# PROYECTO FIN DE GRADO

**TÍTULO**: Sensorización en Smart Cities usando comunicación LoRa

**AUTOR/A:** María Izuzquiza Fernández

**TITULACIÓN**: Electrónica de Comunicaciones

**DIRECTOR/A**: José Carlos Calvo Tudela

**TUTOR/A**: Julián Nieto Valhondo

**DEPARTAMENTO**: Telemática y Electrónica

**VºBº TUTOR/A**

**Miembros del Tribunal Calificador:**

**PRESIDENTE/A**: Raquel Águeda Maté

**TUTOR/A**: Julián Nieto Valhondo

**SECRETARIO/A**: Eduardo Barrera López de Turiso

**Fecha de lectura:**

**Calificación:**

**El Secretario/La Secretaria,**

# Agradecimientos

En primer lugar, quiero expresar mis agradecimientos a Nazaríes Intelligenia por brindarme la oportunidad de llevar a cabo este proyecto. También agradecer a Julián, que ha sido mi profesor y tutor durante este proyecto final, por su dedicación y orientación a lo largo de los años de estudio.

Además, quiero extender un profundo agradecimiento a mis compañeros que me han ayudado y apoyado a lo largo de cursos académicos.

Por último, pero no menos importante, agradecer a mis padres y familia por su apoyo constante, el cual ha sido fundamental para alcanzar mis metas durante la realización del Grado de Electrónica de Comunicaciones.

# Resumen

La importancia de las tecnologías para proyectos de lo que se conoce como Internet de las Cosas (IoT) ha tenido un crecimiento exponencial en las últimas décadas. Este proyecto titulado “Sensorización en Smart Cities usando comunicación LoRa”, busca ampliar el conocimiento del uso de esta tecnología que hoy en día sigue creciendo y desarrollándose.

El propósito general de este proyecto es diseñar e implementar una red de sensores que utilice tecnología de comunicación LoRa, capaz de recopilar y transmitir datos obtenidos del entorno. Tecnológicamente, el proyecto supone un desafío al tener que integrar tanto *hardware* como *software* para crear una red robusta y eficiente. Desde el punto de vista medioambiental, no solo se busca una solución eficiente energéticamente, sino que busca aportar un beneficio al planeta pudiendo utilizar esta red en aplicaciones beneficiosas para el control ambiental en zonas metropolitanas.

El proyecto se desarrolla como una prueba de desarrollo de la tecnología LoRa por lo que se considera el uso de componentes de bajo coste que puedan cumplir con los requisitos mínimos para un correcto diseño de la red.

Durante la realización de este se pasan por distintas fases de aprendizaje y desarrollo. Desde una primera etapa de búsqueda y adquisición de conocimientos, pasando por el desarrollo del *hardware* y *software* necesario, hasta las pruebas finales en un entorno real. Y, finalmente, concluir en un primer prototipo de la red, eficiente y con gran capacidad de crecimiento para posteriores proyectos.

**Palabras clave:** IoT, Smart Cities, LoRa, sensor, red.

# Abstract

The importance of technologies for projects known as the Internet of Things (IoT) has seen exponential growth in recent decades. This project aims to expand the knowledge of the use of this technology, which continues to grow and develop today.

The general purpose of this project is to design and implement a sensor network using LoRa communication technology. Which is capable of collecting and transmitting data obtained from sensors. Technologically, the project raises a challenge by integrating both hardware and software parts. This is to achieve to a robust and efficient network. From an environmental perspective, the goal consist on achieving an energy-efficient solution. It also provides an environmental control benefit in metropolitan areas for environmental.

The project development consist on testing LoRa technology, so low-cost components that can meet the minimum requirements for proper network design are considered.

Throughout the execution of the project, various phases of learning and development are undertaken. These range from state of the art, development of necessary hardware and software, to final tests in real environment. All phases culminate in an initial prototype of the network, with great potential for future projects.

**Keywords:** IoT, Smart Cities, LoRa, sensors, network.

# Índice de contenidos

# Índice de figuras

# Índice de tablas

# Lista de acrónimos

*Tabla 1: Lista de acrónimos*

| Acrónimo | Descripción |
|---|---|
| **IoT** | Internet Of Things |
| **LPWAN** | Low Power Wide Area Network |
| **LoRa** | Long Range |
| **TCP** | Transmission Control Protocol |
| **IP** | Internet Protocol |
| **TIC** | Tecnologías de la Información y la Comunicación |
| **WSN** | Wireless Sensor Network |
| **FFG** | Full Function Device |
| **RFD** | Reduced Function Device |
| **ISM** | Industrial, Scientific, Medical |
| **LTE** | Long Term Evolution |
| **GSM** | Global System for Mobile Communication |
| **FOTA** | Firmware Over-The-Air |
| **PAN** | Personal Area Network |
| **WAN** | Wide Area Network |
| **AODV** | Ad Hoc On demand Distance Vector |
| **OLSR** | Optimized Link State Routing |
| **RREQ** | Route Request |
| **RREP** | Route Reply |
| **RERR** | Route Error |
| **TC** | Topology Control |
| **MPR** | Multipoint Relay |
| **WWAN** | Wireless Wide Area Network |
| **WMAN** | Wireless Metropolitan Area Network |
| **WLAN** | Wireless Local Area Network |
| **WPAN** | Wireless Personal Area Network |
| **RFID** | Radio Frequency Identification |

| | |
|---|---|
| **LiPo** | Lithium Polymer |
| **PFG** | Proyecto Fin de Grado |
| **ID** | Identification |
| **LCD** | Liquid Crystal Display |
| **WiFi** | Wireless Fidelity |
| **GPIO** | General Purpose Input/Output |
| **HTTP** | Hypertext Transfer Protocol |
| **AWS** | Amazon Web Services |
| **PCB** | Printed Circuit Board |
| **ACK** | Acknowledgement |
| **RSSI** | Received Signal Strength Indicator |
| **ETSIST** | Escuela Técnica Superior de Ingeniería y Sistemas de Telecomunicación |
| **MQTT** | Message Queuing Telemetry Transport |

# 1. Introducción

En este primer capítulo se expone tanto la motivación, como los objetivos a lograr y la estructura que se ha llevado en la memoria.

## 1.1. Marco y motivación del proyecto

En la última década, la implementación de sistemas en el ámbito del Internet de las Cosas (IoT), término definido y analizado en profundidad en el apartado 2.1 de esta memoria, ha experimentado un crecimiento exponencial en la sociedad. De manera paralela, el desarrollo de nuevas tecnologías ha acompañado al Internet de las Cosas, contribuyendo a la mejora de la vida cotidiana. Un ejemplo es el uso de Redes de Sensores Inalámbricos en redes de área amplia y de baja potencia, conocidas en inglés como *Low Power Wide Area Network* (LPWAN), cuyo análisis se presenta en los apartados 2.4 y 2.5. En concreto, se evalúa el protocolo de comunicaciones LoRa de entre todos los protocolos disponibles en el mercado actual.

La motivación principal de este proyecto radica en la necesidad de investigar e implementar una Red de Sensores Inalámbricos, además de analizar todo lo que implica el proceso de desarrollo de un sistema funcional y eficiente que pueda llegar a ser utilizado en el futuro en nuestra sociedad. De igual forma, se busca comprender el funcionamiento de los algoritmos de enrutamiento centrados en las redes inalámbricas para desarrollar un algoritmo de enrutamiento capaz de proporcionar independencia ante fallos. Todo ello haciendo uso de dispositivos con módulos LoRa, que proviene del inglés *Long Range* traducido como Largo Alcance , permitiendo la comunicación entre otros.

## 1.2. Objetivos técnicos y académicos

El objetivo principal se ha fijado en la implementación de una red de sensores para el control de una ciudad inteligente haciendo uso de una red mallada, conocida en inglés como *Mesh Network* y preparada para utilizar el protocolo de comunicaciones LoRa. En la Figura 1 se muestra un esquema del sistema que se propone desarrollar.

Esta red mallada permite una comunicación robusta y autoorganizada, habilitada mediante la tecnología LoRa, reconocida por su eficiencia en términos de consumo de energía y su capacidad para abarcar distancias considerables.

Adicionalmente, se han establecido dos objetivos secundarios para la ejecución del proyecto. Uno de ellos es la implementación de un sistema de enrutamiento confiable que garantice la transmisión exitosa de los datos recopilados por los sensores hacia el nodo central de la red *Mesh*. Este enrutamiento es esencial para la funcionalidad de la red. Por otro lado, el segundo objetivo se enfoca en el envío de datos al servidor, lo que permite el almacenamiento y análisis de los datos generados por los sensores. El servidor lo

proporciona la empresa Nazaríes Intelligenia y dispone de una plataforma [1] robusta para la gestión de los datos que se manejan en la red.

Es relevante destacar que este proyecto se concibe como un primer prototipo, y no busca la obtención de un producto final completamente optimizado. Por lo tanto, en caso de alcanzar con éxito los objetivos previamente mencionados, se llevará a cabo un análisis detallado del consumo energético en la red.

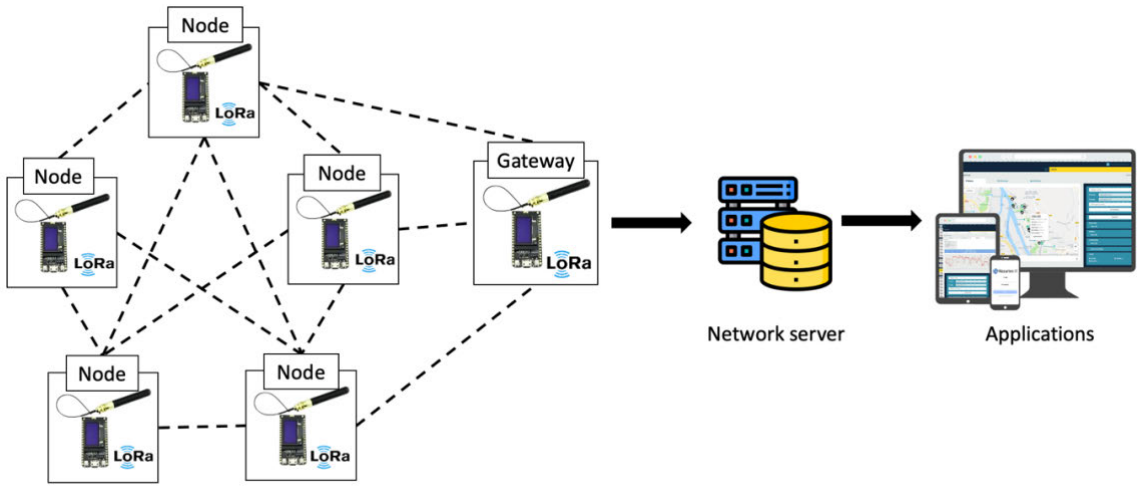


*Figura 1: Esquema del proyecto a desarrollar.*

## 1.3. Estructura de la memoria

En la siguiente sección se hace una breve explicación de la estructura de la memoria.

- **Capítulo 1: Introducción.**

Se proporciona una descripción resumida de la motivación y los objetivos que se llevan a cabo en el desarrollo del proyecto, así como la estructura de la memoria.

- **Capítulo 2: Marco tecnológico.**

Se estudian en profundidad los aspectos teóricos necesarios para la implantación de un sistema basado en el Internet de las Cosas con comunicación LoRa.

- **Capítulo 3: Especificaciones y restricciones del diseño.**

Se presentan las especificaciones y restricciones del sistema que deben considerarse para el desarrollo, con relación directa a los objetivos establecidos.

- **Capítulo 4: Descripción de la solución propuesta.**

Se explica de manera precisa los elementos que intervienen en el desarrollo del proyecto, tanto hardware como de software.

- **Capítulo 5: Resultados**

Se documentan las pruebas realizadas durante el desarrollo del sistema y los resultados obtenidos en cada una de ellas.

