UNIVERSIDAD POLITÉCNICA DE CARTAGENA

Escuela Técnica Superior de Ingeniería de Telecomunicación

SISTEMA DE MONITORIZACIÓN PARA ELEMENTOS MÓVILES IOT MEDIANTE REDES LPWAN

TRABAJO FIN DE GRADO

GRADO EN INGENIERÍA TELEMÁTICA

Autora: Marina Marín Cava

Director: José Santa Lozano

Cartagena, octubre de 2020

AGRADECIMIENTOS

Quisiera agradecer a varias personas el apoyo y ayuda que me han prestado a la hora de llevar a cabo la realización de este Trabajo Fin de Grado con el que finalizo mi etapa como estudiante en la Universidad Politécnica de Cartagena. En primer lugar, me gustaría agradecer a José Santa Lozano y a Pablo Pavón Mariño, tutores de este proyecto, por confiar en mí y abrirme las puertas al ámbito de la investigación en este campo tan interesante como son las redes de sensores y, en concreto, la tecnología LoRa. Gracias a ellos, este trabajo ha sido parcialmente subvencionado por el Ministerio de Ciencia, Innovación y Universidades, bajo el proyecto Go2Edge (RED 2018-102585-T), algo que agradezco enormemente.

En segundo lugar, quiero mostrar mi agradecimiento a mis compañeros dentro del grupo de investigación GIRTEL, por acogerme tan bien desde el primer minuto y enseñarme tantas cosas.

A mi gran amiga y compañera de piso María Teresa Jódar, por acompañarme desde el primer momento en este largo camino y no dejarme caer nunca. Por infinitas horas de estudio juntas, ayudándonos la una a la otra, apoyo, consejos, comprensión. Estoy realmente orgullosa de lo que hemos conseguido amiga.

A mi estrella del cielo que, sin estar presente, siempre lo ha estado, por guiarme en todo momento bien en mi camino.

Finalmente, a mi familia, por su paciencia y apoyo estos años, por sentirse orgullosos de mí en cada paso y cada decisión que he tomado, y por darme la oportunidad de llegar a dónde he llegado.

Gracias a todos.

RESUMEN

Este proyecto presenta una plataforma telemática basada en tecnología LoRa para la conectividad de nodos IoT en entornos móviles, la recogida de información y su representación. La estructura básica de esta red de comunicaciones consta de varios nodos, incluyendo *gateway*, servidor de red y servidor de aplicaciones.

En el caso de estudio que se presenta, los nodos están instalados en una unidad móvil provista en un vehículo de movilidad personal, como puede ser un patinete eléctrico o una bicicleta, como es el caso en las pruebas de validación realizadas en el proyecto.

En este proyecto se muestran las técnicas y herramientas necesarias para desplegar y testear una red de comunicaciones LoRaWAN. Para ello, se ha construido un *gateway* empleando una Raspberry Pi 3 B+ y un concentrador LoRa conectado a ella, que es el que proporciona la capacidad de establecer una comunicación con esta tecnología. Además, se ha instalado el sistema operativo necesario para el correcto funcionamiento del *gateway*.

Para desarrollar la red de comunicaciones LoRaWAN y gestionar la comunicación entre los distintos nodos o sensores y el *gateway,* se ha instalado un servidor de red y de aplicaciones, el cual es proporcionado por la comunidad ChirpStack, para construir una red de sensores global. En esta aplicación empleamos la interfaz web proporcionada por ChirpStack, para configurar los parámetros de conexión necesarios y conectar los distintos nodos a la red LoRa.

Para la conexión de los nodos monitorizados con el servidor, recolección de datos del entorno y envío de datos al servidor, se ha desarrollado un software que corre en un SOC dentro de la unidad a bordo. Este realiza una recogida constante de datos de navegación (GPS, inercial y giróscopo) y meteorológicos/clima (CO_2, temperatura, humedad). Además de enviar estos datos mediante la red LoRaWAN, los principales parámetros se muestran en una pantalla OLED dispuesta en la misma unidad.

Tras la recolección de los datos por parte el servidor, los datos son almacenados en una base de datos de series de tiempo, de donde son extraídos para crear gráficas de visualización en un entorno Web accesible mediante las credenciales apropiadas.

ABSTRACT

This project presents a telematics platform based on LoRa technology for the connectivity of IoT nodes in mobile environments, the collection of information and its representation. The basic structure of this communication network consists of several nodes, including a gateway, network server, and application server.

In the case study presented, the nodes are installed in a mobile unit provided in a personal mobility vehicle, such as an electric scooter or a bicycle, as is the case in the validation tests carried out in the project.

This project shows the techniques and tools necessary to deploy and test a LoRaWAN communications network. For this, a gateway has been built using a Raspberry Pi 3 B+ and a LoRa hub connected to it, which provides communication capabilities. In addition, the necessary operating system for the correct functioning of the gateway has been installed.

To develop the LoRaWAN communications network and manage the communication between the different nodes or sensors and the gateway, a network and application server has been installed using the ChirpStack distribution to build a global sensor network. In this application, we use the web interface provided by ChirpStack, to configure the connection parameters necessary to connect the different nodes to the LoRa network.

To connect the monitored nodes with the server, collect data from the environment and send data to the server, a software has been developed that runs in a SOC inside the mobile unit. This application collects navigation data (GPS, inertial and gyroscope) and meteorological / climate (CO2, temperature, humidity) continuously. In addition to sending this data through the LoRaWAN network, the main parameters are displayed on an OLED screen arranged in the same unit.

Data is collected by the server and then stored in a time series database, from where they are extracted to plot graphs in a Web environment accessible through the appropriate credentials.

ACRÓNIMOS

- **ABP**: Activation By Personalization
- **CLI:** Command Line Interface
- **GPIO**: General Purpose Input/Output
- **GPS**: Global Positioning System
- **GPU**: Graphics Processing Unit
- **HTTP**: HyperText Transfer Protocol
- **I2C:** Inter Integrated Circuits
- **IoT**: Internet of Things
- **ISM**: Industrial Scientific Medical
- **JSON**: JavaScript Object Notation
- **LoRa**: Long Range
- **LoRaWAN**: Long Range Wide Area Network
- **LPWAN**: Low Power Wide Area Network
- **M2M**: Machine to Machine
- **MAC**: Medium Access Control
- **MQTT**: Message Queuing Telemetry Transport
- **OBU**: On-Board Unit
- **OTAA**: Over The Air Activation
- **RAM**: Random Access Memory
- **SNR**: Signal to Noise Ratio
- **SPI:** Serial Peripheral Interface
- **SQL**: Structured Query Database
- **TSDB**: Time Series Database
- **TVOC:** Total Volatile Organic Compounds
- **UART**: Universal Asynchronous Receiver-Transmitter
- **USB**: Universal Serial Bus

ÍNDICE DE CONTENIDO

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

CAPÍTULO 1. INTRODUCCIÓN

Actualmente vivimos en una sociedad que busca estar cada vez más conectada, no solo con el resto de personas, sino con objetos de su entorno cotidiano. Vivimos rodeados de máquinas y objetos conectados a las redes de comunicaciones, que envían o reciben información para realizar determinadas tareas. Es ahí de donde nace el concepto de Internet de las Cosas o *Internet of Things* (IoT).

IoT se define como la interconexión digital de elementos diarios con Internet. En los últimos años, este término ha sido uno de los más populares en la industria tecnológica, ya que permite que un gran número de dispositivos y sensores puedan conectarse a Internet. Las aplicaciones IoT son muy variadas, entre ellas, ciudades inteligentes, medio ambiente, industria, seguridad, etc.

Ante la necesidad de conectar dispositivos en entornos IoT, aparecen numerosas soluciones tecnológicas, entre las que se encuentran las redes LPWAN o redes de área extensa de baja potencia. Sus principales características son que proporcionan una conexión inalámbrica con una potencia baja, a un gran alcance, y con pequeño coste de los elementos. Es por ello que son redes idóneas en el ámbito IoT, ya que los dispositivos y sensores que se conectan, en la mayoría de los casos, a largas distancias, transmiten poco volumen de información y, a veces, no de forma constante en el tiempo.

1.1. Motivación y objetivos

Hoy en día existen tecnologías que son capaces de conectar varios objetos de nuestro alrededor, los cuales crean grandes cantidades de datos. Uno de los campos en los que más se investiga actualmente dentro de IoT es en las redes de sensores inalámbricas, que se encargan de medir distintos parámetros, procesar los datos obtenidos y enviarlos a la red.

Dado el interés por este tipo de redes, nos resultaba interesante llevar a cabo un proyecto que estuviera ligado con el desarrollo de una red de sensores de este tipo, experimentando con la tecnología LoRa.

Los principales objetivos son:

- Realizar un análisis del estado del arte de la tecnología LoRa y LoRaWAN para evaluar sus características de su funcionamiento y sus parámetros de configuración.
- Llevar a cabo la construcción de una pasarela LoRa compacta y de bajo coste mediante el uso de componentes de electrónica de consumo.
- Conseguir el funcionamiento y la obtención de datos de cada uno de los sensores elegidos.
- Desarrollo de una solución software para la recogida de datos de sensores de una unidad móvil IoT
- Configuración del camino de red para el envío de datos al *gateway* y a continuación al servidor LoRaWAN.
- Representación de datos de los sensores mediante visualización en series de tiempo y cartografía digital.
- Validación de la plataforma telemática del proyecto mediante el análisis de los datos recogidos.

1.2. Estructura del proyecto

En este apartado se va a resumir el contenido de los distintos capítulos del proyecto.

- Capítulo 1: Introducción.

En este capítulo, capítulo actual, se realiza una introducción a la tecnología y se exponen los motivos y objetivos de la realización de este proyecto, así como su estructura y planificación en el tiempo.

- Capítulo 2: Estado del Arte.

En esta sección se resumen las ideas necesarias que se deben conocer antes de ahondar en los detalles el proyecto. Además, se incluyen distintas soluciones actuales de las tecnologías ya existentes y algunas investigaciones ya realizadas.

- Capítulo 3: Tecnologías empleadas.

Se realiza la explicación detallada de cada una de las tecnologías que se emplean en este trabajo. En concreto, se tratan las especificaciones de LoRa y LoRaWAN, y se detalla la información sobre sistemas embebidos, sistemas operativos, métodos de acceso a sensores, entornos de programación, software para el *gateway* LoRa y LoRa Server, Grafana, e InfluxDB.

- Capítulo 4: Arquitectura del sistema.

En este capítulo se realiza un esquema principal de la arquitectura del sistema y se informa de los detalles de la arquitectura de cada uno de los componentes.

- Capítulo 5: Implementación del *gateway.*

El Capítulo 5 trata sobre los componentes y su coste, y los pasos a seguir para la configuración de un *gateway LoRa*, así como su unión al servidor.

- Capítulo 6: Programación de un nodo móvil.

En este capítulo se presenta una unidad móvil IoT tomada como referencia en este proyecto, cuyos sensores han sido usado para la recogida de datos. Se exponen el flujograma del software desarrollado y detalles sobre los datos obtenidos.

- Capítulo 7: Desarrollo de la solución.

En el Capítulo 7 se explica con detalle el despliegue realizado para la validación de todo el sistema. Se comenta la configuración de cada uno de los componentes del sistema y su funcionamiento, para después estudiar los datos recogidos.

- Capítulo 8: Conclusiones y futuras líneas de investigación.

En este capítulo final se hace una recapitulación de los resultados obtenidos tras la validación del sistema, y se enumeran las conclusiones alcanzadas con este proyecto.

Adicionalmente se incluyen una serie de anexos con detalles sobre desarrollos de interés en el trabajo:

- Anexo I: Instalación software *gateway* LoRa y pila del servidor de red Chirpstack.

En este primer anexo se incluyen los pasos necesarios referentes a la instalación del software en el *gateway* LoRa para su correcto funcionamiento y la pila del servidor de red ChirpStack.

- Anexo II: Instalación librerías para el control de los sensores.

En el segundo anexo se detallan las distintas librerías que tienen que ser instaladas para poder controlar de forma correcta cada uno de los sensores, y poder extraer datos de ellos de forma sencilla.

- Anexo III: Partes de código Python más importantes.

En este último anexo, se incluyen las partes del código Python más relevantes que se han realizado en este proyecto para el correcto funcionamiento del sistema.

CAPÍTULO 2. ESTADO DEL ARTE

En este capítulo se realiza un análisis previo de las tecnologías LPWAN o redes de área extensa de baja potencia, las cuales son muy populares actualmente, ya que se utilizan en aplicaciones de IoT en escenarios de Industria 4.0, *Smart Cities, Intelligent Transportation Systems, Smart Agriculture*, etc.

Para llevar a cabo nuestra propia red, se van a valorar las distintas soluciones del mercado disponibles, así como los estudios sobre diferentes propuestas ya adoptadas. Actualmente existen varias opciones para el desarrollo electrónico con comunicaciones IoT integradas, para una eficaz puesta en marcha de soluciones. En cuanto al entorno software, se van a considerar varias opciones de plataformas ya existentes para realizar una infraestructura de red IoT sin tener que desarrollar servidores propios para la comunicación con los dispositivos.

2.1. Redes de área amplia de baja potencia (LPWAN)

Los despliegues IoT que involucran sensores remotos tienen requisitos específicos tales como largo alcance, baja velocidad de datos y bajo consumo de energía. Las tecnologías radio de corto alcance ampliamente utilizadas (por ejemplo, ZigBee y Bluetooth) no están adaptadas para los escenarios que requieren una transmisión de largo alcance. Las soluciones basadas en comunicaciones celulares (por ejemplo, 2G, 3G, y 4G) pueden proporcionar una mayor cobertura, pero consumen demasiada energía del dispositivo. Por lo tanto, los requisitos de las aplicaciones IoT han impulsado el seguimiento de una nueva tecnología de comunicación inalámbrica: la red de área amplia de baja potencia (LPWAN) (Mekki, Bajic, Chaxel, & Meyer, 2018).

LPWAN es una clase de estándares de comunicación IoT inalámbrica y una solución con características tales como grandes áreas de cobertura, bajas velocidades de transmisión de datos con tamaños de paquetes pequeños y operación con bajo consumo de batería. La tecnología LPWAN se está desplegando globalmente y está demostrando un enorme potencial para la amplia gama de aplicaciones en IoT y M2M, especialmente en entornos restringidos (Yegin, et al., 2020).

Las redes LPWAN generalmente consisten en redes de un solo salto, donde cada nodo está conectado con al menos un *gateway* a través de enlaces de Radio Frecuencia (RF) que forman una topología en estrella (Aftab, Raza Zaidi, & McLernon, 2019).

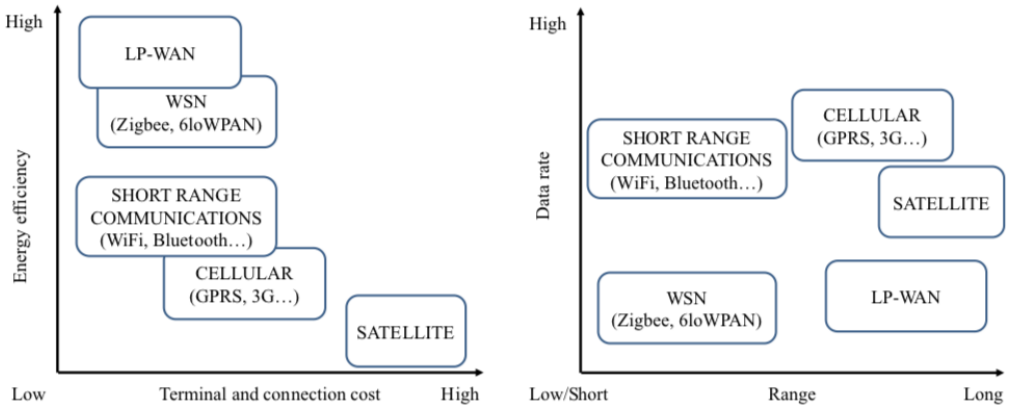

Figura 1. Principales características de las tecnologías para IoT. (Cano & Sanchez-Iborra, 2016)

La mayoría de las redes LPWAN trabajan en la banda ISM libre sin licencia, con frecuencias por debajo de 1 GHz. Para hacer frente a la cantidad limitada de ancho de banda disponible en ese espectro, a menudo se aplican regulaciones del ciclo de trabajo, potencia radiada y ancho de banda. La banda más utilizada en Europa es la de 868 MHz.

Además de sus ventajas, las LPWANs también tienen algunos inconvenientes tales como el ancho de banda, el tamaño de las tramas y la tasa de datos. El ancho de banda es reducido, comprendido entre los 50 bits/s hasta los 250 kbits/s teniendo ciclos de trabajo de entre 0.1% y 10%; las tramas son excesivamente limitadas, incluyendo un *payload* desde los 12 bytes hasta los 222 bytes, y existe una gran probabilidad de pérdida de paquetes, causada por colisiones o condiciones adversas en la transmisión de los mensajes.

Dentro de las redes LPWAN existen varias tecnologías. Entre las más utilizadas en IoT se encuentran LoRaWAN y SigFox. En este trabajo nos enfocamos en LoRaWAN. Aunque ambas ofrecen características similares a sus usuarios, cada una adopta enfoques tecnológicos diferentes. Por un lado, Sigfox presenta un esquema de transmisión altamente cerrado y limitado con muy pocas capacidades de adaptación, ya que emplea una tecnología patentada de banda ultra estrecha con una velocidad máxima de datos de enlace ascendente de 100 bps y una carga útil máxima de 12 bytes por cada paquete (Sanchez-Iborra, Gomez, Viñas, Cano, & F.Skarmeta, 2018). A su vez, LoRaWAN presenta una solución más avanzada que permite al usuario adaptar el sistema a sus

necesidades. Específicamente, LoRaWAN define las capas de acceso físico (PHY) y de Acceso al Medio (MAC), y propone una topología de estrella de estrellas con puertas de enlace que sirven como puentes transparentes entre los dispositivos finales y la red central de datos, donde se almacenan y se ponen a disposición del suscriptor. Los dispositivos finales se conectan a los puntos de acceso a través de enlaces de un salto utilizando la modulación *Long Range* (LoRa), la cual sigue el esquema de la modulación de espectro ensanchado, que se enfoca en proporcionar solidez a las transmisiones de largo alcance

(Cano & Sanchez-Iborra, 2016). LoRa presenta tres parámetros configurables diferentes, el Factor de Dispersión (*Spreading Factor* - SF), la Velocidad de Codificación (*Coding Rate* - CR) y el Ancho de Banda (*Bandwidth* - BW). Dependiendo de la configuración de estos, la velocidad de los datos varía desde 0.25 kbps hasta 50 kbps. En este caso, la longitud máxima permitida de la carga útil es de 242 Bytes (Sanchez-Iborra, Gomez, Viñas, Cano, & F.Skarmeta, 2018)

En la Tabla 1 se puede observar un resumen de las principales características de las tecnologías LPWAN mencionadas anteriormente.

	LoRaWAN	**Sigfox**
Banda	433/868/780/915 MHz	868/915 MHz
Máxima velocidad de datos	50 kbps	100 bps
Rango (urbano)	5 km	10 km
Tamaño del paquete	Max. 256 B	12 B
Topología	Estrella de estrellas	Estrella
Itinerancia	Sí	Sí
Seguridad	Totalmente abordado	Parcialmente abordado
Propiedad del protocolo	Parcialmente propietario	Propietario

Tabla 1. *LoRa vs Sigfox (Cano & Sanchez-Iborra, 2016)*

2.2. Estudios previos

Aunque LoRaWAN es una tecnología que ha aparecido recientemente, se han publicado diversos trabajos para evaluar o analizar su desempeño en distintos escenarios o proponer mejoras a la versión estándar de LoRaWAN.

En este apartado he tratado de plasmar distintos trabajos que he buscado en relación con mi proyecto, que me han aportado conocimientos sobre esta tecnología.

En (Petric, Goessens, Nuaymi, Toutain, & Pelov, 2016), tuvo lugar el despliegue de una red LoRaWAN para tratar de encontrar la correlación entre la elevación y el rendimiento. Finalmente, tras varias mediciones, se concluyó que la ubicación y elevación de la antena de la estación base tienen una gran repercusión en el rendimiento de la red.

Por otro lado, se han llevado a cabo experimentos como el de (P.Radcliffe, K.Chavez, P.Beckett, Spangaro, & C.Jakob, 2017), en el que se colocó una estación base en el techo de la Universidad RMIT en el corazón de Melbourne, Australia, para medir la capacidad de penetración de LoRa hasta el nivel de la calle de un CBD (Distrito Central de Negocios). Para llevar a cabo el experimento, rastrearon a una persona que caminaba por el CBC, por varias rutas, midiendo la pérdida de paquetes y la atenuación de la señal. La conclusión que obtuvieron fue que la capacidad de LoRaWAN para realizar esa acción era mucho menor de lo que se esperaba. La comunicación sin pérdidas se limitó a 200 metros aproximadamente de la estación base, y a unos 600 metros hubo una pérdida total de transmisión.

En el documento (Voigt, Bor, Roedig, & Alonso, 2016), los autores han evaluado el impacto de las interferencias en las redes LoRa, y han demostrado que éstas pueden reducir drásticamente el rendimiento de la red. Los resultados de la investigación demuestran que las antenas direccionales y el uso de múltiples estaciones base, pueden mejorar el rendimiento frente a las interferencias.

En el trabajo de (Aftab, Raza Zaidi, & McLernon, 2019), se realizó una investigación sobre la efectividad de un *gateway* LoRa en presencia de otras puertas de enlace LoRa. Tras la investigación, se concluye que la presencia de más puertas de enlace LoRa en la misma región geográfica, deteriora el rendimiento de éstas.

También, cabe destacar que, según un estudio realizado en Bangkok, (Vatcharatiansakul, P.Tuwant, & C.Pornavalai, 2017), el alcance de la comunicación es de 2 km en áreas rurales al aire libre y de 55-100 metros en un ambiente urbano interior. Según los resultados, el alcance de la comunicación está influenciado por las propiedades de las antenas, tales como la ganancia, la dirección y la altura de la antena.

Otros autores (Petajajarvi, Mikhaylov, Roivainen, Hanninen, & Pettissalo, 2015), llevaron a cabo la investigación de la cobertura de LoRaWAN en diferentes entornos, colocando el dispositivo final a bordo de un coche y de un barco. Los resultados de esta

investigación concluyeron que, más del 60% de los paquetes se recibieron correctamente a distancias de 5 a 10 km a bordo de un coche, y en el agua, se alcanzó un rango de comunicación de casi 30 km.

