ULPGC
Universidad de
Las Palmas de
Gran Canaria

eITe
ESCUELA DE INGENIERÍA
DE TELECOMUNICACIÓN Y ELECTRÓNICA

ESCUELA DE INGENIERÍA DE

TELECOMUNICACIÓN Y ELECTRÓNICA

TRABAJO FIN DE MÁSTER

APLICACIÓN IOT LORAWAN PARA EL ANÁLISIS DE PATRONES DE COMPORTAMIENTO BASADA EN MÓDULOS HW DE DETECCIÓN DE PERSONAS

Titulación: Máster Universitario en Ingeniería de Telecomunicación

Autor: D. Julio Francisco Estela Bravo

Tutores: Dr. D. Valentín De Armas Sosa

Dr. D. Félix Bernardo Tobajas Guerrero

Fecha: Julio de 2024

Agradecimientos

En primer lugar, me gustaría agradecer a mis tutores el Dr. D. Valentín de Armas Sosa y el Dr. D. Félix B. Tobajas Guerrero por ser grandes profesores y enormes educadores, al profesorado de la Escuela de Ingeniería de Telecomunicación y Electrónica (EITE) de la Universidad de Las Palmas de Gran Canaria (ULPGC) por el trabajo realizado en mi formación como profesional, y a mi familia por apoyarme en todo momento.

Resumen

En este Trabajo Fin de Máster (TFM) se aborda la implementación de una aplicación IoT (*Internet of Things*) enfocada en la detección de rostros utilizando nodos finales basados en el dispositivo Arduino MKRWAN 1310 y el módulo *Person Sensor* de *Useful Sensor*. Se destaca la importancia de la conectividad y la eficiencia energética en dispositivos autónomos, operados por baterías. Se emplea el protocolo LoRaWAN por su bajo consumo y largo alcance, gestionando la comunicación y eficiencia energética de forma óptima. La plataforma *The Things Network* (TTN) y el sistema de almacenamiento InfluxDB se utilizan para procesar y almacenar los datos, proporcionando flexibilidad para el almacenamiento local y en la nube. Además, se utiliza Grafana para la visualización gráfica de los datos, ofreciendo una interfaz intuitiva y poderosa para el análisis de la información capturada. Este TFM demuestra la aplicabilidad de estrategias de conectividad, gestión de datos y visualización en IoT, contribuyendo al avance académico y profesional en este campo.

Abstract

This Master's Thesis focuses on the implementation of an IoT application for face detection using Arduino MKRWAN 1310 devices and the Person Sensor module by Useful Sensor. The study emphasizes the importance of connectivity and energy efficiency in battery-operated autonomous devices. The LoRaWAN protocol is employed for its low power consumption and long-range capabilities, managing communication and energy efficiency optimally. The Things Network (TTN) platform and InfluxDB storage system are used to process and store data, providing flexibility for both local and cloud storage. Additionally, Grafana is utilized for data visualization, offering an intuitive and powerful interface for analyzing captured information. This Master's Thesis demonstrates the applicability of connectivity, data management, and visualization strategies in IoT, contributing to academic and professional advancements in this field.

Índice

Índice de Figuras

Índice de Tablas

Listado de acrónimos

ABP	Activation By Personalization
ADR	Adaptive Data Rate
AES	Advanced Encryption Standard
AP	Access Point
API	Application Programming Interface
APM	Application Performance Monitoring
ASML	Advanced Semiconductor Materials Lithography
BLE	Bluetooth Low Energy
CPU	Central Processing Unit
CR	Coding Rate
CSS	Channel Spreading Sequence
DevEUI	Device Extended Unique Identifier
DHCP	Dynamic Host Configuration Protocol
DUP	Duplicate
EITE	Escuela de Ingeniería de Telecomunicación y Electrónica
ESSID	Extended Service Set Identifier
ETSI	European Telecommunications Standards Institute
EUI	Extended Unique Identifier
FUOTA	Firmware Update Over-The-Air
GPU	Graphics Processing Unit
GSM	Global System for Mobile Communications

HTTP	Hypertext Transfer Protocol
HW	Hardware
I_2C	Inter-Integrated Circuit
IDE	Integrated Development Environment
IEEE	The Institute of Electrical and Electronics Engineers
IIoT	Industrial Internet of Things
IoT	Internet of Things
ISM	Industrial, Scientific, and Medical
ITU	International Telecommunication Union
JoinEUI	Join End-Device Identifier
LoRa	Long Range
LoRaWAN	Long Range Wide Area Network
LED	Light Emitting Diode
LEO	Low Earth Orbit
LNS	LoRaWAN Network Server
LTE	Long-Term Evolution
LWT	Last Will and Testament
LPWAN	Low Power Wide Area Network
M2M	Machine to Machine
MAC	Media Access Control
MBED	Microcontroller Board Embedded Device
microSD	Micro Secure Digital
MQTT	Message Queuing Telemetry Transport

MCU	Microcontroller Unit
MUIT	Máster Universitario en Ingeniería de Telecomunicación
NAT	Network Address Translation
NB-IoT	Narrowband Internet of Things
NVM	Non-Volatile Memory
OTAA	Over-The-Air-Activation
OUI	Organizational Unique Identifier
PC	Personal Computer
PLC	Programmable Logic Controller
PoE	Power over Ethernet
QoS	Quality of Service
RFID	Radio-Frequency Identification
RTC	Real-Time Clock
SF	Spread Factor
SLA	Service Level Agreement
SAMD	Smart Arm-based Microcontroller Devices
SSID	Service Set Identifier
SPI	Serial Peripheral Interface
TFM	Trabajo Fin de Máster
TSMC	Taiwan Semiconductor Manufacturing Company
TTN	The Things Network
UIT	Unión Internacional de Telecomunicaciones
ULPGC	Universidad de Las Palmas de Gran Canaria

URL	Uniform Resource Locator
VoLTE	Voice over Long-Term Evolution
WPA3	Wi-Fi Protected Access 3
XR	Extended Reality

1 Introducción

En la actualidad, la conectividad de los dispositivos se ha convertido en una característica casi indispensable, tanto para fabricantes como para comerciantes, debido a su relevancia en la monitorización, almacenamiento y análisis en tiempo real de diversos parámetros. Esta tendencia evidencia un cambio paradigmático en la concepción y comercialización de productos tecnológicos, donde la interconexión se percibe como un elemento fundamental para la competitividad y la eficiencia en el mercado.

La comercialización de dispositivos conectados implica la necesidad de garantizar un rendimiento máximo en términos de comunicación y eficiencia energética. En su mayoría, estos dispositivos operan de manera autónoma, alimentados por baterías, por lo que una gestión eficaz del consumo energético se erige como un factor crucial para optimizar su funcionalidad a largo plazo. Con esta finalidad, se implementan diversas técnicas, como la programación a nivel de hardware y la optimización de la conexión a la red, estrategias que buscan equilibrar la eficacia operativa con el ahorro de energía.

El concepto de *Internet of Things* (IoT) aúna la interconexión y comunicación entre dispositivos electrónicos, incluyendo sensores y otros componentes, a través de Internet [1]. La eficiencia del consumo energético en el contexto de IoT se ve respaldada por protocolos especializados, entre los que destaca *Long Range Wide Area Network* (LoRaWAN), que representa una tecnología de bajo consumo energético y largo alcance para redes inalámbricas, lo que la convierte en una opción atractiva para aplicaciones IoT con requisitos de energía limitados [2]. LoRaWAN opera bajo una banda que no necesita licencia y permite desplegar miles de nodos sin interferir con otras redes, resultando conveniente para conectar dispositivos que demanden una baja tasa de transmisión de datos a larga distancia, con un reducido consumo de potencia.

El protocolo LoRaWAN asume además la responsabilidad de administrar la comunicación, la tasa de datos, y las frecuencias de energía para cada dispositivo de forma individual. En una red LoRaWAN, los nodos finales envían los datos de manera asíncrona a cualquier puerta de enlace, o *Gateway*, que se encuentre dentro de su área de cobertura, para ser transferidos a un *Network Server* (*NS*) a través de Internet, de acuerdo con una topología

de red basada en un esquema similar al de una disposición en estrella, como se representa en la Figura 1 [2].

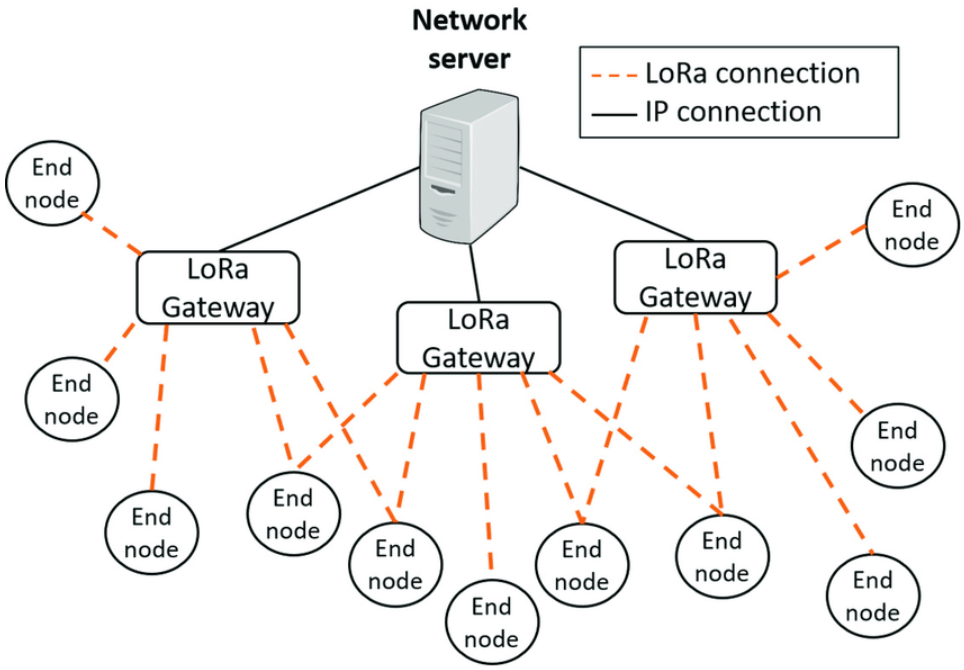


Figura 1 Topología de una red LoRaWAN.

Además, existen entornos de desarrollo y plataformas que ofrecen herramientas y recursos para la implementación de aplicaciones IoT basadas en LoRaWAN. Estos entornos, orientados por lo general a placas de prototipado basadas en MCU (*Micro Controller Unit*) y plataformas IoT en la nube, proporcionan opciones tanto gratuitas como de suscripción, que permiten a los desarrolladores adaptar los proyectos a sus necesidades específicas de conectividad y almacenamiento de datos.

En este sentido, Arduino representa un entorno de desarrollo de código abierto que integra tanto hardware como software [3]. Las plataformas de desarrollo Arduino permiten adquirir datos provenientes de diversas fuentes, como sensores, o incluso mensajes de redes sociales, y transformar posteriormente estos datos en respuestas concretas, tales como el accionamiento de motores, la iluminación de diodos emisores de luz (LED) o la publicación de información en línea. Este proceso se logra mediante la transmisión de un conjunto de instrucciones al microcontrolador integrado en la plataforma, valiéndose del lenguaje de programación Arduino, basado en *Wiring*, y

haciendo uso del entorno de desarrollo integrado o IDE *(Integrated Development Environment)*, que tiene su origen en *Processing* [3].

Por otro lado, la plataforma IoT en la nube *The Things Network* (TTN) es un recurso que proporciona, entre otros elementos, un *Network Server* (NS) comunitario orientado a redes LoRaWAN. Las redes comunitarias, como es el caso de TTN, se erigen siguiendo un modelo que parte desde sus usuarios y se desarrolla hacia la creación de las infraestructuras necesarias para brindar los servicios requeridos. Este enfoque implica un proceso en el que la comunidad de usuarios desempeña un papel fundamental en la creación y operación de la red, en lugar de depender exclusivamente de entidades externas para su implementación y mantenimiento [4].

En cuanto al almacenamiento de datos provenientes de los nodos finales de una red LoRaWAN, se plantea la necesidad de seleccionar plataformas que garanticen la seguridad y privacidad de la información capturada por los dispositivos IoT. Tanto el almacenamiento en la nube como en servidores locales ofrecen ventajas y desafíos, por lo que es importante evaluar cuidadosamente las opciones disponibles en función de los requisitos del proyecto y las regulaciones de privacidad aplicables [5].

La plataforma InfluxDB Cloud proporciona un conjunto completo de herramientas y servicios para recopilar y acumular métricas y datos de eventos, analizar los datos y actuar sobre ellos mediante visualizaciones y notificaciones. Ya sea que los datos provengan de sensores o dispositivos, InfluxDB permite a los desarrolladores implementar aplicaciones de monitorización, análisis, e IoT de próxima generación de manera rápida, sencilla y escalable, ofreciendo valor comercial real [5].

El presente Trabajo Fin de Máster (TFM) contempla la implementación de una aplicación IoT en diversos escenarios, incorporando condiciones esenciales como bajo consumo, el protocolo de comunicación LoRaWAN, y las plataformas TTN, InfluxDB y Grafana. El dispositivo principal utilizado en los nodos finales será el modelo MKRWAN 1310 de Arduino, complementado con el módulo *Person Sensor* de la empresa *Useful Sensor*. La aplicación se orienta hacia la detección de rostros de personas mediante estos dispositivos, con el posterior procesamiento y almacenamiento de los datos, ofreciendo

flexibilidad tanto para el almacenamiento local como para el uso de plataformas en la nube.

Así, el presente TFM aborda de manera integral los desafíos y oportunidades asociados con la conectividad de dispositivos, la eficiencia energética y la gestión de datos en el contexto de IoT. La implementación práctica de diferentes propuestas demuestra la aplicabilidad y relevancia de las estrategias planteadas en situaciones reales, contribuyendo así al avance y la comprensión más profunda de *Internet of Things* en el ámbito académico y profesional.

1.1 Objetivos

El objetivo general del presente Trabajo Fin de Máster (TFM) consiste en el desarrollo de una aplicación IoT basada en el protocolo LoRaWAN, que permita realizar el análisis de patrones de comportamiento relacionados con la detección de personas a partir de una serie de nodos implementados como resultado de la integración de un módulo *Person Sensor* con un dispositivo Arduino MKRWAN 1310. Se evaluarán diferentes soluciones basadas, tanto en servicios disponibles en la nube, como en recursos locales, a partir del uso del protocolo MQTT para la comunicación entre los diferentes elementos que formen parte de la aplicación IoT LoRaWAN, con el fin de adaptarse a la disponibilidad de diferentes requisitos y recursos. La aplicación IoT se integrará con plataformas que permitan el almacenamiento y el análisis de las series temporales de datos registrados por el sensor, así como su representación gráfica. Este objetivo general se puede desglosar en los siguientes objetivos específicos:

O1: Estudio de la integración de módulo HW *Person Sensor* con el dispositivo Arduino MKRWAN 1310, ventajas e inconvenientes.

O2: Análisis de los recursos necesarios para la implementación de diferentes soluciones para el desarrollo de la aplicación IoT basada en LoRaWAN.

O3: Implementación de la estructura de las diferentes soluciones para el desarrollo de la aplicación IoT basada en LoRaWAN.

O4: Validación de la funcionalidad de las soluciones implementadas para el desarrollo de la aplicación IoT basada en LoRaWAN.

O5: Análisis de los datos registrados en la aplicación IoT a partir de su procesamiento para la obtención de información relacionada con patrones de comportamiento.

1.2 Peticionario

El peticionario del presente Trabajo Fin de Máster es la Escuela de Ingeniería de Telecomunicación y Electrónica (EITE) de la Universidad de Las Palmas de Gran Canaria (ULPGC), en calidad de institución pública que solicita la realización de dicho trabajo con el fin de superar los requisitos establecidos en la asignatura Trabajo Fin de Máster en el plan de estudios de la titulación Máster Universitario en Ingeniería de Telecomunicación (MUIT).

1.3 Estructura del documento

El documento correspondiente a la memoria del presente Trabajo Fin de Máster (TFM) se estructura en varias secciones claramente definidas:

En primer lugar, se presenta un apartado preliminar que incluye la portada, la hoja de firmas, un resumen del documento, un *abstract* en inglés, y los distintos índices de capítulos, figuras y tablas.

En segundo lugar, la memoria del proyecto abarca toda la información relevante, incluyendo los objetivos, una explicación detallada de todas las etapas del proceso, los resultados obtenidos y su correspondiente visualización. Esta sección también contiene las conclusiones derivadas del estudio y las posibles líneas de investigación futura, además de la bibliografía consultada para poder realizar el TFM.

En tercer lugar, se incluye una sección dedicada al presupuesto, donde se detallan todos los recursos necesarios para la realización del proyecto, y el pliego de condiciones en el que se definen los requisitos técnicos y operativos.

Finalmente, se presenta un anexo en el que se proporciona información adicional que complementa y amplía los contenidos expuestos en la memoria.

PARTE I. MEMORIA

Capítulo 1. Introducción

En este capítulo se introduce el contexto de la realización de este TFM, se describen los objetivos que se pretenden alcanzar, se presenta un resumen de los campos abordados en este estudio, y se ofrece una explicación detallada de la estructura del proyecto.

Capítulo 2. Antecedentes

Este capítulo examina los fundamentos de *Internet of Things* (IoT), incluyendo sus conceptos básicos, su impacto, y su crecimiento. Además, se abordan las principales tecnologías LPWAN (*Low Power Wide Area Network*) y su estado actual.

Capítulo 3. La tecnología LoRa® y el protocolo LoRaWAN.

La tecnología LoRa® es crucial en IoT por su conectividad de largo alcance y bajo consumo energético. Por su parte, LoRaWAN constituye un protocolo eficiente de área extensa, que proporciona una infraestructura robusta para la transmisión de datos en diversos entornos IoT. Este capítulo analiza las características fundamentales de LoRa® y LoRaWAN.

Capítulo 4. Plataforma HW/SW del nodo IoT

En este capítulo se describe la elección del dispositivo MKRWAN 1310 de Arduino como la plataforma principal para desarrollar e implementar el nodo final en soluciones IoT basadas en LoRaWAN. Se explica que esta elección se debe a su versatilidad, capacidad de comunicación LoRaWAN y soporte para diferentes tipos de aplicaciones IoT. Además, se detalla la integración del módulo *Person Sensor* con el dispositivo Arduino MKRWAN 1310

con el propósito de poder analizar patrones de comportamiento a partir de la detección de rostros de personas.

Capítulo 5. Desarrollo de la aplicación IoT basada en LoRaWAN

En este capítulo se presenta la primera propuesta planteada en el desarrollo del presente TFM, detallándose el proceso de integración del nodo IoT con la plataforma TTN. Se describen las características principales de los *Gateways* LoRaWAN utilizados, correspondientes a los modelos HT-M00 de Heltec, y RAK7268 de *RAK Wireless*, además de explicarse el procedimiento necesario para completar su registro y conexión a la plataforma IoT mediante tecnología WiFi.

Capítulo 6. Integración con InfluxDB

En este capítulo se detallará la integración de TTN con InfluxDB utilizando el agente *Telegraf*. Esta integración es fundamental para aprovechar las capacidades de TTN en la gestión de redes LoRaWAN y combinarla con la eficiencia de InfluxDB Cloud en el almacenamiento y análisis de datos de series temporales.

Capítulo 7. Visualización de datos con Grafana

En este capítulo se explorará cómo Grafana ofrece una plataforma flexible para la visualización de datos, capaz de integrarse con una amplia variedad de fuentes de datos, incluyendo bases de datos de series temporales como InfluxDB.

Capítulo 8. Aplicación IoT LoRaWAN basada en *Network Server* integrado en RAK7268

En este capítulo se presenta la segunda de las propuestas planteada para la implementación de una aplicación IoT basada en el protocolo LoRaWAN que permita realizar el análisis de patrones de comportamiento relacionados con la detección de personas. Se explicará en detalle la configuración del *Gateway* RAK7268 como *Network Server* en una red LoRaWAN, así como del *Broker* MQTT que integra.

Capítulo 9. Conclusiones y Líneas Futuras

En este capítulo se destacan los principales resultados del Trabajo Fin de Máster y se plantean posibles direcciones para futuros desarrollos. Se resumen los resultados

obtenidos, enfocándose en la efectividad de integrar el dispositivo Arduino MKRWAN 1310 con el módulo *Person Sensor* y el planteamiento de diferentes alternativas para la implementación de plataformas IoT orientadas al registro y análisis de datos en redes LoRaWAN. Finalmente, se evalúa la funcionalidad de esta infraestructura en términos de conectividad, eficiencia energética y precisión en la detección de patrones de comportamiento de personas.

BIBLIOGRAFÍA

Compilación de todas aquellas fuentes, como trabajos académicos, libros, páginas web, guías y otros recursos, que han brindado respaldo y han fundamentado los conocimientos expuestos en este trabajo.

PARTE II. PRESUPUESTO

Incluye un desglose del presupuesto necesario para la realización de este Trabajo Fin de Máster. Se presenta una recopilación de los costes humanos requeridos, así como aquellos relacionados con la adquisición de los materiales necesarios para la implementación de la plataforma y el coste de redacción de este documento, entre otros.

PARTE III. PLIEGO DE CONDICIONES

Se detallan los parámetros y límites dentro de los cuales se ha desarrollado el proyecto. En esta sección se describirán las características mínimas necesarias de los materiales y las versiones de los programas utilizados durante el desarrollo del proyecto.

PARTE IV. ANEXO

Por último, se encuentra un anexo en el que se proporciona información adicional que complementa los detalles presentados en el desarrollo del trabajo.

2 Antecedentes

En el contexto actual de rápida evolución tecnológica, la proliferación de dispositivos conectados a la red, impulsada por *Internet of Things* (IoT), ha transformado significativamente la forma de interactuar con el entorno digital y físico. Este fenómeno ha generado un creciente interés, tanto en la academia como en la industria, motivando una amplia gama de investigaciones y desarrollos en áreas como la comunicación inalámbrica, la eficiencia energética, el procesamiento de datos y la ciberseguridad. La expansión de IoT ha abierto nuevas perspectivas y desafíos en diversos campos, desde la automatización industrial hasta la salud digital, estableciendo así un contexto dinámico y multidisciplinar para la exploración y el análisis en el marco de la investigación tecnológica.

2.1 Internet of Things

El término *Internet of Things* fue utilizado por primera vez en 1999 por Kevin Ashton durante una presentación de *Procter and Gamble*. En esta presentación, Ashton explicó los posibles beneficios de utilizar la tecnología *Radio Frequency Identification* (RFID) en la gestión de mercancías. Al equipar los productos con dispositivos especiales, podrían obtener información de interés (estado, trazabilidad, etc.). De esta manera, las denominadas *things* podrían proporcionar información sobre su estado y el mundo circundante, pero de una manera mucho más eficiente [6].

2.1.1 Concepto

El paradigma de IoT se refiere a un sistema de dispositivos interconectados entre sí, equipados con capacidad computacional (*smart objects*), identificables y habilitados para transferir datos a través de una red, sin requerir interacción humana. El concepto detrás

de este paradigma es la presencia generalizada de dispositivos inteligentes que, al cooperar entre sí, e interactuar con los seres humanos, logran objetivos comunes [6].

IoT es una de las tecnologías clave que respalda la cuarta Revolución Industrial y el concepto de Industria 4.0. Los dispositivos y sistemas de IoT permiten que se detecten, recopilen y almacenen grandes cantidades de datos para su posterior procesamiento mediante dispositivos conectados. Generalmente, los datos se procesan en servidores centralizados basados en la nube que permiten garantizar un alto rendimiento. Los resultados pueden utilizarse para la toma de decisiones en diferentes niveles y facilitar la interacción en tiempo real entre diferentes sectores. A pesar de los posibles beneficios de IoT, aún quedan varios desafíos y limitaciones por abordar, entre los que se encuentran seguridad y privacidad, comunicación, hardware/software, o regulación y legislación [1].

2.1.2 Impacto y crecimiento

El informe más reciente sobre el estado de IoT elaborado por la empresa *IoT Analytics*, denominado *"State of IoT—Spring 2023"*, muestra que el número de conexiones IoT a nivel mundial creció un 18% en 2022, alcanzando los 14.3 mil millones de dispositivos finales IoT activos, como se observa en la Figura 2. *IoT Analytics* espera que el número global de dispositivos IoT conectados a Internet crezca otro 16%, llegando a 16.7 mil millones de nodos finales activos [7]. Según el análisis de *IoT Analytics*, para 2027 se prevé que existan más de 29 mil millones de conexiones de IoT.

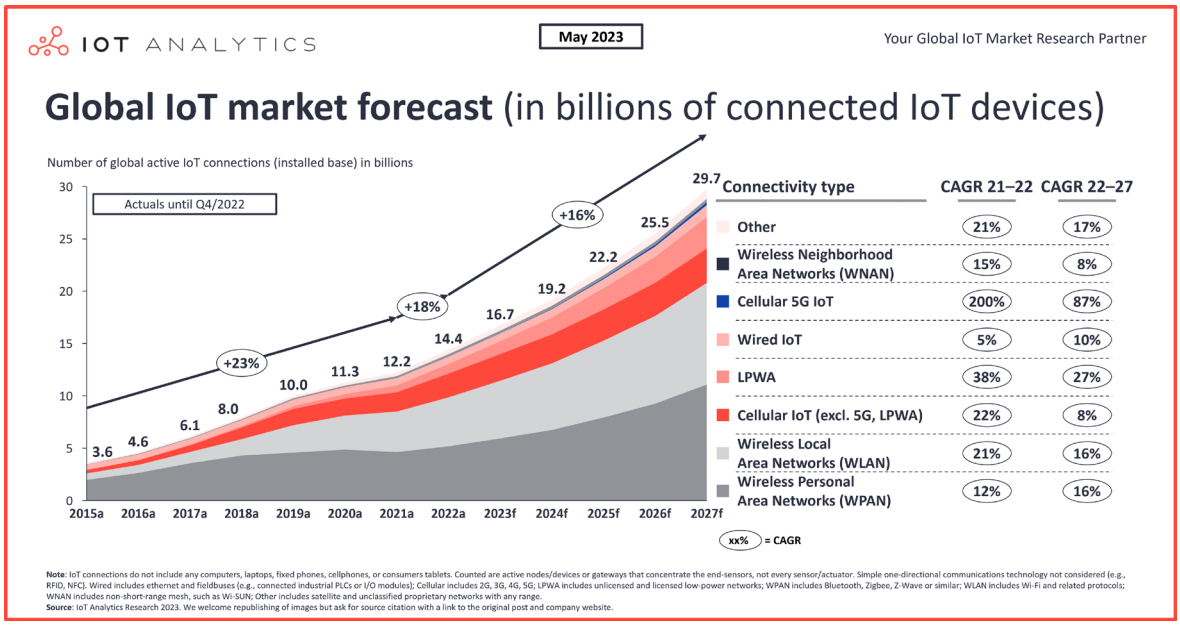


Figura 2 Estado y predicción del crecimiento de dispositivos IoT en los últimos años [7].

En los últimos años, China se ha posicionado como el líder en nuevas conexiones de dispositivos IoT, con más de dos mil millones de conexiones celulares activas en 2022. Sin embargo, es probable que estos años de crecimiento intenso lleguen a su fin debido a diversos problemas que debe afrontar el país. Entre estos problemas destaca la escasez de suministros tecnológicos, debido a las tensiones comerciales renovadas entre Estados Unidos y China, especialmente en la industria de semiconductores. En octubre de 2022, Estados Unidos impuso un embargo de exportación a China, lo que causó una interrupción significativa en estas industrias. Como resultado, empresas de circuitos integrados están trasladando parte de su producción fuera de China, como es el caso de Infineon, TSMC *(Taiwan Semiconductor Manufacturing Company)* y *Advanced Semiconductor Materials Lithography* (ASML) [7].

A pesar de que el panorama general de la conectividad de IoT evoluciona gradualmente, con dispositivos que permanecen conectados durante períodos prolongados de tiempo, las nuevas tecnologías de conectividad IoT tienen un impacto significativo en el panorama a largo plazo. Entre estos avances destacan dos desarrollos de especial interés que están siendo objeto de seguimiento.

Por un lado, en 2022, la industria *Low Power Wide Area Network (*LPWAN) experimentó dos eventos significativos que cambiaron el enfoque de la competencia entre tecnologías

de conectividad LPWAN a la coexistencia y convergencia de esas tecnologías. La adquisición de Sigfox por parte de UnaBiz y la adquisición de Sierra Wireless por la empresa Semtech allanaron el camino para que las empresas LPWAN proporcionen multi-conectividad de varias tecnologías LPWAN. Para hacer que se de esta convergencia, y poder implementar cualquier tecnología LPWAN, UnaBiz, por ejemplo, colabora en la actualidad con plataformas como *The Things Network* (TTN), *Actility*, *Soracom* o *LORIOT*, entre otras. De esta manera, UnaBiz se ha convertido en más que un proveedor de tecnología, proporcionando soluciones que agrupan diferentes tecnologías en su propia plataforma software. UnaBiz es solo un ejemplo que muestra cómo la industria está pasando de una visión de tecnología LPWAN única, a soluciones basadas en multi-conectividad [7].

Por otro lado, la conectividad satelital *Low Earth Orbit* (LEO) para IoT está ganando popularidad al brindar amplia cobertura, retardos mínimos y gran fiabilidad. Esta tecnología es especialmente útil en las industrias de agricultura, marítima y logística. Los satélites LEO están más cerca de la Tierra que los satélites tradicionales, lo que resulta en una latencia reducida y una transmisión de datos más rápida, aspectos esenciales para el procesamiento de datos en tiempo real. Este tipo de conectividad es más resistente y fiable, garantizando una comunicación consistente, incluso en entornos adversos o ante desastres naturales. Los avances en la conectividad IoT vía satélite LEO continúan optimizando su rendimiento y mejorando la experiencia del usuario [7].

Según *IoT Analytics*, se espera que las conexiones de IoT vía satélite crezcan de seis millones a 22 millones entre 2022 y 2027, con una tasa de crecimiento anual compuesta del 25%. Si bien se espera que este crecimiento tenga un efecto menor en el mercado general, la integración de opciones de conectividad satelital en *chipsets* LPWAN por empresas como Qualcomm, podría acelerar la adopción. Se espera que esta integración de conectividad satelital en chipsets LPWAN impulse una mayor innovación y crecimiento en el mercado de IoT [7].

2.2 Tecnologías LPWAN

En el ámbito de las aplicaciones IoT se pueden encontrar diversos protocolos, tales como Bluetooth, Zigbee, *Wireless Fidelity* (Wi-Fi), 2G, 3G, 4G, 5G o *Near Field Communication* (NFC), entre otros. Por lo general, estos se clasifican en función de su ancho de banda y alcance, como se ilustra en la Figura 3. A primera vista, podría parecer que los protocolos ubicados en la parte superior derecha son considerablemente superiores, ofreciendo un mejor alcance y ancho de banda, mientras que aquellos en la parte inferior izquierda podrían considerarse menos efectivos. Sin embargo, es importante destacar que esta representación gráfica no refleja el consumo de energía asociado con cada protocolo, que representa un aspecto importante a considerar [8].

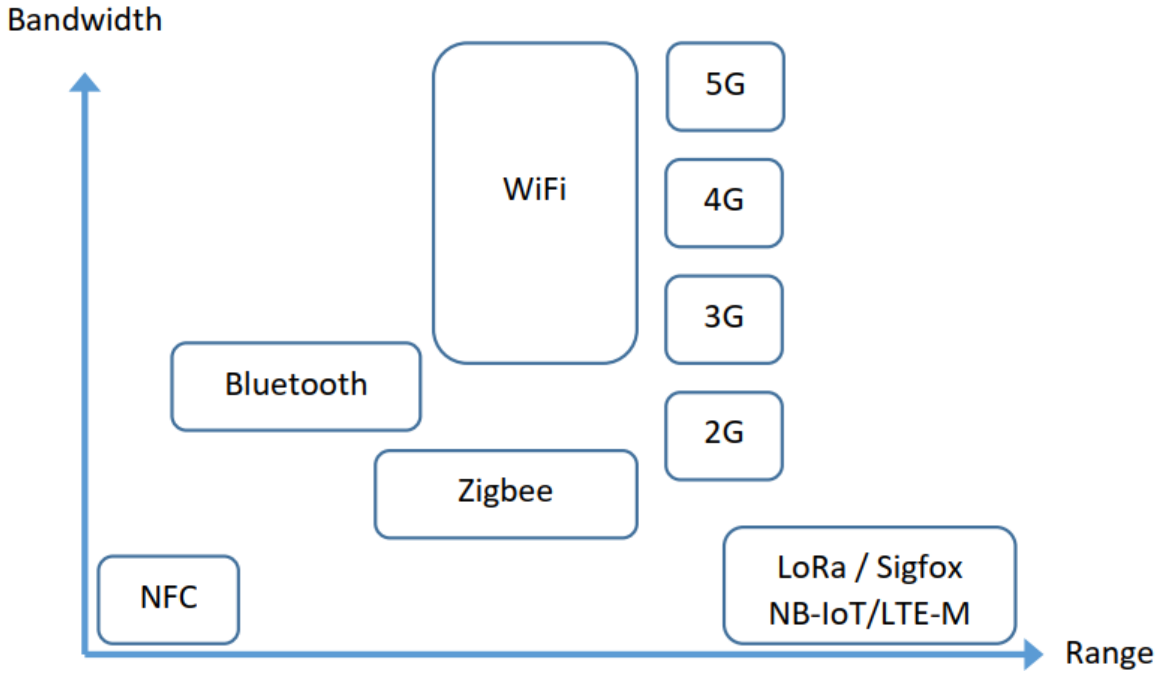


Figura 3 Protocolos usados en IoT [8].

En la parte inferior derecha se encuentran los protocolos LPWAN de baja velocidad de transmisión y largo alcance, que presentan requisitos de energía muy bajos, como *Narrowband Internet of Things* (NB-IoT) y *Long Term Evolution for Machines* (LTE-M). Sigfox y LoRaWAN se consideran protocolos de alcance extremadamente largo y sumamente eficientes en términos de consumo de energía [8].

En este contexto, destacan tres tecnologías clave: Wi-Fi, LTE-M/NB-IoT y LoRaWAN. Cada una de estas tecnologías juega un papel crucial al ofrecer distintos niveles de alcance, velocidad y eficiencia energética, adaptándose a las diversas necesidades de los dispositivos IoT. En esta sección se presentan las principales características de estas tecnologías, analizando su integración para la implementación de soluciones robustas en el vasto ecosistema del IoT.

2.2.1 Wi-Fi

La tecnología Wi-Fi ha sido una fuerza transformadora en la forma de trabajar, vivir y de entretenerse de las personas. Ha impulsado numerosas innovaciones en diversas industrias y aplicaciones, desde dispositivos y servicios hasta segmentos de mercado completos, como hogares inteligentes y *Extended Reality* (XR). Con más de 21 mil millones de dispositivos Wi-Fi en uso actualmente, su amplia presencia en *smartphones* y *tablets*, así como su papel crucial en la conectividad para hogares, empresas y proveedores de servicios, ha trascendido las expectativas, convirtiéndose en un motor de crecimiento económico y progreso social [9].

A lo largo del último cuarto de siglo, la tecnología Wi-Fi ha evolucionado y expandido sus capacidades, logrando avances significativos en rendimiento, alcance, fiabilidad y seguridad. En la actualidad, Wi-Fi CERTIFIED 7™ habilita casos de uso avanzados anteriormente impracticables, demostrando el compromiso continuo de *Wi-Fi Alliance* en ofrecer soluciones y características que enriquezcan las experiencias de usuario a nivel global [9].

La tecnología Wi-Fi destaca por su enfoque en la interoperabilidad basada en estándares, brindando una plataforma común para diversas aplicaciones de IoT. Además, su conectividad facilita el control eficiente de sistemas de IoT mediante dispositivos móviles. En cuanto a seguridad, Wi-Fi ofrece la tecnología *Wi-Fi Protected Access* (WPA3), asegurando la protección de información sensible tanto en entornos personales como empresariales [10].

La implementación de Wi-Fi destaca por su asequibilidad y simplicidad, eliminando la necesidad de *Gateways* o puertas de paso para aplicaciones de IoT. La compatibilidad en todos sus estándares garantiza una inversión duradera y minimiza el riesgo de obsolescencia prematura, facilitando la conexión de dispositivos heredados a las redes domésticas. En términos de conectividad, las redes Wi-Fi ofrecen eficiencia, diagnósticos y capacidades de optimización avanzadas, mejorando significativamente la eficiencia energética para dispositivos alimentados por batería en implementaciones de IoT doméstico [10].

Wi-Fi 6 y Wi-Fi 6E gestionan múltiples dispositivos conectados simultáneamente con alto rendimiento y baja latencia, siendo ideales para aplicaciones exigentes como transmisión de video 4K, o realidad aumentada. Wi-Fi CERTIFIED HaLow™ mejora el alcance y el acceso en áreas densas, aumenta el número de dispositivos por punto de acceso y ofrece conectividad de bajo consumo para dispositivos alimentados por baterías de Litio. La topología de red flexible de Wi-Fi CERTIFIED EasyMesh® extiende la cobertura de Wi-Fi por todo el hogar, manteniendo conectados los dispositivos de IoT mediante múltiples puntos de acceso que trabajan en conjunto para formar una red unificada [10]

En conjunto, estas características resaltan la posición destacada de Wi-Fi como un componente esencial para el éxito y la expansión continua de IoT, ofreciendo un amplio soporte para diversas aplicaciones y consolidando su papel como habilitador clave en la conectividad global.

2.2.2 LTE-M/NB-IoT

En el ámbito de IoT, en el que la eficiencia energética resulta especialmente importante, NB-IoT y LTE-M son tecnologías de radiocomunicación móvil especialmente eficientes desde el punto de vista del consumo de energía [11]. Ambas tecnologías de radio son idóneas para dispositivos finales que sólo transmiten pequeñas cantidades de datos entre periodos prolongados de tiempo. Los módulos de radio sencillos, que se limitan a las funciones básicas necesarias, hacen que ambos estándares de red presenten un bajo coste y reducido consumo de energía.

A pesar de todas las similitudes entre estas dos tecnologías, también hay algunas diferencias entre NB-IoT y LTE-M, de forma que cada una de ellas presenta sus propios puntos fuertes y, por tanto, resultan especialmente adecuadas para determinados ámbitos de aplicación [11]. LTE-M es una tecnología móvil más reciente desarrollada específicamente para IoT. El estándar de radio móvil LTE-M se basa en 4G y está destinado a la conectividad móvil en IoT. Por ello, se utiliza principalmente para conectar objetos físicos de forma inalámbrica a Internet. Esta conectividad inalámbrica a través de LTE-M es la base de nuevas e interesantes aplicaciones [11].

En comparación con NB-IoT, LTE-M ofrece mayores velocidades de datos (hasta 1 Mbit/s) y menores latencias. Tanto el enlace ascendente como el descendente pueden establecerse en cualquier momento. A largo plazo, LTE-M también debería ser compatible con *Voice over Long Term Evolution* (VoLTE), lo cual beneficiaría, sobre todo, a las soluciones de seguridad en las que se activan las llamadas de emergencia en los coches o los ascensores. Debido a que las bandas de frecuencias tienen licencia, se necesitan tarjetas SIM M2M especiales para el uso de LTE-M. Los principales operadores de red, así como diferentes operadores independientes, proporcionan tarjetas SIM LTE-M [11].

El hecho de que LTE-M se base en el estándar LTE 3GPP lo hace apto para 5G. El uso de esta tecnología sólo requiere actualizar el software en la red existente. Se espera que la mayoría de los operadores de red acaben ofreciendo ambas tecnologías como parte de su estrategia de IoT, en lugar de optar por LTE-M o NB-IoT [11].

LTE-M, a diferencia de NB-IoT, es capaz de gestionar el cambio de celda de radio, o *handover*, mientras que un dispositivo NB-IoT tiene que desconectarse y volver a conectarse cada vez que cambia de una célula de radio a otra. Con LTE-M, este cambio se produce sin interrupción, lo que hace que resulte especialmente adecuada para aplicaciones con sensores que se encuentren en movimiento. En este caso, las aplicaciones también se benefician de la mayor velocidad de datos, de hasta 1 Mbit/s, lo que significa que se pueden transmitir más datos en menos tiempo que con NB-IoT. En general, el tiempo de transmisión de un paquete de datos desde el origen hasta el destino sigue siendo algo menor que con NB-IoT. Esto hace también que LTE-M resulte adecuada para aplicaciones en las que sea importante recibir los datos lo más cerca posible del momento del procesamiento. En cambio, la penetración en el edificio de LTE-M es algo

peor que con NB-IoT. Por tanto, la recepción en sótanos, aparcamientos y zonas más protegidas es mayor con NB-IoT que con LTE-M. Además, el consumo de energía –aunque resulte ya muy reducido con LTE-M– es aún menor con NB-IoT. Por lo general, los módulos hardware para LTE-M también presentan un coste algo mayor que los de NB-IoT [11].

Tanto los módulos NB-IoT como LTE-M están diseñados con un enfoque específico en las características de radio y en la implementación de funciones destinadas al ahorro de energía, tales como el modo de reposo o actualizaciones periódicas de la zona de seguimiento, lo que asegura una eficiencia energética óptima en ambas tecnologías [11].

Además, en términos de cobertura de red, NB-IoT y LTE-M superan a GSM (*Global System For Mobile Comunication*) al ofrecer una densidad de potencia hasta 20 decibelios (NB-IoT) y 15 decibelios (LTE-M) mayor, a partir del uso de métodos de modulación de banda estrecha y repeticiones múltiples de transmisión, especialmente beneficiosas para entornos con edificaciones. La instalación, tanto de NB-IoT como de LTE-M se simplifica enormemente mediante la tecnología *Plug & Play*, que no requiere la instalación de redes locales o puertas de enlace, permitiendo que los sensores se conecten directamente a la red NB-IoT y/o LTE-M [11].

Finalmente, al ser tecnologías estandarizadas a nivel mundial, en NB-IoT y LTE-M son prioritarias la seguridad y la normalización, implementando mecanismos de seguridad LTE según las pautas del comité 3GPP, con un constante proceso de revisión y mejora de las funciones de seguridad [11].

2.2.3 LoRaWAN

Entre los estándares LPWAN, el protocolo LoRaWAN ha experimentado un crecimiento notable en los últimos años y se anticipa que su adopción continuará aumentando. La conectividad inalámbrica proporcionada por la organización *LoRa Alliance®* destaca por su innovación, seguridad y asequibilidad, facilitando una gestión simplificada, una implementación sencilla y una escalabilidad efectiva [12].

En los últimos cinco años, el despliegue de redes LoRaWAN ha experimentado un continuo crecimiento, contando actualmente con 133 operadores de red que

implementan esta tecnología. Es importante destacar que esta cifra no engloba los despliegues privados de la tecnología, lo que eleva en total a 143 países con algún tipo de red basada en LoRaWAN. Este crecimiento sustancial se atribuye principalmente a la creciente demanda del mercado en busca de conectividad eficiente que aborda de manera específica las necesidades de un nicho de mercado. Además, el ecosistema continúa expandiéndose, con la presencia en el mercado de 245 dispositivos certificados por la organización LoRa Alliance [13].

LoRaWAN es una de las tecnologías LPWAN más prometedoras, proporcionando comunicación con bajo consumo, reducido coste, largo alcance y baja velocidad de transferencia de datos. El sistema de comunicación LoRaWAN consta de dispositivos finales, puertas de enlace o *gateways*, Servidores de Red *(Network Servers)* y Servidores de Aplicaciones *(Application Servers)* [13].

El sistema de comunicación LoRaWAN se enfoca principalmente en las capas física y MAC (*Media Access Control*) de su pila de protocolos. *Long Range* (LoRa®) es la implementación de la capa física de LoRaWAN estandarizada por el consorcio *LoRa Alliance®*, mientras que LoRaWAN es el protocolo de capa MAC promovido por *LoRaWAN Alliance* [3]. LoRa® permite la conexión de dispositivos a larga distancia con una baja tasa de bits y un reducido consumo de potencia. Por esta razón, es uno de los candidatos más adecuados para la mayoría de las aplicaciones de IoT. Sin embargo, la implementación de dispositivos basados en LoRa® está restringida a aquellos escenarios que admiten una reducida tasa de bits. Además, el éxito de LoRaWAN depende de la seguridad de la red. En Internet, se puede obtener una mejor seguridad y facilidad de gestión mediante la implementación de un control de seguridad basado en roles que se centre en la separación dinámica de tareas. Además, el éxito del sistema de comunicación LoRa® radica en su apertura y la disponibilidad de su soporte de software de código abierto [14].

La adopción global de LoRaWAN revela un sólido crecimiento en áreas como edificaciones inteligentes, servicios públicos, ciudades, agricultura e industria. *LoRa Alliance®* también muestra una continua diversificación en su ecosistema colaborativo, fomentando soluciones integrales de IoT en todas las etapas del proceso. En las solicitudes de propuestas, las ciudades inteligentes están incorporando cada vez más dispositivos finales certificados por LoRaWAN. La tecnología ha consolidado su posición de liderazgo en la

Industria 5.0, cumpliendo con los requisitos de sostenibilidad, eficiencia y calidad de vida [15].

En resumen, las tecnologías de comunicación inalámbrica como Wi-Fi, LoRaWAN, LTE-M o NB-IoT ofrecen diversas ventajas para proyectos de IoT en términos de eficiencia energética, coste, cobertura y facilidad de instalación. En el presente TFM se ha decidido profundizar en el protocolo LoRaWAN debido a sus características específicas. La elección de LoRaWAN se fundamenta en su capacidad para proporcionar una cobertura amplia y fiable en entornos donde otras tecnologías pueden presentar limitaciones, como áreas rurales o urbanas densamente pobladas. Su reducido consumo de potencia y larga duración de la batería hacen que resulte ideal para dispositivos IoT que requieren operar de manera autónoma durante largos períodos sin intervención humana. Además, la infraestructura descentralizada y la capacidad de operar en redes privadas ofrecen flexibilidad y control sobre la gestión de los datos y la seguridad, lo que resulta fundamental.

En consecuencia, las características particulares del protocolo LoRaWAN se ajustan de manera precisa a los requisitos y objetivos específicos establecidos en el desarrollo de este Trabajo Fin de Máster, proporcionando una solución de conectividad en IoT fiable y eficiente para las necesidades que se desean cubrir.

3 La tecnología LoRa® y el protocolo LoRaWAN.

En el ámbito de las tecnologías de comunicación inalámbrica, LoRa® destaca como elemento clave en el escenario de IoT. LoRa® es una tecnología de modulación patentada que ofrece una conectividad de largo alcance y bajo consumo de energía, mientras que LoRaWAN representa un protocolo de comunicación de área extensa y alta eficiencia energética, que proporciona una sólida infraestructura para la transmisión de datos en una variedad de entornos IoT a través de Internet. Este capítulo tiene como objetivo analizar las principales características de la tecnología LoRa® y el protocolo LoRaWAN.

3.1 LoRa®

LoRa®, acrónimo de *Long Range*, es una tecnología inalámbrica de transmisión de datos vía radio basada en una variante de la modulación CSS (Chirp Spread Spectrum), o modulación de espectro ensanchado de frecuencia pulsada, que permite establecer comunicaciones a larga distancia con un reducido ancho de banda y alta inmunidad al ruido, minimizando el consumo de potencia [16]. Estas características hacen que la tecnología LoRa® resulte idónea para aplicaciones en las que se requiera transmitir reducidas cantidades de datos a larga distancia, con un mínimo consumo de potencia, como en el caso de IoT.

Figura 4 Logo de LoRa®.

La técnica de modulación LoRa® fue inventada en el año 2010 por la *start-up* francesa Cycleo, que fue adquirida por la empresa Semtech Corporation en 2012, siendo una

tecnología propietaria. Las aplicaciones idóneas para la tecnología LoRa® son aquellas que requieren la transmisión de una reducida cantidad de datos a baja velocidad. Su capacidad de alcanzar distancias notables, de hasta 30Km –dependiendo de las condiciones ambientales y de la configuración usada– con consumos de potencia del orden de mW, supera a otras tecnologías inalámbricas como Wi-Fi o BLE *(Bluetooth Low Energy)*, aunque con una reducida tasa de datos (≤ 50Kbps), como se observa en la Figura 5 [16].

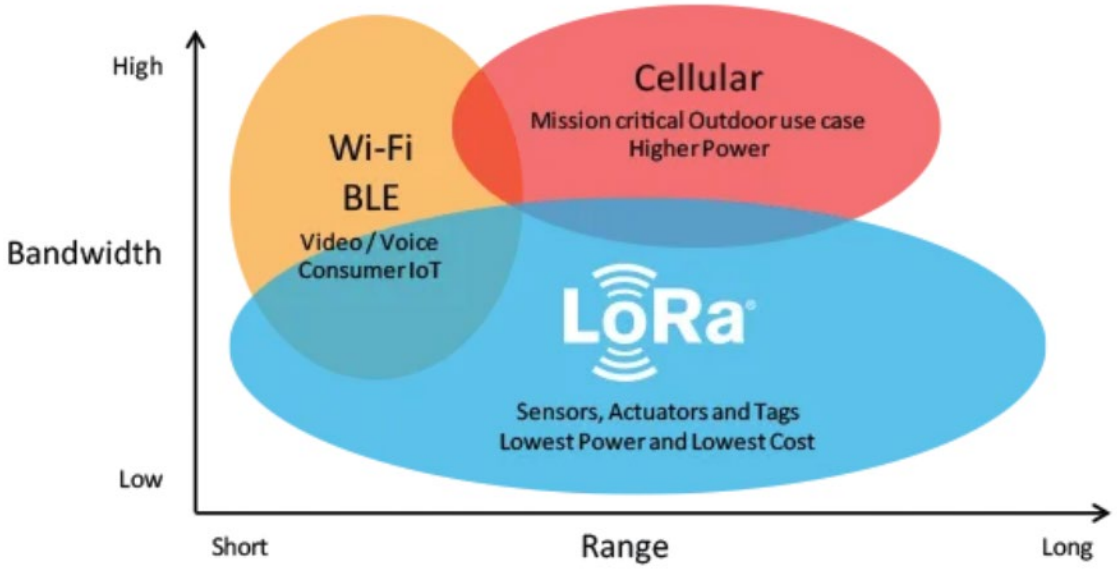


Figura 5 Comparación entre ancho de banda y rango por tecnologías [16].

Un aspecto significativo de la tecnología LoRa® es que opera en bandas sin licencia, tales como 915 MHz, 868 MHz y 433 MHz. Asimismo, puede operar en la frecuencia de 2.4 GHz para obtener tasas de transmisión de datos superiores en comparación con las bandas inferiores a 1 GHz, si bien esto se traduce en una disminución del alcance efectivo. Cabe destacar que estas frecuencias se encuentran dentro de las bandas *Industrial, Scientific, and Medical* (ISM), internacionalmente reservadas para propósitos de índole industrial, científica y médica [16].

La técnica de modulación CSS fue desarrollara alrededor del año 1940 para aplicaciones de radar, siendo utilizada tradicionalmente en aplicaciones militares basadas en comunicaciones seguras. En los últimos 20 años, esta técnica ha sido adoptada en numerosas aplicaciones para la comunicación de datos, debido fundamentalmente a su

reducido consumo de potencia e inherente inmunidad ante la degradación de canal provocada por interferencias o pérdidas de señal.

En concreto, la modulación CSS en la que se basa la tecnología propietaria LoRa®, codifica los datos a partir de un *chirp* –barrido de frecuencia–, que representa esencialmente una señal sinusoidal cuya frecuencia aumenta (*up-chirp*) o disminuye (*down-chirp*) linealmente con el tiempo en un rango determinado por el ancho de banda, como se representa en la Figura 6, lo que permite reducir el efecto de las posibles interferencias.

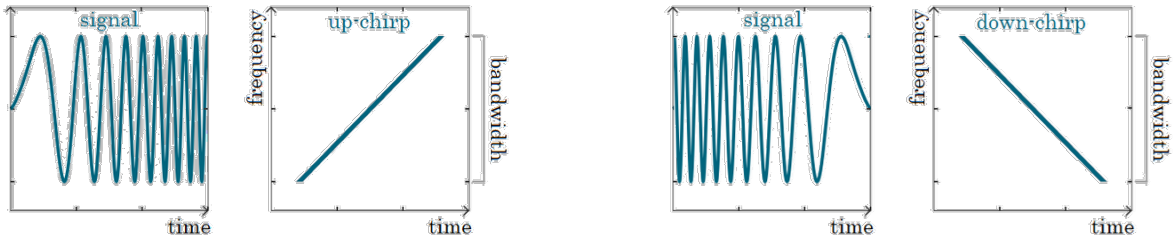


Figura 6 Up-chirp y down-chirp en la modulación LoRa®.

Por otro lado, la velocidad a la que varía la frecuencia del *chirp* dentro del ancho de banda, está determinada por un parámetro denominado SF (*Spreading Factor*). En consecuencia, dependiendo del valor del BW seleccionado, el parámetro SF determina el tiempo asociado al barrido de frecuencia de un *chirp* –y en consecuencia, el Tiempo de Símbolo (TS)–, será mayor o menor.

La modulación LoRa® soporta un total de 6 factores de dispersión, desde SF7 a SF12, cuyo valor influye en la velocidad de transferencia de los datos, el tiempo de transmisión en el aire, la duración de la batería, y la sensibilidad del receptor, como se representa en la Figura 7 [17].

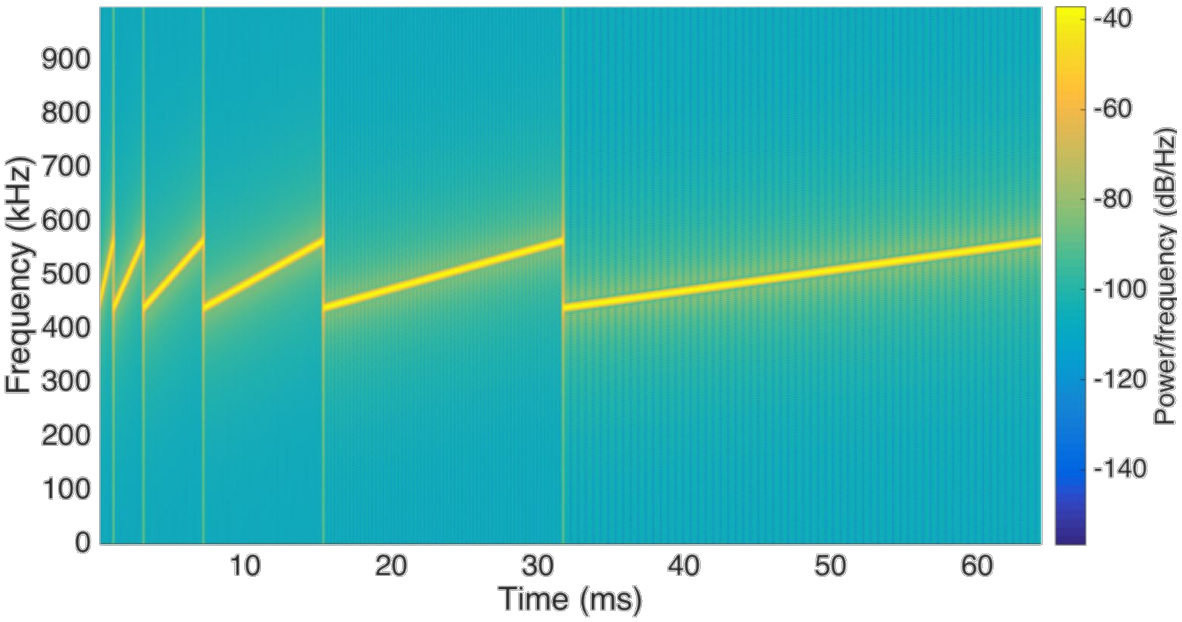


Figura 7 Influencia del parámetro SF en la modulación LoRa®.

En términos más específicos, cabe destacar que factores de dispersión más bajos están asociados con *chirps* más rápidos, lo que implica una tasa de transmisión de datos mayor. Por el contrario, un incremento en el valor del factor de dispersión conlleva una reducción a la mitad en la velocidad de barrido de *chirp* y, por consiguiente, una disminución equivalente en la tasa de transmisión de datos [17].

La selección de factores de dispersión más bajos tiene como consecuencia la reducción en el alcance de las transmisiones LoRa®. Esto se debe a que los factores de dispersión más bajos limitan la ganancia de procesamiento y aumentan la velocidad de bits, lo que da como resultado un alcance reducido. En última instancia, el ajuste del factor de dispersión otorga a la red la capacidad de modular la velocidad de datos para cada dispositivo final, aunque esto puede traducirse en una disminución en el alcance de la comunicación [17].

Además, los factores de dispersión se pueden emplear como un mecanismo eficaz para gestionar la congestión de la comunicación. Esta estrategia se basa en la ortogonalidad de los factores de dispersión, lo que significa que las señales moduladas con diferentes factores de dispersión, y transmitidas en el mismo canal de frecuencia, pueden coexistir sin interferirse mutuamente [17].

Por lo general, un menor factor de dispersión conlleva la obtención de una tasa de bits superior, manteniendo constantes tanto el ancho de banda como la tasa de codificación o *Coding Rate* (CR) [17].

Es relevante destacar que duplicar el ancho de banda conlleva automáticamente una duplicación en la tasa de bits, siempre que se mantengan constantes el factor de dispersión y la velocidad de codificación. En la Tabla 1 se presentan datos específicos sobre las tasas de bits considerando SF7 y una tasa de codificación (CR) de 1, en el contexto de anchos de banda de 125 KHz, 250 KHz y 500 kHz [17].

Tabla 1 Comparación de ancho de banda y velocidad de transmisión [17].

Spread Factor (SF)	**Ancho de Banda**	**Velocidad de bits (kbits/s)**
7	125 KHz	5.5
7	250 KHz	10.9
7	500 KHz	21.9

Por otro lado, factores de dispersión mayores conllevan una ganancia de procesamiento mayor y, por lo tanto, una señal modulada con un factor de dispersión mayor se puede recibir con menos errores en comparación con una señal con un factor de dispersión inferior y, con ello, alcanzar una distancia mayor [17]. En consecuencia, factores de dispersión más altos proporcionan una mayor sensibilidad del receptor, por lo que generalmente LoRa® utiliza factores de dispersión más altos cuando la señal es débil. La Tabla 2 muestra cómo los factores de dispersión afectan la sensibilidad del receptor.

Tabla 2 Sensibilidad necesaria para diferentes factores de dispersión [17].

Spread Factor (SF)	**Sensibilidad del receptor para ancho de banda fijado en 125 kHz**
SF7	-123dBm
SF8	-126dBm
SF9	-129 dBm
SF10	-132dBm
SF11	-134,5dBm
SF12	-137 dBm

Sin embargo, en comparación con un factor de dispersión más bajo, transferir una determinada cantidad de datos como carga útil, con un factor de dispersión mayor y un ancho de banda fijo, requiere de un mayor tiempo de transmisión en el aire [17].

Finalmente, la duración de la batería de un dispositivo final depende en gran medida del factor de dispersión utilizado. Factores de dispersión más elevados dan como resultado tiempos activos más largos para los transceptores de radio, y con ello una vida útil más corta de la batería [17].

En la Figura 8 se representa de forma gráfica la relación entre el parámetro SF, la tasa de transferencia de datos, el tiempo de transferencia en el aire, la distancia alcanzada y el consumo de potencia en la tecnología de modulación LoRa®.

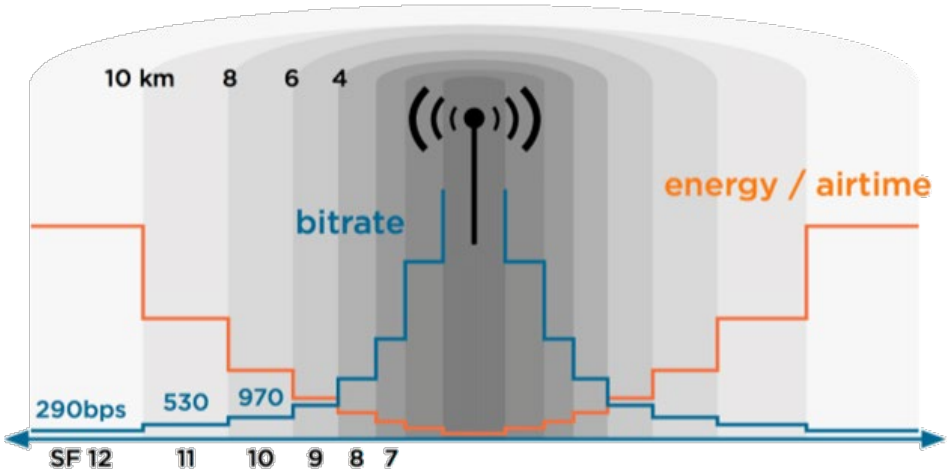


Figura 8 Relación entre el parámetro SF y las prestaciones de la modulación LoRa® [17].

3.2 LoRaWAN

La tecnología LoRa® puede utilizarse, tanto para establecer comunicaciones punto a punto entre dispositivos, como para crear redes de dispositivos. En las comunicaciones punto a punto, la transmisión/recepción de datos entre dispositivos se realiza de forma directa mediante señales RF moduladas/demoduladas en un transceptor LoRa®. Por otro lado, en una red de dispositivos LoRa®, la transmisión/recepción de datos se basa en el protocolo LoRaWAN (*Long Range Wide Area Network*).

Figura 9 Logo de LoRaWAN.

LoRaWAN se configura como un protocolo de capa de control de acceso al medio (MAC) que establece los parámetros de uso de la tecnología de comunicación LoRa®, incluyendo aspectos como el momento de transmisión y el formato de los mensajes [16].

El protocolo LoRaWAN es una especificación de red LPWAN creada a partir de la tecnología LoRA®, desarrollado y estandarizado por la organización *Lora Alliance*®. La primera especificación de LoRaWAN se públicó en el año 2015, sucediéndose diversas versiones a lo largo del tiempo, recogidas en la Tabla 3 [16].

Tabla 3 Historial de versiones de especificaciones LoRaWan [16].

Versión	**Fecha de lanzamiento**
1.0	enero 2015
1.0.1	febrero 2016
1.0.2	julio 2016
1.1	octubre 2017
1.0.3	julio 2018
1.0.4	octubre 2020

LoRaWAN es un protocolo adecuado para transmitir cargas útiles de pequeño tamaño, como datos de sensores a largas distancias. La modulación LoRa® proporciona un rango de comunicación significativamente mayor, con menores anchos de banda bajos que otras tecnologías de transmisión de datos inalámbricas.

3.2.1 LoRa Alliance®

LoRa Alliance® es una entidad sin ánimo de lucro establecida en el año 2015, que destaca como una organización de alcance global. Su principal misión radica en respaldar el

desarrollo y la promoción de LoRaWAN, además de asegurar la interoperabilidad de todos los productos y tecnologías relacionadas con este protocolo. En la actualidad, esta congrega a una membresía internacional que supera los 500 miembros [16].

Figura 10 Logo de Lora Alliance [18].

Un importante servicio que brinda *LoRa Alliance®* consiste en la certificación de dispositivos finales compatibles con LoRaWAN. Los dispositivos que obtienen esta certificación confieren a los usuarios la confianza de ser confiables y que cumplen estrictamente con los parámetros y especificaciones definidos por LoRaWAN. Cabe señalar que el acceso a esta certificación se encuentra reservado exclusivamente para aquellos fabricantes de dispositivos que ostentan su condición de miembros de *LoRa Alliance®*. Una vez que un dispositivo ha obtenido la certificación, el fabricante se encuentra habilitado para emplear la marca de LoRaWAN en el producto certificado, lo que aporta una valiosa señal de calidad y cumplimiento normativo [16].

Adicionalmente, es relevante destacar que LoRaWAN ha alcanzado el estatus de estándar reconocido por la Unión Internacional de Telecomunicaciones (UIT). Este importante logro fue anunciado por la LoRa Alliance® el 7 de diciembre de 2021 y consolida a LoRaWAN como un estándar oficialmente respaldado para LPWAN. Este reconocimiento a nivel internacional subraya la relevancia y confiabilidad de LoRaWAN como un protocolo para aplicaciones de comunicación de amplio alcance y bajo consumo energético [16].

3.2.2 Arquitectura de una red LoRaWAN

Las redes LoRaWAN se despliegan siguiendo una topología en estrella, como se representa en la Figura 11. Por lo general, la configuración típica de una red LoRaWAN contempla los siguientes componentes [19]:

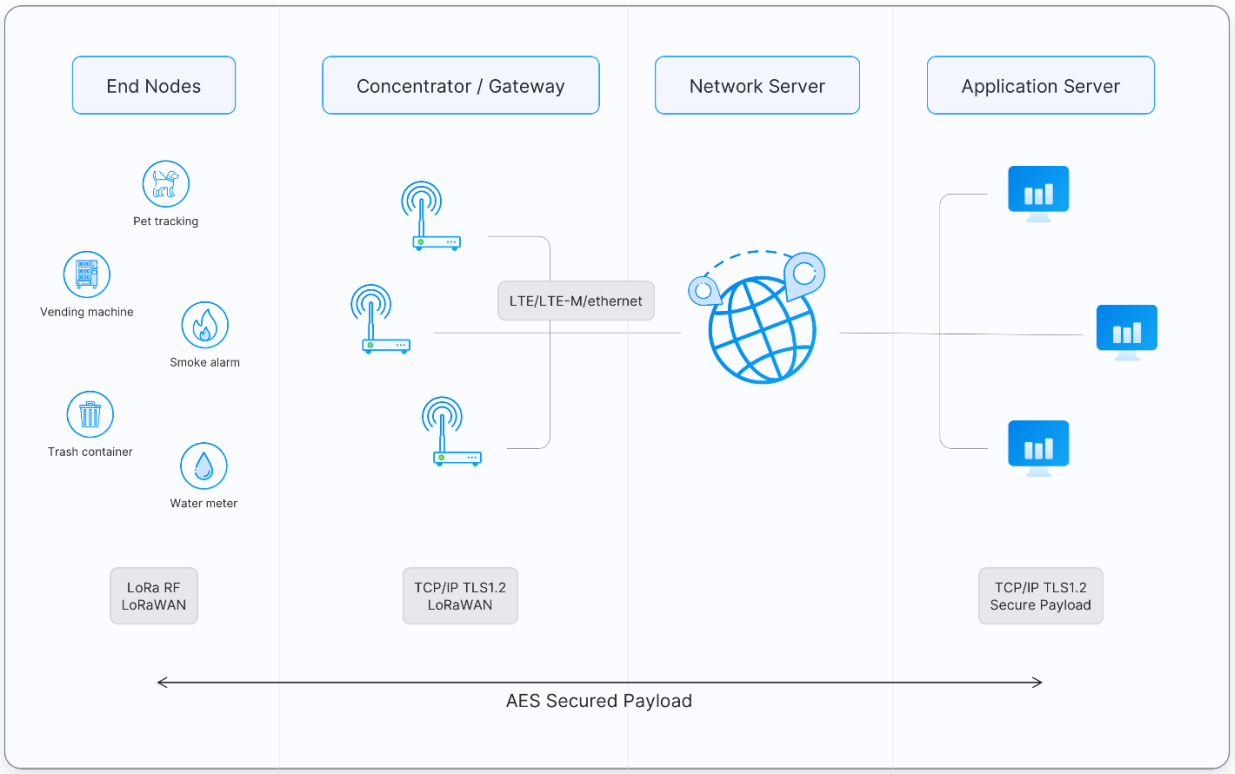


Figura 11 Arquitectura típica tecnología LoRaWAN [19].

1. *End Nodes*: Estos dispositivos, que pueden ser sensores o actuadores, son responsables de la transmisión de mensajes inalámbricos mediante la modulación LoRa®. Asimismo, tienen la capacidad de recibir mensajes de forma inalámbrica procedentes de las puertas de enlace.
2. *Gateways*: Las puertas de enlace constituyen un nodo fundamental en la red, ya que reciben los mensajes transmitidos por los dispositivos finales y los reenvían al Servidor de Red. Desempeñan un papel crucial en la conectividad de los dispositivos con el Servidor Central.
3. *Network Server*: Este elemento software se aloja en un servidor y asume la responsabilidad de gestionar la totalidad de la infraestructura de la red LoRaWAN. Supervisa la comunicación entre dispositivos finales, puertas de enlace y Servidores de Aplicaciones.
4. *Application Server*: Los Servidores de Aplicaciones, también ejecutados como software en servidores, desempeñan un rol esencial en el procesamiento seguro de los datos de la aplicación. Su función principal es recibir, analizar y actuar sobre

la información transmitida por los dispositivos finales, asegurando que se lleve a cabo de manera adecuada y en conformidad con los requerimientos de la aplicación.

5. *Join Server*: Aunque no se refleje de manera explícita en la Figura 11, representa una instancia software que se encuentra alojada en un servidor y se encarga de procesar los mensajes de solicitud de unión enviados por los dispositivos finales. Este servidor juega un papel crítico en el proceso de autenticación y registro de los dispositivos en la red.

Los dispositivos finales se comunican con puertas de enlace cercanas, y cada puerta de enlace está conectada al Servidor de Red. Las redes LoRaWAN utilizan un protocolo basado en ALOHA, por lo que los dispositivos finales no necesitan conectarse con puertas de enlace específicas. Los mensajes enviados desde dispositivos finales son transferidos a través de todas las puertas de enlace dentro del alcance. Estos mensajes son recibidos por el Servidor de Red. En caso de que un Servidor de Red reciba varias copias de un mismo mensaje, conserva una única copia del mensaje y descarta otras [16].

3.2.3 Parámetros Regionales

LoRaWAN opera en un espectro de radio sin licencia, de manera similar a la tecnología Wi-Fi, que utiliza las bandas ISM de 2.4 GHz y 5 GHz en todo el mundo [20]. El protocolo LoRaWAN utiliza frecuencias de radio más bajas con un alcance mayor. El hecho de que las frecuencias tengan un mayor alcance también conlleva más restricciones, que a menudo son específicas de cada país. Como resultado, LoRaWAN se especifica para diferentes bandas en estas regiones. Estas bandas son lo suficientemente similares como para admitir un protocolo independiente de la región, pero tienen una serie de consecuencias para la implementación de los sistemas *backend* [20].

- LoRaWAN presenta parámetros regionales oficiales que únicamente dan cobertura a una región. Por ejemplo, los parámetros regionales de LoRaWAN para Asia especifican únicamente un subconjunto común de canales, pero existen

variaciones entre las regulaciones de los países asiáticos. Además, cada operador de Servidor de Red es libre de seleccionar parámetros complementarios, como canales de emisión adicionales [20].

- En algunos países, se puede utilizar más de un plan de frecuencia. Por ejemplo, en los Países Bajos se puede utilizar tanto EU868-870 como EU433 [20].
- Los parámetros regionales incluyen parámetros de capa física tales como planes de frecuencia (planes de canales), frecuencias de canales obligatorias, y velocidades de datos para mensajes de solicitud de incorporación. Los parámetros regionales también incluyen parámetros de la capa LoRaWAN, como el tamaño máximo de carga útil [20].

3.2.3.1 Planes de frecuencia común

LoRaWAN opera en las bandas ISM sin licencia. En la Tabla 4 se enumeran los planes de frecuencia y denominaciones más recientes [20].

Tabla 4 Planes de frecuencia y sus denominaciones [20].

Plan de canales	Denominación
UE863-870	UE868
US902-928	US915
CN779-787	CN779
UE433	UE433
AU915-928	AU915
CN470-510	CN470
AS923	AS923
KR920-923	KR920
IN865-867	IN865
RU864-870	RU864

Banda EU863-870

La banda EU863-870 se puede aplicar a cualquier región donde el uso del espectro de radio esté definido por el estándar ETSI [EN300.220]. La banda EU863-870 se utiliza en todos los países europeos, y algunos países fuera de Europa, como por ejemplo Bahrein (BH), ubicado en Oriente Medio. La banda EU863-870 implica que la banda de frecuencia oscila entre 863 MHz y 870 MHz, pero algunos países utilizan rangos de frecuencia ligeramente diferentes. Por ejemplo, Albania (AL) utiliza 863-873 MHz [20].

3.2.4 Clases de dispositivos

La especificación del protocolo LoRaWAN presenta una clasificación de dispositivos en tres categorías distintas, conocidas como Clase A, Clase B y Clase C. Es relevante destacar que la implementación de la Clase A es un requisito esencial para todos los dispositivos LoRaWAN, mientras que las Clases B y C se perfilan como extensiones específicas de la especificación destinadas a los dispositivos pertenecientes a la Clase A. Todas las clases de dispositivos proporcionan el soporte necesario para establecer comunicaciones bidireccionales, lo que implica la capacidad de transmitir tanto en el enlace ascendente como en el enlace descendente. En el contexto de las actualizaciones de firmware inalámbricas, comúnmente denominadas FUOTA (*Firmware Update Over the Air*), se hace imperativo que un dispositivo efectúe la transición hacia la Clase B o Clase C. Sin embargo, es esencial enfatizar que, durante dicho proceso, los dispositivos finales se encuentran inhabilitados para transmitir mensajes en el enlace ascendente mientras se hallen en proceso de recepción de mensajes en el enlace descendente [21].

3.2.4.1 Clase A

Todos los dispositivos finales LoRaWAN deben admitir la implementación de Clase A. Un dispositivo Clase A puede enviar un mensaje de enlace ascendente en cualquier momento. Una vez que se completa la transmisión de enlace ascendente, el dispositivo abre dos ventanas de recepción breves para recibir mensajes de enlace descendente de la

red. Existe un retardo entre el final de la transmisión del enlace ascendente y el inicio de cada ventana de recepción, denominadas RX1 y RX2, respectivamente, en la Figura 12. En caso de que el Servidor de Red no responda durante estas dos ventanas de recepción, el siguiente enlace descendente se programará inmediatamente después de la siguiente transmisión del enlace ascendente [21].

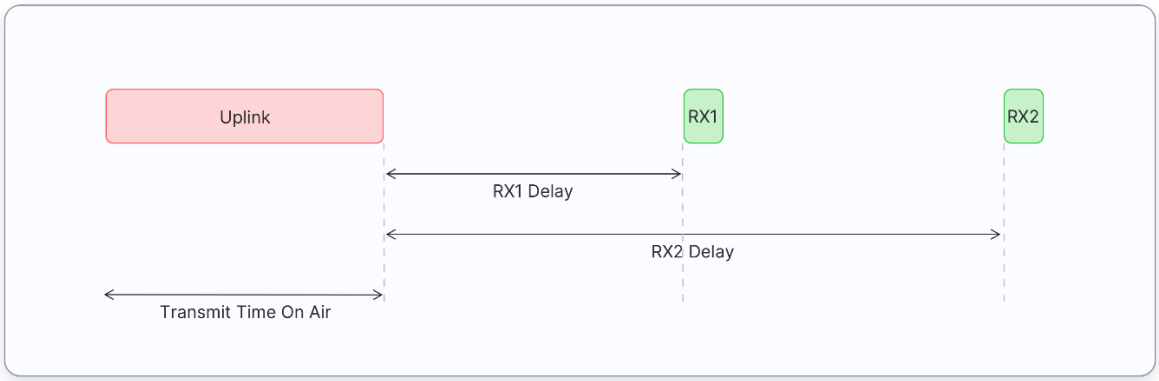


Figura 12 Ejemplo de retrasos en enlace clase A [21].

El Servidor de Red puede responder durante la primera ventana de recepción (RX1) o la segunda ventana de recepción (RX2), pero no utilizar ambas ventanas [21]. Se consideran tres situaciones para mensajes de enlace descendente, como se muestra en la Figura 13.

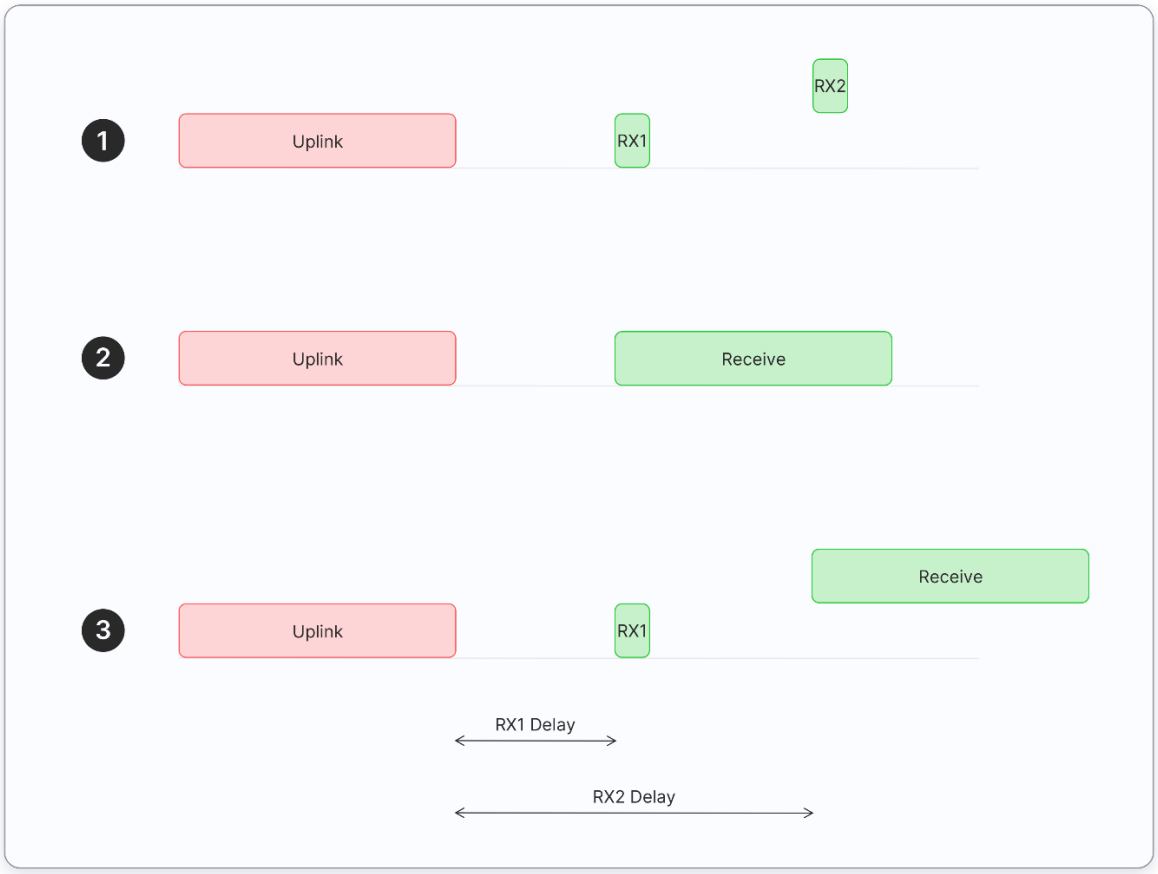


Figura 13 Posibles situaciones de un enlace clase A [21].

1. El dispositivo final abre ambas ventanas de recepción, pero no recibe un mensaje de enlace descendente durante ninguna de las ventanas de recepción.
2. El dispositivo final recibe un enlace descendente durante la primera ventana de recepción y, por lo tanto, no abre la segunda ventana de recepción.
3. El dispositivo final abre la primera ventana de recepción, pero no recibe un enlace descendente. Por lo tanto, abre la segunda ventana de recepción y recibe un enlace descendente durante la segunda ventana de recepción.

Los dispositivos finales de Clase A presentan un consumo de energía muy bajo. Por lo tanto, pueden operar con batería, al pasar la mayor parte del tiempo en modo de suspensión con intervalos de tiempo prolongados entre enlaces ascendentes. Además, los dispositivos de Clase A tienen una elevada latencia de enlace descendente, ya que requieren enviar un enlace ascendente para recibir un enlace descendente [21].

3.2.4.2 Clase B

Los dispositivos Clase B amplían las capacidades de la Clase A al abrir periódicamente ventanas de recepción denominadas *ping slots* para recibir mensajes de enlace descendente. La red transmite periódicamente una baliza sincronizada en el tiempo (unidifusión y multidifusión) a través de las puertas de enlace, que es recibida por los dispositivos finales. Estas balizas proporcionan una referencia de sincronización para los dispositivos finales, permitiéndoles alinear sus relojes internos con la red. Esto permite que el Servidor de Red sepa cuándo enviar un enlace descendente a un dispositivo específico o a un grupo de dispositivos [21].

Tras un enlace ascendente, las dos ventanas de recepción RX1 y RX2 se abrirán de manera similar a los dispositivos de Clase A, como se muestra en la Figura 14 [21].

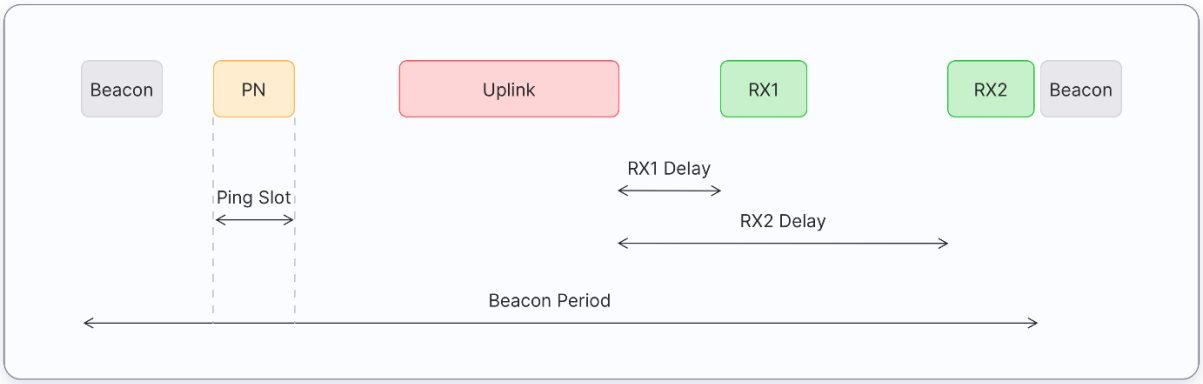


Figura 14 Ejemplo de enlace clase B [21].

Los dispositivos finales Clase B presentan una reducida latencia para los enlaces descendentes en comparación con los dispositivos finales de Clase A. Los dispositivos Clase B suelen operar también con baterías, si bien su duración es menor en comparación con los de Clase A, puesto que los dispositivos pasan más tiempo en modo activo. Debido a la baja latencia de los enlaces descendentes, el modo Clase B se puede utilizar en dispositivos que requieren una actuación crítica de nivel medio, como los medidores de servicios públicos [21]. Los dispositivos Clase B también pueden funcionar en modo Clase A.

3.2.4.3 Clase C

Los dispositivos de Clase C amplían las capacidades de la Clase A manteniendo abiertas las ventanas de recepción a menos que transmitan un enlace ascendente, como se muestra en la Figura 15. Por lo tanto, los dispositivos de Clase C pueden recibir mensajes de enlace descendente en casi cualquier momento, por lo que presentan una latencia muy baja para los enlaces descendentes [21].

Los dispositivos de Clase C abren dos ventanas de recepción, RX1 y RX2, de forma similar a los de Clase A. Sin embargo, la ventana de recepción RX2 permanece abierta hasta la siguiente transmisión de enlace ascendente. Después de que el dispositivo envíe un enlace ascendente, se abre una ventana de recepción RX2, seguida de una ventana de recepción RX1, abriéndose finalmente la ventana de recepción continua de RX2. Esta ventana de recepción RX2 permanece abierta hasta que se programe el siguiente enlace ascendente. Los enlaces ascendentes se envían cuando no hay ningún enlace descendente en curso [21].

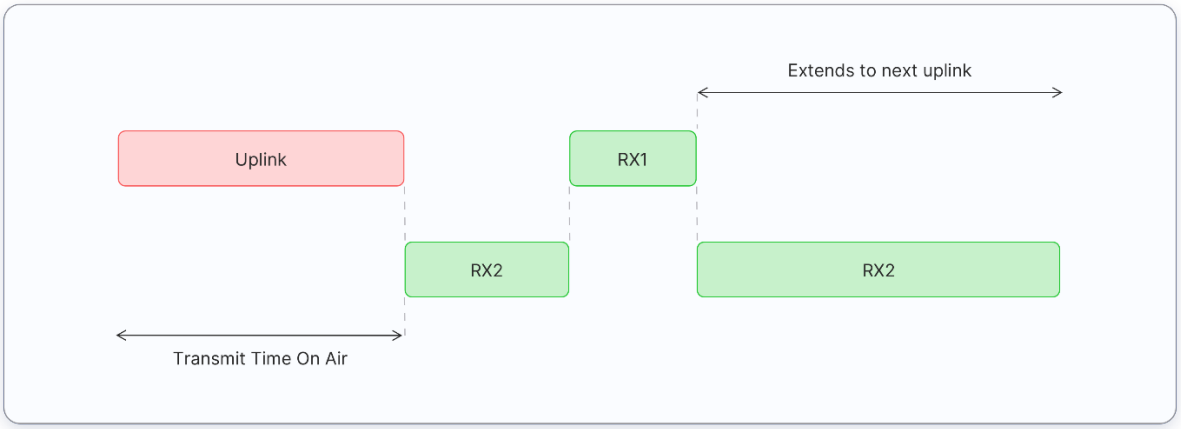


Figura 15 Ejemplo enlace Clase C [21].

En comparación con los dispositivos de Clase A y Clase B, los dispositivos de Clase C presentan una menor latencia. Sin embargo, consumen más energía debido a la necesidad de abrir intervalos de recepción continua. Como resultado, estos dispositivos no pueden operar por lo general con baterías durante largos periodos de tiempo, por lo que suelen

funcionar con alimentación eléctrica [21]. Los dispositivos Clase C también pueden funcionar en modo Clase A.

3.2.5 Métodos de activación de los dispositivos finales

Cada dispositivo final debe estar registrado en una red LoRaWAN antes de enviar y recibir mensajes. Este procedimiento se conoce como activación. Existen dos métodos de activación disponibles:

- *Over-The-Air-Activation* (OTAA): es el método de activación más seguro y recomendado para los dispositivos finales. Los dispositivos realizan un procedimiento de unión con la red, durante el cual se asigna una dirección de dispositivo dinámica y se negocian las claves de seguridad con el dispositivo [22].
- *Activation By Personalization* (ABP): se requiere codificar manualmente la dirección del dispositivo y las claves de seguridad en el dispositivo. ABP es menos seguro que OTAA y también tiene la desventaja de que los dispositivos no pueden cambiar de proveedor de red sin modificar manualmente las claves en el dispositivo [22].

Ambos métodos, OTAA y ABP, generan claves de sesión secretas tanto en el dispositivo final como en el Servidor de Red. Estas claves se utilizan para proteger los mensajes enviados y recibidos a través de la red. En el caso del método ABP, las claves de sesión se almacenan en el dispositivo final durante el proceso de fabricación. Por otro lado, en OTAA, las claves de sesión se generan durante una serie de intercambios entre el dispositivo y el Servidor de Red [23].

El proceso de activación establece tres claves en el dispositivo, denominadas *Network Session Key* (NwkSKey), *Application Session Key* (AppSKey) y *Device Address* (DevAddr). Estos se generan y almacenan cuando el dispositivo intenta conectarse por primera vez a la red a través del método de activación OTAA, o se almacenan directamente en el dispositivo en el momento de la fabricación para dispositivos que implementan el método de activación ABP [23].

- ***Network Session Key* (NwkSKey):** tanto el Servidor de Red como el dispositivo final utilizan la clave (NwkSKey) para verificar la integridad de todos los datos.
- ***Application Session Key* (AppSKey):** el Servidor de Aplicaciones y el dispositivo final utilizan la clave (AppSKey) para cifrar y descifrar el *Frame Payload* (FRMPayload) de las tramas de datos específicas de la aplicación.
- ***Device Address* (DevAddr)**: la dirección del dispositivo (DevAddr) es un identificador de 32 bits asignado por el Servidor de Red. DevAddr se utiliza junto con NwkSKey para identificar el dispositivo en la red actual.

3.2.5.1 OTAA

En LoRaWAN 1.0.x y 1.1.x, el método de activación inalámbrica OTAA requiere el intercambio de dos mensajes MAC entre el dispositivo final y el servidor: una solicitud de unión enviada desde el dispositivo final al servidor, y una aceptación de unión enviada desde el servidor al dispositivo final. En ambas versiones, antes de la activación, los parámetros DevEUI y AppKey deben estar almacenados en el dispositivo final. La AppKey es una clave secreta AES de 128 bits conocida como clave raíz. En LoRaWAN 1.1.x, también es necesario almacenar JoinEUI y NwkKey, siendo estas últimas también claves secretas AES de 128 bits. Estas claves, junto con el DevEUI coincidente, deben estar aprovisionadas en el servidor de unión para facilitar el procesamiento del procedimiento de unión y la derivación de la clave de sesión. Los identificadores JoinEUI y DevEUI no son secretos y son visibles para todos. Es importante destacar que las claves AppKey y NwkKey nunca se envían a través de la red [22]. En la Figura 16 se muestra el procedimiento de activación OTAA para LoRaWAN 1.0.x, y en la Figura 17 para LoRaWAN 1.1.x.

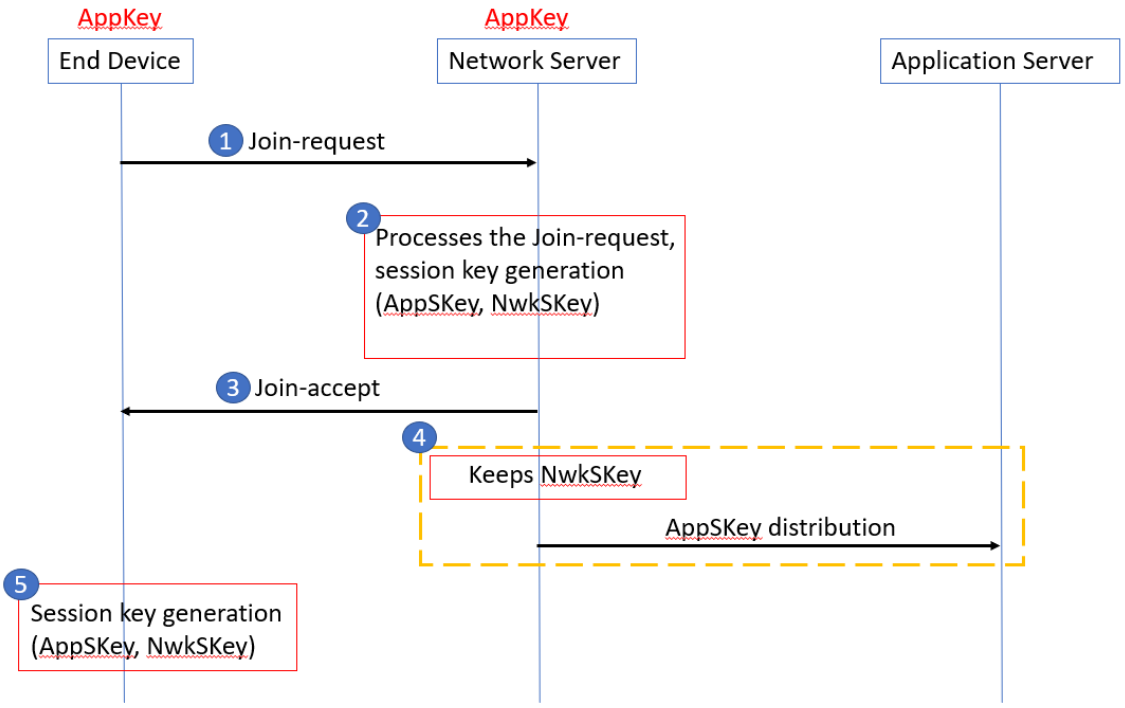


Figura 16 Flujo de mensajes OTAA en LoRaWAN 1.0.X [22].

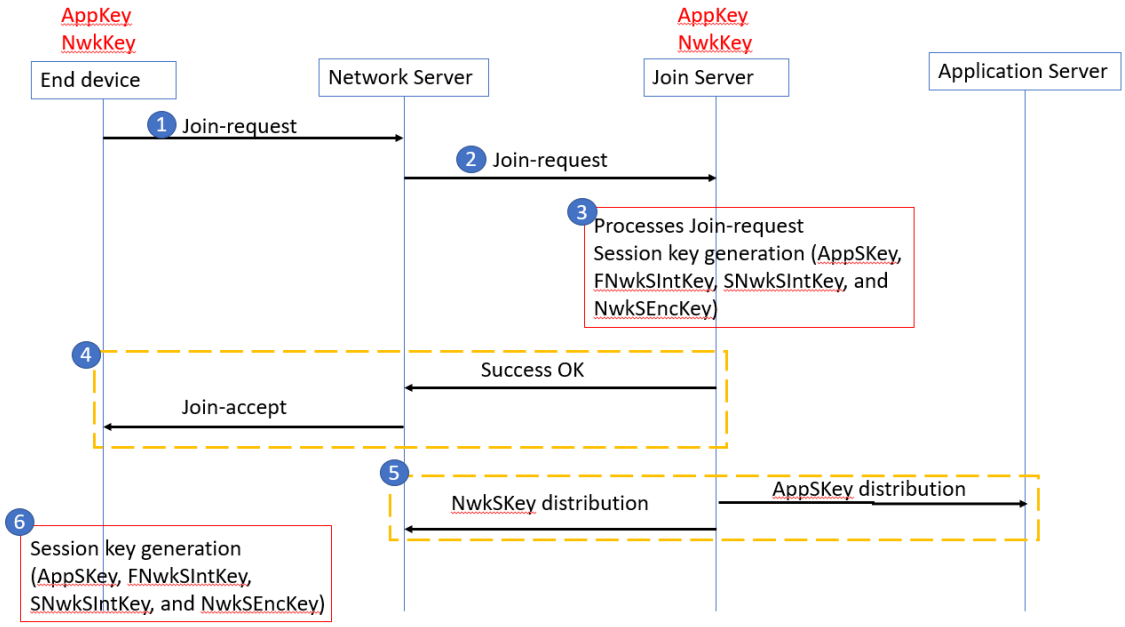


Figura 17 Flujo de mensajes OTAA en LoRaWAN 1.1.X [22].

3.2.5.2 ABP

El método de activación ABP vincula directamente un dispositivo final a una red preseleccionada, sin pasar por el procedimiento de activación inalámbrica. El método de

activación ABP es menos seguro y también tiene la desventaja de que los dispositivos no pueden cambiar de proveedor de red sin modificar manualmente las claves en el dispositivo. Un dispositivo final activado mediante el método ABP solo puede funcionar con una única red y mantiene la misma sesión de seguridad durante toda su vida [22].

