UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

Desarrollo de una red LoRa/LoRaWAN para el monitoreo del sistema de abastecimiento de agua potable PUMAGUA

TESIS

Que para obtener el título de

Ingeniero Eléctrico Electrónico

P R E S E N T A

Néstor Gabriel León Mendoza

DIRECTOR DE TESIS

M.I. Mario Alberto Hernández Flores

Ciudad Universitaria, Cd. Mx., 2025

PROTESTA UNIVERSITARIA DE INTEGRIDAD Y HONESTIDAD ACADÉMICA Y PROFESIONAL
(Titulación con trabajo escrito)

De conformidad con lo dispuesto en los artículos 87, fracción V, del Estatuto General, 68, primer párrafo, del Reglamento General de Estudios Universitarios y 26, fracción I, y 35 del Reglamento General de Exámenes, me comprometo en todo tiempo a honrar a la institución y a cumplir con los principios establecidos en el Código de Ética de la Universidad Nacional Autónoma de México, especialmente con los de integridad y honestidad académica.

De acuerdo con lo anterior, manifiesto que el trabajo escrito titulado DESARROLLO DE UNA RED LORA/LORAWAN PARA EL MONITOREO DEL SISTEMA DE ABASTECIMIENTO DE AGUA POTABLE PUMAGUA que presenté para obtener el titulo de INGENIERO ELÉCTRICO ELECTRÓNICO es original, de mi autoría y lo realicé con el rigor metodológico exigido por mi Entidad Académica, citando las fuentes de ideas, textos, imágenes, gráficos u otro tipo de obras empleadas para su desarrollo.

En consecuencia, acepto que la falta de cumplimiento de las disposiciones reglamentarias y normativas de la Universidad, en particular las ya referidas en el Código de Ética, llevará a la nulidad de los actos de carácter académico administrativo del proceso de titulación.

NESTOR GABRIEL LEON MENDOZA
Número de cuenta: 316257136

Agradecimientos

Quiero expresar mi más sincero agradecimiento al proyecto PAPIIT-DGAPA "Diseño, análisis e implementación de una red celular 5G-LoRa en la UNAM con un fronthaul DWDM para aplicaciones IoTy servicios en una ciudad inteligente", con clave AG100724, cuyo financiamiento fue decisivo para hacer realidad esta tesis. El apoyo económico que recibí a través de la beca no solo me permitió dedicar el tiempo necesario a esta investigación, sino que también fue fundamental para desarrollar el proyecto completo y lograr los resultados que aquí presento. Sin este respaldo, no habría sido posible llevar a cabo un trabajo de esta envergadura.

También quiero agradecer a todo el personal de PUMAGUA, particularmente al Dr. Fernando González Villarreal, Director de PUMAGUA, por abrirme las puertas de su infraestructura y por todas las facilidades que me brindaron durante el proceso. Gracias a ellos pude recopilar la información necesaria y realizar las pruebas que el proyecto requería. Un agradecimiento especial por permitirme publicar material relacionado con sus instalaciones, incluyendo fotografías, ubicaciones específicas, así como por brindar el equipo necesario para adecuar su infraestructura a este trabajo.

No puedo dejar pasar por alto a todo el equipo de trabajo del Laboratorio de Redes Inalámbricas de la Facultad de Ingeniería, especialmente a los estudiantes de servicio social y a los demás becarios que participaron en distintas etapas del proyecto, no sería justo mencionar nombres ya que se olvidaría alguno. Su esfuerzo, dedicación y trabajo en equipo fueron piezas fundamentales en este proceso, cada uno puso su grano de arena, esencial para llegar al resultado final.

Agradezco al Dr. Víctor Rangel Licea, cuyo seguimiento del proceso fue invaluable, su asesoría constante, su orientación técnica siempre precisa, su dominio del tema y su compromiso real con el seguimiento del proyecto fueron decisivos para que pudiera terminar esta tesis exitosamente. Su paciencia y disponibilidad para resolver dudas marcaron una gran diferencia en todo momento.

De igual manera, quiero agradecer al Dr. Daniel Enrique Ceballos y al Dr. Ramón Gutiérrez Castrejón, cuya asesoría y perspectivas enriquecieron significativamente este trabajo, sus comentarios y observaciones siempre fueron oportunos y me ayudaron a ver aspectos que de otra manera hubieran pasado desapercibidos. Asimismo, expreso mi más profundo agradecimiento a mi asesor, el M.I. Mario Alberto Hernández Flores, cuyo acompañamiento constante fue fundamental en todo este proceso, su guía paciente, su disponibilidad para resolver mis dudas y su apoyo incondicional no sólo fueron clave para el desarrollo técnico de esta tesis, sino también para mantenerme motivado durante los momentos más desafiantes. Su asesoría ha dejado una huella importante en mi formación profesional.

Por último, pero no menos importante, agradezco a todas las personas que, de una forma u otra, aportaron su tiempo, sus conocimientos y su apoyo para que este proyecto saliera adelante, no sólo en el tiempo que duró este trabajo, sino durante toda mi trayectoria a lo largo de la licenciatura. Quiero dedicar un agradecimiento muy especial a mis amigos, quienes fueron mi red de apoyo durante todo este proceso, brindándome palabras de aliento en los momentos de estrés y celebrando conmigo cada pequeño avance. A mis compañeros de carrera, con quienes compartí no sólo clases y proyectos, sino también esas largas jornadas de estudio y ese intercambio de ideas que fortalecieron mi formación. Sobre todo, agradezco a mi familia, cuyo amor incondicional, paciencia y comprensión fueron el cimiento sobre el cual pude construir este logro; a mis padres y hermano, por sus sacrificios y por creer siempre en mí y a mi pareja, por ser mi compañía constante y por ser ese apoyo inquebrantable que me impulsó a seguir adelante incluso en los momentos más demandantes.

Índice general

Índice de figuras

Índice de tablas

Lista de acrónimos

IoT	Internet of Things
LoRa	Long Range
CSS	Chirp Spread Spectrum
TTN	The Things Network
LPWAN	Low Power Wide Area Network
SF	Spread Factor
BW	Band width
SNR	Signal to Noise Ratio
RSSI	Received Signal Strength Indicator
MAC	Media Access Control
SoA	Service-oriented Architecture
RFID	Radio Frequency Identification
OTAA	Over the air activation
ABP	Activation by personalization

Capítulo 1

Introducción

1.1 Antecedentes

El uso de Internet de las Cosas o Internet of Things (IoT) en aplicaciones de monitoreo es cada vez mayor gracias a tecnologías como Long Range (LoRa) o Sigfox, que permiten cubrir grandes zonas con alto nivel de penetración y un consumo de energía reducido. Estas características son clave cuando se habla de entornos rurales o ciudades inteligentes, donde no siempre hay acceso a la red eléctrica, se requiere cubrir grandes distancias o la instalación de una red de monitoreo cableada no es rentable.

El enfoque anterior es importante ya que esta tesis se centró en el desarrollo de una red IoT con todas las características mencionadas anteriormente. Existe un gran número de variables que pueden y han sido censadas a través de IoT, una de ellas es el consumo de agua, fundamental para el uso eficiente de este recurso esencial para la vida. El reto al que se enfrenta esta red IoT son las condiciones debidas a las ubicaciones de los dispositivos o nodos terminales, pues habrá casos donde el entorno sea más similar a un entorno rural, y habrá casos donde será más similar a un entorno urbano.

Otro enfoque importante abordado por esta tesis es el consumo de energía. Según las características descritas, los nodos terminales que se encuentren ubicados en entornos apartados de edificaciones, o bien sin acceso a la red eléctrica, deberán ser capaces de permanecer funcionando unos cuantos años antes de recibir mantenimiento. El tema del consumo de energía, además de demostrar la operación del sistema y analizar su desempeño, es quizás el más importante para esta tesis y, a su vez, es motivo de la elección de la tecnología LoRa y el protocolo de comunicación LoRaWAN, lo anterior debido a que se ha demostrado en la literatura y en los estudios y desarrollos previos del Laboratorio de Redes Inalámbricas, perteneciente al Departamento de Telecomunicaciones de la Facultad de Ingeniería, que las aplicaciones que llevan estas tecnologías cumplen con los requerimientos que una Red de Área Amplia de Bajo Consumo o *Low Power Wide Area Network (LPWAN)* demanda para los dispositivos.

1.2 Definición del problema

El sistema de tuberías de distribución para el manejo sustentable del agua potable con el que cuenta el programa de PUMAGUA en el campus de Ciudad Universitaria se encuentra en constante monitoreo, haciendo uso de una red de IoT basada en la tecnología LoRa. El sistema de censado actual ha estado operativo desde el 2012 aproximadamente y, a parte del censado del consumo de agua por parte de la comunidad universitaria, las características de este monitoreo presentan la posibilidad de identificar y detectar fugas aparentes en algún punto del sistema de distribución de PUMAGUA.

El inconveniente del sistema es que la vida útil de las baterías eléctricas, que alimentan los nodos terminales que transmiten los datos de flujo de agua de las tuberías, está llegando a su fin, lo cual es esperado y sorprendente considerando la fecha de inicio de operación. El problema se agrandó cuando se supo que una renovación de todos los nodos terminales mediante el proveedor de servicios particular resulta ser muy costosa, además de que PUMAGUA tendría un acceso limitado a la operación y mantenimiento de la red.

Como solución a las necesidades de monitoreo de PUMAGUA se planteó el diseño y la implementación de una red de IoT, con las características de una red LPWAN, iniciando en el Conjunto Sur de la Facultad de Ingeniería. El despliegue de los nodos terminales será con base en las ubicaciones de los medidores de flujo de agua instalados en la red de tuberías de PUMAGUA, estos dispositivos establecerán una comunicación con el Gateway central, ubicado en la azotea del Edificio Q de la Facultad de Ingeniería.

Los resultados obtenidos con esta red de IoT servirían como base para diseñar e implementar una red de monitoreo de PUMAGUA en todo el campus de Ciudad Universitaria. Puesto que en los sitios de monitoreo no existen las condiciones para mantener una alimentación de energía eléctrica en los nodos terminales, uno de los requerimientos esenciales en el diseño de la red LoRa/LoRaWAN es que el periodo de mantenimiento de su sistema de alimentación sea de al menos un par de años, por lo tanto, se necesitan implementar estrategias de bajo consumo en la programación y, particularmente, con un análisis que permita estimar o calcular la vida útil del nodo terminal considerando una batería de capacidad conocida y un intervalo de valores para el ciclo de trabajo o periodo de transmisión de los nodos.

1.3 Hipótesis

El uso e implementación de la tecnología LoRa junto con el protocolo LoRaWAN harán posible la creación de la red IoT experimental de bajo consumo energético, cuyo diseño, desempeño y operación demostrará la factibilidad de el despliegue de la red IoT permitiría a PUMAGUA sustituir su sistema de monitoreo actual, por lo tanto, la red y el sistema serían aptos para detectar y localizar fugas o irregularidades en el consumo de las tuberías.

La ubicación que se defina para los nodos terminales provocará que sus parámetros de transmisión LoRa varíen entre sí, lo que generará un impacto significativo y negativo en la duración de la batería que alimente a los dispositivos. Este comportamiento, aunque negativo, servirá como herramienta para analizar y caracterizar el desempeño del sistema, particularmente de los nodos, en diferentes condiciones, considerando los factores que LoRaWAN modifica para la capa física (LoRa).

1.4 Objetivos de la tesis

1.4.1 Objetivo general

Diseñar e implementar una red de IoT, usando la tecnología LoRa y el protocolo de comunicación LoRaWAN, compuesta por dos o más nodos terminales y un Gateway central, para monitorear el consumo de agua potable de la red de distribución del sistema PUMAGUA de Ciudad Universitaria.

1.4.2 Objetivos específicos

1. Integrar la infraestructura de PUMAGUA con los nodos terminales.
2. Implementar estrategias de bajo consumo energético en el diseño y análisis del nodo terminal.
3. Diseñar e implementar sistemas y métodos de medición de corriente adecuados para caracterizar el consumo de energía del nodo terminal.
4. Estimar la duración de la batería que alimente los dispositivos con base en mediciones reales de consumo de energía.
5. Analizar el desempeño y funcionamiento de la red, a través de pruebas de campo.

1.5 Metodología

La metodología o el proceso llevado a cabo en esta tesis es en gran parte empírico, ya que el despliegue de una red LoRaWAN las condiciones de la infraestructura de PUMAGUA así lo demanda. La estimación de la vida útil de las baterías con base en mediciones de corriente reales, constituye una parte esencial de esta tesis, por lo tanto, las etapas necesarias para cumplir con el objetivo del despliegue de la red se enlistan a continuación.

1.5.1 Etapa 1: Comprensión de los dispositivos LoRaWAN

Esta etapa es de suma importancia para el diseño de la red LoRaWAN ya que en ella se pretende identificar y comprender la función de los elementos que componen dicha red, además de conocer las diferentes formas de operar de cada nodo terminal según su clasificación. Otro aspecto clave de esta etapa es conocer las capacidades, la viabilidad y la escalabilidad de la red, asimismo se busca conocer modelos teóricos de consumo energético, todo lo anterior con base en redes y estudios previamente documentados por la comunidad científica especializada en el área.

1.5.2 Etapa 2: Integración de la infraestructura de PUMAGUA

Como su nombre lo indica, la etapa dos consiste en adaptar la infraestructura con la que cuenta PUMAGUA, que básicamente son los medidores de flujo de agua o caudalímetros, con los nuevos dispositivos que serán diseñados. Esta etapa es fundamental para la confiabilidad del sistema, es decir, la correcta integración de los medidores hará posible tener lecturas y datos confiables para el monitoreo del consumo de agua de las tuberías.

1.5.3 Etapa 3: Despliegue de la red LoRaWAN

El despliegue de la red LoRaWAN parte de la integración de los medidores de flujo de agua de PUMAGUA al nodo terminal. Esta etapa es la de mayor extensión debido a que en ella se requiere de toda la instalación, configuración y registro del Gateway central, la construcción de los nodos terminales, la programación del protocolo LoRaWAN, así como el registro de estos en una aplicación web que permita la recepción, visualización y almacenamiento de los datos enviados al Gateway y, finalmente, la instalación de los nodos en las tuberías de PUMAGUA.

1.5.4 Etapa 4: Pruebas de resistencia y mediciones de corriente

El objetivo de esta etapa es poner a prueba la resistencia de los nodos terminales a las exigencias del ambiente al que se enfrentarán en las ubicaciones donde serán instalados, para ello se diseñaron

prototipos que permiten una fácil conexión e instalación de los dispositivos con los medidores de flujo.

Las pruebas de resistencia y funcionamiento se acompañan de las mediciones de corriente, esto para obtener y estudiar datos de consumo energético reales de las ubicaciones de los nodos, el único objetivo es adquirir la mayor información posible del comportamiento y funcionamiento de los dispositivos en las diferentes condiciones en las que se encontrarán.

1.5.5 Etapa 5: Análisis de consumo energético

Analizar el consumo energético requiere, en primer lugar, la caracterización y comprensión de todos los procesos que el nodo lleva a cabo durante un ciclo de operación del protocolo LoRaWAN y. En segundo lugar, se necesita calcular el consumo de energía del nodo asociado a dicho ciclo de operación, con base en las mediciones previas de corriente, y lograr estimar la vida útil del nodo.

1.5.6 Etapa 6: Evaluación del desempeño del sistema

A partir de los cálculos y las pruebas de laboratorio y de campo, implementadas en etapas previas, se consigue un análisis completo del trabajo, con ello se evalúa su potencial para sustituir el sistema de PUMAGUA y la posibilidad de ser desplegado a mayor escala en más zonas que requieran de un monitoreo de flujo de agua. Se pretende que el análisis y los procedimientos desarrollados en este trabajo puedan aplicarse, adaptarse y escalarse a redes LoRaWAN con distintos enfoques o aplicaciones.

1.6 Contribuciones

La principal contribución de este trabajo de tesis es la estimación de la vida útil de las baterías que alimentan los nodos terminales y su análisis de comportamiento, con base en mediciones reales de consumo de corriente, a partir de los respectivos métodos y cálculos propuestos de acuerdo con el funcionamiento de los nodos terminales bajo el protocolo LoRaWAN.

Por otro lado, esta tesis contribuye con el desarrollo de diferentes sistemas o equipos dedicados a la medición de corriente o energía de nodo terminal, bajo las características de consumo que presentan. En este punto, es vital que los sistemas desarrollados o usados cuenten con una gran exactitud, precisión y en gran medida que sean accesibles y libres para el uso futuro de la comunidad científica y de desarrolladores.

Finalmente, otra contribución es la implementación de algoritmos o metodologías alternativos

para el cálculo de consumo energético y estimación de la vida útil de los dispositivos, a partir de sus consumos de energía medidos. En otras palabras, se pretende dejar de lado los modelos matemáticos y metodologías de cálculo de consumo de energía más clásicos en la literatura relacionado, por uno donde la precisión y la exactitud sean mayores.

1.7 Estructura del documento

El trabajo de esta tesis se encuentra distribuido en 6 capítulos, en los cuales se desarrollan todos los puntos de la metodología propuesta en este capítulo. La distribución y el contenido de los capítulos está diseñada para comprender los principios de una red LPWAN, conceptos básicos, desarrollo de la red y el alcance del trabajo realizado.

- **Capítulo 2**
  En este se abordan los conceptos más relevantes relacionados con el uso de la tecnología LoRa y el protocolo LoRaWAN para el desarrollo de redes LPWAN para aplicaciones IoT. También se describen algunos de los casos de implementación de redes LoRaWAN para monitoreo de agua, sea consumo o calidad, destacando sus características clave. Finalmente, se incluyen algunos modelos o trabajos relacionados con la medición de corriente en esta clase de dispositivos.
- **Capítulo 3**
  Este capítulo describe y detalla el proceso del desarrollo de la red LoRaWAN, pasando por la propuesta, análisis y diseño de la red, incluyendo el nodo terminal, Gateway central y aplicación web. El principal enfoque de este capítulo es hacia los nodos terminales, debido a su importancia en el acondicionamiento de la infraestructura previa de PUMAGUA y los requerimientos para la duración de la batería.
- **Capítulo 4**
  Aquí se estudia de forma detallada el consumo energético que presenta el nodo diseñado y construido en el capítulo 3. El objetivo es estimar la vida útil de una batería en particular y, para lograr lo anterior, se analizó el comportamiento del nodo terminal por medio de la implementación de diversos métodos de medición de corriente, así como cálculos de consumo de energía eléctrica, cuya finalidad es obtener un resultado lo más cercano a la realidad.
- **Capítulo 5**
  Este capítulo consiste en pruebas del sistema completo, el interés de las pruebas del sistema es determinar la fiabilidad en las mediciones de consumo de agua, la resistencia a condiciones climáticas, la fiabilidad en la transmisión de paquetes y la consistencia entre el estudio de consumo energético y la vida útil de los nodos terminales. Para lograr lo descrito se diseñaron

pruebas en condiciones controladas y condiciones no controladas o reales.

- **Capítulo 6**

  Por último, el capítulo 6 enlista las conclusiones generales obtenidas de los resultados y las conclusiones más particulares del sistema, que permiten aceptar o rechazar la hipótesis propuesta. Por otro lado, también se abordan las áreas de oportunidad y el trabajo a futuro para la mejora del sistema, principalmente de cara a la optimización y análisis del consumo energético de los nodos terminales.

Capítulo 2

Marco teórico

2.1 Introducción

En este capítulo se presentan los fundamentos de las tecnologías LoRa y LoRaWAN, esencial para comprender el funcionamiento y capacidades de la red diseñada, describiendo sus características principales y ventajas en aplicaciones para comunicaciones de largo alcance y bajo consumo energético. Además de los fundamentos de LoRa y LoRaWAN, se revisaron y analizaron casos documentados de redes IoT para monitoreo de calidad y consumo de agua, que sirven como referencia y contexto sobre la importancia de estas aplicaciones y, sobre todo, de la propuesta de este trabajo.

Finalmente, puesto que uno de los objetivos principales de esta tesis es la estimación de la vida útil de los nodos terminales, se estudiaron diferentes trabajos y metodologías relacionados con la medición de corriente y la estimación de la vida útil de este tipo de dispositivos IoT, de esta manera, el marco teórico proporciona lo necesario para comprender los retos presentes en el desarrollo de redes IoT y los desafíos de cara al consumo energético en los dispositivos o nodos terminales que componen estas redes.

2.2 Tecnologías LoRa y LoRaWAN para el desarrollo de aplicaciones IoT

Antes que todo, es importante entender que LoRa y LoRaWAN son cosas diferentes, LoRa se trata de la tecnología encargada de la transmisión de datos por radiofrecuencia, la modulación con la que funciona proporciona ciertas características particulares que le dan la capacidad de cubrir grandes distancias y atravesar grandes obstáculos, LoRaWAN es algo similar a un software que controla la comunicación entre diferentes dispositivos a través de LoRa, dichas características y diferencias se detallarán más adelante en el desarrollo del marco teórico.

Estas tecnologías tienen gran relevancia en la actualidad ya que todos los dispositivos que las emplean han revolucionado las aplicaciones en el IoT, esto debido principalmente a sus características de bajo consumo de energía y largo alcance, lo que las hace ideales para aplicaciones de

monitoreo con una amplia posibilidad de sensores para zonas rurales o interiores, incluyendo agricultura inteligente, ciudades inteligentes, IoT industrial (IIoT), medio ambiente inteligente, hogares y edificios inteligentes, servicios públicos, así como logística y gestión de cadenas de suministros [22].

2.2.1 Internet de las Cosas

Aunque no existe una definición en concreto de lo que es el Internet de las Cosas, este término se refiere a escenarios en los que la conectividad de red y la capacidad de computación se extienden a objetos, sensores y artículos cotidianos que normalmente no se consideran computadoras. Esto permite que dichos dispositivos generen, intercambien y consuman datos con una mínima intervención humana [23].

El Internet de las Cosas es un paradigma que ha estado ganando terreno en el ámbito de las telecomunicaciones inalámbricas modernas. Según se menciona en [24], el concepto o la idea elemental del Internet de las Cosas es la presencia de una gran variedad de dispositivos con etiquetas de identificación por radiofrecuencia o Radio Frequency Identification (RFID), que cumplen la función de sensores o actuadores, con esquemas de direccionamiento único y la capacidad de interactuar entre sí y cooperar con sus vecinos para alcanzar objetivos comunes.

Las raíces del IoT se encuentran en el Instituto de Tecnología de Massachusetts (MIT), concretamente, en el trabajo que se realiza en el Auto-ID Center. Este grupo, fundado en 1999, trabajaba en el campo de la identificación por radiofrecuencia (RFID) en red y en el de las nuevas tecnologías de detección por sensores [2]. Algunos autores han clasificado la evolución del internet en cinco eras, de acuerdo con los usos y aplicaciones que han ido surgiendo a lo largo de su historia, culminando en el Internet de las Cosas [25].

1. El Internet de Documentos: bibliotecas electrónicas y páginas web basadas en documentos.
2. El Internet del Comercio: comercio electrónico, banca en línea y sitios de negociación de acciones.
3. El Internet de Aplicaciones: Web 2.0.
4. El Internet de las Personas: redes sociales.
5. El Internet de las Cosas: máquinas y dispositivos conectados.

La arquitectura en la que se basan las redes IoT debe garantizar el funcionamiento de todos los objetos interconectados, el diseño de dicha arquitectura debe considerar la escalabilidad e interoperabilidad entre dispositivos. Debido a que los objetos pueden moverse geográficamente y necesitan

interactuar en tiempo real, la arquitectura IoT debe ser adaptable para que los dispositivos interactúen dinámicamente con otros objetos y permitan la comunicación clara de eventos. Además, el IoT debe poseer una naturaleza descentralizada y heterogénea [1].

En IoT la arquitectura encargada de lograr las características anteriores se conoce como Service-oriented Architecture (SoA). Esta arquitectura proporciona las funcionalidades necesarias para la composición de servicios individuales ofrecidos por los objetos interconectados. Un aspecto importante para los servicios es contar con un repositorio de todas las instancias en servicio conectadas, las cuales se ejecutan de forma que puedan construir servicios compuestos [24].

La Figura 2.1 ilustra una arquitectura SoA genérica, la cual consta de cuatro capas, cada una con las siguientes funciones: [1]

1. **Capa de sensado:** está integrada con objetos de hardware disponibles para detectar el estado de las cosas.
2. **Capa de red:** es la infraestructura que soporta conexiones inalámbricas o cableadas entre los objetos.
3. **Capa de servicio:** crea y gestiona los servicios requeridos por los usuarios o aplicaciones.
4. **Capa de interfaces:** consiste en los métodos de interacción con los usuarios o aplicaciones.

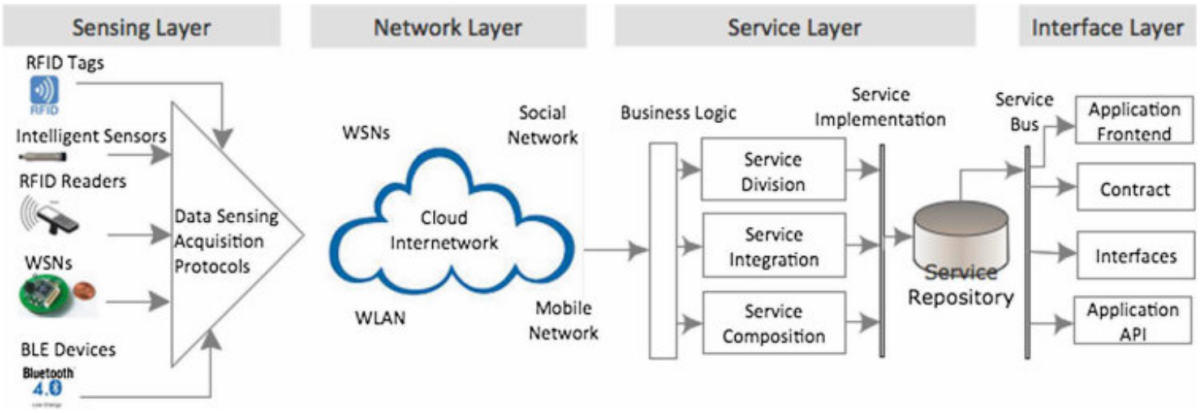


Figura 2.1: Arquitectura SoA para IoT [1].

Finalmente, una forma de darse cuenta del impacto que ha generado el Internet de las Cosas es observar la cantidad de objetos conectados a internet. En 2003, había aproximadamente 6,300 millones de personas en el planeta y 500 millones de dispositivos conectados a internet, por lo que el número de dispositivos conectados entre la población mundial da un aproximado de 0.08 dispositivos por persona. Con el crecimiento exponencial de los teléfonos inteligentes y tabletas, se

elevó el número de dispositivos conectados a internet a 12,500 millones en 2010, con una población mundial de 6,800 millones, lo que significa que el número de dispositivos conectados por persona era de 1.84, superando más de un dispositivo por persona por primera vez en la historia [2]. Lo anterior es sólo un ejemplo de cómo se ha expandido el uso de internet; no obstante, la figura 2.2 presenta estadísticas más actuales, donde se puede observar el gran crecimiento de los objetos conectados a internet.

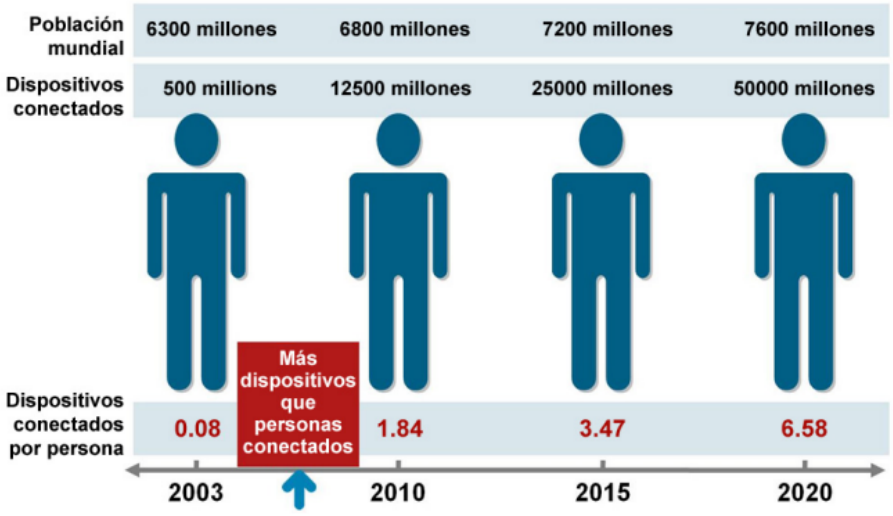


Figura 2.2: Crecimiento de los dispositivos conectados a internet [2].

2.2.2 LoRa

LoRa, por sus siglas en inglés, es una técnica de modulación de espectro ensanchado derivada de la tecnología de espectro ensanchado por barrido de frecuencia o Chirp Spread Spectrum (CSS). Originalmente desarrollada en 2009 por la compañía de origen francés Cycleo y después adquirida por la empresa estadounidense Semtech, LoRa es una plataforma inalámbrica de largo alcance y bajo consumo de energía que se ha convertido en la plataforma inalámbrica por defecto para aplicaciones IoT [26]. Los dispositivos y redes LoRa, como LoRaWAN, permiten aplicaciones IoT que resuelven algunos de los mayores desafíos que enfrenta el planeta: la gestión energética, la reducción de recursos naturales, el control de la contaminación, la eficiencia de infraestructuras y la prevención de desastres [22].

En cuanto a las frecuencias de operación, también llamadas planes de frecuencia, LoRa utiliza frecuencias por debajo de 1 GHz para el envío de datos. estas frecuencias se dividen en bandas dedicadas de acuerdo con las diferentes regiones establecidas alrededor del mundo, según lo muestra la tabla 2.1. Como se puede apreciar, no existe un ancho de banda específico en el caso de Méxi-

co, esto es principalmente debido a su cercanía con Estados unidos, de modo que ambos países comparten el mismo plan de frecuencia (US915) [19].

Región	**Frecuencia (MHz)**
Japón (AS923-1)	920 - 928
Sudeste Asiático (AS923-2)	923 - 925
Australia (AU915)	915 - 928
China (CN470)	470 - 510
China (CN779)	779 - 787
Europa (EU433)	433 - 434
Europa (EU868)	863 - 870
Corea del Sur (KR920)	920 - 923
India (IN865)	865 - 867
Estados Unidos (US915)	902 - 928
Rusia (RU864)	864 - 870

Tabla 2.1: Regiones y frecuencias establecidas para LoRa/LoRaWAN [19].

Una de las grandes ventajas de la tecnología LoRa es su largo rango de alcance, con posibilidad de cubrir distancias de hasta cinco kilómetros en áreas urbanas y quince kilómetros o más en zonas rurales con línea de vista, aspecto clave de las soluciones basadas en LoRa. Además, su consumo de energía ultra bajo, comparado con otras tecnologías inalámbricas, como Bluetooth, Wi-Fi o redes de telefonía celular 5G o 4g, permite la creación de dispositivos alimentados por baterías que pueden tener una vida útil de hasta 10 años [5].

Funcionamiento

De acuerdo con lo mencionado, LoRa funciona mediante la modulación de espectro ensanchado por barrido de frecuencia o (CSS), esta técnica de modulación es lo que le permite tener un largo alcance con un bajo consumo de potencia, además, facilita la instalación de redes fácilmente escalables, ya que múltiples dispositivos pueden comunicarse con mínima interferencia gracias a que se pueden distinguir eficientemente las señales mediante el uso de diferentes factores de propagación o Spread Factor (SF) [27]. Esta modulación ofrece velocidades de transmisión de datos ajustables sólo con ajustar el Spread Factor o el ancho de banda, en consecuencia, esta modulación permite un equilibrio entre capacidad, alcance y consumo de energía [4].

La modulación CSS es esencial para LoRa de acuerdo con lo ya mencionado, en resumen, esta modulación es clave básicamente porque permite la comunicación a larga distancia entre

dispositivos LoRaWAN con un bajo deterioro de la señal y una fuerte resistencia a las interferencias. Esta resiliencia es crucial en entornos urbanos cuando se utilizan varios dispositivos al mismo tiempo.[27] Además, el espectro extendido de chirp es resistente al efecto Doppler, que es típico en aplicaciones de radio móvil [28].

El elemento clave para la modulación CSS son, como su nombre lo indica, los *Chirps*, el término Chirp se refiere a una señal sinusoidal cuya frecuencia varía en el tiempo. cuando la variación en frecuencia es lineal, esta se puede llamar Chirp lineal o también modulación de frecuencia lineal [27] . La figura 2.3 ilustra los casos en que la frecuencia de la señal aumenta con el tiempo y cuando disminuye con el tiempo, estos casos se conocen como Chirp Up y Chirp Down, respectivamente.

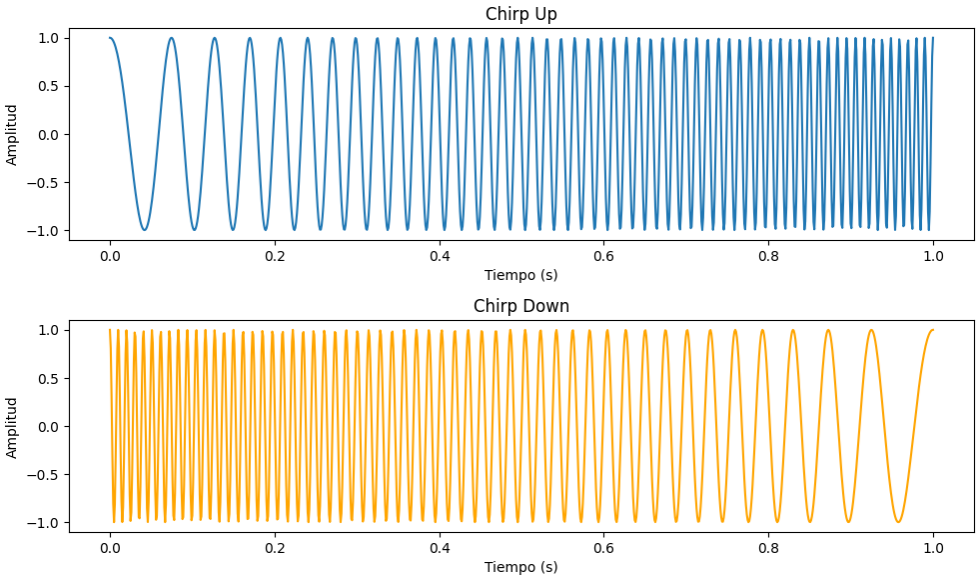


Figura 2.3: Chirp Up y Chirp Down.

En esencia, la modulación CSS consiste en representar o modular los datos a través de símbolos creados con estas señales, en las que la frecuencia de la señal aumenta (Chirp ascendente o Chirp Up) o disminuye (Chirp descendente o Chirp Down) con el tiempo a lo largo del ancho de banda o en inglés Band width (BW) [27]. Por lo tanto, en el envío de estos símbolos se ven involucrados dos parámetros:

1. La frecuencia inicial y final del Chirp, también llamado ancho de banda.
2. El sweep rate, que es la velocidad en que la frecuencia cambia, también se conoce como el spread factor.

Lo anterior se ilustra en el espectrograma de la figura 2.4, donde el eje 'Y' representa el ancho

de banda o las frecuencias que abarca el Chirp, mientras que el eje 'X' es el tiempo, los cambios en el SF se observan en el tiempo de duración de cada símbolo, yendo desde un SF de 7 hasta 12; esto implica que, cuando aumenta el SF, aumenta el tiempo en que se envía cada símbolo, por esto es que el aumentar el SF extiende el alcance de la comunicación LoRa a costa de una disminución en la velocidad de transmisión de datos [4].

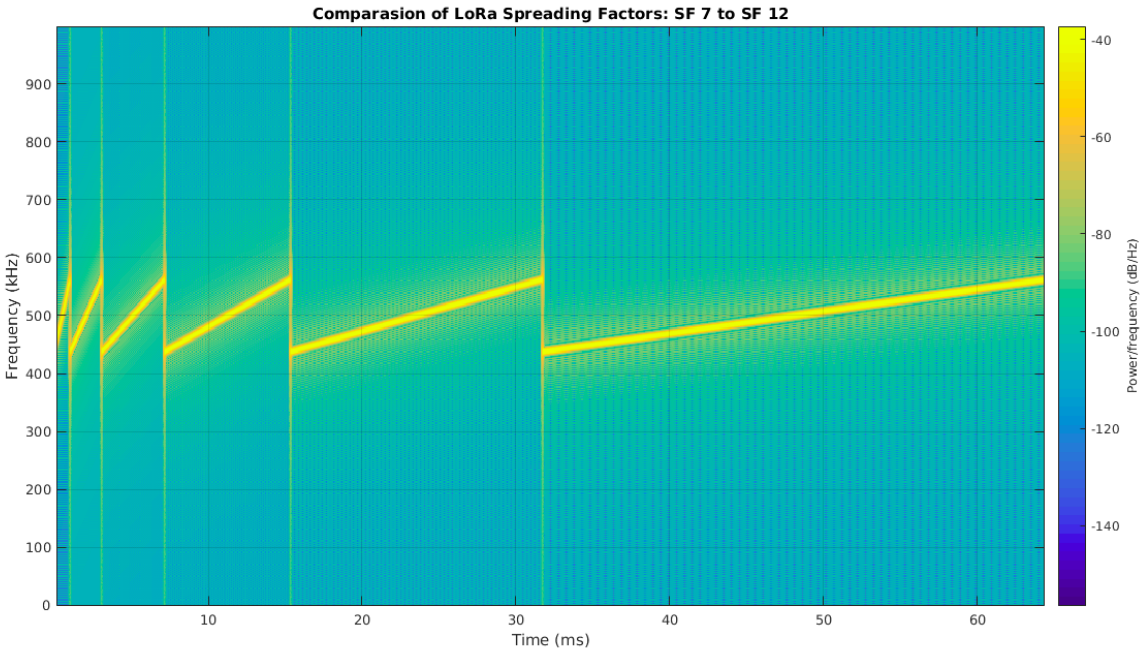
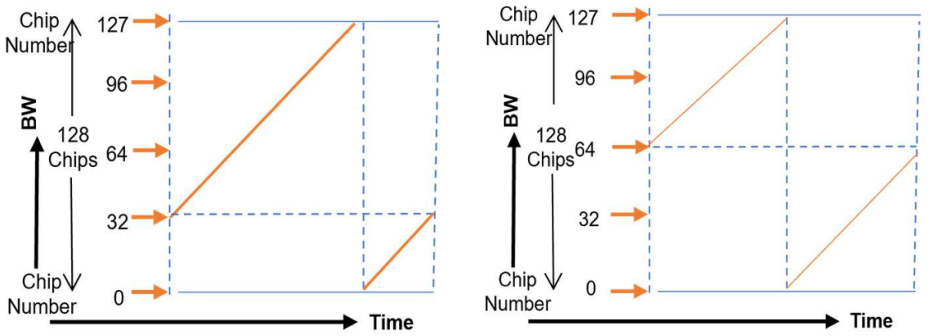


Figura 2.4: Espectrograma de símbolos LoRa para diferentes valores de SF [3].

Como ya se mencionó, la información se transmite por LoRa a través de símbolos, que a su vez están conformados por diferentes Chirps ascendentes o Chirps descendentes, por lo que para entender cómo se forman los diferentes símbolos se necesita introducir el concepto de *chip*. En [4] se define un chip como un pulso que comienza a barrer desde la frecuencia baja hasta la frecuencia alta, siendo que la suma de los chips más cercanos forma un símbolo. LoRa utiliza 6 valores de SF (desde SF 7 hasta SF 12) y 3 anchos de banda (125 kHz, 250 kHz y 500 kHz), por lo que existen 2^{SF} pares de símbolos posibles (128, 256, 512, 1024, 2048, 4096) que pueden transmitirse sobre el ancho de banda disponible [4], [3].

En las figuras 2.5a y 2.5b se ilustra cómo se forman diferentes símbolos (32 y 64) a partir de chirps ascendentes con un SF de 7, por lo que estos símbolos contienen 128 chips, cuya representación en binario es (0100000 y 1000000) [4].



(a) Chirp ascendente del símbolo = 32.

(b) Chirp ascendente del símbolo = 64.

Figura 2.5: Desplazamiento cíclico de los símbolos de información chirp ascendentes en formato decimal [4].

A partir de la creación de los símbolos, LoRa emplea tramas de datos formadas por diversos símbolos, que incluyen un preámbulo, una parte destinada a la sincronización y la carga últil o payload. Antes del denominado payload existen múltiples chiprs ascendentes que conforman el preámbulo, seguidos por dos señales de chirps descendentes correspondientes a la parte de sincronización, adicionalmente, después de los chirps descendentes, la trama puede contener un encabezado opcional con detalles de la tasa de bits, para finalmente dar paso al payload [4], [27]. Lo anterior descrito se observa en la figura 2.6

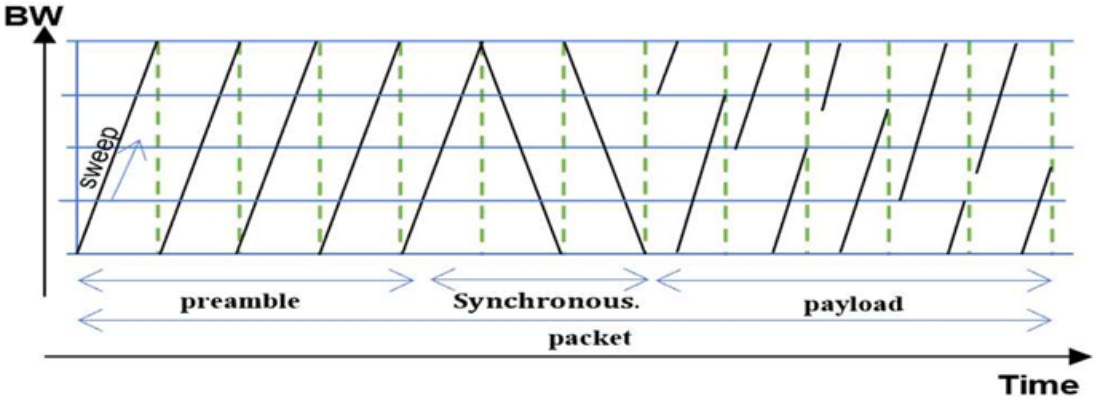


Figura 2.6: Estructura de la trama de una paquete LoRa [4].

2.2.3 LoRaWAN

En el mercado existen varias tecnologías Low Power Wide Area Network (LPWAN) como SigFox, NB-IoT o LoRaWAN. Las empresas de telecomunicaciones ofrecen NB-IoT como una alterna-

tiva de comunicación IoT a las tecnologías LPWAN sub-GHz, como NB-IoT opera en un espectro con licencia, ofrece una mayor confiabilidad del tráfico en comparación con otras tecnologías sub-GHz. A diferencia de SigFox y NB-IoT, LoRaWAN ofrece la posibilidad de implementaciones de redes privadas y una fácil integración con varias plataformas de red mundiales (por ejemplo, The Things Network (TTN)). Debido a esto y a sus especificaciones de acceso abierto, LoRaWAN atrajo la atención de la comunidad de investigación desde su primera aparición en el mercado [29].

En palabras de los creadores de LoRa, Semtech, LoRa es la capa física de LoRaWAN y LoRaWAN es un protocolo de red abierto que ofrece servicios de comunicación bidireccional segura, movilidad y localización estandarizados y mantenidos por LoRa Alliance [5], esto se visualiza en el stack de la figura 2.7.

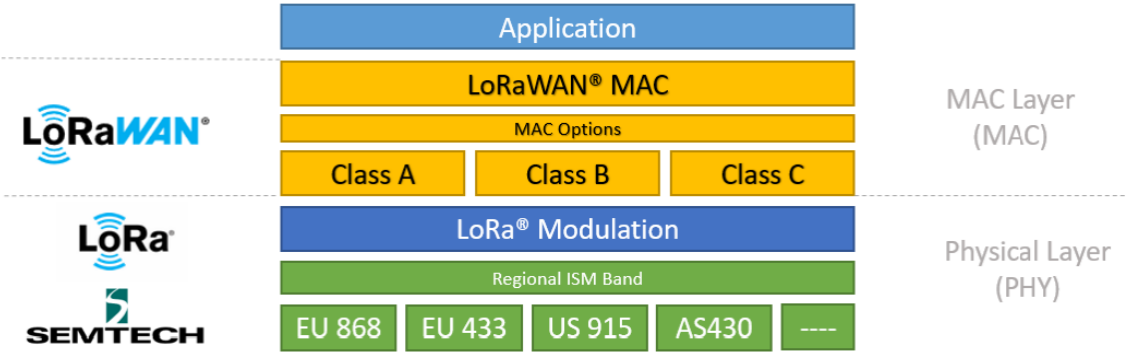


Figura 2.7: Stack LoRaWAN [5].

Básicamente, la relación entre LoRa y LoRaWAN, según lo definen [7] y [29], es que LoRaWAN es un protocolo de capa de control de acceso a medios o Media Access Control (MAC) que opera sobre LoRa y funciona como una capa de software que regula la utilización del hardware LoRa por parte de los dispositivos, dictando acciones como la transmisión y el formato de los mensajes entre dispositivos como nodos terminales y Gateways centrales. No se debe pasar por alto que las redes LoRaWAN están organizadas en una topología de estrella, en la que los Gateways transmiten mensajes entre los nodos terminales y un servidor de red central [30], por lo tanto, los Gateways son los responsables de encapsular los paquetes y reenviar los datos de los nodos terminales hacia el servidor de red hasta terminar en la aplicación del servidor [29]. Con lo anterior la arquitectura de red LoRaWAN resultante se muestra en la figura 2.8.

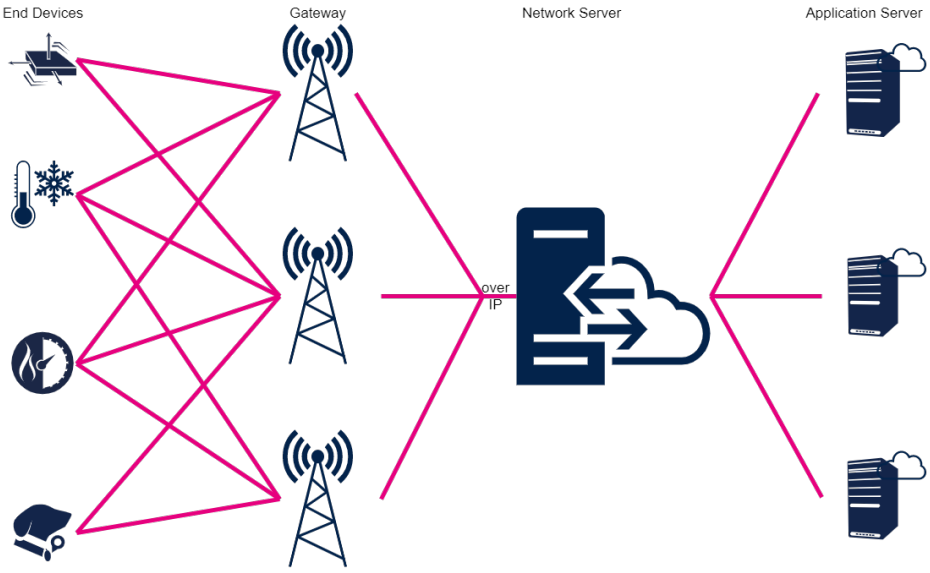


Figura 2.8: Topología de red LoRaWAN [6].

Clases de dispositivos

Volviendo a la figura 2.7, se observa que dentro del protocolo LoRaWAN están definidas las clases A, B y C; que, en resumen, definen la forma en la que los dispositivos o nodos terminales se comunican con el Gateway, esto también define la fiabilidad de los mensajes ya que para la comunicación bidireccional existen confirmaciones entre las recepciones de paquetes [30]. Cada clase tiene un comportamiento distinto que influye directamente en el consumo de energía del nodo terminal, por ello la importancia de entender las diferencias entre el funcionamiento de las clases A, B y C.

- Clase A: Retomando lo definido por el fabricante Semtech [5], la clase A es, por decirlo de alguna manera, la clase más básica, donde el nodo terminal permanece dormido hasta que un evento programado lo despierta para comenzar una transmisión (Tx), conocida como *uplink*. Finalizada la transmisión existe la posibilidad de que se presenten una o dos ventanas de recepción (Rx), llamadas *downlik*, que sirven como respuesta de confirmación del Gateway. Cada ventana de recepción está programada con un retardo de tiempo específico y únicamente existen después de que el nodo terminal inicia una transmisión, adicionalmente a las dos ventanas de recepción se pueden configurar algunos reintentos en la transmisión hasta que se reciba respuesta de confirmación del Gateway.

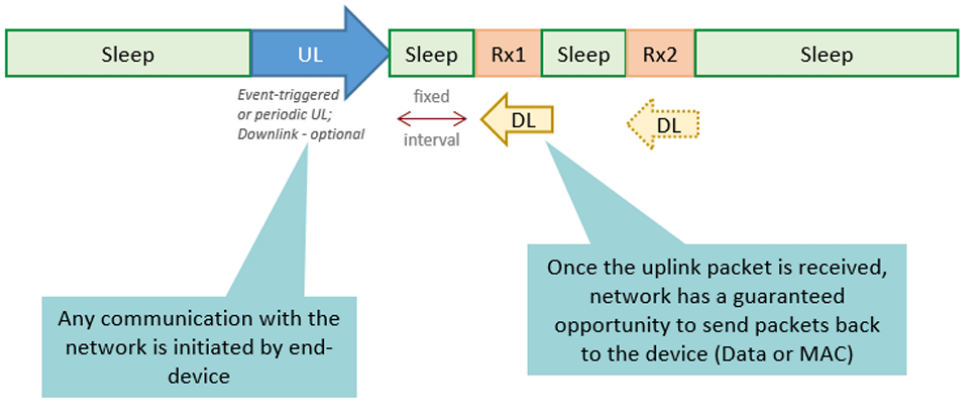


Figura 2.9: Funcionamiento de LoRaWAN clase A [5].

- Clase B: Conociendo la clase A de LoRaWAN resulta sencillo entender la clase B. Al igual que en el caso anterior, se recomienda observar la figura 2.10, que ilustra el funcionamiento de la clase B. La principal diferencia es que en esta clase los nodos terminales presentan ventanas de recepción de manera programada, ademas de las ventanas producidas por los eventos de transmisión programados, cabe mencionar que en las ventanas de recepción el servidor envía un mensaje al nodo terminal llamado *beacon*, cuyo objetivo es sincronizar la hora de los nodos con la del servidor [5], [29], [30].

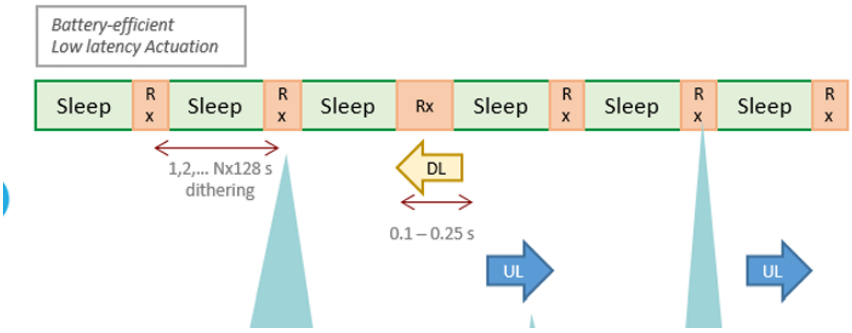


Figura 2.10: Funcionamiento de LoRaWAN clase B [5].