- **Capítulo 6: Presupuesto**

Se presenta el presupuesto invertido en la realización del proyecto.

- **Capítulo 7: Impacto del proyecto**

Se realiza un estudio del impacto del proyecto en la vida real.

- **Capítulo 8: Conclusiones y propuesta de mejoras.**

Se exponen las conclusiones alcanzadas tras la realización del proyecto, además de la propuesta de mejoras futuras.

- **Anexo I:**

Se incluyen fragmentos del código de la configuración de los dispositivos.

- **Anexo II: Diagramas de Flujo**

Se incluyen las figuras de los diagramas de flujo.

- **Anexo III: Código prueba punto a punto**

Se incluyen fragmentos del código implementado para la conexión punto a punto de una red simple.

- **Anexo IV: Código prueba de algoritmo de enrutamiento**

Se incluyen fragmentos del código implementado para la prueba de un algoritmo de enrutamiento en una red simple.

# 2. Marco tecnológico

En este capítulo, se aborda la relevancia del Internet de las Cosas en el contexto actual, así como el innovador concepto de *Smart Cities*.

Exploraremos el origen de la idea de una red de sensores inalámbrica, examinando su historia y características. Asimismo, se llevará a cabo un análisis de los algoritmos de enrutamiento[1].

A continuación, se discutirán las diversas tecnologías inalámbricas disponibles en la actualidad, con especial atención en las redes de área amplia de baja potencia (LPWAN, por sus siglas en inglés, *Low Power Wide Area Network*). Finalmente, se procederá a examinar los dispositivos que se encuentran vigentes del mercado.

## 2.1. Internet de las Cosas

En la actualidad, no disponemos de una definición estandarizada del término IoT. En cambio, existen estudios íntegramente dedicados a definir este concepto. Según el artículo "*Internet of Things: A Definition & Taxonomy*" [2] se define IoT como "grupo de infraestructuras interconectadas de objetos conectados que permiten su gestión, minería de datos y el acceso a los datos que generan." Otros autores lo definen como: "El Internet de las cosas describe objetos físicos (o grupos de estos) con sensores, capacidad de procesamiento, software y otras que se conectan e intercambian datos con otros dispositivos y sistemas a través de internet u otras redes de comunicación."[3].

Se llama IoT a la red de objetos físicos conectados y a la tecnología que ayuda a la comunicación entre ellos y, a su vez, con la nube. Teniendo como propósito la recopilación de datos y su distribución. Los beneficios de esta red varían según el contexto en el que se aplique, pero en su mayoría persiguen un objetivo común, el proporcionar datos en tiempo real a los usuarios.

Por otro lado, representa una tecnología de amplio espectro, lo cual indica su capacidad para manejar un volumen significativo de información. Dicha información se transmite a través de la red, y dada la complejidad de sus datos, es necesario tener en cuenta un amplio rango de protocolos. Esto incluye la exploración de alternativas a los métodos de comunicación tradicionales, como puede ser el Protocolo de Control de Transmisión/Protocolo de Internet (TCP/IP, por sus siglas en inglés, *Transmission Control Protocol/Internet Protocol*). A continuación, se explorarán las diversas arquitecturas utilizadas en el Internet de las Cosas (IoT).

[1] Enrutamiento: proceso de selección de rutas.

### 2.1.1. Arquitecturas IoT

Los estudios no presentan una arquitectura única para los proyectos de IoT [4], sin embargo, los objetivos de esta tecnología son similares en la gran mayoría de los casos. Nos referimos a que una red IoT debe ser escalable, resistente a fallos y flexible ante los cambios.

- Escalabilidad: capacidad de la red para crecer y adaptarse a medida que se agregan nuevos dispositivos, y como consecuencia que tiene esto, al crecimiento del manejo de datos. Permitiendo su expansión sin comprometer el rendimiento.
- Resistente a fallos: capacidad de mantener el funcionamiento de la red incluso en situaciones de pérdidas de conexión en dispositivos individuales o de componentes de la red.
- Flexibilidad: capacidad de adaptación en diferentes contextos y condiciones cambiantes, tanto en términos de requisitos de aplicación como de entornos operativos.

Aunque no existen dos proyectos de IoT iguales, las capas principales se han mantenido constantes a lo largo de los años [2][5]. La Figura 2 muestra un ejemplo de una arquitectura de tres capas, la cual es el modelo dominante de las aplicaciones IoT.

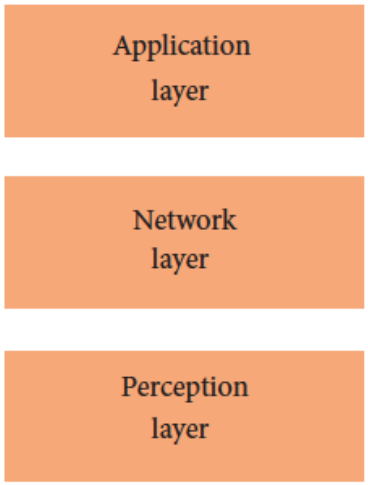


*Figura 2: Arquitectura IoT de tres capas. [6]*

En la primera capa (capa de percepción) se encuentran los dispositivos responsables de recolectar los datos del entorno. Estos pueden comunicarse haciendo uso de redes cableadas como pueden ser Ethernet o fibra óptica; pero para nuestro estudio nos centraremos en las conexiones inalámbricas (*Wireless links*), que se analizarán en detalle en el apartado 2.4.

La segunda capa (capa de red) es la encargada de transportar el tráfico de dispositivos hacia la nube o los servidores.

Y, por último, la tercera capa (capa de aplicación) es responsable de proporcionar servicios específicos de la aplicación al usuario. Hay diversas aplicaciones en las que se puede implementar IoT, como puede ser para controlar un ecosistema doméstico.

Con el desarrollo de esta tecnología, una arquitectura de tres capas puede llegar a tener un alcance limitado, por lo que se han ido desarrollando otras capas adicionales o modificando las que previamente hemos explicado. Una arquitectura popular que ha surgido como mejora de la arquitectura de tres capas es la arquitectura de cinco capas, como se muestra en la Figura 3.

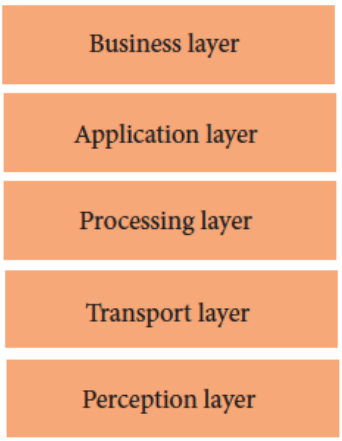


*Figura 3: Arquitectura de cinco capas. [6]*

Teniendo en este caso una capa de transporte, encargada de conectar la capa de percepción y la capa de procesamiento, definiendo el modo de transmisión de los datos obtenidos por los sensores. Esta capa remplaza a la que se ha definido antes como capa de red. Se incluye una capa de procesamiento conocida como capa de Middleware, encargada de almacenar, analizar y procesar los datos procedentes de la capa anterior. Finalmente, se añade una capa de negocio en la cual se realiza la toma de decisiones de los datos encontrados en la capa de aplicación. Esta capa varía según las partes interesadas.

## 2.2. Ciudades Inteligentes o *Smart Cities*[2]

En las últimas décadas, el avance tecnológico ha desencadenado una transformación sin precedentes en nuestra interacción con el entorno que nos rodea. Un ejemplo de este progreso tecnológico en esta era digital es la fusión de dos conceptos fundamentales como es el Internet de las Cosas (IoT) y el surgimiento de las ciudades inteligentes. Una vez dada la explicación del término IoT explicaremos como se define y las características que tiene una ciudad inteligente.

Se entiende por Ciudad Inteligente o también conocido en inglés como *Smart City,* al entorno urbano, seguro y mayormente eficiente, basado en nuevas tecnologías. Tecnologías que estimulan el aumento de la economía y mejorando el nivel de vida de la población. Algunas interpretaciones destacan la integración de las TIC como factor fundamental para el desarrollo de una ciudad inteligente. A pesar de que los científicos tienen opiniones variadas

[2] *Smart Cities*: es la traducción al inglés de Ciudades Inteligentes.

del modo de desarrollo, todos coinciden en que estos proyectos deben basarse en la interacción constante entre diversas estructuras. Se pueden identificar los siguientes bloques, mostrados en la Figura 4.

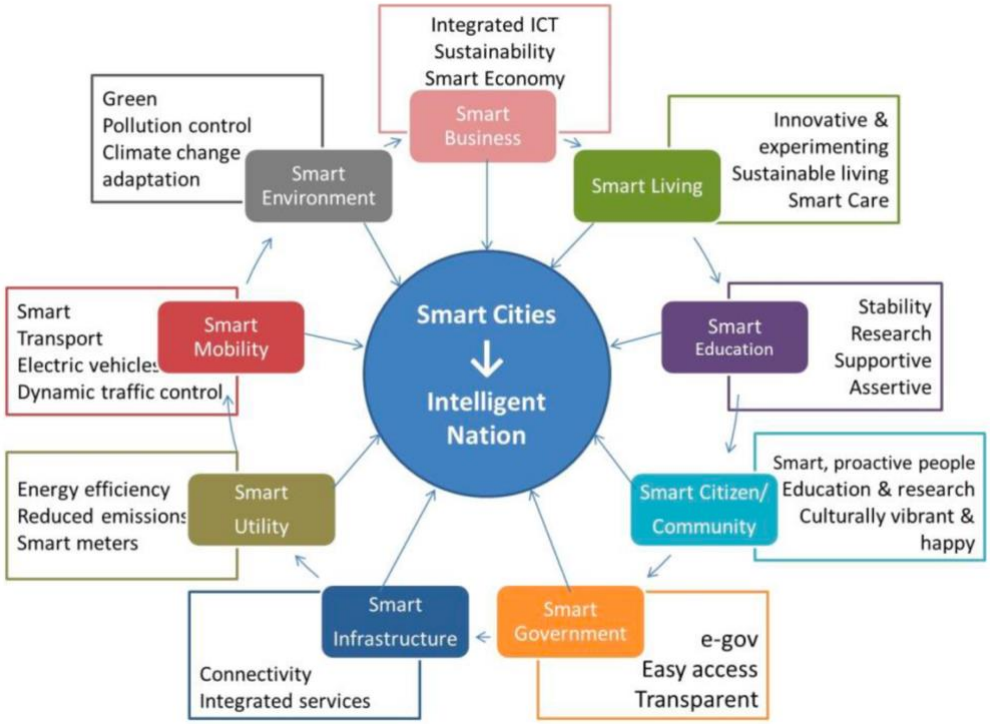


*Figura 4: Modelo de interacción de una Ciudad Inteligente. [7]*

## 2.3. Internet de las Cosas para Ciudades Inteligentes

Una vez definido qué es una Ciudad Inteligente hay que destacar que no se puede hablar de este concepto sin mencionar el IoT. En el marco de una *Smart City*, el IoT es la tecnología que realiza la interacción entre las distintas estructuras que la componen.

Como se ha mencionado anteriormente, el IoT promete la digitalización de todos los aspectos de nuestra vida, por lo que este proceso de digitalización en una Ciudad Inteligente implica la expansión de sensores en todas las áreas que conforman el funcionamiento de una ciudad. Esto supone un reto para los desarrolladores debido al amplio ámbito de aplicación y su posterior despliegue. En la Figura 5 se muestran los diversos desafíos que surgen al implementar sistemas de IoT en ciudades inteligentes, como la seguridad y privacidad, los sensores inteligentes, las redes y el análisis de grandes volúmenes de datos [8].