En LoRaWAN, el método de activación ABP requiere almacenar directamente en el dispositivo final el DevAddr (Device Address) y las claves de sesión. En LoRaWAN 1.0.x, estas claves son NwkSKey (*Network Session Key*) y AppSKey (*Application Session Key*), mientras que en LoRaWAN 1.1, son *Forwarding Network Session Integrity Key* (FNwkSIntKey), *Serving Network Session Integrity Key* (SNwkSIntKey), *Network Session Encryption Key* (NwkSEncKey) y AppSKey. Cada dispositivo final debe poseer un conjunto único de estas claves de sesión. Paralelamente, DevAddr y las claves de sesión correspondientes deben estar registradas en el Servidor de Red, mientras que la AppSKey debe estar registrada en el servidor de aplicaciones [22]. En la Figura 18 y la Figura 19 se presentan los diagramas de flujo que ilustran el método de activación utilizando ABP.

Figura 18 Flujo de mensajes en ABP versión LoRaWAN 1.0.X [22].

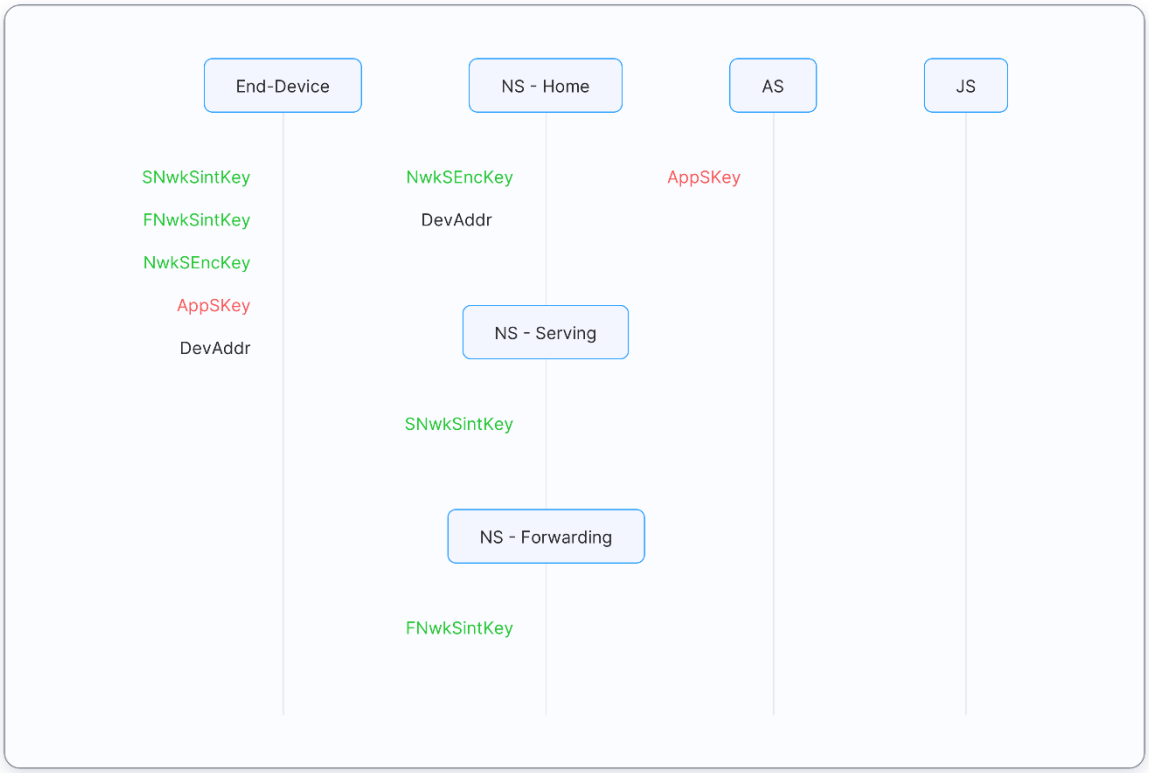


Figura 19 Flujo de mensajes en ABP versión LoRaWAN 1.1.X [22].

3.2.5.3 OTAA vs ABP

Para optimizar la eficiencia y la seguridad en la activación y gestión de dispositivos en redes LoRaWAN, se recomienda evitar la implementación de ABP en la medida de lo posible. En los casos en que su uso sea inevitable, es imperativo garantizar la disponibilidad simultánea de OTAA. Además, resulta fundamental emplear el identificador único de dispositivo (DevEUI) proporcionado por el fabricante del módulo de radio, o bien, adquirir un DevEUI válido y globalmente único [24].

En el escenario de utilizar un Servidor de Unión independiente, se debe utilizar el JoinEUI proporcionado por dicho servidor. En caso contrario, se debe designar un único identificador único de entidad (EUI) como JoinEUI para todos los dispositivos. Respecto a la generación de la clave de aplicación (AppKey), se enfatiza la importancia de seguir las directrices para una generación segura de claves raíz, seleccionando un enfoque que mitigue los ataques de retransmisión de aceptación de unión [24].

Para asegurar que un ciclo de encendido no requiera la reintegración del dispositivo, es esencial almacenar la AppSKey, la NwkSKey, el último DevNonce utilizado, y cualquier valor de JoinNonce en una memoria no volátil (NVM). En el caso específico de utilizar ABP, se debe almacenar también el contador de tramas ascendentes (FCntUp) en la NVM [24].

3.3 Propuesta de soluciones para la aplicación IoT basada en LoRaWAN

Dentro del marco del desarrollo del presente Trabajo Fin de Máster, se proponen dos soluciones para la implementación de una aplicación IoT basada en el protocolo LoRaWAN que permita realizar el análisis de patrones de comportamiento relacionados con la detección de personas a partir de una serie de nodos con el objetivo de acceder a esta información, almacenarla, procesarla y visualizarla de manera eficiente. Cada propuesta aborda consideraciones específicas, desde la escalabilidad y eficiencia energética hasta la privacidad y seguridad de los datos que permitirán discernir cuál de los escenarios se ajusta a los requisitos particulares demandados.

A continuación, se presenta un análisis de cada propuesta, indicando las ventajas y argumentos que respaldan su viabilidad en entornos de aplicación específicos, así como los recursos asociados a la implementación de cada solución.

En todos los casos, para la detección de personas se contará con el módulo *Person Sensor* de la empresa *Useful Sensors*, que se integrará con el dispositivo Arduino MKRWAN 1310 mediante el protocolo I_2C (*Inter-Integrated Circuit*) para la implementación de los nodos finales IoT. La función del dispositivo Arduino MKRWAN 1310 consistirá básicamente en procesar los datos proporcionados por el/los módulo/s *Person Sensor*, y procesarlos para su transmisión a una plataforma IoT en una red LoRaWAN.

En la primera solución, representada esquemáticamente en la Figura 20, se plantea la interconexión, mediante la tecnología LoRa®, del dispositivo Arduino MKRWAN 1310 que forma parte del nodo final IoT, con las puertas de enlace LoRaWAN correspondientes a los modelos HT-M00 y RAK7268, de las empresas *Heltec* y *RAK Wireless*, respectivamente. Estas puertas de enlace, por su parte, se registrarán en la plataforma TTN (*The Things Network*), desde la que se reenviará la información recibida, haciendo uso del protocolo

MQTT (*Message Queuing Telemetry Transport*), hacia la base de datos InfluxDB para su almacenamiento y su posterior análisis y visualización en la herramienta Grafana. La elección de LoRa® se justifica por su bajo consumo de energía, mientras que TTN proporciona una red global escalable, e InfluxDB destaca por su eficiencia en el almacenamiento de series temporales de datos.

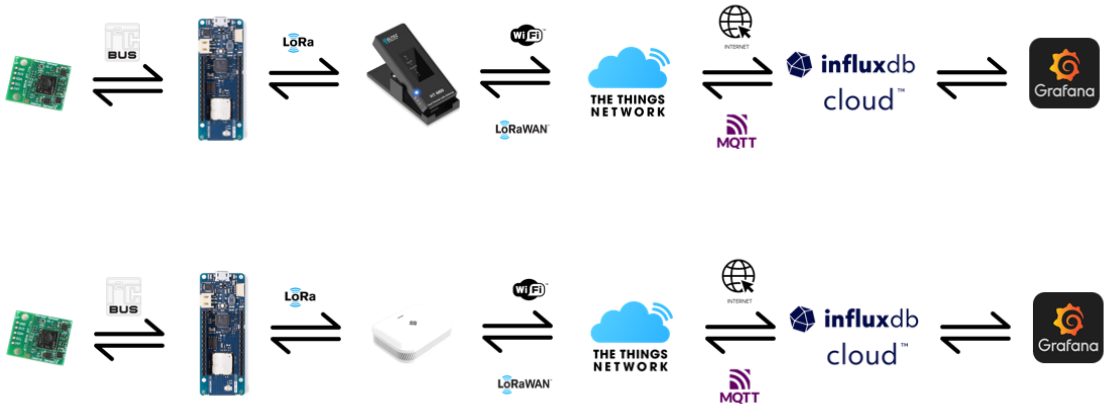


Figura 20 Arquitecturas genéricas basadas en los Gateways LoRaWAN HT-M00 y RAK7268, respectivamente.

En la segunda propuesta, representada esquemáticamente en la Figura 21, se contempla la conexión del dispositivo Arduino MKRWAN 1310 con el *Gateway* RAK7268, pero asumiendo en este caso el rol de *Network Server* de la red LoRaWAN, facilitando la comunicación con la base de datos InfluxDB mediante un *broker* MQTT integrado, y visualizando los datos de igual forma en la herramienta Grafana. La posibilidad de configurar el *Gateway* RAK7268 como *Network Server* proporciona versatilidad y flexibilidad al sistema, facilitando la integración entre el dispositivo y la plataforma de almacenamiento. Además, el uso del protocolo MQTT simplifica tanto la configuración como el intercambio de datos, lo que contribuye significativamente a la viabilidad y eficacia de la propuesta en entornos de IoT.

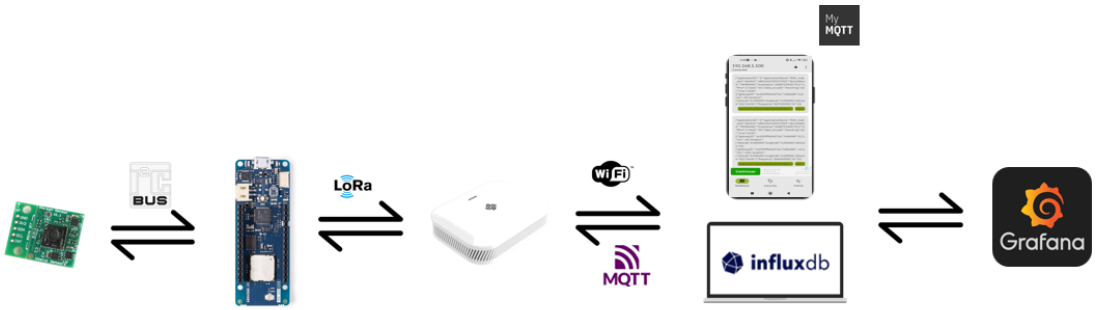


Figura 21 Arquitectura Built-In MQTT Broker.

Cada propuesta aborda diversos aspectos, desde la tecnología de comunicación hasta la gestión de datos, presentándose opciones adaptables a los requisitos específicos que se requieran. En capítulos subsiguientes, se introducirán los dispositivos, plataformas, protocolos y herramientas de comunicación que se usan en las soluciones propuestas. Este enfoque permitirá alcanzar conclusiones más concretas y precisas, además de realizar una comparativa detallada, exponiendo los beneficios y desafíos inherentes a cada una de las arquitecturas planteadas.

4 Plataforma HW/SW del nodo IoT

En el presente Trabajo Fin de Máster se ha seleccionado el dispositivo MKRWAN 1310 de Arduino como elemento principal para el desarrollo y la implementación del nodo final en las diferentes soluciones propuestas para la aplicación de IoT basada en el protocolo LoRaWAN establecida como objetivo. Esta elección se fundamenta en diversas razones, entre las que destaca su versatilidad, su capacidad de comunicación en redes LoRaWAN, y el soporte para el desarrollo de aplicaciones en el ámbito de IoT. Por otro lado, con el fin de poder realizar el análisis de patrones de comportamiento relacionados con la detección de personas, en la implementación del nodo IoT se integrará el módulo *Person Sensor* con el dispositivo Arduino MKRWAN 1310 mediante una interfaz I_2C, como se representa en la Figura 22.

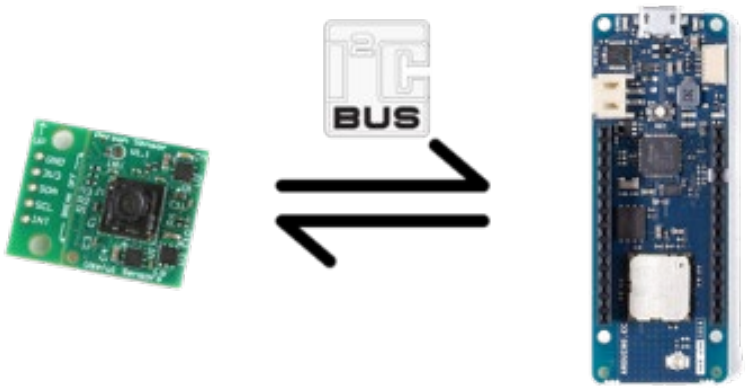


Figura 22 Elementos de la plataforma HW del nodo IoT.

4.1 Dispositivo Arduino MKRWAN 1310

El dispositivo Arduino MKRWAN 1310, mostrado en la Figura 23, es un módulo HW de reducido tamaño que integra un procesador Cortex M0+ SAMD21, un módulo Murata CMWX1ZZABZ LoRa®, un chip criptográfico (ECC508) y una memoria flash SPI de 2 Mbytes [25].

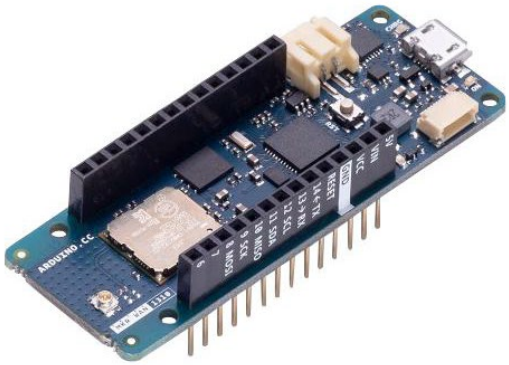


Figura 23 Arduino MKRWAN 1310 [26].

El dispositivo Arduino MKRWAN 1310 soporta comunicación inalámbrica a través de LoRa® y el protocolo LoRaWAN. En la Tabla 5 se proporciona una visión estructurada y detallada de las especificaciones del dispositivo Arduino MKRWAN 1310, lo cual resulta fundamental para comprender de manera rápida y precisa las prestaciones que proporciona este dispositivo.

Tabla 5 Características funcionales del Módulo Arduino MKRWAN 1310 [26].

Board	**Name**	**Arduino® MKRWAN 1310**
	SKU	ABX00029
	Compatibility	MKR
Microcontroller	SAMD21 Cortex®-M0+ 32bit low power ARM MCU	
USB connector	Micro USB (USB-B)	
Pins	**Built-in LED Pin**	6
	Digital I/O Pins	8
	Analog Input Pins	7 (ADC 8/10/12 bit)
	Analog Output Pins	1 (DAC 10 bit)
	PMW Pins	13 (0 - 8, 10, 12, A3, A4)
	External interrupts	10 (0, 1, 4, 5, 6, 7, 8 ,9, A1, A2)
Connectivity	**LoRa**	Murata CMWX1ZZABZ
	Carrier frequency	433/868/915 MHz
	Region	EU/US

	Secure element	ATECC508A
Communication	**UART**	Yes
	I2C	Yes
	SPI	Yes
Power	**I/O Voltage**	3.3V
	Input Voltage (nominal)	5-7V
	DC Current per I/O pin	7 mA
	Supported battery	Li-Po Single Cell, 3.7V, 1024mAh Minimum
	Battery connector	JST PH
	Lowest power consumption	104uA
Clock speed	**Processor**	48 MHz
	RTC	32.768 kHz
Memory	**SAMD21G18A**	256KB Flash, 32KB SRAM
	Murata CMWX1ZZABZ	192KB Flash, 20KB RAM
Dimensions	**Weight**	32 g
	Width	25 mm
	Length	67.6 mm

Es importante tener en cuenta que, de acuerdo con estas especificaciones, el dispositivo Arduino MKRWAN 1310 es compatible únicamente con señales de E/S de 3.3V.

4.1.1 Descripción General

4.1.1.1 Microcontrolador

El microcontrolador integrado en el dispositivo Arduino MKRWAN 1310 es un Cortex M0+ de ARM que opera a una frecuencia de hasta 48MHz. La mayoría de sus pines están conectados a los terminales externos, si bien algunos están reservados para la comunicación interna con el módulo de comunicación LoRa® y con los periféricos internos SPI e I_2C (memoria flash SPI y criptografía) [25].

4.1.1.2 Módulo LoRa

El módulo Murata CMWX1ZZABZ proporciona conectividad LoRa®. Este módulo contiene un procesador STM32L0 junto con una radio Semtech SX1276. El procesador funciona con un *firmware* de código abierto de Arduino basado en el código de Semtech [25].

La comunicación del módulo LoRa® Murata CMWX1ZZABZ con el microcontrolador Cortex M0+ se realiza a través de las interfaces I_2C y SPI, utilizando los pines que se muestran en la Tabla 6 [25].

Tabla 6 Pines del Procesador Cortex M0+ [25].

SAMD21 Pin	SAMD21 Acronym	NINA Pin	CMWX1ZZABZ Acronym	Description
39	PA27	34	MCU_NRST	Reset
41	PA28	22	GPIO0	IRQ pin
8	PB09	43	boot_0	Boot
23	PA14	17	GPIO5	SPI CS
22	PA13	16	GPIO18	SPI CLK
24	PA15	15	GPIO21	SPI MISO
21	PA12	14	GPIO12	SPI MOSI
14	PA08	36	PB9/SDA1	I2C SDA
13	PA09	35	PB8/SCL1	I2C SCL

Este transmisor de radio [IC:26792-ABX00029] ha sido aprobado para operar con los tipos de antena que se enumeran en la Tabla 7, con la ganancia máxima permitida indicada [25].

Tabla 7 Tipos de antena que soporta el módulo LoRa [25].

Antenna Manufacturer:	**Dynaflex srl**
Antenna Model:	505012UFL
Antenna type:	External omnidirectional dipole antenna
Antenna gain:	-1dBi

4.1.2 Módulo Crypto (ECC508)

El chip criptográfico integrado en el dispositivo Arduino MKRWAN 1310 proporciona una forma segura de almacenar certificados, además de permitir la aceleración de protocolos seguros [25].

4.1.3 Módulo memoria flash (W25Q16)

La memoria flash SPI externa puede proporcionar hasta 16Mb de almacenamiento en el dispositivo Arduino MKRWAN 1310, y se puede utilizar como un periférico SPI para el almacenamiento de datos [25].

4.1.4 Rangos de trabajo

En la Tabla 8, la Tabla 9, la Tabla 10 y la Tabla 11 se detallan los rangos de funcionamiento del dispositivo Arduino MKRWAN 1310 en términos de niveles de tensión admitidos, temperaturas tolerables, condiciones de operación, y niveles de consumo de energía. Estos datos son esenciales para garantizar un desempeño óptimo y una utilización eficiente de este módulo en diversas aplicaciones, proporcionando así una visión completa de sus capacidades y restricciones en una variedad de situaciones y entornos.

Al igual que la mayoría de las placas que presentan el factor de forma Arduino MKR, el dispositivo MKRWAN 1310 puede alimentarse mediante un conector USB, a través de los pines de E/S o conectando una batería de litio o polímero de litio al cargador de batería incorporado (BQ24195L) [25].

El cargador de batería del módulo Arduino MKRWAN 1310 tiene una corriente mínima de carga de 512 mA. Se debe asegurar que esta corriente de carga sea compatible con la batería que se vaya a utilizar [25].

Tabla 8 Voltajes [25].

Símbolo	Descripción	Min	Max	Unidad
VIN $_{Max}$	Máximo voltaje de entrada de VIN	-0.3	5.5	V
VUSB $_{Max}$	Máximo voltaje de	-0.3	5.5	V

	entrada del conector USB		
P_{Max}	Máximo consumo de potencia	TBC	mW

Tabla 9 Temperaturas [25].

Símbolo	Descripción	Min	Max	Unidad
T_{ST}	Temperatura de almacenamiento	-40	85	°C
T_{OP}	Temperatura de trabajo	-40	85	°C

Tabla 10 Condiciones óptimas de trabajo [25].

Símbolo	Descripción	Min	Typ	Max	Unidad
V_{IN}	Voltaje de entrada V_{IN}	4.5	5	5.5	V
V_{USB}	Voltaje de entrada del conector USB	4.8	5	5.5	V
V_{3V3}	Salida de Voltaje del Pin 3V3		3.3		V
I_{3V3}	Corriente disponible del pin 3V3			300	mA
V_{IH}	Input high-level voltage	1.815	-	-	V
V_{IL}	Input low-level voltage	-	-	0.99	V
V_{OH}	Output high-level voltage @I OH Max, PORT.PINCFG.DRVSTR=1	-	-	7	mA
V_{OL}	Output low-level voltage @I OL Max, PORT.PINCFG.DRVSTR=1	-	-	10	mA

Tabla 11 Consumo de potencia [25].

Símbolo	Descripción	Min	Typ	Max	Unit
P_{BL}	Potencia de consumo en modo de trabajo continuo		TBC		mW
P_{LP}	Potencia de consumo en modo de bajo consumo		TBC		mW
P_{MAX}	Máximo consumo de Potencia		TBC		mW

4.1.5 Conexionado de Pines

4.1.5.1 Headers

El dispositivo Arduino MKRWAN 1310 incluye dos conectores de 28 pines. En la Tabla 12 se presenta una detallada información que incluye el identificador de cada pin, su función, el tipo de conexión, y una breve descripción de sus características particulares.

Tabla 12 Pines del Arduino MKRWAN 1310 [25].

Pin	Function	Type	Description
1	AREF	Analog	Analog Reference.
2	A0/DAC0	Analog	ADC in/DAC out, can be used as GPIO
3	A1	Analog	ADC in, can be used as GPIO
4	A2	Analog	ADC in, can be used as GPIO
5	A3	Analog	ADC in, can be used as GPIO
6	A4/SDA	Analog	ADC in, I2C SDA, can be used as GPIO
7	A5/SCL	Analog	ADC in, I2C SCL, can be used as GPIO
8	A6	Analog	ADC in, can be used as GPIO
9	D0	Digital	GPIO, can be used as PWM
10	D1	Digital	GPIO, can be used as PWM
11	D2/PWM	Digital	GPIO, can be used as PWM
12	D3/PWM	Digital	GPIO, can be used as PWM
13	D4/PWM	Digital	GPIO, can be used as PWM
14	D5/PWM	Digital	GPIO, can be used as PWM
15	D6	Digital	GPIO, can be used as PWM
16	D7	Digital	GPIO can be used as PWM
17	D8/MOSI	Digital	SPI MOSI, can be used as GPIO, can be used as PWM
18	D9/SCK	Digital	SPI SCK, can be used

			as GPIO, can be used as PWM
19	D10/MISO	Digital	SPI MISO, can be used as GPIO
20	D11/SDA	Digital	I2C SDA, can be used as GPIO
21	D12/SCL	Digital	I2C SCL, can be used as GPIO
22	D13/RX	Digital	USART RX, can be used as GPIO
23	D14/TX	Digital	USART TX, can be used as GPIO
24	RESETN	Digital	Reset input
25	GND	Power	Power Ground
26	+3V3	Power Out	
27	VIN	Power In	Vin Power input
28	+5V	Power Out	Power Ground

Es importante destacar que el pin +5V no suministra tensión por defecto, pudiendo conectarse a través de un puente ubicado cerca del pin V_{USB}, a la entrada de alimentación USB [25]. En la Figura 24 se muestra la distribución de los pines en el dispositivo Arduino MKRWAN 1310.

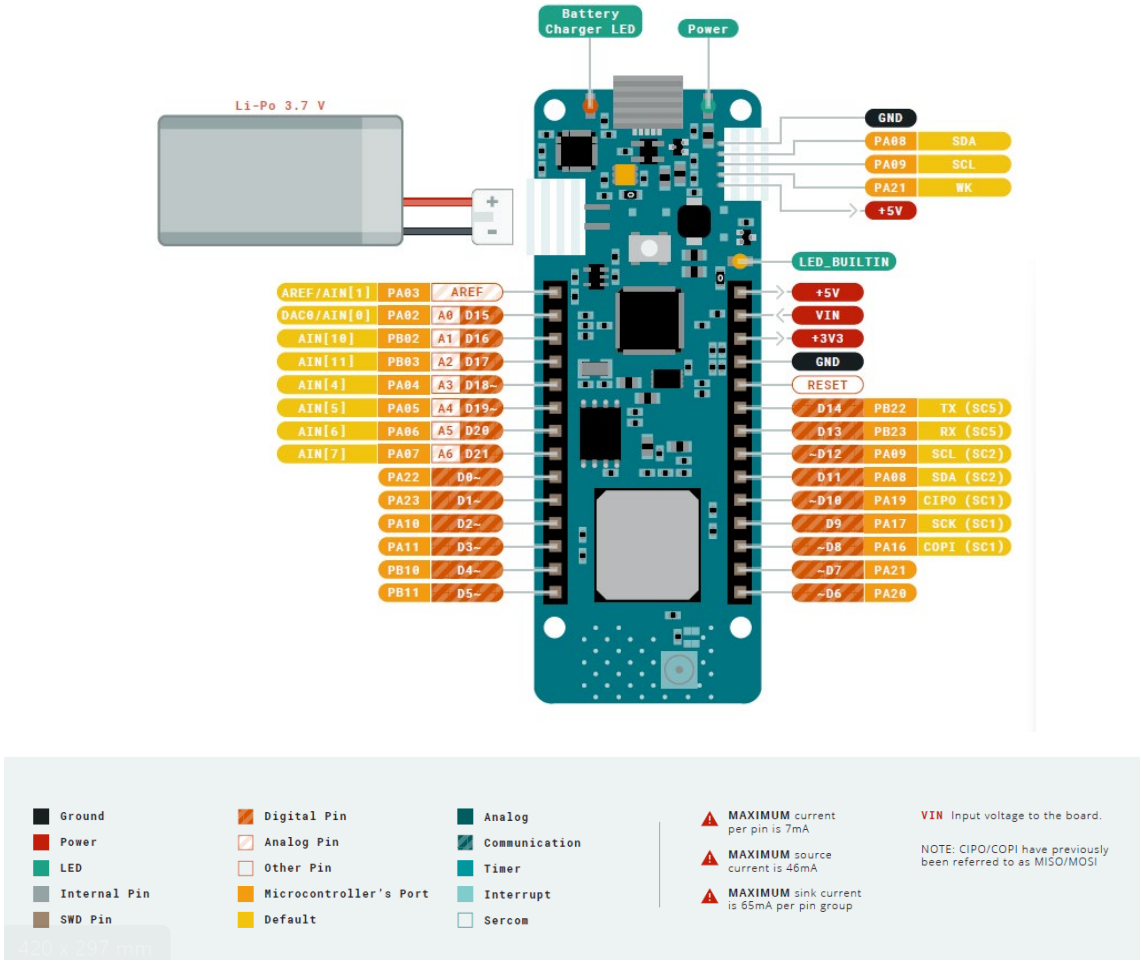


Figura 24 Arduino MKRWAN 1310 [26].

Finalmente, en la Figura 25 se representan los diferentes módulos que componen el dispositivo Arduino MKRWAN 1310.

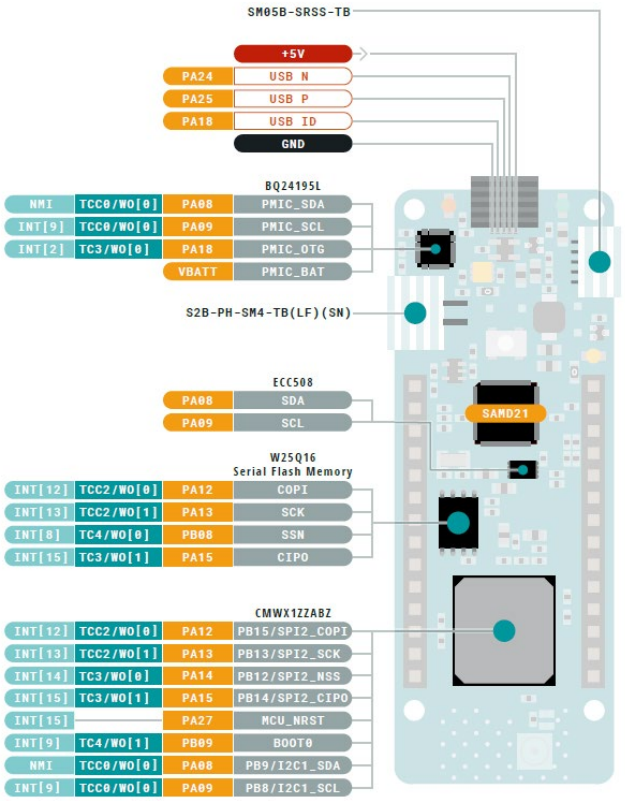


Figura 25 Módulos del Arduino MKRWAN 1310 [26].

4.2 Módulo *Person Sensor*

El módulo *Person Sensor*, desarrollado por la empresa *Useful Sensors*, es un dispositivo HW de bajo coste capaz de detectar rostros de personas cercanas y proporcionar información sobre cuántas personas hay, o dónde se encuentran en relación con el dispositivo, además de poder realizar opcionalmente reconocimiento facial. En la Figura 26 se representa el aspecto del módulo *Person Sensor*.

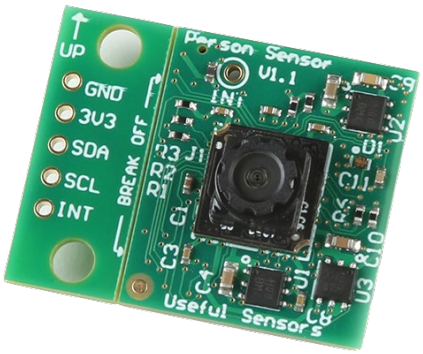


Figura 26 Módulo Person Sensor [27].

Este módulo está diseñado para ser utilizado como entrada en un sistema de mayor entidad, como, por ejemplo, para activar la pantalla de un quiosco cuando alguien se acerque, silenciar un micrófono cuando no hay nadie presente [28], o como en un posible caso de uso del presente TFM, detectar la atención que puede recibir un producto en un comercio.

4.2.1 Características

El módulo *Person Sensor* integra un sensor de imagen y un microcontrolador en una única placa. El sensor de imagen integrado en este módulo presenta un funcionamiento adecuado incluso en condiciones de poca luz -aunque no se dispone de información detallada sobre cómo varía el rendimiento de este módulo con los lúmenes [28], y no incluye un filtro de infrarrojos, por lo que es posible utilizar iluminación de infrarrojos cercanos a partir de un LED en situaciones en la que no se disponga de la luz visible-.

El sensor abarca un campo de visión de aproximadamente 110 grados. El modelo se actualiza aproximadamente siete veces por segundo cuando no está en modo de reconocimiento facial, y alrededor de cinco veces por segundo cuando está habilitado. La latencia es de aproximadamente 200 milisegundos, que es el tiempo que lleva completar una iteración del modelo. Además, la versión del módulo utilizado requiere una alimentación de 3.3V y consume aproximadamente 150 mW de energía, de los cuales alrededor de 5 mW se destinan al LED cuando está en funcionamiento [28].

4.2.2 Conexionado

El módulo *Person Sensor* funciona como cualquier otro sensor, proporcionándole alimentación y obteniendo información a través de una interfaz hardware I_2C estándar con un conector Qwiic [28]. Su diseño hace que este tipo de conector solo se pueda acoplar en una posición, mostrándose en la Figura 27 la distribución del cableado. Si se utiliza un conector estándar, los colores de los cables son amarillo para SCL, azul para SDA, rojo para 3.3V, y negro para GND. También existe un pin de interrupción independiente, denominado TP1, que se puede utilizar para indicar si se han detectado personas. Esto solo es necesario en caso de que se desee despertar el sistema desde el modo de

suspensión para ahorrar energía. De lo contrario, se puede ignorar, ya que únicamente indica que hay información que leer. Muchas aplicaciones pueden simplemente consultar repetidamente el bus I_2C. El sensor admite velocidades de bus I_2C de hasta 400k baudios y está diseñado para ser alimentado con 3.3V [28].

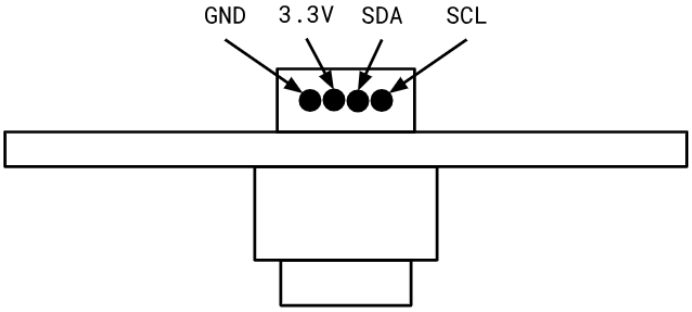


Figura 27 Distribución del cableado del conector Qwiic.

4.2.3 Montaje

Para el montaje del módulo *Person Sensor* es necesario asegurarse de que la lente del sensor tenga un campo de visión claro y se encuentre en la posición correcta, ya que la cámara puede separarse de la placa, al no estar fijada al conector. Para identificar la posición correcta, la persona debe ser capaz de ver el lado de la placa que alberga el sensor, observando la pequeña lente en su centro. De esta manera, se sabrá que se encuentra en la posición adecuada cuando la serigrafía *Useful Sensors Person Sensor V1.0* esté alineada correctamente. En la Figura 28 y la Figura 29 se indican las ubicaciones de la lente, el conector I_2C, y el posicionamiento correcto del sensor en el módulo, respectivamente [23].

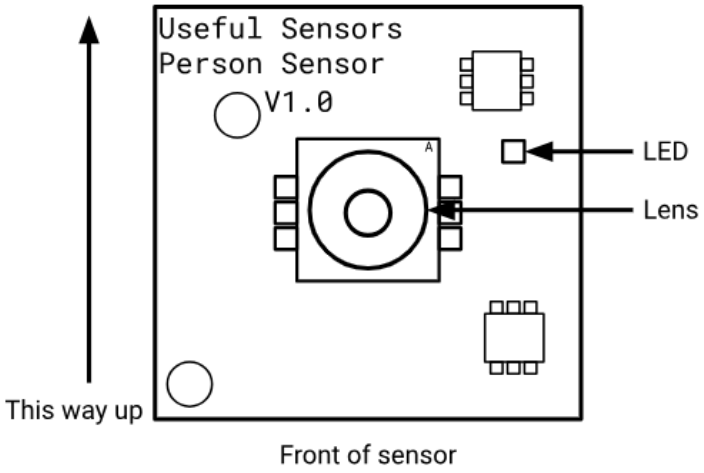


Figura 28 Vista frontal del módulo Person Sensor [28].

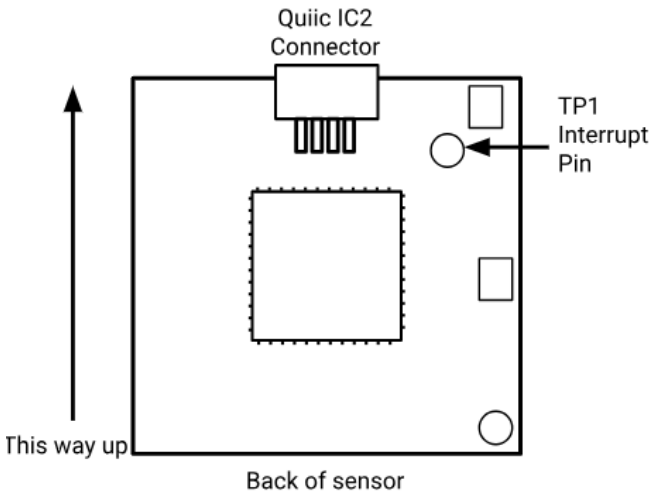


Figura 29 Vista trasera del módulo Person Sensor [28].

Una vez que el módulo *Person Sensor* se encuentre conectado y correctamente montado, se puede comenzar a leer los datos que proporcione en cada momento.

4.2.4 Lectura de datos

El procedimiento de encendido del módulo es el siguiente [28]:

1. Se deben conectar las líneas GND y 3.3V del microcontrolador al módulo *Person Sensor*. Si se utiliza un zócalo Qwiic en el dispositivo basado en MCU, se conectarán automáticamente las líneas SDA y SDC.

2. Con el módulo alimentado, se mueve el sensor para que la cámara apunte hacia el rostro de alguna persona, de manera que, si el módulo *Person Sensor* detecta un rostro, se ilumina el LED verde integrado en la placa.
3. Esto permitirá verificar que el sensor está funcionando correctamente y que está detectando un rostro.

Para el control del módulo *Person Sensor* desde el dispositivo Arduino MKRWAN 1310 es necesario conocer el formato de los datos que proporcionará el módulo. La dirección I_2C del módulo *Person Sensor* es `0x62` (7 bits). Se debe tener en cuenta que en algunos sistemas se incluye el bit de lectura/escritura en la dirección, por lo que en este caso la dirección de 8 bits del módulo *Person Sensor* sería `0xC4` [28].

4.2.5 Formato de los datos

El resultado de las inferencias se almacena en el módulo *Person Sensor* como un vector de bytes, con el formato de los datos que se muestra en la Tabla 13 [28].

Tabla 13 Formato de los Datos del módulo Person Sensor [28].

Byte Offset	**Meaning**
0	Reserved
1	Reserved
2	Data Length (first byte)
3	Data Length (second byte)
4	Number of Faces
5	Face #0 Box Confidence
6	Face #0 Box Left
7	Face #0 Box Top
8	Face #0 Box Right
9	Face #0 Box Bottom
10	Face #0 Recognition Confidence

11	Face #0 Recognition ID
12	Face #0 Is Facing
13	Face #1 Box Confidence
14	Face #1 Box Left
15	Face #1 Box Top
16	Face #1 Box Right
17	Face #1 Box Bottom
18	Face #1 Recognition Confidence
19	Face #1 Recognition ID
20	Face #1 Is Facing
21	Face #2 Box Confidence
22	Face #2 Box Left
23	Face #2 Box Top
24	Face #2 Box Right
25	Face #2 Box Bottom
26	Face #2 Recognition Confidence
27	Face #2 Recognition ID
28	Face #2 Is Facing
29	Face #3 Box Confidence
30	Face #3 Box Left
31	Face #3 Box Top
32	Face #3 Box Right
33	Face #3 Box Bottom
34	Face #3 Recognition Confidence
35	Face #3 Recognition ID
36	Face #3 Is Facing
37	Checksum (first byte)
38	Checksum (second byte)

4.2.5.1 Number of Faces

Este parámetro indica, mediante 1 byte, el número de rostros detectados por el sensor, hasta un máximo de 4.

4.2.5.2 Face Box location

La ubicación del cuadro delimitador alrededor de cada rostro se almacena en un vector mediante 4 bytes. Todos los valores se encuentran en el rango de 0 a 255, y representan la posición o tamaño relativo dentro del campo de visión. Únicamente se proporcionan las coordenadas bidimensionales del rostro dentro de la vista, si bien el tamaño del cuadro delimitador se puede utilizar como una aproximación de la distancia [28].

Dentro del vector de bytes, los valores pueden representar, por ejemplo, la coordenada *x* del punto superior izquierdo del cuadro delimitador, la coordenada *y* del punto superior izquierdo, la anchura y la altura del cuadro delimitador. Estos valores se pueden utilizar para determinar la posición y el tamaño del rostro detectado en relación con el campo de visión, como se muestra en la Figura 30 [28].

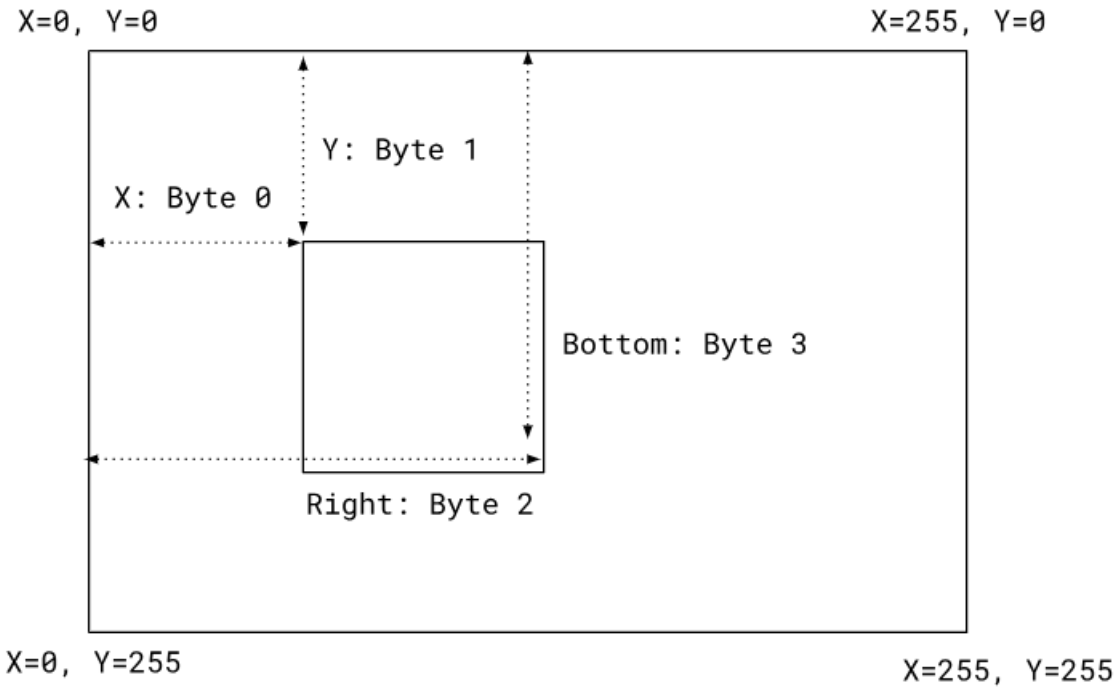


Figura 30 Ejemplo de representación de los datos capturados del módulo Person Sensor en una imagen [28].

Es importante tener en cuenta que la escala y la interpretación de los valores pueden variar según la implementación específica del sensor y la biblioteca utilizada [28].

4.2.5.3 Face Confidence

El parámetro *Face Confidence* refleja el nivel de certeza del sensor respecto a si lo capturado se corresponde realmente con un rostro, y no simplemente con un patrón parecido a los rasgos de un rostro humano. El valor de confianza oscilará entre 0, menos seguro, y 100, más seguro, almacenándose como un valor de un solo byte [28].

Al utilizar el valor de *Face Confidence* se puede establecer un umbral para determinar qué acciones tomar en función del nivel de confianza detectado. Por ejemplo, si se necesita una elevada confianza para realizar una acción crítica basada en el reconocimiento facial, se puede establecer un umbral más alto y tomar la acción únicamente cuando la confianza supere ese umbral [28].

4.2.5.4 Is facing

El parámetro *Is Facing* indica si el rostro detectado se encuentra mirando directamente a la cámara integrada en el módulo, a partir de la posición de la cabeza, por lo que los ojos de una persona aún pueden estar mirando en otra dirección [28].

4.2.5.5 Identity

En el módulo *Person Sensor*, la identificación de usuarios es una característica compleja, ya que busca personalizar respuestas sin exponer información biométrica por motivos de privacidad. Así, se asigna un ID comprendido entre los valores 0 y 7 a las personas reconocidas, requiriéndose de un proceso de calibración inicial para nuevas personas. El reconocimiento facial funciona mejor cuando el rostro está directamente frente al sensor. No se debe confiar exclusivamente en la identificación facial para la seguridad, ya que no es muy precisa y se enfoca en aplicaciones poco críticas, como la personalización de la experiencia del usuario. La asignación de ID se pierde con la pérdida de alimentación, pero se puede almacenar de forma persistente mediante configuración específica [28].

4.2.6 Configuración

El fabricante y los desarrolladores del módulo *Person Sensor* han intentado que el comportamiento predeterminado del sistema resulte lo más sencillo posible, si bien hay algunas opciones que pueden resultar útiles para optimizar el rendimiento o el consumo de energía del sistema. Se pueden aplicar estos cambios de configuración escribiendo un byte con la dirección del registro, seguido de un byte para indicar el valor, en el bus I_2C hacia el ID del módulo [28]. En la Tabla 14 se muestra la dirección y el byte correspondiente al módulo *Person Sensor*, en caso de que se requiera realizar alguna modificación en su configuración, mientras que en la Tabla 15 se pueden observar los modos de trabajo del módulo *Person Sensor.*

Tabla 14 Dirección y byte de configuración del módulo Person Sensor [28].

Address	Name	Default	Description
0x01	Mode	0x01 (continuous)	Mode. See mode table below.
0x02	Enable ID	0x00 (False)	Enable / Disable the ID model. With this flag set to False, only capture bounding boxes.
0x03	Single shot	0x00	Trigger a single-shot inference. Only works if the sensor is in standby mode.
0x04	Label next ID	0x00	Calibrate the next identified frame as person N, from 0 to 7. If two frames pass with no person, this label is discarded.
0x05	Persist IDs	0x01 (True)	Store any recognized IDs even when unpowered.
0x06	Erase IDs	0x0	Wipe any recognized IDs from storage.
0x07	Debug Mode	0x01 (True)	Whether to enable the LED indicator on the sensor.

Tabla 15 Modos de trabajo del módulo Person Sensor [28].

Mode	Name	Description
0x00	Standby	Lowest power mode, sensor is in standby and not capturing.
0x01	Continuous	Capture continuously, setting the GPIO trigger pin to high if a face is detected.

4.2.7 Privacidad

El módulo *Person Sensor* incluye un sensor de imagen, y se requiere que este hecho no represente una amenaza para la privacidad de los usuarios. Con este propósito, el fabricante diseñó el módulo de manera que resulte lo más resistente posible a que alguien acceda a los datos de imagen en bruto, disponiendo únicamente de los metadatos derivados de cada fotograma. Además, la interfaz con el módulo es muy limitada con el fin de reducir la posibilidad de interferir maliciosamente y simplificar las auditorías por parte de terceros [28].

Este enfoque limita lo que los desarrolladores pueden hacer con este módulo. Una restricción obvia es que no permite el acceso a los datos de imagen, pero tampoco admite la actualización del *firmware* o del modelo funcional, ya que hacerlo podría permitir la introducción de cambios sin control en el comportamiento del sensor [28].

4.3 Integración del módulo *Person Sensor* con el dispositivo Arduino MKRWAN 1310

Para establecer la conexión entre el módulo *Person Sensor* y el dispositivo Arduino MKRWAN 1310, se requiere del uso de la interfaz I_2C. Como se ha señalado en secciones previas, el módulo *Person Sensor* integra un conector estándar Qwiic para la interfaz I_2C con los siguientes códigos de color para los cables: amarillo para la señal SCL, azul para SDA, rojo para 3.3V, y negro para GND, como se observa en la Figura 31.

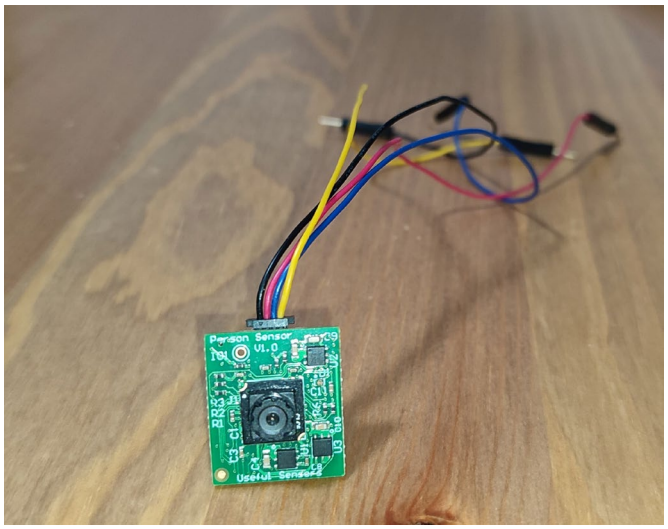


Figura 31 Módulo Person Sensor con conector QWiiC.

En el caso del dispositivo Arduino MKRWAN 1310 se emplea inicialmente el pin VCC para proporcionar al módulo *Person Sensor* la tensión de alimentación de 3.3V, el pin GND, y los pines digitales 11 y 12 para las señales SDA y SCL de la interfaz I_2C, de acuerdo con sus especificaciones. En la Figura 32 se pueden ver los pines referenciados.

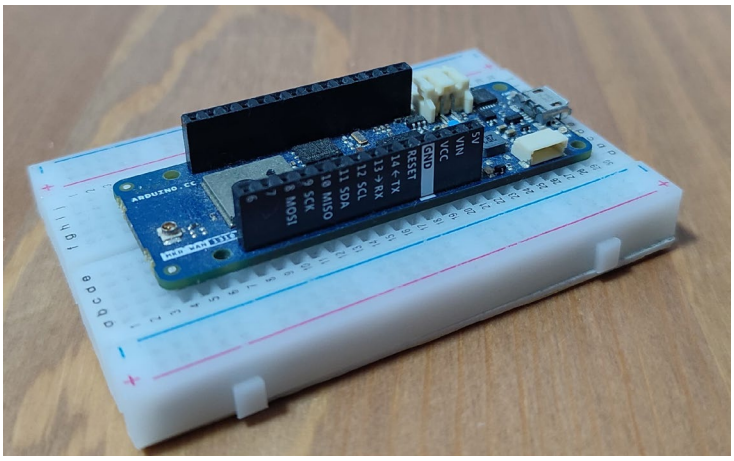


Figura 32 Arduino MKRWAN 1310.

En la Figura 33 se puede apreciar la interconexión física establecida inicialmente entre el módulo *Person Sensor* y el dispositivo Arduino MKRWAN 1310 mediante la interfaz I_2C.

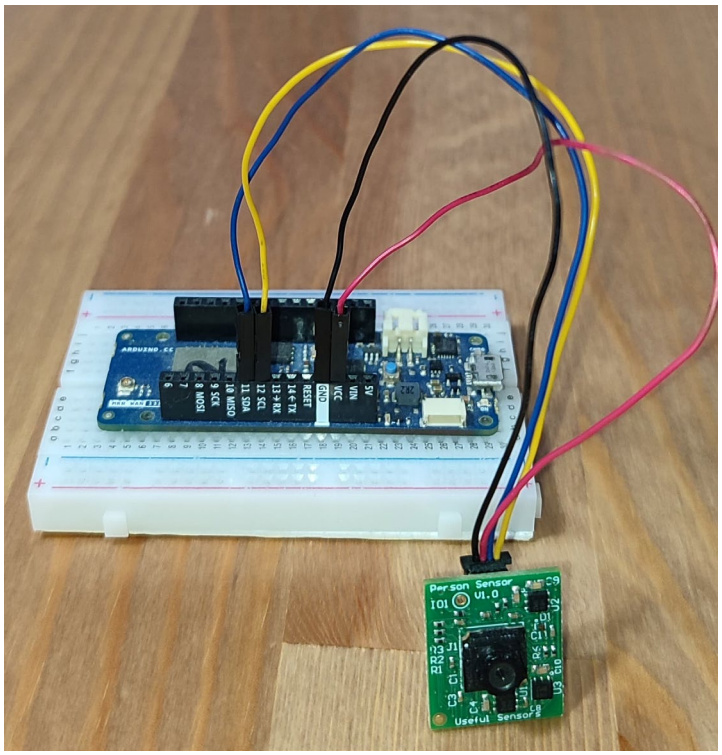


Figura 33 Dispositivo Arduino MKRWAN 1310 y dispositivo Person Sensor conectados a través de interfaz I2C.

Con el objetivo de poder obtener un conocimiento más detallado del funcionamiento del módulo *Person Sensor*, se integró inicialmente con el dispositivo Arduino MKRWAN 1310 una pantalla TFT modelo Adafruit ILI9341. Esta incorporación permitiría obtener una base sólida acerca del correcto procesamiento de los datos proporcionados por el módulo *Person Sensor* en su funcionamiento, así como una depuración más completa del código de los *sketches* que se desarrollen.

A partir del código de referencia disponible en los repositorios de GitHub de los fabricantes del módulo *Person Sensor* [29] [30], se elaboró un primer *sketch*, denominado `person_sensor_TFT.ino`, en el que se integra este módulo con el dispositivo Arduino MKRWAN 1310 y la pantalla LCD TFT Adafruit ILI9341.

`person_sensor_TFT.ino`

```
#include <Wire.h>
#include "person_sensor.h"
#include "Adafruit_GFX.h"
#include "Adafruit_ILI9341.h"
```

```
#include "SPI.h"
```

En la sección inicial del código desarrollado se incluyen las bibliotecas necesarias para acceder a las funciones y recursos específicos que se incluyen.

`#include <Wire.h>`: La biblioteca oficial `Wire` de Arduino proporciona funcionalidad para la comunicación basada en el protocolo I_2C en dispositivos Arduino que, en este caso, se utiliza para la comunicación con el módulo *Person Sensor*.

`#include "person_sensor.h"`: Esta biblioteca específica define las constantes, estructuras de datos y funciones necesarias para interactuar con el módulo *Person Sensor*.

`#include "Adafruit_GFX.h"` y `#include "Adafruit_ILI9341.h"`: La biblioteca `Adafruit_GFX.h`, desarrollada por Adafruit Industries Inc. proporciona funcionalidad gráfica general, mientras que `Adafruit_ILI9341.h` está específicamente diseñada para la integración de pantallas TFT basadas en el controlador ILI9341.

`#include "SPI.h"`: La biblioteca `SPI.h` se utiliza en este caso para la comunicación con la pantalla TFT mediante el protocolo *Serial Peripheral Interface* (SPI).

```
#define TFT_DC 1
#define TFT_CS 2
Adafruit_ILI9341 tft = Adafruit_ILI9341(TFT_CS, TFT_DC);
uint16_t color[4] = {ILI9341_CYAN , ILI9341_GREEN , ILI9341_RED ,
ILI9341_YELLOW};
const int32_t SAMPLE_DELAY_MS = 200;
```

En esta parte del código se crea la instancia del controlador de la pantalla TFT y su configuración. También se establece un vector de colores y se establece el tiempo de espera entre lecturas de los datos proporcionados por el módulo *Person Sensor*.

```
void setup() {
  Serial.begin(9600);
  Wire.begin();
  tft.begin();
}
```

En esta sección del código se define la función `setup()`, en la que se realizan las inicializaciones necesarias. En primer lugar, se inicializa la comunicación serie USB a una velocidad de 9600 baudios mediante la sentencia `Serial.begin(9600)`. Posteriormente, se habilita la comunicación I_2C con los parámetros por defecto mediante la sentencia `Wire.begin()`. Finalmente, la línea `tft.begin()` se utiliza para inicializar la pantalla TFT conectada, permitiendo que se realicen operaciones de visualización.

```
void loop() {
  person_sensor_results_t results = {};
  if (!person_sensor_read(&results)) {
    Serial.println("No person sensor results found on the i2c bus");
    delay(SAMPLE_DELAY_MS);
    return;
  }

  Serial.print(results.num_faces);
  Serial.println(" faces found");
  tft.fillScreen(ILI9341_BLACK);
  for (int i = 0; i < results.num_faces; ++i) {
    const person_sensor_face_t* face = &results.faces[i];
    Serial.print("Face #");
    Serial.print(i);
    Serial.print(": ");
    Serial.print(face->box_confidence);
    Serial.print(" confidence, (");
    Serial.print(face->box_left);
    Serial.print(", ");
    Serial.print(face->box_top);
```

```
    Serial.print("), (");
    Serial.print(face->box_right);
    Serial.print(", ");
    Serial.print(face->box_bottom);
    Serial.print("), ");
    if (face->is_facing) {
      Serial.println("facing");
    } else {
      Serial.println("not facing");
    }
    if (face->box_confidence > 90) {
       tft.drawRect(uint16_t (face->box_left), uint16_t (face->box_top),
                    uint16_t (face->box_right - face->box_left), uint16_t
(face->box_bottom - face->box_top),
                    color[i]);
       if (face->is_facing) {
          tft.drawLine(uint16_t (face->box_left), uint16_t (face->box_top),
uint16_t (face->box_right), uint16_t (face->box_bottom), color[i]);
          tft.drawLine(uint16_t (face->box_left), uint16_t (face->box_bottom),
uint16_t (face->box_right), uint16_t (face->box_top), color[i]);
       }
    }
  }
  delay(SAMPLE_DELAY_MS);
}
```

En esta sección del código se establece una estructura correspondiente al bucle principal `loop()`, en la que se declaran e inicializan las variables y estructuras necesarias. `person_sensor_results_t results = {};` se utiliza para crear una estructura denominada `results` que almacenará los datos obtenidos del módulo *Person Sensor*. A continuación, se verifica mediante la llamada a la función `person_sensor_read(&results)` en caso de haber obtenido nuevos datos desde el sensor, y en caso contrario, se muestra un mensaje de error a través del terminal serie y se introduce el tiempo de espera entre lecturas establecido, antes de continuar.

En caso de que la lectura del sensor haya proporcionado nuevos datos, se procede a mostrar a través del terminal serie el número de caras detectadas. A continuación, se itera a través de las caras detectadas, mostrando para cada una de ellas la información relativa el nivel de confianza de la detección, correspondiente al parámetro `box_confidence`, las coordenadas de la caja que rodea la cara, y si la persona está mirando o no, a partir del parámetro `is_facing`. En función de la confianza de la detección y la orientación de la cara, se dibujan rectángulos y líneas en la pantalla TFT.

En la Figura 34 se representa el esquema de interconexión entre el dispositivo Arduino MKRWAN 1310, el sensor *Person Sensor*, y la pantalla TFT, mientras que la Figura 35 y en la Figura 36 se muestra el montaje en funcionamiento del sistema realizado, detectando respectivamente un rostro y dos rostros en el entorno.

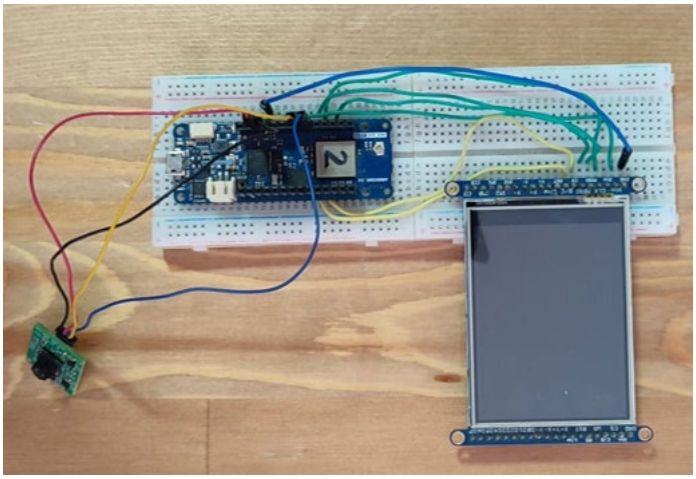


Figura 34 Circuito conformado por el Arduino MKRWAN 1310, el Person Sensor y la pantalla TFT.

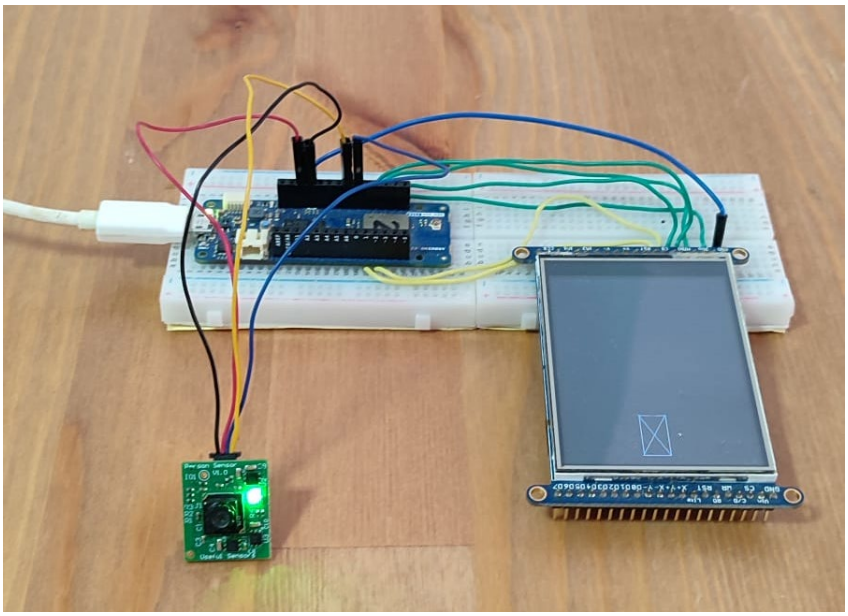


Figura 35 Ejemplo del sistema formado por Arduino MKRWAN 1310, Person Sensor y pantalla TFT con un rostro detectado.

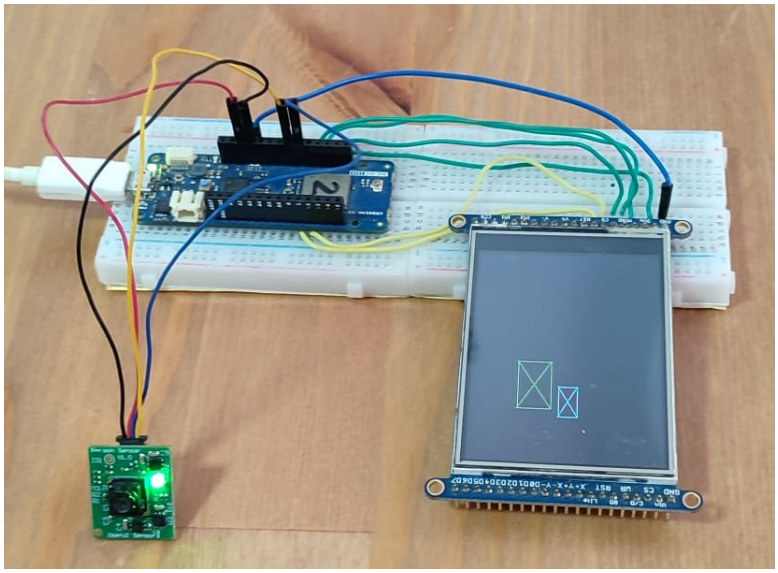


Figura 36 Ejemplo del sistema formado por Arduino MKRWAN 1310, Person Sensor y pantalla TFT con dos rostros detectados.

4.4 Gestión de consumo energético

En el ámbito de IoT, la gestión eficiente del consumo de potencia es un elemento fundamental, existiendo en la actualidad múltiples bibliotecas que permiten controlar y optimizar el uso de la energía en dispositivos basados en Arduino.

La medición precisa del consumo de energía de un sensor es esencial en numerosos proyectos y aplicaciones, especialmente cuando se busca optimizar la eficiencia energética. En este apartado se explorarán los recursos disponibles en el dispositivo Arduino MKRWAN 1310, así como las bibliotecas específicas para este tipo de dispositivos que se han considerado con el fin de obtener una optimización del consumo de energía.

4.4.1 Tipos de alimentación de los dispositivos Arduino

Las plataformas de desarrollo Arduino pueden alimentarse de diversas maneras mediante el uso, tanto de conectores dedicados (puertos USB, conectores de batería, …), como de pines dedicados. Esta sección describe las características de los principales métodos de alimentación del dispositivo Arduino MKRWAN 1310, así como los diferentes pines y conectores disponibles.

4.4.1.1 Conector USB

La forma más común y sencilla para alimentar el dispositivo Arduino MKRWAN 1310 es mediante el conector micro-USB integrado. Este conector USB proporciona una línea regulada de 5V para alimentar los componentes electrónicos de la placa de prototipado. Sin embargo, los 5V del conector USB también pueden utilizarse para alimentar componentes externos a través del pin +5V, que se encuentra en la interfaz de E/S el dispositivo Arduino MKRWAN 1310 [31]. Como referencia, en la Figura 37 se muestra el conector micro-USB disponible en el dispositivo Arduino Nano RP2040, el cual es idéntico al utilizado en el modelo Arduino MKRWAN 1310.

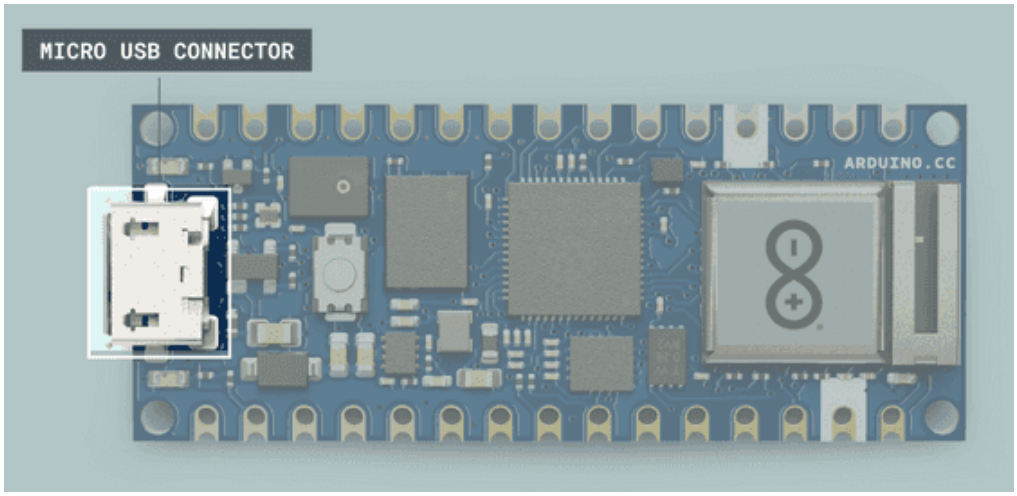


Figura 37 Conector micro-USB del Arduino Nano RP2040 [31].

Algo importante acerca de la alimentación a través del conector micro-USB es la clasificación de corriente del dispositivo *host*. Por ejemplo, el dispositivo *host* USB puede ser un ordenador, de forma que la fuente de alimentación de 5V del dispositivo Arduino se corresponda en este caso con uno de los puertos USB del ordenador. Además del puerto USB de un ordenador, también podrían utilizarse como fuente de alimentación baterías o bancos de energía. Los bancos de energía suelen tener una o más salidas USB que proporcionan líneas reguladas de 5V con diferentes clasificaciones de corriente. Los dispositivos Arduino que funcionan con 5V utilizan directamente la línea de 5V regulada asociada al conector USB, mientras que las placas que operan con 3V3 regulan la línea de 5V del conector USB a 3.3V mediante un regulador de tensión incorporado. La clasificación de corriente de salida desde el pin de 5V variará según la fuente de alimentación utilizada [31].

4.4.1.2 Conector de Batería

Algunos dispositivos Arduino incorporan un conector específico para conectar una batería externa y utilizarla como fuente de alimentación principal o secundaria. Los dispositivos de la familia Arduino MKR utilizan un conector de batería SMD PH de 2 pines y 2 mm [31], como se muestra en la Figura 38.

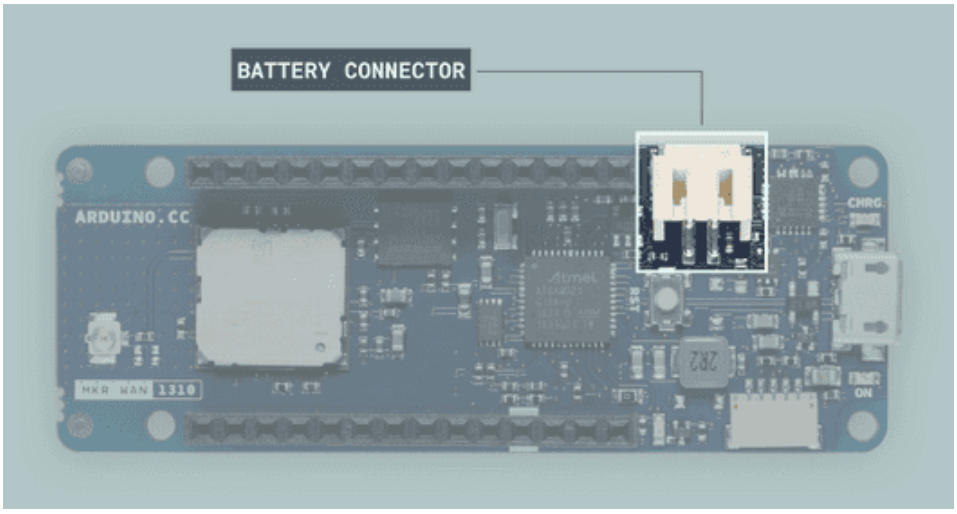


Figura 38 Conector de batería del Arduino MKRWAN 1310 [31].

En concreto, el dispositivo Arduino MKRWAN 1310 dispone de un circuito de gestión de carga de batería integrado en la placa. Este circuito integra las funciones más comunes de gestión de batería y alimentación, como un cargador de batería, un regulador de tensión, y un interruptor de carga [31].

4.4.1.3 Pin V_{IN}

En los dispositivos Arduino, V_{IN} representa por lo general un pin de alimentación con una función dual. Por un lado, este pin puede operar como entrada de tensión para fuentes de alimentación externas reguladas que no utilicen un conector *jack*. Por otro lado, en algunos dispositivos Arduino puede funcionar también como una salida de tensión. Una consideración importante es que el pin V_{IN} está conectado directamente al pin de entrada del regulador de tensión incorporado en las placas Arduino, por lo no proporciona protección contra polaridad inversa [31].

4.4.1.4 Pin 3V3/+5V

Los pines 3V3 y +5V también son pines de alimentación con una función dual. Por un lado, pueden operar como salidas de alimentación, ya que estos pines están directamente conectados a las salidas de los reguladores de tensión de 3.3V y +5V incorporados en el dispositivo (dependiendo del modelo). Además, los pines 3V3 y +5V también pueden

utilizarse como entradas de alimentación en caso de no conectarse ninguna fuente de alimentación regulada a través de los otros conectores de alimentación (puerto USB, conector *jack* o pin V_{IN}) [31].

Aunque los pines 3V3 y +5V se pueden utilizar como entradas de alimentación, no se recomienda hacerlo si no se conecta ninguna fuente a través del puerto USB, o el pin V_{IN}. Los pines 3V3 y +5V están directamente conectados al pin de salida del regulador de tensión incorporado. Si la tensión en el pin de salida del regulador de tensión es mayor que la tensión de entrada, podría circular una corriente importante desde el pin de salida del regulador de tensión hacia su pin de entrada. Esta corriente podría dañar permanentemente el regulador de tensión de la placa. Por lo tanto, no se recomienda utilizar los pines 3V3 y +5V como entradas de alimentación, a menos que haya una fuente de alimentación regulada conectada a través del puerto USB o el pin V_{IN} [31].

4.4.2 Bibliotecas Arduino para la gestión del consumo de potencia

4.4.2.1 Arduino Low Power

La biblioteca oficial *Arduino Low Power* permite gestionar los estados de bajo consumo de energía en los dispositivos Arduino más recientes [32]. Para usar esta biblioteca se debe incluir en el *sketch* la sentencia `#include "<ArduinoLowPower.h>"`

Compatibilidad:

La biblioteca `Arduino Low Power` es compatible con las arquitecturas SAMD y nRF52, por lo que se puede usar en los siguientes dispositivos Arduino:

- Arduino MKRFOX 1200.
- Arduino MKRGSM 1400.
- Arduino MKRNB 1500.
- Arduino MKRVIDOR 4000.
- Arduino MKRWAN 1300 (LoRa connectivity).
- **Arduino MKRWAN 1310.**
- Arduino MKRWI-FI 1010.

- Arduino MKRZERO (I2S bus & SD for sound, music & digital audio data).
- Arduino MKR1000 Wi-Fi.
- Arduino Nano 33 IoT.
- Arduino Zero.

Métodos:

- `LowPower.idle()`
- `LowPower.sleep()`
- `LowPower.deepSleep()`
- `LowPower.attachInterruptWakeup()`
- `LowPower.CompanionLowPowerCallback()`
- `LowPower.companionSleep()`
- `LowPower.companionWakeup()`

4.4.2.2 RTCZero

Por su parte, la biblioteca `RTCZero` permite hacer uso de las funcionalidades del RTC (*Real-Time Clock*) integrado en algunos dispositivos Arduino, como Arduino Zero, MKRZero y MKR1000 [33]. Esta biblioteca permite utilizar el RTC interno de los dispositivos Arduino basados en arquitecturas SAMD. Un RTC es un reloj que lleva un registro de la hora actual, y que puede utilizarse para programar acciones en un momento específico. La mayoría de los RTC utilizan un oscilador a cristal (como en el caso del dispositivo Arduino Zero) cuya frecuencia es de 32.768 kHz (la misma frecuencia utilizada en relojes de cuarzo). Esta frecuencia es igual a 2^15 ciclos por segundo y, por lo tanto, es una velocidad conveniente para usar con circuitos de contadores binarios simples [33].

Además, el RTC puede seguir funcionando en cualquier modo de suspensión, por lo que se puede utilizar para despertar el dispositivo de los modos de suspensión de manera programada. Cada vez que se alimenta la placa, el RTC se reinicia y comienza desde una fecha estándar. Para mantener la hora y el funcionamiento del RTC, es necesario que el dispositivo permanezca alimentado. Una batería de litio de botón, o cualquier batería en el rango de 3V conectada a través de un diodo al pin de 3.3V, sería suficiente para

mantener el RTC activo si el microcontrolador se configura en modo de suspensión antes de desconectar la alimentación estándar USB o a través del pin V_{IN} [33]. Para usar esta biblioteca se debe incluir en el *sketch* la sentencia `#include <RTCZero.h>`

Compatibilidad:

La biblioteca `RTCZero` es compatible con la arquitectura SAMD, por lo que puede utilizarse en las siguientes placas Arduino:

- Arduino MKRFOX 1200.
- Arduino MKRGSM 1400.
- Arduino MKRNB 1500.
- Arduino MKRVIDOR 4000.
- Arduino MKRWAN 1300 (LoRa connectivity).
- **Arduino MKRWAN 1310.**
- Arduino MKRWI-FI 1010.
- Arduino MKRZERO (I2S bus & SD for sound, music & digital audio data).
- Arduino MKR1000 Wi-Fi.
- Arduino Nano 33 IoT.
- Arduino Zero.

Métodos:

- `begin()`
- `setHours()`
- `setMinutes()`
- `setSeconds()`
- `setTime()`
- `setYear()`
- `setMonth()`
- `setDay()`
- `setDate()`
- `getHours()`
- `getMinutes()`

- `getSeconds()`
- `getYear()`
- `getMonth()`
- `getDay()`
- `setAlarmHours()`
- `setAlarmMinutes()`
- `setAlarmSeconds()`
- `setAlarmTime()`
- `setAlarmYear()`
- `setAlarmMonth()`
- `setAlarmDay()`
- `setAlarmDate()`
- `enableAlarm()`
- `disableAlarm()`
- `attachInterrupt()`
- `detachInterrupt()`
- `standbyMode()`

4.4.3 Medidas iniciales de consumo de energía

En primer lugar, se procedió a evaluar el consumo del dispositivo Arduino MKRWAN 1310 con diferentes métodos de alimentación y en múltiples escenarios, pasando de alimentar el circuito a través del puerto micro USB, a hacerlo a través del pin V_{IN}, y mediante el conector de batería externa utilizando una batería modelo BAT536 M6 Li-Po de 1400mAh de 3.7V. Este procedimiento se llevó a cabo en dos escenarios diferentes para cada caso, uno basado en la ejecución de un *sketch* en el que se integra la funcionalidad del dispositivo Arduino MKRWAN 1310 con el módulo *Person Sensor*, y otro en el que se hace uso de las bibliotecas `Arduino Low Power` y `RTCZero` para caracterizar el mínimo consumo de potencia posible en el dispositivo Arduino MKRWAN 1310.

Como referencia, para la evaluación de los resultados obtenidos se utilizaron los datos de consumo mínimo disponibles en la página web oficial de Arduino [26], así como mediciones previamente realizadas por otros usuarios y publicadas en el Foro oficial de

Arduino [34]. Estos datos proporcionaron un marco de referencia significativo para contextualizar y comparar los resultados experimentales obtenidos en este trabajo.

4.4.3.1 Evaluación de medidas de consumo con la integración del dispositivo Arduino MKRWAN 1310 y el módulo Person Sensor

En este apartado se muestran los resultados obtenidos a partir de las medidas de consumo de energía del dispositivo Arduino MKRWAN 1310 cuando el módulo *Person Sensor* se encuentra en los estados de detección y no detección de un rostro. El *sketch* que se desarrolló con este propósito se denominó `only_person_sensor.ino`, y es similar al *sketch* `person_sensor_TFT.ino` presentado en el apartado anterior, aunque eliminando la funcionalidad asociada a la pantalla TFT, y aumentando el intervalo de tiempo entre muestras con el fin de poder realizar la medición del consumo de energía instantáneo. En este *sketch*, el módulo *Person Sensor* se configura inicialmente en modo s*tandby* (ver Tabla 14 y Tabla 15) y posteriormente se toma una única captura para la detección de rostros.

`only_person_sensor.ino`

```
#include <Wire.h>
#include "person_sensor.h"
#include "SPI.h"

const int32_t SAMPLE_DELAY_MS = 2000;

void setup() {
  Serial.begin(9600);
  Wire.begin();
  person_sensor_write_reg(PERSON_SENSOR_REG_MODE, 0x00);
}

void loop() {
  person_sensor_write_reg(PERSON_SENSOR_REG_SINGLE_SHOT, 0x01);
  delay(SAMPLE_DELAY_MS);


  person_sensor_results_t results = {};
```

```
  if (!person_sensor_read(&results)) {
    Serial.println("No person sensor results found on the i2c bus");
    delay(SAMPLE_DELAY_MS);
    return;
  }

  Serial.print(results.num_faces);
  Serial.println(" faces found");

  for (int i = 0; i < results.num_faces; ++i) {
    const person_sensor_face_t* face = &results.faces[i];
    Serial.print("Face #");
    Serial.print(i);
    Serial.print(": ");
    Serial.print(face->box_confidence);
    Serial.print(" confidence, (");
    Serial.print("), ");
    if (face->is_facing) {
      Serial.println("facing");
    } else {
      Serial.println("not facing");
    }
  }
  delay(SAMPLE_DELAY_MS);
}
```

Como se aprecia en la Figura 39, el consumo permanece relativamente constante ante ambos estados de funcionamiento del módulo *Person Sensor* cuando se alimenta mediante un banco de energía a través del conector USB, presentando una variación mínima de tan solo 1.8 mA que puede atribuirse principalmente al consumo del LED integrado en la placa del módulo.

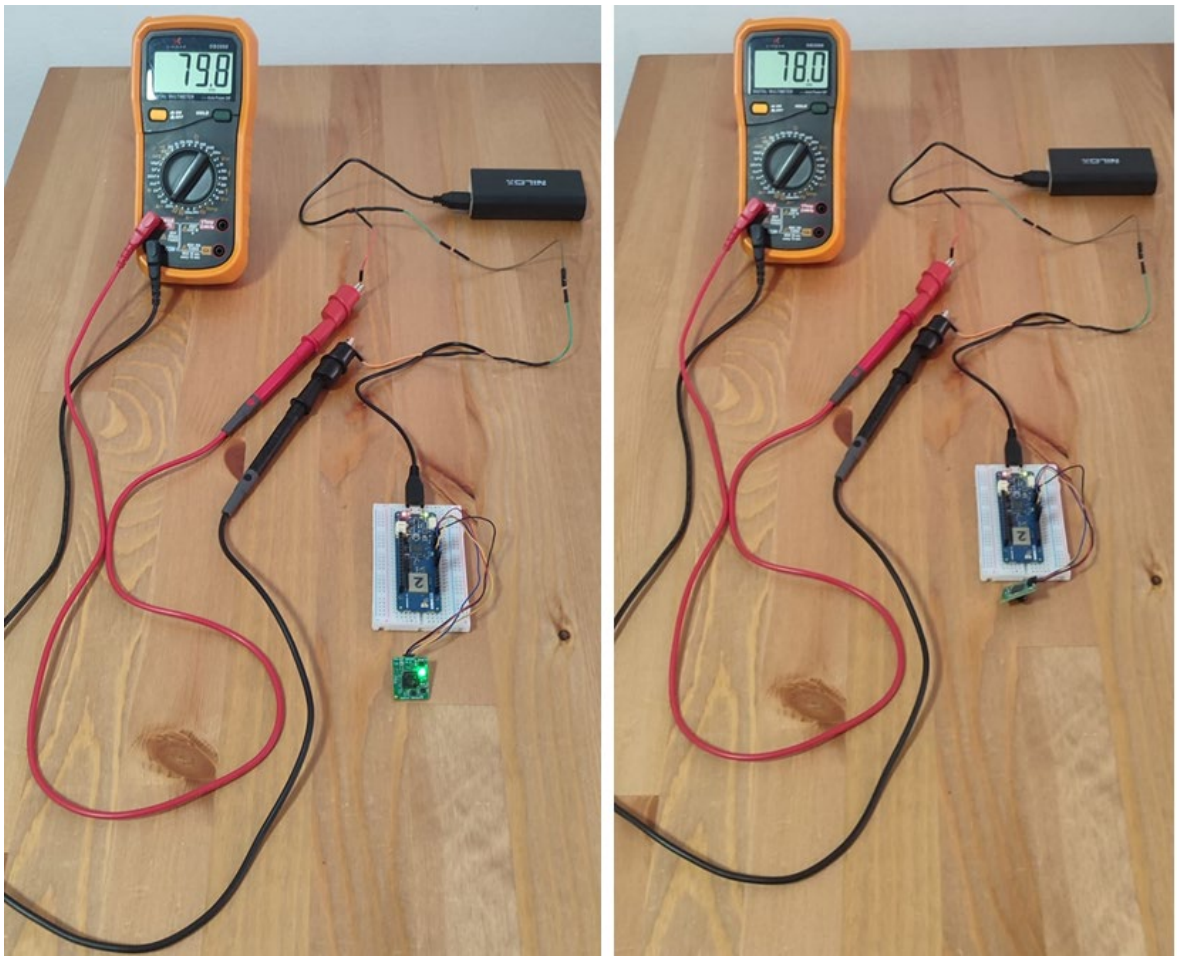


Figura 39 Medida de consumo del dispositivo Arduino MKRWAN 1310 conectado al módulo Person Sensor alimentado mediante el conector micro USB y ejecutando el sketch only_person_sensor.ino.

Al realizar la medida de consumo cuando el dispositivo Arduino MKRWAN 1310 se alimenta a través del pin V_{IN}, se observa que el resultado es congruente con la medición previa, mostrando una variación de consumo similar, que en este caso asciende a 2.1 mA, en comparación con la diferencia anterior de 1.8 mA, como se aprecia en la Figura 40.

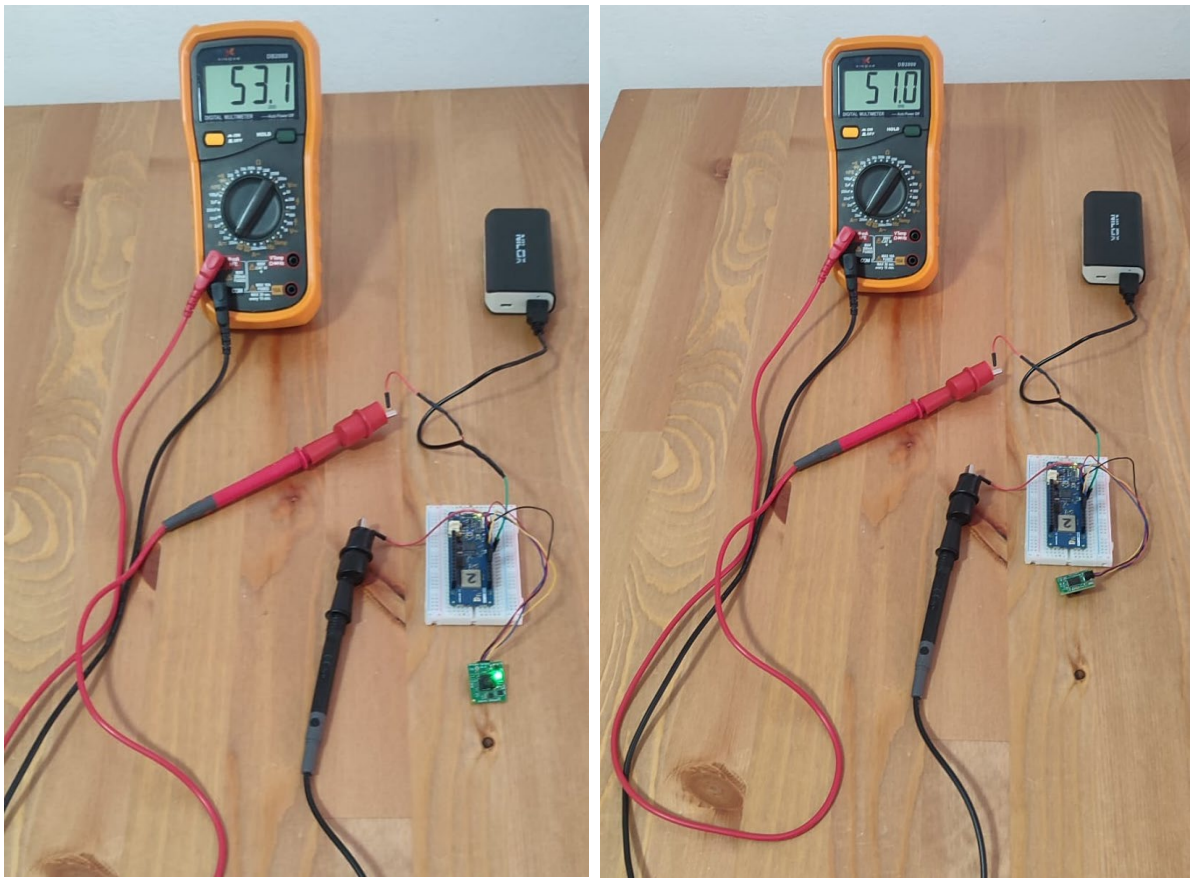


Figura 40 Medida de consumo del dispositivo Arduino MKRWAN 1310 conectado al módulo Person Sensor alimentado mediante el pin V_{IN} y ejecutando el sketch only_person_sensor.ino.

Finalmente, cuando se alimenta el dispositivo Arduino MKRWAN 1310 directamente a través del conector de batería externa, la diferencia de consumo correspondiente a la detección y no detección de rostro en el módulo *Person Sensor* es de 1.8mA, como se observa en la Figura 41.

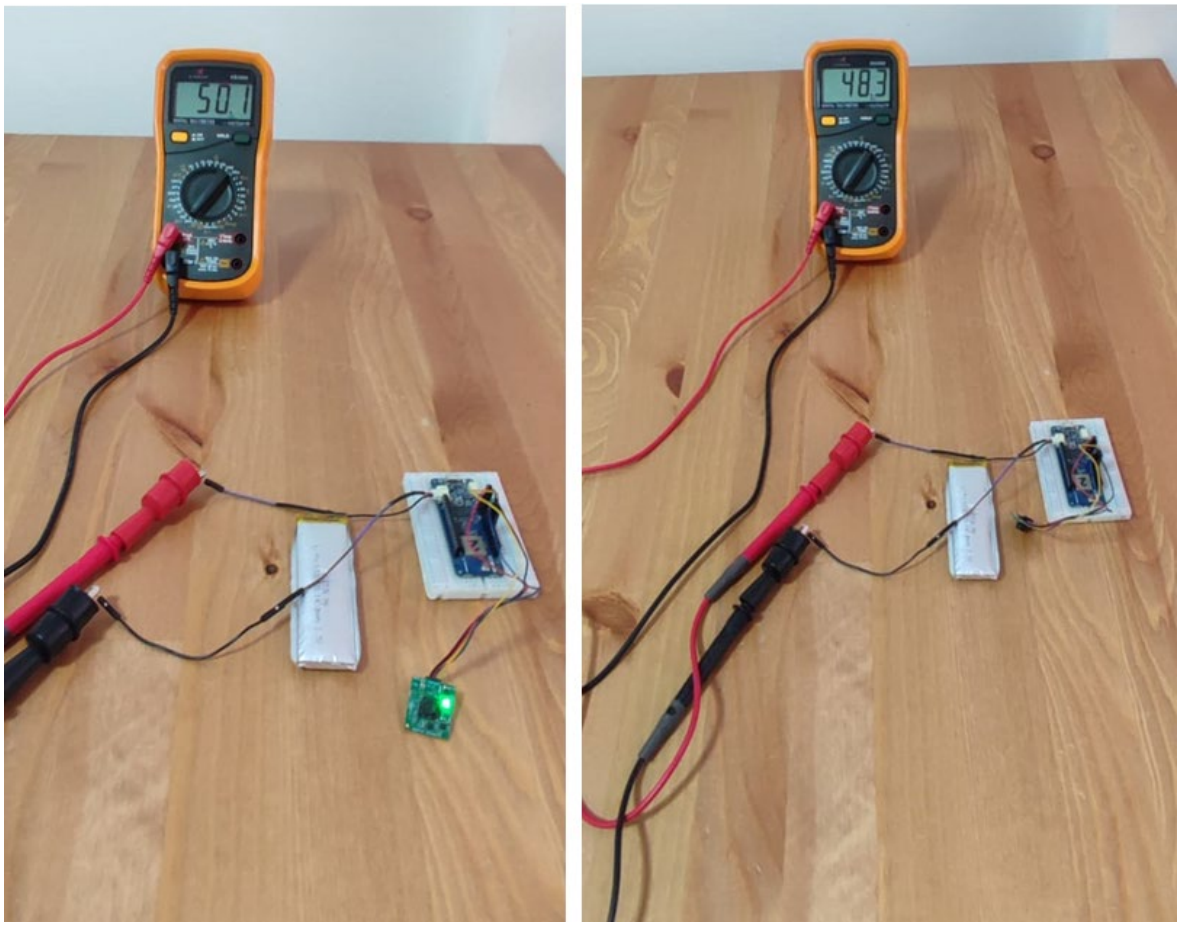


Figura 41 Medida de consumo del dispositivo Arduino MKRWAN 1310 conectado al módulo Person Sensor alimentado mediante el conector de batería y ejecutando el sketch only_person_sensor.ino.

Como conclusión inicial, a partir del *sketch* desarrollado, y haciendo uso de diferentes fuentes y métodos para la alimentación de los dispositivos, se deduce que la variación en el consumo se mantiene debido a una consistencia en las condiciones de funcionamiento, estableciéndose que la forma de alimentar el dispositivo Arduino MKRWAN 1310 ejerce una influencia constante en los niveles de consumo en ambos estados de funcionamiento del módulo *Person Sensor*.

4.4.3.2 Evaluación de medidas de mínimo consumo del dispositivo Arduino MKRWAN 1310

En este apartado se muestran los resultados obtenidos a partir de la medición del mínimo consumo de energía del dispositivo Arduino MKRWAN 1310 haciendo uso de las bibliotecas `Arduino Low Power` y `RTCZero`. El *sketch* que se desarrolló con este propósito se denominó `mkr1310_lowpower_v2.ino`.

mkr1310_lowpower_v2.ino

```
#include <Wire.h>
#include <RTCZero.h>
#include <ArduinoLowPower.h>
#include <MKRWAN.h>
```

En primer lugar, se incluyen las bibliotecas necesarias, que se han comentado y descrito en secciones anteriores, a excepción de MKRWAN, que posibilitan la comunicación basada en la tecnología LoRa® y redes LoRaWAN en los dispositivos MKRWAN 1300/1310.

```
#define RTC_MATCH_MINUTE 1
#define RTC_MATCH_SECOND 10

RTCZero rtc;
LoRaModem modem;
```

Esta sección del código define constantes para configurar alarmas de coincidencia de tiempo y crea la instancia de dos objetos: rtc y modem. La constante RTC_MATCH_MINUTE se establece al valor 1, y RTC_MATCH_SECOND al valor 10, con el objetivo de programar alarmas de tiempo específicas. El objeto rtc se utiliza para gestionar en todo momento el instante de tiempo real en el dispositivo Arduino. Por otro lado, el objeto modem está relacionado con el transceptor LoRa® Murata CMWX1ZZABZ integrado en la placa Arduino MKRWAN 1310, y se utiliza para configurar y gestionar la comunicación basada en la tecnología LoRa®.

```
boolean Modules_begin() {
  digitalWrite(LED_BUILTIN, HIGH);

  if (!modem.connected()) {
    if (!modem.begin(EU868)) {
    };
  }
```

```
}
```

La función `Modules_begin()` lleva a cabo una serie de operaciones fundamentales. En su inicio, realiza la activación del LED incorporado en el dispositivo Arduino MKRWAN 1310, lo que permite disponer de una señal visual del estado en el que se encuentra en todo momento. Posteriormente, lleva a cabo una verificación del estado de conexión con el transceptor LoRa® Murata CMWX1ZZABZ integrado en la placa Arduino MKRWAN 1310. En caso de establecerse esta conexión, se procede a su inicialización en la región correspondiente a Europa (EU868). La inclusión de esta función es crucial, ya que su ejecución es necesaria para evitar bloqueos del programa y permitir el adecuado estado de suspensión del transceptor LoRa® cuando no se encuentre en uso.

```
void sleep() {
  delay(2000);

  modem.sleep();

  pinMode(LORA_IRQ_DUMB, INPUT);
  pinMode(LORA_BOOT0, INPUT);
  pinMode(LORA_RESET, INPUT);

  digitalWrite(LORA_IRQ_DUMB, LOW);
  digitalWrite(LORA_BOOT0, LOW);
  digitalWrite(LORA_RESET, LOW);

  int rtc_alarm = (rtc.getMinutes() + RTC_MATCH_MINUTE) % 60;
  rtc.setAlarmMinutes(rtc_alarm);
  rtc.setAlarmSeconds(RTC_MATCH_SECOND);
  rtc.enableAlarm(rtc.MATCH_MMSS);

  USBDevice.detach();

  digitalWrite(LED_BUILTIN, LOW);

  LowPower.deepSleep();
}
```

La función `sleep()` cumple en el código una serie de funciones destinadas a la gestión del estado de suspensión o inactividad del dispositivo Arduino MKRWAN 1310. En primera instancia, se introduce un retardo de 2 segundos con el fin de permitir que el sistema se prepare para entrar en un estado de bajo consumo de potencia. A continuación, se activa el modo de suspensión del transceptor LoRa® mediante la función `modem.sleep()` con el fin de minimizar el consumo de energía durante este período. Posteriormente, se establecen los pines de la interfaz del transceptor LoRa® (`LORA_IRQ_DUMB`, `LORA_BOOT0`, `LORA_RESET`) como entradas (`INPUT`) y se asegura que la tensión en estos pines se encuentre a nivel bajo (`LOW`), lo que se ha comprobado que contribuye al ahorro de energía y evita posibles interferencias. Esta función calcula también el tiempo, en minutos, hasta el próximo evento de alarma para la configuración del reloj en tiempo real (RTC), basándose en el valor definido como `RTC_MATCH_MINUTE`. Además, configura la alarma del RTC para que se active en el minuto calculado y en el segundo definido en la constante `RTC_MATCH_SECOND`, empleando las funciones `rtc.setAlarmMinutes` y `rtc.setAlarmSeconds` de la biblioteca `RTCZero`. Por otro lado, se utiliza la función `rtc.enableAlarm` para habilitar la alarma, configurándose para que coincida con un evento que considere minutos y segundos. Para reducir aún más el consumo de energía, se desconecta el dispositivo USB mediante la llamada a la función `USBDevice.detach()`.