- Clase C: Finalmente, la clase C de LoRaWAN permite al servidor comunicarse de manera continua con el nodo terminal (figura 2.11), por lo que el nodo presenta ventanas de recepción casi todo el tiempo, con excepción de los eventos de transmisión [6].

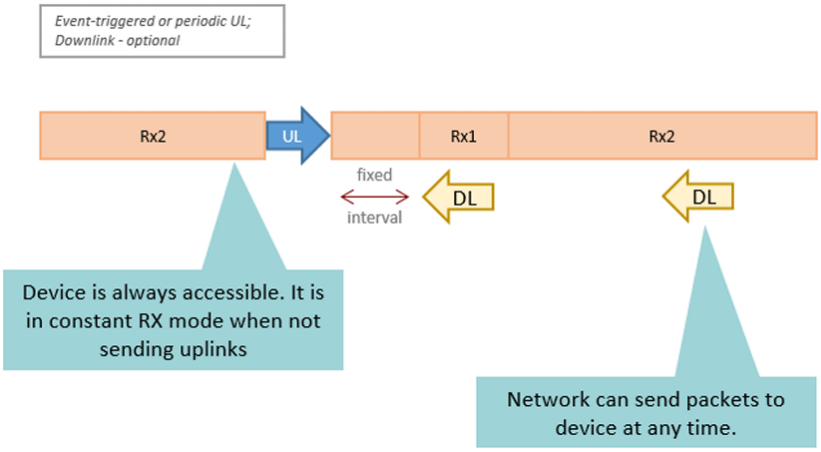


Figura 2.11: Funcionamiento de LoRaWAN clase C [5].

La razón de existir de las clases A, B y C de LoRaWAN es la funcionalidad que se necesita en los nodos terminales, ya que en esencia, los nodos terminales son sensores o actuadores [6]. Analizando las características de cada clase, se puede notar que una diferencia común entre las clases es la capacidad de recepción del nodo terminal, esta capacidad de recepción permite que los nodos terminales tengan la posibilidad de funcionar como actuadores a cambio de incrementar su consumo energético. La tabla 2.2 resume lo más destacado para entender cada clase y sus posibles aplicaciones.

Clase	**Aplicación**	**Características**
A	Sensores de bajo consumo y transmisión ocasional	▪ Modo más eficiente en energía. ▪ Solo recibe datos tras enviar un mensaje (*uplink*). ▪ Tiene hasta dos ventanas de recepción (*downlink*) tras cada transmisión. ▪ No recibe mensajes en otros momentos.
B	Sensores y actuadores con sincronización periódica y eventos programados	▪ Similar a la Clase A, pero con ventanas de recepción adicionales. ▪ Sincronización con el servidor mediante *beacons*. ▪ Mayor consumo energético en comparación con la Clase A.
C	Actuadores principales y dispositivos que requieren comunicación constante	▪ Recepción de mensajes casi continua, excepto al transmitir. ▪ Mayor consumo energético, ya que el receptor permanece activo. ▪ Baja latencia en la recepción de mensajes.

Tabla 2.2: Comparación de las Clases A, B y C en LoRaWAN.

Mecanismos de conexión a una red LoRaWAN.

Para que una red de LoRaWAN sea accesible únicamente entre los nodos terminales y Gateways propios de la red, los nodos terminales deben ser activados antes de poder enviar o recibir mensajes. Para ello, existen dos métodos diferentes, *Over the air activation (OTAA)* y *Activation by personalization (ABP)* [6], [7], [29].

- **Over the air activation:** El mecanismo OTAA para LoRaWAN 1.0 es el más seguro de los dos posibles mecanismos de conexión, lo que a su vez provoca que sea el más empleado. Su seguridad se basa en la creación de identificadores personalizados previamente definidos en el nodo terminal, para este mecanismo el nodo terminal manda una solicitud de *join request* con estos identificadores, después el nodo terminal abre una ventana de recepción con el objetivo de recibir el mensaje de *join accept* por parte del servidor que confirme su exitosa conexión a la red LoRaWAN [29], [31].
  Los identificadores usado por OTTA son los siguientes [31], [7]:
  - DevEui: Identificador global de dispositivo, consta de 64 bits en el espacio de direcciones IEEE EUI64 e identifica de forma única al nodo terminal.
  - AppEUI o JoinEUI: Identificador de aplicación único, consta de 64 bits para clasificar los dispositivos por aplicación.
  - AppKey: Es una clave secreta de 128 bits compartida entre el dispositivo terminal y la red.
  - DevNonce: Contador de 2 bytes, que comienza en 0 cuando el dispositivo se enciende y se incrementa con cada solicitud de *join request*.

  El procedimiento que sigue el mecanismo OTAA se puede observar en la figura 2.12

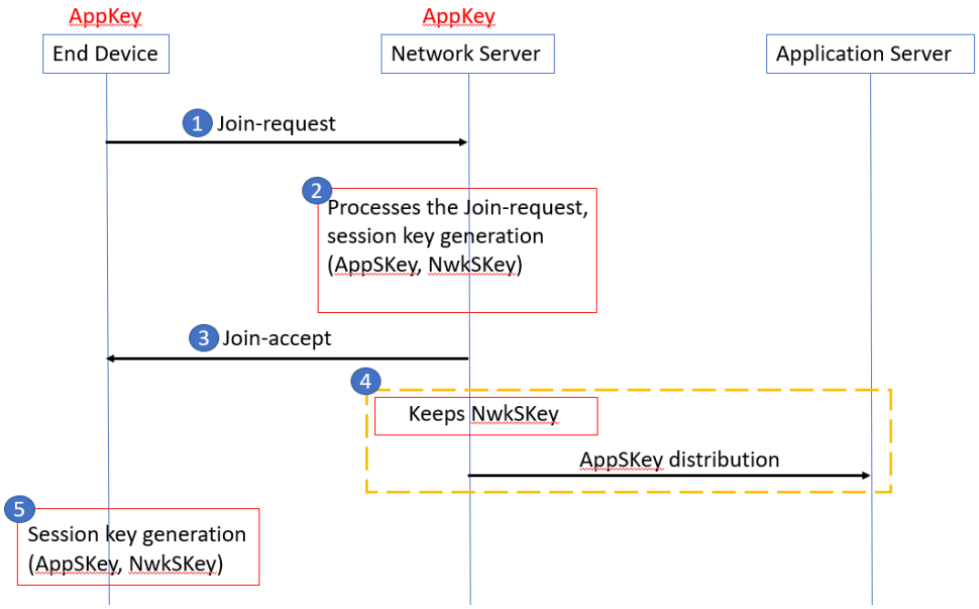


Figura 2.12: Mecanismo Over the Air Activation [7].

Se puede notar que durante el proceso de conexión se crean y comparten más claves o identificadores de inicio de sesión entre el servidor y el nodo terminal (network session key (NwkSKey), application session key (AppSKey)) [6], [7].

- **ABP:** Este es el método más simple para LoRaWAN 1.0 y, por lo tanto, el menos seguro; su simplicidad es gracias a que las claves NwkSKey y AppSKey están almacenadas en el nodo terminal, lo que permite que el nodo envíe datos directamente a la red LoRaWAN. Una desventaja es que los nodos terminales no pueden cambiar de servidor sin cambiar manualmente las claves en el dispositivo [6], [7].
  El procedimiento que sigue el mecanismo ABP se puede observar en la figura 2.13

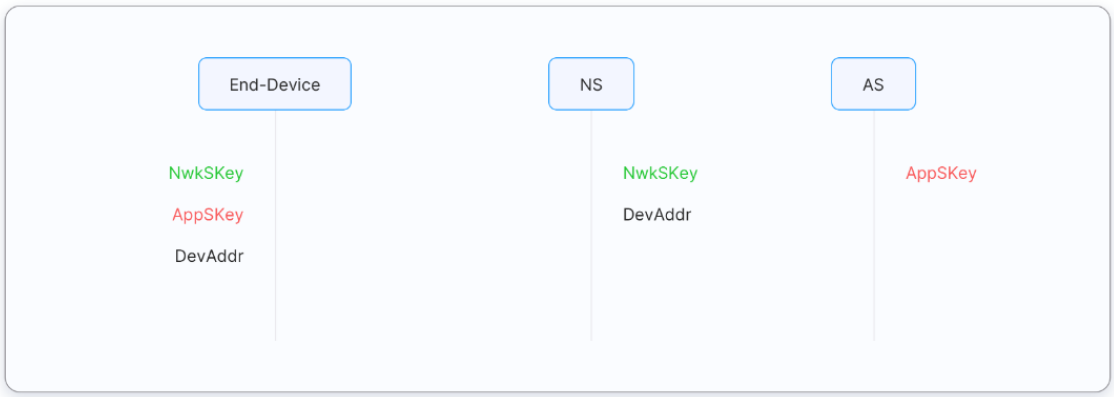


Figura 2.13: Mecanismo Activation by personalization [7].

Estructuras de paquetes LoRa y LoRaWAN.

Conocer de qué se componen los mensajes de LoRa es muy útil no sólo para comprender por completo que factores intervienen cuando se envían los mensajes, sino que es fundamental para determinar el tiempo en el aire de cada mensaje LoRa y posteriormente de LoRaWAN. La estructura de los mensajes se puede encontrar más detallada en sitios como [32], o en las especificaciones de LoRaWAN [33]. Este breve resumen de la tabla 2.3 sirve para aproximar la longitud de las tramas o símbolos que el nodo terminal envía en los procesos de *Join request*, transmisiones y ventanas de recepción, algunos eventos como el *join accept* o el *Join request* son muy similares a la ventana de recepción y transmisión, respectivamente, por lo que en esta sección no se profundizará en la composición de todas las tramas de LoRaWAN.

Campo	**Longitud / Descripción**
Preamble	8–65 535 símbolos "up-chirp" (configurable)
Sync Word	2 símbolos
PHDR (Header)	8 bits (modulación, CRC, `IF_CRC`)
PHDR CRC	16 bits
Payload	N bytes de datos
CRC de paquete	16 bits (opcional)

Tabla 2.3: Estructura de la trama física (PHY) de LoRa.

Con la información anterior, es importante aclarar la diferencia entre símbolos y bits o bytes, de acuerdo con la modulación LoRa y con base en los ejemplos de la figura 2.5, un símbolo puede representar diferentes valores dependiendo del spread factor, es decir, un símbolo puede representar 2^{SF} valores, de modo que esa representación de valores se puede traducir a bits como se muestra en la tabla 2.4.

Spread Factor (SF)	**Valores posibles**	**Bits por símbolo**
SF7	$2^7 = 128$	7
SF8	$2^8 = 256$	8
SF9	$2^9 = 512$	9
SF10	$2^{10} = 1024$	10
SF11	$2^{11} = 2048$	11
SF12	$2^{12} = 4096$	12

Tabla 2.4: Relación entre SF, número de símbolos y bits por símbolo en LoRa.

En cuanto a la estructura de los paquetes de subida (*uplink*) de LoRaWAN, estos comparten la estructura básica de los mensajes de LoRa PHY, con la diferencia de que en el payload se anexan todos los datos requeridos por LoRaWAN. Esta estructura se encuentra definida en la documentación de Semtech [8] y se resume en la tabla 2.5.

Campo	**Tamaño / Descripción**
MAC Header (MHDR)	
MHDR	1 byte (Tipo de mensaje: Unconfirmed/Confirmed Data Up)
MAC Payload	
FHDR (Frame Header)	Variable
DevAddr	4 bytes (Dirección del dispositivo)
FCtrl	1 byte (Octeto de control de trama)
FCnt	2 bytes (Contador de uplink)
FOpts	0–15 bytes (Comandos MAC "piggybacked")
FPort	1 byte (0 = comandos MAC en FRMPayload; 1–223 = datos de aplicación)
FRMPayload	0–N bytes (Datos de aplicación cifrados con AppSKey)
Message Integrity Code (MIC)	
MIC	4 bytes (Código de integridad con NwkSKey)
Detalle de FCtrl (Octeto de control)	
Bits	7 — 6 — 5 — 4 — 3–0
Significado	ADR — ADRACKReq — ACK — ClassB — FOptsLen

Tabla 2.5: Estructura jerárquica de la trama LoRaWAN basada en la trama del paquete.

La información anterior se ve plasmada en la figura 2.14, donde se descompone y aprecia en su totalidad la estructura de los mensajes LoRa y LoRaWAN, ya sea para *uplink* o *downlink*.

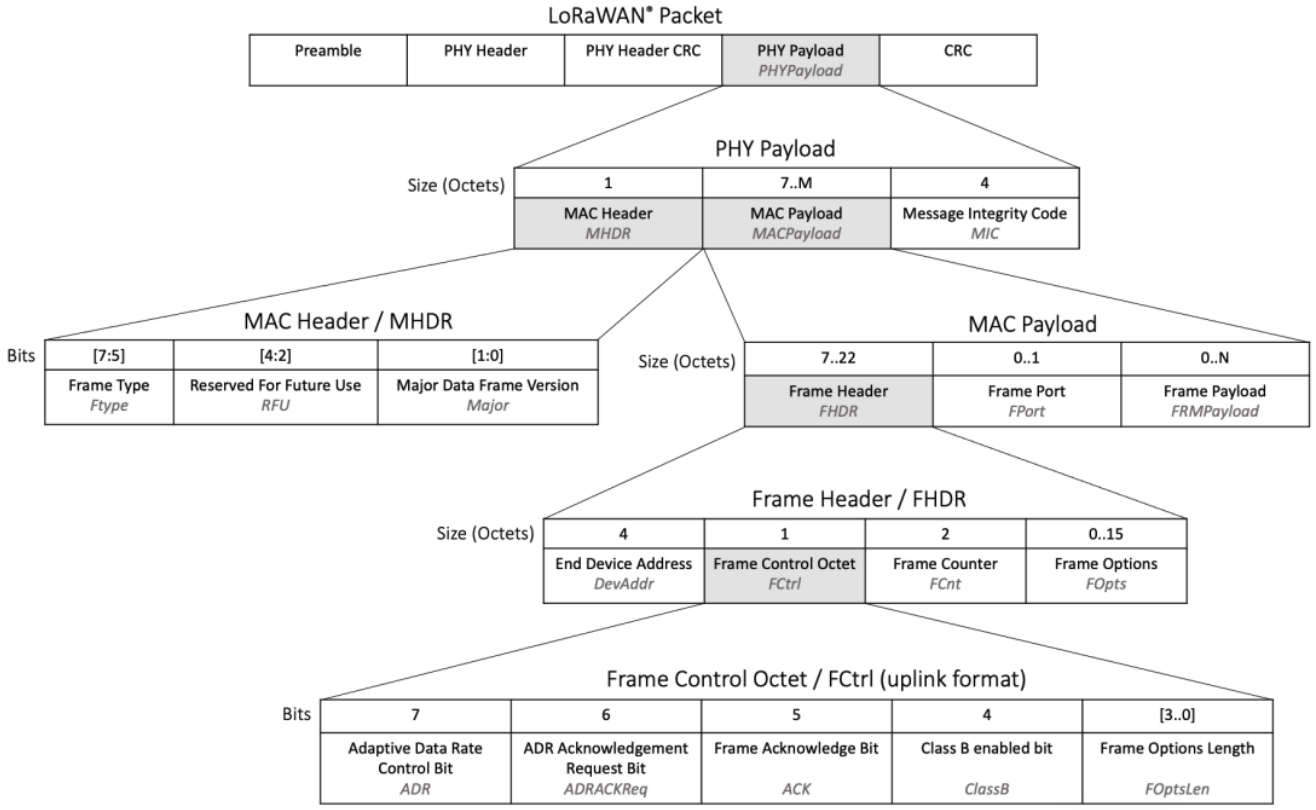


Figura 2.14: Estructura general de la trama de los paquetes LoRaWAN [8].

2.3 Casos previos de redes IoT/LPWAN para el monitoreo de agua potable

Es bien sabido que el agua es un recurso elemental para la vida humana y en general, para todos los organismos vivos, la gran demanda de agua provocada por la urbanización y la industrialización de las ciudades ha generado preocupación en la sociedad, tanto por garantizar el suministro, como la calidad de la misma. Esta problemática ha dado paso al desarrollo de fuentes de suministro alternas, como lo son las aguas subterráneas y superficiales, no obstante, se prevee que la escasez de agua empeore en muchos países en desarrollo [34], [35].

Con el contexto de la importancia del agua para la vida en la tierra, y lo ya mencionado en las secciones 2.2.1, 2.2.2 y 2.2.3, resulta evidente que el uso de IoT, junto con la tecnología LoRa y el protocolo LoRaWAN, son una solución de gran peso para resolver las demandas de la problemática del agua.

2.3.1 Casos previos de monitoreo de calidad del agua

El almacenamiento de agua también provoca acumulación de residuos sólidos, ademas de sustancias no deseadas como óxido y metales provenientes de las paredes de tuberías antiguas, sedimentos y lodo de tuberías dañadas. Partiendo de lo anterior, para evitar la mala calidad del agua y la escasez en tiempos de sequía, la tendencia actual es desarrollar sistemas de monitoreo que gestionen el suministro y calidad del agua, estas tecnologías han sido bien aceptadas ya que tienen la capacidad de realizar dicho monitoreo de forma automática [34], [35].

En la literatura sobre IoT y el monitoreo de agua existen dos ramas de mayor relevancia, el monitoreo de la calidad del agua y el consumo del agua. Un ejemplo de monitoreo de la calidad del agua es el articulo llamado *"IoT-Based Water Monitoring Systems: A Systematic Review" [35]*, para este articulo se se revisaron publicaciones de sitios como IEEE Xplore, ScienceDirect, Scopus y Web of Science durante 5 años, dado como resultado que de 946 artículos revisados, solo 50 se centraron en el estudio de la calidad del agua con enfoque a la agricultura. Este artículo es muy enriquecedor debido a la gran cantidad de información resumida en él, abordando desde los métodos más tradicionales de monitoreo de la calidad del agua hasta modelos que usan inteligencia artificial y machine learning para el monitoreo en tiempo real.

2.3.2 Casos previos de monitoreo de consumo de agua potable

Un estudio más enfocado de cara al monitoreo del consumo de agua, como lo es esta tesis, es el artículo *"Water Monitoring System Embedded with Internet of Things (IoT) Device: A Review"* [34], de manera similar al artículo anterior, éste hace un resumen de muchos artículos centrados en el monitoreo de consumo de agua con IoT, particularmente, se hace énfasis en cómo se minimiza la intervención humana en la mayoría de los procedimientos de monitoreo, lo que da paso a algoritmos autónomos de monitoreo de consumo de agua en tiempo real.

Por otro lado, el artículo que lleva el nombre *"Smart water consumption measurement system for houses using IoT and cloud computing"* [36], se centra especialmente en el uso de mecanismos inteligentes de monitoreo en tiempo real, cuyo objetivo es detectar fugas de agua y, por ende, evitar el desperdicio de la misma. Resulta interesante que los autores aseguran un 100 % de precisión en el algoritmo de detección de fugas, el cual está basado en reglas, contexto histórico y ubicación del usuario, considerando 10 posibles escenarios de consumo de agua, que van desde el consumo normal hasta un consumo anómalo. Aunque este artículo tiene el mismo objetivo que la tesis, existe una gran diferencia en las tecnologías usadas, debido a su enfoque para hogares, el sistema desarrollado en el artículo utiliza Wi-Fi como tecnología inalámbrica de transmisión, haciendo gran contraste

con el uso de LoRa.

2.4 Casos previos de medición de corriente y estimación de la vida útil de dispositivos IoT

Considerando que gran parte del enfoque de esta tesis está en la estimación de la vida útil de las baterías, es fundamental tener contexto sobre los métodos y herramientas existentes para dicho objetivo. Dentro del modelado del consumo energético de dispositivos LoRaWAN existen trabajos como el de [37], [38] y [39], donde se desarrollan modelos matemáticos considerando los eventos producidos y/o definidos por LoRaWAN según la clase, es decir, los autores proponen un modelo de estimación de consumo energético basado en la duración y demanda de corriente de los envíos, recepciones de paquetes e incluso en las preparaciones de las tramas.

Los trabajos presentados tienen un enfoque teórico, ya que determinan el consumo de corriente a partir de hojas de datos, así como determinan la duración o tiempo en el aire de los eventos con base en los parámetros de modulación LoRa y la composición de sus tramas, por lo tanto, los modelos desarrollados para estimar el consumo energético de un dispositivo LoRaWAN se pueden resumir o describir como la suma de las potencias o energías consumidas en cada evento. Lo anterior es abordado con mayor detalle por [40] en su trabajo de tesis, pues, a partir de un modelo basado en el cálculo y sumatoria de las potencias consumidas, logra estimar y dar valores de tiempo de vida útil en el dispositivo estudiado.

Aunque estos trabajos son de gran utilidad para comprender teóricamente los factores a considerar en la estimación de consumo energético en dispositivos LoRaWAN, su limitante es que se quedan con puros resultados teóricos, por lo que algunos otro autores, como [41], decidieron estudiar experimentalmente el comportamiento de los dispositivos, esta vez con mediciones de corriente, similar a lo propuesto en este trabajo de tesis.

Estudios como el de [41] y [42] se enfocaron en medir con diferentes equipos la corriente que demandaba su dispositivo en particular, a partir de sus mediciones obtuvieron valores promedio de duración y corriente, que después sustituyeron en modelos similares a los ya descritos. La ventaja de este enfoque es que considera valores reales, por lo que deberían ser mejores aproximaciones, a pesar de ello, estos trabajos no profundizan en modelos o procedimientos más complejos que tomen en cuenta todas las variaciones de parámetros que puede existir en transmisiones y recepciones, por ello, la propuesta de esta tesis es implementar esta clase de procedimientos y tener estimaciones aun más exactas.

Capítulo 3

Diseño y configuración del sistema

3.1 Introducción

El diseño de la red LoRaWAN es un paso fundamental ya que en este se integran todos los componentes de hardware y software requeridos, el proceso abarca desde el acondicionamiento del medidor de flujo de agua y la construcción del nodo terminal hasta la configuración de este, el Gateway central y la aplicación web en la que se despliegan los datos, no hace falta mencionar que todo el proceso y las decisiones tomadas son críticas para que la red funcione de la manera más óptima.

En el capítulo se detalla todo el proceso de acondicionamiento del medidor de flujo de agua, abarcando desde la explicación de su funcionamiento y hasta la conexión y/o integración con el microcontrolador que servirá como nodo terminal, además, se explica la programación del nodo bajo LoRaWAN y con base en las especificaciones requeridas en el sistema. Al igual que se detalla el proceso llevado a cabo para la creación del nodo terminal, también se describen todas las configuraciones y aspectos clave que permiten el despliegue de la red.

3.2 Especificaciones del sistema

Las especificaciones del sistema están dadas principalmente por los requerimientos de una red LoRaWAN, es decir, a grandes rasgos se trata de los dispositivos o nodos terminales, el Gateway y la red de medidores de flujo de agua con la que cuenta PUMAGUA. En resumen, se optó por dividir las especificaciones por requerimientos, principalmente de hardware y posteriormente de funcionalidad.

Especificaciones de Hardware

- Medidor de flujo (proporcionado por PUMAGUA): Salida digital.
- Microcontrolador: Bajo consumo, entrada digital para pulsos, temporizador, chip LoRa, compatibilidad con LoRaWAN.
- Batería de alimentación: Batería de litio 3.7 V.

- Gateway LoRaWAN: Conectividad a internet (Ethernet o 4G), compatible con The Things Network, ChirpStack o Helium.

Funciones del Sistema

- Conteo acumulativo de pulsos provenientes del medidor de flujo.
- Transmisión periódica del consumo de agua (cada 15 min a 1 h, configurable).
- Almacenamiento y visualización en servidor.

3.2.1 Infraestructura de PUMAGUA

PUMAGUA cuenta con una red de más de doscientos nodos encargados de registrar el caudal de agua que fluye a través del sistema de tuberías que abastece con agua potable los bebederos de Ciudad Universitaria, específicamente en el conjunto sur de la Facultad de Ingeniería. Cada uno de estos nodos está compuesto por un medidor de flujo de agua o caudalímetro y un dispositivo electrónico que registra y transmite las señales provenientes del caudalímetro al Gateway central.

Tanto los caudalímetros como el dispositivo electrónico pertenecen a la empresa Badger Meter, no obstante, debido a que el dispositivo electrónico se encuentra inmerso en acrílico y no se piensa reutilizar o analizar más a fondo, este no será especificado en los siguientes apartados, como sí lo será el sensor de flujo de agua.

3.2.2 Medidor de flujo de agua

Dentro de Ciudad Universitaria se encuentran instalados medidores de flujo de agua llamados por la empresa *medidores de desplazamiento positivo Badger Meter Recordall (RCDL)*, la estructura interna general del medidor se presenta en la figura 3.1.

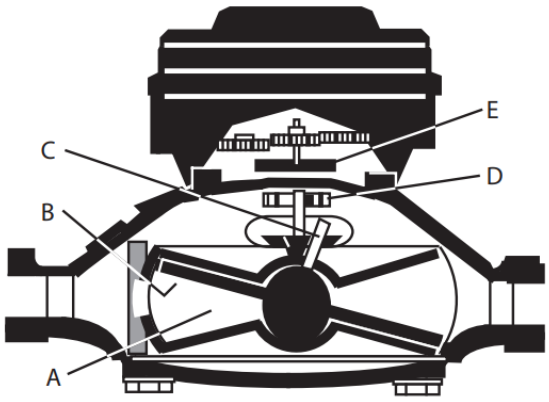


Figura 3.1: Estructura interna del medidor de flujo de agua [9].

- A: cámara del medidor
- B: disco
- C: husillo
- D: imán de accionamiento
- E: segundo imán o estilo variado de captación de sensor.

Como su nombre lo indica, el principio de funcionamiento de éstos es el desplazamiento positivo, el cual se basa en el llenado y descarga continua de la cámara de medición. Las separaciones entre el disco y la cámara producen la medición precisa por ciclo de volumen, es decir, a medida que el disco oscila, el eje central hace girar un imán que es detectado por medio de otro imán seguidor u otros sensores, por lo tanto, cada vuelta es equivalente a un volumen fijo de fluido.

El funcionamiento del medidor se puede resumir en los siguientes pasos [9]:

1. El líquido que fluye a través de la cámara del medidor (A) hace que un disco (B) oscile o se tambalee.
2. El movimiento del disco (b) genera la rotación de un husillo (C) y un imán de accionamiento (D).
3. La rotación se transmite a través de la pared del medidor a un segundo imán (E) encargado de registrar la rotación.

Dado que el medidor en especifico no puede dar una señal eléctrica directa, esta se obtiene con un registrador que básicamente realiza la captación de la rotación de los imanes por medio de caratulas que se acoplan mecánicamente al medidor, denominadas *Recordall Transmitter Register (RTR)* (mostrado en la figura 3.2). La empresa las define como un *transmisor de pulso/totalizador simple y económico diseñado para usarse con todos los medidores Recordall de las series Disc, Turbo y Compound*, es decir, es el encargado de registrar los datos del sensor y, por lo tanto, este fue conectado al microcontrolador del nodo terminal, encargado de registrar sus pulsos y enviar los datos al Gateway central por medio de LoRa.

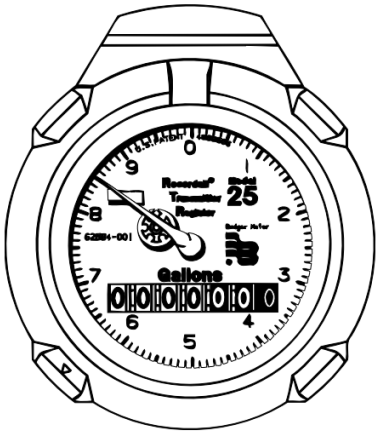


Figura 3.2: Caratula del medidor de flujo de agua [10].

Alimentado por una fuente externa, el RTR se alimenta con una tensión máxima de 30V DC a 1 mA (25°C). La salida del RTR representa 1/10 del círculo de prueba del medidor, lo que significa que cada vez que el medidor realiza un ciclo completo de prueba, el RTR genera una señal equivalente a 1/10 de ese ciclo. De acuerdo con el fabricante [10], la salida cuenta con las características de un transistor de efecto de campo (FET) de drenaje abierto. La condición de encendido es el cierre de un interruptor en estado sólido, mientras que la condición de apagado es un circuito abierto.

3.2.3 Microcontrolador del nodo terminal

Con base en investigaciones realizadas previamente para el desarrollo de esta red LoRa/LoRaWAN, el microcontrolador empleado es el Heltec HTCC-AB01_V2, mejor conocido como CubeCell, y puede ser conseguido como chip o como tarjeta de desarrollo. Para fines prácticos se escogió el uso de la tarjeta de desarrollo mostrada en la figura 3.3, cuyas especificaciones de interés se muestran en la tabla 3.1

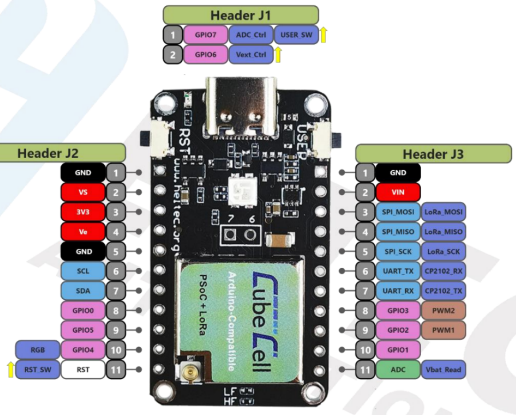


Figura 3.3: Microcontrolador CubeCell con chip LoRa [11].

Esta tarjeta de desarrollo es adecuada ya que cumple con las características para poder operar con LoRa y LoRaWAN en la región de Norteamérica, e incluso más regiones alrededor del mundo, de igual forma, el consumo de corriente en modo sueño o sleep es ideal para lograr que el consumo energético sea lo suficientemente bajo para que los nodos puedan funcionar el tiempo deseado. Otro factor clave es el uso del Framework de Arduino para su programación, aunque quizás no sea la mejor opción para proyectos complejos, sí es la opción más sencilla de usar considerando que la aplicación de monitoreo de flujo de agua no requiere una programación extremadamente compleja, o a bajo nivel, dadas las librerías proporcionadas por el fabricante.

Características	**Especificaciones**
Chip principal	ASR6501 (ARM Cortex M0+ MCU + SX1262 LoRa)
Frecuencia LoRa	433 MHz / 470 MHz / 868 MHz / 915 MHz
Conectividad LoRaWAN	clase A y clase C
RAM	64 KB (SRAM)
Flash	128 KB
Potencia de transmisión máxima	21 ± 1 dBm
Consumo en modo sueño (Sleep)	3.5 µA
Perifericos	1*SPI; 1*I2C; 1*UART; 1*12-bit ADC; 1*SWD; 8*GPIO; 2*PWM; 8-Channel DMA engine
Interfaz de programación	Arduino IDE
Voltaje de operación	3.3 - 5 V

Tabla 3.1: Características del CubeCell HTCC-AB01_V2 [11].

Consumo de energía

Dado el enfoque de una aplicación de bajo consumo para esta tesis, resulta crucial conocer los consumos de energía que el nodo terminal requiere en sus diferentes etapas de ciclo de trabajo correspondientes al protocolo LoRaWAN, específicamente en su clase A. El consumo que el nodo terminal presenta en las etapas del ciclo de trabajo está determinado por dos componentes esenciales: *el chip LoRa* y *el microprocesador*, cada uno de ellos presenta un consumo en específico, de modo que, a continuación, se presenta la recompilación de especificaciones proporcionadas por los fabricantes.

CubeCell HTCC-AB01_V2

Como primer valor de referencia se tiene la tabla 3.2 donde el fabricante Heltec proporciona una pequeña lista de consumos en los diferentes modos en los que la tarjeta de desarrollo opera.

Modo	**Condición**	**Consumo típico**
TX	+14dBm, alimentado por USB	150 mA
	+17dBm, alimentado por USB	170 mA
	+22dBm, alimentado por USB	185 mA
RX	Recepción continua	30 mA
Reposo	Alimentado por USB	10 mA
	Alimentado por batería (VBAT)	11 µA
	Alimentado por cabezal de 3.3V	3.5 µA

Tabla 3.2: Consumos típicos de corriente del CubeCell HTCC-AB01_V2 [11].

Esta tabla de consumos, teóricamente, es la más adecuada ya que es la proporcionada específicamente para la tarjeta de desarrollo, por lo que debe considerar el consumo de todos los demás componentes que constituyen dicha tarjeta, no obstante, estos valores son comparados con las hojas de datos de los fabricantes de los chips empleados en la tarjeta.

Chip LoRa sx1262

Se considera analizar las especificaciones de consumo de corriente del chip LoRa, ya que es el encargado de la transmisión y recepción, eventos donde se presentan los mayores consumos por parte del nodo terminal. De manera análoga, la tabla 3.3 contiene los valores de consumo de corriente de importancia, con la ventaja de que el fabricante (Semtech) proporciona una mayor extensión de valores bajo diferentes características.

Modo	Condición	Consumo Típico
Reposo	Modo SLEEP	160 nA
	Configuración retenida	600 nA
	Configuración retenida	1.2 μA
RX	Recepción continua	10.1 mA
TX	+14dBm	90 mA
	+17dBm	95 mA
	+20dBm	102 mA
	+22dBm	118 mA

Tabla 3.3: Consumos típicos de corriente por modo de operación del chip SX1262 [20].

Una breve comparativa entre las tablas 3.2 y 3.3 muestra una diferencia considerable en los modos de transmisión y recepción, esto será motivo de un pequeño análisis, considerando el consumo del chip principal que conforma al CubeCell, que se presenta en el siguiente apartado.

Chip ASR6501

Considerar los consumo de la hoja de datos del fabricante del chip principal es otra gran referencia debido a que se incluye el consumo en conjunto del chip LoRa junto con el microprocesador que se encarga de manejar el chip LoRa, que además, agrega un consumo extra. Teniendo en cuenta las especificaciones de estas tres hojas de datos se puede llegar a un consenso en lo que podrían ser los consumos reales medidos en el próximo capitulo. Por lo tanto, los valores resumidos de corriente que el fabricante del chip ASR6501 proporciona se ven reflejados en la tabla 3.4.

Modo	Condición	Consumo Típico
Reposo	Sin retención RF ni RTC	2 μA
	Sin retención RF, con RTC	2.7 μA
	Con retención RF y RTC	3.1 μA
RX	Recepción continua	10 mA
TX	+22 dBm	108 mA
	+21 dBm	106 mA
	+20 dBm	98 mA
	+17 dBm	90 mA
	+14 dBm	78 mA
	+10 dBm	59 mA
	+5 dBm	47 mA

Tabla 3.4: Consumos típicos de corriente por modo de operación del chip ASR6501 [21].

Comparación de consumos entre chips

Esta breve comparación se resume en la tabla 3.5 y su objetivo es proporcionar un panorama claro sobre el comportamiento de consumos que tendrá el nodo terminal. En esta se comparan los valores de corriente en los modos de reposo, recepción (RX) y transmisión (TX) de los dispositivos ASR6501, SX1262 y CubeCell.

Modo	**Condición**	**ASR6501**	**SX1262**	**CubeCell**
Reposo	Configuración mínima	2 μA	160 nA	-
	Retención parcial	2.7 μA	600 nA	-
	Retención completa	3.1 μA	1.2 μA	3.5 μA
RX	Recepción continua	10 mA	10.1 mA	30 mA
TX	14 dBm	78 mA	90 mA	150 mA
	17 dBm	90 mA	95 mA	170 mA
	22 dBm	108 mA	118 mA	185 mA

Tabla 3.5: Comparativa de consumos de los módulos LoRa según el fabricante.

De la Tabla 3.5 se puede concluir que el SX1262 presenta el menor consumo en modo de reposo, lo que lo hace más adecuado para aplicaciones de ultra bajo consumo y operación a largo plazo, no obstante, en la transmisión presenta un consumo más elevado respecto al chip ASR6501.

La comparativa es complicada ya que hay una diferencia considerable entre chips, esto no debería ser así ya que en esencia el ASR6501 y el CubeCell son la combinación del SX1262 con el microprocesador, esto puede deberse al diseño interno, la gestión de energía o incluso el firmware del CubeCell para el ASR6501.

Con lo anterior, los valores mostrados deben considerarse como referencia, mientras que para determinar los valores reales harán mediciones experimentales en los diferentes modos, las cuales serán abordadas en el siguiente capítulo.

3.2.4 Gateway central

Retomando el marco teórico, un Gateway central en un sistema IoT actúa como un maestro o puente entre los nodos terminales y el servidor web junto con la aplicación. Es un componente fundamental que permite gestionar, procesar y transmitir los datos de manera eficiente y segura, por lo que sus principales funciones son:

- Agrupación de datos: Reúne la información de múltiples dispositivos IoT, reduciendo el tráfico de red y simplificando la gestión.
- Preprocesamiento de datos: Realiza tareas básicas de procesamiento de datos, como filtrado,

agregación o transformación, antes de enviarlos a la nube. Esto reduce la carga computacional en la nube y mejora la eficiencia.

- Protocolos de traducción: Convierte los protocolos de comunicación utilizados por los diferentes dispositivos IoT en un formato estándar para facilitar la comunicación con la nube.
- Gestión de energía: Optimiza el consumo de energía de los dispositivos IoT, especialmente aquellos con baterías limitadas.
- Seguridad: Implementa medidas de seguridad para proteger los datos y los dispositivos de ataques cibernéticos.
- Conectividad: Proporciona conectividad a dispositivos que no tienen conexión directa a internet, como aquellos que utilizan tecnologías de corto alcance, como Zigbee o Bluetooth.
- Escalabilidad: Permite agregar nuevos dispositivos al sistema de manera sencilla y escalable.

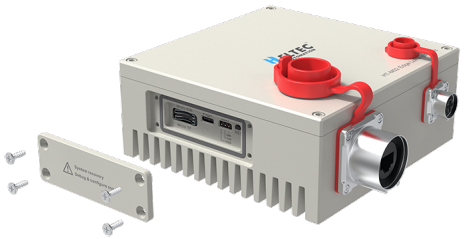


Figura 3.4: Gateway central Heltec HT-M02 [12].

El **HT-M02** (véase la figura 3.4) es un Gateway de LoRaWAN desarrollado por Heltec Automation (R), optimizado para redes de baja potencia y largo alcance *Low Power Wide Area Networks* (LPWAN) y está diseñado para facilitar la comunicación entre nodos LoRa y la red de internet, actuando como puente entre dispositivos distribuidos y los servidores de aplicaciones.

La elección de este Gateway central se debe a las características mostradas en la tabla 3.6, pues ofrece gran versatilidad para una gran variedad de aplicaciones, incluyendo aquellas que involucren Wi-Fi y redes de telefonía celular 4G. Otra de las características que lo hacen superior a otras alternativas es que posee 1 GB de memoria RAM y, finalmente, su capacidad para poder ser instalado en exteriores lo hace la mejor alternativa del mercado para el Gateway central.

Característica	Descripción
Frecuencias soportadas	EU868, US915, AS923, AU915, CN470, IN865, KR920
Capacidad de Conexión	Hasta 10,000 nodos finales
Procesador Principal	ARM Cortex-A53, 64 bits a 1.2 GHz
Almacenamiento y RAM	8 GB de almacenamiento, 1 GB de memoria RAM
Conectividad	WiFi 2.4GHz (802.11 b/g/n), Ethernet Gigabit RJ45, USB 2.0
Potencia de Transmisión	Hasta 27 dBm (500 mW), configurable
Sistema Operativo	Basado en Linux (Ubuntu Core)
Compatibilidad	Compatible con The Things Network (TTN), ChirpStack, entre otras plataformas LoRaWAN
Alimentación	PoE (*Power over Ethernet*) o 12V DC
Dimensiones y Peso	115 x 115 x 30 mm, 450 g
Rango de Temperatura Operativa	-40°C a 85°C
Certificaciones	CE, FCC, RoHS

Tabla 3.6: Características del Gateway HT-M02 [12].

3.3 Acondicionamiento del medidor de flujo de agua

De acuerdo con las especificaciones del medidor de flujo de agua, éste puede ser alimentado con un voltaje máximo de 30 V, por lo que el voltaje de las baterías no es un inconveniente y, por lo tanto, este puede ser conectado y acoplado de una manera relativamente sencilla al microcontrolador.

Puesto que la señal de las carátulas registradoras son en esencia cortos circuitos debido al FET con drenaje abierto, el medidor puede ser acoplado al microcontrolador del nodo terminal como si fuera un interruptor, esto significa que existen dos opciones y/o formas de conexión a las entradas digitales del microcontrolador: con resistencia de pull-up y pull-down. Usar alguna de estas configuraciones es de suma importancia debido a que definen un valor o estado por defecto en la entrada del microcontrolador mientras no hay señal, esto es útil para evitar estados indefinidos y errores en el registro de pulsos.

- Resistencia Pull-Up: Esta resistencia conecta la entrada digital al voltaje de alimentación (Vcc), esto implica que siempre habra un valor lógico de 1 cuando no haya señal.
- Resistencia Pull-Down: Esta resistencia conecta la entrada digital a tierra (GND), esto implica que siempre habrá un valor lógico de 0 cuando se recibe la señal del pulso.

Analizando las opciones, la resistencia de pull-up, al estar conectada a Vcc, implica que siempre

habrá una corriente circulando por dicha resistencia. Por otro lado, en la resistencia de pull-down no circula ninguna corriente por la resistencia, ya que está conectada a GND, esto provoca que solo habrá consumo de corriente cuando llegue la señal del medidor.

Con base en lo anterior, y teniendo en cuenta que se busca minimizar el consumo, la opción más acertada es usar la resistencia de pull-down. El diagrama de conexión se presenta en la figura 3.5.

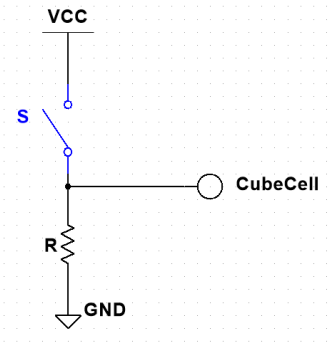


Figura 3.5: Conexión de resistencia pull-down.

Para entender un poco más, R es la resistencia de pull-down que conecta la entrada digital del microcontrolador del nodo terminal, en este caso el CubeCell, a tierra o GND; S es un interruptor que representa el registrador del medidor de flujo de agua, una terminal (+) se conecta la fuente de alimentación y la otra (-) a la entrada digital del CubeCell, de este modo, cuando el medidor mande un pulso el interruptor se cerrará, provocando que el voltaje de alimentación llegue a la entrada digital del microcontrolador.

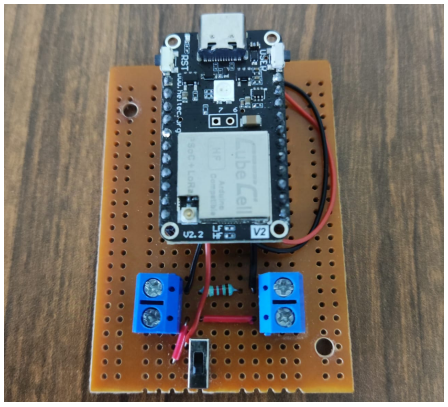
El cálculo de la resistencia de pull-down depende de la corriente que se necesite en la carga conectada, en este caso, la carga es la entrada digital del microcontrolador. Las entradas digitales o GPIOs de los microcontroladores por lo general demandan muy poca corriente (en el orden de μA) cuando realizan una lectura debido a su alta impedancia, esta corriente se conoce como corriente de fuga, entonces, la resistencia se puede calcular en función de la corriente de fuga y el voltaje al que está conectada. El chip del CubeCell que contiene la parte de LoRa y la unidad de procesamiento es el ASR6501, las hojas de datos de este chip indican que tiene resistencias internas de pull-up y pull-down programables con un valor típico de $45[k\Omega]$, con un voltaje típico de $3.3[V]$, por lo tanto, con 3.1 se puede determinar que la corriente de fuga para el CubeCell es de $73.3[\mu A]$.

$$I = \frac{Vcc}{R} = \frac{3.3}{45000} \approx 73.3[\mu A] R = \frac{Vcc}{I} = \frac{3.3}{100x10^{-9}} = 37[M\Omega] \tag{3.1}$$

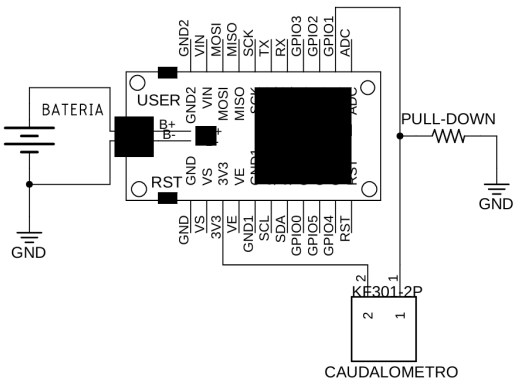
3.3.1 Construcción del circuito

Partiendo de los cálculos para el acondicionamiento del medidor de flujo de agua y con la finalidad de tener mayor robustez en el sistema, se decidió no utilizar la resistencia interna de pull-down programable, sino, que se agrega una resistencia de pull-down externa que garantiza la conexión ante posibles errores de software, además de que ofrece protección adicional al microcontrolador.

Aunque el valor teórico la resistencia de pull-down es de 37$[M\Omega]$, este valor es muy grande, por lo que no existe un valor comercial, por otro lado, entre mayor es el valor de resistencia, mayor será el ruido electrónico, lo que podría generar niveles de tensión flotante poco confiables y bajas velocidades de respuesta [43]. Para mitigar los efectos negativos de las resistencias de valor elevado, se recomienda el uso de resistencias de pull-up y pull-down con valores de entre 1 y 10$[k\Omega]$, valores que ayudan a reducir la velocidad de respuesta y ruido; de este modo, para la aplicación, que prioriza el consumo energético, se decidió usar un valor de 10$[k\Omega]$. Con lo anterior, se construyó el circuito que se muestra en la figura 3.6a, correspondiente al esquema de conexión de la figura 3.6b.



(a) Construcción en placa fenólica perforada.



(b) Diagrama de conexión.

Figura 3.6: Construcción del circuito para el nodo terminal.

Una vez realizada la conexión, el siguiente paso es configurar el IDE para programar las funciones de conteo pulsos del medidor, lecturas del voltaje de la batería que lo alimenta y la comunicación a través de LoRaWAN.

3.4 Configuración y operación del nodo terminal

En este apartado se abordan todos los aspectos necesarios para la programación del nodo terminal, esto se hace con base en la información que proporciona el fabricante. Dentro de los

puntos que se verán en esta sección, los más importantes son: programación de las variables que se medirán (pulsos del sensor de flujo de agua y voltaje de la batería) y la integración de dichas variables con LoRa y LoRaWAN.

3.4.1 Configuración del entorno de programación para el CubeCell

El microcontrolador del nodo terminal, como ya se mencionó, se programa mediante el framework de Arduino. La configuración del editor Arduino IDE para el CubeCell se realiza a través de los siguientes pasos:

1. Descargar el driver que permite la comunicación serial entre el CubeCell y la computadora, el CubeCell utiliza el chip CP2102 como medio de comunicación entre la computadora y el chip central del microcontrolador, este driver se puede descargar en el sitio oficial de descargas de Silicon Labs.
2. Ingresar el siguiente enlace: `https://github.com/HelTecAutomation/CubeCell-Arduino/releases/download/V1.5.0/package_CubeCell_index.json` en la opción de administrador de tarjetas adicionales que se encuentra en la barra de herramientas, el apartado de *Archivo* > *Preferencias*. Lo anterior se ilustra en la figura 3.7.

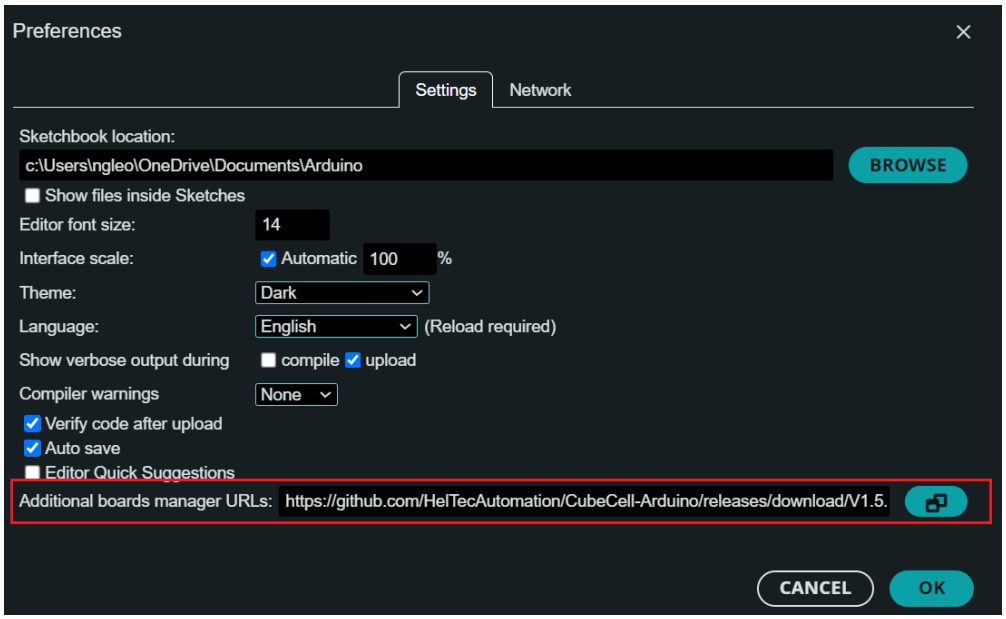


Figura 3.7: Registro del CubeCell en las preferencias de tarjetas de Arduino.

3. Después de haber ingresado el enlace, lo siguiente es descargar e instalar las librerías/drivers necesarios para la tarjeta a través del administrador de tarjetas, esto se logra buscando *CubeCell* en dicho administrador, según se observa en la figura 3.8.

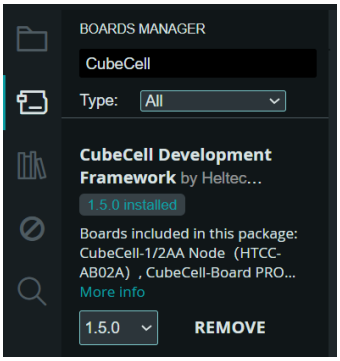


Figura 3.8: Descarga del framework del CubeCell para Arduino.

4. Para poder usar la tarjeta, es necesario ir al menú de *herramientas* y seleccionar el modelo de la placa en específico, de igual manera el puerto COM detectado por la computadora (véase la figura 3.9).

