT payloads are encoded. Valid options are:
# * protobuf:  Protobuf encoding
# * json:      JSON encoding (easier for debugging, but less compact than 'protobuf')
# * json_v3:   v3 JSON (will be removed in the next major release)
marshaler="json"
```

Fig. 47 Configuración "mashaler" del Servidor de Aplicación

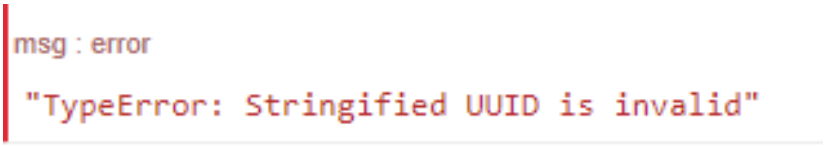

Fig. 48 Error obtenido al recibir mensajes del dispositivo

Una vez hecho este cambio, hay que reiniciar el *Application Server* y *Node-Red*. Ahora podemos ver en *Node-Red* los mensajes que llegan del dispositivo en el apartado de "*debug*", *Fig. 49*.

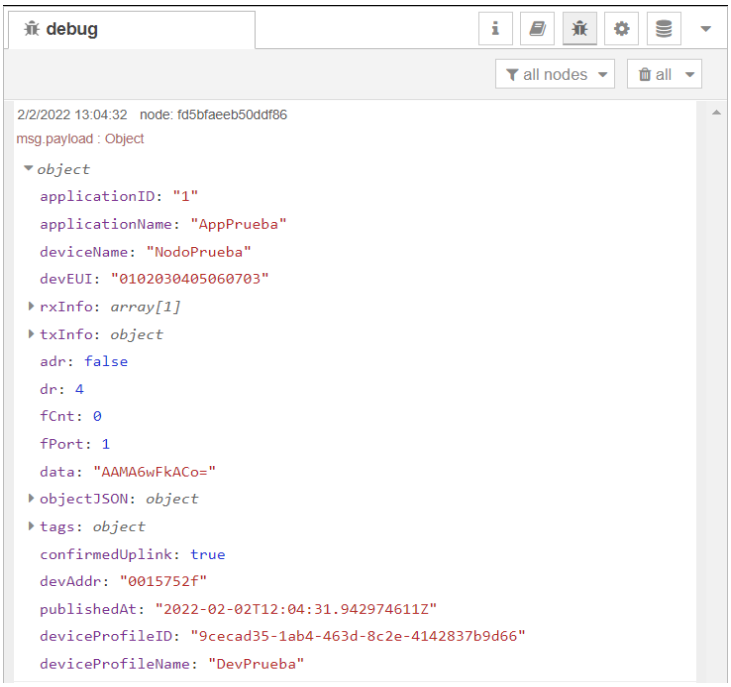

Fig. 49 Mensaje enviado por el dispositivo visto desde Node-Red

Para poder ver los datos decodificados que se reciben del dispositivo, hay una opción dentro de la interfaz web de *ChirpStack* donde se puede crear una función que permita decodificar la información del dispositivo. Esto se hace entrando en perfil de dispositivo correspondiente, y una vez ahí, en el apartado "CODEC", podemos elegir entre tres opciones, que se muestran en la *Fig. 50*.

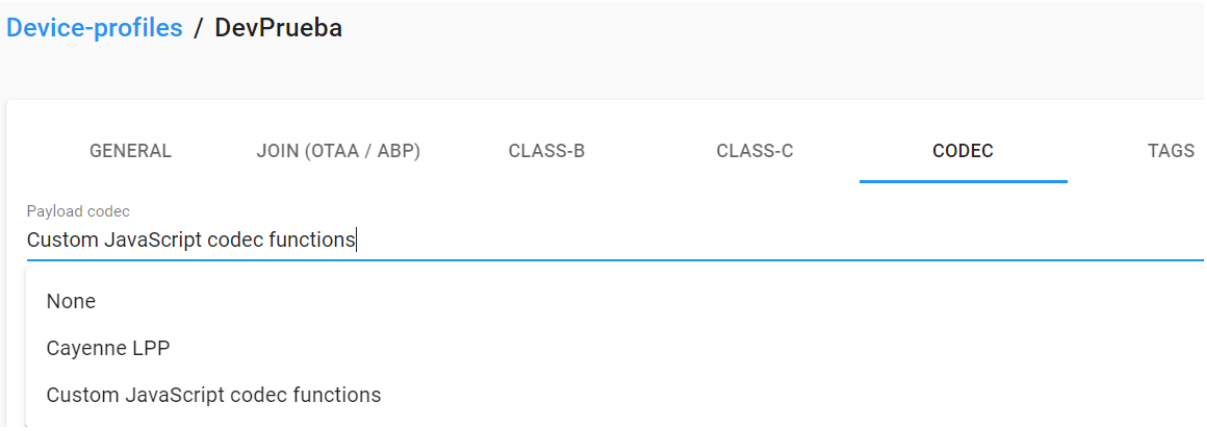

Fig. 50 Opciones de decodificación dentro de la interfaz web de ChirpStack

Hay que elegir última opción, ya que de esta forma se pueden crear una función para la codificación y otra para la decodificación. En este caso, el dispositivo de prueba es un dispositivo que mide la temperatura y la humedad, por lo que la función necesaria para decodificar la información es relativamente sencilla, y se muestra en la *Fig. 51*.

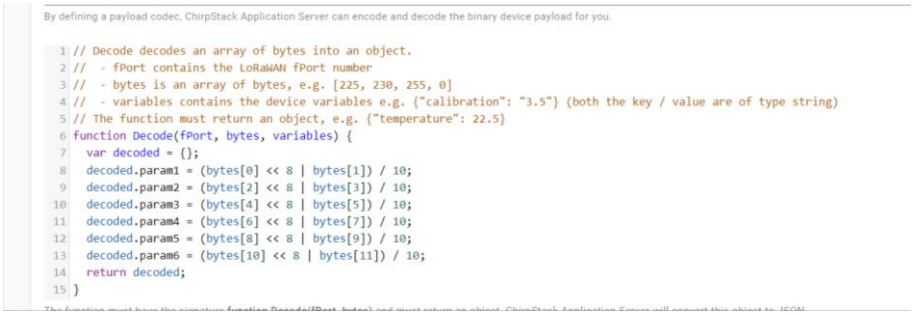

Fig. 51 Función empleada para la decodificación de los datos enviados por el dispositivo

Una vez hecho esto, en *Node-Red* cuando se reciban mensajes del dispositivo, dentro del mensaje, en el apartado "*objectJSON*" podremos ver los parámetros decodificados, como en la *Fig. 52*.

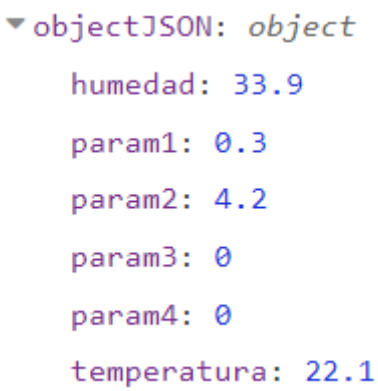

Fig. 52 Parámetros enviados por el dispositivo decodificados

4. Resultados

Una vez finalizado el proceso de implementación, teniendo una puerta de enlace funcional, se pasa a la realización de pruebas para comprobar que el funcionamiento del *gateway* es el esperado. Dentro de esta fase de pruebas se han realizado dos tipos de pruebas: la primera prueba se ha realizado en un interior, teniendo dos nodos conectados al *gateway*, y la finalidad de esta prueba era ver cómo afectaba la existencia de más de un dispositivo final conectado a la puerta de enlace a los principales parámetros (SNR, RSSI y errores). La segunda prueba consiste en, con solo uno de los dos nodos, ver hasta qué distancia llega la cobertura ofrecida por el *gateway*.

Los dispositivos finales empleados eran dispositivos encargados de la medición de temperatura y humedad, ambos con activación OTAA. El primer dispositivo, al que llamaremos *Nodo 1*, es un dispositivo de Clase A alimentado por una batería, con una antena de transmisión similar a la que tiene el *gateway*. El segundo dispositivo, al que llamaremos *Nodo 2*, es un dispositivo de Clase A con soporte de Clase C. Este dispositivo no está alimentado por una batería, sino que se conecta directamente a la corriente.

Además de los resultados obtenidos tras la realización de las pruebas mencionadas, en este apartado también se va a explicar el montaje del *gateway* y se va a realizar un cálculo del balance de enlace, ya que tanto el montaje como el balance de enlace favorecerán una mejor comprensión de los resultados obtenidos.

4.1. Montaje

Antes de explicar del montaje cabe mencionar que este paso, a pesar de que se esté explicando en este apartado, ha de realizarse previo a la implementación del *gateway* para que esta se pueda realizar de forma correcta.

Para empezar con el montaje hay que comprobar que la *Raspberry* no esté conectada a la corriente para poder acoplar el adaptador WM1302 P hat al cabezal de 40 pines de la *Raspberry*. Una vez hecho esto, se inserta el módulo WM1302 en el adaptador y se atornilla para que quede como se muestra en la *Fig. 53*. Adicionalmente, para proporcionar una mayor sujeción, se puede atornillar el adaptador a la *Raspberry*.

Fig. 53 Módulo montado en la raspberry pi.

Después, tenemos que añadir las antenas, tanto la de GPS como la antena para la transmisión. La antena de GPS hay que conectarla al adaptador del módulo, en el conector "ANT", como podemos ver en la *Fig. 54*. La antena de transmisión se conecta al módulo WM1302 en el conector "RFIO". Una vez hecho esto, podemos encender la raspberry, conectar el cable *ethernet* para proporcionar la conexión a la red necesaria, y continuar con la instalación de *ChirpStack*.

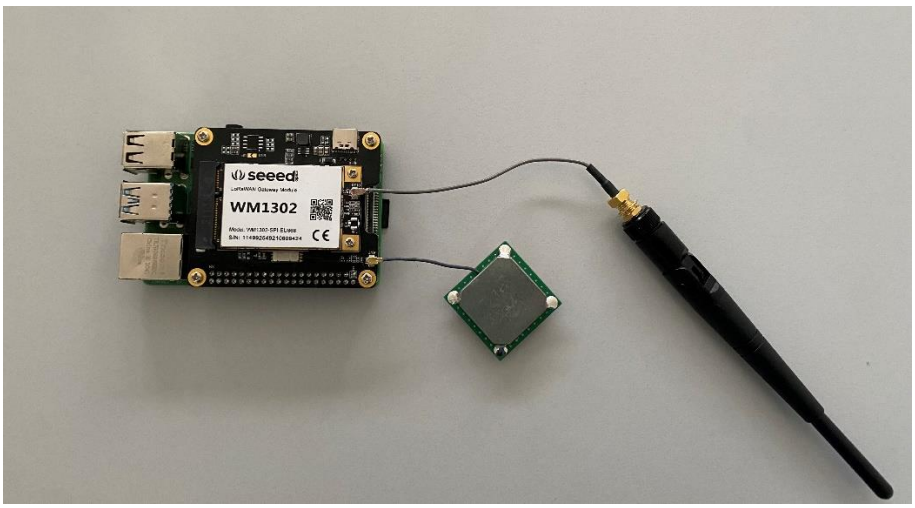

Fig. 54 Módulo WM1302 con las antenas de GPS y transmisión.

El *gateway* se ha colocado en ambas pruebas en una superficie lisa a, aproximadamente, 1 metro del suelo. Como se puede ver en la *Fig. 54*, la antena de transmisión empleada no tiene ningún punto de sujeción que la mantenga erguida, de forma que en ambas pruebas la antena se ha colocado en horizontalmente.

4.2. Primera prueba

Esta prueba consiste en conectar los dos dispositivos, *Nodo 1* y *Nodo 2*, al *gateway* y ver cómo se comporta teniendo que manejar los datos de los dos dispositivos, y después realizar esto mismo, pero utilizando únicamente el *Nodo 1*. La finalidad de esta prueba es, como se ha mencionado brevemente previamente, ver si la puerta de enlace funciona de igual manera teniendo solo un dispositivo o dos, o si, por el contrario, al añadir más de un dispositivo a la red del *gateway* empeoran las comunicaciones de este con los dispositivos finales. Para comprobar el funcionamiento de la puerta de enlace se van a tener en cuenta los siguientes parámetros: SNR, RSSI y errores.

El lugar de realización de la prueba ha sido el interior de un edificio, dentro de un entorno industrial en el que, además de los dispositivos y el *gateway* mencionados, había otros dispositivos y *gateways* comunicándose entre ellos. Entre la puerta de enlace y los dispositivos empleados no había ningún obstáculo, ya que se encontraban en la misma sala. La sala estaba rodeada por paredes, de las cuales una estaba cubierta por ventanas.

La primera parte de la prueba se ha realizado a lo largo de cinco días, y los resultados obtenidos se encuentran reflejados en las tablas 2 y 3. En las *Fig. 55*, *56* y *57* se comparan los valores obtenidos de los tres parámetros seleccionados anteriormente por los dos nodos.

	SNR (en dB)	**RSSI (en dBm)**	**Errores**
Día 1	12,89	-50	12
Día 2	12,93	-39,93	15
Día 3	12,65	-44,41	2
Día 4	12,91	-32,87	6
Día 5	12,77	-38,95	12

Tabla 2. Resultados obtenidos con el Nodo 1

	SNR (en dB)	**RSSI (en dBm)**	**Errores**
Día 1	12,6	-38	0
Día 2	12,88	-34,17	0
Día 3	12,95	-43,08	0
Día 4	13,18	-40,42	5
Día 5	12,96	-41,83	0

Tabla 3. Resultados obtenidos con el Nodo 2

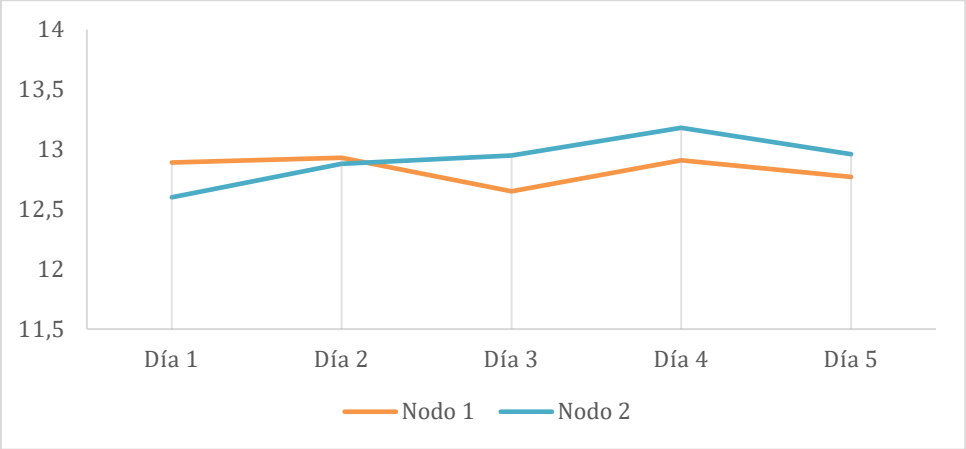

Fig. 55 SNRs obtenidas con Nodo 1 y Nodo 2

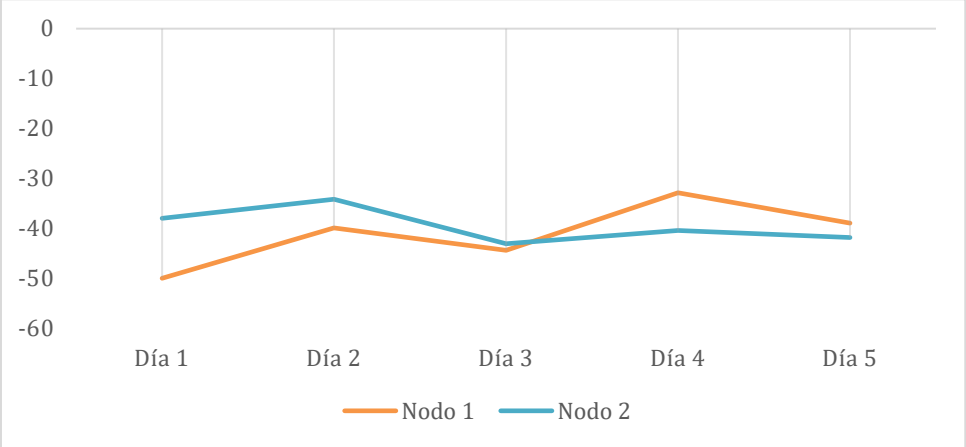

Fig. 56 RSSI Nodo 1 y Nodo 2

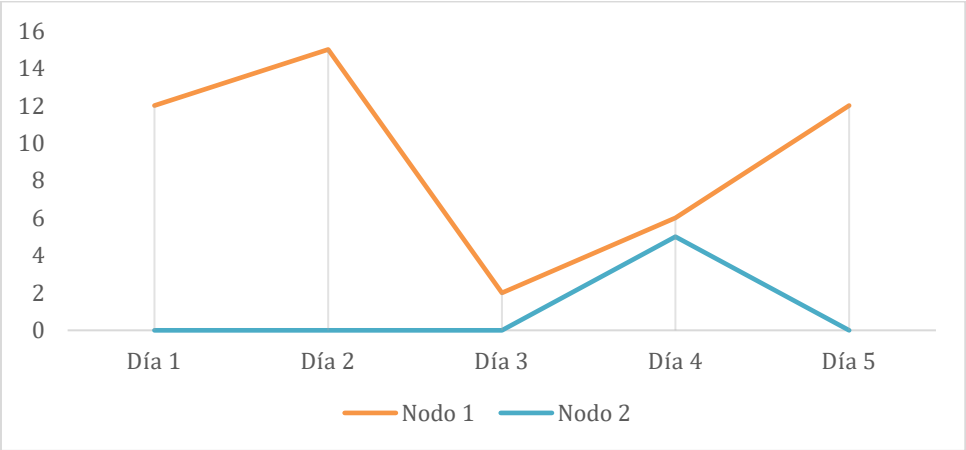

Fig. 57 Número de errores obtenidos Nodo 1 y Nodo 2

La segunda parte de la prueba ha tenido una duración menor que la primera, por lo que los resultados son orientativos, ya que al no tener las mismas condiciones que en la primera parte no representan con exactitud el funcionamiento de la puerta de enlace. Los resultados obtenidos en cuanto a SNR, en torno a los 12,9dB, son muy similares a los obtenidos en la primera parte con el mismo nodo. En cuanto al RSSI, se han obtenido resultados en torno a los -40dBm, que no difieren mucho de los resultados obtenidos anteriormente, que si se realiza el promedio de esos resultados se obtiene un valor de unos -41dBm. En cambio, en los errores sí que se puede ver diferencia ya que en esta ocasión no se obtuvieron errores mientras que al tener ambos nodos funcionando simultáneamente sí.

Con los resultados mencionados anteriormente podemos ver que, exceptuando el número de errores, los otros dos parámetros no cambian excesivamente entre las dos situaciones. Cabe resaltar que esta prueba se ha realizado en un interior en el que, además de los dispositivos empleados en la realización de esta, había otros dispositivos pertenecientes a otras redes que pueden haber afectado a los resultados. Hay que tener en cuenta que, al no haber tenido la misma duración ambas partes de la prueba, no se puede afirmar de forma categórica que el funcionamiento del *gateway* sea el mismo, pero sí se puede confirmar que el funcionamiento sigue siendo lo suficientemente bueno como para gestionar a, al menos, 2 dispositivos.

4.3. Segunda prueba

En esta segunda prueba lo que se quiere comprobar es hasta qué distancia se puede mantener la comunicación entre el *gateway* y el *Nodo 1*, y si esta distancia coincide con la distancia teórica. Este nodo se escogió ya que se alimenta con una batería y, por lo tanto, se puede desplazar a grandes distancias.

La prueba se ha realizado colocando la puerta de enlace en un interior y moviendo el dispositivo final por los alrededores. El *gateway* se ha colocado en una habitación de un chalé adosado, al nivel del suelo, la cual tiene una pared cubierta de ventanas. El chalé se encuentra en una urbanización en una zona más rural, teniendo enfrente un amplio parque rodeado por varias urbanizaciones. Por lo tanto, las señales se verán obstaculizadas por edificios de baja altura.

Durante la realización de la prueba se pudo ver que, al aumentar la distancia entre la puerta de enlace y el *Nodo 1*, tanto la SNR como RSSI disminuían, hasta llegar a valores en torno a los -11 dB y -94 dBm, respectivamente, momento en el que se perdía la comunicación, a unos 500 m de distancia del *gateway*. Para volver a establecer la comunicación entre la puerta de enlace y el dispositivo la distancia tenía que disminuir ligeramente, a unos 485 m, pero los valores de SNR y RSSI no se veían a penas afectados, e incluso en ocasiones empeoraban con respecto al momento en el que se interrumpía la conexión.

4.4. Cálculo del balance de enlace

Como se ha mencionado en el capítulo 2, el balance de enlace es una forma de cuantificar el rendimiento del enlace y la calidad de la transmisión. Para poder realizar este cálculo se van a emplear las fórmulas mencionadas en el apartado 2, por lo que necesitamos saber las distintas variables necesarias.

Primero, hay que realizar el cálculo de las pérdidas del trayecto, para la cual se necesita saber la frecuencia empleada y la distancia. La frecuencia a la que trabaja el *gateway* es de 868 MHz, y para la distancia se va a utilizar la distancia máxima hasta la que se ha comprobado que el *gateway* mantenía la comunicación con el dispositivo final, que es de 500 m o 0,5 km. De esta forma las pérdidas del trayecto obtenidas, cuyo cálculo se muestra en la *Fig. 58*, son de 85,2 dB.

$$\text{Loss (dB)} = 32{,}45 + 20\log_{10}(\text{frecuency in MHz}) + 20\log_{10}(\text{distance in km})$$
$$\text{Loss (dB)} = 32{,}45 + 20\log_{10}(868\ \text{MHz}) + 20\log_{10}(0{,}5\ \text{km}) = 85{,}2\ \text{dB}$$

Fig. 58 Cálculo de pérdidas del trayecto

A continuación, para obtener la potencia recibida es necesario conocer las ganancias de las antenas de transmisión y recepción, las pérdidas de alimentación del transmisor y del receptor, y la potencia del transmisor. Esta última se obtiene de la hoja de especificaciones del módulo WM1302 [31], en la que indica que su valor es de 26 dBm, unos 400mW. Las demás variables se han estimado, teniendo así la ganancia de la antena transmisora - G_{ta} de 3 dB, las pérdidas de alimentación del transmisor - L_{tf} de 1 dB, la ganancia de la antena receptora - G_{ra} de 4 dB, y las pérdidas de alimentación del receptor - L_{rf} de 0,5 dB. En la *Fig. 59* se muestra el cálculo de la potencia recibida.

$$P_r(\text{dBm}) = P_t(\text{dBm}) + G_{ta}(\text{dB}) - L_{tf}(\text{dB}) - L_{path} + G_{ra}(\text{dB}) - L_{rf}(\text{dB})$$
$$P_r(\text{dBm}) = 26\text{dBm} + 3\text{dB} - 1\text{dB} - 85{,}2\text{dB} + 4\text{dB} - 0{,}5\ \text{dB} = -54\text{dBm}$$

Fig. 59 Cálculo de la potencia recibida

De esta forma obtenemos que la potencia recibida es de -54 dBm. Si lo comparamos con la sensibilidad del módulo indicada en la hoja de especificaciones [31], -139 dBm, podemos ver que el *gateway* debería de ser capaz de captar la señal, pero en la práctica, como se ha señalado en la segunda prueba, esto no es lo que ocurría.

Hay que tener en cuenta que este balance de enlace, por un lado, es una estimación, y, por otro lado, no se han tenido en cuenta las pérdidas por interferencias o las perdidas por despolarización.

Mientras que en la primera prueba esto no afecta o, por lo menos, de forma notoria, ya que se ha realizado en un interior en la que los dispositivos estaban estáticos, la segunda prueba se ha realizado en una zona en la que había edificios en los que las señales de radio han podido rebotar de forma que se han podido producir interferencias que contribuyan al deterioro de la señal. Además, cabe señalar, que la puerta de enlace se encontraba en un interior, a diferencia del nodo, lo cual, añade más obstáculos a la señal y contribuye, una vez más, al deterioro de la señal.

El montaje del *gateway* también puede afectar de forma negativa ya que, como se ha podido ver en el apartado anterior, el *gateway* no se ha metido en ningún tipo de estructura que pueda proporcionar sujeción o firmeza a la antena, lo cual se une a que el nodo con el que se ha realizado la prueba se encuentre en movimiento, de forma que cada antena puede encontrarse en una posición distinta, lo cual afecta a las pérdidas por despolarización.