Finalmente, se apaga el LED incorporado en el dispositivo Arduino MKRWAN 1310 (`LED_BUILTIN`) con el fin de minimizar el consumo de energía en este estado de suspensión, entrando el sistema en un estado de bajo consumo de energía mediante la llamada a la función `LowPower.deepSleep()` de la biblioteca `Arduino Low Power`. La función `sleep` se encarga de gestionar de manera eficiente el estado de suspensión del dispositivo, optimizando el consumo de energía y preparando el sistema para despertar en respuesta a eventos específicos o alarmas definidas mediante el reloj en tiempo real.

```
void wakeup() {

  USBDevice.attach();

  delay(1000);
```

```
}
```

La función `wakeup()` realiza una serie de acciones destinadas a despertar al dispositivo Arduino MKRWAN 1310 de su estado de suspensión o inactividad. En primer lugar, se vuelve a habilitar la conexión USB del dispositivo mediante la función `USBDevice.attach()`, lo que permite reestablecer la comunicación serie a través del puerto USB y la interacción con el sistema.

Posteriormente, se introduce una latencia de 1 segundo mediante la función `delay(1000)`, que proporciona un breve período de tiempo para que el dispositivo se estabilice y complete cualquier inicialización necesaria tras salir del estado de suspensión.

```
void setup() {
  pinMode(LED_BUILTIN, OUTPUT);
  digitalWrite(LED_BUILTIN, LOW);

  rtc.begin();

  for (int i = 0; i < 15; i++) {
    pinMode(i, INPUT_PULLUP);
  }
}
```

La función `setup()` del código *sketch* desarrollado se encarga de configurar el entorno y los pines del dispositivo Arduino MKRWAN 1310. En primer lugar, se define la configuración del pin `LED_BUILTIN` como una salida (`OUTPUT`) y se establece su estado inicial como apagado (`LOW`).

A continuación, se inicia el RTC mediante la llamada a la función `rtc.begin()`, realizándose una configuración específica para varios pines del dispositivo. Se utiliza un bucle `for` que recorre los pines desde 0 hasta 14, configurando en cada iteración del bucle el pin correspondiente como `INPUT_PULLUP`, de forma que se habilite una

resistencia interna de *pull-up* que ayuda a mantener el estado alto de los pines cuando no se les aplica ninguna tensión.

```
void loop() {
  Modules_begin();
  sleep();
  wakeup();
}
```

Por último, en la función `loop()` del *sketch* se llama a la función `Modules_begin()` comentada anteriormente, y se configura el dispositivo Arduino MKRWAN 1310 en un estado de bajo consumo de energía a través de la llamada a la función `sleep()`, lo que permitirá estimar el consumo mínimo de potencia que puede obtenerse. Finalmente, la función `wakeup()` se encarga de despertar al dispositivo tras un período de suspensión. Este ciclo se repite continuamente, de forma que cuando el dispositivo vuelve del estado de suspensión profunda, la ejecución del código continua desde la línea que entró en suspensión.

Una vez explicado en detalle el código del *sketch* desarrollado, se presentan a continuación los resultados obtenidos a partir de las medidas experimentales del consumo de energía del dispositivo Arduino MKRWAN 1310 cuando se encuentra en el estado normal de funcionamiento, así como en el estado de suspensión profunda, para cada una de las formas de alimentación consideradas.

En la Figura 42, el dispositivo Arduino MKRWAN 1310 se alimenta a través del conector micro-USB. La imagen de la izquierda ilustra los resultados obtenidos durante el funcionamiento normal del dispositivo, mientras que en la imagen de la derecha se representan las medidas realizadas cuando el dispositivo se encuentra en modo de suspensión profunda. Como se observa, la diferencia de consumo es de 19 mA entre ambos estados, con un consumo de 24.7 mA en funcionamiento normal, y de 8.7 mA en el estado de suspensión profunda.

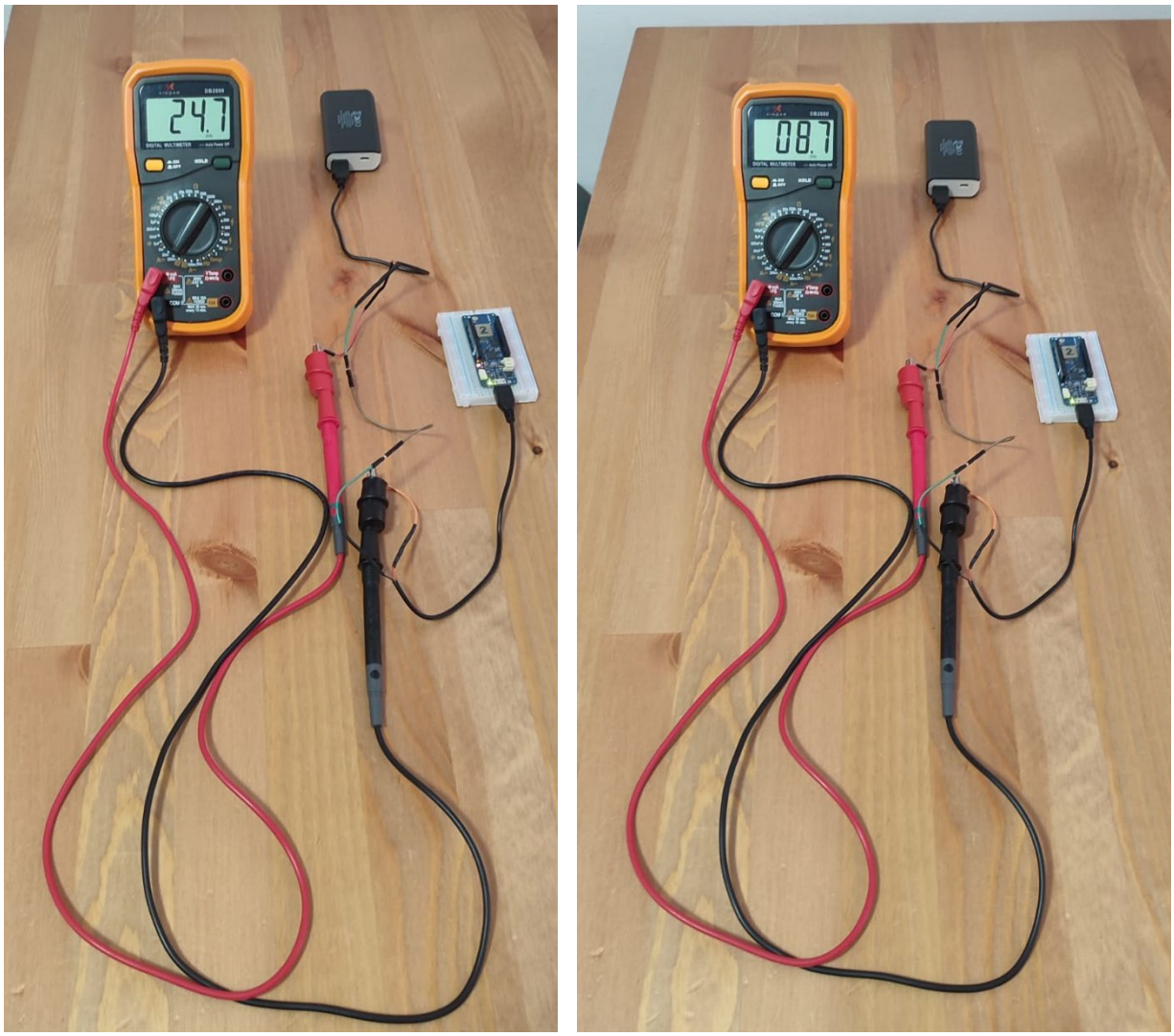


Figura 42 Medida de consumo del dispositivo Arduino MKRWAN 1310 alimentado mediante el conector micro USB y ejecutando el sketch "mkr1310_lowpower_v2".

En la Figura 43, el dispositivo Arduino MKRWAN 1310 se alimenta a través del pin V_{IN}. A partir de los resultados obtenidos se observan variaciones significativas con respecto al escenario anterior. En funcionamiento normal se observa una reducción del consumo a 23.8 mA, mientras que, en el estado de suspensión profunda, el consumo aumenta a 9.19 mA.

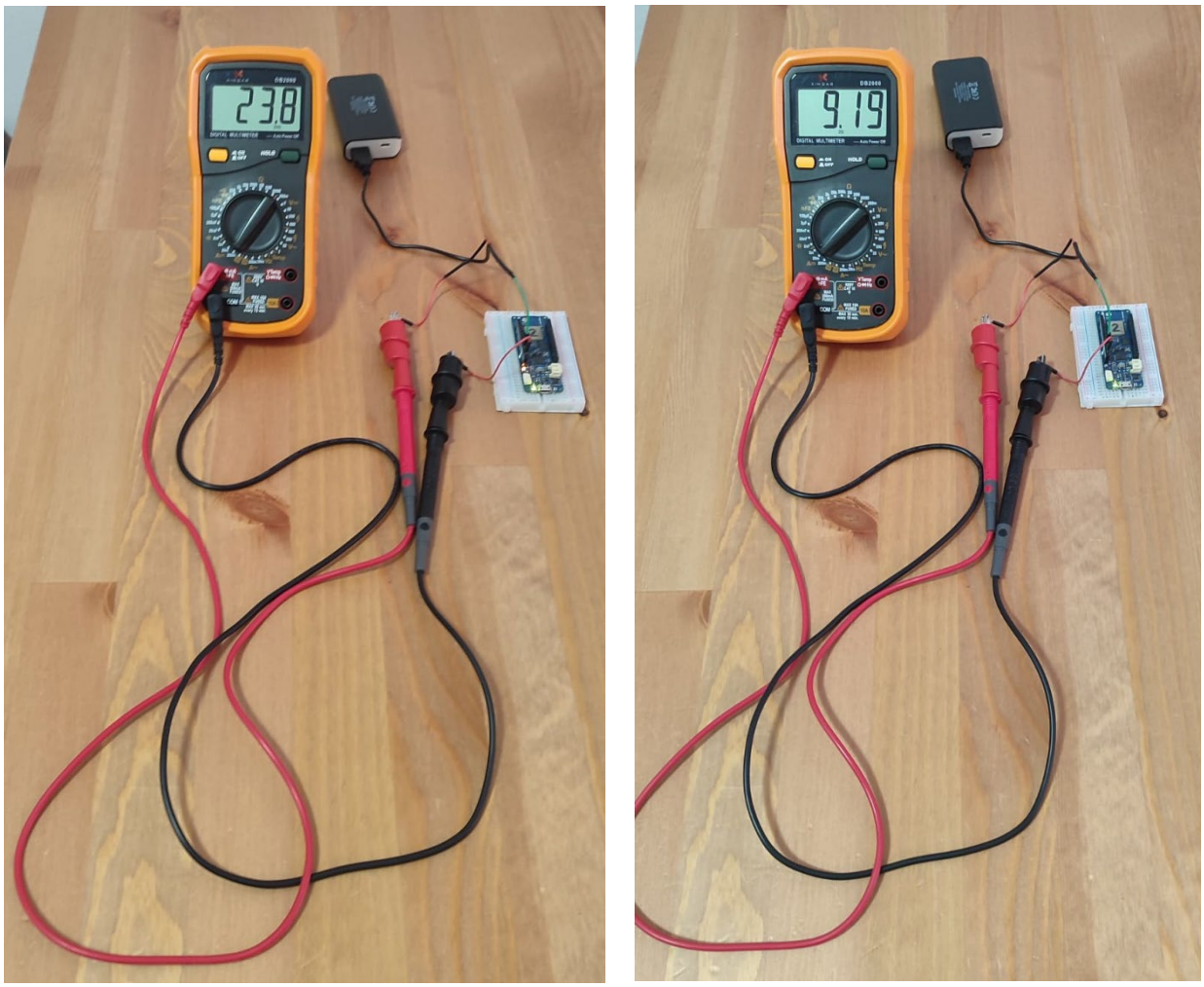


Figura 43 Medida de consumo del dispositivo Arduino MKRWAN 1310 alimentado mediante el pin V_{IN} y ejecutando el sketch "mkr1310_lowpower_v2".

Finalmente, en la Figura 44 se observan los resultados obtenidos cuando el dispositivo Arduino MKRWAN 1310 se alimenta directamente a través del conector de batería externa, obteniéndose una reducción de consumo máxima, ya sea en estado de funcionamiento normal o en modo de suspensión profunda, obteniéndose valores de 15.05mA y 0.42mA, respectivamente.

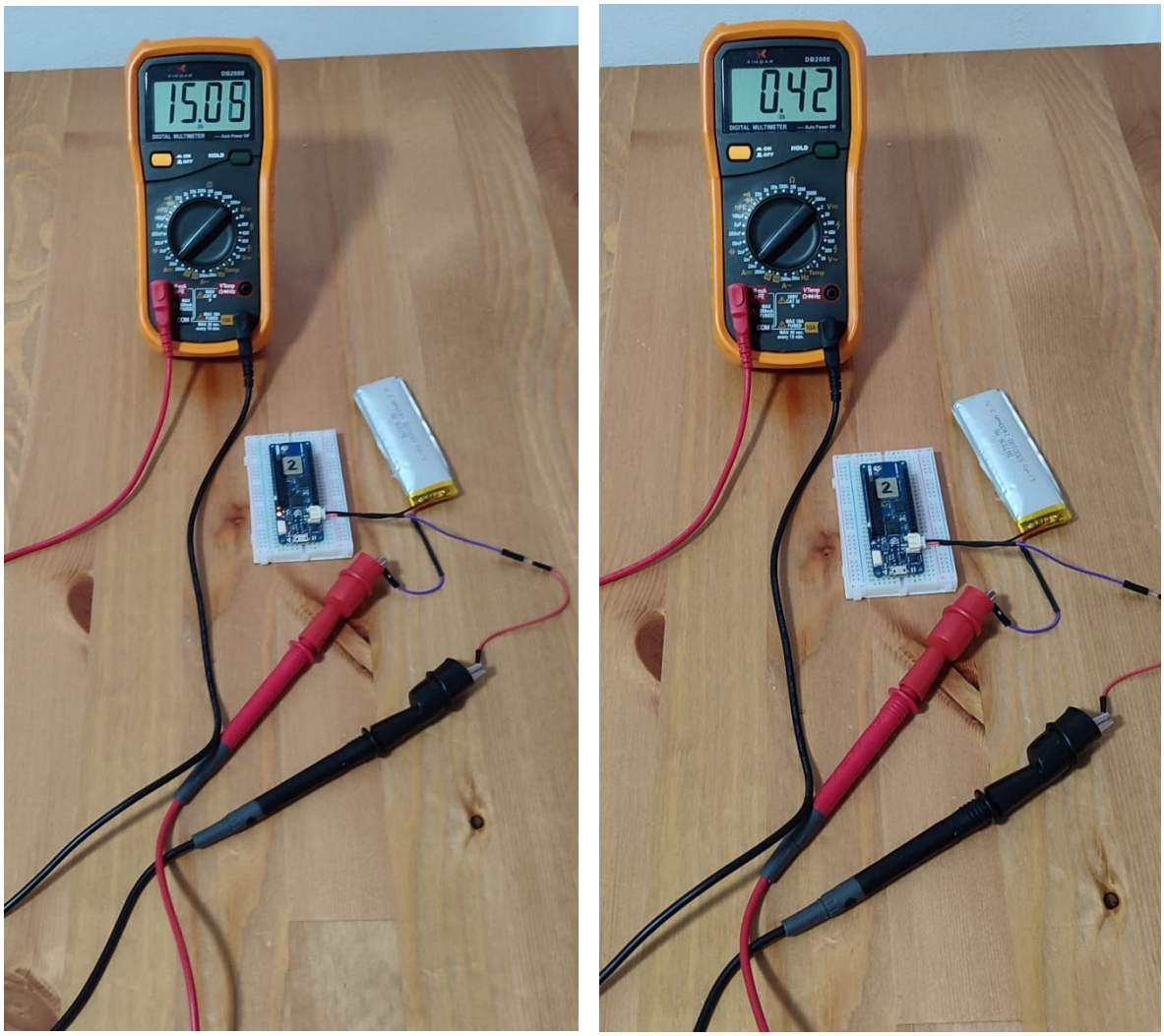


Figura 44 Medida de consumo del dispositivo Arduino MKRWAN 1310 alimentado mediante el conector de la batería y ejecutando el sketch "mkr1310_lowpower_v2".

En consecuencia, en ningún caso se ha logrado obtener el mínimo consumo especificado por el fabricante en la documentación del dispositivo Arduino MKRWAN 1310, que es de aproximadamente 104µA [35], por lo que se plantea la posibilidad de tener que realizar modificaciones hardware orientadas inicialmente a analizar el efecto de eliminar el puente de soldadura SJ1 que se muestra en la Figura 45.

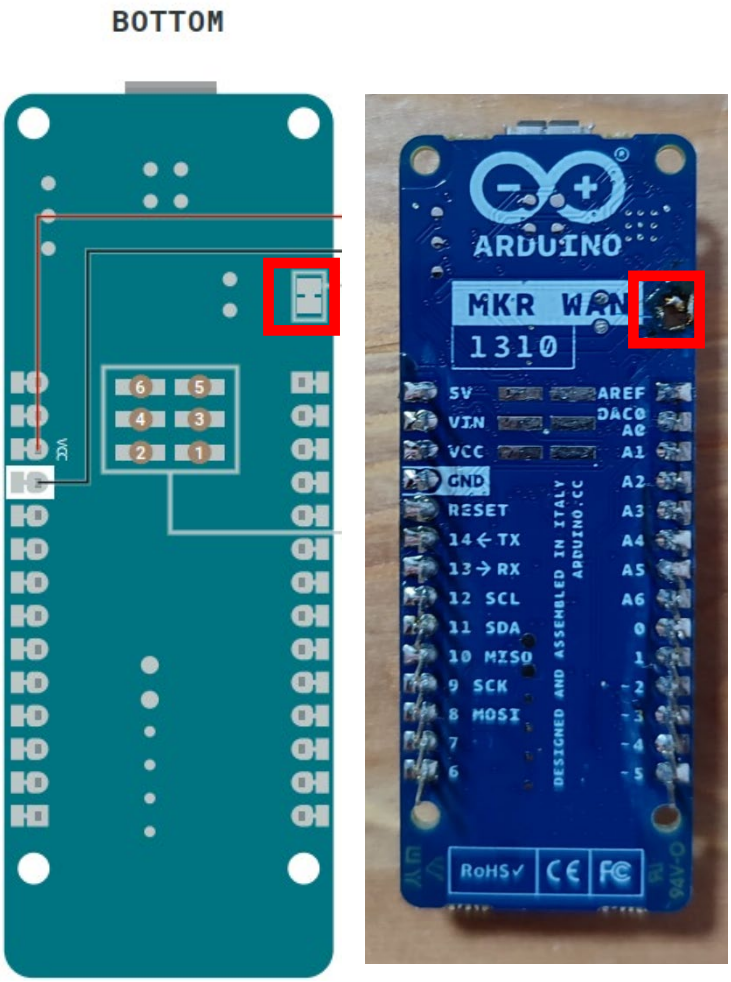


Figura 45 Eliminación de la soldadura SJ1.

El esquemático del dispositivo Arduino MKRWAN 1310, representado en la Figura 46, muestra la conexión del pin V_{IN}, la alimentación USB, y el conector de batería externa, al chip de gestión de energía BQ24195L, seguido por el regulador de tensión AP2112K. La salida de 3.3 V del regulador se conecta al bus de alimentación +3V3 mediante el puente de soldadura SJ1. Todos los demás circuitos integrados de la placa, como el ATSAMD21G18 y el transceptor LoRa®, obtienen energía del bus +3V3. Por tanto, al cortar el puente SJ1, el pin V_{IN}, la alimentación USB, y el conector de batería externa se desconectarían por completo del bus +3V3.

El propósito de este puente radica en el consumo de corriente en reposo que presentan los componentes BQ24195L y AP2112K, incluso cuando se alimenta el dispositivo directamente a través del pin V_{CC}. Para minimizar el consumo de energía, es necesario por tanto eliminar estos elementos del circuito.

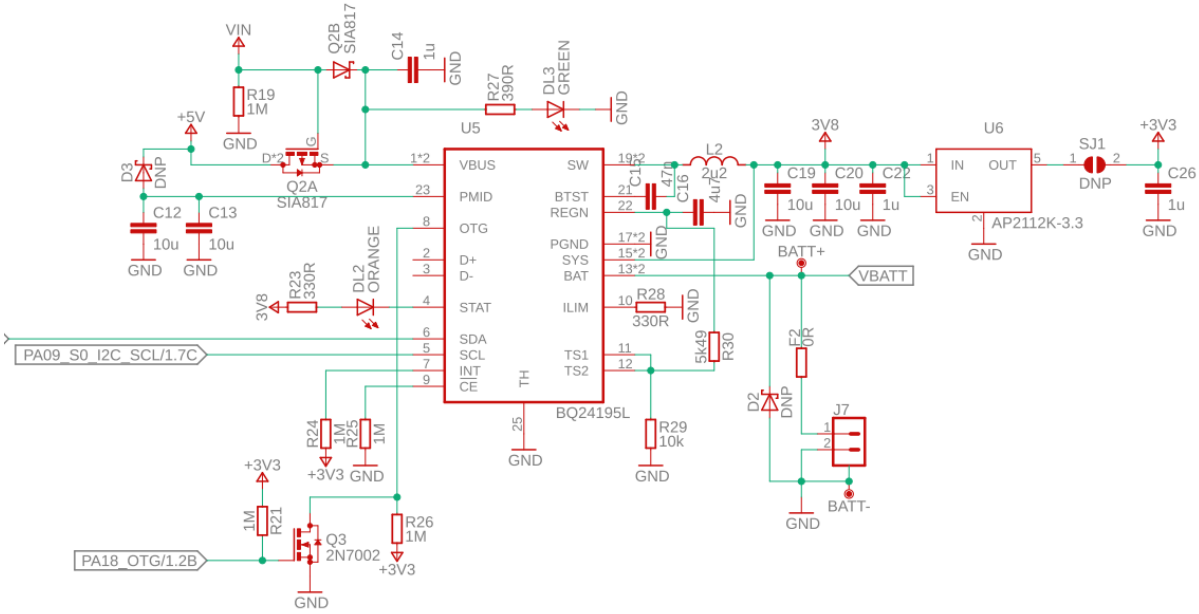


Figura 46 Esquema eléctrico del circuito MKRWAN 1310.

Tras retirar la soldadura, es imperativo proporcionar la alimentación del circuito a través de los pines VCC y GND con una consideración precisa en cuanto al nivel de tensión suministrado, ya que cualquier variación podría incidir en su funcionamiento e incluso causar daños. En la Figura 47 se muestran los pines del circuito alimentados directamente desde una batería externa.

Cuando se requiera conectar el circuito a un ordenador para, por ejemplo, realizar la programación del dispositivo Arduino MKRWAN 1310, la alimentación se debe establecer a través de los conectores VCC y GND. Esto es necesario para que el sistema sea reconocido por el puerto micro-USB, y permitir así la transferencia de datos y su programación.

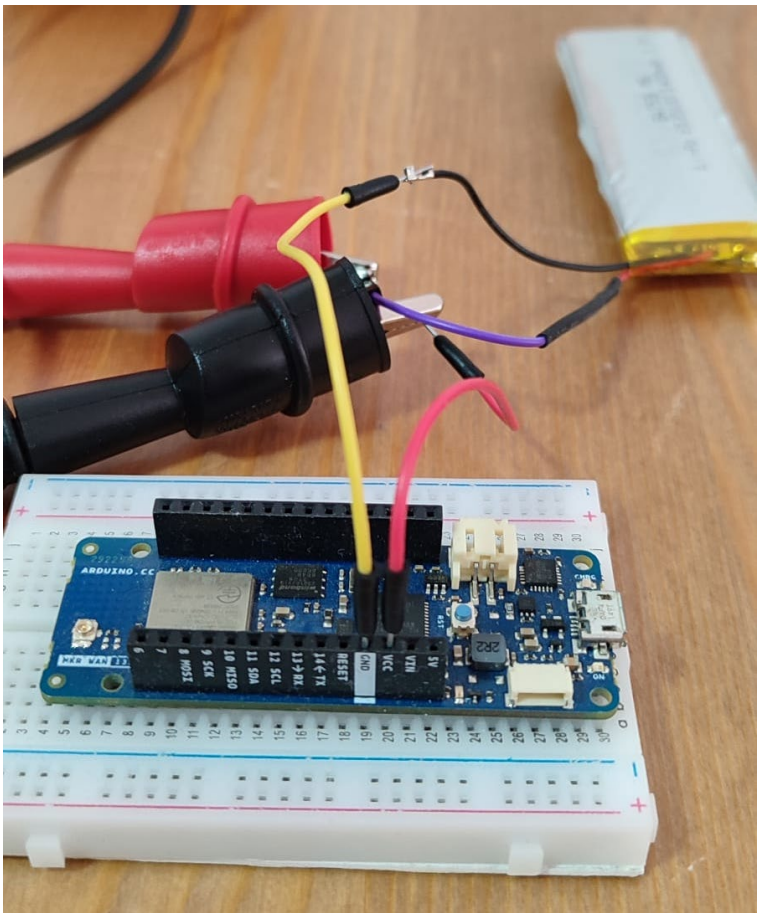


Figura 47 Alimentación del Arduino MKRWAN 1310 a través de los pines VCC y GND.

La ejecución de estas modificaciones hardware en el dispositivo Arduino MKRWAN 1310 abrió paso a la realización de nuevas mediciones de consumo de potencia, contemplando los dos estados mencionados al comienzo de esta sección. Los resultados obtenidos resultaron ser verdaderamente notables, ya que, al operar en el estado de suspensión profunda, se logró un consumo de tan solo 13 µA.

Estos resultados son especialmente destacables debido a la significativa reducción que se aprecia en el consumo de energía, lo que refleja la eficacia de las modificaciones implementadas en el dispositivo. La capacidad de lograr un consumo tan bajo en modo de suspensión profunda tiene implicaciones importantes para la vida útil de la batería y la eficiencia general del dispositivo. En la Figura 48 se muestran estos resultados.

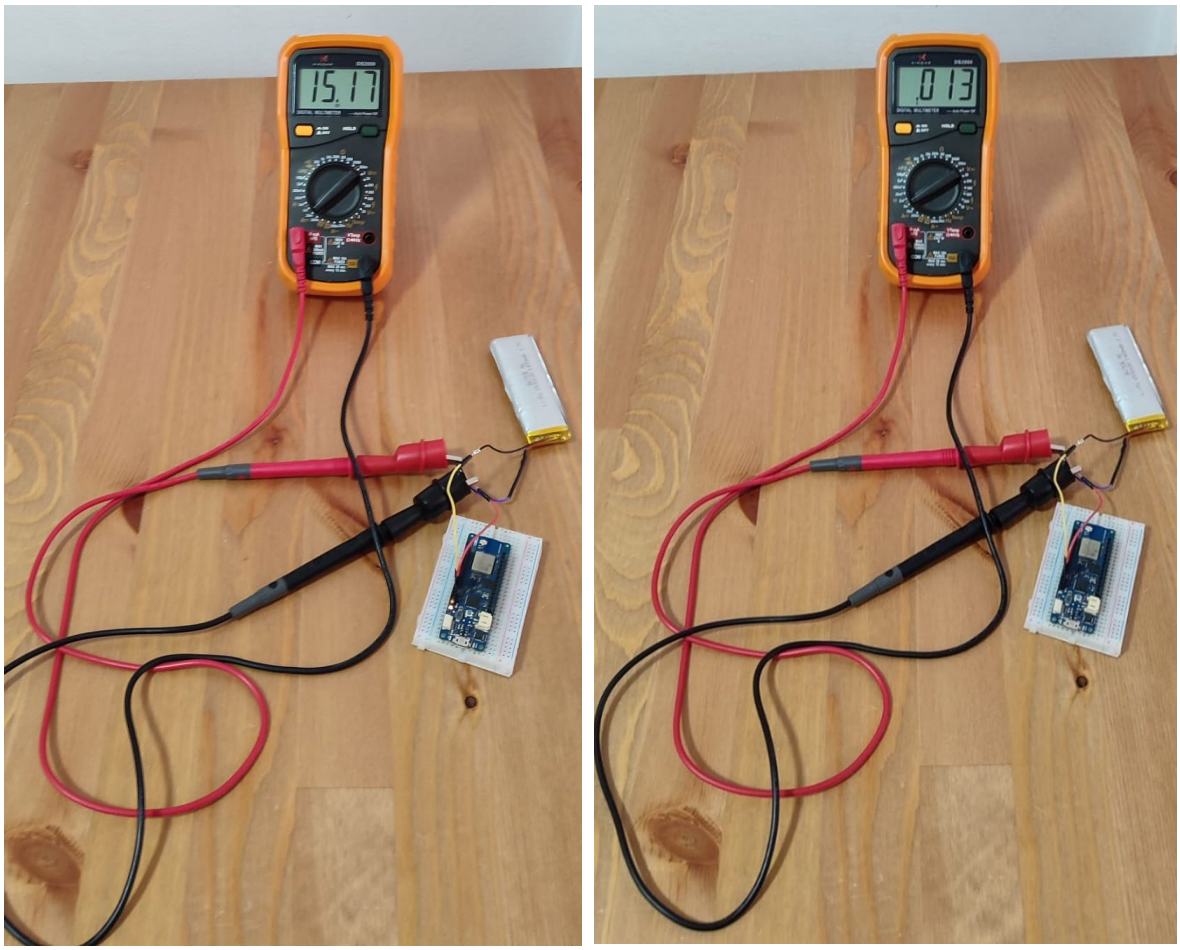


Figura 48 Medida de consumo del dispositivo Arduino MKRWAN 1310 alimentado mediante pin VCC y GND con el puente de soldadura SJ1 eliminado, ejecutando el sketch "mkr1310_lowpower_v2_crudo".

A partir de la Tabla 16, en la que se representa el consumo de energía del dispositivo Arduino MKRWAN 1310 en diferentes estados de funcionamiento y con diversas fuentes de alimentación, se pueden extraer varias conclusiones significativas.

En primer lugar, se observa que el consumo de energía en el modo de suspensión profunda es sustancialmente inferior al que presenta en funcionamiento normal, independientemente de la fuente de alimentación utilizada. Esto resalta la eficacia de la gestión de energía del dispositivo al reducir drásticamente su consumo durante los períodos de inactividad.

Además, resulta evidente que la alimentación a través de batería externa proporciona los niveles más bajos de consumo de energía en comparación con otras fuentes, tanto en funcionamiento normal como en suspensión profunda. Esto confirma que la batería es

una opción altamente eficiente desde el punto de vista energético para alimentar el dispositivo en aplicaciones que requieren de una larga duración.

La eliminación de la soldadura SJ1, que desconecta el chip de administración de energía y otros componentes, muestra una apreciable reducción en el consumo, especialmente en el modo de suspensión profunda. Esta modificación puede ser valiosa para aplicaciones sensibles al consumo de energía.

Los resultados obtenidos demuestran que el dispositivo Arduino MKRWAN 1310 es capaz de operar de manera altamente eficiente en términos de consumo de energía, especialmente en el modo de suspensión profunda, lo que lo convierte en una elección válida para aplicaciones de baja potencia y alimentación con batería. Además, la capacidad de modificar la configuración para reducir aún más el consumo puede adaptarse a escenarios específicos y requisitos de aplicaciones.

Tabla 16 Resultados medición de consumo Arduino MKRWAN 1310, ejecutando el sketch "mkr1310_lowpower_v2"

	Consumo Funcionamiento Arduino MKRWAN 1310 (mA)	
Alimentación	normal	Suspensión profunda
Micro USB	24.7	8.7
Pin VIN	23.8	9.19
Batería	15.05	0.42
Pin VCC y GND con soldadura SJ1 eliminada	15.17	0.13

5 Desarrollo de la aplicación IoT basada en LoRaWAN

Las plataformas de IoT desempeñan un papel fundamental en la gestión y conectividad de dispositivos interconectados. Estas soluciones integrales facilitan la recopilación, procesamiento y análisis de datos provenientes de sensores y dispositivos IoT, permitiendo a las organizaciones y usuarios tomar decisiones informadas. Desde la monitorización remota hasta la automatización, las plataformas de IoT ofrecen herramientas para la creación, despliegue y mantenimiento eficiente de aplicaciones IoT, impulsando la innovación en diversos sectores industriales y optimizando la eficiencia operativa.

En este sentido, la plataforma IoT *The Things Network* (TTN) representa un extenso ecosistema colaborativo a nivel global en el ámbito de IoT, donde se desarrollan redes, dispositivos y soluciones empleando la tecnología LoRaWAN. TTN ha realizado esfuerzos considerables en el desarrollo de una infraestructura de servidor respalda por una red de *Gateways* desplegados en diversas ubicaciones alrededor del mundo [36].

En este capítulo se recoge el proceso de integración del nodo final, formado por el dispositivo Arduino MKRWAN 1310 y el módulo *Person Sensor*, con la plataforma IoT TTN, como se representa en la Figura 49. Para ello, se describirán las principales características de las puertas de enlace o *Gateways* LoRaWAN utilizados en la red LoRaWAN, correspondientes a los modelos HT-M00 y RAK7268 de las empresas Heltec y *RAK Wireless*, respectivamente, así como su registro y conexión a través de Internet con la plataforma IoT mediante el uso de la tecnología WiFi.

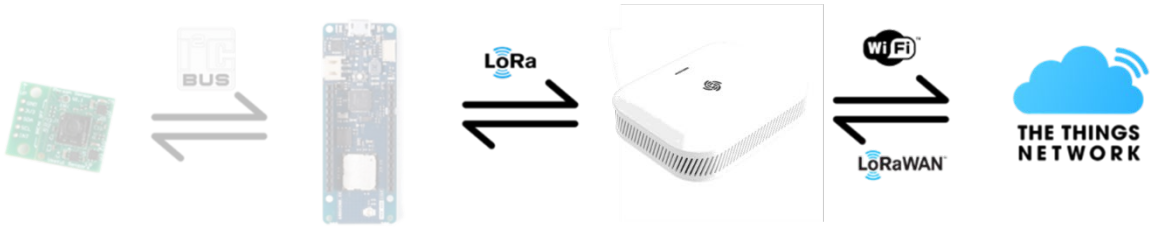


Figura 49 Arquitectura de integración del dispositivo Arduino MKRWAN 1310 y el módulo Person Sensor con la plataforma IoT TTN, con los Gateways LoRaWAN HT-M00 y RAK7268.

Este proceso completaría prácticamente la implementación de la primera de las alternativas propuestas en el desarrollo del presente TFM, quedando pendiente la integración con InfluxDB mediante el uso del protocolo MQTT, y la herramienta Grafana, procesos que ser abordarán en capítulos posteriores.

5.1 The Thing Network (TTN)

TTN proporciona una gama de capacidades de integración que abarcan tanto el protocolo HTTP como MQTT, además de ofrecer una serie de interfaces de programación de aplicaciones (API) compatibles con diversos lenguajes de programación, tales como Go, Java, Node-RED y Node.js. Estas API permiten la creación integral de aplicaciones, posibilitando la integración de nodos IoT y *Gateways* con el servidor de TTN y otras plataformas IoT para construir soluciones IoT de extremo a extremo [36].

El *backend* de TTN se encarga de gestionar aspectos críticos como la detección y la gestión de mensajes duplicados, la coordinación de mensajes de transmisión descendente, y la integración con diversas plataformas IoT, entre otras funciones clave [36].

En cuanto a la seguridad y privacidad, TTN es una red pública altamente segura que admite un cifrado extremo a extremo y soporte para diferentes claves de cifrado de 128 bits para cada dispositivo final [36].

Para acceder a TTN los usuarios necesitan disponer de una cuenta. En la red pública comunitaria, las cuentas de usuario se almacenan en un servidor específico. Estas cuentas

se identifican con un nombre de usuario o dirección de correo electrónico y se protegen mediante el uso de una contraseña. En configuraciones de redes privadas, los usuarios pueden implementar sus propios medios de autenticación y autorización [36].

Los usuarios pueden crear aplicaciones y autorizar el acceso de otros usuarios a las aplicaciones. Las aplicaciones se identifican mediante una ID de aplicación única. Cada aplicación tiene una o más llaves de acceso para acceder a los datos de la aplicación y / o administrar dispositivos [36].

5.2 The Things Stack

The Things Stack es un servidor LoRaWAN de nivel empresarial que incluye las funciones de Servidor de Red (*Network Server*) y de Servidor de Aplicaciones (*Application Server*) en la arquitectura de una red LoRaWAN. Además, *The Things Stack* contiene servicios y herramientas que permiten gestionar de forma segura millones de dispositivos LoRaWAN en producción [38].

The Things Stack ofrece las siguientes opciones para la implementación de redes LoRaWAN orientadas a proyectos comunitarios simples y pruebas locales:

- ***The Things Stack Community Edition****:* Es una versión gratuita y limitada de *The Things Stack*, sin garantías ni SLA, disponible para los usuarios de la comunidad de *The Things Network* con el fin de realizar pruebas de conceptos y proyectos comunitarios simples [38].
- ***The Things Stack Open Source*****:** Las funciones principales de *The Things Stack* son de código abierto. Tiene características limitadas y no resulta adecuado para la producción [38].

Entre estas opciones, *The Things Stack Community Edition* resulta particularmente adecuada para este TFM debido a que sus características permiten una mayor personalización y un aprendizaje profundo de los sistemas IoT sin incurrir en costes

significativos. Además, la participación en una comunidad activa proporciona recursos valiosos y soporte durante el desarrollo del proyecto.

Por otro lado, plataformas como AWS IoT Core, Microsoft Azure IoT Hub, o Google Cloud IoT Core, ofrecen una escalabilidad y seguridad superiores, además de integraciones robustas con otros servicios en la nube. Sin embargo, estos beneficios conllevan un coste mayor y una mayor complejidad, lo que puede no ser ideal para un proyecto académico con recursos limitados y un enfoque en el aprendizaje y la investigación.

5.3 Gateway Heltec HT-M00

El Gateway HT-M00 de la empresa Heltec es una puerta de enlace económica de doble canal LoRa® que utiliza una interfaz USB Tipo-C, como se observa en la Figura 50. El *Gateway* Heltec HT-M00 utiliza dos chips SX1276 basados en ESP32. La función principal del *Gateway* Heltec HT-M00 es proporcionar una red LoRaWAN para grandes espacios, o bien para cubrir las zonas sin cobertura de señal [39].

Figura 50 Heltec HT-M00 [39].

Según especifica el fabricante, para utilizar la puerta de enlace Heltec HT-M00 es conveniente modificar la longitud del preámbulo de los paquetes LoRa® en el nodo IoT a 16 bytes (por defecto es 8 bytes). En caso de que la longitud del preámbulo sea de 8

bytes, el SF mínimo y el SF máximo deben ser iguales, o bien solo se recibirán los paquetes transmitidos con SF mínimo. Por ejemplo, si la longitud del preámbulo del nodo es de 8 bytes y la puerta de enlace establece un SF mínimo de 7 y un SF máximo de 12, se recibirán únicamente paquetes con SF7 [39].

5.3.1 Configuración del Gateway Heltec HT-M00

En el proceso de producción de la puerta de enlace Heltec HT-M00 se ha preinstalado un *firmware* que permite su uso inmediato [39].

5.3.1.1 Versión firmware V2.0 y superiores

Tras activar la pasarela mediante el uso de un cable de datos USB Tipo-C, se requiere mantener pulsado el botón identificado como "CFG" y, posteriormente, presionar el botón designado como "RST", mostrándose la ubicación de estos botones en la Figura 51. A continuación, se procederá a liberar el botón "RST". Una vez que la puerta de enlace acceda a la interfaz, tal como se ilustra en la Figura 52, se procederá a soltar el botón "CFG" [40].

Figura 51 Ubicación de los botonesde configuración en el gateway HT-M00 [40].

Figura 52 Menú de información del Gateway HT-M00 [40].

A partir de este instante, se procederá a localizar la red WiFi inalámbrica denominada `M00_XXXX` con el propósito de establecer una conexión mediante la clave de acceso `heltec.org`. Acto seguido, en el dispositivo conectado a este punto de acceso WiFi se introducirá la dirección IP `192.168.4.1` en un navegador web, lo que permitirá el acceso a la interfaz de configuración de la puerta de enlace [40].

En la interfaz, mostrada en la Figura 53, se procede a establecer los parámetros de configuración del *Gateway* Heltec HT-M00. Estos ajustes incluyen la configuración del punto de acceso Wi-Fi necesario para la conectividad a Internet, la identificación (ID) del *Gateway*, la selección de la frecuencia de los canales de la puerta de enlace, los valores mínimo y máximo de SF (*Spreading Factor*), la especificación de la dirección del Servidor de Red LoRaWAN, correspondiente en este caso al Servidor de Red de la plataforma TTN, junto con su correspondiente puerto de acceso, el periodo para mantener activa la conexión, y la configuración de la zona horaria, procediendo finalmente a aplicar la configuración establecida, mediante la opción *Submit*. Simultáneamente, se posibilita la actualización del *firmware* de la puerta de enlace HT-M00, y se efectúa la actualización mediante la selección de la función *Firmware Update* [40].

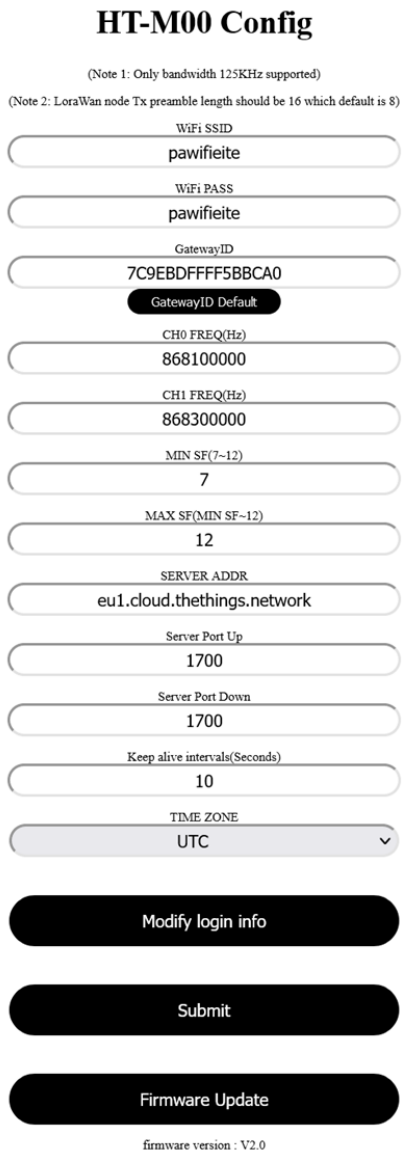


Figura 53 Interfaz web de configuración del gateway HT-M00 en versiones firmware V2.0 y superiores[40] .

Es importante mencionar que, tras la completar el proceso de configuración, la puerta de enlace experimentará un reinicio. Una vez reiniciada, establecerá de forma automática la conexión con la red Wi-Fi previamente configurada. En el caso de que la conexión no se establezca satisfactoriamente, la puerta de enlace se reiniciará de manera reiterada hasta que se logre establecer la conexión, proceso que se muestra en la Figura 54.

Figura 54 Proceso de inicializacion y conexión del gateway HT-M00 [40][

5.3.2 Información del Gateway

Una vez que se logra establecer con éxito la conexión con el punto de acceso WiFi configurado, la puerta de enlace Heltec HT-M00 mostrará en su pantalla integrada la interfaz que se observa en la Figura 55.

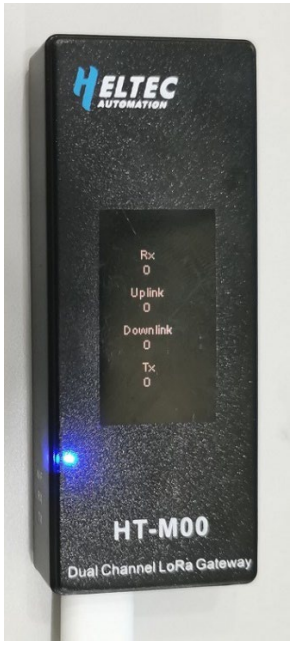


Figura 55 Interfaz de estado de los enlaces del gateway HT-M00 [40].

Al presionar el botón denominado "STA", que se muestra en la Figura 56, es posible modificar el contenido que se muestra en la pantalla, como se observa en la Figura 57. En esta visualización se pueden observar detalles tales como la hora actual, el registro de los tiempos de envío y recepción más recientes, la identificación (ID) de la puerta de enlace, la dirección del Servidor de Red, o la frecuencia del canal, entre otros. En la versión del firmware V2.0 y posteriores, también se proporciona información relativa a la IP local, que permite un acceso directo a la interfaz de configuración desde un dispositivo conectado a la misma red local que la puerta de enlace.

Figura 56 Ubicación boton STA del gateway HT-M00 [40].

Figura 57 Información del gateway HT-M00 a través de la interfaz de estado [40].

5.3.3 Registro del Gateway HT-M00 en la plataforma TTN

El registro de una puerta de enlace es un requisito esencial para la transmisión de datos desde/hacia nodos IoT en una red LoRaWAN basada en la plataforma *The Things Network*. Este proceso comienza con la selección de un plan dentro de las opciones proporcionadas por TTN, contemplando tanto alternativas particulares como empresariales, como se representa en la Figura 58 y la Figura 59, con el propósito de adecuarse a diversos requisitos.

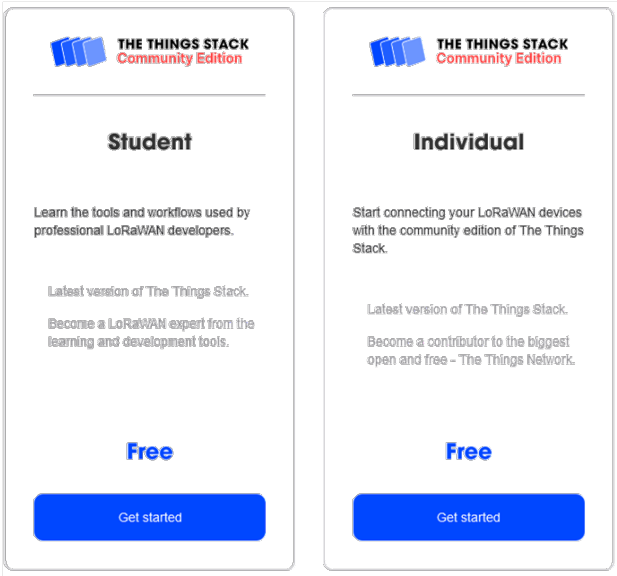


Figura 58 Planes de suscripción a la plataforma TTN [41].

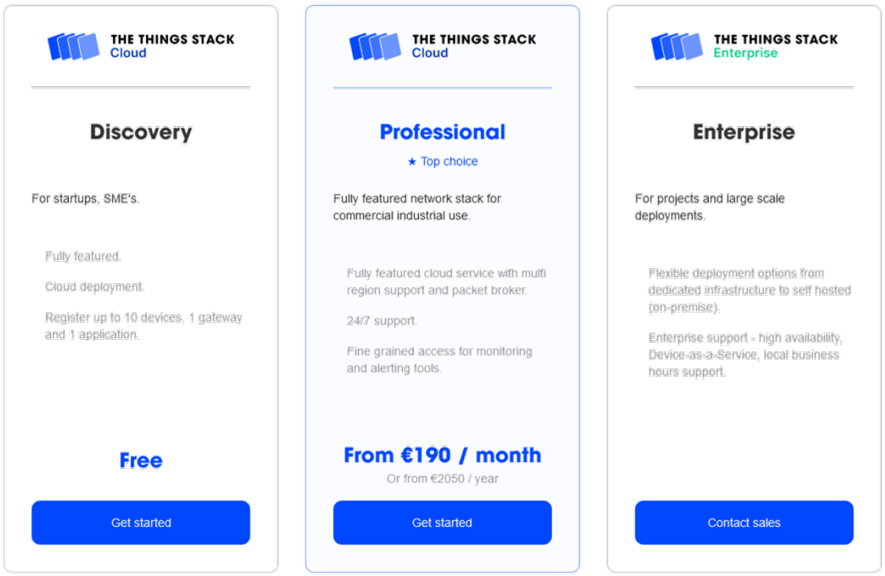


Figura 59 Planes de suscripción a la plataforma TTN [41].

En el caso concreto del desarrollo del presente TFM, se accederá a los servicios gratuitos que ofrece la plataforma TTN a los usuarios de una cuenta Individual. Para registrarse es necesario completar el formulario disponible en la página web https://console.cloud.thethings.network/, representado en la Figura 60, tras haber seleccionado el clúster correspondiente a la región Europe 1 (eu1), como se indica en la Figura 61.

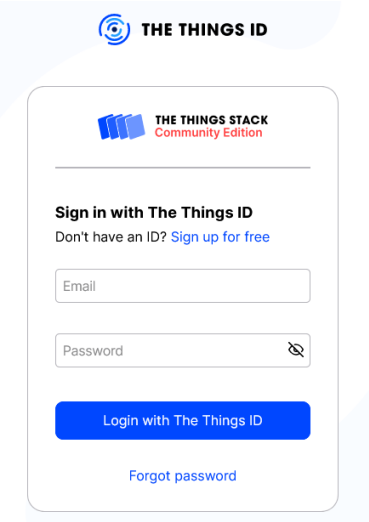


Figura 60 Página Web de inicio para registrarse o iniciar sesión en la plataforma TTN [41].

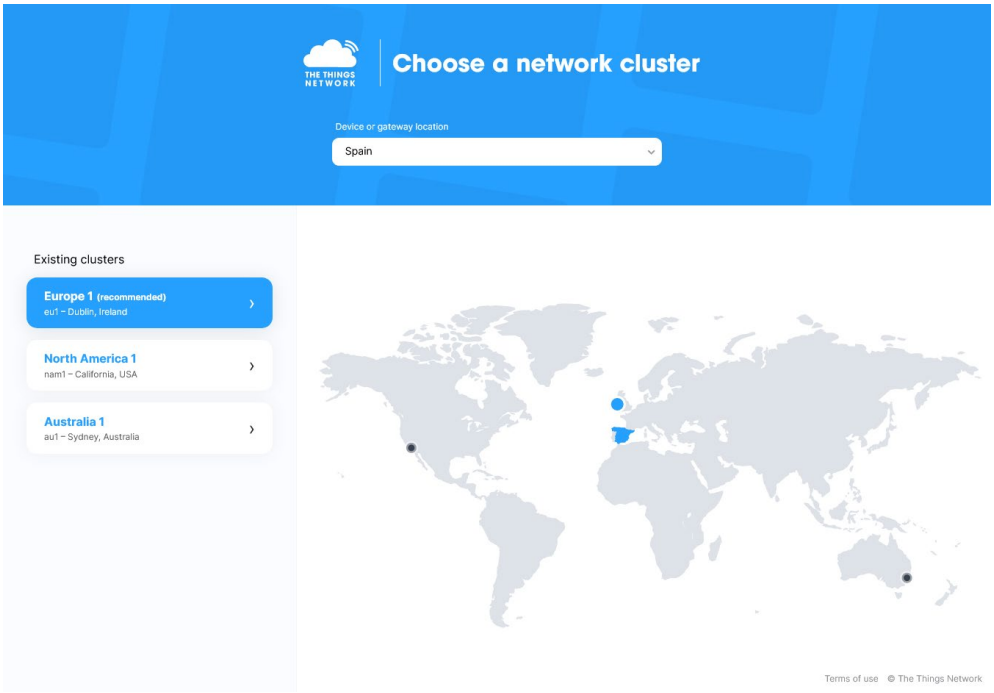


Figura 61 Network Clusters de la plataforma TTN [41].

Una vez completado el registro como usuario, se podrá acceder a la cuenta creada desde la página principal de TTN a través de la opción *Log In* directamente desde la consola asociada a la cuenta TTN https://console.cloud.thethings.network, mostrada en la Figura 62, o bien directamente en el clúster Europe1, a través del enlace https://eu1.cloud.thethings.network/console.

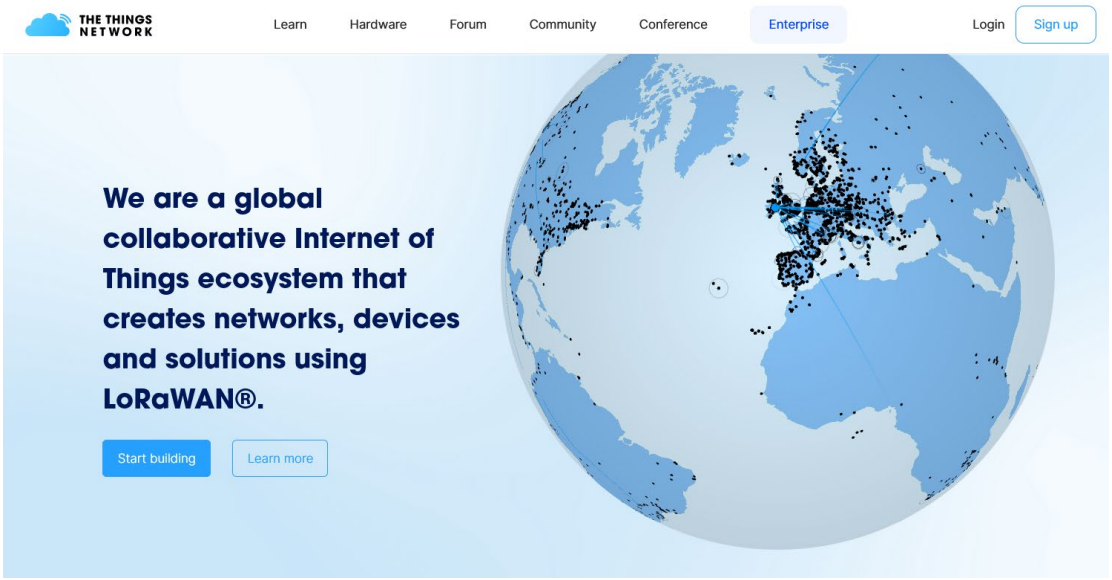


Figura 62 Página Web de Inicio de la plataforma TTN [41].

La consola de la plataforma TTN proporciona esencialmente dos opciones, representadas en la Figura 63. En primer lugar, posibilita a los usuarios la creación y gestión de aplicaciones a través de las cuales se puede supervisar y dirigir el intercambio de datos entre los diversos módulos de la red LoRaWAN, permitiendo además la integración con otras plataformas. En segundo lugar, permite el registro y la administración del uso de dispositivos comerciales que actúen como puerta de enlace en la red LoRaWAN, en situaciones en las que no resulte viable hacer uso de los *Gateways* integrados actualmente en TTN.

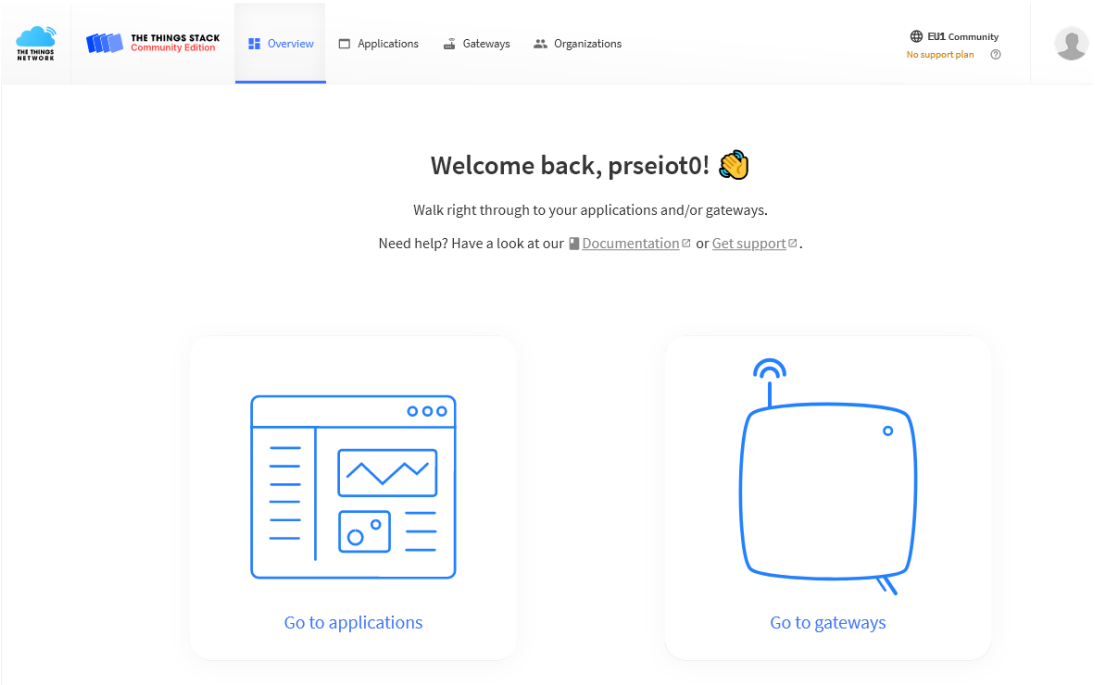


Figura 63 Consola de plataforma TTN [41].

En el proceso de registro de una puerta de enlace, se procede en primer lugar a su identificación mediante el uso del parámetro *Extended Unique Identifier* (EUI) específico para cada dispositivo, como se observa en la Figura 64.

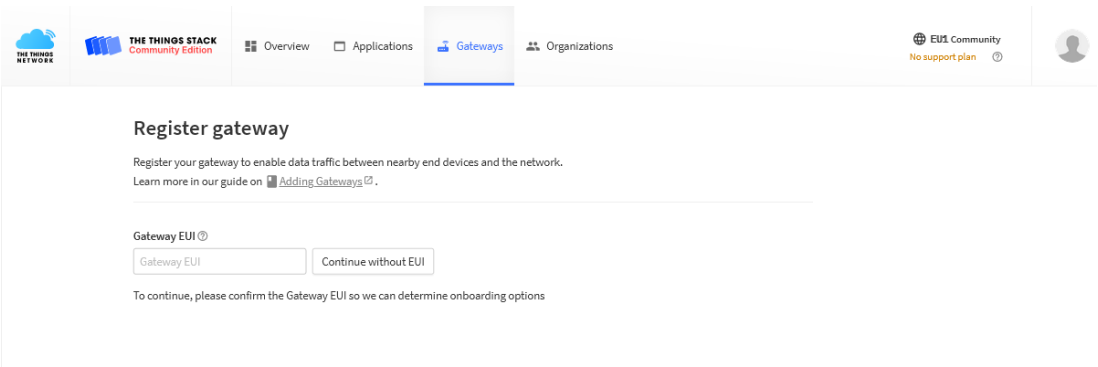


Figura 64 Registro del Gateway HT-M00.

En el contexto de la puerta de enlace Heltec HT-M00, resulta imperativo establecer una conexión serie con el propósito de obtener el valor del parámetro *Gateway EUI*. En la Figura 67 se destacan los parámetros de la conexión serie establecida con la puerta de

enlace mediante el uso de la aplicación *Putty*, siendo el valor del parámetro *Gateway EUI* el correspondiente a `GateWayID`.

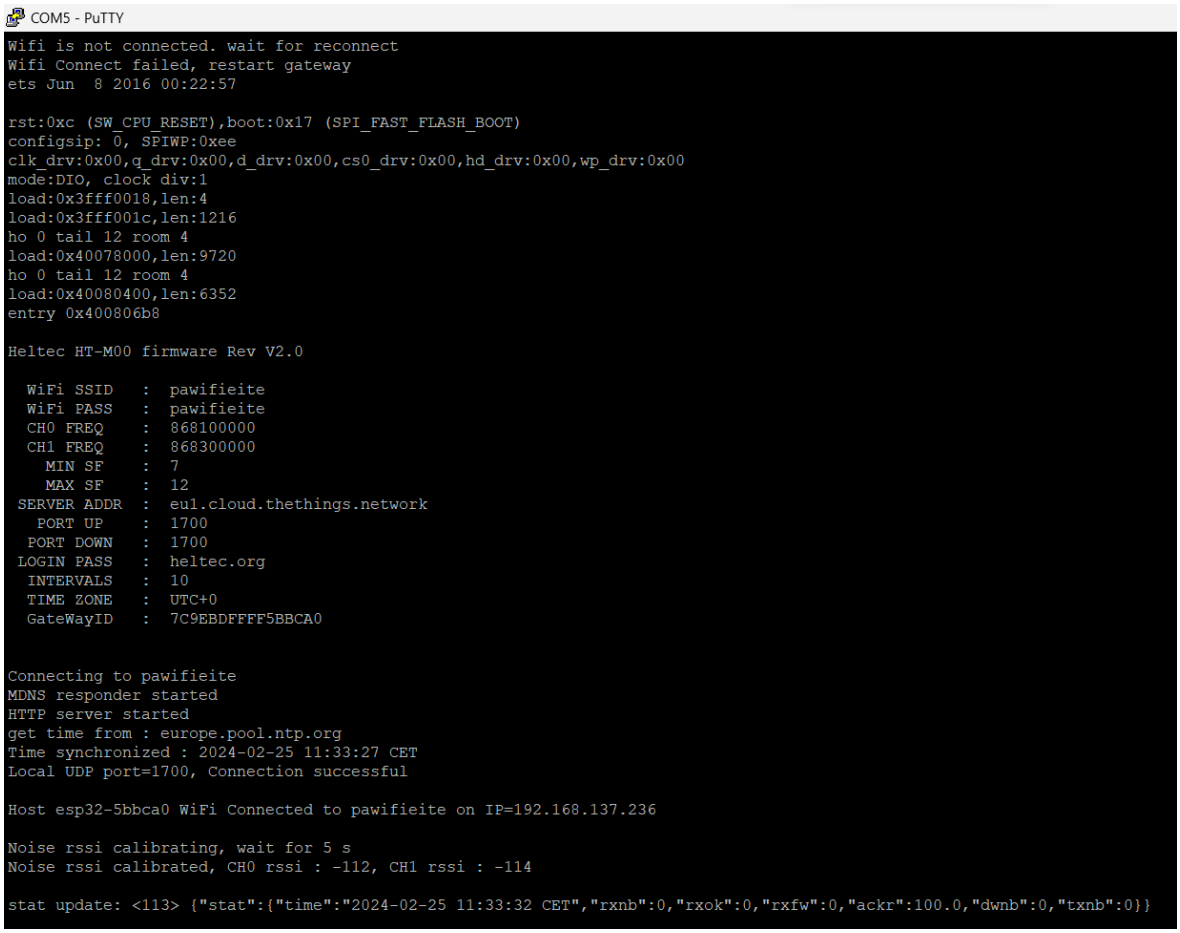


Figura 65 EUI del dispositivo Gateway HTM-00

Además del EUI, en la plataforma TTN se genera automáticamente el parámetro *Gateway ID*, que se corresponderá con la identificación única del *Gateway* en la red LoRaWAN, si bien puede ser modificado en caso necesario. Cabe destacar que este identificador debe contener únicamente letras minúsculas, números y guiones. Asimismo, se proporciona la opción de asignar un nombre al *Gateway*. En cuanto al parámetro *Frequency Plan*, se refiere al plan de frecuencias utilizado por la puerta de enlace en función de su ubicación. En la Figura 66 se puede observar el valor de los parámetros establecidos en estos campos, mientras que en la Figura 67 el *Gateway* correctamente registrado en la plataforma TTN.

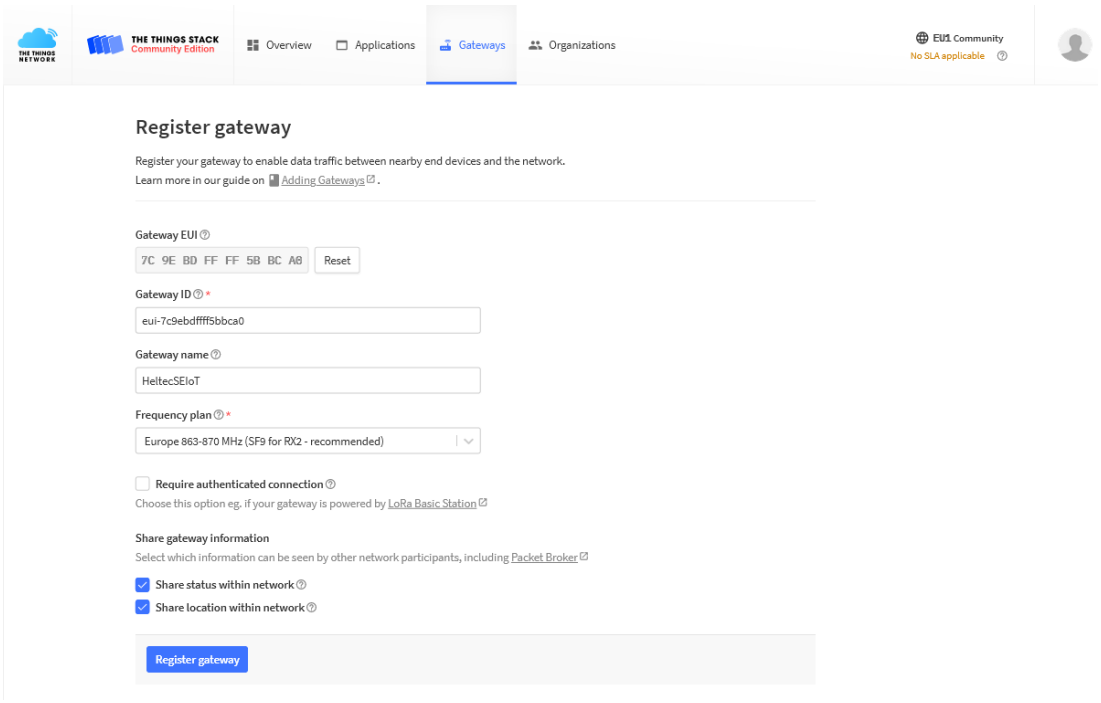


Figura 66 Relleno de los campos de registro del Gateway en TTN.

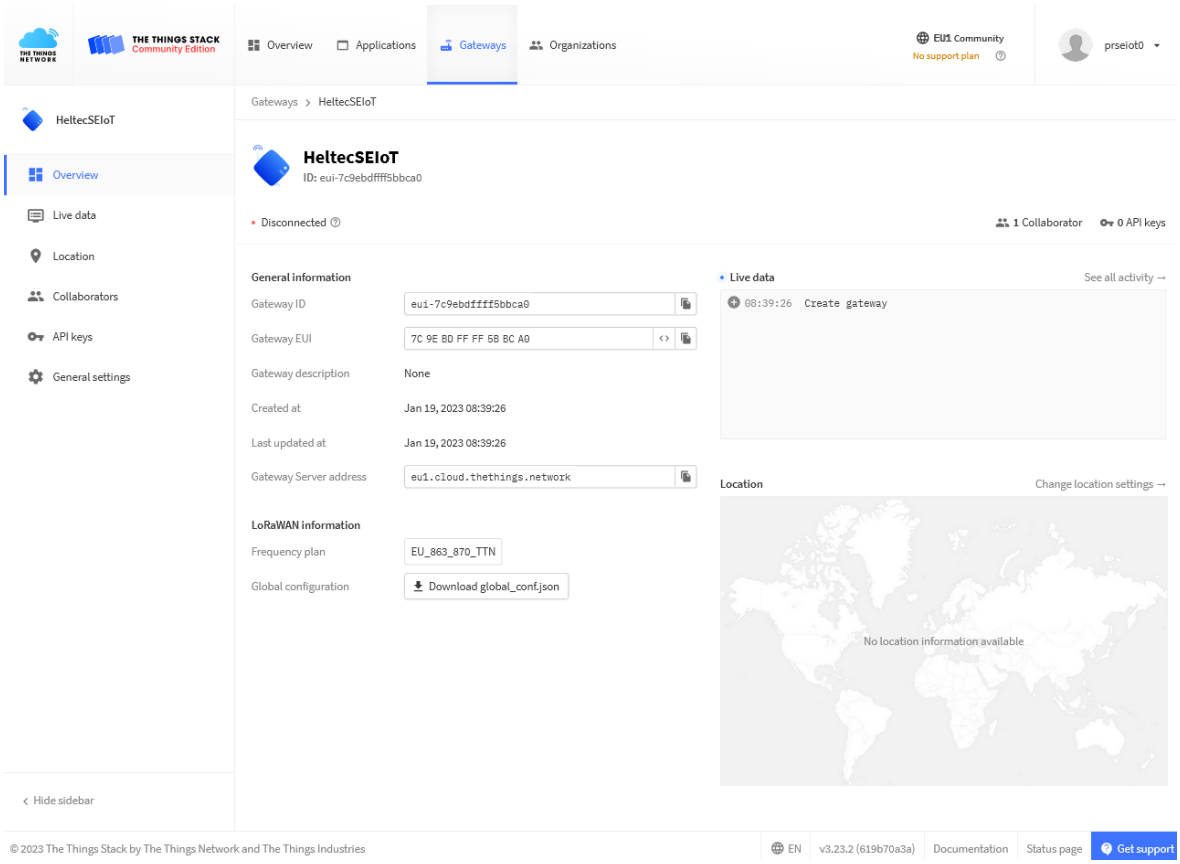


Figura 67 Gateway HT-M00 registrado en TTN.

La culminación exitosa del proceso de registro del *Gateway* implica la definición precisa de su ubicación geográfica en la plataforma TTN. Este paso garantiza que el dispositivo pueda ser identificado por otros usuarios que busquen transmitir y recibir información a través de él. En la Figura 68 y la Figura 69 se ilustra este procedimiento, ofreciendo una representación visual del contexto geográfico asignado a la puerta de enlace registrada.

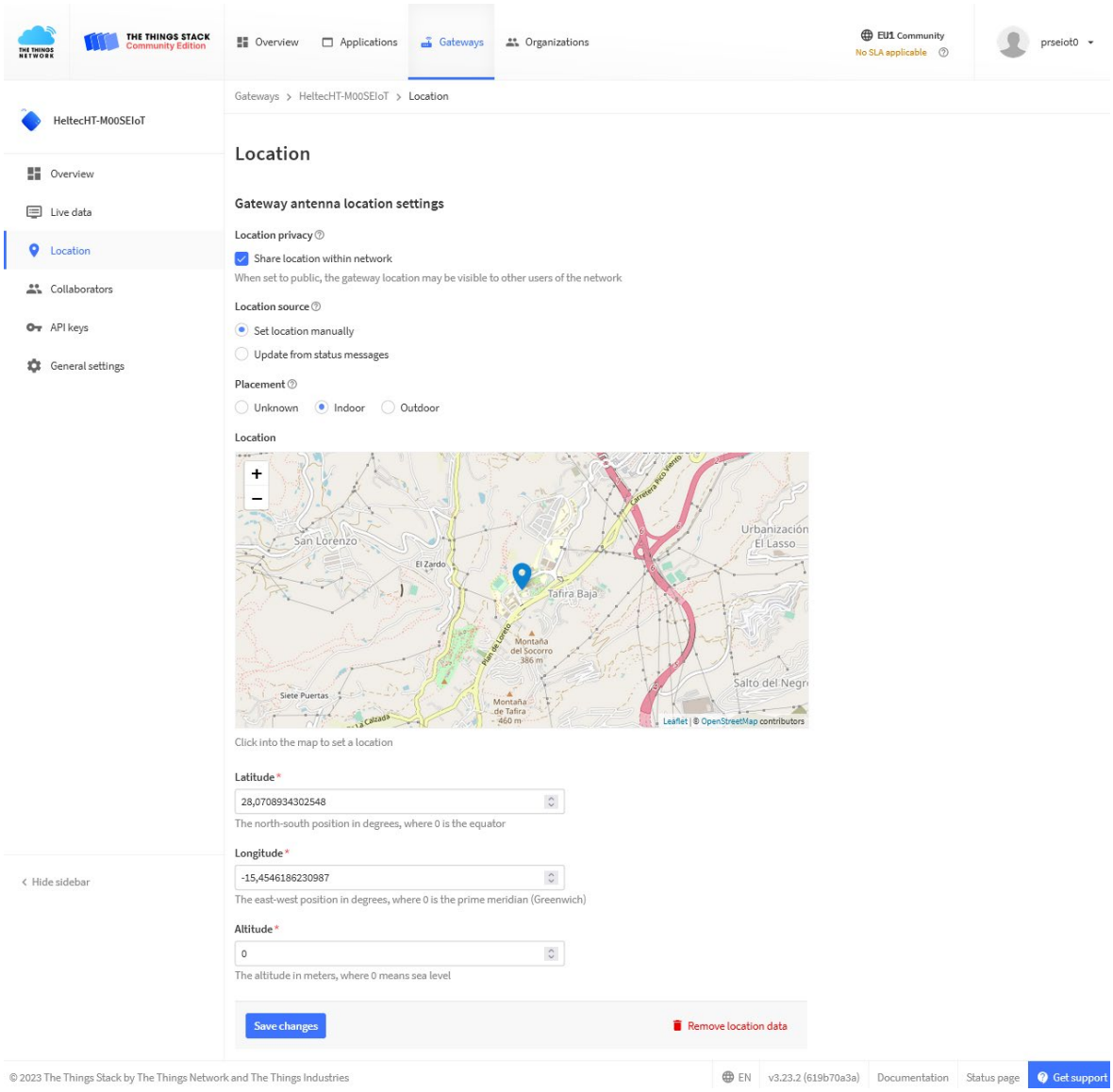


Figura 68 Ubicación física del Gateway HT-M00.

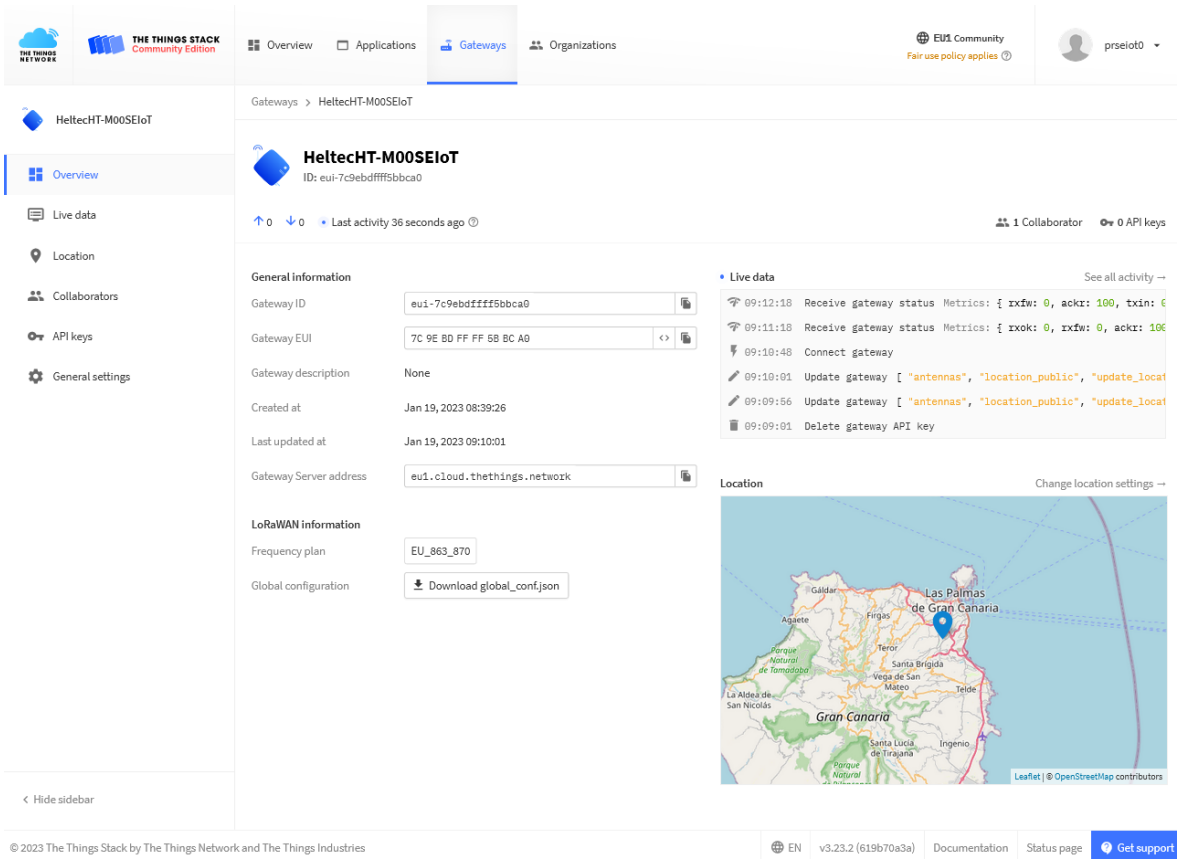


Figura 69 Estado del Gateway HT-M00 de forma generalizada.

5.4 Creación de una aplicación TTN basada en el dispositivo Arduino MKRWAN 1310

Para crear una aplicación en la consola de TTN, se selecciona inicialmente la pestaña *Go to Application* y se elige la opción *Create Application*. Posteriormente, se proporciona un identificador de aplicación único (*Application ID*), y se indica un nombre para la aplicación, como se observa en la Figura 70.

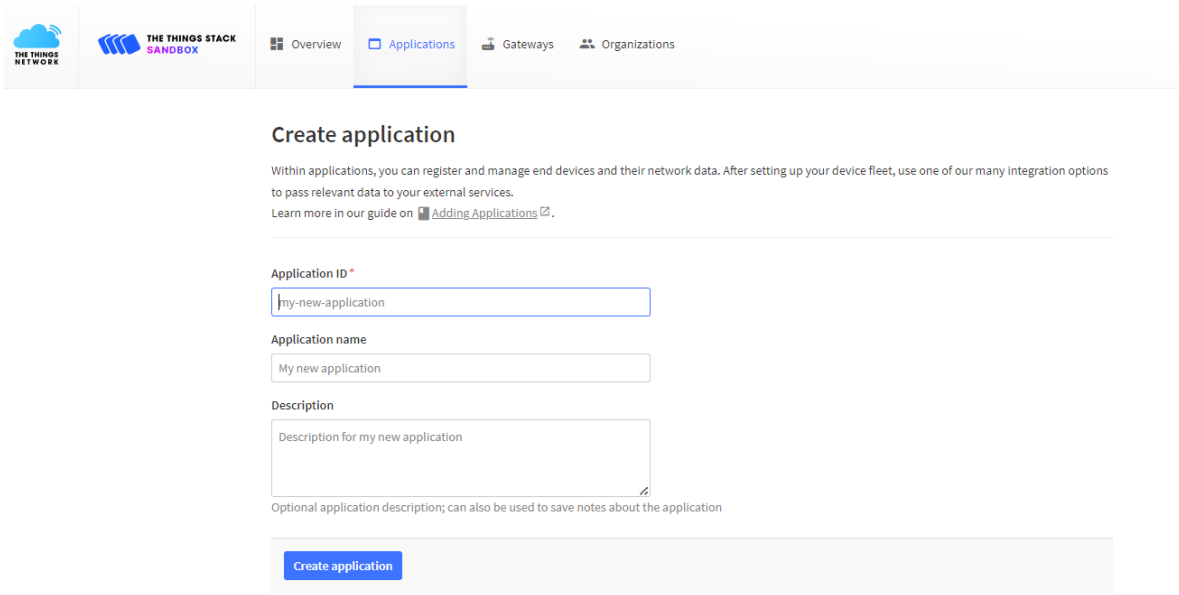


Figura 70 Creación de la aplicación en TTN.

En este caso, la denominación inicial de la aplicación desarrollada es `Sensor_Person_consumption`, estableciéndose como *Application ID test-v1-23-03-2023*, como se ilustra en la Figura 71.

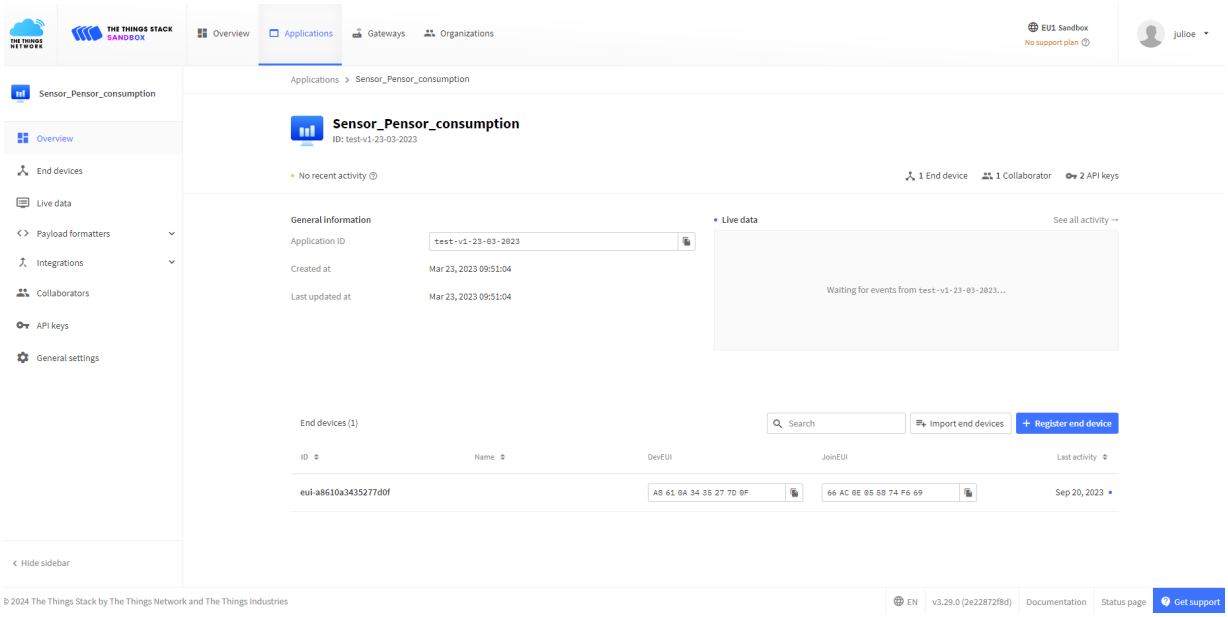


Figura 71 Parámetros de la aplicación.

Una vez creada la aplicación, es posible asociar a ella dispositivos específicos, lo que facilita la administración y visualización de los datos procedentes de los nodos IoT conectados a la red LoRaWAN. Este procedimiento resulta fundamental para establecer una interfaz de gestión eficiente, posibilitando así la integración fluida de dispositivos en la plataforma TTN.

5.4.1 Integración de nodos IoT basados en dispositivos Arduino en redes LoRaWAN

5.4.1.1 Biblioteca MKRWAN

Entre las bibliotecas oficiales proporcionadas por Arduino para aprovechar las capacidades de conectividad de los módulos radio incorporados en diversas placas de desarrollo IoT, como Wi-Fi, Bluetooth®, LoRa®, GSM, NB-IoT y Sigfox, destaca la biblioteca MKRWAN. Diseñada como soporte específico para los dispositivos Arduino MKRWAN 1300/1310, esta biblioteca ofrece interfaces de programación de aplicaciones (API) que posibilitan la comunicación basada en la tecnología LoRa® y redes LoRaWAN. A partir del uso de esta biblioteca, es posible utilizar eficazmente el transceptor LoRa® Murata CMWX1ZZABZ integrado en la placa Arduino MKRWAN 1310. [31]. Para utilizar esta biblioteca se debe incluir en el código del *sketch* la sentencia `#include <MKRWAN.h>`

Esta biblioteca es compatible con las arquitecturas SAMD, MBED, y MBED_PORTENTA, por lo que es posible utilizarla en las siguientes placas Arduino:

- Arduino MKRFOX 1200.
- Arduino MKRGSM 1400.
- Arduino MKRNB 1500.
- Arduino MKRVIDOR 4000.
- Arduino MKRWAN 1300 (*LoRa connectivity*).
- **Arduino MKRWAN 1310.**
- Arduino MKRWI-FI 1010.
- Arduino MKRZERO (*I_2S bus & SD for sound, music & digital audio data*).
- Arduino MKR1000 WI-FI.
- Arduino Nano 33 IoT.
- Arduino Zero.

- Portenta H7.

La biblioteca MKRWAN de Arduino proporciona varios métodos y funciones para facilitar la integración de nodos IoT basados en placas de desarrollo como MKRWAN 1300/1310, en redes LoRaWAN, entre los que destacan los siguientes:

- `begin()`
- `restart()`
- `version()`
- `deviceEUI()`
- `joinOTAA()`
- `joinABP()`
- `beginPacket()`
- `write()`
- `read()`
- `print()`
- `peek()`
- `available()`
- `parsePacket()`
- `endPacket()`
- `modem.configureBand()`

En epígrafes posteriores se explicarán con más detalle algunos de los métodos utilizados en el código de los *sketches* creados en el desarrollo del presente TFM para la integración del nodo IoT basado en el dispositivo Arduino MKRWAN 1310, en la plataforma TTN.

5.4.2 Registro del dispositivo Arduino MKRWAN 1310 en una aplicación TTN

Tras la creación de la aplicación en la plataforma TTN, el siguiente paso en la creación de la red LoRaWAN conlleva el registro del dispositivo Arduino MKRWAN 1310 que actuará como nodo IoT. Estos módulos tienen la capacidad de transmitir y recibir datos hacia y

desde las puertas de enlace de la red LoRaWAN en comunicaciones *uplink* y *downlink*, respectivamente.

Para ello, en el proceso de registro del dispositivo que actuará como nodo IoT en la aplicación creada en TTN, se selecciona el dispositivo Arduino MKRWAN 1310 entre los disponibles en el repositorio de la plataforma, así como el modelo, la versión de hardware y *firmware*, y el plan de frecuencias, que el caso de Europa se encuentra entre 863 MHz y 870MHz, como puede observar en la Figura 72.

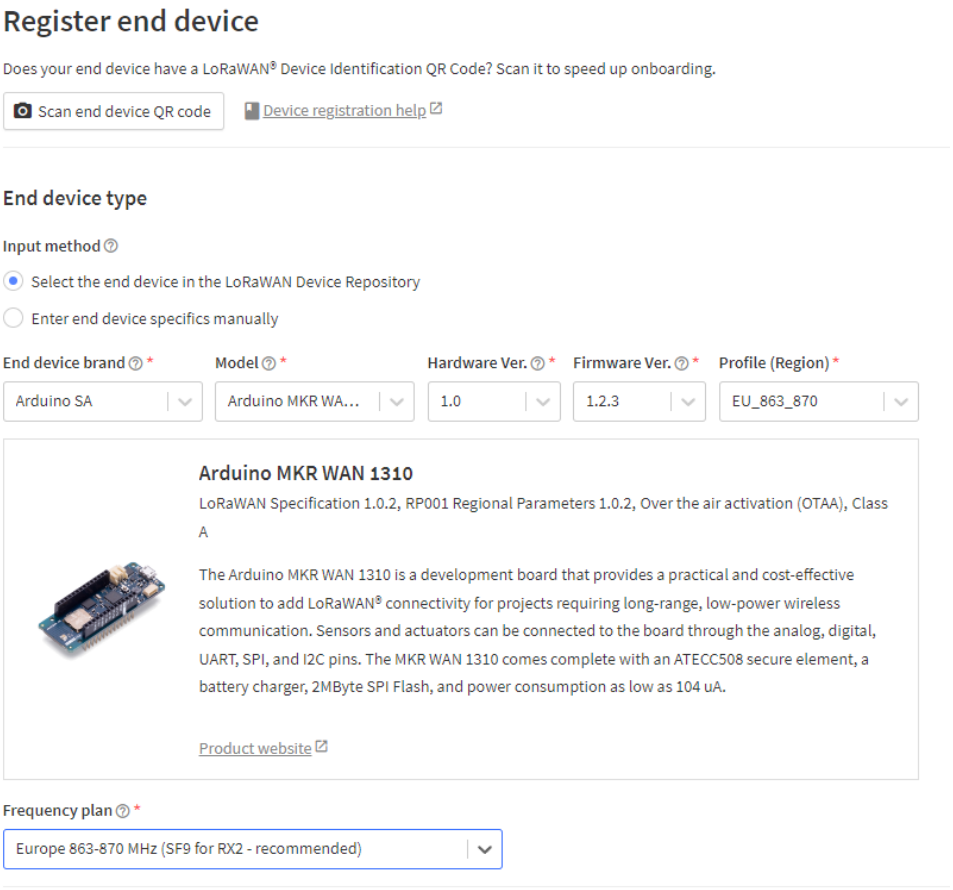


Figura 72 Registro del dispositivo Arduino MKRWAN 1310.

A continuación, se cumplimentan los campos que aparecen en la Figura 73 con los parámetros que se indican en cada caso:

- **JoinEUI** (*AppEUI - Application Extended Unique Identifier*): Identificador único de 64 bits del Servidor de Aplicaciones y es el destino de los datos enviados por los dispositivos finales. Debe ser único para que los dispositivos finales sepan a dónde

enviar sus paquetes. El parámetro *JoinEUI* se puede generar utilizando, entre otras opciones, el generador de EUI aleatorios disponible en https://descartes.co.uk/CreateEUIKey.html

- **DevEUI** (*Device Extended Unique Identifier*): Identificador único de 64 bits del dispositivo final, asignado por el fabricante. La asignación de un *DevEUI* requiere disponer de un OUI (*Organizational Unique Identifier*) de la Autoridad de Registro de IEEE. El identificador DevEUI se obtiene del dispositivo Arduino MKRWAN 1310, a partir de la ejecución de un *sketch* específico proporcionado con la biblioteca MKRWAN.
- **AppKey:** Para generar una clave de aplicación para este dispositivo, simplemente es necesario hacer clic en el botón de generación adyacente al campo *AppKey*, como se muestra en la Figura 73. Este parámetro es esencial para el cifrado de datos, utilizándose para codificar los mensajes entre los nodos finales y el Servidor de Aplicaciones. Por lo general, esta clave consiste en un número generado aleatoriamente, el cual se programa en los nodos finales y se comparte con el Servidor de Aplicaciones, posibilitando así la decodificación efectiva de los mensajes.

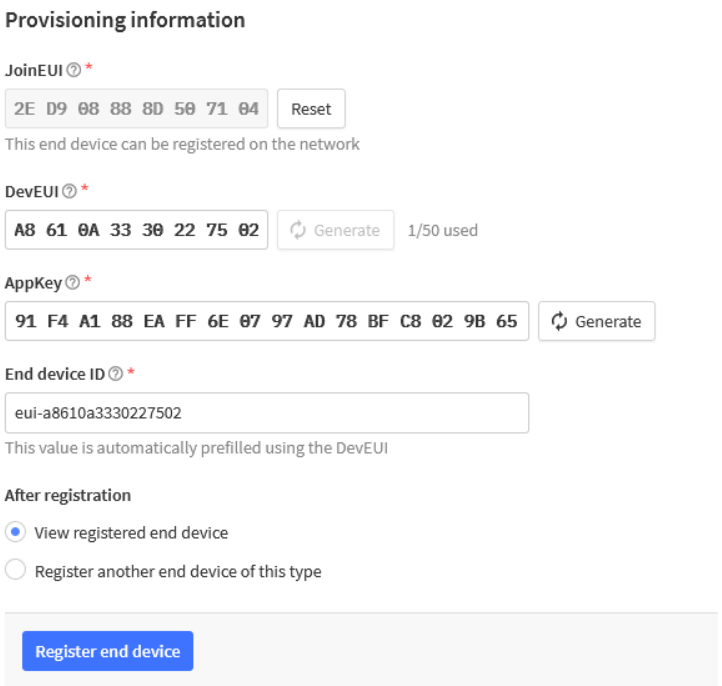


Figura 73 Campos necesarios para registrar el dispositivo final.

Para obtener el identificador *DevEUI* del módulo Arduino MKRWAN 1310, se deben llevar a cabo los siguientes pasos: en primer lugar, abrir en el entorno Arduino IDE el *sketch* de referencia `FirstConfiguration.ino`, disponible en la ruta `File > Examples > MKRWAN > FirstConfiguration`, como se muestra en la Figura 74.