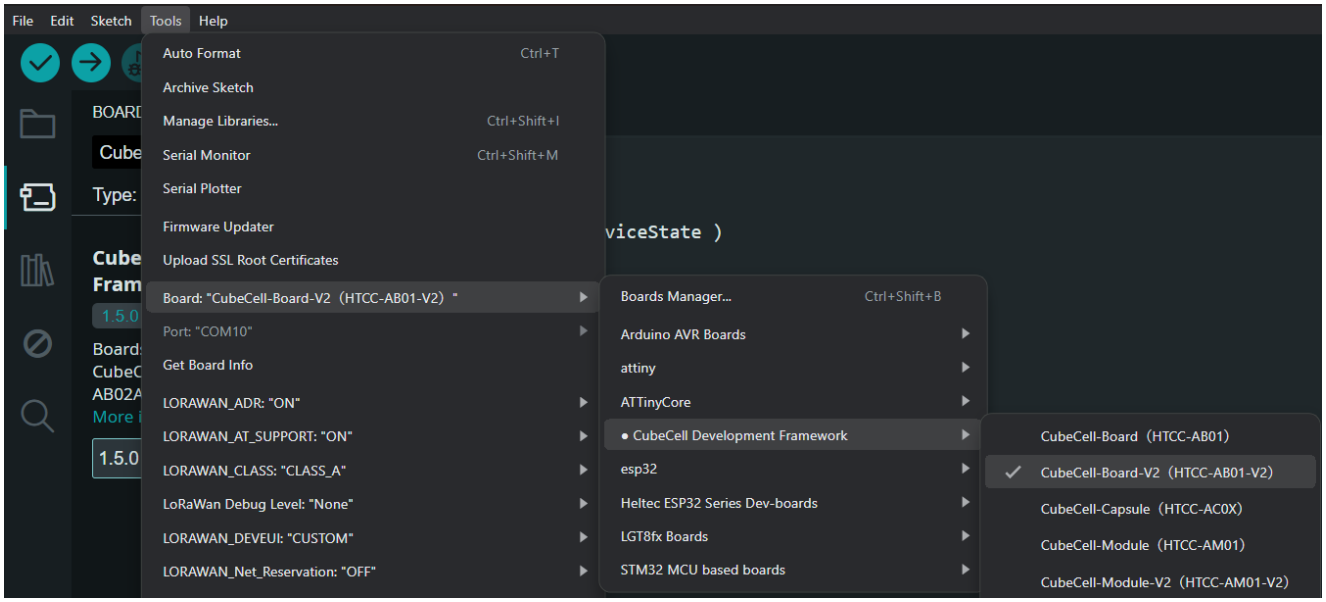


Figura 3.9: Selección del modelo de tarjeta.

5. Elegir un programa de ejemplo para verificar el correcto funcionamiento del entorno de programación, estos se encuentran en la barra de herramientas en el menú de *Archivo* > *Ejemplos* y en el apartado de *Ejemplos* aparece la sección exclusiva para el CubeCell (figura 3.10).

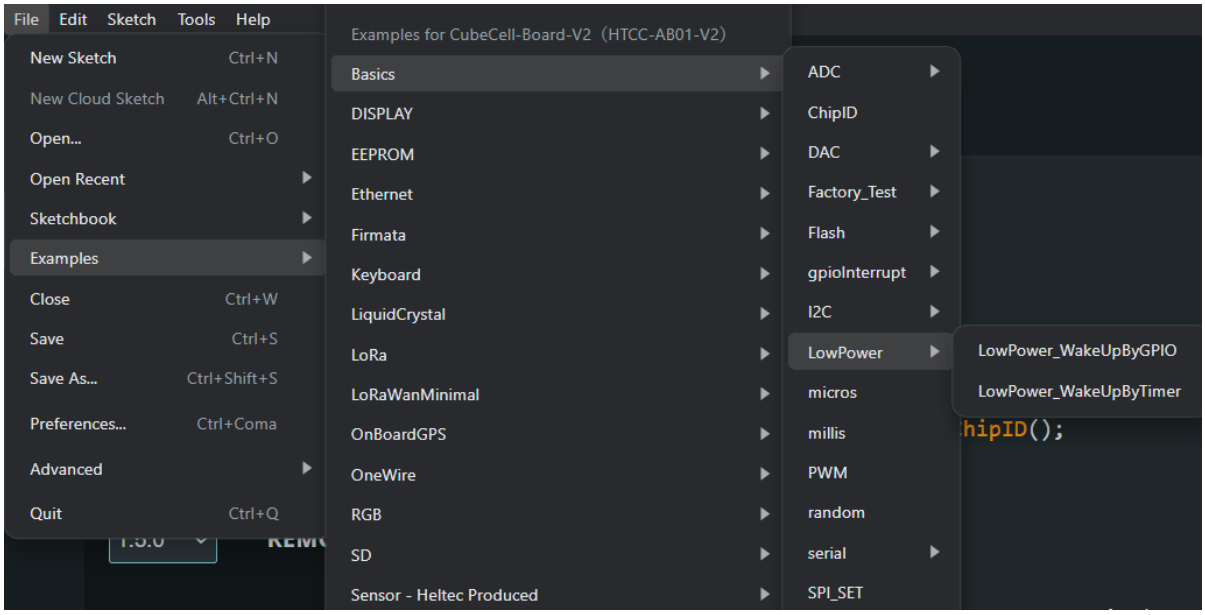


Figura 3.10: Programas ejemplo para el CubeCell.

6. Después de abrir el ejemplo, simplemente se carga el programa con el botón de la flecha que apunta hacia la derecha (ubicado abajo de la barra de herramientas). Según el ejemplo seleccionado en el punto anterior, el programa cargado consiste en entrar y salir del modo de bajo consumo cuando se presiona el botón de *USER KEY* integrado en la tarjeta.
   La figura 3.11 presenta los mensajes que el nodo terminal manda al monitor serial de Arduino cuando entra y sale del modo de bajo consumo, con lo cual queda claro que la configuración del IDE de Arduino y el CubeCell funcionan adecuadamente.

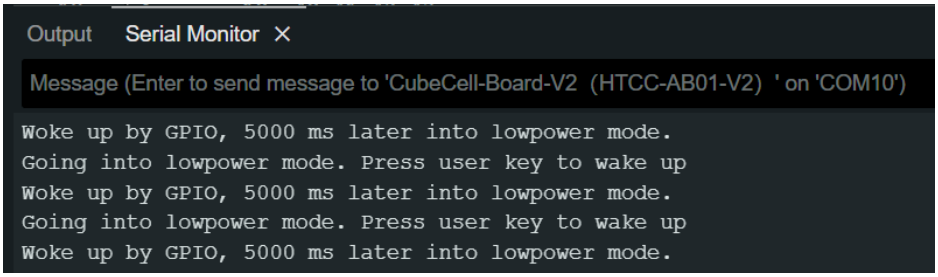


Figura 3.11: Salida por puerto serial del programa $LowPower_WakeUpByGPIO$ para el CubeCell.

3.4.2 Programación del nodo terminal.

Como primer punto de la programación del nodo terminal, se tiene el apartado encargado de la adquisición de los datos o variables a censar en la aplicación, esencialmente, se trata del conteo de los pulsos del medidor de flujo de agua, no obstante, también se considera medir el voltaje de

la batería con la que se alimenta el nodo, principalmente para fines de estudio del consumo de energía en las baterías.

Programación del conteo de pulsos

La manera más adecuada para el conteo de pulsos es mediante interrupciones; una interrupción es un evento de hardware, por lo que, cuando este evento ocurre, la unidad de procesamiento atiende inmediatamente el evento con una rutina de instrucciones que se programa dependiendo de la aplicación. Para el caso del medidor de flujo de agua, la rutina de instrucciones incrementará un contador, de este modo, el registro de consumo se estará almacenando en una variable en la memoria interna del microcontrolador, que podrá ser enviada a través de LoRa y LoRaWAN.

La forma en la que se configura la interrupción para el CubeCell es muy sencilla ya que se trata de Arduino, básicamente, se usa una instrucción que configura el pin o GPIO como entrada y otra instrucción que configura la interrupción y asocia una rutina de instrucciones (véase el código 3.1).

```
pinMode(GPIO1, INPUT);
attachInterrupt(GPIO1,cntIncrease,RISING);
```

Código 3.1: Configuración de la rutina de interrupción.

En relación con la segunda línea de código, esta indica que se ejecutará la rutina de instrucciones *cntIncrease* cuando se detecte un flanco positivo en el GPIO1. En la rutina de la interrupción del código 3.2 se tiene la variable del contador (cnt), que incrementa en uno cada vez que se ejecuta la rutina o, bien, se presenta el pulso por parte del medidor de flujo de agua.

```
void cntIncrease(){
  cnt++;
  Serial.println(cnt);
  delay(50);
}
```

Código 3.2: Rutina de interrupción.

El uso de un retardo (delay) al final de la rutina, aunque no se ajusta con las buenas prácticas de diseño de rutinas de interrupción, que recomiendan evitar bloqueos en el procesador y mantener rutinas lo más breves posible; este se implementó como solución pragmática para garantizar el correcto registro de cada pulso por los siguientes motivos:

1. Durante las pruebas de verificación de conteo de pulsos se presentó un conteo inconsistente, de acuerdo con los pulsos observados en el osciloscopio, cuyo origen se asocia con los rebotes eléctricos generados por el medidor de flujo de agua.

2. Considerando que la frecuencia con la que se presentan pulsos del medidor es muy baja, incluso cuando fluyen grandes cantidades de agua, el uso de un retardo que bloquee brevemente el procesador hasta que el pulso termine y se garantice que no hay rebotes no afecta el conteo de pulsos.
3. Puesto que el nodo terminal se encuentra en su mayoría de tiempo en estado *sleep*, y los eventos de comunicación LoRaWAN son poco frecuentes y espaciados entre ellos, bloquear el procesador tampoco interfiere con con el funcionamiento general del noto terminal.
4. Como solución pragmática, es la manera más sencilla y eficaz de resolver el problema de inconsistencia en el conteo de los pulsos del medidor de flujo de agua.

El valor del retardo se estableció con base en mediciones de la duración del pulso emitido por el medidor de flujo de agua, la figura 3.12 muestra una captura de pantalla de un osciloscopio donde se observa la magnitud y el tiempo que dura el pulso. Según lo visto, este tiempo es de aproximadamente 30 $[ms]$, por lo que se agregó una ventana de seguridad de 20 $[ms]$ (retardo de 50 $[ms]$ en total) para garantizar que, terminada la rutina de interrupción, no se vuelvan a registrar pulsos por los posibles "rebotes"de la señal.

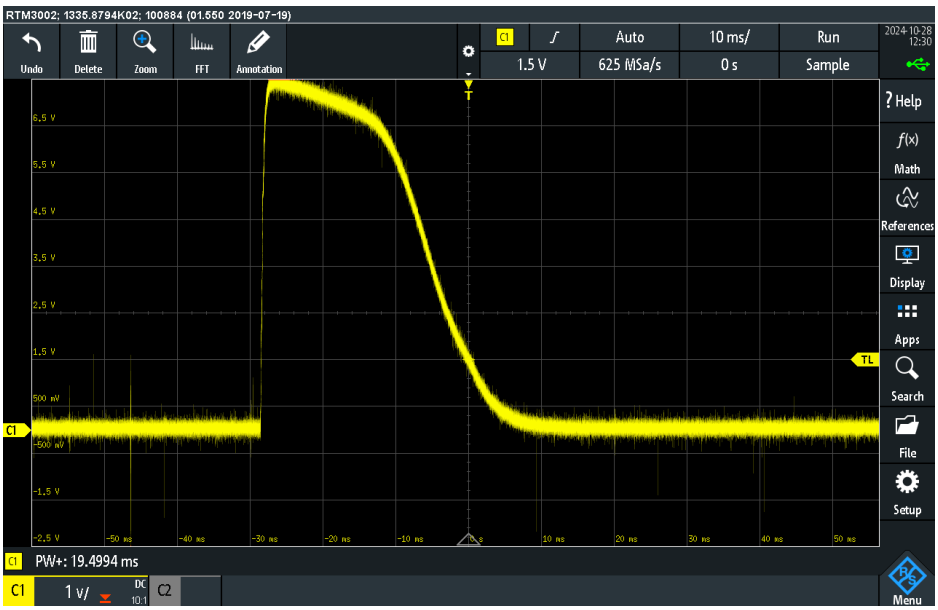


Figura 3.12: Duración del pulso del medidor de flujo de agua.

Programación de lectura de voltaje de la batería.

Aunque el registro/conteo de pulsos del sensor de flujo de agua es lo más importante en esta aplicación, conocer el voltaje de la batería con la que se está alimentando el nodo terminal propor-

ciona mucha información sobre su comportamiento y estado. Además, este monitoreo permitiría dar mantenimiento preventivo o, en su defecto, ayuda a saber cuando el nodo terminal dejará de funcionar por un nivel de voltaje insuficiente.

Al igual que con la rutina de conteo de pulsos, la lectura del voltaje de la batería se logra con base en un programa ejemplo proporcionado por el fabricante. Internamente el CubeCell tiene un convertidor analógico digital (ADC) dedicado para el voltaje de la batería, las instrucciones que se usan en el programa para leer y mostrar el valor están en el código 3.3.

```
void loop() {
  uint16_t voltage = getBatteryVoltage();
  Serial.println(voltage);
  delay(1000);
}
```

Código 3.3: Lectura de voltaje de batería.

Básicamente, la primera línea crea una variable de 16 bits donde se almacena el valor que retorna la función *getBatteryVoltage()*, después dicho valor se imprime en el monitor serial de manera continua cada segundo. Se puede comprobar que las lecturas de voltaje del microcontrolador son correctas al comparar dicho valor mostrado en el monitor serial con el que es medido con un multímetro, esto se ilustra en la figura 3.13.

(a) Lectura con el multímetro. (b) Lectura con el ADC.

Figura 3.13: Lecturas de voltaje de la batería.

Aunque el valor obtenido con el multímetro difiere en algunos [mV] con respecto al valor leído con el CubeCell, esta diferencia no es significativa, de modo que se pude confiar en los datos proporcionados por el nodo para monitorear el estado de la batería.

3.4.3 Programación del LoRa y LoRaWAN

Este apartado de configuración y programación podría ser el más importante, y a su vez, el más complicado debido a que LoRaWAN utiliza diversos protocolos de autenticación y confirmación en el envío y recepción de paquetes de datos. A pesar de las dificultades que implementar LoRaWAN pudiera representar, el fabricante del CubeCell (Heltec) provee ejemplos muy completos que incluyen toda la configuración del hardware de LoRa, así como todo lo necesario para el uso de LoRaWAN.

Los ejemplos de LoRa y LoRaWAN para el CubeCell que el fabricante proporciona se pueden encontrar, al igual que los mostrados en las secciones anteriores, en el entorno de Arduino junto con todos los demás ejemplos, según se mostró en la sección 3.4.1. El ejemplo elegido para trabajar con LoRaWAN se encuentra específicamente dentro de los ejemplos de *LoRa > LoRaWAN* y finalmente el programa lleva el nombre de *LoRaWAN*. Al abrir el programa se pueden identificar diversos bloques de configuración para los parámetros de LoRaWAN, los parámetros más importantes, y los que se modificarán en esta sección son los siguientes:

1. Identificadores y claves de seguridad: Para la activación de los nodos terminales por Over the Air Activation (OTAA), de acuerdo con el marco teórico, estas claves de LoRaWAN son las requeridas para registrar los nodos terminales en la plataforma a la que enviarán datos.

```
/* OTAA para*/
uint8_t devEui[] = { 0x22, 0x32, 0x33, 0x80, 0x00, 0x88, 0x88, 0x81 };
uint8_t appEui[] = { 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x81 };
uint8_t appKey[] = { 0x88, 0x88, 0x88, 0x88, 0x88, 0x88, 0x88, 0x88, 0x88, 0x88, 0x88, 0
x88, 0x88, 0x88, 0x66, 0x81 };
```

Código 3.4: Identificadores OTAA para LoRaWAN.

2. Tiempo de transmisión entre los paquetes: Este es muy importante ya que define el periodo en que el nodo terminal despierta del bajo consumo para enviar los datos registrados hasta ese momento.

```
/*the application data transmission duty cycle.  value in [ms].*/
uint32_t appTxDutyCycle = 1800000;
```

Código 3.5: Periodo de transmisión.

3. Puerto de la aplicación: Este dato está estrechamente relacionado con la configuración del Gateway, en otras palabras, tanto el Gateway como el nodo terminal deberán conectarse al mismo puerto.

```
/* Application port */
uint8_t appPort = 1700;
```

Código 3.6: Puerto de conexión con el Gateway central.

4. Número de reintentos y preparación de los paquetes: Se trata de una función donde se establece el tamaño (en bytes) y contenido del paquete que se enviará, en dicha función se observa cómo se inicializa uno a uno los bytes que conforman el paquete, esta función es llamada cada vez que el nodo se despierta para enviar datos, de forma que aquí se deberá llenar el paquete con los pulsos registrados y el voltaje de la batería.

```
uint8_t confirmedNbTrials = 4;
/* Prepares the payload of the frame */
static void prepareTxFrame( uint8_t port )
{
    appDataSize = 4;
    appData[0] = 0x00;
    appData[1] = 0x01;
    appData[2] = 0x02;
    appData[3] = 0x03;
}
```

Código 3.7: Preparación del payload.

5. Región de operación: Es importante verificar que la región en la que se está configurando el nodo terminal sea la correcta, en otras palabras, la región del nodo terminal y del Gateway debe ser coincidir. Según lo visto en el capítulo 2, la región adecuada para México es la $US915$, lo anterior significa que la región es Estados Unidos con una frecuencia de 915 $[MHz]$. La configuración de la región en el nodo terminal no se encuentra en código como en los fragmentos mostrados anteriormente, sino que para ello hay que entrar al apartado de *tools* de la barra de herramientas y seleccionar la región $US915$, como se muestra en la figura 3.14

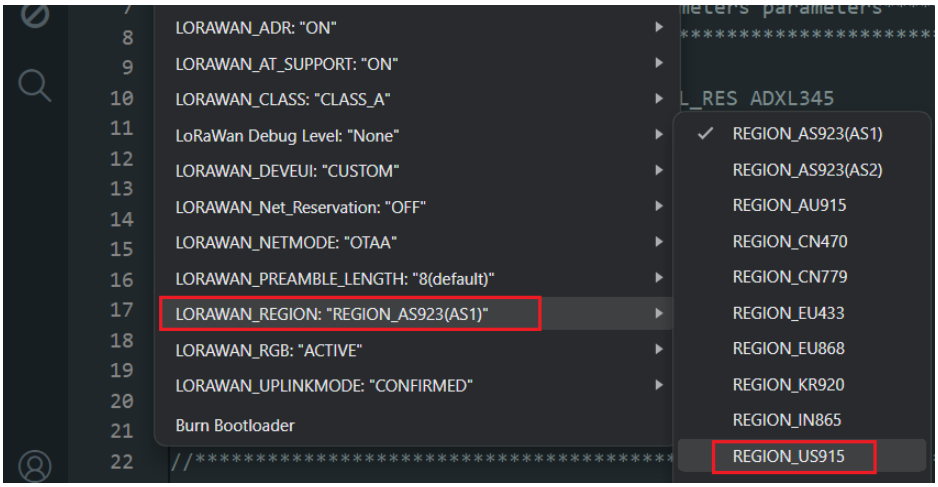


Figura 3.14: Selección de la región para LoRaWAN.

Dentro de las otras opciones de configuración en este menú, las más importantes y que deben ser establecidas son las siguientes:

- $LoRaWAN\_CLASS$: configura la clase en la que trabaja el nodo terminal. Se usa clase A puesto que es el más apto para aplicaciones de ultra bajo consumo.
- $LoRaWAN\_DEVEUI$: permite que la configuración de los identificadores y claves de seguridad para OTAA.
- $LoRaWAN\_RGB$: habilita y deshabilita el led RGB del CubeCell, se debe deshabilitar para reducir el consumo de corriente.

6. Los últimos apartados relevantes corresponden a la configuración del nodo terminal con los parámetros ya mencionados, y el ciclo principal que consiste en una máquina de estados que gestiona las funciones del nodo terminal con base en los eventos que surgen por parte de LoRaWAN. Con lo anterior, el siguiente paso es integrar los códigos realizados para el conteo de pulsos y lectura de voltaje de la batería en este programa ejemplo del fabricante.

Habiendo conocido las partes clave para la programación de LoRaWAN en el ejemplo proporcionado por el fabricante, resta integrar las funcionalidades requeridas por el nodo terminal, para ello se agregarán la función de interrupción y la configuración del GPIO que lee los pulsos del sensor de flujo de agua (códigos 3.2 y 3.1, respectivamente) de la sección anterior.

También es necesario agregar la instrucción para leer el voltaje de la batería del nodo terminal y enviarlo junto con el contador de pulsos. Esta parte de la programación es de especial atención ya que se debe llenar la variable de datos correspondiente al payload que se enviará. De acuerdo con el código del fabricante, el payload está compuesto por un arreglo de bytes cuya dimensión puede ser establecida previamente (código 3.7). El inconveniente que se presenta con este arreglo de datos es que la dimensión de las variables del contador y de voltaje de la batería son de dos bytes, por lo que en la preparación del payload se necesitan separar en dos partes el valor del contador y de voltaje de la batería, asimismo, este manejo de datos deberá ser considerado en la interpretación del payload por parte del receptor, lo anterior mencionado se puede ver programado en el código 3.8.

```
static void prepareTxFrame( uint8_t port )
{
    uint16_t voltage = getBatteryVoltage();
    appDataSize = 4;
    appData[0] = (cnt >> 8) & 0xFF;       // High byte of cnt
    appData[1] = cnt & 0xFF;              // Low byte of cnt
    appData[2] = (voltage >> 8) & 0xFF;   // High byte of voltage
    appData[3] = voltage & 0xFF;          // Low byte of voltage
    cnt = 0;
}
```

Código 3.8: Modificación del payload.

De acuerdo con el código, el primer byte del payload corresponde a la parte alta de la variable del contador, el segundo byte a la parte baja del contador, el tercer byte a la parte alta de la

variable de voltaje de la batería y el cuarto byte a la parte baja del voltaje de la batería. También, cabe resaltar que al final se reinicia el contador de pulsos, de modo que cada envío de datos envíe el número de pulsos correspondiente a ese periodo de tiempo.

Finalmente, lo único que resta es cargar el código completo (adjunto en el anexo A) al nodo terminal para su posterior registro en la aplicación web que se usará para la visualización de los datos (The Things Network).

3.4.4 Registro de nodos terminales en The Things Network

Para que el Gateway pueda reconocer los mensajes de los nodos terminales y desplegar la información de sus paquetes es necesario que se registren los nodos terminales en la plataforma TTN, esto se logra a través de los diferentes identificadores y claves de seguridad como el *devEui, appEui y appKey*, establecidas en la sección 3.4.3 donde se programó la parte de LoRaWAN en el CubeCell. El registro de los nodos se lleva a cabo con los siguientes pasos:

1. Una vez estando dentro de la aplicación creada previamente en la plataforma TTN y en el apartado de *End devices* se presiona el botón con etiqueta *Register end device*, como se muestra en la figura 3.15.

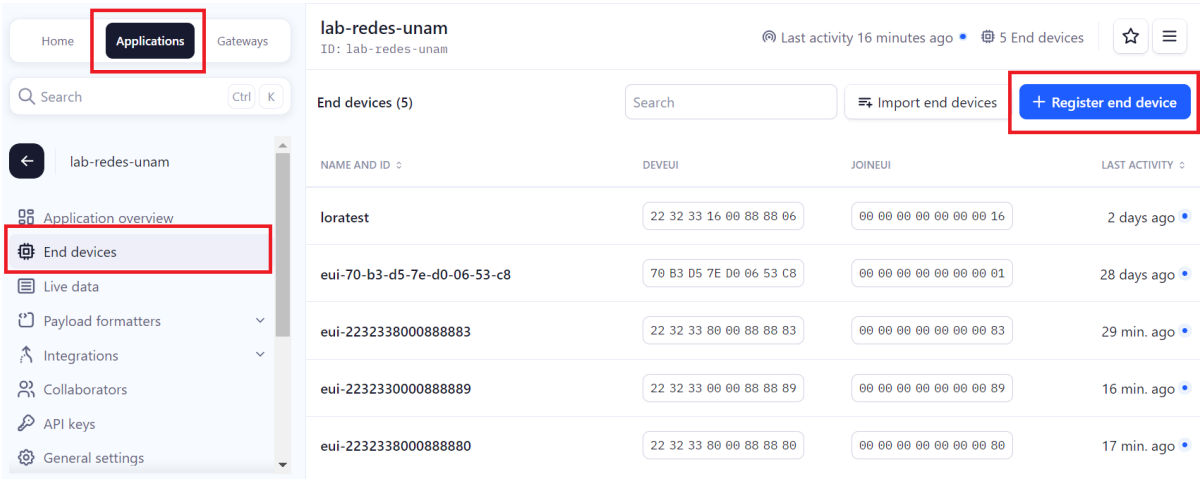


Figura 3.15: Registro de nodo terminal en TTN.

2. La figura 3.16 corresponde a la ventana inicial que se despliega para registrar los nodos terminales, el primer apartado solicita el tipo de dispositivo que se registrará, en este caso se selecciona la opción de *Enter end device specifics manually.*

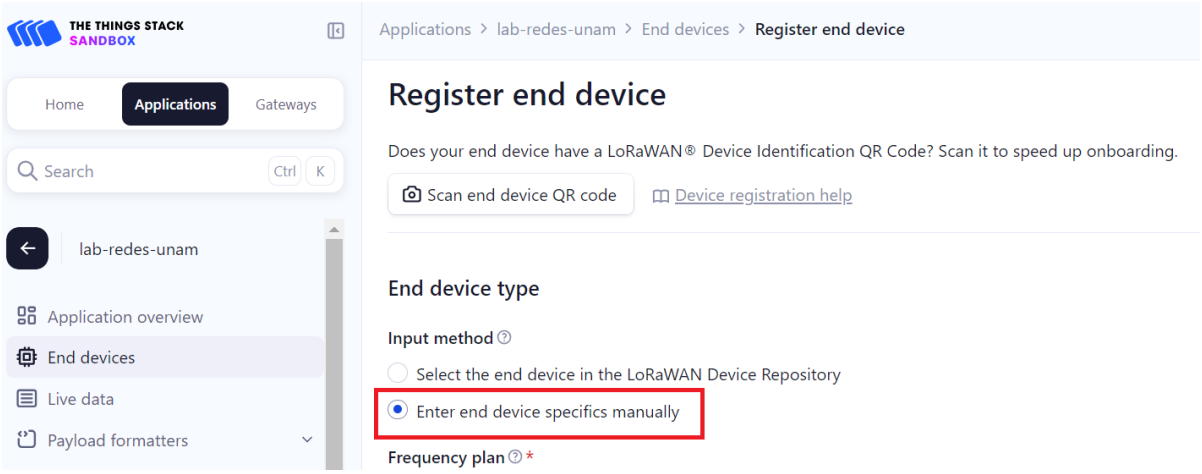
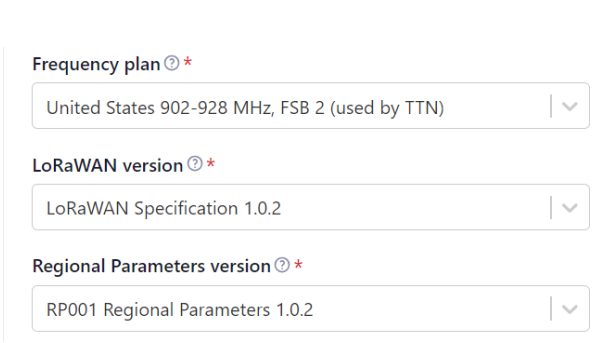
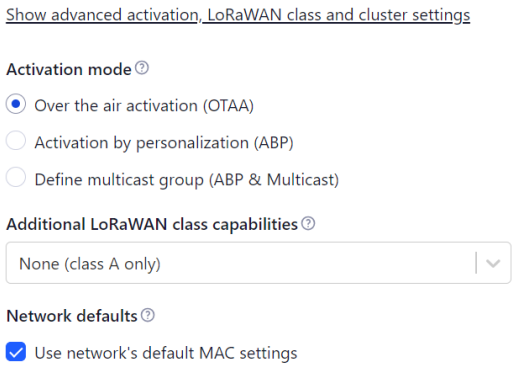


Figura 3.16: Selección del tipo de dispositivo a registrar.

3. Para registrar un nodo terminal de forma manual se requieren todos los datos que despliega la ventana de la figura 3.17. Dada la ubicación y el plan de frecuencia de la región, los campos solicitados se deben llenar como lo muestra la figura 3.17a, mientras que en el apartado de *Show advanced activation, LoRaWAN class and cluster settings* se pueden dejar las configuraciones por defecto mostradas en la figura 3.17b.



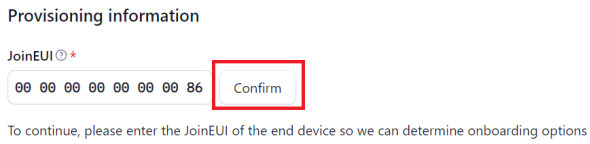
(a) Selección del plan de frecuencia y versión de LoRaWAN.



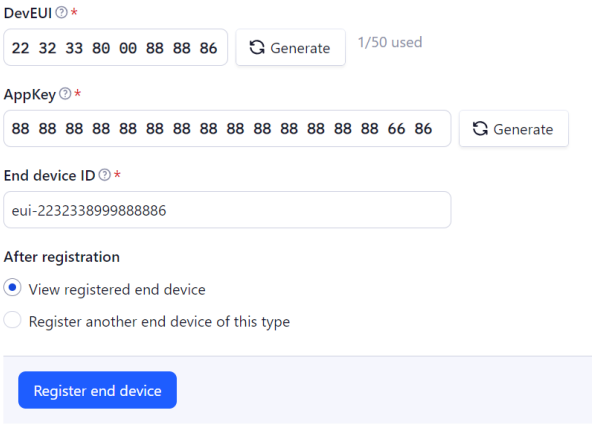
(b) Selección de la clase y método de activación.

Figura 3.17: Configuraciones generales para el registro del nodo terminal en The Things Network.

4. El siguiente apartado lleva la etiqueta de *Provisioning information* y es aquí donde se ingresa la primera de las claves con las que se identifica el nodo terminal. En la figura 3.18a aparece la ventana donde se ingresa la clave *appEui* del nodo terminal en el apartado de *joinEui*. Al dar clic en el botón confirmar se despliegan los apartados donde se ingresan las claves *devEui* y *appKey*, así como el nombre que tendrá el nodo terminal (figura 3.18b).



(a) Registro de la clave *appEui* del nodo terminal.



(b) Registro de las claves *devEui y appKey* del nodo terminal.

Figura 3.18: Registro final del nodo terminal en The Things Network.

5. Finalmente, el último paso es dar clic en el botón de *Register end device*, de modo que se puedan visualizar los datos que el nodo envía en la consola de datos de TTN.

3.5 Configuración y operación del Gateway central

El Gateway se requiere configurar por varios motivos, entre ellos, el principal y necesario es que se necesita cambiar la dirección del servidor al que se envían los datos.

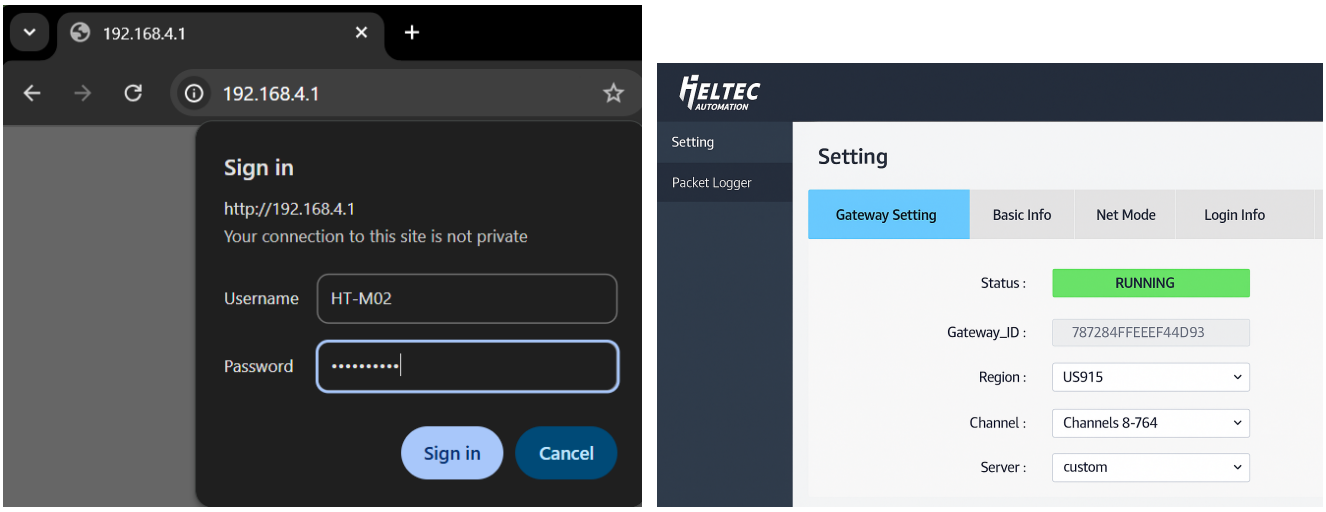
Las configuraciones necesarias para que el Gateway funcione debidamente en la región correcta se pueden realizar por dos métodos diferentes: a través de la interfaz gráfica del Gateway y por medio de un adaptador de USB a comunicación serial. Este segundo método resulta más complicado, pero también más confiable, ya que toda la configuración se hace a bajo nivel desde la terminal de una computadora, es decir, el Gateway sólo recibe las instrucciones a través de un adaptador que se conecta a unos pines del Gateway.

3.5.1 Comprobación del funcionamiento del Gateway

El primer paso para la configuración del Gateway es la verificación de su funcionamiento y algunos parámetros básicos, como la región, el canal en que opera y su identificador (Gateway ID), la forma de conseguir lo anterior es alimentar eléctricamente el Gateway y conectarlo a internet vía cable ethernet, esto hará que el Gateway cree una red Wi-Fi con el nombre *HT-M02-AP*, para conectarse a la red desde un dispositivo externo se requiere ingresar la contraseña: *heltec.org*.

Estando conectado a la red Wi-Fi del Gateway se puede ingresar la dirección IP (192.168.4.1)

del Gateway a través de un navegador, esto desplegara una ventana (figura 3.19a) donde se solicita un nombre de usuario (HT-M02) y contraseña (heltec.org), al iniciar sesión se muestra otra pantalla donde se visualiza el estado del Gateway, así como otros datos asociados a este (figura 3.19b). Esta ventana es importante ya que muestra la información clave para su funcionamiento y su posterior registro en la aplicación web.

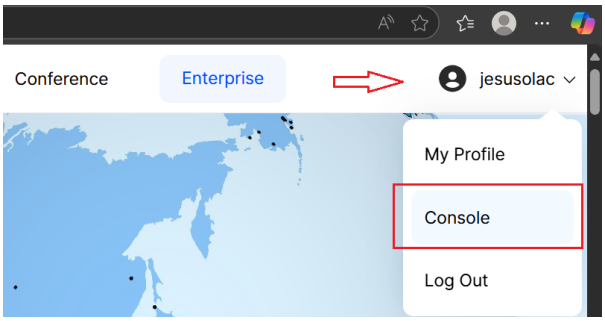


(a) Inicio de sesión del Gateway vía navegador. (b) Vista del estado operativo del Gateway.

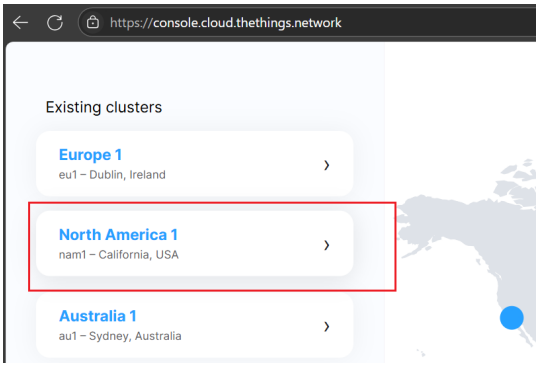
Figura 3.19: Interfaz web del Gateway y acceso a su información de estado.

3.5.2 Registro del Gateway en The Things Network

Con base en la información anterior, se puede proceder con el registro del Gateway en la plataforma The Things Network (TTN), para eso es necesario tener una cuenta en dicha plataforma, no obstante, la creación de la cuenta quedará fuera de esta tesis. Una vez iniciada sesión en TTN se accede al apartado de *console*, ubicado en el menú del perfil que se encuentra en la esquina superior derecha (figura 3.20a) y después habrá que seleccionar el servidor adecuado de acuerdo a la ubicación del Gateway, que en este caso será el de Norteamérica 1 (figura 3.20b).



(a) Acceso a la consola del perfil en The Things Network.



(b) Elección del servidor de conexión para el Gateway.

Figura 3.20: Interfaz inicial de usuario en TTN para el registro del Gateway.

La nueva ventana muestra una vista general de todos los Gateways y aplicaciones presentes o hechas en la cuenta, nuevamente en la esquina superior derecha existe la opción de registrar un Gateway (*Register gateway*), según lo muestra la figura 3.21.

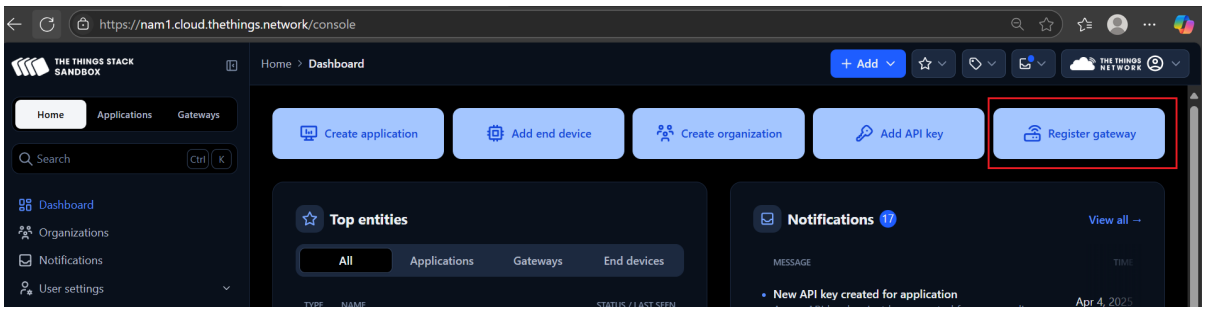


Figura 3.21: Acceso al registro de un Gateway en The Things Network.

Dentro de la ventana para el registro del Gateway, el primer parámetro a ingresar es el llamado *Gateway ID*, cuyo valor es el que se despliega como ID del Gateway en la figura 3.19b. Al confirmar el *Gateway ID* se despliegan más opciones para completar, estos valores son: el nombre del Gateway y el plan de frecuencia. También se seleccionan las casillas de modo que el registro del Gateway se ve como en la figura 3.22.

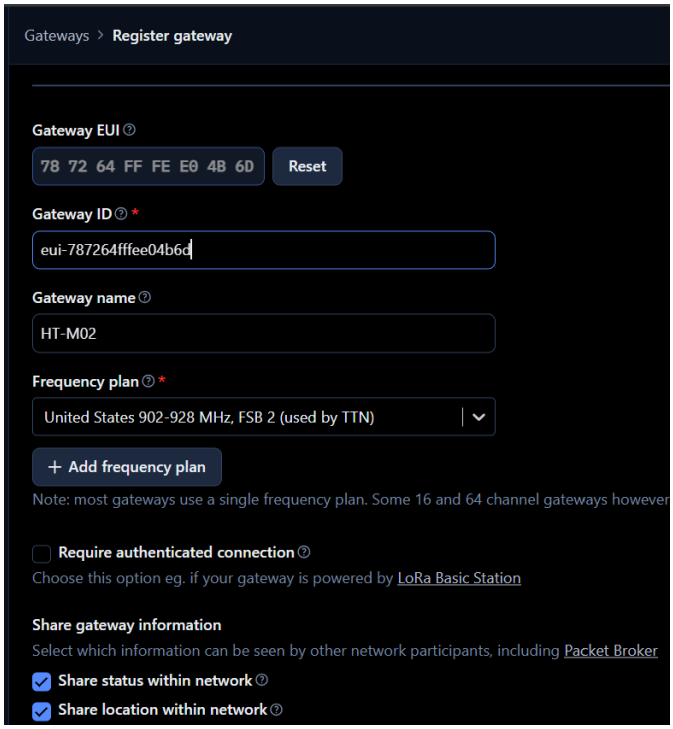


Figura 3.22: Parámetros e identificadores del registro del Gateway.

El proceso de registro termina dando clic en el botón con la etiqueta *Register gateway*, ubicado en la parte inferior de la ventana de la figura anterior. El registro se puede verificar al ver el Gateway en la lista de Gateways registrados de la sección del menú lateral de la consola de The Things Network (figura 3.23).

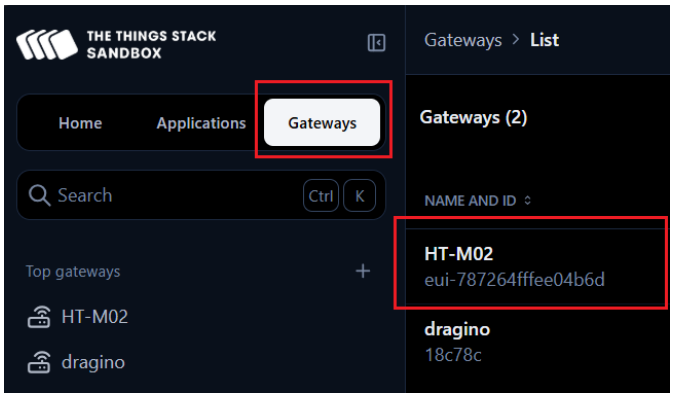
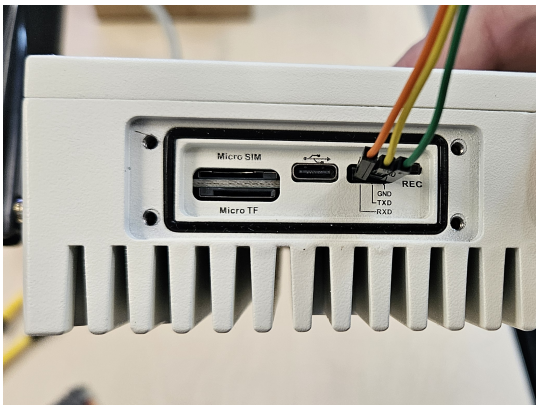


Figura 3.23: Listado de Gateways registrados en The Things Network.

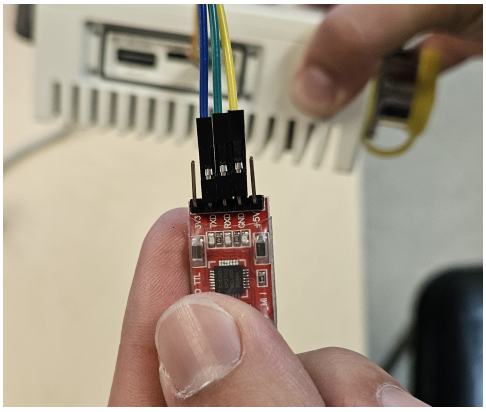
3.5.3 Configuración de la dirección del servidor en el Gateway

Cuando el Gateway se encuentra registrado en la plataforma TTN, es requerido cambiar la dirección del servidor donde se enviarán los datos, con base en el registro que TTN generó para el Gateway. Esta configuración se hará por medio de depuración UART, cuya conexión está situada en la interfaz de expansión presente en el costado del Gateway (figura 3.4), dichas interfaces se muestran a detalle en la figura 3.24a, donde además se observan los pines necesarios ya conectados (GND, TX y RX).

En cuanto al adaptador de USB a puerto serial, se usa el FTDI ft232, cuya conexión se presenta en la figura 3.24b. En el protocolo de comunicación UART, al ser un protocolo asíncrono (no hay un reloj común entre dispositivos), cada dispositivo transmite por el pin TX y recibe por el pin RX; de este modo, para garantizar la correcta transmisión y recepción se debe realizar una conexión cruzada, es decir, se conectar el pin TX del FTDI al pin RX del Gateway, y el pin RX del FTDI al pin TX del Gateway.



(a) Pines puerto serial del Gateway.



(b) Pines FTDI ft232r.

Figura 3.24: Conexión para la configuración del Gateway.

El siguiente paso posterior a realizar la conexión física entre el Gateway, el adaptador USB y la computadora, es establecer la comunicación por medio de un monitor serial, para esto se optó por usar PuTTY, de modo que para configurar la comunicación se necesita configurar el puerto COM del adaptador y la velocidad de la comunicación dentro de la aplicación, esto se logra accediendo al apartado de *Session* y fijando los parámetros correctos, como se muestra en la figura 3.25, es importante que la velocidad se fije en 115200 baudios.

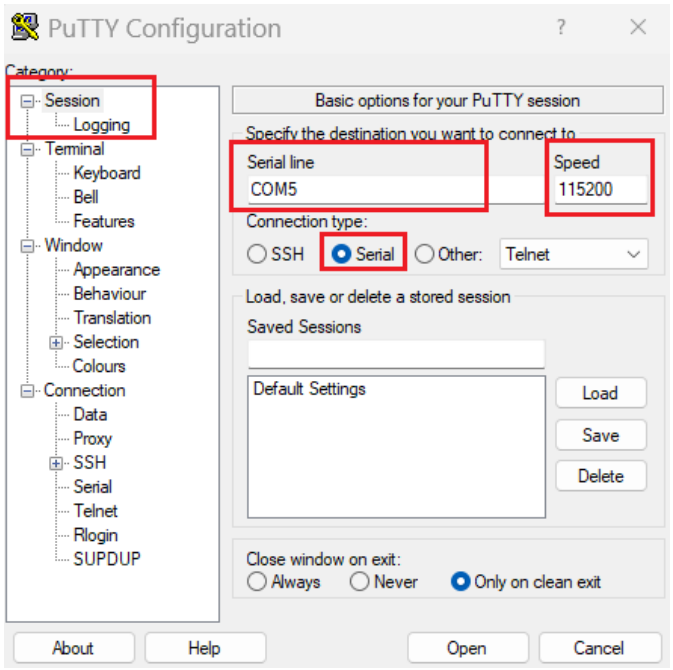


Figura 3.25: Configuración de la comunicación UART en PuTTY.

Al dar clic en el botón de *open* se despliega una nueva ventana indicando que se estableció la conexión correctamente, el primer mensaje que manda el Gateway es una solicitud de inicio de sesión, cuyo login y contraseña son *root* y *heltec.org*, respectivamente, al presionar la tecla enter el Gateway iniciará sesión mostrando el mensaje de la figura 3.26.

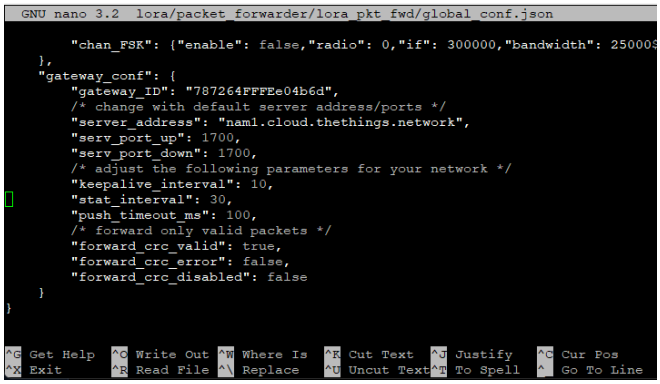
Figura 3.26: Conexión entre PC.

La dirección del servidor se puede observar y modificar ingresando el comando *sudo nano lora/packet_forwarder/lora_pkt_fwd/global_conf.json*, dicho comando accede al archivo donde se encuentran configurados la dirección del servidor y otros valores relevantes como el puerto, estos se encuentran al bajar por la información contenida en el archivo. La nueva dirección se encuentra en

la información general del Gateway, accesible desde TTN en el apartado de los Gateways (figura 3.27a).

Para modificar la dirección del servidor en la terminal de PuTTY primero se copia la dirección desde TTN, después, al seleccionar la antigua dirección en la terminal de PuTTY y dar clic derecho, esta se actualizará; finalmente, se guardan los cambios con los siguientes comandos: *Ctrl+X* → *Y* → *Ctrl+C*. La información actualizada se debe mostrar como en la figura 3.27b.

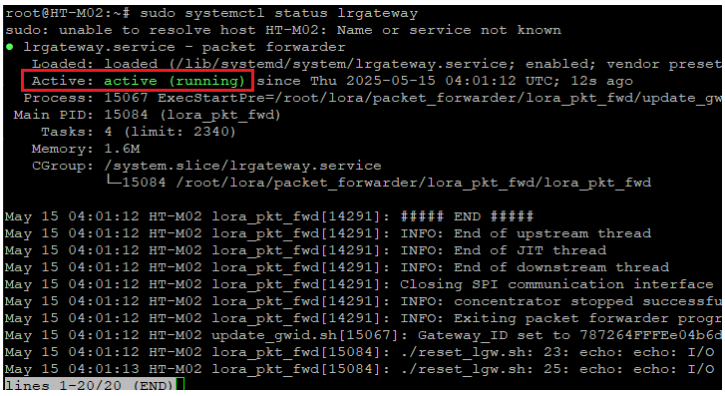
(a) Información general del Gateway en The Things Network.



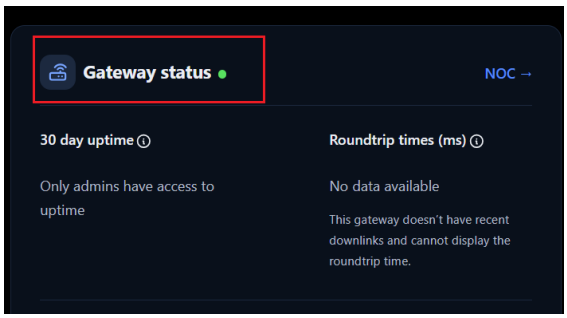
(b) Dirección del servidor actualizada.

Figura 3.27: Configuración de la dirección del servidor para el Gateway.

El paso final para completar la configuración y poner a funcionar el Gateway es reiniciarlo con el comando: *sudo systemctl restart lrgateway*, verificar que esté operando con el comando: *sudo systemctl status lrgateway*, que despliega una ventana con el estado del Gateway (figura 3.28a) y, por último, observar que el Gateway aparezca como conectado en TTN (figura 3.28b).



(a) Estado del Gateway en depuración UART



(b) Estado del Gateway en The Things Network.

Figura 3.28: Estado del Gateway.

Capítulo 4

Análisis del consumo energético del sistema

4.1 Introducción

Este capítulo se enfoca en analizar y determinar el consumo de energía de los nodos terminales de la red. Este análisis consiste en medir la corriente que el nodo terminal demanda por ciclo de trabajo o por transmisión de paquetes, donde un ciclo de trabajo abarca la lectura de los pulsos del medidor, preparación del paquete, envío del paquete y confirmación de recepción del paquete, y determinar el consumo promedio del nodo terminal.