2.3. Gateways comerciales LoRaWAN

Existen numerosos *gateways* comerciales que se basan en la tecnología LoRaWAN. Por su popularidad en despliegues existentes, se pueden encontrar los siguientes:

- *MultiTech Programmable Gateway for the Internet of Things* (MTCDT Series), en concreto MTCDT-L4N1-246A-915-US es un *gateway* LoRa que tiene varias opciones de conectividad de red para su plataforma de administración de datos preferida, tales como 4G-LTE, 3G, 2G y Ethernet. Tiene un precio de 719€. (Multitech, 2020)

Figura 2. Gateway Multitech. (Multitech, 2020)

- *Gateway* Cisco LoRaWAN: Se trata de un *gateway* que permite implementaciones de IoT que requieren velocidades de datos bajas y dispositivos finales alimentados por baterías a través de la conectividad a larga distancia. Además, proporciona hasta 16 canales de enlace ascendente y admite la geolocalización a través de TDo A y RSSI. El precio de este *gateway* es de 2.500 € aproximadamente.

Figura 3. Gateway LoRa Cisco. (Cisco, 2020)

- Lorrier LR2: gateway de nivel de operador mejorada para construir una red profesional de IoT basada en el protocolo LoRaWAN. Dispositivo externo

destinado a establecer una amplia red de cobertura por parte de operadores de telecomunicaciones y una red local por parte de individuos o proveedores de servicios de conectividad IoT. El precio de este gateway es de 400€ aproximadamente.

Figura 4. Gateway Lorrier LR2 (The Things Network, 2020).

CAPÍTULO 3. TECNOLOGÍAS EMPLEADAS

3.1. LoRa

LoRa es una tecnología de capa física que se basa en la modulación de señales en la banda ISM por debajo de 1GHz, mediante la utilización de una técnica denominada de espectro ensanchado (Mekki, Bajic, Chaxel, & Meyer, 2018). Se trata de una tecnología de modulación de Radio Frecuencia (RF).

LoRa fue desarrollado por primera vez por la start-up Cycleo en 2009 (en Grenoble, Francia), y Semtech (EE. UU), lo compró tres años después. En 2015, LoRa fue estandarizado por LoRa-Alliance y se implementa en 42 países. Todavía se está siendo desplegado en otros países debido a la inversión de varios operadores móviles (por ejemplo, Bouygues y Orange en Francia, KPN en Países Bajos y Fastnet en Sudáfrica). (Mekki, Bajic, Chaxel, & Meyer, 2018)

Actualmente Semtech posee la patente, mientras que la fundación LoRa Alliance es la encargada ahora del desarrollo del estándar y de su evolución (LoRaWAN, 2020). LoRa es utilizada en enlaces de comunicación de largo alcance, siendo una de sus principales características su bajo consumo de potencia, permitiendo la creación de dispositivos que funcionan con baterías que pueden durar casi 10 años, y su gran rango de comunicación, alcanzando hasta cinco kilómetros en áreas urbanas, y hasta 15 kilómetros en áreas rurales. (Semtech, LoRa® and LoRaWAN®: A technical overview, 2019)

3.1.1. Modulación LoRa

La técnica de modulación empleada por LoRa se basa en técnicas de espectro ensanchado, y es una modificación de la tecnología *Chirp Spread Spectrum* (CSS) ya existente, la cual se caracteriza por modular los datos en diferentes velocidades y canales, empleando *Forward Error Correction* (FEC) para realizar la corrección de los errores. Esta modulación mejora significativamente la sensibilidad del receptor, y consiste en emplear un mayor ancho de banda del que se necesita teóricamente, para permitir así la recepción al mismo tiempo de gran cantidad de señales con distinta velocidad. Con esto se consigue una mayor solidez, pero debido a realizar una decodificación de la señal más complicada.

Todo ello, logrando tasas en la transmisión desde los 0.3 kbps hasta los 38.4 kbps, debido a la sensibilidad del receptor por la ordenación de codificación.

LoRa emplea seis factores de dispersión (*spreading factor*), de SF7 a SF12, para adecuar la tasa de los datos y el rango de compensación. El factor de dispersión (SF) define el número de bits usado para codificar un símbolo. Cuanto mayor sea el factor de dispersión, mayor será la sensibilidad en el receptor y el alcance, teniendo una velocidad de datos más baja, y viceversa. (Mekki, Bajic, Chaxel, & Meyer, 2018)

Además, Lora emplea un mecanismo de velocidad de datos adaptativa (*Adaptive Rate* - ADR). El objetivo principal de este mecanismo es ahorrar la energía de la batería de los nodos finales LoRaWAN. Para ello, permite a los dispositivos ajustar sus parámetros dependiendo de su distancia a la pasarela y del tamaño del mensaje que van a transmitir, consiguiendo así una velocidad de transmisión óptima y una mayor eficiencia.

3.1.2. Canales y rangos de frecuencias

La tecnología LoRa emplea frecuencias que se pueden utilizar sin necesidad de licencia. Estas frecuencias son básicamente las de la banda ISM, aunque podría emplear cualquier frecuencia por debajo de 1GHz. LoRa suele operar en las bandas 433 MHz, 868 MHz y 915 MHz, pero dependiendo del país, algunas de las bandas pueden estar restringidas. En Europa no se puede usar la de 915 MHz, y normalmente se utiliza la de 868 MHz. La banda ISM empleada en Europa es la de 863 a 870MHz, moderada por el Instituto Europeo de Normas de Telecomunicaciones. Emplea 8 canales, los cuales se eligen arbitrariamente, y cuentan con 0.3MHz de ancho de banda por canal.

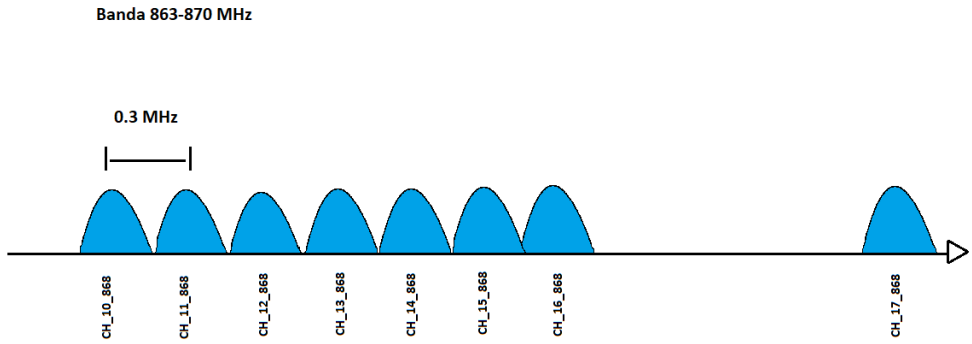

Figura 5. Canales LoRaWAN.

La banda de frecuencias que se utiliza en USA, Singapur, Canadá, Israel o Australia, es la banda ISM de 902 a 928, la cual emplea 13 canales con 2,16 MHz de ancho de banda cada uno de ellos.

3.2 LoRaWAN

Lora es la capa física (PHY), es decir, la modulación utilizada para crear el enlace de comunicación de largo alcance. LoRaWAN es un protocolo, una especificación de redes LPWAN, diseñada para conectar de forma inalámbrica elementos a Internet. Dentro de los niveles OSI, se puede incluir en el nivel 2, ya que se define como un protocolo MAC (Media Access Control), cuyo objetivo es unir los distintos dispositivos LoRa, gestionando sus canales y parámetros de conexión, tales como el ancho de banda, el canal, el cifrado de los datos, etc.

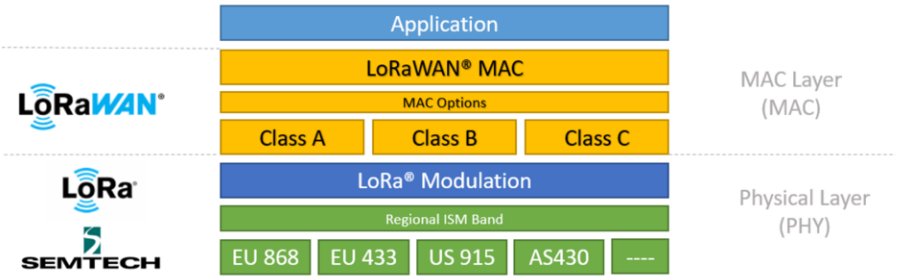

Figura 6. Estructura de red LoRaWAN. (Semtech, LoRa Developer Portal, 2020)

3.2.1 Arquitectura de red LoRaWAN

En general, la topología de las redes LoRaWAN es de estrella de estrellas, en la que los dispositivos finales establecen una comunicación con el *gateway*, el cual retransmite los mensajes entre los nodos finales y el servidor de red.

Dentro de la red LoRaWAN podemos destacar la existencia de tres elementos (Semtech, LoRa Developer Portal, 2020), tal y como se muestra en la Figura 7:

- Dispositivos finales: elementos tales como sensores o actuadores que recogen información y la envían al *gateway* de la red.
- *Gateways* o concentradores: Son estaciones base, las cuales implementan LoRa, las cuales reciben información de múltiples dispositivos finales y se encargan de reenviarla a los servidores de red. Cada *gateway* reenviará los paquetes recibidos desde el nodo final al servidor de red a través de conexiones IP estándar.

- Servidores de red: Se encargan de gestionar toda la red, controlando los parámetros de esta para adaptar el sistema a las condiciones siempre cambiantes, y establecer conexiones seguras para el transporte de datos extremo a extremo. El servidor de red asegura la autenticidad de cada sensor en la red y la integridad de cada mensaje. Concretamente, el servidor de red se encarga de comprobar la dirección del dispositivo, autenticar la trama y gestionar el contador de trama, adoptar las velocidades de datos mediante el protocolo ADR, reenviar cargas útiles de enlace ascendente a los servidores de aplicaciones apropiados, poner en cola las cargas útiles de enlace descendente que provienen de cualquier servidor de aplicaciones conectado a la red, reenviar mensajes de solicitud de unión y de aceptación de unión entre los dispositivos y el servidor de unión, etc.
- Servidores de aplicación: Su función es manejar, administrar e interpretar los datos de la aplicación del sensor de manera segura. A través de ellos, el usuario obtiene la información.

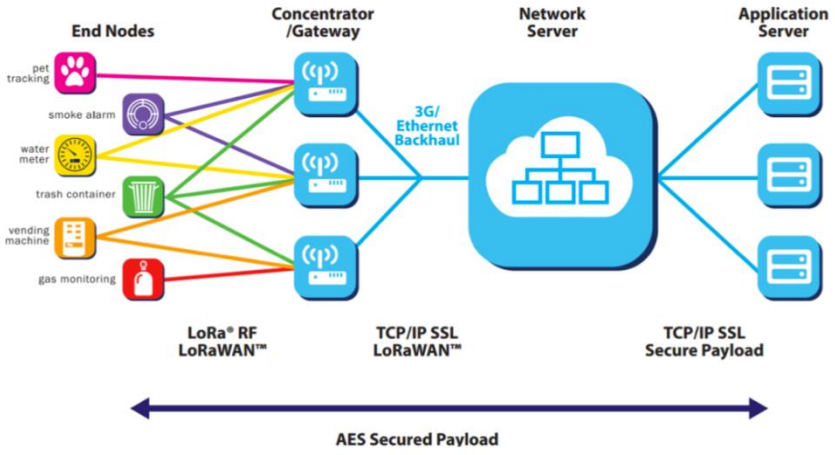

Figura 7. Arquitectura de red LoRa. (LoRa Alliance, LoRaWAN. What is it?, 2015)

En las redes LoRaWAN no existe una asociación fija entre un dispositivo final y un *gateway* específico; los nodos pueden enviar sus datos y que estos sean recibidos por varias pasarelas a la vez. Las puertas de enlace LoRaWAN operan totalmente en la capa física, y se encargan de reenviar mensajes de radio LoRa por la red infraestructura.

Cada *gateway* se encargará de enviar los datos recibidos al servidor de red a través de otras tecnologías como Wifi, Ethernet o red móvil.

La complejidad y la capacidad intelectual del sistema está en el servidor de red, el cual se encarga de realizar acciones de seguridad, filtrar los paquetes redundantes y gestionar la velocidad de datos, etc.

Cabe destacar que la comunicación entre los nodos de una red LoRaWAN es asíncrona. Los nodos se comunican cuando tienen datos listos para enviar siguiendo el protocolo Aloha. Cuando una red tiene topología de malla, los dispositivos finales o nodos tienen que "activarse o despertar" para realizar la sincronización con la red y verificar si hay mensajes. El hecho de realizar esta sincronización, supone un gran consumo de energía, siendo este el principal motivo de la pérdida de la batería de los dispositivos.

3.2.2 Canales y velocidades de transmisión

Para realizar una comunicación entre los dispositivos finales de la red y los *gateways* o pasarelas, se emplean diferentes tasas de datos y canales de frecuencia. Las comunicaciones que emplean distintas tasas, gracias a la técnica de espectro ensanchado, no producen interferencias entre ellas. Para maximizar la duración de la batería de los dispositivos y la capacidad de la red, LoRaWAN es capaz de administrar la tasa de datos y la potencia de salida de radio frecuencia para cada uno de los dispositivos de la red, utilizando un esquema adaptativo de velocidad de datos (ADR).

Por lo tanto, los dispositivos finales pueden realizar en cualquier momento una transmisión en cualquiera de los canales disponibles, empleando una tasa de datos cualquiera, cumpliendo siempre las siguientes características (Yegin, et al., 2020):

- El nodo final cambia los canales de forma aleatoria para cada una de las transmisiones de radio. Esto da lugar a que el sistema tenga más robustez frente a las interferencias que se produzcan.
- El nodo no sobrepasa el ciclo de trabajo máximo para transmitir de acuerdo con la sub-banda en la que está operando, cumpliendo con las normas establecidas de forma local.
- El nodo no supera la duración de transmisión máxima relacionada son la sub-banda que está utilizando.

Las bandas de 433 MHz y 868 MHz (en Europa) disponen de 16 canales, y la banda de 900MHz de 72 canales. Como he mencionado anteriormente, en este trabajo se emplea la banda de 868 MHz. En la Tabla 2 podemos observar los parámetros para los distintos canales siguiendo la normativa que se aplica en Europa:

Canal	Parámetros	Bandas de frecuencias	
		868 MHz	**433 MHz**
0	Frecuencia	868100 kHz	433175 kHz
	Ciclo de trabajo	0.33 %	0.33 %
	Data Rate	0 – 5	0 - 5
	Estado	On	On
1	Frecuencia	868300 kHz	433375 kHz
	Ciclo de trabajo	0.33 %	0.33 %
	Data Rate	0-5	0-5
	Estado	On	On
2	Frecuencia	868500 kHz	433575 kHz
	Ciclo de trabajo	0.33 %	0.33 %
	Data Rate	0 – 5	0 - 5
	Estado	On	On
3-15	Frecuencia	868325 kHz-869750 kHz	433050 kHz – 434790 kHz
	Ciclo de trabajo	*	*
	Data Rate	0 -5	0 – 5
	Estado	Off	Off

Tabla 2. Tabla de parámetros de cada canal para LoRaWAN. (Libelium, 2020)

Como podemos ver en la tabla anterior, los canales del 3 al 15 se encuentran por defecto apagados. Al realizar la activación, el responsable de la red debe adaptar el ciclo de trabajo del resto de los canales para que el número total de estos no supere el 1 % permitido como máximo en la banda de frecuencias. También, estos canales del 3 al 15, pueden ser configurados con los *data rate* 6 y 7, aunque no aparezca en la Tabla 2, de forma distinta a los canales configurados por defecto, que emplearían *data rate* de 0 a 5.

Existen limitaciones que establecen cuánto tiempo puede estar activado o transmitiendo el nodo. LoRaWAN exige una limitación del ciclo de trabajo en cada sub-banda. La misma, no puede ser usada de nuevo a lo largo de un intervalo de tiempo, el cual viene sujeto al ciclo de trabajo de la sub-banda y al tiempo en el aire. El cálculo se puede realizar con la siguiente fórmula:

$$Toff = \frac{TimeOnAir}{DutyCycle\ subband} - TimeOnAir$$

Ecuación 1. Limitación de tiempo. (Yegin, et al., 2020)

La tasa o velocidad de datos que se emplea en la tabla anterior viene determinada por los factores de la siguiente Tabla 3:

Tasa de datos	Configuración	Tasa de bit
0	LoRa: SF12 / 125 kHz	250 bps
1	LoRa: SF11 / 125 kHz	440 bps
2	LoRa: SF10 / 125 kHz	980 bps
3	LoRa: SF9 / 125 kHz	1760 bps
4	LoRa: SF8 / 125 kHz	3125 bps
5	LoRa: SF7 / 125 kHz	5470 bps
6	LoRa: SF7 / 250 kHz	11000 bps
7	FSK: 50 kbps	50000 bps

Tabla 3. Tasa de datos EU863-870 TX. (LoRa Alliance, LoRaWAN Regional Parameters, 2020)

El SF (Spreading Factor) establece el número de datos repetidos que son envíados en la transmisión de un mensaje. La tasa de datos está relacionada con este parámetro porque, cuanto más grande es éste, más cantidad de información repetida se enviará, y ante esto, mayor será la solidez de la comunicación y su alcance, pero teniendo una velocidad menor en la transmisión.

3.2.3 Clases LoRaWAN

Dentro de las redes LoRaWAN existen tres clases diferentes de dispositivos, Clase A, Clase B y Clase C, siendo posible encontrar dispositivos de todas las clases coexistiendo dentro de una misma red (LoRa Alliance, LoRaWAN Specification, 2018). El objetivo de esta clasificación reside en las diferentes necesidades de una amplia gama de aplicaciones IoT.

- Clase A: Dentro de esta clase nos encontramos con dispositivos que permiten comunicaciones bidireccionales, teniendo en cuenta que la transmisión de enlace ascendente puede realizarse en cualquier momento, y la de enlace descendente sólo puede realizarse si ha habido antes una transmisión ascendente. Esto ocurre porque tras el envío de un paquete, se despliegan un par de ventanas de recepción de enlace descendente, que permiten en envío de un paquete de respuesta. Mientras tanto, el dispositivo final puede entrar en modo reposo, dando lugar a un menor consumo energético. Por ello se puede decir que los dispositivos de esta clase con los más eficientes energéticamente.
- Clase B: A esta clase pertenecen los dispositivos que tienen la capacidad de recibir datos sin haber enviado un paquete anteriormente. De esta forma, se permite que

la aplicación envíe datos a los dispositivos de forma programada. Para conseguir una sincronización de la pasarela con el nodo final, se envían tramas *beacon* periódicamente para planificar el tiempo que la ventana de recepción debe estar abierta. Los dispositivos de esta clase tienen un mayor gasto energético que los de la clase anterior.

- Clase C: Los dispositivos de esta clase son aquellos que permanecen periódicamente en modo escucha, es decir, son capaces de recibir información en cualquier instante, a excepción de cuando ellos estén realizando un envío. Esta clase consigue los menores valores de latencia, a consta de un gasto de energía mucho mayor, en comparación con las otras dos clases. Se recomienda usar dispositivos de esta clase cuando éstos cuentan con una fuente de alimentación externa.

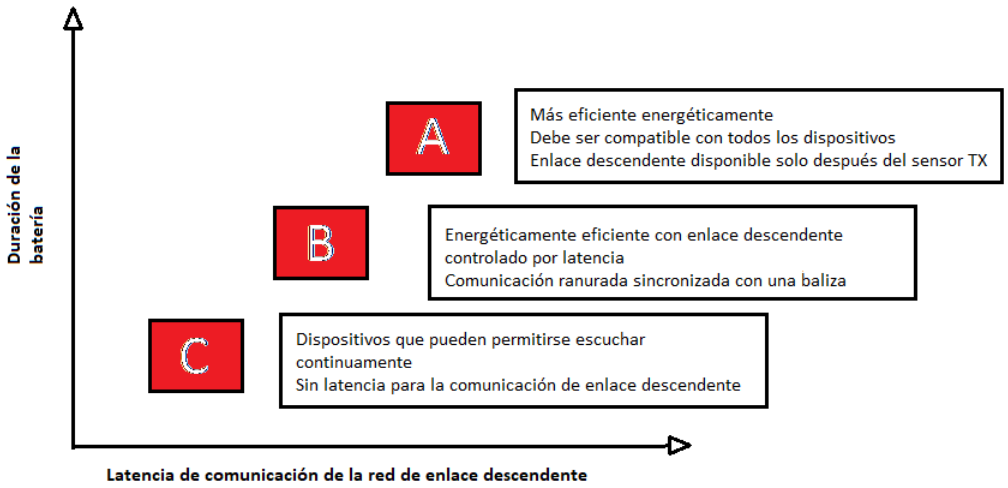

Figura 8. Comparación clases LoRa en función de la latencia y la duración de la batería.

3.2.4 Seguridad en LoRaWAN

En LoRaWAN se utilizan dos niveles de seguridad, uno para la red y otro para la aplicación, utilizando para ello cifrado AES.

- Seguridad de red: Se emplean claves de 128 bits para garantizar la seguridad en la red, la llamada *Network Session Key*.
- Seguridad de aplicación: Se emplean claves de 128 bits que garantizan que el operador de red no pueda acceder a los datos del usuario. Las claves que se emplean en este nivel son la *Application Session Key* y la *Application Key*.

3.2.5 Métodos de activación

Para conectarse a una red LoRaWAN, cada dispositivo final tiene que ser personalizado y activado. La activación de los dispositivos finales se puede realizar de dos formas:

- *Activation by Personalization* (ABP): Este modo de activación es más sencillo que el anterior. En este caso se activa el dispositivo final mediante su personalización, lo cual conlleva el almacenamiento del *Device Address* (DevAddr) y de las dos claves de sesión *Network Session Key* (NwkSKey) y *Application Session Key* (AppSKey) en el propio dispositivo. Por lo tanto, el dispositivo final ya dispone de la información requerida para participar en una red LoRa tras el inicio de ésta. Una de las ventajas de este modo de conexión es que el dispositivo final no debe realizar el proceso de unión a la red para poder enviar datos, y la confirmación por parte del servidor no es necesaria, ya que la conexión está manualmente asignada. Este tipo de conexión se recomienda para dispositivos que están en movimiento o no tienen muy buena recepción.

  Los parámetros, ya mencionados, necesarios en esta conexión son:

  -DevAddress: 32 bits que sirven para identificar el nodo final en la red.

  -**NetworkSessionKey:** Clave de sesión de red propia para cada dispositivo. Esta es utilizada por el servidor de red y por el dispositivo final para deducir y verificar el MIC (Código de integridad del mensaje) de todos los mensajes para garantizar la integridad.

  -**ApplicationSessionKey:** Clave de sesión de aplicación propia para cada dispositivo. Esta clave es usada por el servidor de aplicaciones y el dispositivo final para codificar la carga útil de los mensajes específicos de la aplicación.
- *Over-the-Air Activation* (OTAA): El modo de activación por aire es el modo más seguro para realizar la conexión a una red LoRaWAN. Para llevarlo a cabo, los dispositivos finales deben seguir un procedimiento de unión, antes del intercambio de mensajes, el cual requiere la personalización del dispositivo final con un identificador del dispositivo final único (DevEUI), un identificador de aplicación (AppEUI) y una clave AES-128 (AppKey). Este procedimiento se debe realizar también cada vez que un dispositivo pierda la información de la sesión. La ventaja principal de este tipo de conexión es que la sesión es muy segura ya que se crea bajo demanda y es renovada cada vez que el dispositivo se apaga, reinicia, o pierde la conexión.