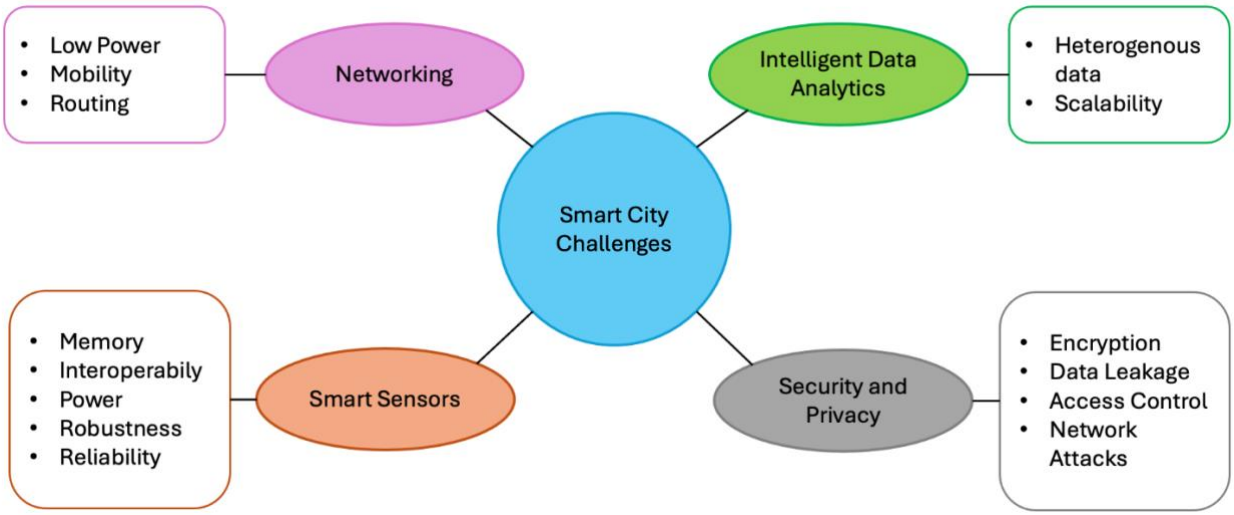


*Figura 5: Retos del IoT en Ciudades Inteligentes [8].*

- Seguridad y privacidad.

La seguridad y la privacidad son aspectos de suma importancia en el contexto en el que nos encontramos. Dado que estas ciudades involucran la conexión en línea de infraestructuras vitales, cualquier fallo en los servicios humanos podría ocasionar poner en riesgo a las personas. En un entorno actual, donde la ciberdelincuencia utiliza tácticas cada vez más sofisticadas, las ciudades inteligentes están expuestas a ataques malintencionados.

Por otro lado, la participación ciudadana es fundamental para el éxito de estos proyectos. Sin embargo, la recopilación de datos sobre la actividad cotidiana de las personas plantea preocupaciones sobre la exposición de la privacidad de los ciudadanos ante terceros sin autorización. Además, cabe la posibilidad que las empresas o entidades de la red IoT puedan utilizar esta información, sin consentimiento, con fines publicitarios.

Para abordar estos desafíos se necesitan soluciones que aseguren el anonimato en la recolección de datos, y que a su vez se mantenga la integridad del contexto necesario para una toma de decisiones adecuada.

- Sensores inteligentes.

Los dispositivos de sensores inteligentes desempeñan un papel fundamental en la recopilación de datos en entornos urbanos inteligentes. Estos dispositivos son fabricados por una amplia gama de proveedores, cada uno siguiendo distintos estándares y protocolos en cuanto a detección, medición, formatos de datos y conectividad. Sin embargo, para lograr el despliegue efectivo de una ciudad inteligente es crucial que estos dispositivos puedan intercambiar datos entre sí, coordinar tareas y procesar información para obtener conclusiones significativas.

Una posible solución a este desafío es el desarrollo y la implementación de protocolos y formatos de datos abiertos. Esto permitiría a los fabricantes crear dispositivos compatibles que puedan comunicarse sin problemas entre sí, lo que facilitaría aún más la expansión de los sistemas de IoT en entornos urbanos.

Un desafío adicional para los sensores inteligentes es garantizar su fiabilidad y robustez, aspectos cruciales para la integridad del sistema IoT. Dado que el IoT constituye el núcleo de las ciudades inteligentes del futuro y es fundamental para su funcionamiento eficiente, es imprescindible que ofrezca una experiencia de usuario fluida. Esto implica que las solicitudes de servicio de los usuarios deben recibir respuestas precisas y oportunas, asegurando una calidad de servicio óptima para todos los habitantes de la ciudad inteligente.

Además, se necesitan avances en tecnologías de almacenamiento y memoria para gestionar los datos recopilados por estos sensores. El almacenamiento eficiente de grandes volúmenes de datos implica el desarrollo de algoritmos de compresión y esquemas de bases de datos avanzados. Para abordar los desafíos energéticos, se requiere el desarrollo de nuevas tecnologías de baterías y posiblemente la integración de mecanismos de captación de energía en los dispositivos para garantizar la prestación de servicios sostenibles a largo plazo.

- Redes.

El Internet de las Cosas (IoT) depende en gran medida de la capacidad de los dispositivos sensores y otros dispositivos para intercambiar información entre sí y con la nube. En el contexto del desarrollo de nuevas aplicaciones para ciudades inteligentes, el desafío de proporcionar redes que mantengan conectados estos dispositivos es considerable. Las infraestructuras de red actuales no están adecuadamente adaptadas para ofrecer servicios de red a los componentes de las ciudades inteligentes. Muchos de estos dispositivos tienen requisitos específicos de movilidad y ancho de banda que deben cumplirse para garantizar una calidad de servicio aceptable. Para abordar este problema se han propuesto diversos enfoques, incluyendo la definición de puntos de acceso y redes locales. Además, es crucial trabajar en el desarrollo de protocolos de enrutamiento eficientes y dinámicos que puedan satisfacer las demandas del IoT, especialmente aquellos capaces de adaptarse a dispositivos tanto estacionarios como en movimiento, una capacidad que muchos protocolos actuales no proporcionan adecuadamente.

- Análisis de datos.

La creciente interconexión de dispositivos en el Internet de las Cosas (IoT) plantea desafíos significativos para el análisis de datos en ciudades inteligentes. Se requiere el desarrollo de algoritmos avanzados que puedan manejar la diversidad de datos generados en este entorno, tanto estructurados como no estructurados. La aplicación de técnicas de fusión de datos efectivas es esencial para integrar y extraer información significativa de estos conjuntos de datos.

Otra consideración es la importancia de garantizar que los algoritmos sean adaptables y escalables a diferentes contextos y conjuntos de datos. La evolución continua de los datos presenta un desafío adicional, ya que las propiedades pueden cambiar con el tiempo, lo que requiere enfoques como el aprendizaje incremental.

## 2.4. Red de sensores Inalámbricos

Una Red de Sensores Inalámbricos (WSN, por sus siglas en inglés, *Wireless Sensor Network*) se define como una red de dispositivos distribuidos capaces de monitorizar diversas condiciones, como temperatura, humedad y presión entre otras.

Estos dispositivos, también conocidos como nodos o motas[3], son unidades independientes de tamaño reducido que destacan por su autonomía. En la Figura 6 se presenta un esquema de un dispositivo autónomo de una Red de Sensores Inalámbricos [9].

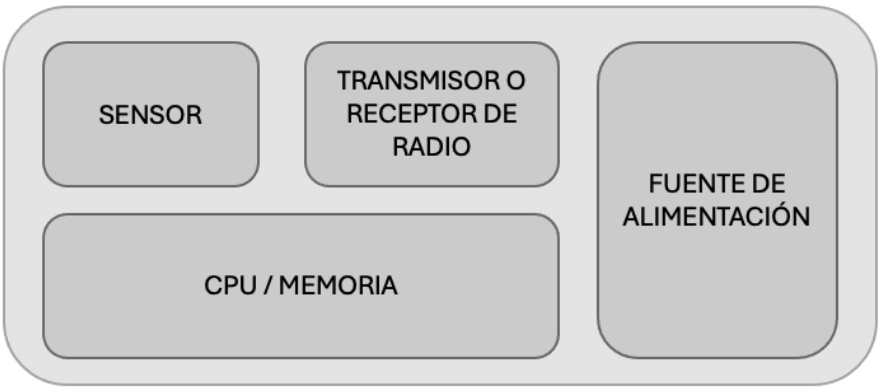


*Figura 6: Dispositivo autónomo, nodo o mota [9].*

Las redes de sensores se despliegan en áreas específicas y pueden consistir en nodos estacionarios o móviles. En el caso de una red de nodos móviles, se habla de una red *ad hoc*, la cual es capaz de realizar enrutamiento entre sí. Su configuración es autónoma y se adapta a una topología física arbitraria, sujeta a cambios frecuentes debido a movimientos, ingresos, salidas y fallos de los nodos participantes.

Los nodos tienen diversas funciones típicas como pueden ser:

- Medición de parámetros ambientales como temperatura, humedad, presión o detección de humos, y luminosidad entre otras.
- Detección de eventos como movimiento, vibración o presencia.
- Estimación de parámetros de velocidad y dirección.

[3] Mota: nombre que se asigna a los nodos que están equipados con sensores. El término proviene del inglés ***mote***, que en español puede ser traducido como **mota**.

Un ejemplo real de un despliegue de nodos IoT lo encontramos España con un proyecto llevado a cabo por Telefónica, Ferrovial Servicios y Tellink Sistemas de Comunicación [10]. Este proyecto se centra en la digitalización del alumbrado público para la mejora de la gestión y mantenimiento del alumbrado, contribuyendo al ahorro energético. Hace uso de la tecnología llamada NB-IoT explicada en detalle en la sección 2.5.1.1.

### 2.4.1. Características

En este estudio exploraremos las características fundamentales a tener en cuenta en el diseño y desarrollo de redes de sensores. Las redes de sensores poseen una variedad de características que deben ser consideradas en función de las necesidades específicas de cada proyecto. A continuación, nos centraremos en las características más comunes y relevantes para la realización de proyectos de este tipo.

- **Topología y mantenimiento:** la topología de la red y los procedimientos de mantenimiento son aspectos críticos que considerar en el diseño de redes de sensores. La selección de la topología adecuada puede influir en la eficiencia de la comunicación entre los nodos, mientras que las estrategias de mantenimiento pueden garantizar el rendimiento y la durabilidad de la red a lo largo del tiempo.
- **Limitaciones energéticas:** dada la naturaleza inalámbrica y, en muchos casos, autónoma de las redes de sensores, es crucial tener en cuenta las limitaciones energéticas de los nodos. El diseño de estrategias eficientes de gestión de energía puede prolongar la vida útil de la red y garantizar su operatividad continua.
- ***Hardware* y *software* específico:** la selección de hardware y software específico para las aplicaciones de la red de sensores es fundamental para garantizar un rendimiento óptimo. Los componentes deben ser capaces de soportar las condiciones ambientales y las demandas operativas del proyecto en cuestión.
- **Enrutamiento dinámico:** el enrutamiento dinámico permite adaptar la ruta de comunicación entre los nodos de la red en función de las condiciones cambiantes del entorno. Esta capacidad es especialmente importante en redes de sensores desplegadas en entornos dinámicos o móviles.
- **Restricciones temporales:** Algunas aplicaciones de redes de sensores pueden tener restricciones temporales estrictas, donde la recolección y transmisión de datos deben realizarse dentro de ciertos límites de tiempo. Es crucial diseñar la red para garantizar que estas restricciones se cumplan de manera eficiente y fiable.

Además de estas características fundamentales, existen otros aspectos que pueden ser relevantes dependiendo del contexto específico del proyecto. Entre ellos se encuentran la facilidad de despliegue, el bajo consumo de recursos, el coste reducido, el tamaño compacto y la posibilidad de no necesitar un mantenimiento a largo plazo.

### 2.4.2. Arquitectura

La implementación de una red de sensores inalámbricos requiere la integración de tecnologías que provienen de tres áreas de investigación principales: detección, comunicación y computación. Estas áreas abarcan tanto *hardware* como *software*, así como algoritmos.