Para medir estas corrientes de consumo se necesitan equipos de adquisición de datos que tengan la frecuencia de muestreo, rango y resolución adecuadas para detectar los eventos, estos eventos suelen ser de pequeña duración debido a las especificaciones de LoRa y LoRaWAN, por lo que no es tan sencillo detectarlos. Dado que al inicio de esta actividad no se contaba con el equipo adecuado, las mediciones se realizaron con dos métodos que serán explicados en su respectivo apartado: *método de la resistencia shunt con osciloscopio* y *la lectura de un sensor de corriente con un microcontrolador*; posteriormente se adquirió un equipo especializado que, de acuerdo con sus especificaciones técnicas, promete dar mejores resultados que los métodos anteriores.

De acuerdo con el marco teórico, el protocolo LoRaWAN en su clase A tiene tres estados de operación elementales: la parte de transmisión, el estado *idle* y la o las ventanas de recepción. Estos eventos tienen diferentes características en cuanto a magnitud y duración, siendo la duración del evento lo más difícil de cuantificar, por lo anterior, los métodos y sistemas de medición propuestos deben contar con la suficiente frecuencia de muestreo como para captar y caracterizar con el mayor detalle posible dichos eventos.

4.2 Especificaciones de la alimentación de energía eléctrica

Las especificaciones de la alimentación se basan en las necesidades del proyecto. Debido a que no es posible alimentar de manera continua a la red eléctrica los nodos terminales, se requiere una batería para su alimentación, esta debe cumplir con las especificaciones del CubeCell, es decir, su voltaje debe estar en un rango entre 3.3 y 5 $[V]$.

En lo que a la capacidad de la batería se refiere, este valor dependerá del análisis y estimación de la vida útil de las baterías según el consumo de energía que el CubeCell necesite para su operación. De este modo, las características eléctricas de la batería, en particular su voltaje y capacidad, son factores que deben seleccionarse cuidadosamente para garantizar el funcionamiento continuo de los nodos terminales.

Con base en el voltaje de alimentación del CubeCell, y a partir de la disponibilidad de baterías en Ciudad de México, para la mayoría de las pruebas se seleccionaron baterías recargables de iones de litio (LI-ION) marca UNIT Electronics, modelo 18650 y con voltaje nominal de 3.7 $[V]$, su capacidad es de 3000 $[mAh]$ y se puede adquirir y consultar más información en el sitio web de la tienda.

4.3 Sistemas para la medición y adquisición de datos de corriente

De acuerdo con la metodología propuesta para el análisis de consumo energético del nodo terminal, es fundamental tener un método para adquirir datos de corriente que sea fiable, por lo que, para garantizar que dichas mediciones sean las más exactas y precisas, se emplearon tres métodos diferentes para la adquisición de datos. Estos son:

1. Adquisición de datos con un microcontrolador y sensor digital de corriente.
2. Adquisición de datos mediante osciloscopio con el método de la resistencia shunt.
3. Adquisición de datos con el equipo Otii ARC Pro (Qoitech AB, Suecia).

A partir de todos los datos recabados de las mediciones fue posible determinar qué el método de adquisición de datos más adecuado en cuanto a facilidad y confiabilidad es el equipo Otii Arc pro, de modo que este fue el punto de referencia para las comparación de los sistemas propuestos.

La finalidad de adquirir y guardar los datos de las mediciones es poder elaborar un análisis más detallado del consumo de energía utilizando diversas herramientas. Esto permitió comparar los resultados del consumo por ciclo de trabajo obtenidos a partir de cálculos con los resultados arrojados por el equipo Otii ARC Pro. Para llevar a cabo lo anterior, se propone el siguiente procedimiento:

1. Comparar qué método y sistema presentan mejores características en la adquisición de datos de corriente.
2. Calcular experimentalmente la energía, en mAh, que el nodo consume por ciclo de trabajo

con base en los datos adquiridos.
3. Calcular, de manera teórica, la energía, en mAh, que el nodo consume por ciclo de trabajo.
4. Comparar los resultados de los cálculos teóricos y los cálculos experimentales.

4.3.1 Medición de corriente con microcontrolador

Para este sistema de medición se necesitan principalmente tres elementos: un microcontrolador, un sensor de corriente y una herramienta para almacenar y procesar los datos adquiridos. El microcontrolador elegido es el ESP32, principalmente debido a que opera a una frecuencia máxima de 240 $[MHz]$, esta frecuencia de operación es útil para ejecutar la adquisición y transmisión de datos en el menor tiempo posible, con el objetivo de lograr la mayor frecuencia de muestreo.

Por otro lado, se seleccionó el sensor de corriente INA226, debido a su disponibilidad en el mercado, la compatibilidad directa con el ESP32 utilizando Arduino IDE y su capacidad para medir corrientes en el rango deseado.

Una limitante importante a considerar es la velocidad con la que se pueden adquirir y almacenar los datos del sensor, esto es fundamental, ya que define la frecuencia de muestreo del sistema. Aunque el ESP32 puede trabajar a frecuencias de hasta 240 $[MHz]$, la adquisición de los datos está limitada por tres factores:

1. El protocolo de comunicación entre el sensor INA226 y el ESP32 es I2C, cuya frecuencia máxima de operación es de 400 $[kHz]$.
2. La cantidad de operaciones necesarias para leer los datos del sensor, registrar el tiempo transcurrido en cada muestra y preparar los datos para ser enviados por puerto serial a la PC.
3. La velocidad del protocolo (UART) utilizado para enviar los datos desde el ESP32 a la PC para su posterior procesamiento.

Otro inconveniente es que el INA226 no puede medir corrientes en el rango de $[\mu\text{A}]$. Una solución es medir el consumo de corriente del nodo terminal cuando está en modo sleep. La figura 4.1 muestra dicho consumo de corriente cuando el nodo está alimentado por la batería de litio. Posteriormente, este valor se emplea como referencia cuando el INA226 registra corrientes inferiores a su rango medible.

Figura 4.1: Medición de corriente del CubeCell en modo sleep.

El diagrama de conexión se presenta en la figura 4.2, en este diagrama se muestran todas las conexiones necesarias entre el ESP32, el sensor INA226 y el nodo terminal, que actúa como carga. Una característica útil del INA226 es su capacidad para alimentar la carga con una batería externa, lo que permite realizar mediciones en condiciones más reales. La configuración y lecturas del sensor se realizan con las funciones de la librería hecha por *wollewald* [44], la cual se puede encontrar y consultar en su repositorio de GitHub.

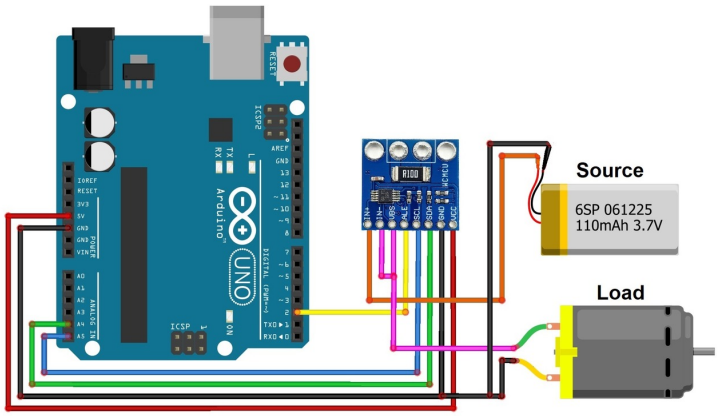


Figura 4.2: Diagrama de conexión del sistema [13].

A partir de la librería del sensor y las especificaciones de los protocolos mediante los cuales se transmiten los datos, en la configuración final del sistema se fijaron tales parámetros que, para estimar la frecuencia de muestreo se consideran los siguientes valores de tiempo:

1. Tiempo de conversión del INA226: Fijado en 1.1 ms por defecto del fabricante para tener balance entre velocidad de adquisición y precisión en las lecturas.
2. Lectura por I2C: A 400 kHz, el tiempo de lectura de dos registros de 16 bits se puede estimar

agregando 50 % de los bits leídos, a estos se les llama overhead de protocolo y son generados por los Acknowledgment (ACK) y Negative Acknowledgment (NACK).

$$N_{bits} = 32 + 16 = 48 \text{ bits útiles}$$

$$t_{\text{I2C}} = \frac{N_{\text{bits totales}}}{f_{\text{I2C}}} = \frac{48}{400 \times 10^3} = 120\ \mu s$$

3. Transmisión UART: A 921600 baudios, el paquete de 13 bytes formado por la instrucción `sprintf(txpacket, "%0.3f,%0.2f", current_mA, loadVoltage_V);` incluyendo delimitadores y decimales:
$$t_{UART} = \frac{13 * 8\ bits}{921600\ bits/s} \approx 0.1128\ ms$$
4. Retardo: Se incluye un retardo estático entre lecturas ($t_{retardo} = 0.3\ ms$).

El tiempo que le toma a los datos de cada muestra ser leídos y enviados por UART es:

$$t_p = t_{I2C} + t_{UART} + t_{retardo} = 0.120 + 0.1128 + 0.3 = 0.5328\ ms$$

t_p es pequeño respecto al tiempo de conversión del sensor (1.1 ms), incluso si se consideraran latencias entre transferencias y operaciones, de modo que el parámetro que define la frecuencia de muestreo del sistema es el tiempo de conversión del sensor y, por lo tanto, la frecuencia de muestreo aproximada del sistema es:

$$f_s = \frac{1}{t_{\text{total}}} = \frac{1}{1.1 \times 10^{-3}} \approx 909 \text{ Hz}$$

La función principal, mostrada en el código 4.1, se encarga de leer los datos proporcionados por el sensor y enviar los valores de corriente y voltaje por el puerto serial, en un formato separado por comas, con el objetivo de simplificar su recepción y procesamiento mediante herramientas de software. De acuerdo con las mediciones mostradas en la figura 4.1, el CubeCell consume aproximadamente 30 [μA] en reposo, por lo tanto, este valor se le asignará a todas las lecturas cuyo valor sea menor a 1 [mA].

```
void loop(void) {
.
.
.
  // Transmisión serial optimizada
  sprintf(txpacket,"%0.3f, %0.2f",current_mA, loadVoltage_V);  //start a package
  Serial.println(txpacket);
  delayMicroseconds(300);
}
```

Código 4.1: Lectura y envío de datos por el puerto serial.

La herramienta o software empleado para el análisis de la información es MATLAB, debido a su dominio y capacidad para procesar grandes cantidades de datos, tanto el programa de MATLAB, como el de Arduino para el INA226, se encuentran completos en el Anexo B. Las instrucciones que se encargan de recibir y almacenar los datos de corriente, voltaje y tiempo de cada muestra enviados por el microcontrolador se presentan el código 4.2.

```
% Lectura de datos durante 30 segundos
while milliseconds(datetime('now') - startTime) < tiempoMaximo
    .
    .
    .
    if length(valores) == 2  % Asegura que hay dos valores (corriente y voltaje)
        elapsedTime = milliseconds(datetime('now') - startTime); % Tiempo en ms desde el inicio
        % Almacena los valores leídos
        timeData = [timeData; elapsedTime];
        corriente = [corriente; valores(1)];
        voltaje = [voltaje; valores(2)];
    end
end
```

Código 4.2: Recepción y almacenamiento de datos.

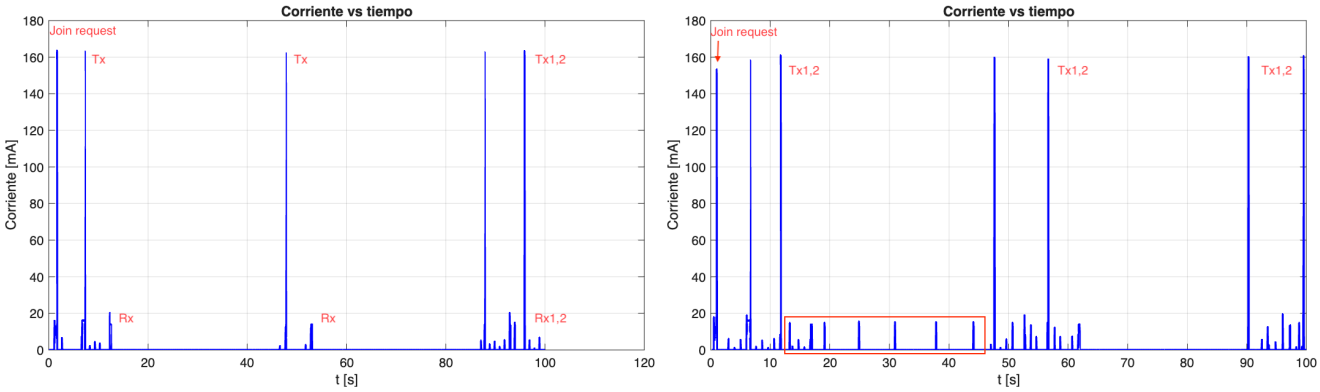
Para las mediciones del sistema se propusieron las siguientes características, las cuales tienen la única finalidad de facilitar el análisis y visualización de cada uno de los estados y procesos realizados por el nodo terminal:

- Se registran los eventos desde que se energiza el nodo terminal.
- Intervalo de envío de paquetes de 40 segundos.
- Una loza y un muro de concreto obstaculizaban la línea de vista entre el nodo terminal y el Gateway.
- La frecuencia de muestreo es de 0.909 kSa/s según lo calculado.

Como ejemplo de las mediciones con las características descritas, la figura 4.3a muestra los picos de consumo de corriente de acuerdo con los diferentes estados por los que pasa el nodo terminal. Adicionalmente, la figura 4.3b muestra el mismo comportamiento, con la diferencia que en esta medición se aprecian los consumos de corriente asociados a un conteo de pulsos por parte del medidor de flujo de agua (encerrados en rojo).

Las gráficas presentadas son muy útiles ya que están registrados procesos como el *Join request* y múltiples envíos de datos, donde se observan diferencias entre ellos, principalmente porque en algunos existen dos ventanas de recepción o hay más pulsos que parecen envíos de datos que, a simple vista, no corresponden a la programación del nodo terminal. El aspecto de mayor relevancia es que se notan pequeños pulsos de corriente entre el envío de datos y las ventanas de recepción, estos pequeños pulsos podrían significar algún proceso interno del microcontrolador, sin embargo, la frecuencia de muestreo lograda con este sistema de adquisición de datos no logra capturar los

consumos de corriente.



(a) Medición sin medidor de flujo de agua. (b) Medición con medidor de flujo de agua.

Figura 4.3: Consumo de corriente del nodo terminal medido con el sensor INA226.

En la figura 4.4 se ven de cerca los pulsos entre el envío de datos y las ventanas de recepción, aunque la resolución de la muestra no permite observar con todo detalle la forma de estos pulsos, se puede comprobar que el comportamiento de estos pulsos es periódico, ya que la separación entre ellos es de aproximadamente 1 segundo.

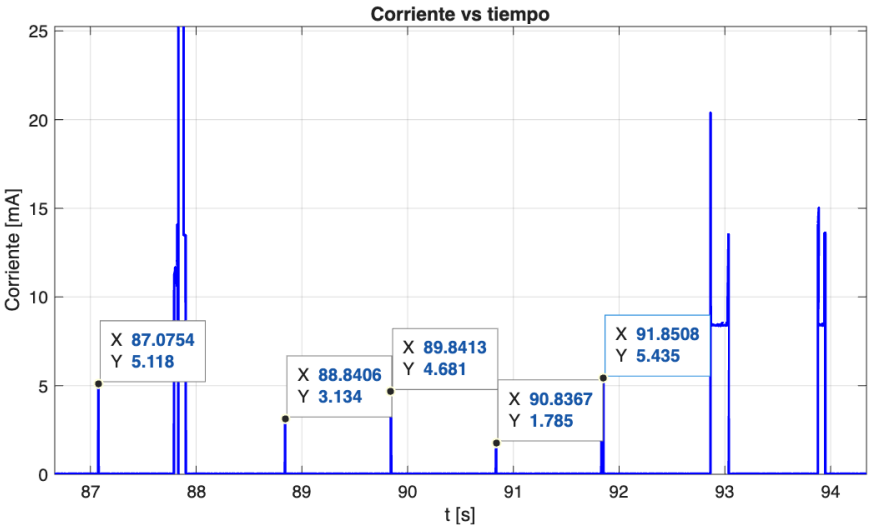


Figura 4.4: Pulsos de consumo entre la transmisión y ventanas de recepción.

Con las gráficas resultantes se concluye que este sistema es adecuado para detectar de buena manera los eventos principales del nodo terminal en su ciclo de trabajo, con excepción de los pequeños pulsos de la figura 4.4. Ademas, las mediciones permiten comprobar que el nodo terminal se comporta correctamente al detectar los pulsos provenientes del sensor de flujo de agua. Por

último, también se logró identificar pulsos de consumo idénticos a los pulsos de transmisión, cuyo origen probablemente se deba a retransmisiones.

La tabla 4.1 presenta los valores promedio de consumo de corriente y duración de los eventos registrados con este sistema, los cuales serán útiles para futuras comparativas.

Modo o Evento	**Consumo [mA]**	**Duración [ms]**
Join request	26.16	546.25
Transmisión Tx	83.17	99.95
Recepción Rx1	8.29	76.05
Pulsos de preparación de transmisión y recepción	4.51	1.06
Sleep	0.03	-

Tabla 4.1: Valores promedio de pulsos de consumo de corriente en condiciones ideales

Este sistema resulta eficaz considerando su accesibilidad y que cumple con el objetivo de detectar los eventos del ciclo de trabajo, por lo que es adecuado para analizar el comportamiento del nodo terminal durante su ciclo de trabajo, sin embargo, los siguientes factores impiden el uso del sistema basado en el módulo comercial INA226 y el ESP32 para cálculos energéticos confiables:

- La frecuencia de muestreo alcanzada con la configuración no logra registrar con detalle todos los pulsos de consumo del nodo terminal, principalmente causado por el tiempo de conversión del sensor INA226.
- El rango de medición establecido por el valor de resistencia shunt del modulo impide lecturas de corriente inexactas en el estado sleep del nodo terminal.

4.3.2 Medición de corriente con osciloscopio

La ventaja de adquirir datos con un osciloscopio es su alta frecuencia de muestreo, esto permite registrar eventos de menor duración que los captados con el sistema diseñado con el microcontrolador y el sensor de corriente, sin embargo, al no poder medir corriente de forma directa con este instrumento se requiere diseñar un circuito basado en el método de la resistencia shunt, el cual consiste en medir el voltaje que cae en una resistencia conectada en serie con la carga [45].

En la figura 4.5 se observa el circuito utilizado en este método, para ello se escogió una resistencia shunt con un valor de 1 [Ω] debido a las siguientes razones: es un valor fácil de conseguir, es lo suficientemente grande para que haya una caída de voltaje fácil de medir cuando el nodo está dormido, no es tan grande como para afectar la alimentación del nodo durante la transmisión y,

por último, simplifica en gran medida los cálculos.

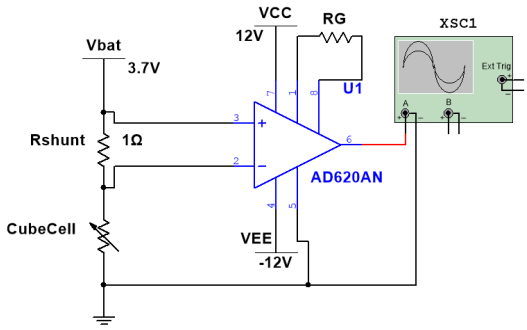


Figura 4.5: Esquema de medición de corriente por el método de la resistencia shunt.

El voltaje que cae en la resistencia shunt puede ser visto con el osciloscopio al ser amplificado con un amplificador de instrumentación gracias a su alta ganancia, inmunidad al ruido y elevada impedancia de entrada. Se escogió el amplificador AD620 gracias a que su configuración, mostrada en la figura 4.6, permite ajustar la ganancia con una sola resistencia R_G y cuyo valor se determina con la expresión 4.1 [14].

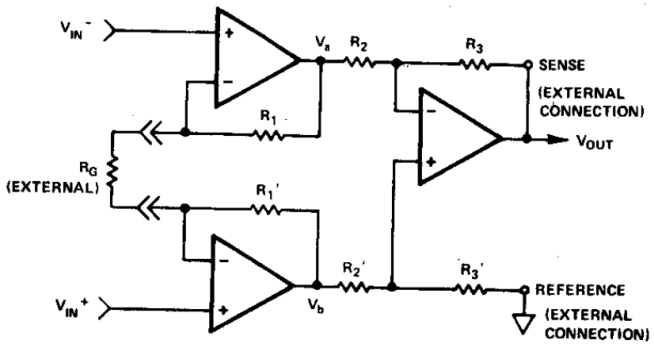


Figura 4.6: Estructura clásica de un amplificador de instrumentación con 3 amplificadores operacionales [14].

$$R_G = \frac{49.4\ [k\Omega]}{A_V - 1}\ [\Omega] \tag{4.1}$$

Para establecer la ganancia adecuada en las mediciones del nodo terminal se consideran los resultados obtenidos con el sistema de microcontrolador. Considerando que, según lo medido en la figura 4.1, cuando el nodo terminal se encuentra en estado *sleep* y su consumo es de 30 μA, el voltaje en la resistencia shunt es:

$$V_R = R \cdot I = 1 \cdot 30 \times 10^{-6} = 30\ [\mu V]$$

Por otro lado, cuando el nodo terminal transmite su consumo de corriente es de 185 mA, de acuerdo con las especificaciones de los chips resumidos en la sección 3.4.3, no obstante, con base en las mediciones hechas con el sensor INA226 se estableció que la corriente máxima demandada es de 150 mA, por lo que el voltaje en la resistencia shunt en esta etapa del ciclo de trabajo es:

$$V_R = R \cdot I = 1 \cdot 150 \times 10^{-3} = 150 \ [mV]$$

Con este valor como entrada máxima del amplificador, se propone que el valor máximo a la salida del amplificador sea de 8 V. Por lo tanto, la ganancia de voltaje requerida se calcula como:

$$A_V = \frac{V_O}{V_R} = \frac{8}{150 \times 10^{-3}} = 53.33 \ [V/V]$$

Esta ganancia, con la ecuación 4.1, se traduce en una resistencia de:

$$R_G = \frac{49.4 \ [k\Omega]}{53.33 - 1} = 944 \ [\Omega]$$

Puesto que el valor resultante no existe comercialmente, se estableció $R_G = 1 \ [k\Omega]$.

Es importante mencionar que estos valores tendrán que ser ajustados con base en una medición del valor de resistencia real, causado por la tolerancia en los valores de las resistencias, tanto en la shunt, como la resistencia (R_G). Con lo anterior, la construcción del circuito en protoboard se muestra en la figura 4.7, donde se aprecian las dos resistencias junto con el nodo terminal conectado como carga.

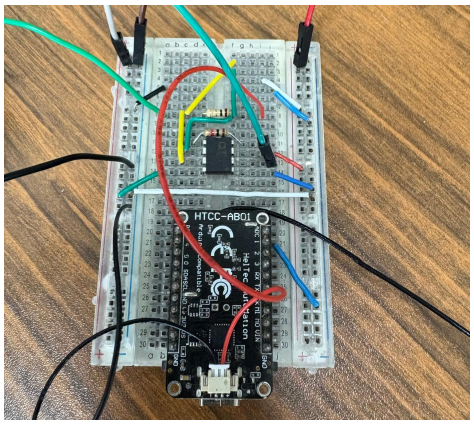


Figura 4.7: Construcción práctica del circuito para la medición del voltaje en la resistencia shunt.

Para la adquisición de datos se utilizó el osciloscopio Rohde & Schwarz RTM3002 (figura 4.8). Este osciloscopio ofrece una gran cantidad de funciones y características, entre ellas, las más importantes se encuentran resumidas en la tabla 4.2.

Figura 4.8: Osciloscopio Rohde & Schwarz RTM3002 [15].

Las características más relevantes para la aplicación en cuestión son la frecuencia de muestreo y la capacidad de realizar grabaciones y exportarlas en archivos de formato .csv mediante USB, con esto, se busca manipular y analizar los datos de la misma forma en que se hizo con el sistema anterior.

Característica	**Descripción**
Canales	2 canales analógicos
Ancho de banda	100 MHz, 200 MHz o 350 MHz (según el modelo)
Velocidad de muestreo	Hasta 5 GSa/s por canal
Memoria	40 MSa por canal estándar, expandible hasta 400 MSa
Pantalla	Pantalla táctil de 10.1"(1280 x 800 píxeles)
Modos de disparo	Patrón, duración/ancho de pulso, configuración en tiempo
FFT y análisis	FFT de hasta 128 k puntos
Interfaces	USB, LAN, salida VGA
Funciones matemáticas	Integración, derivación y funciones de usuario
Peso	Aproximadamente 4.2 kg
Dimensiones	390 mm × 220 mm × 152 mm

Tabla 4.2: Características principales del osciloscopio Rohde & Schwarz RTM3002 [15].

Para obtener las lecturas más realistas, se debe considerar que las operaciones dependen de los valores reales de resistencia usados en la construcción del circuito, por lo tanto, estos valores se obtuvieron a partir de mediciones hechas con un multímetro, dando como resultado: $R_G = 995$ [Ω] y $R = 2.02$ [Ω] (figura 4.9a y 4.9b, respectivamente).

(a) Medición real de R_G. (b) Medición real de R.

Figura 4.9: Mediciones reales de resistencias

Con lo anterior, la conversión de voltaje a corriente se realiza con las siguientes operaciones:

$$i[mA] = \frac{V_R[V]}{R[\Omega]} * 1000$$

Considerando que:

$$V_R[V] = \frac{V_o[V]}{Av[V/V]}$$

Por lo que primero se calcula la ganancia real Av_{real} como:

$$Av_{real} = \frac{49.4[k\Omega]}{R_G} + 1 = \frac{49.4[k\Omega]}{995} + 1 = 50.648\ [V/V]$$

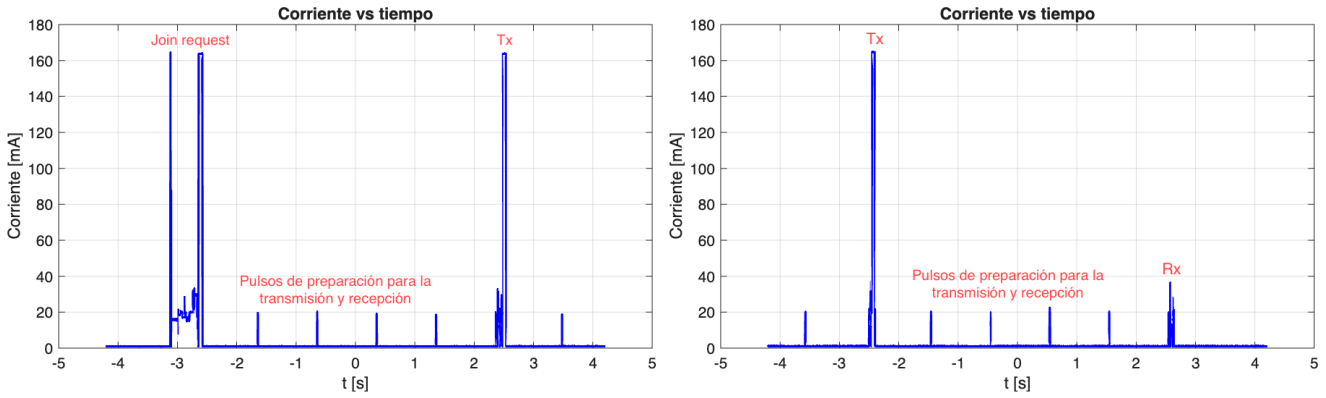
Finalmente, la operación se reduce a la siguiente:

$$i[mA] = \frac{V_o[V]}{R[\Omega]Av_{real}} * 1000 = \frac{V_o[V]}{2.02[\Omega]50.648} * 1000$$

De la misma forma que se hizo con el sistema de adquisición de datos hecho con el microcontrolador y el sensor INA226, para estas mediciones se han establecido las siguientes especificaciones:

- Se registran los eventos desde que se energiza el nodo terminal.
- Intervalo de envío de paquetes de 40 segundos.
- Se tiene la loza y el muro de concreto como obstáculos entre la línea de vista del nodo terminal y el Gateway.
- La frecuencia de muestreo fue fijada en *15.6 kSa/s* por el osciloscopio a partir de la escala horizontal establecida manualmente que permitió la captura de los eventos completos.
- El tiempo de grabación esta limitado a 1 minuto por especificación del fabricante en su versión de software gratuito.

Después de haber programado en MATLAB la conversión de voltaje a corriente, en $[mA]$, y del tiempo en $[s]$, al ejecutar el programa, este grafica los consumos captados para los eventos de *Join request* y un ciclo completo de transmisión y recepción, que se observan en las gráficas de la figura 4.10a y 4.10b, respectivamente.



(a) Join request.

(b) Transmisión y recepción.

Figura 4.10: Eventos capturados por el osciloscopio.

A pesar de que las mediciones se ven limitadas por el poco tiempo de grabación, las gráficas muestran más detalles sobre los pequeños picos de consumo entre la transmisión y las ventanas de recepción. En las mediciones hechas con el microcontrolador y el sensor INA226 (figura 4.4), apenas y se pueden notar estos pulsos, mientras que en la gráfica de la figura 4.10, obtenida con el osciloscopio, son claramente visibles.

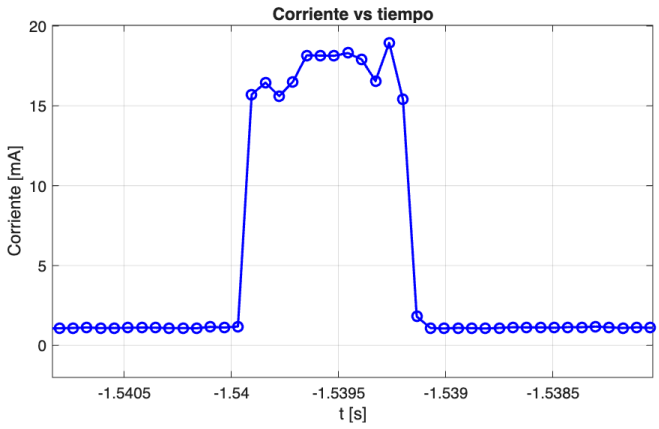
Lo anterior confirma el uso de estos eventos como parte del funcionamiento del nodo terminal bajo el protocolo de LoRaWAN en su clase A, al estar presentes antes de la transmisión y durante la recepción, se puede considerar que los eventos están relacionados con la sincronización y preparación de los eventos de transmisión y recepción.

Los valores promedio de consumo de corriente y duración de los eventos del ciclo de trabajo medidos hasta el momento con los sistemas de adquisición de datos empleados se ven resumidos en la tabla 4.3.

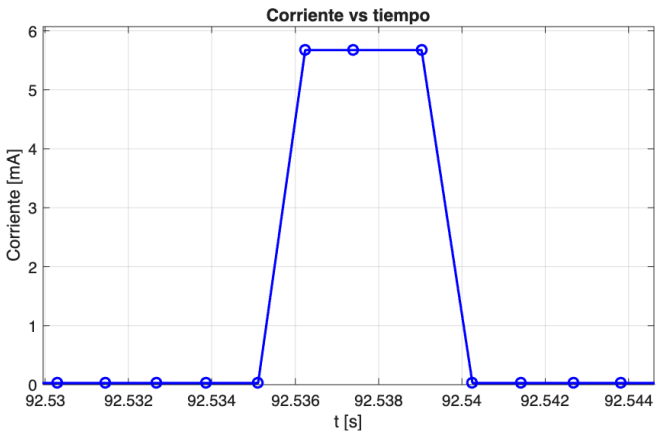
Modo o Evento	**Equipo**	**Corriente [mA]**	**Duración [ms]**
Join request	Osciloscopio	33.84	533
	Microcontrolador	26.16	546.25
Transmisión Tx	Osciloscopio	92.33	99.49
	Microcontrolador	83.17	99.95
Recepción Rx1	Osciloscopio	14.23	57.73
	Microcontrolador	8.29	76.05
Pulsos de preparación para la transmisión y recepción	Osciloscopio	12.99	0.8981
	Microcontrolador	4.51	1.06
Sleep	Osciloscopio	0.04	–
	Microcontrolador	0.03	–

Tabla 4.3: Valores promedio de pulsos de consumo de corriente en condiciones ideales

La diferencia entre la resolución de muestreo del microcontrolador y el osciloscopio se nota en los eventos o pulsos de preparación de transmisión y recepción descritos, la definición en la forma del pulso obtenida por el osciloscopio (figura 4.11a) es claramente mayor respecto a la forma del pulso captado por el microcontrolador (figura 4.11b).



(a) Pulsos vistos por el osciloscopio a *15.6 kSa/s*.



(b) Pulsos vistos por el microcontrolador a *0.909 kSa/s*.

Figura 4.11: Comparación de pulsos de preparación para la transmisión y recepción entre el osciloscopio y el microcontrolador.

A partir de las mediciones de la tabla 4.3 se puede concluir que las magnitudes de corriente en la transmisión concuerdan con las medidas con el sensor INA226, no obstante, existe una

diferencia considerable en las ventanas de recepción y los pulsos de preparación para la transmisión y recepción, la causa principal de esta diferencia se puede asociar a la conversión de voltaje a corriente dados por los errores entre los valores de resistencias medidos respecto a los reales, ya dichos valores de resistencia fueron obtenidos por un multímetro y no por un equipo especializado para medir impedancia.

En cuanto a la corriente medida para el nodo terminal en estado sleep, esta se puede observar y/o determinar con mayor precisión al cambiar la escala del osciloscopio, el resultado se muestra en la figura 4.12. Como se puede notar, aunque se presenta gran cantidad de ruido, la señal captada ronda los $40[\mu A]$, lo cual se acerca a lo medido con el multímetro para el sistema anterior, no obstante, es un valor lejano a lo especificado por el fabricante, por lo que restara verificar si estos valores son correctos a partir de las mediciones realizadas en la siguiente sección.

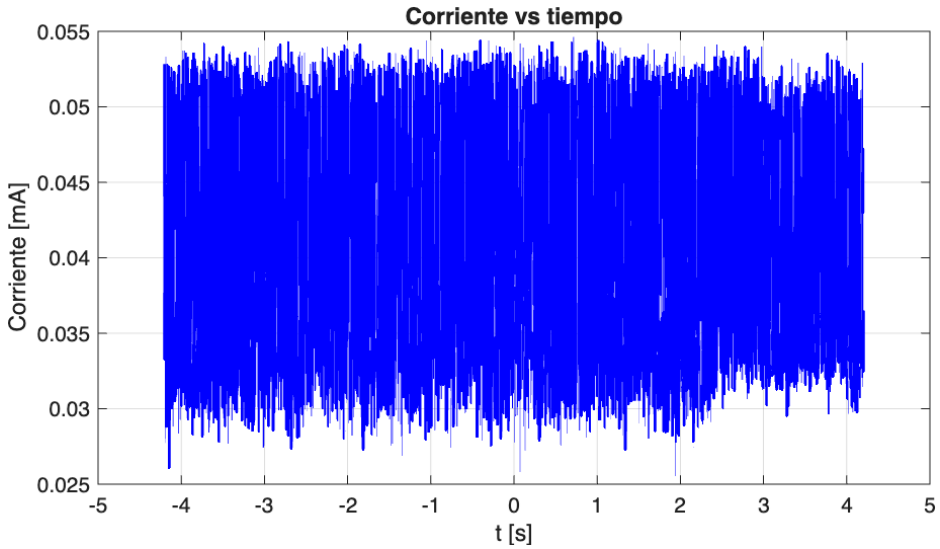


Figura 4.12: Corriente en modo sleep medida con el osciloscopio.

4.3.3 Medición de corriente con el equipo Otii Arc pro

Este equipo de medición de corriente, mostrado en la figura 4.13, promete ser mejor en la adquisición de datos respecto a los sistemas desarrollados y presentados anteriormente, esto se puede confirmar a partir de sus especificaciones, resumidas en la tabla 4.4. Además de las ventajas en sus especificaciones, su facilidad de uso sumada con el software Otii 3, que permite al usuario configurar los parámetros del equipo, así como visualizar y manipular los datos adquiridos, lo convierten en la opción más adecuada para las necesidades de la aplicación.

Figura 4.13: Otii Arc pro [16].

Especificación	**Detalle**
Precisión de corriente	▪ < 19 mA: $\pm(0.1\,\% + 50$ nA) ▪ > 19 mA: $\pm(0.1\,\% + 150$ µA)
Resolución de corriente	5 nA
Convertidor ADC	24 bits con cambio automático entre rangos
Precisión de voltaje	$\pm(0.1\,\% + 1.5$ mV)
Frecuencia de muestreo	▪ Canal principal: 4 kSa/s ▪ Secundarios: 1 kSa/s (voltaje, corriente ADC, etc.)
Rangos de alimentación	**Modo USB:** 0.5–3.75 V (auto) / 0.5–4.2 V (alto) **Modo DC:** 0.5–4.55 V (auto) / 0.5–5.0 V (alto)
Corriente de operación	–2.5 A a 5 A

Tabla 4.4: Especificaciones técnicas del analizador de potencia Otii Arc Pro [16].

Dentro de las especificaciones destacan su resolución de hasta 5 nA, precisión del 0.1 %, y una frecuencia de muestreo de hasta 4 kSa/s, lo que lo convierte en una opción potente para estudios de consumo energético del nodo terminal.

La descarga del software Otii 3 se hace desde la página oficial de descargas de Otii. La figura 4.14 muestra la ventana donde aparecen las descargas de acuerdo al sistema operativo. Para poder descargar hay que aceptar los términos del fabricante y dar clic en el archivo según el sistema operativo, que en este caso será la versión para Windows.

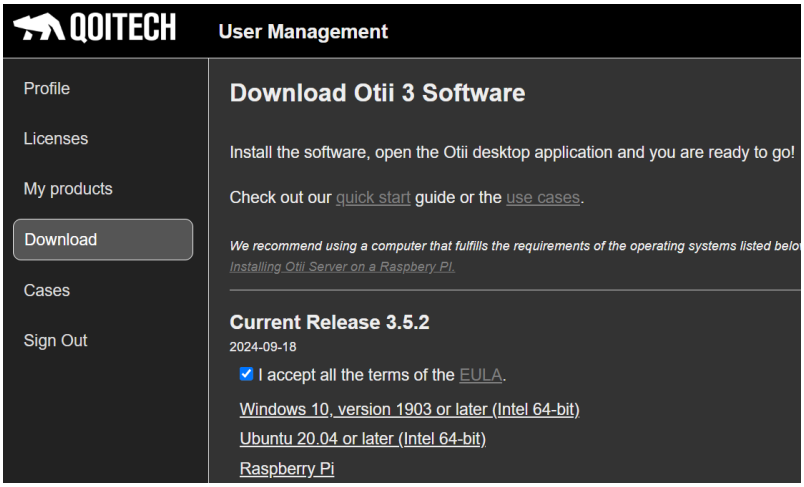


Figura 4.14: Enlaces de descarga de Otii 3.

El software Otii 3 no requiere de ninguna instalación, por lo que se ejecuta inmediatamente después de hacer doble clic. En la figura 4.15 se tiene la vista que genera el software Otii 3 al ser ejecutado por primera vez, en esta ventana se puede abrir un antiguo proyecto o crear uno nuevo, para este caso se crea un nuevo proyecto.

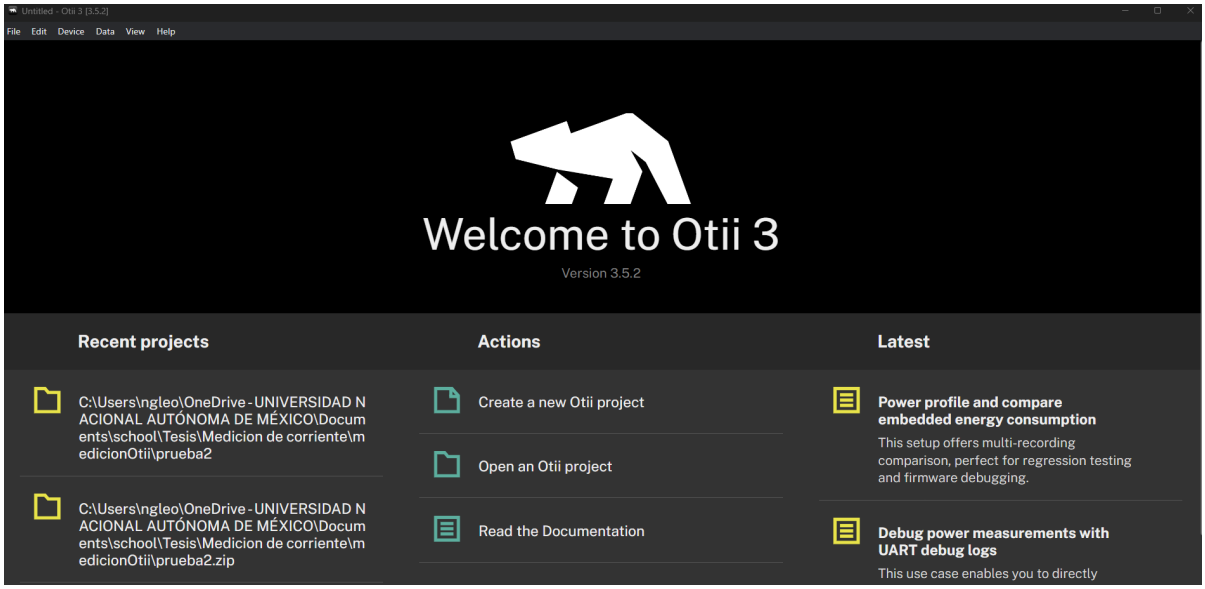
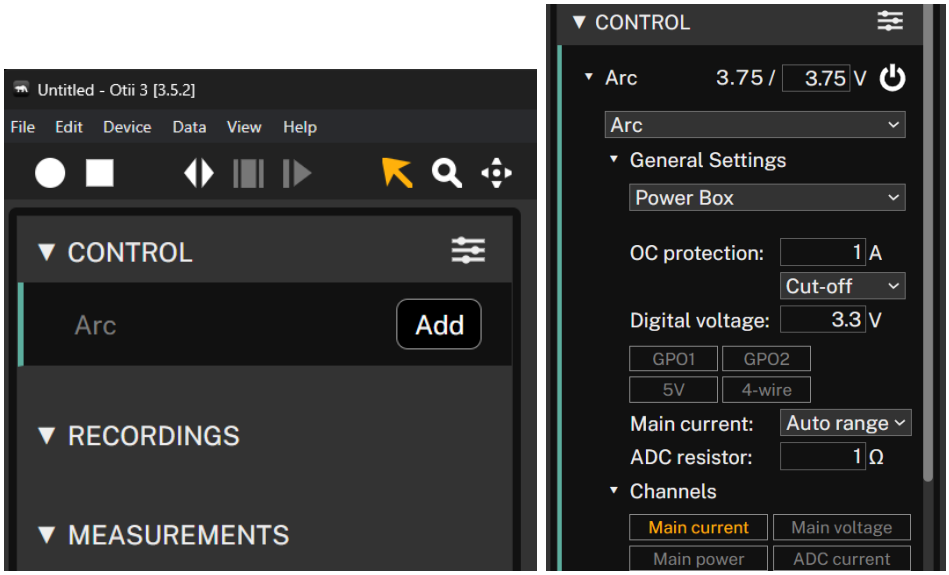


Figura 4.15: Pantalla de inicio Otii 3.

Para la conexión y puesta en funcionamiento del equipo, basta conectarlo mediante cable micro USB a la computadora, el software reconocerá automáticamente el dispositivo, puesto que aparece un botón con la etiqueta *Add* en el apartado de control de la ventana del nuevo proyecto, esto se ilustra en la figura 4.16a. Al dar clic en el botón *Add*, se activan todos los apartados de configuración

del equipo y archivos del proyecto, la figura 4.16b muestra la configuración del equipo con base en las siguientes especificaciones de operación del sistema:

- En el apartado de *Ajustes generales*:
  - Voltaje de alimentación del nodo terminal (3.7 [V]).
  - Corriente máxima de protección (1 [A])
  - Rango de corriente automático
  - Resistencia de ADC (1 [Ω])
- En el apartado de *Cananles*
  - Corriente principal



(a) Agregar el dispositivo. (b) Configuraciones.

Figura 4.16: Configuración del Otii arc Pro en el software Otii 3.

El nodo terminal es alimentado por el Otii arc pro, de acuerdo con la conexión mostrada en la figura 4.17. Cabe mencionar que la fuente se enciende y apaga con el botón de encendido que aparece en la esquina superior derecha del apartado de control del software Otii, que ya se presentó en la figura 4.16b.

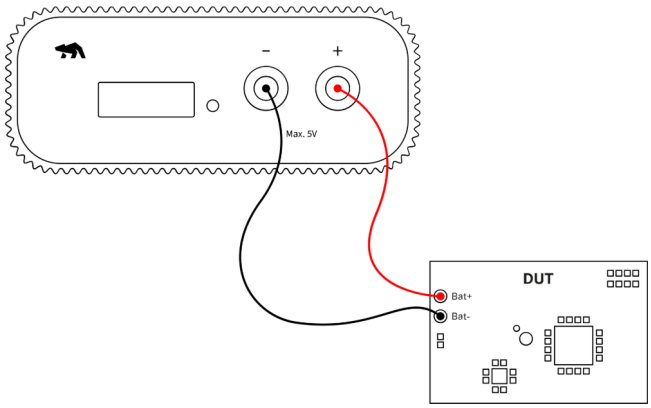


Figura 4.17: Alimentación de la carga con el Otii Arc pro [16].

Finalmente, se pueden iniciar y detener las grabaciones con los botones de control que aparecen en la barra superior de la figura 4.16a. Todas las grabaciones se visualizan en tiempo real mientras están en proceso, las grabaciones se exportan a un archivo en formato .csv al dar clic derecho en la grabación, ubicada en el apartado de *grabaciones*, debajo del apartado de control del Otii Arc pro (figura 4.18) y seleccionar la opción de *Export to CSV*.

Figura 4.18: Exportación grabación Otii 3.

Continuando con la metodología de los sistemas de adquisición de datos anteriores, para estas mediciones se establecieron las siguientes especificaciones, como siempre, priorizando la máxima frecuencia de muestreo.

- Se registran los eventos desde que se energiza el nodo terminal.

- Intervalo de envío de paquetes de 40 segundos.
-
- Se tiene la loza y el muro de concreto como obstáculos entre la línea de vista del nodo terminal y el Gateway.
- Frecuencia de muestreo de $4kSa/s$, por especificación del equipo Otii Arc pro.

Los archivos resultantes, al igual que los obtenidos en los métodos anteriores, se graficaron en MATLAB. Un ejemplo de las mediciones resultantes se presenta en la figura 4.19, en la cual se muestran cuatro ciclos de trabajo, con todos los estados ya observados en los sistemas anteriores.

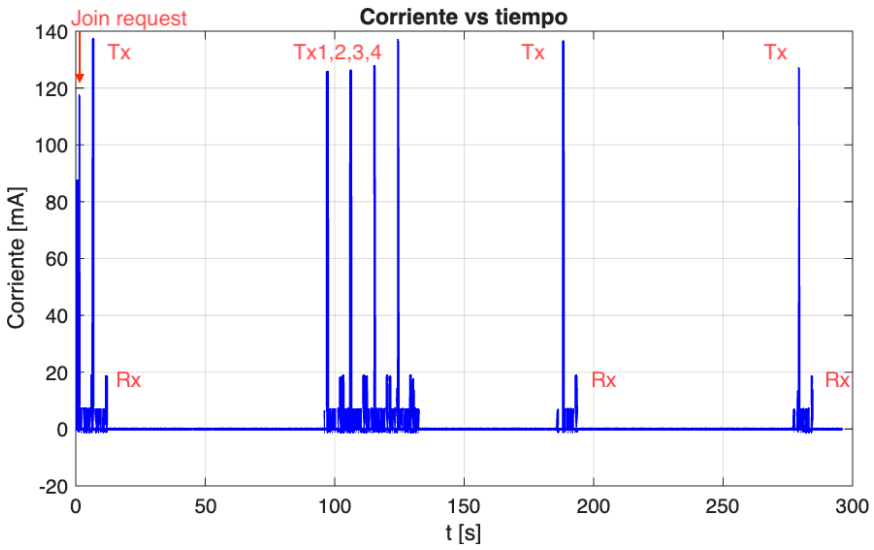


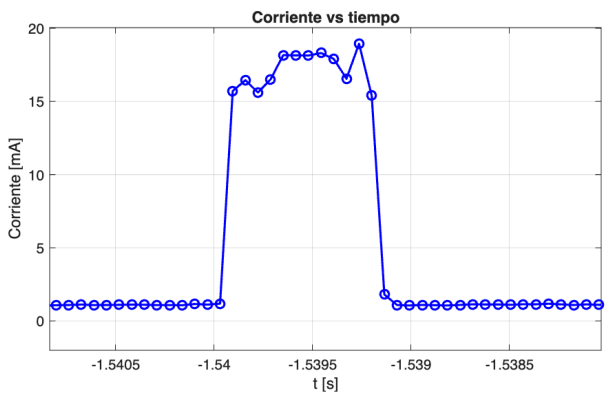
Figura 4.19: Gráfica de consumo de corriente captada con el equipo Otii Arc pro.

En cuanto a las magnitudes, mientras que en los datos arrojados por el microcontrolador los consumos en las transmisiones rondan los 150-160 mA, los valores dados por el equipo Otii están entre 120 y 140 mA, lo que se debe considerar en la elección final del sistema que se usará para calcular la vida útil de las baterías del nodo terminal. Lo anterior se ve reflejado en la tabla comparativa 4.5, donde se resumen los valores promedio recopilados entre los tres sistemas.

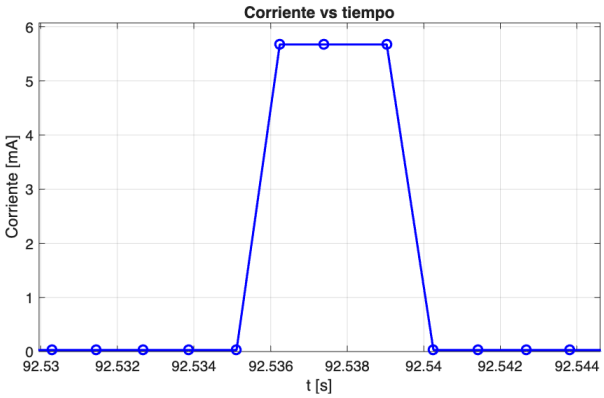
Modo o Evento	**Equipo**	**Corriente [mA]**	**Duración [ms]**
Join request	Osciloscopio	33.84	533
	Microcontrolador	26.16	546.25
	Otii Arc Pro	24.87	530.25
Transmisión Tx	Osciloscopio	92.33	99.49
	Microcontrolador	83.17	99.95
	Otii Arc Pro	72.02	105.62
Recepción Rx1	Osciloscopio	14.23	57.73
	Microcontrolador	8.29	76.05
	Otii Arc Pro	7.96	59.74
Pulsos de preparación para la transmisión y recepción	Osciloscopio	12.99	0.8981
	Microcontrolador	4.51	1.06
	Otii Arc Pro	3.31	2
Sleep	Osciloscopio	0.04	-
	Microcontrolador	0.03	-
	Otii Arc Pro	0.013	-

Tabla 4.5: Valores promedio de pulsos de consumo de corriente captados por los tres sistemas de adquisición de datos.