3.2.6 Estructura de paquetes LoRaWAN

En las redes LoRaWAN se pueden distinguir dos categorías de mensajes, *uplink* y *downlink*:

- Mensajes *uplink* (enlace ascendente): Estos mensajes se envían desde los dispositivos finales hacia el servidor de red atravesando una o varias puertas de enlace. Los mensajes de enlace ascendente están formados por un encabezado físico (PHDR) y la carga útil (PHYPayload), ambos con su CRC (PHDR_CRC y CRC). (LoRa Alliance, LoRaWAN Specification, 2018)

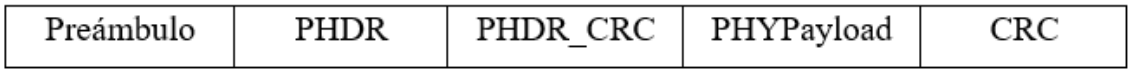

Figura 9. Estructura mensaje ascendente PHY. (LoRa Alliance, LoRaWAN Specification, 2018)

- Mensajes *downlink* (enlace descendente): Los mensajes de enlace descendente se envían desde el servidor de red hacia un solo dispositivo final, atravesando un único *gateway*. En estos mensajes, además de la carga útil se incluyen un encabezado físico (PHDR) y un encabezado CRC (PHDR_CRC). En este caso no se incluye el CRC del *Payload.* (LoRa Alliance, LoRaWAN Specification, 2018)

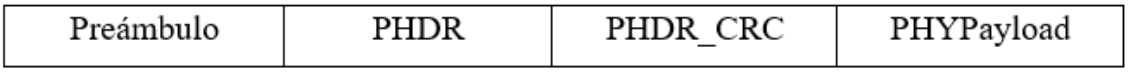

Figura 10. Estructura mensaje descendente PHY. (LoRa Alliance, LoRaWAN Specification, 2018)

Todos los mensajes, tanto ascendentes como descendentes, contienen una carga útil (PHYPayload), la cual se compone de un encabezado MAC (MHDR), una carga útil MAC (MACPayload) y finaliza con un Código de integridad de mensaje (MIC). El formato y el tamaño de cada uno de ellos es el siguiente:

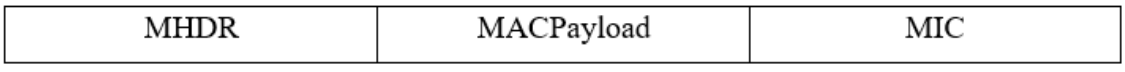

Figura 11. Formato de carga útil PHY. (LoRa Alliance, LoRaWAN Specification, 2018)

En la cabecera MAC o MHDR se detalla el modelo o tipo de mensaje y la versión del formato de la trama con la que se ha realizado la codificación . En LoRaWAN se pueden enumerar 6 tipos diferentes de mensajes:

MType	Descripción
000	*Join Request*
001	*Join Accept*
010	*Unconfirmed Data Up*
011	*Unconfirmed Data Down*
100	*Confirmed Data Up*
101	*Confirmed Data Down*
110	*RFU*
111	*Proprietary*

Figura 12. Tipos de mensaje MAC. (LoRa Alliance, LoRaWAN Specification, 2018)

El MACPayload lo forman una cabecera de trama (FHDR), un puerto que es opcional y un *Payload* de trama también opcional, como se puede observar en la siguiente Figura 13:

FHDR	FPort	FRM Payload
7-23 bytes	0-1 bytes	0-N bytes

Figura 13. Carga útil MAC. (LoRa Alliance, LoRaWAN Specification, 2018)

La carga útil MAC contiene una cabecera de la trama (FHDR), la cual incluye la dirección para identificar al dispositivo en el interior de la red LoRaWAN, DevAddr, un campo FCtrl para poder habilitar el *Adaptive data rate*, un contador FCnt y un campo FOps si lo que se transmite es un comando MAC. El FPort se utiliza para indicar si el FRM Payload incluye comandos MAC o datos de aplicación. (LoRa Alliance, LoRaWAN Specification, 2018)

DevAddr	FCtrl	FCnt	FOpsts
4 bytes	1 byte	2 bytes	0-15 bytes

Figura 14. Datos de la cabecera de la trama. (LoRa Alliance, LoRaWAN Specification, 2018)

En el DevAddr, los 7 bits más significativos se utilizan para el identificador de red (NwkID), y los veinticinco sobrantes pertenecen a la dirección de red (NwkAddr), que la puede asignar la persona que se encargue de la administración de la red.

Teniendo todo esto en cuenta, el límite de tamaño del MacPayload (de tamaño M en la Figura 15) puede cambiar dependiendo de la banda de frecuencia en que se trabaja, la

tasa de datos y la falta del campo de control (FOpts de tamaño N), como podemos observar en la Figura 15:

Tasa de datos	M (bytes)	N (bytes)
0	59	51
1	59	51
2	59	51
3	123	115
4	230	222
5	230	222
6	230	222
7	230	222

Figura 15. Tamaño máximo de datos. (LoRa Alliance, LoRaWAN Regional Parameters, 2020)

3.3. Control y acceso a sistemas embebidos

3.3.1 Sistemas embebidos

Un sistema integrado o embebido es un sistema de computación que está diseñado para controlar una función o un rango de funciones dedicadas. Al contrario de lo que ocurre con los ordenadores de propósito general, los cuales están diseñados para cubrir un amplio rango de necesidades, los sistemas embebidos sirven para cubrir necesidades específicas. (Heath, 2002)

En un sistema embebido, la mayoría de los componentes se encuentran incluidos en la placa base (tarjeta de vídeo, audio, módem, etc.).

Puesto que los sistemas embebidos se pueden fabricar por decenas de millares o por millones de unidades, una de las principales preocupaciones es reducir costes. Los sistemas embebidos suelen usar un procesador relativamente pequeño y una memoria pequeña para ello. Los primeros equipos embebidos fueron desarrollados por IBM en los años 1980.

Existen también plataformas desarrolladas por distintos fabricantes que proporcionan herramientas para el desarrollo y diseño de aplicaciones y prototipos con sistemas embebidos desde entornos gráficos. Algunos ejemplos de éstas son Raspberry Pi, Arduino o BeagleBone (Wikipedia, 2020).

3.3.2. Raspberry Pi

Hoy en día existen varios modelos de Raspberry Pi, todos ellos incluyen un procesador Broadcom, GPU, memoria RAM, puertos USB, HDMI, Ethernet, y 40 pines GPIO, a los cuales se puede acceder a través de los protocolos como UART, SPI e I2C.

Todas las versiones de Raspberry Pi utilizan una tarjeta SD (*Secure Digital*) para almacenar el sistema operativo.

En cuanto a la velocidad del procesador, va desde 700MHz hasta 1.5 GHz en el último modelo de Raspberry Pi.

3.3.4. Acceso a sensores en Raspberry Pi

Desde la primera versión de Raspberry Pi, ésta incluye 40 pines GPIO (*General Purpose Input/Output*) que se pueden configurar como entradas o salidas digitales para poder controlar periféricos, sensores, etc.

La mayoría de los pines son de propósito general y pueden configurarse como entradas o salidas. La descripción de los pines es la siguiente:

- Los pines 3 y 5 pueden configurarse como interfaz I2C.
- Los pines 8 y 10 pueden ser configurados como interfaz UART.
- El pin 12 puede ser configurado como salida PWM.
- Los pines 19,21,23,24, 26 se pueden configurar como interfaz SPI.
- En cuanto a los pines 27 y 28, estos no están disponibles, están reservados para añadir de forma opcional una memoria serie en las placas de expansión.
- Los pines 29,31,32,33,35,36,37,38 y 40 dan acceso a nuevas patas GPIO, las cuales pueden llevar a cabo otros usos adicionales.

Como se ha mencionado anteriormente, Raspberry Pi incluye pines que soportan protocolos de comunicación tales como I2C, SPI y UART.

El protocolo de comunicación en serie I2C (*Inter-Integrated Circuit*) fue desarrollado por Philips Semiconductors en 1982, y se utiliza sobre todo de forma interna para comunicar las distintas partes de un circuito. El bus I2C cuenta con sólo dos hilos, una línea de reloj (SCL) y una línea de datos (SDA), y tiene una arquitectura maestro-esclavo, en la que el dispositivo maestro inicia una comunicación con los esclavos y puede enviar o recibir datos de ellos. Por otro lado, los esclavos no pueden comenzar una conversación ni hablar

entre ellos de forma directa. Además, se trata de un bus síncrono en el que la señal de reloj siempre es controlada por el maestro, y compartida entre éste y el esclavo.

Por otro lado, el protocolo de comunicación SPI (Serial Peripheral Interface), es un protocolo utilizado por miles de dispositivos, el cual, a diferencia de I2C, utiliza más pines, pero consigue una mayor velocidad. Se trata de un protocolo serie, en el que se utilizan patas específicas para seleccionar la dirección de los dispositivos y, además, existe un maestro que es el que toma el control de la comunicación, y unos esclavos, los cuales asumen un papel pasivo.

Por último, el protocolo de comunicación UART (*Universal Asynchronous Receiver Transmiter*) es el dominante en los microprocesadores de gama baja. Hoy en día es casi imposible encontrar esta interfaz serie en un ordenador, ya que ha sido reemplazada por USB. Antes de USB, UART era utilizado para conectar el ratón, el teclado o el módem de comunicaciones, incluso era la interfaz utilizada por terminales para poder interactuar con un ordenador. Existe adaptadores UART-USB que permiten la comunicación de la UART de la Raspberry Pi con un puerto USB de un ordenador o de otra Raspberry Pi, lo cual permite usar la Raspberry Pi sin disponer de un monitor y sin tener conexión a ninguna red.

3.3.5. Sistemas Operativos en Raspberry Pi

El sistema operativo por excelencia para los modelos de Raspberry Pi es Raspberry Pi OS, también llamado Raspbian. Se trata de un sistema operativo adecuado para tareas tanto generales como específicas (Raspberrypi, 2020).Además de este, existen numerosos sistemas operativos para este sistema embebido. NOOBS es la opción más recomendada para los principiantes, ya que proporciona la opción de instalar fácilmente otros sistemas operativos, descargando estos previamente de Internet. RetroPie es otro sistema operativo cuyo propósito es más específico, ya que se emplea para emular distintos sistemas de ocio. OSMC es un sistema operativo que está pensado para poder usar la Raspberry Pi a modo de centro multimedia.

ChirpStack Gateway OS es el sistema operativo que se va a emplear en este proyecto, ya que tiene el propósito específico que este requiere. Se trata de un sistema operativo basado en Linux, el cual es de código abierto, y puede ser ejecutado en distintos modelos de *gateway* LoRa. Su objetivo principal es facilitar la configuración de un *gateway* LoRa,

con el menor número de pasos, así como el inicio de LoRaWAN y la pila de su servidor de red (ChirpStack, 2020).

3.4. Sección sobre entornos de programación (Python)

Python es un lenguaje de programación interpretado, el cual soporta orientación a objetos, programación funcional e imperativa. Para elegir este lenguaje en el proyecto, se han tenido en cuenta sus ventajas con respecto al ámbito de IoT, las cuales son:

- Es un lenguaje que no necesita ser compilado, algo que beneficia a nodos con el hardware restringido.
- Existencia de librerías para comunicarse con el hardware de los distintos nodos, así como para interactuar usando los distintos protocolos de IoT.

3.5. Presentación de la distribución de software para GW y Lora Server

ChirpStack es una pila LoRaWAN que proporciona componentes de código abierto para redes LoRaWAN. Todos ellos juntos forman una solución lista para usar que incluye una interfaz web fácil de utilizar para la administración de los dispositivos (ChirpStack, 2020).

Los componentes que se proporcionan son los siguientes (ChirpStack, 2020):

- *ChirpStack Gateway Bridge*
- *ChirpStack Network Server*
- *ChirpStack Application Server*
- *ChirpStack Gateway OS*

3.5.1. Gateway

Para la configuración del *gateway*, como se ha comentado anteriormente, se ha instalado el sistema operativo ChirpStack Gateway OS. Éste se distribuye ya compilado para distintas versiones de Raspberry.

Existen dos tipos de esta imagen que se distinguen según lo que ofrecen cada uno de ellos. ChirpStack Gateway OS Base ofrece ChirpStack Gateway Bridge, servicio que convierte los paquetes LoRa transportados en un JSON o Protobuf sobre MQTT, y ChirpStack Concentratord, demonio concentrador que se encarga de la configuración y comunicación con los chips basado en SX1301/SX1302, ya preinstalados, así como una utilidad CLI para poder configurar fácilmente el *gateway*. ChirpStack Gateway OS Full, además de

todas las características de la imagen anterior, proporciona ChirpStack Network Server y ChirpStack Application Server.

En este caso, se ha instalado la imagen ChirpStack Gateway OS Base, ya que es lo necesario para poner en funcionamiento un *gateway* en un Raspberry Pi junto con el concentrador iC880A. Por otro lado, se ha instalado ChirpStack Network Server y ChirpStack Application Server en un ordenador distinto porque se ha determinado mejor tener la pila del servidor separada del propio *gateway*.

La configuración de nuestro *gateway* se encuentra en el ANEXO I. INSTALACIÓN SOFTWARE GATEWAY LORA Y PILA DEL SERVIDOR DE RED CHIRPSTACK

3.5.2. LoRa Server

El servidor de red es uno de los componentes de la pila ChirpStack, cuya función es decodificar las tramas LoRaWAN que se reciben de puertas de enlace LoRa, de las que manipula su autenticación, los comandos MAC, la comunicación con el servidor de aplicaciones ChirpStack, y la programación de tramas de enlace descendente. Para publicar y recibir tramas de distintas aplicaciones, el servidor emplea Mosquitto.

Mosquitto es un servidor MQTT de código abierto, cuyo funcionamiento se basa en un servicio de mensajería con un publicador y un suscriptor, en el que los clientes se conectan con un servidor llamado *broker*. Los mensajes se disponen en distintos *topics* que están organizados, con la posibilidad que de cada cliente pueda publicar un mensaje en cada uno de ellos, y otros clientes se puedan suscribir a él, para que el *broker* les haga llegar los mensajes.

Por otro lado, el servidor ChirpStack almacena los datos del *gateway* en PostgreSQL, una base de datos relacional orientada a objetos.

Por último, Redis se emplea para almacenar los datos de sesión de los dispositivos, y los datos que no persisten tales como bloqueos distribuidos, conjuntos de des duplicación y metadatos.

Dentro de la pila del servidor de red LoRaWAN se encuentra el servidor de aplicaciones ChirpStack, el cual se encarga del tratamiento de los datos recogidos desde un dispositivo LoRa, el manejo y cifrado de la carga útil de la aplicación, y el control de la solicitud de unión de los dispositivos.

Este servidor de aplicaciones ChirpStack, proporciona una interfaz web para poder administrar organizaciones, dispositivos, usuarios y aplicaciones. Además, los datos de un dispositivo pueden enviarse y/o recibirse mediante MQTT, HTT y escribirse de forma directa en InfluxDB.

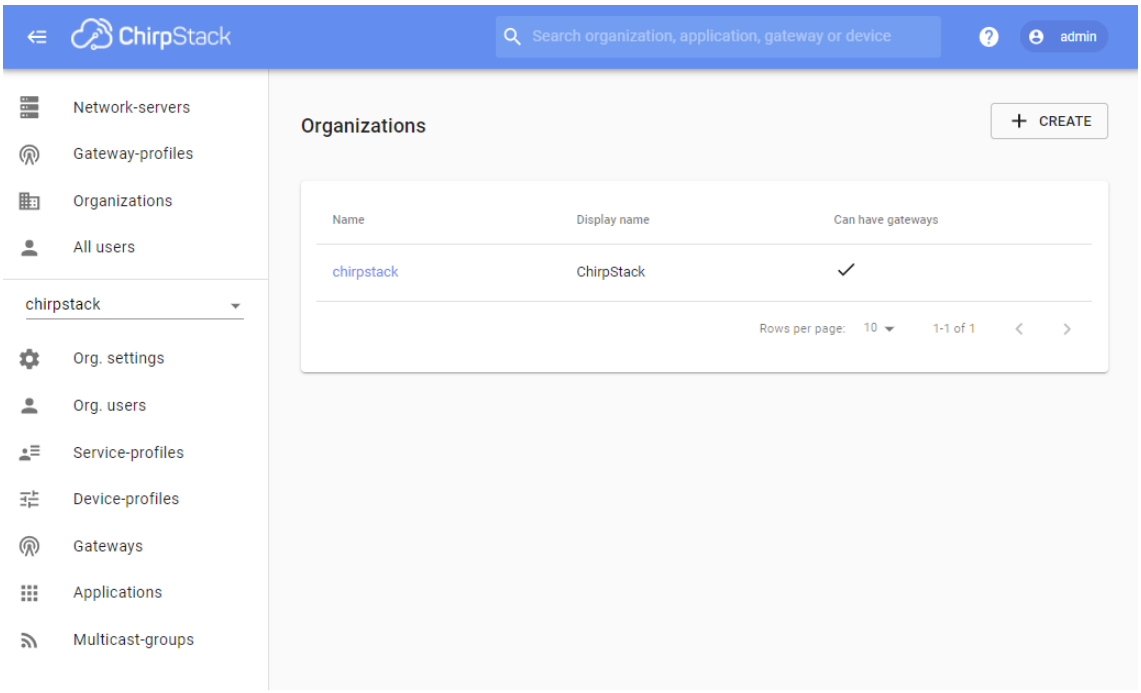

Figura 16. Interfaz Web del servidor ChirpStack.

3.6. InfluxDB

InfluxDB (InfluxData, 2020) es la base de datos que se ha utilizado en este proyecto. Ésta fue elegida ya que se trata de un servicio de código abierto ligero y potente, y con un lenguaje similar a SQL. Es una base de datos de series de tiempo, escrita en Go, que fue desarrollada por InfluxData, la cual está optimizada para un rápido almacenamiento de los datos y recuperación de ellos. Usualmente es utilizada para campos como telemetría, monitoreo de operaciones, datos de sensores de IoT, análisis en tiempo real, etc.

InfluxDB escucha en el puerto 8086. En cuanto a la estructura de los datos, está formada por puntos, series y medidas. Los puntos están constituidos por pares clave-valor, y cuando estos se agrupan en conjuntos de pares, definen una serie. Por último, la agrupación de series por un identificador de cadena da lugar a una medida.

3.7. Grafana

Grafana (Grafana, 2020) es un software de código abierto que se utiliza para visualizar datos y analizar aplicaciones, aunque también para otros sectores como el industrial, la

domótica, el clima, etc. Permite crear gráficos a partir de múltiples fuentes, entre las que se incluyen bases de datos de series de tiempo tales como OpenTSDB e InfluxDB.

Grafana se conecta a las fuentes de datos y transmite la información al navegador de cada uno de los clientes. También proporciona una forma de consultar la fuente de datos especificando una ventana de tiempo y granularidad o simplemente mostrar los datos entrantes en tiempo real tal y como se entregan desde la fuente. Permite configurar alertas empleando umbrales programables en una o más métricas.

En este proyecto se va a utilizar este software para representar de una manera gráfica intuitiva los datos almacenados para demostrar un correcto funcionamiento de todo el sistema.

CAPÍTULO 4. ARQUITECTURA DEL SISTEMA

La arquitectura del sistema en este proyecto se muestra en la Figura 17. El enfoque principal del trabajo es el montaje y correcto funcionamiento del *gateway* LoRa, así como de la unidad a bordo (OBU).

El *gateway*, está formado por una Raspberry Pi, un concentrador LoRa iC880A, y un *Gateway* Backplane ic880A, que conecta la Raspberry con el concentrador. Estos tres componentes forman el *gateway* LoRa que se puede ver en la Figura 17. Éste se encarga de reenviar los mensajes que recibe de la unidad a bordo en este caso, a un servidor Lora, utilizando protocolos como MQTT.

La unidad a bordo usada en el proyecto (Santa, Bernal-Escobedo, & Sanchez-Iborra, 2020), provista de conectividad LoRaWAN, es capaz de alcanzar los servicios finales en Internet para proporcionar datos, a los que se puede acceder mediante distintas plataformas para ofrecer funciones como el monitoreo de la contaminación, rutas inteligentes, monitoreo del estado de la unidad móvil, etc. Se compone de dos placas principales, de las cuales la primera de ellas incluye la CPU y la memoria del sistema usando un chip (SoC) o computadora a pequeña escala. La comunicación de bus a través de I2C y USB se emplean para conectar el módulo de la CPU con los sensores y transceptores de comunicación.

En las tareas de evaluación del proyecto, la unidad es colocada en una bicicleta, como se explicará en la sección 7.1.4 Unidad a bordo, pero también se podría colocar en otros sistemas móviles como patinetes eléctricos, motos, etc.

Los sensores que se incluyen en la unidad son: una unidad inercial, que proporciona datos de aceleración y velocidad angular; un conjunto de sensores de medición meteorológica para temperatura, humedad o presión atmosférica; y una unidad de contaminación del aire, capaz de detectar niveles de CO2, y partículas orgánicas volátiles. Además, se hace uso de una pantalla de estado para informar al usuario de las principales condiciones de funcionamiento y de las mediciones realizadas en la propia unidad móvil.

Dado que los transmisores de comunicación están normalmente provistos de una única interfaz serie, y que estos computadores de pequeña escala tienen un número limitado de UART, la unidad incluye una placa de comunicación adicional al diseño. Ésta placa integra una solución de USB a serie para conectar chips de comunicación como GPS y nuestro transceptor LPWAN de referencia para LoRaWAN. Se han colocado también las antenas adecuadas, conectadas a los transceptores de comunicación, para conseguir una solución a pequeña escala con el objetivo de poder reducir el tamaño de la unidad.