Para comprender mejor el funcionamiento de la red, en la Figura 7 se muestra su estructura. En ella se pueden observar los sensores que recopilan los datos del entorno. Además, encontramos dispositivos conocidos como *mote*, distribuidos a lo largo de la red. Estos nodos tienen la capacidad tanto de recolectar datos como de enrutarlos al *gateway*[4].

El *gateway*, a su vez, se comunica con el servidor a través de Internet. El servidor almacena y presenta los datos recibidos. El sistema completo permite la recopilación y gestión eficiente de la información generada por los sensores distribuidos en la red.

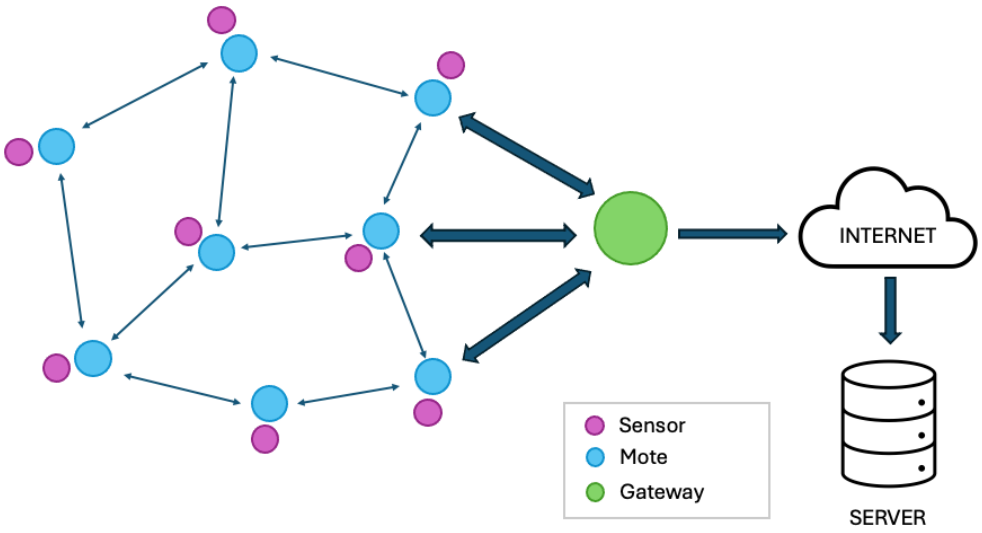


*Figura 7: Estructura de una red de sensores.*

Es importante destacar que el diseño de una red, tal como se ha descrito anteriormente, puede estar fuertemente condicionado por una serie de factores clave, entre los cuales se incluyen:

- **Tolerancia a fallos:** En caso de daño físico de los nodos o una falla debido a la falta de alimentación o por interferencias del medio ambiente, es fundamental que el sistema esté preparado para que estos eventos no comprometan la integridad de la red.

[4] *Gateway*: se refiere al nodo recolector en una red. El término proviene del inglés **"gateway"** que se traduce en español como **"puerta de enlace"**. También puede ser conocido como **"sink node"**

- **Escalabilidad:** El diseño debe ser capaz de soportar un número elevado de nodos, del orden de centenares, miles e incluso, en alguna ocasión, de millones.
- **Consumo energético:** Al tratarse de dispositivos cuyo objetivo se centra en que tengan un tamaño reducido y que sean inalámbricos, es importante tener en cuenta que disponen de una fuente de energía limitada.

### 2.4.3. Topologías

Se define como topología de la red al mapa físico o lógico de una red para intercambiar datos [11]. Como se ha comentado antes, el tipo de topología de red desplegada es crucial para una correcta implementación de la red. Un despliegue con un gran número de nodos requiere una topología adecuada y un mantenimiento constante. Existen varios tipos de dispositivos en una red de sensores [12].

- FFD (*Full-Function Device*): dispositivos capaces de funcionar en cualquier topología.
- RFD (*Reduced Function Device*): dispositivos que únicamente pueden ser miembros de una red con topología estrella.
- PAN (*Personal Area Network*): coordinador principal de la red.

Los FFD tiene todas las funcionalidades y puede desempeñar el papel de coordinador (PAN) de la red, tal y como lo hemos nombrado antes este sería la puerta de enlace o *gateway*. Mientras que los RFD se utilizan como dispositivos con funciones limitadas como, por ejemplo: tienen menos memoria, recursos informáticos y, además, consumen menos energía.

Existen varios tipos de topologías de red, cada una con sus propias características y aplicaciones específicas. Algunas de las topologías más comunes son la topología en estrella, en malla y en árbol. Cada una de estas topologías tiene ventajas y desventajas particulares en términos de escalabilidad, redundancia, facilidad de implementación y tolerancia a fallos. La elección de la topología adecuada depende de los requisitos específicos de la red y del entorno en el que se despliegue.

En los siguientes apartados, exploraremos las características distintivas de cada una de estas topologías y discutiremos sus aplicaciones más comunes [13].

#### 2.4.3.1. Topología en Estrella

En la topología en estrella, representada en la Figura 8, los nodos se encuentran conectados a un único nodo receptor o nodo central. Este nodo central tiene la función de coordinador de la red (PAN). Este nodo receptor es el encargado de llevar a cabo la toma de decisiones y el enrutamiento de los datos.

Una de las ventajas clave de esta topología es la sencillez de la red y su facilidad de implementación. Además, debido a su estructura centralizada, la topología en estrella puede ofrecer cierto nivel de seguridad, ya que los datos pueden ser más fáciles de controlar y monitorear en comparación con otras topologías.

Sin embargo, es importante tener en cuenta que la topología en estrella también presenta algunas desventajas significativas. Por ejemplo, si el nodo central falla, toda la red se ve afectada, lo que puede crear un punto único de fallo y comprometer la fiabilidad de la red. Además, el mantenimiento y la reparación del nodo central pueden ser costoso y complicado.

Otra limitación importante de la topología en estrella es su bajo nivel de escalabilidad. Agregar nuevos nodos a la red puede requerir cambios significativos en la infraestructura existente, lo que puede resultar costoso y poco práctico en entornos en crecimiento.

En cuanto a las aplicaciones y casos de uso, la topología en estrella puede ser especialmente adecuada para redes donde se requiere un control centralizado y una fácil administración de los datos. Sin embargo, puede no ser la mejor opción para entornos donde la redundancia y la resistencia a fallos son críticas.


*Figura 8: Topología en Estrella.*

#### 2.4.3.2. Topología en Malla

La topología en malla, también conocida como *Mesh Network,* es una estructura de red en la que cada nodo está conectado directamente a todos los demás nodos, formando una red completamente interconectada, como se puede observar en la Figura 9. Esto significa que cada nodo tiene múltiples rutas disponibles para comunicarse con cualquier otro nodo en la red.

Una de las principales ventajas de la topología en malla es su alta redundancia y robustez. Debido a la gran cantidad de conexiones cruzadas, la pérdida de un nodo o enlace no afecta significativamente la conectividad de la red, ya que todavía hay múltiples rutas disponibles para el tráfico de datos.

Sin embargo, esta redundancia también puede ser una desventaja en términos de costes y complejidad. La topología en malla puede requerir una gran cantidad de cables y *hardware* de red para mantener todas las conexiones, lo que puede ser costoso de implementar y mantener. Además, el enrutamiento de datos en una red en malla puede ser más complejo que en otras topologías, lo que puede requerir un mayor nivel de configuración y monitorización.

En términos de aplicaciones y casos de uso, la topología en malla es especialmente adecuada para entornos donde la confiabilidad y la tolerancia a fallos son críticas, como en sistemas de control industrial, redes militares o aplicaciones de misión crítica donde la conectividad ininterrumpida es esencial.

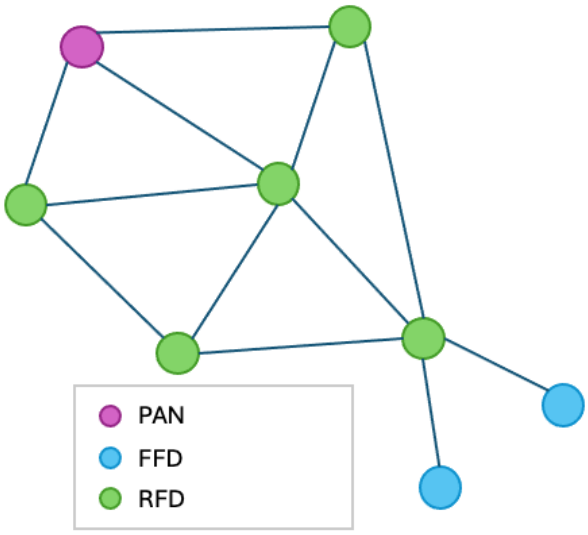


*Figura 9: Topología en Malla.*

#### 2.4.3.3. Topología en Árbol

La topología en árbol es una estructura de red en la que los nodos están organizados en una jerarquía de ramas que se extienden desde un nodo central. Cada nodo en la red se conecta a un nodo padre, excepto el nodo raíz que se encuentra en la cima de la jerarquía, véase Figura 10.

Una de las principales ventajas de la topología en árbol es su capacidad para manejar grandes cantidades de nodos de manera eficiente, ya que la carga de la red se distribuye a través de las ramas del árbol. Además, esta topología puede ofrecer un alto nivel de

redundancia, ya que múltiples caminos pueden estar disponibles entre cualquier par de nodos.

Sin embargo, la topología en árbol también tiene algunas limitaciones importantes. Por ejemplo, si el nodo raíz falla, toda la red se ve afectada, lo que puede resultar en una interrupción generalizada del servicio. Además, la adición o eliminación de nodos en la red puede requerir cambios significativos en la estructura del árbol, lo que puede ser complejo y costoso de manejar.

En términos de aplicaciones y casos de uso, la topología en árbol puede ser especialmente adecuada para redes de área amplia (WAN) donde se requiere una estructura jerárquica para manejar grandes volúmenes de tráfico de datos, como en empresas o proveedores de servicios de Internet.

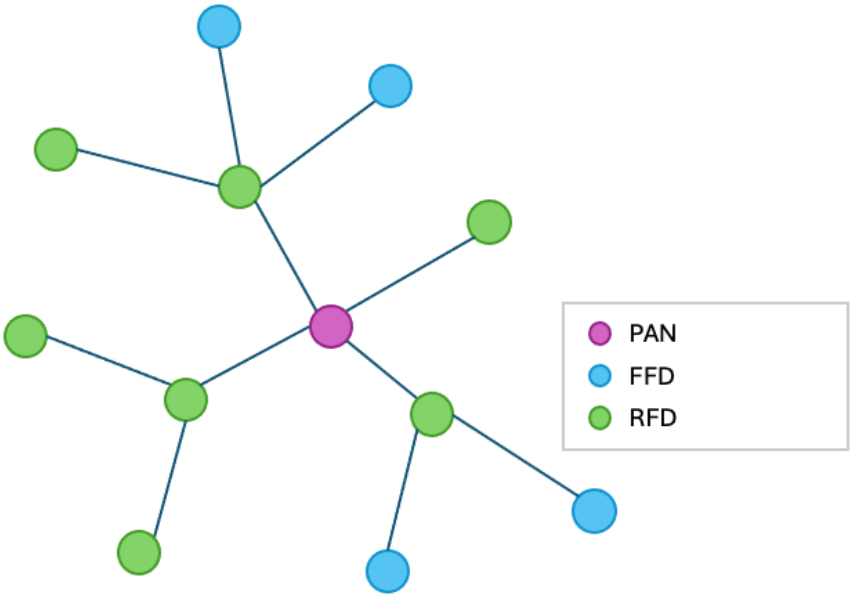


*Figura 10: Topología en Árbol.*

### 2.4.4. Protocolos de enrutamiento

El enrutamiento se refiere al proceso de definir rutas desde un punto de origen específico hacia un destino designado, permitiendo la utilización de nodos intermedios para alcanzar dicho destino final [14]. Los nodos no tienen conocimiento del tipo de topología que define la red, deben descubrirla y para conseguirlo se utilizan los protocolos de enrutamiento.