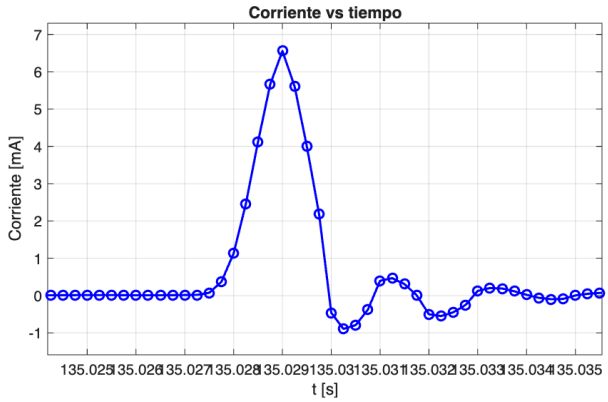
La diferencia entre la resolución de las muestras es apreciable en las figuras 4.20a, 4.20b y 4.20c, en dichas gráficas se comparan, igual que se hizo anteriormente, los pulsos de preparación para la transmisión y recepción, es notorio que la muestra dada por el osciloscopio capta con mayor detalle la forma del pulso, lo que en primera instancia, es mejor para los cálculos precisos. A pesar de que el osciloscopio arroja mayor detalle en la forma del pulso, la poca capacidad de grabación, la inexactitud de la medición en estado sleep y los errores de medición provocados por los valores de resistencia siguen siendo limitantes que el equipo Otii Arc pro no presenta.



(a) Pulsos vistos por el osciloscopio.



(b) Pulsos vistos por el microcontrolador.



(c) Pulsos vistos por el equipo Otii.

Figura 4.20: Comparación de pulsos de preparación para la transmisión y recepción entre los sistemas de adquisición de datos.

Tomando en cuenta lo anterior mencionado, sumado con la mayor exactitud en la magnitud de sus mediciones, se considerarán únicamente datos recabados con el equipo Otii para los posteriores cálculos del consumo de energía por ciclo de trabajo y la estimación de la vida útil de las baterías.

4.4 Estimación de la vida útil de las baterías eléctricas

Esta tesis se enfoca principalmente en la estimación de la vida útil de las baterías con base en los datos adquiridos por los sistemas de la sección anterior, a pesar de ello se hizo un cálculo basado en la teoría de LoRa y LoRaWAN, con la única finalidad de hacer una comparación entre los resultados arrojados por medio de los valores teóricos y los medidos en las pruebas. Para dicha estimación se propusieron los siguientes pasos:

1. Caracterizar cada uno de los pulsos de consumo medidos.
2. Usar un modelo matemático adecuado para el cálculo teórico de energía consumido por el

nodo en un ciclo de trabajo.

3. Obtener un valor de energía consumida por ciclo de trabajo con base en las mediciones de corriente.
4. Estimar la vida útil de una batería con determinada capacidad a partir de los resultados teóricos y experimentales.
5. Comparar resultados de los cálculos anteriores.

4.4.1 Análisis y caracterización de la actividad en el ciclo de trabajo del nodo terminal

Identificar e interpretar cada uno de los pulsos de consumo que se observan durante el ciclo de trabajo del nodo terminal se logra con el apoyo del sitio The Things Network (TTN), ya que este despliega la información sobre el envío de paquetes, la confirmación de recepción de los paquetes, así como otros procesos que se realizan entre el nodo terminal y el Gateway central.

Para el análisis se registraron todos los eventos transcurridos en dos transmisiones, incluidos las solicitudes de *Join request*, para ello se estableció el tiempo de transmisión entre paquetes de 60 segundos. En la figura 4.21 se muestran los consumos del caso descrito, cabe aclarar que este análisis se hizo con datos recabados por el sistema de microcontrolador, ya que era con el único equipo con se contaba.

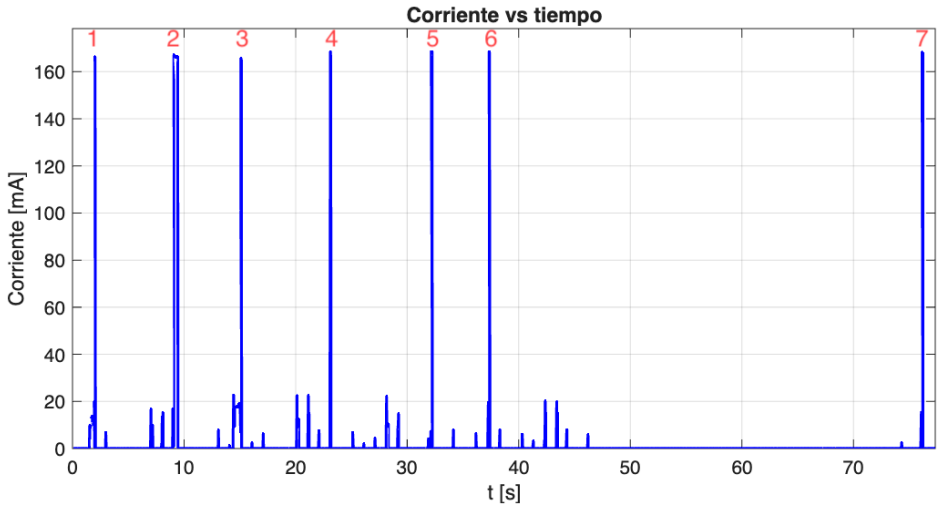


Figura 4.21: Gráfica de consumo para el análisis de actividad del nodo terminal.

Figura 4.22: Mensajes de la plataforma TTN entre nodo y Gateway.

Con ayuda de los datos desplegados por la plataforma TTN, mostrados en la figura 4.22, se logró identificar a qué etiqueta corresponde cada uno de los pulsos de consumo registrados.

- Los dos primeros pulsos con la mayor magnitud (pulso 1 y 2), corresponden a solicitudes de conexión o *Join request*, la existencia de dos pulsos indica que el segundo de ellos es un reintento del *oin request* .
  - Estos se identifican en la plataforma TTN como los primeros dos mensajes con la etiqueta: *"Forward Join-accept message"*.
- El tercer pulso (pulso 3) de mayor magnitud observado en la gráfica corresponde al primer envío de datos.
  - Estos pulsos de envío de datos aparecen en TTN como *"Forward uplink data message"*.
- Como el CubeCell fue configurado para enviar datos cada sesenta segundos, el séptimo pulso de mayor magnitud observado en la gráfica corresponde al segundo paquete de datos recibido (pulso 7).

Entre los dos primeros pulsos de *Join request* con etiqueta *"Forward Join-accept message"* se pueden observar dos ventanas de consumo de corriente que corresponden a los siguientes mensajes:

- La primera ventana es del mensaje con la etiqueta "*Accept Join-request"* debido a que transcurrieron 5 segundos entre ese mensaje y el primer pulso de envío.
- La segunda ventana, dado que ocurre en el mismo segundo que la primera ventana, corresponde al mensaje con la etiqueta *"Successfully processed Join"*.

Se logró identificar que las ventanas de recepción, así como la posterior al *Join request*, están asociadas a los mensajes *"Successfully processed data"*

Después del primer envío de datos (pulso 3) existen tres picos de corriente de amplitud similar a los de envío de datos, los cuales coinciden con los tres mensajes que llevan la etiqueta *"Successfully processed data"*, por lo que estos picos de corriente, además de coincidir con las etiquetas mencionadas, significan eventos de retransmisión de datos ocasionados por una mala recepción del paquete por parte del Gateway central, esto concuerda con la programación del nodo terminal, donde se definió un número máximo de cuatro intentos para el envío de datos.

Finalmente, los pequeños pulsos llamados *pulsos de preparación para la transmisión y recepción* en las mediciones, están relacionados con los mensajes de TTN que tienen la etiqueta *"Schedule data downlink for transmission on Gateway Server"*

Con el análisis anterior y con base en todas las mediciones realizadas con el equipo Otti Arc pro bajo las condiciones de transmisión descritas en su sección correspondiente, en la tabla 4.6 se encuentran registrados los valores promedio de duración y consumo de corriente para cada evento del ciclo de trabajo. El consumo promedio de corriente se calculó para todo el tiempo en que el nodo terminal está activo, este cálculo fue hecho con la herramienta de hojas de cálculo Excel, específicamente con la función que obtiene el promedio de los valores en el rango de tiempo definido, el cual está dado por los índices con los que se determinó la duración de los eventos.

Tanto para la tabla 4.6, como para las tablas 4.1, 4.3 y 4.5, los valores se determinaron a partir de un total de 10 muestras de cada evento. Para la duración de los eventos, al igual que para el cálculo de consumo promedio de corriente, se contabilizó desde que el primer instante en que la gráfica o el evento cambia de estado sleep y hasta que vuelve entrar, es decir, se considera todo el proceso en que el nodo terminal está activo, y no sólo la parte donde transmite. Como observación particular, la tabla también incluye los valores asociados a una segunda ventana de recepción.

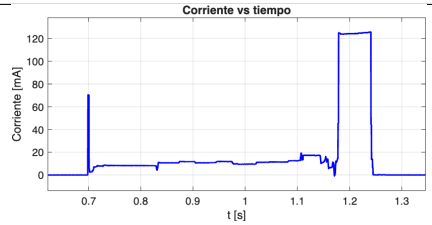
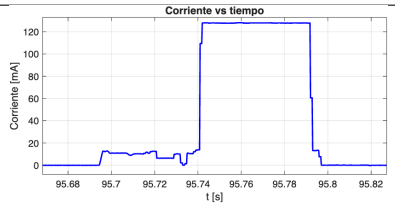
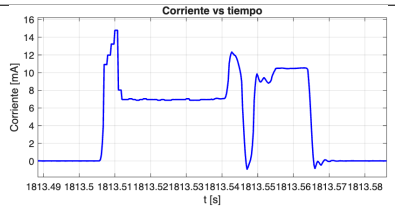
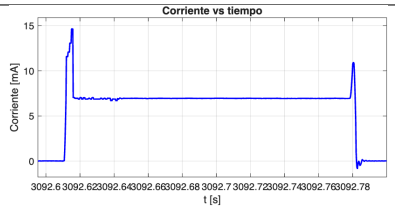
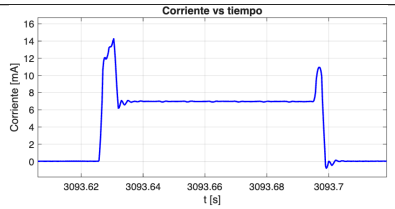
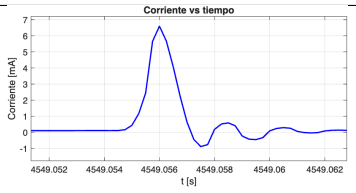
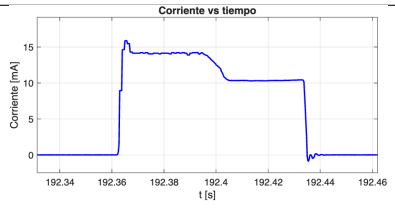
Modo o Evento	**Consumo [mA]**	**Duración [ms]**	**Imagen**
Join request	24.87	530.25	
Transmisión Tx	72.02	105.62	
Recepción Rx	7.96	59.74	
Recepción Rx1	7.07	171.52	
Recepción Rx2	7.21	73.25	
Pulsos de preparación para la transmisión y recepción	3.31	2	
Pulso medidor de flujo	12.66	53.25	
Sleep	0.013	-	-

Tabla 4.6: Eventos de ciclo del trabajo.

Después de haber analizado el comportamiento del nodo terminal se llegó a la conclusión de que la energía que consumirá el nodo terminal depende totalmente de las características de su ubicación, puesto que de ello depende el número de retransmisiones, doble ventana de recepción y el Spread Factor, por lo anterior, se decidió realizar mediciones en dos escenarios diferentes: en el que exista una cantidad abundante de retransmisiones, con el cambio en el Spread Factor que esto implique y otro donde haya casi nula presencia de retransmisiones. Estas mediciones se realizaron en conjunto con las pruebas de campo documentadas en el capítulo cinco de esta tesis.

4.4.2 Cálculo teórico de energía por ciclo de trabajo

El consumo energético de un nodo terminal por ciclo de trabajo está estrictamente relacionado con la clase y método de activación con el que esté configurado, de acuerdo con la teoría para la versión de LoRaWAN presentada en el marco teórico de la sección 2.2.3, cuando el nodo terminal opera en la clase A mediante OTAA, se producen los siguientes eventos:

- Join request del nodo terminal hacia la red.
- Join accept de la red hacia el nodo terminal

Una vez el nodo terminal se conecta a la red, este envía su primer *uplink* o paquete de datos, lo que genera los siguientes eventos según la clase A de LoRaWAN:

- Uplink o envío de datos.
- Ventana de recepción 1.
- Ventana de recepción 2.

Considerando lo anterior se puede construir la gráfica de la figura 4.23, similar a las obtenidas con las mediciones, pero con base en las especificaciones (tabla 3.5) de consumo de corriente de los chips que componen la tarjeta de desarrollo usada y con base en el tiempo en el aire de cada paquete generado, a partir del tamaño de su respectivo payload.

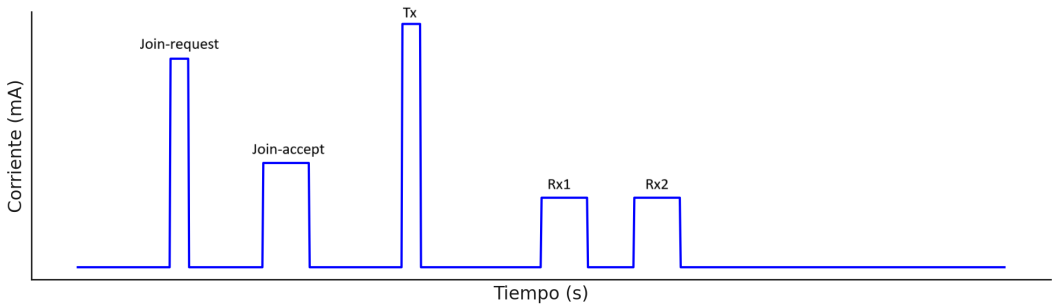


Figura 4.23: Gráfica teórica de consumo para el nodo bajo OTA y la clase A de LoRaWAN.

Debido a la diferencia entre los valores de corriente especificados entre las diferentes hojas de datos, se propuso hacer una medición de los consumos asociados a cada potencia con ayuda de los programas de LoRa en su capa física proporcionados por el fabricante y el equipo Otii Arc pro, las condiciones son las siguientes:

- Se usará programa ejemplo de recepción y transmisión proporcionado por el fabricante Heltec.
- Las tarjetas estarán a una distancia cercana con el fin de tener buena comunicación.
- El único valor que se cambiará según la prueba es la potencia de transmisión.
- Se establecen los siguientes parámetros de configuración de LoRa:
  - SF: 7
  - BW: 125 [KHz]
  - CR: 4/5
  - Preámbulo: 8 símbolos

El resultado de las mediciones se ve reflejado en la tabla 4.7, para la tabla se obtuvo el promedio de consumo de corriente de un total de 10 muestras para cada caso. Adicionalmente, se realizaron las mismas mediciones de corriente con las mismas potencias, pero modificando el spread factor, con el objetivo de determinar si la corriente cambia respecto al spread factor. Al observar que el spread factor no afecta la corriente demandada se decidió no incluir estos valores en la tabla.

Recepción	
Modo	**Corriente promedio [mA]**
RX	7.31
Transmisión	
Potencia [dBm]	**Corriente promedio [mA]**
5	78.74
10	102.5
14	127.45
16	140.95
18	155.39
20	165.13
21	169.01
22	172.31

Tabla 4.7: Consumo de corriente en recepción y transmisión a diferentes potencias de salida.

La duración de los eventos o tiempo en el aire de cada paquete se obtiene de la suma del tiempo necesario para generar los símbolos del preámbulo de la capa física de LoRa y del tiempo requerido para el generar el payload, compuesto por todos los bytes de información necesarios y/o agregados por la capa física de LoRa y el protocolo LoRaWAN en su versión 1.0.2, especificada en la sección

2.2.3. Conociendo lo anterior, y de acuerdo lo especificado en documentos como [46] y [40], las expresiones necesarias para el cálculo son las siguientes:

Tiempo en el aire

$$T_{packet}[s] = (N_{preamble} + N_{payload}) \cdot T_s[s] \tag{4.2}$$

Tiempo de símbolo

$$T_s[s] = \frac{2^{\mathrm{SF}}}{\mathrm{BW}} \tag{4.3}$$

Donde:

- SF: Spreading Factor
- BW: Bandwidth [Hz]

Número de símbolos de preámbulo

$$N_{preamble} = n_{preamble} + 4.25 \tag{4.4}$$

Donde:

- $n_{preamble}$: Longitud del preámbulo [símbolos].

Número de símbolos del payload

$$N_{payload} = 8 + \text{máx}\left(ceil\left[\frac{8PL - 4SF + 28 + 16CRC - 20H}{4(SF - 2DE)} \cdot (CR + 4)\right], 0\right) \tag{4.5}$$

Donde:

- PL: Longitud del payload [bytes]
- CRC : CRC habilitado o deshabilitado
- H : Encabezado implícito o explícito
- DE : Optimización de baja tasa de datos
- CR: Coding Rate (4/5, 4/6, etc)

Mientras que para la longitud del payload en los eventos representados en la figura 4.23, de acuerdo con la especificación de LoRaWAN en la región US915, se tiene lo siguiente:

- Join request: 23 bytes
- Join accept: 17 bytes (sin CFList bytes)
- uplink: 17 bytes (considerando una carga útil de 4 bytes y sin FOpts bytes)

- Ventana Rx1: 12 bytes
- Ventana Rx2: 12 bytes

Con la información obtenida y con base en trabajos como el realizados por [37] o [47], se pueden construir modelos para evaluar el desempeño energético de LoRaWAN, no obstante, para este cálculo se iniciará específicamente con el modelo empleado en la tesis [40], donde se calcula la energía consumida a partir de la ecuación 4.6

$$E[\text{Wh}] = P[\text{W}] \cdot t[\text{h}] \tag{4.6}$$

Donde t[h] es el tiempo en el que se aplica la potencia, P[W] es la potencia en Watts y se calcula como:

$$P[\text{W}] = I[A] \cdot V_{cc}[V]$$

De modo que, al sustituir en la expresión 4.6, la energía consumida se calcularía como:

$$E[\text{Wh}] = I[A] \cdot V_{cc}[V] \cdot t[\text{h}] \tag{4.7}$$

La expresión 4.7 resulta útil para obtener la energía consumida por periodo de tiempo, no obstante, para este cálculo y análisis, más que la energía se requiere conocer la carga eléctrica consumida; esto porque las baterías especifican su carga, no su energía, por lo que al tener la carga consumida por ciclo se evita una operación extra. De esta manera, la expresión 4.7 se puede reescribir en la forma de la expresión 4.8, la cual es fácil de convertir a unidades de mAh.

$$C[\text{C}] = C[A \cdot s] = I[A] \cdot t[\text{s}] \tag{4.8}$$

Al descomponer las corrientes y los tiempos de cada evento descrito, considerando el tiempo en que el nodo está en modo sleep, la expresión resultante es:

$$\begin{aligned} C[\text{mAh}] = \Big[&(I_{\text{JR}} \cdot t_{\text{JR}}) + (I_{\text{JA}} \cdot t_{\text{JA}}) + (I_{\text{Tx}} \cdot t_{\text{Tx}}) + (I_{\text{Rx1}} \cdot t_{\text{Rx1}}) \\ &+ (I_{\text{Rx2}} \cdot t_{\text{Rx2}}) + (I_{\text{sleep}} \cdot t_{\text{sleep}}) \Big] \cdot \frac{10}{36} \end{aligned} \tag{4.9}$$

El consumo de carga que se obtiene con la expresión 4.9 corresponde a un ciclo de trabajo que abarca desde el *Join request* hasta la segunda ventana de recepción, lo cual es adecuado si se desea conocer toda esta carga, sin embargo, para la estimación de la vida útil de las baterías del nodo terminal se deben considerar únicamente los eventos de transmisión y recepción. El hecho

de considerar únicamente los eventos de transmisión y recepción para la estimación de la vida útil de los nodos es debido al mecanismo de conexión OTAA, pues este sólo requiere del primer *Join request* y *Join accept* para establecer conexión con el Gateway, una vez establecida la conexión, el protocolo LoRaWAN la desactiva con el fin de ahorrar energía.

Habiendo definido lo anterior, el primer paso para obtener la carga consumida por ciclo de trabajo es calcular el tiempo en el aire para cada mensaje, para esto se elaboró una hoja de cálculo que efectúa las operaciones necesarias. En la figura 4.24 se presentan los valores obtenidos.

SF	CR	BW [kHz]	Preambulo [simbolos]	CRC	H	DE	Payload Tx [bits]	Payload Rx [bits]
7	1.00	125	8	1	1	0	23	17

	Ts [ms]	TPreamble[ms]	TPayLoad [ms]	Time on Air [ms]	BitRate [bps]
TX	1.024	12.544	44.032	56.576	5468.75
RX	1.024	12.544	33.792	46.336	5468.75

(a) Tiempo en el aire para el Join request y Join accept.

SF	CR	BW [kHz]	Preambulo [simbolos]	CRC	H	DE	Payload Tx [bits]	Payload Rx [bits]
7	1.00	125	8	1	1	0	17	12

	Ts [ms]	TPreamble[ms]	TPayLoad [ms]	Time on Air [ms]	BitRate [bps]
TX	1.024	12.544	33.792	46.336	5468.75
RX	1.024	12.544	28.672	41.216	5468.75

(b) Tiempo en el aire para el uplink y ventanas de recepción.

Figura 4.24: Tiempo en el aire de los eventos del ciclo de trabajo del nodo terminal.

A partir de la información recabada y generada a lo largo de esta sección, se construye la tabla 4.8, en la cual se definen y simplifican los valores de corriente y tiempo obtenidos para cada evento. El objetivo es visualizar de manera sencilla cómo se compone un ciclo de trabajo para el nodo terminal.

Potencia [dBm]	Evento	SF	BW [kHz]	CR	Corriente [mA]	Duración [ms]	Carga [mAh]
14	Join request	7	125	4/5	127.45	56.592	0.002004
	Join accept	7	125	4/5	7.310	46.336	0.000094
	Uplink	7	125	4/5	127.45	46.336	0.001640
	Recepción 1	7	125	4/5	7.310	41.216	0.000084
	Recepción 2	7	125	4/5	7.310	41.216	0.000084
	Sleep	–	–	–	0.0035	1799780.304	0.001750
	Carga total	–	–	–	–	–	**0.005655**
20	Join request	7	125	4/5	165.13	56.592	0.002596
	Join accept	7	125	4/5	7.310	46.336	0.000094
	Uplink	7	125	4/5	165.13	46.336	0.002125
	Recepción 1	7	125	4/5	7.310	41.216	0.000084
	Recepción 2	7	125	4/5	7.310	41.216	0.000084
	Sleep	–	–	–	0.0035	1799780.304	0.001750
	Carga total	–	–	–	–	–	**0.006733**

Tabla 4.8: Consumos de carga totales y por evento para un intervalo de transmisión de 30 minutos.

De este modo, la duración de la vida útil de la batería se estima con base en el número de eventos o transmisiones totales, los cuales se calculan a partir de la capacidad de la batería y la carga consumida por evento de transmisión: (Uplink, ventanas de recepción y tiempo en modo sleep):

$$Eventos\ totales = \frac{capacidad\ de\ la\ batería}{consumo\ por\ ciclo\ de\ trabajo}$$

Al excluir el consumo asociado al *Join request* y *Join accept*, según se explicó, se tiene una carga de $0.004043[mAh]$ considerando una potencia de transmisión de $20dBm$, de este modo, al suponer una batería con capacidad de $1000[mAh]$ se tiene que:

$$Eventos\ totales = \frac{1000[mAh]}{0.004043[mAh]} \approx 247,341$$

Por último, con la transmisión fijada en 30 minutos, se presentan 48 eventos por día, dando como resultado que, el total de días de transmisiones posibles es:

$$Días\ de\ transmisiones = \frac{Eventos\ totales}{Eventos\ por\ día} = \frac{247{,}341}{48} \approx 5,152 \text{ días} \approx 14.11 \text{ años}$$

Este resultado a simple vista es muy optimista, pues supera el valor esperado y/o propuesto en el objetivo de la tesis, a pesar de ello, al compararlo con los resultados obtenidos por [40], cuyo modelo (similar al empleado en este trabajo) considera igualmente los parámetros de modulación, composición de trama de paquetes y eventos del ciclo de trabajo y, que son considerados y modelados de manera similar en otros trabajos como [37] y [38]; resulta que los 14.11 años estimados, no son un valor alejado de la realidad conociendo los trabajos y dispositivos existentes. Aunque este método basado en un modelo es prometedor, las conclusiones importantes provienen del cálculo basado en datos de mediciones experimentales.

4.4.3 Cálculo experimental de energía por ciclo de trabajo

El cálculo experimental de la energía o carga eléctrica que el nodo terminal consume por ciclo de trabajo, es con base en todas las mediciones realizadas en las pruebas de campo del capítulo 5. El objetivo principal es obtener un valor de carga, en mAh y, finalmente, estimar la vida útil que el nodo terminal tendría considerando una capacidad específica de la batería instalada y el periodo de transmisión especificado para la aplicación de PUMAGUA.

La propuesta para este cálculo es determinar el área bajo la curva de cada ciclo de trabajo, de modo que se pueda encontrar un valor promedio de consumo. Se propuso esta metodología ya que, del análisis de la actividad del nodo, resultó ser que los diferentes eventos no siempre tienen las mismas características en cuanto a transmisión y recepción, por lo que no basta con realizar cálculos entre muestras que tengan características similares.

Por último, se pretende asociar un consumo de corriente a cada ciclo de trabajo con la finalidad de apreciar la diferencia de consumo en aquellos ciclos de trabajo donde se presente una transmisión, dos retransmisiones y hasta tres retransmisiones, con lo anterior, sería posible obtener estadísticas sobre la frecuencia con la que se presentan las retransmisiones y su consumo asociado.

Energía consumida por ciclo de trabajo por el método del área bajo la curva

Este método surgió ya que al tener el área bajo la curva con los datos recabados las unidades son $mAms$, por lo que se puede hacer una conversión fácilmente para tener un resultado en mAh. Para obtener el área bajo la curva se requiere de métodos numéricos debido a la complejidad de las curvas que se producen, en esencia se hará uso de las fórmulas de Newton-Cotes, que son los tipos de integración numérica más comunes, los cuales se basan en la estrategia de reemplazar una función complicada o datos tabulados por un polinomio de aproximación que es fácil de integrar

[48].

Existen formas cerradas y abiertas de las fórmulas de Newton-Cotes. Las formas cerradas son aquellas donde se conocen los datos al inicio y al final de los límites de integración y las formas abiertas tienen límites de integración que se extienden más allá del intervalo de los datos [48], por lo tanto, para los datos recabados se requieren formas cerradas que cumplen con la siguiente expresión:

$$I = \int_a^b f(x)dx \cong \int_a^b f_n(x)dx$$

donde $f_n(x)$ = un polinomio de la forma $f_n(x) = a_0 + a_1x + \cdots + a_{n-1}x^{n-1} + a_nx^n$ y n es el orden del polinomio.

Dicho de manera más sencilla, es posible utilizar funciones o curvas de menor orden para aproximar la forma de una función más compleja.

Debido a la gran cantidad de datos almacenados en las mediciones, se usarán librerías existentes para MATLAB. Los métodos que se emplearán para dicho cálculo son: la regla del trapecio y la regla de Simpson; de modo que, a continuación, se presentan las principales características de cada método, con la única intención de comprender su funcionamiento.

1. **Regla del trapecio:** Este método consiste en aproximar el área bajo la curva al dividir el intervalo $[a, b]$ en n subintervalos de Δx ancho y sumar las áreas de cada trapecio formado en cada subintervalo, en la figura 4.25 se ilustra este comportamiento.

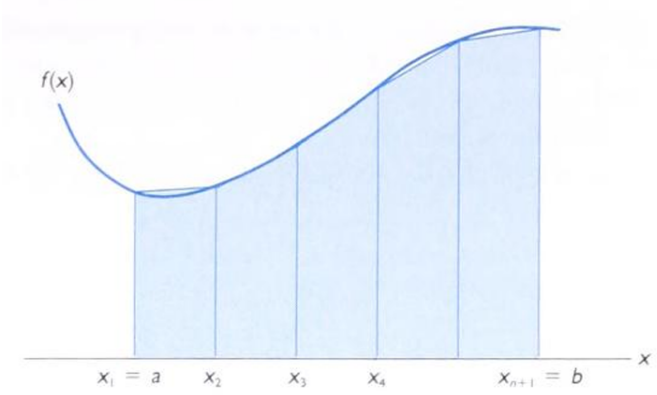


Figura 4.25: Aproximación del área bajo la curva de una función por la regla del trapecio [17].

Para conseguir lo anterior se debe considerar el área de un trapecio, según lo ilustra la figura 4.25, este se calcula como:

$$A = \Delta x \frac{f(x_i)+f(x_{i+1})}{2} = \frac{\Delta x}{2} \cdot (f_i + f_{i+1})$$

De este modo, se puede aproximar el área bajo la curva de la función $f(x)$ con la sumatoria de los n intervalos resultantes:

$$\int_a^b f(x)\,dx \approx \sum_{i=1}^{n} \frac{\Delta x}{2}(f_i + f_{i+1}) = \frac{\Delta x}{2}(f_1 + f_2 + f_2 + f_3 + \cdots + f_n + f_{n+1}) \tag{4.10}$$

De la expresión 4.10, se puede observar que los términos f_2 se repiten, este comportamiento se repite a medida que crece el número de intervalos, por lo que la expresión simplificada para el área bajo la curva de la función $f(x)$ en el intervalo $[a, b]$, subdividido en n intervalos de Δx ancho es:

$$\int_a^b f(x)\,dx = \frac{\Delta x}{2}(f_1 + 2f_2 + 2f_3 + \cdots + 2f_n + f_{n+1}) \tag{4.11}$$

Al contar con grabaciones donde se tiene registrado cada valor de corriente en cada instante de tiempo, el método se simplifica al sustituir los valores de $f(x_i)$ en la expresión 4.11
El término de error para la regla trapezoidal implica f'', por lo que la regla da el resultado exacto cuando se aplica a cualquier función cuya segunda derivada es idénticamente cero, es decir, cualquier polinomio de grado uno o menos [18].

2. **Regla de Simpson:** Similar a la regla del trapecio, esta regla consiste en aproximar la señal original con trazos de otras funciones, en lugar de usar rectas o polinomios de primer orden, la regla de Simpson usa polinomios de segundo y tercer orden, de ahí que existen las variantes conocidas como regla de Simpson $\frac{1}{3}$ y regla de Simpson $\frac{3}{8}$, respectivamente. Dado que las señales de corriente no se presentan curvas suaves de alto orden, sino que presenta cambios abruptos, aproximar con un polinomio de segundo grado es más que suficiente, por esta razón se decidió usar la regla de Simpson $\frac{1}{3}$.
   La curva $y = f(x)$ de la figura 4.26 se puede aproximar con un polinomio de Lagrange de segundo orden $y = P_2(x)$ cuando el intervalo $[a, b]$ se divide en los n subintervalos x_n.

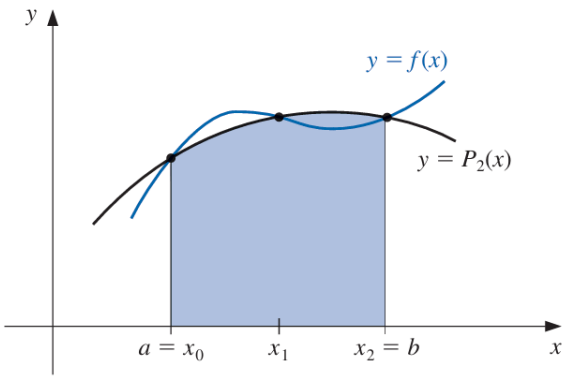


Figura 4.26: Aproximación del área bajo la curva de una función por la regla de Simpson $\frac{1}{3}$ [18].

Al usar el polinomio de Lagrange indirectamente se está interpolando la señal de corriente, esto se debe a que el polinomio de segundo orden aproxima la curva entre los puntos $f(x_0) = y_0$ y $f(x_1) = y_1$ de cada subintervalo de la siguiente forma: [18], [48]

$$P(x) = L_0(x)f(x_0) + L_1(x)f(x_1) + L_2(x)f(x_2)$$

$$P(x) = \frac{(x-x_1)(x-x_2)}{(x_0-x_1)(x_0-x_2)}f_0 + \frac{(x-x_0)(x-x_2)}{(x_1-x_0)(x_1-x_2)}f_1 + \frac{(x-x_0)(x-x_1)}{(x_2-x_0)(x_2-x_1)}f_2$$

Con lo anterior, al sustituir el polinomio de Lagrange, la integral de $f(x)$ en el intervalo $[a, b]$ se aproxima con la expresión:

$$\begin{aligned}\int_a^b f(x)\,dx = \int_{x_0}^{x_2} \Bigg[&\frac{(x-x_1)(x-x_2)}{(x_0-x_1)(x_0-x_2)}f_0 \\ &+ \frac{(x-x_0)(x-x_2)}{(x_1-x_0)(x_1-x_2)}f_1 \\ &+ \frac{(x-x_0)(x-x_1)}{(x_2-x_0)(x_2-x_1)}f_2 \Bigg] dx\end{aligned} \tag{4.12}$$

Existen diferentes formas de resolver la integral anterior, no obstante, de acuerdo con [18], la mejor forma es expandir $f(x)$ en el tercer polinomio de Taylor, por ende, al resolver la integral de la expresión 4.12 se obtiene la expresión 4.13:

$$\int_{x_0}^{x_2} f(x)\,dx \approx \frac{h}{3}\left[f_0 + 4f_1 + f_2\right] \tag{4.13}$$

Donde $h = x_2 - x_1 = x_1 - x_0$ es el ancho de cada subintervalo o espaciado entre muestras, el cual debe ser uniforme.

La expresión 4.13 aplica para el caso de la figura 4.26 donde solo hay 3 subintervalos, cuando este número de subintervalos crece, la expresión general obtiene la forma:

$$\int_a^b f(x)\,dx \approx \frac{h}{3}\left[f_1 + 4f_2 + 2f_3 + 4f_4 + 2f_5 + \cdots + 2f_{n-1} + 4f_n + f_{n+1}\right] \tag{4.14}$$

A partir de la expresión 4.14 se aprecia que el área bajo la curva de $f(x)$ se puede calcular conociendo el valor de $f(x)$ evaluada en cada punto, en otras palabras, se requiere sustituir los valores de las grabaciones en la expresión considerando que el espaciado ya es uniforme en todas las mediciones recabadas.

Para tener resultados lo más cercanos a la realidad se decidió realizar mediciones de corriente de los nodos terminales en algunas de las ubicaciones donde PUMAGUA tiene instalados medidores de flujo de agua y con periodos de transmisión de 30 minutos. En concreto, para estos cálculos y análisis de consumo se optó por estudiar dos casos de ubicaciones: el primero donde exista poca o nula presencia de obstáculos como edificios, muros y la distancia sea menor a cincuenta o cien metros; el segundo caso donde los obstáculos y la distancia sean considerables como para afectar los parámetros de transmisión como el SF, BW y potencia de transmisión. Las ubicaciones, a su vez, forman parte de las pruebas de campo del capítulo 5, donde se analizarán otros aspectos relacionados con la transmisión de los paquetes.

El programa creado para obtener el área bajo la curva y, por lo tanto, la energía consumida por ciclo de trabajo comienza con dividir la señal de los archivos en intervalos de transmisión, es decir, identifica los periodos de transmisión para analizarlos individualmente. Como ejemplo, se tiene la gráfica de la figura 4.27, que ilustra el segundo caso, donde se puede decir que la calidad de la transmisión es mala debido a los obstáculos y las distancias; visualmente se pueden identificar 5 transmisiones, incluidas las solicitudes de *Join request*, en consecuencia, el programa obtendrá automáticamente los 5 intervalos correspondientes a cada ciclo de trabajo.

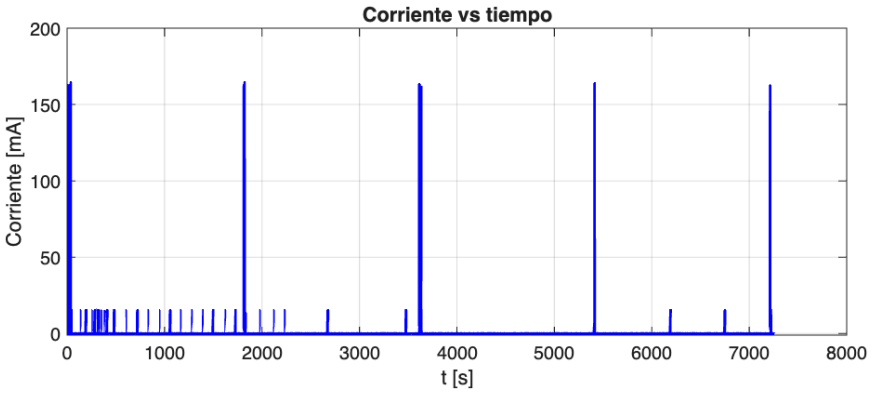


Figura 4.27: Análisis gráfico de ciclos de trabajo del nodo terminal.

Con los intervalos establecidos, el programa se encarga de realizar los cálculos de área bajo la curva para cada intervalo con los métodos mencionados, así como para la curva completa; además, este realiza un conteo del número de transmisiones por ciclo de trabajo junto con sus respectivas ventanas de recepción. El programa completo se presenta en el anexo C y el resultado se puede observar, en primera instancia, en la gráfica actualizada de la (figura 4.28), donde la información mostrada es:

- Las líneas punteadas verticales representan los intervalos de tiempo o ciclos de trabajo en los que fue dividida la grabación.
- La esquina superior izquierda muestra el consumo de carga asociado a cada intervalo de tiempo dividido.
- La columna de datos central presenta el número de transmisiones registradas por cada ciclo de trabajo. Para el primer intervalo se incluye el *Join request*.
- La esquina superior derecha muestra el valor de carga consumido por toda la gráfica con ambos métodos, además, obtiene el promedio de todos los ciclos de trabajo.

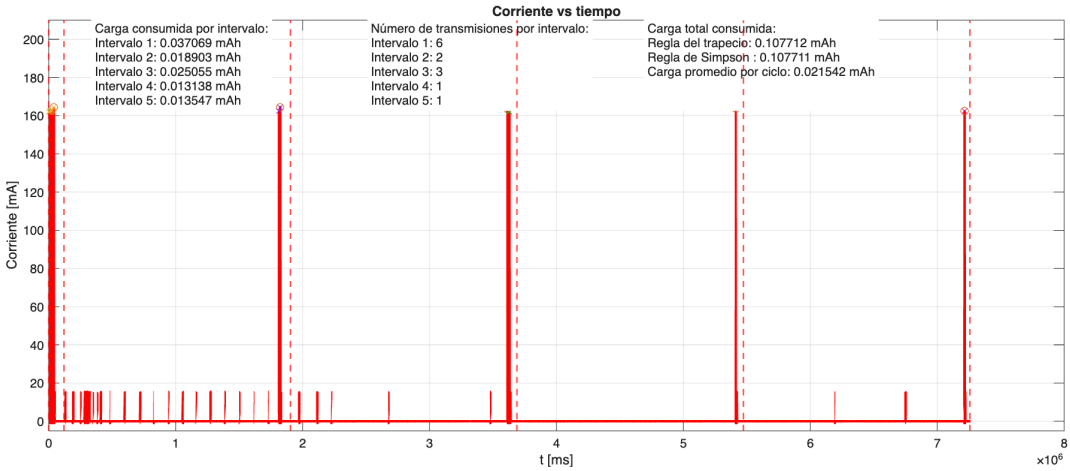


Figura 4.28: Gráfica actualizada del nodo con mala calidad de transmisión.

A diferencia de las gráficas anteriores, las gráficas creadas por este programa se manejan en milisegundos y no en segundos, esta decisión surge de la necesidad de mantener una alta resolución en el tiempo, ya que los algoritmos encargados de detectar los picos de transmisión y recepción, así como de delimitar los intervalos de análisis, requieren trabajar con precisión en escalas de tiempo reducidas, por ello es que se mantiene el uso de milisegundos en todo el procesamiento.

Después de observar múltiples resultados para el caso con mala calidad de recepción, existe un gran contraste respecto al primer caso donde se puede denominar que la calidad de transmisión es buena ya que la distancia entre el nodo y el Gateway es corta y casi no existen obstáculos (figura 4.29), pues el consumo de corriente cambia significativamente con el número de transmisiones y la cantidad de pulsos provenientes del medidor de flujo de agua. Por lo anterior, se decidió complementar el programa para que almacene en un nuevo archivo el número de transmisiones, ventanas de recepción y consumo asociado a cada intervalo, con esto se pretende obtener datos estadísticos que permitan realizar aproximaciones de consumo más exactas, de acuerdo a las características de la ubicación que presenta cada nodo.

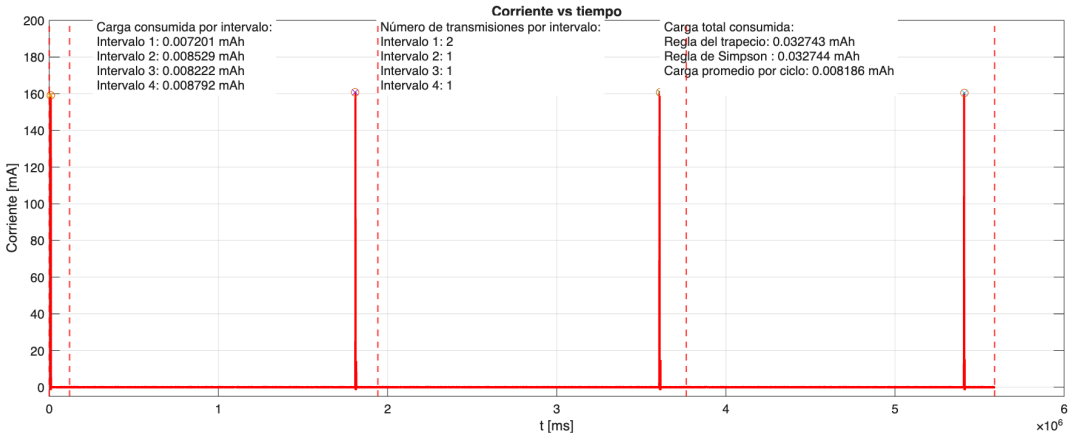


Figura 4.29: Gráfica actualizada del nodo con buena calidad de transmisión.

Todos los datos asociados a cada intervalo de transmisión o ciclo de trabajo fueron almacenados con el programa anterior en un archivo de extensión .csv, de tal modo que al leer dichos datos se pueden construir histogramas para cada uno de los datos, es decir, se puede determinar estadísticamente la frecuencia de cada número de transmisiones y ventanas de recepción asociadas al proceso de *Join request* y transmisión de paquetes.

El programa encargado de obtener los datos estadísticos asociados al ciclo de trabajo de los nodos se presenta en el anexo C, este obtiene los histogramas del número de picos de transmisión para el *Join request*, con sus respectivas ventanas de recepción, y el histograma de consumo de carga, también hace lo mismo para todos los ciclos de trabajo restantes, que corresponden a las transmisiones de paquetes. Lo anterior se presenta en la figura 4.30

Para el nodo con mala calidad de transmisión se recaudaron y procesaron un total de 73 ciclos de trabajo de transmisiones y recepciones regulares, en otras palabras, los histogramas de consumo de carga más frecuente para las transmisiones, recepciones, numero de retransmisiones y número de ventanas de recepción fueron construidos con 73 muestras de ciclos de trabajo. Para los histogramas de consumo, transmisiones y recepciones asociados al proceso de **Join request** se recaudaron y procesaron 9 eventos.

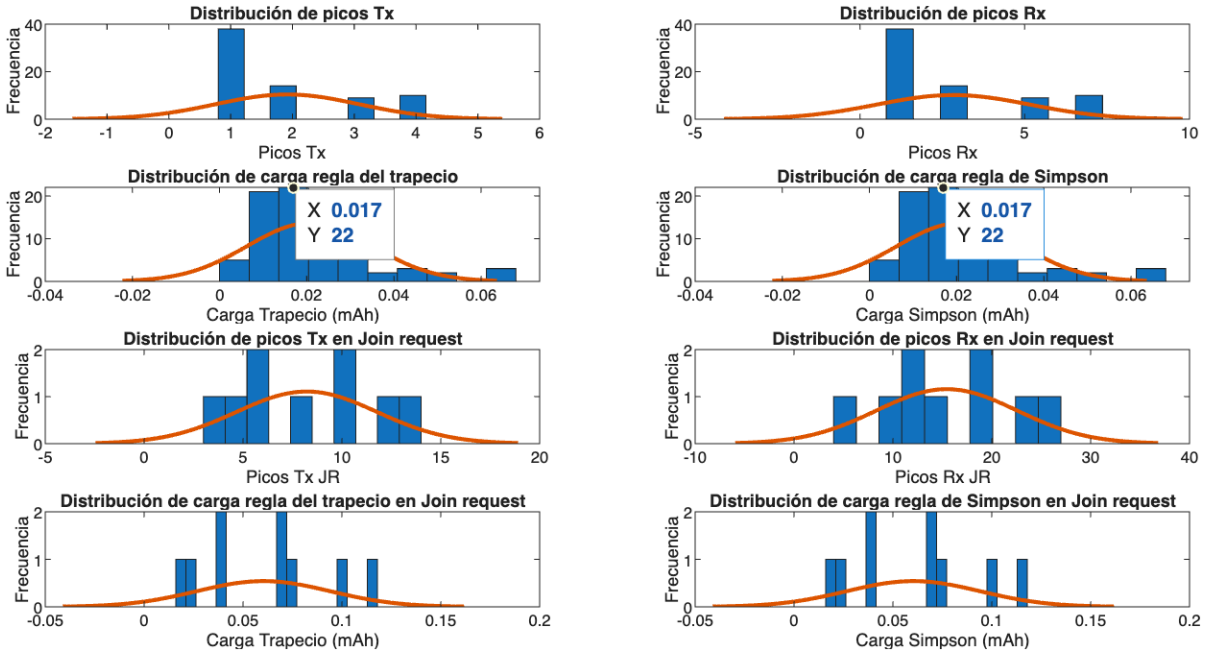


Figura 4.30: Histogramas de eventos y consumos para el nodo con mala calidad de transmisión.

De manera análoga, para el nodo con buena calidad de transmisión, los histogramas se presentan en la figura 4.31. En el caso de este nodo, los histogramas de consumo de carga, número de transmisiones y ventanas de recepción se construyeron al procesar un total de 56 ciclos de trabajo regular, mientras que los histogramas asociados al *Join request* se construyeron con 4 muestras.

En ambos nodos terminales, no se optó por recaudar más muestras para la construcción de los histogramas asociados al *Join request* ya que para la estimación de la vida útil de las baterías estos eventos no son relevantes.

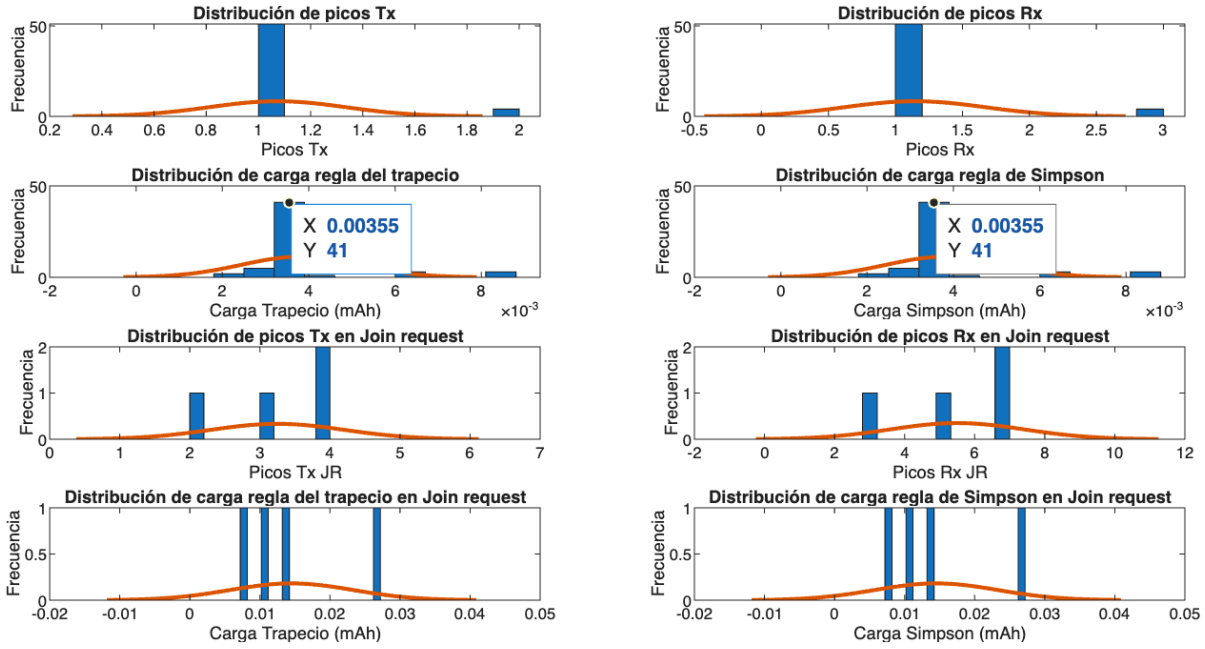


Figura 4.31: Histogramas de eventos y consumos para el nodo con buena calidad de transmisión.

La comparación anterior entre las medidas estadísticas mostradas confirman el hecho de que el consumo por ciclo de trabajo depende de las condiciones de transmisión particulares de la ubicación de cada nodo, siendo que el reto en la estimación de la vida útil de las baterías se presenta en aquellos nodos donde la calidad de la transmisión es mala, principalmente provocado por el constante cambio en el número de retransmisiones y parámetros como la potencia de transmisión y el spread factor. Por lo anterior, es de gran apoyo contar con la estadística generada, pues esta permite asociar un valor de consumo fijo al ciclo de trabajo del nodo.

Finalmente, al igual que con el cálculo teórico, se obtiene el número posible de eventos totales y días de transmisiones, considerando la media de consumo en un ciclo de trabajo para el nodo con buena calidad de transmisión (figura 4.31), con un intervalo de transmisión de 30 minutos.

Suponiendo la misma batería del caso teórico, con capacidad de $1000[mAh]$ y considerando $0.00355[mAh]$ como el valor de carga eléctrica mas frecuente consumida por el nodo, leído del histograma de la figura 4.31, se tiene que:

$$Eventos\ totales = \frac{1000[mAh]}{0.00355[mAh]} \approx 281,690$$

Con lo que la duración de la batería en días y años, contemplando los 48 eventos por día, es:

$$Días\ de\ transmisiones = \frac{Eventos\ totales}{Eventos\ por\ día} = \frac{281,690}{48} \approx 5,868 \text{ días} \approx 16 \text{ años}$$

O bien, considerando el peor de los casos, donde el nodo tiene mala calidad en la transmisión, con un consumo por ciclo de trabajo de $0.017[mAh]$ leído del histograma de la figura 4.30, se tiene el siguiente resultado:

$$Eventos\ totales = \frac{1000[mAh]}{0.017[mAh]} \approx 58,823$$

Con lo que la duración de la batería en días y años, contemplando los 48 eventos por día, es:

$$Días\ de\ transmisiones = \frac{Eventos\ totales}{Eventos\ por\ día} = \frac{58,823}{48} \approx 1,225 \text{ días} \approx 3.35 \text{ años}$$

El tiempo de vida estimado con esta metodología y cálculos, comparados con los obtenidos en trabajos como [41] y [42], donde también estima, con datos experimentales, la vida útil de dispositivos LoRaWAN alimentados por baterías que transmiten a periodos de tiempo mayores a 5 o 10 minutos; resulta coherente, pues el rango estimado de entre 3.35 y 16 años según las condiciones de transmisión son valores alcanzados y posibles de acuerdo con los trabajos mencionados.