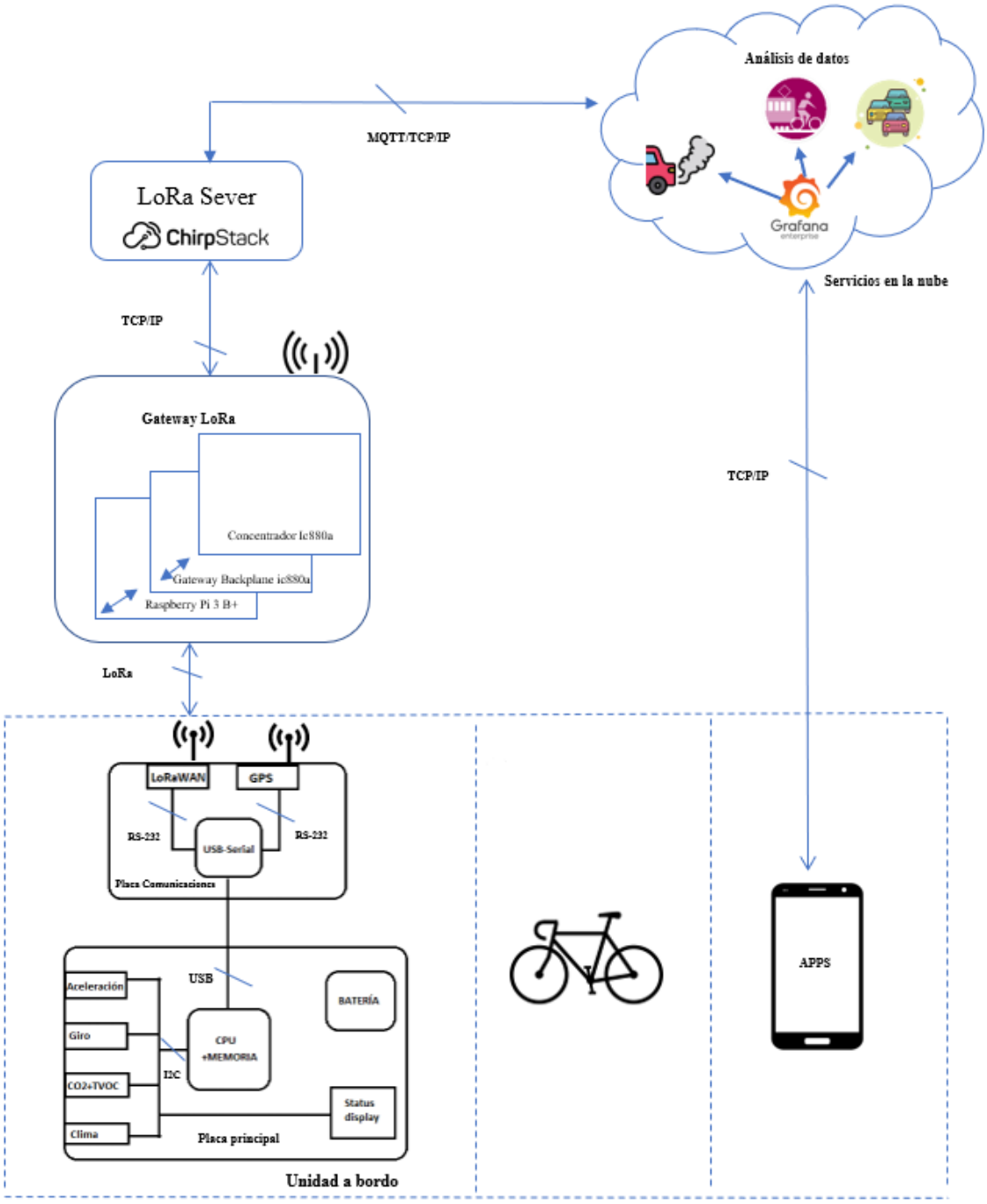

Figura 17. Arquitectura del sistema.

CAPÍTULO 5. IMPLEMENTACIÓN DEL GATEWAY

En este apartado se va a analizar la arquitectura del *gateway*, así como su configuración y registro para poder recibir los paquetes de datos de los sensores mediante LoRa, y a continuación enviarlos a un servidor online.

5.1. Componentes utilizados

El *gateway* ensamblado en este proyecto está constituido por un concentrador iC880A, una Raspberry Pi, y una antena.

El iC880A es un concentrador LoRaWAN diseñado para recibir paquetes de datos de distintos dispositivos finales usando hasta 8 canales en paralelo, los cuales envían los datos con diferentes factores de dispersión (SF) y anchos de banda.

Figura 18. Concentrador iC880A.

El iC880A LoRaWAN Gateway Backplane v2.1 es una placa adaptadora que se utiliza para conectar el concentrador iC880A-SPI a una Raspberry Pi de 40 pines.

Figura 19. LoRaWAN Gateway Backplane.

La Raspberry Pi 3 modelo B+ se emplea para el procesamiento de los paquetes entrantes y salientes de LoRaWAN, así como el intercambio de estos con la placa concentradora. Se conecta al concentrador iC880A a través de la placa adaptadora anterior. Mediante la Raspberry Pi, la placa iC880A es alimentada, y los paquetes de datos son enviados al servidor.

Figura 20. Raspberry Pi 3 B+.

Para el puerto de antena iC880A-SPI, es necesario un cable "*pigtail*" hembra EE. UU. a SMA, de una longitud aproximada de 100mm.

Figura 21. Cable pigtail para antena.

En este proyecto, se ha utilizado una antena SMA para el iC880A-SPI con una ganancia de 2 dBi, cuyo rango de frecuencia se encuentra entre 824 y 896 MHz.

Figura 22. Antena SMA para iC880A-SPI.

Una vez vistos los componentes, se va a realizar un desglose de los precios de cada uno de ellos para el montaje del *gateway*, simplemente teniendo en cuenta la parte del hardware, para comparar el precio de nuestra solución con el precio de otras pasarelas comerciales que hemos presentado anteriormente y que ofrecen una funcionalidad base equivalente.

CONCEPTO	CANTIDAD	PRECIO
Concentrador iC880A LoRaWAN	1	189,11 €
iC880A LoRaWAN Gateway Backplane v2.1	1	84,55€
Raspberry Pi 3 Modelo B	1	37,44€
Cable Pigtail para iC880A-SPI	1	6,50€
Antena para iC880A-SPI and Life Gateway	1	6,50€
	TOTAL	324,10€

Tabla 4. Presupuesto para el Gateway LoRa.

Tras el cálculo total del coste de todos los componentes, podemos apreciar que es mucho menor que el de los *gateways* comerciales mencionados anteriormente, cuyos precios eran de 719€, 2.500€ y 400€.

5.2. Configuración del Gateway

Este apartado se va a enfocar únicamente en la configuración del software para poner en funcionamiento el *gateway*.

En primer lugar, establecemos el Network Server para que se puedan ver los datos recibidos por el *gateway* en la web. Para ello, nombramos el Network Server e indicamos su IP y su puerto, que como podemos ver en la siguiente imagen, hemos usado localhost y el puerto 8000, puerto por defecto establecido en la configuración del *gateway*.

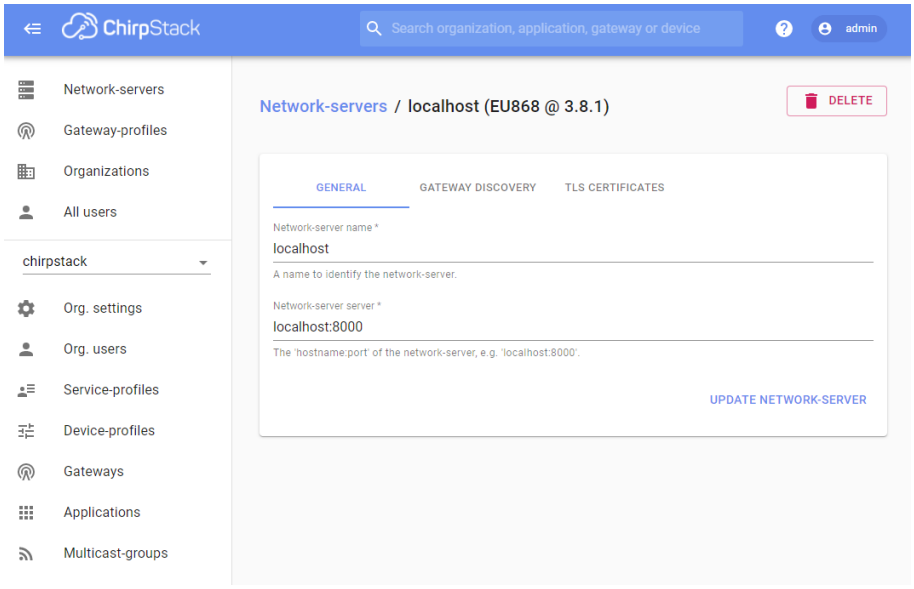

Figura 23. Pantalla Servidores de Red.

A continuación, creamos un perfil para el *gateway*, es decir, introducimos su nombre, los canales que queremos que se encuentren escuchando, y si queremos algún canal extra, introducimos la frecuencia, el ancho de banda (125, 250, 500) kHz y el Factor de dispersión.

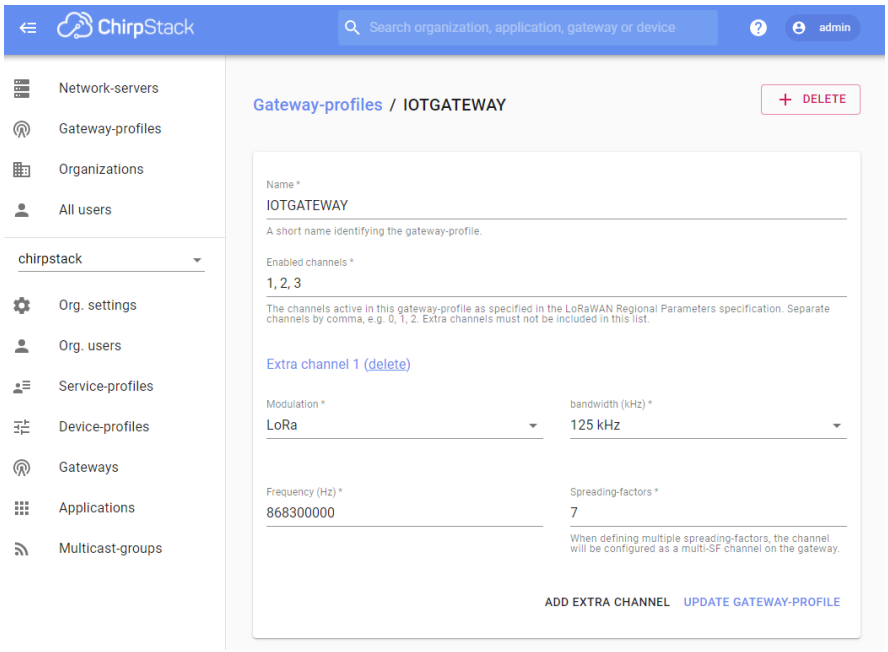

Figura 24. Captura configuración perfil Gateway.

El siguiente paso es añadir nuestro propio *gateway*, introduciendo su nombre y descripción, y seleccionando el perfil de *gateway* que hemos creado anteriormente.

También podemos incluir las coordenadas GPS de la Raspberry, añadiendo la latitud, la longitud y la altura.

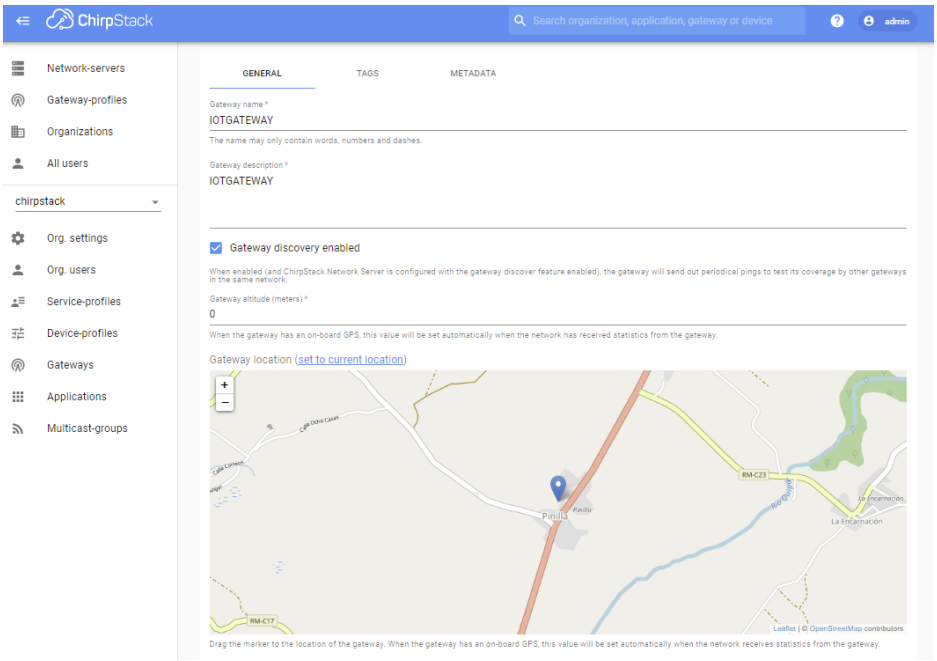

Figura 25. Captura de la Localización del Gateway.

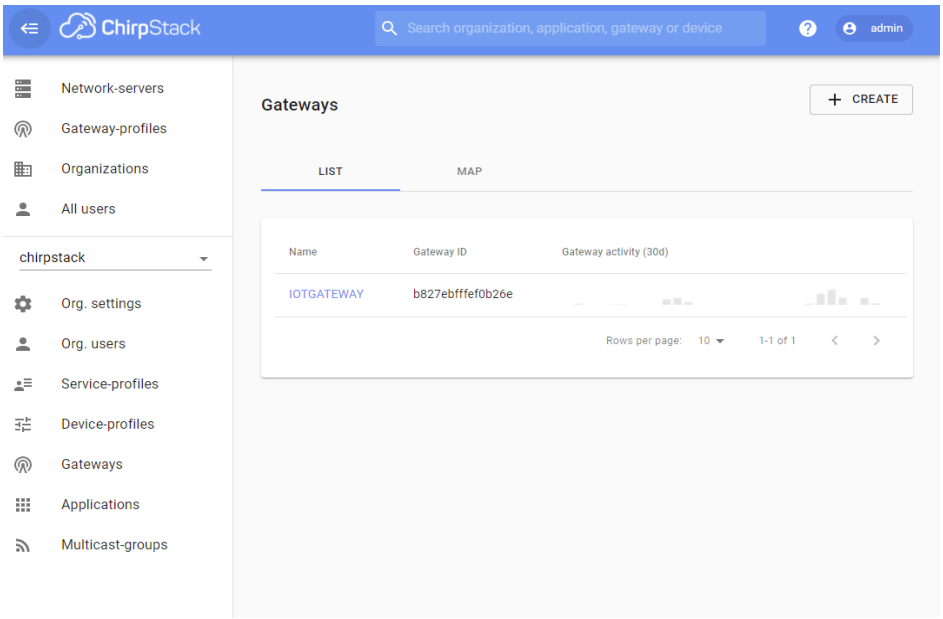

Figura 26. Listado de Gateways configuradas.

Acto seguido, creamos el Perfil de Servicio, el cual define determinadas características que pueden ser usadas en una organización.

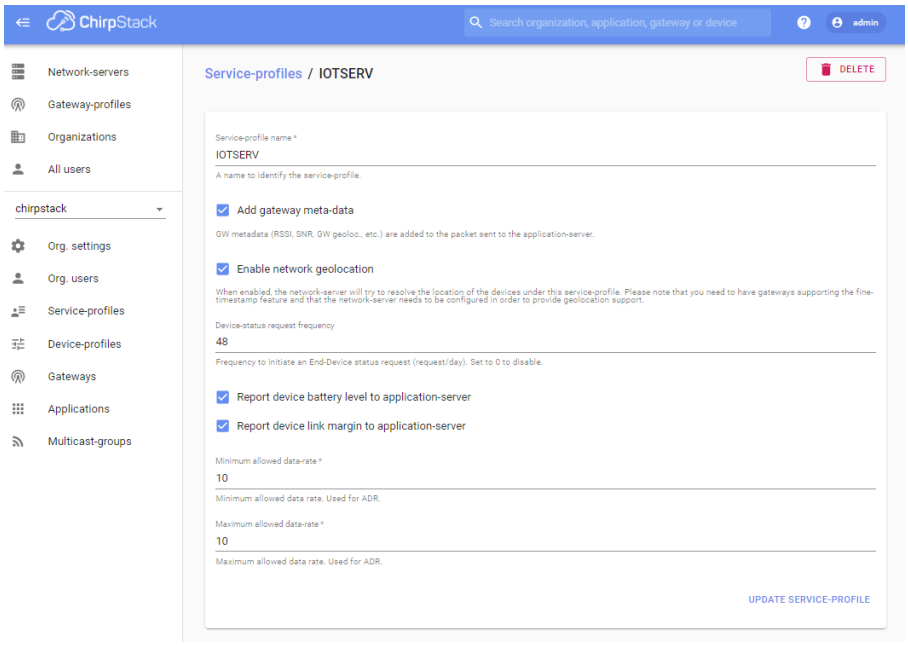

Figura 27. Configuración de los perfiles de servicio.

Seguidamente, definimos un Perfil de Dispositivo para definir todas las propiedades de nuestro nodo, para lo que establecemos por ejemplo la versión de LoRaWAN, la clase, el tipo de activación (ABP u OTAA), el códec de *payload* (Cayenne LPP, JavaScript personalizado o ningún otro), etc.

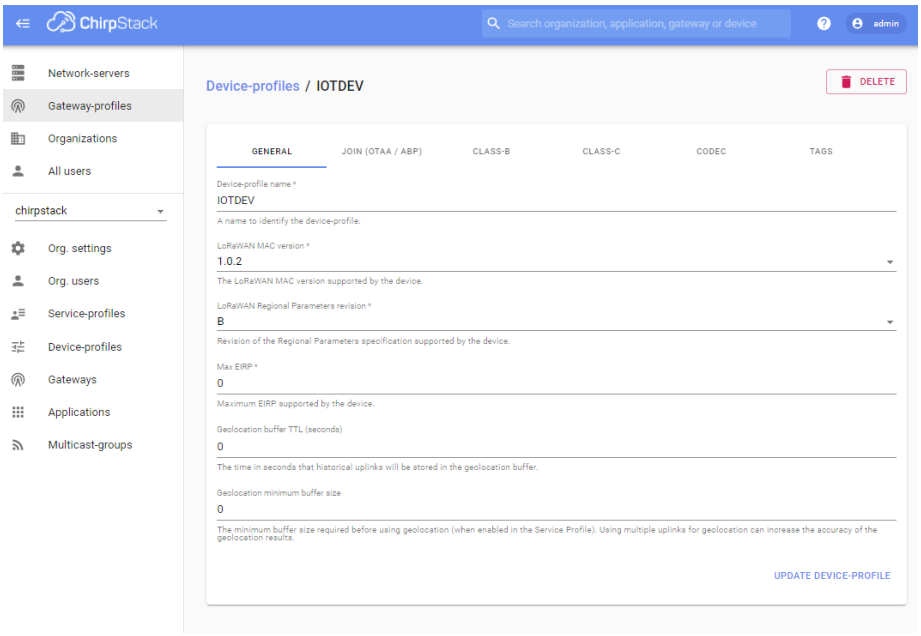

Figura 28. Configuración perfil de dispositivo.

Una vez realizados estos pasos, vamos a añadir nuestra aplicación, de la que tendremos que añadir su nombre, su descripción, así como un perfil de dispositivo.

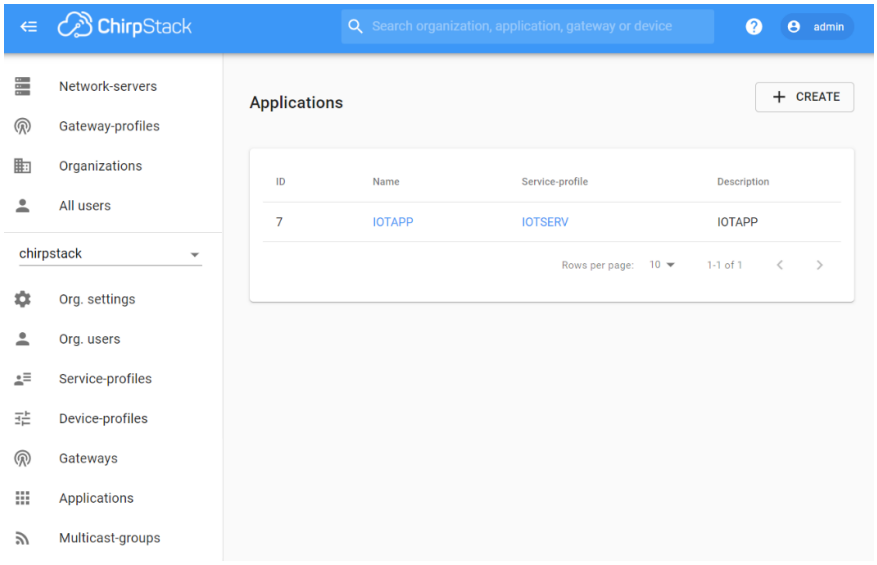

Figura 29. Lista de aplicaciones configuradas.

Dentro de la aplicación se encuentra el apartado de Integraciones en el que aparecen dos opciones, Influx DB y HTTP. En este caso hemos usado Influx DB para almacenar los datos de los distintos sensores y después extraerlos para su posterior representación.

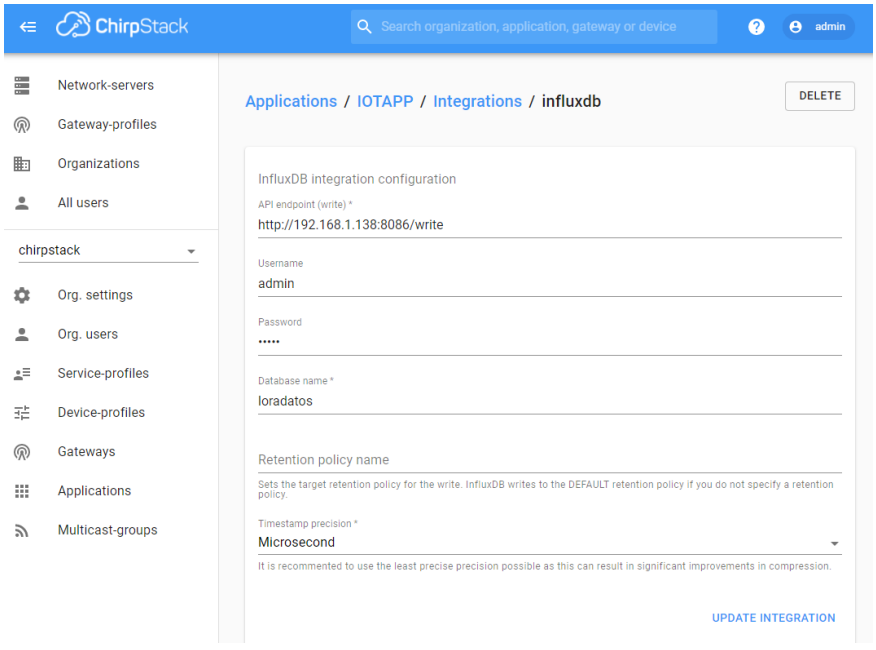

Figura 30. Integración InfluxDB.

Una vez añadida la aplicación, dentro de su pestaña, tendremos que establecer un nuevo dispositivo con su nombre, descripción y perfil del dispositivo al que pertenece. Además,

es necesario configurar las distintas claves correspondientes, según el tipo de unión (ABP u OTAA), siendo necesarias la *Network Sesion Key*, *Application Session Key* y *Application key para ABP*, y una única clave *Application Key* para OTAA.