5. Conclusiones

5.1. Comparación con *gateways* comerciales

En este apartado se va a realizar una comparación entre algunas de las soluciones comerciales ya existentes con el *gateway* desarrollado para este proyecto con la finalidad de ver cuál sería más rentable.

Primero, vamos a describir algunos de los *gateways* comerciales más populares:

- *Multitech Conduit*: esta puerta de enlace, *Fig. 60*, presenta una solución configurable, manejable y escalable para aplicaciones industriales de IoT. Además, tiene varias opciones de conectividad de red para su plataforma de administración de datos preferida, tales como 4G-LTE, 3G, 2G y Ethernet. [36] Tiene un precio de 851,05€.

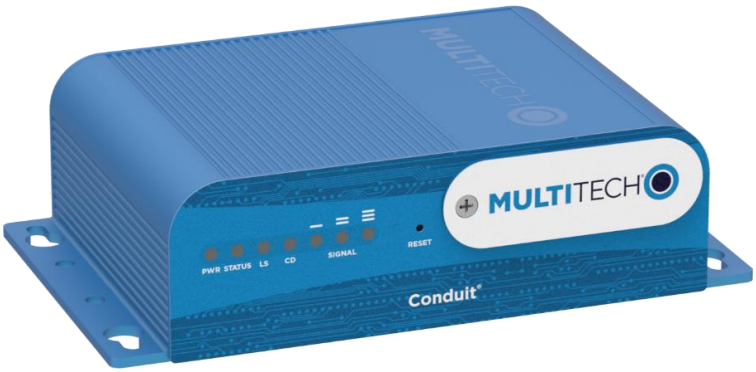

Fig. 60 Multitech Conduit [36]

- *Cisco Wireless Gateway*: esta puerta de enlace, *Fig. 61*, está especialmente diseñada para ser utilizada en exteriores, aunque también es adecuada para aplicaciones en interiores. Ofrece hasta 16 canales de *uplink* y proporciona geolocalización a través de TDOA (*Time Difference of Arrival*, diferencia en el tiempo de llegada) y RSSI (*Received Signal Strength Indication*, indicador de fuerza de la señal recibida). [37] Su precio es de 2242,05€.

Fig. 61 Cisco Wireless Gateway [37]

Anteriormente se han mencionado los distintos componentes empleados y, a continuación, se va a hacer un desglose de los precios de los distintos componentes, así como de las horas invertidas en el desarrollo del *gateway*, para poder comparar el precio de la solución ofrecida en este proyecto con algunas soluciones comerciales ya existentes.

Componentes	**Cantidad**	**Precio**
Raspberry Pi 4 Modelo B (2GB RAM)	1	54,99€
Módulo LoRa SeeedStudio WM1302	1	93,16€
Adaptador para el módulo WM1302	1	17,41€
Antena	1	10,89€
	Precio Total	173,45€

Tabla 4. Precio total de materiales

A este precio hay que sumarle las horas invertidas en el desarrollo, que en total fueron unas 400 horas. Según el Boletín Oficial del Estado en el año 2022, el salario base para un ingeniero es de 1.722€ mensuales, lo que hace un total de 10,76 €/hora. De esta forma, el precio total de la puerta de enlace sería de 4.478,45€.

Teniendo en cuenta únicamente el precio de los componentes empleados para el montaje del *gateway*, se puede apreciar que es mucho más barato que los *gateways* comerciales mencionados anteriormente. En cambio, al añadir las horas dedicadas a su desarrollo esto cambia, ya que pasaría a ser el más caro de las tres opciones. Cabe apuntar que, una vez hecho todo el proceso de desarrollo por primera vez, al implementar un segundo *gateway* la cantidad de horas empleadas se reduciría significativamente, por lo que, aunque a priori puede parecer

una peor opción, en caso de necesitar varios *gateways* sería una buena opción para tener en cuenta.

Hay que tener en cuenta que, al no tener los *gateways* comerciales para poder hacer pruebas y ver todas las funcionalidades que ofrecen, no se puede afirmar de forma rotunda que estos sean mejores que el desarrollado a lo largo de este proyecto y viceversa.

5.2. Conclusiones

Con este proyecto se pretendía obtener un *gateway* o puerta de enlace de bajo coste con el que se pudieran realizar comunicaciones de largo alcance y utilizar un alto número de nodos.

A lo largo de este proyecto se ha conseguido implementar un *gateway* funcional a partir de un módulo comercial con el cual se han podido realizar distintas pruebas a través de las cuales se ha podido comprender mejor el funcionamiento de la tecnología LoRa. Por una parte, durante la realización de pruebas se ha podido comprobar que con al menos dos dispositivos la puerta de enlace era capaz de gestionar los mensajes de ambos, ya que las comunicaciones de ninguno se veían afectadas de forma que no pudieran seguir con su funcionamiento normal, además de ser capaz de recibir mensajes de nodos pertenecientes a otras redes y procesarlos de la forma correspondiente. Por otro lado, en cuanto al alcance del radioenlace, se ha podido comprobar que, a pesar del razonamiento teórico tras la tecnología LoRa, hay muchos factores que afectan de forma negativa a las señales, como la zona de realización de las pruebas, la posición del nodo respecto a la puerta de enlace, o incluso el montaje de los distintos componentes del *gateway*.

Desde el punto de vista económico, se pretendía obtener una puerta de enlace de bajo coste a partir de un módulo comercial. Si solo se tienen en cuenta los gastos relativos a los materiales, se puede afirmar que el *gateway* que se ha desarrollado es una opción mucho más asequible que las principales opciones comerciales disponibles y, por lo tanto, se puede decir que de bajo coste. Si además se tienen en cuenta las horas empleadas en el montaje y desarrollo, esto cambia ya que el coste se ve incrementado de forma significativa. En el caso de que se necesitase una sola puerta de enlace, quizás esta opción no sería la más adecuada, pero si, por el contrario, se necesitasen varias puertas de enlace para distintas aplicaciones, esta podría ser una alternativa a tener en cuenta ya que, una vez realizado todo el proceso de desarrollo por primera vez, en siguientes ocasiones este sería mucho más sencillo y rápido, lo cual no incrementaría excesivamente el coste final y podría resultar más asequible.

6. Líneas futuras

Para finalizar, en este apartado se van a comentar algunas opciones de mejora y posibilidades de continuación tras el desarrollo de este proyecto.

Una de las primeras opciones de mejora de este proyecto sería poder diseñar una carcasa o protector que aporte una estructura más estable y firme para los distintos componentes del *gateway*, así como protección ante agentes externos, lo cual además sería beneficioso también en caso de querer o necesitar utilizarlo en exteriores.

Por otro lado, en este proyecto se han comprobado algunas de las funcionalidades del *gateway*, pero de forma algo limitada por los recursos disponibles, por lo que sería interesante ampliar en este sentido. Para empezar, sería interesante poder realizar pruebas con más nodos de forma simultánea para poder comprobar de forma más realista la capacidad de la puerta de enlace ya que, por lo general, las puertas de enlace tienen que dar cobertura a varios dispositivos finales. Otro aspecto en que se podría mejorar serían las pruebas realizadas, por un lado, sería interesante realizar pruebas en varias ubicaciones diferentes y poder comparar los resultados obtenidos y, por otro lado, también podría resultar interesante realizar estas mismas pruebas añadiendo nodos, como se ha mencionado antes, para ver cómo afecta el número de nodos en estas situaciones.

7. Anexos

Anexo 1. Definiciones

En este anexo se definen de forma más extensa los principales parámetros empleados en la activación de los dispositivos LoRaWAN.

·*JoinEUI*: es un identificador de aplicación global que utiliza el protocolo de direccionamiento IEEE EUI64 que identifica de forma exclusiva el servidor de unión que puede ayudar en el procesamiento del procedimiento de unión y la obtención de las claves de sesión. [27] En anteriores versiones se denominaba *AppEUI*. [29]
En los dispositivos OTAA, el *JoinEUI* debe de estar almacenado en el dispositivo antes de que se realice el proceso de unión. En el caso de dispositivos que sólo permiten ABP el *JoinEUI* no es necesario.

·*DevEUI*: es un identificador global de dispositivo final que utiliza el protocolo de direccionamiento IEEE EUI64 que identifica de forma única al dispositivo. El *DevEUI* es el identificador recomendado por los servidores de red, independiente del procedimiento de activación utilizado, para identificar un dispositivo en itinerancia a través de redes.
Para dispositivos OTAA, el *DevEUI* debe de estar almacenado en el dispositivo antes de que se realice el proceso de unión. En el caso de dispositivos que sólo permiten ABP, no es necesario que el *DevEUI* esté almacenado en el dispositivo, pero se recomienda que si se haga. [27]

·*AppKey* y *NwkKey*: son claves raíz AES-128 (*Advanced Encryption Standard*) de 128 bits, específicas de cada dispositivo final que se le asignan durante la fabricación. Cuando un dispositivo se une a una red a través de activación *over-the-air*, la clave *AppKey* se utiliza para obtener la clave de sesión *AppSKey*, y la clave *NwkKey* se utiliza para obtener las claves de sesión de red *FNwkSIntKey, SNwkSIntKey* y *NwkSEncKey*. El dispositivo y el servidor deben almacenar la misma *AppKey*.
Para dispositivos que quieran utilizar activación *over-the-air*, estas claves deben almacenarse en el mismo. Para dispositivos que sólo permiten ABP no son necesarias. [27]

·*FNwkSIntKey*: clave de sesión de red específica para el dispositivo final. El dispositivo la utiliza para calcular el MIC (*Message Integrity Code*), similar al *checksum* pero que también evita la manipulación intencional de los mensajes [30], completa o parcialmente, de todos los mensajes de *uplink* para asegurar la integridad de los datos. Esta clave debe de almacenarse de forma que se prevenga su extracción y reutilización por parte de actores externos. [27]

·*SNwkSIntKey*: clave de sesión de red específica para el dispositivo final. El dispositivo la utiliza para verificar el MIC de todos los mensajes de *downlink* para asegurar la integridad

de los datos y calcular el MIC de la mitad de los mensajes de *uplink*. Esta clave debe de almacenarse de forma que se prevenga su extracción y reutilización por parte de actores externos. [27]

·*NwkSEncKey*: clave de sesión de red específica para el dispositivo final. Se utiliza para encriptar y desencriptar comandos MAC de *uplink* y *downlink*. Esta clave debe de almacenarse de forma que se prevenga su extracción y reutilización por parte de actores externos. [27]

·*DevAddr*: consta de 32 bits e identifica al dispositivo final dentro de la red actual. El servidor de red asigna al dispositivo el *DevAddr*.

·*AppSKey*: es una clave de sesión de aplicación específica para el dispositivo final. La usan tanto el dispositivo final como el servidor de aplicación para encriptar y desencriptar la carga útil en mensajes de datos específicos de la aplicación. Esta clave debe de almacenarse de forma que se prevenga su extracción y reutilización por parte de actores externos.

·*JoinNonce*: es un contador de 24 bits específico de cada dispositivo, que nunca se repite, proporcionado por el servidor de unión y el dispositivo final lo usa para obtener las claves de sesión de red y la clave *AppSKey*. Se incrementa con cada *join-request*. En versiones anteriores se conocía como *AppNonce*.

·*DevNonce*: es un contador de 16 bits que comienza en 0 cuando el dispositivo se enciende, y se incrementa con cada *join-request*. Este valor no se puede utilizar nunca para un mismo *JoinEUI*. Para cada dispositivo final, el servidor de red supervisa los últimos valores del *DevNonce* usados por el dispositivo, e ignora los *join-requests* si el *DevNonce* no se ha incrementado.

Anexo 2. Implementación *gateway*

En este anexo se va a explicar de forma más extensa la implementación del *gateway*, con todos los comandos y código correspondientes, además de explicar los distintos problemas que han ido apareciendo durante la misma, y las soluciones llevadas a cabo en cada caso.

Para poder empezar con la implementación, hay que asegurarse de que la *Raspberry* esté apagada, y se extrae la tarjeta SD. A continuación, hay que descargarse la imagen "Raspberry Pi OS with desktop and recommended software", obtenida de la página oficial de *Raspberry*, e instalarla en la tarjeta SD. Antes de introducirla en la *Raspberry*, para poder conectar nuestro equipo, un ordenador portátil, a la *Raspberry* a través de SSH en la línea de comandos, hay que habilitarlo de forma manual, ya que este servicio está deshabilitado por defecto para evitar que el acceso externo sea demasiado fácil. Hay distintas formas para habilitar SSH, de las cuales la más sencilla es crear un archivo SSH sin extensión en el directorio de arranque. Otras opciones para habilitar SSH: activar el servidor SSH en el escritorio, conectando la Raspberry a un monitor, un ratón y un teclado … Una vez hecho esto, se introduce la tarjeta SD en la *Raspberry*.

Ahora hay que conectar nuestro equipo con la *Raspberry* mediante SSH, para lo cual necesitamos saber la dirección IP de la Raspberry, esto se hace a través de la línea de comandos introduciendo lo siguiente: `ping raspberrypi.local`. Hay que tener en cuenta que para poder utilizar este comando la Raspberry y el ordenador que estemos utilizando tienen que estar conectados a la misma red. Sabiendo la dirección IP de la *Raspberry*, utilizamos el comando `ssh pi@[direccion IP]` para conectar ambos equipos.

Al conectar con la *Raspberry*, se pide introducir la contraseña, que por defecto es 'raspberry', y cambiarla con el comando `passwd`. Una vez hecho esto, hay que seguir los pasos indicados en la página de *ChirpStack*. Primero hay que instalar los siguientes servicios para que puedan funcionar los servicios específicos de *ChirpStack*:

-*MQTT broker* - Un protocolo de publicación / suscripción que permite a los usuarios publicar información sobre temas a los que otros pueden suscribirse. Una implementación popular del protocolo MQTT es *Mosquitto*.
-*Redis* - motor de base de datos en memoria, basado en el almacenamiento en tablas de hashes pero que opcionalmente puede ser usada como una base de datos durable o persistente.
-*PostgreSQL* - sistema de gestión de bases de datos relacional orientado a objetos y de código abierto.

Esto se hace con siguientes comandos:

```
sudo apt install mosquitto mosquitto-clients
sudo apt install redis-server redis-tools
sudo apt install postgresql
```

Una vez tenemos esto instalado, hay que configurarlo. Empezaremos por configurar las bases de datos y usuarios de PostgreSQL. Con el comando `sudo -u postgres psql` accedemos a *postgresql*, y desde ahí podemos empezar la configuración. Primero creamos los usuarios, uno para el *Application Server* y otro para *Network Server*, con sus respectivas contraseñas, y luego creamos las bases de datos correspondientes a los usuarios anteriormente creados. Una vez creados, entramos en la base de datos de *Application Server* y habilitamos las extensiones *pg_trgm* y *hstore*. La extensión *pg_trgm* proporciona funciones y operadores para determinar la similitud de texto alfanumérico en función de la coincidencia de trigramas, así como clases de operadores de índice que admiten la búsqueda rápida de cadenas similares. *Hstore* es una extensión de *PostgreSQL* que implementa el tipo de datos *hstore*, tipo de datos para almacenar conjuntos de parejas clave/valor dentro de un único valor de *PostgreSQL*. En la *Fig. 62* aparecen los comandos empleados.

```
-- set up the users and the passwords
-- (note that it is important to use single quotes and a semicolon at the end!)
create role chirpstack_as with login password 'dbpassword';
create role chirpstack_ns with login password 'dbpassword';