Tanto el resultado obtenido experimental y teóricamente concuerdan y cumplen con el objetivo de la tesis, sin embargo, en ambos casos se despreció el factor de autodescarga de las baterías, el cual está presente en todo tipo de baterías y que, seguramente, afecta gravemente y es el principal limitante en el tiempo de vida del nodo terminal.

4.4.4 Comparativa de cálculos

La finalidad de esta comparativa entre cálculos es observar de manera más detallada y clara la diferencia del tiempo de vida útil de la batería y como cambia con respecto a los parámetros de transmisión y condiciones que se pueden presentar para cada nodo terminal. Para esto, se propuso una primera comparativa entre el caso teórico y el caso experimental y, posteriormente, entre los dos casos experimentales documentados.

Cálculo teórico y cálculo experimental

Resulta interesante apreciar la diferencia entre lo que puede consumir un nodo terminal considerando únicamente la teoría de funcionamiento de LoRa y LoRaWAN y lo que consume el protocolo implementado en un dispositivo real. Los principales aspectos inesperados en el consumo del nodo terminal, apreciados a lo largo del capítulo, son los siguientes:

1. Existen los pequeños pulsos de preparación para transmisión y recepción de paquetes.
2. El delay entre el envío de datos (uplink) y ventanas de recepción es mayor a lo especificado.
3. Cuando la calidad de la transmisión/recepción es buena, sólo existe una ventana de recepción.

Considerando el número de días de transmisiones posibles obtenido en el caso teórico $(5, 152)$ como valor de referencia y el obtenido en el caso experimental para el nodo con buena calidad en la transmisión y recepción $(5, 868)$, el porcentaje de error entre ambos resultados es:

$$\text{Error porcentual} = \left|\frac{5{,}868-5{,}152}{5{,}152}\right| \times 100 \approx 13.9\,\%$$

Lo anterior sugiere que el cálculo experimental tendría una duración de batería mayor, en aproximadamente $13.9\,\%$, lo cual puede atribuirse a dos posibles causas:

- En el caso experimental, cuando el nodo transmitió registro corrientes cercanas a 160[mA], mientras que en la tabla 4.7, para la potencia de 20 [dBm] la corriente es de 165.13[mA], por lo que el nodo estaría transmitiendo a una potencia menor a la considerada en el calculo teórico.
- Solamente se presenta una ventana de recepción en las mediciones experimentales, mientras que en el calculo teórico se considero en todo momento la doble ventana de recepción.

A pesar de la diferencia, un tanto inesperada, este resultado es favorable ya que, aunque el consumo de modo sleep es mayor y se presentan los pequeños pulsos de preparación de transmisión y recepción de paquetes, el experimento demuestra que es posible lograr una mayor duración de la batería en condiciones reales con el uso del ADR de LoRaWAN. Además, este resultado valida que, hasta cierto punto, el modelo teórico es confiable como referencia para el diseño y estimación de la vida útil de los nodos terminales en distintos escenarios.

Cálculos experimentales

Esta comparativa es la más significativa, considerando que para tener la mejor exactitud en la estimación de la duración de las baterías se requiere tomar en cuenta todos los factores que intervienen en una transmisión. En este caso, las principales diferencias entre un nodo con buena calidad en la transmisión y recepción y uno con mala calidad, como ya se ha establecido, son:

1. Cuando la calidad de la transmisión/recepción es buena, sólo existe una ventana de recepción.

2. Se presentan las retransmisiones de manera constante cuando la calidad de la transmisión/-recepción es mala.

De manera análoga a lo realizado en la comparativa anterior, el porcentaje de diferencia entre la duración de la batería para un nodo y otro es de:

$$\text{Error porcentual} = \left|\frac{1{,}225-5{,}868}{5{,}868}\right| \times 100 \approx 79.12\,\%$$

El resultado anterior es crítico debido a la magnitud del impacto en la batería que un nodo en condiciones desfavorables puede presentar, esto puede representar un gran problema de mantenimiento respecto a nodos con mejores condiciones, o simplemente que no es viable la instalación de nodos en puntos retirados del Gateway o que presenten muchos obstáculos como edificios o árboles, por lo que, para el diseño de una red a gran escala se deben considerar diferentes estrategias que optimicen el consumo de la batería.

El resumen de las breves comparativas se presenta en la tabla 4.9

Cálculo	**Duración estimada [días]**	**Consumo por ciclo [mAh]**	**Error respecto al teórico [%]**
Teórico	5,212	0.004043	–
Experimental (buenas condiciones)	5,868	0.00355	13.9
Experimental (malas condiciones)	1,225	0.017	79.12

Tabla 4.9: Comparación entre duración de batería y consumo por ciclo de trabajo

Capítulo 5

Pruebas experimentales del sistema

5.1 Introducción

El objetivo de las pruebas realizadas al nodo terminal y al sistema completo desarrollado (compuesto por el Gateway, la aplicación web y los nodos terminales), es conocer su desempeño en escenarios donde las condiciones en las que se encuentran los nodos terminales varían y no son ideales, al conocer el desempeño del sistema, y particularmente de los nodos terminales, es posible determinar si cumple los objetivos propuestos y, por lo tanto, si es apto para su implementación definitiva en reemplazo de los dispositivos con los que PUMAGUA monitorea el flujo de agua de su infraestructura.

La metodología llevada a cabo para las pruebas consiste en dos partes principales: las pruebas en laboratorio y las pruebas en campo. Las pruebas en laboratorio tienen el objetivo específico de validar que los nodos terminales registran, se comunican y transmiten datos correctamente al Gateway central en un entorno ideal y controlado, es decir, no hay grandes obstáculos ni largas distancias entre los nodos terminales y el Gateway; validar esta correcta comunicación entre nodos y Gateway da paso a las pruebas en campo, donde las condiciones como la humedad del ambiente, los obstáculos, distancias e interferencias, ya tienen un impacto negativo en la transmisión de datos. Las pruebas en campo son las más significativas puesto que los nodos terminales se instalarán en ubicaciones de PUMAGUA donde se presentan este tipo de anomalías o dificultades en el análisis del desempeño del sistema.

Tanto para las pruebas de laboratorio, como las pruebas de campo, se empleó un Gateway central instalado en la azotea del edifico Q de la Facultad de Ingeniería, mismo edificio donde se encuentra el laboratorio, lo que facilita estas pruebas; además, la ubicación de este edificio es clave puesto que a sus alrededores existe la infraestructura de PUMAGUA, como las tuberías y los medidores de flujo de agua necesarios para cubrir con las condiciones de estudio establecidas para las pruebas de campo. En el anexo D se adjuntan imágenes que permiten observar las características del entorno en las que el Gateway central se encontró instalado.

5.2 Experimentos en laboratorio

5.2.1 Infraestructura y definición de los experimentos

La infraestructura necesaria para llevar a cabo estas pruebas es simple, consta de uno o más nodos terminales conectados al Gateway central, los nodos terminales se energizarán ya sea por medio de baterías externas o por el puerto USB del CubeCell. Se emplea una aspiradora cuya función es proporcionar un flujo de aire al medidor a manera de simular un flujo de agua. La decisión de usar flujo de aire en lugar de agua para simular el funcionamiento del medidor de flujo de agua fue debido a su facilidad de empleo en laboratorio.

La definición de las características de los experimentos que se realizan se presenta a continuación:

1. El primer experimento tiene como objetivo verificar que el nodo terminal esté registrando correctamente todos los pulsos del medidor de flujo de agua, esto es esencial para determinar la confiabilidad de las mediciones.
2. Para el segundo experimento, se planea el envío de datos de un nodo terminal al Gateway central cuando el nodo se encuentre relativamente cerca y sin obstáculos, la finalidad es comprobar que los paquetes de datos llegan sin errores al Gateway.
3. En el tercer experimento de laboratorio, se enviarán datos de dos o más nodos terminales al Gateway central, considerando un entorno con obstáculos y/o interferencia.
4. Como último experimento de laboratorio, se propone dejar un nodo terminal enviando datos cada 15 segundos, el objetivo es observar el impacto de consumo de energía en las baterías cuando la frecuencia de transmisión de datos es grande comparada con la requerida por PUMAGUA.

5.2.2 Experimento 1

Según el objetivo de este experimento, será necesario determinar que el registro de pulsos es correcto, con ayuda del osciloscopio se verifica la existencia de los pulsos por parte del medidor de flujo de agua. Este punto fue realizado y explicado más detalladamente en la sección 3.4.2 del capítulo 3, específicamente la figura 3.11 muestra la forma y duración del pulso, por lo que el procedimiento para verificar el correcto registro del pulso por parte del microcontrolador del nodo terminal fue el siguiente:

1. Con base en la programación del conteo de pulsos para el medidor de flujo de agua, se cargó

en el nodo el programa ejemplo de interrupción con la rutina que incrementa un contador cada vez que se detecta un pulso en el GPIO seleccionado (código 3.2).

2. Se verifica en el monitor serial del IDE de Arduino que la variable del contador imprima el valor correcto de acuerdo con los pulsos vistos con el osciloscopio cuando se le proporciona un flujo de aire constante al medidor de flujo de agua.
   En la figura 5.1 se encuentran las impresiones con el valor del contador en el monitor serial de Arduino. Gracias a la visualización de los pulsos en el osciloscopio es que se pudo demostrar el correcto funcionamiento de la configuración de la rutina de interrupción y, por lo tanto, el correcto registro de los pulsos del medidor de flujo de agua.

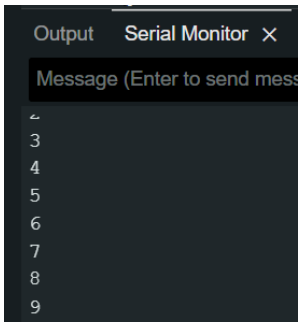


Figura 5.1: Registro de pulsos en el microcontrolador del nodo terminal.

5.2.3 Experimento 2

La comunicación entre un nodo terminal y el Gateway central se basa en la programación del nodo y el registro del mismo en la plataforma TTN, según lo mostrado en la sección 3.4.4, el nodo terminal está programado para enviar el valor del contador y el voltaje de alimentación hasta que se cumple el tiempo establecido para el envío del paquete al Gateway (sección 3.4.3), teniendo en cuenta lo anterior, en la primera prueba realizada se estableció un tiempo de envió entre paquetes de 5 segundos, donde además, sólo se envían datos de voltaje de batería, puesto que no se conectó el medidor de flujo de agua.

La comprobación de recepción de paquetes se hace desde la plataforma TTN, en el apartado de *End devices* (mostrado en la figura 3.12). Dentro de las aplicaciones creadas en la plataforma TTN se enlistan todos los nodos terminales registrados, para acceder a la información asociada a cada uno de ellos basta con hacer clic en su nombre y se abrirá una ventana como la de la figura 5.2, en el apartado de *Live data* se pueden observar todos los datos recibidos por el Gateway, similar a lo mostrado en el análisis del ciclo de trabajo del nodo terminal del capítulo 4.

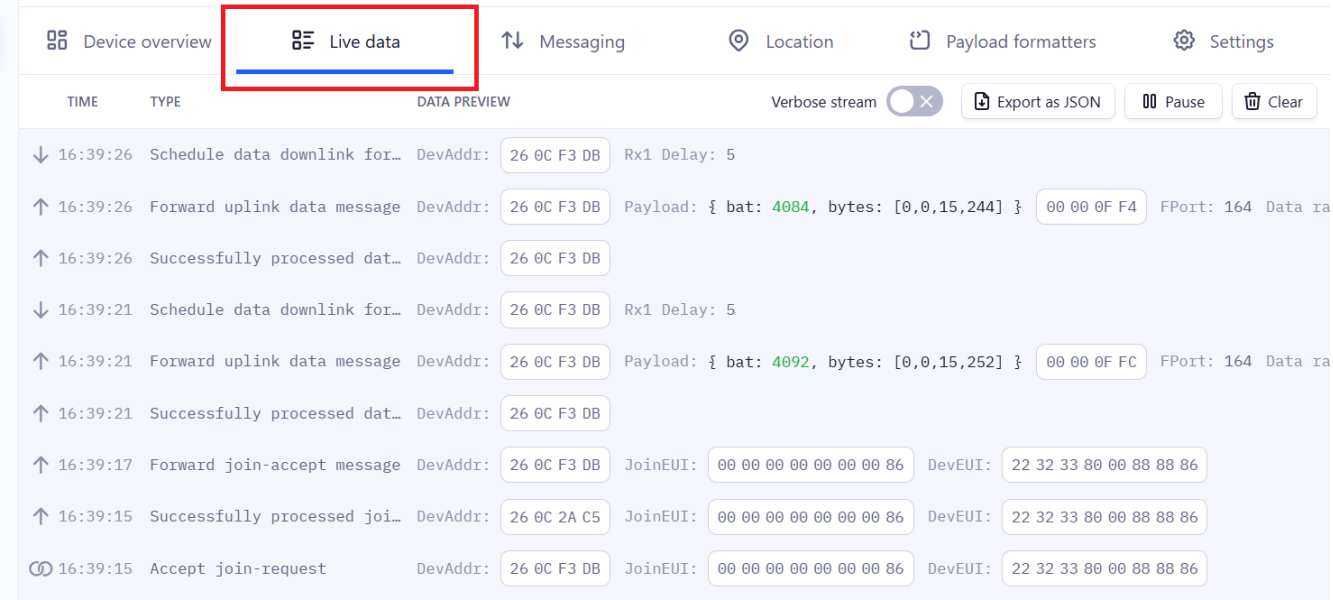


Figura 5.2: Mensajes del nodo terminal sin el medidor de flujo de agua.

Dentro de los mensajes desplegados se tienen las solicitudes de *join request*, las confirmaciones de aceptación o *join acept*, los mensajes con el payload y otros mensajes de confirmación y de recibido por parte del Gateway. De acuerdo con la programación de la sección 3.4.3, se definió que el voltaje de la batería y el contador se envía en dos partes de 8 bits, por lo que, para interpretar el payload es necesario unir los dos primeros dos bytes para el contador y los últimos dos bytes para el voltaje de la batería; de este modo, al hacer la operación el valor hexadecimal *FF4* convertido a decimal es *4084*, en otras palabras, $4084[mV]$.

Para la segunda parte de este experimento se estableció un tiempo entre envío de paquetes de 20 segundos con el objetivo de darle tiempo al medidor de flujo de agua de emitir un par de pulsos y que el CubeCell los registre y envíe para verse reflejados en la plataforma. En la figura 5.3 se encuentra la ventana con los mensajes recibidos por el Gateway, en estos se observan datos tanto de voltaje, como el número de pulsos registrados por el nodo terminal.

Figura 5.3: Mensajes del nodo terminal con el medidor de flujo de agua.

Con base en los resultados de este experimento se comprobó que el envío de datos por parte del nodo terminal al Gateway central funciona correctamente en condiciones *ideales*. Con lo anterior, es posible seguir probando el nodo y dar paso al experimento 3.

5.2.4 Experimento 3

Siguiendo con la definición y los objetivos de los experimentos, en este se abordarán los casos donde dos o más nodos envían datos al Gateway central en condiciones que simulan un entorno más realista al que se enfrentarán los nodos terminales, en este caso los obstáculos y la interferencia son creados por los muros y pisos del edificio donde se encuentra el laboratorio. Dado que el Gateway se encuentra en la azotea del edificio, los nodos terminales estarán en movimiento constante por los diferentes pasillos y pisos del edificio, simulando un entorno cambiante.

En la figura 5.4 están listados los dos nodos registrados. En el costado derecho está el tiempo transcurrido desde la última actividad, lo cual es importante para verificar que no existan pérdidas de paquetes. A partir de lo descrito, se decidió fijar el tiempo de transmisión en 20 segundos.

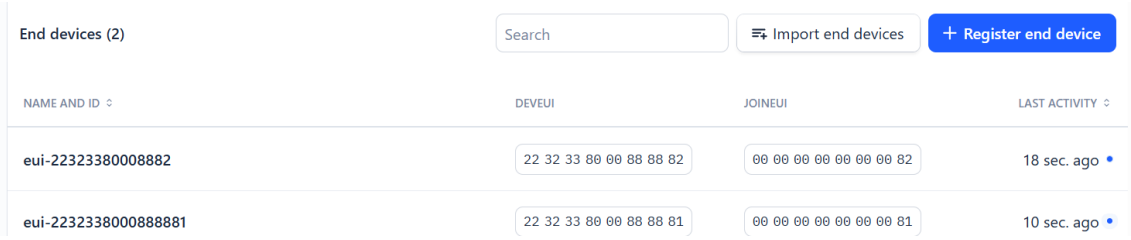
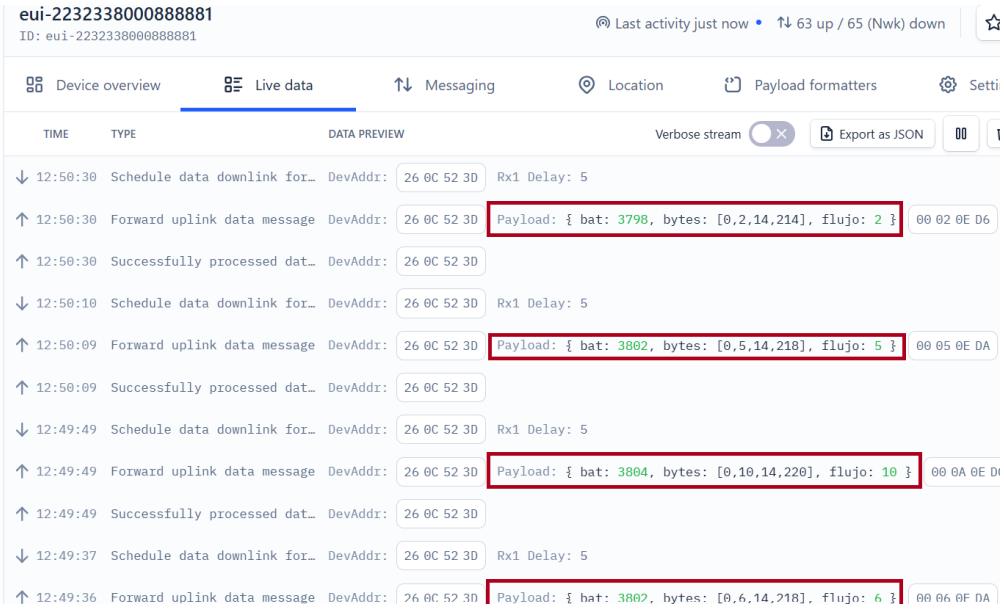


Figura 5.4: Nodos terminales registrados.

El correcto funcionamiento de los nodos se puede observar en el payload recibido por cada

uno de ellos, La figura 5.5 muestra los mensajes recibidos por ambos nodos, en uno de ellos no se conecta el medidor de flujo de agua, mientras que en el otro sí, por ello, en la figura 5.5a el valor de flujo es 0 y en la figura 5.5b el valor de flujo es diferente de 0; también se aprecia que el voltaje por el que se encuentran alimentados es diferente.

(a) Nodo terminal con clave Eui:2232338000888882.



(b) Nodo terminal con clave Eui:2232338000888881.

Figura 5.5: Mensajes recibidos por dos nodos terminales.

5.2.5 Experimento 4

Para cumplir de la mejor manera con el objetivo de esta prueba, se empleó el circuito construido en la figura 3.6a, alimentado por una batería de litio recargable con capacidad de 2000 $[mAh]$, en las mismas condiciones con las que se realiza el experimento uno de las pruebas de campo (figura 5.9). Una vez registrado en la plataforma TTN se realizó un monitoreo periódico en el que se registraron los valores de voltaje de la batería 5 veces por día, durante tres semanas, dando como resultado la gráfica de la figura 5.6.

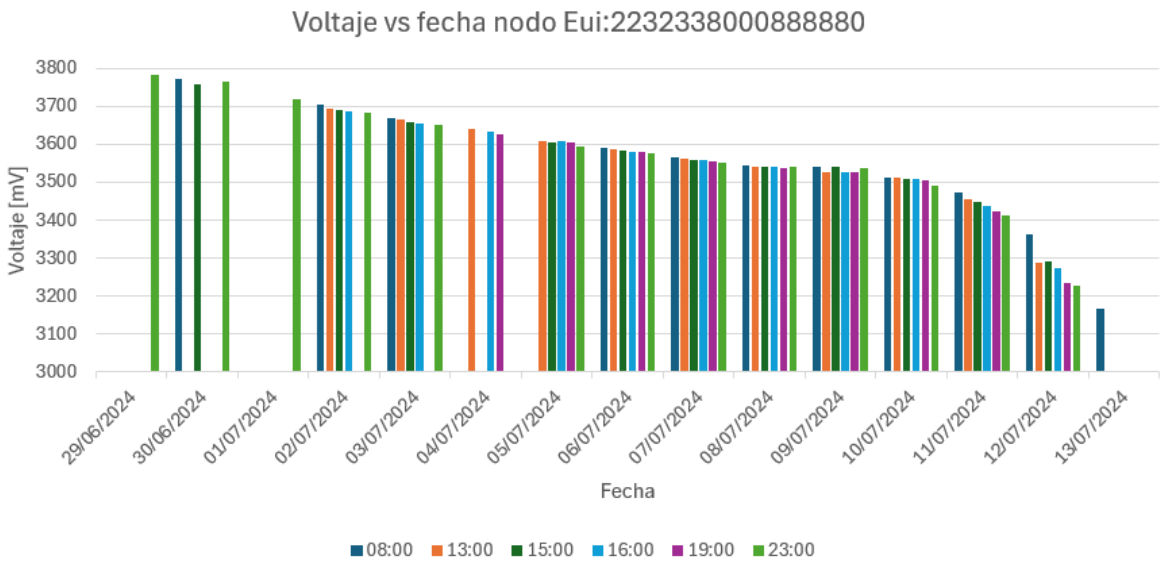


Figura 5.6: Voltaje de la batería registrado en el nodo terminal.

La gráfica resultante muestra el claro comportamiento de descarga de una batería con respecto al voltaje que entrega. Se observa una tendencia decreciente en el voltaje de la batería a lo largo de las muestras tomadas, esto indica que el consumo energético del nodo terminal es constante y, además, los valores registrados muestran variaciones mínimas, lo que concuerda con el bajo valor de consumo de corriente que se le demanda. La principal observación ocurre a partir del 10 de julio de 2024, cuando se ha agotado aproximadamente el 75 % de la capacidad de la batería, después de esta fecha el voltaje comienza a disminuir más rápidamente, indicando que la batería se acerca a su nivel de descarga crítica, esto evidencía la importancia de un monitoreo continuo para evaluar el rendimiento energético del nodo y determinar el momento adecuado para reemplazar o recargar la batería.

Como complemento del análisis de la gráfica, durante el registro de datos se pudo apreciar que el nodo terminal registró pulsos del medidor de flujo de agua a partir de que el voltaje fue menor a

3550[mV], como si tuviera conectado un medidor de flujo de agua que emitía pulsos constantes. no obstante, este número de pulsos se puede asociar al mal funcionamiento del nodo terminal debido al bajo nivel de voltaje; por lo que se asume que estos pulsos registrados son consecuencia de dicha causa, ya que los pulsos registrados, en su mayoría, siempre fueron de dos y uno, según se observa en la tabla de la figura 5.7.

Dia		8:00		13:00		15:00		16:00		19:00		23:00
29/06/24											0	3784
30/06/24	0	3774			0	3760					0	3765
01/07/24											0	3718
02/07/24	0	3704	0	3695	0	3692	0	3988			0	3682
03/07/24	0	3668	0	3664	0	3660	0	3656			0	3650
04/07/24			0	3640			0	3632	0	3626		
05/07/24			0	3610	0	3606	0	3608	0	3604	0	3596
06/07/24	0	3590	0	3586	0	3582	0	3580	0	3580	0	3576
07/07/24	0	3566	0	3562	0	3558	0	3558	0	3556	0	3550
08/07/24	0	3544	2	3542	0	3542	0	3540	2	3538	2	3542
09/07/24	2	3540	2	3528	0	3540	2	3528	1	3526	2	3536
10/07/24	2	3514	2	3514	2	3508	2	3510	2	3504	2	3491
11/07/24	2	3473	2	3454	2	3447	2	3436	2	3424	2	3412
12/07/24	2	3363	2	3288	2	3292	2	3272	2	3234	2	3227
13/07/24	2	3167	---------									

Figura 5.7: Registro de datos del nodo terminal con Eui:2232338000888880.

5.3 Experimentos en campo

5.3.1 Infraestructura y definición de los experimentos

La definición de los experimentos de campo surge con base en los experimentos de laboratorio. Una vez comprobada la correcta comunicación entre los nodos terminales y el Gateway central, es posible definir los experimentos de campo. En estos, como ya se mencionó, se probó el desempeño de los nodos terminales en diversos escenarios que desafían la comunicación entre ellos y el Gateway central.

El principal objetivo es conocer los límites o circunstancias en las que los paquetes de datos son recibidos correctamente por el Gateway central, y en qué circunstancias dichos paquetes ya no son recibidos. De acuerdo con lo anterior, se definen los siguientes experimentos:

1. Como primer experimento, se pretende determinar la distancia máxima con linea de vista en la cual el Gateway central recibe datos del nodo terminal, el objetivo es determinar si el alcance del CubeCell es suficiente para cubrir las grandes áreas a las que se enfrentará en Ciudad Universitaria.
2. El segundo experimento consiste en instalar el prototipo del nodo terminal en uno de los registros con los que cuenta PUMAGUA, allí se probó el desempeño y resistencia del prototipo en una ubicación real, donde existe gran humedad en el ambiente y atenuación por obstáculos

y la misma tapa del registro.

3. Para el tercer experimento, se colocaron dos o más nodos terminales ubicados en diferentes registros de las tuberías de PUMAGUA, con esto se pretende recabar datos suficientes para hacer un análisis detallado del consumo energético de cada nodo según las características de su ubicación, además de probar la resistencia a la humedad de un nuevo prototipo de encapsulado para el circuito electrónico.

5.3.2 Experimento 1

Para este primer experimento se tiene como objetivo validar y cuantificar el alcance que puede tener un CubeCell a través de la comunicación LoRa y el protocolo LoRaWAN, para ello, se requiere llevar un nodo terminal a una distancia lo suficientemente grande que permita validar que el alcance de la comunicación es suficiente para alcanzar todos los puntos requeridos de Ciudad Universitaria por PUMAGUA.

Gracias a la ayuda de los integrantes del laboratorio de redes inalámbricas es que fue posible ubicar el nodo terminal en el edificio High Park Gran Sur, cuya distancia al edificio Q de la Facultad de Ingeniería, donde se encuentra el Gateway central, es de aproximadamente 3.087$[km]$, de acuerdo con el mapa mostrado en la figura 5.8.

Figura 5.8: Distancia de línea de vista entre el Gateway central y el nodo terminal.

Durante este experimento se monitoreó constantemente la recepción de los datos enviados por el nodo terminal durante su trayecto. Durante la mayoría del trayecto se perdió la comunicación entre el Gateway y el nodo terminal, desde que este salió de Ciudad Universitaria y se reestableció hasta que se encontró en la azotea del edificio High Park (figura 5.9), la pérdida de comunicación se le atribuye a la cantidad de edificios y árboles presentes en el trayecto.

Figura 5.9: Condiciones del entorno del nodo terminal en la azotea del edificio High Park.

El resultado obtenido fue parcialmente exitoso puesto que el Gateway central recibió datos del nodo terminal a una distancia de 3.087$[km]$ con linea de vista, lo que es suficiente para abarcar toda Ciudad Universitaria. Por otro lado, el experimento mostró las deficiencias en la transmisión de datos cuando la distancia y los obstáculos aumentan o están cambiando constantemente, por lo tanto, las futuras pruebas deberán concentrarse en encontrar la distancia y condiciones en las que los nodos terminales son capaces de enviar datos exitosamente hasta el Gateway central.

5.3.3 Experimento 2

Como primer objetivo de este experimento, se tiene el monitoreo de flujo/consumo de agua en una de las tuberías de PUMAGUA durante un periodo de tiempo que demuestre el correcto funcionamiento del sistema y que, además, muestre información sobre el comportamiento del consumo de agua en esa tubería durante el periodo de observación. Dado que el nodo terminal se encontrará inmerso en un registro con un alto grado de humedad, el segundo objetivo consiste en probar la resistencia y durabilidad a largo plazo del circuito del nodo terminal diseñado.

El monitoreo de los datos transmitidos por el nodo terminal se realizó en una hoja de cálculo, la razón principal de esto es que la plataforma TTN no permite almacenar los datos recibidos, sino que se eliminan después de cierto tiempo o cierta cantidad de mensajes. Aunque TTN dispone de planes de almacenamiento de datos, éstos implican un costo, lo que lo vuelve inaccesible a largo plazo.

Para este experimento se construyó el prototipo de nodo terminal que se observa en la figura 5.10. Este prototipo consiste en una caja de plástico y acrílico donde se encuentra el circuito construido en la figura 3.6a, dentro se encuentra la batería de litio recargable con capacidad de

2000$[mAh]$ (idéntica a la mostrada en la figura 5.9), que se encarga de alimentar el CubeCell, para agregar mayor protección al circuito contra la humedad del ambiente, esta caja fue sellada con silicón frío transparente para colocación de ventanas.

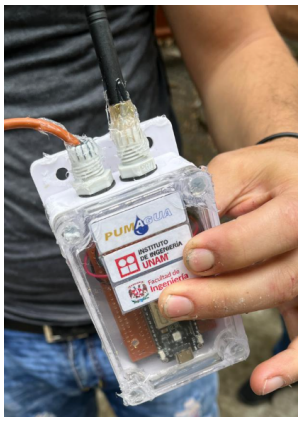
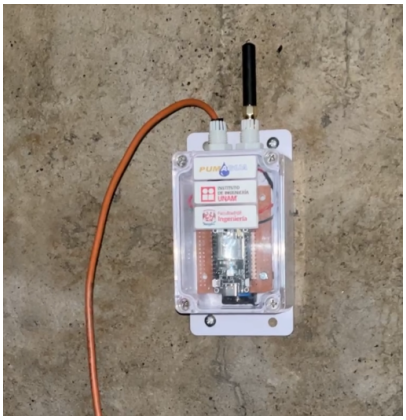


Figura 5.10: Prototipo de nodo terminal.

El nodo terminal fue programado para enviar el paquete cada 30 minutos, este fue ubicado en un registro a las afueras del edificio 5 del Instituto de Ingeniería. Se escogió esta ubicación debido a la cercanía (113 metros) con el Gateway central, además, en este registro de PUMAGUA existe el modelo adecuado de medidor de flujo de agua compatibles con la caratula acondicionada al prototipo del nodo terminal. La figura 5.11 presenta la instalación y conexión del nodo terminal al medidor de flujo de agua instalado en las tuberías de PUMAGUA.



(a) Nodo terminal.

(b) medidor de flujo de agua.

Figura 5.11: Instalación en registro de PUMAGUA del nodo terminal.

El periodo en el que se registraron los datos de pulsos del medidor y el voltaje de la batería, en un total de 6 veces por día, abarca del 29 de junio de 2024 al 25 de julio de 2024, lo que comprende

casi todo un mes, esto se decidió ya que se trataba de un periodo vacacional y es un periodo de tiempo considerable para cumplir con el primer objetivo del experimento. Lo anterior se muestra reflejado en la tabla de la figura 5.12 y se interpreta de la siguiente manera:

- Cada hora de registro tiene dos columnas.
- La primer columna indica la cantidad de pulsos registrados/contados por el nodo terminal.
- La segunda columna es el voltaje, en $[mV]$, que el nodo registra de la batería que lo alimenta.

A grandes rasgos, en la tabla se puede notar que hay una pequeña cantidad de pulsos registrados por el nodo según la hora en que se registraron, además, se puede notar como el voltaje de la batería disminuye conforme pasan los días. Con base en los datos, se puede concluir que la tubería no presenta una gran cantidad de flujo de agua, se asume que este poco flujo de agua es debido principalmente a que Ciudad Universitaria se encontraba en periodo vacacional, por lo que realmente no existió un consumo constante o normal.

Dia	8:00		13:00		15:00		16:00		19:00		23:00	
29/06/24											1	3908
30/06/24	0	3908			0	3908					0	3908
01/07/24											1	3908
02/07/24	3	3908			6	3906	0	3906			0	3096
03/07/24	0	3908	2	3906	6	3906	0	3906			0	3906
04/07/24			0				0	3906	0	3906		
05/07/24			0	3906	5	3906	0	3906	5	3906	0	3906
06/07/24	0	3906	0	3906	4	3906	0	3906	0	3904	0	3904
07/07/24	0	3904	0	3904	0	3904	0	3904	0	3904	0	3904
08/07/24	1	3904	0	3904	1	3904	0	3904	1	3904	0	3904
09/07/24	0	3904	3	3904	0	3904	0	3904	2	3904	0	3904
10/07/24	3	3904	6	3904	0	3904	3	3904	1	3904	1	3904
11/07/24	0	3904	4	3904	0	3904	0	3904	0	3904	0	3904
12/07/24	0	3902	0	3904	1	3902	0	3904	0	3902	0	3902
13/07/24	0	3902	0	3902	0	3902	0	3902	1	3902	1	3902
14/07/24	0	3902	0	3902	0	3902	0	3902	0	3902	0	3902
15/07/24	0	3902	6	3902	0	3902	0	3902	1	3902	0	3902
16/07/24	0	3900	0	3900	0	3900	0	3900	1	3900	0	3900
17/07/24	1	3900	0	3900	0	3900	0	3900	1	3900	0	3900
18/07/24	0	3900	1	3900	2	3900	1	3900	2	3900	0	3900
19/07/24	0	3900	1	3900	0	3900	0	3900	0	3900	0	3900
20/07/24	0	3900	3	3900	0	3900	0	3900	1	3900	0	3900
21/07/24	0	3900	0	3900	0	3900	2	3900	0	3898	0	3898
22/07/24	4	3900	12	3900	9	3900	2	3900	4	3898	1	3898
23/07/24	3	3898	1	3898	3	3898	0	3898	2	3898	3	3898
24/07/24	7	3898	9	3898	7	3898	0	3898	4	3898	1	3898
25/07/24	8	3898	14	3898	5	3898	4	3898	2	3898		3898

Figura 5.12: Registro de datos del nodo terminal con Eui:2232330000888889.

Adicionalmente, se construyó una gráfica de barras donde aprecia más claramente el comportamiento que tiene la descarga de la batería (figura 5.13) con el paso de los días, a simple vista la gráfica promete que la duración de la batería podría alcanzar al menos diez meses, ya que durante el periodo de registro el voltaje de la batería cayó unicamente $10[mV]$.

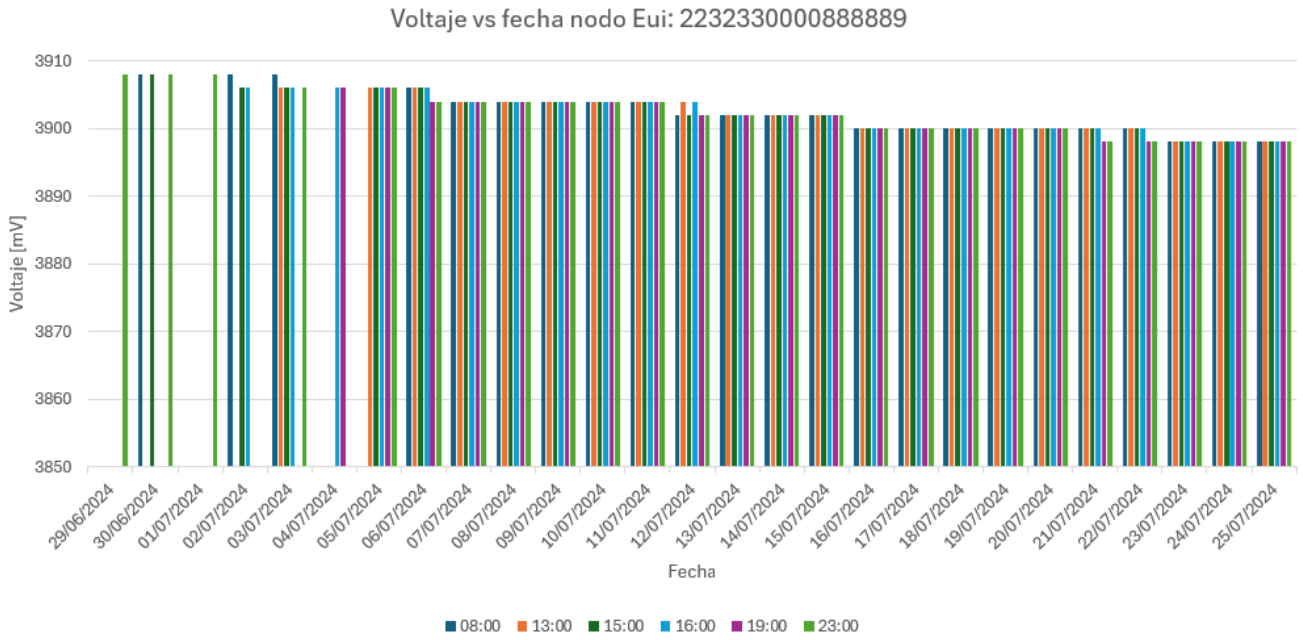


Figura 5.13: Voltaje de la batería del nodo terminal con Eui:2232330000888889.

Cumplir con el segundo objetivo de este experimento tomó un tiempo aproximado de seis meses, durante ese tiempo el nodo terminal se mantuvo enviando datos al Gateway central de manera exitosa, pasado ese periodo de tiempo se comenzó a notar un comportamiento similar al observado en el cuarto experimento de laboratorio y, finalmente, se observó que dejaban de llegar los datos. Con lo anterior se decidió retirar el nodo de su ubicación y observar el estado en que se encontraba, según se aprecia en la figura 5.14, el nodo terminal terminó siendo afectado en gran medida por la humedad a pesar de las medidas tomadas para evitarlo.

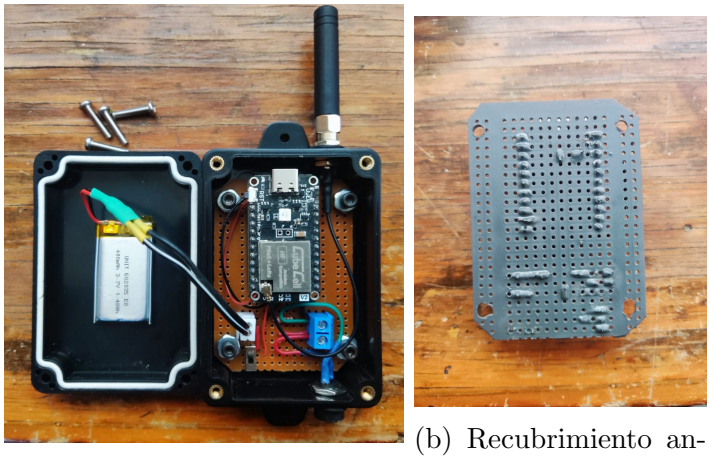
Figura 5.14: Deterioro del nodo terminal después de seis meses de operación.

5.3.4 Experimento 3

Como último experimento diseñado, este tiene el objetivo principal de monitorear y estudiar más a detalle el comportamiento del nodo a través de los parámetros de la señal LoRa como el Received Signal Strength Indicator (RSSI), el Signal to Noise Ratio (SNR), y spread factor, además del flujo de agua de las ubicaciones donde se instalarán. Asimismo, se pretende mejorar la resistencia y, por lo tanto, el desempeño del nodo, a partir de un nuevo prototipo que resista mejor las condiciones de humedad que deterioran el circuito, según lo visto en la figura 5.14.

Diseño del experimento

Con base en los objetivos planteados para este experimento, se creó un prototipo final más adecuado para las pruebas de funcionamiento bajo las condiciones de humedad vistas. El prototipo creado se presenta en la figura 5.15a, este encapsula el circuito electrónico en una caja de plástico ABS con un empaque de sellado en su contorno, se hicieron dos perforaciones por las que se extrae la antena y se conecta el medidor de flujo de agua; estas, a su vez, están selladas mediante empaques de goma para plomería. Por último, pero no menos importante, la parte de la placa fenólica donde se encuentran las conexiones con soldadura cuenta con un recubrimiento de pintura anticorrosiva (figura 5.15b).



(a) Prototipo nodo terminal. (b) Recubrimiento anticorrosivo.

Figura 5.15: Construcción del prototipo para el nodo terminal.

Para estas pruebas se cambiaron las baterías de litio anteriores por baterías de polímero de litio con una capacidad de 1000$[mAh]$, por lo que no se espera una larga duración del nodo terminal

en funcionamiento, La decisión se tomó por dos motivos: registrar el comportamiento de descarga de la batería en un periodo de tiempo reducido y tener consistencia con el análisis de consumo energético realizado en el capítulo cuatro (4), por otro lado, la reducción en el tamaño del prototipo limitó la instalación de una batería de mayor capacidad.

El objetivo de tener diferentes ubicaciones para los nodos terminales es tener condiciones de transmisión diferentes, es decir, dependiendo su ubicación cada nodo se configurará con diferentes parámetros de spread factor y potencia de transmisión, de igual forma presentarán doble ventana de recepción o múltiples retransmisiones. Para estudiar y observar este fenómeno se propuso instalar nodos terminales en ubicaciones dentro de los siguientes rangos de distancia:

- Distancias cercanas entre 0 y 200 metros.
- Distancias intermedias entre 200 y 600 metros.
- Distancias lejanas mayores a 600 metros.

Otro factor que se pretende considerar y estudiar es ubicar nodos en distancias similares, pero que presenten diferente densidad de obstáculos, de esta manera se podría estudiar el impacto que tienen los obstáculos más allá de la distancia.

Desarrollo del experimento

La elección de las ubicaciones, como es de suponerse, depende en gran medida de los registros y sensores disponibles por parte de PUMAGUA, después de informar sobre los requerimientos y negociar las ubicaciones, se logró establecer las siete ubicaciones presentadas en el mapa de la figura 5.16.

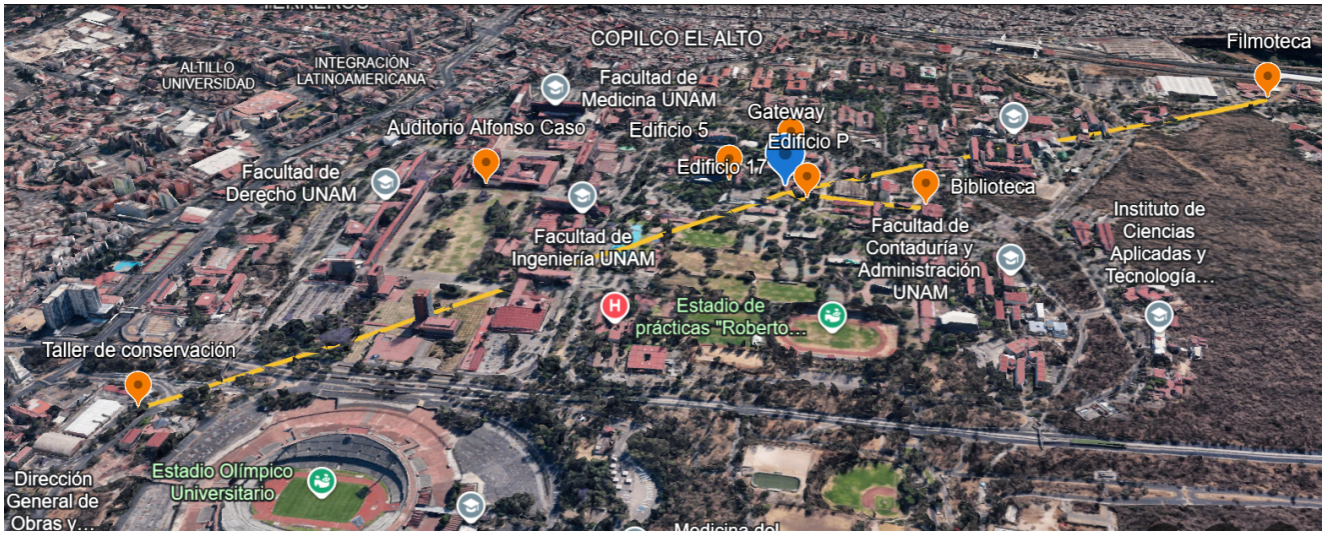


Figura 5.16: Mapa de ubicaciones de los nodos terminales.

Para comprender la elección de cada ubicación, a continuación se presentan las características de cada punto establecido, además, en el anexo D se adjuntan las imágenes correspondientes a la documentación de la instalación de todos los nodos descritos:

- Gateway: Este se encuentra en el edificio Q de la Facultad de Ingeniería, se puede notar que es un punto céntrico respecto a los nodos terminales.
- Taller de conservación: Corresponde a los nodos ubicados a más de 600 metros del Gateway, con una distancia de 1,222 metros aproximadamente, el principal obstáculo es la tapa metálica del registro, seguido de árboles y edificios.
- Filmoteca: Similar al nodo ubicado en el taller de conservación, este está ubicado a 1,173 metros del Gateway, por lo que ambos están a más de 600 metros y cuentan con una cantidad de obstáculos parecidos, incluida la tapa metálica.
- Auditorio Alfonso Caso: Con una distancia al Gateway de 585 metros, entra en el caso de los nodos con distancia entre 200 y 600 metros, con una cantidad de obstáculos considerablemente baja, debida principalmente a la tapa metálica del registro y árboles.
- Biblioteca: Ubicado en la biblioteca Enrique Rivero Borrell del Anexo de la Facultad de Ingeniería, y con una distancia al Gateway de 267 metros, entra en la misma categoría que el nodo del Auditorio Alfonso Caso, con la diferencia de que este presenta como obstáculos únicamente edificios y no la tapa metálica, ya que se encuentra en un medidor externo.
- Edificio 5 (Instituto de Ingeniería): Este se ubica a 113 metros del Gateway y presenta grandes obstáculos ocasionados por edificios, árboles y una tapa de concreto, se considera para los nodos entre 0 y 200 metros.

- Edificio P (Facultad de Ingeniería): Al presentar una distancia de 57 metros del Gateway, este es el nodo más cercano y con menor presencia de obstáculos, por ello fue empleado para el análisis de consumo energético del capítulo anterior, en el caso de estudio del nodo con buena calidad de transmisión/recepción.
- Edifico 17 (Instituto de Ingeniería): Finalmente, con 157 metros de distancia al Gateway, este nodo presenta una gran cantidad de obstáculos y fue elegido como contraparte de estudio para el nodo con mala calidad de transmisión/recepción del capítulo cuatro, gracias a su fácil acceso.

Una limitante para el monitoreo de flujo de agua de los sensores fue la disponibilidad para conectarse a estos en cada ubicación; es por ello que, de entre las 7 ubicaciones descritas, sólo fue posible conectar 3 nodos terminales a los sensores de flujo de agua de la red de PUMAGUA y con un periodo de envío de datos de una hora, estos son:

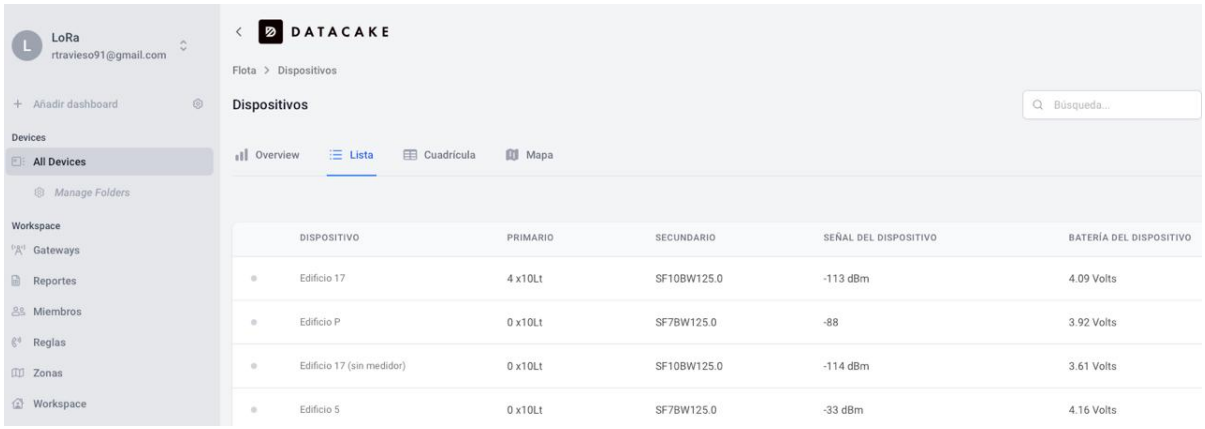
1. Edificio 5.
2. Biblioteca.
3. Edificio 17: En esta ubicación se instalaron dos nodos, un nodo conectado al medidor de flujo de agua y otro no.

De acuerdo con lo anterior, todos los demás nodos tendrán la función de mandar paquetes sin conteo de pulsos por parte del medidor con un periodo de 10 minutos, no obstante, son de gran importancia ya que servirán para recolectar la información sobre sus señales LoRa. La recolección de todos los datos relacionados a su configuración de transmisión, voltaje y conteo de pulsos se logra mediante la aplicación web llamada Datacake, entre sus principales funciones están:

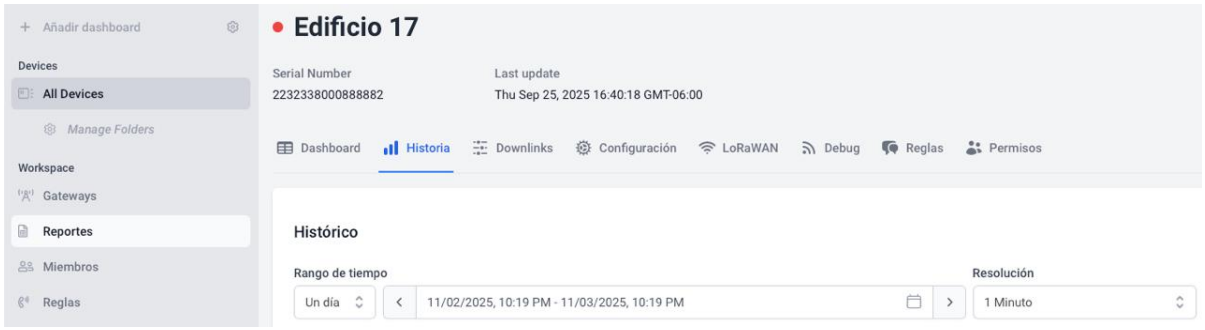
- Conexión de dispositivos (sensores y Gateways) desde TTN.
- Visualización de datos en ventanas interactivas.
- Almacenamiento y análisis de mediciones en la nube.
- Configuración de alertas en tiempo real.