Una vez realizada toda la configuración, podemos observar que en el apartado del *gateway* aparecen sus datos y la última vez que ha estado activa, así como un gráfico con el detalle de la cantidad de tramas recibidas y transmitidas.

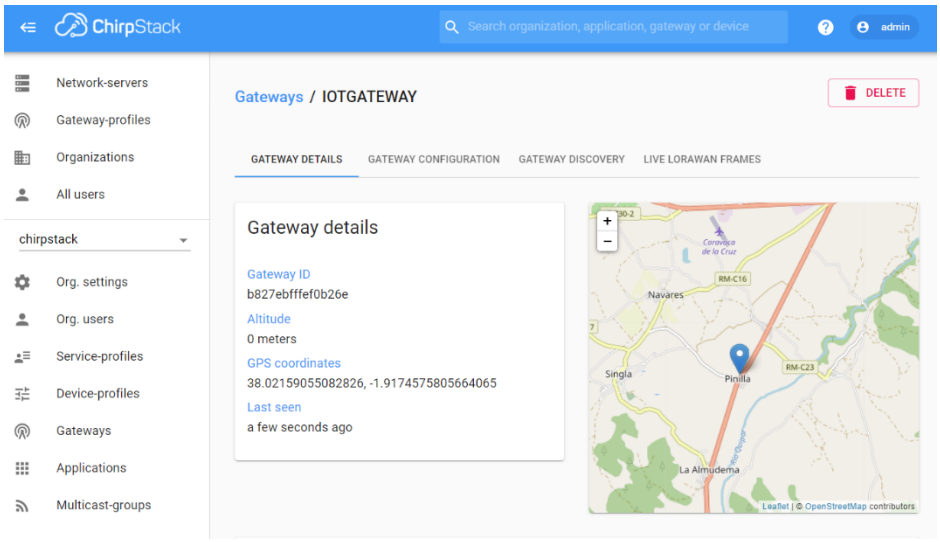

Figura 31. Detalle configuración Gateway.

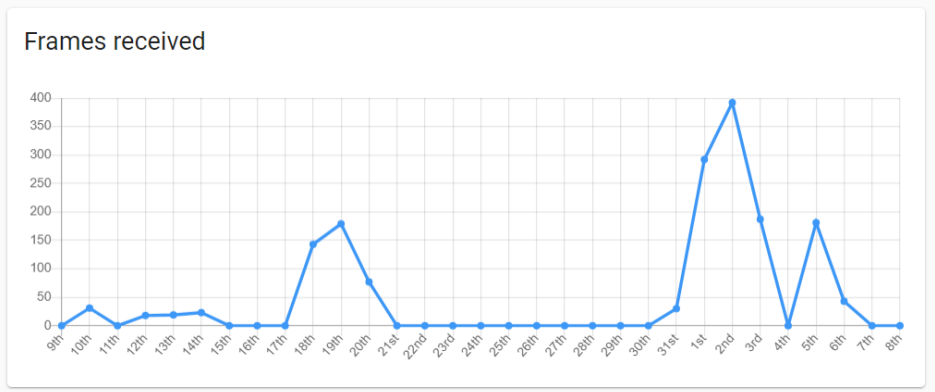

Figura 32. Tramas recibidas.

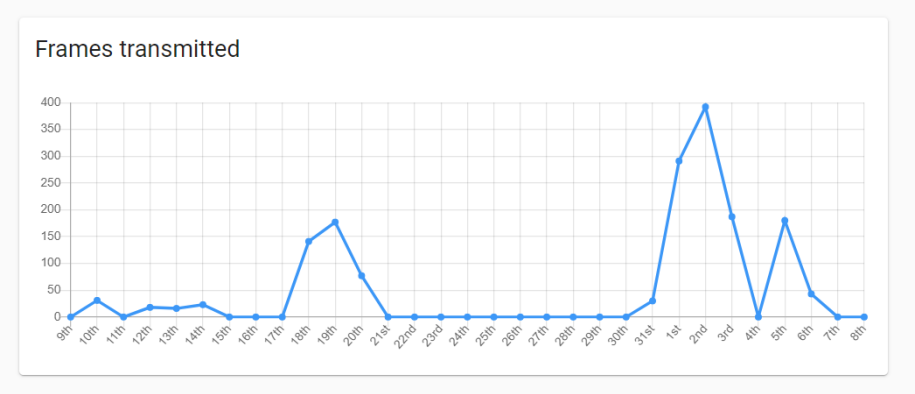

Figura 33. Tramas transmitidas.

En el apartado de dispositivos también podemos observar la última vez que éste fue visto en la red, el nivel de batería, y el margen de enlace en dB, que es el resultado de restar la sensibilidad del receptor y la potencia mínima que se espera recibir.

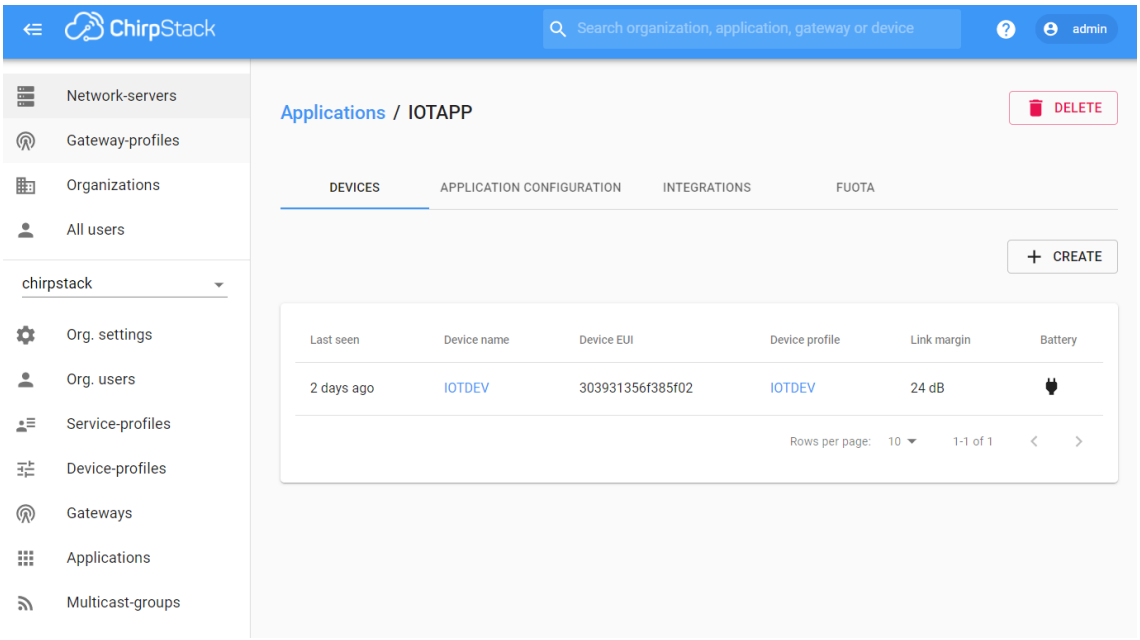

Figura 34. Detalles del dispositivo.

CAPÍTULO 6. PROGRAMACIÓN DE UN NODO MÓVIL

En este capítulo se explica cómo se ha realizado la programación de la unidad móvil, dando detalles de cada uno de los componentes utilizados para su funcionamiento.

En la Figura 35 se representa un diagrama de flujo que resume el funcionamiento de la programación software, el código Python que hace funcionar los sensores descritos en este mismo apartado, para extraer datos de ellos y poder representarlos.

En primer lugar, se inicializa LoRa, GPS y Serial. Serial se inicializa cuando nos conectamos mediante puerto serie a la placa de comunicaciones, LoRa al conectar el dispositivo final o en este caso unidad móvil a la red, estableciendo unas determinadas claves y parámetros de unión y, por último, inicializamos el GPS de nuestra placa de comunicaciones.

Una vez realizada la inicialización, comprobamos si los datos del GPS están disponibles. En determinadas ocasiones, los datos del GPS no están disponibles si nos encontramos por ejemplo en interiores, si el dispositivo no se ha inicializado correctamente, o simplemente porque hay que esperar unos segundos para que éste empiece a tomar datos.

Si los datos del GPS están disponibles, los procesamos, y continuamos leyendo los datos del resto de los sensores, tales como los inerciales, de temperatura, contaminación, etc. Tras extraer los datos, formamos una trama LoRa, y procedemos a su envío.

A continuación, ciertos datos serán mostrados por la pantalla OLED para que el usuario los pueda conocer en todo momento, y el sistema se pondrá en reposo durante 10 segundos.

Por otro lado, si los datos del GPS no están disponibles, esperamos 5 segundos y volvemos al paso de la inicialización, es decir, al primer paso para continuar de nuevo todo el diagrama.

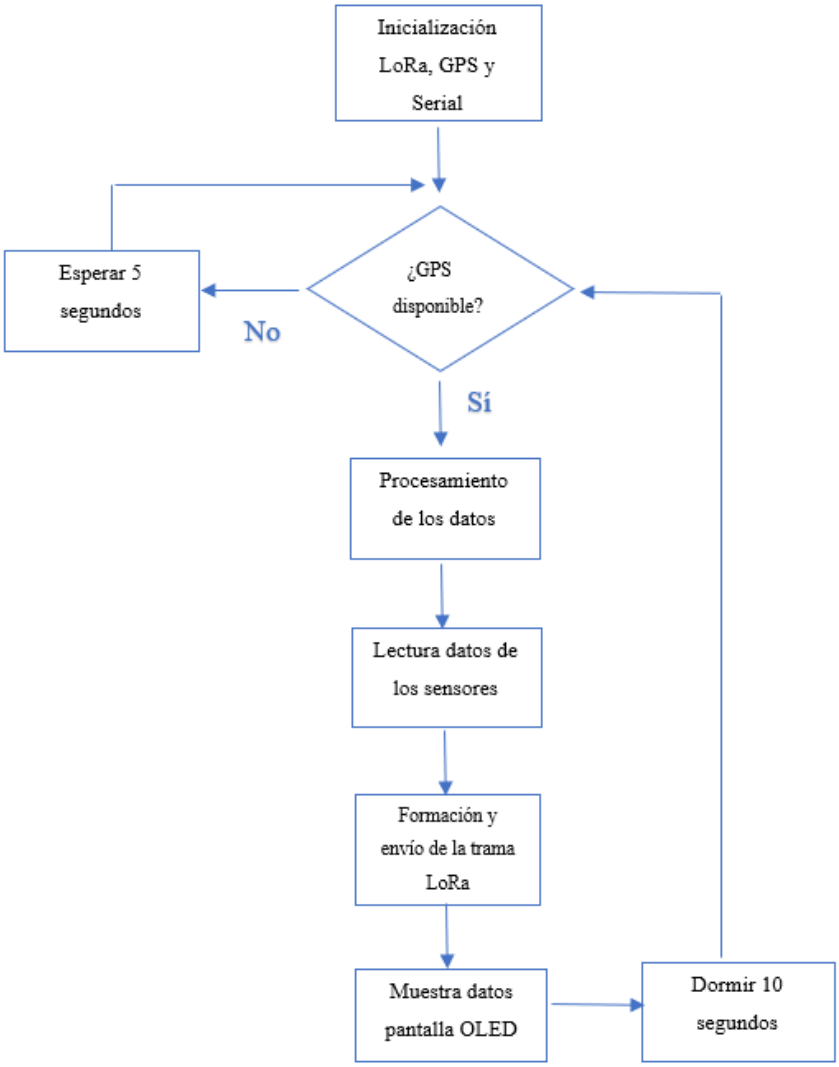

Figura 35. Diagrama de flujo del software desarrollado para el nodo móvil.

6.1. Prototipo de la unidad de a bordo

La unidad de procesamiento y comunicación descrita anteriormente se ha desarrollado utilizando componentes comerciales y produciendo nuevas piezas desde cero en el momento que ha sido necesario. Esta unidad se ha diseñado dentro del grupo de investigación GIRTEL de la UPCT (Santa, Bernal-Escobedo, & Sanchez-Iborra, 2020). La

Figura 36 muestra el interior de la unidad. Sus dimensiones son de 15x7x5 cm.

Las partes principales, siendo también las más grandes, son el SOC utilizado como CPU, la batería y la placa de comunicaciones. El SOC es una Raspberry Pi Zero W, con una CPU ARM de 1 GHz y 512 MB de RAM.

La batería es el modelo RS PRO-PB-A5200, la cual trabaja a 5V y con 5 Ah de capacidad.

La placa de comunicaciones es un nuevo diseño destinado a integrar transceptores. Se coloca sobre el SOC, como se muestra en la

Figura 36, y resuelve el problema de tener un único puerto serie efectivo, y también brinda flexibilidad al separar las interfaces de comunicación. Esta placa integra un chip FTDI FT4332 para crear cuartos puertos serie desde la interfaz USB proporcionada por la Raspberry Pi Zero. El receptor GPS Lantronix A2235-H, el transceptor Murata CMWX1ZZABZ-093 LoRaWAN y el módem Quectel BC95-G NB-IoT, están fijados a esta placa. Además, se emplean antenas regulares de sub-giga hertzios y de 1 a 2 Ghz para LoRa y NB-IoT. Se incluye una pantalla OLED Solomon SSD1306 en el lado inferior derecho, como se puede ver en la Figura 37. La versión actual del prototipo incluye el siguiente conjunto de sensores para permitir aplicaciones de monitoreo de ciudades inteligentes: una unidad combinada de presión, humedad, temperatura, compuestos orgánicos volátiles y CO2 con sensores CCS811 y BME280, y un sensor inercial LSM9DS1 (Santa, Bernal-Escobedo, & Sanchez-Iborra, 2020). El presente proyecto ha desarrollado un software embebido que corre en la unidad y accede a estos sensores para recoger datos de forma continua.

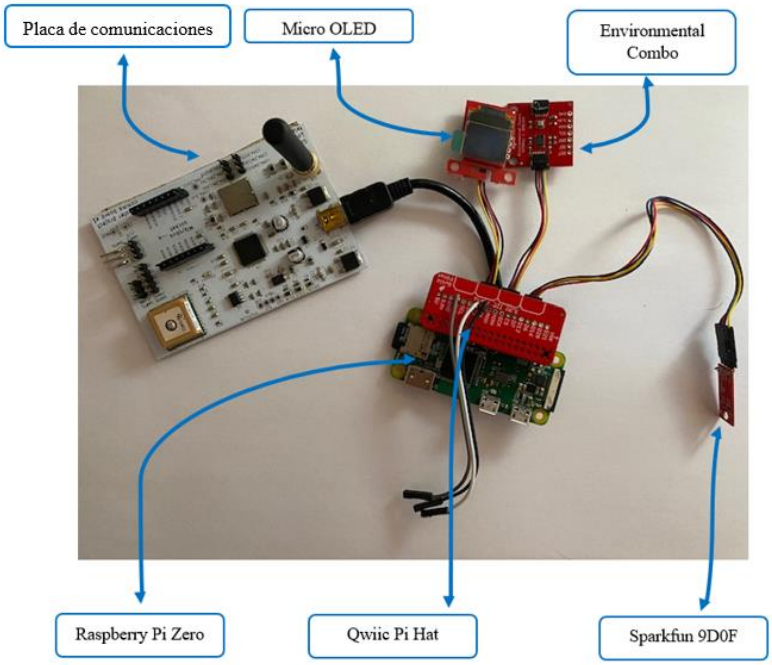

Figura 36. Componentes incluidos en la Unidad a bordo (OBU).

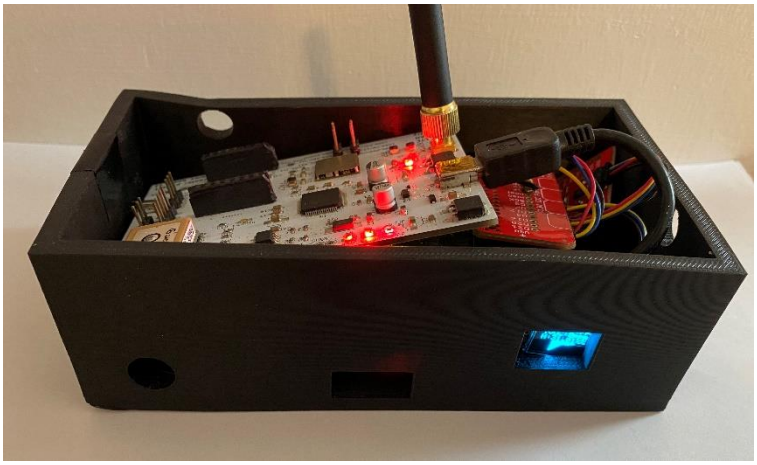

Figura 37. Unidad a bordo (OBU).

A continuación, se van a explicar con detalle los principales componentes empleados para la monitorización.

6.1.1 Qwiic HAT para Raspberry Pi

Con este componente podemos utilizar de forma sencilla el ecosistema Qwiic de Sparkfun manteniendo el uso de la Raspberry Pi. Su función es conectar el bus I2C de la Raspberry a una serie de conectores Qwiic.

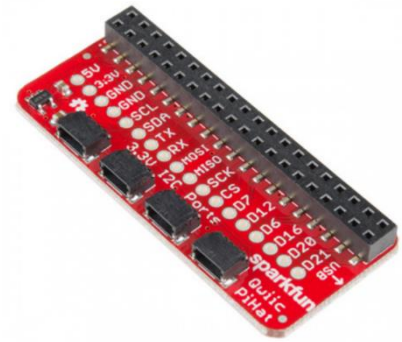

Figura 38. Qwiic HAT.

6.1.2 Sparkfun Environmental Combo CCS811/BME280

Este sensor se encarga de obtener valores sobre la calidad del aire gracias a los circuitos integrados CCS811, el cual mide el CO2 en partes por millón (PPM) y los compuestos orgánicos volátiles totales en partes por millón (PPM), y BME280, que proporciona lecturas de humedad, presión barométrica y temperatura.

El Sparkfun Environmental Combo se comunica exclusivamente a través de I2C, a través del uso del sistema Qwiic, el cual emplea los conectores Qwiic polarizados de 4 pines.

La dirección I2C del CCS811 por defecto es la 0X5B, la cual se puede cambiar a 0x5A agregando una soldadura por la parte posterior para cerrar el puente ADR1, y la dirección por defecto del BME280 es la 0x77, y también se puede cambiar a 0x76 añadiendo una soldadura en el puente ADR2.

En este proyecto, hemos empleado este sensor para medir el CO2, los compuestos orgánicos volátiles totales, la humedad, la presión y la temperatura, en distintos escenarios.

Figura 39. Sparkfun Environmental Combo.

6.1.3 Sparkfun 9D0F

Este sensor emplea el sistema de detección de movimiento LSM9DS1, el cual incluye un acelerómetro de 3 ejes, un giroscopio de 3 ejes y un magnetómetro de 2 ejes. Se trata de una IMU de 9 grados de libertad (9DoF), siendo capaz de medir tres propiedades claves en el movimiento (la velocidad angular, la aceleración y el rumbo) en un único circuito integrado. Cada uno de los sensores LSM9DS1 mide cada una de estas propiedades de movimiento en tres dimensiones, dando lugar a nueve datos.

Los sensores admiten un gran espectro de rangos, siendo la escala del acelerómetro de ± 2, 4, 8 o 16 g, la del giroscopio de ± 245, 500 y 2000 ° / s, y la del magnetómetro de ± 4, 8, 12 o 16 gauss.

El LSM9DS1 emplea la interfaz serie I2C, siendo la dirección del magnetómetro la 0x1E y la dirección del acelerómetro y el giroscopio la 0x6B.

Con este sensor inercial hemos medido la aceleración de la unidad móvil y el giro de ésta. En este caso los datos del magnetómetro no los hemos usado.

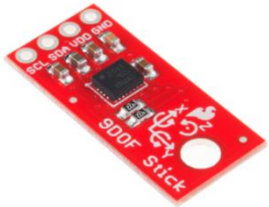

Figura 40. Sparkfun 9D0F.

6.1.4 Sparkfun Micro OLED

La SparkFun Micro OLED incluye una pequeña pantalla nítida en la que se pueden mostrar cantidad de gráficos e información de diagnóstico sin recurrir a la salida en serie. Lo más importante de este Micro OLED es la facilidad para controlarlo mediante la interfaz SPI o I2C.

El Micro OLED Breakout facilita el acceso a 16 de los pines del OLED, aunque solamente son necesarios aproximadamente la mitad para que la pantalla funcione.

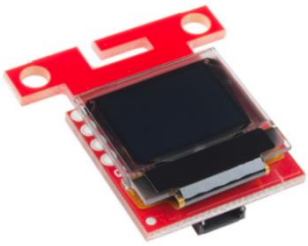

Figura 41. Sparkfun MicroOLED.

En este caso, en la pantalla se muestran determinados datos de la monitorización, tales como el CO2 en ppm, los compuestos orgánicos volátiles totales (TVOC) en ppm, la temperatura, y la humedad.

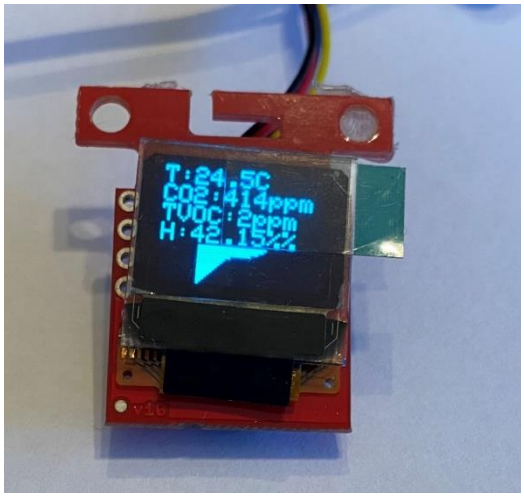

Figura 42. Salida por pantalla de los distintos valores.

Para comprobar que las direcciones de los sensores son las correctas y que han sido reconocidas por la Raspberry Pi zero, empleamos el siguiente comando:

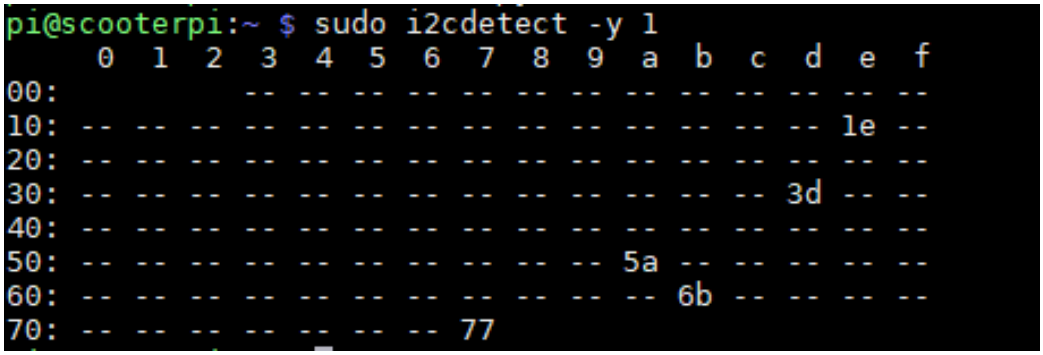

Figura 43. Comprobación de direcciones.