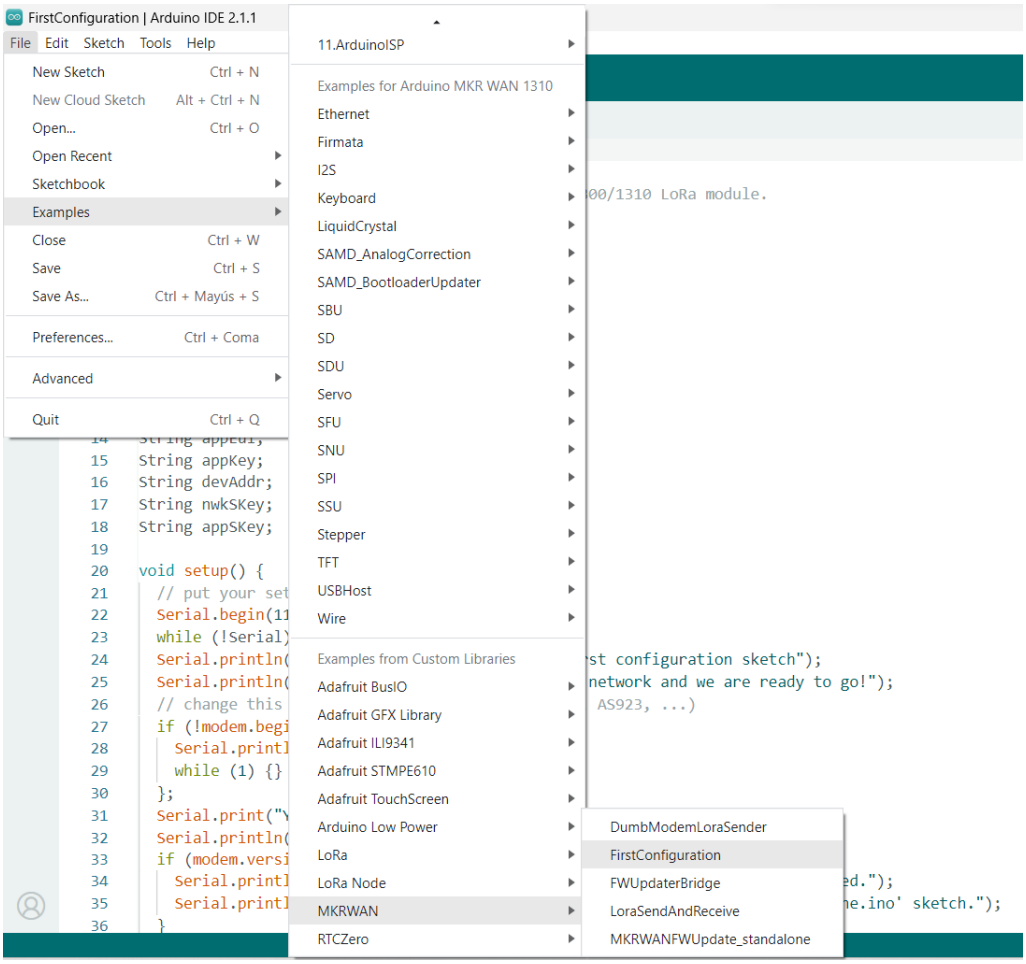


Figura 74 ruta de Sketch FirstConfiguration.ino.

A continuación, programar el dispositivo Arduino MKRWAN 1310 para la ejecución de este *sketch*. Finalmente, abrir el *Serial Monitor* disponible en el entorno Arduino IDE, en el que se mostrará, entre otra información, el identificador *DevEUI* del dispositivo, esencial para la configuración y comunicación efectiva del nodo IoT en la plataforma TTN. En este caso particular, este identificador es `a8610a3330227502`, como se muestra en la Figura 75.

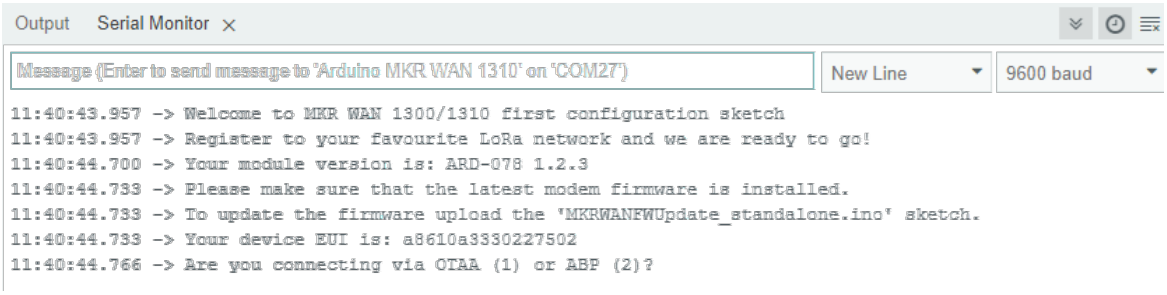


Figura 75 Datos en la interfaz Serie del dispositivo MKRWAN 1310.

En la Figura 76 se muestra la información necesaria para autenticar el dispositivo registrado en la aplicación creada en la plataforma TTN.

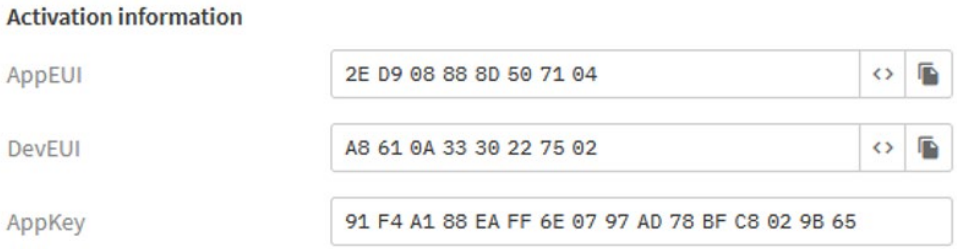


Figura 76 Datos necesarios para la autenticar el dispositivo MKRWAN 1310 en la aplicación alojada en TTN.

Una vez concluido el proceso de registro del dispositivo Arduino MKRWAN 1310, se procederá a validar el envío de datos mediante una comunicación *uplink*. Esta transmisión se realizará desde el dispositivo a través del *Gateway* Heltec HT-M00 previamente registrado en la plataforma TTN, y los datos transferidos se dirigirán hacia la aplicación alojada en el *Network Server* de la red LoRaWAN implementada. Para llevar a cabo esta validación, se empleará de nuevo el *sketch* de referencia `FirstConfiguration.ino` proporcionado con la biblioteca MKRWAN, especificando en este caso que la conexión se establecerá desde el dispositivo utilizando el método de autenticación OTAA (*Over-The-Air Activation*) con el parámetro (`1`), como se muestra en la Figura 78. Este proceso permitirá verificar con éxito la transmisión de datos en la red LoRaWAN implementada.

Para ello, será necesario introducir, tanto el identificador *AppEUI* como la clave *AppKey*, los cuales fueron proporcionados durante el proceso de registro del módulo Arduino

MKRWAN 1310 en la plataforma TTN. Una vez completada esta etapa, en el caso de que se confirme con éxito el envío de datos desde el dispositivo Arduino MKRWAN 1310 hacia la aplicación establecida en TTN mediante la red LoRaWAN, se visualizará un mensaje de confirmación, validando así la comunicación exitosa entre el dispositivo y la plataforma, como se muestra en la propia Figura 77.

```
12:57:12.397 -> Welcome to MKR WAN 1300/1310 first configuration sketch
12:57:12.397 -> Register to your favourite LoRa network and we are ready to go!
12:57:12.893 -> Your module version is: ARD-078 1.2.3
12:57:13.063 -> Please make sure that the latest modem firmware is installed.
12:57:13.063 -> To update the firmware upload the 'MKRWANFWUpdate_standalone.ino' sketch.
12:57:13.063 -> Your device EUI is: a8610a3330227502
12:57:13.063 -> Are you connecting via OTAA (1) or ABP (2)?
12:57:58.419 -> Enter your ABP EUI
12:58:21.784 -> Enter your ABP KEY
12:58:38.833 -> Message sent correctly!
```

Figura 77 Proceso de introducción de parámetros y conformación de comunicación del dispositivo MKRWAN 1310 con la aplicación en TTN.

En la sección *Overview* del dispositivo registrado en la aplicación se presentan de manera detallada los parámetros fundamentales de la conexión establecida entre el módulo Arduino MKRWAN 1310 y la plataforma TTN a través de la puerta de enlace Heltec HT-M00, como se ilustra en la Figura 78, la Figura 79 y la Figura 80.

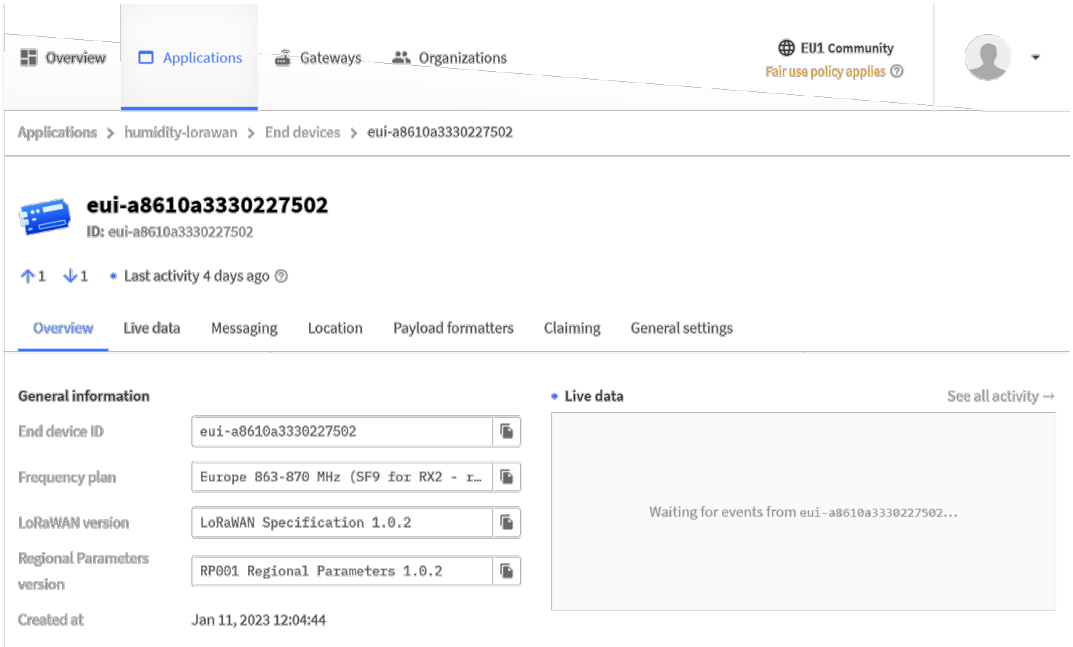


Figura 78 Parámetros de la conexión entre el módulo Arduino MKRWAN 1310 y TTN.

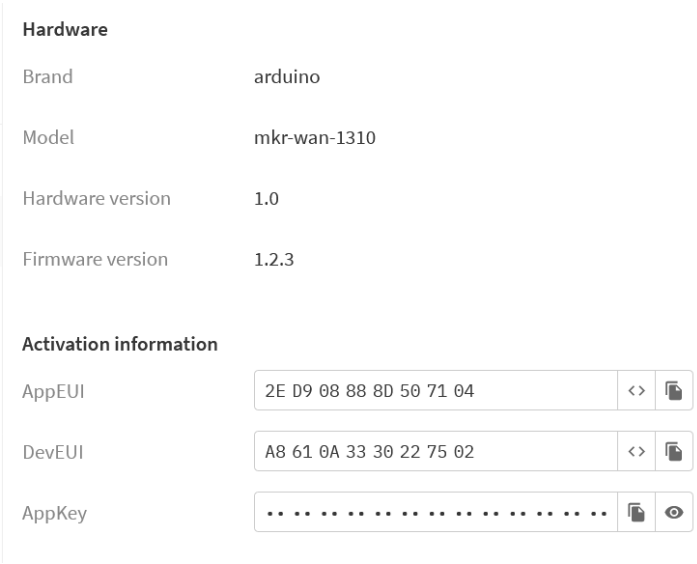


Figura 79 Información del Hardware Arduino MKRWAN 1310.

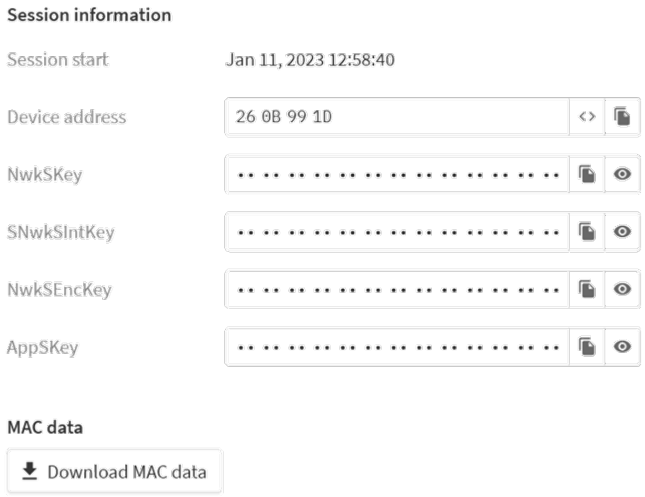


Figura 80 Información de la sesión entre el dispositivo Arduino MKRWAN 1310 y la aplicación.

En la sección *Live Data* de la Figura 81 se dispone de la posibilidad de verificar en este caso la recepción de un mensaje *uplink* transferido desde el dispositivo. La representación hexadecimal de este mensaje, expresada como `48 65 4C 6F 52 41 20 77 6F 72 6C 64 21`, se traduce en la codificación ASCII que corresponde al mensaje "`HeLoRa world`". Este proceso de validación, que implica la interpretación del *payload* enviado en formato hexadecimal, asegura de manera efectiva la transmisión y recepción precisa de información entre el dispositivo y la plataforma. Mediante el uso de un decodificador ASCII online, en la Figura 82 se puede validar el contenido del mensaje recibido.

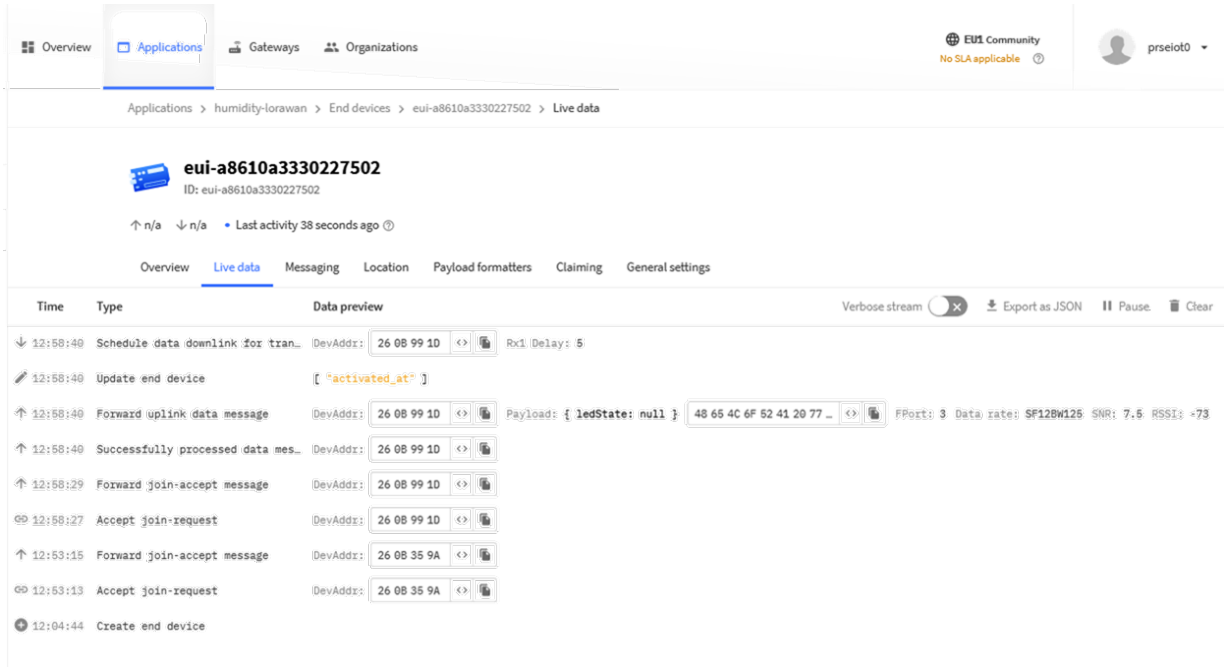


Figura 81 Recepción de datos desde la aplicación en TTN desde el dispositivo Arduino MKRWAN 1310.

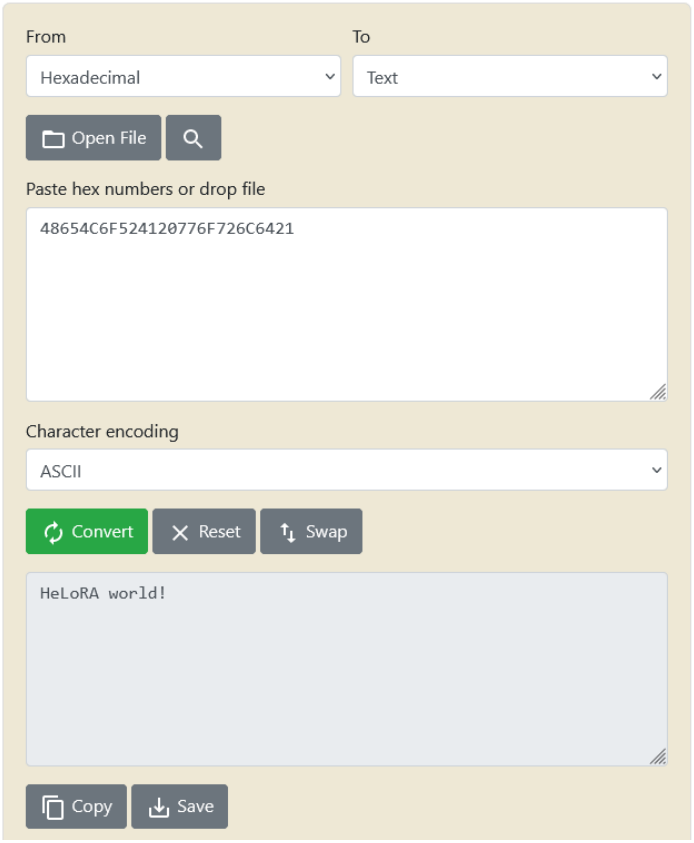


Figura 82 Herramienta conversora del sistema hexadecimal a sistema ASCII [42].

5.4.3 Integración del dispositivo Arduino MKRWAN 1310 con el Gateway Heltec HT-M00 y la plataforma TTN

Una vez validada la integración básica del dispositivo Arduino MKRWAN 1310 con la aplicación creada en la plataforma TTN, en esta sección se presenta el código del *sketch* desarrollado en el presente TFM para establecer una comunicación efectiva entre el dispositivo Arduino MKRWAN 1310 y el *Gateway* Heltec HT-M00, teniendo en cuenta algunas de sus características particulares.

En este *sketch*, denominado `mkr1310-lorawan-lowpower-v2_heltec.ino`, se procede en primer lugar a incluir la biblioteca `MKRWAN` que habilita la comunicación LoRa® a través del módulo Murata CMWX1ZZABZ integrado en el dispositivo Arduino MKRWAN 1310, definiéndose a continuación el objeto `modem` de la clase `LoRamodem`, que se utilizará para interactuar con el módulo LoRa®.

`mkr1310-lorawan-lowpower-v2_heltec.ino`

```
#include <MKRWAN.h>

LoRaModem modem;
```

La función `ttn_begin()` resulta esencial para la configuración inicial del módulo LoRa® en el dispositivo Arduino MKRWAN 1310, permitiendo la conexión con la red LoRaWAN creada a partir de la plataforma TTN. Esta función enciende el LED integrado en la placa para comprobar el estado del dispositivo y saber así en qué fase se encuentra, sin necesidad de utilizar la conexión con un terminal serie, además de inicializar el módulo LoRa® con los parámetros correspondientes a la región de la Unión Europea (EU868) y proporcionar información relevante sobre el módulo y el dispositivo, a través del puerto serie USB.

```
boolean ttn_begin() {

digitalWrite(LED_BUILTIN, HIGH);

if (!modem.connected()) {
   // initialize LoRa module
   if (!modem.begin(EU868)) {
```

```
        Serial.println("* Failed to start MKRWAN 1310 module!");
        while (1) {}
    };
    Serial.println("* Successfully started MKRWAN 1310 module");
    Serial.print("   MKRWAN 1310 module version is: ");
    Serial.println(modem.version());
    Serial.print("   MKRWAN 1310 device EUI is: ");
    Serial.println(modem.deviceEUI());
  }
}
```

La función `ttn_join()` tiene la responsabilidad de establecer la conexión con el Servidor de Red de la plataforma TTN mediante el procedimiento de Activación OTAA. Inicialmente, intenta establecer esta conexión y, en caso de éxito, muestra a través del terminal serie un mensaje indicando que el dispositivo está conectado al *Network Server* de la plataforma TTN. Posteriormente, establece la configuración de una serie de parámetros, entre los que se encuentra la desactivación de ajuste automático de la tasa de datos (*Automatic Data Rate, ADR*) con el fin de fijar la tasa de datos en SF7 con un ancho de banda de canal de 125 kHz, de acuerdo con la información mostrada en la Figura 83, y la limitación del *Gateway* Heltec HT-M00 según la cual, en caso de no modificar la longitud por defecto del preámbulo de los paquetes LoRa®, como se ha establecido en este caso, en el dispositivo Arduino MKRWAN 1310 se recibirán únicamente paquetes con el mínimo SF establecido en la configuración de la puerta de enlace, es decir, SF7.

```
/*
        DataRate        Modulation      SF      BW      bit/s
        0       LoRa    12      125     250
        1       LoRa    11      125     440
        2       LoRa    10      125     980
        3       LoRa    9       125     1'760
        4       LoRa    8       125     3'125
        5       LoRa    7       125     5'470
        6       LoRa    7       250     11'000
*/
```

Figura 83 Relación entre el parámetro dataRate y los valores de SF y BW.

Posteriormente, con el fin de establecer una correcta comunicación LoRa® entre el dispositivo Arduino MKRWAN 1310 y la puerta de enlace Heltec HT-M00, que es de doble canal, es necesario establecer una correspondencia entre los canales utilizados por ambos elementos. En consecuencia, en la función `ttn_join()` se configura la máscara de canal para habilitar únicamente los canales 0 (868.1MHz) y 1 (868.3MHz), de acuerdo con la información mostrada en la Figura 84, deshabilitando los canales 2 al 7, de acuerdo con la configuración establecida en la puerta de enlace, reproducida en la Figura 85.

Modulation	Bandwidth [kHz]	Channel Frequency [MHz]	FSK Bitrate or LoRa DR / Bitrate	Nb Channels	Duty cycle
LoRa	125	868.10 868.30 868.50	DR0 to DR5 / 0.3-5 kbps	3	< 1%

Figura 84 Frecuencia de los canales correspondientes a un BW de 125 KHz.

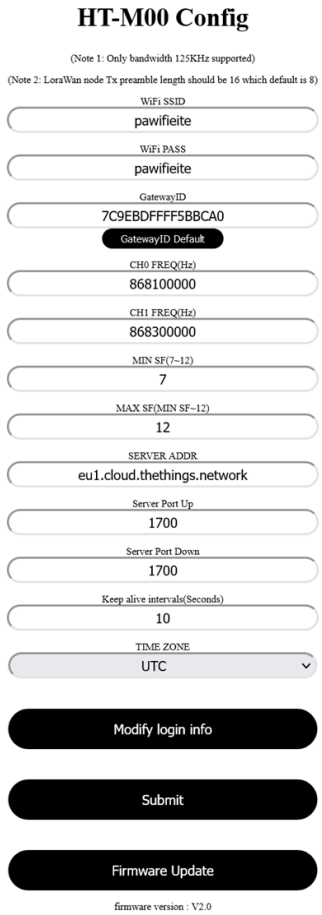


Figura 85 Configuración del Gateway Heltec HT-M00.

Estos procedimientos resultan críticos para asegurar una conexión robusta y eficiente, optimizando la configuración de la comunicación LoRa® para la región específica y asegurando coherencia con los parámetros configurados en el *Gateway* Heltec HT-M00. Esta función también muestra información relevante a través del puerto serie con el fin de informar sobre la configuración establecida, como la máscara aplicada a los canales, así como los dos canales habilitados.

```
boolean ttn_join() {
    // attempt to connect to TTN network server:
    Serial.print("* Attempting to connect to TTN network server (OTAA) ");
    int connected = modem.joinOTAA(appEui, appKey);
    if (!connected) {
       Serial.println("Failed to connect to TTN!");
       return(false);
    }
    Serial.println("You're connected to TTN network server");

    modem.setADR(true);   // turn off Automatic Data Rate mode
    modem.dataRate(5);    // set data rate to be SF7 and channel //bandwidth to
125kHz
    Serial.print("   - Current channel mask: ");
    Serial.println(modem.getChannelMask());
    Serial.print("   - Disabling CH2 to CH7 (enable only CH0-868.1MHz & CH1-
868.3MHz)");

    for (unsigned int i = 2; i < 8; i++) {
      if (modem.disableChannel(i))
      {//Serial.print(".");
      }
      else {
        //Serial.print("F");
        }
    }
    Serial.print(" - channel mask: ");
    Serial.println(modem.getChannelMask());

    return(true);
    delay(1000);
```

}

Algunas de las líneas anteriores surgieron como resultado de diversas pruebas de configuración y la identificación de errores que se manifestaron durante las fases de comunicación LoRa® entre el dispositivo Arduino MKRWAN 1310 y la puerta de enlace Heltec HT-M00. Estas experiencias condujeron a las conclusiones y refinamientos de los bloques de código explicados anteriormente.

5.5 Gateway Wis Gate Edge Lite 2 model RAK7268

El *Gateway* RAK7268 WisGate Edge Lite 2 desarrollado por la empresa *RAK Wireless* y representado en la Figura 86, constituye una puerta de enlace integral de 8 canales, con conectividad Ethernet incorporada, orientada a la implementación del protocolo LoRaWAN. Además, proporciona un punto de acceso Wi-Fi integrado que facilita su configuración [43].

Figura 86 RAK7268 WisGate Edge Lite 2 [43].

Al igual que otros dispositivos de la línea de productos RAK Wireless Industrial *Gateways*, la puerta de enlace RAK7268 incorpora la tecnología *Power Over Ethernet* (PoE), lo que elimina la necesidad de instalar infraestructura eléctrica de alimentación adicional [43].

El software de código abierto destinado a la administración y configuración de esta puerta de enlace se basa en *OpenWRT*, caracterizándose por incluir un remitente de paquetes LoRa® integrado, así como una interfaz gráfica de usuario. Estas funcionalidades permiten una configuración ágil sin comprometer la flexibilidad de una solución completamente personalizable. El *Gateway* RAK7268 también es compatible con la función *MQTT Bridge*, pudiendo emplear el protocolo MQTT integrado en plataformas de terceros [43].

5.5.1 Características

El *Gateway* RAK7268 ofrece una completa compatibilidad con la pila de protocolo *LoRaWAN Stack* (Versión 1.0.3), haciendo uso del módulo Semtech SX1302. Además, facilita la configuración a través de su soporte para redes Wi-Fi de 2.4 GHz en modo AP *(Access Point)* [43].

Este dispositivo incorpora conectividad Ethernet Base-T de 100M con funcionalidad PoE, lo que simplifica las operaciones de montaje y proporciona múltiples opciones de *back-haul*, incluyendo Ethernet, Wi-Fi, y opcionalmente, conexiones celulares. La gestión y configuración de la puerta de enlace RAK7268 se basa en el software *OpenWRT*, brindando una interfaz de usuario web que simplifica estas tareas [43].

Este producto permite la integración, tanto con Servidores de Red privados, como *ChirpStack*, como con Servidores de Red públicos, como TTN. También dispone de una tarjeta TF que se puede utilizar para almacenar en búfer tramas LoRa® en caso de experimentarse fallos de conexión. Asimismo, presenta un Servidor de Red integrado que facilita la implementación de aplicaciones y la integración de *Gateways*. En última instancia, se encuentra disponible la opción de conectividad LTE Cat 4, aunque es opcional [43].

5.5.2 Especificaciones

La descripción general del *Gateway* RAK7268 se muestra en el diagrama de bloques de la Figura 87, en el que se representa la arquitectura interna de la placa.

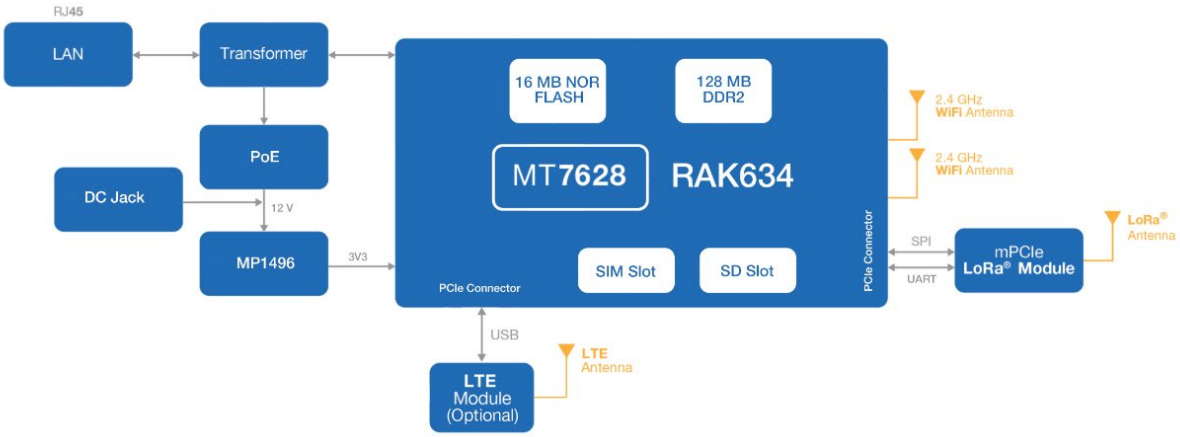


Figura 87 Diagrama de bloques de WisGate Edge Lite 2 [43].

5.5.2.1 Interfaces

La puerta de enlace RAK7268 cuenta con diversas interfaces hardware, como se observa en la Figura 88, entre las que se incluyen: una interfaz de entrada de tensión de 12 V para la alimentación, una interfaz ETH para la conexión Ethernet, una interfaz de consola, un botón de reinicio, una ranura destinada a una tarjeta microSD (*Micro Secure Digital*), indicadores LED que muestran el estado de funcionamiento, y un conector específico para la antena LoRa® [43]. En la Tabla 17 se muestran más detalladas estas interfaces y algunas características técnicas del dispositivo [43].

Figura 88 Interfaces WisGate Edge Lite 2 [43].

Tabla 17 Caracteristicas técnicas del dispositivo Wis Gate Edge Lite 2 model RAK7268 [43].

Computing	**MT7628, DDR2 RAM 128 MB**
Wi-Fi feature	Frequency: 2.4 GHz (802.11b/g/n)
	RX Sensitivity: -95 dBm (Min)
	TX Power: 20 dBm (Max)
	Operation channels: 2.4 GHz: 1-13
LoRa feature	SX1302 Mini PCIe card
	8 Channels
	RX Sensitivity: -139 dBm (Min)
	TX Power: 27 dBm (Max)
	Frequency:

	EU433/CN470/EU868/US915/AS923/AU915/IN865/KR920
Cellular	Supports Quectel EG95-E/EG95-NA (IoT/M2M -optimized LTE Cat 4 Module)
	EG95-E for EMEA Region - LTE FDD: B1/B3/B7/B8/B20/B28A - WCDMA: B1/B8 - GSM/EDGE: B3/B8
	EG95-NA for North America Region - LTE FDD: B2/B4/B5/B12/B13 - WCDMA: B2/B4/B5
	Optional supports other PCIe LTE module for Global Region
Power supply	DC 12 V - 1 A
	PoE (IEEE 802.3 af), 36~57 VDC
Power consumption	12 W (typical)
ETH	RJ45 (10/100 M)
Console	Type-C USB
Antenna	LoRa: RP-SMA female connector
	LTE: Internal antenna
	Wi-Fi: Internal antenna
LEDs	POWER LED
	Breathing LED (Top side)
	ETH LED (On ETH connector)
	LoRa LED
	WLAN LED
	LTE LED

5.5.3 Especificaciones de Radiofrecuencia

En la Tabla 18, la Tabla 19 y la Tabla 20 se exponen las especificaciones de radiofrecuencia de la puerta de enlace RAK7268, dividiéndolas en tres categorías fundamentales: especificaciones de conectividad Wi-Fi, características relacionadas con la tecnología LoRa® y, finalmente, aspectos específicos vinculados a la tecnología LTE.

Tabla 18 Especificaciones Wi-Fi Radio [43].

Características	**Especificaciones**
Wireless Standard	IEEE 802.11b/g/n
Operating Frequency	ISM band: 2.412~2.472 GHz
Operation Channels	2.4 GHz: 1-13
Transmit Power **(The max power maybe different depending on local regulations) - per chain**	802.11b 19 dBm @1 Mbps 19 dBm @11 Mbps
	802.11g 18 dBm @6 Mbps 16 dBm @54 Mbps
	802.11n (2.4G) 18 dBm @MCS0 (HT20) 16 dBm @MCS7 (HT20) 17 dBm @MCS0 (HT40) 15 dBm @MCS7 (HT40)
Receiver Sensitivity (Typical)	802.11b -95 dBm @1 Mbps -88 dBm @11 Mbps
	802.11g -90 dBm @6 Mbps -75 dBm @54 Mbps
	802.11n (2.4G) -89 dBm @MCS0 (HT20) -72 dBm @MCS7 (HT20) -86 dBm @MCS0 (HT40) -68 dBm @MCS7 (HT40)

Tabla 19 Especificaciones LoRa-Radio [43].

Características	**Especificaciones**

Operating Frequency	EU433/CN470/EU868/US915/AS923/AU915/IN865/KR920
Transmit Power	27 dBm (Max)
Receiver Sensitivity	-139 dBm (Min)

Tabla 20 Especificaciones LTE-Radio [43].

Características	**Especificaciones**
EG95-E for EMEA Region	LTE FDD: B1/B3/B7/B8/B20/B28A WCDMA: B1/B8 GSM/EDGE: B3/B8
EG95-NA for North America Region	LTE FDD: B2/B4/B5/B12/B13 WCDMA: B2/B4/B5 Optional supports other PCIE LTE module for Global Region

5.5.4 Requisitos eléctricos

El *Gateway* RAK7268 incluye un adaptador de corriente nominal de 12V-1A y ofrece compatibilidad plena con la tecnología PoE, según la norma IEEE 802.3af, en un rango de tensión que oscila entre 36 VCC y 57 VCC. El consumo de energía característico se encuentra en el orden de 12 W [43].

5.5.5 Requisitos medioambientales

La carcasa tiene clasificación IP30 y está fabricada en plástico. En la parte posterior incluye una serie de orificios para un soporte que permite simplificar su montaje en la pared [43]. En la Tabla 21 se muestran las características físicas del *Gateway* RAK7268.

Tabla 21 Caracteristicas fisicas del dispositivo RAK7268/C WisGte Edge Lite 2 [43].

Parámetro	**Valor**
Dimensiones	66x127x36mm

Peso	0,3 kilos
Temperatura de funcionamiento	-10 a 55°C
Ingress protection	IP30
Enclosure material	Plastic
Método de instalación	Wall mounting

5.5.6 Firmware

El *firmware* de la puerta de enlace RAK7268 se basa en *OpenWRT*, lo que permite su personalización. Dispone de una interfaz de usuario web que simplifica la configuración y la gestión del dispositivo [43].

Tabla 22 Versión de Firmware del dispositivo RAK7268/C WisGte Edge Lite 2. [43].

Modelo	**Versión del firmware**
RAK7268 WisGate Edge Lite 2	WisGateOS V1.3.9

5.5.7 Funciones software

El software del *Gateway* RAK7268 incorpora una serie de funciones clave para facilitar la configuración, administración y monitorización de los dispositivos IoT conectados. Estas funciones se resumen en la Tabla 23.

Tabla 23 Funciones del Software del Roter Router RAK7268 WisGate Edge Lite 2 [43].

LoRaWAN	**Red**	**Gestión**
Soporta clase A, C	Modo AP Wi-Fi	Gestión WEB
Paquete LoRa adelante	Copia de seguridad de enlace ascendente	SSH2
Configuración del código de país	802.1q	Actualización de firmware

Configuración de energía TX	Servidor/Cliente DHCP	NTP
Registrador de datos	Módulo de enrutador NAT	Configuración del reenviador de paquetes LoRa
Estadísticas	Cortafuegos	Servidor incorporado
Configuración de ubicación	Configuración de APN LTE	OpenVPN, perro guardián de ping
Dirección del servidor y configuración del puerto		MQTT Bridge

5.5.8 Configuración del dispositivo RAK7268

La configuración inicial del *Gateway* RAK7268 contempla los siguientes pasos:

1. Se conecta la antena LoRa®: Se atornilla la antena al conector RP-SMA en el panel posterior del RAK7268 (Figura 89). No se debe encender el dispositivo en caso de que el puerto de la antena LoRa® se deje abierto para evitar posibles daños.
2. Se alimenta el *Gateway*: Se recomienda utilizar el adaptador de 12 V CC que se incluye con la puerta de enlace RAK7268. Opcionalmente, se puede utilizar el propio cable Ethernet, ya que el dispositivo soporta PoE.

Figura 89 Vista superior de RAK7268/C WisGte Edge Lite 2 [43].

5.5.9 Indicadores LED de estado

En la Tabla 24 se muestra el significado de los diferentes estados de los LED integrados en el *Gateway* RAK7268.

Tabla 24 Funcionamiento de los LED de estados del dispositivo RAK7268/C WisGte Edge Lite 2 [43].

LED	**Estado Indicación Descripción**
LED DE ENERGÍA	Indicador de encendido: el LED está encendido cuando el dispositivo está encendido.
LED de respiración	Respirar después de que el sistema esté activo.
LED ETH	ENCENDIDO: el enlace está activo. APAGADO: el enlace no funciona. Parpadeo: transferencia de datos en curso.
LED LoRa	ENCENDIDO: LoRa está activo. APAGADO: LoRa está inactivo. Parpadeo: transferencia de datos en curso.
LED inalámbrico	Modo AP: - ON - El AP está activo. - APAGADO - El AP está caído. - Parpadeo - Transferencia de datos en curso. Modo STA: - Parpadeo lento (1 Hz) – Desconectado. - ENCENDIDO – Conectado. - Parpadeo - Transferencia de datos en curso.
LED LTE (se iluminará solo en RAK7268C)	Parpadeo lento (1800 ms alto / 200 ms bajo) - Búsqueda de red. Parpadeo lento (200 ms alto / 1800 ms bajo) - Inactivo Parpadeo rápido (125 ms alto / 125 ms bajo) - Transferencia de datos en curso

5.5.10 Acceso a la puerta de enlace

5.5.10.1 Modo AP Wi-Fi

Por defecto, la puerta de enlace RAK7268 opera en el modo de *Access Point* (AP) Wi-Fi. En este modo, los usuarios pueden localizar una red con el SSID `RAK7268_XXXX` en la lista de redes Wi-Fi disponibles, donde `XXXX` se corresponde con los dos últimos bytes de la

dirección MAC del *Gateway*. Para acceder a la interfaz de configuración, se puede usar un navegador web, utilizándose en la página de inicio de sesión, representada en la Figura 90, las credenciales que se indican a continuación [43]:

- `Nombre de usuario: root`
- `Contraseña: root`

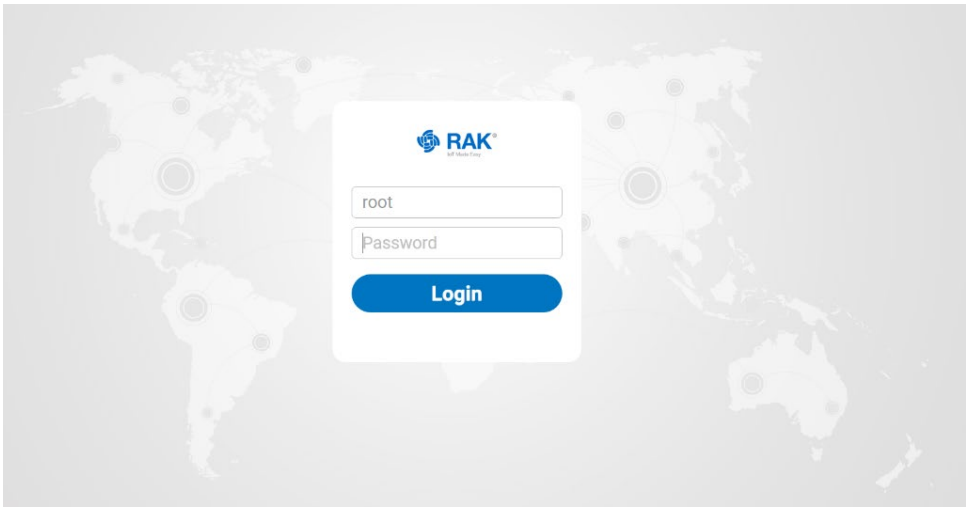


Figura 90 Página de inicio de sesión de la interfaz de usuario web [43].

5.5.11 Acceso a Internet

5.5.11.1 A través de Wi-Fi

Para configurar los parámetros de conexión de la puerta de enlace RAK7268 a Internet a través de un determinado punto de acceso WiFi, en el menú de configuración de red debe seleccionarse la sección denominada *Wi-Fi*, como se muestra en la Figura 91, asegurándose de seleccionar la opción *Wireless Client*. En esta sección, se proporciona la posibilidad de especificar, o bien de manera explícita o bien ejecutando la función *Scan*, el *Extended Service Set Identifier* (ESSID) de la red WiFi. Asimismo, es esencial que se elija el método de cifrado adecuado de acuerdo con las especificaciones de seguridad de la red a la que se desea conectar, y posteriormente, introducir la contraseña correspondiente para completar la configuración del cliente WiFi [43].

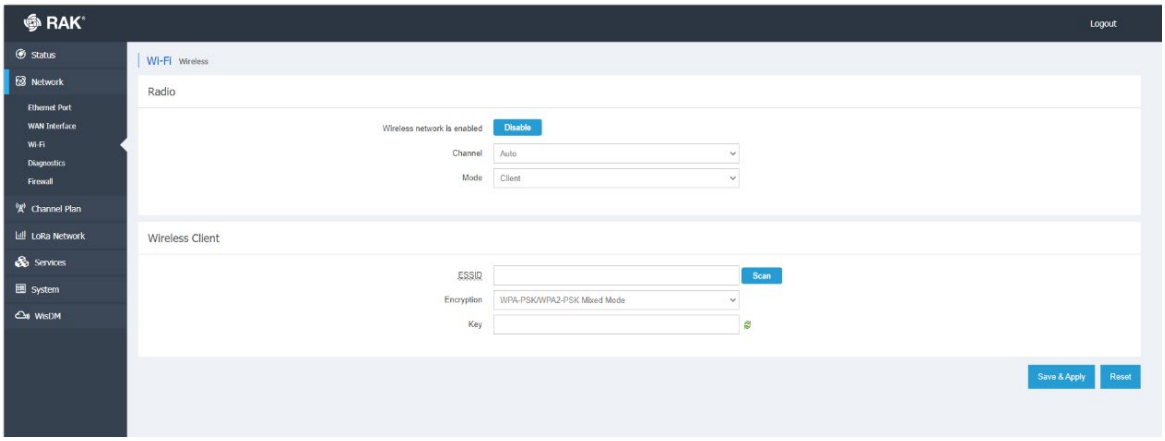


Figura 91 Conexión a través de credenciales Wi-Fi [43].

En caso de haber introducido correctamente el valor de estos parámetros, se debería recibir la asignación de una dirección IP por parte del servidor DHCP incorporado en el *router* del punto de acceso WiFi. Con esta nueva dirección IP, se podrá acceder al *Gateway* RAK7268 directamente a través de un navegador web, siguiendo el mismo procedimiento que en el modo de punto de acceso AP [43].

5.5.11.2 A través de Ethernet

Para configurar los parámetros de conexión de la puerta de enlace RAK7268 a Internet a través de Ethernet, se debe conectar un extremo del cable Ethernet en la puerta de enlace y el otro extremo al *router*. El servidor DHCP del *router* deberá asignar a continuación una dirección IP a la puerta de enlace, como se muestra en la Figura 92 [43].

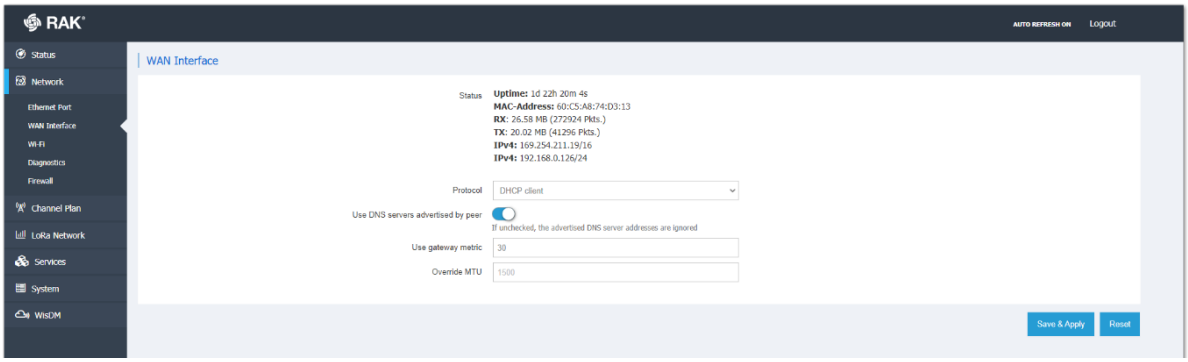


Figura 92 Datos del estado de la conexión a través de Ethernet [43].

5.5.12 Registro del Gateway RAK7268 en la plataforma TTN

La interconexión entre dispositivos IoT es crucial en la creación de soluciones inteligentes. Por consiguiente, resulta imperativo llevar a cabo el registro adecuado del *Gateway* RAK7268 en la plataforma TTN, tal como se realizó previamente con la puerta de enlace Heltec HT-M00.

Inicialmente, se procederá con el registro del *Gateway* RAK7268 en la plataforma TTN con el fin de que los dispositivos Arduino MKRWAN 1310 que se integren en la red LoRaWAN, puedan tener acceso al *Network Server* TTN. Para ello, se elige la opción *Register Gateway* de la plataforma TTN, como se muestra en la Figura 93.

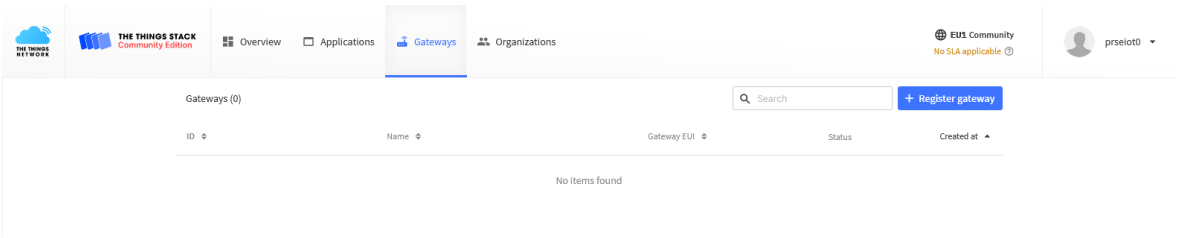


Figura 93 Registro del Gateway RAK7268.

El parámetro EUI del *Gateway* RAK7268 puede localizarse, ya sea en la etiqueta adherida a la carcasa del dispositivo, como se muestra en la Figura 94, o bien consultando la sección de configuración de la red LoRa® en el menú de la puerta de enlace, accesible a través de la interfaz de usuario representada en la Figura 95.

Figura 94 EUI del Gateway RAK en el revestimiento del dispositivo.

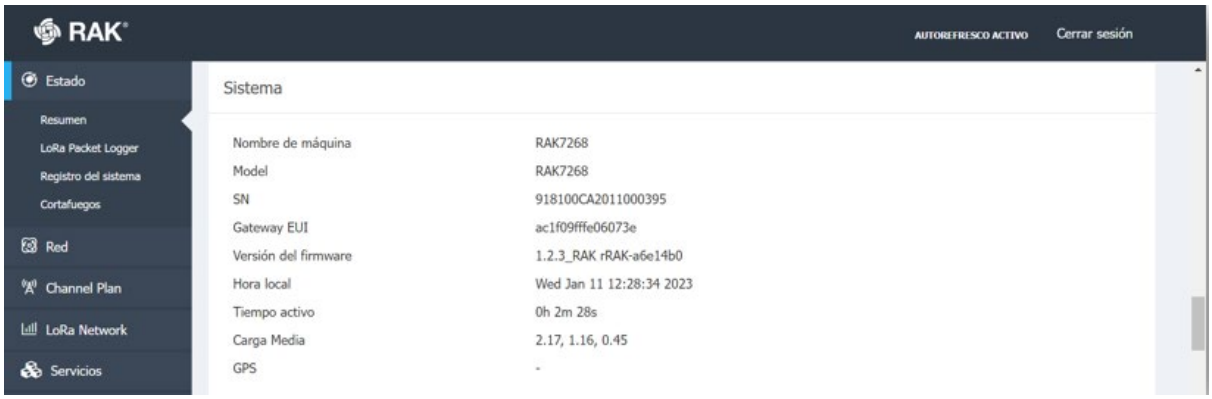


Figura 95 EUI del Gateway RAK a través de la interfaz web.

A continuación, se procede a la selección del plan de frecuencias adecuado, así como a la asignación de un nombre distintivo a la puerta de enlace, como se observa en la Figura 96.

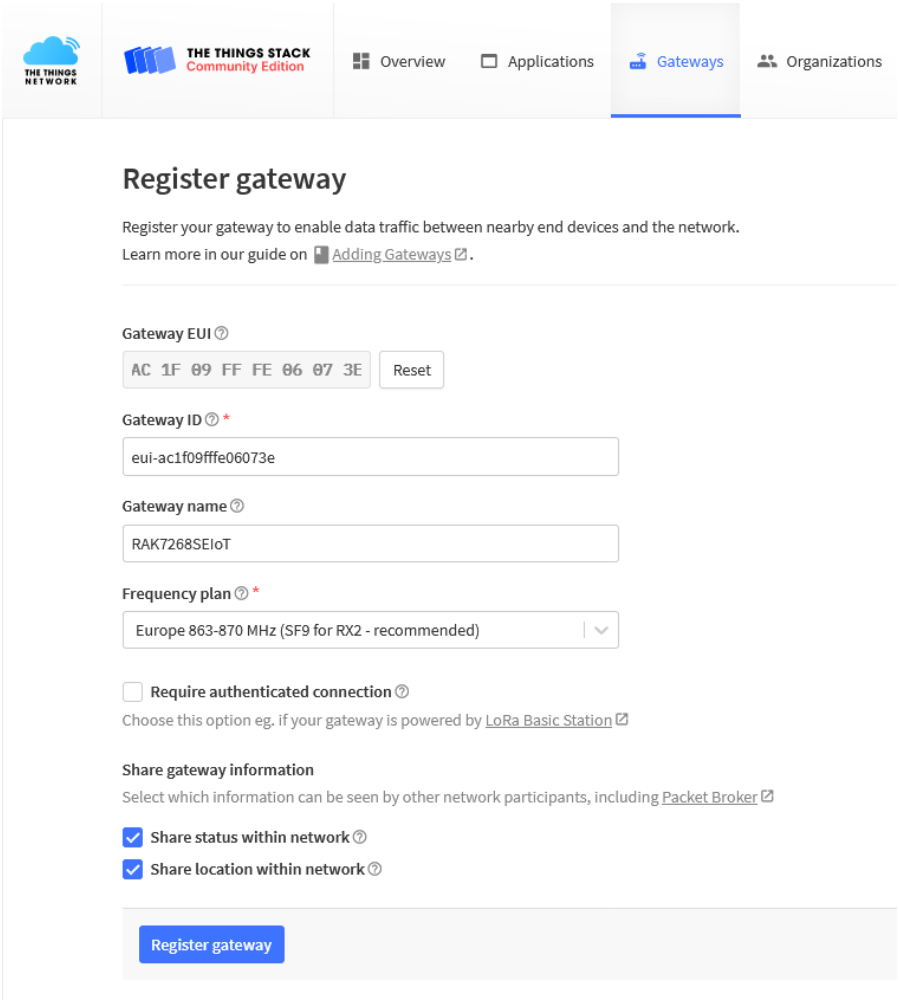


Figura 96 Registro del Gateway RAK7268/C WisGate Edge Lite 2 en TTN.

En la Figura 97 y la Figura 98 se muestran de manera detallada y simplificada el resultado del proceso seguido para el registro del *Gateway* RAK7268, y su información de estado en la plataforma TTN, respectivamente.

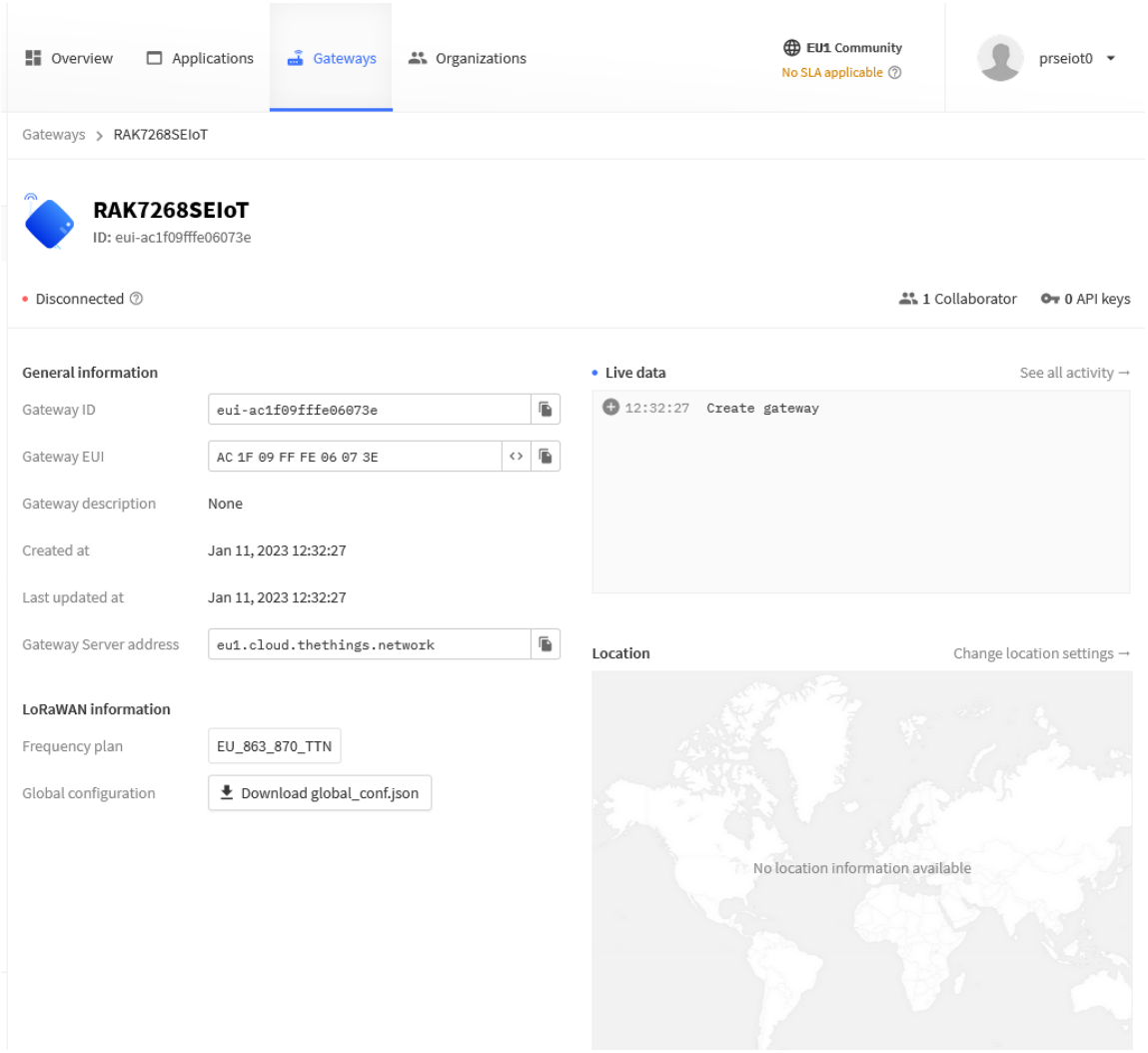


Figura 97 Estado del Gateway RAK7268 una vez registrado en TTN.

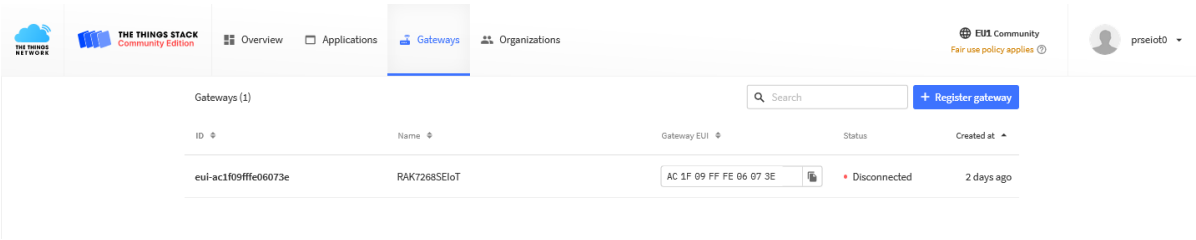


Figura 98 Estado del Gateway RAK7268 resumido una vez registrado en TTN.

Una vez concluido el proceso de registro del *Gateway* RAK7268 en el Servidor de Red TTN, resta únicamente establecer la conexión entre ambos para la transferencia de datos en las comunicaciones *uplink*/*downlink*. Esta conexión se basa en el protocolo LNS (*LoRaWAN Network Server*). TTN respalda el modo de autenticación *TLS Server & Client Token*

Authentication, por lo que es imperativo que un dispositivo *Gateway* cuente con un certificado digital y una clave de API que autentiquen su identidad. Estos elementos son esenciales para la configuración del protocolo LNS en el *Gateway* RAK7268 y, consecuentemente, posibilitan la conexión efectiva a la red LoRaWAN. En la Figura 99 y la Figura 100 se muestra el proceso de generación de una nueva *API Key* en la plataforma TTN.

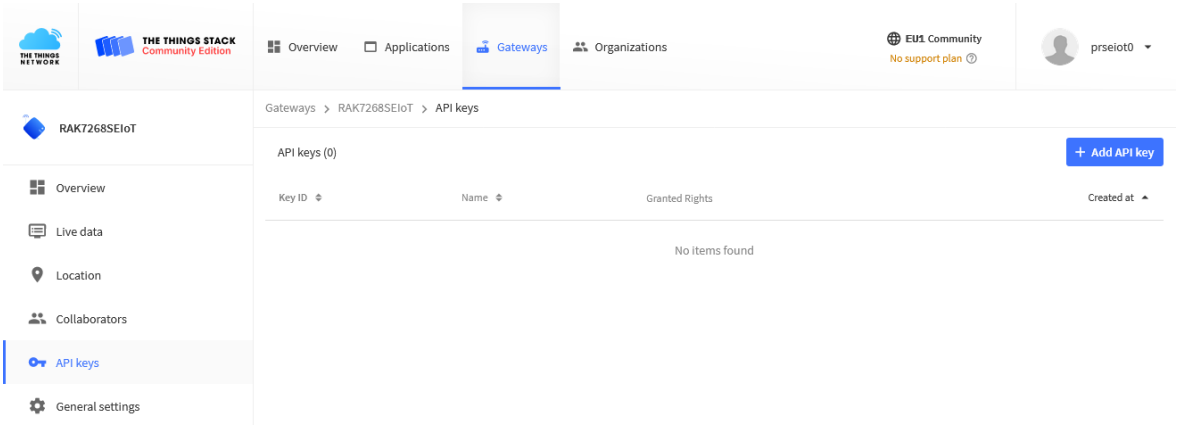


Figura 99 Creación de la API Key.

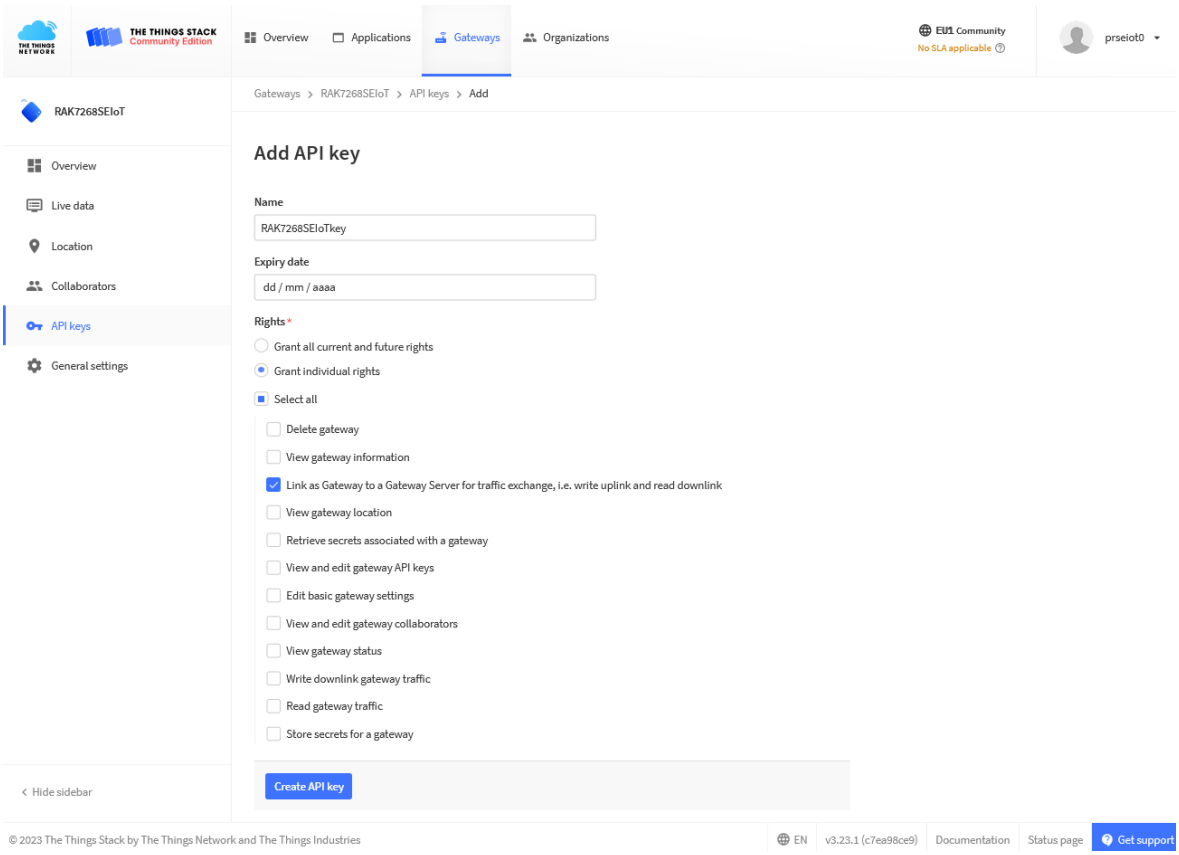


Figura 100 Configuración de la API Key.

En la Figura 101 se presenta, de manera visual, la *API Key* que ha sido generada en la plataforma TTN.

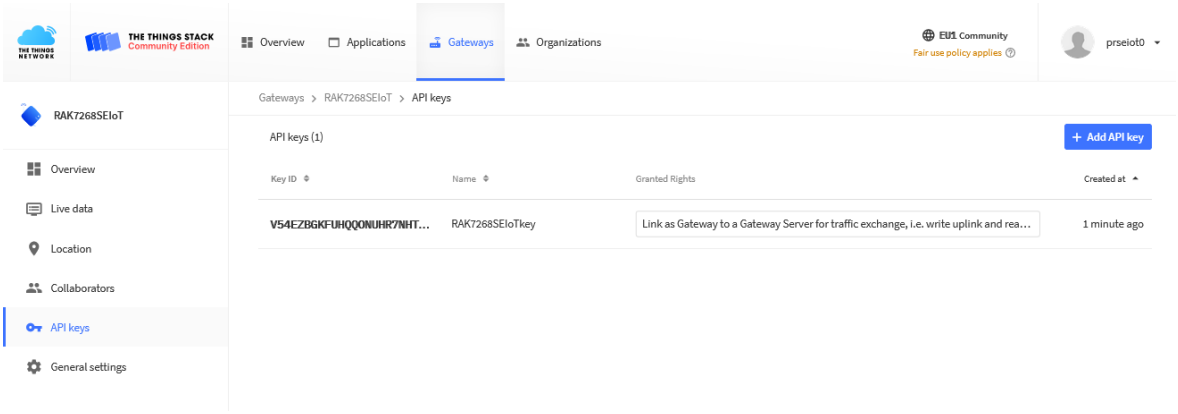


Figura 101 API Key.

En esta fase, se requiere acceder al *Gateway* RAK7268 con el fin de configurar los parámetros que permitirán su conexión con el Servidor de Red TTN. Durante este proceso, se logra identificar la dirección IP de la puerta de enlace RAK7268, correspondiente a `192.168.1.101`. Posteriormente, en la sección *LoRa Network* se selecciona la opción *Network Settings* para llevar a cabo las configuraciones necesarias, como se observa en la Figura 102.

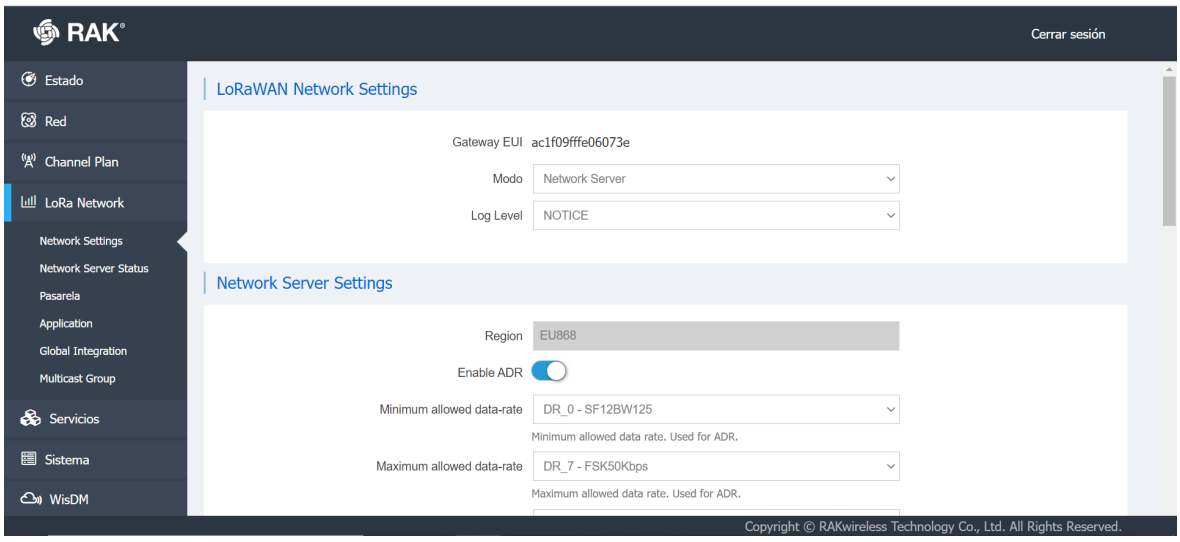


Figura 102 Parámetros de configuración de Red del Gateway RAK7268/C WisGate Edge Lite 2

Como se muestra en la Figura 103, en esta configuración se establece inicialmente el modo *LoRa Basics Station*. Este programa opera en el anfitrión de un *Gateway* basado en LoRa®, dirigiendo paquetes de radiofrecuencia recibidos por el *Gateway* (*Uplinks*) hacia un Servidor de Red LoRaWAN mediante un enlace IP seguro, y transmitiendo paquetes de radiofrecuencia enviados desde el Servidor de Red LoRaWAN (*Downlinks*) hacia uno o varios dispositivos finales.

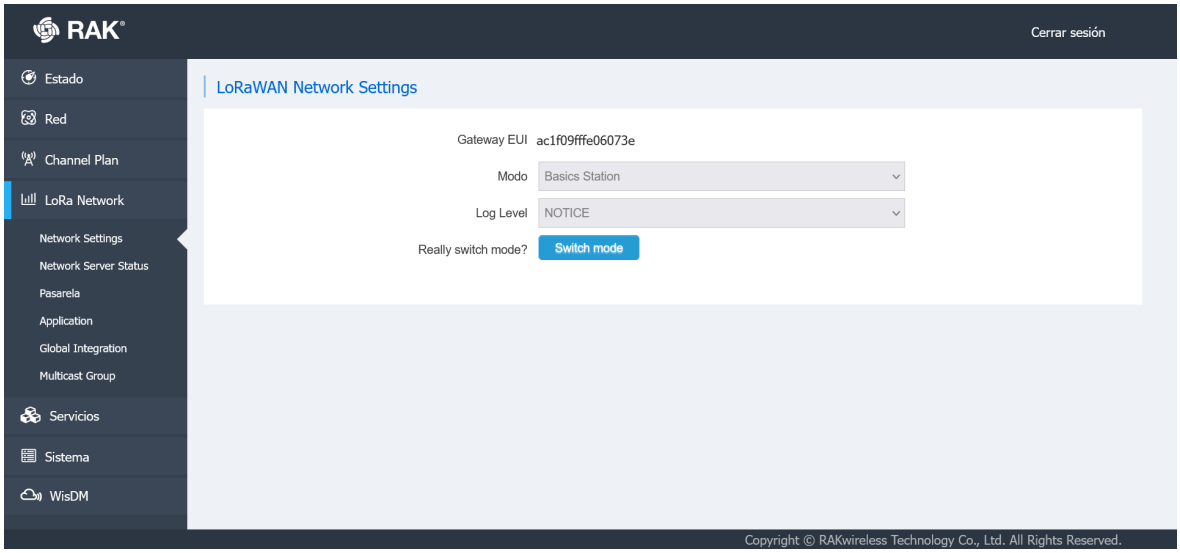


Figura 103 LoRaWAN Network Settings en el Gateway RAK7268.

El servidor LNS se identifica en este caso con la URL específica de la plataforma TTN para el clúster europeo, que es `wss://eu1.cloud.thethings.network`, a través del puerto `8887`, como se muestra en la Figura 104. Además, se configura un modo de autenticación que combina la autenticación del servidor TLS y el *Token* del Cliente. En términos de confianza, se utiliza el certificado `Let's Encrypt ISRG ROOT X1 Trust`. Además, se establece la vinculación mediante el uso del token, que corresponde a la *API Key* generada desde la plataforma TTN. Este conjunto de parámetros proporciona una configuración segura y fiable para la interconexión del *Gateway* con el servidor LNS. Cabe destacar que la implementación de certificados de confianza, como el mencionado, garantiza la integridad y autenticidad de las comunicaciones en la red LoRaWAN.

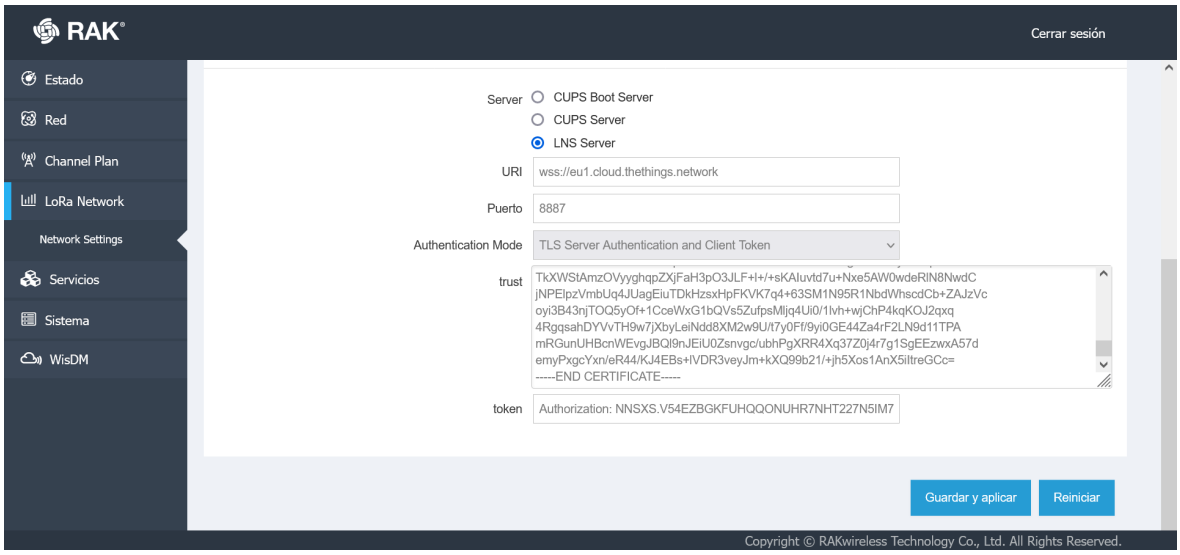


Figura 104 Configuración del servidor LNS en el Gateway RAK7268.

Al retornar a la consola de la plataforma TTN, se constata que la conexión con el *Gateway* RAK7268 se ha establecido con éxito, como se observa en la Figura 105.

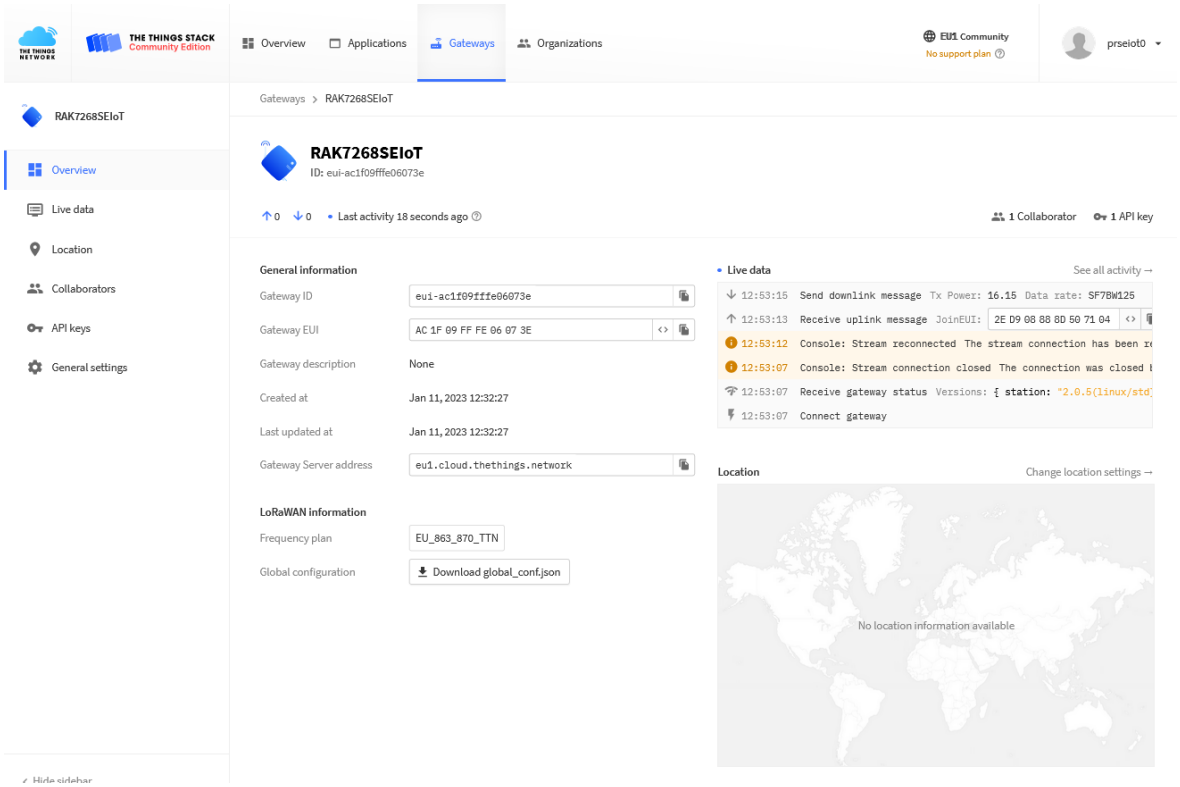


Figura 105 Estado del dispositivo RAK7268 en TTN.

Finalmente, en la Figura 106 se establece la localización geográfica del *Gateway* RAK7268 registrado en la plataforma TTN, al igual que se hizo con la puerta de enlace Heltec HT-M00.

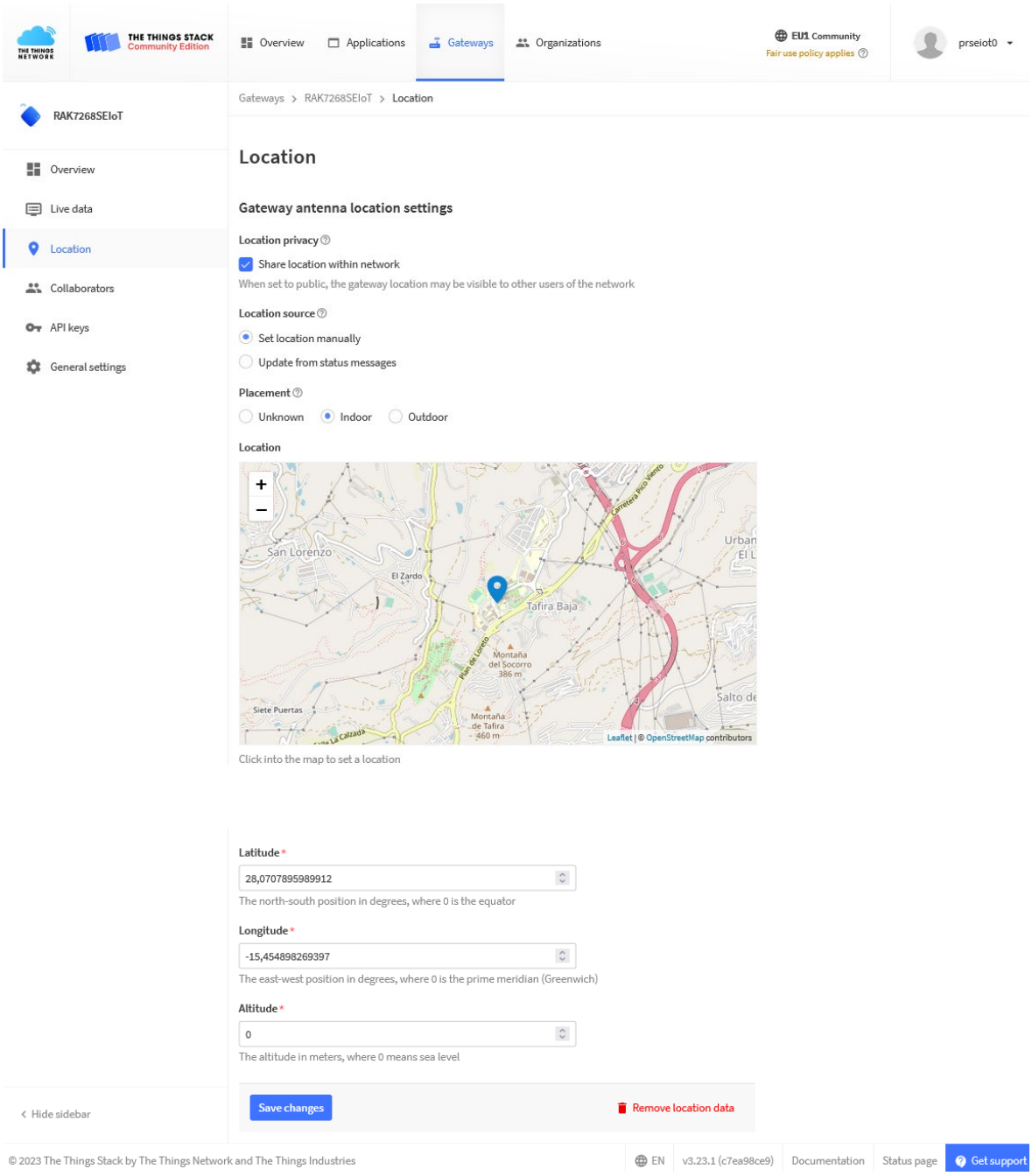


Figura 106 Localización del dispositivo Gateway RAK.

La Figura 107 proporciona una visión del estado generalizado del *Gateway* RAK7268, contextualizado con su información de ubicación. Esta representación visual no solo

ofrece una visión inmediata del estado operativo del dispositivo, sino que también facilita una evaluación rápida de su posición geográfica registrada. La visualización conjunta de estos datos resulta esencial para una supervisión efectiva y una toma de decisiones informada en el entorno de la red LoRaWAN.

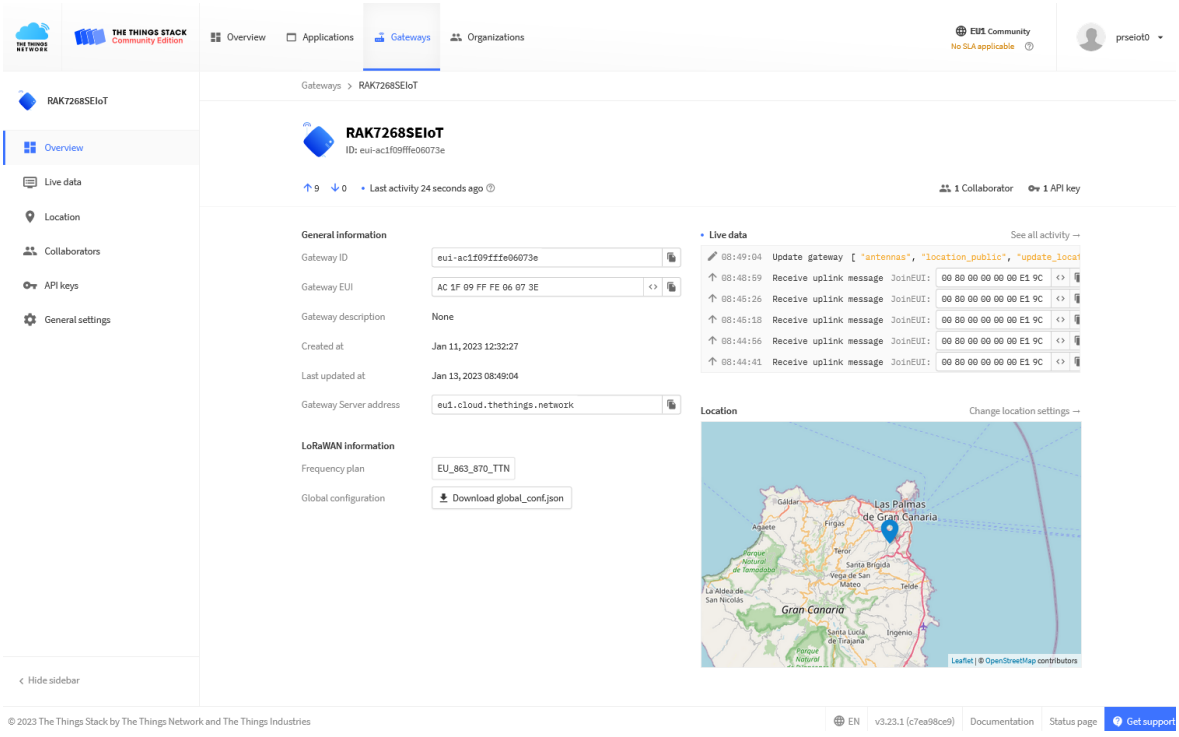


Figura 107 Estado del Gateway RAK en la plataforma TTN.

Como información adicional, en la Figura 108 se muestran diversos enlaces y algunos dispositivos registrados en la plataforma TTN desde las Islas Canarias. Este detalle complementario proporciona una visión más completa de la infraestructura y conectividad presentes en la región, enriqueciendo la comprensión del ecosistema LoRaWAN en el archipiélago.

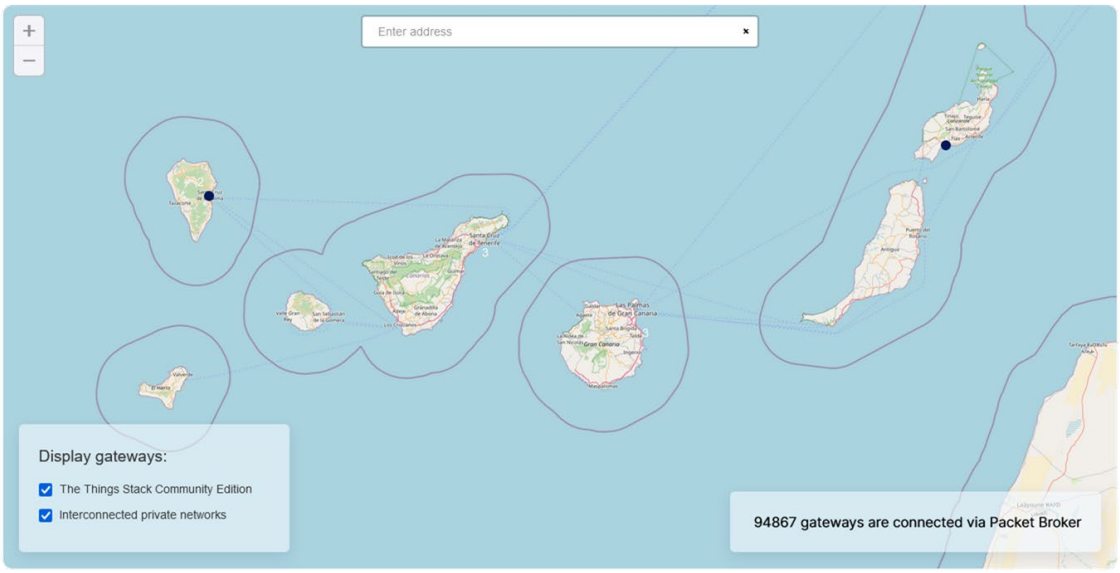


Figura 108 Estado de la Red de TTN en las Islas Canarias.

5.5.13 Integración del dispositivo Arduino MKRWAN 1310 con el *Gateway* RAK7268 y la plataforma TTN

El *sketch* desarrollado en este caso para establecer la comunicación entre el dispositivo Arduino MKRWAN 13010 y la plataforma TTN a través del *Gateway* RAK7268 se basa en el implementado en el caso del uso de la puerta de enlace Heltec HT-M00, aunque con una diferencia significativa, ya que se ha eliminado la restricción que limitaba el número de canales utilizados. Esta modificación se realiza debido a que el *Gateway* RAK7268 es integral, soportando 8 canales en las comunicaciones LoRa®. Al eliminar esta restricción, el *Gateway* RAK7268 puede adaptarse dinámicamente a las necesidades en cada caso, lo que resulta en un mejor rendimiento en entornos donde la disponibilidad de canales es variable. Esta característica permite al *Gateway* RAK7268 aprovechar al máximo los recursos de comunicación disponibles, lo que potencialmente hace que resulte más efectivo en entornos con interferencias o alta congestión de canales. Sin embargo, es importante tener en cuenta que esta modificación podría requerir de un mayor consumo de energía o de recursos hardware, en comparación con la puerta de enlace Heltec HT-M00.

5.5.14 Implementación de la primera propuesta de aplicación IoT

En secciones anteriores se ha presentado en detalle el planteamiento correspondiente a la primera de las plataformas que se proponen para el desarrollo del presente TFM. En este epígrafe se concluirá dicho planteamiento mediante la integración de los elementos implicados en la aplicación IoT asociada a la primera de las plataformas propuestas, reproducida por conveniencia en la Figura 109, a falta de completar la integración con InfluxDB, mediante el uso del protocolo MQTT, y la herramienta Grafana, que se abordarán en capítulos posteriores. Esta integración se ha llevado a cabo a partir del desarrollo de un *sketch* denominado `mkr1310_lorawan_person_sensor_v0.2.ino`, que se ejecutará en el dispositivo Arduino MKRWAN 1310.

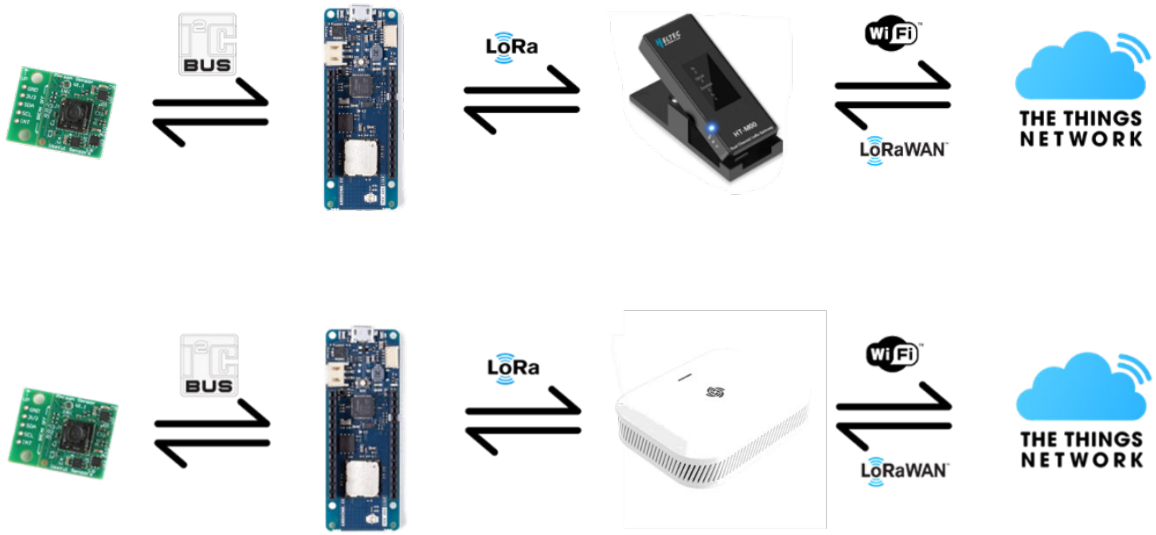


Figura 109 Arquitectura de integración del dispositivo Arduino MKRWAN 1310 y el módulo Person Sensor con la plataforma IoT TTN mediante los Gateways LoRaWAN HT-M00 y RAK7268.