-- create the database for the servers
create database chirpstack_as with owner chirpstack_as;
create database chirpstack_ns with owner chirpstack_ns;

-- change to the ChirpStack Application Server database
\c chirpstack_as

-- enable the pq_trgm and hstore extensions
-- (this is needed to facilitate the search feature)
create extension pg_trgm;
-- (this is needed to store additional k/v meta-data)
create extension hstore;

-- exit psql
\q
```

Fig. 62 Creación de los usuarios y bases de datos de PostgreSQL [34]

Después hay que configurar el repositorio del software de *ChirpStack*. *ChirpStack* proporciona un repositorio que es compatible con el sistema de paquetes *apt* de *Ubuntu*. Primero, hay que asegurarse de que `dirmngr` y `apt-transport-https` estén instalados con los siguientes comandos: `sudo apt install apt-transport-https dirmngr`. A continuación, hay que configurar la clave para este nuevo repositorio: `sudo apt-key adv --keyserver keyserver.ubuntu.com --recv-keys 1CE2AFD36DBCCA00`. Y agregar el repositorio a la lista de repositorios creando un nuevo archivo: `sudo echo "deb https://artifacts.chirpstack.io/packages/3.x/deb stable main" | sudo tee /etc/apt/sources.list.d/chirpstack.list`. Por último, se actualizas la caché del paquete *apt*: `sudo apt update`.

Ahora, hay que instalar *ChirpStack Gateway Bridge* con el siguiente comando: `sudo apt install chirpstack-gateway-bridge`. El archivo de configuración se encuentra en la ruta /etc/chirpstack-gateway-bridge/ chirpstack-gateway-bridge.toml. Ahora instalamos el *ChirpStack Network Server* con este comando: `sudo apt install chirpstack-network-server`. El archivo de configuración se encuentra en la ruta /etc/chirpstack-network-server/ chirpstack-network-server.toml, y este archivo debe actualizarse para que coincida con la configuración de la base de datos y la banda. Por último, hay que instalar el *ChirpStack Application Server*: `sudo apt install chirpstack-application-server`. El archivo de configuración se encuentra en la ruta /etc/chirpstack-application-server/ chirpstack-application-server.toml, y este archivo también debe ser actualizado para que coincida con la configuración de la base de datos. Dentro del archivo de configuración hay que reemplazar el *jwt_secret* por uno seguro, esto lo hice de la forma que se indicaba en la página de Chirpstack, utilizando el comando: `openssl rand -base64 32`. Así se genera un *jwt_secret* aleatorio, una vez generado se copia lo obtenido y se pone en el archivo de configuración sustituyendo lo que hubiera anteriormente. En la *Fig. 63* se muestra un ejemplo de clave *jwt_secret*.

```
# JWT secret used for api authentication / authorization
# You could generate this by executing 'openssl rand -base64 32' for example
jwt_secret="oQcCYJXWnZo69v+Sm4/CaB4oM0L2prqczJ+Y7mGJPxU="
```

Fig. 63 Ejemplo de clave jwt_secret

Al estar los archivos de configuración en las direcciones `/etc/chirpstack-application-server/chirpstack-application-server.toml`, `/etc/chirpstck-network-server/chirpstack-network-server.toml` y `/etc/chirpstack-gateway-bridge/chirpstack-gateway-bridge.toml`, para poder acceder a ellos hay que ejecutar `sudo su`, que permite a los usuarios ejecutar programas con los privilegios de seguridad de otro usuario, y después accedemos al directorio correspondiente con `cd`. Una vez ahí, para editar los archivos utilizamos el comando `nano`. Dentro del archivo de configuración del *Network Server* y del *Application Server* hay que cambiar, en ambos archivos no solo en uno de ellos, la línea: `dsn="postgres://localhost/chirpstack_ns?sslmode=disable"` por:

-dsn="postgres://chirpstack_ns:dbpassword@localhost/chirpstack_ns?sslmode=disable", en el caso del *Network Server*.

-dsn="postgres://chirpstack_as:dbpassword@localhost/chirpstack_as?sslmode=disable", en el caso del *Application Server*.

Los cambios en los archivos de configuración mencionados previamente también se realizarían de esta forma. Una vez realizados los cambios necesarios, utilizamos el comando `exit`.

Ahora que ya están instalados todos los servicios, hay que iniciarlos y comprobar que no haya errores en los archivos de configuración:

```
sudo systemctl start chirpstack-application-server
sudo systemctl enable chirpstack-application-server
sudo journalctl -f -n 100 -u chirpstack-application-server

sudo systemctl start chirpstack-network-server
sudo systemctl enable chirpstack-network-server
sudo journalctl -f -n 100 -u chirpstack-network-server

sudo systemctl start chirpstack-gateway-bridge
sudo systemctl enable chirpstack-gateway-bridge
sudo journalctl -f -n 100 -u chirpstack-gateway-bridge
```

El comando `sudo systemctl start [servicio]` inicia el servicio, de forma análoga se podría utilizar el comando `sudo systemctl stop [servicio]` para parar el servicio y para reiniciarlo el comando a utilizar es `sudo systemctl restart [servicio]`, este comando se utilizará en caso de que hiciéramos cambios en los archivos de configuración posteriormente mencionados. El comando `sudo systemctl enable [servicio]` se utiliza para que se inicie el servicio en el arranque. El comando `sudo journalctl -f -n 100 -u [servicio]` imprime el resultado del registro del servicio.

A continuación, hay que instalar el módulo WM1302, previamente montado, siguiendo la guía del módulo WM1302 [32]. Primero hay que comprobar que en el archivo de configuración del *Gateway bridge*, `chirpstack-gateway-bridge.toml`, este configurado el tipo de *backend* como se indica en la *Fig. 64*, ya que los otros tipos disponibles en *ChirpStack* no son compatibles con el módulo WM1302.

```
# Gateway backend configuration.
[backend]
# Backend type.
type="semtech_udp"
```

Fig. 64 Tipo de backend compatible con el módulo WM1302

El módulo WM1302 se comunica con la *Raspberry* a través de SPI e I2C, pero estas dos interfaces no están habilitadas de forma predeterminada, por lo que hay que habilitarlas antes de usar el módulo WM1302. Para ello, se ejecuta el comando `sudo raspi-config` con el que se abre la ventana que se muestra en la *Fig. 65*.

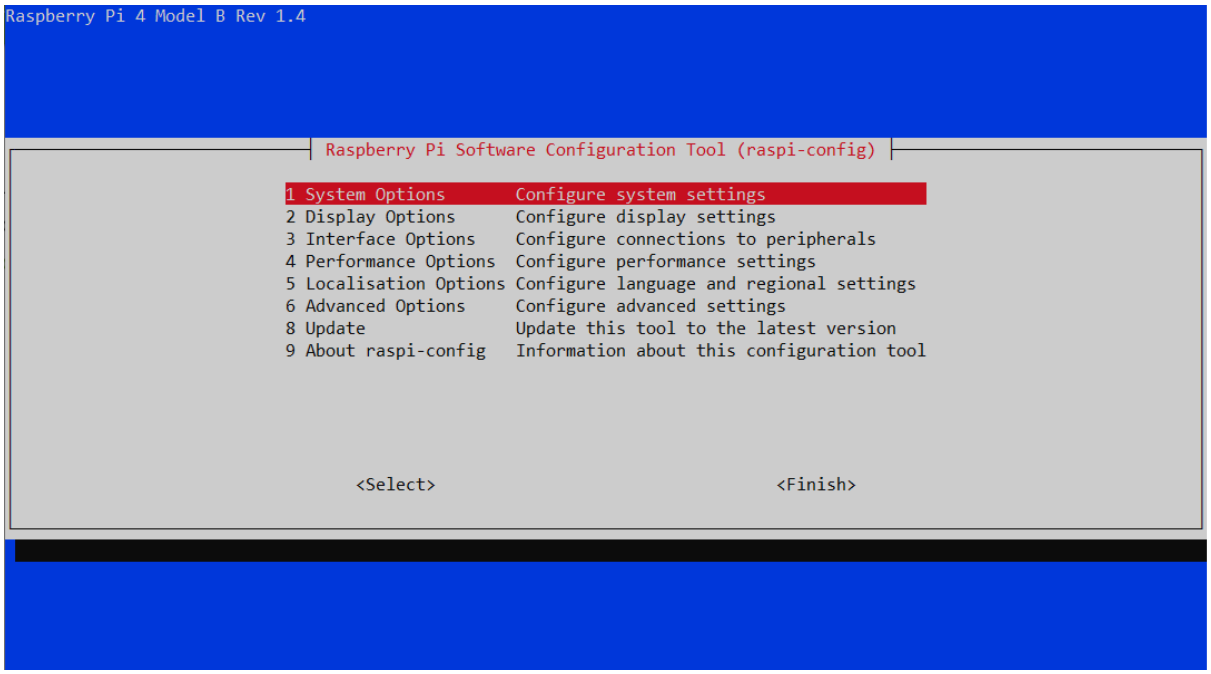

Fig. 65 Configuración de la Raspberry

Aquí se selecciona la opción "*Interface Options*", dentro de esta opción seleccionamos "*SPI*", y luego "*Yes*" para habilitarlo, seleccionamos "*I2C*" y luego "*Yes*" para habilitarlo. Por último, seleccionamos "*Serial Port*" y luego "*No*" para "*Would you like a login shell...*" y "*Yes*" para *"Would you like the serial port hardware...*". Una vez hecho esto, se reinicia la *Raspberry* para asegurar que la configuración funcione.

Ahora hay que descargar `sx1302_hal`, biblioteca y programas para *SX1302 LoRa Gateway*, de GitHub, con el comando `git clone https://github.com/Lora-net/sx1302_hal`. Si GitHub no estuviera instalado habría que ejecutar los siguientes comandos previamente para instalarlo: `sudo apt update` y `sudo apt install -y git`. Una vez hecho esto, hay que ponerse en la carpeta de `sx1302_hal` y compilar todo, con los comandos `cd sx1302_hal` y `make`. A continuación, hay que modificar el pin de reset de los chips SX1302 y SX1262 dentro del archivo `reset_lgw.sh`, el cual se encuentra en la carpeta *tools*, por lo que para editarlo se ejecuta la siguiente línea: `nano tools/reset_lgw.sh`. Dentro del archivo, hay que modificar la parte que se muestra en la *Fig. 66* para que quede de como en la *Fig. 67*. Con este cambio lo que se consigue es cambiar el pin de reset del SX1302 del 23 al 25 y el pin de reset del SX1262 del 22 al 17, para hacer que coincidan los "pineados" de la *Raspberry* y el módulo WM1302.