Estos protocolos se pueden clasificar en tres categorías: reactivos (baja demanda), proactivos (basados en tablas) e híbridos. Los protocolos reactivos establecen rutas únicamente cuando son requeridas para enviar datos entre un nodo origen y un nodo destino. Por otro lado, los protocolos proactivos establecen rutas antes de que sean necesarias, minimizando así los retrasos en la búsqueda de rutas. A menudo, estos protocolos se

denominan basados en tablas porque utilizan tablas de enrutamiento predefinidas. Sin embargo, los protocolos proactivos pueden generar tráfico pesado y sobrecarga de enrutamiento al construir tablas de enrutamiento extensas, lo que puede resultar en errores de enrutamiento y pérdida de paquetes.

Aunque se han mencionado tres categorías de protocolos de enrutamiento, existen más y dentro de cada una hay múltiples subtipos, como se puede observar en la Figura 11.

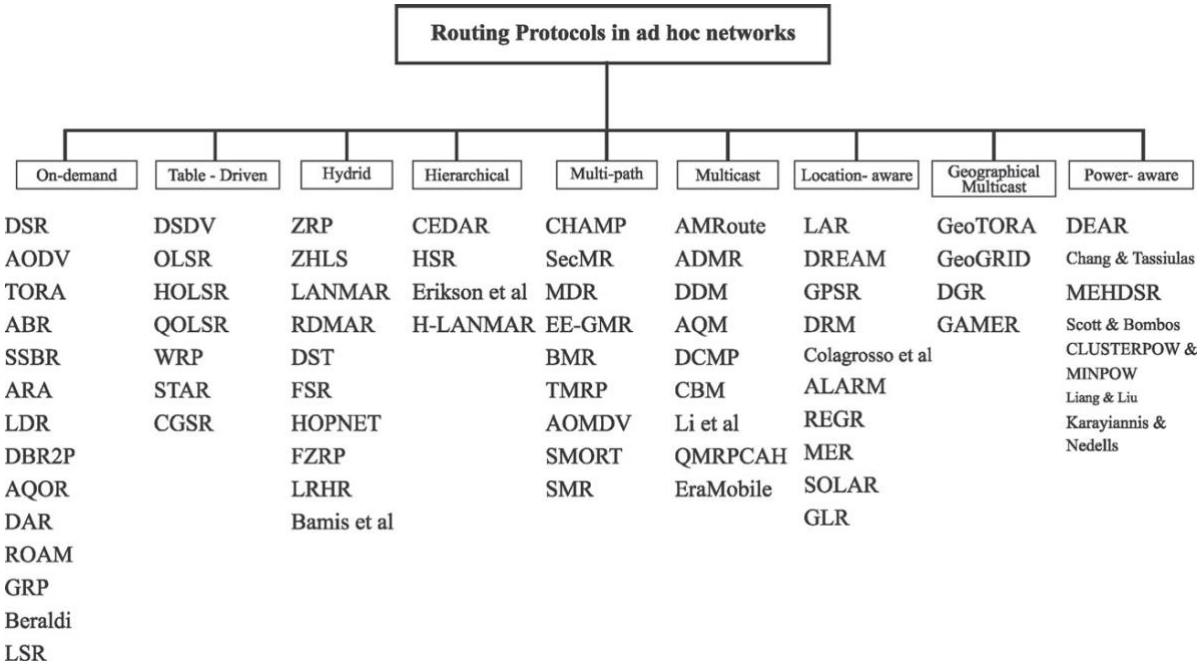


*Figura 11: Categorías de protocolos de enrutamiento ad hoc [15].*

Para el desarrollo del proyecto se va a estudiar en profundidad el protocolo de enrutamiento llamado *Ad Hoc On demand Distance Vector* (AODV), que pertenece a la categoría de protocolo reactivo, y también el enrutamiento *Optimized Link State Routing* (OLSR), perteneciente a la categoría de protocolos proactivos.

#### 2.4.4.1. Ad Hoc On demand Distance Vector (AODV)

Al tratarse de un protocolo reactivo, las rutas únicamente se establecen cuando es necesario. En la Figura 12 se muestran los intercambios de mensajes que se producen durante el proceso de establecimiento de rutas [16]. Existen varios tipos de mensajes durante este proceso.

El mensaje *Hello* se utiliza para el mantenimiento de la conectividad con los nodos vecinos. Estos mensajes se envían periódicamente para confirmar la presencia de vecinos y mantener actualizada la tabla de enrutamiento. Para iniciar el proceso de descubrimiento de rutas se utiliza el mensaje *Route Request* (RREQ), el cual se envía cuando el nodo origen necesita una ruta a un destino y no dispone de una ruta válida en su tabla de enrutamiento.

Cuando un nodo intermedio o el nodo destino recibe un RREQ y dispone de una ruta válida al destino, envía un *Route Reply* (RREP) de vuelta al nodo origen para establecer la ruta. Una vez establecida una ruta válida, se envía el mensaje con los datos reales entre los nodos. Por último, cuando una ruta ya no es válida, se genera un mensaje de error de ruta, *Route Error* (RERR), que se envía a los nodos vecinos para que puedan actualizar sus tablas de enrutamiento.

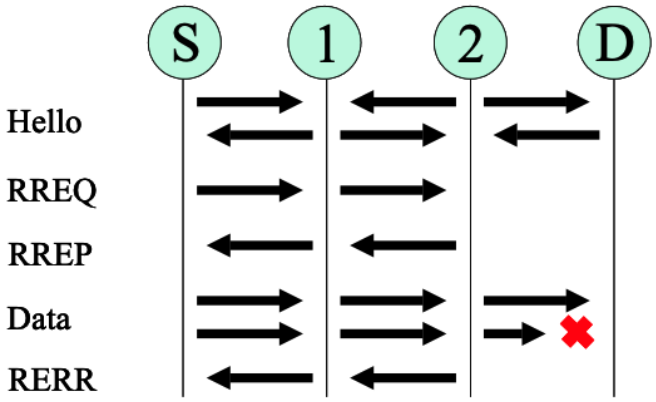


*Figura 12: Intercambio de mensajes del protocolo AODV [16].*

#### 2.4.4.2. Optimized Link State Routing (OLSR)

En este caso, se trata de un protocolo proactivo, esto significa que las rutas están siempre disponibles cuando se necesitan. Esto consigue que se reduzca el tiempo de descubrimiento de rutas [14]. En la Figura 13 se muestra el diagrama de flujo de datos del protocolo OLSR.

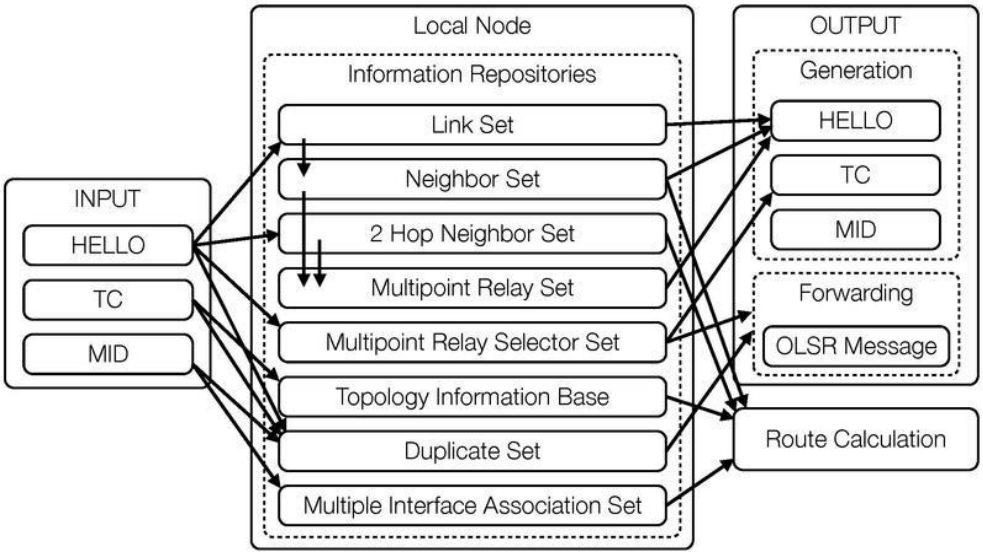


*Figura 13: Diagrama de flujo de datos OLSR [17].*

En la Figura 13 se observan los mensajes que se intercambian durante el proceso de creación y mantenimiento de las rutas. El mensaje *HELLO* se encarga de encontrar a los

vecinos, y mediante la respuesta de estos, se obtienen los vecinos de dos saltos. Además, el protocolo OLSR utiliza mensajes de control denominados TC, del inglés, *Topology Control*, que se envían a todos los nodos de la red permitiendo obtener la topología completa de la red.

Mediante el uso de estos dos tipos de mensajes, que realizan un proceso de inundación[5], el protocolo determina que nodos ejercen de repetidor multipunto (MPR del inglés, *multipoint relay)*. Estos repetidores son los encargados de transmitir la topología de la red y, a diferencia de los procesos de inundación tradicionales, no todos los nodos tienen permitido enviar mensajes con la topología de la red, logrando así una reducción del tráfico y, evitando transmisiones duplicadas. En la Figura 14 se muestra un esquema de un proceso de inundación clásico (imagen de la izquierda) y un proceso de inundación mediante nodos MPR (imagen de la derecha), indicados con los puntos pintados de negro, estos nodos son los únicos encargados de transmitir los mensajes a otros nodos.

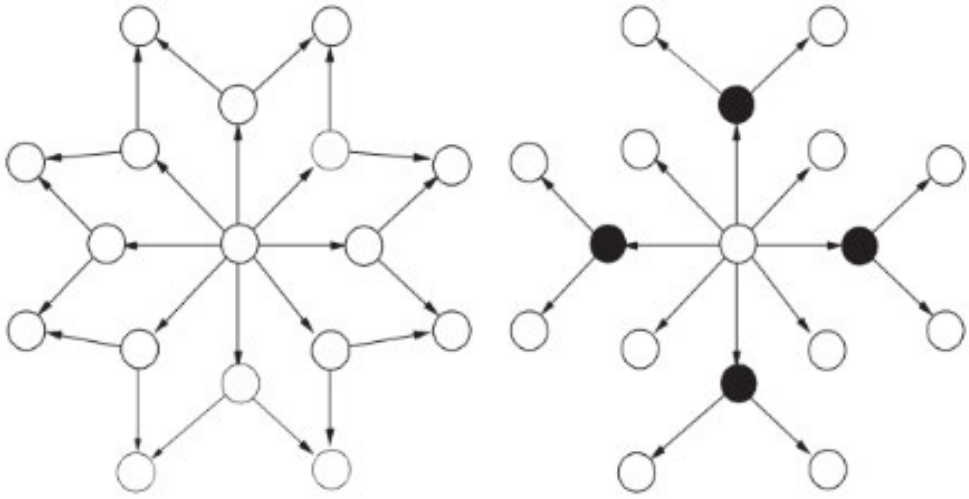

*Figura 14: Proceso de inundación clásico (izquierda) y proceso de inundación mediante MPR (derecha) [14].*

## 2.5. Tecnologías de conectividad inalámbrica: LPWAN

Una vez explicado el concepto de red de sensores es esencial detallar las tecnologías de conectividad inalámbrica que componen estas redes. Estas tecnologías se clasifican en cuatro grupos según el rango de cobertura, como se observa en la Figura 15.

Las categorías de estas redes son: redes de área amplia (WWAN), redes de área metropolitana (WMAN), redes de área local (WLAN) y redes de área personal (WPAN). Cabe

[5] Proceso de inundación: algoritmo simple de enrutamiento que envía todos los paquetes que entran por todas las salidas disponibles, excepto por la que se ha recibido. Término conocido en inglés como *flooding*.

destacar que en la Figura 15 se utiliza el término WNAN para referirse a las redes de área metropolitana, aunque en este documento se utilizará el término WMAN.