El código del *sketch* `mkr1310_lorawan_person_sensor_v0.2.ino` comienza incluyendo varias bibliotecas esenciales. Así, se utilizan de nuevo las bibliotecas `Wire.h` y `RTCZero` para la comunicación I_2C y el control del reloj en tiempo real (RTC) integrado en el dispositivo Arduino MKRWAN 1310, respectivamente. Por su parte, la biblioteca `ArduinoLowPower.h` permite gestionar los modos de bajo consumo del microcontrolador SAMD21, mientras que `MKRWAN.h` se usa para la comunicación LoRa®. Finalmente, la biblioteca "`person_sensor.h`" proporciona los controladores necesarios

para la integración del módulo *Person Sensor*, mientras que el fichero "ttn_credentials.h" almacenará las credenciales para la conexión con la red LoRaWAN

mkr1310_lorawan_person_sensor_v0.2.ino

```
#include <Wire.h>
#include <RTCZero.h>
#include <ArduinoLowPower.h>
#include <MKRWAN.h>
#include "ttn_credentials.h"
#include "person_sensor.h"
```

Se definen las constantes para la configuración del RTC y se declaran variables globales. RTC_MATCH_MINUTE y RTC_MATCH_SECOND mediante las que se configura el intervalo de tiempo asociado a la alarma del RTC. Por otro lado, se crean objetos y variables necesarias para el uso del RTC, el módulo *Person Sensor*, el control del estado de la batería, y el módem LoRa®. En la estructura SensorResults se definen los parámetros del módulo *Person Sensor* que se van a enviar desde el nodo final, así como el porcentaje de carga de la batería.

```
#define RTC_MATCH_MINUTE 1
#define RTC_MATCH_SECOND 30
#define MAX_num_faces 4

RTCZero rtc;
uint8_t counter = 0;

int alimentacion_sensor_person = 1;

float voltaje_bateria;

LoRaModem modem;

const int32_t SAMPLE_DELAY_MS = 200;
```

```
struct SensorResults {
  uint8_t num_faces;
  uint8_t is_facing;
  uint8_t box_confidence[MAX_num_faces];
  uint16_t porcentaje_carga;
};
```

La función `ttn_begin()` tiene por objeto inicializar el módulo LoRa® y asegurar que se encuentre correctamente conectado y operativo, encendiendo el LED integrado en el dispositivo Arduino MKRWAN 1310 para indicar visualmente que el proceso ha comenzado, además de verificar la conexión con el módem, inicializarlo y, si tiene éxito, mostrar información relevante sobre el módulo a través del terminal serie. En caso de fallo, se muestra un mensaje de error y el programa se detiene en un bucle infinito, ya que esta función es crucial para asegurar que el dispositivo pueda comunicarse correctamente a través de la red LoRaWAN.

```
boolean ttn_begin() {
  digitalWrite(LED_BUILTIN, HIGH);

  if (!modem.connected()) {
    if (!modem.begin(EU868)) {
      Serial.println("* Failed to start MKR WAN 1310 module!");
      while (1) {}
    }
    Serial.println("* Successfully started MKR WAN 1310 module");
    Serial.print("   MKR WAN 1310 module version is: ");
    Serial.println(modem.version());
    Serial.print("   MKR WAN 1310 device EUI is: ");
    Serial.println(modem.deviceEUI());
  }
}
```

La función `ttn_join()` se encarga de conectar el dispositivo Arduino MKRWAN 1310 a la plataforma TTN utilizando el método *Over The Air Activation* (OTAA) definido en el protocolo LoRaWAN. En primer lugar, muestra un mensaje indicando que está intentando

establecer la conexión con la plataforma TTN de acuerdo con el método OTAA, y a partir del uso de las credenciales `appEui` y `appKey` obtenidas en el proceso de integración del dispositivo en la aplicación implementada en la plataforma TTN. En caso de que la conexión falle, se muestra un mensaje de error y retorna `false`. En caso de que se establezca la conexión con éxito, muestra un mensaje de confirmación. A continuación, se activa el modo de *Adaptive Data Rate* (ADR) y establece la tasa de datos en SF7 con un ancho de banda de 125 kHz, de acuerdo con las limitaciones establecidas por la puerta de enlace Heltec HT-M00. Posteriormente, muestra la máscara de canal actual y procede a desactivar los canales 2 a 7, dejando habilitados únicamente los canales 0 (868.1 MHz) y 1 (868.3 MHz), representando un punto (.) para cada canal desactivado con éxito, o la letra 'F' en caso de que la desactivación falle. Finalmente, muestra la nueva máscara de canal y retorna `true` para indicar el éxito en el establecimiento de la conexión.

```
boolean ttn_join() {
  Serial.print("* Attempting to connect to TTN network server (OTAA) ... ");
  int connected = modem.joinOTAA(appEui, appKey);
  if (!connected) {
    Serial.println("Failed to connect to TTN!");
    return false;
  }
  Serial.println("You're connected to TTN network server");

  modem.setADR(true);
  modem.dataRate(5);

  Serial.print("   - Current channel mask: ");
  Serial.println(modem.getChannelMask());
  Serial.print("   - Disabling CH2 to CH7 (enable only CH0-868.1MHz & CH1-
868.3MHz)");

  for (unsigned int i = 2; i < 8; i++) {
    if (modem.disableChannel(i)) {
      Serial.print(".");
    } else {
      Serial.print("F");
    }
```

```
  }
  Serial.print(" - channel mask: ");
  Serial.println(modem.getChannelMask());

  return true;
  delay(1000);
}
```

Por su parte, la función `camera_person_sensor()` se encarga de leer los datos del sensor de personas y procesar los resultados obtenidos. Inicialmente, define una estructura `SensorResults` y establece el número de rostros detectados, `num_faces`, en 0. A continuación intenta leer los resultados del sensor de personas a través de la conexión I_2C. En caso de que el proceso de lectura falle, muestra un mensaje de error y retorna los resultados con `num_faces` en 0. En caso de que la lectura se realice con éxito, muestra el valor de `num_faces` detectado y, para cada rostro, muestra los valores `box_confidence` e `is_facing`. Posteriormente, actualiza los resultados almacenados en la estructura `SensorResults`, incluyendo el valor de `box_confidence` de la detección, y si el rostro está mirando al sensor. Finalmente, retorna la estructura `SensorResults` actualizada con los datos procesados del módulo *Person Sensor*.

```
SensorResults camera_person_sensor() {
  SensorResults sensor_results;
  sensor_results.num_faces = 0;
  person_sensor_results_t results = {};

  if (!person_sensor_read(&results)) {
    Serial.println("No person sensor results found on the i2c bus");
    delay(SAMPLE_DELAY_MS);
    return sensor_results;
  }

  Serial.println("NUM_FACES = " + String(results.num_faces));
  for (int i = 0; i < MAX_num_faces; ++i) {
    if (i < results.num_faces) {
```

```
      Serial.println("BOX_CONFIDENCE[" + String(i) + "] = " +
String(results.faces[i].box_confidence));
      Serial.println("IS_FACING[" + String(i) + "] = " +
String(results.faces[i].is_facing));
      sensor_results.box_confidence[i] = results.faces[i].box_confidence;
      sensor_results.is_facing = (results.faces[i].is_facing == 1) ?
bitWrite(sensor_results.is_facing, i, 1) : bitWrite(sensor_results.is_facing,
i, 0);
      Serial.println(sensor_results.is_facing, HEX);
    } else {
      sensor_results.box_confidence[i] = 0;
      bitWrite(sensor_results.is_facing, i, 0);
      Serial.println(sensor_results.is_facing, HEX);
    }
  }

  sensor_results.num_faces = results.num_faces;
  return sensor_results;
}
```

La función `ttn_send()` se encarga de enviar los datos procesados al Servidor de Red TTN mediante el protocolo LoRaWAN. Para ello, toma como argumentos un puntero a la estructura `SensorResults`, que contiene los datos a enviar y el tamaño de los datos. En primer lugar, muestra un mensaje indicando que se están enviando datos a la plataforma TTN, seguido del tamaño de los datos, y posteriormente, de los datos en formato hexadecimal. A continuación, muestra el estado actual del modo ADR, la tasa de datos y la máscara de canal. En caso de que la máscara de canal no sea la esperada, se ajusta y se muestra la máscara actualizada. Posteriormente, inicializa el paquete LoRaWAN y escribe los datos en el paquete. En caso de que el proceso de envío del paquete sea exitoso, muestra un mensaje de confirmación; de lo contrario, muestra un mensaje de error. Finalmente, espera un tiempo para recibir posibles paquetes de respuesta desde el servidor de la plataforma TTN. En caso de que reciba un paquete *downlink*, lo procesa y muestra los datos recibidos.

```
void ttn_send(struct SensorResults* data, uint16_t dataSize) {
```

```
int err;
Serial.print("- Sending message to TTN with data (");
Serial.print(String(dataSize) + ") ");
for (unsigned int j = 0; j < dataSize; j++) {
  Serial.print(((uint8_t*)data)[j], HEX);
  Serial.print(" ");
}
Serial.println();

Serial.print("   - Current ADR / DR / Channel mask: ");
Serial.print(modem.getADR());
Serial.print(" / ");
Serial.print(modem.getDataRate());
Serial.print(" / ");
Serial.println(modem.getChannelMask());
Serial.print("   - Current channel mask: ");
Serial.println(modem.getChannelMask());

if (modem.getChannelMask() != "0003") {
  for (unsigned int i = 2; i < 8; i++) {
    if (modem.disableChannel(i)) {
      Serial.print(".");
    } else {
      Serial.print("F");
    }
  }
  Serial.print(" - channel mask: ");
  Serial.println(modem.getChannelMask());
}

modem.beginPacket();
modem.write((uint8_t*)data, dataSize);
err = modem.endPacket();
if (err > 0) {
  Serial.println("- Message sent correctly!");
} else {
  Serial.println("- Error sending message!");
}

delay(15000);
```

```
  int packetSize = modem.parsePacket();
  if (packetSize) {
    Serial.print("- Received message from TTN with data: ");
    char data[64];
    int i = 0;
    while (modem.available()) {
      data[i++] = (char)modem.read();
    }
    for (unsigned int j = 0; j < i; j++) {
      Serial.print(data[j] >> 4, HEX);
      Serial.print(data[j] & 0x0F, HEX);
      Serial.print(" ");
    }
    Serial.println();
  }
}
```

La función `ttn_data_string` toma como argumentos un puntero a la estructura `SensorResults`, que contiene los datos a enviar y el tamaño de los datos. A continuación, itera sobre los datos, convirtiendo cada byte en formato hexadecimal, y concatenándolos en una cadena de texto junto con un mensaje que indica que se están enviando datos a la plataforma TTN. Finalmente, devuelve la cadena de texto resultante.

```
String ttn_data_string(struct SensorResults* data, uint16_t dataSize) {
  String result = "- Sending message to TTN with data: ";
  for (unsigned int j = 0; j < dataSize; j++) {
    char hex_str[3];
    sprintf(hex_str, "%02X", ((uint8_t*)data)[j]);
    result += hex_str;
    result += " ";
  }
  return result;
}
```

Por su parte, la función `sleep()` se inicia estableciendo el dispositivo en modo de bajo consumo de energía y mostrando un mensaje que indica este estado. A continuación, configura el módulo LoRa® del dispositivo Arduino MKRWAN 1310 en modo de bajo consumo, así como el módulo *Person Sensor*, además de configurar los pines asociados al módulo LoRa® a nivel bajo. Posteriormente configura una alarma mediante el uso del RTC del dispositivo Arduino MKRWAN 1310 con el fin de salir del modo de bajo consumo después de un cierto período de tiempo. Tras desactivar el controlador USB, se apaga el LED integrado en el dispositivo Arduino MKRWAN 1310 para, finalmente, entrar en modo de bajo consumo profundo utilizando la función `LowPower.deepSleep()`.

```
void sleep() {
  Serial.print("  ... Putting device into deepSleep mode ... ");
  modem.sleep();
  digitalWrite(alimentacion_sensor_person, LOW);
  digitalWrite(LORA_IRQ_DUMB, LOW);
  digitalWrite(LORA_BOOT0, LOW);
  digitalWrite(LORA_RESET, LOW);
  pinMode(LORA_IRQ_DUMB, INPUT);
  pinMode(LORA_BOOT0, INPUT);
  pinMode(LORA_RESET, INPUT);

  int rtc_alarm = (rtc.getMinutes() + RTC_MATCH_MINUTE) % 60;
  rtc.setAlarmMinutes(rtc_alarm);
  rtc.setAlarmSeconds(RTC_MATCH_SECOND);
  rtc.enableAlarm(rtc.MATCH_MMSS);
  rtc.attachInterrupt(alarmMatch);

  USBDevice.detach();
  digitalWrite(LED_BUILTIN, LOW);
  LowPower.deepSleep();
}
```

La función `wakeup()` comienza activando el sensor de personas estableciendo el pin `alimentacion_sensor_person` a nivel alto. A continuación, se vuelve a activar el controlador USB y se espera un breve período de tiempo para permitir que el puerto serie

se abra. Finalmente, muestra un mensaje indicando que el dispositivo Arduino MKRWAN 1310 ha salido del modo de bajo consumo.

```
void wakeup() {
  digitalWrite(alimentacion_sensor_person, HIGH);
  USBDevice.attach();
  delay(1000);
  Serial.println(" wake up from deepSleep!");
  Serial.println();
}
```

En la función `setup()` del *sketch* se configuran los pines de salida para el LED integrado en el dispositivo Arduino MKRWAN 1310 y el control de la alimentación del módulo *Person Sensor*. El pin de alimentación se establece a nivel alto para alimentar el sensor. Se inicializa la comunicación I_2C con `Wire.begin()` y se inicia el módulo de tiempo real (RTC) con `rtc.begin()`.

```
void setup() {
  pinMode(LED_BUILTIN, OUTPUT);
  pinMode(alimentacion_sensor_person, OUTPUT);
  digitalWrite(alimentacion_sensor_person, HIGH);
  Wire.begin();
  rtc.begin();
}
```

En el bucle principal `loop()` del *sketch*, se intenta en primer lugar establecer la conexión con el servidor de la plataforma TTN, invocando la función `ttn_begin()`. En caso de que la conexión se realice con éxito, se procede a realizar las siguientes acciones: se crea una instancia de la estructura `SensorResults`, llamada `resultados_sensor_person`, que se inicializa con ceros utilizando la función `` `memset` ``, se obtienen los datos del sensor de personas mediante la función `camera_person_sensor()`, y se asignan a esta instancia. A continuación, se lee el valor analógico del pin `A1` y se convierte en porcentaje

de carga de la batería del dispositivo Arduino MKRWAN 1310, almacenándose el valor en `resultados_sensor_person.porcentaje_carga`. Posteriormente, se envían los datos al servidor de la plataforma TTN utilizando las funciones `ttn_send()` y `ttn_data_string()` con el fin de enviar la estructura de datos y convertirla en una cadena de texto, respectivamente. Después de completar estas acciones, el dispositivo Arduino MKRWAN 1310 entra en un estado de bajo consumo mediante la llamada a la función `sleep()`. Una vez que el dispositivo sale del modo de bajo consumo, tras invocar la función `wakeup()`, se incrementa el contador `counter` para llevar un registro de las iteraciones del bucle.

```
void loop() {
  ttn_begin();
  if (ttn_join()) {
    SensorResults resultados_sensor_person;
    memset(&resultados_sensor_person, 0, sizeof(resultados_sensor_person));
    resultados_sensor_person = camera_person_sensor();
    int valor_analogico = analogRead(A1);
    resultados_sensor_person.porcentaje_carga = (uint16_t)map(valor_analogico,
402, 620, 0, 100);
    ttn_send((struct SensorResults*)&resultados_sensor_person,
sizeof(resultados_sensor_person));
    ttn_data_string((struct SensorResults*)&resultados_sensor_person,
sizeof(resultados_sensor_person));
  }
  sleep();
  wakeup();
  counter++;
}
```

Una vez interconectado el módulo *Person Sensor* con el dispositivo Arduino MKRWAN 1310, y éste al puerto serie USB de un PC y con ambos *Gateways* operativos, en primer lugar, la puerta de enlace RAK7268, y posteriormente el *Gateway* Heltec HT-M00, se procede a la captura y análisis del tráfico de la red LoRaWAN.

Como se observa en la Figura 110, el registro que se muestra a través del terminal serie USB a partir de la ejecución del *sketch* en el dispositivo Arduino MKRWAN 1310, revela en primer lugar que el proceso de inicialización se ha realizado satisfactoriamente.

```
22:13:01.641 -> Iteration Number: 49
22:13:01.734 -> * Attempting to connect to TTN network server (OTAA) ... You're connected to TTN network server
22:13:08.323 ->  - channel mask: 0003
22:13:08.363 -> NUM_FACES = 0
22:13:08.363 -> BOX_CONFIDENCE[0] = 0
22:13:08.363 -> IS_FACING[0] = 0
22:13:08.363 -> BOX_CONFIDENCE[1] = 0
22:13:08.363 -> IS_FACING[1] = 0
22:13:08.363 -> BOX_CONFIDENCE[2] = 0
22:13:08.363 -> IS_FACING[2] = 0
22:13:08.363 -> BOX_CONFIDENCE[3] = 0
22:13:08.363 -> IS_FACING[3] = 0
22:13:08.363 -> 569
22:13:08.363 -> 41
22:13:08.363 -> - Sending message to TTN with data (8) 0 0 0 0 0 0 41 0
22:13:08.363 ->    - Current ADR / DR / Channel mask: 1 / 5 / 0003
22:13:08.397 ->    - Current channel mask: 0003
22:13:08.486 -> - Message sent correctly!
22:13:23.549 ->   ... Putting device into deepSleep mode ...  wake up from deepSleep!
22:14:01.793 ->
22:14:01.793 -> Iteration Number: 50
22:14:01.825 -> * Attempting to connect to TTN network server (OTAA) ... Failed to connect to TTN!
22:15:02.875 ->   ... Putting device into deepSleep mode ...
```

Figura 110 Secuencia de Inicialización y Operación del Dispositivo Arduino MKRWAN 1310.

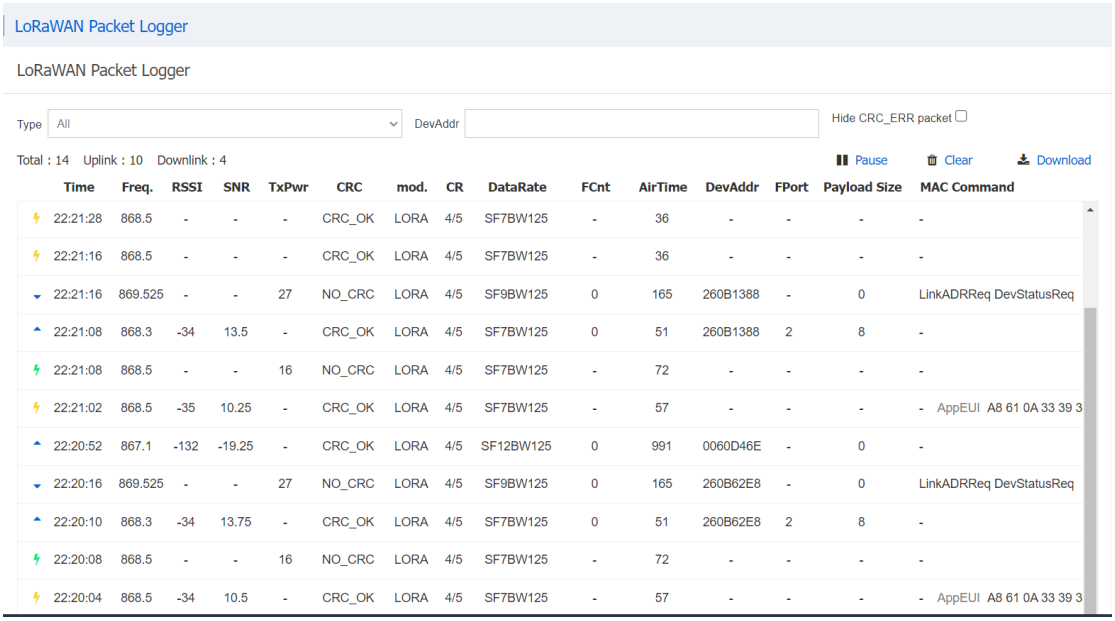
Inicialmente, se confirma la inicialización del dispositivo Arduino MKRWAN 1310, junto con la especificación de la versión del *firmware*, en este caso `ARD-078 1.2.3`. Además, se indica el EUI del dispositivo como `a8610a3339306710`. Posteriormente, se lleva a cabo un intento de conexión al servidor TTN de la red LoRaWAN utilizando el método de activación *Over The Air* (OTAA), culminando con éxito la conexión con la plataforma TTN.

La máscara de canal configurada se especifica en formato hexadecimal como `0x0003`, indicando que solo los canales 0 y 1 están habilitados para la comunicación LoRa®, lo que refleja un enfoque de comunicación selectiva determinado por las limitaciones de la puerta de enlace Heltec HT-M00, como se ha comentado en secciones precedentes. Inicialmente no se detecta la presencia de rostros por parte del módulo *Person Sensor*, como se evidencia tanto por el valor de `NUM_FACES` igual a 0, como por los valores nulos en los campos `BOX_CONFIDENCE` e `IS_FACING`.

A continuación, se envía un mensaje a la plataforma TTN con los datos recopilados, seguido por la confirmación de que el mensaje se ha enviado correctamente. El dispositivo opera con una tasa de datos de DR5, como se indica en el registro. Finalmente,

el dispositivo Arduino MKRWAN 1310 entra en modo de suspensión profunda (*deepSleep*) con el objetivo de optimizar el consumo de energía, lo que indica una práctica de gestión de recursos para optimizar la eficiencia energética del dispositivo durante los periodos de inactividad.

Los datos presentados en la Figura 111 muestran el tráfico asociado a la comunicación LoRa® entre el dispositivo Arduino MKRWAN 1310 y la puerta de enlace RAK7268, registrados en intervalos de tiempo específicos.



Time	Freq.	RSSI	SNR	TxPwr	CRC	mod.	CR	DataRate	FCnt	AirTime	DevAddr	FPort	Payload Size	MAC Command
22:21:28	868.5	-	-	-	CRC_OK	LORA	4/5	SF7BW125	-	36	-	-	-	-
22:21:16	868.5	-	-	-	CRC_OK	LORA	4/5	SF7BW125	-	36	-	-	-	-
22:21:16	869.525	-	-	27	NO_CRC	LORA	4/5	SF9BW125	0	165	260B1388	-	0	LinkADRReq DevStatusReq
22:21:08	868.3	-34	13.5	-	CRC_OK	LORA	4/5	SF7BW125	0	51	260B1388	2	8	-
22:21:08	868.5	-	-	16	NO_CRC	LORA	4/5	SF7BW125	-	72	-	-	-	-
22:21:02	868.5	-35	10.25	-	CRC_OK	LORA	4/5	SF7BW125	-	57	-	-	-	- AppEUI A8 61 0A 33 39 3
22:20:52	867.1	-132	-19.25	-	CRC_OK	LORA	4/5	SF12BW125	0	991	0060D46E	-	0	-
22:20:16	869.525	-	-	27	NO_CRC	LORA	4/5	SF9BW125	0	165	260B62E8	-	0	LinkADRReq DevStatusReq
22:20:10	868.3	-34	13.75	-	CRC_OK	LORA	4/5	SF7BW125	0	51	260B62E8	2	8	-
22:20:08	868.5	-	-	16	NO_CRC	LORA	4/5	SF7BW125	-	72	-	-	-	-
22:20:04	868.5	-34	10.5	-	CRC_OK	LORA	4/5	SF7BW125	-	57	-	-	-	- AppEUI A8 61 0A 33 39 3

Figura 111 Paquetes LoRaWAN filtrados por el Gateway RAK7268.

A lo largo del registro, se observan múltiples solicitudes de tipo *Join Request* enviadas por el dispositivo Arduino MKRWAN 1310 en diferentes momentos. Estas solicitudes fueron recibidas correctamente, como se indica por la presencia de `CRC_OK`, utilizando la modulación LoRa® con una tasa de codificación de 4/5 y una tasa de datos de SF7BW125. La ausencia de valores para RSSI y SNR en estas solicitudes indica que en esos instantes no se midieron o registraron estos parámetros.

Durante la operación, también se registraron datos no confirmados de subida y bajada (*Unconfirmed* Data Up y *Unconfirmed Data Down*). Como referencia, a las 22:21:08, se registró un paquete de datos no confirmados de subida a una frecuencia de 868.3 MHz, con un RSSI de -34 dBm y un SNR de 13.5 dB. Este paquete, que tenía un tamaño de 8 bytes, incluía un *Payload* específico y fue correctamente recibido con `CRC_OK`. Similarmente, se registraron varios paquetes de datos *downlink* no confirmados a frecuencias de 869.525 MHz y 868.3 MHz, con detalles específicos sobre la ausencia de CRC y el tamaño del *Payload.* Las solicitudes de aceptación de unión *Join Accept* también se registraron, indicando que el dispositivo fue autorizado para unirse a la red LoRaWAN. Estos eventos se producen en diferentes momentos y frecuencias, con la modulación y la tasa de codificación correspondientes.

En la Figura 112, la Figura 113, la Figura 114 y la Figura 115 se muestran, de manera desglosada, los parámetros asociados a algunos de estos paquetes.

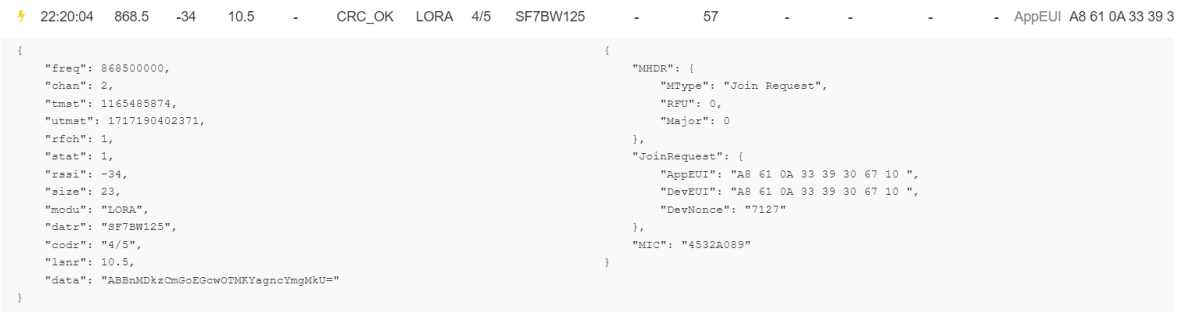


Figura 112 Paquete LoRaWAN (I).

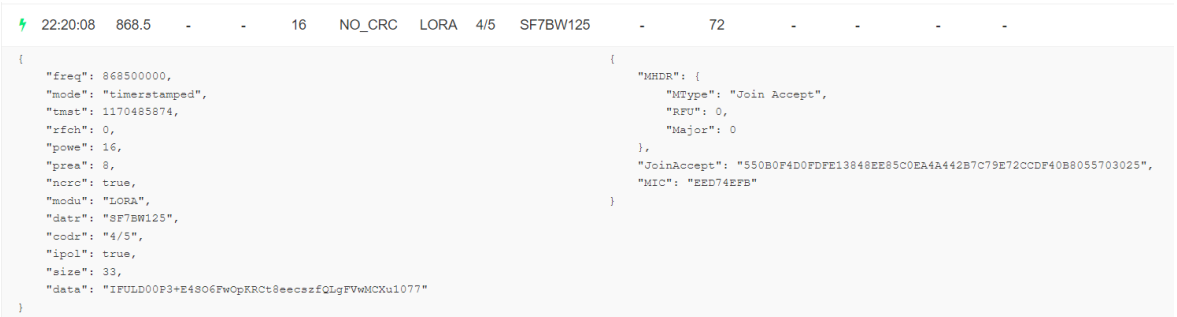


Figura 113 Paquete LoRaWAN (II).

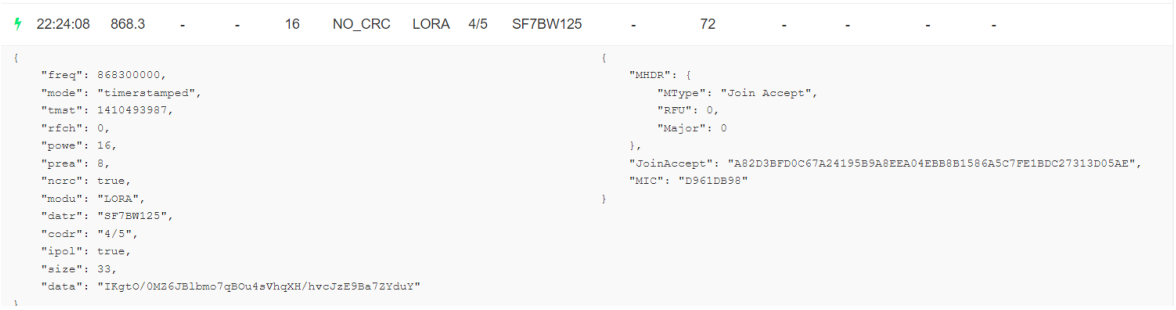


Figura 114 Paquete LoRaWAN (III).

Figura 115 Paquete LoRaWAN (IV).

A partir de la ejecución del *sketch* `mkr1310_lorawan_person_sensor_v0.2.ino`, el dispositivo Arduino MKRWAN 1310 está habilitado para la transferencia de paquetes LoRa® en las frecuencias correspondientes a los canales 0 y 1 (868.1 MHz y 868.3 MHz), lo que hace que la mayor parte del tráfico gestionado por la puerta de enlace se concentre en estos valores, como se muestra en la Figura 116.

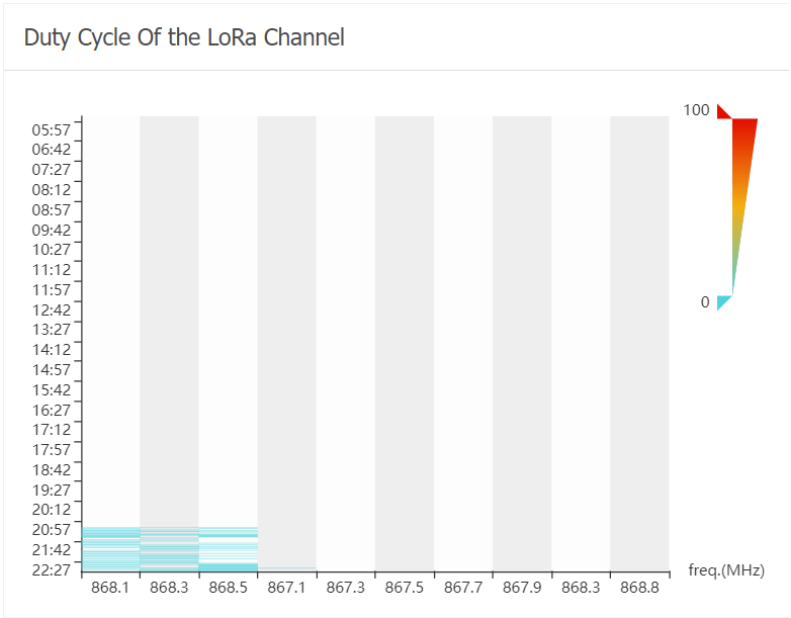


Figura 116 Frecuencias de trabajo del dispositivo Arduino MKRWAN 1310 y el Gateway RAK7268.

En el contexto de LoRaWAN correspondiente a la región EU868, los dispositivos pueden operar en varias frecuencias, siendo 868.1 MHz, 868.3 MHz y 868.5 MHz frecuencias obligatorias para todos ellos. Estas frecuencias son utilizadas para enviar solicitudes de unión (*join-requests*) y deben tener un ancho de banda de 125 kHz, operando en tasas de datos (*Data Rates*) de DR0 a DR5. El ciclo de trabajo (*duty cycle*) para estas frecuencias está limitado al 1%, lo que significa que un dispositivo puede transmitir durante un máximo del 1% del tiempo, para evitar interferencias perjudiciales con otros dispositivos en la misma banda [44].

El dispositivo Arduino MKR WAN1310 puede transmitir con tasas de datos distintas a DR5, como se muestra en la Figura 117 y en la Figura 118, a pesar de estar configurado principalmente para DR5 debido a los mecanismos de tasa de datos adaptativa (ADR) y las optimizaciones de red implementadas en LoRaWAN. Esta flexibilidad en el uso de la tasa de datos ayuda a optimizar el rendimiento de la red y la vida útil de la batería del dispositivo. La característica ADR de LoRaWAN permite ajustar dinámicamente la tasa de datos de cada dispositivo en función de las condiciones actuales de la red, la calidad de la señal y otros parámetros para asegurar una comunicación eficiente [45].

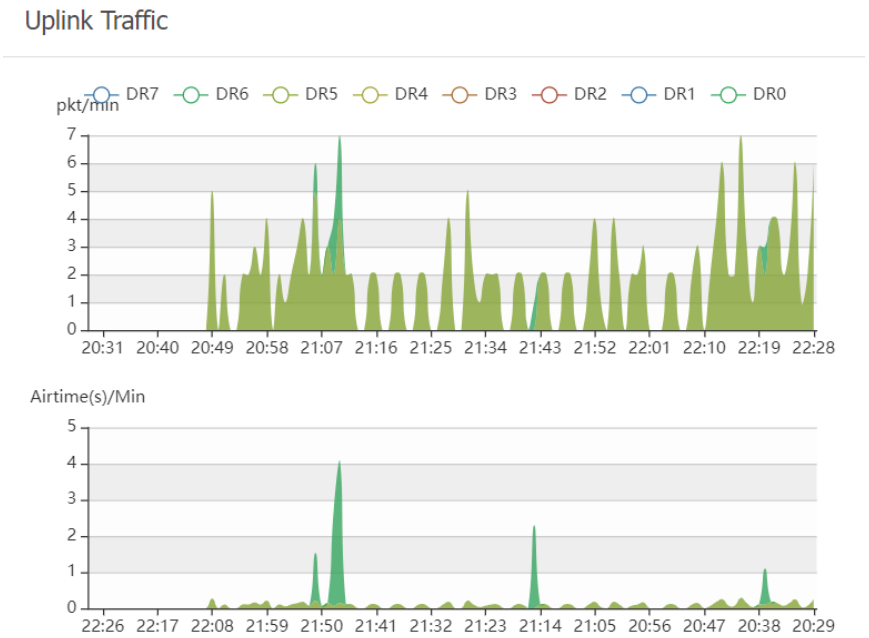


Figura 117 Tasas de datos de Recepción del Gateway RAK.

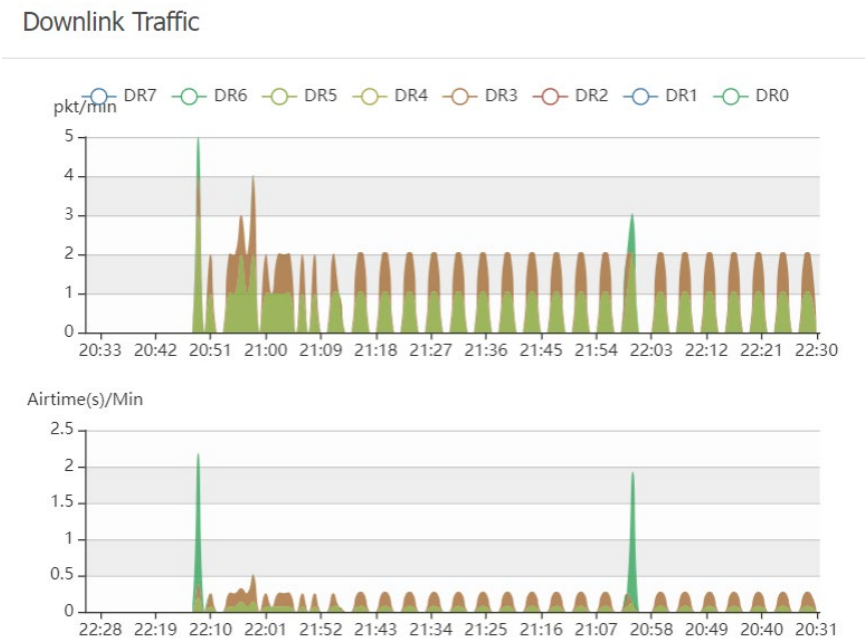


Figura 118 tasa de datos de transmisión del Gateway RAK.

Por otra parte, en la Figura 119 se pueden observar los datos trasferidos desde/hacia la plataforma TTN a través de la puerta de enlace RAK7268, mientras que en la Figura 120 se muestran los datos correspondientes al dispositivo Arduino MKRWAN 1310 integrado en la aplicación creada en TTN.

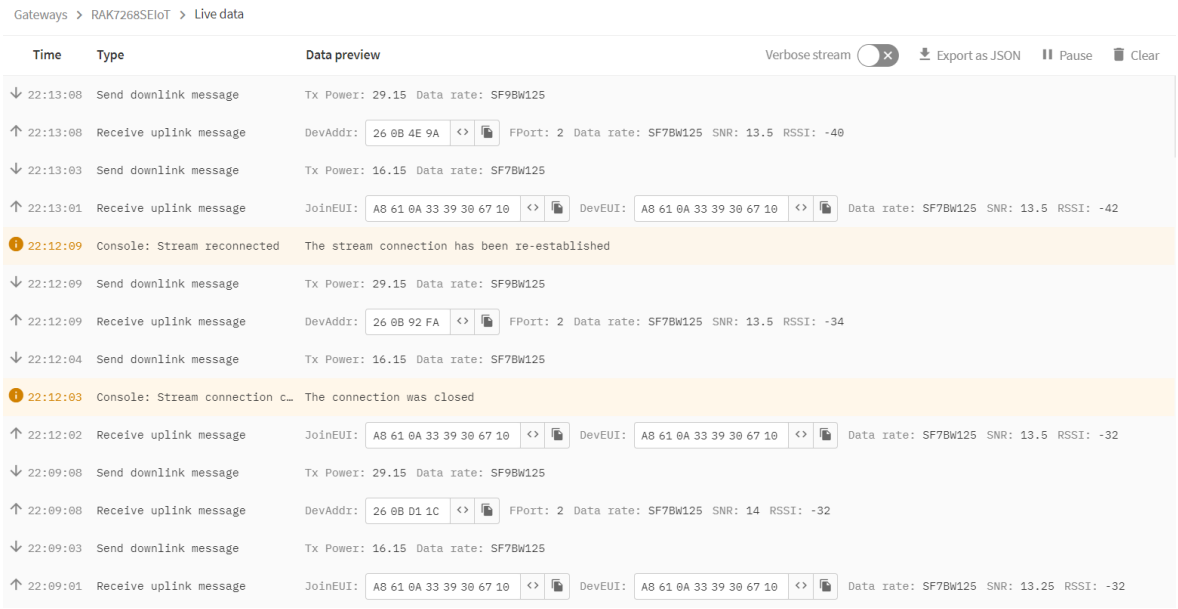


Figura 119 Live Data en plataforma TTN de dispositivo RAK7268.

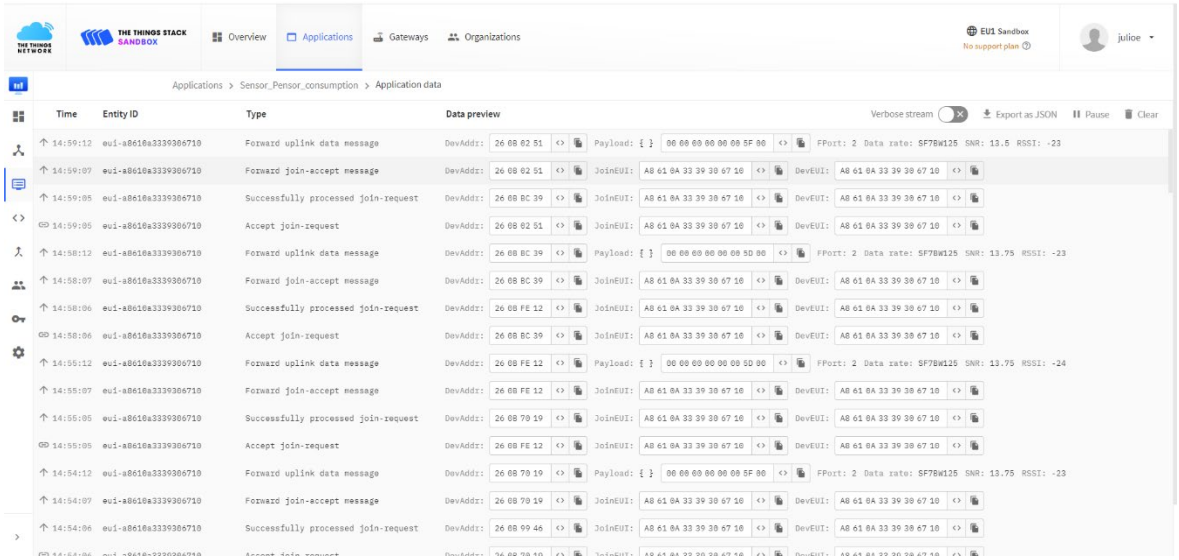


Figura 120 Live Data en plataforma TTN de dispositivo Arduino MKRWAN 1310.

Sin embargo, la decodificación precisa de los datos recibidos en la plataforma TTN puede resultar fundamental para analizar e interpretar correctamente la información recibida. Así, la función `Decoder` implementada en el desarrollo del presente TFM, permite transformar los datos crudos recibidos en la plataforma TTN desde el dispositivo Arduino MKRWAN 1310, en información estructurada y comprensible. Este proceso es esencial para la monitorización y la gestión eficiente de la información. En la Figura 121 se presenta la función implementada para decodificar y organizar la información recibida desde el dispositivo Arduino MKRWAN 1310 registrado, disponible en la sección *end devices* de la plataforma TTN, en concreto, en el apartado *payload formatters* para mensajes *uplink*.

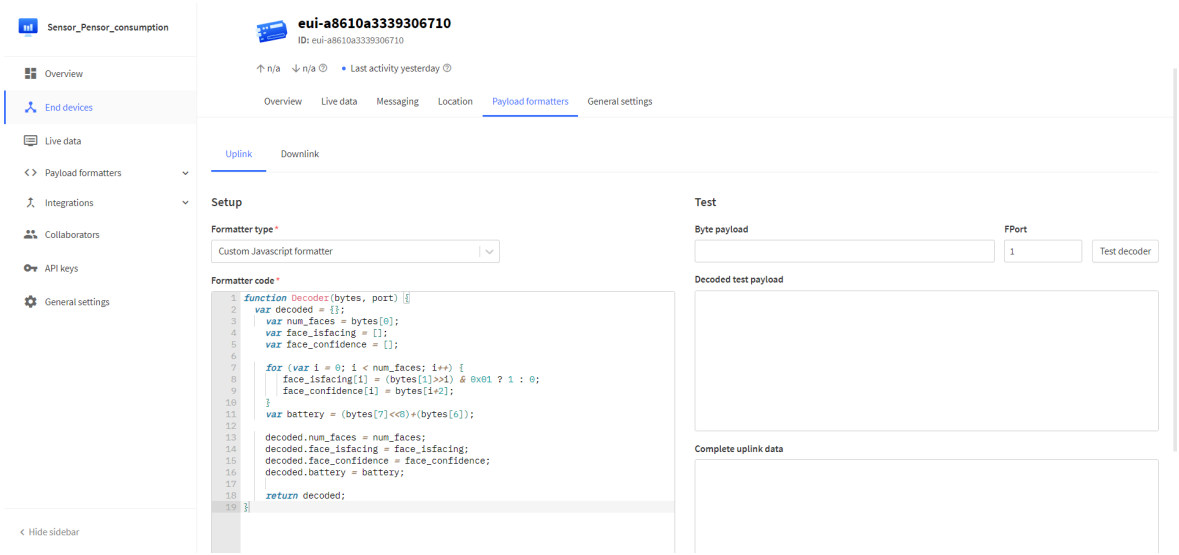


Figura 121 Funcion decodificadora de Payload en TTN.

Esta función, denominada `Decoder`, recibe dos parámetros: `bytes`, que constituye un array de bytes del mensaje *uplink*, y `port`, que indica el puerto desde el cual se recibió el mensaje. Inicialmente, se define un objeto vacío de tipo `decoded` destinado a almacenar los datos decodificados. Seguidamente, se extrae el número de caras detectadas del primer byte del array `bytes`, almacenándose este valor en la variable `num_faces`. Para registrar si cada rostro está orientado hacia el sensor, así como la confianza en la detección de cada cara, se inicializan los arrays `face_isfacing` y `face_confidence`.

```
function Decoder(bytes, port) {
  var decoded = {};
    var num_faces = bytes[0];
        var face_isfacing = [];
        var face_confidence = [];

    for (var i = 0; i < num_faces; i++) {
       face_isfacing[i] = (bytes[1]>>i) & 0x01 ? 1 : 0;
       face_confidence[i] = bytes[i+2];
```

```
    }
    var battery = (bytes[7]<<8)+(bytes[6]);

    decoded.num_faces = num_faces;
    decoded.face_isfacing = face_isfacing;
    decoded.face_confidence = face_confidence;
    decoded.battery = battery;

  return decoded;
```

Por otro lado, dentro de un bucle, que se ejecuta según el número de rostros que detecte, se determina si cada rostro observa al sensor mediante una operación bit a bit, almacenándose resultado en el array `face_isfacing`. La confianza en la detección de cada rostro se extrae de los bytes correspondientes y se almacena en `face_confidence`.

Posteriormente, se calcula el nivel de batería combinando los bytes 6 y 7 del array bytes, y se almacena en la variable `battery`. Finalmente, las variables `num_faces`, `face_isfacing`, `face_confidence` y `battery` se asignan al objeto `decoded`. Este objeto, que ahora contiene una estructura de datos clara y organizada con la información decodificada del mensaje *uplink*, se devuelve al final de la función.

Esta implementación facilita una interpretación efectiva y estructurada de los datos transmitidos desde los nodos IoT a través de TTN, permitiendo la extracción y análisis de información crítica, como el número de rostros detectados, su orientación, la confianza en las detecciones y el nivel de batería del dispositivo.

6 Integración con InfluxDB

Los datos de series temporales han estado asociados históricamente con aplicaciones en el campo de las finanzas. Sin embargo, a medida que los desarrolladores y las empresas avanzan en la instrumentación de más servidores, aplicaciones, infraestructura de red y el mundo físico, las series temporales se están convirtiendo en el estándar de facto para el almacenamiento, análisis y recuperación de estos datos en tiempo real [5].

Las series temporales representan mediciones o eventos que se registran, se supervisan, y se agregan a lo largo del tiempo. Esto puede incluir métricas de servidores, seguimiento del rendimiento de aplicaciones, datos de redes, datos de sensores, eventos, transacciones en un mercado, y muchos otros tipos de datos analíticos. La diferencia clave que diferencia los datos de series temporales de los datos regulares es que siempre se están formulando consultas sobre ellos a lo largo del tiempo [5].

En los últimos años, las series temporales se han convertido en un caso de uso común en muchas industrias y en una categoría de base de datos propia. Las métricas, eventos y otros datos basados en el tiempo se generan a un ritmo exponencial, existiendo una creciente necesidad de analizar entornos complejos [5].

En la actualidad, muchas empresas en todo el mundo optan por el uso de InfluxDB en el desarrollo de aplicaciones de IoT, al ofrecer la posibilidad de integrarse con otras soluciones para satisfacer diferentes requisitos [5]. A diferencia de las bases de datos relacionales o de documentos, la plataforma de código abierto InfluxDB organiza los datos de series temporales con una estructura definida. Esta estructura facilita a los desarrolladores la creación de herramientas en torno a las API que proporciona InfluxData.

6.1 InfluxDB Cloud

InfluxDB Cloud es una plataforma avanzada diseñada específicamente para el registro, almacenamiento y consulta de datos de series temporales. Esta plataforma está soportada por el motor de almacenamiento InfluxDB 3.0, el cual ofrece múltiples beneficios,

incluyendo una cardinalidad de series casi ilimitada, un rendimiento de consultas mejorado, y una interoperabilidad eficiente con diversas plataformas y herramientas de procesamiento de datos ampliamente utilizadas.

Los datos de series temporales son secuencias de puntos de datos indexados cronológicamente. Cada punto de datos generalmente representa una medición sucesiva de una misma fuente, permitiendo así el seguimiento de cambios a lo largo del tiempo. Ejemplos de estos datos incluyen los provenientes de sensores industriales, métricas de rendimiento de servidores, actividad eléctrica cerebral, mediciones de lluvia, o precios de acciones.

6.1.1 Organización de los Datos

El modelo de datos de *InfluxDB Cloud* organiza los datos de series temporales en bases de datos y mediciones. Una base de datos puede contener múltiples mediciones. Las mediciones contienen múltiples etiquetas y campos.

- ***Database***: Ubicación en la que se almacenan los datos de series temporales. Una base de datos puede contener múltiples mediciones.
- ***Measurement***: Agrupación lógica para datos de series temporales. Todos los puntos asociados a una medición deben tener las mismas etiquetas. Una medición contiene múltiples etiquetas y campos.
- ***Tags***: Pares clave/valor que proporcionan metadatos para cada punto, por ejemplo, algo que identifique la fuente o el contexto de los datos como *host*, ubicación, estación, etc. Los valores de las etiquetas pueden ser nulos.
- ***Fields***: Pares clave/valor con valores que cambian con el tiempo, por ejemplo, temperatura, presión, etc. Los valores de los campos pueden ser nulos, pero al menos un valor de campo debe ser no nulo en cualquier fila.
- ***Timestamp***: Marca de tiempo asociada con los datos. Cuando se almacenan en disco y se consultan, todos los datos están ordenados en función del tiempo. Una marca de tiempo nunca es nula.

6.1.2 Esquema de escritura

Al usar InfluxDB, se puede definir el esquema a medida que se escriben los datos, sin necesidad de crear mediciones (equivalentes a una tabla relacional) ni de definir explícitamente el esquema de la medición. Los esquemas de las mediciones se determinan por el esquema de los datos a medida que se escriben.

Por otro lado, al usar InfluxDB resulta fundamental entender algunas definiciones clave. Así, se denomina *Point* a un registro de datos único identificado por su medición, claves de etiquetas, valores de etiquetas, clave de campo y marca de tiempo. Por otro lado, se llama *Series* a un grupo de puntos que comparten la misma medición, claves y valores de etiquetas, y clave de campo. El término *Primary key* se refiere a las columnas utilizadas para identificar de manera única cada fila en una tabla. Las filas se identifican de manera única por su marca de tiempo y conjunto de etiquetas, excluyendo las etiquetas con valores nulos del conjunto de etiquetas.

En InfluxDB, todos los datos se escriben utilizando *Line Protocol*, un formato basado en texto que permite proporcionar la información necesaria para escribir un punto de datos en InfluxDB. Cada línea de *Line Protocol* contiene los siguientes elementos:

- ***measurement***: Cadena que identifica la medición en la que se almacenarán los datos.
- ***tag set***: Lista de pares clave/valor delimitadas por comas, representando cada uno una etiqueta. Las claves y valores de las etiquetas son cadenas sin comillas.
- ***field set***: Lista de pares clave/valor delimitadas por comas, cada uno representando un campo. Las claves de los campos son cadenas sin comillas. Los valores de los campos pueden ser cadenas (entre comillas), floats, enteros, enteros sin signo o booleanos.
- ***timestamp***: Marca de tiempo Unix asociada con los datos. InfluxDB soporta precisiones de nanosegundos. En caso de que la precisión de la marca de tiempo no sea de nanosegundos, es necesario especificar la precisión al escribir los datos en InfluxDB.

Este protocolo organiza cada punto de datos en base a estos elementos clave, como se representa en la Figura 123:

- ***Measurement***: El campo anterior a la primera coma que se encuentra antes del primer espacio en blanco.
- ***tag set***: Pares clave/valor entre la primera coma, y el primer espacio en blanco.
- ***field set***: Pares clave/valor entre el primer y el segundo espacio en blanco.
- ***timestamp***: Valor entero que se encuentra tras el segundo espacio en blanco.

Line protocol es sensible a los espacios en blanco, y las líneas se diferencian mediante el carácter de nueva línea (`\n`).

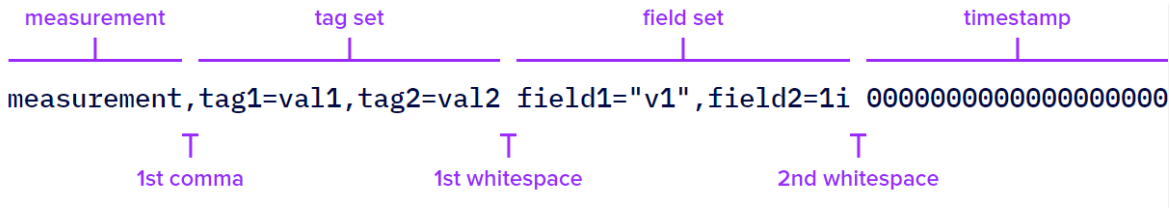


Figura 122 Estructura del Line Protocol

6.2 Telegraf

Telegraf es un agente que forma parte del ecosistema de InfluxData para recopilar datos y reportar métricas sin código. Su amplia biblioteca de complementos de entrada y su arquitectura *plug-and-play* le permiten recopilar rápida y fácilmente métricas de múltiples fuentes [46].

Telegraf es un agente escrito en Go que se basa en complementos que se habilitan y configuran en su archivo de configuración (`telegraf.conf`). Cada configuración de *Telegraf* debe tener al menos un *plugin* de entrada y un *plugin* de salida [46]. Los complementos de entrada de *Telegraf* recuperan métricas de diferentes fuentes, mientras que los complementos de salida de *Telegraf* escriben esas métricas en un destino.

6.2.1 Telegraf Plugin

Telegraf es un agente altamente extensible y modular, gracias a su arquitectura basada en *plugins*. Los *plugins* de *Telegraf* se dividen en cuatro categorías principales: *Input Plugins*, *Output Plugins*, *Processor Plugins* y *Aggregator Plugins* como se muestra en la Figura 122.

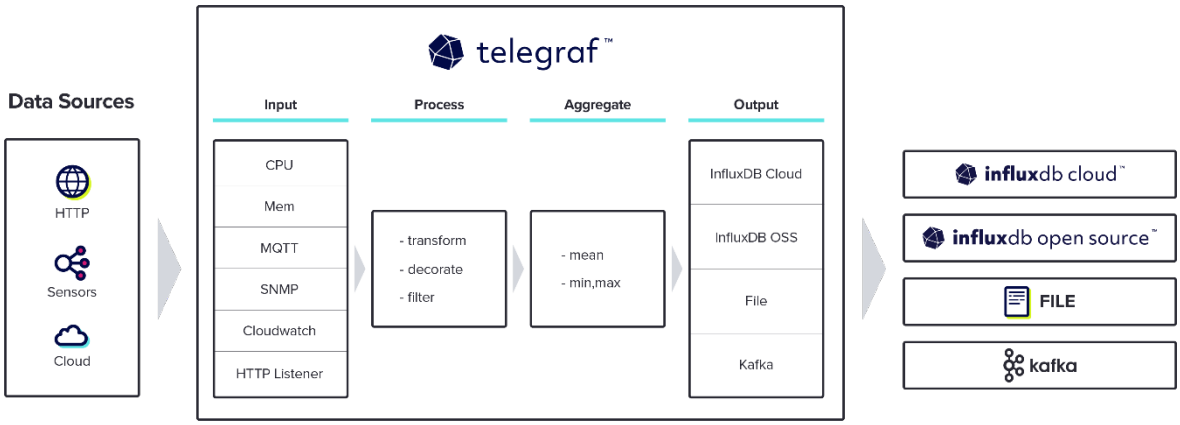


Figura 123 Visión General de Telegraf [46].

A continuación, se describen brevemente cada una de estas categorías, mencionándose algunos ejemplos de *plugins* para cada una de ellas.

1. ***Input Plugins***: Responsables de recopilar datos de diversas fuentes y enviarlos a *Telegraf*. Pueden recoger métricas de sistemas, servicios, bases de datos y aplicaciones [46]. Algunos ejemplos comunes incluyen:
    - *CPU:* Recopila métricas del uso de la CPU.
    - *Memory:* Recopila métricas del uso de la memoria.
    - *Disk:* Recopila métricas de la utilización del disco.
    - *HTTP:* Permite recopilar datos desde un *endpoint* HTTP.
    - *SNMP:* Recopila datos a través de SNMP.
2. ***Output Plugins***: Permiten enviar las métricas recopiladas por *Telegraf* a diferentes destinos, como bases de datos, servicios de monitoreo, u otros sistemas [46]. Ejemplos de *output plugins* incluyen:
    - *InfluxDB:* Envía métricas a una base de datos InfluxDB.

- *Prometheus:* Expone métricas en un formato en el que los datos pueden ser extraídos por *Prometheus*.
- *Kafka*: Envía métricas a un clúster de *Apache Kafka*.
- *Graphite*: Envía métricas al sistema de monitoreo *Graphite*.

3. ***Processor Plugins***: Permiten modificar o transformar las métricas tras haber sido recopiladas, pero antes de que sean enviadas a su destino [46]. Algunos ejemplos incluyen:
   - *Rename:* Permite cambiar el nombre de las métricas, *tags* y *fields*.
   - *Regex:* Utiliza expresiones regulares para modificar las métricas.
   - *Starlark:* Permite escribir scripts en el lenguaje *Starlark* para procesar las métricas.
4. ***Aggregator Plugins***: Permiten agregar métricas antes de enviarlas a su destino. Pueden ser útiles para reducir la cantidad de datos enviados o para calcular estadísticas a partir de las métricas recopiladas [46]. Ejemplos incluyen:
   - *Basicstats:* Calcula estadísticas básicas como media, mediana y percentiles.
   - *Histogram:* Genera histogramas a partir de las métricas.
   - *Minmax:* Calcula los valores mínimo y máximo de las métricas.

Como referencia, en la recolección de datos que se realiza en el presente TFM se utiliza el *output plugin* `outputs.influxdb_v2` para escribir métricas recopiladas por *Telegraf* en InfluxDB Cloud. La configuración de este complemento es la siguiente:

```
# ...

[[outputs.influxdb_v2]]
  urls = ["https://eu-central-1-1.aws.cloud2.influxdata.com"]
  token = "${INFLUX_TOKEN}"
  organization = "TFM"
```

```
  bucket = "Person_Sensor "


# ...
```

Adicionalmente, se utiliza el *input plugin* `inputs.mqtt_consumer` para suscribirse y recopilar datos desde un servidor MQTT de TTN. La configuración para este complemento es la siguiente:

```
[inputs.mqtt_consumer]]

  ## Broker URLs for the MQTT server or cluster.  To connect to multiple

  ## clusters or standalone servers, use a separate plugin instance.

  servers = ["eu1.cloud.thethings.network:1883"]

  ## Topics that will be subscribed to.

  topics = [

## Username and password to connect MQTT server.

   username = "test-v1-23-03-2023@ttn"

   password = ""
```

Esta configuración permite a *Telegraf* recopilar métricas de sensores que se comunican a través del protocolo MQTT haciendo uso del *broker* MQTT de TTN, y enviarlas a InfluxDB Cloud para su almacenamiento y posterior análisis.

6.3 Configuración e Integración de InfluxDB Cloud con Telegraf y TTN

InfluxDB Cloud 2.0 ofrece planes de suscripción adaptados a diferentes requisitos de aplicaciones IoT. En el caso particular del presente TFM, se hará uso de los servicios del plan gratuito de InfluxDB Cloud 2.0. Para acceder a este plan, es necesario registrarse como usuario en https://cloud2.influxdata.com/signup . El plan gratuito de InfluxDB Cloud 2.0 permite almacenar hasta un máximo de 10.000 series temporales de datos y ofrece un período de almacenamiento de hasta 30 días, como se indica en la Figura 124, la Figura 125, la Figura 126 y la Figura 127 [47]. Esta opción resulta particularmente beneficiosa para proyectos iniciales y evaluaciones, ya que proporciona una plataforma robusta sin incurrir en costes adicionales.

InfluxDB Cloud Offerings

		Serverless			Dedicated
		Free Plan	**Usage-Based Plan**	**Annual Plans**	**Dedicated**
		Rate limited plan for quickly and easily learning InfluxDB and prototyping. Single click upgrade to Usage-Based Plan	For more extensive testing and development and smaller or early stage production workloads. Pay only for what you use - no minimums or long term commitment.	Committed use plans for production or high volume workloads offering scale discounts, elevated support, and enhanced feature access	Single-tenant environment for high volume production workloads offering workload isolation, private networking, elevated support and enhanced feature access
General					
	Fully managed	✓	✓	✓	✓
	Service Environment	Serverless, Multi-tenant	Serverless, Multi-tenant	Serverless, Multi-tenant	Single-tenant
	Uptime SLA	–	–	✓	✓
	Pricing	Free	• Volume of data in: $0.0025/MB • Query count: $0.012 per 100 query executions • Storage (total disk usage): $0.002/GB-hour • Volume of data out: $0.09/GB	Usage-based, committed scale discounts	Annual subscription based on dedicated instance CPU / RAM

Figura 124 Planes de InfluxDB Cloud [47].

Scale					
	Writes	5MB/5 mins	300MB/5 mins	Contact Sales	No applied service quota
	Queries- Data Read	300MB/5 mins	3000MB/5 mins	Contact Sales	No applied service quota
	Data Retention	30 days	Unlimited	Unlimited	Unlimited
	Cardinality	Contact Sales about how to scale for high-cardinality datasets	Contact Sales about how to scale for high-cardinality datasets	Contact Sales	No applied service quota
	Database (Buckets)	2	Unlimited	Unlimited	Unlimited
	Notification Rules	2 Notification Rules	Unlimited	Unlimited	Unlimited
	Separate Organizations Allowed Per Account	1	3	20	N/A
Cloud					
	Cloud Marketplace	—	AWS, Google Cloud, Microsoft Azure	Private Offers available Contact Sales	Private Offers available Contact Sales

Figura 125 Planes de InfluxDB Cloud [47].

Security					
	End to end encryption	✓	✓	✓	✓
	SAML/SSO	—	—	✓ Available as an add-on	✓ Available as an add-on
	Private networking	—	—	—	✓ Available as an add-on
Compliance					
	SOC 2 Type 2	✓	✓	✓	✓
	ISO 27001	✓	✓	✓	✓

Figura 126 Planes de InfluxDB Cloud [47].

InfluxDB Cloud Support

	Free Plan	Usage-Based Plan	Annual Plans	Dedicated
Community forum	✓	✓	✓	✓
Slack	✓	✓	✓	✓
Cloud infrastructure monitored 24x7 (*)	✓	✓	✓	✓
Uptime SLA	Not available	Not available	✓	✓
InfluxData Help Desk Support (**)	Not available	Contact Sales	✓	✓
Dedicated customer success	Not available	Not available	✓ With qualifying annual plan	✓ With qualifying plan
Professional Services	Not available	Not available	Contact Sales	Contact Sales

(*) does not include for support for questions about using, optimizing or troubleshooting InfluxDB itself.
(**) provides access to support engineers to help troubleshoot issues and answer questions about using and optimizing InfluxDB.

Figura 127 Planes de InfluxDB Cloud [47]

InfluxDB Cloud está disponible en múltiples proveedores en la nube, así como en diferentes regiones. Cada región dispone de una URL de InfluxDB Cloud y un API *endpoint*

único, como se observa en la Figura 128 y la Figura 129. Algunas regiones de InfluxDB Cloud disponen de varios clústeres de Cloud, cada uno con una URL única [47].

Servicios web de Amazon (AWS)

Región	Ubicación	URL(s)
Oeste de EE. UU. (Oregón)	Oregón, Estados Unidos	nosotros-west-2-1: https://us-west-2-1.aws.cloud2.influxdata.com nosotros-west-2-2: https://us-west-2-2.aws.cloud2.influxdata.com
Este de EE. UU. (Virginia)	Virginia, Estados Unidos	https://us-east-1-1.aws.cloud2.influxdata.com
UE Fráncfort	Frankfurt, Alemania	https://eu-central-1-1.aws.cloud2.influxdata.com

Figura 128 Servicios Web de Amazon (AWS)[47].

Plataforma en la nube de Google (GCP)

Región	Ubicación	URL(s)
Centro de EE. UU. (Iowa)	Iowa, Estados Unidos	https://us-central1-1.gcp.cloud2.influxdata.com

MicrosoftAzure

Región	Ubicación	URL(s)
Europa Occidental (Ámsterdam)	Amsterdam, Holanda	https://westeurope-1.azure.cloud2.influxdata.com
Este de EE. UU. (Virginia)	Virginia, Estados Unidos	https://eastus-1.azure.cloud2.influxdata.com

Figura 129 Servicios Web de Google y Azure [47].

Una vez completado el registro como usuario, se puede acceder a la cuenta creada desde la página principal de InfluxDB Cloud 2.0 utilizando la opción LOG IN, y proporcionando las credenciales de correo electrónico y contraseña asociadas a la cuenta, como se muestra en la Figura 130.

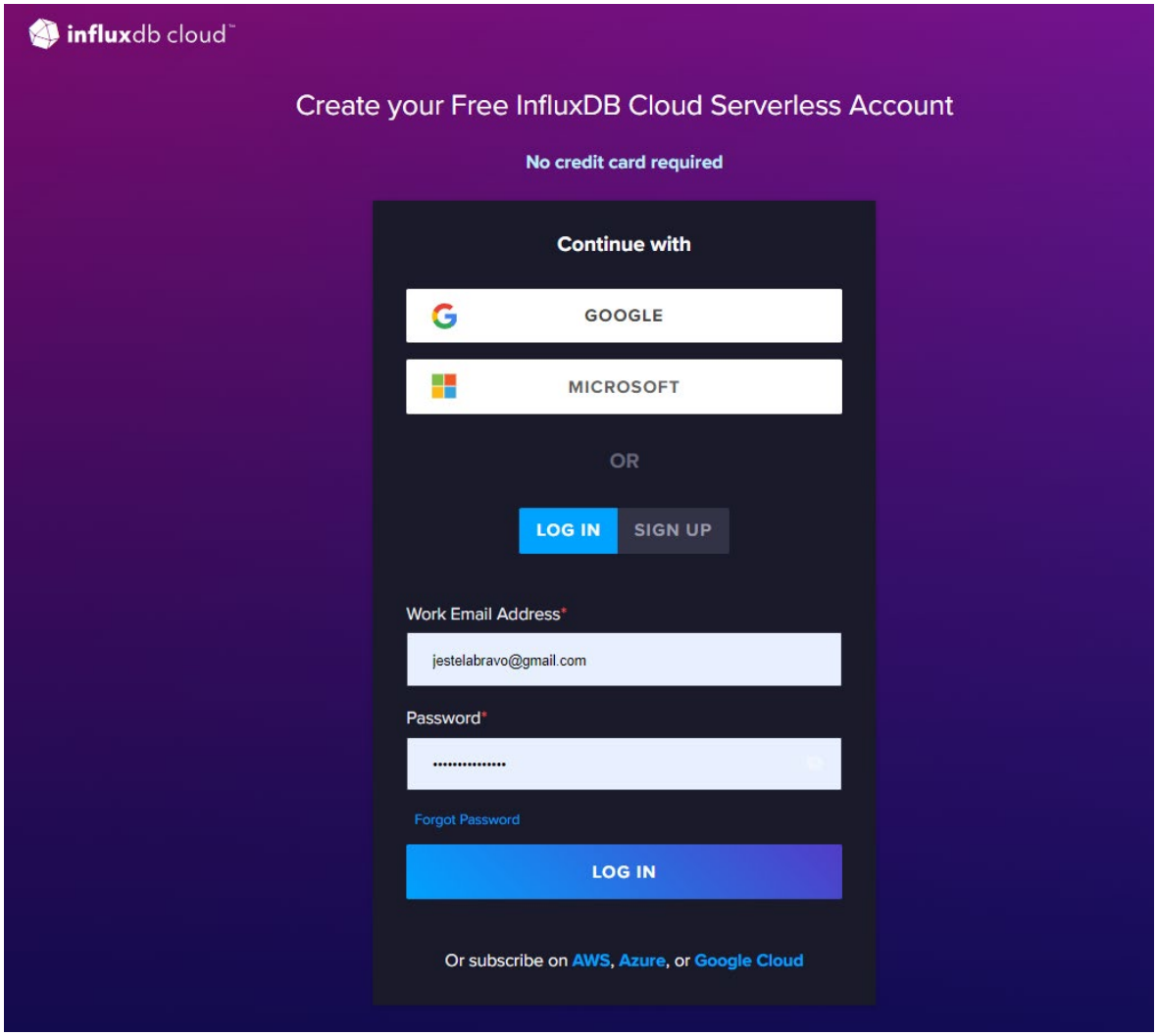


Figura 130 LOG IN en InfluxdB.

Antes de explorar su uso en el desarrollo de aplicaciones IoT, es esencial comprender la organización y el almacenamiento de las series temporales de datos en InfluxDB Cloud 2.0, así como familiarizarse con sus componentes fundamentales. Así, en InfluxDB Cloud los datos se estructuran en series temporales alojadas en contenedores denominados *buckets*. Cada *bucket* puede contener múltiples medidas (*measurements*), las cuales se componen de etiquetas (tags) y campos (*fields*). Estos componentes están asociados con

marcas temporales (*timestamps*), permitiendo un seguimiento preciso del tiempo en que se recopilaron los datos, como se representa en la Figura 131.

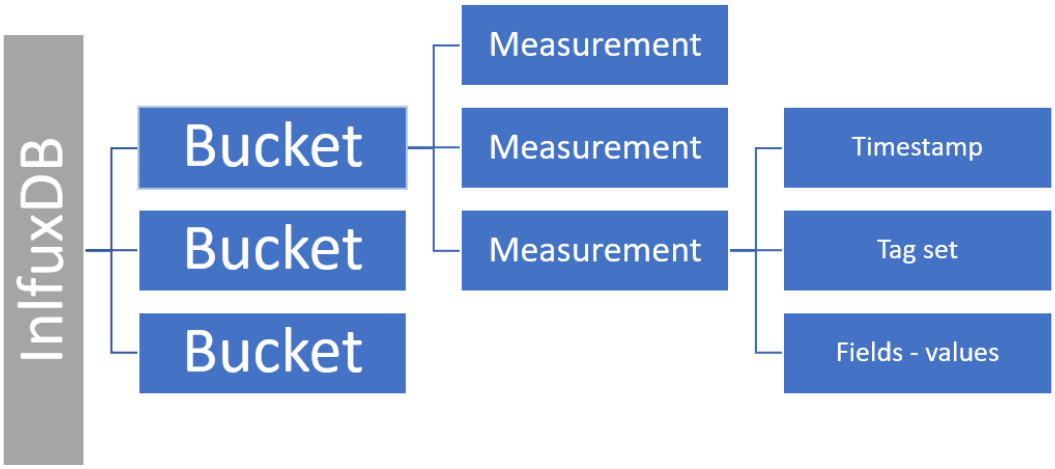


Figura 131 Estructura de los datos en InfluxDB.

Por tanto, en el proceso inicial del desarrollo de una aplicación IoT con InfluxDB Cloud 2.0, el primer paso consiste en la creación de un *bucket*. Este *bucket* se utilizará para almacenar las series temporales de datos generadas, en este caso, por el módulo Arduino MKRWAN 1310. Representa el repositorio principal para toda la información transferida desde el dispositivo, asegurando una gestión estructurada y eficiente de los datos en la aplicación.

Para la creación del *bucket* se accede a la página principal, como aparece en la Figura 132, y se selecciona la opción *Buckets*.

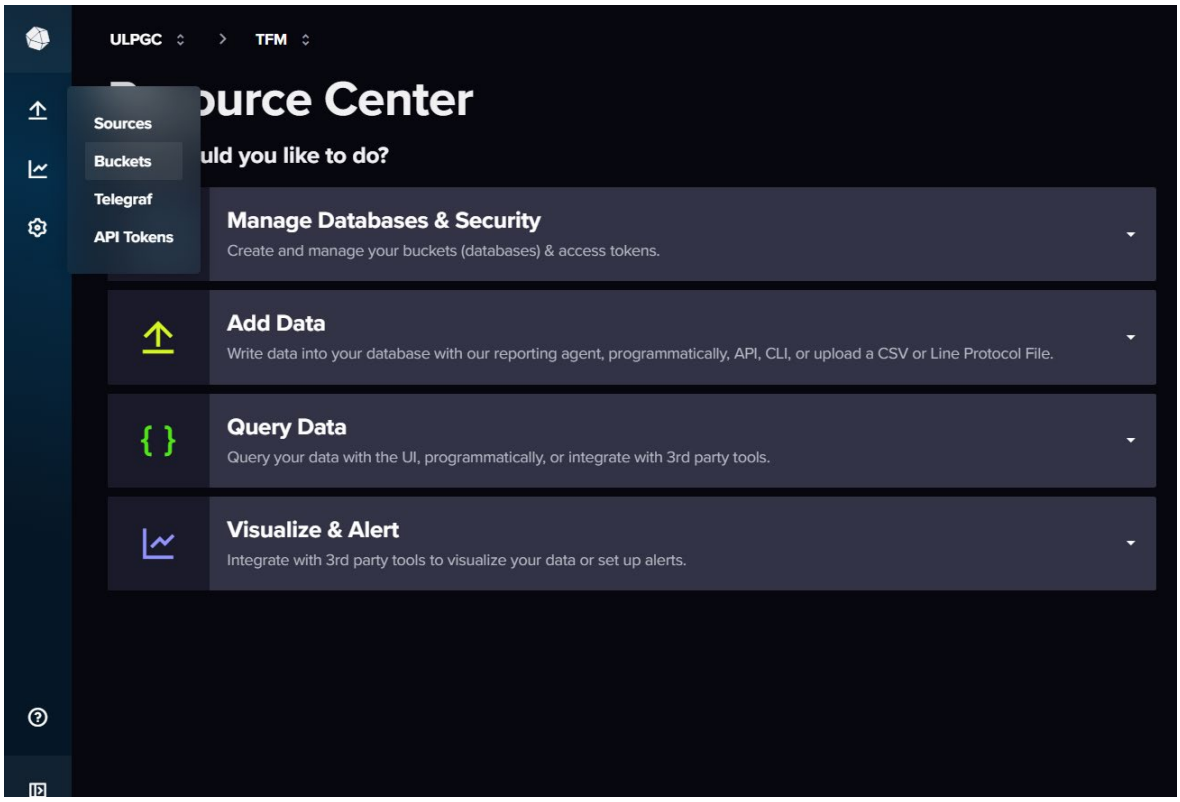


Figura 132 Página Principal de InfluxDB Cloud.

A continuación, se procede a crear el *bucket* e identificarlo con un nombre, que en este caso será `Person_Sensor`, como se muestra en la Figura 133 y la Figura 134.

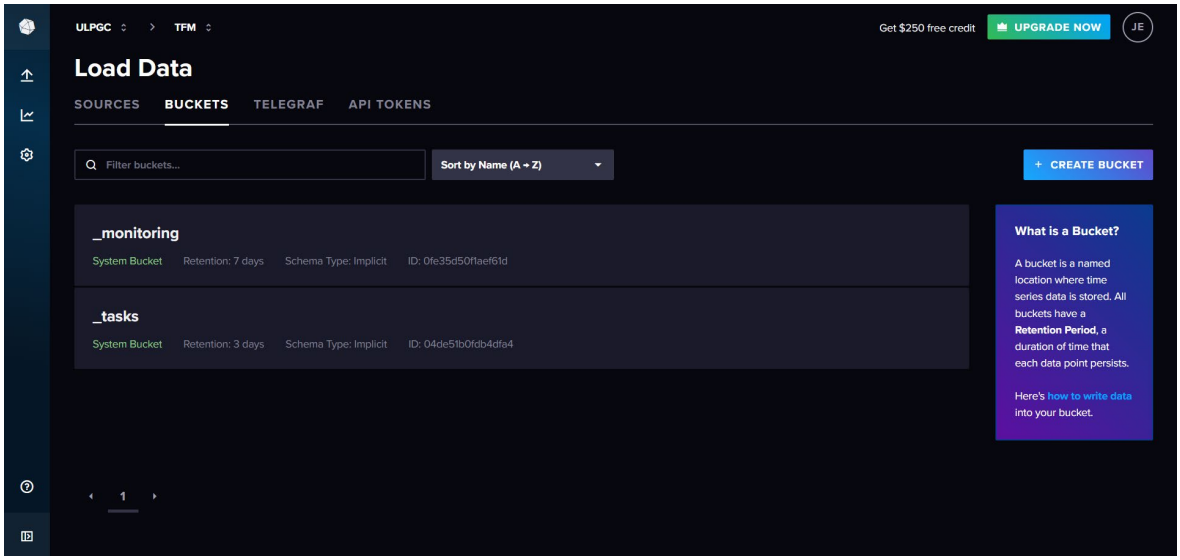


Figura 133 Panel de Buckets en InfluxdB.

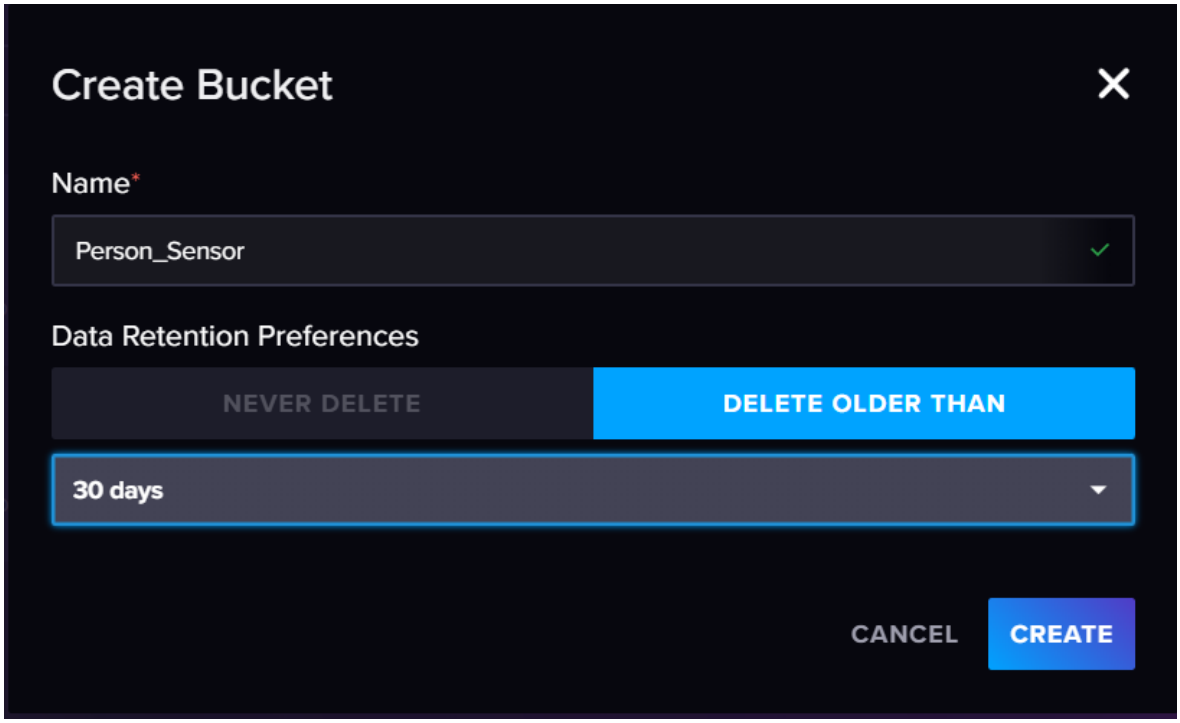


Figura 134 Creación de Buckets.

Una vez creado el *bucket*, como se muestra en la Figura 135, debe especificarse la fuente de los datos que se recibirán en el *bucket* `Person_Sensor` creado en InfluxDB Cloud 2.0. Este paso es crucial para garantizar que los datos provenientes del dispositivo Arduino MKRWAN 1310 se integren y almacenen correctamente en el *bucket*. Al definir la fuente de los datos, se facilita la gestión y el análisis de las series temporales, optimizando la eficacia de la aplicación IoT. Para ello, se proporciona un conjunto de bibliotecas que permiten la integración directa con diferentes tipos de clientes, así como numerosos complementos que permiten recopilar datos de diversos servicios y plataformas, en su mayor parte mediante el uso del agente *Telegraf*.

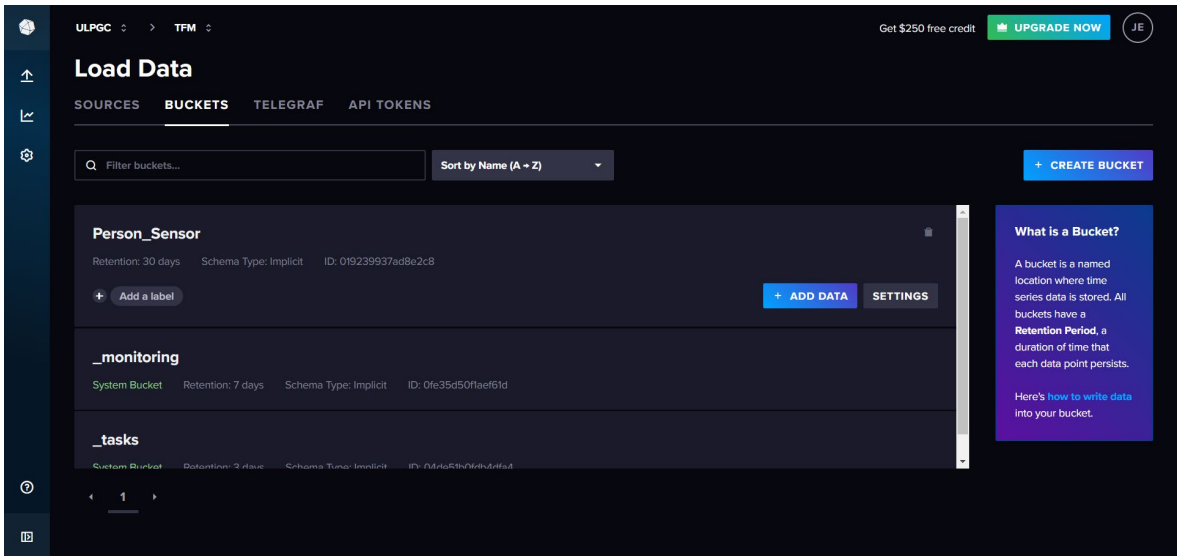


Figura 135 Visualización del Bucket creado Person_Sensor.

En concreto, para poder interactuar con el *broker* MQTT de la plataforma *The Things Network*, se procederá a la instalación del *plugin* MQTT en InfluxDB Cloud. Este *plugin* permite establecer una conexión directa y eficiente para la entrada de datos generados por dispositivos IoT. Al configurar el *plugin* con los parámetros adecuados, como la dirección del *Broker* MQTT de TTN y el nombre del *bucket* en el que se almacenarán los datos, se facilita la integración de la infraestructura de IoT con la plataforma de gestión y análisis de datos de InfluxDB Cloud. En este caso, se procede a especificar una fuente de datos a través del *plugin* MQTT, como se observa en la Figura 136 y la Figura 137.

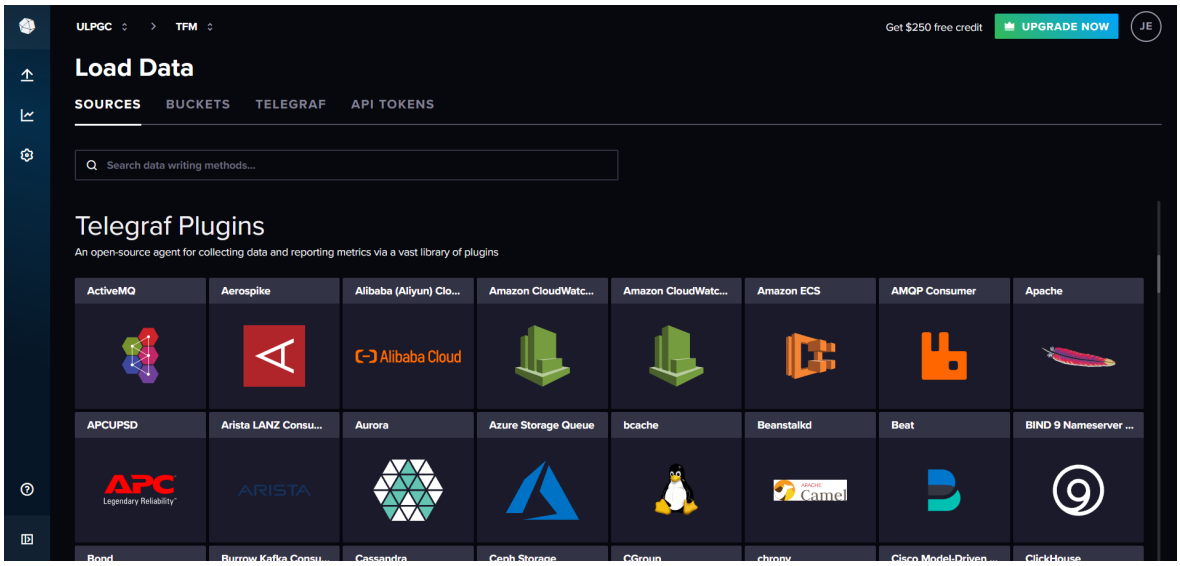


Figura 136 Telegraf Plugin.

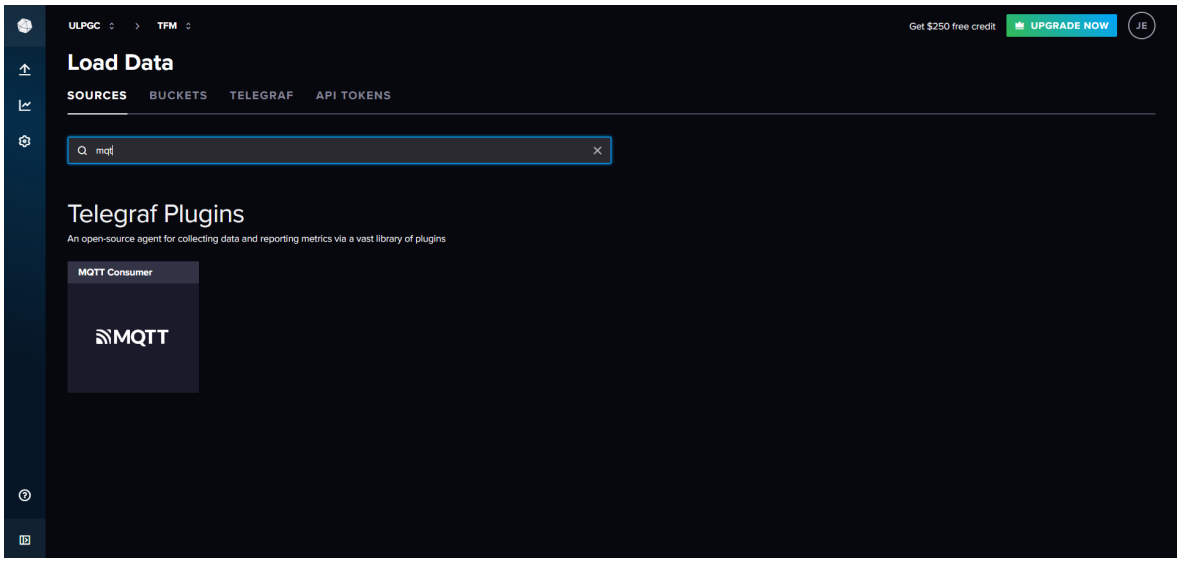


Figura 137 MQTT Plugin

Una vez seleccionado el *plugin*, se da nombre al agente *Telegraf* y se selecciona el *bucket* de salida, correspondiente en este caso a `Person_Sensor`, como se observa en la Figura 138.

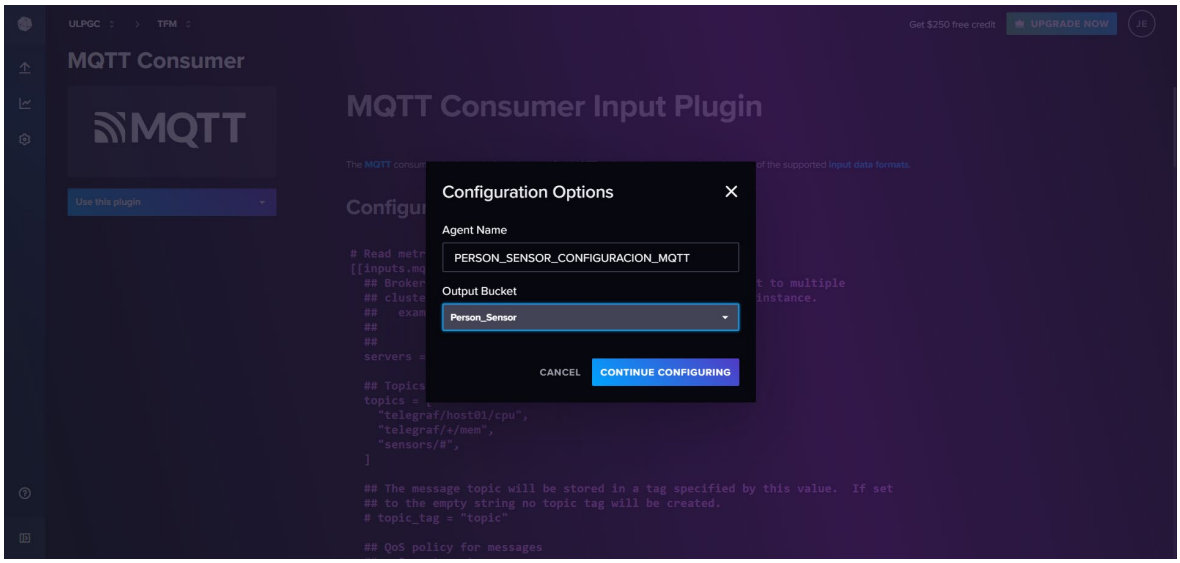


Figura 138 Configuración del Plugin MQTT.

Tras haber instalado el *plugin* MQTT en InfluxDB Cloud, se procede a revisar y configurar el archivo de configuración *Telegraf*, como se observa en la Figura 139. Este archivo de configuración debe ajustarse a las necesidades específicas de cada aplicación, estableciendo los servidores de entrada y de salida de datos, así como las credenciales necesarias para los procesos de autenticación. Esta configuración asegura que *Telegraf* pueda transmitir correctamente los datos desde los dispositivos IoT a InfluxDB Cloud, permitiendo una integración fluida y eficiente entre los componentes del sistema.

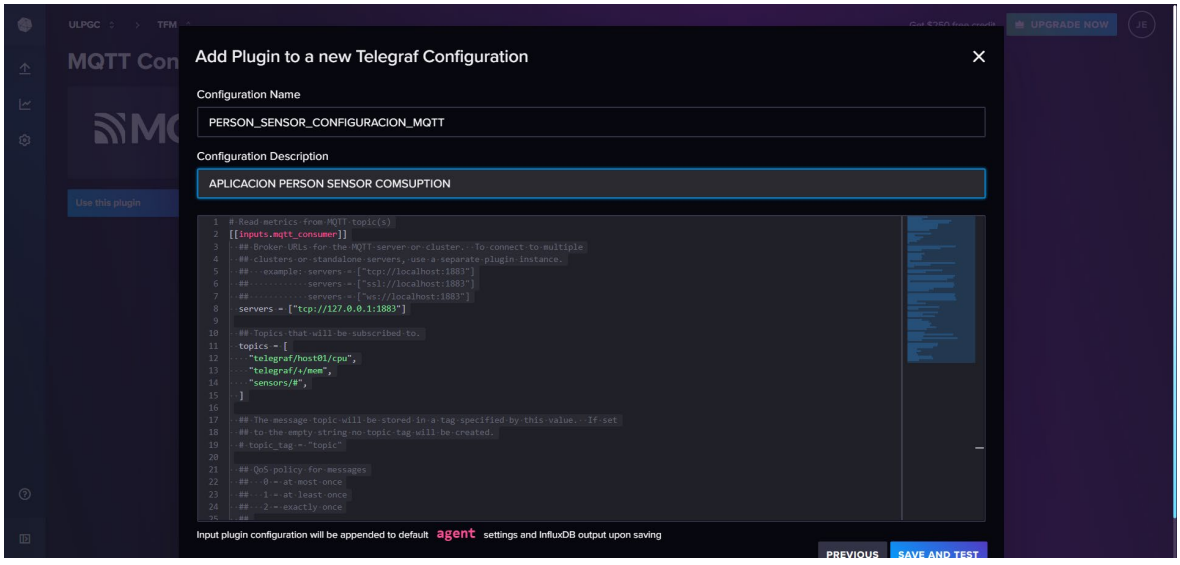


Figura 139 Archivo de configuración de Telegraf.

Con esta finalidad, es necesario acceder a la plataforma TTN para generar las claves de autenticación y definir los permisos requeridos para la recopilación de datos de la aplicación. En el caso concreto del presente TFM se deben generar las credenciales necesarias para la integración con el servidor MQTT de la plataforma TTN, como se muestra en la Figura 140, incorporándolas al archivo de configuración de *Telegraf*, junto con la dirección del servidor.

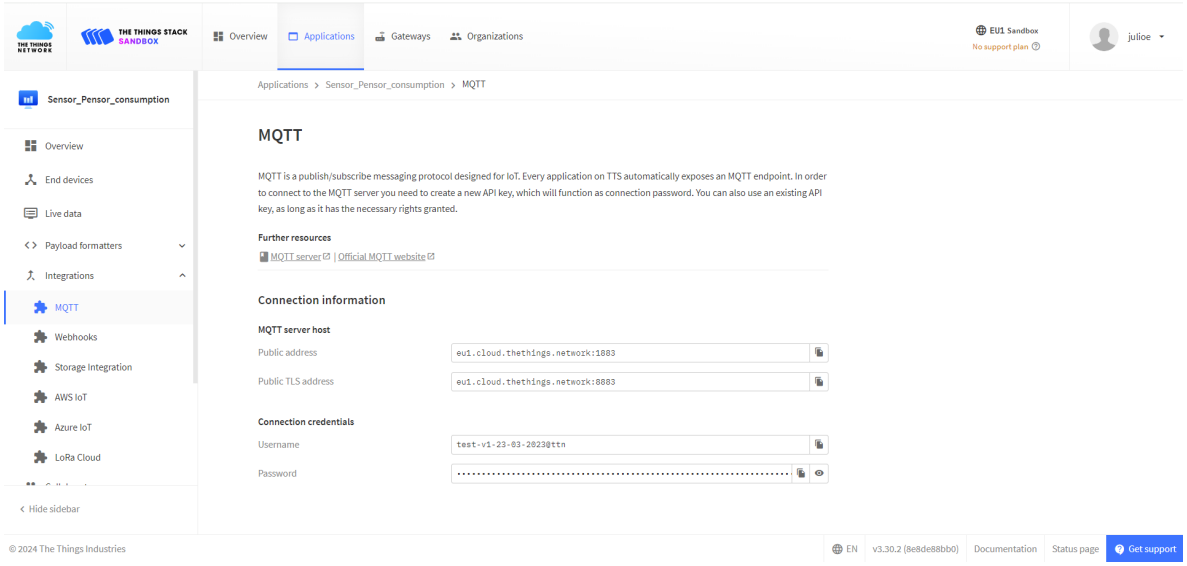


Figura 140 Generación de credenciales en TTN para la conexión basada en el protocolo MQTT.

Una vez que se ha configurado correctamente el archivo de configuración, el siguiente paso consiste en descargar *Telegraf* localmente, siguiendo los pasos mostrados en la Figura 141. *Telegraf* actúa como intermediario entre TTN e InfluxDB Cloud, facilitando la transmisión de datos. Es esencial copiar el parámetro `INFLUX TOKEN`, ya que será necesario para la configuración de *Telegraf*. Además, es importante crear una variable de entorno con este `TOKEN` para asegurar que *Telegraf* tenga los permisos adecuados para interactuar con InfluxDB Cloud.

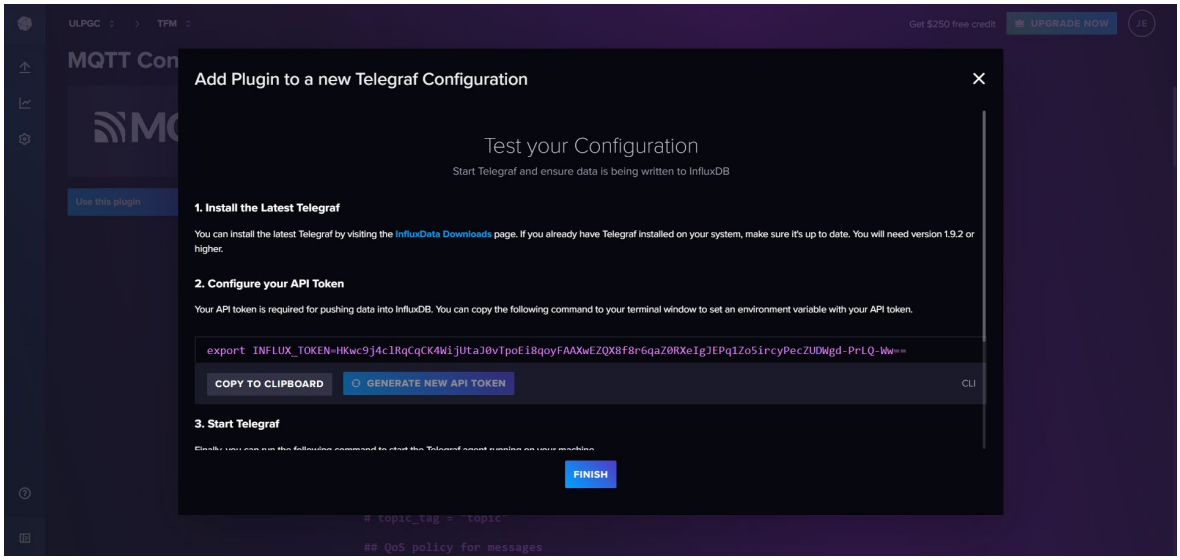


Figura 141 Pasos para la instalación y configuración de Telegraf de forma local.

A partir del archivo de configuración generado en InfluxDB Cloud, se podrá inicializar el agente *Telegraf* en un servidor local, como se representa en la Figura 142. Este agente tiene la tarea de enviar los datos recibidos desde TTN a través del *Broker* MQTT de la plataforma TTN ubicado en `eu1.cloud.thethings.network:1883`, utilizando el protocolo MQTT. Los datos son transmitidos en este caso al *bucket* `Person_Sensor`, previamente creado en InfluxDB Cloud.

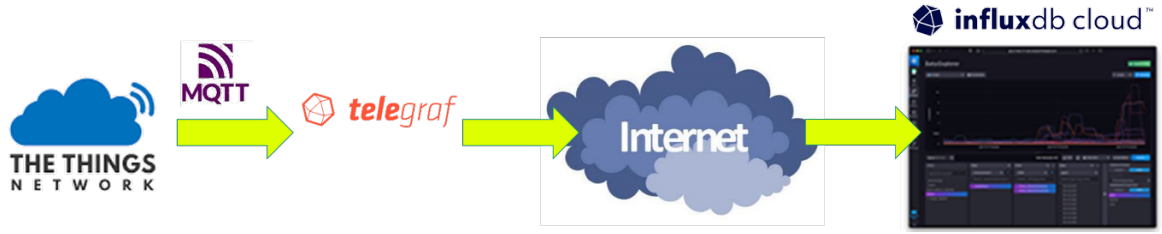


Figura 142 Agente local Telegraf para la integración de TTN e InfluxDB Cloud mediante MQTT.