6.2. Extracción y envío de datos de los sensores

Para extraer datos de cada uno de los sensores, hemos instalado las librerías necesarias que se detallan en el ANEXO II. INSTALACIÓN LIBRERÍAS PARA EL CONTROL DE LOS SENSORES

Sin embargo, para enviar esos datos, hemos tenido que comunicarnos con el módem LoRa, y para ello emplear los denominados comandos “AT”.

Se denominan comandos “AT” porque su significado es “Atención”, y se utilizan para la comunicación con los módems, su configuración y parametrización.

En nuestro caso empleamos un módem LoRa para enviar mensajes LoRa hacia el servidor con datos recogidos de los distintos sensores. Lo que hace el módem básicamente es añadir los datos recogidos de los sensores a una trama LoRaWAN.

Los comandos AT que han sido utilizados en este proyecto son los siguientes:

Comando AT	Descripción
AT	Este comando se usa para chequear si en enlace está funcionando adecuadamente.
AT+TCONF=?	Con este comando se puede acceder a la configuración LoRa establecida.
AT+APPEUI=	Este comando permite al usuario establecer el identificador global de la aplicación.
AT+APPKEY=	Este comando sirve para establecer la clave de sesión de aplicación.
AT+NJM=1	Este comando sirve para establecer el modo de unión. Con el 0 se establece el modo de unión ABP, y con el 1, como en nuestro caso, el OTAA.
AT+CLASS=C	Establecemos la clase del dispositivo como la clase C.
AT+JOIN	Con este comando se hace una petición de unión a la red.
AT+NJS=?	Este comando sirve para que el usuario pueda conocer el estado de la unión, es decir, si el dispositivo se ha unido a la red o no.

AT+SEND=20	Este comando permite enviar datos de texto a través del puerto indicado de la aplicación. En este caso el puerto es el 20.

Tabla 5. Comandos AT.

Existen muchos más comandos AT para comunicarse con el módem, pero éstos son los que se han utilizado en este proyecto para la comunicación y el envío de los datos de los sensores hacia el servidor.

CAPÍTULO 7. EVALUACIÓN DEL SISTEMA

7.1. Diseño del montaje

7.1.2 Gateway

Tras varias pruebas de cobertura, el *gateway* se ha colocado en una vivienda de la localidad de Pinilla de San José, Caravaca, concretamente en la repisa de una de las ventanas que da cobertura a toda la zona de la carretera principal, donde se van a realizar las pruebas. Debido a que no tiene encapsulado aún, se ha colocado ahí el *gateway* para poder protegerlo en el caso de que se den condiciones meteorológicas adversas que puedan dañarlo, como lluvia, viento, etc.

Debido al uso para la validación del sistema de una antena de 2 dB, los valores de alcance y cobertura no llegan a ser extremadamente amplios, pero sí suficientes para realizar las pruebas requeridas.

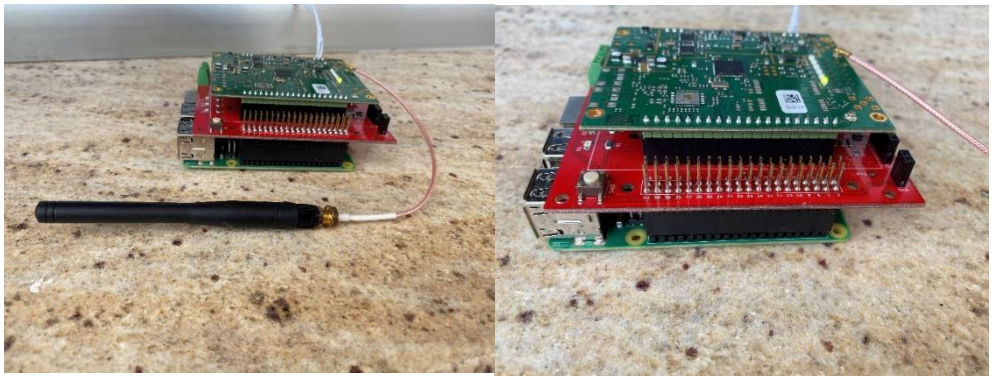

Figura 44. Situación Gateway LoRa.

7.1.3 Servidor LoRa

El servidor LoRa ha sido instalado en un portátil MSI que incluye un procesador Intel Core i7 y 16 GB de memoria RAM. El portátil se encuentra situado en el interior de la vivienda, conectado a una red Wifi, a través de la cual mantienen una conexión el Servidor LoRa y el *gateway*. Gracias a ésta es posible recibir datos del *gateway* para poder representarlos.

Para poder enviar datos desde el nodo móvil al *gateway*, en primer lugar, es necesario que el dispositivo se una a la red, para lo cual es necesario el envío por parte de éste de una serie de claves. Una vez aceptada la unión, el dispositivo ya puede empezar a enviar

los datos al servidor. En la Figura 45 se pueden observar los mensajes de solicitud y aceptación de la unión.

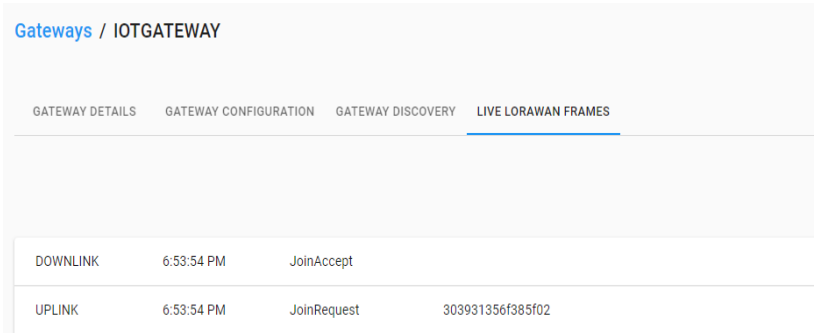

Figura 45. Mensajes de unión en el servidor.

Cuando la unidad envía datos al servidor, éstos aparecen en la pestaña de tramas LoRaWAN al igual que los de unión. Observando dichos mensajes en detalle, aparecen rellenos todos los campos del mensaje LoRa, con las características de esa trama en concreto, así como los datos que se han enviado en JSON. Esto se puede apreciar a continuación.

```
ApplicationID:”7”
ApplicationName:”IOTAPP”
DevEUI:”303931356f385f02”
rxInfo:
        0:
        gatewayID:”b827ebfffef0b26e”
        tiempo: nulo
        timeSinceGPSEpoch: nulo
        rssi: -93
        loRaSNR: 10,8
        canal: 0
        rfChain: 1
        tablero: 0
        antena: 0
        ubicación:
                latitud:
                longitud:
        uplink: “16ab4d2b-3740-4851-84b3-f99369c0545b”
        crcStatus: “CRC_OK”
txInfo:
        frecuencia: 868100000
        modulación: “LORA”
loRaModulationInfo:
```

```
        ancho de banda: 125
        spreadingFactor:12
        codeRate:”4/5”
        polarización Inversión: falso
dr: 0
datos:” ZXI5OGszc2RnZmt1LC05Ljc5NiwtMC4zNDksMC41NzcsMi43OCwwLjMzLDUuNjQsNDA4”
objectJSON: “{“GPS”:” eyy8k3sdgfku “,” aceleración_x “:” -9.796 “,” aceleración_y “:”
-0.349 “,” aceleración_z “:” 0.577 “,” co2 “:” 408 “,” giroscopio_x “:” 2.78 “,”
giroscopio_y “:” 0.33 “,” giroscopio_z “:” 5.64 “}”
```

7.1.4 Unidad a bordo

La unidad se ha colocado en una bicicleta, a la que la hemos atado con bridas para tener una sujeción correcta. Con esta unidad móvil se van a realizar todas las pruebas requeridas para la evaluación completa del sistema. En la Figura 46 podemos ver la unidad a bordo sobre la bicicleta.

Figura 46. Unidad a bordo colocada en una bicicleta.

7.1.5 Base de datos

Para poder utilizar Influx DB, hemos tenido que descargar el paquete InfluxDB, descomprimirlo, y ejecutar los archivos “influx d” e “influx”. Esto lo hemos realizado en el mismo portátil que hemos instalado el servidor LoRa. Una vez realizados estos pasos, hemos creado la base de datos necesaria para el proyecto empleando los siguientes comandos:

```
>create database loradatos

>show databases

>use loradatos

>show measurements
```

Con el último comando podemos ver los datos o medidas que se están añadiendo a la base de datos.

7.1.6 Grafana

En nuestro caso hemos optado por instalar Grafana en el mismo equipo que el Servidor LoRa y la base de datos. Se ha procedido a la instalación del software descargando la carpeta comprimida de la web oficial, descomprimiendo los ejecutables y ejecutando "grafana-sever".

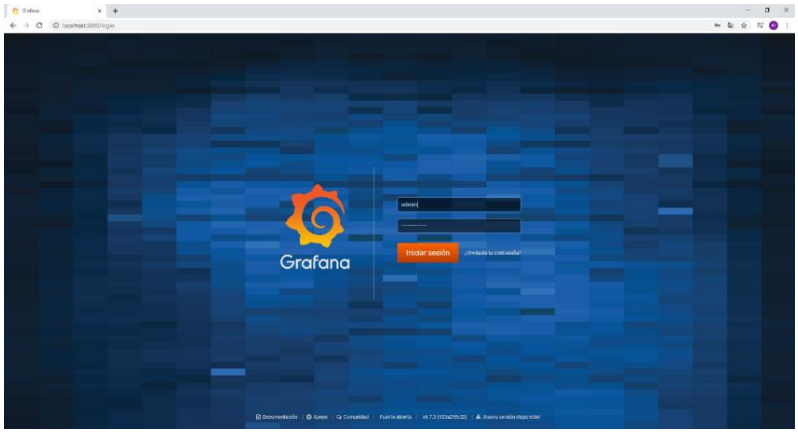

Figura 47. Página de login de Grafana.

Una vez realizados estos pasos, accedemos al servidor poniendo en el navegador la IP del PC donde está instalado y el puerto 3000. Como podemos ver en la Figura 47, Grafana está instalado en el PC local, por lo que hemos puesto en el navegador "localhost:3000", y nos aparece la página de "*login*".

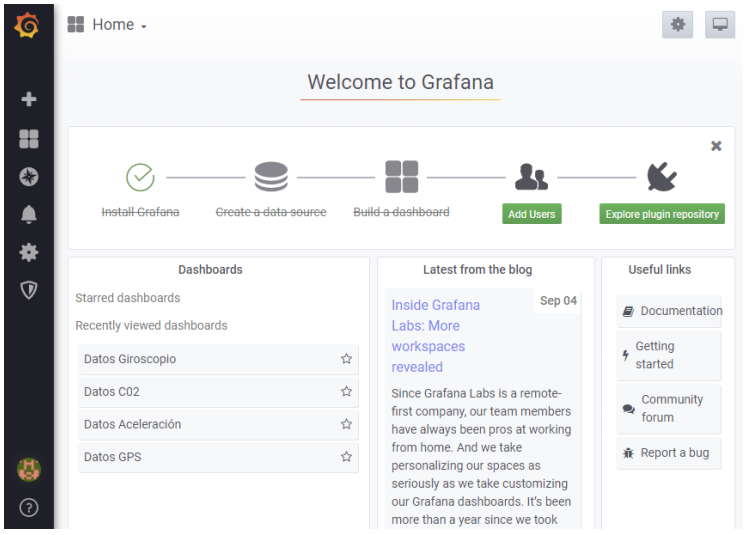

Figura 48. Página principal de Grafana

Tras acceder con nuestro usuario "Admin" y la contraseña que hayamos elegido, procedemos a crear nuestra primera representación de los datos. Para ello seguimos los pasos indicados en la Figura 48.

En primer lugar, agregamos una fuente de datos, según se muestra en la Figura 49, en nuestro caso será InfluxDB. A continuación, elegimos la visualización para la representación de nuestros datos y agregamos una consulta a la base de datos para extraer la información que necesitemos representar (Figura 50). Además, podemos instalar complementos para cambiar las visualizaciones de Grafana. En este caso se ha instalado "*Worldmap Panel Plugin*" para la representación de los datos de GPS.

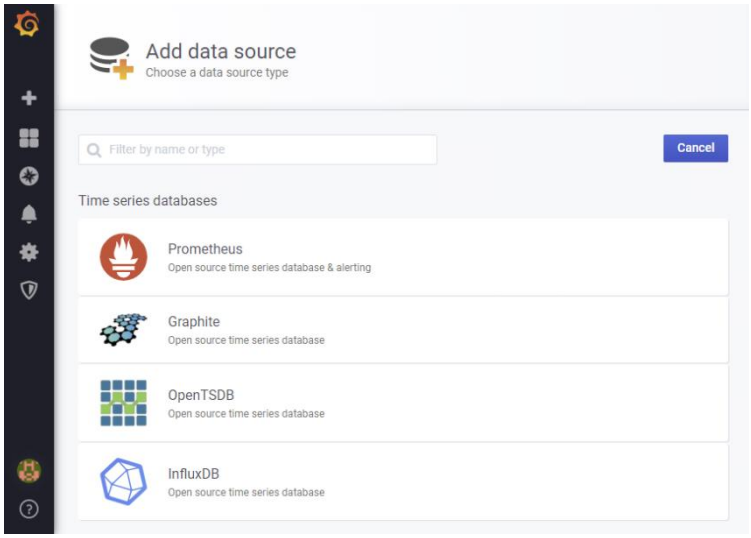

Figura 49. Añadir fuente de datos en Grafana.

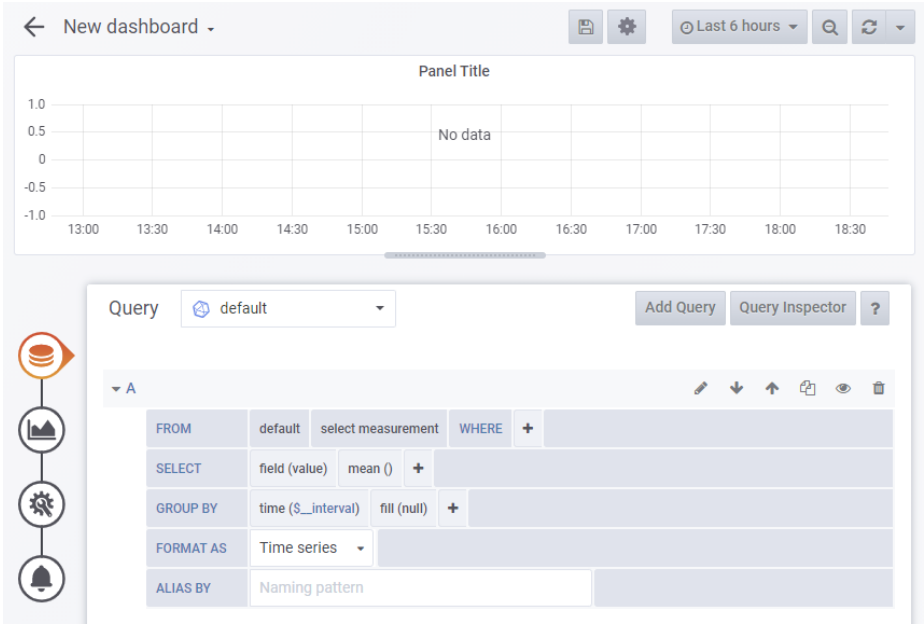

Figura 50. Agregar la consulta a la base de datos en Grafana.

7.2. Metodología

El escenario de pruebas ha sido la localidad Pinilla de San José, como se ha indicado, una pequeña pedanía de tan sólo 200 habitantes aproximadamente, que pertenece al municipio de Caravaca de la Cruz y que se encuentra a 13 km de ésta. Las pruebas se han realizado en la carretera principal, que une Caravaca de la Cruz con Lorca, como se puede ver en la Figura 51. Las pruebas consisten en realizar un recorrido a lo largo de esta carretera para obtener datos de validación y rendimiento de la unidad a bordo. Para ello, se llevaron a cabo varias pruebas completas en las que se recopilaron: datos de aceleración, en fuerza g; datos de giro, en º/s; datos de CO2, en partes por millón (ppm); datos de compuestos orgánicos volátiles totales (TVOC), en partes por millón (ppm); humedad, en %; temperatura, en ºC; y datos de la relación señal a ruido (SNR), en dB.

Figura 51. Señalización escenario de pruebas.

Las pruebas en movilidad realizadas con la bicicleta mostrada en la Figura 51 han sido las siguientes:

- Para obtener datos de la aceleración significativos, se han tomado datos con la bicicleta en reposo, con la bicicleta a una velocidad constante y con la bicicleta experimentando una aceleración. Primero se realizó una prueba con la bicicleta en reposo, para obtener unos datos de la aceleración constantes, sin ningún cambio. A continuación, se llevó a cabo una prueba con la bicicleta a un ritmo constante, y en un determinado momento se empezó a acelerar, para que así se pueda observar en los resultados cómo bajo una aceleración constante, los valores recogidos sufren variación.

- En cuanto a los datos del giroscopio, se han realizado pruebas con distintas situaciones de la bicicleta, para detectar por ejemplo cuándo se produce un giro muy brusco y la bicicleta puede caerse. Se tomaron datos con la bicicleta en reposo, además de registrar giros bruscos hacia los lados simulando una caída, para que en los resultados finales pudieran apreciarse las variaciones.
- Para los datos de contaminación se han tomado datos en distintos escenarios. Uno de ellos fue al aire libre, en la calle, y otro poniendo la unidad abordo tras el tubo de escape de una motocicleta, que desprende dióxido de carbono, entre otros, y es lo que nos interesa en este caso. En los resultados se puede observar una subida brusca del valor de CO_2 en partes por millón. En cuanto a los datos de TVOC, no se ha realizado ninguna representación de éstos, pero sí que se muestran por la pantalla OLED, en la que la persona que circule con la bicicleta en este caso puede ver los valores en cada momento por dónde va circulando.
- Los datos de la humedad y la temperatura, al igual que en el caso anterior, no se representan, simplemente aparecen en la pantalla para que el usuario pueda saber en cada instante la temperatura y la humedad de cada zona de circulación. Estos datos iban apareciendo en todo momento en la pantalla mientras se hacía el recorrido descrito anteriormente. La diferencia de temperatura se observaba, por ejemplo, al entrar en el interior de la vivienda con la unidad a bordo. Se observó que la temperatura variaba con claridad. También lo hacía entre diferentes horas del día.
- Finalmente, empleando la unidad de GPS que contiene la placa de comunicaciones, hemos utilizado la localización de la unidad para representar en ella el valor de la señal a ruido de cada uno de los mensajes. Para obtener el valor de SNR, hemos extraído de la base de datos ese valor en concreto, el cual es agregado por el *gateway* como un metadato. En la sección 5.2. Configuración del *Gateway*, más concretamente en la Figura 27 se puede observar cómo hemos seleccionado la opción que permite que el *gateway* añada metadatos como el RSSI, SNR, etc. Para representar el valor junto a la ubicación, hemos empleado el "*WorldMap Panel Plugin*" de Grafana, que pinta cada ubicación de un color, dependiendo del valor de SNR, y, además, indica este dato.

Todos estos valores han sido obtenidos ejecutando un script de Python en la OBU siguiendo la explicación dada en el ANEXO II. INSTALACIÓN LIBRERÍAS PARA EL

CONTROL DE LOS SENSORES, y luego se han mostrado utilizando la interfaz de Grafana.

7.3. Resultados

De las pruebas realizadas para obtener los datos de la aceleración, en la Figura 52 se muestran los datos obtenidos cuando la bicicleta está en reposo. Se puede observar que los valores de la aceleración son constantes en todo momento y cercanos a cero. En la leyenda se muestran las series representadas. En verde se muestra la aceleración en el eje X, en amarillo la aceleración en el eje Y, y en azul la aceleración en el eje Z. En la Figura 53 se puede observar cómo la aceleración empieza a aumentar ligeramente porque la bicicleta pasa de un estado de reposo a un estado de movimiento, es decir, la bicicleta comienza a moverse en ese momento. En la última gráfica de la aceleración, en la Figura 54, se muestra de nuevo la aceleración cuando la bicicleta pasa de un ritmo medio, a un aumento más brusco de la aceleración.

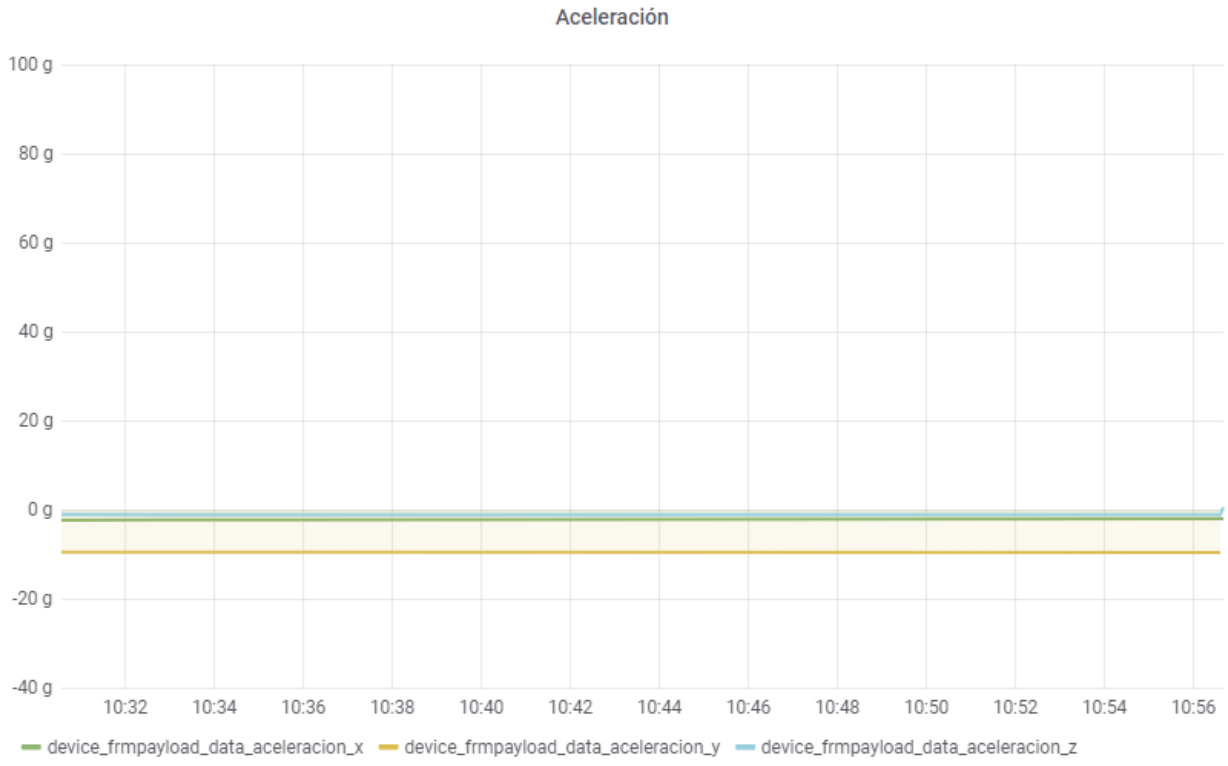

Figura 52. Aceleración con la unidad en reposo.