Esta fue configurada por un compañero del grupo de trabajo y para acceder a ella se inicia sesión en el siguiente enlace, con las siguientes credenciales:

- Email address: rtravieso91@gmail.com
- Password: Heltec.0rg



(a) Ventana de inicio de sesión en Datacake.



(b) Ventanas *Dashboard* e *Historia* del dispositivo registrado den Datacake.

Figura 5.17: Interfaz de usuario inicial de Datacake

Una vez iniciada sesión, de manera muy similar a TTN, se enlistan los dispositivos registrados (figura 5.17a). Al dar clic en cualquiera de los dispositivos se abre una ventana interactiva llamada *Dashboard*, junto a esta ventana existe otra llamada *Historia* (figura 5.17b), en la que se puede configurar una gráfica con los valores históricos almacenados durante el máximo de siete días, arrojando como resultado gráficas como la que se presenta en la figura 5.18.

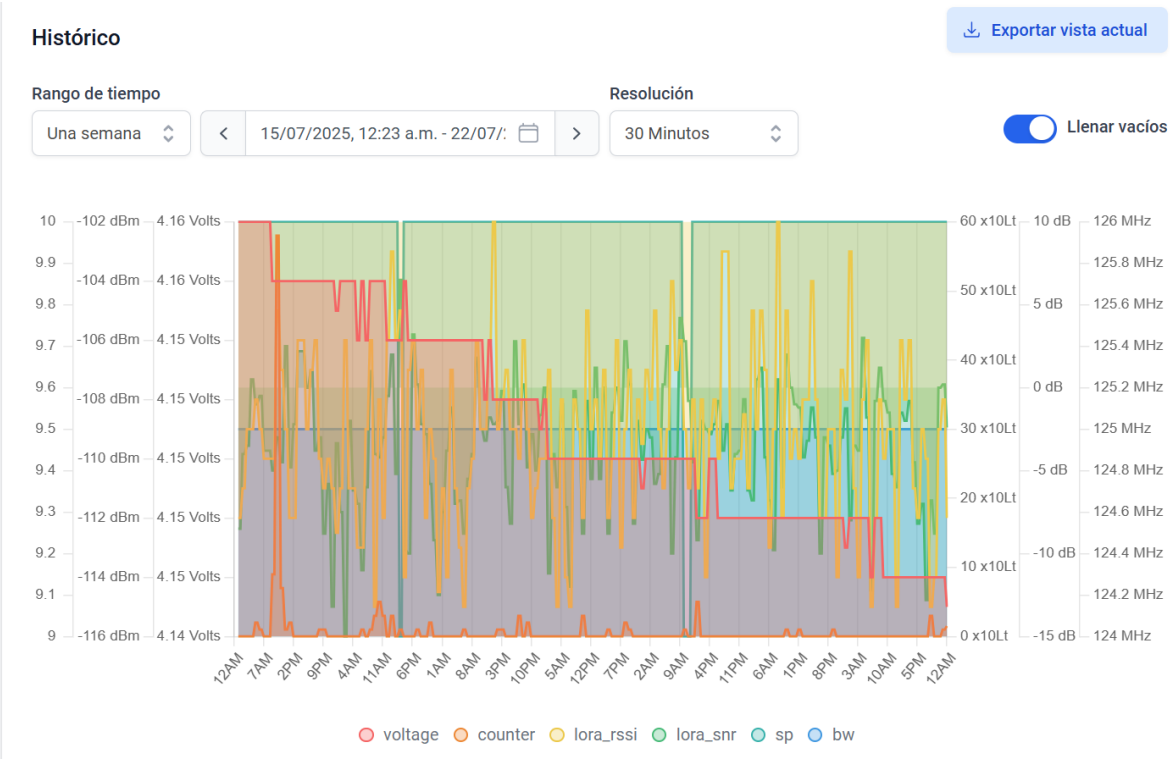


Figura 5.18: Visualización de los datos de los nodos terminales en Datacake.

Esta aplicación web, como se observa, permite almacenar hasta por 7 días los datos de los nodos en su versión gratuita, aunque es relativamente poco tiempo, esta aplicación es suficiente ya que también da la posibilidad de exportar los datos a un archivo con formato .csv para su posterior análisis. Las variables almacenadas para cada nodo, según se aprecia, son las siguientes:

- Voltaje de la batería [V].
- Conteo de pulsos del medidor de flujo de agua.
- RSSI de la señal LoRa recibida [dBm].
- SNR de la señal LoRa recibida [dB].
- Spread factor.
- Ancho de banda [Hz].

Habiendo establecido lo anterior y habiendo completado la instalación y registro de todos los nodos terminales en TTN y Datacake, resta almacenar y analizar los datos en la siguiente sección.

5.4 Análisis del desempeño del sistema

El análisis propuesto para esta sección está basado en los datos recabados de los nodos terminales instalados a lo largo de n semanas, además de evaluar el desempeño del sistema, también se busca encontrar una relación entre el consumo energético (afectado por el número de retransmisiones) y la calidad de las señales que llegan al Gateway. Para encontrar la relación descrita es imprescindible conocer lo que el SNR y el RSSI representan.

- SNR: La relación señal a ruido es un parámetro que indica la relación entre la potencia de la señal transmitida y la potencia de ruido del ambiente [49]. Teóricamente, un receptor necesita de un SNR positivo para demodular la señal, sin embargo, LoRa tiene la capacidad de demodular señales con un SNR negativo, es decir, puede demodular señales a pesar de que la potencia de ruido sea mayor que la potencia de la señal recibida.

$$SNR[dB] = 10log\frac{S}{N} \tag{5.1}$$

  Donde:

  - $S[dBm]$: potencia de la señal.
  - $N[dBm]$: potencia de ruido.

- RSSI: El indicador de intensidad de la señal recibida, o Received Signal Strength Indicator en inglés, es un indicador auxiliar para determinar si la señal recibida tiene la suficiente potencia como para establecer conexiones estables entre transmisor y receptor [50]. Al igual que la potencia de ruido, este indicador tiene como unidades los dBm, lo cual es una medida de potencia relativa con referencia a un miliWatt.
  El valor de potencia de ruido varía dependiendo las lecturas, las anteriores, por ejemplo, hablan de valores de entre -80 y -95 [dBm], este rango de valores es el más adoptado en la literatura, a pesar de ello, la forma exacta de obtener este valor es con base en los fundamentos de la modulación LoRa [46].

$$\text{Noise Floor}[dBm] = 10 \cdot \log_{10}(k \cdot T \cdot BW \cdot 1000) \tag{5.2}$$

  Donde:

  - $k = 1.38 \times 10^{-23}\,\text{J K}^{-1}$ es la constante de Boltzmann.
  - $T = 293\,\text{K}$ es la temperatura ambiente estándar.
  - BW es el ancho de banda en Hz.
  - El factor 1000 convierte de W a mW.

Sustituyendo los valores constantes y conocidos y, considerando un ancho de banda de 125 $[kHz]$:

$$\text{Noise Floor} = 10 \cdot \log_{10}(1.38x10^{-23} \cdot 293 \cdot 125000 \cdot 1000)$$

$$\text{Noise Floor} = -174 + 10 \cdot \log_{10}(125000)$$

$$\text{Noise Floor} \approx -123.03\ [dBm]$$

De esta manera, el valor de potencia de ruido esperado en los nodos terminales debe ser cercano a $-123[dBm]$, considerando que el ancho de banda será el mismo en todos. La mejor forma de analizar y relacionar los datos recabados, al igual que se hizo en el capítulo 4, es obtener sus respectivos histogramas de frecuencias; en consecuencia, se juntaron todos los datos de cada nodo terminal en un solo archivo formato .csv con el que posteriormente un programa de MATLAB (presente en el anexo E) obtendrá los histogramas. El análisis debe ser individual, de este modo, a continuación se desarrolla lo necesario para cada nodo terminal.

5.4.1 Taller de conservación y Filmoteca

Los nodos terminales instalados en estas ubicaciones son la excepción y no se puede realizar un análisis debido a que no lograron establecer una conexión exitosa con el Gateway central a pesar de emplear como estrategia de optimización la extracción de sus antenas a través de las tapas metálicas.

Aunque no se logró una conexión exitosa, se pudo observar que el Gateway logró recibir sus señales con intentos de *Join request* durante más de una semana. Este resultado es desfavorable en un principio, no obstante, también es muy útil ya que permitió identificar que no es posible instalar nodos terminales a grandes distancias y en las condiciones de obstáculos a las que se enfrentarán estos dos nodos, dejando de lado por completo todas las técnicas de optimización, con excepción de la colocación de otro u otros Gateways distribuidos de mejor manera en Ciudad Universitaria.

5.4.2 Auditorio Alfonso Caso

A partir de los histogramas presentados en la figura 5.19 se aprecia que los valores de SNR, y RSSI más frecuentes son -7.65[dB] y -111.25[dBm], respectivamente; esto indica que, en la mayoría de las transmisiones las condiciones de las señales recibidas son considerablemente más débiles que la potencia de ruido (-103.6[dBm]), calculada con la expresión 5.1, esto queda claro dado que el valor de SNR es negativo.

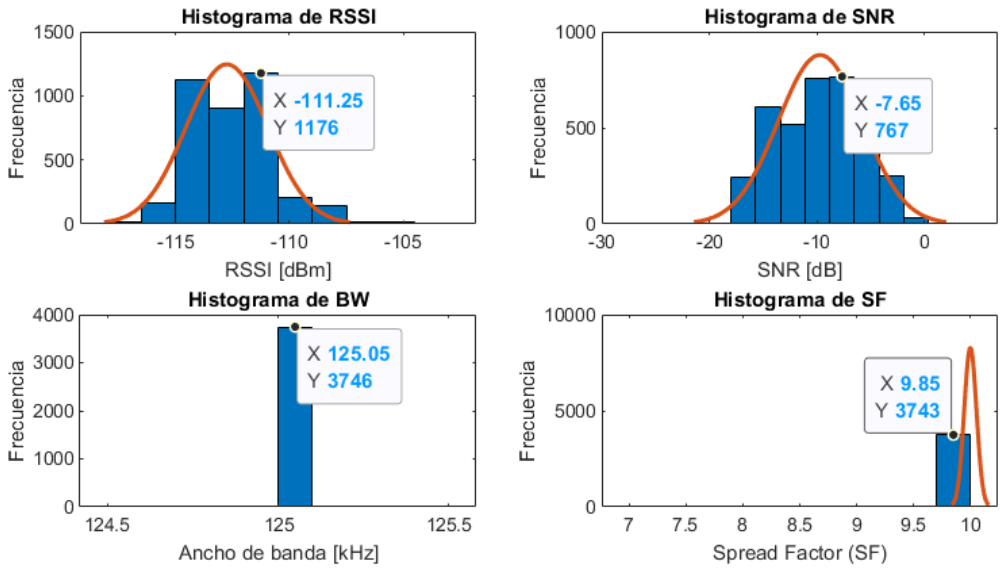


Figura 5.19: Histogramas de datos recabados en el nodo del Auditorio Alfonso Caso.

El hecho de recibir señales débiles de este nodo impacta directamente en el spread factor que el ADR de LoRaWAN configura, esto se comprueba igualmente en el histograma de spread factor presentado, al tener señales débiles LoRa requiere incrementar el spread factor arriba del valor inicial de 7 para poder recibir con éxito los paquetes, en este caso se mantuvo prácticamente en 10.

Otra forma de visualizar el impacto que tiene esta mala calidad en la recepción y transmisión del nodo es a través de los registros históricos, en este caso, el más importante es el voltaje de la batería. La figura 5.20 presenta la gráfica con los históricos de todos los datos recabados, además de verse los cambios en el tiempo del spread factor, SNR y RSSI, lo más destacable es ver cómo va cayendo el voltaje de la batería a través del tiempo; se aprecia un incremento al inicio ya que se recargó la batería, a fin de tener su máxima capacidad para la prueba.

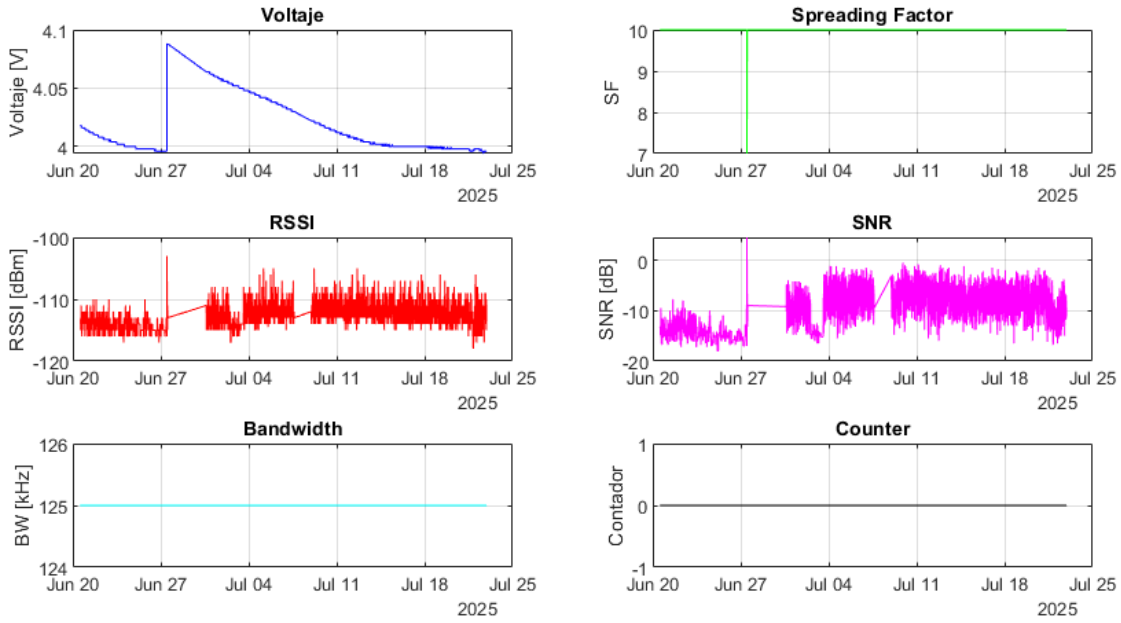


Figura 5.20: Histórico de datos recabados en el nodo del Auditorio Alfonso Caso.

No está de más mencionar que el contador se mantuvo constante, pues este nodo no se conecto a ningún a un medidor de PUMAGUA. También el ancho de banda se mantiene constante en 125[kHz], este debido a que un aumento de este reduciría el alcance y, para aumentar el alcance, LoRaWAN modifica el spread factor y no el ancho de banda, de este modo, se espera el mismo comportamiento para el ancho de banda en todos los nodos del experimento.

5.4.3 Biblioteca Enrique Rivero Borrell

En el caso de este nodo los histogramas de SNR, RSSI, spread factor y ancho de banda se muestran en la figura 5.21. Los valores de SNR, y RSSI más frecuentes son 1.5[dB] y -108.81[dBm], respectivamente. Siguiendo con el análisis de estos parámetros, la calidad de la señal recibida para el nodo fue mejor comparada con la del nodo analizado previamente, es un caso curioso porque, aunque el RSSI indica mayor potencia de la señal recibida, el SNR indica que la potencia de ruido también incrementó respecto al nodo anterior (-110.31[dBm]).

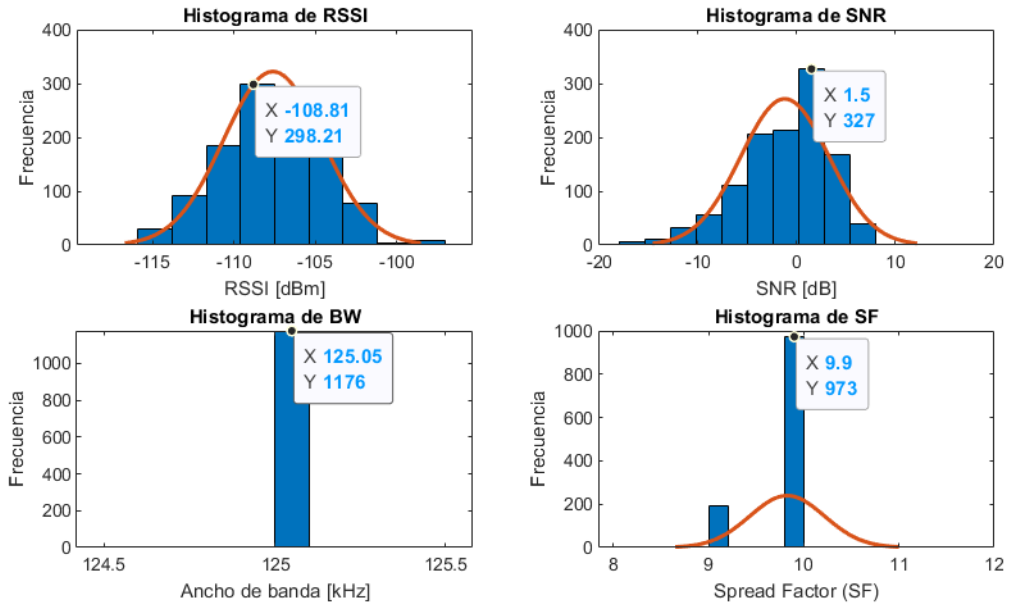


Figura 5.21: Histogramas de datos recabados en el nodo de la Biblioteca Enrique Rivero Borrell.

El cambio en la mejora de la señal se debe a que este nodo no se encuentra dentro de un registro con tapa de metal, si no que al estar en un registro exterior, sólo se enfrenta a obstáculos como árboles y edificios, además de que su distancia al Gateway es menor.

Por otro lado, aunque existe una mejora en la calidad de la recepción y transmisión, el hecho de que el spread factor se mantenga en 10 indicaría que el comportamiento entre este nodo y el anterior sería similar en consumo de batería, a pesar de ello, en la gráfica de voltaje de la batería de la figura 5.22 se nota que la caída es menor, lo que podría sugerir que se presenta una menor cantidad de retransmisiones por cada envío de datos.

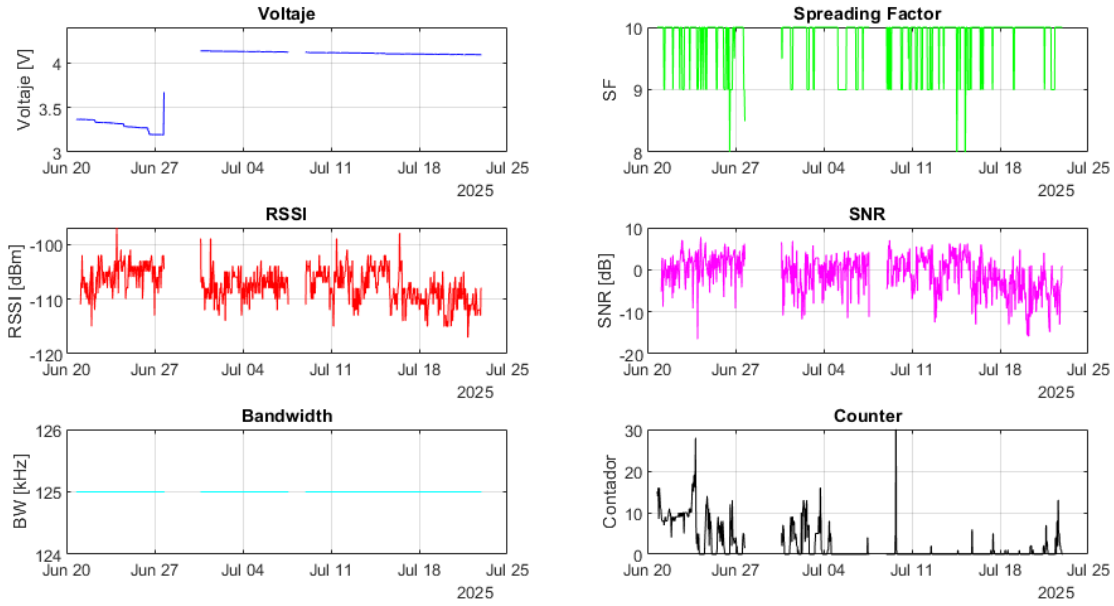


Figura 5.22: Histórico de datos recabados en el nodo de la Biblioteca Enrique Rivero Borrell.

Dado que este es uno de los tres nodos conectado a un medidor de flujo de agua, la gráfica del contador presenta estos picos de consumo, aunque es difícil determinar si se trata de un consumo habitual, el hecho de tener estos *picos* indica lectura correctas por parte del nodo.

5.4.4 Edificio 5

Los histogramas de este nodo, mostrados en la figura 5.23, ofrecen resultados interesantes ya que la calidad de la recepción y transmisión mejora en gran medida, a pesar de que este nodo se encuentra dentro de un registro con tapa de concreto. Los valores de SNR y RSSI,de 10.95[dB] y -85[dBm], respectivamente, indican que la potencia de la señal recibida por el Gateway es considerablemente mayor que la potencia de ruido, que resulta ser menor que en los nodos anteriores (-95.95[dBm]); la combinación de lo observado se ve reflejado en un spread factor de 7, prácticamente constante en todos los eventos registrados.

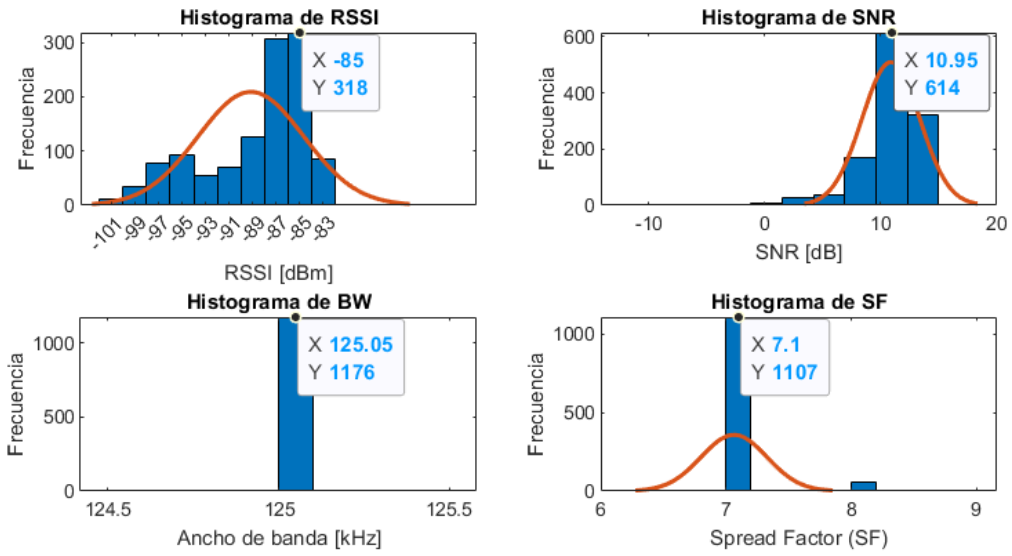


Figura 5.23: Histogramas de datos recabados en el nodo del edificio 5 del Instituto de Ingeniería.

Lo anterior concuerda con una muy poca caída en el voltaje de la batería, según se observa en la figura 5.24, la mejora es sorprendente ya que el nodo, además de estar bajo una tapa de concreto, también se enfrenta a varios obstáculos; sin embargo, el factor clave en la mejora es la distancia, pues hay una diferencia de 154 metros entre el nodo y el Gateway con respecto al nodo de la Biblioteca Enrique Rivero Borrell.

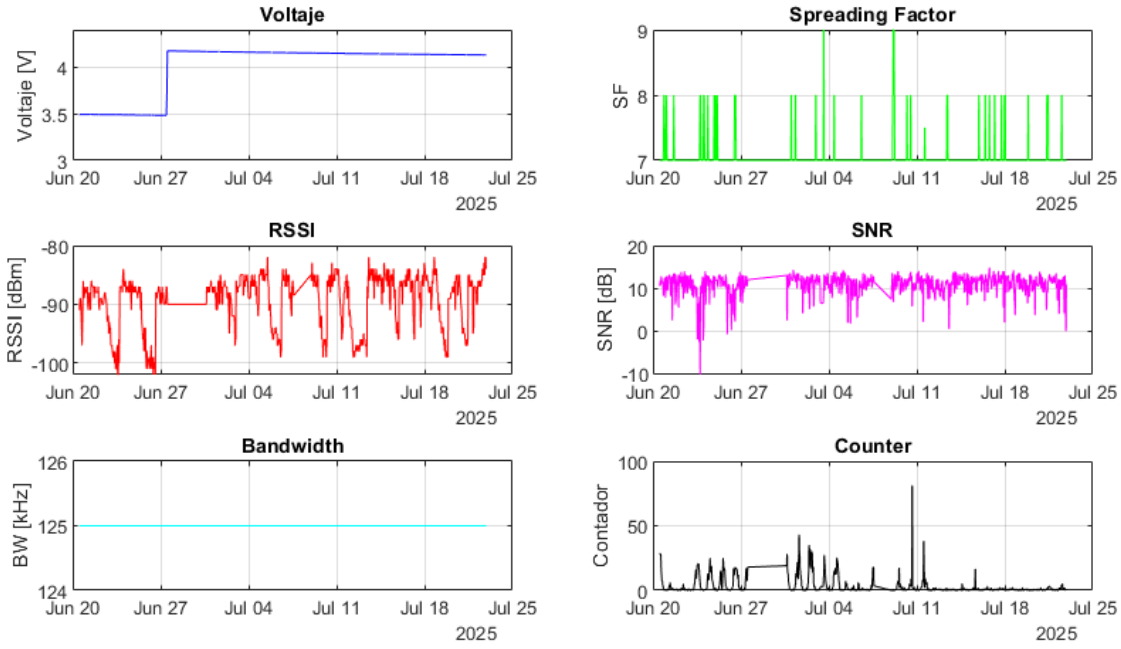


Figura 5.24: Histórico de datos recabados en el nodo del edificio 5 del Instituto de Ingeniería.

Similar al nodo ubicado en la Biblioteca Enrique Rivero Borrell, este nodo se encuentra conectado a un medidor de flujo de agua, la gráfica del contador presenta picos de consumo que, de igual manera, sugieren un correcto funcionamiento de las lecturas del medidor por parte del nodo.

5.4.5 Edifico 17 (con medidor de flujo de agua)

A partir de los histogramas de la figura 5.25, donde se tienen valores de SNR, y RSSI de -0.05[dB] y -106.2[dBm], respectivamente, se puede concluir que este nodo se enfrenta a condiciones similares a las del nodo de la Biblioteca Enrique Rivero Borrell, tanto por SNR y RSSI como por spread factor. Aunque la señal de ambos nodos debe atravesar por obstáculos similares, la similitud en sus parámetros de transmisión podría indicar que la tapa metálica del registro de este nodo atenúa la señal de manera equivalente a la distancia extra que presenta el nodo de la Biblioteca Enrique Rivero Borrell, siendo un total de 110 metros extra.

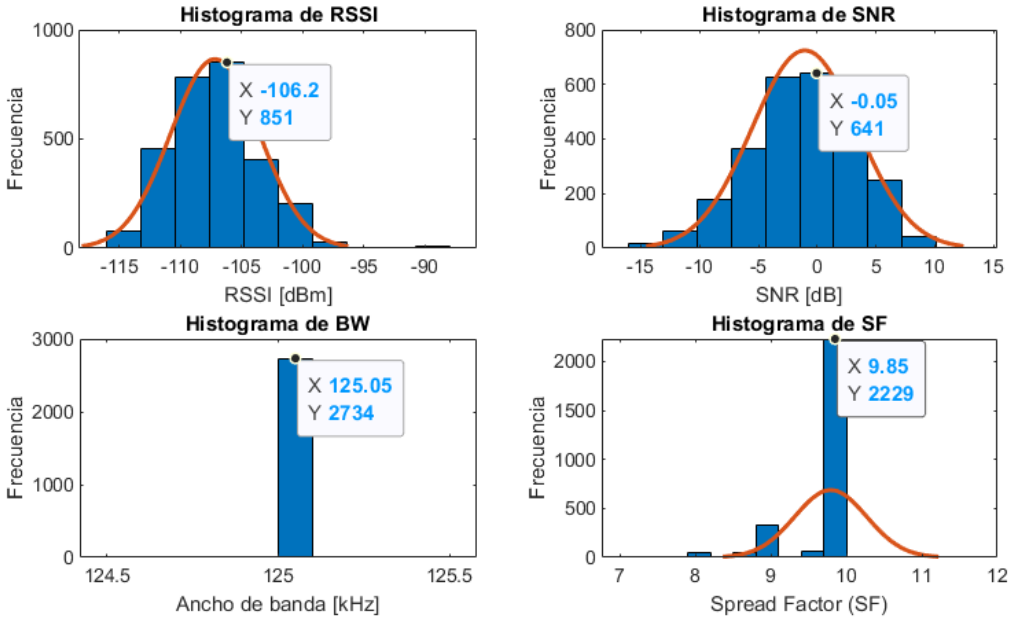


Figura 5.25: Histogramas de datos recabados en el nodo del edificio 17 del Instituto de Ingeniería.

Nuevamente, se puede confirmar la similitud de condiciones puesto que en las mediciones de voltaje de la batería (figura 5.26) tampoco se presenta una caída abrupta, sino que la curva es similar, con una mínima caída de voltaje.

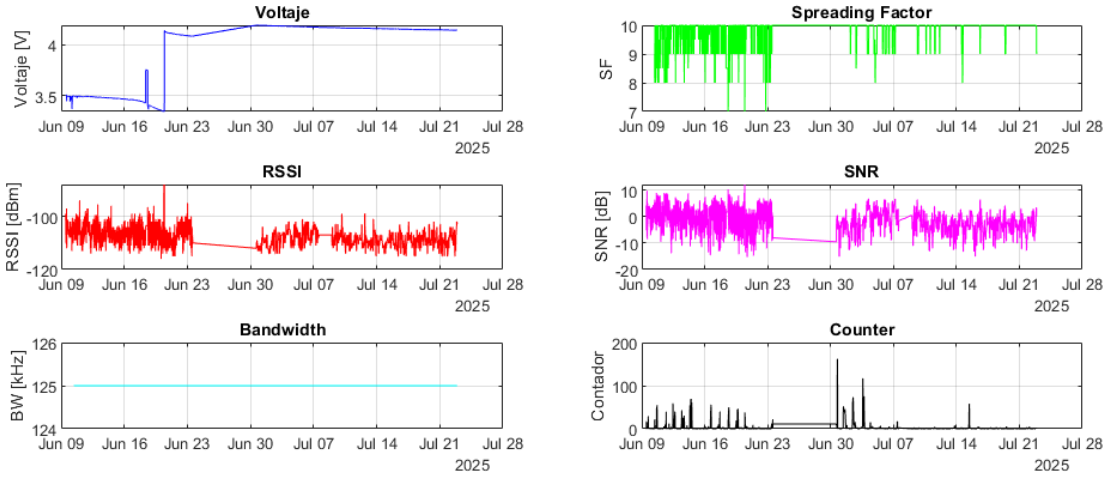


Figura 5.26: Histórico de datos recabados en el nodo del edificio 17 del Instituto de Ingeniería.

En este caso, es más notorio el correcto funcionamiento de conteo de pulsos provenientes del medidor de flujo de agua, gracias a que en las primeras semanas se logra apreciar un consumo

periódico cada día, mientras que en las semanas siguientes, este consumo desaparece debido al inicio del periodo vacacional en Ciudad Universitaria.

5.4.6 Edifico 17 (sin medidor de flujo de agua)

Puesto que este nodo se encuentra en las mismas condiciones que el anterior, no se espera un cambio significativo en los valores de SNR o RSSI, lo que se confirma en los histogramas de la figura 5.27, con valores de SNR de -4.95[dB], RSSI de -110.57[dBm] y spread factor de 10.

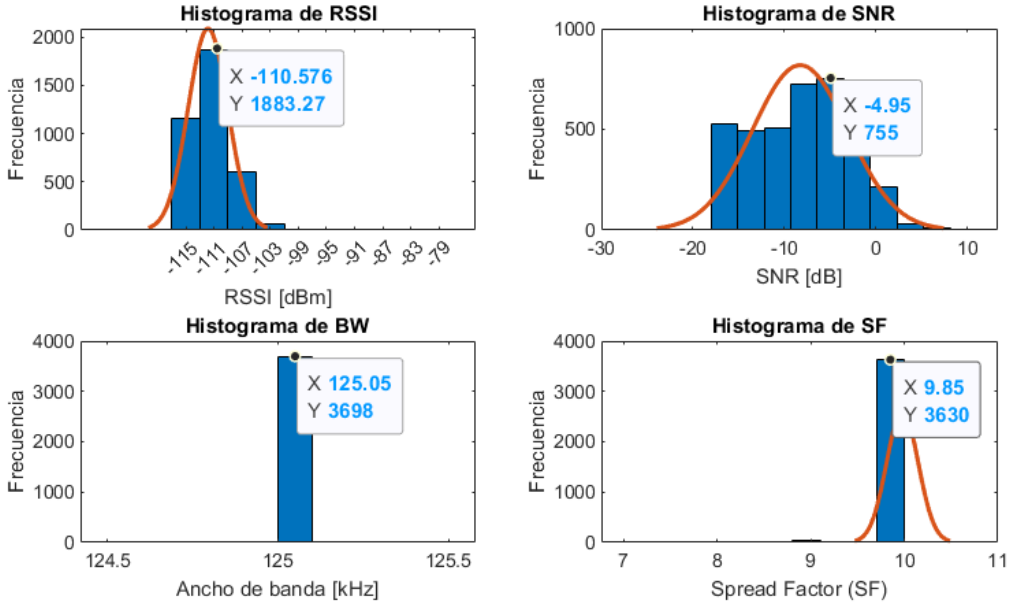


Figura 5.27: Histogramas de datos recabados en el nodo del edificio 17 del Instituto de Ingeniería.

El objetivo de tener dos nodos en la misma ubicación, pero uno conectado a un medidor de flujo de agua y otro no, es observar la diferencia en consumo de corriente debido al conteo de pulsos. Después de comparar la diferencia en las caídas de voltaje de sus respectivas baterías, se tiene un comportamiento inesperado, siendo que el nodo con medidor de flujo de agua presenta una caída de 48$[mV]$ desde su punto más alto, mientras que el nodo que no cuenta con medidor conectado presenta una caída de 102$[mV]$.

En este sentido, dado que este nodo sin medidor de flujo de agua presenta una mayor caída, la conclusión es que no es posible determinar en tan poco tiempo el impacto en la batería del conteo de pulsos, por lo que para este estudio se requiere recabar datos por un periodo de tiempo mayor a un mes, como lo fue en la recopilación de este trabajo.

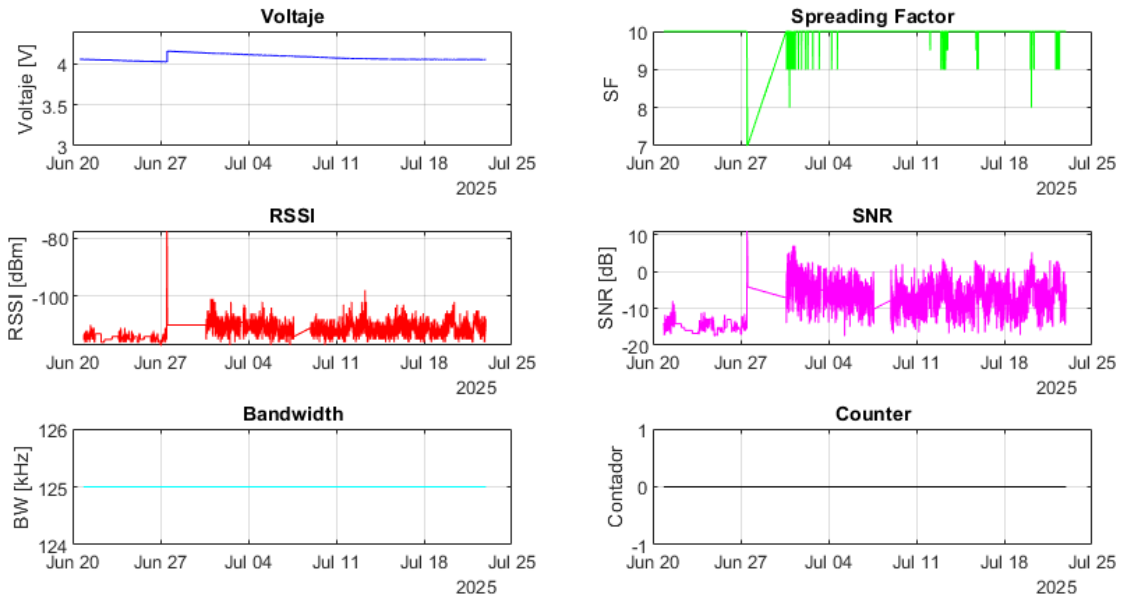


Figura 5.28: Histórico de datos recabados en el nodo del edificio 17 del Instituto de Ingeniería.

Al no tener un medidor de flujo de agua conectado, la gráfica correspondiente tiene un valor constante de 0.

5.4.7 Edifico P

Para este nodo, en la figura 5.29, se observan valores SNR de 11.55[dB], RSSI de -96.4[dBm], respectivamente, de tal modo que se tiene el mejor valor de SNR y, por lo tanto, se cumple la hipótesis de que este nodo sería el que presentaría mejores características de transmisión y recepción gracias a su corta distancia al Gateway, cosa que concuerda con su spread factor de 7.

Es importante mencionar su similitud en parámetros respecto al nodo del edificio 5, y, de igual manera, se le puede atribuir a que la atenuación debida a la tapa metálica es similar a la atenuación debida a la diferencia de distancia al Gateway que existe entre este nodo y el del edificio 5. Otra conclusión significativa es que las tapas de concreto no atenúan las señales tanto como lo hacen las tapas metálicas, lo cual es un factor importante a considerar en futuras instalaciones.

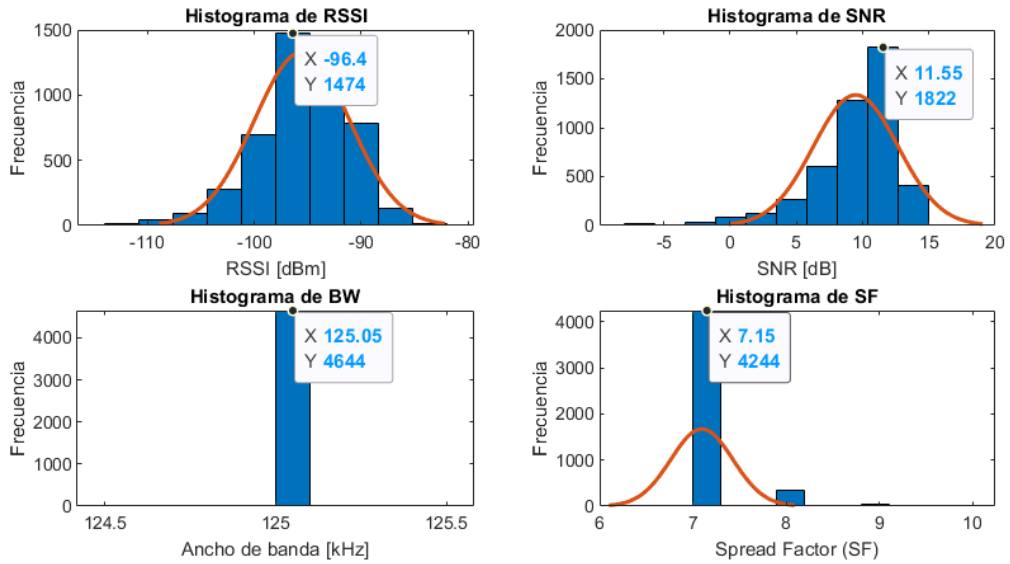


Figura 5.29: Histogramas de datos recabados en el nodo del edificio P del Anexo de la Facultad de Ingeniería.

Por último, como era de esperarse, el voltaje en la batería de este nodo es muy pequeña ($32[mV]$), cumpliendo con lo esperado.

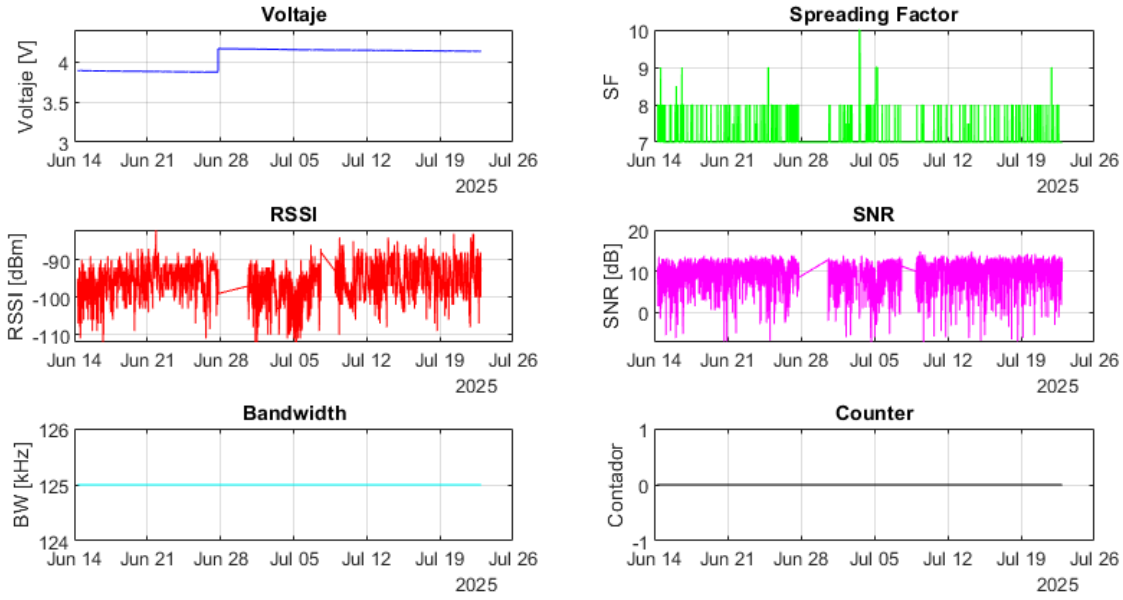


Figura 5.30: Histórico de datos recabados en el nodo del edificio P del Anexo de la Facultad de Ingeniería.

Este nodo tampoco se encuentra conectado a un medidor, por lo que la gráfica correspondiente tiene un valor constante de 0.

5.4.8 Comparativa

En esta comparativa final se resumen los datos recabados y analizados de cada nodo terminal del experimento 3 de las pruebas de campo, de tal manera se construyó la tabla 5.1, donde se puede apreciar la caída de voltaje en la batería de cada nodo y de esta forma, relacionarla con sus parámetros de calidad de la señal recibida.

Nodo	**Caída de voltaje [mV]**	**SNR [dB]**	**RSSI [dBm]**	**SF**	**medidor flujo**	**Distancia [m]**	**Obstáculos**
Auditorio Alfonso Caso	92	-7.65	-111.25	10	No	585	Muchos
Biblioteca E.R. Borrell	42	1.5	-108.81	10	Sí	267	Muchos
Edificio 5	44	10.95	-85	7	Sí	113	Intermedio
Edificio 17 (con sensor)	48	-0.05	-106.2	10	Sí	157	Muchos
Edificio 17 (sin sensor)	102	-4.95	-110.57	10	No	157	Muchos
Edificio P	32	11.55	-96.4	7	No	57	Pocos

Tabla 5.1: Resumen de parámetros por nodo.

En resumen, se puede concluir que las tapas metálicas de los registros son de los elementos que más atenúan las señales, seguidos por la distancia y obstáculos como lo son los edificios y árboles .

Para poder notar el efecto de la descarga de las baterías debido al conteo de pulsos por parte de los medidores es necesario hacer un análisis con una muestra de datos unas cuantas veces mayor. Según los cálculos del capítulo 4, esta batería podría durar hasta casi 4 años en el peor de los casos y más de 16 en el mejor de los casos, considerando condiciones ideales, por lo que un mes de recopilación de datos no es suficiente para comprobarlo. Otro motivo que evidencia la necesidad de un mayor tiempo de recopilación de datos es que no se logró apreciar una diferencia considerable de caída de voltaje en los nodos que transmiten cada diez minutos respecto a los que transmiten cada media hora, como lo sugiere la hipótesis.

El resultado de las pruebas es positivo ya que se ha mantenido la conexión entre nodos, Gateway central y la aplicación web Datacake, la cual fue clave para poder verificar el correcto funcionamiento de conteo de pulsos y, por lo tanto, el buen monitoreo de flujo de agua por parte de los nodos diseñados. Finalmente, el resultado más favorable es observar que las caídas de voltaje son realmente pequeñas, lo que promete una duración de la batería acorde a lo estimado en el capítulo 4.

5.5 Metodologías para la optimización del consumo energético

Puesto que se ha demostrado que la duración de la batería depende de las condiciones en las que el nodo terminal transmite, para esta sección se proponen dos metodologías para la optimización de su consumo: optimización a través de hardware o métodos que impliquen cambios físicos en la configuración del sistema, y optimización a través de software, ambas tienen el mismo objetivo de mejorar la calidad de la transmisión y, por ende, el consumo de energía por ciclo de trabajo. A continuación se presentan las propuestas para cada una de las metodologías mencionadas.

5.5.1 Optimización por hardware

Este tipo de optimización es la más evidente ya que el consumo de energía depende de la calidad de la transmisión; al mejorar las condiciones en las que el nodo transmite, se pueden reducir el número de retransmisiones en cada ciclo de trabajo. De acuerdo con las pruebas documentadas se observó que, al cerrarse las tapas de los registros de PUMAGUA (principalmente metálicas), la calidad de la señal disminuye significativamente y, en algunos casos, deja de ser detectada por el Gateway, por tal motivo se tienen las siguientes propuestas:

1. Emplear una antena de mayor calidad y ganancia: Esta propuesta ayuda principalmente a compensar la atenuación sufrida por las tapas en el momento de la transmisión, aunque el fabricante de la tarjeta de desarrollo (Heltec) no ofrece especificaciones técnicas sobre la antena LoRa incluida y, usada en los nodos, en algunos sitios web se puede encontrar que la ganancia de este tipo de antenas es de 3[dB], por lo que, antenas que garanticen ganancias de 5[dB], de acuerdo con lo explicado en [51], supondrían una mejora en el rango de transmisión y recepción de casi el doble.
2. Colocar los nodos/antenas fuera de los registros: Con esto se busca reducir la atenuación de la señal provocada por las tapas metálicas, esto implica mejoras en la transmisión, sin necesidad de aumentar la potencia o el spread factor y en la recepción, lo que reduciría en

gran medida el número de retransmisiones. Para cumplir lo anterior existen tres opciones:

- Elevar el nodo con un mástil.
- Sacar la antena por un agujero en las tapas.
- Elevar el Gateway central.

En los dos primeros casos, la instalación de los nodos o las antenas se complica, lo que puede resultar inviable considerando que se planea tener una red numerosa de nodos. El tercer caso es el más viable, pues en lugar de subir cientos de antenas de los nodos, sólo se sube el Gateway, no obstante, podría ser redundante con la siguiente estrategia propuesta.

3. Implementar otro Gateway más cercano a las zonas más remotas: Esta propuesta, aunque tiene un costo inicial mayor a las anteriores, es probablemente la mejor solución ya que reduciría las distancias entre los nodos y el Gateway, por lo que la señal sería recibida sin tanta atenuación y sin necesidad de aumentar la potencia, ni invertir en infraestructura adicional o antenas de calidad.

Dentro de las propuestas, las más viables son hacer una combinación entre sacar las antenas de los registros y usar mejores antenas, ya que el costo podría ser menor (dependiendo de la antena) y el procedimiento más sencillo que colocar un mástil o implementar otro Gateway. Sin embargo, después de hacer las pruebas correspondientes del capítulo, se observó que estas soluciones pueden no ser suficientes para las ubicaciones más remotas, por lo que la solución restante, y la mejor, es la implementación de uno o más Gateways distribuidos estratégicamente para cubrir toda Ciudad Universitaria.

5.5.2 Optimización por software

Este tipo de optimización, aunque quizás menos obvia, puede ser una de las más sencillas de implementar y que podría dar mejores resultados. La idea también surge a partir de las pruebas de campo, al notar que el spread factor puede variar entre transmisiones y retransmisiones, se abre la posibilidad de implementar las siguientes estrategias:

1. Desactivar el ADR y fijar el spread factor: Con base en las mediciones de corriente y con base en las pruebas de este capítulo, se observó que el ADR de LoRaWAN cambia constantemente los parámetros de transmisión entre los diferentes envíos de paquetes, principalmente el spread factor y, en menor medida, la potencia de transmisión, por ello, al desactivar esta función y fijar estos parámetros según los valores más frecuentes registrados, se puede forzar al sistema para que evite reducir el spread factor o la potencia, de modo que siempre estaría transmitiendo con los parámetros que garanticen la recepción por parte del Gateway y del

nodo terminal, lo que se traduce en una cantidad menor de retransmisiones.

2. Reducir el número de retransmisiones permitidas: Esta estrategia funge como complemento de la anterior, pues al garantizar que la señal será recibida por el Gateway y el nodo terminal, serán necesarios menos intentos de retransmisión o, en el mejor de los casos, un solo evento de transmisión bastará para recibir los paquetes con éxito, cosa que impacta crucialmente en el consumo por ciclo de trabajo.

Ambas estrategias parecen sencillas de implementar, no obstante, el código de LoRaWAN proporcionado por Heltec para el CubeCell permite desactivar el ADR, pero no existe una opción o forma que permita fijar el spread factor a placer. Por otro lado, reducir el número de reintentos es muy sencillo, puesto que solo se necesita modificar la variable de retransmisiones en el código de la sección 3.4.3. Si bien lo mejor sería implementar ambas estrategias, las limitantes de firmware presentes en la tarjeta solo hacen posible reducir el número de reintento de transmisiones. Ademas de lo anterior, tomar una de estas acciones requiere de un mayor estudio del comportamiento de los parámetros de las señales, de acuerdo a lo concluido en el análisis del sistema.

Capítulo 6

Conclusiones

6.1 Logros de la tesis

A lo largo del desarrollo de esta tesis se fueron alcanzando los objetivos específicos planteados y relacionados con la implementación de redes LPWAN para aplicaciones IoT, particularmente en el monitoreo de consumo de agua y con el uso de la tecnología LoRa y el protocolo LoRaWAN. Cada uno de estos logros contribuyó al despliegue de la red diseñada, al cumplimiento del objetivo general y, finalmente, a aceptar las hipótesis planteadas en la introducción de este trabajo, pues se analizó y validó exitosamente que el desempeño de la red es el adecuado. Los logros puntuales se resumen en los siguientes puntos:

- Se logró el desarrollo de una red LoRaWAN funcional de bajo consumo energético y resistente a condiciones de alta humedad para monitorear el consumo de agua potable de las tuberías de PUMAGUA, el cual es el principal objetivo de la tesis.
- Se logró el correcto acondicionamiento y adaptación de la infraestructura de PUMAGUA, especialmente de los medidores de flujo de agua y hasta su instalación en los respectivos registros.
- Se realizó el despliegue exitoso de seis nodos terminales conectados al Gateway central, los cuales transmiten datos fiables de consumo de agua.
- Se desarrollaron metodologías para la estimación de la vida útil de las baterías, las cuales comprenden la medición de corriente, análisis del comportamiento del nodo terminal y métodos de cálculo de energía.