Una vez creada la variable de entorno asociada parámetro `INFLUX TOKEN`, y tras seguir los pasos indicados en la Figura 141, en la Figura 143 se puede comprobar que inicialmente no se pudo establecer la conexión con InfluxDB Cloud.

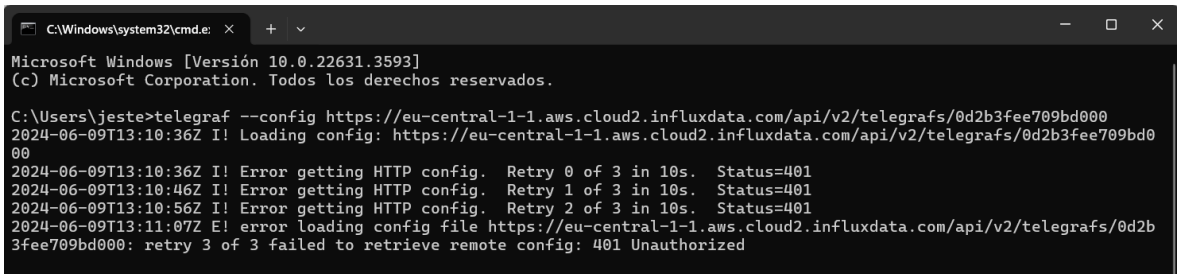


Figura 143 Proceso de inicio de Telegraf cargando el archivo de configuración ubicado en InfluxDB.

Tras intentar iniciar el agente *Telegraf* cargando el archivo de configuración de forma remota, fue necesario descargar el archivo y ejecutarlo de forma local. Para ello, se abre la consola de Windows como administrador, y se ejecuta el siguiente comando:

```
telegraf –debug –config
C:\Users\jeste\OneDrive\Escritorio\name_this_configuration.conf
```

En la Figura 144 se observa cómo, en este caso, sí se pudo iniciar correctamente el agente *Telegraf* a partir del estado de la conexión de entrada y la conexión de salida.

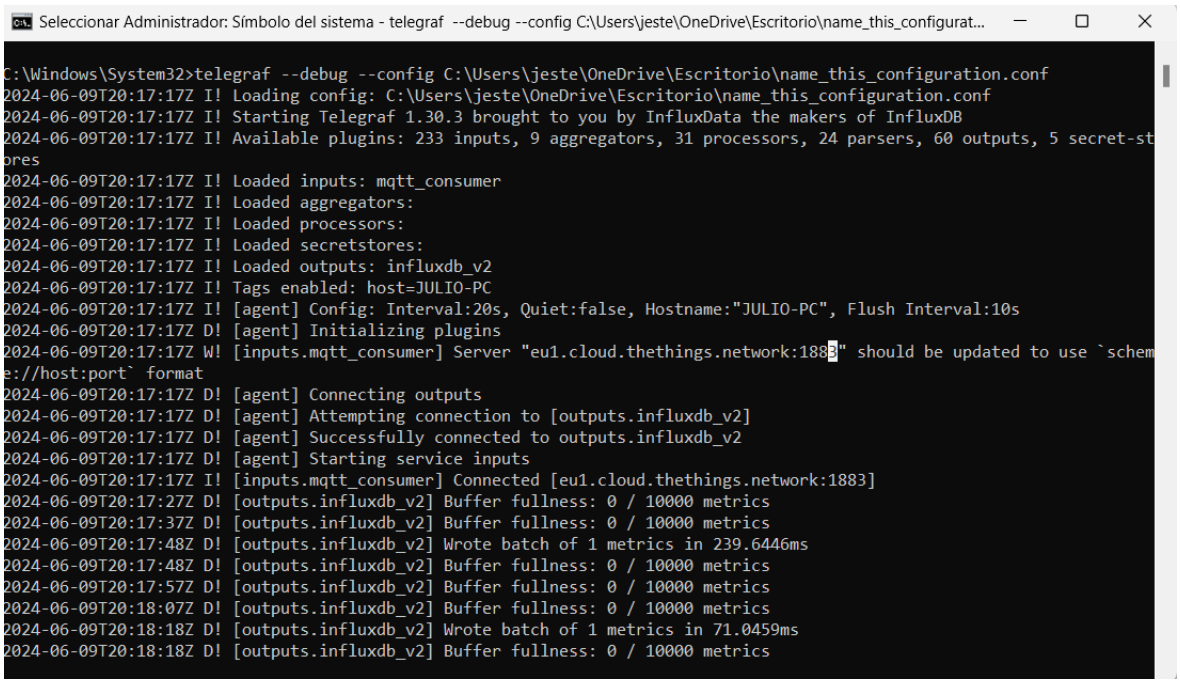


Figura 144 Proceso de inicio de Telegraf cargando el archivo de configuración ubicado en el ordenador.

6.4 Protocolo MQTT

MQTT (*Message Queuing Telemetry Transport*) es un protocolo ligero de mensajería basado en publicación/suscripción, diseñado para telemetría *Machine to Machine* (M2M) en entornos de bajo ancho de banda. Fue diseñado por Andy Stanford-Clark (IBM) y Arlen Nipper en 1999 para conectar sistemas de telemetría de oleoductos a través de satélite. Aunque comenzó siendo un protocolo propietario, se lanzó sin derechos de autor en 2010 y se convirtió en un estándar de OASIS en 2014. MQTT se está convirtiendo rápidamente en uno de los principales protocolos para implementaciones de aplicaciones IoT [48].

El protocolo MQTT incluye un conjunto de reglas que define cómo los dispositivos IoT pueden publicar y suscribirse a datos a través de Internet. MQTT se utiliza para mensajería e intercambio de datos entre dispositivos IoT, como sensores, PLC, etc. El protocolo está controlado por eventos y conecta dispositivos mediante un patrón *Publisher*/*Subscriber* (*Pub/Sub*). El remitente (*Publisher*) y el receptor (*Subscriber*) se comunican a través de las especificaciones de *Topics* o Temas, y están desacoplados entre sí. La conexión entre ellos es gestionada por un *Broker* MQTT. El *Broker* MQTT filtra todos los mensajes entrantes y los distribuye correctamente a los suscriptores [48].

A diferencia del paradigma Cliente/Servidor del protocolo HTTP, MQTT opera a partir de eventos, lo que permite enviar mensajes a los Clientes. Este enfoque resulta altamente escalable al desacoplar a los productores de los consumidores de datos, eliminando las dependencias entre ellos. Así, los componentes claves para establecer una conexión MQTT para la publicación y suscripción de mensajes son los Clientes MQTT y el *Broker* MQTT, representados en la Figura 145 [49].

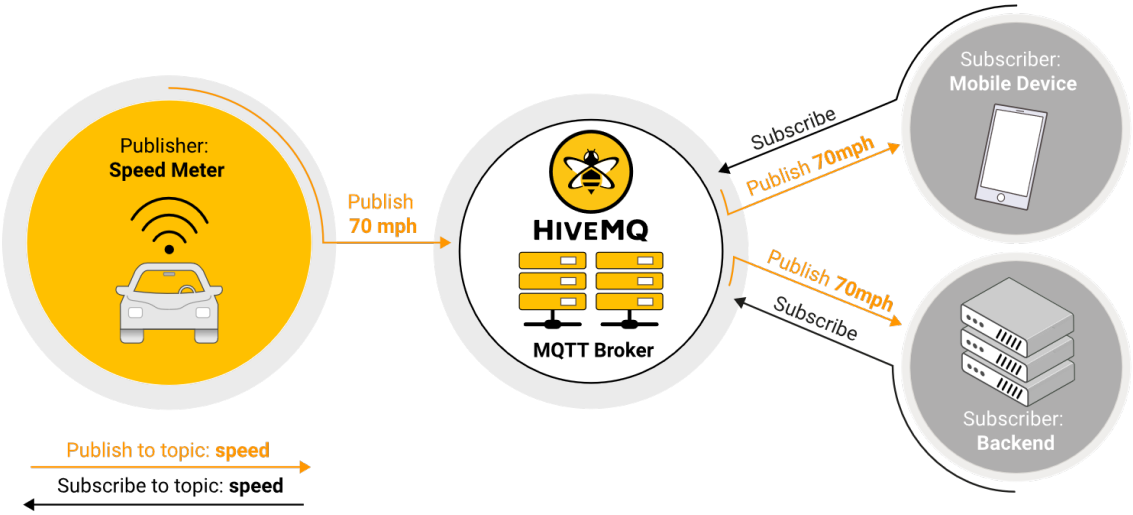


Figura 145 Arquitectura de Publicación/suscripción de MQTT [49].

El protocolo MQTT ha encontrado aplicaciones significativas en múltiples sectores, siendo utilizado por empresas de la industria automotriz, telecomunicaciones, energía, seguridad pública y conectividad de productos, entre otras. Este protocolo se ha convertido en una herramienta esencial para la comunicación eficiente y confiable en una amplia gama de aplicaciones y sectores industriales.

6.4.1 Beneficios del protocolo MQTT

Los beneficios inherentes al protocolo MQTT son múltiples y se distinguen por su relevancia en diversos contextos [49]:

- Eficiencia y ligereza: MQTT destaca por su eficiencia y capacidad para minimizar los recursos demandados por los clientes, así como el ancho de banda utilizado.

- Comunicación bidireccional: Facilita la comunicación versátil entre dispositivos y servidores, posibilitando tanto la publicación como la suscripción a eventos. Además, permite la transmisión de mensajes a grupos de dispositivos, optimizando la interacción.
- Escalabilidad: MQTT exhibe una notable escalabilidad, siendo apto para respaldar la operación de millones de dispositivos dentro de ecosistemas relacionados con IoT o IIoT.
- Niveles de Calidad de Servicio (QoS): Proporciona distintos niveles de QoS, garantizando una entrega confiable de los mensajes, lo que resulta esencial en aplicaciones críticas.
- Sesiones persistentes: MQTT admite sesiones persistentes entre dispositivos y servidores, reduciendo los tiempos de reconexión en entornos de red poco confiables, lo cual es de gran utilidad para mantener una comunicación ininterrumpida.
- Funciones de seguridad avanzadas: Ofrece funciones de seguridad robustas, incluyendo cifrado TLS para asegurar la confidencialidad de los mensajes y protocolos de autenticación para verificar la identidad de los clientes.
- Amplio rango de aplicaciones: MQTT se aplica en una amplia variedad de sectores y casos de uso, lo que evidencia su versatilidad y aplicabilidad en contextos del mundo real.

En el núcleo de MQTT se encuentran los *Brokers* y los clientes MQTT. El *Broker* MQTT opera como un intermediario que facilita la comunicación entre los emisores y receptores, asegurando que los mensajes lleguen a los destinatarios apropiados. Por su parte, los clientes MQTT tienen la capacidad de publicar mensajes hacia el *Broker*, mientras que otros clientes pueden suscribirse a *Topics* o Temas específicos para recibir mensajes de su interés. Cada mensaje MQTT incorpora un *Topic*, y los clientes establecen suscripciones a Temas particulares de acuerdo con sus necesidades. El *Broker* MQTT mantiene un registro de suscriptores y lo utiliza para dirigir los mensajes hacia los clientes pertinentes [49].

Adicionalmente, MQTT posibilita que el *Broker* almacene mensajes para clientes que se encuentren desconectados, lo que garantiza la entrega confiable de mensajes, incluso en

condiciones de red poco fiables. Para lograr este propósito, MQTT implementa tres niveles de Calidad de Servicio (QoS): 0 (como máximo una vez), 1 (al menos una vez) y 2 (exactamente una vez) [49].

Es importante mencionar que en la actualidad existen dos versiones de la especificación MQTT, denominadas MQTT 3.1.1 y MQTT 5. Aunque la mayoría de los *Brokers* MQTT comerciales actualmente admiten MQTT 5, ciertos servicios gestionados en la nube relacionados con IoT continúan soportando mayoritariamente MQTT 3.1.1 [49].

6.4.2 Cliente MQTT

Dentro del contexto de IoT, un cliente MQTT generalmente se divide en dos categorías principales: *publishers* y *subscribers*. Un *publisher* es aquel cliente encargado de enviar mensajes, mientras que un *subscriber* es el cliente que recibe estos mensajes. Sin embargo, es importante destacar que un Cliente MQTT tiene la capacidad de desempeñar el rol de *publishe*r, *subscrib*er, o ambas funciones simultáneamente [50].

Un Cliente MQTT, en términos generales, puede abarcar una amplia variedad de dispositivos, siempre y cuando ejecuten una biblioteca MQTT y se conecten a un *Broker* MQTT a través de una red [50]. MQTT está diseñado específicamente para operar sobre el protocolo TCP/IP, lo que significa que cualquier dispositivo capaz de establecer comunicación mediante TCP/IP, y que cuente con una implementación del protocolo MQTT, puede considerarse un Cliente MQTT. La implementación del Cliente MQTT se caracteriza por su simplicidad y eficiencia, lo que lo convierte en una opción idónea para dispositivos de recursos limitados [50].

6.4.3 Broker MQTT

Un *Broker* MQTT representa el núcleo central en el sistema de mensajería de publicación y suscripción, recibiendo mensajes de los *publishers* y los distribuyéndolos a los *subscribers*. Desempeña un papel crítico en la gestión del flujo de comunicación entre los Clientes MQTT y garantiza una entrega de mensajes fiable [50]. Las funcionalidades de un *Broker* MQTT incluyen las siguientes:

1. Gestión de un gran número de conexiones concurrentes: Dependiendo de la implementación, un *Broker* MQTT puede gestionar millones de clientes conectados simultáneamente. Esto permite la comunicación entre diferentes dispositivos, redes y sistemas de software al actuar como puente entre ellos [50].
2. Filtrado y encaminamiento de mensajes: El *Broker* MQTT filtra los mensajes en función del *Topic* de suscripción y determina qué cliente o clientes deben recibir cada mensaje [50].
3. Gestión de sesiones: El *Broker* MQTT mantiene los datos de sesión de todos los clientes conectados, incluyendo suscripciones y mensajes perdidos, para los clientes con sesiones persistentes [50].
4. Autenticación y Autorización: El *Broker* MQTT es responsable de autenticar y autorizar a los clientes basándose en las credenciales que proporcionen. Además de la autenticación y autorización, los *Brokers* MQTT pueden ofrecer otras características de seguridad, como la encriptación de mensajes y listas de control de acceso [50].
5. Escalabilidad, Integración y Monitorización: Un *Broker* MQTT debe ser escalable para ser capaz de gestionar grandes volúmenes de mensajes y clientes, fácil de monitorear, y resistente a fallos [50].

En el Anexo del presente documento se incluye una descripción detallada del proceso de comunicación entre los Clientes MQTT y un *Broker* MQTT.

6.5 Visualización de los Datos en InfluxDB Cloud

En la esta sección se procederá a la visualización de los datos almacenados en InfluxDB Cloud, tras su integración con la plataforma TTN haciendo uso del protocolo MQTT. Para ello, se utilizarán los recursos que proporciona InfluxDB Cloud con el fin de crear gráficos que representen los diferentes valores de interés asociados a la aplicación IoT desarrollada en este TFM con el fin de poder realizar el análisis de comportamientos a partir de la detección de rostros personas. Con este propósito, en primer lugar, se selecciona en InfluxDB Cloud el *bucket* `Person_Sensor` creado, y el *measurement*

correspondiente en cada caso, definiéndose el intervalo de tiempo deseado, y ejecutando finalmente la consulta para obtener los datos asociados.

Como referencia, en la Figura 146 se puede apreciar la visualización de los valores correspondientes al porcentaje del nivel de la batería del dispositivo Arduino MKRWAN 1310 en un período de tiempo determinado, mientras que en la Figura 147 se muestra la representación gráfica del número de rostros detectados. Con el fin de mejorar la comprensión de los datos visualizados, se pueden ajustar las configuraciones de visualización en el panel *Query Modifier* ubicado a la derecha de la interfaz de usuario.

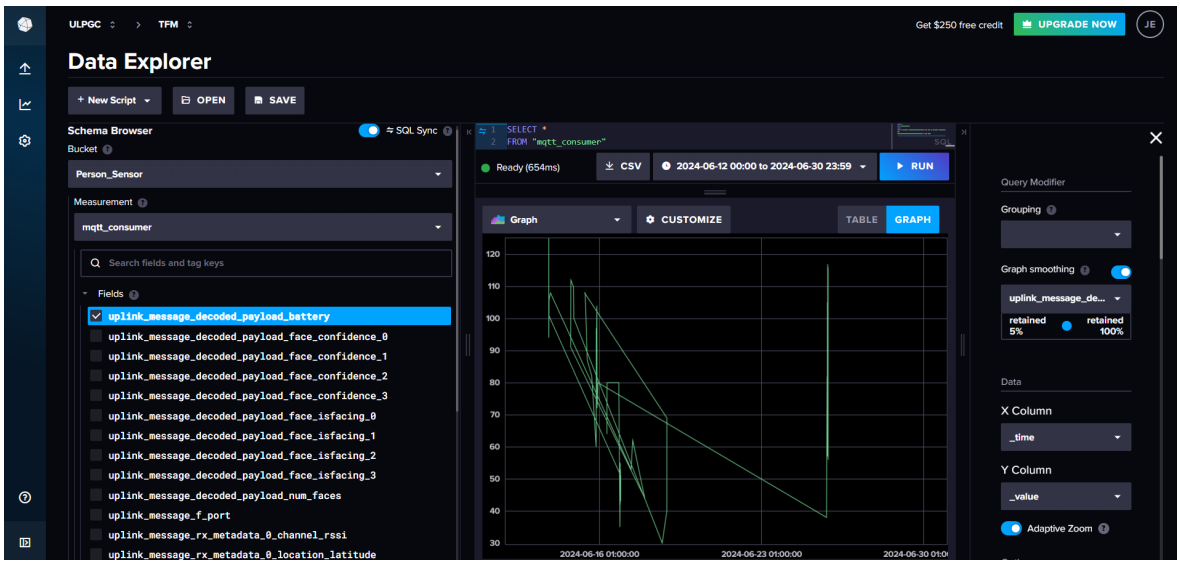


Figura 146 Nivel de la Batería.

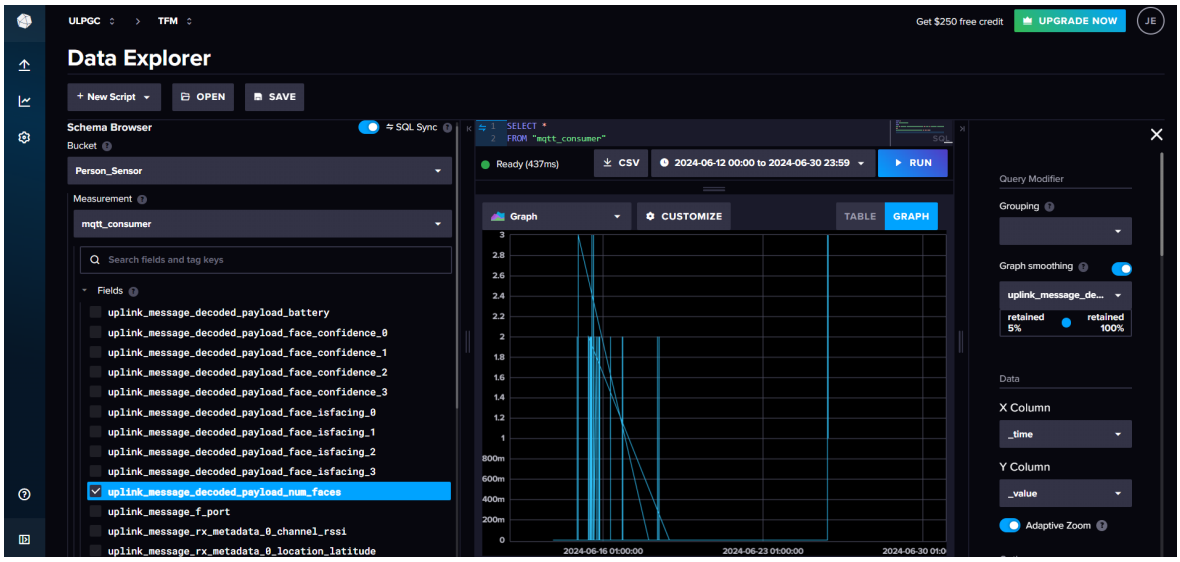


Figura 147 Número de Rostros.

En la Figura 148, la Figura 149, la Figura 150 y la Figura 151 se representa la vista del estado de las miradas de los rostros detectados por el módulo *Person Sensor* integrado en el nodo final de la plataforma IoT desarrollada.

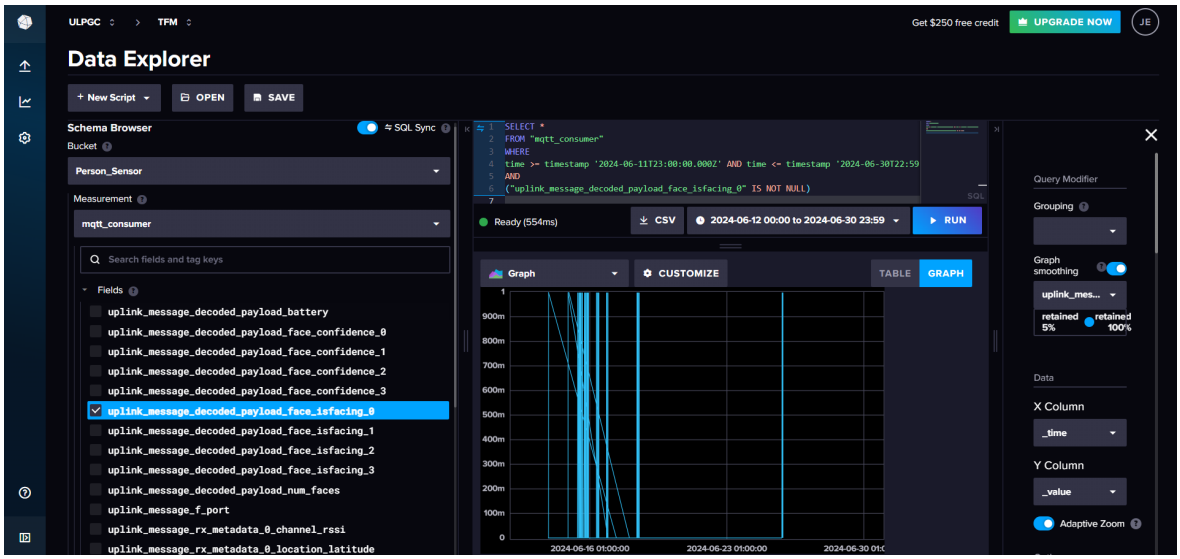


Figura 148 Estado de la mirada del rostro (1).

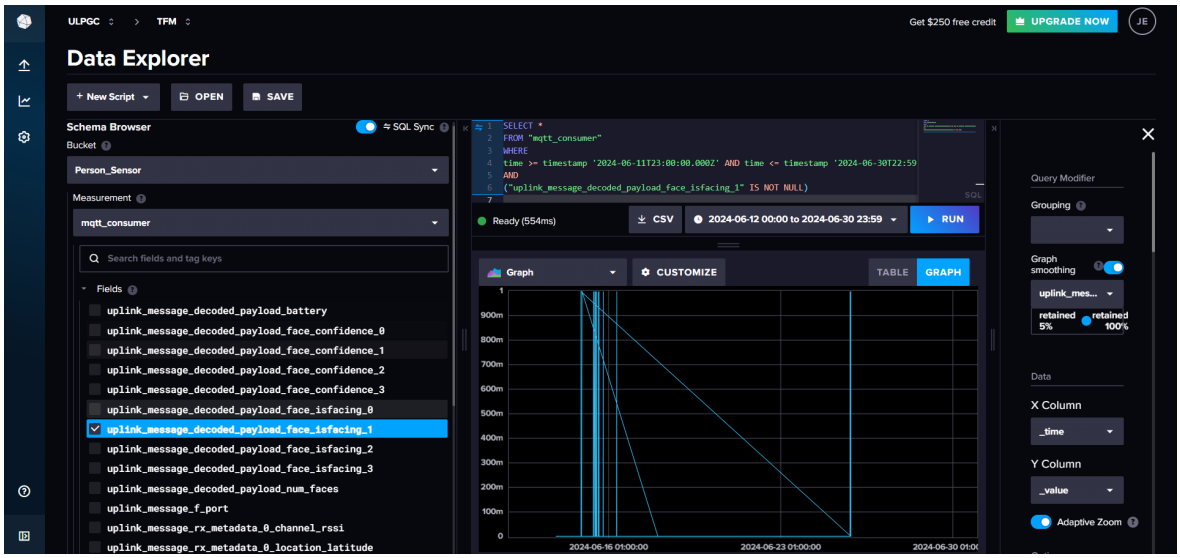


Figura 149 Estado de la mirada del rostro (2)

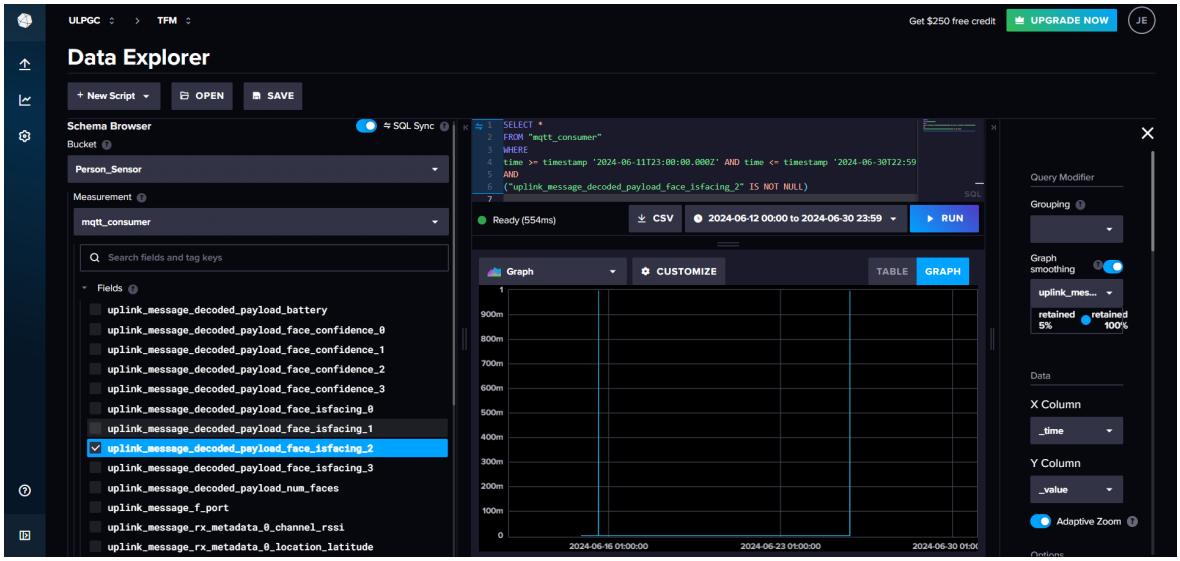


Figura 150 Estado de la mirada del rostro (3).

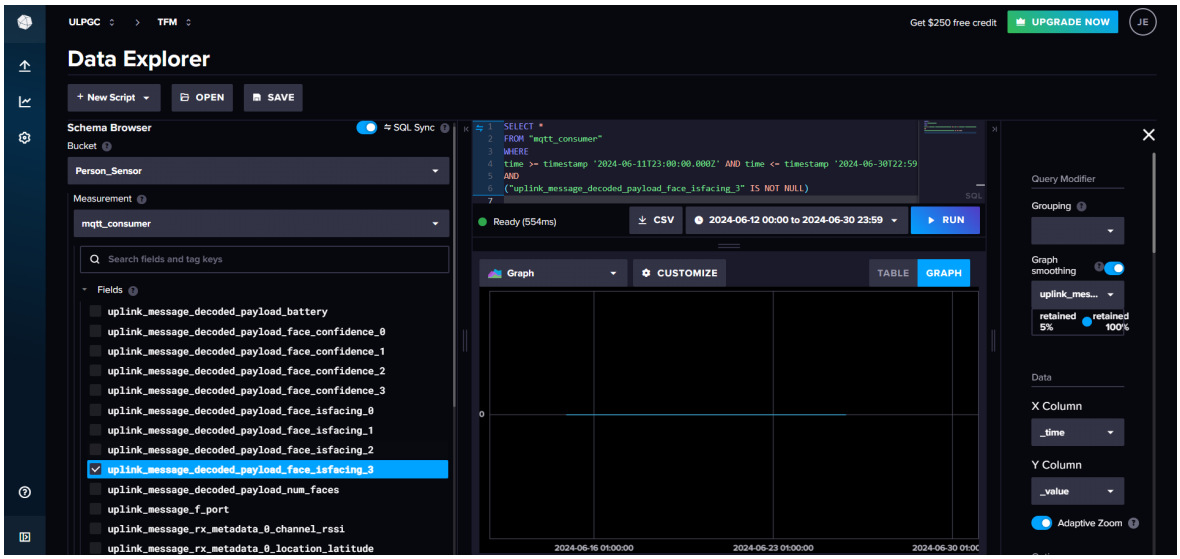


Figura 151 Estado de la mirada del rostro (4).

Finalmente, en la Figura 152, la Figura 153, la Figura 154 y la Figura 155 se muestra la visualización del porcentaje de confianza en la detección de los rostros.

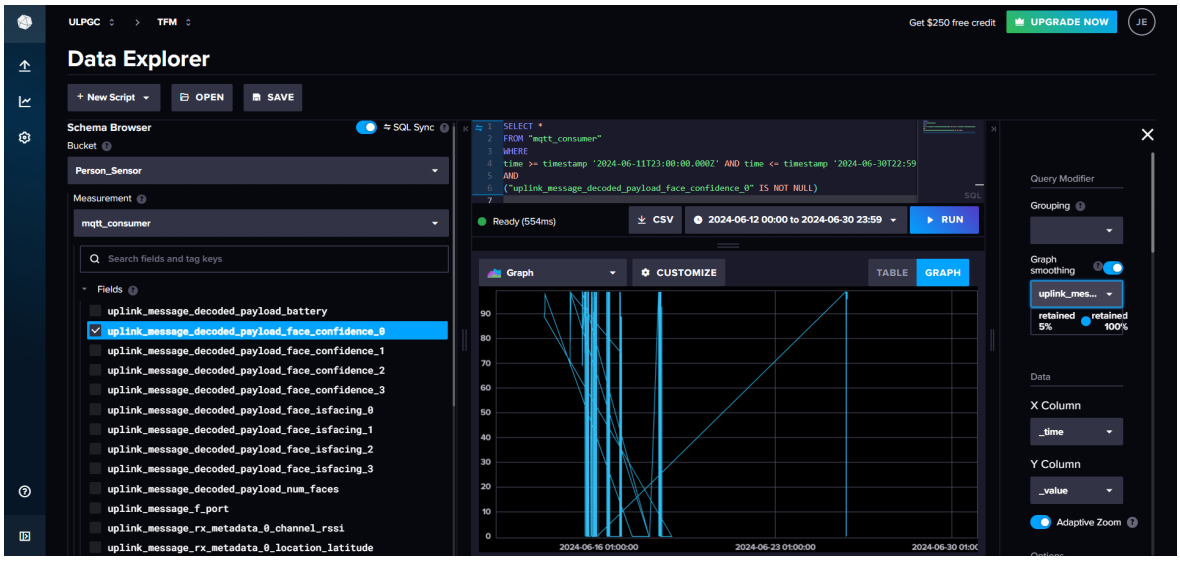


Figura 152 Porciento de Confidencialidad del rostro (1).

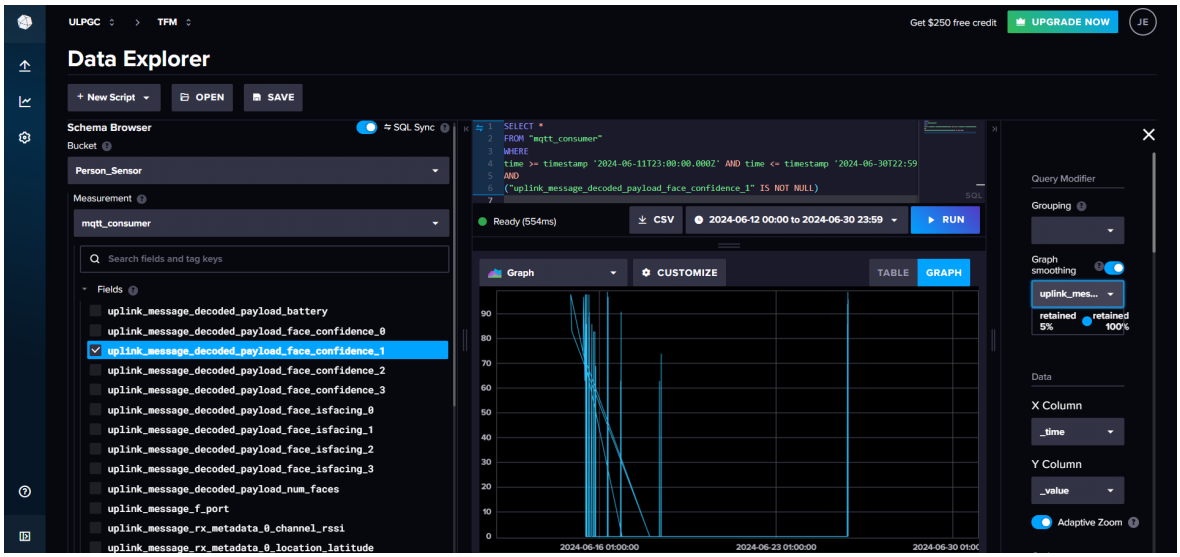


Figura 153 Porciento de Confidencialidad del rostro (2).

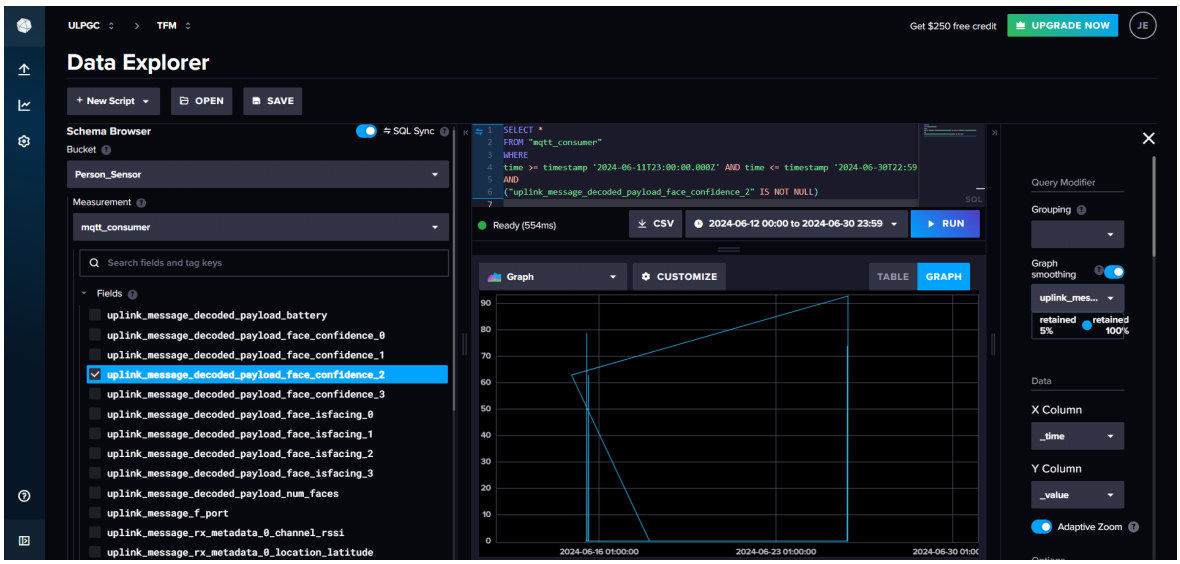


Figura 154 Porciento de Confidencialidad del rostro (3).

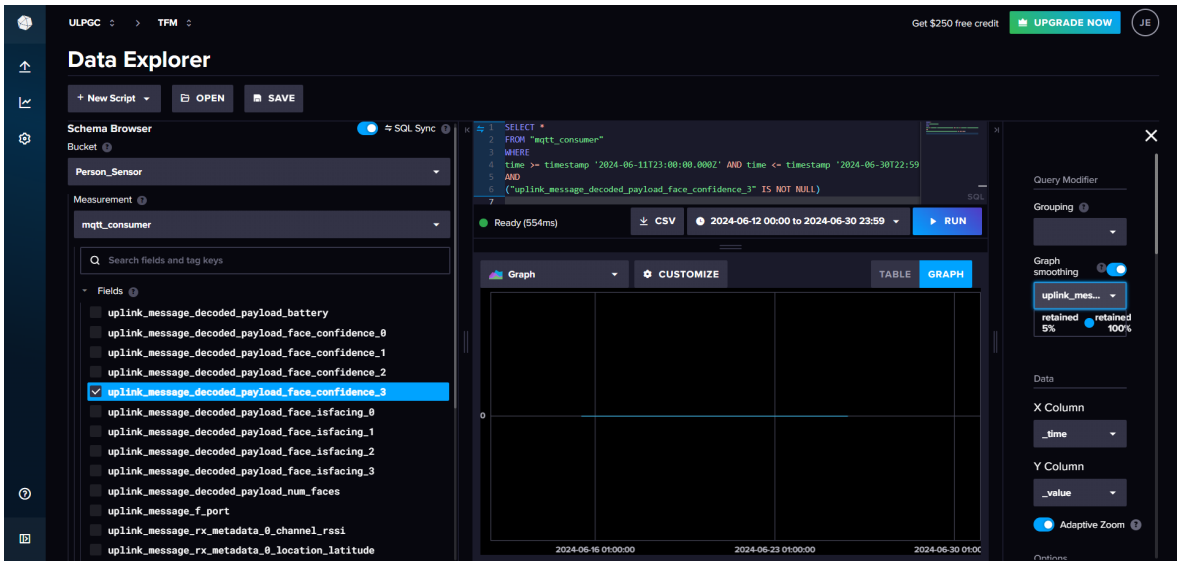


Figura 155 Porciento de Confidencialidad del rostro (4).

Las limitaciones en la visualización de gráficos directamente en InfluxDB Cloud se centran en su enfoque principalmente orientado al almacenamiento y la consulta de series temporales de datos, lo cual puede restringir la capacidad de personalización avanzada y la diversidad de tipos de gráficos disponibles. Para superar estas limitaciones y mejorar el aspecto visual de los gráficos de usuario, se decidió incorporar la herramienta Grafana a la aplicación IoT desarrollada en el presente TFM, como se presentará en el siguiente capítulo.

Grafana se integra con InfluxDB para proporcionar una interfaz más robusta y flexible en la creación de visualizaciones personalizadas, soportando una mayor variedad de gráficos, paneles interactivos y opciones de diseño. Esto no solo mejora la estética de los gráficos, sino que también facilita la visualización de datos complejos y la gestión de múltiples fuentes de datos en entornos IoT y de monitorización.

7 Visualización de datos con Grafana

Durante años, el lenguaje básico de la visualización de datos se ha basado en el uso de gráficos e histogramas. Sin embargo, cada vez era más necesario recrear estos gráficos, no en horas o días, sino en segundos o incluso milisegundos. Esto requiere de una potencia de procesamiento que permita representar miles de puntos de datos en el tiempo que tarda en actualizarse su visualización en la pantalla de un ordenador [51].

Sin embargo, varias tendencias han convergido para generar un renacimiento en la adquisición y visualización de datos, haciéndolo accesible, no solo para los profesionales del campo, sino también para personas técnicamente competentes del público en general. Estas tendencias incluyen, entre otras, la disponibilidad de CPU y GPU de propósito general económicas, el almacenamiento de alta capacidad, los estándares y tecnologías web como JavaScript y CSS, las herramientas software de código abierto, la computación en la nube escalable a precios asequibles, y las redes de banda ancha para empresas, hogares y dispositivos móviles [51].

7.1 Beneficios de la integración de Grafana

Aunque existen muchas soluciones en el ámbito de la visualización de datos, Grafana, cuyo logotipo se muestra en la Figura 156, destaca como una de las más relevantes en el ámbito de IoT. Esto se debe a su rápido crecimiento en alcance y características, amplias opciones para despliegue y soporte, y una creciente comunidad que contribuye a su futuro desarrollo [51].

Figura 156 Logotipo de Grafana

Grafana ofrece una plataforma flexible para la visualización de datos, capaz de integrarse con una amplia variedad de fuentes, incluyendo bases de datos de series temporales como InfluxDB, *Prometheus*, y otras. Su capacidad para crear *dashboards* interactivos y personalizados facilita la interpretación de grandes volúmenes de datos en tiempo real, lo que es crucial para una toma de decisiones informada y eficiente [51].

La combinación de estas capacidades con su naturaleza de código abierto, su facilidad de uso y su escalabilidad, hacen de Grafana una herramienta indispensable para profesionales de diferentes industrias que necesitan una solución robusta para el monitoreo y análisis de datos [51].

Aunque existen bastantes herramientas de análisis de datos que cumplen estas funciones, Grafana tiene una serie de características que hacen que resulte una opción atractiva. El *backend* de Grafana está escrito en lenguaje Go, desarrollado por Google, lo que lo hace extremadamente eficiente para la consulta de fuentes de datos [51].

Admite un modelo de *plugins* para sus paneles de control y fuentes de datos. El número de complementos crece constantemente a medida que la comunidad de Grafana contribuye al proyecto. Esta flexibilidad permite una personalización y expansión continuas de sus capacidades [51].

Por otro lado, Grafana utiliza la biblioteca D3. Muchas de las herramientas populares de paneles de control, como Datadog y Zabbix, pueden generar rápidamente gráficos visualmente atractivos a partir de miles de canales de datos, pero solo ofrecen un control limitado sobre los elementos de visualización. Sin embargo, Grafana proporciona un control detallado sobre la mayoría de los elementos gráficos, incluidos ejes, líneas, puntos, anotaciones o leyendas [51].

7.2 Instalación

Para la instalación local de Grafana es necesario dirigirse a la página de descargas https://grafana.com/grafana/download?platform=windows y seleccionar la versión que se desea instalar, como se observa en la Figura 157. Por defecto se selecciona la versión más

reciente, mostrándose únicamente versiones etiquetadas en el campo de versión. En el caso particular del presente TFM se puede seleccionar la edición *Enterprise*, que es la recomendada por las funciones adicionales disponibles con licencia, u *Open Source*, funcionalmente idéntica a la edición *Enterprise*, pero sin características licenciadas [52].

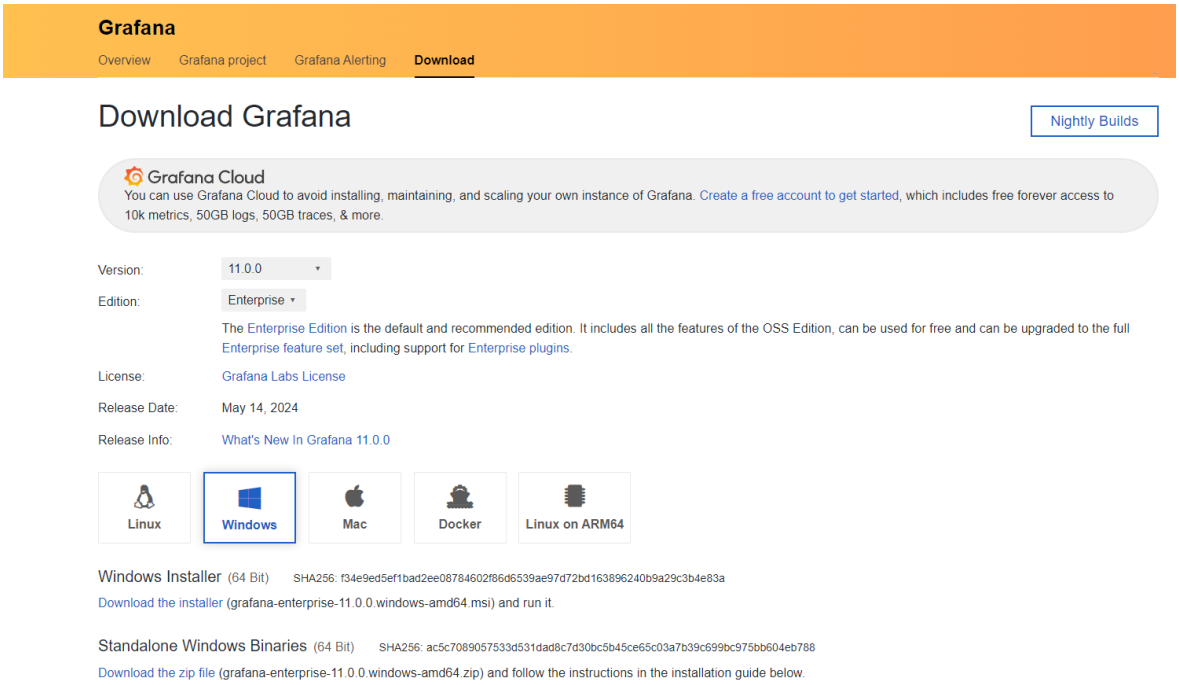


Figura 157 Página de descarga de Grafana [53].

Para la versión binaria independiente de Windows, se descarga el archivo zip, se descomprime y se instala como cualquier programa de Windows, accediendo a Grafana desde el navegador en el puerto predeterminado (por lo general `http://localhost:3000/`), como se muestra en la Figura 158 [52].

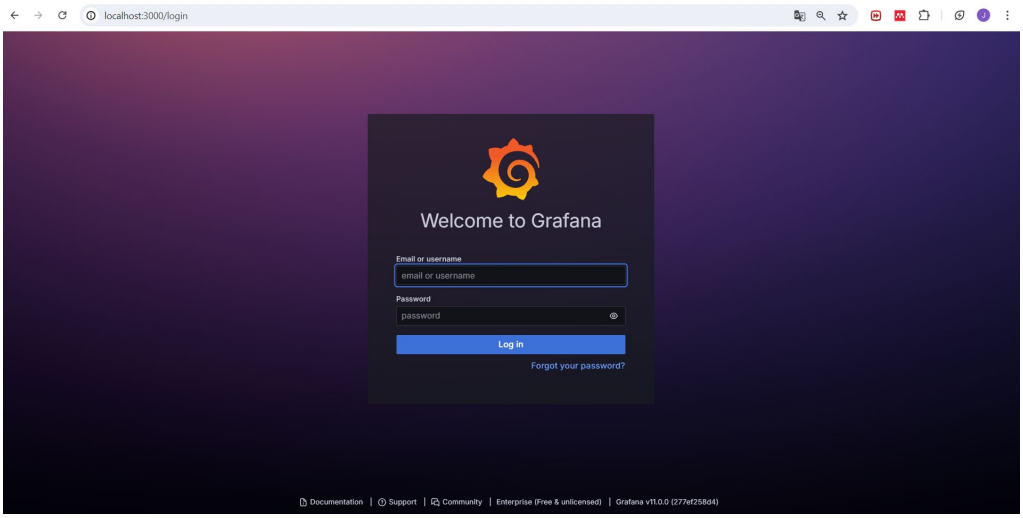


Figura 158 Página de inicio de Grafana.

Si el puerto `3000` requiere permisos adicionales en Windows, o no está disponible, se modifica el puerto copiando el fichero `sample.ini` a `custom.ini` en el directorio `conf`, se edita el fichero `custom.ini` para descomentar la opción `http_port`, y se establece un puerto de preferencia, como por ejemplo `8080`, que generalmente no requiere de privilegios adicionales [52]. Una vez dentro, se observa la interfaz principal de Grafana, como se muestra en la Figura 159.

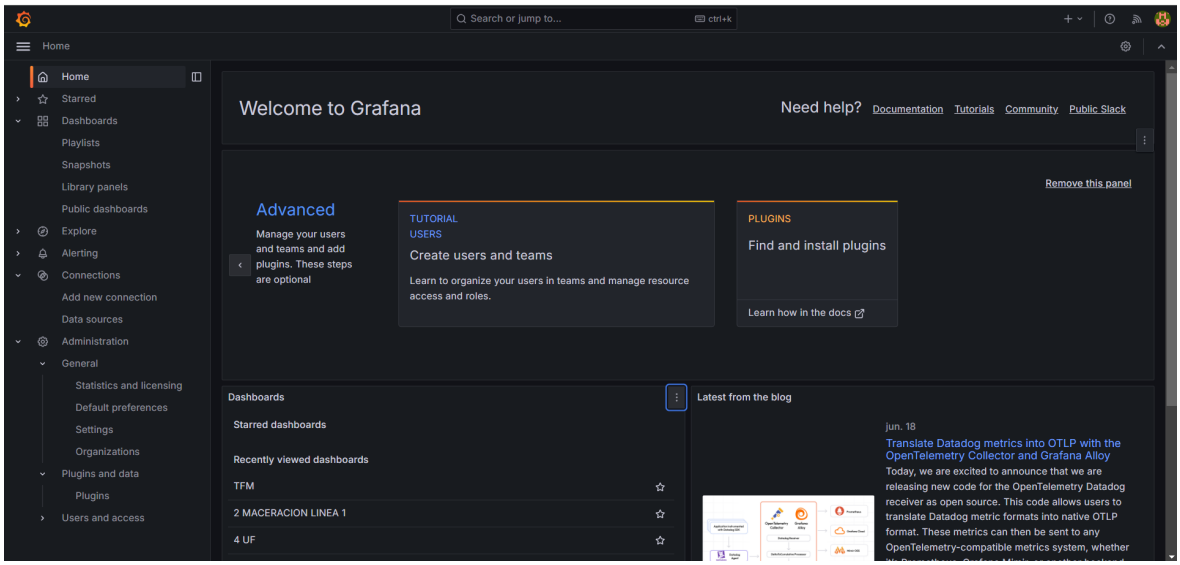


Figura 159 Interfaz Principal de Grafana.

7.3 Integración de Grafana con InfluxDB Cloud

Para establecer una conexión entre Grafana e InfluxDB Cloud, es necesario preparar en primer lugar InfluxDB Cloud, y configurar posteriormente Grafana. Así, para lograr la integración de Grafana con InfluxDB Cloud es necesario generar un `API TOKEN` en este último, como se observa en la Figura 160, seleccionando la opción GENERATE API TOKEN, y el campo *All Access API Token*, generándose con ello un `API TOKEN` con todos los permisos necesarios, como se observa en la Figura 161 y la Figura 162.

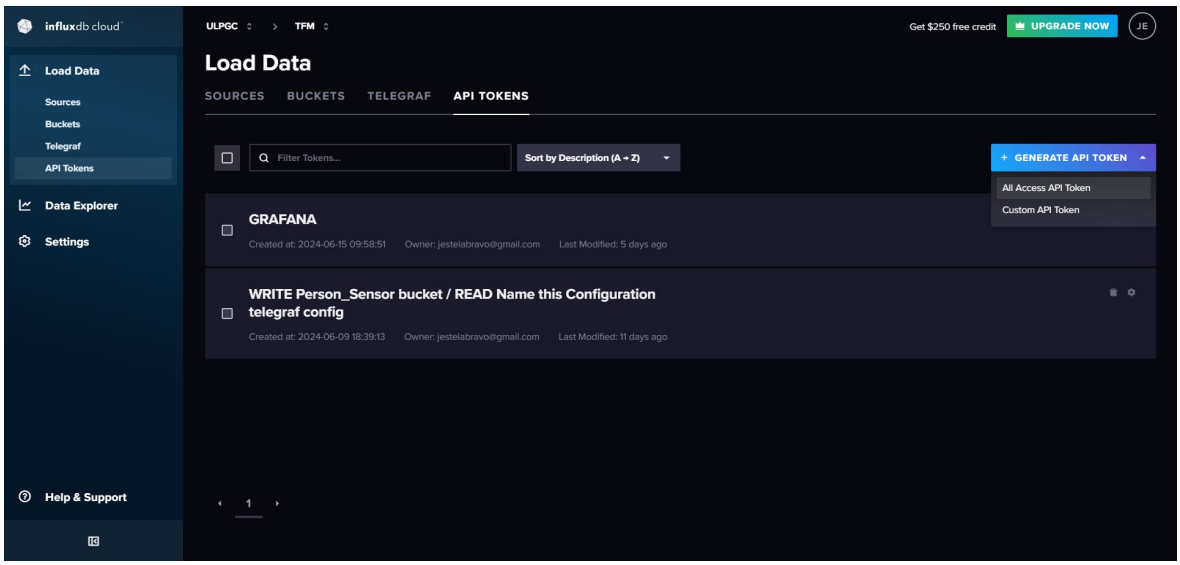


Figura 160 Creación de API Token en InfluxDB.

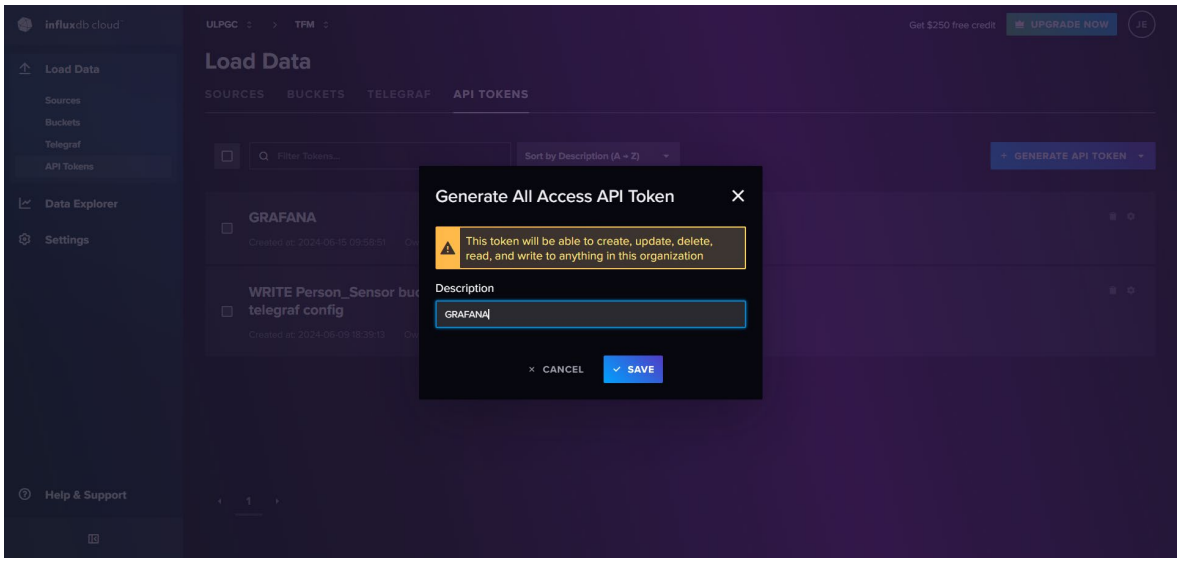


Figura 161 Nombrando el API Token.

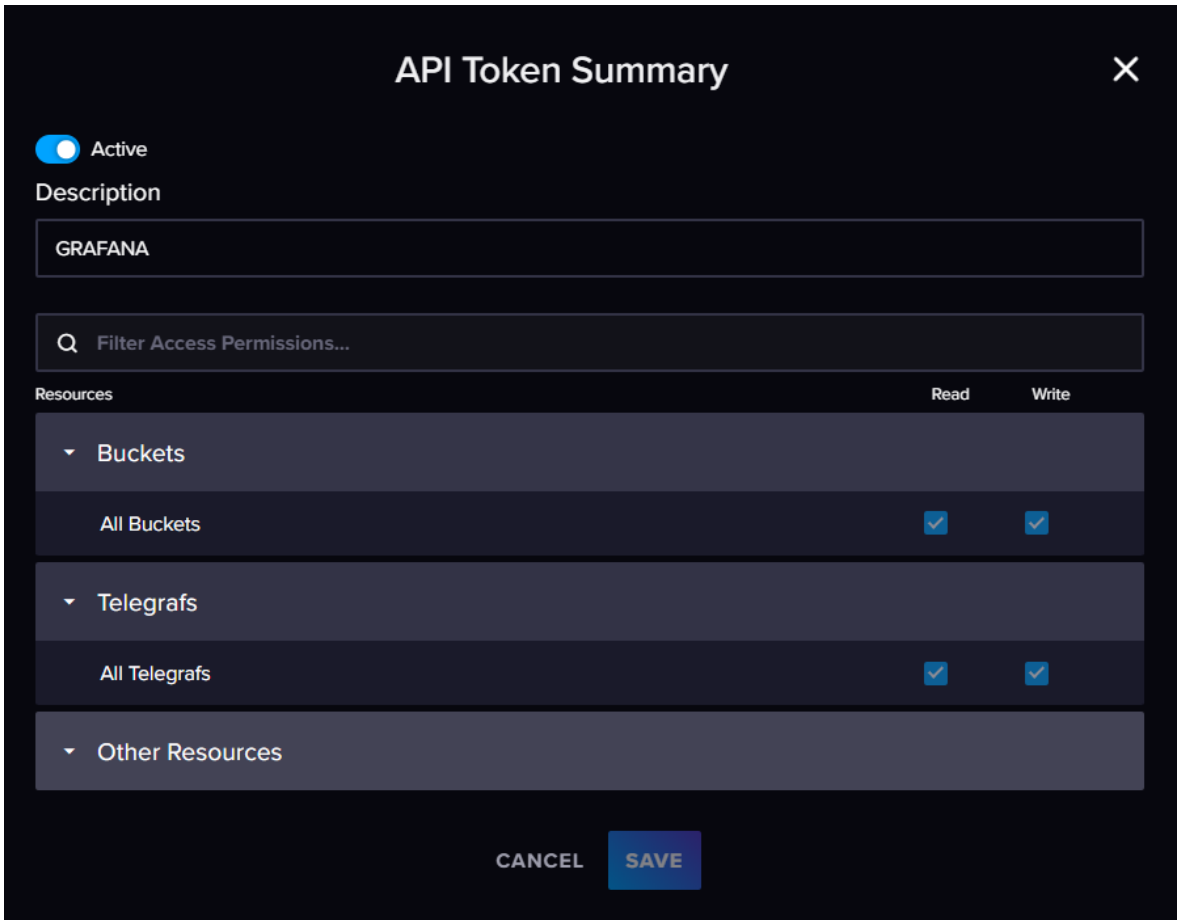


Figura 162 Permisos Generados en la API TOKEN.

A continuación, se debe copiar el `API TOKEN` generado para su posterior uso en Grafana, como se muestra en la Figura 163.

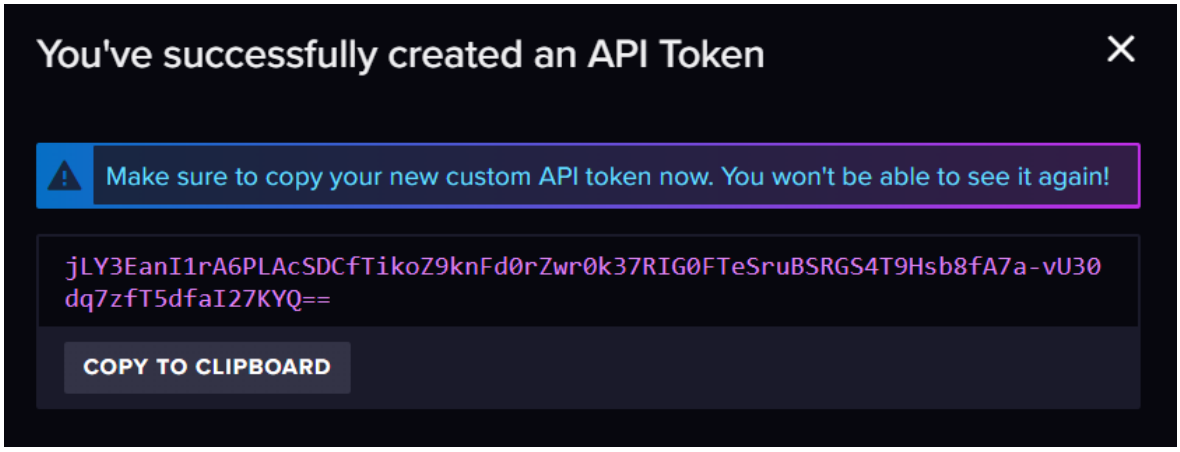


Figura 163 API Token.

Posteriormente se retorna a Grafana y se añade una nueva conexión, como se muestra en la Figura 164.

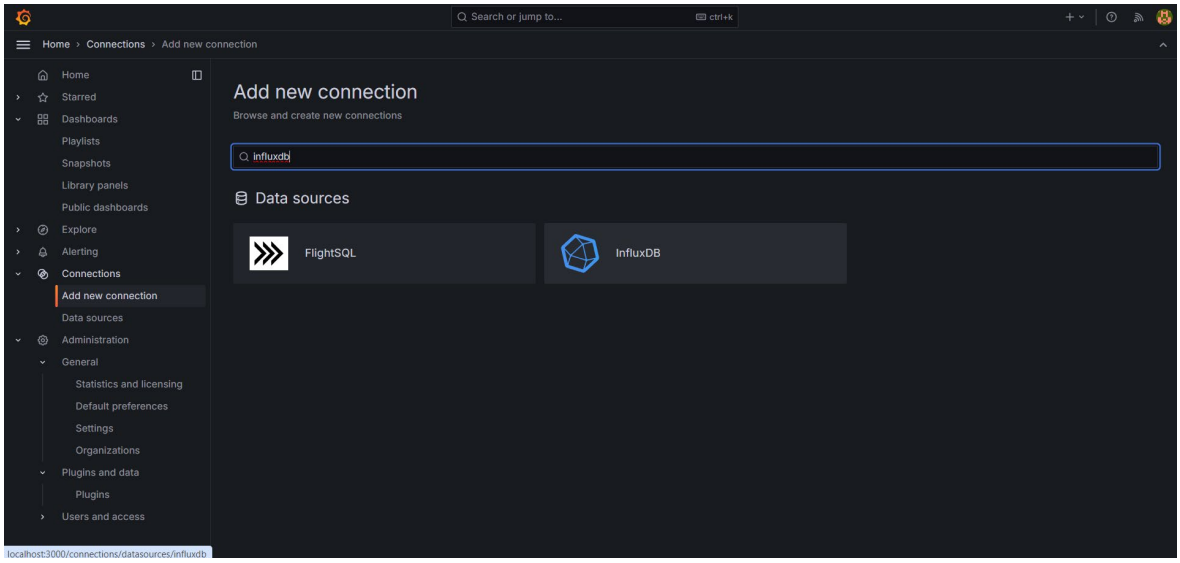


Figura 164 Añadiendo una nueva conexión a Grafana.

En este caso, la conexión se establece con la base de datos InfluxDB Cloud, como se observa en la Figura 165.

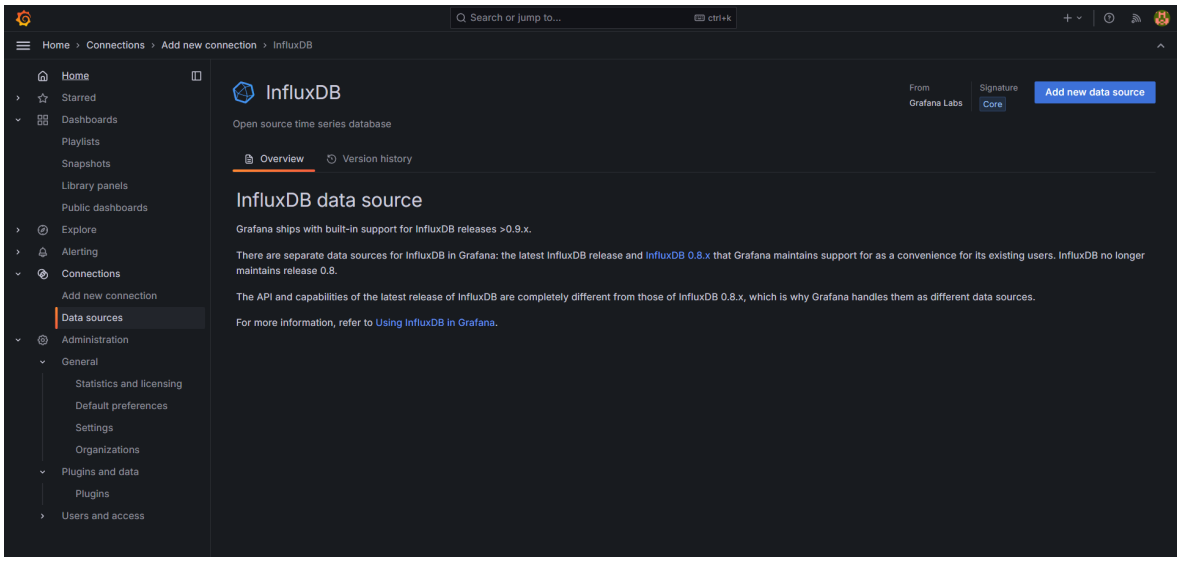


Figura 165 Conectando a InfluxDB.

En la configuración de la fuente de datos se debe introducir la URL de la instancia de InfluxDB Cloud (`https://eu-central-1-1.aws.cloud2.influxdata.com`), como se muestra en la Figura 166. En la sección *Authentication* se debe incluir el `API TOKEN` generado, en el campo correspondiente. A continuación, se debe especificar el *bucket* y la organización que se configuró en InfluxDB Cloud, añadiéndose como *bucket*, `Person_Sensor`, y como organización, `TFM`. Finalmente, en el campo *Auth* se selecciona la opción `With credentials`, como se observa en la Figura 166 y la Figura 167.

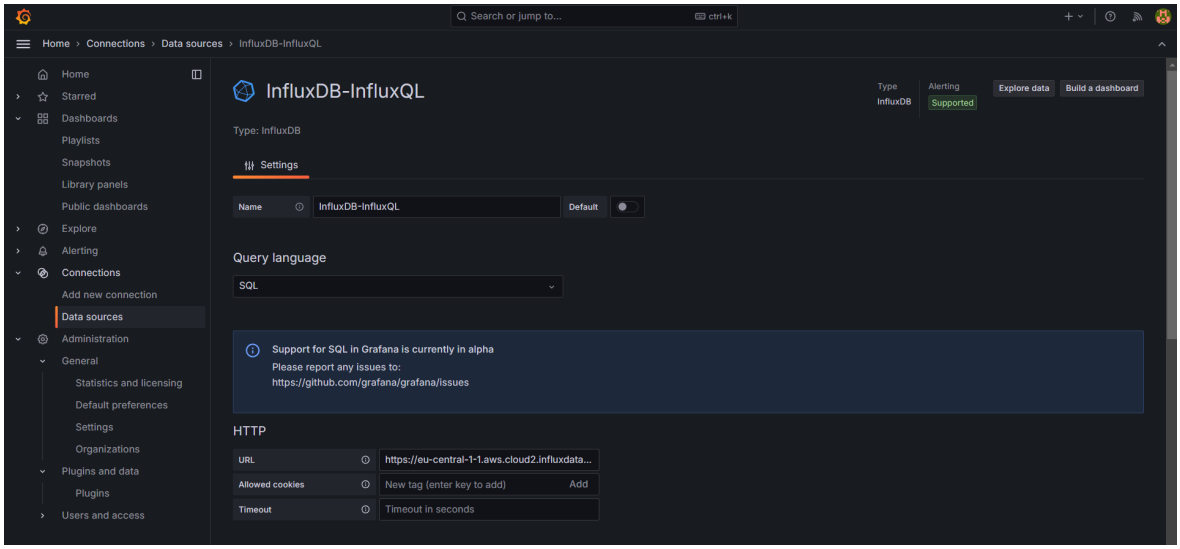


Figura 166 Configuración de la conexión entre Grafana e InfluxDB Cloud.

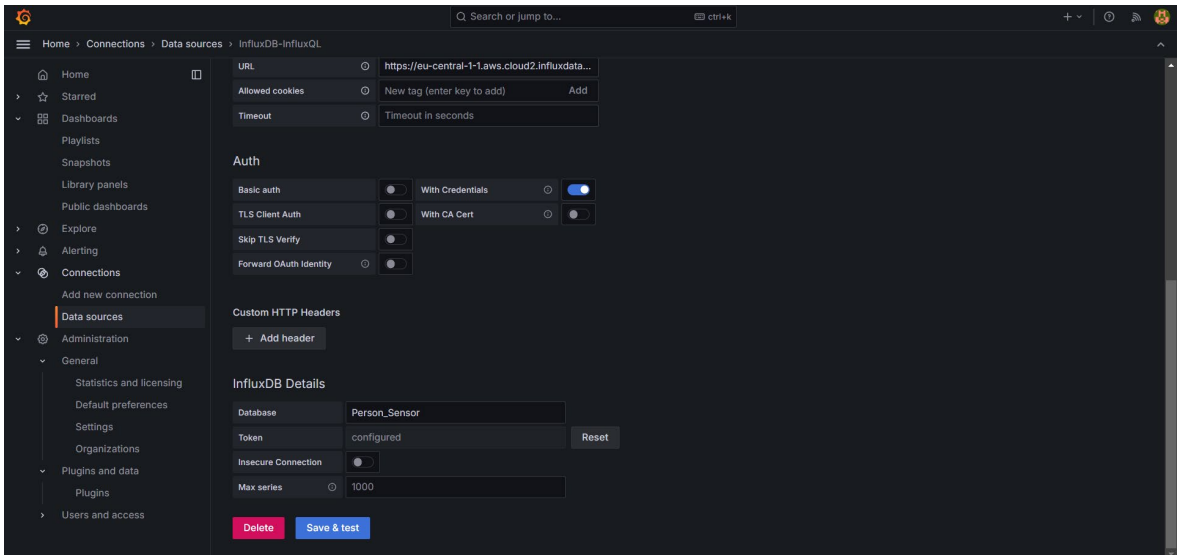


Figura 167 Configuración de la conexión entre Grafana e InfluxDB Cloud.

Una vez completados estos pasos, se debe hacer clic en la opción *Save & Test* para verificar la conexión. Si todo está configurado correctamente, Grafana mostrará un mensaje de éxito, como se observa en la Figura 168, indicando que la conexión con InfluxDB Cloud se ha establecido satisfactoriamente. A partir de este momento, se puede comenzar a crear paneles y visualizaciones utilizando los datos almacenados en InfluxDB Cloud.

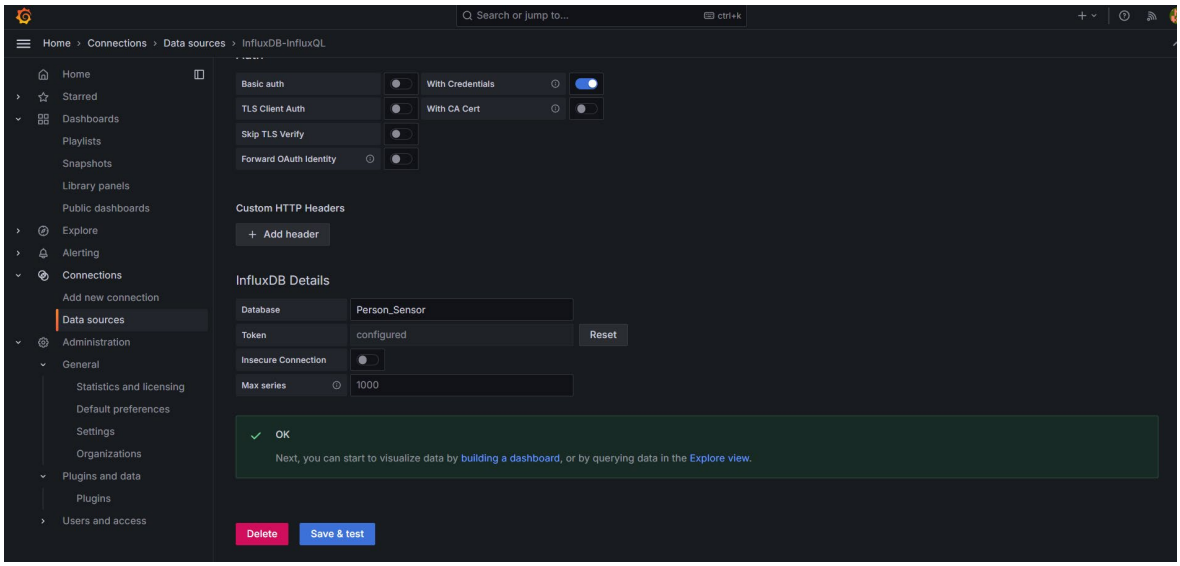


Figura 168 Prueba de conexión correcta entre InfluxDB y Grafana.

7.4 Creación de Dashboard en Grafana

En Grafana, un *dashboard* representa un conjunto de uno o más paneles, organizados y dispuestos en una o más filas, que proporcionan una vista rápida de información relacionada. Estos paneles se crean utilizando componentes que consultan y transforman datos en bruto de fuentes de datos, en gráficos y otras visualizaciones [54].

Un *Data Source* puede ser una base de datos SQL, Grafana Loki, Grafana Mimir, o una API basada en JSON, entre otras opciones. Los *data source plugins* toman una consulta, recuperan los datos de la fuente de datos y los adaptan al modelo de los paneles de Grafana [54].

Las consultas, o *queries*, permiten reducir la totalidad de los datos a un conjunto específico, proporcionando una visualización más manejable. Dado que las fuentes de datos tienen sus propios lenguajes de consulta, los *dashboards* de Grafana proporcionan un panel, denominado *Query editor*, que permite adaptar estas diferencias [54].

Un *panel* es el contenedor que muestra una visualización y proporciona varios controles. Las opciones del panel permiten personalizar muchos aspectos de la visualización, y las opciones difieren según la visualización que se seleccione. Cuando el formato de datos en una visualización no cumple con sus requisitos, se puede aplicar una *transformation* que manipula los datos proporcionados como respuesta a una consulta [54].

Para crear un *dashboard* en Grafana, se navega a la sección *Dashboards* y se hace clic en el icono de *Dashboards*. Desde aquí, se selecciona la opción *New Dashboard*, como se observa en la Figura 169, lo que llevará a una pantalla que mostrará un panel vacío.

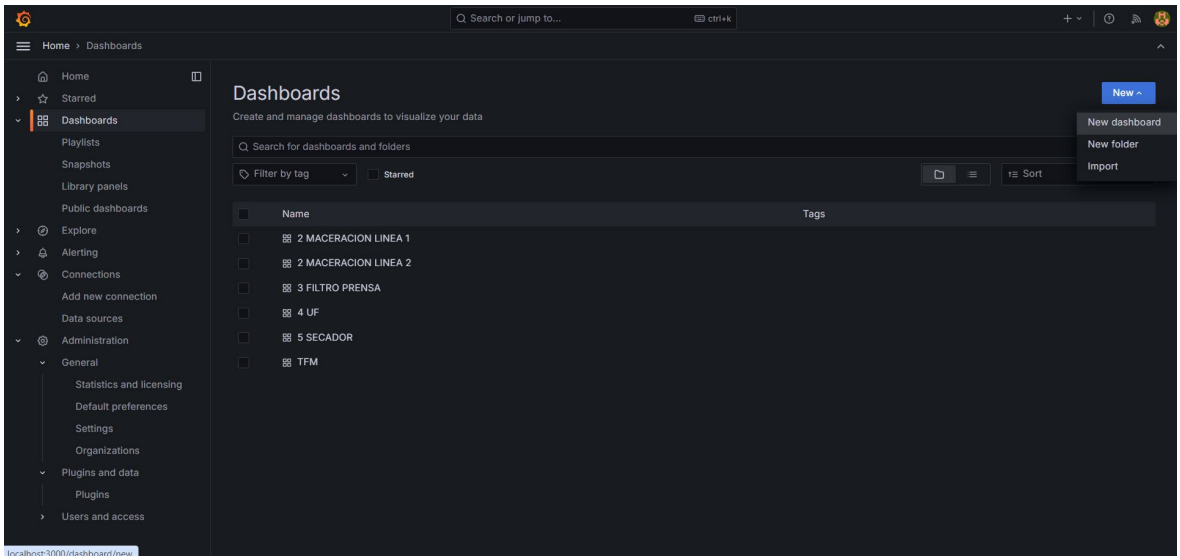


Figura 169 Creación de Dashboard.

Para agregar una nueva visualización se selecciona la opción *Add Visualization*, como se muestra en la Figura 170. Las visualizaciones proporcionan diferentes formas de representar los datos dentro de un panel, dependiendo de lo que mejor se adapte a los datos y a sus necesidades. Las opciones de visualización de Grafana abarcan desde gráficos de series temporales, hasta mapas de calor y representaciones 3D [55].

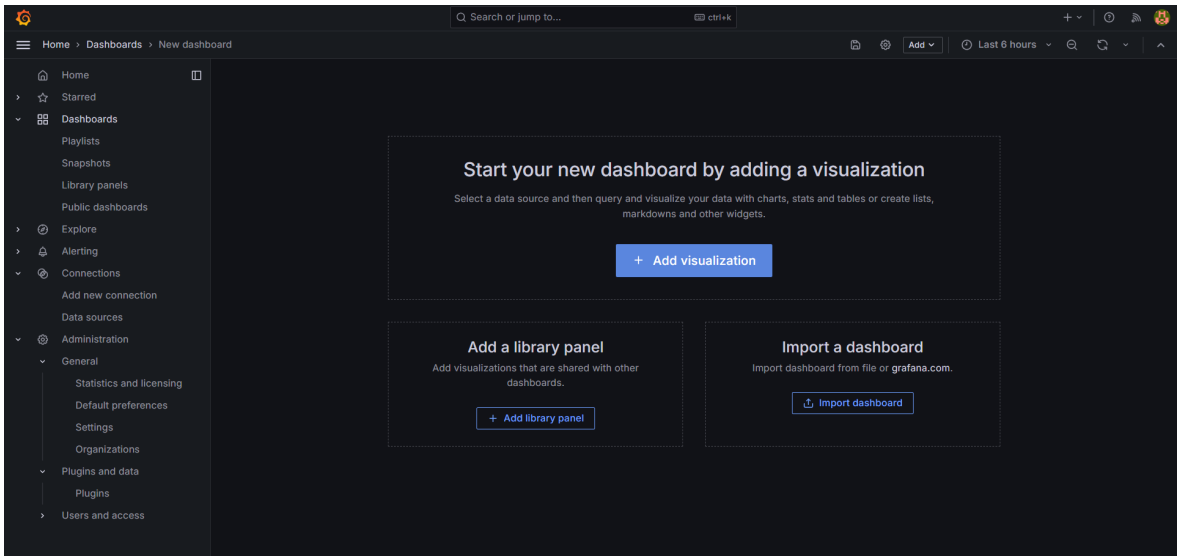


Figura 170 Creación de una Visualización en Grafana.

En este caso, se selecciona en el editor de paneles la fuente de datos que se ha configurado previamente como InfluxDB Cloud, como se representa en la Figura 171.

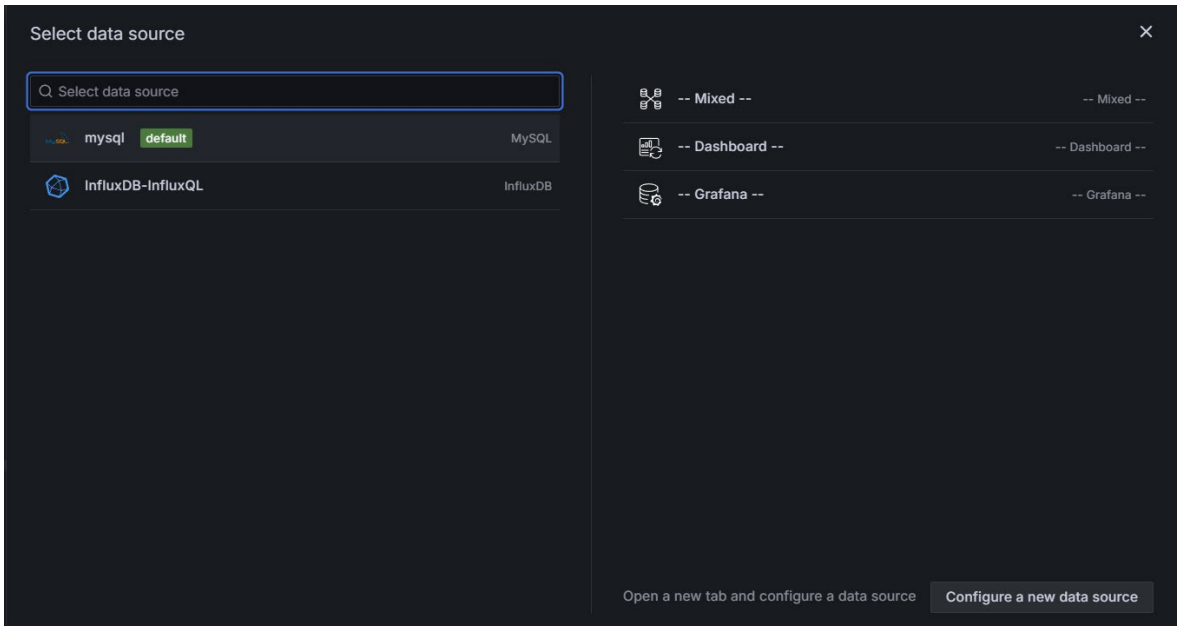


Figura 171 fuente de Datos para el Dashboard.

Posteriormente, se selecciona la tabla de donde proceden los datos, como se muestra en la Figura 172, siendo su nombre en este caso `mQTT_consumer` –similar al *plugin* de *Telegraf* que se ha utilizado–, seleccionándose una columna que representará el *timestamp*, y otra en la que se mostrará la variable a representar gráficamente, como se muestra en la Figura 173.

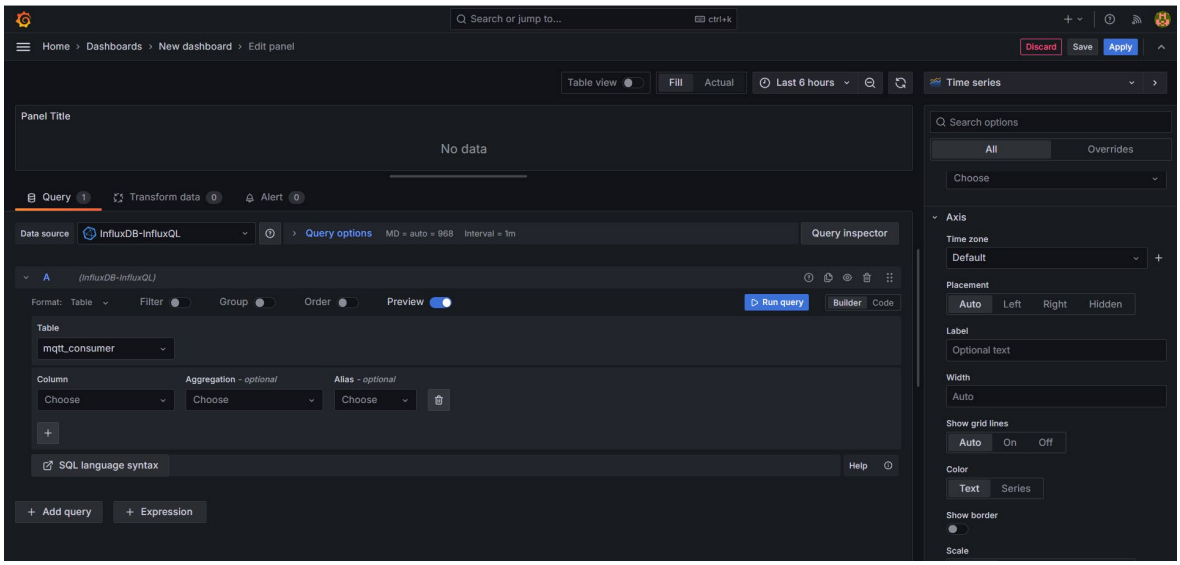


Figura 172 Edición del Panel de Visualización.

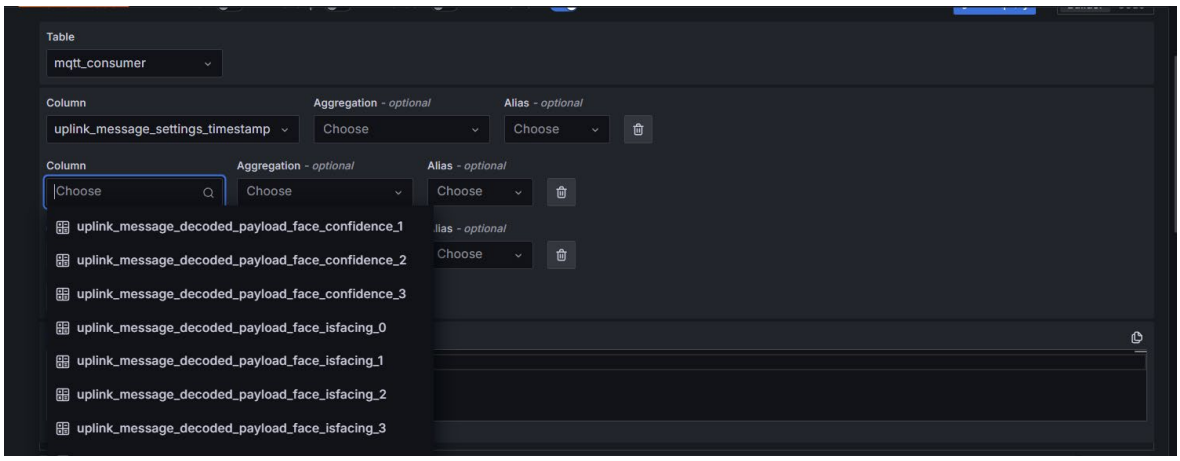


Figura 173 Datos a representar en Grafana.

En la pestaña *Visualization* se configura la visualización del panel, seleccionándose el tipo de gráfico que se desea utilizar. Posteriormente se configuran las opciones de visualización, se ajusta el estilo de las líneas, colores, ejes, leyendas y otras opciones según las necesidades del elemento a mostrar. Una vez configurado el panel, se hace clic en *Apply*, en la parte superior derecha del editor, para almacenar el panel.

El proceso se repite para agregar y configurar nuevos paneles hasta completar el *dashboard*. Cuando se ha terminado con el diseño y el contenido del *dashboard*, se selecciona en la parte superior de la pantalla la opción *Save Dashboard* en el menú desplegable, se le asigna un nombre y, opcionalmente, se proporciona una descripción.

7.5 Representación de los datos almacenados en InfluxDB

El *dashboard* Grafana desarrollado en el presente TFM, representado en la Figura 174 presenta un análisis integral de los datos capturados en el nodo final IoT por el módulo *Person Sensor*, demostrando la capacidad del sistema para monitorizar en tiempo real diversas métricas de rendimiento y detección.

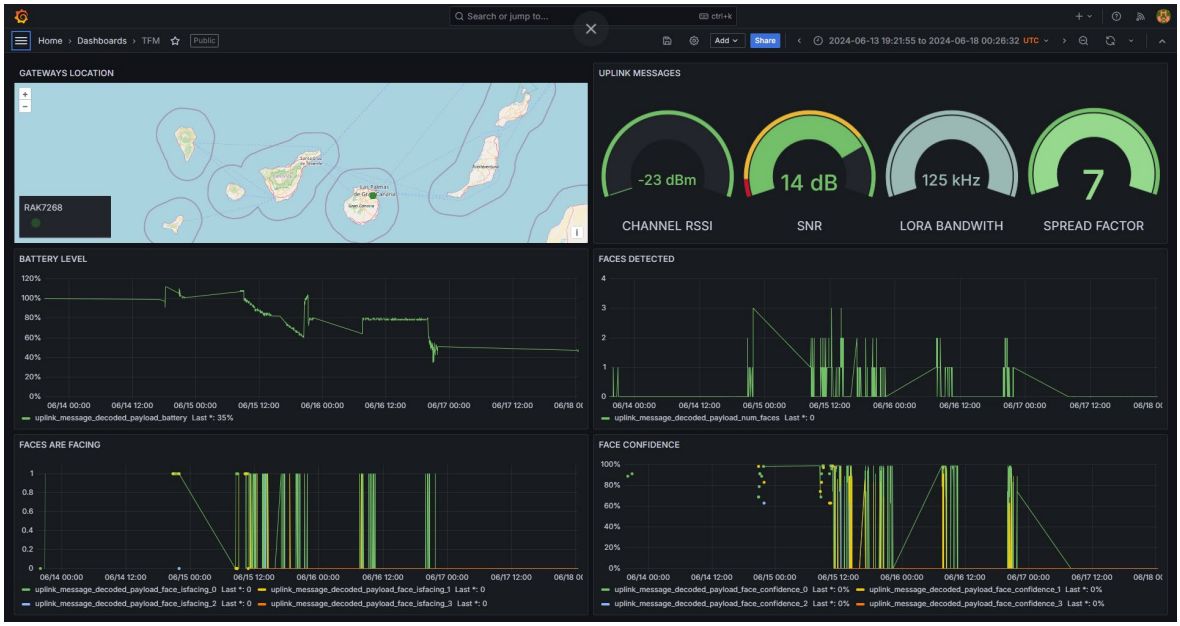


Figura 174 Dashboard TFM.

En la sección superior izquierda, el mapa interactivo muestra la ubicación precisa, en este caso particular del *Gateway* RAK7288, que sirve como nodo central para la recopilación de datos desde los nodos finales en la red LoRaWAN. En la Figura 175 se puede observar con mayor detalle.

Figura 175 Mapa de ubicación del Gateway.

Justo al lado, una serie de indicadores proporcionan información crítica sobre la calidad de la señal LoRa®, incluyendo el nivel de señal recibida (RSSI), la relación señal/ruido (SNR), el ancho de banda utilizado o el *Spread Factor*, como se muestra en la Figura 176, todos ellos parámetros esenciales para evaluar la robustez y eficiencia de la comunicación inalámbrica.

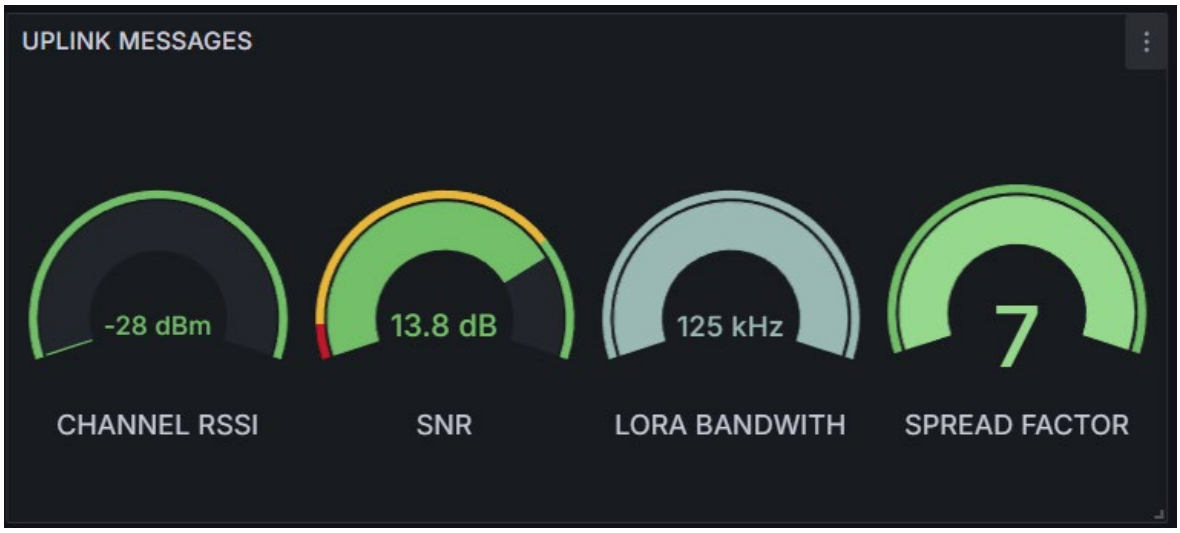


Figura 176 Parámetros de la conexión Uplink.

El gráfico de nivel de batería ofrece una visión temporal del consumo energético del dispositivo Arduino MKRWAN 1310, destacando patrones de uso y posibles puntos de fallo, carga y descarga, como se observa en la Figura 177.

Figura 177 Nivel de Batería.

Por su parte, en el gráfico asociado a la información de detección de rostros, se indica la cantidad de detecciones realizadas por el módulo *Person Sensor* integrado en el nodo final IoT en intervalos específicos, proporcionando datos relevantes a partir de la frecuencia y condiciones de las detecciones.

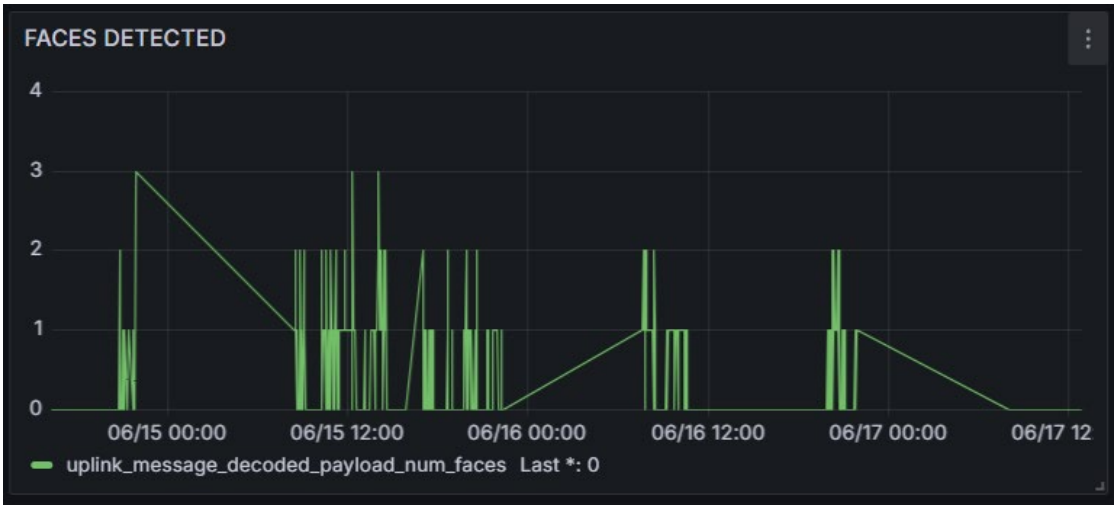


Figura 178 Rostros detectados.