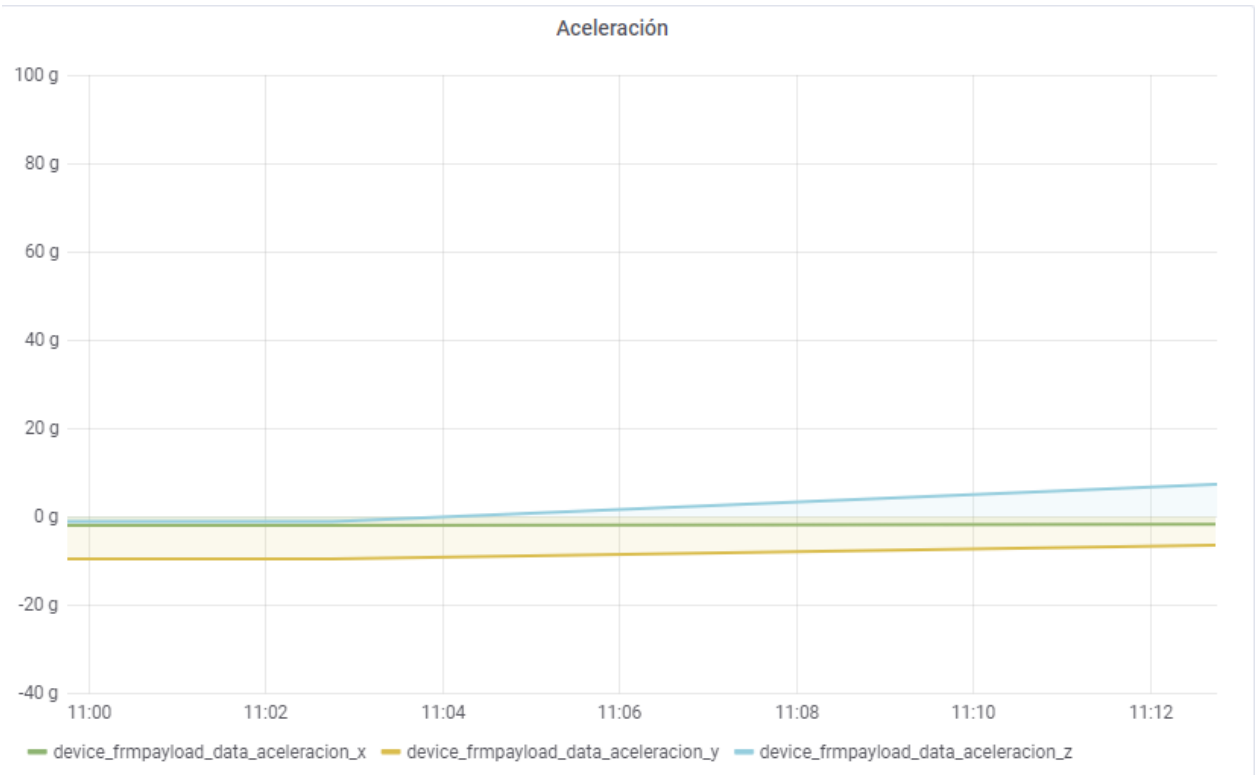

Figura 53. Datos de aceleración cuando la unidad comienza el movimiento.

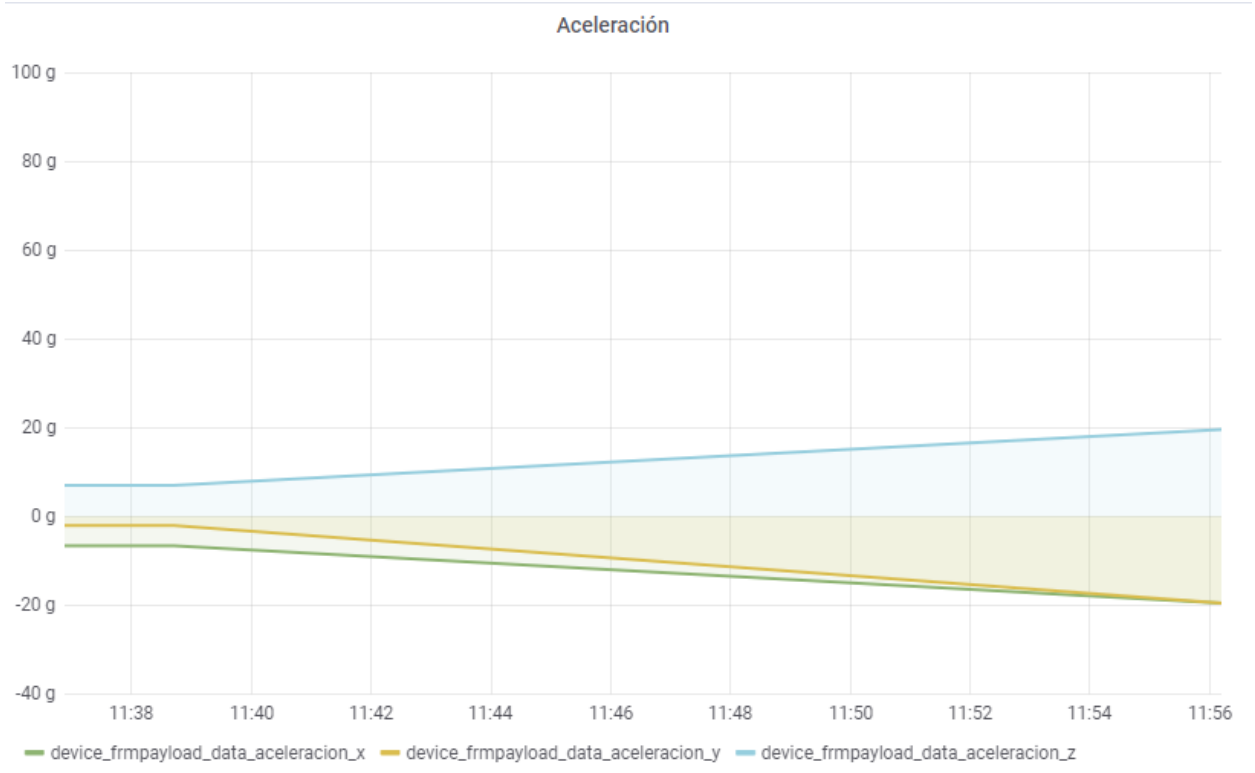

Figura 54. Aceleración al pasar de un ritmo constante a una velocidad más rápida.

De las pruebas de giro realizadas, se han obtenido los datos que se pueden observar en la gráfica de la Figura 55. Se puede apreciar que cuando la bicicleta está en reposo, los valores del giroscopio son constantes y muy cercanos a cero, y cuando se produce un giro brusco, por ejemplo, cuando la bicicleta se gira hacia un lado como si se fuera a caer, en la gráfica se observa que los valores de giro aumentan en cada uno de sus ejes, en verde el eje X, en amarillo el eje Y, y en azul el eje Z.

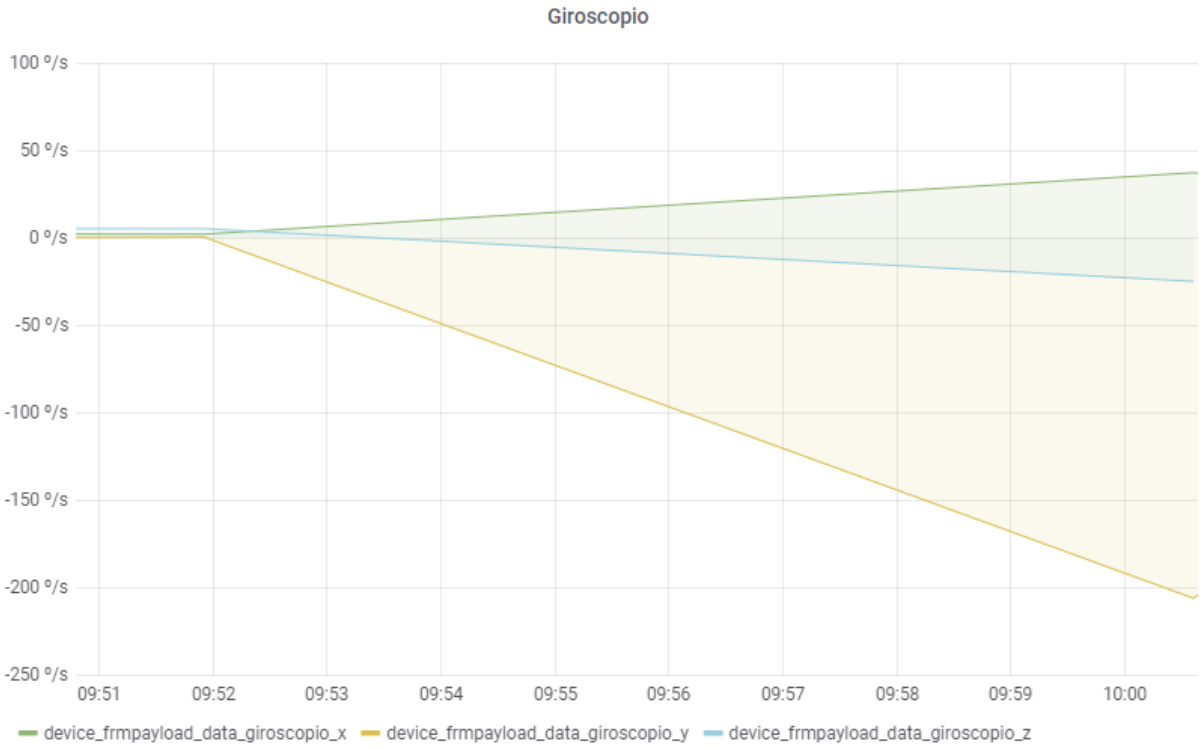

Figura 55. Captura de datos de giro de la unidad.

En cuanto a los datos obtenidos de $C0_2$, en la gráfica de la Figura 56 se puede apreciar cómo aumenta bruscamente el valor del CO2 en ppm cuando se coloca la unidad detrás del tubo de escape de la motocicleta. En la zona de las pruebas no hay mucha contaminación de este tipo ya que es un pueblo muy pequeño, rodeado de naturaleza, y por el que hay un tráfico mínimo. Por ello, el valor inicial de contaminación en la calle no es muy alto, sabiendo que éste oscila entre 360 ppm en áreas de aire limpio y 700 ppm en las ciudades.

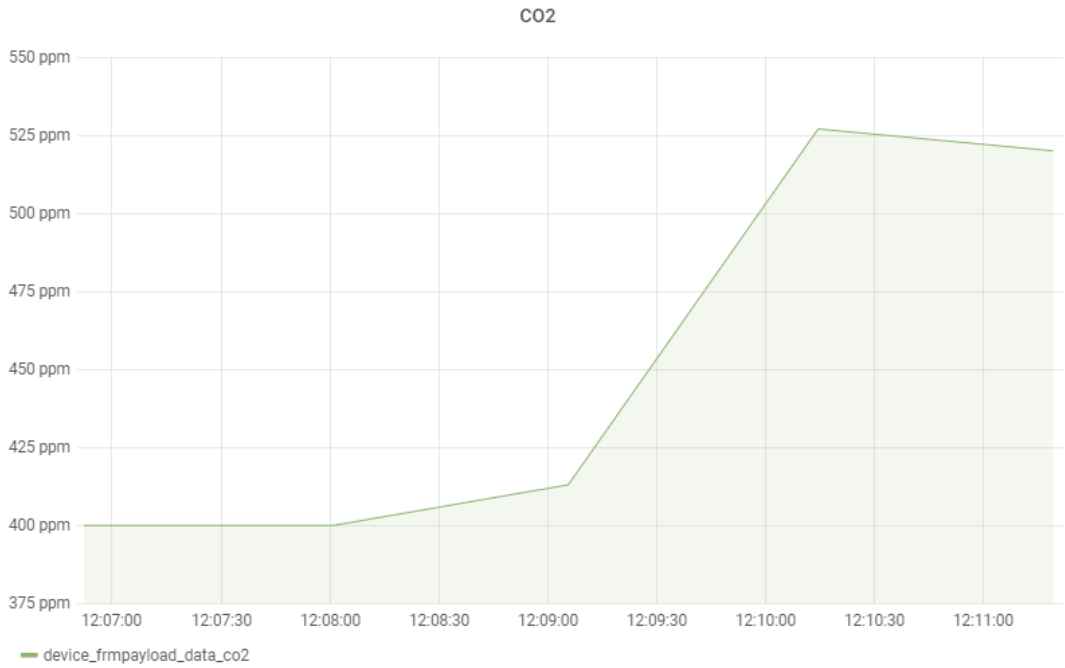

Figura 56. Datos de CO2.

En la Figura 57, se puede observar el mapa de Grafana en el que se representa el valor de SNR para cada posición. Al situar la flecha encima de una ubicación en concreto, aparece el valor del *geohash* de ésta, seguido de dos puntos, y a continuación del valor de SNR, como se puede apreciar en la Figura 58.

Los puntos que aparecen en color verde se corresponden a la localización más cercana al *gateway*, es decir, cuando la unidad móvil se encuentra saliendo de la vivienda.

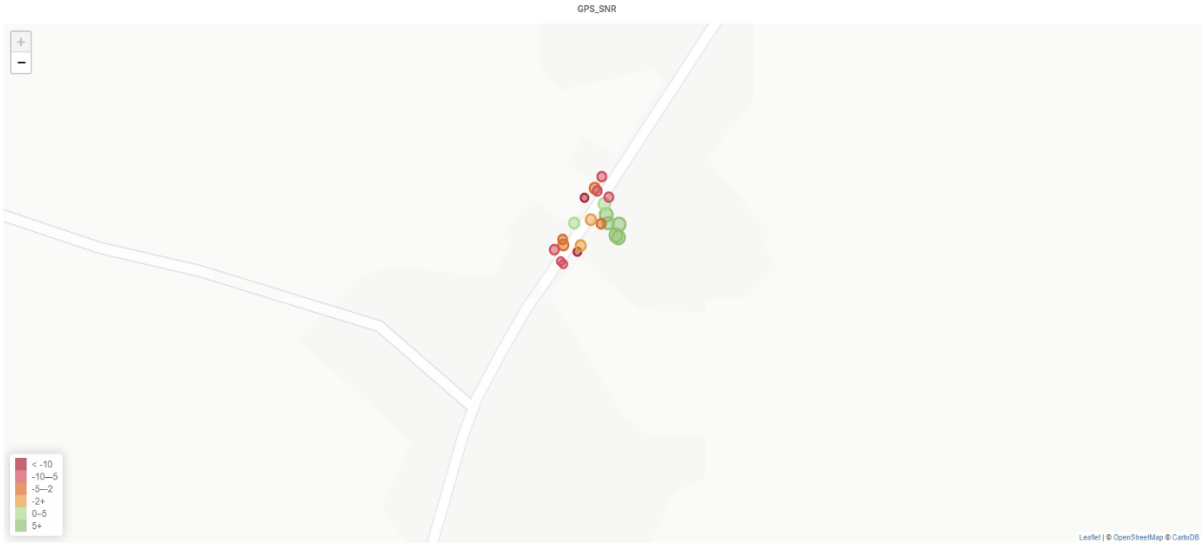

Figura 57. Mapa de recorrido y SNR.

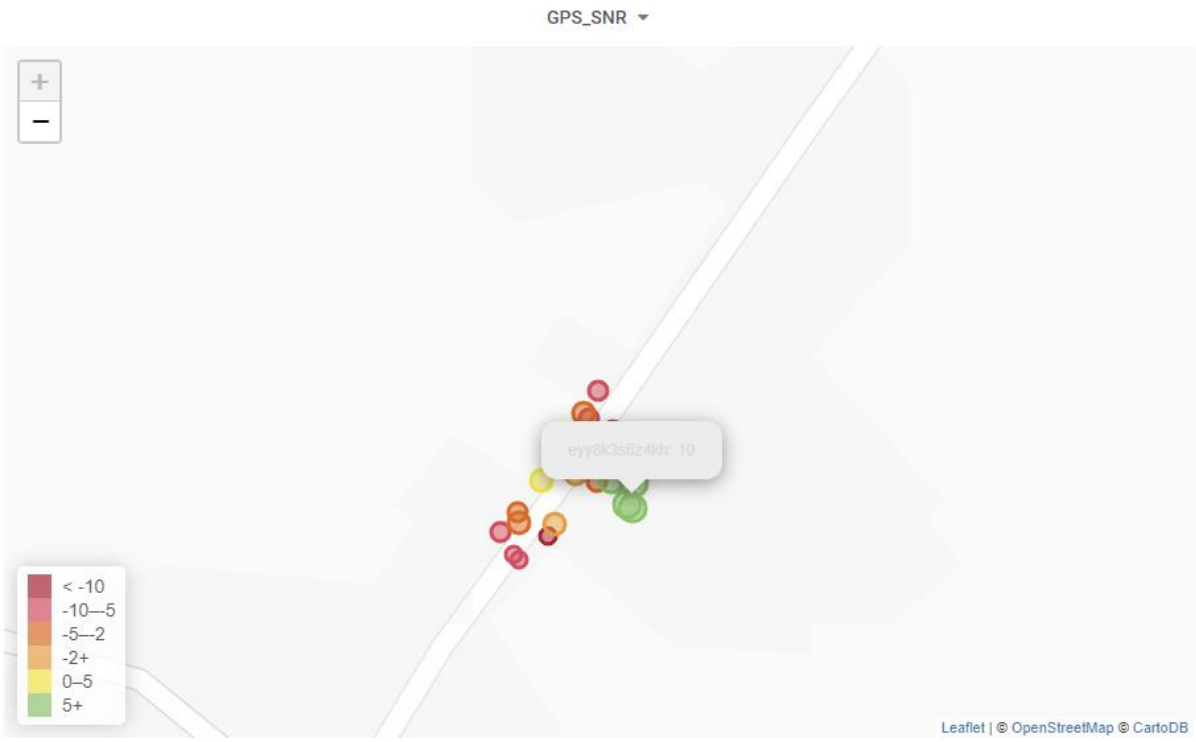

Figura 58. Especificación datos en el mapa.

CAPÍTULO 8. CONCLUSIÓN Y FUTURAS LÍNEAS DE INVESTIGACIÓN

8.1. Conclusión

En este trabajo se ha llevado a cabo la implementación de un *gateway* y servidor LoRa con éxito, realizando una evaluación de la conectividad de nodos finales, cuyos datos han sido reenviados por parte del *gateway* y recolectados por el servidor. Se ha usado como referencia una unidad de a bordo para vehículos de movilidad personal, que representa a un nodo LoRa móvil. Se ha realizado la extracción de datos de distintos sensores, entre ellos: sensores inerciales, de temperatura y de contaminación. Estos han aportado información válida durante las pruebas de validación, demostrando el correcto funcionamiento de la plataforma desarrollada, desde el nodo móvil hasta la representación de los parámetros recogidos de forma gráfica.

En cuanto a los objetivos del proyecto, se han alcanzado todos y cada uno de ellos, ya que se ha realizado un análisis del estado del arte de la tecnología LoRa y LoRaWAN; se ha llevado a cabo la construcción de un *gateway* LoRa mediante el uso del concentrador iC880A; se ha programado un nodo móvil obteniendo datos de sensores de navegación y factores medioambientales; y se han realizado desarrollos para la representación de dichos valores en distintas gráficas en Grafana, lo que ha permitido analizar los datos obtenidos.

Gracias a este trabajo, he conocido esta nueva tecnología, LoRaWAN, la cual presenta un gran potencial en el mundo IoT. Además, he aprendido a planificar y organizar mis tareas, avanzando de forma ordenada en el proyecto.

En cuanto a la temática del proyecto, me ha resultado muy interesante obtener datos de los sensores de un prototipo bajo desarrollo en el grupo GIRTEL, enviarlos utilizando esta nueva tecnología, representarlos, y poder sacar conclusiones sobre el funcionamiento de la unidad y posibles servicios futuros en los ámbitos de la telemetría y la monitorización de factores contaminantes, entre otros.

8.2. Futuras líneas de investigación

Una propuesta de futuro sería incluir un mapa de cobertura, indicando la localización de la unidad móvil en cada momento y la cobertura a la red LoRa que tiene. También, se podrían definir posibles estados adicionales de la unidad móvil y crear alertas en distintas situaciones. Por ejemplo, si el dato obtenido del giroscopio supera un determinado valor, se podría crear una alerta indicando que el vehículo ha sufrido una caída o, por ejemplo, si supera una cierta aceleración, también se podría crear otra alerta, etc.

Por otro lado, también sería interesante crear mapas de contaminación en las ciudades, indicando las calles por las que va circulando la unidad móvil y la contaminación en cada una de ellas. Con esto se podría evitar el tráfico de automóviles por determinadas zonas donde haya más contaminación.

BIBLIOGRAFÍA

Aftab, N., Raza Zaidi, S. A., & McLernon, D. (2019). Scalability analysis of multiple LoRa gateways using stochastic geometry. *Elsevier*, 10.

Bor, M. C., Roedi, U., Voigt, T., & Alonso, J. M. (2016). Do Lora Low-Power Wide-Area Networks Scale? *MSWiM '16: Proceedings of the 19th ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, (págs. 59-67).

Cano, M. D., & Sanchez-Iborra, R. (2016). State of the Art in LP-WAN Solutions for Industrial IoT Services. *Sensors*, 14.

Cava, M. M. (2019). *LoraWAN.* Everest.

ChirpStack. (2020). LoRaWAN Network Server. *Available online: https://www.chirpstack.io/gateway-os/*.

Cisco. (2020). Cisco Wireless Gateway for LoRaWAN. *Available online: https://www.cisco.com/c/en/us/products/routers/wireless-gateway-lorawan/index.html*.

Grafana. (2020). Página principal Grafana. *Available online: https://grafana.com/grafana/*.

Heath, S. (2002). What is an embedded system? En S. Heath, *Embedded Systems Design* (págs. 1-14). EDN.

InfluxData. (2020). InfluxDB. *Available online: https://www.influxdata.com/products/influxdb-overview/*.

Libelium. (2020). LoRaWAN Networking Guide. *Available online: http://www.libelium.com/development/waspmote/documentation/waspmote-lorawan-networking-guide/*.

LoRa Alliance. (2015). LoRaWAN. What is it? *Available online: https://lora-alliance.org/sites/default/files/2018-04/what-is-lorawan.pdf*, 20.

LoRa Alliance. (2018). LoRaWAN Specification. *Available online: https://lora-alliance.org/sites/default/files/2018-07/lorawan1.0.3.pdf*.