```
# GPIO mapping has to be adapted with HW
#

SX1302_RESET_PIN=23     # SX1302 reset
SX1302_POWER_EN_PIN=18  # SX1302 power enable
SX1262_RESET_PIN=22     # SX1262 reset (LBT / Spectral Scan)
AD5338R_RESET_PIN=13    # AD5338R reset (full-duplex CN490 reference design)
```

Fig. 66 Archivo reset_lgw.sh antes de los cambios del "pineado"

```
# GPIO mapping has to be adapted with HW
#

SX1302_RESET_PIN=25     # SX1302 reset
SX1302_POWER_EN_PIN=18  # SX1302 power enable
SX1262_RESET_PIN=17     # SX1262 reset (LBT / Spectral Scan)
AD5338R_RESET_PIN=13    # AD5338R reset (full-duplex CN490 reference design)
```

Fig. 67 Archivo reset_lgw.sh tras los cambios del "pineado"

Además de este cambio, dentro de este mismo archivo hay que modificar otra parte del código, que aparece en la *Fig. 68*, para que no haya errores al terminar la ejecución del *packet forwarder*. Tras la modificación, el código debe de quedar como en la *Fig. 69*.

```
  # write output for SX1302 CoreCell power_enable and reset
  echo "1" > /sys/class/gpio/gpio$SX1302_POWER_EN_PIN/value; WAIT_GPIO

  echo "1" > /sys/class/gpio/gpio$SX1302_RESET_PIN/value; WAIT_GPIO
  echo "0" > /sys/class/gpio/gpio$SX1302_RESET_PIN/value; WAIT_GPIO

  echo "0" > /sys/class/gpio/gpio$SX1262_RESET_PIN/value; WAIT_GPIO
  echo "1" > /sys/class/gpio/gpio$SX1262_RESET_PIN/value; WAIT_GPIO

  echo "0" > /sys/class/gpio/gpio$AD5338R_RESET_PIN/value; WAIT_GPIO
  echo "1" > /sys/class/gpio/gpio$AD5338R_RESET_PIN/value; WAIT_GPIO
```

Fig. 68 Parte del código del archivo reset_lgw.sh a modificar

```
# write output for SX1302 CoreCell power_enable and reset
echo "1" > /sys/class/gpio/gpio$SX1302_POWER_EN_PIN/value; WAIT_GPIO

# echo "1" > /sys/class/gpio/gpio$SX1302_RESET_PIN/value; WAIT_GPIO
# echo "0" > /sys/class/gpio/gpio$SX1302_RESET_PIN/value; WAIT_GPIO

# echo "0" > /sys/class/gpio/gpio$SX1262_RESET_PIN/value; WAIT_GPIO
# echo "1" > /sys/class/gpio/gpio$SX1262_RESET_PIN/value; WAIT_GPIO

echo "0" > /sys/class/gpio/gpio$AD5338R_RESET_PIN/value; WAIT_GPIO
echo "1" > /sys/class/gpio/gpio$AD5338R_RESET_PIN/value; WAIT_GPIO
```

Fig. 69 Archivo reset_lgw.sh tras los cambios

Una vez hechos estos cambios, hay que copiar este archivo a la carpeta packet_forwarder, con los comandos cp tools/reset_lgw.sh packet_forwarder/ y cd packet_forwarder, para poder ejecutar lora_pkt_fwd. Para ejecutar lora_pkt_fwd hay que seleccionar un archivo de configuración global_conf.json.sx1250.xxxx.

```
pi@raspberrypi:~/sx1302_hal/packet_forwarder $ ls
global_conf.json.sx1250.AS923.USB          inc
global_conf.json.sx1250.CN490              lora_pkt_fwd
global_conf.json.sx1250.CN490.USB          Makefile
global_conf.json.sx1250.EU868              obj
global_conf.json.sx1250.EU868.USB          PROTOCOL.md
global_conf.json.sx1250.US915              readme.md
global_conf.json.sx1250.US915.USB          reset_lgw.sh
global_conf.json.sx1255.CN490.full-duplex  src
global_conf.json.sx1257.EU868
```

Fig. 70 Archivos de configuración disponible según la frecuencia empleada por el módulo

En este caso el archivo a utilizar, dentro de las opciones que se muestran en la *Fig. 67*, es global_conf.json.sx1250.EU868, dentro de este hay que modificar los parámetros *gateway_ID*, *server_address*, *serv_port_up* y *serv_port_down*, *Fig. 71*, para que se reenvíen los paquetes al servidor de manera correcta.

```
    "gateway_conf": {
        "gateway_ID": "AA555A0000000000",
        /* change with default server address/ports */
        "server_address": "localhost",
        "serv_port_up": 1700,
        "serv_port_down": 1700,
```

Fig. 71 Parámetros de configuración

Una vez completados estos pasos, se puede arrancar el *packet forwarder*, el cual se encarga de gestionar los paquetes recibidos y reenviarlos hacia el servidor, lo cual es imprescindible para que la puerta de enlace pueda funcionar. Esto se hace con el comando `./lora_pkt_fwd -c global_conf.json.sx1250.EU868`. Al hacer esto, el gateway ya funciona, pero al parar el proceso y volver a arrancarlo, saldrán los errores que se pueden ver en la *Fig. 72*.

```
Note: chip version is 0x10 (v1.0)
ERROR: Failed to set SX1250_0 in STANDBY_RC mode
ERROR: failed to setup radio 0
ERROR: [main] failed to start the concentrator
```

Fig. 72 Errores producidos al arrancar el packet forwarder tras haberlo parado

Esto hace que, si se quiere volver a iniciar el *gateway*, haya que reiniciar la *Raspberry*, desconectándola de la corriente y volviéndola a conectar, y después ejecutar el comando de nuevo. Para solucionar esto hay que añadir en el archivo `reset_lgw.sh` una nueva función, que hemos llamado "reset_fin", *Fig. 73*, que coincide con la función que anteriormente hemos comentado.

```
reset_fin() {
    echo "CoreCell reset through GPIO$SX1302_RESET_PIN..."
    echo "SX1262 reset through GPIO$SX1302_RESET_PIN..."
    echo "CoreCell power enable through GPIO$SX1302_POWER_EN_PIN..."
    echo "CoreCell ADC reset through GPIO$AD5338R_RESET_PIN..."

    # write output for SX1302 CoreCell power_enable and reset
    echo "1" > /sys/class/gpio/gpio$SX1302_POWER_EN_PIN/value; WAIT_GPIO

    echo "1" > /sys/class/gpio/gpio$SX1302_RESET_PIN/value; WAIT_GPIO
    echo "0" > /sys/class/gpio/gpio$SX1302_RESET_PIN/value; WAIT_GPIO

    echo "0" > /sys/class/gpio/gpio$SX1262_RESET_PIN/value; WAIT_GPIO
    echo "1" > /sys/class/gpio/gpio$SX1262_RESET_PIN/value; WAIT_GPIO

    echo "0" > /sys/class/gpio/gpio$AD5338R_RESET_PIN/value; WAIT_GPIO
    echo "1" > /sys/class/gpio/gpio$AD5338R_RESET_PIN/value; WAIT_GPIO
}
```

Fig. 73 Nueva función "reset_fin"

Esta función se utilizará al parar el concentrador, mientras que la otra se utilizará al iniciarlo. Esto hay que reflejarlo al final del archivo haciendo el cambio reflejado en la *Fig. 74*, que consiste en sustituir "reset" por "*reset_fin*" en el apartado de "*stop*".

```
case "$1" in
    start)
    term # just in case
    init
    reset
    ;;
    stop)
    reset
    term
    ;;
    *)
    echo "Usage: $0 {start|stop}"
    exit 1
    ;;
esac
```

```
case "$1" in
    start)
    term # just in case
    init
    reset
    ;;
    stop)
    reset_fin
    term
    ;;
    *)
    echo "Usage: $0 {start|stop}"
    exit 1
    ;;
esac
```

Fig. 74 Cambios en las funciones del archivo reset_lgw.sh

Una vez hecho esto, hay que crear un archivo "*initLora*" en la ruta `/etc/init.d`, ejecutando `sudo nano initLora`, lo cual crea un archivo vacío en el que hay que poner el código reflejado en la *Fig. 75*.