El gráfico *Faces are Facing* desglosa la orientación de las caras detectadas por el módulo *Person Sensor* integrado en el nodo final IoT, como se indica en la Figura 179, lo que puede ser útil para analizar patrones de comportamiento a partir de la interacción de las personas con el entorno monitoreado.

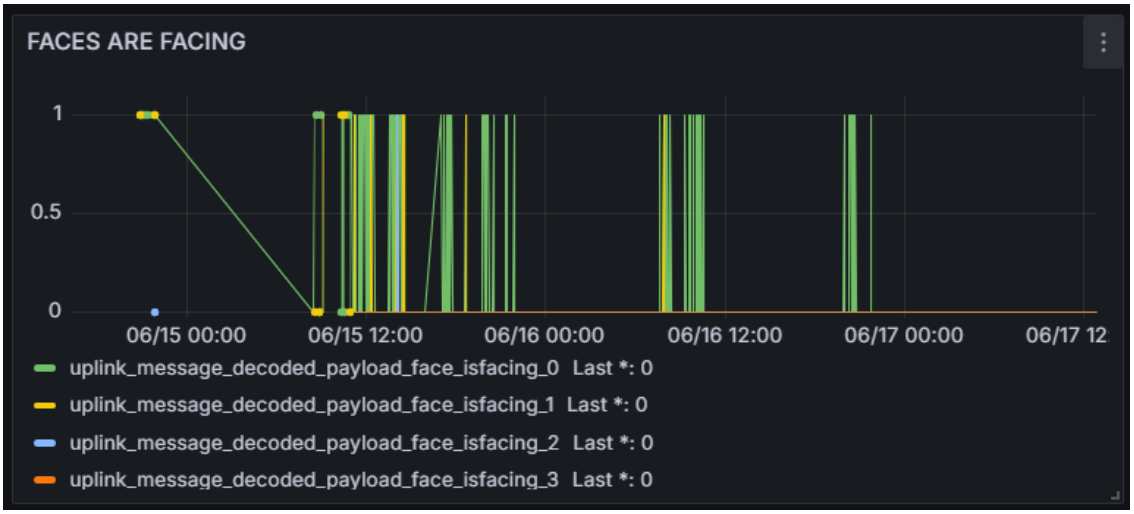


Figura 179 Rostros observando directamente el dispositivo Person Sensor.

Finalmente, la representación *Face Confidence* muestra la confiabilidad de las detecciones realizadas por el módulo *Person Sensor*, como se observa en la Figura 180, indicando con qué precisión se identifican las caras a lo largo del tiempo.

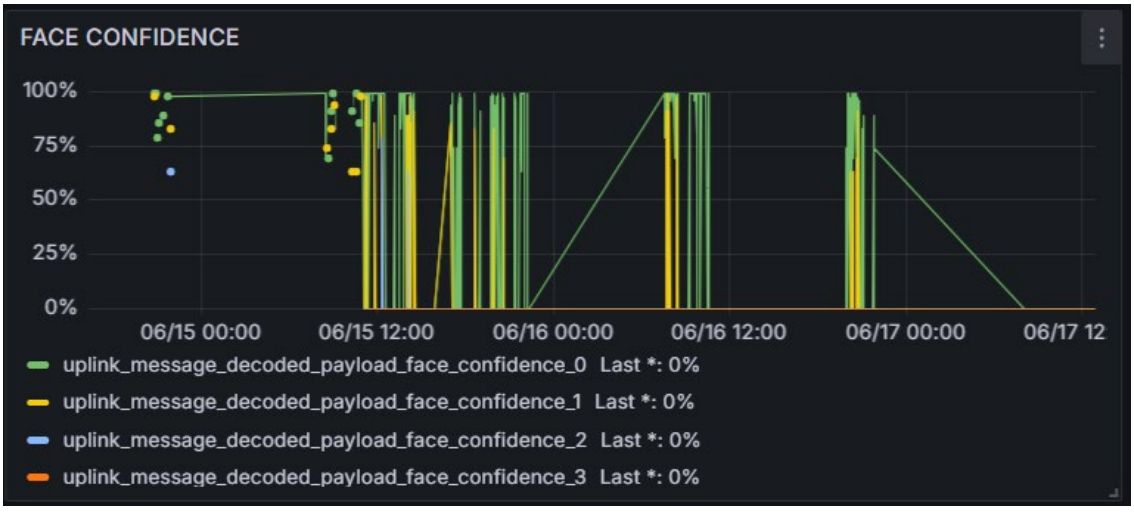


Figura 180 Confidencialidad de los rostros detectados.

En conjunto, este *dashboard* Grafana no solo facilita la supervisión del rendimiento, entre otros elementos de la red LoRaWAN, del módulo *Person Sensor* integrado en los nodos IoT, sino que también proporciona información valiosa sobre el entorno monitorizado, permitiendo así optimizar la configuración y operación del sistema de manera eficiente.

7.6 Análisis del flujo de Datos en la aplicación IoT

En este epígrafe final se presenta un análisis exhaustivo del flujo de datos asociado a la aplicación IoT desarrollada en este TFM como primera propuesta, detallándose cada etapa, desde la captura inicial de la información registrada por el módulo *Person Sensor*, hasta la visualización final en el *dashboard* Grafana. El esquema de la solución propuesta se reproduce por conveniencia en la Figura 181, en la que se proporciona una representación visual que facilita la comprensión completa y estructurada del proceso.

Figura 181 Arquitectura final de la primera solución de aplicación IoT propuesta.

Para verificar la integridad de los datos proporcionados por el módulo *Person Sensor*, y su paso por todas las etapas descritas en la Figura 181, se realizará el análisis de una muestra específica con el fin de asegurar que los valores enviados se corresponden con la información mostrada en Grafana.

Inicialmente, se revisan los datos obtenidos desde el módulo *Person Sensor* conectado al dispositivo Arduino MKRWAN 1310, a partir de su representación en el terminal serie USB. Como referencia, en la Figura 182 se muestran los valores que se envían a la plataforma TTN, siendo en este caso el *payload* del mensaje enviado `02 03 63 59 00 00 68 00`, en formato hexadecimal, transmitido a las 20:59 horas.

```
20:59:23.441 -> * Attempting to connect to TTN network server (OTAA) ... You're connected to TTN network server
20:59:30.028 ->  - channel mask: 0003
20:59:30.028 -> NUM_FACES = 2
20:59:30.070 -> BOX_CONFIDENCE[0] = 99
20:59:30.070 -> IS_FACING[0] = 1
20:59:30.070 -> BOX_CONFIDENCE[1] = 89
20:59:30.070 -> IS_FACING[1] = 1
20:59:30.070 -> BOX_CONFIDENCE[2] = 0
20:59:30.070 -> IS_FACING[2] = 0
20:59:30.070 -> BOX_CONFIDENCE[3] = 0
20:59:30.070 -> IS_FACING[3] = 0
20:59:30.070 -> 1
20:59:30.070 -> 3
20:59:30.070 -> 3
20:59:30.070 -> 3
20:59:30.070 -> 626
20:59:30.070 -> 68
20:59:30.070 -> - Sending message to TTN with data (8) 2 3 63 59 0 0 68 0
20:59:30.070 ->    - Current ADR / DR / Channel mask: 1 / 5 / 0003
20:59:30.116 ->    - Current channel mask: 0003
20:59:30.151 -> - Message sent correctly!
20:59:45.271 ->   ... Putting device into deepSleep mode ...  wake up from deepSleep!
```

Figura 182 Visualización de los datos directamente conectados al dispositivo Arduino MKRWAN 1310.

Tras la revisión de los datos en la plataforma IoT TTN, se verificó que todos muestran la misma secuencia: `02 03 63 59 00 00 68 00`, como se observa en la Figura 183. Tras su decodificación, se puede determinar que el nivel de carga de la batería es de 104%, que el módulo *Person Sensor* detectó la presencia de dos rostros con niveles de confianza del 99% y 89%, respectivamente, y que ambos individuos se encontraban mirando directamente hacia la cámara, de acuerdo con el valor del parámetro *Is Facing*.

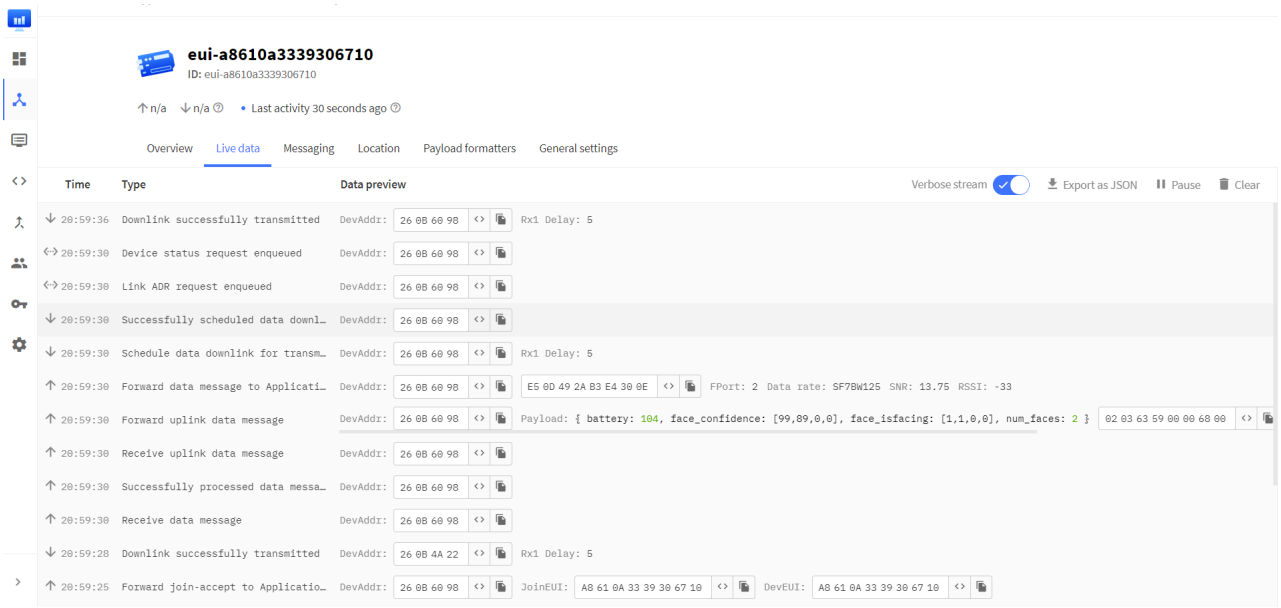


Figura 183 Live data en TTN desde dispositivo final MKRWAN 1310.

En la Figura 184 se muestra una referencia de los datos enviados a InfluxDB Cloud desde la plataforma TTN a través del agente *Telegraf* configurado, donde se puede apreciar que a las 19:59 horas se registró un punto de datos. Sin embargo, debido a la configuración de la franja horaria, se refleja una diferencia de una hora menos.

```
2024-06-25T19:57:28Z D! [outputs.influxdb_v2] Buffer fullness: 0 / 10000 metrics
2024-06-25T19:57:38Z D! [outputs.influxdb_v2] Buffer fullness: 0 / 10000 metrics
2024-06-25T19:57:48Z D! [outputs.influxdb_v2] Buffer fullness: 0 / 10000 metrics
2024-06-25T19:57:58Z D! [outputs.influxdb_v2] Buffer fullness: 0 / 10000 metrics
2024-06-25T19:58:08Z D! [outputs.influxdb_v2] Buffer fullness: 0 / 10000 metrics
2024-06-25T19:58:18Z D! [outputs.influxdb_v2] Buffer fullness: 0 / 10000 metrics
2024-06-25T19:58:28Z D! [outputs.influxdb_v2] Buffer fullness: 0 / 10000 metrics
2024-06-25T19:58:38Z D! [outputs.influxdb_v2] Wrote batch of 1 metrics in 239.2554ms
2024-06-25T19:58:38Z D! [outputs.influxdb_v2] Buffer fullness: 0 / 10000 metrics
2024-06-25T19:58:48Z D! [outputs.influxdb_v2] Buffer fullness: 0 / 10000 metrics
2024-06-25T19:58:58Z D! [outputs.influxdb_v2] Buffer fullness: 0 / 10000 metrics
2024-06-25T19:59:08Z D! [outputs.influxdb_v2] Buffer fullness: 0 / 10000 metrics
2024-06-25T19:59:18Z D! [outputs.influxdb_v2] Buffer fullness: 0 / 10000 metrics
2024-06-25T19:59:28Z D! [outputs.influxdb_v2] Buffer fullness: 0 / 10000 metrics
2024-06-25T19:59:38Z D! [outputs.influxdb_v2] Wrote batch of 1 metrics in 76.5907ms
2024-06-25T19:59:38Z D! [outputs.influxdb_v2] Buffer fullness: 0 / 10000 metrics
2024-06-25T19:59:48Z D! [outputs.influxdb_v2] Buffer fullness: 0 / 10000 metrics
2024-06-25T19:59:58Z D! [outputs.influxdb_v2] Buffer fullness: 0 / 10000 metrics
```

Figura 184 Métricas enviadas por Telegraf a InfluxDB.

Al revisar la información mostrada en la consola de InfluxDB Cloud, se filtran los datos en función de la hora asociada a la muestra enviada, apreciándose que los datos recibidos son idénticos a los enviados desde el dispositivo Arduino MKRWAN 1310, como se muestra en la Figura 185, la Figura 186, la Figura 187, la Figura 188, la Figura 189, la Figura 190 y la Figura 191. Estos datos están representados en forma de tabla para facilitar su visualización y análisis.

Figura 185 Datos de la muestra en InfluxDB (1).

Figura 186 Datos de la muestra en InfluxDB (2)

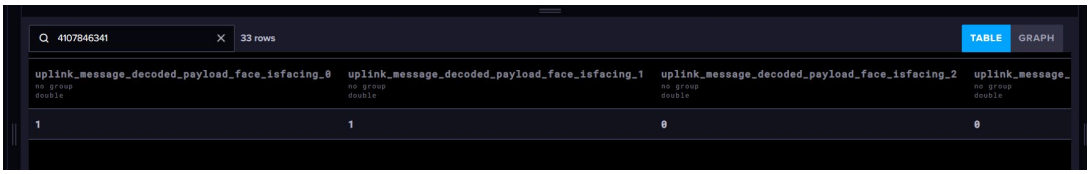


Figura 187 Datos de la muestra en InfluxDB (3).

Figura 188 Datos de la muestra en InfluxDB (4).

Figura 189 Datos de la muestra en InfluxDB (5).

Figura 190 Datos de la muestra en InfluxDB (6).

Figura 191 Datos de la muestra en InfluxDB (7).

Una vez en Grafana, al desglosar las visualizaciones creadas individualmente, se destaca la variable correspondiente al nivel de la batería. Como se puede observar en la Figura 192, a las 19:59 horas se registra un valor de 104%, consistente con los valores observados en etapas anteriores para ese mismo horario.

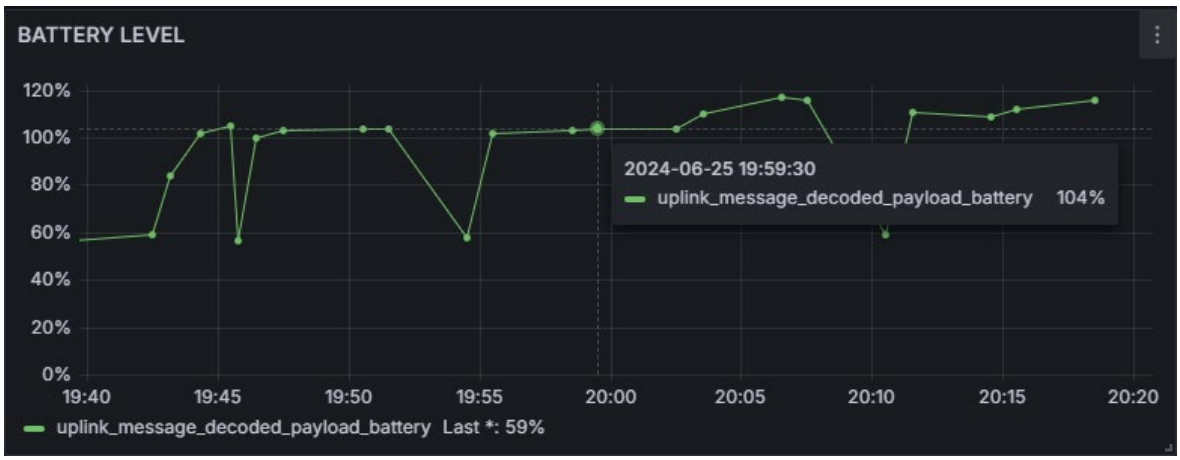


Figura 192 Nivel de Batería conectada al dispositivo Arduino MKRWAN 1310.

De igual modo, el número de rostros detectados también coincide con las mediciones anteriores, como se aprecia en la Figura 193, donde se registran un total de 2 rostros a las 19:59 horas.

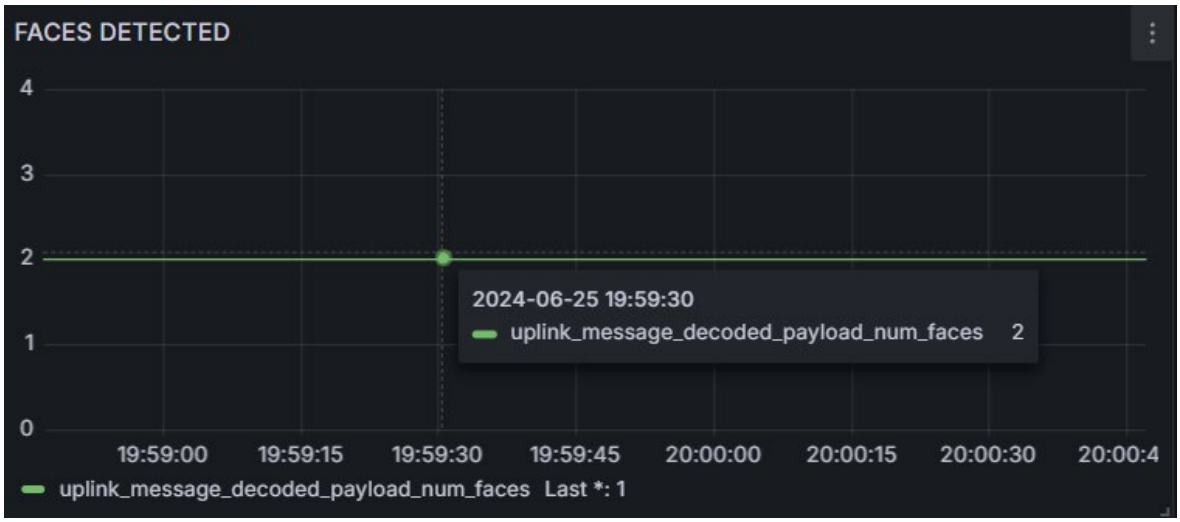


Figura 193 Rostros Detectados.

En la Figura 194 se indica que los dos rostros detectados están mirando directamente hacia el módulo *Person Sensor*.

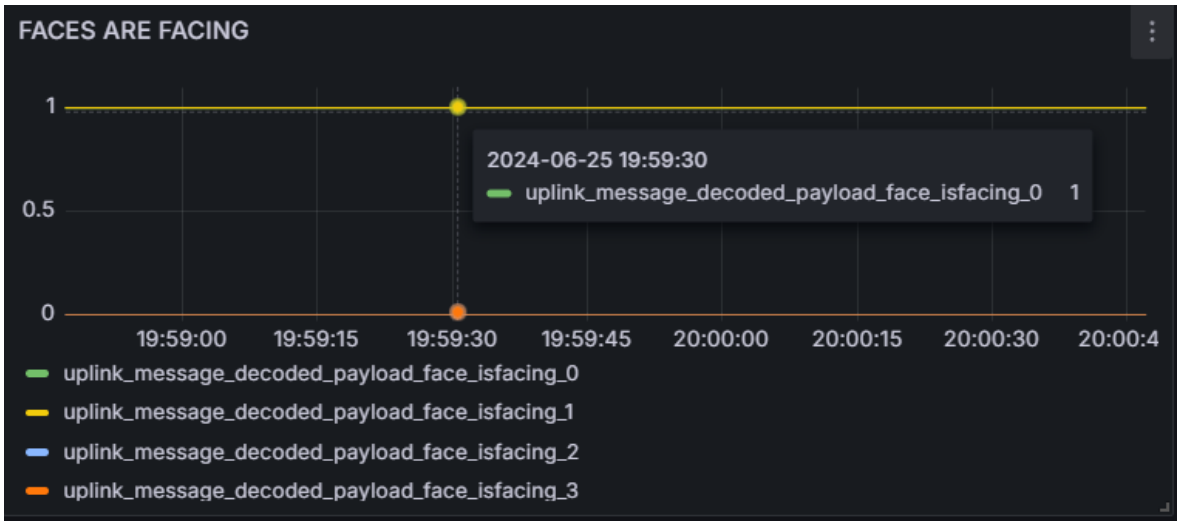


Figura 194 Parámetro que evalúa la mirada directa de los rostros al dispositivo Person Sensor.

En la Figura 195 y la Figura 196 se observa que el valor de confianza para el primer rostro detectado es de 99%, mientras que para el segundo rostro es de 89%.

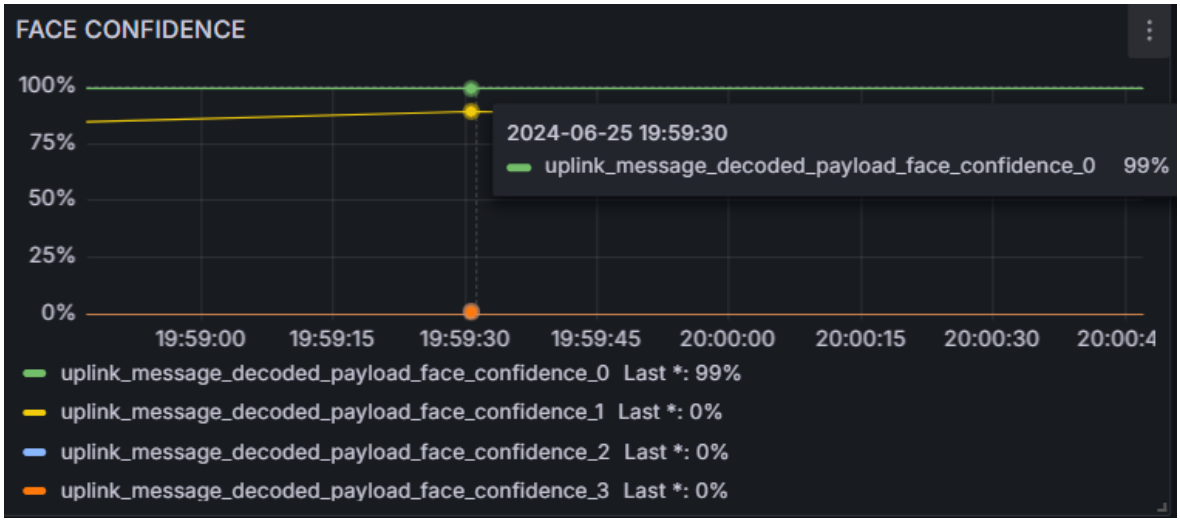


Figura 195 Confidencialidad de los rostros detectados, rostro 1.

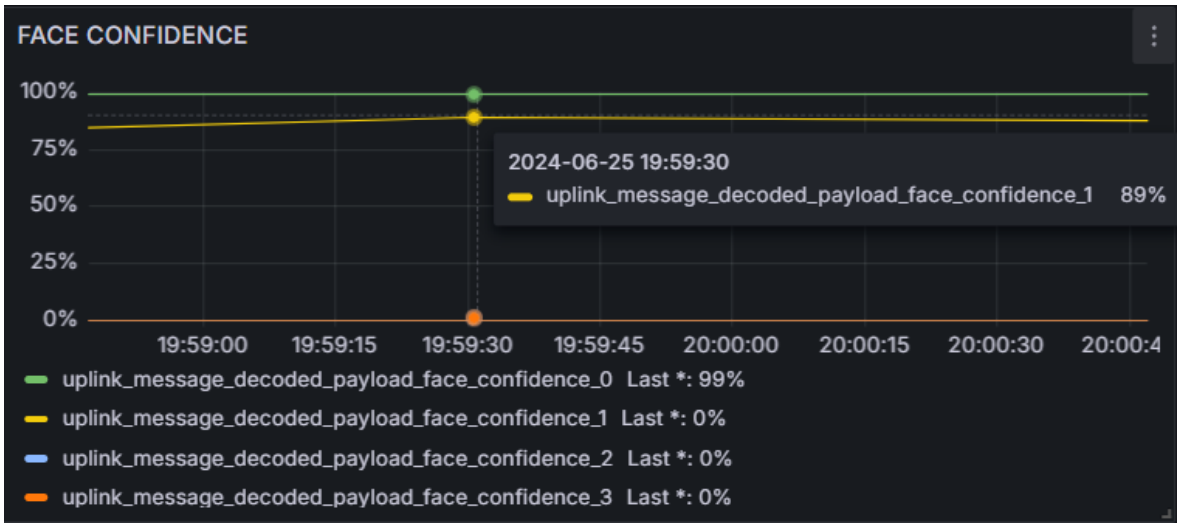


Figura 196 Confidencialidad de los rostros detectados, rostro 2.

En consecuencia, el análisis del flujo de datos revela una consistencia en las métricas registradas en la implementación de la aplicación IoT asociada a la primera de las propuestas planteadas en este TFM. Los datos obtenidos mostraron coherencia en el nivel de la batería, la detección de rostros y sus niveles de confianza a lo largo del tiempo. Además, se confirmó que los rostros detectados estaban directamente frente al módulo *Person Sensor* en el momento de la captura de datos. Este proceso subraya la efectividad de la configuración y el despliegue de los sistemas de monitoreo, proporcionando una base sólida para análisis continuos y la toma de decisiones.

8 Aplicación IoT LoRaWAN basada en *Network Server* integrado en RAK7268

En este capítulo se presenta la segunda propuesta planteada en el desarrollo del presente TFM para la implementación de una aplicación IoT basada en el protocolo LoRaWAN, que permita realizar el análisis de patrones de comportamiento a partir de la detección de rostros de personas. Esta propuesta, que se reproduce por conveniencia en la Figura 197, contempla la conexión del dispositivo Arduino MKRWAN 1310 con el *Gateway* RAK7268, que en este caso asumirá el rol de *Network Server* (*NS*) además de proporcionar un *broker* MQTT integrado que facilitará la comunicación directa con la base de datos InfluxDB Cloud mediante el uso de una nueva configuración del agente Telegraf, sin necesidad de recurrir a la plataforma TTN, y visualizando finalmente los datos en la herramienta Grafana.

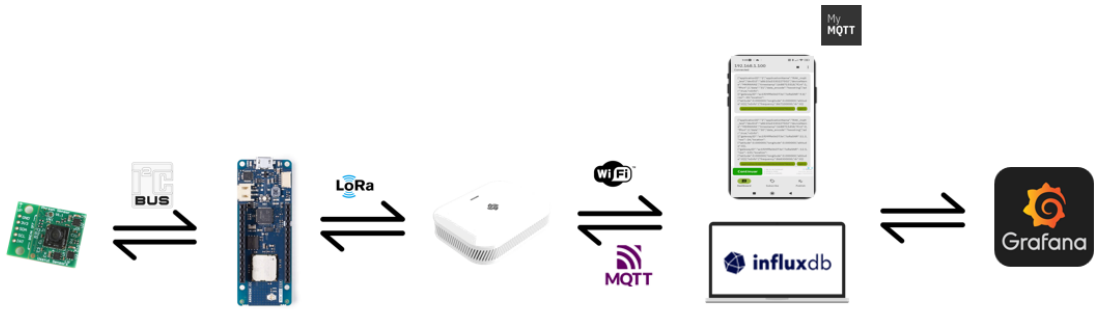


Figura 197 Implementación Arquitectura Built-In MQTT Broker

La posibilidad de configurar el *Gateway* RAK7268 como *Network Server* de la red LoRaWAN proporciona versatilidad y flexibilidad al sistema, facilitando la integración entre los nodos finales y la plataforma de almacenamiento InfluxDB Cloud. Además, el uso del *broker* MQTT integrado, en el que el Gateway RAK7268 actúa al mismo tiempo como *Publisher* y como *Broker* simplifica, tanto la configuración como el intercambio de datos, lo que contribuye significativamente a la viabilidad y eficacia de esta propuesta en entornos de IoT.

8.1 Configuración del modo *Network Server* en el Gateway RAK7268

Para iniciar la implementación de esta propuesta, es necesario modificar en primer lugar la configuración establecida inicialmente el *Gateway* RAK7268 para que funcione en modo *Network Server*, en lugar de en modo *LoRa Basics Station*, tal como se muestra en la Figura 198.

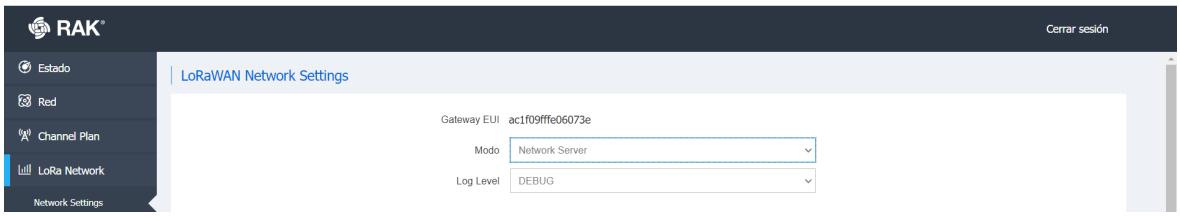


Figura 198 Configuración RAK 7268 como Network Server.

En la Figura 199 y la Figura 200 se presenta en detalle la configuración de los parámetros del servidor LoRaWAN en el *Gateway* RAK7268. Estos parámetros incluyen *Data Rate*, que determina la tasa de transmisión de los datos, y la región EU868, que establece la frecuencia predeterminada para asegurar la compatibilidad con los dispositivos LoRa® en esa zona. Además, se ajustan otros parámetros esenciales para garantizar una comunicación eficiente y un funcionamiento óptimo del sistema, permitiendo una transmisión de datos fiable y segura en el entorno de la red LoRaWAN.

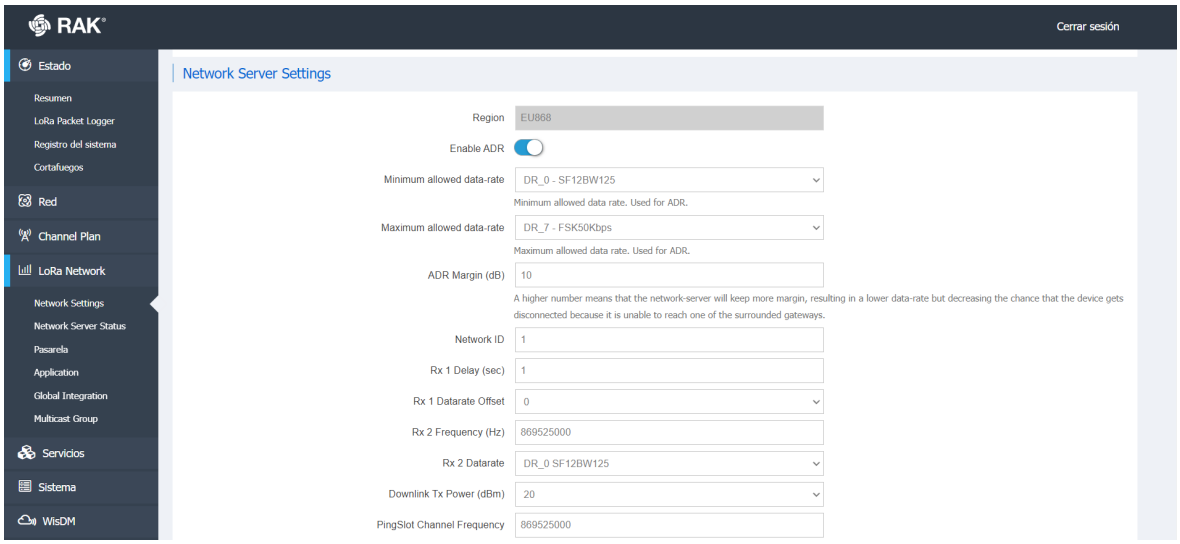


Figura 199 Parámetros del RAK7268 trabajando como Network Server (1).

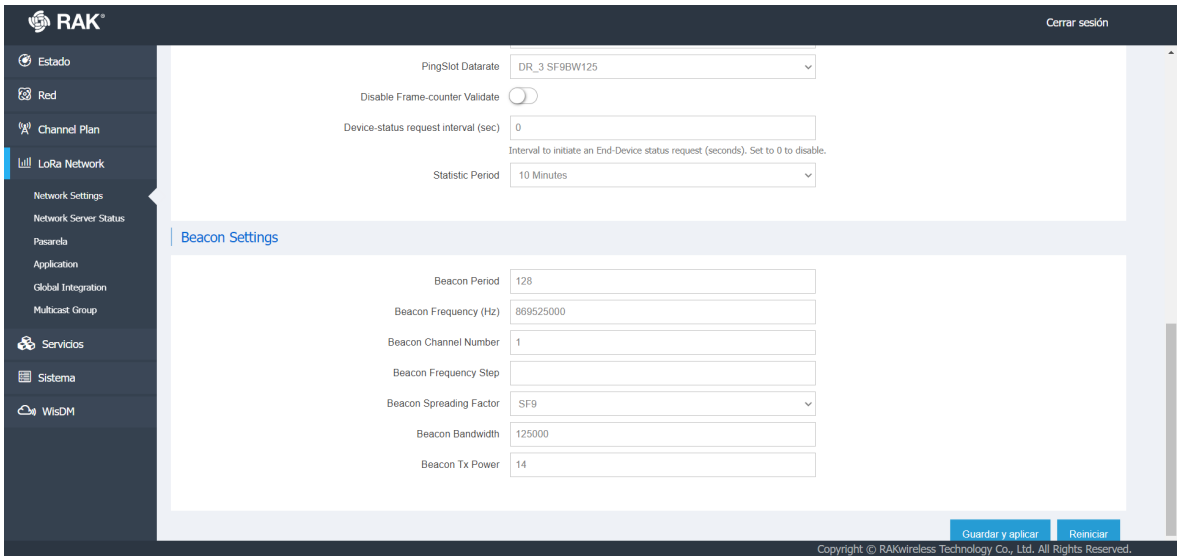


Figura 200 Parámetros del RAK7268 trabajando como Network Server (2).

A continuación, se agregan los datos específicos de la puerta de enlace RAK7268. Esto incluye asignarle un nombre distintivo y proporcionar su ubicación precisa. Para este proyecto, se toma como referencia el Edificio de Electrónica y Telecomunicación de la Universidad de las Palmas de Gran Canaria, tal como se muestra en la Figura 201 y la Figura 202.

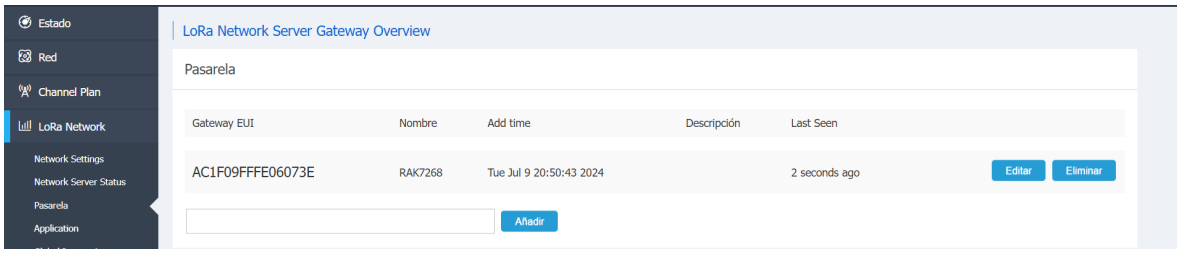


Figura 201 Datos de la Pasarela.

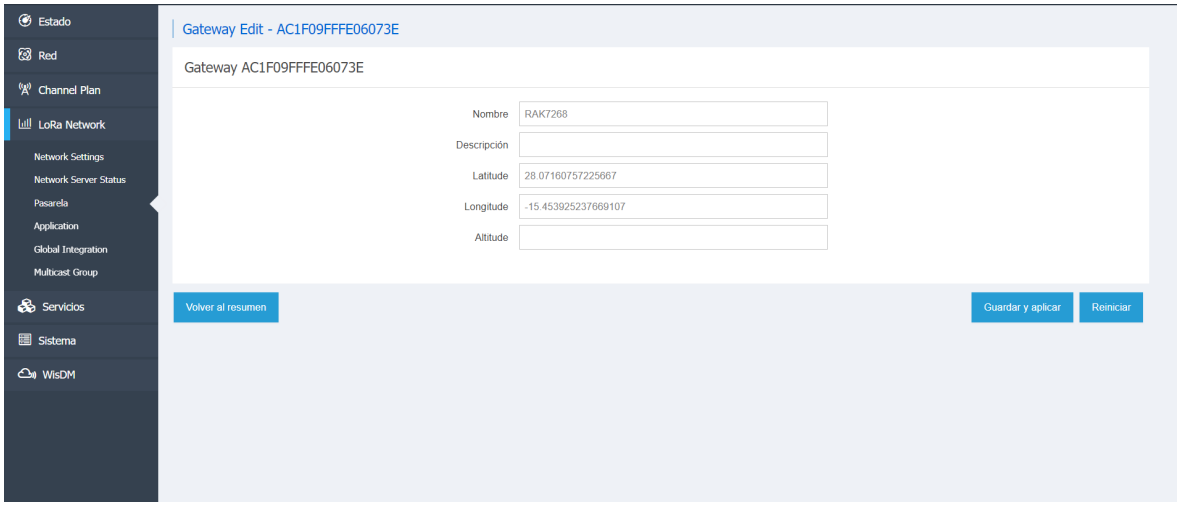


Figura 202 Configuración de la pasarela RAK7268.

En la Figura 203 se muestra la configuración de los parámetros del *Broker* MQTT integrado en el *Gateway* RAK7268, especificando la versión, y habilitando la autenticación en caso necesario. La configuración adecuada del *Broker* MQTT resulta crucial para garantizar una comunicación segura y eficiente entre los dispositivos IoT y el servidor. Al habilitar la autenticación, se añade una capa adicional de seguridad, protegiendo los datos transmitidos y asegurando que únicamente los dispositivos autorizados puedan interactuar con el servidor, lo que es vital para mantener la integridad y la confidencialidad de la información en la red.

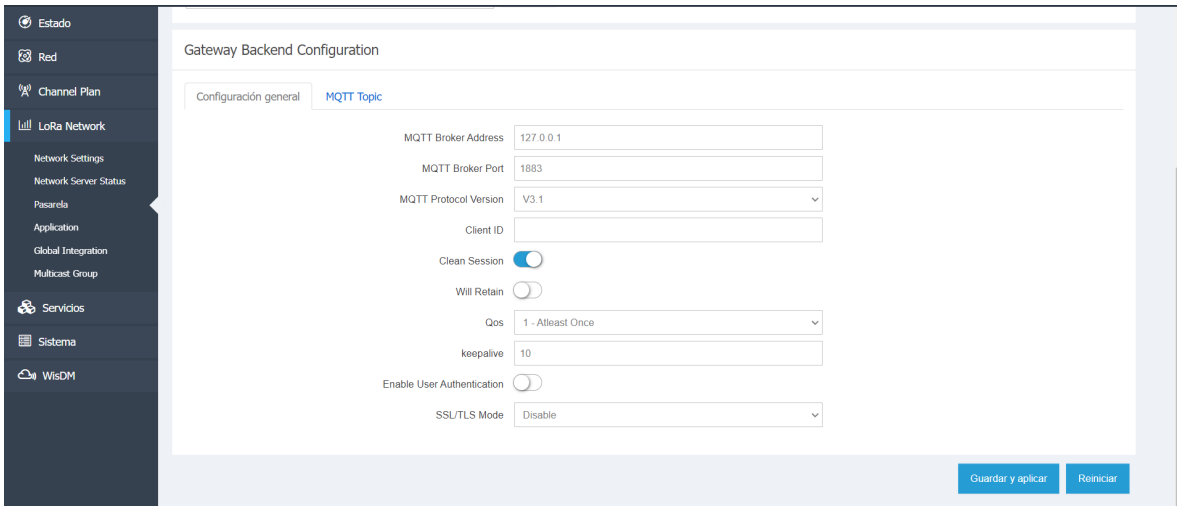


Figura 203 Configuración del Broker MQTT.

8.1.1 Creación de la aplicación

En la Figura 204 se ilustra la creación de una aplicación denominada `Person_Sensor` en el *Network Server* integrado en el *Gateway* RAK7268, mientras que en la Figura 205 se muestra en detalle el proceso de añadir el dispositivo Arduino MKRWAN 1310 a la aplicación a partir de la especificación de su EUI.

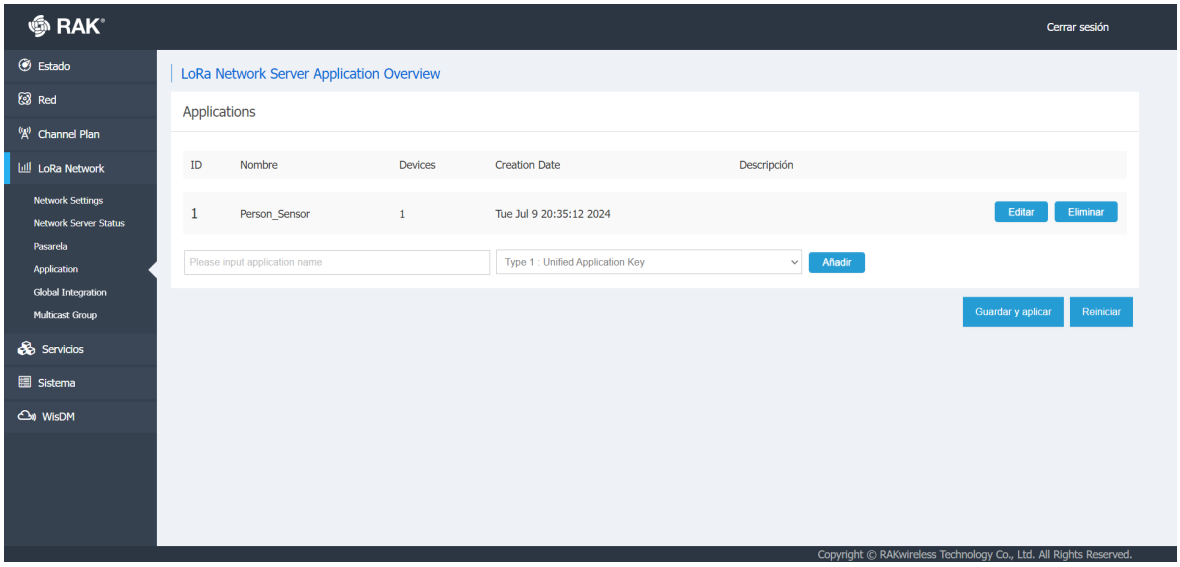


Figura 204 Aplicación Person_Sensor.

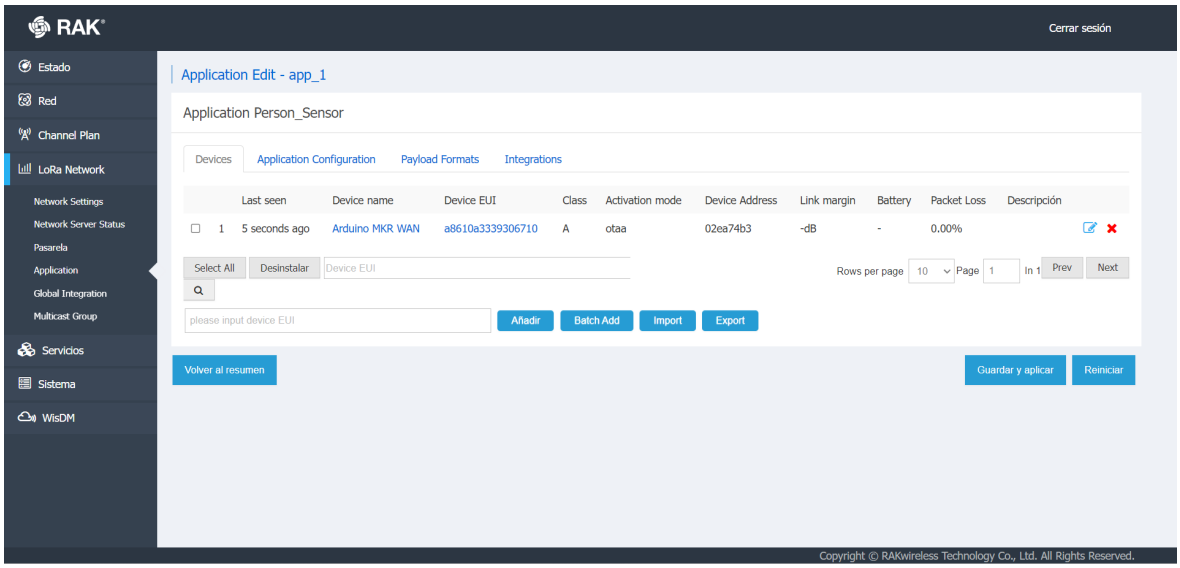


Figura 205 Adición de Dispositivo MKRWAN a la aplicación Person_Sensor.

En la sección *"Application Configuration"* del menú de configuración de la aplicación creada, se debe seleccionar la opción "*Auto Add LoRa Device*" con el fin de detectar automáticamente los dispositivos que cuenten con las credenciales correctas, entre las que se encuentran las claves *Application EUI* y *Application Key*, tal como se muestra en la Figura 206. Estos parámetros serán necesarios para incluirlos en el *sketch* del dispositivo Arduino MKRWAN 1310, como se observa en la Figura 207, asegurando así la correcta integración de éste en la red LoRaWAN.

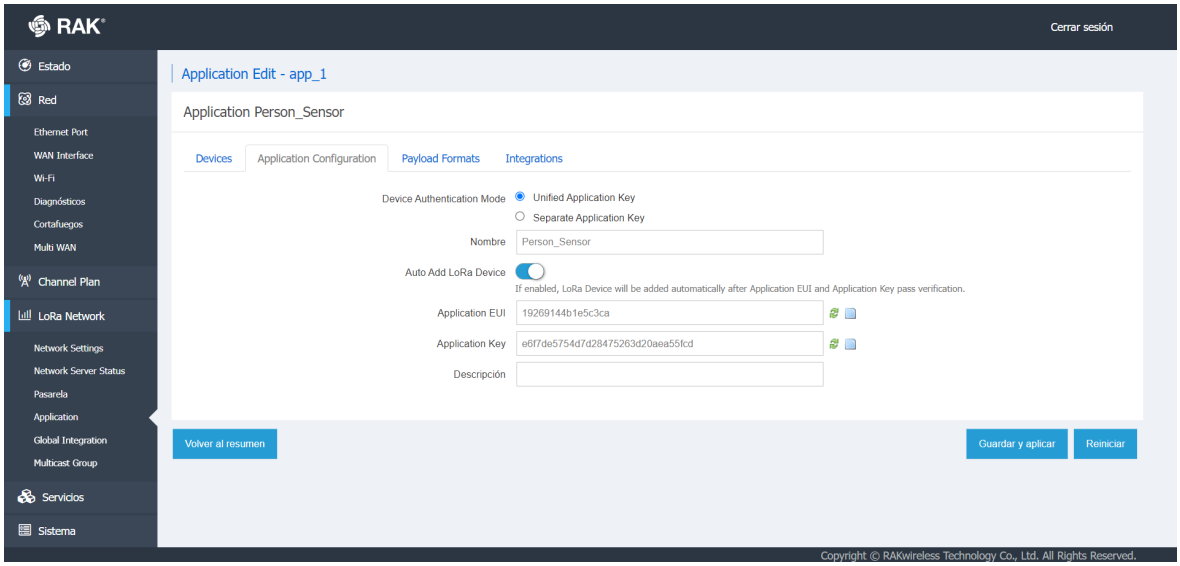


Figura 206 Configuración de la Aplicación.

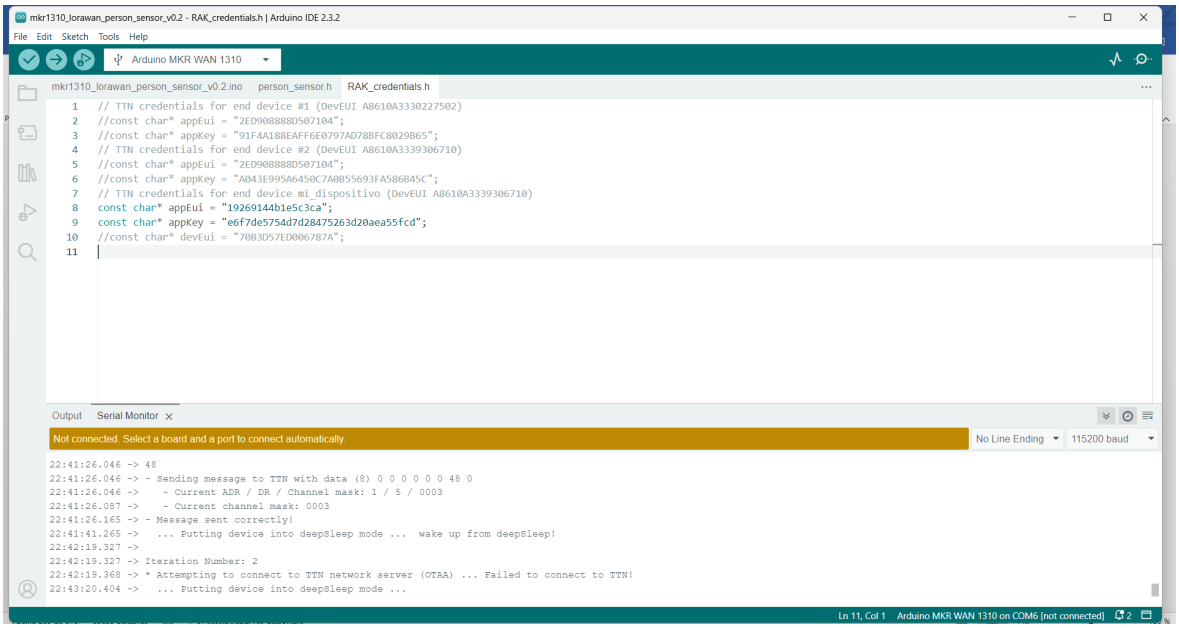


Figura 207 Configuración del AppEUI y el AppKey en el sketch para la segunda propuesta del TFM

Por otro lado, en la sección "*Payload Formats*" de la aplicación se puede especificar el formato de la carga útil (*Payload Format*), configurado por defecto al valor "*None*", como se muestra en la Figura 208. Esta opción permite definir cómo se estructuran los datos que se envían a través del sistema, lo cual es crucial para asegurar que la información se interprete correctamente en etapas posteriores del procesamiento de los datos. La configuración "*None*" indica que no se especifica un formato específico de *payload*, lo que puede resultar útil durante las fases iniciales de desarrollo. Además, en esta sección se encuentra la opción "*Only forward the parsed data object*" que se puede activar o desactivar. Esta opción proporciona un control adicional sobre los datos transmitidos, permitiendo al usuario decidir si solo se deben enviar los datos ya analizados y procesados (*parsed data*), o los datos en bruto.

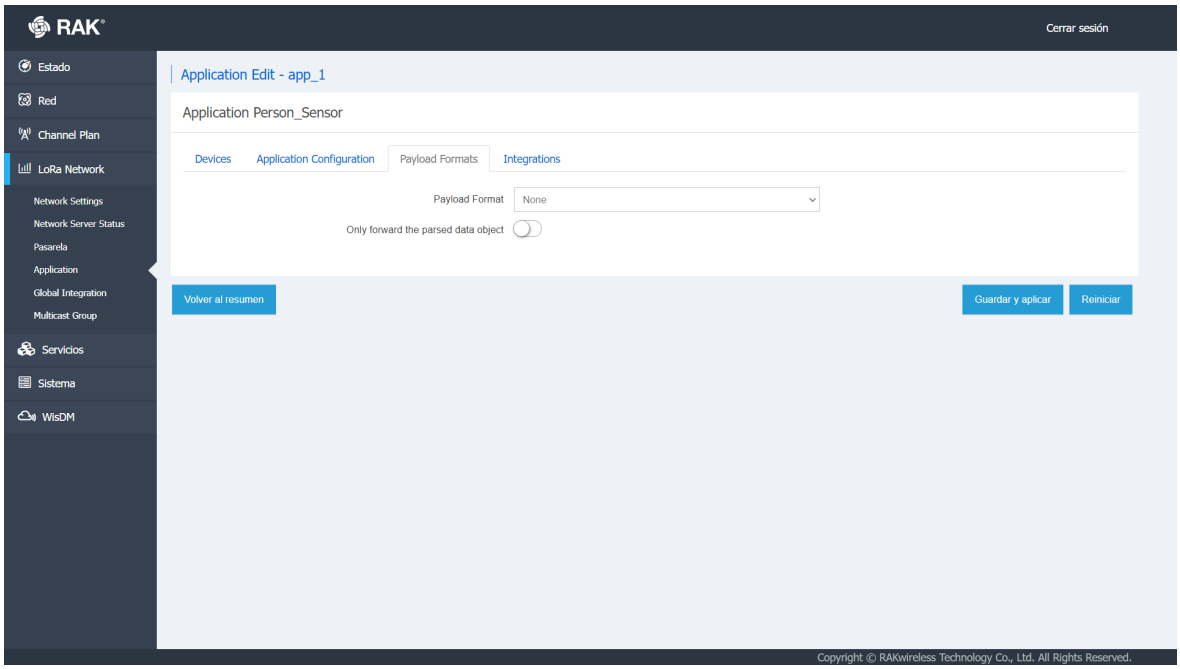


Figura 208 Configuración de Payload Formats en la aplicación.

Finalmente, en la Figura 209 se muestra la sección *"Integrations"* de la aplicación, en la que se pueden configurar diversas opciones. Entre los elementos destacados se encuentran la selección del tipo de codificación/decodificación de los datos a través de un menú desplegable, configurado por defecto en "HEX String", y la activación/desactivación de la información de radio LoRa® mediante un interruptor dedicado. También se puede habilitar la integración HTTP/HTTPS, la cual proporciona campos adicionales para configurar *headers* específicos. Además, existen campos que permiten incluir URL asociadas a diferentes tipos de notificaciones y datos, como la URL del enlace ascendente, notificaciones de unión, o reconocimiento y estado del dispositivo. Otros ajustes incluyen la especificación del máximo número de conexiones simultáneas y la longitud máxima de la cola. Finalmente, se ofrece la opción de activar o desactivar la integración FTP, completando así las opciones disponibles para configurar esta aplicación en el entorno del *Gateway* RAK7268.

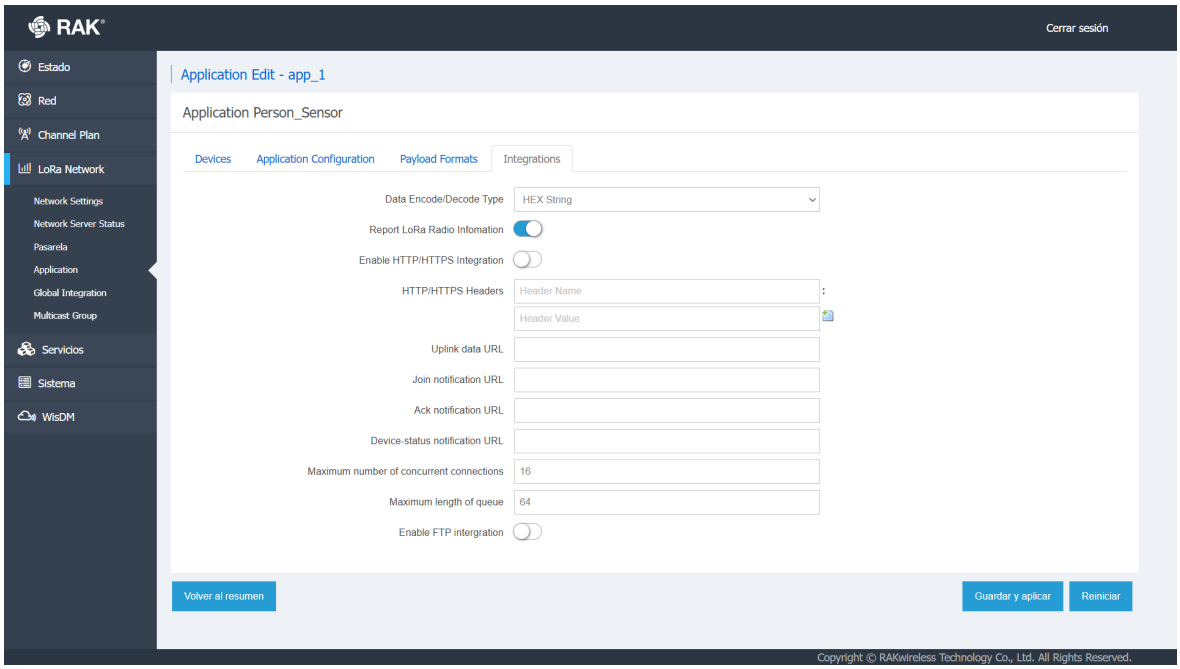


Figura 209 Configuración de Integrations en la aplicación

8.1.2 Registro del dispositivo Arduino MKRWAN 1310

Tras revisar la configuración de la aplicación creada en el *Network Server* LoRaWAN integrado en el *Gateway* RAK7268, es necesario registrar el dispositivo que forma parte del nodo final IoT y revisar los parámetros de configuración, así como los parámetros del enlace. En la Figura 210 se presenta la configuración de los parámetros LoRaWAN del dispositivo Arduino MKRWAN 1310 especificando, entre otros, la clase del dispositivo, o el método de activación OTAA.

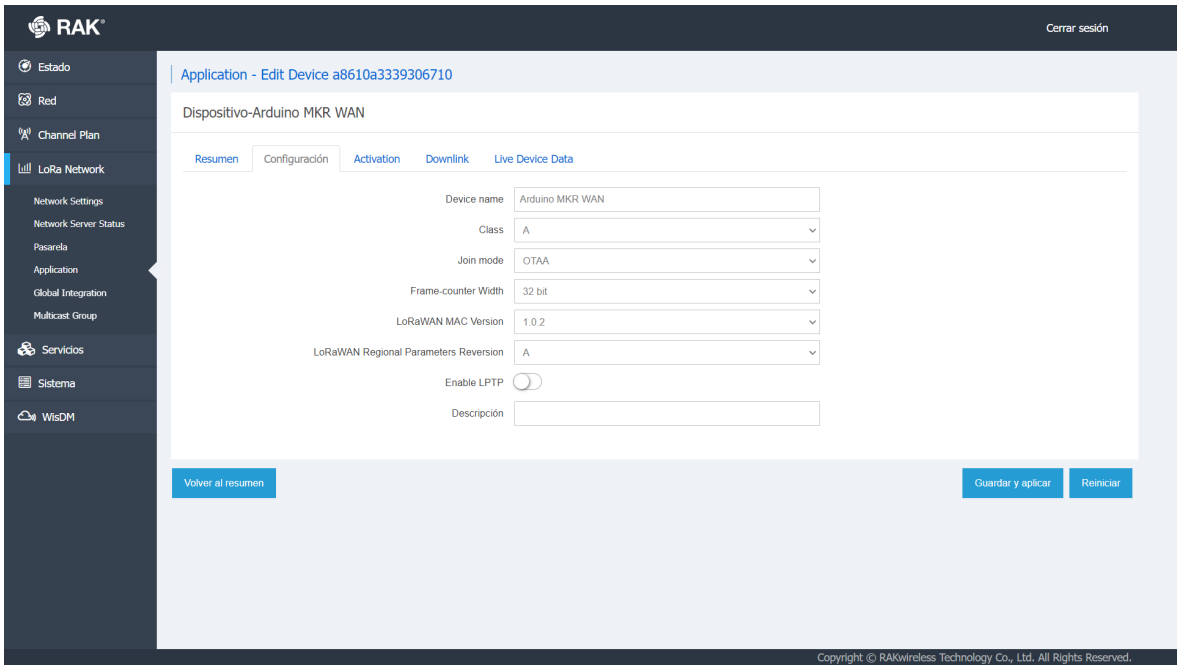


Figura 210 Configuración del Dispositivo Arduino MKRWAN 1310.

En la Figura 211 se muestra la información disponible en la sección *"Activation"* del menú de configuración del dispositivo, en la que se proporcionan las credenciales necesarias para establecer su conexión con el *Network Server* integrado en el *Gateway* RAK7268.

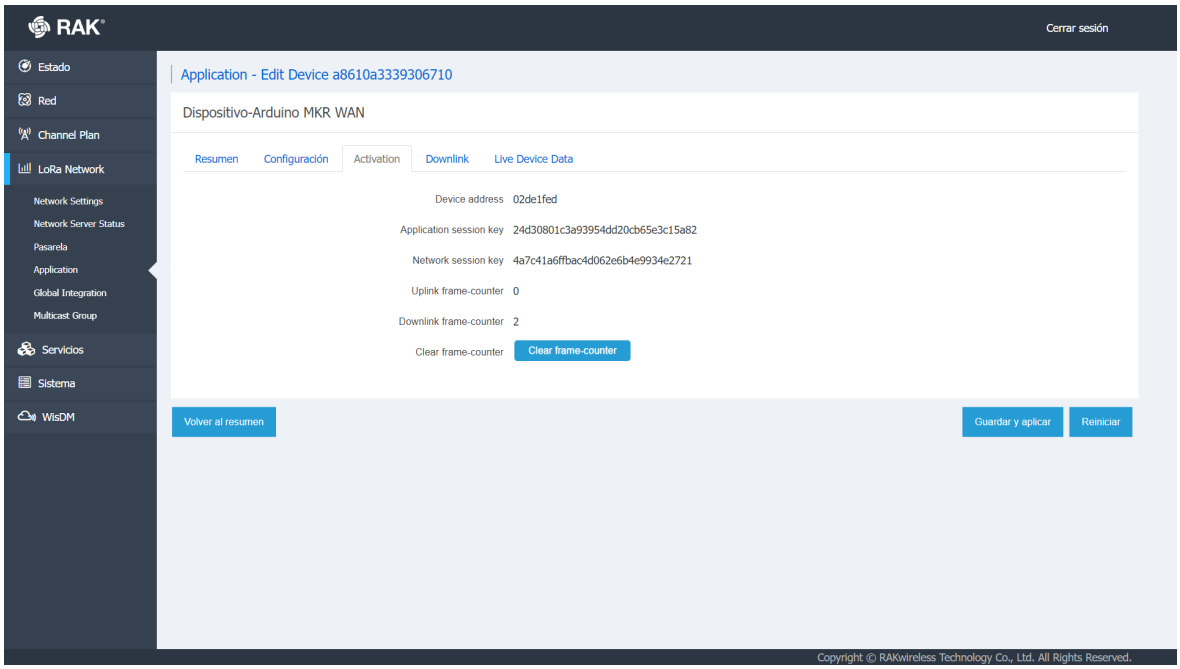


Figura 211 Parámetros de Autenticación de la aplicación Person_Sensor_RAK.

En caso de que se desee realizar una prueba funcional de la conexión configurada, en la sección *"Downlink"* de las opciones de registro del dispositivo se permite el envío de datos a través del enlace *dowlink*, como se muestra en la Figura 212.

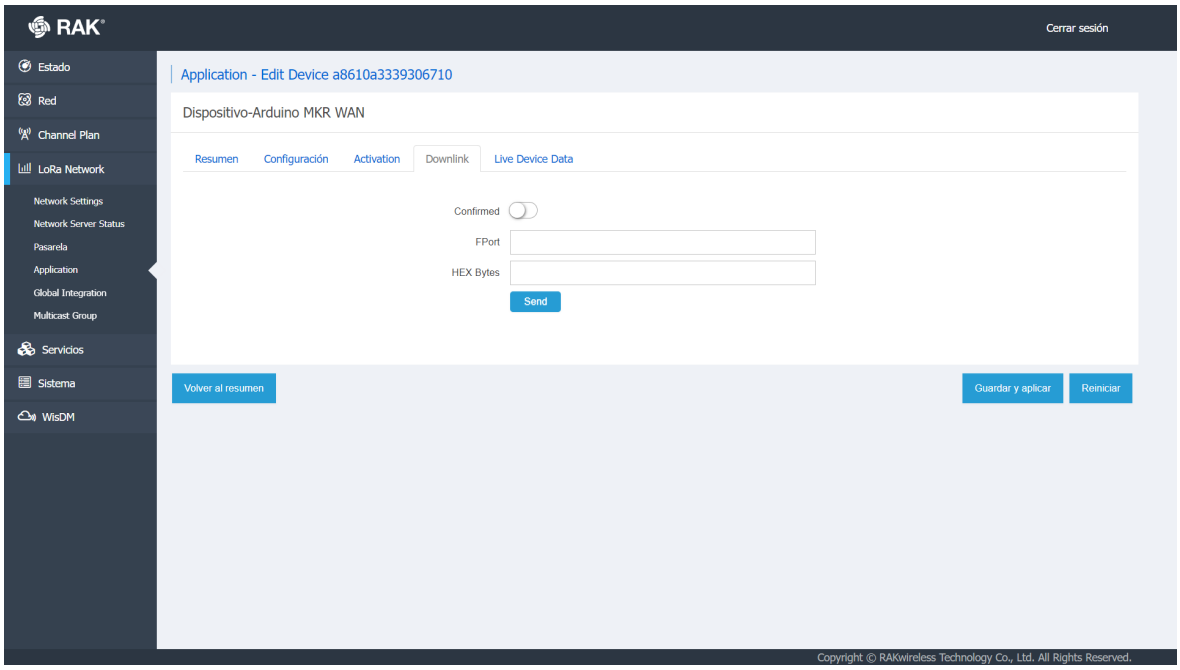


Figura 212 Simulación enlace Downlink en la aplicación Person_Sensor_RAK.

Por otro lado, en la Figura 213 se pueden apreciar, dentro de la sección *"Live Device Data"*, los datos recibidos en el *Network Server* integrado en el *Gateway* RAK7268 desde el dispositivo Arduino MKRWAN 1310, presentados en formato HexString. Este formato permite una transmisión eficiente y compacta de la información, facilitando su interpretación y procesamiento por parte del servidor. La visualización de estos datos confirma la correcta comunicación entre los dispositivos finales y la infraestructura de red LoRaWAN.

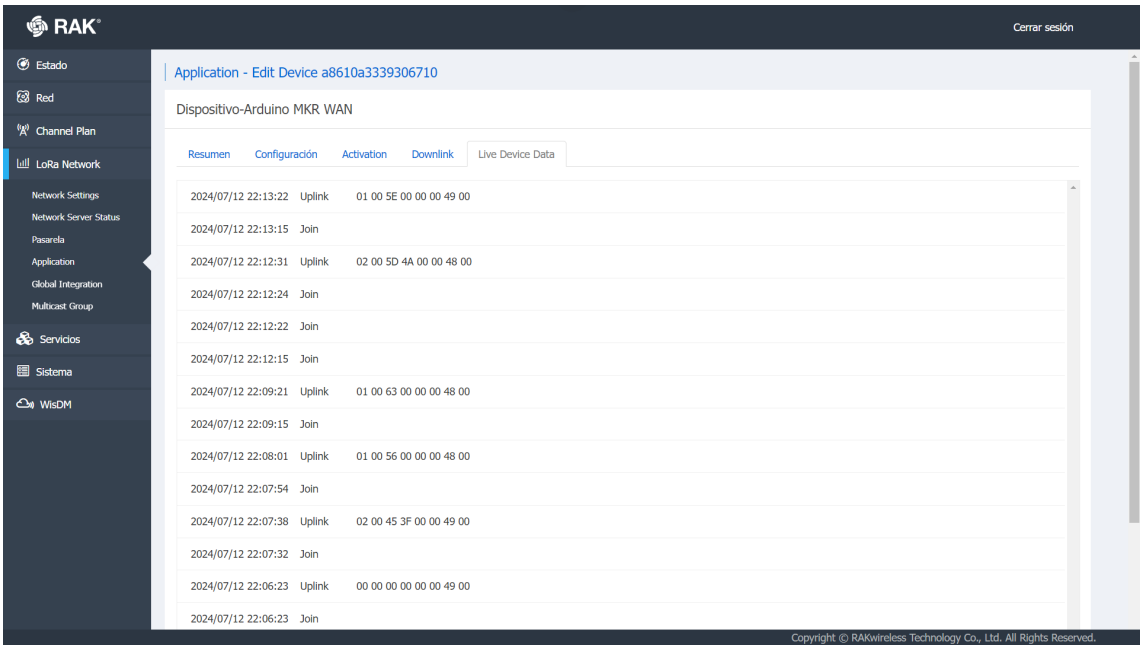


Figura 213 Live Data del Arduino MKRWAN 1310.

8.1.3 Integración del *Broker* MQTT integrado en el Gateway RAK7268

La Figura 214 y la Figura 215 muestran la sección "*Application Server Integration*" de la interfaz de configuración del *Gateway* RAK7268. En esta sección se configura la integración con un Servidor de Aplicaciones utilizando el protocolo MQTT. En este caso se selecciona el modo de integración "*Generic MQTT*" y se especifica la dirección del *Broker* MQTT como `127.0.0.1` y el puerto `1883`, lo cual indica que la dirección del *Broker* es la propia del *Gateway* RAK7268 en la red.

La versión del protocolo MQTT se establece en `V3.1`, y el campo "*Client ID*" está vacío, sugiriendo que se usará un identificador de cliente generado automáticamente, o no especificado. La opción "*Clean Session*" está activada por defecto, lo que implica que el *Broker* MQTT no mantendrá el estado de la sesión entre conexiones, mientras que la opción "*Will Retain*" está desactivada, y el nivel de calidad de servicio (QoS) está configurado en `1 - Atleast Once`, garantizando que cada mensaje se entregue al menos una vez. Finalmente, el intervalo de "*Keepalive*" está configurado en `10` segundos, especificando el tiempo para enviar mensajes ping y mantener la conexión activa.

Por otro lado, la autenticación de usuario está habilitada, siendo el nombre de usuario `rak_person_sensor` y proporcionándose una contraseña. El modo SSL/TLS está

desactivado, lo que indica que la conexión no utiliza cifrado. Adicionalmente, se especifican varios *Topics* MQTT para diferentes tipos de mensajes: "*Join Topic*" para los eventos de unión de dispositivos, "*Uplink Topic*" para los datos enviados desde el nodo final `Person_Sensor/19269144b1e5c3ca/device/a8610a3339306710/rx`, "*Downlink Topic*" para el envío de datos desde el *Network Server* integrado en el *Gateway* RAK7268 `Person_Sensor/19269144b1e5c3ca/device/a8610a3339306710/tx`, "*Ack Topic*" para las confirmaciones de tramas descendentes, "*Status Topic*" para el estado de la batería de los dispositivos, y "*Multicast Downlink Topic*" para la programación de datos de descendentes *multicast*. Todos estos ajustes resultan fundamentales para garantizar una comunicación efectiva y segura con el Servidor de Aplicaciones mediante el protocolo MQTT.

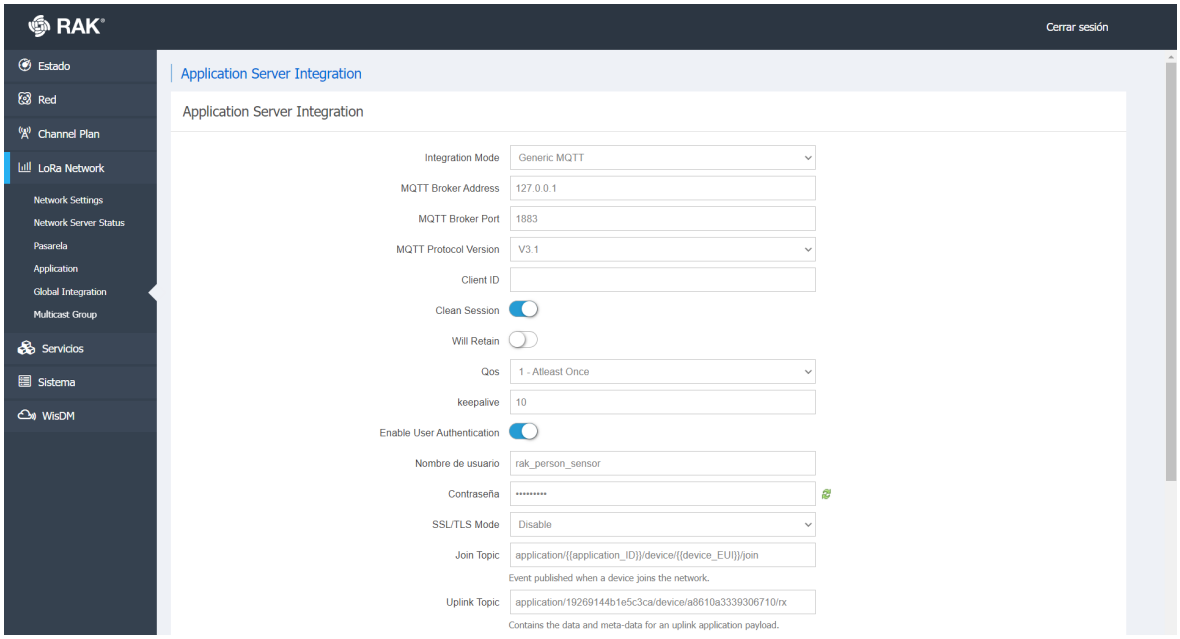


Figura 214 Sección Application Server Integration del RAK7268 (1).

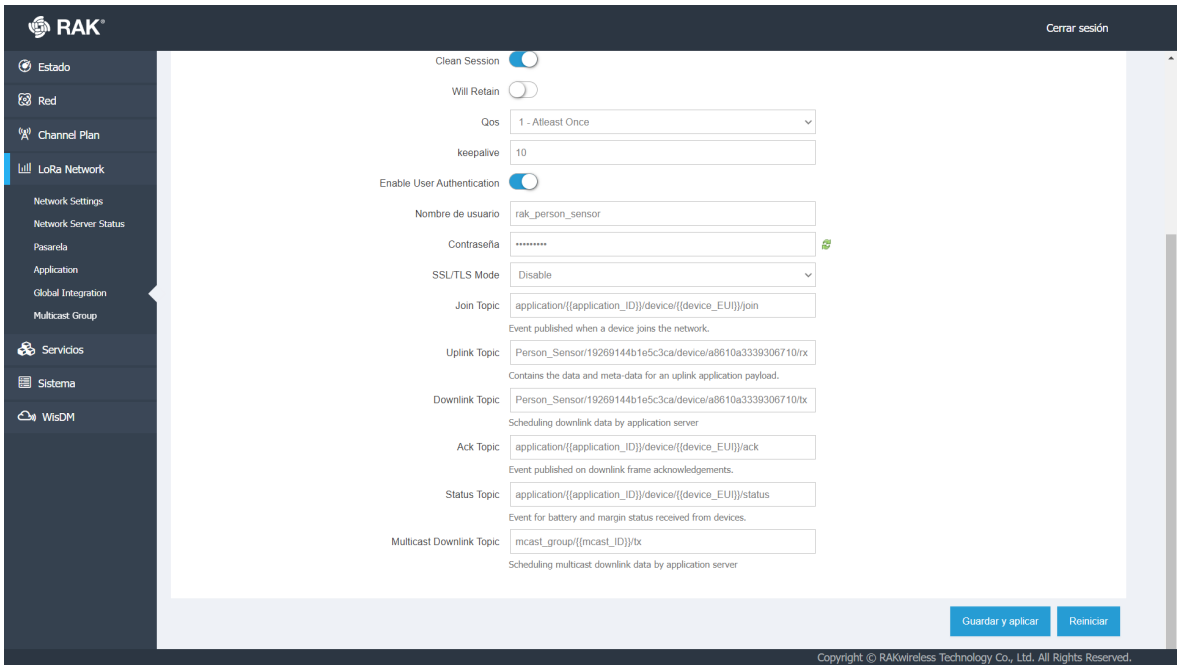


Figura 215 Sección Application Server Integration del RAK7268 (2).

8.1.4 Integración con InfluxDB

Con el fin de almacenar en InfluxDB Cloud los datos enviados desde el *Broker* MQTT asociado al *Network Server* integrado en el *Gateway* RAK7268, se optó por crear un nuevo *bucket* denominado `Person_Sensor_RAK`, como se observa en la Figura 216, con el fin de almacenar los datos enviados desde el dispositivo Arduino MKRWAN 1310 a partir de una nueva configuración del agente Telegraf.

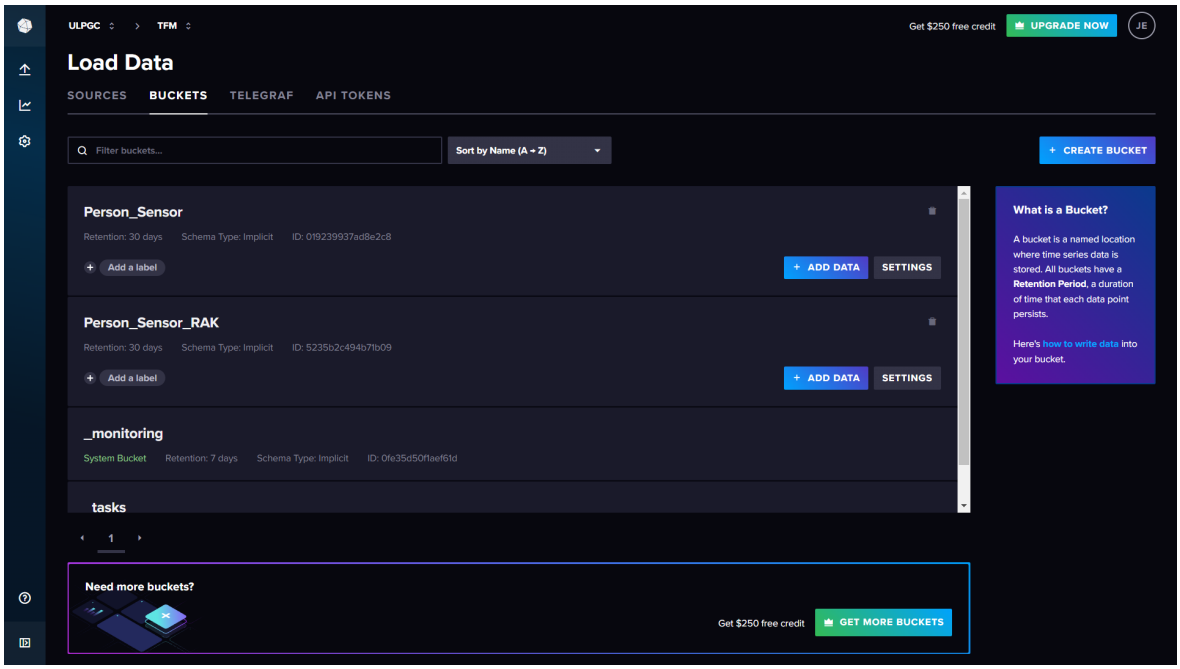


Figura 216 Buckets en InfluxDB.

Así, en la Figura 217 se puede apreciar la creación de un nuevo archivo de configuración del agente Telegraf, identificado en este caso como `RAK_INFLUXDB`, para establecer la conexión entre la información proveniente del *Broker* MQTT integrado en el *Gateway* RAK7268, y los datos enviados hacia InfluxDB Cloud.

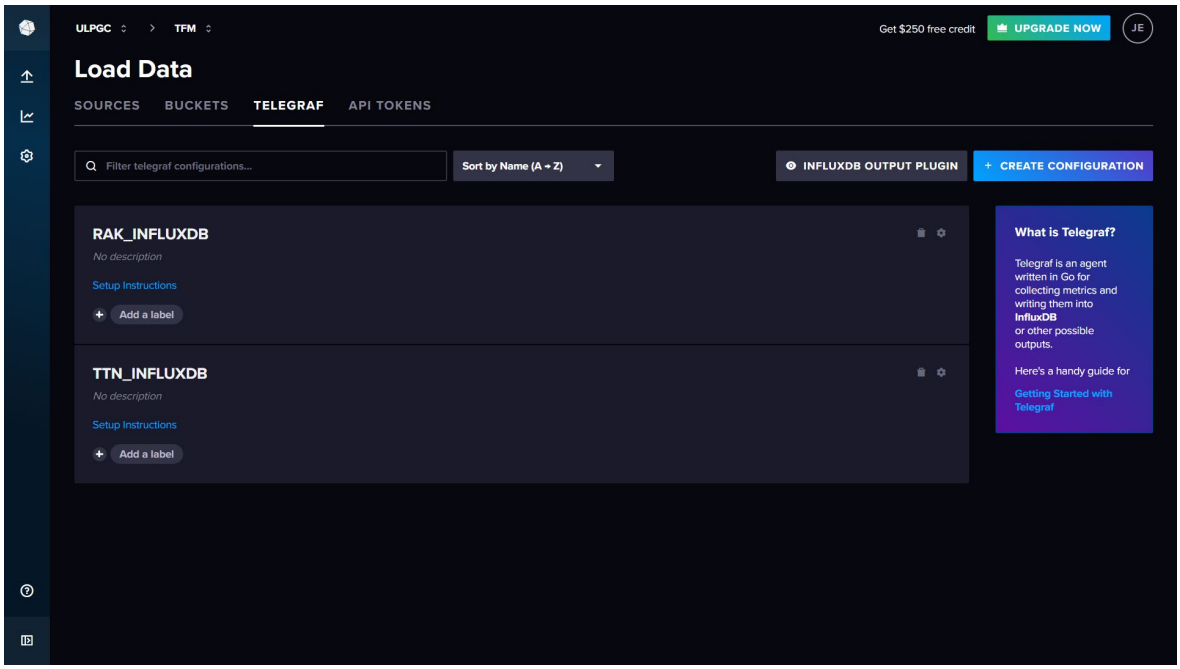


Figura 217 Archivos de configuración de Telegraf.

En la nueva configuración del agente Telegraf, el *plugin* de entrada actúa como consumidor MQTT, suscrito a un tema específico donde se reciben datos en formato JSON. Utilizando un script *Starlark* personalizado, Telegraf transformará los datos hexadecimales en estructuras legibles, extrayendo métricas como el número de caras detectadas, su orientación, y la confianza de cada detección. Finalmente, en el *plugin* de salida del agente Telegraf, los datos procesados se envían hacia el *bucket* creado en InfluxDB Cloud, asegurando una integración eficiente y escalable para su posterior análisis y visualización.

```
[[inputs.mqtt_consumer]]
  servers = ["tcp://192.168.1.97:1883"]
  topics                                                    =
["Person_Sensor/19269144b1e5c3ca/device/a8610a3339306710/rx"]
  username = "rak_person_sensor"
  password = "123456789"
  data_format = "json"
  tag_keys = ["topic"]
  json_string_fields = ["data", "data_encode", "adr"]
```

En este bloque, Telegraf se configura como un Cliente MQTT. Se especifica la dirección del *Broker* MQTT (`servers`) y el *Topic* al que se desea suscribir (`topics`). En este caso el servidor MQTT se encuentra accesible en la dirección `192.168.1.97` y utiliza el puerto `1883`. A partir de esta configuración, Telegraf espera recibir mensajes asociados al *Topic* `Person_Sensor/19269144b1e5c3ca/device/a8610a3339306710/rx`, desde el *Broker* MQTT integrado en el Gateway RAK7268.

Las credenciales de autenticación (*username* y *password*) se proporcionan para acceder al *Broker* MQTT como el usuario `rak_person_sensor`. El formato de los datos esperados (`data_format`) es JSON, lo que indica que los mensajes recibidos serán interpretados como objetos JSON para su procesamiento posterior.

Adicionalmente, se especifican las etiquetas (`tag_keys`) que se extraerán del objeto JSON para etiquetar los datos en InfluxDB Cloud a partir del campo `topic`. Finalmente, se indican los campos del objeto JSON que deben tratarse como *strings* en lugar de ser interpretados como valores numéricos durante su procesamiento (`json_string_fields`).

```
[[outputs.influxdb_v2]]
  urls = ["https://eu-central-1-1.aws.cloud2.influxdata.com"]
  token = "qA1yYSgSdicpnTbNc7yjD5vbjezolKnneCCFLQ3UA0CZv4_2O4XV0wk6bGNkqye0OET6dnl6V_pvDzi_hjXx3w=="
  organization = "TFM"
  bucket = "Person_Sensor_RAK"
```

En este bloque, se configura Telegraf para enviar los datos procesados a InfluxDB Cloud. Se especifica la URL de conexión (`urls`) del servicio de InfluxDB, que en este caso es `https://eu-central-1-1.aws.cloud2.influxdata.com`. Adicionalmente, se proporciona un *token* de autorización (`token`) para autenticar Telegraf y permitirle escribir datos procesados en el *bucket* (`bucket`) específico denominado `Person_Sensor_RAK` asociado a la organización (`organization`) denominada `TFM` dentro de InfluxDB Cloud.

```
[[processors.starlark]]
  source = '''
```

```
def hex_to_bytes(hex_string):
    bytes = []
    for i in range(0, len(hex_string), 2):
        bytes.append(int(hex_string[i:i+2], 16))
    return bytes


def apply(metric):
    if "data" not in metric.fields:
        return None
    data = metric.fields["data"]
    bytes = hex_to_bytes(data)
    decoded = {}


    num_faces = bytes[0]
    face_isfacing = []
    face_confidence = []


    for i in range(4):
        face_isfacing.append((bytes[1] >> i) & 0x01)
        face_confidence.append(bytes[i + 2])


    battery = (bytes[7] << 8) + bytes[6]


    decoded["num_faces"] = num_faces
```

```
    decoded["face_isfacing_1"] = face_isfacing[0]
    decoded["face_isfacing_2"] = face_isfacing[1]
    decoded["face_isfacing_3"] = face_isfacing[2]
    decoded["face_isfacing_4"] = face_isfacing[3]
    decoded["face_confidence_1"] = face_confidence[0]
    decoded["face_confidence_2"] = face_confidence[1]
    decoded["face_confidence_3"] = face_confidence[2]
    decoded["face_confidence_4"] = face_confidence[3]
    decoded["battery"] = battery

    for key, value in decoded.items():
        metric.fields[key] = value
    return metric
'''
```

Por último, en este bloque de configuración del agente Telegraf se define un procesador *Starlark* que transforma los datos recibidos antes de enviarlos a InfluxDB Cloud. La función `hex_to_bytes` convierte una cadena hexadecimal (`data`) en una lista de bytes. A continuación, la función `apply` se aplica a cada una de las métricas recibidas por el agente Telegraf. La función `apply` verifica si el campo `data` está presente en la métrica recibida. Si es así, convierte la cadena hexadecimal a bytes, y se decodifica en este caso para obtener el número de rostros (`num_faces`), la dirección de las caras (`face_isfacing`), y la confianza en el reconocimiento de los rostros (`face_confidence`), además de decodificarse el valor del nivel de batería (`battery`).

Finalmente, los datos decodificados se asignan como campos de la métrica y se devuelven para ser enviados a InfluxDB Cloud.

Esta estructura configura el agente Telegraf para capturar los datos enviados desde los nodos finales de la red LoRaWAN hacia el *Network Server* integrado en el *Gateway* RAK7268, publicados a través del *Broker* MQTT que proporciona, procesarlos utilizando un *script Starlark* personalizado, y enviar la información procesada a InfluxDB Cloud para su almacenamiento y posterior análisis.

La Figura 218 muestra la ventana de comandos del servidor local en el que se está ejecutando el agente Telegraf con la opción de depuración activada (`-debug`). En esta configuración, se está utilizando un archivo de configuración específico (`name_this_configuration_otro.conf`) donde se encuentra la configuración analizada previamente. A partir de esta información se deduce que Telegraf ha cargado con éxito diversos complementos de entrada, de procesamiento y de salida, incluyendo los complementos `mqtt_consumer` e `influxdb_v2`. El agente ha iniciado correctamente estos *plugins* y ha comenzado a conectar las salidas. Se observa además que las métricas se están enviando correctamente hacia `influxdb_v2`, con registros detallados de la escritura de lotes de métricas en tiempos específicos, junto con el estado del buffer de métricas, que permanece vacío. Este registro de actividad demuestra que la configuración del agente Telegraf está funcionando según lo esperado, transfiriéndose los datos adecuadamente desde el consumidor MQTT, hacia la base de datos InfluxDB Cloud.

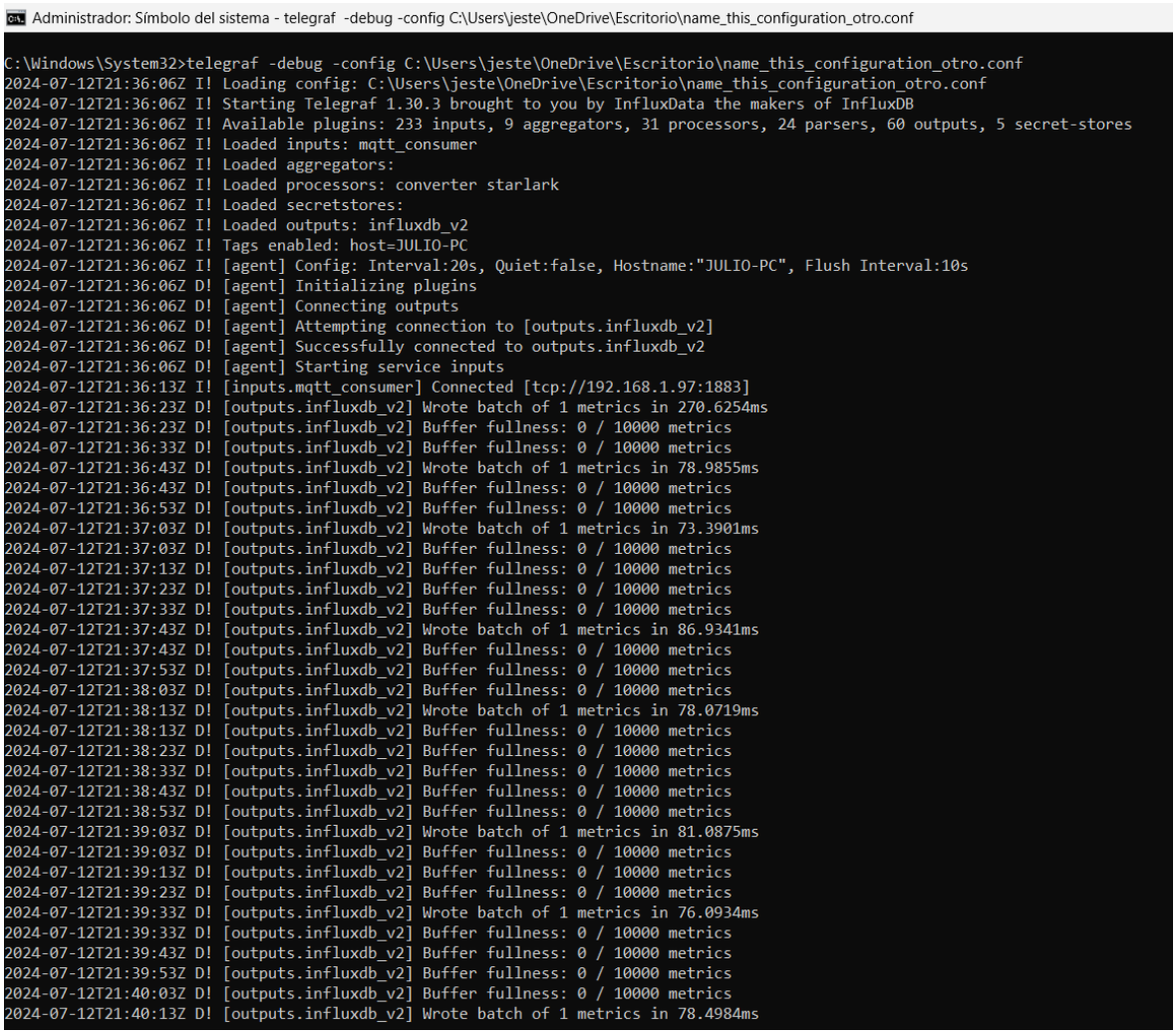


Figura 218 Ventana de comandos con Telegraf en funcionamiento.

En la Figura 219 se pueden observar los datos recopilados en el *bucket* `Person_Sensor_RAK` creado en InfluxDB Cloud. Sin embargo, al igual que en la primera de las propuestas presentadas en este documento, se consideró conveniente integrar en la aplicación IoT la herramienta Grafana para lograr una representación gráfica óptima de los datos, a partir de visualizaciones más intuitivas y comprensibles.

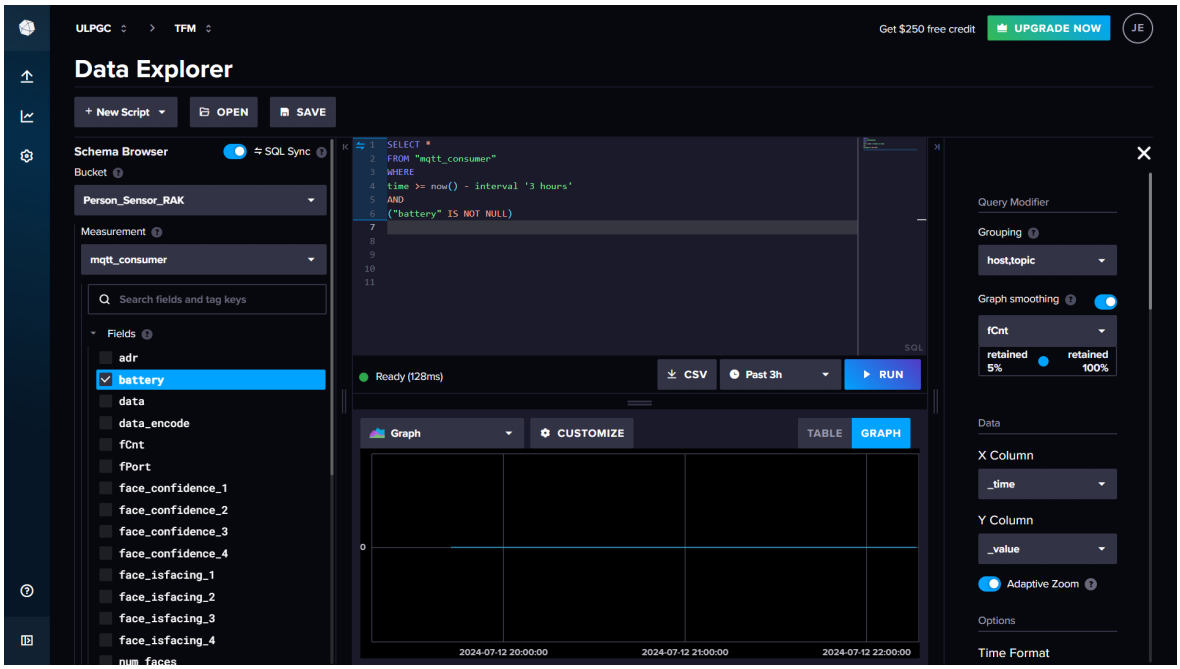


Figura 219 Datos representados en InfluxDB.