LoRa Alliance. (2020). LoRaWAN Regional Parameters. *Available online: https://lora-alliance.org/sites/default/files/2020-06/rp_2-1.0.1.pdf*, 40-41.

LoRaWAN. (2020).

Mekki, K., Bajic, E., Chaxel, F., & Meyer, F. (2018). A comparatie study of LPWAN technologies for large-scale IoT deployment. *ELSEVIER*, 7.

Multitech. (2020). Multitech Lora Gateway. *Available online: https://shop.multitech.com/mtcdt-l4n1-246a-915-us.html*.

P.Radcliffe, K.Chavez, P.Beckett, Spangaro, J., & C.Jakob. (2017). Usability of LoRaWAN Technology in a Central Business District. *IEEE 85th Vehicular Technology Conference (VTC Spring)*.

Petajajarvi, J., Mikhaylov, K., Roivainen, A., Hanninen, T., & Pettissalo, M. (2015). On the Coverage of LPWANs: Range Evaluation and Channel Attenuation Model for LoRa Technology. *14th International Conference on ITS Telecommunications (ITST)*, (págs. 55-59). Copenhagen, Denmark.

Petric, T., Goessens, M., Nuaymi, L., Toutain, L., & Pelov, A. (2016). Measurements, performance and analysis of LoRa FABIAN, a real-world implementation of LPWAN. *IEEE 27th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC)*, 1-7.

Raspberrypi. (2020). Downloads. *Available online: https://www.raspberrypi.org/downloads/*.

Sanchez-Iborra, R., Gomez, J. S., Viñas, J. B., Cano, M. D., & F.Skarmeta, A. (2018). Performance Evaluation of LoRa Considering Scenario Conditions. *Sensors*.

Santa, J., Bernal-Escobedo, L., & Sanchez-Iborra, R. (2020). On-board unir to connect personal mobility vehicles to the IoT. *Elsevier*.

Semtech. (2019). LoRa® and LoRaWAN®: A technical overview. *Available online: https://lora-*

developers.semtech.com/uploads/documents/files/LoRa_and_LoRaWAN-A_Tech_Overview-Downloadable.pdf, 26.

Semtech. (2020). LoRa Developer Portal. *Available online: https://lora-developers.semtech.com/library/tech-papers-and-guides/lora-and-lorawan/*.

Sparkfun. (2020). Qwiic products. *Available online: https://www.sparkfun.com/qwiic#products*.

The Things Network. (2020). Gateways LoRaWAN. *Available online: https://www.thethingsnetwork.org/docs/gateways/lorrier/*.

Vatcharatiansakul, N., P.Tuwant, & C.Pornavalai. (2017). Experimental performance evaluation of LoRaWAN: A case study in Bangkok. *14th International Joint Conference on Computer Science and Software engineering (JCSSE).* Nakhon Si Thammarat, Tailandia.

Voigt, T., Bor, M., Roedig, U., & Alonso, J. (2016). Mitigating Inter-network Interference in LoRa Networks. *ArXiv*.

Wikipedia. (2020). Sistema embebido. *Available online: https://es.wikipedia.org/wiki/Sistema_embebido*.

Yegin, A., Krampt, T., Dufour, P., Gupta, R., Soss, R., Hersent, O., . . . Sornin, N. (2020). LoRaWAN protocol: specifications, security, and capabilities. In B. S. Chaudhari, & M. Zennaro, *LPWAN Technologies for IoT and M2M Applications* (pp. 37-63). Academic Press.

ANEXO I. INSTALACIÓN SOFTWARE GATEWAY LORA Y PILA DEL SERVIDOR DE RED CHIRPSTACK

A1.1. Gateway

En primer lugar, se ha llevado a cabo la instalación del sistema operativo en la Raspberry Pi. Se ha instalado Chirpstack Gateway OS Base, software creado por ChirpStack para crear un nodo *gateway*. Para ello se ha descargado el sistema operativo de la web oficial de ChirpStack, cuya imagen ha sido instalada en la tarjetea microSD que se introduce en la Raspberry Pi. Para llevar a cabo ese proceso se ha empleado el programa Etcher como herramienta para seleccionar la imagen y la tarjeta microSD donde se quiere escribir esta.

Una vez finalizada la instalación del sistema operativo e introducida la tarjeta microSD en la Raspberry, se lleva a cabo la configuración de la Raspberry, ya sea conectando ésta a un monitor HDMI, o bien usando un cliente SSH como PuTTY para acceder a ella.

Para realizar la configuración del sistema se llevan a cabo los siguientes pasos:

1. En primer lugar, nos conectamos al *gateway* mediante un monitor HDMI y un teclado USB o mediante un cliente SSH, como hemos comentado anteriormente
2. A continuación, una vez dentro, ejecutamos el comando "*sudo gateway-config*" en el terminal, el cual nos llevará a la pantalla que aparece en la Figura 59.

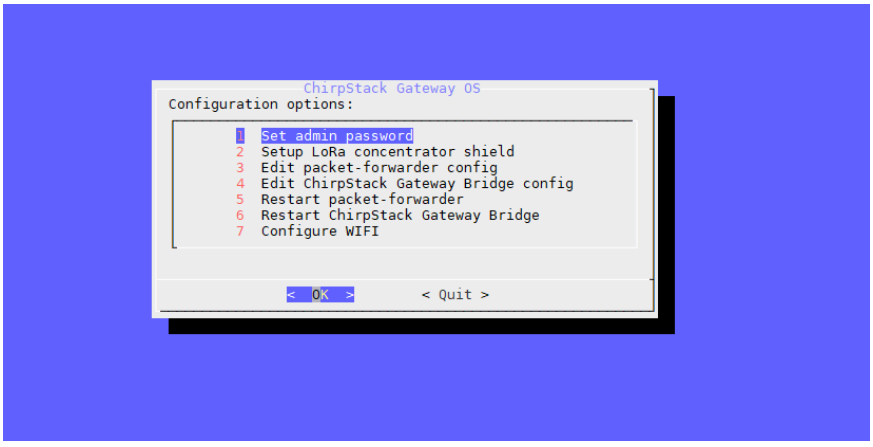

Figura 59. Opciones de configuración en ChirpStack Gateway OS.

3. En esa pantalla seguimos con la configuración, y establecemos nuestro concentrador (iC880A) que podemos ver en la Figura 60.

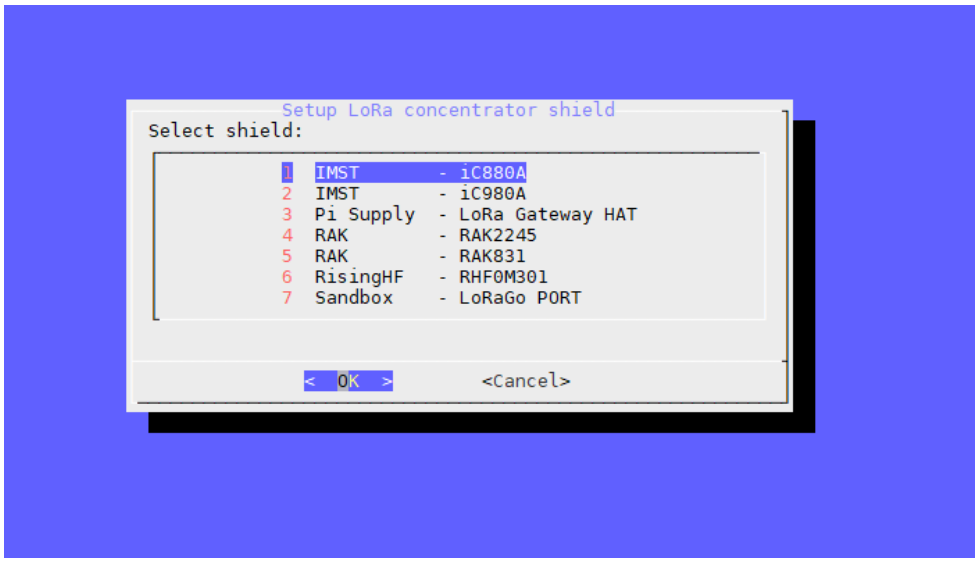

Figura 60. Configuración del concentrador iC880A.

4. Una vez elegido el concentrador, seleccionamos el canal que éste va a usar (868 MHz EU).
5. Por último, seleccionamos la opción de "Editar la configuración del Packet Forwarder", la cual nos lleva a un fichero de configuración, en el que, si todo ha ido correctamente, se habrá cambiado la ID de nuestro *gateway*. Cabe destacar que a la ID es necesario añadir "FFFE" en la mitad para completar los 8 pares. Por ello nuestra gateway_ID aparece como b8273bFFFEf0b26e, cuando en realidad es b827ebf0b26e.

```
"gateway_conf": {
    "gateway_ID": "b827ebFFFEf0b26e",
    "server_address": "192.168.1.164",
    "serv_port_up": 1700,
    "serv_port_down": 1700,
    "keepalive_interval": 10,
    "stat_interval": 30,
    "push_timeout_ms": 100,
    "forward_crc_valid": true,
    "forward_crc_error": false,
    "forward_crc_disabled": false,
    "dr":4
}
```

Figura 61. Configuración gateway en el reenviador de paquetes (Packet Forwarder).

6. Para comprobar que todo ha ido configurado correctamente, podemos ejecutar en consola el comando "*sudo monit status*" para ver el estado de cada uno de los componentes del *gateway*.

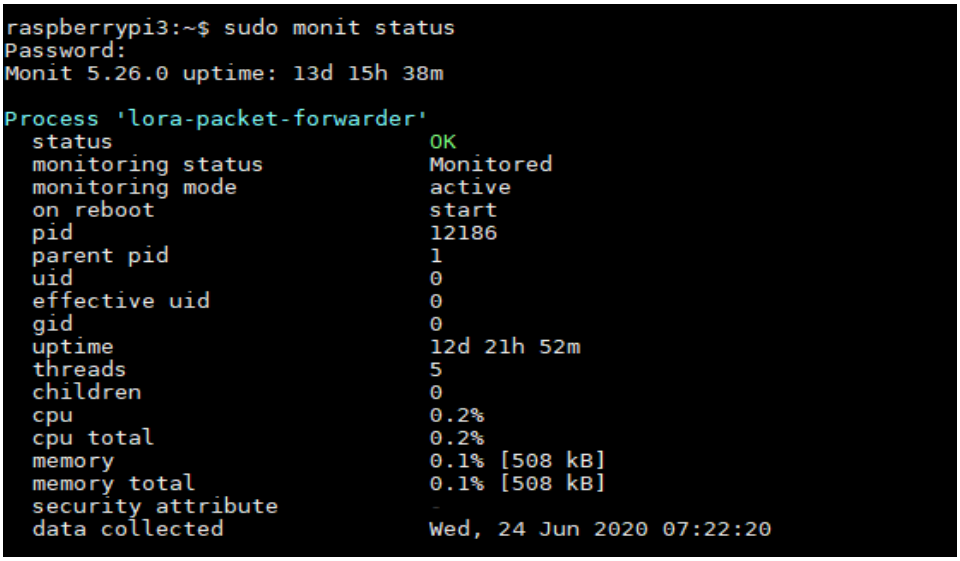

Figura 62. Salida del comando "sudo monit status".

A1.2. Pila ChirpStack

A1.2.1. ChirpStack Gateway Bridge

Para hacer que ChirpStack Gateway Bridge tenga un mejor rendimiento y sea altamente disponible, aunque ya se encuentre preinstalado en la Raspberry con el sistema operativo ChirpStack Gateway OS Base, se puede ejecutar en varios servidores, los cuales se conectan al mismo intermediario MQTT. Por ello, se ha procedido a instalar este componente en el servidor, además de en el propio *gateway.*

Primero, es necesario instalar las dependencias MQTT, Redis y PostgreSQL.

El comando para su instalación es el siguiente:

A continuación, procedemos a configurar la base de datos y usuarios de PostgreSQL. Para ello empleamos los siguientes comandos:

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

```
sudo -u postgres psql
```

```
-- 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
```

El siguiente paso es configurar la clave para el nuevo repositorio de ChirpStack Gateway Bridge.

```
sudo apt-key adv -keyserver keyserver.ubuntu.com -recv-keys
1CE2AFD36DBCCA00
```

Agregamos el repositorio a la lista de repositorios mediante la creación de 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
```

Actualizamos la caché del paquete apt:

```
sudo apt update
```

Por último, instalamos el puente del *gateway* de ChirpStack:

```
sudo apt install chirpstack-gateway-bridge
```

Iniciamos el servicio ChirpStack Gateway Bridge:

```
# start chirpstack-gateway-bridge

sudo systemctl start chirpstack-gateway-bridge

# start chirpstack-gateway-bridge on boot

sudo systemctl enable chirpstack-gateway-brige
```

```
sudo apt install chirpstack-network-server
```

Iniciamos el servicio del servidor ChirpStack:

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

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

A1.2.2. Servidor de aplicaciones ChirpStack

Instalación del paquete:

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

Instalación servidor de red ChirpStack

```
sudo apt install chirpstack-application-server
```

Resultado del registro del servidor de aplicaciones ChirpStack:

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

ANEXO II. INSTALACIÓN LIBRERÍAS PARA EL CONTROL DE LOS SENSORES

En primer lugar, habilitamos el bus I2C de la Raspberry Pi Zero para poder comunicarnos con los sensores. Para ello seleccionamos la opción 3 de "*Interfacing Options*" como se muestra en la Figura 63 y a continuación marcamos I2C.

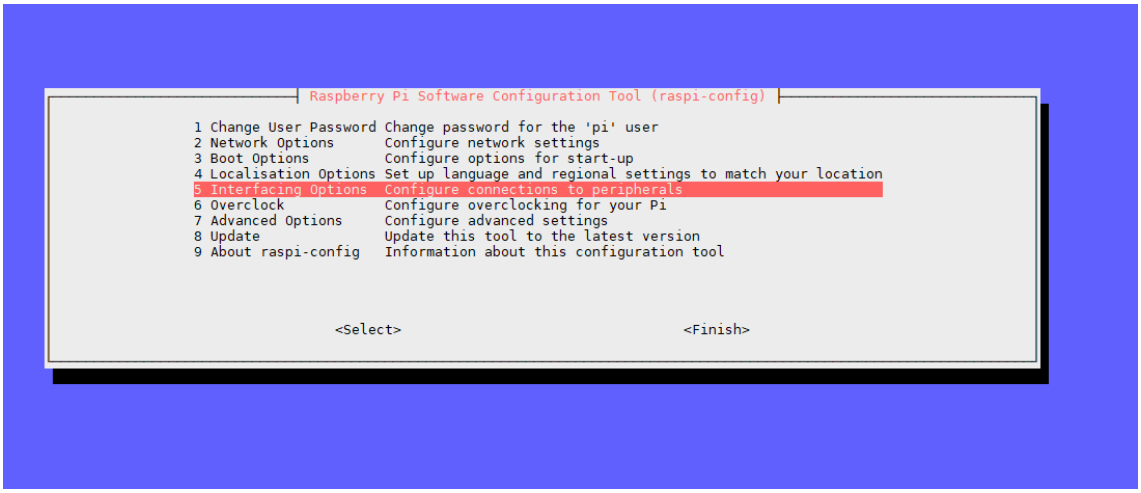

Figura 63. Configurar conexión a periféricos.

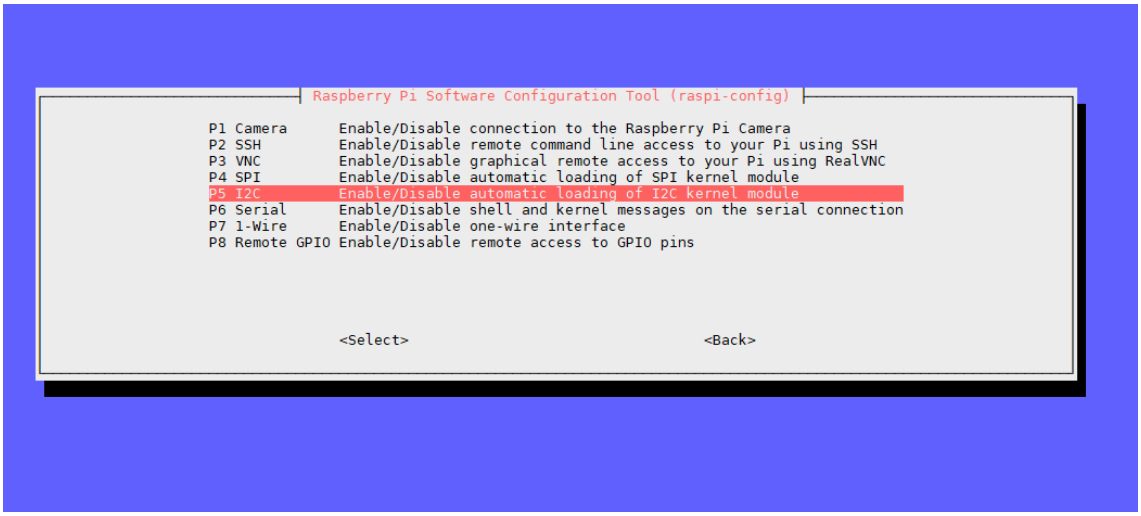

Figura 64. Seleccionamos la opción I2C.

En segundo lugar, de la misma forma, seleccionamos la segunda opción de "*Network Options*" para configurar la conexión a Internet. Una vez seleccionada esa opción, en un fichero de configuración sólo tenemos que añadir el nombre y la contraseña de la red a la que queremos conectarnos. Este paso es necesario para poder descargar de Internet las librerías necesarias para el siguiente paso, que son las siguientes:

```
sudo pip install pyserial
```

Instalamos esta librería ya que nos permite las comunicaciones a través de serial (RS-232), lo cual puede ser muy útil para mandar o recibir datos de periféricos sin tener que complicar la programación.

GPS

```
sudo pip install pynmea2
```

Pynmea2 es una biblioteca de Python para el protocolo NMEA 0183. Este protocolo es un medio a través del cual los instrumentos marítimos y también la mayoría de receptores GPS pueden comunicarse unos con los otros.

```
sudo pip install python-geohash
```

Como ya se ha comentado anteriormente, contamos con un GPS del que obtenemos los datos de la localización en cada momento. Estos datos se representan también en Grafana en un mapa, para lo cual enviamos los datos en formato Geohash.

Geohash es una forma de expresar los datos de geolocalización, en la que se codifican la latitud y la longitud como una cadena de números y letras. Empleamos esta codificación ya que el plugin de Grafana "*WorldMap Panel*" emplea este formato para la localización.

Aceleración

```
sudo pip3 install adafruit-circuitpython-lsm9ds1
```

Esta librería es instalada para facilitar el uso del sensor lsm9ds1 con circuitpython. Permite escribir código Python que lee el acelerómetro, magnetómetro y giroscopio del sensor.

Sensor medio ambiental

```
sudo pip3 install adafruit-circuitpython-ccs811
```

```
sudo pip3 install adafruit-circuitpython-bme280
```

Al igual que en el caso anterior, estas librerías sirven para acceder fácilmente a los datos que proporcionan estos dos sensores, tales como el C02 en ppm, el TVOC en ppm, la humedad, presión, altitud, etc.

Pantalla micro OLED

```
sudo pip install sparkfun-qwiic-micro-oled
```

Instalamos esta biblioteca para poder acceder al micro oled, y poder mostrar la información por pantalla.

ANEXO III. PARTES DE CÓDIGO PYTHON MÁS IMPORTANTES

Configuración APPEUI para unión a la red

```
def lora_APPEUI_union(lora_modem: serial.Serial):
    #ID de la aplicación
    print("Modem union init")
    try:
      lora_modem.write("AT+APPEUI=07:07:07:07:07:07:07:07\r".encode("ASCII"))
        response = lora_modem.readline()
        response = lora_modem.readline()

        if 'OK' in str(response):
            print("LoRa UNION working")
        else:
            print("LoRa UNION not working")
            exit(0)

    except Exception as e:
        print(e)
```

Comprobación del estado de unión

```
def estado_union(lora_modem: serial.Serial):

    #Comprobación estado de unión

    try:

        lora_modem.write("AT+NJS=?\r".encode("ASCII"))

        result = ""

        while True:

            result = (

             result+(bytes.decode(lora_modem.readline())).replace("\r\n",''))

            if 'OK' in str(result):

                if '1' in str(result):
```

```
                    print("Union establecida")
                if '0' in str(result):
                    print("Union no establecida")
                break
  except ValueError:
        print("Oops! se ha producido un error")
```

Extracción de datos de los sensores

```
    while True:
        accel_x, accel_y, accel_z = sensor.acceleration
        mag_x, mag_y, mag_z = sensor.magnetic
        gyro_x, gyro_y, gyro_z = sensor.gyro
        temp = sensor.temperature
        humid = environment.humidity
        pressure = environment.pressure
        altitude = environment.altitude

        co2 = gases.eco2
        tvoc = gases.tvoc
        print(
           "Acceleration(m/s^2):({0:0.3f},{1:0.3f},{2:0.3f})".format(accel_x,
accel_y,  accel_z))
        print(
            "Gyroscope(degrees/sec):
({0:0.3f},{1:0.3f},{2:0.3f})".format(gyro_x, gyro_y,  gyro_z))

        print("Temperature: {0:0.3f}".format(temp))
        print("Humid:{0:0.3f}".format(humid))
```

```
        print("C02: %1.0f PPM" % co2)
        print("TVOC: %1.0f PPM" % tvoc)
        print("Pressure:{0:0.3f}".format(pressure))
        print("Altitude:{0:0.3f}".format(altitude))
        time.sleep(1.0)
```

Envío de datos de GPS y de los sensores

```
  if gps_gga.is_valid:
        thisgeohash = geohash.encode(
        float("" + str(gps_gga.latitude)), float("" + str(gps_gga.longitude)))

        datapoint = thisgeohash+","+datoacce_x+"," + datoacce_y + "," + \
          datoacce_z+","+datogyro_x+","+datogyro_y+","+datogyro_z+","+datoco2

        atcmd = "AT+SEND=12:"+datapoint+"\r"
        lora_modem.write(atcmd.encode("ASCII"))
        print("Datapoint: " + datapoint)
        time.sleep(1)

        response = (bytes.decode(lora_modem.readline())).replace("\\r\\n", '')
        response = (bytes.decode(lora_modem.readline())).replace("\\r\\n", '')

        print("  got: " + response)

    if not gps_gga.is_valid:
```

```
        datapoint = datoacce_x+","+datoacce_y+","+datoacce_z + \
            ","+datogyro_x+","+datogyro_y+","+datogyro_z+","+datoco2


        atcmd = "AT+SEND=12:"+datapoint+"\r"
        lora_modem.write(atcmd.encode("ASCII"))
        print("Datapoint: " + datapoint)
        time.sleep(1)
        response = (bytes.decode(lora_modem.readline())).replace("\\r\\n", '')
        response = (bytes.decode(lora_modem.readline())).replace("\\r\\n", '')


        print("  got: " + response)


        print("ERROR GGA")


print("Modem sending telemetry packet")
```