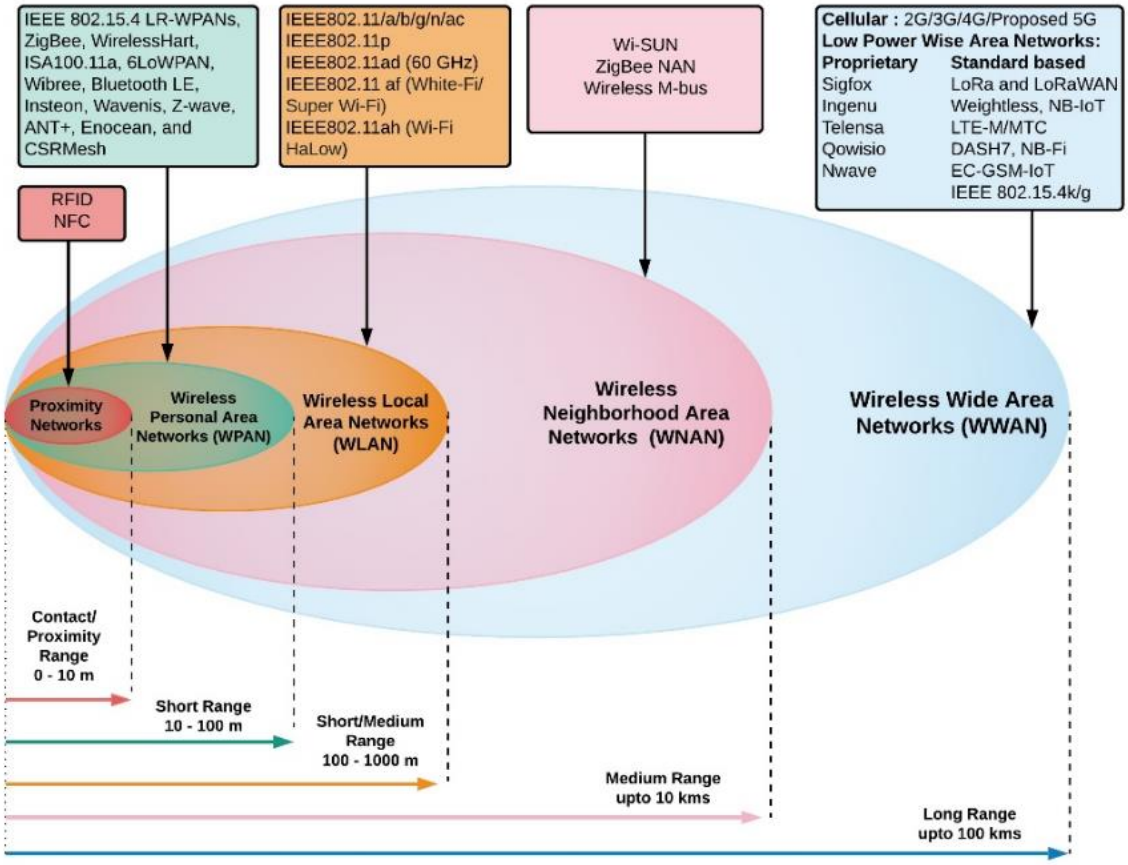


*Figura 15: Clasificación de las redes inalámbricas [18].*

En este estudio se profundizará en las redes de área amplia, especialmente en las redes de área amplia y de baja potencia (LPWAN), ya que representan el protocolo básico para la implementación de sistemas IoT [18] [20]. Antes de ello, en la Tabla 2 se muestran las características de estas tecnologías.

*Tabla 2: Características de las Tecnologías de Redes Inalámbricas.*

| Tipo de red | Rango de Cobertura | Ejemplos de Tecnologías | Frecuencia | Tasa de Datos | Aplicaciones principales |
|---|---|---|---|---|---|
| **WWAN** | Global/Nacional | 3G, 4G, 5G, LTE | Varias (700 MHz a 2.6 GHz) | Alta | Comunicaciones móviles, Internet. |
| **WMAN** | Ciudad/Área Metropolitana | WiMAX, LTE-A | 2 GHz a 11 GHz | Alta | Conectividad urbana, servicios municipales. |

| Tipo de red | Rango de Cobertura | Ejemplos de Tecnologías | Frecuencia | Tasa de Datos | Aplicaciones principales |
|---|---|---|---|---|---|
| **WLAN** | Edificio/Área Limitada | WiFi (IEEE 802.11) | 2.4 GHz, 5 GHz | Alta | Redes domésticas, oficinas, etc. |
| **WPAN** | Habitación/Alcance Personal | Bluetooth, Zigbee, Z-Wave | 2.4 GHz | Baja a Media | Dispositivos personales, automatización. |

Esta tabla resume las principales características de las diferentes tecnologías de redes inalámbricas según su rango de cobertura, frecuencia, tasa de datos y aplicaciones principales, proporcionando una visión general esencial para comprender su implementación en sistemas de sensores.

Las redes de alcance personal, WPAN, se caracterizan por ser de corta distancia, como las utilizadas en identificadores de radiofrecuencia (RFID, del inglés Radio Frequency Identification), transmiten a una velocidad baja y con una buena eficiencia energética. En cambio, una red de área local, WLAN, ofrece una velocidad de transmisión de datos significativamente mayor, pero se limita a una distancia de unos cientos de metros.

Por otro lado, una red de área metropolitana, WMAN, es una evolución de las redes de área local, principalmente utilizada en el ámbito público. Tiene un alcance menor que las redes de área amplia, WWAN. Estas últimas se caracterizan por su uso variable según la cobertura, eficiencia energética, velocidad de datos, etc. Las redes WWAN se pueden subdividir en dos grandes grupos: redes móviles y LPWAN.

Las redes móviles, conocidas como 3G, 4G o 5G, transmiten datos a una velocidad muy alta y cubren distancias de aproximadamente de 10 kilómetros. Estas redes proporcionan un amplio rango de movimiento con una cobertura continua. Por otro lado, las redes LPWAN también son redes inalámbricas con un largo alcance y bajo consumo, con la diferencia que tiene una tasa de envío de datos mucho menor. Las aplicaciones que usan este tipo de redes destacan por el uso de un gran un número de dispositivos en la red.

En la Figura 16 se muestran los distintos tipos de redes dependiendo de la velocidad de transmisión, consumo de energía y distancia. Debido que el proyecto se centra en la utilización de tecnología LoRa a continuación se estudia más en detalle las redes LPWAN.

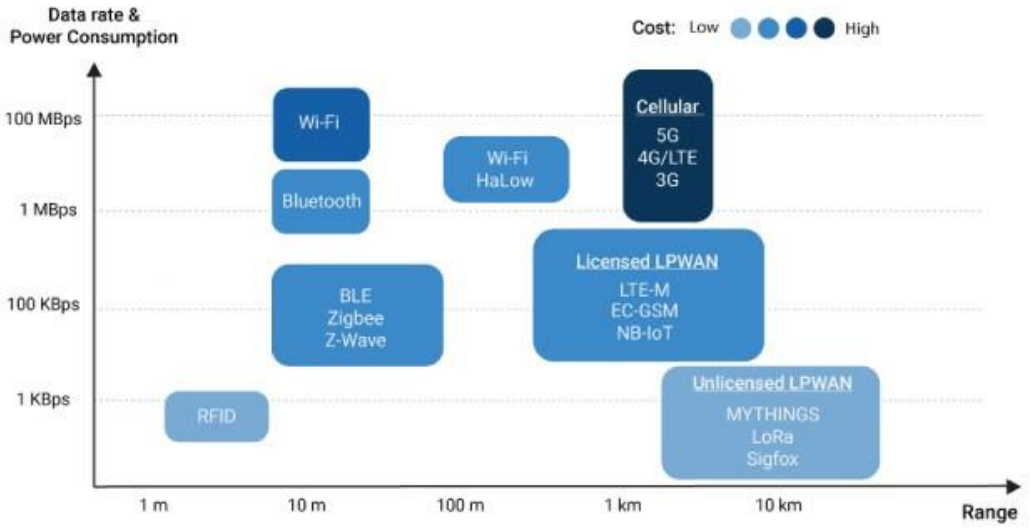


*Figura 16: Clases de redes inalámbricas [21].*

### 2.5.1. Redes de área amplia y de baja potencia (LPWAN)

Una red LPWAN, como su nombre indica, *Low Power Wide Area Network*, es una red de comunicaciones inalámbrica de área amplia, baja potencia y bajo ancho de banda. Las características más importantes de este tipo de redes son las siguientes:

- Alcance geográfico: diseñado para el transporte inalámbrico de datos entre dispositivos con distancias en el rango de los kilómetros.
- Cantidad de datos transmitidos: transporte de pequeñas cantidades de datos y no de manera constante.
- Bajo consumo eléctrico: uso principal en dispositivos que usen baterías y que estas tengan una vida útil de varios años.

El artículo “LPWAN Technologies for IoT Applications: a review” [22] explica que las redes LPWAN funcionan en un espectro ISM, del inglés *Industrial, Scientific and Medical*, esto significa que internacionalmente se han reservado unas bandas de radio para uso industrial, científico y médico. Además, estas bandas de frecuencia se pueden categorizar como bandas con licencia (redes móviles) o sin licencia. Las bandas con licencia suelen tener un coste muy alto por lo que son utilizas principalmente por empresas de telecomunicaciones, en cambio, las bandas sin licencia son accesibles al público, con la desventaja de tener limitada la velocidad de transmisión de los datos. En la Figura 17 se presenta una visión general de las tecnologías LPWAN en sus dos modalidades, con licencia o sin licencia. Para el caso de estudio se va a centrar la atención en las tecnologías NB-IoT, Sigfox y LoRa[23].

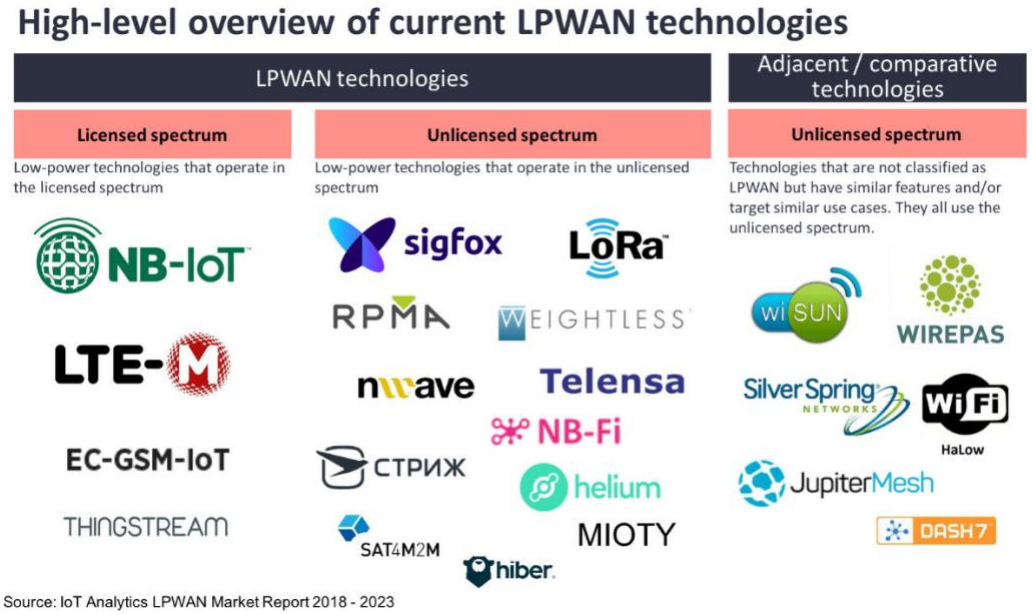


*Figura 17: Visión general de las tecnologías LPWAN [24].*

#### 2.5.1.1. NB-IoT

NarrowBand-IoT, o NB-IoT es una iniciativa de 3GPP [25], esta organización se desarrolla en el campo de la tecnología móvil. Es una tecnología licenciada que ocupa un ancho de banda de 200 kHz y convive con los estándares de comunicación inalámbrica LTE y GSM. NB-IoT se basa en el protocolo LTE reduciendo al mínimo este protocolo y lo adapta para el uso en aplicaciones IoT. Tiene una velocidad de transmisión de 200 kbps en enlace descendente[6] y 20 kbps en enlace ascendente[7], con un tamaño de mensaje de 1600 bytes.