```
#!/bin/sh
if [ -f /tmp/logLora.txt ]; then
    rm /tmp/logLora.txt
fi
cd /home/pi/sx1302_hal/packet_forwarder
./reset_lgw.sh stop >> /tmp/logLora.txt
./lora_pkt_fwd - c global_conf.json.sx1250.EU868 >> /tmp/logLora.txt &
```

Fig. 75 Código del nuevo archivo initLora

Tras esto, ahora hay que darle permiso de ejecución con `sudo chmod +x initLora`. Una vez creado, hay que añadir ese proceso solo en `/etc/rc2.d`, `/etc/rc3.d`, `/etc/rc4.d` y `/etc/rc5.d`, ya que en `/etc/rc0.d`, `/etc/rc1.d` y `/etc/rc6.d` solo hay procesos de "*kill*". Se ejecuta el comando `sudo ln -s /etc/init.d/initLora S02initLora` en cada una de las rutas anteriormente mencionadas. El proceso se añade como "S02" para darle la prioridad más baja, para que no influya en ninguno de los otros procesos.

Una vez completados estos pasos, la puerta de enlace funciona de forma correcta, y se puede continuar con la configuración del *gateway* y, posteriormente, con la configuración de un dispositivo. Todo esto se hace desde la interfaz web del *Application Server* de *ChirpStack* que, una vez instalado y funcionando, se puede acceder a la interfaz web: http://localhost:8080/. Al entrar, habrá que iniciar sesión con las credenciales por defecto:

-Usuario: admin

-Contraseña: admin

Inicialmente, al abrir la página, se mostrarán todos los elementos vacíos. Primero, hay que añadir un *Network Server* o Servidor de Red, *Fig. 76*.

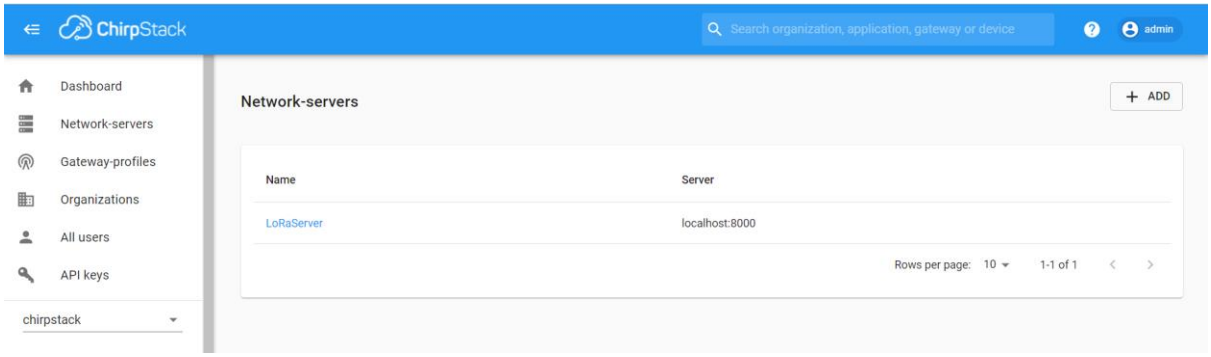

Fig. 76 Servidores de red en la interfaz web de ChirpStack

Se rellenan los campos requeridos, *Fig. 77*, con los parámetros usados en la instalación.

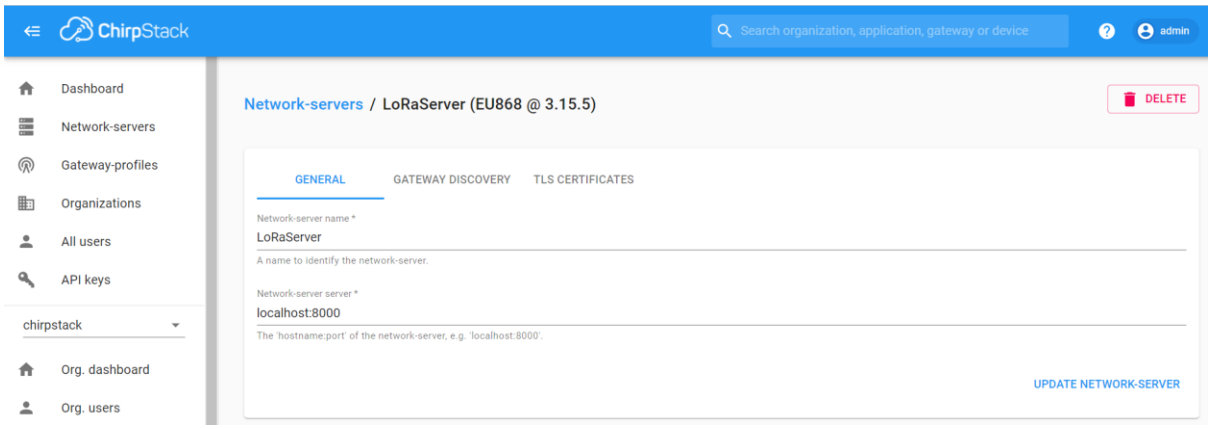

Fig. 77 Campos requeridos para añadir un Servidor de Red

Se requiere disponer de un *gateway profile*, o perfil de gateway, para después poder añadir un *gateway*, *Fig. 78*.

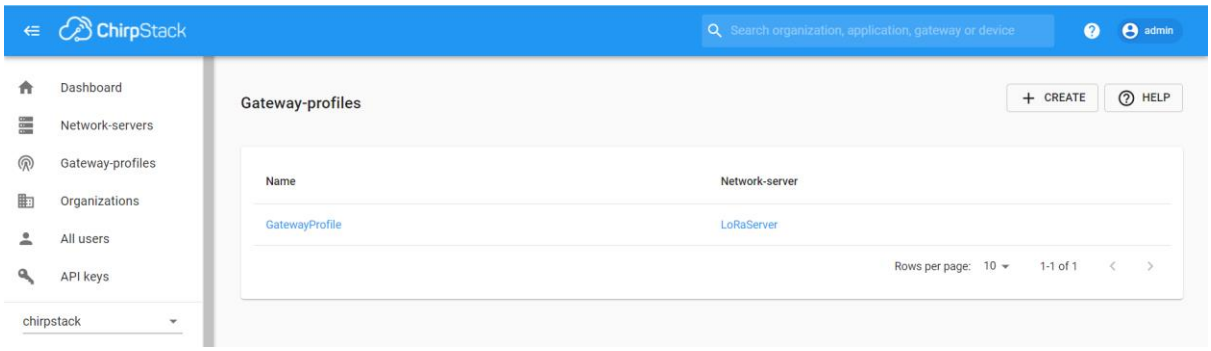

Fig. 78Perfiles de gateway en la interfaz web de ChirpStack

Dentro de este, se indican los canales que usa el *gateway*, el intervalo de tiempo en el que el *gateway* informa de sus estadísticas, y el servidor de red creado anteriormente, *Fig. 79*.

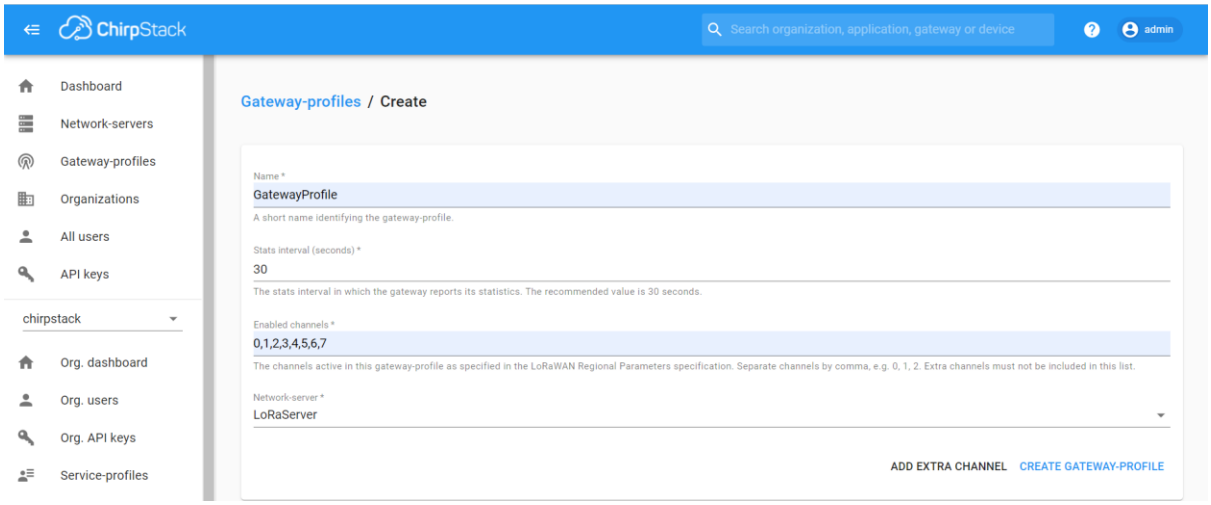

Fig. 79 Campos requeridos para añadir un perfil de gateway

A continuación, es necesario crear un *Service Profile*, o perfil de servicio, para los *gateways*, *Fig. 80*.

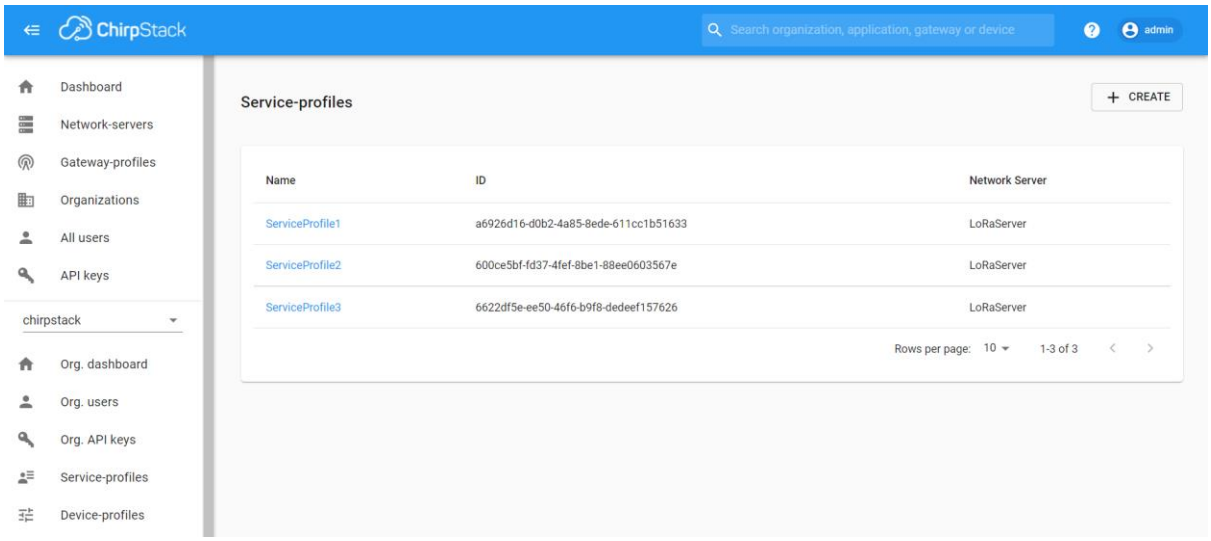

Fig. 80 Perfiles de servicios en la interfaz web de ChirpStack

Dentro de este perfil se indica a que *Network Server* se conectará nuestro *gateway* y parámetros relacionados con la señal, *Fig. 81*.

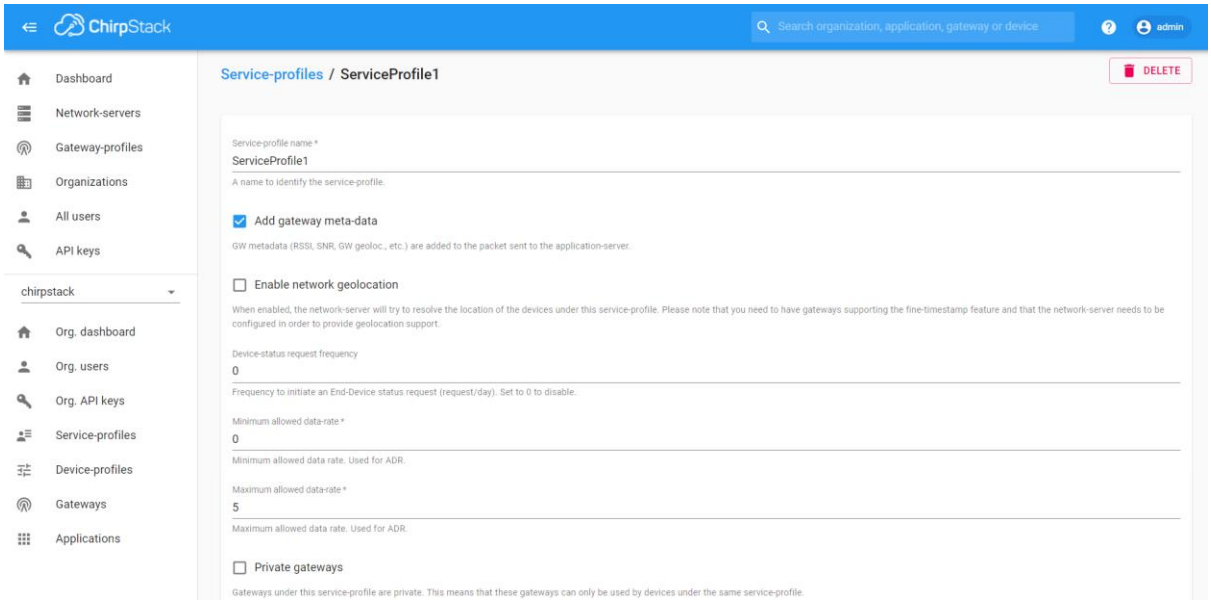

Fig. 81 Campos requeridos para añadir un perfil de servicio

Una vez completados estos pasos, se puede añadir un *gateway*, *Fig. 82*.

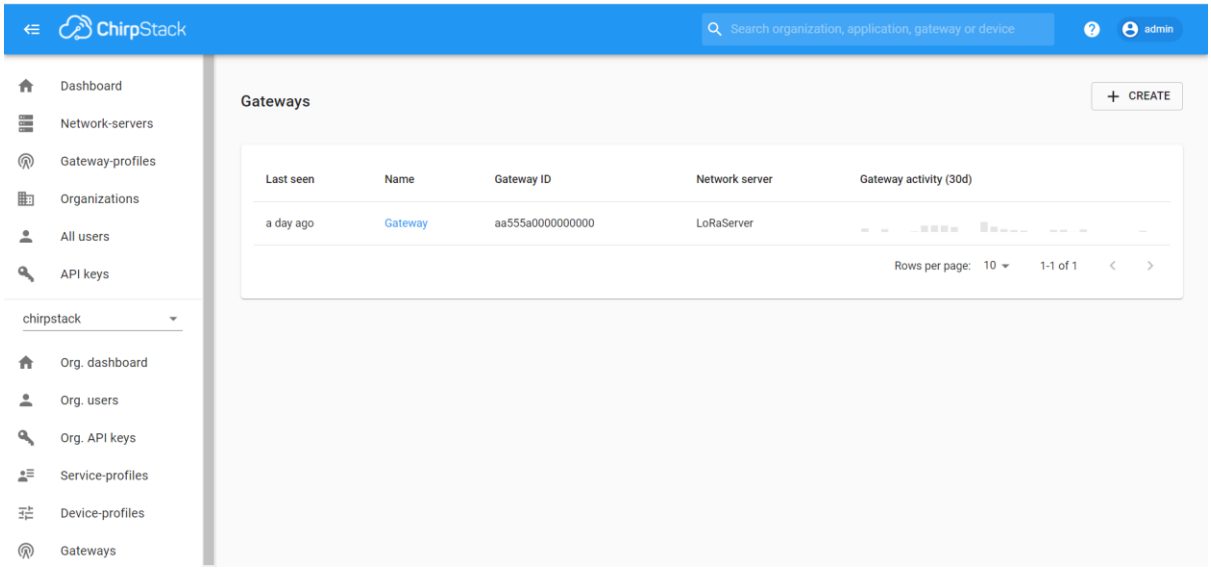

Fig. 82 Gateways en la interfaz web de ChirpStack

Para poder añadir el *gateway* necesitamos el parámetro *gateway_ID*, *Fig. 83*, configurado anteriormente en el archivo de configuración del *packet forwarder*.

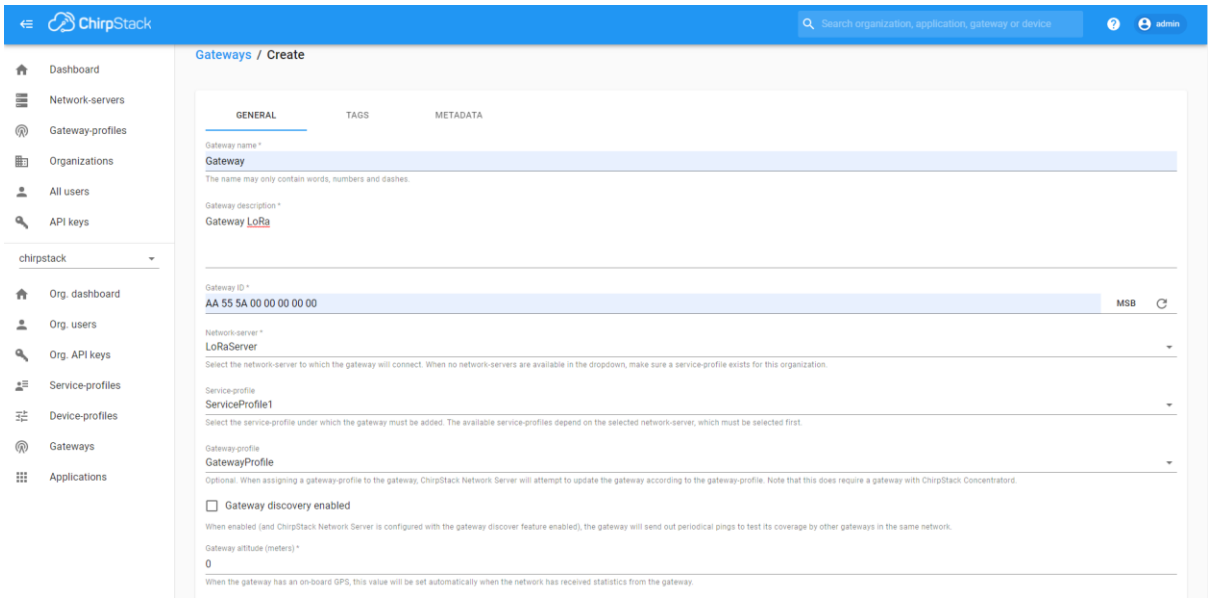

Fig. 83 Campos requeridos para añadir un gateway

Una vez completados estos pasos, hay que comprobar que el *gateway* funciona correctamente. La primera forma, y más sencilla es comprobar que en el apartado *Last seen* del menú de *gateways* ponga *a few seconds ago*, como en la *Fig. 82*, ya que, aunque el *gateway* no esté recibiendo datos, periódicamente envía sus estadísticas, en este caso cada 30 segundos, y cuando recibe estas estadísticas el *Application Server* actualiza el *Last seen*. Otra forma es ver que en el apartado *Dashboard* el *gateway* está activo, como en la *Fig. 84*.

Fig. 84 Dashboard de la interfaz web de ChirpStack

Por último, también se puede comprobar entrando en la página del *gateway*, y una vez ahí abrir la pestaña *Live LoRaWAN Frames*, esto mostrará todas las tramas *LoRaWAN* que recibe y envía el *gateway*. En el caso de tramas recibidas, aquí se pueden ver tramas recibidas de dispositivos que no son los míos y/o que aún no están configurados. Por lo tanto, esta pantalla es útil para validar si el *gateway* puede recibir tramas LoRaWAN y reenviarlas a *ChirpStack*.