6.2 Conclusiones generales

Conociendo los logros obtenidos, se pueden rescatar como conclusiones positivas que el objetivo general de la tesis se cumplió de manera satisfactoria, partiendo del hecho de que se desplegó una red de 6 nodos terminales que transmiten datos correctamente al Gateway central y que, además, se pueden visualizar en la aplicación web, ya sea TTN o Datacake. En consecuencia de lo anterior, se valida que la tecnología LoRa y el protocolo LoRaWAN son aptos para el desarrollo de redes LPWAN en entornos urbanos o rurales, en los que existen diversas condiciones obstaculizan las

señales, como lo es Ciudad Universitaria.

Las pruebas de los nodos terminales dejan en evidencia que se debe cumplir estrictamente con un encapsulado que proporcione protección contra el agua y la humedad presentes en las condiciones del ambiente en que estos se instalan, lo que debe ser considerado meticulosamente en el diseño de una versión mejorada para la aplicación. En cuanto a la tecnología usada, el CubeCell resultó cumplir con las capacidades necesarias para los nodos terminales, dado que estos demostraron ser eficientes en consumo, resistentes a entornos húmedos y, sobre todo, fiables en la transmisión de datos, asegurando su funcionamiento a largo plazo.

Aunque el dispositivo desarrollado logra un consumo de energía lo suficientemente bajo como para tener una autonomía superior a tres años, este podría verse aun más reducido mediante hardware con un diseño electrónico más a la medida, en otras palabras, evitar usar componentes innecesarios en la tarjeta de desarrollo, así como emplear componentes de mayor calidad en la construcción del dispositivo, un ejemplo de ello podrían ser antenas mejor adaptadas a las características de instalación del nodo.

El análisis de consumo energético demostró la necesidad de crear modelos matemáticos más exactos para determinar dicho consumo, o bien, el desarrollo de métodos de estimación más avanzados que consideren todos los factores y/o parámetros cambiantes involucrados en la transmisión de paquetes por medio del protocolo LoRaWAN. En este sentido, la conclusión más relevante es la necesidad de implementar las estrategias de optimización de consumo energético propuestas en el capítulo 5, otra opción es desarrollar e implementar estrategias alternativas que permitan reducir dicho consumo y que, a su vez, hagan más sencilla la estimación de la vida útil de las baterías, por ejemplo, al reducir los efectos de los cambios de parámetros como el SF o el número de retransmisiones.

Por último, pero no menos importante, las pruebas finales del sistema, descritas en su respectivo capítulo dan pie al despliegue de más nodos terminales conectados a la red de medidores de PUMAGUA, esto gracias a que se ha observado un correcto conteo de pulsos proveniente de los medidores, De esta manera, se puede pensar en analizar dichos datos para detectar posibles fugas o simplemente empezar a monitorear de forma normal el consumo de las tuberías donde se encuentran instalados los dispositivos.

6.3 Trabajo a futuro

El trabajo a futuro de la tesis consiste en la mejora de las cualidades más relevantes del sistema, como lo es el bajo consumo energético de los nodos terminales, su resistencia a las condiciones del ambiente, la mejora en la calidad de las transmisiones y, por supuesto, el crecimiento de la red. Estas mejoras están encaminadas con los siguientes enfoques:

- **Optimización de las transmisiones de los paquetes LoRaWAN.**
  En este apartado se propone aplicar las estrategias de optimización, previamente mencionadas en el capítulo 5, más viables y de mayor impacto, el reto es aplicar estas estrategias de forma que el costo de estas sea mínimo respecto a los beneficios agregados al sistema. Aplicar estrategias, como la fijación del SF, implican un mayor estudio de las ubicaciones de los nodos y la cobertura del, o de los Gateways, por ello, se requiere analizar más en detalle de cada una de las metodologías o, alternativamente, idear otras.
- **Implementación de mejores algoritmos para la estimación de la vida útil de las baterías.**
  La idea de tener mejores algoritmos para estimar la vida útil de las baterías es lograr tener valores numéricos y gráficas que ilustren el comportamiento de la vida del nodo de forma precisa y en el menor tiempo y estudio posible. El objetivo propuesto o fijado con esta mejora es que un programa o algoritmo se encargue de determinar automáticamente todas las estadísticas relacionadas con las transmisiones de cada nodo y, con base en ello, estimar la vida útil de las baterías, además, sería de gran utilidad que el algoritmo diseñado, a partir de los datos de corriente, pudiera sugerir alguna estrategia de optimización particular para cada nodo de acuerdo con las características presentadas según su ubicación.
- **Diseño de un nodo terminal adaptado a las necesidades.**
  Con un nuevo diseño del dispositivo se pretende mejorar las características actuales en cuanto a resistencia, fiabilidad y consumo energético. Como ya se mencionó, un diseño electrónico con los componentes únicamente esenciales podría reducir el consumo energético, disminuir el tamaño del circuito, mejorar las conexiones con la batería, con el medidor de flujo y la antena, evitando posibles fallos causados por falsos contactos o componentes parásitos del circuito.
  Para un rediseño completo del dispositivo, se podría ajustar el tamaño del encapsulado a partir de la capacidad de la batería necesaria para cumplir con el tiempo de vida requerido en el nodo terminal. Este cambio o ajuste en las baterías exige una investigación y un estudio de las diferentes tecnologías de baterías existentes, su objetivo sería identificar aquellas tecnologías cuyas características sean las que mejor se adapten a las necesidades, en otras

palabras, se busca implementar baterías capaces de alimentar los nodos terminales durante el periodo más largo posible, y así, obtener un resultado final lo más cercano a lo estimado. En resumen, este punto sugiere implementar baterías con el menor porcentaje de autodescarga, la mayor capacidad de retención de carga y, sobre todo, que sus especificaciones estén garantizadas por algún fabricante.

- **Escalamiento de la red IoT.**

  Escalar la red no sólo requiere de un aumento en el número de nodos terminales, sino que también en el número de Gateways operativos, ubicar los Gateways de forma estratégica tendría como objetivo lograr una distribución más equitativa entre nodos y los Gateways y, por lo tanto, reduciría la carga de nodos al Gateway y la distancia entre ellos. Este aspecto se relaciona estrechamente con la optimización de las transmisiones, pues al haber más Gateways repartidos en Ciudad Universitaria, los nodos establecerán conexión con el más cercano o con el que ofrezca la comunicación de mejor calidad; de este modo, se esperaría que los nodos no tengan que retransmitir o hacer grandes cambios en el SF.

Apéndice A

Código principal nodo esclavo

```
#include <softSerial.h>
#include "LoRaWan_APP.h"
#include "Arduino.h"

uint16_t cnt = 0;
void cntIncrease(){
  cnt++;
  Serial.println(cnt);
  delay(50);
}

/*
 * set LoraWan_RGB to Active,the RGB active in loraWan
 * RGB red means sending;
 * RGB purple means joined done;
 * RGB blue means RxWindow1;
 * RGB yellow means RxWindow2;
 * RGB green means received done;
 */

/* OTAA para*/
uint8_t devEui[] = { 0x22, 0x32, 0x33, 0x80, 0x00, 0x88, 0x88, 0x81 };
uint8_t appEui[] = { 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x81 };
uint8_t appKey[] = { 0x88, 0x88, 0x88, 0x88, 0x88, 0x88, 0x88, 0x88, 0x88, 0x88, 0x88, 0x88,
     0x88, 0x66, 0x81 };

/* ABP para*/
uint8_t nwkSKey[] = { 0x15, 0xb1, 0xd0, 0xef, 0xa4, 0x63, 0xdf, 0xbe, 0x3d, 0x11, 0x18, 0x1e, 0x1e
    , 0xc7, 0xda,0x85 };
uint8_t appSKey[] = { 0xd7, 0x2c, 0x78, 0x75, 0x8c, 0xdc, 0xca, 0xbf, 0x55, 0xee, 0x4a, 0x77, 0x8d
    , 0x16, 0xef,0x67 };
uint32_t devAddr =  ( uint32_t )0x007e6ae1;

/*LoraWan channelsmask, default channels 0-7*/
uint16_t userChannelsMask[6]={ 0x00FF,0x0000,0x0000,0x0000,0x0000,0x0000 };

/*LoraWan region, select in arduino IDE tools*/
LoRaMacRegion_t loraWanRegion = ACTIVE_REGION;

/*LoraWan Class, Class A and Class C are supported*/
DeviceClass_t  loraWanClass = LORAWAN_CLASS;

/*the application data transmission duty cycle.  value in [ms].*/
uint32_t appTxDutyCycle = 20000;

/*OTAA or ABP*/
bool overTheAirActivation = LORAWAN_NETMODE;

/*ADR enable*/
bool loraWanAdr = LORAWAN_ADR;

/* set LORAWAN_Net_Reserve ON, the node could save the network info to flash, when node reset not
    need to join again */
bool keepNet = LORAWAN_NET_RESERVE;

/* Indicates if the node is sending confirmed or unconfirmed messages */
bool isTxConfirmed = LORAWAN_UPLINKMODE;

/* Application port */
uint8_t appPort = 1700;
/*!
```

```
* Number of trials to transmit the frame, if the LoRaMAC layer did not
* receive an acknowledgment. The MAC performs a datarate adaptation,
* according to the LoRaWAN Specification V1.0.2, chapter 18.4, according
* to the following table:
*
* Transmission nb | Data Rate
* ----------------|-----------
* 1 (first)       | DR
* 2               | DR
* 3               | max(DR-1,0)
* 4               | max(DR-1,0)
* 5               | max(DR-2,0)
* 6               | max(DR-2,0)
* 7               | max(DR-3,0)
* 8               | max(DR-3,0)
*
* Note, that if NbTrials is set to 1 or 2, the MAC will not decrease
* the datarate, in case the LoRaMAC layer did not receive an acknowledgment*/

uint8_t confirmedNbTrials = 4;

/* Prepares the payload of the frame */
static void prepareTxFrame( uint8_t port )
{
 /*appData size is LORAWAN_APP_DATA_MAX_SIZE which is defined in "commissioning.h".
 *appDataSize max value is LORAWAN_APP_DATA_MAX_SIZE.
 *if enabled AT, don't modify LORAWAN_APP_DATA_MAX_SIZE, it may cause system hanging or failure.
 *if disabled AT, LORAWAN_APP_DATA_MAX_SIZE can be modified, the max value is reference to lorawan
     region and SF.
 *for example, if use REGION_CN470,
 *the max value for different DR can be found in MaxPayloadOfDatarateCN470 refer to DataratesCN470
     and BandwidthsCN470 in "RegionCN470.h".
 */
    uint16_t voltage = getBatteryVoltage();
    appDataSize = 4;
    appData[0] = (cnt >> 8) & 0xFF;       // High byte of cnt
    appData[1] = cnt & 0xFF;              // Low byte of cnt
    appData[2] = (voltage >> 8) & 0xFF;   // High byte of voltage
    appData[3] = voltage & 0xFF;          // Low byte of voltage
    cnt = 0;
}


void setup() {
 Serial.begin(115200);
#if(AT_SUPPORT)
 enableAt();
#endif
 deviceState = DEVICE_STATE_INIT;
 LoRaWAN.ifskipjoin();
  //PINMODE_INPUT_PULLDOWN(GPIO1);
  pinMode(GPIO1, INPUT);
  attachInterrupt(GPIO1,cntIncrease,RISING);
}

void loop()
{
 switch( deviceState )
 {
  case DEVICE_STATE_INIT:
  {
#if(LORAWAN_DEVEUI_AUTO)
   LoRaWAN.generateDeveuiByChipID();
#endif
#if(AT_SUPPORT)
   getDevParam();
#endif
   printDevParam();
   LoRaWAN.init(loraWanClass,loraWanRegion);
   deviceState = DEVICE_STATE_JOIN;
   break;
  }
  case DEVICE_STATE_JOIN:
```

```
  {
   LoRaWAN.join();
   break;
  }
  case DEVICE_STATE_SEND:
  {
   prepareTxFrame( appPort );
   LoRaWAN.send();
   deviceState = DEVICE_STATE_CYCLE;
   break;
  }
  case DEVICE_STATE_CYCLE:
  {
   // Schedule next packet transmission
   txDutyCycleTime = appTxDutyCycle + randr( 0, APP_TX_DUTYCYCLE_RND );
   LoRaWAN.cycle(txDutyCycleTime);
   deviceState = DEVICE_STATE_SLEEP;
   break;
  }
  case DEVICE_STATE_SLEEP:
  {
   LoRaWAN.sleep();
   break;
  }
  default:
  {
   deviceState = DEVICE_STATE_INIT;
   break;
  }
 }
}
```

Código A.1: Programa LoRaWAN para el nodo terminal.[1]

[1]Este código fue tomado y modificado de los programas ejemplo del fabricante del CubeCell (Heltec). El código fuente se puede consultar en el siguiente enlace.

Apéndice B

Códigos de adquisición de corriente para el sistema de microcontrolador y el sensor INA226

```
/***************************************************************************
* Example sketch for the INA226_WE library
*
* This sketch shows how to use the INA226 module in continuous mode.
*
* Further information can be found on:
* https://wolles-elektronikkiste.de/ina226 (German)
* https://wolles-elektronikkiste.de/en/ina226-current-and-power-sensor (English)
*
***************************************************************************/
#include <Wire.h>
#include <INA226_WE.h>
#define I2C_ADDRESS 0x40

char txpacket [5];

/* There are several ways to create your INA226 object:
 * INA226_WE ina226 = INA226_WE(); -> uses I2C Address = 0x40 / Wire
 * INA226_WE ina226 = INA226_WE(I2C_ADDRESS);
 * INA226_WE ina226 = INA226_WE(&Wire); -> uses I2C_ADDRESS = 0x40, pass any Wire Object
 * INA226_WE ina226 = INA226_WE(&Wire, I2C_ADDRESS);
 */
INA226_WE ina226 = INA226_WE(I2C_ADDRESS);

void setup() {
  Serial.begin(921600);
  Wire.begin();
  Wire.setClock(400000);  // Establece I2C a 400 kHz para mejorar la velocidad
  if(!ina226.init()){
    Serial.println("Failed to init INA226. Check your wiring.");
    while(1){}
  }
  /* Set Number of measurements for shunt and bus voltage which shall be averaged
  * Mode *     * Number of samples *
  AVERAGE_1            1 (default)
  AVERAGE_4            4
  AVERAGE_16          16
  AVERAGE_64          64
  AVERAGE_128        128
  AVERAGE_256        256
  AVERAGE_512        512
  AVERAGE_1024      1024
  */
  //ina226.setAverage(AVERAGE_16); // choose mode and uncomment for change of default
  /* Set conversion time in microseconds
     One set of shunt and bus voltage conversion will take:
     number of samples to be averaged x conversion time x 2
     * Mode *         * conversion time *
     CONV_TIME_140          140 µs
     CONV_TIME_204          204 µs
     CONV_TIME_332          332 µs
     CONV_TIME_588          588 µs
     CONV_TIME_1100         1.1 ms (default)
     CONV_TIME_2116       2.116 ms
     CONV_TIME_4156       4.156 ms
     CONV_TIME_8244       8.244 ms
  */
  //ina226.setConversionTime(CONV_TIME_1100); //choose conversion time and uncomment for change of
```

```
    default
  /* Set measure mode
  POWER_DOWN - INA226 switched off
  TRIGGERED  - measurement on demand
  CONTINUOUS  - continuous measurements (default)
  */
  ina226.setMeasureMode(CONTINUOUS); // choose mode and uncomment for change of default
  /* Set Resistor and Current Range
     if resistor is 5.0 mOhm, current range is up to 10.0 A
     default is 100 mOhm and about 1.3 A*/
  ina226.setResistorRange(0.1, 1.3); // choose resistor 0.1 Ohm and gain range up to 1.3A
  /* If the current values delivered by the INA226 differ by a constant factor
     from values obtained with calibrated equipment you can define a correction factor.
     Correction factor = current delivered from calibrated equipment / current delivered by INA226
  */
  ina226.setCorrectionFactor(0.91);
  Serial.println("INA226 Current Sensor Example Sketch - Continuous");
  ina226.waitUntilConversionCompleted(); //if you comment this line the first data might be zero
}

void loop() {
  float shuntVoltage_mV = ina226.getShuntVoltage_mV();
  float busVoltage_V = ina226.getBusVoltage_V();
  float current_mA = ina226.getCurrent_mA();
  //float power_mW = ina226.getBusPower();
  float loadVoltage_V = busVoltage_V + (shuntVoltage_mV/1000);
  if (current_mA < 0.8){current_mA = 0.030;}

  // Transmisión serial optimizada
  sprintf(txpacket,"%0.3f, %0.2f",current_mA, loadVoltage_V);  //start a package
  Serial.println(txpacket);
  delayMicroseconds(300);
}

void checkForI2cErrors(){
  byte errorCode = ina226.getI2cErrorCode();
  if(errorCode){
    Serial.print("I2C error: ");
    Serial.println(errorCode);
    switch(errorCode){
      case 1:
        Serial.println("Data too long to fit in transmit buffer");
        break;
      case 2:
        Serial.println("Received NACK on transmit of address");
        break;
      case 3:
        Serial.println("Received NACK on transmit of data");
        break;
      case 4:
        Serial.println("Other error");
        break;
      case 5:
        Serial.println("Timeout");
        break;
      default:
        Serial.println("Can't identify the error");
    }
    if(errorCode){
      while(1){}
    }
  }
}
```

Código B.1: Programa de adquisición de datos del sensor INA226.[1]

```
    timeData = [];                  % Para almacenar los tiempos en ms
    corriente = [];                 % Para almacenar los valores de corriente
```

[1]El código B.1 fue tomado y modificado de la librería creada por el usuario *wollewald*, la cual se puede consultar en el siguiente enlace.

```
voltaje = [];                    % Para almacenar los valores de voltaje

% Configuración del puerto serial
serialPort = 'COM10';            % Puerto serial usado
baudRate = 921600;               % Tasa de baudios del dispositivo
serialObj = serialport(serialPort, baudRate);

% Tiempo máximo de lectura (en milisegundos)
tiempoMaximo = 720*60*1000;            % Establecido en 8 horas
startTime = datetime('now');     % Tiempo de inicio

% Lectura de datos durante el tiempo establecido
while milliseconds(datetime('now') - startTime) < tiempoMaximo
    data = readline(serialObj);                % Lee los datos del puerto serial
    valores = str2double(split(data, ','));    % Convierte los datos leídos en valores numé
ricos
    if length(valores) == 3                     % Asegura que hay tres valores (tiempo,
corriente y voltaje)
        timeData = [timeData; valores(1)];     % Almacena el tiempo
        corriente = [corriente; valores(2)];     % Almacena el valor de corriente
        voltaje = [voltaje; valores(3)];         % Almacena el valor de voltaje
    end
end
clear serialObj;  % Limpieza: cierra el puerto serial

% Calcular la potencia
potencia = corriente .* voltaje;

figure
% Crear la primera gráfica (corriente)
subplot(3,1,1);
plot(timeData, corriente, '-b', 'LineWidth', 1.5);
xlabel('t [ms]');
ylabel('Corriente [mA]');
title('Corriente vs tiempo');
grid on;

% Crear la segunda gráfica (voltaje)
subplot(3,1,2);
plot(timeData, voltaje, '-r', 'LineWidth', 1.5);
xlabel('t [ms]');
ylabel('Voltaje [V]');
title('Voltaje vs tiempo');
grid on;

% Crear la tercera gráfica (potencia)
subplot(3,1,3);
plot(timeData, potencia, '-g', 'LineWidth', 1.5);
xlabel('t [ms]');
ylabel('Potencia [mW]');
title('Potencia vs tiempo');
grid on;

% Encabezados
header = {'Tiempo [ms]', 'Corriente [mA]', 'Voltaje [V]', 'Potencia [mW]'};
% Nombre del archivo
filename = 'descargaBateria2.csv';
% Escribe los encabezados
writecell(header, filename);  % Escribe los encabezados
% Escribe los datos numéricos
data = [timeData, corriente, voltaje, potencia];
writematrix(data, filename, 'WriteMode', 'append');  % Añade los datos debajo de los
encabezados
```

Código B.2: Recepción, almacenamiento y graficación de los valores de correinte en MATLAB

Apéndice C

Códigos del cálculo de consumo energético.

```
% Abrir archivo de Excel
filename = 'nodo2_13_06_2025.csv'; % Nombre del archivo Excel
opts = detectImportOptions(filename, 'VariableNamingRule', 'preserve');
data = readtable(filename, opts); % Leer la tabla del archivo Excel con opciones

% Asignar las columnas de datos usando los nombres exactos
timeData = data.('Timestamp');      % Tiempo en ms
corriente = data.('Value');         % Corriente en mA
timeData = timeData * 1000;         % Convertir tiempo a ms
corriente = corriente * 1000;       % Convertir corriente a mA

% Crear la primera gráfica (datos completos)
figure;
plot(timeData, corriente, 'r-', 'LineWidth', 2, 'DisplayName', 'Datos reales');
xlabel('t [ms]');
ylabel('Corriente [mA]');
title('Corriente vs tiempo');
grid on;
% Establecer el rango del eje y entre -5 y 200
ylim([-5, 200]);

% Solicitar al usuario el intervalo de tiempo
%transmisiones = input('Por favor, ingrese la cantidad de transmisiones en la gráfica: ');

umbral_Tx = 70;                 % mA, altura mínima del pico
tiempo_entre_Tx = 280000;       % ms = 4 minutos
[pks, locs] = findpeaks(corriente, timeData, ...
    'MinPeakHeight', umbral_Tx, ...
    'MinPeakDistance', tiempo_entre_Tx);
hold on
plot(locs, pks, 'o')  % marca los picos
transmisiones = length(pks);

% Dividir el tiempo en intervalos iguales
primer_intervalo = 120000;
% Punto de división después del primer intervalo
tiempo_inicial = timeData(1);
tiempo_final = timeData(end);  % Último valor del vector timeData
tiempo_restante_inicio = tiempo_inicial + primer_intervalo;
tiempo_restante = tiempo_final - tiempo_restante_inicio;
% Dividir el tiempo restante en (transmisiones) intervalos iguales
intervalos_restantes = linspace(tiempo_restante_inicio, tiempo_final, transmisiones);
% Combinar el primer intervalo fijo con los intervalos restantes
intervalos = [tiempo_inicial, tiempo_restante_inicio, intervalos_restantes(2:end)];

% Calcular el área bajo la curva usando la función trapz (regla del trapecio)
area = trapz(timeData, corriente);
area = area / 3600000; % Convertir de mAs a mAh
fprintf(['El área bajo la curva entre %.2f ms y %.2f ms con el método del trapecio es:' ...
    ' %.6f mAh\n'], tiempo_inicial, tiempo_final, area);

% Calcular el área bajo la curva usando regla de simpson
area1 = simpson(timeData, corriente,[],'1/3'); % Simpson's 1/3 rule
area1 = area1 / 3600000;
fprintf(['El área bajo la curva entre %.2f ms y %.2f ms con la regla de simpson es:' ...
    ' %.6f mAh\n'], tiempo_inicial, tiempo_final, area1);

carga_promedio = (area + area1) / 2;
carga_promedio = carga_promedio/transmisiones;

tiempos_intervalos = cell(1, transmisiones);  % Para almacenar tiempos de cada intervalo
```

```
corriente_intervalos = cell(1, transmisiones);  % Para almacenar tiempos de cada intervalo
areas_trapecio = zeros(1, transmisiones);   % Para áreas
areas_simpson = zeros(1, transmisiones);   % Para áreas
picos_tx_intervalos = zeros(1, transmisiones); % Para contar picos de transmision
picos_rx_intervalos = zeros(1, transmisiones); % Para contar picos de recepcion
% Parámetros para findpeaks
umbral_Tx = 70;  % Máximo valor de pico (mA)
tiempo_entre_Tx = 500;  %valor el ms
% Extraer los tiempos para cada intervalo
for i = 1:transmisiones
    % Encontrar los índices de los tiempos dentro del intervalo actual
    idx = (timeData >= intervalos(i)) & (timeData < intervalos(i+1));
    % Almacenar los tiempos del intervalo actual en la celda
    tiempos_intervalos{i} = timeData(idx);
    corriente_intervalos{i} = corriente(idx);
    areas_trapecio(i) = trapz(tiempos_intervalos{i}, corriente_intervalos{i})/3600000;
    areas_simpson(i) = simpson(tiempos_intervalos{i}, corriente_intervalos{i},[],'1/3')/3600000;

    % Detectar picos de tx
    [pks, locs] = findpeaks(corriente_intervalos{i}, tiempos_intervalos{i}, ...
        'MinPeakHeight', umbral_Tx, 'MinPeakDistance',tiempo_entre_Tx);
    hold on
    plot(locs,pks,'x')
    picos_tx_intervalos(i) = length(pks); % Contar picos válidos

    % Estimar picos de rx
    if picos_tx_intervalos(i) == 1
        picos_rx_intervalos(i) = 1;
    elseif picos_tx_intervalos(i) >= 2
        picos_rx_intervalos(i) = 2 * picos_tx_intervalos(i)-1;
    end
end

% Crear el primer bloque de texto (Carga consumida)
text_content1 = cell(1 + transmisiones, 1);
text_content1{1} = 'Carga total consumida:';
text_content1{2} = sprintf('Regla del trapecio: %.6f mAh', area);
text_content1{3} = sprintf('Regla de simpson : %.6f mAh', area1);
text_content1{4} = sprintf('Carga promedio por ciclo: %.6f mAh', carga_promedio);

% Crear el segundo bloque de texto (Número de picos)
text_content2 = cell(1 + transmisiones, 1);
text_content2{1} = 'Número de transmisiones por intervalo:';
for i = 1:transmisiones
    text_content2{1 + i} = sprintf('Intervalo %d: %d ', i, picos_tx_intervalos(i));
end

% Crear el tercer bloque de texto (Carga por intervalo)
text_content3 = cell(1 + transmisiones, 1);
text_content3{1} = 'Carga consumida por intervalo:';
% Agregar el texto en la esquina inferior izquierda
for i = 1:transmisiones
    text_content3{1 + i} = sprintf('Intervalo %d: %.6f mAh', i, areas_trapecio(i));
end

% Dibujar líneas verticales para los límites de los intervalos
for i = 1:length(intervalos)
    xline(intervalos(i), '--r', 'LineWidth', 1.2); % Color rojo
end

text_position_y = max(corriente);
text_position_x1 = min(timeData) + 0.05 * (max(timeData) - min(timeData));
text_position_x2 = text_position_x1 + 0.3 * (max(timeData) - min(timeData));
text_position_x3 = text_position_x2 + 0.3 * (max(timeData) - min(timeData));

% Agregar los bloques de texto a la gráfica
text(text_position_x1, text_position_y, text_content3, ...
    'HorizontalAlignment', 'left', 'VerticalAlignment', 'bottom', ...
    'BackgroundColor', 'white', 'FontSize', 10);
text(text_position_x2, text_position_y, text_content2, ...
    'HorizontalAlignment', 'left', 'VerticalAlignment', 'bottom', ...
    'BackgroundColor', 'white', 'FontSize', 10);
text(text_position_x3, text_position_y, text_content1, ...
```

```
    'HorizontalAlignment', 'left', 'VerticalAlignment', 'bottom', ...
    'BackgroundColor', 'white', 'FontSize', 10);
hold off;  % Liberar la figura


% Crear arreglo de índices de intervalo (1, 2, ..., transmisiones)
interval_idx = (1:transmisiones)';
% Tabla por intervalo
detalle_intervalos = table;
detalle_intervalos.NombreArchivo = repmat({filename}, transmisiones, 1);
detalle_intervalos.Intervalo = interval_idx;
detalle_intervalos.CargaTrapecio_mAh = areas_trapecio';
detalle_intervalos.CargaSimpson_mAh = areas_simpson';
detalle_intervalos.PicosTx = picos_tx_intervalos';
detalle_intervalos.PicosRx = picos_rx_intervalos';

% Nombre del archivo por intervalo
detalle_filename = 'consumos_nodo2.csv';

% Leer el archivo si existe
if isfile(detalle_filename)
    resumen_existente = readtable(detalle_filename);
    % Revisar si ya se procesó este archivo
    if any(strcmp(resumen_existente.NombreArchivo, filename))
        fprintf('El archivo "%s" ya fue procesado. No se volverá a registrar.\n', filename);
        return;  % Detener aquí para evitar duplicados
    end
    % Si no está en el registro, añadir al archivo existente
    writetable(detalle_intervalos, detalle_filename, 'WriteMode', 'append');
    fprintf('El archivo "%s" ha sido registrado con exito.\n', filename);
else
    % Si el archivo no existe, crearlo con cabecera
    writetable(detalle_intervalos, detalle_filename);
    fprintf('El archivo "%s" se ha creado con éxito.\n', detalle_filename);
end
```

Código C.1: programa de cálculo de consumo de energía de los nodos terminales en MATLAB

```
filename = 'consumos_nodo2.csv';
data = readtable(filename);

% 1. Extraer datos de Join Request (Intervalo == 1)
joinData = data(data.Intervalo == 1, :);
joinTx = joinData.PicosTx;
joinRx = joinData.PicosRx;
joinCargaTrap = joinData.CargaTrapecio_mAh;
joinCargaSimpson = joinData.CargaSimpson_mAh;

% 2. Extraer el resto de los datos (Intervalo != 1)
restData = data(data.Intervalo ~= 1, :);
restTx = restData.PicosTx;
restRx = restData.PicosRx;
restCargaTrap = restData.CargaTrapecio_mAh;
restCargaSimpson = restData.CargaSimpson_mAh;

% 3. Calcular estadísticas para Join Request
joinStatsTx.mean = mean(joinTx);
joinStatsTx.std = std(joinTx);  % Corregido: std() en lugar de mean()
joinStatsRx.mean = mean(joinRx);
joinStatsRx.std = std(joinRx);  % Corregido: std() en lugar de mean()
joinStatsCargaTrap.mean = mean(joinCargaTrap);
joinStatsCargaTrap.std = std(joinCargaTrap);
joinStatsCargaSimpson.mean = mean(joinCargaSimpson);
joinStatsCargaSimpson.std = std(joinCargaSimpson);

% 4. Calcular estadísticas para el resto
statsTx.mean = mean(restTx);
statsTx.std = std(restTx);
statsRx.mean = mean(restRx);
statsRx.std = std(restRx);
statsCargaTrap.mean = mean(restCargaTrap);
statsCargaTrap.std = std(restCargaTrap);
```

```
statsCargaSimpson.mean = mean(restCargaSimpson);
statsCargaSimpson.std = std(restCargaSimpson);

% 5. Mostrar resultados
fprintf(" Estadísticas:\n");
fprintf("- Join request Tx: Media = %.2f, Desviación = %.2f\n", joinStatsTx.mean, joinStatsTx.std)
    ;
fprintf("- Join request Rx: Media = %.2f, Desviación = %.2f\n", joinStatsRx.mean, joinStatsRx.std)
    ;
fprintf("- Join request carga Trapecio: Media = %.6f, Desviación = %.6f\n", joinStatsCargaTrap.
    mean, joinStatsCargaTrap.std);
fprintf("- Join request carga Interp: Media = %.6f, Desviación = %.6f\n", joinStatsCargaSimpson.
    mean, joinStatsCargaSimpson.std);
fprintf("- Picos Tx: Media = %.2f, Desviación = %.2f\n", statsTx.mean, statsTx.std);
fprintf("- Picos Rx: Media = %.2f, Desviación = %.2f\n", statsRx.mean, statsRx.std);
fprintf("- Carga Trapecio: Media = %.6f, Desviación = %.6f\n", statsCargaTrap.mean, statsCargaTrap
    .std);
fprintf("- Carga Interp: Media = %.6f, Desviación = %.6f\n", statsCargaSimpson.mean,
    statsCargaSimpson.std);


% Histogramas
figure;

subplot(4,2,1);
histfit(restTx, 10, 'normal'); xlabel('Picos Tx'); ylabel('Frecuencia');
title('Distribución de picos Tx');

subplot(4,2,2);
histfit(restRx, 10, 'normal'); xlabel('Picos Rx'); ylabel('Frecuencia');
title('Distribución de picos Rx');

subplot(4,2,3);
histfit(restCargaTrap, 10, 'normal'); xlabel('Carga Trapecio (mAh)'); ylabel('Frecuencia');
title('Distribución de Carga Trapecio');

subplot(4,2,4);
histfit(restCargaSimpson, 10, 'normal'); xlabel('Carga Simpson (mAh)'); ylabel('Frecuencia');
title('Distribución de Carga Simpson');

subplot(4,2,5);
histfit(joinTx, 10, 'normal'); xlabel('Picos Tx JR'); ylabel('Frecuencia');
title('Distribución de picos Tx en Join request');

subplot(4,2,6);
histfit(joinRx, 10, 'normal'); xlabel('Picos Rx JR'); ylabel('Frecuencia');
title('Distribución de picos Rx en Join request');

subplot(4,2,7);
histfit(joinCargaTrap, 20, 'normal'); xlabel('Carga Trapecio (mAh)'); ylabel('Frecuencia');
title('Distribución de Carga Trapecio en Join request');

subplot(4,2,8);
histfit(joinCargaSimpson, 20, 'normal'); xlabel('Carga Simpson (mAh)'); ylabel('Frecuencia');
title('Distribución de Carga Simpson en Join request');
```

Código C.2: Programa de cálculo de estadística de consumos de transmisión de los nodos terminales en MATLAB

Apéndice D

Instalación de los nodos terminales y el Gateway central.

D.1 Instalación del Gateway central para las pruebas de laboratorio y de campo.

(a) Gabinete de protección con salida de las antenas Wi-Fi y 4G.

(b) Perspectiva de vista del Gateway e instalación de la antena LoRa.

Figura D.1: Instalación del Gateway central en la Azotea del Edificio Q de la Facultad de Ingeniería.

D.2 Instalación de los nodos terminales para las pruebas de campo finales.

D.2.1 Taller de conservación

(a) Registro de PUMAGUA

(b) Vista en dirección al Gateway

(c) Colocación con antena

(d) Extracción de la antena

Figura D.2: Instalación del nodo del Taller de conservación.

D.2.2 Filmoteca

(a) Vista en dirección al Gateway

(b) Registro de PUMAGUA

(c) Colocación del nodo

(d) Extracción de la antena

Figura D.3: Instalación del nodo de la Filmoteca.

D.2.3 Auditorio Alfonso Caso

(a) Vista en dirección al Gateway

(b) Registro de PUMAGUA

(c) Colocación del nodo

(d) Vista al a la ubicación del nodo

Figura D.4: Instalación del nodo del Auditorio Alfonso Caso.

D.2.4 Biblioteca

(a) Registro de PUMAGUA

(b) Colocación del nodo

(c) Vista en dirección al Gateway

Figura D.5: Instalación del nodo de la biblioteca Enrique Rivero Borrell.

D.2.5 Edificio 5

(a) Registro de PUMAGUA

(b) Vista en dirección al Gateway

(c) Colocación del nodo

Figura D.6: Instalación del nodo del Edifico 5.

D.2.6 Edificio P

(a) Registro de PUMAGUA

(b) Colocación del nodo

(c) Vista en dirección al Gateway

Figura D.7: Instalación del nodo del Edifico P.

D.2.7 Edifico 17

(a) Registro de PUMAGUA

(b) Vista en dirección al Gateway

(c) Colocación del nodo con medidor

(d) Colocación del nodo sin medidor

Figura D.8: Instalación del nodo del Edificio 17.

Apéndice E

Código de cálculo de estadística de los paráme-tros LoRa de Datacake.

```
% Leer el archivo CSV
filename = 'Biblioteca.csv'; % <- Cambia esto por el nombre real
data = readtable(filename);

data.time = strrep(data.time, 'p.m.', 'PM');
data.time = strrep(data.time, 'a.m.', 'AM');

% Convertir a datetime con el formato correcto
data.time = datetime(data.time, 'InputFormat', 'd/M/yyyy, hh:mm:ss a');

% Variables de interés para histogramas
rssi = data.rssi;
snr = data.snr;
bw = data.bw;
sf = data.sf;

% Figura con histogramas
figure('Name','Histogramas SNR, RSSI, BW, SF');

subplot(2,2,1);
histfit(rssi, 10, 'normal');
xlabel('RSSI [dBm]'); ylabel('Frecuencia');
title('Histograma de RSSI');

subplot(2,2,2);
histfit(snr, 10, 'normal');
xlabel('SNR [dB]'); ylabel('Frecuencia');
title('Histograma de SNR');

subplot(2,2,3);
histfit(bw, 10, 'normal');
xlabel('Ancho de banda [kHz]'); ylabel('Frecuencia');
title('Histograma de BW');

subplot(2,2,4);
histfit(sf, 10, 'normal');
xlabel('Spread Factor (SF)'); ylabel('Frecuencia');
title('Histograma de SF');

% Figura con todas las variables en el tiempo
figure('Name','Evolución temporal de parámetros');
t = data.time;

subplot(3,2,1);
plot(t, data.voltage, 'b'); grid on;
ylim([3 4.4])
ylabel('Voltaje [V]');
title('Voltaje');

subplot(3,2,2);
plot(t, sf, 'g'); grid on;
ylabel('SF'); title('Spreading Factor');

subplot(3,2,3);
plot(t, rssi, 'r'); grid on;
ylabel('RSSI [dBm]'); title('RSSI');

subplot(3,2,4);
```

```
plot(t, snr, 'm'); grid on;
ylabel('SNR [dB]'); title('SNR');

subplot(3,2,5);
plot(t, bw, 'c'); grid on;
ylabel('BW [kHz]'); title('Bandwidth');

subplot(3,2,6);
plot(t, data.counter, 'k'); grid on;
ylabel('Contador'); title('Counter');
```

Código E.1: Programa de cálculo de estadística de los parámetros de las señales LoRa de Datacake en MATLAB

Bibliografía

[1] S. Li, L. D. Xu, and S. Zhao, "The internet of things: A survey," *Springer Science+Business Media New York*, 2014, accessed via Telefonica Tech Media. [Online]. Available: https://www.academia.edu/27394432/The_internet_of_things_a_survey

[2] D. Evans, "The internet of things," *Evolution of the Internet is Changing Everything*, 2011, accessed via Telefonica Tech Media. [Online]. Available: https://media.telefonicatech.com/telefonicatech/uploads/2021/1/126528_Internet_of_Things_IoT_IBSG_0411FINAL.pdf

[3] D.-H. Kim, E.-K. Lee, and J. Kim, "Experiencing lora network establishment on a smart energy campus testbed," *Sustainability*, vol. 11, p. 1917, 03 2019.

[4] Z. J. R. Kamoonah and M. Ilyas, "Investigating the performance of lora communication for nominal lora and interleaved chirp spreading lora," in *2022 International Conference on Artificial Intelligence of Things (ICAIoT)*. Istanbul, Turkey: IEEE, 2022. [Online]. Available: https://ieeexplore.ieee.org/document/10002861

[5] Semtech. (2020) Lora and lorawan: A technical overview. Accessed: Oct. 1, 2023. [Online]. Available: https://www.semtech.com/uploads/technology/LoRa/lora-and-lorawan.pdf

[6] STMicroelectronics, "Connectivity: Introduction to lorawan," 2024, accessed: Feb. 6, 2025. [Online]. Available: https://wiki.st.com/stm32mcu/wiki/Connectivity:Introduction_to_LoRaWAN

[7] The Things Network, "Lorawan," The Things Network, 2024, accessed: Apr. 2024. [Online]. Available: https://www.thethingsnetwork.org/docs/lorawan/

[8] S. Corporation, "Sending and receiving messages," Semtech Corporation, Tech. Rep. AN1200.88, Mar 2024, application Note. [Online]. Available: https://www.semtech.com/uploads/technology/LoRa/sending-and-receiving-messages.pdf

[9] *Medidores industriales Recordall: Medidor de disco de nutación, Manual del usuario*, Badger Meter, 2020, accessed: 2025-10-21. [Online]. Available: https://cf-store.widencdn.net/badgermeter/2/d/e/2de25479-66e3-4656-93a1-5750dae86f27.pdf

[10] Badger Meter, "Recordall® Transmitter Register (RTR)," https://www.badgermeter.com/en-gb/products/registers-transmitters/recordall-transmitter-register-rtr/, accessed: Oct. 20, 2024.

[11] HelTec Automation Technology Co., Ltd., *HTCC-AB01 V2 LoRa Node Development Kit (Rev1.1)*, HelTec Automation Technology Co., Ltd., Chengdu, Sichuan, China, September 2022, versión 1.1. [Online]. Available: https://resource.heltec.cn/download/CubeCell/HTCC-AB01_V2/HTCC-AB01_V2(Rev1.1).pdf

[12] H. Automation. (2023) Ht-m02 edge lora gateway (v2). Accessed: 2025-10-21. [Online]. Available: https://heltec.org/project/ht-m02-v2/

[13] M. Alam. (2024) How to use ina226 dc current sensor with arduino. Accessed: August 15, 2025. [Online]. Available: https://how2electronics.com/how-to-use-ina226-dc-current-sensor-with-arduino/#google_vignette

[14] Analog Devices, "A Practical Guide to High-Speed Printed-Circuit-Board Layout," October 2002, application Note AN-244. [Online]. Available: https://www.analog.com/media/en/technical-documentation/application-notes/AN-244.pdf

[15] R. . Schwarz. (2025) R&S RTM3000 Oscilloscope. [En línea; accedido: 27-oct-2025]. [Online]. Available: https://www.rohde-schwarz.com/es/productos/test-y-medida/osciloscopios/rs-rtm3000-oscilloscope_63493-427459.html

[16] Qoitech. (2024) Qoitech user manual. Accessed: Feb. 5, 2025. [Online]. Available: https://docs.qoitech.com/user-manual

[17] C. F. Gerald and P. O. Wheatley, *Applied Numerical Analysis*, 5th ed. Boston, MA, USA: Addison-Wesley Longman Publishing Co., 1989.

[18] R. L. Burden and J. D. Faires, *Análisis Numérico*, 10th ed. México: Cengage Learning, 2017.

[19] LoRa Alliance, "LoRa Alliance Regional Parameters 1.0.2," LoRa Alliance, Inc., Fremont, CA, USA, Tech. Rep., 2020, accessed: Sep. 30, 2024. [Online]. Available: https://lora-alliance.org/wp-content/uploads/2020/11/RP_2-1.0.2.pdf

[20] Semtech Corporation, *SX1261/2 LoRa® Sub-GHz Radio Transceiver Datasheet*, Semtech Corporation, Camarillo, CA, USA, 2022, versión 1.1. [Online]. Available: https://www.semtech.com/products/wireless-rf/lora-connect/sx1262

[21] EBYTE, *E32-900M30S LoRa Wireless Module Datasheet*, Chengdu Ebyte Electronic Technology Co., Ltd., Chengdu, Sichuan, China, 2020, versión 1.0. [Online]. Available: https://www.cdebyte.com/pdf-down.aspx?id=1415

[22] Semtech Corporation. (2024) What is LoRa? Accessed: September 30, 2024. [Online]. Available: https://www.semtech.com/lora/what-is-lora

[23] K. Rose, S. Eldridge, and L. Chapin, "The internet of things: An overview," The Internet Society (ISOC), Tech. Rep., 2015, accessed via Academia.edu. [Online]. Available: https://www.academia.edu/28441059/The_Internet_of_Things_An_Overview_Understanding_the_Issues_and_Challenges_of_a_More_Connected_World

[24] L. Atzori, A. Iera, and G. Morabito, "The internet of things: A survey," *Computer Networks*, 2010.

[25] P. Gokhale, O. Bhat, and S. Bhat, "Introduction to iot," *International Advanced Research*, 2018, accessed via ResearchGate. [Online]. Available: https://www.researchgate.net/profile/Omkar-Bhat/publication/330114646_Introduction_to_IOT/links/5c2e31cf299bf12be3ab21eb/Introduction-to-IOT.pdf

[26] Semtech. (2020) A brief history of lora: Three inventors share their personal story at the things conference. Accessed: 2025-10-21. [Online]. Available: https://blog.semtech.com/a-brief-history-of-lora-three-inventors-share-their-personal-story-at-the-things-conference

[27] A. Maleki, H. H. Nguyen, E. Bedeer, and R. Barton, "A tutorial on chirp spread spectrum modulation for lorawan: Basics and key advances," *IEEE Open Journal of the Communications Society*, vol. 5, pp. 4578–4612, 2024.

[28] V. R. Licea, "Redes celulares 4g 5g e iot iot-lora," Material de clase, Facultad de Ingeniería, UNAM, 2024, presentación proporcionada por el Dr. Victor Rangel Licea para el desarrollo de la Tesis.

[29] J. Haxhibeqiri, E. De Poorter, I. Moerman, and J. Hoebeke, "A survey of lorawan for iot: From technology to application," *Sensors*, vol. 18, no. 11, 2018. [Online]. Available: https://www.mdpi.com/1424-8220/18/11/3995

[30] F. Adelantado, X. Vilajosana, P. Tuset-Peiro, B. Martinez, J. Melia-Segui, and T. Watteyne, "Understanding the limits of lorawan," *IEEE Communications Magazine*, vol. 55, no. 9, pp. 34–40, 2017.

[31] M. A. Moya Quimbita, "Evaluación de pasarela lora/lorawan en entornos urbanos," Ph.D. dissertation, Intituto Politecnico Nacional, Quito, Ecuador, 09 2018. [Online]. Available: https://www.researchgate.net/publication/357620788_Evaluacion_de_pasarela_LoRaLoRaWAN_en_entornos_urbanos

[32] The Things Network. (2025) Lora physical layer packet format. En línea; consultado el 2 de diciembre de 2025. [Online]. Available: https://www.thethingsnetwork.org/docs/lorawan/lora-phy-format/

[33] LoRa Alliance Technical Committee, "Lorawan 1.1 specification," LoRa Alliance, Tech. Rep., Oct 2017, version 1.1, Final release. [Online]. Available: https://lora-alliance.org/resource_hub/lorawan-1-1-specification/

[34] N. Kamaruidzaman and S. N. Rahmat, "Water monitoring system embedded with internet of things (iot) device: a review," in *IOP conference series: Earth and environmental science*, vol. 498, no. 1. IoP Publishing, 2020, p. 012068.

[35] C. Z. Zulkifli, S. Garfan, M. Talal, A. Alamoodi, A. Alamleh, I. Y. Ahmaro, S. Sulaiman, A. B. Ibrahim, B. Zaidan, A. R. Ismail *et al.*, "Iot-based water monitoring systems: a systematic review," *Water*, vol. 14, no. 22, p. 3621, 2022.

[36] H. Fuentes and D. Mauricio, "Smart water consumption measurement system for houses using iot and cloud computing," *Environmental Monitoring and Assessment*, vol. 192, no. 9, p. 602, 2020.

[37] L. Casals, B. Mir, R. Vidal, and C. Gomez, "Modeling the energy performance of lorawan," *Sensors*, vol. 17, no. 10, p. 2364, 2017, published: 16 October 2017. [Online]. Available: https://www.mdpi.com/1424-8220/17/10/2364

[38] M. R. Ghaderi and N. Amiri, "Lorawan sensor: Energy analysis and modeling," *Wireless Networks*, vol. 30, no. 2, pp. 1013–1036, 2023, accepted: 3 October 2023; Published online: 30 October 2023. [Online]. Available: https://link-springer-com.pbidi.unam.mx:2443/content/pdf/10.1007/s11276-023-03542-y?pdf=openurl

[39] R. S. Benatti, C. P. de Souza, and O. Baiocchi, "An optimization method based on lora parameters for energy consumption reduction," in *2021 5th International Symposium on Instrumentation Systems, Circuits and Transducers (INSCIT)*. IEEE, 2021, pp. 1–6.

[40] L. J. O. Acevedo, "Despliegue de una red iot lora para potenciar ciudades inteligentes y Áreas rurales en un entorno universitario," Maestría en Telecomunicaciones, Universidad Nacional Autónoma de México (UNAM), Posgrado en Ingeniería Eléctrica, Ciudad de México, México, 2024, director de tesis: Dr. Víctor Rangel Licea.

[41] V. Novák, P. Ambruz, E. Kánská, M. Stočes, J. Vaněk, J. Veselý, and K. Sylvar, "Predictive battery life modeling for lorawan sensors using real-world deployment data," *AGRIS on-line*

Papers in Economics and Informatics, vol. 17, no. 3, pp. 53–62, September 2025. [Online]. Available: https://research-ebsco-com.pbidi.unam.mx:2443/c/df24kt/viewer/pdf/epz2p3z5xf

[42] C. S. B. Collinao, "Desarrollo de un sistema de transmisión de datos a larga distancia utilizando tecnología lora para aplicaciones de internet de las cosas (iot)," Memoria de Título, Universidad de Concepción, Facultad de Ingeniería, Concepción, Chile, junio 2025, profesor guía: Sergio K. Sobarzo G., Ph.D. [Online]. Available: https://repositorio.udec.cl/server/api/core/bitstreams/d3d50aa8-ac90-4f61-8207-b6222894f244/content#page74

[43] EE Power, "Pull-up resistor and pull-down resistor applications," https://eepower.com/resistor-guide/resistor-applications/pull-up-resistor-pull-down-resistor/, 2025, [Accedido: 23-Oct-2024].

[44] W. Waldmann, "Ina226_we: Arduino library for ti ina226 current and power monitor," https://github.com/wollewald/INA226_WE, 2024, accessed: 2025-06-15.

[45] G. Berthou, K. Marquet, T. Risset, and G. Salagnac, "Accurate power consumption evaluation for peripherals in ultra low-power embedded systems," in *2020 Global Internet of Things Summit (GIoTS)*, 2020, pp. 1–6.

[46] Semtech Corporation, "AN1200.22 LoRa Modulation Basics," Semtech Corporation, Tech. Rep., 2016, accessed: Jul. 9, 2025. [Online]. Available: https://www.frugalprototype.com/wp-content/uploads/2016/08/an1200.22.pdf

[47] R. S. Benatti, C. P. de Souza, and O. Baiocchi, "An optimization method based on lora parameters for energy consumption reduction," in *Proc. 5th Int. Symp. on Instrumentation Systems, Circuits and Transducers (INSCIT)*, Campinas, Brazil, Aug. 2021, accessed: Jan. 12, 2024. [Online]. Available: https://doi.org/10.1109/inscit49950.2021.9557241

[48] S. C. Chapra and R. P. Canale, *Métodos Numéricos para Ingenieros*, 5th ed. McGraw-Hill, 2006.

[49] Cadence PCB Solutions, "What is signal to noise ratio and how to calculate it?" Blog post, Jan. 2020, accessed: Oct. 12, 2024. [Online]. Available: https://resources.pcb.cadence.com/blog/2020-what-is-signal-to-noise-ratio-and-how-to-calculate-it

[50] The Things Network. (2023) Rssi and snr. Accessed: Oct. 12, 2024. [Online]. Available: https://www.thethingsnetwork.org/docs/lorawan/rssi-and-snr/

[51] Tektelic Communications. (2023) Understanding gateway antenna gain: What you need to know. Accedido: 4 de noviembre de 2025. [Online]. Available: https://tektelic.com/expertise/gateway-antenna-gain/