NB-IoT es ventajoso en áreas urbanas por su buena cobertura, funcionando correctamente tanto en interiores como en zonas urbanas de alta densidad y, por último, tiene un tiempo de respuesta menor que otras tecnologías, como LoRa. La desventaja que se encuentra es su dificultad en la implantación del *firmware* por aire (FOTA[8]).

[6] Enlace descendente (Downlink): comunicación desde el dispositivo final (nodo) hacia la estación base (gateway).

[7] Enlace ascendente (Uplink): comunicación desde la estación base (gateway) hacia el dispositivo final (nodo).

[8] FOTA: actualización inalámbrica de firmware, del inglés *Firmware Over The Air*

#### 2.5.1.2. Sigfox

La operadora conocida como Sigfox lleva desde 2010 aportando soluciones IoT de forma global. Es una tecnología que opera sin licencia, es decir de forma gratuita, y trabaja en varios países en las respectivas bandas ISM, véase la Tabla 3.

*Tabla 3: Bandas de frecuencia ISM.*

| Región | Frecuencia (MHz) |
|---|---|
| **Europa** | 868 MHz |
| **Estados Unidos** | 915 MHz |
| **Asia** | 433 MHz |

Sigfox es una opción ventajosa cuando se busca un consumo de energía bajo y un coste menor. Esto se consigue gracias a que los dispositivos finales trabajan en una banda de frecuencia de 100 Hz con una velocidad de datos de unos 100 bps. Esto se traduce en unos niveles de ruido bajos, un consumo reducido y una antena de bajo coste.

Como inconveniente, dispone de una longitud máxima de mensaje en su enlace ascendente de 12 bytes, mientras que en un enlace descendente los mensajes se encuentran limitados a 4 mensajes al día. En la página oficial del operador Sigfox[26], el operador ofrece una perspectiva del despliegue de la red.

#### 2.5.1.3. LoRa/LoRaWAN

La tecnología LoRa, como se ha dicho anteriormente de Sigfox, trabaja en un espectro no licenciado. La asociación LoRa-Alliance[27] desde 2015 ha sido la encargada de establecer un estándar para esta tecnología, que llamó LoRaWAN. En Europa se establece un ciclo de servicio de un 1% para así tener un control de la red y evitar un tráfico elevado. Como características tiene una velocidad de transmisión de entre 300 bps y 50 kbps, y una longitud de los mensajes de 246 bytes.

LoRaWAN tiene la capacidad de descartar información redundante y da fiabilidad a la red transmitiendo un mensaje de confirmación de la recepción. Esto supone una ventaja para el uso de dispositivos móviles ya que no pierden la comunicación con la red. Una desventaja es que tiene una velocidad de transmisión más baja y un tiempo de latencia[9] mayor.

El estándar LoRaWAN diferencia en tres clases los dispositivos según la aplicación que se le vaya a dar:

[9] Latencia: tiempo que tarda en llegar un paquete de datos al destino.

- Clase A: es la clase más utilizada por los dispositivos. Tiene la ventaja de ofrecer un mayor ahorro energético al entrar en modo escucha únicamente después de transmitir un mensaje. Ideal para dispositivos que usan baterías.
- Clase B: en este tipo de dispositivos la puerta de enlace establece una base de tiempos para las ventanas de recepción de mensajes. Esta clase se puede usar tanto en dispositivos que utilizan baterías o una fuente de alimentación externa.
- Clase C: menor ahorro de energía al encontrarse los dispositivos en una escucha continua y solo transmite cuando es necesario. Recomendado para dispositivos que dispongan de una fuente de alimentación externa.

En la Figura 18 se presenta una gráfica que ilustra la relación entre la vida útil de la batería y la latencia en las comunicaciones de recepción para cada una de estas clases de dispositivos.

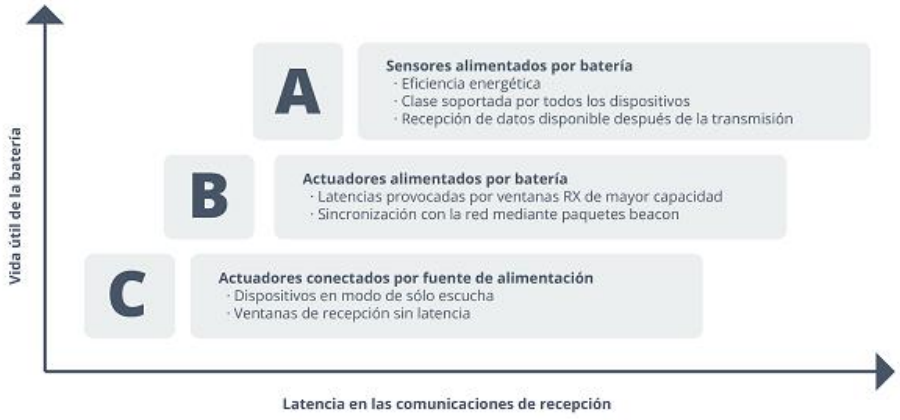


*Figura 18: Clases de dispositivos LoRaWAN[28].*

#### 2.5.1.4. Comparativa NB-IoT, Sigfox y LoRa

Para cada una de estas tecnologías, es importante tener en cuenta los siguientes factores: alcance, cobertura, vida útil del dispositivo y coste, entre otros. En la Tabla 4 y la Figura 19 se muestra una comparativa de estas tres tecnologías: NB-IoT, Sigfox y LoRa/LoRaWAN.

*Tabla 4: Comparativa NB-IoT, Sigfox, LoRa/LoRaWAN.*

| Tecnología | NB-IoT | Sigfox | LoRa/LoRaWAN |
|---|---|---|---|
| **Ancho de Banda** | 200 kHz o compartido | 100 kHz | < 500 kHz |
| **Espectro** | Bandas de frecuencia LTE licenciadas | Bandas de frecuencia ISM no licenciadas (ver Tabla 3) | Bandas de frecuencia ISM no licenciadas (ver Tabla 3) |

| Tecnología | NB-IoT | Sigfox | LoRa/LoRaWAN |
|---|---|---|---|
| **Velocidad de datos** | 200 kbps | 100 bps | >300 bps y <50 kbps |
| **Estándar** | 3GPP | - | LoRa-Aliance |
| **Alcance** | 1 km (urbano)<br>10 km (rural) | 1 km (urbano)<br>20 km (rural) | 10 km (urbano)<br>40 km (rural) |
| **Vida útil (años)** | <10 | <10 | <10 |
| **Consumo de energía** | Bajo | Bajo | Bajo |

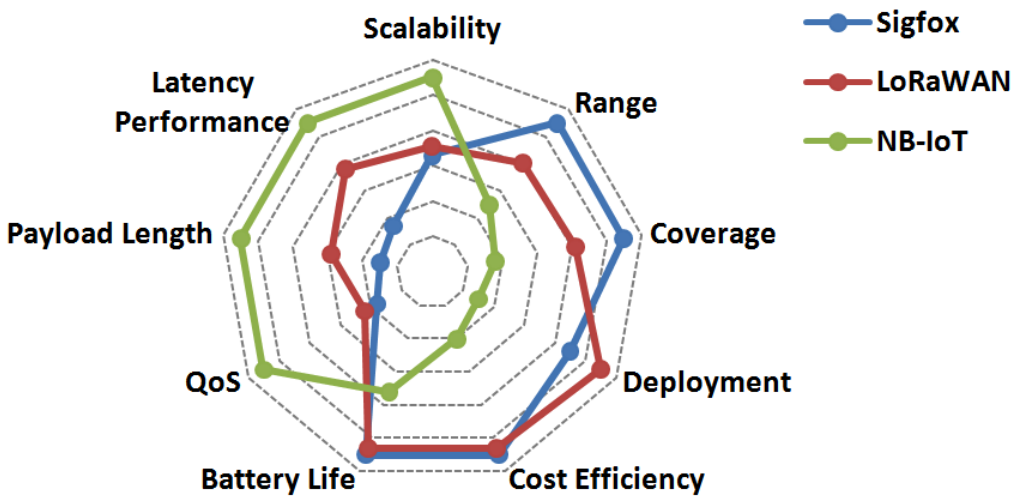


*Figura 19: Comparativa Sigfox, LoRaWAN y NB-IoT [23].*

## 2.6. Dispositivos comerciales

En este apartado se muestran distintos dispositivos comerciales, con tecnología LoRa, que se encuentran en el mercado. La tecnología LoRa (Long Range) es conocida por su capacidad de comunicación a larga distancia con bajo consumo de energía, lo cual es ideal para aplicaciones de Internet de las Cosas (IoT). A continuación, se detallan los tipos de dispositivos y sus fabricantes, abarcando desde módulos de transmisión básicos hasta dispositivos completamente diseñados con infraestructura para facilitar su personalización.

- Módulo de transmisión

Uno de los componentes más básicos en la tecnología LoRa son los módulos de transmisión. Un ejemplo representativo de estos es el SX127x [29], un chip de Semtech que proporciona las capacidades básicas de comunicación LoRa, se muestra en la Figura 20. Este módulo suele ser utilizado por desarrolladores y empresas para integrar la funcionalidad LoRa en sus propios diseños de hardware.


*Figura 20: Chip SX127x de Semtech [29].*

- Microprocesadores con LoRa integrado

Además de los módulos de transmisión básicos, también existen microprocesadores programables que ya tienen integrado un módulo LoRa. Fabricantes como LilyGo, Arduino o Heltec, comercializan en sus páginas web diferentes placas de desarrollo que facilitan la implementación de proyectos IoT. Estas placas incluyen microcontroladores con la capacidad de procesamiento suficiente para ejecutar aplicaciones complejas, como el popular microcontrolador ESP32. Algunos ejemplos de estas placas son la LilyGo TTGO LoRa, la Arduino MKR WAN 1300 y las placas Heltec WiFi LoRa 32, las cuales se muestran en la Figura 21.


*Figura 21: LilyGo TTGO LoRa (arriba izquierda) [30], MKR WAN 1300 (arriba derecha) [31] y Heltec WiFi LoRa 32 (abajo [32]).*

- Dispositivos completamente diseñados

Por último, existen dispositivos comerciales que no solo incluyen la tecnología LoRa, sino que también incluyen la infraestructura necesaria para facilitar su personalización y despliegue. Estos dispositivos son ideales para soluciones IoT rápidas, eliminando la necesidad de desarrollar *hardware* desde cero. En este caso se encuentran marcas como Dragino [33] o Kerlink [34].

Dragino Technology Co., Limited es una empresa que fue fundada en 2010 y se especializan en el diseño, desarrollo y fabricación de productos tecnológicos. Su catálogo incluye una variedad de productos con tecnología LoRa/LoRaWAN, algunos ejemplos se muestran en la Figura 22.

*Figura 22: LoRaWAN To Modbus Gateway (izquierda)[35] y LoRa/LoRaWAN IoT Kit v3 (derecha) [36].*

Por otro lado, Kerlink, fundada en 2004, se enfoca en ofrecer soluciones IoT. Su catálogo incluye principalmente *gateways* LoRaWAN, diseñados para facilitar la implementación de redes IoT robustas y eficientes. En la Figura 23 se muestra dos módulos de este fabricante.