Una vez realizada la configuración del *gateway*, se puede configurar un dispositivo, para lo que, igual que al añadir un *gateway*, hay que realizar unos pasos previos. Primero, hay que crear un nuevo *Service Profile*, de forma análoga a como se ha realizado con el *gateway*. A continuación, hay que crear un *Device Profile*, o perfil de dispositivo, que describe la forma de conexión y autenticación de un grupo de dispositivos, *Fig. 85*.

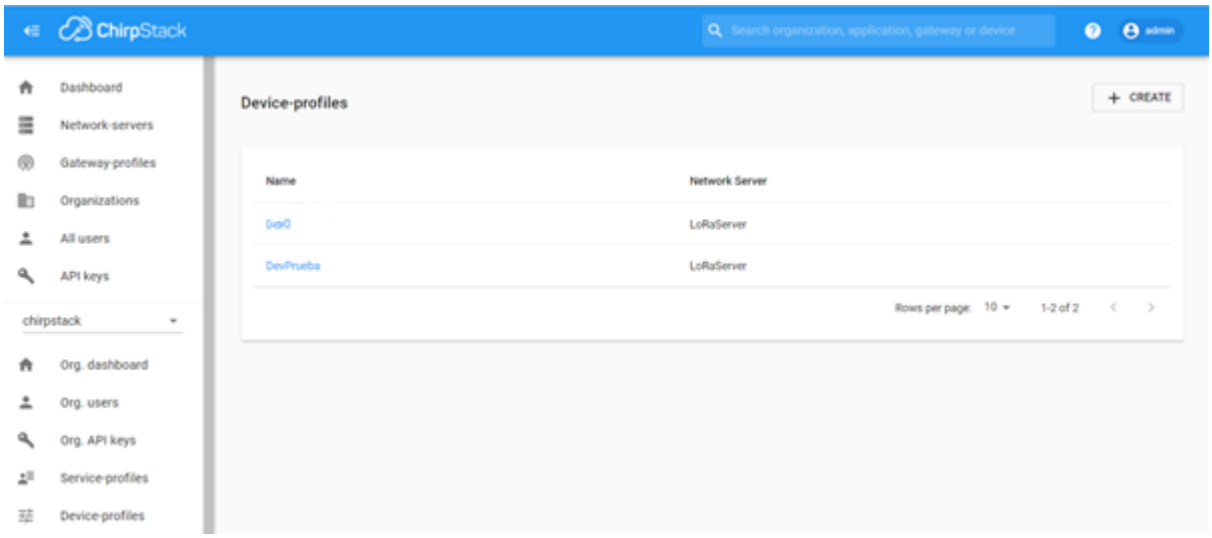

Fig. 85 Perfiles de dispositivo dentro de la interfaz web de ChirpStack

Aquí, *Fig. 86*, es donde se indicará si los dispositivos autentican por medio de OTAA/APB, en este caso la activación es OTAA, si se incluirán dispositivos clase B, clase C y las instrucciones para codificar o decodificar los datos del dispositivo (CODEC).

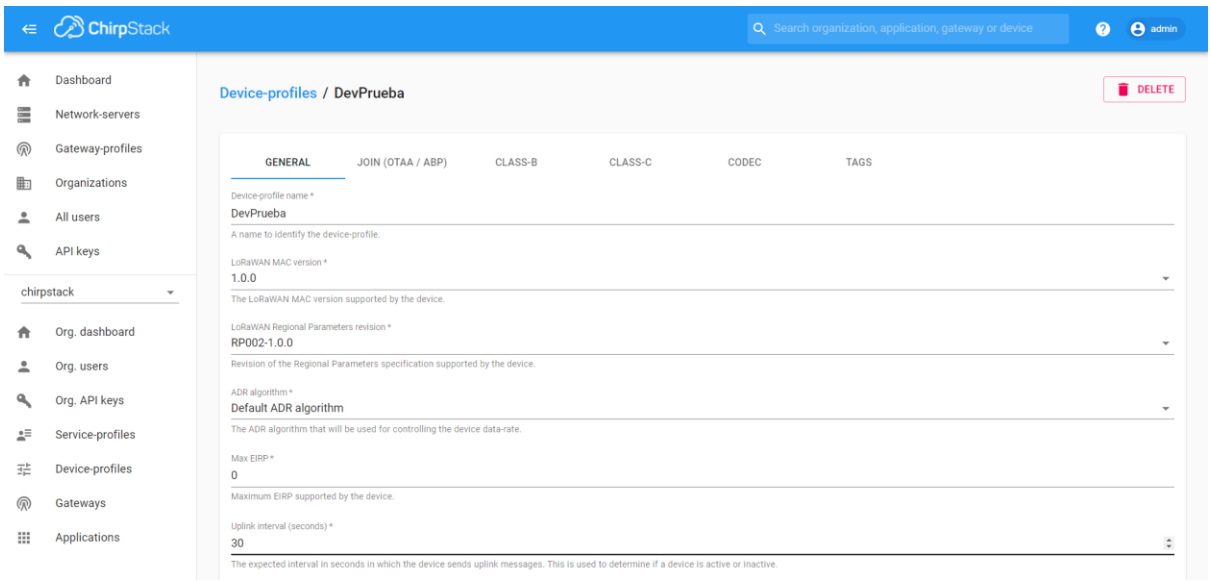

Fig. 86 Campos a rellenar para crear un perfil de dispositivo

Ahora hay que crear una aplicación, *Fig. 87*, que agrupa dispositivos que realizan funciones semejantes, mediante el *Device Profile*. Aquí, *Fig. 88*, solo hay que rellenar el nombre, una pequeña descripción de la función que van a realizar los dispositivos y el *Service Profile*.

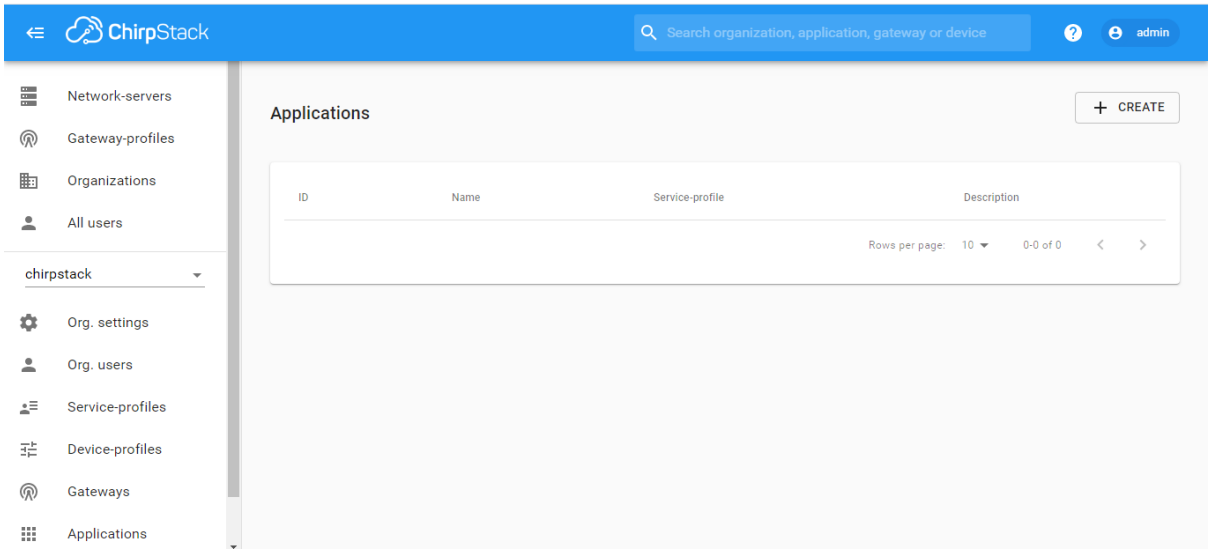

Fig. 87 Aplicaciones en la interfaz web de ChirpStack

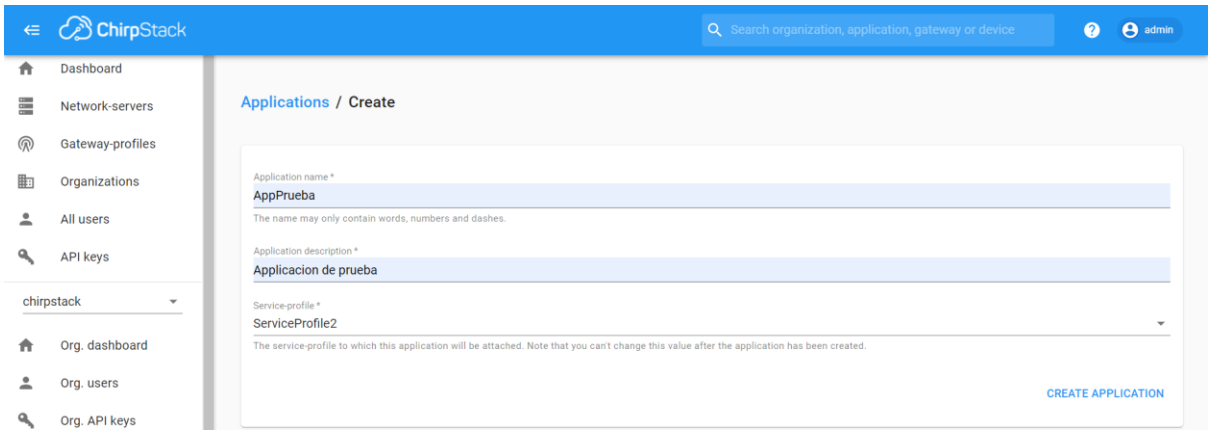

Fig. 88 Campos a rellenar para crear una aplicación

Una vez creada la aplicación, hay que añadir el dispositivo que se quiere usar para esa aplicación, para lo que hay que completar el nombre y la descripción, *Fig. 89*. Lo siguiente es el *device EUI* o *DevEUI*. Al introducirlo hay que tener en cuenta el orden de los bits: MSB o LSB. Al usar conexión OTAA hay que rellenar también el campo de *Application key* o *AppKey*.

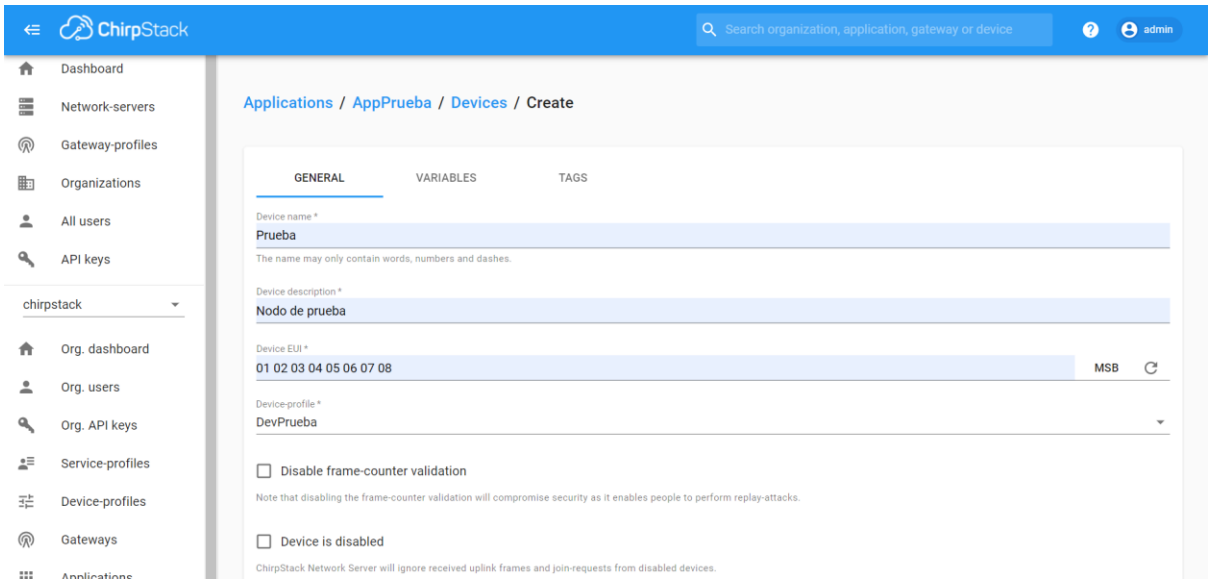

Fig. 89 Campos a rellenar para añadir un dispositivo

En el caso de que no se sepa el *DevEUI*, se puede averiguar de forma sencilla: dentro del apartado del *gateway* hay una pestaña de *Live LoRaWAN Frames* en la que aparecen todas las tramas recibidas, y ahí podemos ver la solicitud de conexión del dispositivo que queremos, *Fig. 90*, en la que aparece su *DevEUI*.

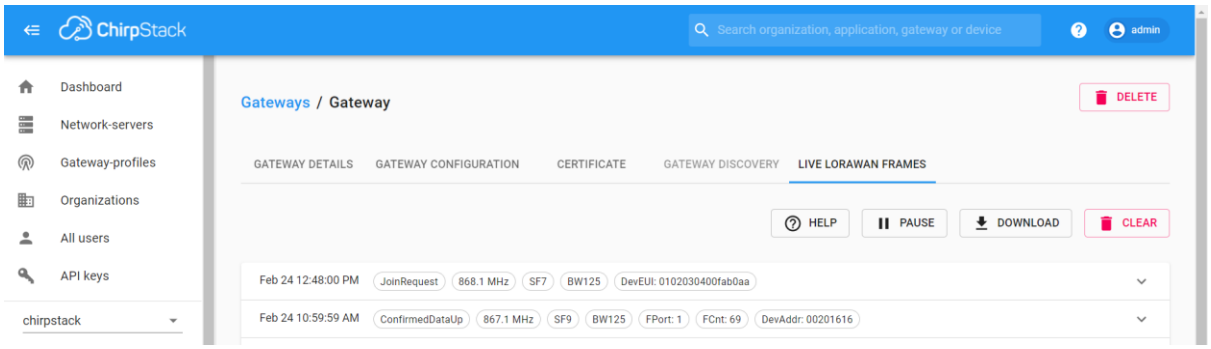

Fig. 90 Mensajes de join-request de un dispositivo no conocido

Para comprobar si el dispositivo funciona correctamente, dentro de la página del dispositivo, en la pestaña *Live LoRaWAN Frames*, al encender el dispositivo y, al ser un dispositivo OTAA, primero debería ver un *JoinRequest* seguido de un mensaje *JoinAccept*, como ocurre en la *Fig. 91*.

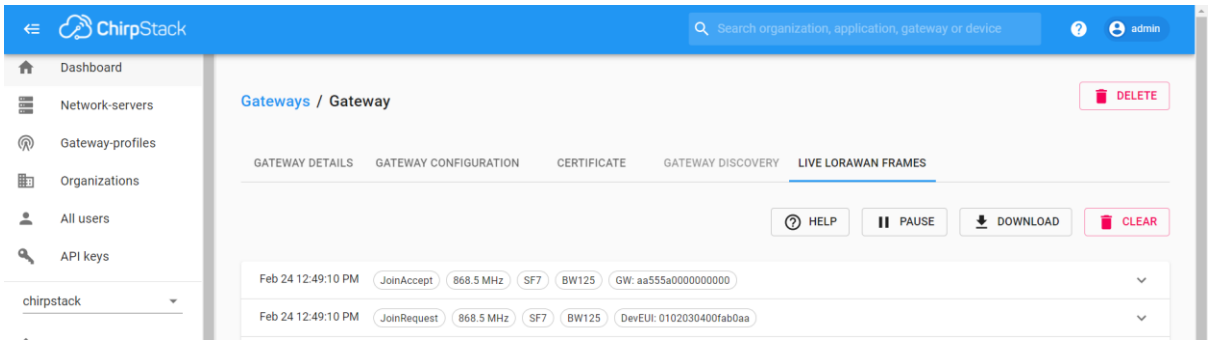

Fig. 91 Mensajes de join-request y join- accept

8. Referencias

[1] "Conoce los tipos de redes más importantes," *IONOS Digitalguide*, Jul. 18, 2019. https://www.ionos.es/digitalguide/servidores/know-how/los-tipos-de-redes-mas-conocidos/#:~:text=El%20t%C3%A9rmino%20red%20hace%20referencia (accessed Jun. 09, 2022).

[2] "¿Qué es una red inalámbrica? - Red Wi-Fi," *Cisco*. https://www.cisco.com/c/es_es/solutions/small-business/resource-center/networking/wireless-network.html (accessed Jun. 20, 2022).

[3] F. J. Santos Macías, "Tecnologías inalámbricas para la comunicación," Sep. 2009. Accessed: Jun. 08, 2022. [Online]. Available: https://www.feandalucia.ccoo.es/docu/p5sd5322.pdf

[4] Equipo Grupo Sinelec, "¿Qué son las redes LPWAN?," *Grupo Sinelec*, May 05, 2021. https://gruposinelec.com/que-son-las-redes-lpwan/#Las_posibilidades_de_las_redes_LPWAN (accessed Jun. 08, 2022).

[5] "¿Qué es la Narrowband IoT (NB-IoT)?," *www.thingsmobile.com*. https://www.thingsmobile.com/es/preguntas-y-respuestas/-que-es-la-narrowband-iot-nb-iot- (accessed Jun. 09, 2022).

[6] "Sigfox: qué es y cuándo hacer uso de su servicio de datos," *Wattabit*, Dec. 19, 2019. https://wattabit.com/sigfox-que-es-y-cuando-hacer-uso-de-su-servicio-de-datos/ (accessed Jun. 08, 2022).

[7] "Sigfox Technology," *www.sigfox.com*. https://www.sigfox.com/en/what-sigfox/technology (accessed Jun. 05, 2022).

[8] "Sigfox España | FAQs," *www.sigfox.es*. https://www.sigfox.es/faqs (accessed Jun. 08, 2022).

[9] H. Subedi, "How To Design A Network Topology | Jones IT," *Jones IT | Managed IT Services, IT Support, IT Consulting*, Nov. 22, 2020. https://www.itjones.com/blogs/2020/11/22/a-guide-to-network-topology (accessed Jun. 15, 2022).

[10] "Diferencias entre NB-IOT y LTE-M · Accent Systems," *Accent Systems*, May 03, 2018. https://accent-systems.com/es/diferencias-nb-iot-lte-m/ (accessed Jun. 09, 2022).