8.1.5 Integración con Grafana

Para lograr una mejor visualización de los datos, en esta segunda propuesta de aplicación IoT se utilizará de nuevo la herramienta Grafana. Inicialmente, se debe añadir una nueva fuente de datos desde InfluxDB Cloud, como se observa en la Figura 220. Para ello, se sigue el mismo procedimiento descrito en epígrafes anteriores, lo que incluye la creación de un nuevo API TOKEN en InfluxDB Cloud con control total sobre el *bucket* `Person_Sensor_RAK` utilizado. Este proceso asegura que Grafana tenga acceso completo a los datos almacenados en InfluxDB Cloud, permitiendo una representación clara y detallada de la información recopilada.

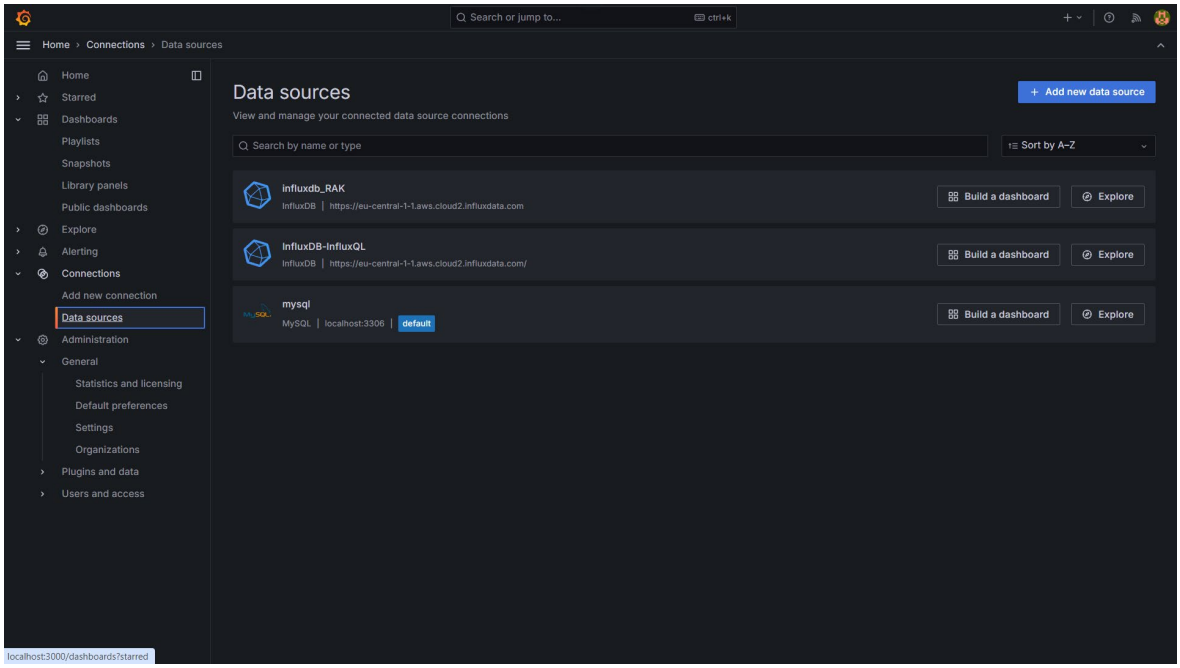


Figura 220 Nueva entrada de datos en InfluxDB.

Además, se ha desarrollado un nuevo *dashboard* en Grafana, denominado `TFM_RAK`, similar al presentado en la propuesta inicial.

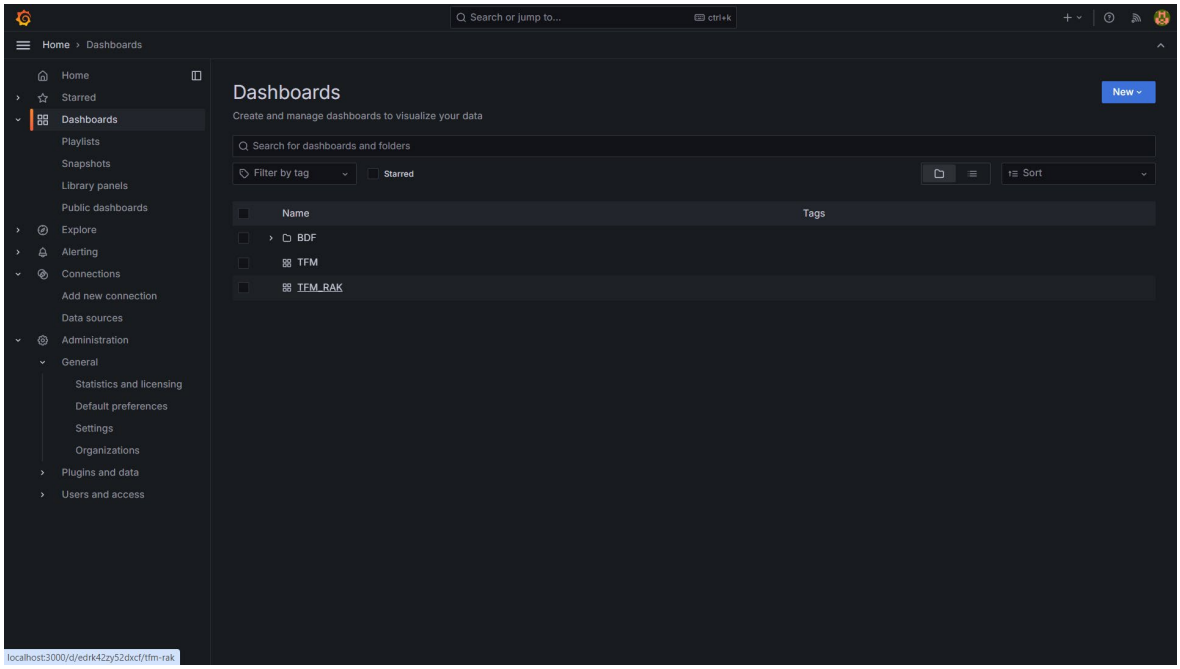


Figura 221 Creación del Dashboard TFM_RAK.

Finalmente, se han creado diferentes visualizaciones correspondientes a los datos necesarios, como se observa en la Figura 222, modificándose únicamente la entrada y los datos a representar, ya que la configuración en la herramienta Grafana se mantiene prácticamente sin cambios.

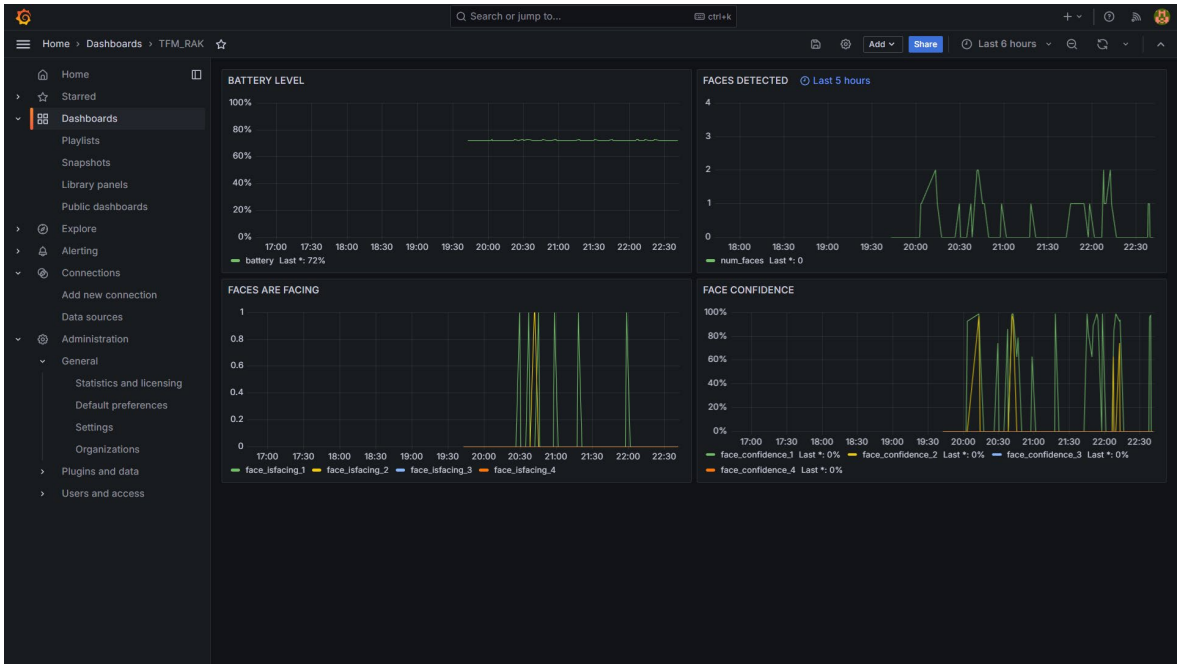


Figura 222 Representación de los datos recopilados con la segunda propuesta del TFM en Grafana.

8.1.6 Integración con el Cliente MQTTx

MQTTx es un cliente MQTT multiplataforma y de código abierto creado por EMQ [57]. En el desarrollo de esta segunda propuesta de aplicación IoT, se ha empleado como complemento para verificar la funcionalidad del *Broker* MQTT integrado en el *Gateway* RAK7268.

Diseñado para ejecutarse en macOS, Linux y Windows, MQTTx presenta una interfaz de usuario basada en chat para enviar y recibir mensajes MQTT, compatibilidad total con MQTT 5.0, 3.1.1 y 3.1, esquemas de colores personalizables para diferentes suscripciones, y soporte para autenticación SSL/TLS unidireccional y bidireccional. Además, MQTTx ofrece capacidades avanzadas, como MQTT sobre *WebSockets*, conversión de formatos de

carga útil (Hex, Base64, JSON y texto sin formato), o *scripts* personalizados para simulaciones *Pub/Sub* de MQTT [57].

Para descargar e instalar MQTTX, se disponen de varias fuentes. Se puede acceder directamente a la página de descarga en el sitio web oficial de MQTTX (`https://mqttx.app/downloads`) o en el sitio web oficial de EMQ (`https://www.emqx.com/en/try?product=MQTTX`), ofreciendo en ambos casos enlaces directos para su instalación [58].

Una vez que se accede a la vista inicial de la interfaz de MQTTx se podrán configurar los parámetros necesarios para actuar como Cliente MQTT. Entre estos se encuentra la información del *broker* MQTT, a partir de la especificación de la dirección del *Host* y el *Puerto*. Al seleccionar la opción "*Actualizar*" junto al parámetro *Client ID*, se puede generar un nuevo identificador aleatorio para el Cliente MQTT. El menú desplegable junto al *Host* permite seleccionar el protocolo de conexión, que puede ser `mqtt://` o `ws://`. Es importante tener en cuenta que, al cambiar el protocolo, también se debe ajustar el puerto de conexión correspondiente [59].

Como se observa en la Figura 223, en este caso se establece el nombre del Cliente y el *Username* y *Password* configurados previamente en el *Broker* MQTT integrado en el *Gateway* RAK7268.

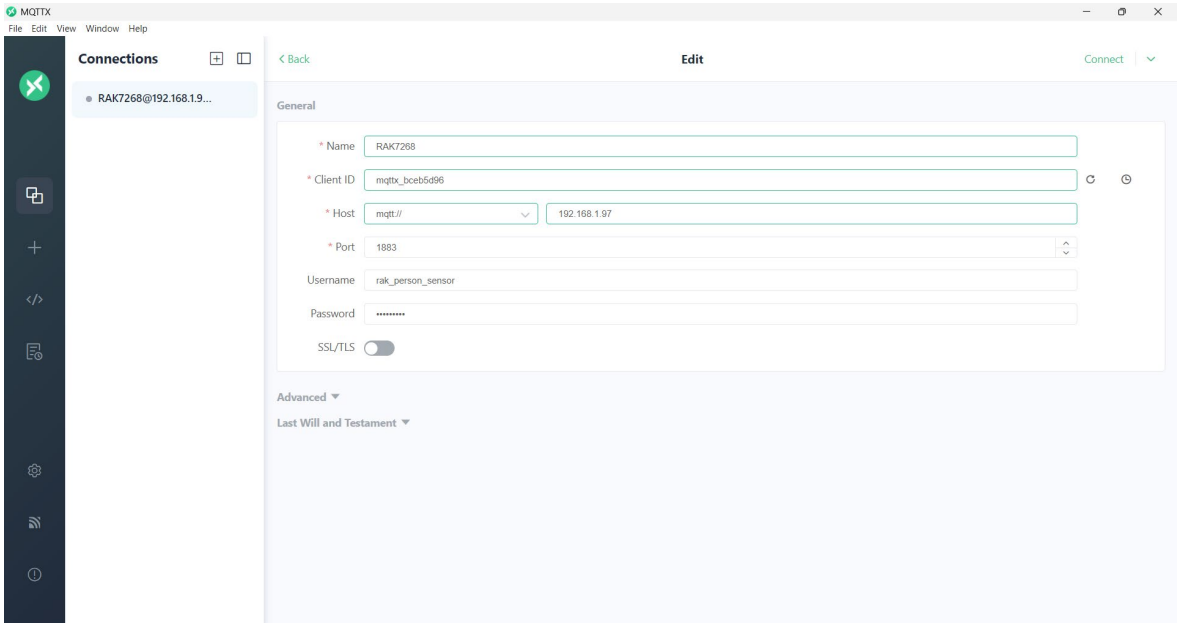


Figura 223 Configuración Cliente MQTT.

Para agregar la suscripción un nuevo *Topic*, se selecciona la opción "*New Subscription*" en la esquina inferior izquierda. Cada *Topic* puede ser etiquetado con un color, el cual puede generarse aleatoriamente o bien seleccionarse mediante el selector de colores. Una vez agregado, al hacer clic en los elementos de los temas suscritos en la lista, se filtrarán los mensajes correspondientes. La vista de mensajes mostrará únicamente el contenido relacionado con el *Topic* actualmente seleccionado. Para cancelar el filtro, simplemente es necesario hacer clic nuevamente en el *Topic* suscrito, o en otros elementos, para ver mensajes adicionales. Además, al hacer clic en el nombre del *Topic*, se podrá copiar rápidamente la información asociada a él. Para enviar un mensaje a este *Topic*, se puede pegar directamente en el cuadro de entrada del tema en la barra de mensajes, modificarlo según sea necesario, y completar la operación [59].

Como referencia, en la Figura 224 se pueden observar los *Topics* suscritos por el Cliente MQTTx. Por un lado, `Person_Sensor/19269144b1e5c3ca/device/a8610a3339306710/rx`, en el que se recibirán los mensajes publicados por el *Gateway* RAK7268 a través del *Broker* MQTT integrado. En este *Topic*, se espera recibir los datos enviados desde el dispositivo Arduino MKRWAN 1310 a partir de la información

proporcionada por el módulo *Person Sensor* integrado en el nodo final. Por otro lado se incluye el *Topic* `Person_Sensor/19269144b1e5c3ca/device/a8610a3339306710/tx`, que puede utilizarse para publicar mensajes desde otro dispositivo o aplicación.

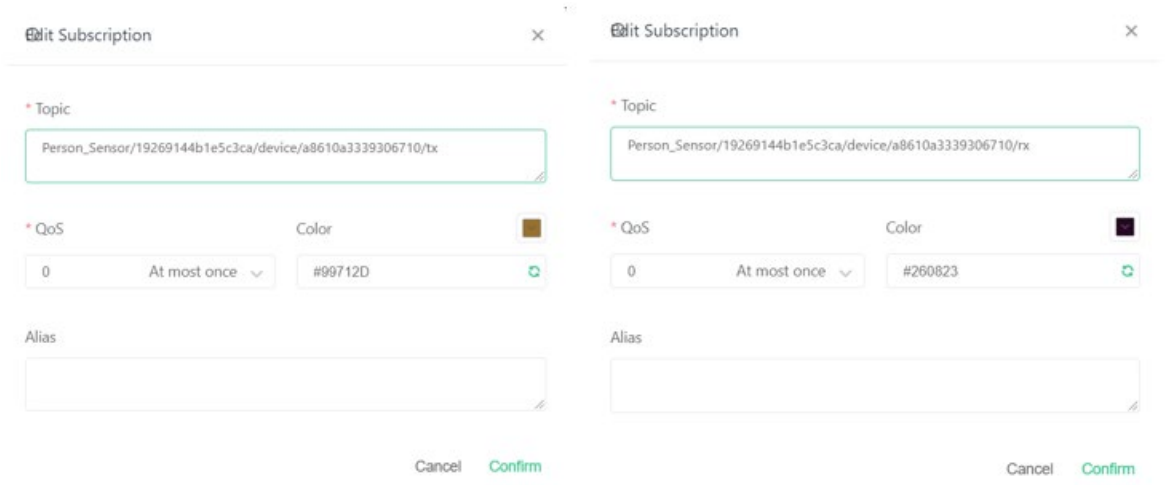


Figura 224 Topics suscritos por el Cliente MQTT.

Una vez completada la configuración de MQTTx como Cliente MQTT, en la Figura 225 se puede observar el rastreo de los mensajes publicados por el *Broker* MQTT integrado en el *Gateway* RAK7268.

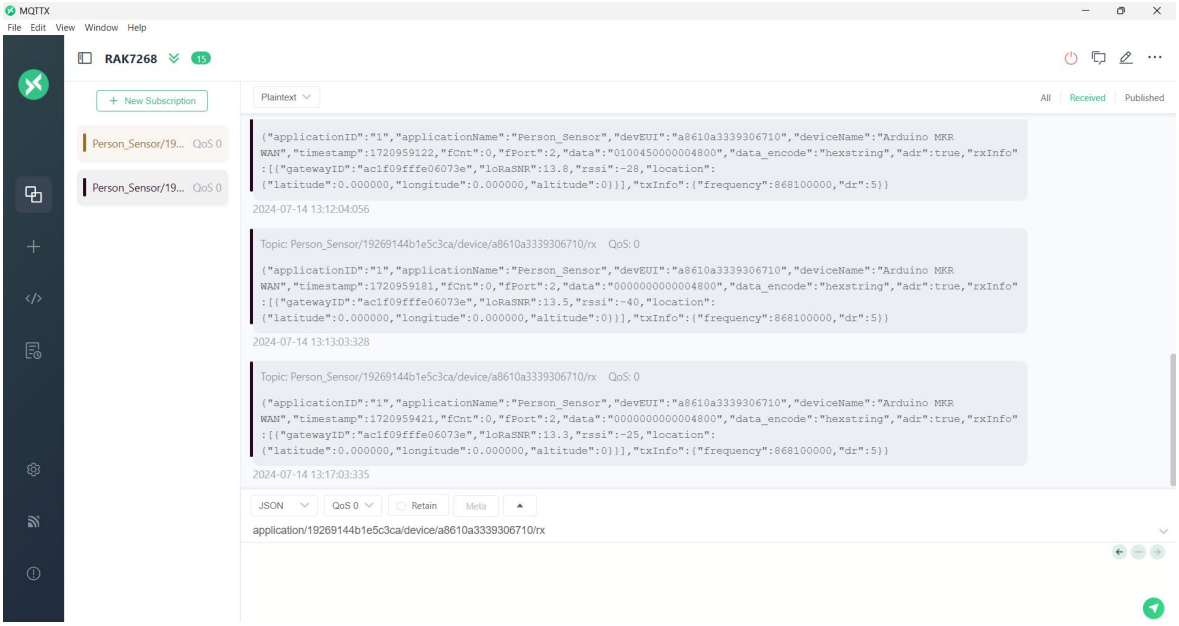


Figura 225 Mensajes recibidos en el cliente MQTT.

A modo de referencia, se realiza a continuación el análisis de uno de los mensajes recibidos a través del *Topic* MQTT `Person_Sensor/19269144b1e5c3ca/device/a8610a3339306710/rx`, representado en la Figura 226.

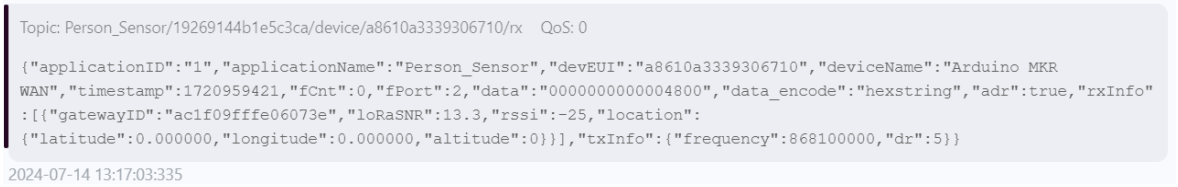


Figura 226 Mensaje recibido en el Topic: Person_Sensor/19269144b1e5c3ca/device/a8610a3339306710/rx

Este mensaje está asociado a la aplicación `"Person_Sensor"` configurada anteriormente en el *Network Server* integrado en el *Gateway* RAK7268, y en concreto al dispositivo Arduino MKRWAN 1310 registrado en ésta, identificado por el devEUI `"a8610a3339306710"`. Este mensaje, en formato JSON, incluye detalles como el ID de la aplicación (`"1"`), el nombre del dispositivo (`"Arduino MKR WAN"`), una marca de tiempo Unix (`"1720959421"`), un contador de tramas (`"fCnt": 0`), el puerto de destino (`"fPort": 2`), y los datos transmitidos en formato hexadecimal (`"data": "0000000000004800"`). La codificación de los datos se especifica como `"hexstring"`, y se confirma que la adaptación de tasa de datos (ADR) está activada (`"adr": true`).

En términos de recepción, el mensaje incluye información del *Gateway* que lo recibió en la red LoRaWAN (`"rxInfo"`), identificado por el `gatewayID` `"ac1f09fffe06073e"`, con una relación señal-ruido LoRa (`LoRaSNR`) de 13.3 dB y una intensidad de señal recibida (`RSSI`) de -25 dBm. En cuanto a los parámetros de transmisión LoRa® desde el dispositivo (`"txInfo"`), se confirma que la frecuencia de transmisión es 868100000 Hz, y el valor de DR (`dr`) es 5.

El contenido de este mensaje MQTT permite validar la correcta interacción del dispositivo Arduino MKRWAN 1310 con el *Network Server* integrado en el *Gateway* RAK7268 y el *broker* MQTT en la segunda propuesta de aplicación IoT planteada en el desarrollo del presente TFM.

9 Conclusiones y Líneas Futuras

9.1 Conclusiones

El desarrollo de la aplicación IoT basada en LoRaWAN para la detección de personas utilizando el módulo *Person Sensor* y el dispositivo Arduino MKRWAN 1310 como elementos básicos de los nodos finales, ha permitido cumplir con los objetivos establecidos inicialmente, proporcionando valiosas conclusiones en cada uno de los aspectos evaluados.

Se han detallado las características hardware del módulo *Person Sensor* y el dispositivo Arduino MKRWAN 1310, sus diferentes componentes, así como los aspectos más importantes a tener en cuenta en su programación. El módulo *Person Sensor* ha demostrado ser efectivo en la obtención de datos precisos y confiables sobre la detección de personas a partir de la identificación de rostros humanos. Por su parte, el dispositivo Arduino MKRWAN 1310 ha facilitado la transmisión de estos datos a través de una red LoRaWAN, destacando por su compatibilidad y bajo consumo energético. En este sentido, se ha logrado obtener consumos mínimos inferiores a los especificados por el fabricante, permitiendo prolongar considerablemente la vida de la batería en futuras aplicaciones mediante ajustes en la programación y la configuración del hardware.

El análisis de la tecnología LoRa® y el protocolo LoRaWAN ha abarcado los temas más relevantes para una mayor comprensión de la aplicación IoT desarrollada. El uso de *Gateways* con distintas prestaciones, en concreto las puertas de enlace Heltec HT-M00 y RAK7268 de *RAK Wireless*, ha permito desarrollar diferentes alternativas a la implementación de una red LoRaWAN, proponiéndose plataformas basadas en distintas arquitecturas, que pueden adaptarse a diferentes requisitos y costes, considerando sus limitaciones y ventajas.

Se ha utilizado la plataforma IoT TTN para la implementación de la aplicación en la red LoRaWAN, detallando y explicando cada paso realizado para contribuir a su creación. Por orto parte, el uso de InfluxDB Cloud ha demostrado ser efectivo para el almacenamiento

de los datos asociados a la aplicación IoT desarrollada, así como su integración con diferentes fuentes de datos mediante el uso del protocolo MQTT. El análisis de los datos obtenidos permitió identificar patrones de comportamiento relevantes, utilizando técnicas de procesamiento de series temporales y herramientas de visualización como Grafana.

En este sentido, Grafana ha demostrado un alto potencial en la visualización de datos, permitiendo la creación de *dashboards* personalizados de manera intuitiva, con la posibilidad de mejorar y añadir métricas de análisis para estudiar el comportamiento de los rostros detectados.

Sin embargo, se han identificado desafíos como la necesidad de una gestión adecuada de la energía y la optimización del código para asegurar la eficiencia del sistema, debido a la limitada memoria del dispositivo Arduino MKRWAN 1310, que puede restringir la complejidad de los algoritmos de procesamiento en el dispositivo.

Las pruebas experimentales de validación confirmaron que las soluciones desarrolladas cumplen con los requisitos funcionales y de rendimiento esperados. La aplicación IoT mostró una alta fiabilidad en la detección de personas y la transmisión de datos. Los sistemas basados en la nube demostraron una elevada capacidad para el procesamiento y almacenamiento de datos a gran escala, facilitando el análisis en tiempo real y la visualización gráfica de la información.

Dentro del marco del desarrollo del presente Trabajo Fin de Máster, se han planteado dos propuestas para la implementación de una aplicación IoT basada en el protocolo LoRaWAN que permita realizar el análisis de patrones de comportamiento relacionados con la detección de rostros de personas. Cada propuesta aborda consideraciones específicas, desde la escalabilidad y eficiencia energética, hasta la privacidad y seguridad de los datos. El propósito del análisis de estos dos escenarios es evaluar los recursos fundamentales para su implementación, y determinar cuál se ajusta mejor a los requisitos particulares que se demanden en cada caso.

Ambas propuestas se implementaron con éxito, logrando resultados satisfactorios en la detección de personas y en la transmisión y almacenamiento de los datos. El análisis detallado de cada propuesta indicó que las dos soluciones son viables para aplicaciones

IoT específicas. La primera propuesta destaca por el uso de una red global escalable y su eficiencia en el almacenamiento de series temporales, mientras que la segunda proporciona una integración más fluida y eficiente entre el nodo final y la plataforma de almacenamiento. Ambos enfoques ofrecen ventajas significativas, y la elección entre ellos dependerá de las necesidades específicas del entorno de aplicación.

La solución desarrollada presenta un gran potencial de uso en diversos escenarios, especialmente en la monitorización y gestión remota de datos en ciudades inteligentes, gestión de recursos y automatización de edificios, o en actividades comerciales. La combinación de LoRaWAN y dispositivos de bajo consumo energético y reducido coste, como Arduino MKRWAN 1310, junto con las capacidades del módulo *Person Sensor*, ofrece una plataforma escalable y eficiente para soluciones IoT.

9.2 Líneas Futuras

Las futuras líneas de investigación y desarrollo pueden centrarse en varias áreas de interés. En primer lugar, se debe profundizar en técnicas avanzadas de gestión energética para prolongar la vida útil de los dispositivos en entornos sin acceso constante a energía. Además, es importante ampliar las capacidades del módulo *Person Sensor* para reconocer una gama más amplia de comportamientos y características individuales. También se deben desarrollar y aplicar medidas robustas para garantizar la seguridad de los datos y la privacidad de los usuarios, especialmente en aplicaciones sensibles. Es esencial explorar la integración con otros protocolos de comunicación y tecnologías emergentes en el ámbito de IoT, como LTE-M, NB-IoT o 5G, así como tecnologías de inteligencia artificial. Finalmente, se deben investigar métodos para mejorar la escalabilidad de la solución y la eficiencia en la gestión y análisis de grandes volúmenes de datos. En resumen, el proyecto ha sentado una base para el desarrollo de soluciones IoT avanzadas y eficientes, con amplias posibilidades de expansión y mejora.

10 Bibliografía

[1] L. B. Furstenau *et al.*, “Internet of things: Conceptual network structure, main challenges and future directions,” *Digital Communications and Networks*, vol. 9, no. 3, pp. 677–687, Jun. 2023, doi: 10.1016/J.DCAN.2022.04.027.

[2] L. Maziero *et al.*, “Monitoring of Electric Parameters in the Federal University of Santa Maria Using LoRaWAN Technology,” *2019 IEEE PES Conference on Innovative Smart Grid Technologies, ISGT Latin America 2019*, Sep. 2019, doi: 10.1109/ISGT-LA.2019.8895425.

[3] “What is Arduino? | Arduino Documentation.” Accessed: Oct. 02, 2023. [Online]. Available: https://docs.arduino.cc/learn/starting-guide/whats-arduino

[4] “Madrid - The Things Network Community.” Accessed: Oct. 03, 2023. [Online]. Available: https://www.thethingsnetwork.org/community/madrid/

[5] P. Dix, “Why Time Series Matters for Metrics, Real-Time Analytics and Sensor Data,” 2021. Accessed: Oct. 04, 2023. [Online]. Available: https://get.influxdata.com/rs/972-GDU-533/images/why%20time%20series.pdf

[6] M. Lombardi, F. Pascale, and D. Santaniello, “Internet of Things: A General Overview between Architectures, Protocols and Applications,” *Information 2021, Vol. 12, Page 87*, vol. 12, no. 2, p. 87, Feb. 2021, doi: 10.3390/INFO12020087.

[7] “Number of connected IoT devices growing 16% to 16.7 billion globally.” Accessed: Oct. 31, 2023. [Online]. Available: https://iot-analytics.com/number-connected-iot-devices/

[8] “Book - LoRaWAN.” Accessed: Mar. 03, 2024. [Online]. Available: https://www.univ-smb.fr/lorawan/en/free-book/

[9] “Wi-Fi Alliance® celebrates 25 years of Wi-Fi® innovation and impact | Wi-Fi Alliance.” Accessed: Mar. 02, 2024. [Online]. Available: https://www.wi-fi.org/news-events/newsroom/wi-fi-alliance-celebrates-25-years-of-wi-fi-innovation-and-impact

[10] “Internet of Things | Wi-Fi Alliance.” Accessed: Mar. 02, 2024. [Online]. Available: https://www.wi-fi.org/discover-wi-fi/internet-things

[11] “SIMs NB-IoT y LTE-M para el IoT.” Accessed: Mar. 03, 2024. [Online]. Available: https://es.whereversim.de/lte-m-m2m-sim

[12] “Future of Massive IoT and the Role of LoRaWAN.” Accessed: Mar. 02, 2024. [Online]. Available: https://www.brighttalk.com/webcast/5567/549664

[13] “Conectando al Planeta de Forma Inteligente con LoRaWAN - LoRa Alliance®.” Accessed: Mar. 03, 2024. [Online]. Available: https://lora-alliance.org/lorawan-news/conectando-al-planeta-de-forma-inteligente-con-lorawan/

[14] H. U. Rahman, M. Ahmad, H. Ahmad, and M. A. Habib, “LoRaWAN: State of the Art, Challenges, Protocols and Research Issues,” *Proceedings - 2020 23rd IEEE International Multi-Topic Conference, INMIC 2020*, Nov. 2020, doi: 10.1109/INMIC50486.2020.9318170.

[15] “LoRa Alliance® Issues 2023 Annual Report Highlighting LoRaWAN® Maturity, Robust Adoption, and Diversity of End-to-End Solutions - LoRa Alliance®.” Accessed: Mar. 02, 2024. [Online]. Available: https://lora-alliance.org/lora-alliance-press-release/lora-alliance-issues-2023-annual-report-highlighting-lorawan-maturity-robust-adoption-and-diversity-of-end-to-end-solutions/

[16] “What are LoRa and LoRaWAN? | The Things Network.” Accessed: Oct. 02, 2023. [Online]. Available: https://www.thethingsnetwork.org/docs/lorawan/what-is-lorawan/

[17] “Spreading Factors | The Things Network.” Accessed: Oct. 02, 2023. [Online]. Available: https://www.thethingsnetwork.org/docs/lorawan/spreading-factors/

[18] “LoRaWAN® Specification v1.1.” Accessed: Oct. 24, 2023. [Online]. Available: https://resources.lora-alliance.org/technical-specifications/lorawan-specification-v1-1

[19] “LoRaWAN Architecture | The Things Network.” Accessed: Oct. 02, 2023. [Online]. Available: https://www.thethingsnetwork.org/docs/lorawan/architecture/

[20] “Regional Parameters | The Things Network.” Accessed: Oct. 02, 2023. [Online]. Available: https://www.thethingsnetwork.org/docs/lorawan/regional-parameters/

[21] “Device Classes | The Things Network.” Accessed: Oct. 02, 2023. [Online]. Available: https://www.thethingsnetwork.org/docs/lorawan/classes/

[22] “End Device Activation | The Things Network.” Accessed: May 20, 2024. [Online]. Available: https://www.thethingsnetwork.org/docs/lorawan/end-device-activation/

[23] “1 - End Device Activation: Introduction.” Accessed: May 21, 2024. [Online]. Available: https://learn.semtech.com/mod/book/view.php?id=171&chapterid=102

[24] “1 - End Device Activation: End Device Activation Best Practices.” Accessed: May 25, 2024. [Online]. Available: https://learn.semtech.com/mod/book/view.php?id=171&chapterid=101

[25] “Product Datasheet.”

[26] “MKR WAN 1310 | Arduino Documentation.” Accessed: Sep. 21, 2023. [Online]. Available: https://docs.arduino.cc/hardware/mkr-wan-1310

[27] “Useful Sensors – AI you can trust.” Accessed: Sep. 30, 2023. [Online]. Available: https://usefulsensors.com/#products

[28] “person_sensor_docs/README.md at main · usefulsensors/person_sensor_docs · GitHub.” Accessed: Sep. 30, 2023. [Online]. Available: https://github.com/usefulsensors/person_sensor_docs/blob/main/README.md

[29] “GitHub - usefulsensors/person_sensor_arduino: How to use a Person Sensor from an Arduino.” Accessed: Oct. 25, 2023. [Online]. Available: https://github.com/usefulsensors/person_sensor_arduino

[30] “GitHub - adafruit/Adafruit_ILI9341: Library for Adafruit ILI9341 displays.” Accessed: Oct. 25, 2023. [Online]. Available: https://github.com/adafruit/Adafruit_ILI9341

[31] “Powering Alternatives for Arduino Boards | Arduino Documentation.” Accessed: Oct. 04, 2023. [Online]. Available: https://docs.arduino.cc/learn/electronics/power-pins

[32] “Arduino Low Power - Arduino Reference.” Accessed: Oct. 04, 2023. [Online]. Available: https://www.arduino.cc/reference/en/libraries/arduino-low-power/

[33] “RTCZero - Arduino Reference.” Accessed: Oct. 04, 2023. [Online]. Available: https://www.arduino.cc/reference/en/libraries/rtczero/

[34] “How can I achieve the 104µA power consumption? - MKR Family / MKRWAN1310 - Arduino Forum.” Accessed: Sep. 21, 2023. [Online]. Available: https://forum.arduino.cc/t/how-can-i-achieve-the-104-a-power-consumption/619331

[35] “MKR WAN 1310 | Arduino Documentation | Arduino Documentation.” Accessed: Feb. 15, 2023. [Online]. Available: https://docs.arduino.cc/hardware/mkr-wan-1310

[36] “The Things Network? - Post - The Things Network.” Accessed: Oct. 03, 2023. [Online]. Available: https://www.thethingsnetwork.org/community/barranquilla/post/the-things-network

[37] “Migrating devices from The Things Network V2 to The Things Stack.” Accessed: Oct. 03, 2023. [Online]. Available: https://www.thethingsnetwork.org/article/migrating-devices-from-the-things-network-v2-to-the-things-stack

[38] “The Things Stack | The Things Stack for LoRaWAN.” Accessed: Oct. 03, 2023. [Online]. Available: https://www.thethingsindustries.com/docs/getting-started/the-things-stack-basics/

[39] “HT-M00 Dual Channel LoRa Gateway — ht-m00 latest documentation.” Accessed: Jun. 08, 2023. [Online]. Available: https://docs.heltec.org/en/gateway/ht-m00/index.html

[40] “HT-M00 Dual Channel LoRa Gateway Quick Start — ht-m00 latest documentation.” Accessed: Oct. 03, 2023. [Online]. Available: https://docs.heltec.org/en/gateway/ht-m00/quick_start.html

[41] “The Things Network.” Accessed: May 31, 2024. [Online]. Available: https://www.thethingsnetwork.org/

[42] “Hex to ASCII Text String Converter.” Accessed: Apr. 06, 2024. [Online]. Available: https://www.rapidtables.com/convert/number/hex-to-ascii.html

[43] “RAK7268 WisGate Edge Lite 2 | RAKwireless Documentation Center.” Accessed: Oct. 03, 2023. [Online]. Available: https://docs.rakwireless.com/Product-Categories/WisGate/RAK7268/Overview/#product-description

[44] “Regional Limitations of RF Use in LoRaWAN | The Things Network.” Accessed: Jun. 02, 2024. [Online]. Available: https://www.thethingsnetwork.org/docs/lorawan/regional-limitations-of-rf-use/

[45] “Adaptive Data Rate | The Things Network.” Accessed: Jun. 02, 2024. [Online]. Available: https://www.thethingsnetwork.org/docs/lorawan/adaptive-data-rate/

[46] “Telegraf | InfluxData.” Accessed: Jun. 19, 2024. [Online]. Available: https://www.influxdata.com/time-series-platform/telegraf/

[47] “InfluxDB Time Series Data Platform | InfluxData.” Accessed: Jun. 19, 2024. [Online]. Available: https://www.influxdata.com/?_gl=1*kyi7w5*_ga*OTU5OTQxMjQ0LjE2OTY0MTE4NDM.*_ga_CNWQ54SDD8*MTcxODc4OTk1MC40OS4xLjE3MTg3OTA1NDcuMTguMC4xNTEwMzkyMDgw*_gcl_au*MTQxODE5MjQ1MS4xNzE3ODc0MDI3

[48] “Beginners Guide To The MQTT Protocol.” Accessed: Oct. 02, 2023. [Online]. Available: http://www.steves-internet-guide.com/mqtt/

[49] “MQTT Tutorial: An Easy Guide to Getting Started with MQTT.” Accessed: Oct. 02, 2023. [Online]. Available: https://www.hivemq.com/article/how-to-get-started-with-mqtt/

[50] “MQTT Client, MQTT Broker, and MQTT Server Connection Establishment Explained – MQTT Essentials: Part 3.” Accessed: Oct. 25, 2023. [Online]. Available: https://www.hivemq.com/blog/mqtt-essentials-part-3-client-broker-connection-establishment/

[51] E. Salituro, “Learn Grafana 10.x : a beginner’s guide to practical data analytics, interactive dashboards, and observability,” p. 542.

[52] “Install Grafana on Windows | Grafana documentation.” Accessed: Jun. 21, 2024. [Online]. Available: https://grafana.com/docs/grafana/latest/setup-grafana/installation/windows/

[53] “Download Grafana | Grafana Labs.” Accessed: Jun. 21, 2024. [Online]. Available: https://grafana.com/grafana/download?platform=windows

[54] “Dashboards | Grafana documentation.” Accessed: Jun. 23, 2024. [Online]. Available: https://grafana.com/docs/grafana/latest/dashboards/

[55] “Panels and visualizations | Grafana documentation.” Accessed: Jun. 23, 2024. [Online]. Available: https://grafana.com/docs/grafana/latest/panels-visualizations/

[56] “Visualizations | Grafana documentation.” Accessed: Jun. 23, 2024. [Online]. Available: https://grafana.com/docs/grafana/latest/panels-visualizations/visualizations/

[57] “Introduction - MQTTX Documentation.” Accessed: Jul. 14, 2024. [Online]. Available: https://mqttx.app/docs

[58] “Installation - MQTTX Documentation.” Accessed: Jul. 14, 2024. [Online]. Available: https://mqttx.app/docs/downloading-and-installation

[59] “Get Started - MQTTX Documentation.” Accessed: Jul. 14, 2024. [Online]. Available: https://mqttx.app/docs/get-started

[60] “Agencia Tributaria: Amortizaciones.” Accessed: Jul. 03, 2024. [Online]. Available: https://sede.agenciatributaria.gob.es/Sede/impuesto-sobre-sociedades/que-base-imponible-se-determina-sociedades/amortizaciones.html?faqId=42c3904421205710VgnVCM100000dc381e0aRCRD

[61] “Person Sensor by Useful Sensors - SEN-21231 - SparkFun Electronics.” Accessed: Jul. 03, 2024. [Online]. Available: https://www.sparkfun.com/products/21231

[62] “Arduino MKR WAN 1310 — Arduino Official Store.” Accessed: Jul. 03, 2024. [Online]. Available: https://store.arduino.cc/products/arduino-mkr-wan-1310

[63] “AOLIKES Reemplazo de batería Lipo recargable de 3.7 V 1400 mAh para cámara de tubo prisma Sena 50R auriculares Bluetooth, paquete de batería de 3 cables : Amazon.es: Electrónica.” Accessed: Jul. 03, 2024. [Online]. Available: https://www.amazon.es/

[64] “HT-M00 Dual Channel LoRa Gateway – Heltec Automation.” Accessed: Jul. 03, 2024. [Online]. Available: https://heltec.org/project/ht-m00/

[65] “Indoor LoRaWAN gateway - 8 Channel SX1302 LoRa Gateway.” Accessed: Jul. 03, 2024. [Online]. Available: https://store.rakwireless.com/products/rak7268-8-channel-indoor-lorawan-gateway?utm_campaign=RAKwireless&utm_medium=Header&utm_source=RAK7268&variant=42316476678342

[66] “Asus TUF Dash F15 FX517ZM - Ordenador Portátil Gaming de 15.6" Full HD 144Hz (Intel Core i7-12650H, 16GB RAM, 512GB SSD, NVIDIA RTX 3060-6GB, Sin Sistema

Operativo) Negro - Teclado QWERTY español : Amazon.es: Informática.” Accessed: Jul. 03, 2024. [Online]. Available: https://www.amazon.es/

[67] “Comprar Windows 11 Home - Microsoft Store es-ES.” Accessed: Jul. 03, 2024. [Online]. Available: https://www.microsoft.com/es-es/d/windows-11-home/dg7gmgf0krt0/000P

[68] “Compara todos los planes de Microsoft 365 (anteriormente Office 365): Microsoft Store.” Accessed: Jul. 03, 2024. [Online]. Available: https://www.microsoft.com/es-es/microsoft-365/buy/compare-all-microsoft-365-products

[69] “2023_retribuciones_pi_contratado_ulpgc”.

[70] “MQTT Publish, MQTT Subscribe & Unsubscribe – MQTT Essentials: Part 4.” Accessed: Oct. 25, 2023. [Online]. Available: https://www.hivemq.com/blog/mqtt-essentials-part-4-mqtt-publish-subscribe-unsubscribe/

11 Presupuesto

En esta sección del documento se desglosa el coste económico asociado al desarrollo del presente TFM. Los aspectos del trabajo que han contribuido a dicho coste incluyen los recursos hardware, recursos software, recursos humanos, y la redacción del documento.

11.1 Recursos Hardware

La Tabla 25 recoge los recursos hardware utilizados y su vida útil. Por su parte, en la Tabla 26 se muestra el coste de los equipos hardware en el momento de la adquisición, su amortización anual, el porcentaje de amortización de los elementos calculado a partir de la tabla de Amortizaciones de la Agencia Tributaria [60], y el coste final de cada dispositivo.

Tabla 25 Recursos Hardware y su vida útil.

Recursos Hardware	**Vida útil**
Person Sensor	5 años
Arduino MKRWAN 1310	5 años
Batería modelo BAT536 M6 Li-Po de 1400mAh de 3.7V	2 años
Gateway Heltec HT-M00	5 años
Gateway RAK7268	5años
Elementos de cableado e interconexión	2 años
Ordenador ASUS TUF Dash F15	5 años

Tabla 26 Coste de Recursos Hardware.

	Recursos Hardware						
Descripción	Tiempo	Valor de Adquisición	Amortización anual	Depreciación Anual	Depreciación Mensual	Depreciación Total (4 meses)	Coste Final
Person Sensor	4 meses	9,95 € [61]	20% [60]	1,99 €	0,16 €	0,66 €	9,28 €
Arduino MKRWAN 1310	4 meses	39,60 € [62]	20% [60]	7,92 €	0,66 €	2,64 €	36,96 €

Batería modelo BAT536 M6 Li-Po de 1400mAh de 3.7V	4 meses	23,99 € [63]	20% [60]	4,79 €	0,39 €	1,59 €	22,39€
Gateway Heltec HT-M00	4 meses	39,00 € [64]	20% [60]	7,80 €	0,65 €	2,60 €	36,40 €
Gateway RAK7268	4 meses	139,00 € [65]	20% [60]	27,80 €	2,31 €	9,26 €	129,73€
Elementos de cableado e interconexión	4 meses	50 €	7 % [60]	3,50 €	0,29 €	1,16 €	48,83 €
Ordenador ASUS TUF Dash F15	4 meses	899,00 € [66]	20% [60]	179,80 €	14,98 €	59,93 €	839,06 €
Total		1254,00 €				77,27 €	1123,27 €

- **Descripción**: El nombre del recurso hardware.
- **Tiempo**: El tiempo de uso en meses.
- **Coste Inicial**: El coste inicial del recurso.
- **Amortización Anual**: El porcentaje de amortización anual.
- **Depreciación Anual**: La cantidad que se deprecia anualmente (Coste Inicial × Amortización Anual).
- **Depreciación Mensual**: La cantidad que se deprecia mensualmente (Depreciación Anual / 12).
- **Depreciación Total** (4 meses): La cantidad total depreciada en 4 meses (Depreciación Mensual × 4).
- **Coste Final**: El coste final después de 4 meses de depreciación (Coste Inicial - Depreciación Total).
- **Coste Inicial Total**: La suma de todos los costes iniciales de los recursos.
- **Depreciación Total** (4 meses): La suma de todas las depreciaciones totales para los 4 meses.
- **Coste Final Total**: La suma de todos los costes finales después de la depreciación de 4 meses.

El coste final de los recursos hardware es de MIL CIENTO VEINTITRÉS EUROS CON VEINTISIETE CÉNTIMOS.

11.2 Recursos Software

La Tabla 27 recoge los recursos software y su vida útil. Seguidamente, en la Tabla 28 se muestra el coste de los recursos software en el momento de la adquisición, su amortización anual, y el coste final de cada uno.

Tabla 27 Recursos Software y su vida útil.

Recurso Software	**Vida útil**
IDE de Arduino V 2.3.2	5 años
Terminal Tera Term V4.106	5 años
Terminal Putty V 0.81	5 años
TTN Console Cloud V 3.31.0	5 años
InfluxDB Cloud Serverless	5 años
Telegraf V 1.30.3	5 años
Grafana Enterprise V 11.0.0	5 años
SO Microsoft Windows 11 Home 10.0.22631 N/D Compilación 22631	5 años
Office 365	1 año

Tabla 28 Coste de Recursos de Software

Recursos Software							
Descripción	Tiempo	Valor de Adquisición	Amortización anual	Depreciación Anual	Depreciación Mensual	Depreciación Total (4 meses)	Coste Final
IDE de Arduino V 2.3.2	4 meses	0,00 €	33 % [60]	0,00 €	0,00 €	0,00 €	0,00 €
Terminal Tera Term V4.106	4 meses	0,00 €	33 % [60]	0,00 €	0,00 €	0,00 €	0,00 €
Terminal Putty V 0.81	4 meses	0,00 €	33 % [60]	0,00 €	0,00 €	0,00 €	0,00 €

TTN Console Cloud V 3.31.0	4 meses	0,00 €	33 % [60]	0,00 €	0,00 €	0,00 €	0,00 €
InfluxDB Cloud Serverless	4 meses	0,00 €	33 % [60]	0,00 €	0,00 €	0,00 €	0,00 €
Telegraf V 1.30.3	4 meses	0,00 €	33 % [60]	0,00 €	0,00 €	0,00 €	0,00 €
Grafana Enterprise V 11.0.0	4 meses	0,00 €	33 % [60]	0,00 €	0,00 €	0,00 €	0,00 €
SO Microsoft Windows 11 Home 10.0.22631 N/D Compilación 22631	4 meses	145,00 € [67]	33 % [60]	47,85 €	3,98 €	15,95 €	129,05 €
Office 365	4 meses	69,00 € [68]	33 % [60]	22,77 €	1,89 €	7,59 €	61,41 €
Total		214,00 €				23,54 €	190,46 €

- **Descripción**: El nombre del recurso software.
- **Tiempo**: El tiempo de uso en meses.
- **Coste Inicial**: El coste inicial del recurso.
- **Amortización Anual**: El porcentaje de amortización anual.
- **Depreciación Anual**: La cantidad que se deprecia anualmente (Coste Inicial × Amortización Anual).
- **Depreciación Mensual**: La cantidad que se deprecia mensualmente (Depreciación Anual / 12).
- **Depreciación Total** (4 meses): La cantidad total depreciada en 4 meses (Depreciación Mensual × 4).
- **Coste Final**: El coste final después de 4 meses de depreciación (Coste Inicial - Depreciación Total).
- **Coste Inicial Total**: La suma de todos los costes iniciales de los recursos.
- **Depreciación Total** (4 meses): La suma de todas las depreciaciones totales para los 4 meses.

- **Coste Final Total**: La suma de todos los costes finales después de la depreciación de 4 meses.

El coste final de los recursos software es de CIENTO NOVENTA EUROS CON CUARENTAISÉIS CÉNTIMOS.

11.3 Recursos Humanos

Según la tabla "RETRIBUCIONES DEL PERSONAL INVESTIGADOR, DEL PERSONAL TÉCNICO Y DEL PERSONAL TÉCNICO DE APOYO CONTRATADO DE LA ULPGC CON CARGO A PROYECTOS, CONVENIOS Y CONTRATOS PARA EL AÑO 2023" publicada en el BOULPGC del 3 de febrero de 2023 [34], para un investigador de Titulación de Máster (MESES 3) bajo una jornada de 20 horas semanales el sueldo base se corresponde con el recogido en la Tabla 29.

Tabla 29 Coste de Recursos Humanos.

Recursos Humanos			
Personal	Coste Total Mensual	Tiempo	Total
Investigador	1389,63 €	4 meses	5558,52 €

El coste final de los recursos humanos es de CINCO MIL QUINIENTOS CINCUENTA Y OCHO EUROS CON CINCUENTA Y DOS CÉNTIMOS.

11.4 Material Fungible

Se entiende como material fungible los recursos asociados como miscelánea como son los folios, utensilios de escritura, tinta, entre otros. Su valor se recoge en la Tabla 30.

Tabla 30 Coste del Material Fungible.

Material Fungible	
Descripción	Total
Miscelánea	150,00 €

El coste total del material fungible es de CIENTO CINCUENTA EUROS.

11.5 Redacción de Documento

Dadas las horas asignadas a la redacción del documento en el anteproyecto, y teniendo en cuenta el coste por hora según el baremo publicado en el BOULPGC del 3 de febrero de 2023 [69], los costes de redacción son los que se recogen en la Tabla 31.

Tabla 31 Coste de Redacción del documento.

Redacción			
Personal	Coste total por hora	Tiempo	Total
Investigador	17,37 €	90 horas	1563,33 €

El coste de redacción asciende a un total de MIL QUINIENTOS SESENTA Y TRES EUROS CON TREINTA Y TRES CÉNTIMOS.

11.6 Aplicación de Impuestos y Coste Final

En la Tabla 32 se muestra el Presupuesto Final del presente TFM, aplicando el correspondiente IGIC (Impuesto General Indirecto Canario).

Tabla 32 Presupuesto Total.

Presupuesto Final	
Partidas	Totales (€)
Recursos Hardware	1123,27
Recursos Software	190,46
Recursos Humanos	5558,52
Material Fungible	150,00

Redacción del Documento	1563,33
Total	8585,58
IGIC (7%)	600,99
Total, Aplicando IGIC	9186,57

El presupuesto total del Trabajo Fin de Máster “APLICACIÓN IOT LORAWAN PARA EL ANÁLISIS DE PATRONES DE COMPORTAMIENTO BASADA EN MÓDULOS HW DE DETECCIÓN DE PERSONAS” asciende a NUEVE MIL CIENTO OCHENTAISÉIS EUROS, con CINCUENTAISIETE céntimos (9186,57 €).

Fdo.: D. Julio Francisco Estela Bravo

En Las Palmas de Gran Canaria a 17 de julio de 2024

12 Pliego de Condiciones

El Pliego de condiciones expone las condiciones bajo las que se ha desarrollado el presente Trabajo Fin de Máster. A continuación, se muestra el conjunto de recursos hardware y software empleados durante su realización.

12.1 Condiciones Hardware

En la Tabla 33 se recogen los equipos y dispositivos hardware utilizados, con sus principales características.

Tabla 33 Recursos Hardware empleados.

Equipo/Dispositivo	**Modelo**	**Fabricante/Comerciante**
Person Sensor	V 1.0	Useful Sensor Inc
Arduino	MKRWAN 1310	Arduino AG
Batería	BAT536 M6 Li-Po de 1400mAh de 3.7V	
Gateway Heltec HT-M00	HT-M00	Heltec Automation
Gateway RAK	RAK7268 WisGate Edge Lite 2	RAKwireless
Ordenador	ASUS TUF Dash F15	ASUS

12.2 Condiciones Software

En la Tabla 34 se recogen las aplicaciones software utilizadas, con su versión correspondiente.

Tabla 34 Recursos software empleados.

Aplicación	**Versión**	**Desarrollador/Comerciante**
Sistema Operativo	Microsoft Windows 11 Home 10.0.22631 N/D Compilación 22631	Microsoft
Microsoft Office	2019	Microsoft
Foxit Reader	V 12.1.3.15356	Foxit Software
Google Chrome	V 126.0.6478.127	Google LLC
IDE de Arduino	V 2.3.2	Arduino AG
Terminal Tera Term	V4.106	Open Source
Terminal Putty	V 0.81	Open Source
TTN Console Cloud	V 3.31.0	The Things Network Foundation
InfluxDB Cloud Serverless	V 3.0	InfluxData
Telegraf	V 1.30.3	InfluxData
Grafana Enterprise	V 11.0.0	Grafana Labs

13 Anexo

13.1 Comunicación entre Clientes MQTT y un *Broker* MQTT

Una de las características clave del protocolo MQTT (*Message Queuing Telemetry Transport*) es su enfoque eficiente y ligero para el intercambio de mensajes entre dispositivos IoT. La base de esta comunicación es la conexión MQTT, que permite a los dispositivos intercambiar datos de manera segura y fiable mediante un *broker* MQTT. En este anexo, se explorará el proceso de establecimiento de una conexión MQTT y los diferentes parámetros involucrados [50].

El protocolo MQTT se basa en TCP/IP, como se representa en la Figura 227, lo que significa que el cliente MQTT y el *broker* MQTT deben contar con una pila TCP/IP [50].

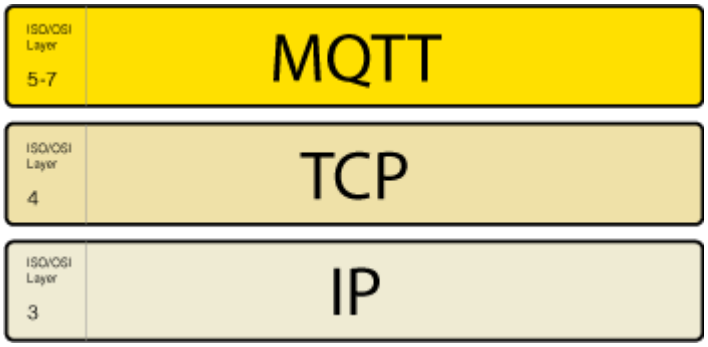


Figura 227 Modelo OSI [50].

Las conexiones MQTT se establecen siempre entre un cliente y un *broker*, de forma que los clientes MQTT nunca se conectan directamente entre sí. Para iniciar una conexión, el cliente MQTT envía un mensaje de tipo CONNECT al *broker* MQTT, que responde con un mensaje de tipo CONNACK y un código de estado, como se representa en la Figura 228. Una vez que se establece la conexión, el *broker* MQTT la mantiene abierta hasta que el cliente envía un comando de desconexión, o bien se produce una interrupción en la conexión.

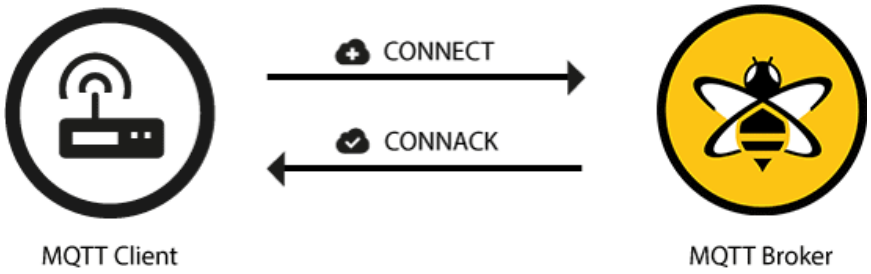


Figura 228 Ejemplo de mensajes de conexión entre cliente y broker MQTT [50].

En muchos casos, los clientes MQTT se encuentran detrás de *routers* que utilizan la traducción de direcciones de red (NAT, por sus siglas en inglés) para convertir direcciones de red privada (como `192.168.x.x` o `10.0.x.x`) en direcciones que son accesibles desde el exterior. Como se mencionó, el cliente MQTT inicia la conexión enviando un mensaje de tipo CONNECT al *broker* MQTT. Dado que el *broker* MQTT dispone de una dirección pública y mantiene la conexión abierta para permitir el envío y la recepción bidireccionales de mensajes (tras el procedimiento CONNECT inicial), los clientes MQTT ubicados tras *routers* NAT no tendrán dificultades.

13.1.1 Mensaje de conexión del Cliente al *Broker* MQTT

En el contexto del protocolo MQTT, el contenido del mensaje de comando CONNECT que envía un cliente al *broker* MQTT con el propósito de iniciar una conexión, es el que se representa en la Figura 229. En situaciones en las que este mensaje adolece de una estructura incorrecta o en las cuales transcurre un período sustancial desde la apertura de un *socket* de red hasta la transmisión del mensaje CONNECT, el *broker* MQTT opta por finalizar la conexión con el fin de prevenir posibles abusos por parte de clientes maliciosos que podrían perjudicar el rendimiento del *broker* MQTT [50].

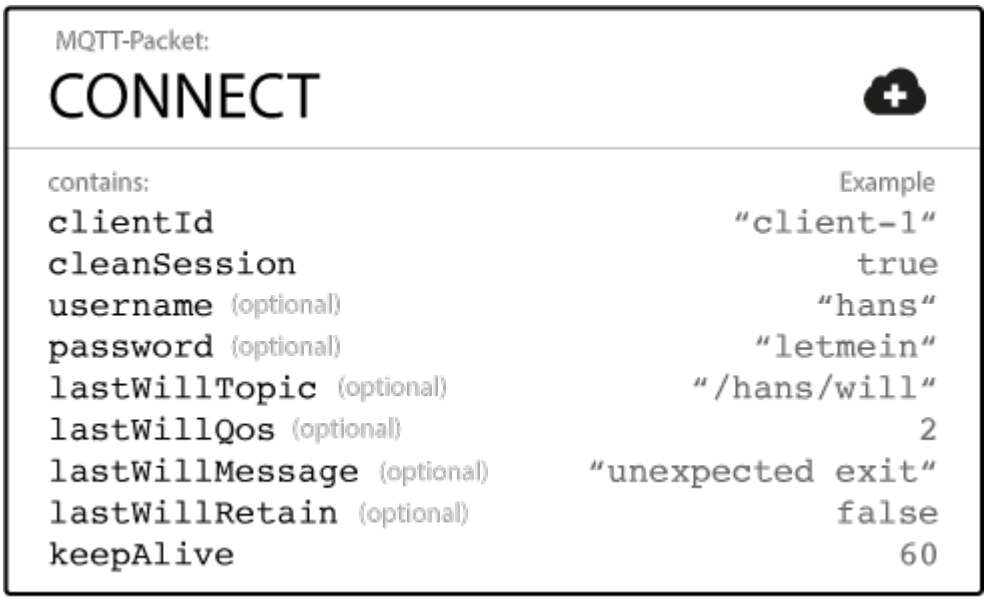


Figura 229 Ejemplo de mensaje del cliente MQTT al broker MQTT [50].

El campo *ClientId* representa un identificador único que distingue a cada cliente MQTT que se conecta a un *broker* MQTT y le permite realizar un seguimiento del estado actual del cliente. Para garantizar la singularidad, el campo *ClientId* debe ser específico para cada cliente y *broker*. MQTT 3.1.1 permite un *ClientId* vacío si no es necesario mantener un estado por parte del *broker*. Sin embargo, esta conexión debe tener el campo *clean session* establecida en verdadero, o el *broker* rechazará la conexión [50].

El campo *CleanSession* indica si el cliente desea establecer una sesión persistente con el *broker*. Cuando *CleanSession* se establece al valor falso (*CleanSession* = *false*), lo que se considera una sesión persistente, el *broker* almacena todas las suscripciones del cliente y todos los mensajes perdidos para el cliente que se suscribió con un nivel de calidad de servicio (QoS) 1 o 2. En contraste, cuando *CleanSession* se establece al valor verdadero (*CleanSession* = *True*), el *broker* no retiene ninguna información para el cliente y descarta cualquier estado anterior de cualquier sesión persistente [50].

MQTT ofrece la opción de incluir un nombre de usuario y una contraseña para la autenticación y autorización del cliente. Sin embargo, es importante tener en cuenta que enviar esta información en texto plano plantea un riesgo de seguridad. Para mitigar este riesgo, se recomienda hacer uso de cifrado o *hashing* (como a través de TLS) con el fin de

proteger las credenciales. También se recomienda utilizar una capa de transporte segura al transmitir datos sensibles [50].

La característica MQTT *Last Will and Testament* (LWT) incluye un último mensaje de voluntad que notifica a otros clientes cuando un cliente se desconecta inesperadamente. Este mensaje puede ser especificado por el cliente dentro del mensaje de tipo CONNECT como un mensaje MQTT y un tema. Cuando el cliente se desconecta abruptamente, el *broker* envía el mensaje LWT en nombre del cliente [50].

La característica *Keep Alive* de MQTT permite que el cliente especifique un intervalo de tiempo en segundos y lo comunique al *broker* al establecer una conexión. Este intervalo determina el período más largo durante el cual el *broker* y el cliente pueden permanecer conectados sin enviar ningún mensaje. Para garantizar que la conexión permanezca activa, el cliente envía mensajes regulares de solicitud de PING, y el *broker* responde con una respuesta de PING. Este método permite que ambos lados determinen si el otro todavía está disponible [50].

Cuando se establece la conexión con un *broker* MQTT desde un cliente MQTT 3.1.1, el intervalo *Keep Alive* resulta esencial. Sin embargo, algunas bibliotecas MQTT tienen opciones de configuración adicionales, como la forma en que se almacenan los mensajes en cola en una implementación específica [50].

13.1.2 Mensaje de respuesta del *Broker* MQTT al Cliente

Cuando un *broker* MQTT recibe un mensaje de tipo CONNECT, está obligado a responder con un mensaje de tipo CONNACK, cuyo contenido se representa en la Figura 230 [50].

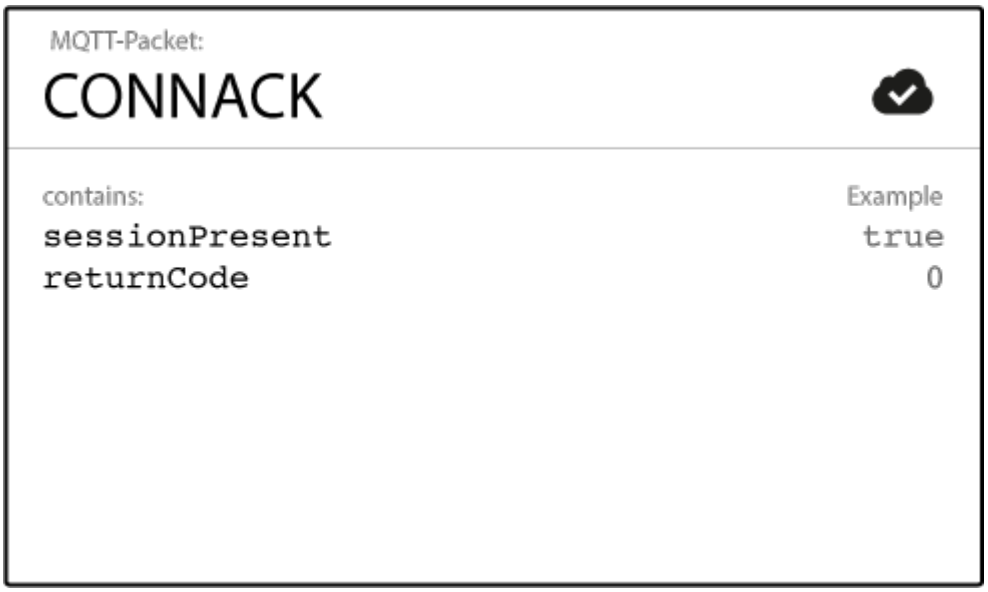


Figura 230 Ejemplo de mensaje de respuesta del bróker MQTT al cliente [50].

El mensaje de tipo CONNACK contiene dos entradas de datos: el indicador de sesión presente (*sessionPresent flag*) y el código de retorno de conexión (*connect return code*) [50].

El indicador *sessionPresent* informa al cliente si se encuentra disponible en el *broker* una sesión anterior. En caso de que el cliente haya solicitado una sesión limpia (*clean session*), este indicador siempre será falso. Sin embargo, si el cliente solicita reanudar una sesión anterior, este indicador será verdadero si el *broker* aún tiene información de alguna sesión almacenada. Este indicador ayuda a los clientes a determinar si necesitan volver a suscribirse a temas o si el *broker* todavía tiene las suscripciones de la sesión anterior [50].

El código de retorno *(returnCode*) es un código de estado que informa al cliente sobre el éxito o el fracaso del intento de conexión. Este código puede indicar varios tipos de errores, como credenciales inválidas o versiones de protocolo no admitidas, recogidas en la Tabla 35 [50].

Tabla 35 Códigos de retorno de mensaje CONNACK [50].

Return Code	**Return Code Response**
0	Connection accepted
1	Connection refused, unacceptable

	protocol version
2	Connection refused, identifier rejected
3	Connection refused, server unavailable
4	Connection refused, bad user name or password
5	Connection refused, not authorized

Es fundamental prestar atención al código de retorno de conexión (*connect returnCode*), ya que puede ayudar a diagnosticar problemas de conexión. Por ejemplo, si el código de retorno indica un fallo de autenticación, el cliente puede intentar reconectar con las credenciales correctas [50].

13.2 Mensaje MQTT PUBLISH

En el protocolo MQTT, un cliente puede publicar mensajes inmediatamente cuando se conecta a un *broker*. Los mensajes se filtran en función de temas (*Topics*), y cada mensaje debe contener un tema que el *broker* puede utilizar para reenviar el mensaje a los clientes interesados. La carga útil (*payload*) de cada mensaje incluye los datos que se van a transmitir en formato de bytes, y el cliente que envía puede elegir enviar cualquier tipo de datos, incluyendo texto, números, imágenes, datos binarios e incluso XML o JSON completos [70], como se muestra en la Figura 231.

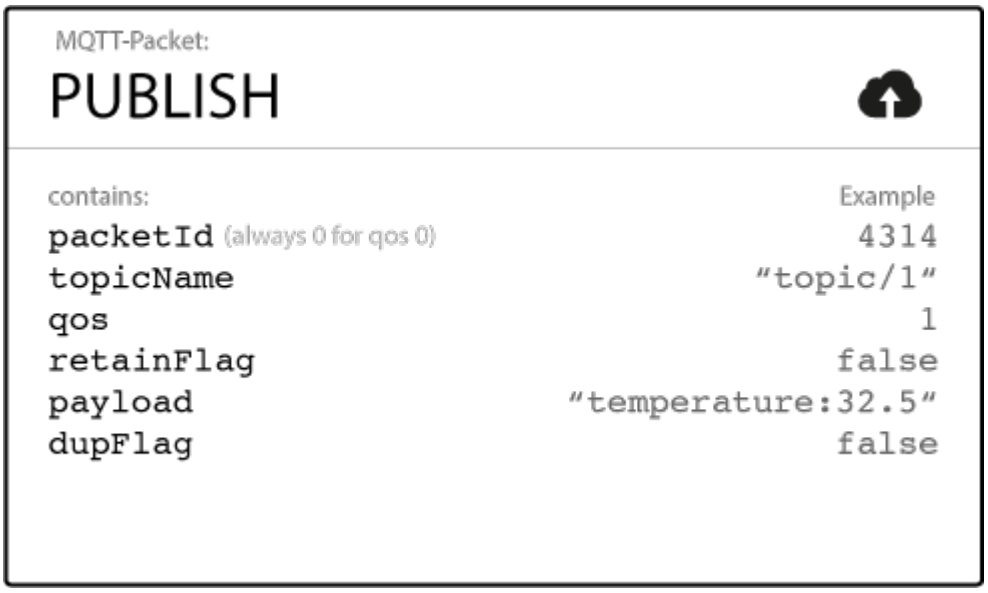


Figura 231 Ejemplo del formato del Payload MQTT [70].

En el protocolo MQTT, el *payload* puede estructurarse según el caso de uso específico del cliente. La carga útil es el contenido principal del mensaje y es lo que los clientes se suscriben, reciben y procesan [70]. Un mensaje de tipo PUBLISH en MQTT tiene varios atributos que determinan su comportamiento, incluyendo el identificador del paquete, el nombre del tema, o el nivel de calidad de servicio, entre otros.

El Identificador de Paquete (*PacketId*) es un atributo esencial en MQTT. Se utiliza para identificar el mensaje específico y garantizar que los mensajes se entreguen en el orden en que se enviaron, especialmente cuando se utilizan niveles de Calidad de Servicio (QoS) superiores a cero. Este campo es asignado por el cliente y se incluye en los mensajes de tipo PUBLISH, PUBREL, PUBREC y PUBCOMP. Cuando el *broker* recibe un mensaje de tipo PUBLISH, asigna un *Packet ID* al mensaje y envía un mensaje de tipo PUBACK al cliente que contiene el *Packet ID* del mensaje de tipo PUBLISH. El cliente utiliza el mensaje de tipo PUBACK para confirmar que el *broker* ha recibido el mensaje [70].

En el protocolo MQTT, los mensajes relacionados con la publicación y la confirmación de mensajes se dividen en varias etapas [70]:

1. Publicación (PUBLISH): Esta es la primera etapa del proceso, e implica que un cliente MQTT publique un mensaje en el *broker*. El mensaje contiene un tema y una carga útil.
2. Publicación Recibida (PUBREC): Tras recibir el mensaje PUBLISH, el *broker* envía un mensaje de tipo PUBREC para confirmar que ha recibido el mensaje. Esta es la segunda etapa del proceso.
3. Liberación de Publicación (PUBREL): Una vez que el cliente recibe el mensaje de tipo PUBREC, envía un mensaje de tipo PUBREL para liberar al *broker* de la responsabilidad de mantener el mensaje en memoria. Esta es la tercera etapa.
4. Publicación Completa (PUBCOMP): Finalmente, el *broker* envía un mensaje de tipo PUBCOMP para confirmar que ha recibido y procesado correctamente el mensaje. Esta es la cuarta y última etapa del proceso.

Estos cuatro mensajes forman parte de los mecanismos de Calidad de Servicio (QoS) del protocolo MQTT, que garantizan una entrega de mensajes confiable. El nivel de QoS determina la cantidad de mensajes intercambiados entre el cliente y el *broker* [70].

Es importante destacar que el identificador de paquete permite reconocer de manera única un mensaje a medida que se transfiere entre el cliente y el *broker*. El identificador de paquete solo es relevante para los niveles de QoS superiores a cero. Esto es válido no solo para mensajes de tipo PUBLISH, sino también para los mensajes de tipo SUBSCRIBE, UNSUBSCRIBE y CONNECT [70].

La biblioteca del cliente y/o el *broker* son responsables de establecer este identificador interno de MQTT. Cuando se utiliza un nivel de QoS superior a cero, el cliente debe esperar un mensaje de tipo PUBACK o PUBREC del *broker* antes de enviar el siguiente mensaje. El cliente también debe llevar un registro de los *Packet ID* que ha enviado y recibido para garantizar que los mensajes no se pierdan ni se dupliquen. En resumen, el campo *Packet ID* es esencial para el mecanismo de fiabilidad de MQTT y ayuda a garantizar que los mensajes se entreguen de manera correcta y eficiente [70].

13.3 Topic MQTT

El concepto de tema o *topic* representa un elemento fundamental del protocolo MQTT. Este elemento se estructura jerárquicamente mediante el uso de barras diagonales (*forward slashes*) como separadores, generando una cadena de texto sencilla. Esto guarda similitud con una ruta URL, pero carece de los componentes de protocolo y dominio. Los temas MQTT desempeñan un papel fundamental al etiquetar los mensajes y ofrecen a los clientes la posibilidad de suscribirse a mensajes específicos [70].

Por ejemplo, un dispositivo que mide la temperatura podría publicar sus lecturas en el tema *sensors/temperature/livingroom*. Un cliente interesado en estas lecturas puede suscribirse a este tema y recibir actualizaciones a medida que se publican.

MQTT proporciona dos tipos de comodines para usar con suscripciones de temas [70]:

- "+" (signo más) se utiliza para coincidir con un solo nivel en la jerarquía. Por ejemplo, una suscripción a *sensors/+/livingroom* coincidiría con *sensors/temperature/livingroom* y *sensors/humidity/livingroom*, pero no con *sensors/temperature/kitchen*.
- "#" (signo numeral) se utiliza para coincidir con varios niveles en la jerarquía. Por ejemplo, una suscripción a *sensors/#* coincidiría con *sensors/temperature/livingroom*, *sensors/humidity/kitchen* y *sensors/power/meter1*.

La suscripción a un gran número de temas puede tener un impacto significativo en el rendimiento del *broker*. Esto se debe a que cada mensaje que se publica en un tema al que un cliente está suscrito, debe ser entregado a ese cliente. Si múltiples clientes se suscriben a varios temas, esto puede convertirse rápidamente en una carga pesada para el *broker* [70].

El uso de comodines para suscribirse a varios temas con una sola suscripción también puede afectar el rendimiento. Cuando un cliente se suscribe a un tema con un comodín, el *broker* debe evaluar cada mensaje publicado en un tema coincidente y determinar si debe reenviarlo al cliente. Si el número de temas coincidentes es grande, esto puede poner a prueba los recursos del *broker* [70].

Para evitar problemas de rendimiento, es importante utilizar suscripciones a temas de manera eficiente. Un enfoque es utilizar filtros de temas más específicos siempre que sea posible, en lugar de depender de comodines. Otro enfoque es utilizar suscripciones compartidas, que permiten a varios clientes compartir una sola suscripción a un tema. Esto puede ayudar a reducir la cantidad de suscripciones y mensajes que el *broker* debe gestionar. Finalmente, supervisar el rendimiento del *broker* y ajustar su configuración según sea necesario es importante para garantizar un rendimiento óptimo [70].

13.4 Calidad de Servicio (QoS) en MQTT

Los niveles de QoS se representan mediante un número que varía en un rango comprendido entre los valores 0 y 2. Cada nivel ofrece un grado diferente de confiabilidad y garantía para la entrega de mensajes.

- QoS 0 (como máximo una vez): Este nivel no garantiza que un mensaje se entregue. El mensaje se envía una vez y, si se pierde o no es recibido por el destinatario, no se volverá a enviar [70].
- QoS 1 (al menos una vez): Este nivel garantiza que un mensaje se entregue al menos una vez, pero puede entregarse varias veces en caso de problemas o fallos de red [70].
- QoS 2 (exactamente una vez): Este nivel proporciona el nivel más alto de garantía para la entrega de mensajes. El mensaje se garantiza que se entregará exactamente una vez, pero este nivel requiere más comunicación entre el remitente y el receptor, lo que puede aumentar la latencia y el tráfico de red [70].

La elección del nivel adecuado de QoS depende del caso de uso específico. Por ejemplo, QoS 0 podría ser idóneo para datos no críticos, mientras que QoS 2 podría ser necesario para datos críticos que requieran niveles de confiabilidad elevados [70].

Es importante tener en cuenta que el nivel de QoS puede afectar el rendimiento del *broker* y la red, por lo que se recomienda utilizar el nivel adecuado para el caso de uso específico [70].

El atributo *Retained Flag* es una característica importante que determina si el *broker* almacena un mensaje como el último valor conocido para un tema específico. Cuando este atributo se establece al valor verdadero (*true*), el *broker* almacenará el mensaje más reciente que coincida con el tema especificado, independientemente de si hay clientes suscritos [70]. Cuando un nuevo cliente se suscribe a un tema con un mensaje retenido, el *broker* envía el último mensaje retenido (en ese tema) al cliente, lo que permite a los clientes recibir la información más reciente y relevante, incluso si no se han suscrito a ese tema anteriormente [70].

Es importante destacar que, al igual que otros parámetros, el uso de mensajes retenidos también puede afectar el rendimiento del *broker*, especialmente si existen muchos mensajes retenidos. Además, si un mensaje retenido se actualiza con frecuencia, puede provocar un aumento en el tráfico de red y afectar el rendimiento de la red [70].

El indicador DUP (*Duplicated*) de MQTT indica que un mensaje es una duplicación y se ha reenviado porque el destinatario previsto (cliente o *broker*) no reconoció el mensaje original. Solo es relevante para mensajes con un QoS mayor que 0. Cuando un cliente o un *broker* recibe un mensaje con el indicador DUP activado, debe ignorar el mensaje si ya ha recibido un mensaje con el mismo ID. Por el contrario, si no lo ha recibido previamente, el cliente o el *broker* debe procesar el mensaje de manera normal [70].

El protocolo MQTT gestiona automáticamente el mecanismo de reenvío y duplicados, pero es importante tener en cuenta que esto puede afectar el rendimiento de la red y aumentar el tráfico de la red[70].

13.5 Mensajes MQTT del proceso de subscripción a Topics

Publicar un mensaje no tiene sentido si nadie lo recibe. Aquí es donde entra en juego el proceso de suscripción. Una vez que un cliente publica un mensaje en un *broker* MQTT, el mensaje debe entregarse a los clientes interesados. Los clientes que desean recibir mensajes sobre temas de interés envían un mensaje de tipo SUBSCRIBE al *broker* [70]. El mensaje de tipo SUBSCRIBE y contiene un identificador de paquete único y una lista de suscripciones, como se muestra en la Figura 232.

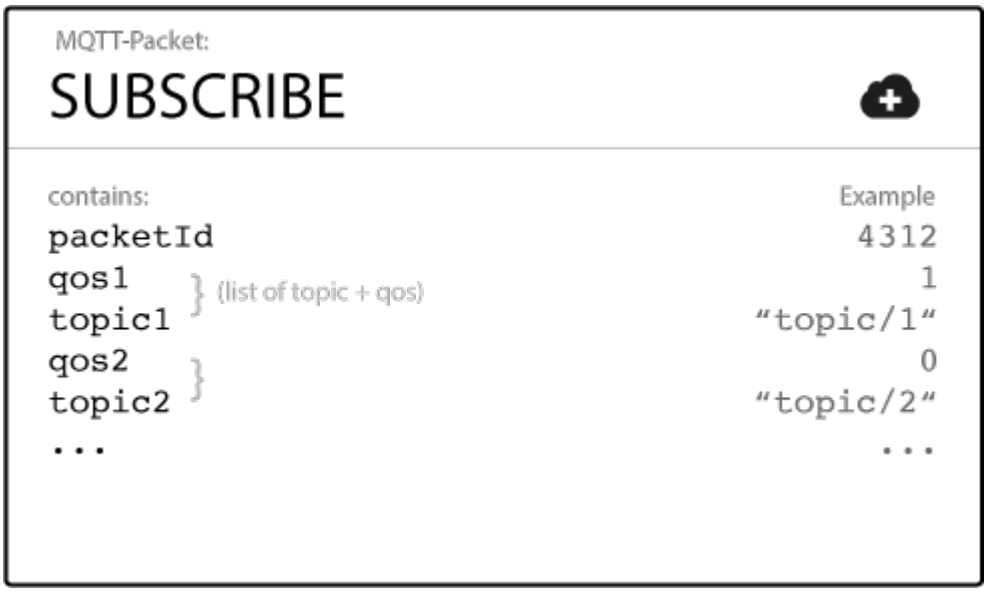


Figura 232 Ejemplo de paquete de suscripción [70].

El identificador de paquete (*packetId*) es único e identifica un mensaje a medida que se transfiere entre el cliente y el *broker*. La biblioteca del cliente o el *broker* son responsables de establecer este identificador MQTT interno [70].

Un mensaje de tipo SUBSCRIBE puede contener múltiples suscripciones para un cliente. Cada suscripción incluye un tema y un nivel de QoS. En un mensaje de tipo SUBSCRIBE, el tema puede contener comodines que permiten suscribirse a un patrón de temas en lugar de a un tema específico. Si hay suscripciones superpuestas para un cliente, el *broker* entrega el mensaje con el nivel de QoS más alto para ese tema [70].

En general, MQTT permite a los clientes suscribirse a temas específicos, recibir mensajes publicados en esos temas y procesar las cargas útiles según su caso de uso específico. El identificador de paquete y el nivel de QoS en un mensaje de tipo SUBSCRIBE garantizan que los mensajes se entreguen de manera confiable y con el nivel de calidad adecuado [70].

Una vez que un cliente envía un mensaje de tipo SUBSCRIBE con la lista de temas deseados y niveles de QoS a un *broker* MQTT, el *broker* responde con un mensaje de tipo SUBACK, como se muestra en la Figura 233, que confirma la suscripción e indica el nivel de QoS máximo que el *broker* entregará.

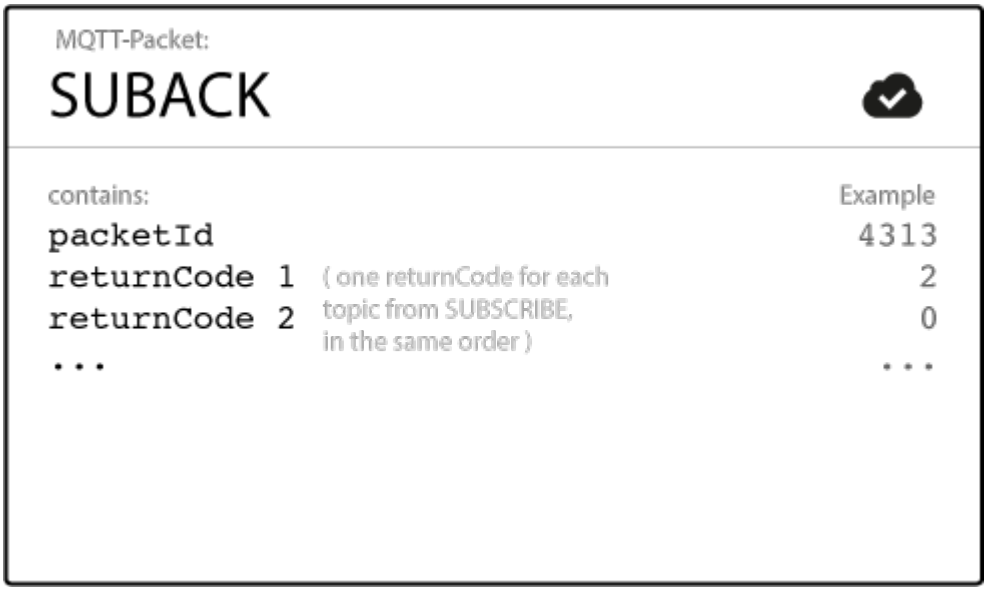


Figura 233 Ejemplo del paquete SUBACK [70].

El mensaje de tipo SUBACK incluye el mismo identificador de paquete (*packetId*) que el cliente incluyó en el mensaje de tipo SUBSCRIBE, lo que permite al cliente emparejar la confirmación con la solicitud original [70].

El mensaje de tipo SUBACK también incluye un código de retorno (*returnCode*) para cada par tema/nivel de QoS especificado en el mensaje de tipo SUBSCRIBE. Los códigos de retorno son valores binarios que indican si el *broker* ha aceptado o rechazado la solicitud de suscripción para cada tema [70], recogidos en la Tabla 36.

El código de retorno de fallo se representa con el valor `0x80`, e indica que el *broker* no acepta la suscripción. Esto puede ocurrir si el cliente no tiene permisos suficientes para suscribirse al tema, si el tema está mal formado, o si hay otro problema con la solicitud de suscripción. Cuando un cliente recibe un código de retorno de fallo, debe volver a intentar la suscripción con un tema o nivel de QoS diferentes o tomar medidas apropiadas para abordar el problema con la solicitud de suscripción [70].

Tabla 36 Códigos de retorno del broker MQTT en respuesta al mensaje SUBSCRIBE [70].

Return Code	Return Code Response

0	Success - Maximum QoS 0
1	Success - Maximum QoS 1
2	Success - Maximum QoS 2
128	Failure

En la Figura 234 se representan los mensajes MQTT involucrados en el proceso de subscripción a un *Topic* y la publicación de mensajes MQTT.

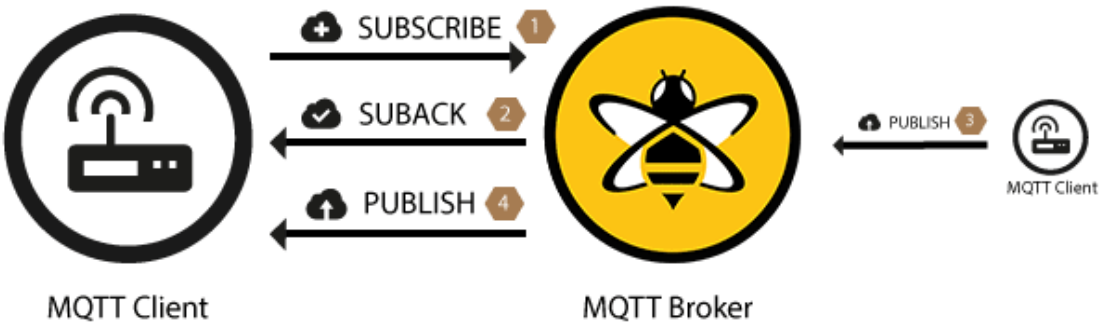


Figura 234 Ejemplo de funcionamiento de mensajes MQTT SUBSCRIBE, SUBACK y PUBLISH [70].

Después de que un cliente se haya suscrito a temas de interés y haya recibido mensajes publicados en esos temas, es posible que en algún momento necesite anular la suscripción. Ahora se hablará del homólogo del mensaje de tipo SUBSCRIBE, el mensaje de tipo UNSUBSCRIBE y el correspondiente mensaje de tipo UNSUBACK que confirma la cancelación de la suscripción [70].

En MQTT, los clientes pueden cancelar la suscripción a los temas a los que se han suscrito enviando un mensaje de tipo UNSUBSCRIBE al *broker*. Al igual que con los mensajes de tipo SUBSCRIBE, este mensaje incluye un identificador de paquete para identificarlo de manera única y una lista de temas de los que cancelar la suscripción, como se muestra en la Figura 235 [70].

Figura 235 Ejemplo de paquete MQTT UNSUBSCRIBE [70].

Al igual que en el mensaje de tipo SUBSCRIBE, el identificador de paquete (*packetId*) en el mensaje de tipo UNSUBSCRIBE sirve como un identificador MQTT interno para el flujo de mensajes entre el cliente y el *broker*. Asegura que el cliente y el *broker* puedan hacer un seguimiento del mensaje y sus correspondientes mensajes de confirmación [70].

La lista de temas (*list of topics*) en el mensaje de tipo UNSUBSCRIBE puede contener uno o varios temas de los que el cliente desea cancelar la suscripción. No es necesario especificar el nivel de QoS, ya que el *broker* cancelará la suscripción al tema independientemente del nivel de QoS con el que se suscribió originalmente [70].

Tras recibir el mensaje de tipo UNSUBSCRIBE, el *broker* envía un mensaje de confirmación de tipo UNSUBACK para confirmar la eliminación de las suscripciones del cliente. Este mensaje incluye el identificador de paquete del mensaje UNSUBSCRIBE y sirve como una confirmación de que el *broker* ha eliminado con éxito los temas de la lista de suscripciones del cliente, como se observa en la Figura 236 [70].

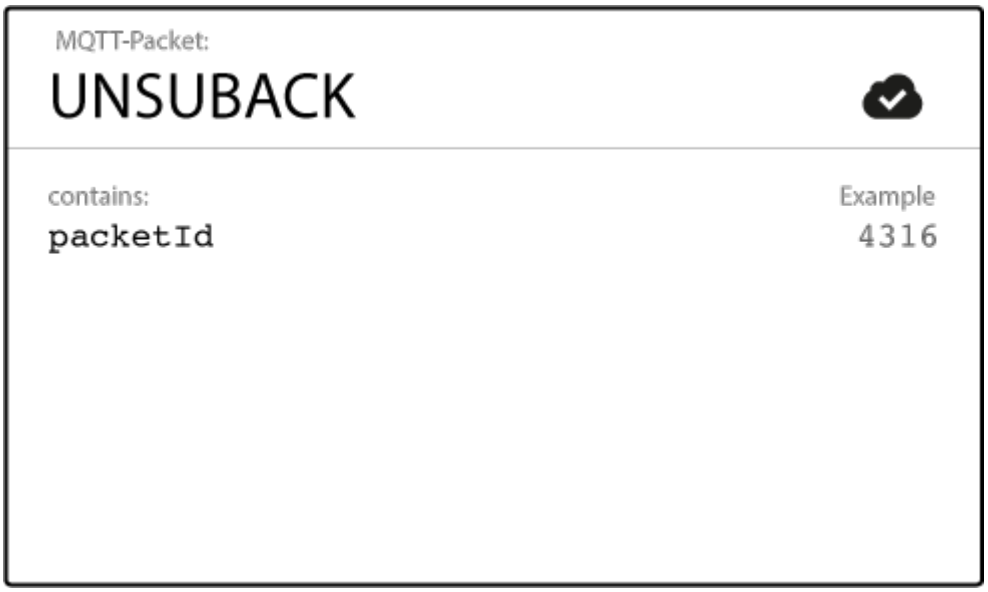


Figura 236 Ejemplo de paquete MQTT UNSUBACK [70].

El identificador de paquete (*packetId*) en el mensaje de tipo UNSUBACK es el mismo que el del mensaje de tipo UNSUBSCRIBE asociado. Esto asegura que el cliente pueda identificar el mensaje de confirmación y correlacionarlo con el mensaje de tipo UNSUBSCRIBE original [70].

El mensaje de tipo UNSUBACK incluye una lista de códigos de retorno para cada par *Topic*/QoS del que se canceló la suscripción. Un código de retorno de valor `0` indica una eliminación exitosa, mientras que un código de retorno de `17` indica una eliminación fallida debido a un tema inválido o mal formado. También pueden especificarse otros códigos de retorno para diferentes escenarios de error [70]. Después de recibir el mensaje de tipo UNSUBACK del *broker*, el cliente puede asumir que las suscripciones en el mensaje de tipo UNSUBSCRIBE se han eliminado [70].

Estos detalles proporcionan una comprensión completa de cómo los clientes pueden cancelar la suscripción a temas, y cómo los *brokers* confirman la eliminación de esas suscripciones a través de los mensajes de tipo UNSUBSCRIBE y UNSUBACK, respectivamente, como se representa en la Figura 237 [70].

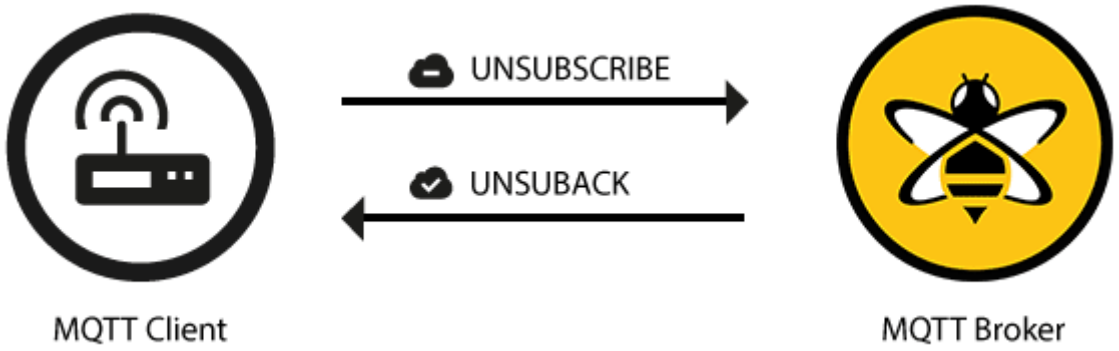


Figura 237 Ejemplo de mensaje MQTT UNSUBACK

El protocolo MQTT ofrece un enfoque flexible de los datos para la publicación de mensajes entre clientes y *brokers*. Al utilizar temas para filtrar mensajes, los clientes pueden suscribirse de manera rápida y sencilla al contenido que les interesa. La carga útil de cada mensaje se puede personalizar para satisfacer las necesidades específicas de cada cliente, y el soporte de MQTT para varios tipos de datos lo convierte en una solución versátil para numerosos casos de uso [70].