[11] "A technical overview of LoRa ® and LoRaWAN ™," 2015. Accessed: Jun. 08, 2022. [Online]. Available: https://lora-alliance.org/resource_hub/what-is-lorawan/

[12] 2CIGroup and J. Yao, “Conceptos de actualidad: LoRa y LoRaWan,” *2cigroup*, Mar. 22, 2021. https://www.2cigroup.com/es/conceptos-de-actualidad-lora-y-lorawan/ (accessed Jun. 08, 2022).

[13] “Spreading Factors,” *The Things Network*. https://www.thethingsnetwork.org/docs/lorawan/spreading-factors/ (accessed Jun. 08, 2022).

[14] Qoitech, “How Spreading Factor affects LoRaWAN device battery life,” *The Things Network*. https://www.thethingsnetwork.org/article/how-spreading-factor-affects-lorawan-device-battery-life

[15] “LoRa and LoRaWAN: Technical overview,” *lora-developers.semtech.com*. https://lora-developers.semtech.com/documentation/tech-papers-and-guides/lora-and-lorawan/ (accessed Jun. 11, 2022).

[16] “Frequency Plans,” *The Things Network*. https://www.thethingsnetwork.org/docs/lorawan/frequency-plans/ (accessed Jun. 08, 2022).

[17] D. Kjendal, “RP002-1.0.2 LoRaWAN® Regional Parameters,” Oct. 2020. Accessed: Jun. 08, 2022. [Online]. Available: https://lora-alliance.org/resource_hub/rp2-102-lorawan-regional-parameters/

[18] “Desarrollo de LoRa y aplicación de Internet de las cosas IV (diseño del sistema LoRaPingPang) - programador clic,” *programmerclick.com*. https://programmerclick.com/article/4887291877/ (accessed Jun. 08, 2022).

[19] E. B, “LoRa Documentation,” 2019. Accessed: Jun. 08, 2022. [Online]. Available: https://buildmedia.readthedocs.org/media/pdf/lora/latest/lora.pdf

[20] I. Poole, *Newnes Guide to Radio and Communications Technology*. 2003. Accessed: Jun. 15, 2022. [Online]. Available: https://almena.uva.es/discovery/fulldisplay?context=L&vid=34BUC_UVA:VU1&search_scope=FISICO_ELECTRO&tab=LibraryCatalog&docid=alma991008145459405774

[21] “The best IoT applications for LoRa™ Technology,” *www.pickdata.net*, Nov. 04, 2020. https://www.pickdata.net/es/noticias/mejores-aplicaciones-iot-tecnologia-lora (accessed Jun. 08, 2022).

[22] E. Gómez Sánchez, “Apuntes Tema 3 de Arquitectura de Redes, Sistemas y Servicios, Universidad de Valladolid,” 2020.

[23] K. Koenen, “Understanding the LoRaWAN® Architecture,” *tech-journal.semtech.com*. https://tech-journal.semtech.com/understanding-the-lorawan-architecture (accessed Jun. 08, 2022).

[24] “Device Classes,” *The Things Network*. https://www.thethingsnetwork.org/docs/lorawan/classes/

[25] “What is LoRaWAN® Specification,” *LoRa Alliance®*. https://lora-alliance.org/about-lorawan/ (accessed Jun. 10, 2022).

[26] “ABP vs OTAA,” *www.thethingsindustries.com*. https://www.thethingsindustries.com/docs/devices/abp-vs-otaa/

[27] N. Sornin and T. Kramp, “LoRaWAN® 1.1 Specification,” Oct. 2017. Accessed: Jun. 08, 2022. [Online]. Available: https://lora-alliance.org/resource_hub/lorawan-specification-v1-1/

[28] “End Device Activation,” *The Things Network*. https://www.thethingsnetwork.org/docs/lorawan/end-device-activation/

[29] N. Sornin, M. Luis, T. Eirich, T. Kramp, and O. Hersent, “LoRaWAN® Specification v1.0,” Jan. 2015. [Online]. Available: https://lora-alliance.org/resource_hub/lorawan-specification-v1-0/

[30] “Security,” *The Things Network*. https://www.thethingsnetwork.org/docs/lorawan/security/

[31] O. Brocaar, “The ChirpStack project,” *www.chirpstack.io*. https://www.chirpstack.io/project/ (accessed Jun. 09, 2022).

[32] “LoRaWAN Gateway Module WM1302 - Seeed Wiki,” *wiki.seeedstudio.com*. https://wiki.seeedstudio.com/WM1302_module/ (accessed Jun. 08, 2022).

[33] “WM1302 Pi Hat for Raspberry Pi - Seeed Wiki,” *wiki.seeedstudio.com*. https://wiki.seeedstudio.com/WM1302_Pi_HAT/ (accessed Jun. 09, 2022).

[34] O. Brocaar, “Quickstart Debian or Ubuntu,” *www.chirpstack.io*. https://www.chirpstack.io/project/guides/debian-ubuntu/ (accessed Jun. 09, 2022).

[35] “Node-RED,” *Nodered.org*, 2019. https://nodered.org/ (accessed Jun. 15, 2022).

[36] “LoRa Gateway | Industrial IoT LoRaWAN Gateways For Sale,” *www.multitech.com*. https://www.multitech.com/brands/multiconnect-conduit (accessed Jun. 23, 2022).

[37] “Cisco Wireless Gateway for LoRaWAN Data Sheet,” *Cisco*. https://www.cisco.com/c/en/us/products/collateral/se/internet-of-things/datasheet-c78-737307.html (accessed Jun. 23, 2022).