*Figura 23: Wirnet iFemtoCell-evolution LoRaWAN Gateway (izquierda) [37] y Wirnet iBTS LoRaWAN Gateway (derecha) [38].*

# 3. Especificaciones y restricciones del diseño

En este capítulo se definen las especificaciones y restricciones del proyecto, fundamentales para la implementación y evaluación de la red LoRa propuesta.

El objetivo del proyecto es la creación de una red de sensores con comunicación LoRa. Esta tecnología destaca por la capacidad de transmitir datos a grandes distancias consumiendo una cantidad baja de energía, lo cual es esencial para las aplicaciones de sensores remotos. Para lograr esto es necesario utilizar integrados o módulos que sean capaces de generar y trabajar con comunicaciones LoRa de manera efectiva.

La información se visualizará en la plataforma web de Nazaríes Intelligenia, lo que permite un control preciso del entorno y facilita el análisis de los datos recopilados. Esta aplicación aporta al proyecto una mejora gracias a que dispone de una interfaz de usuario intuitiva y sencilla para los usuarios.

Se fabricarán prototipos para el entorno de pruebas que validen el diseño y funcionamiento de la red. En el diseño se dará importancia a los materiales para proporcionar durabilidad y firmeza, además de tener un tamaño reducido que facilite su transporte e instalación en cualquier lugar.

La alimentación de los dispositivos es otro aspecto importante para tener en cuenta en el diseño. Estos dispositivos utilizarán baterías recargables que proporcionen una autonomía durante un largo periodo de tiempos sin necesidad de recarga. Esto implica optimizar tanto el software, que debe ser eficiente en el manejo de los recursos, como del hardware, que debe utilizar componentes de bajo consumo.

# 4. Descripción de la solución propuesta

En este capítulo se presenta una descripción detallada de la solución propuesta para el proyecto, teniendo en cuenta las especificaciones del apartado 3. Se explican los motivos detrás de la elección de dispositivos como los modelos TTGO LoRa32 y el sensor de temperatura y humedad, así como las herramientas de software empleadas. Adicionalmente, se describe el desarrollo del diseño implementado, proporcionando una explicación en profundidad de cada uno de los elementos que intervienen en la red.

## 4.1. Selección del material

Una vez que se han establecido los requisitos, se procede al desarrollo de la selección de componentes para la ejecución del proyecto. Esta sección consta de dos subapartados. El primero aborda el aspecto hardware, en el que se revisan las placas de desarrollo y los sensores empleados. El segundo se enfoca en la parte software, detallando los programas y librerías empleadas.

### 4.1.1. Componentes Hardware

#### 4.1.1.1. TTGO LoRa32 v1.0 y TTGO LoRa32 v2.1_1.6

Después de un análisis exhaustivo de varios dispositivos con módulos LoRa, se han elegido las placas de desarrollo TTGO LoRa32 V1.0 [39] (véase Figura 24 y Figura 25) y TTGO LoRa32 V2.1_1.6 [30] (véase Figura 26 y Figura 27) fabricadas por LilyGo para formar parte de este proyecto. Estas placas destacan por sus impresionantes especificaciones técnicas, su diseño compacto y su precio accesible, lo que las convierte en una opción ideal para servir como puerta de enlace y nodos en la red de sensores.

La placa TTGO LoRa32 V1.0 está equipada con un microcontrolador ESP32 de doble núcleo, que ofrece un rendimiento potente y una amplia variedad de periféricos. Además, cuenta con un módulo LoRa SX1276 integrado, lo que le permite comunicarse de manera eficiente a largas distancias con otros dispositivos LoRa. Esta placa también incluye una pantalla OLED integrada de 0.96 pulgadas para facilitar la visualización de datos y una ranura para tarjetas microSD para el almacenamiento de datos locales. En la Tabla 5 se presentan las especificaciones de esta placa.

*Tabla 5: Especificaciones TTGO LoRa32 V1.0 [39].*

| | |
|---|---|
| **MCU** | ESP32 |
| **Chip LoRa** | SX1276 |
| **Flash** | 4M |
| **Chip Serial** | CH910X |
| **Antena** | Antena WiFi 3D<br>Antena LoRa con soporte IPEX |
| **OLED** | 0.96 Inch SSD1306 OLED 128*64 |
| **Alimentación** | 3.3V a 7V |
| **Temperatura de funcionamiento** | -40 ° C to + 90 ° C |

*Figura 24: Placa de desarrollo del fabricante LilyGo LoRa32 V1.0 [39].*

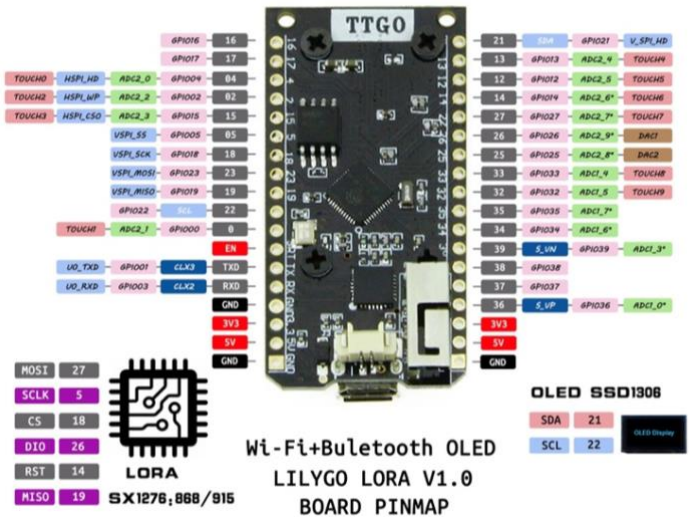


*Figura 25: Pinout LoRa32 V1.0 [39].*

Por otro lado, la placa TTGO LoRa32 V2.1_1.6 también está basada en el microcontrolador ESP32 y cuenta con un módulo LoRa SX127X. Sin embargo, se distingue por su diseño actualizado, que incluye una antena SMA reemplazable para una mejor conectividad y flexibilidad de configuración. Además, esta placa cuenta con una ranura para tarjetas SIM, lo que la hace ideal para aplicaciones que requieren conectividad celular para la transferencia de datos. En la Tabla 6 se presentan las especificaciones de esta placa.

*Tabla 6: Especificaciones TTGO LoRa V2.1_1.6 [30].*

| | |
|---|---|
| **MCU** | ESP32 |
| **Flash** | 4MB |
| **Chip Serial** | CH9102 |
| **Protocolo inalámbrico** | WiFi y Bluetooth 4.2 |
| **Antena** | Antena 3D WiFi (por defecto) (Soporte WiFi IPEX antena externa)<br>Antena LoRa |
| **Alimentación** | Soporta USB Micro / Li-Po Batería de doble fuente de alimentación<br>JST GH 2pin 1.25mm<br>[USB puede alimentar la batería] |


*Figura 26: Placa de desarrollo del fabricante LilyGo LoRa32 V2.1_1.6 [30].*

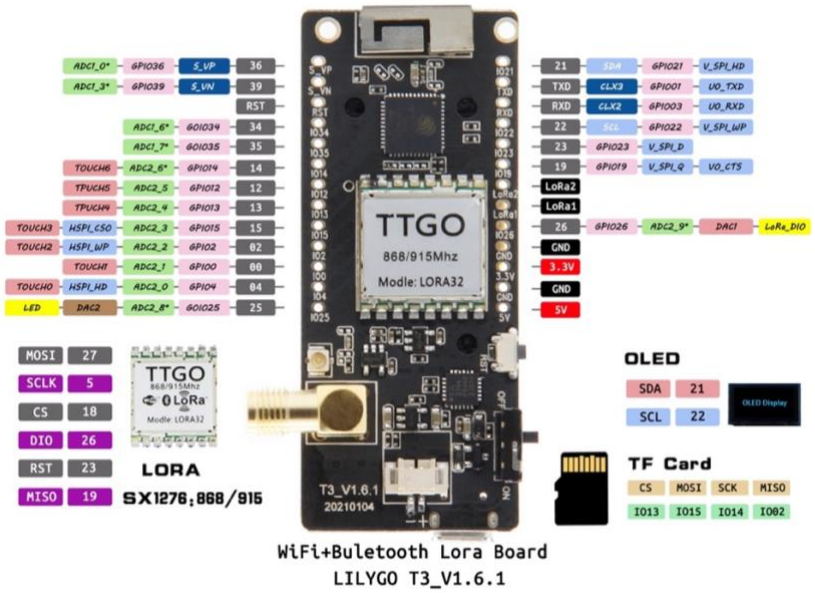


*Figura 27: Pinout LoRa32 V2.1_1.6 [30].*

Ambas placas ofrecen soporte para baterías LiPo, lo que las hace adecuadas para aplicaciones de baja potencia y alta portabilidad. Su compatibilidad con el entorno de desarrollo Arduino y el amplio soporte de la comunidad hacen que estas placas sean fáciles de programar y personalizar para satisfacer las necesidades específicas del proyecto. En resumen, la combinación de sus especificaciones técnicas robustas, su diseño compacto y su precio asequible las convierte en la elección perfecta para servir como componentes clave en la implementación de la red LoRa para este proyecto.

#### 4.1.1.2. Sensor de Temperatura y Humedad

Se ha considerado y analizado una variedad de sensores disponibles en el mercado, como el DHT11, DHT22, DS18B20, SHT21, así como sensores de CO2, sensores de luz, sensores de movimiento, entre otros. Sin embargo, para nuestro propósito, se ha decidido utilizar un sensor de temperatura y humedad debido a su fácil implementación y disponibilidad.

El sensor DHT11, mostrado en la Figura 28, ha sido seleccionado por su sencillez de implementación, su amplia disponibilidad en el mercado y su costo accesible. Este sensor integra un termistor para medir la temperatura y un sensor de humedad capacitivo, proporcionando mediciones relativamente precisas en un paquete compacto y fácil de usar. Además, el DHT11 utiliza un protocolo de comunicación de un solo cable, lo que simplifica su integración en el sistema. En la Tabla 7 se presentan las especificaciones de esta placa.

*Tabla 7: Especificaciones DHT11 del fabricante AZ-Delivery [40].*

| **Características** | |
|---|---|
| **Alimentación** | 3.3V – 5.5V |
| **Consumo** | 0.3 mA durante la medida<br>60 μA en modo reposo |
| **Interfaz de comunicación** | Protocolo de comunicación de un solo cable |
| **Temperatura** | |
| **Rango** | 0ºC a 50ºC |
| **Precisión** | ±2ºC |
| **Humedad Relativa** | |
| **Rango** | 20% a 90% |
| **Precisión** | ±5% |

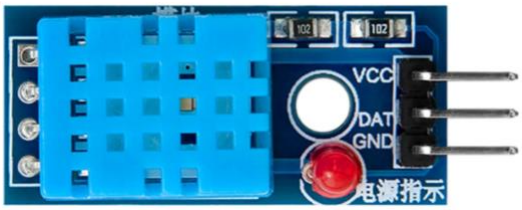


*Figura 28: Sensor DHT11 [40].*

Es esencial tener en cuenta que, aunque la elección del sensor es importante para obtener mediciones precisas, en el contexto de este proyecto, el énfasis principal recae en el diseño y la implementación efectiva de la red de comunicación Mesh. Por lo tanto, se ha priorizado la selección de un sensor que cumpla con los requisitos básicos de medición de temperatura y humedad, permitiendo así enfocar los esfuerzos en el desarrollo de la infraestructura de comunicación necesaria para el funcionamiento óptimo del sistema.

### 4.1.2. Componente Software

Para la programación de los dispositivos, se empleará un software compatible que permita desarrollar y cargar el firmware de manera eficiente. Dos opciones comunes y altamente utilizadas son el Arduino IDE y Visual Studio Code con la extensión de PlatformIO.

El Arduino IDE [41] es una plataforma de desarrollo integrada que ofrece un entorno de programación simple y fácil de usar, especialmente diseñado para trabajar con placas Arduino y compatibles. Ofrece una interfaz gráfica intuitiva y una amplia gama de bibliotecas y ejemplos que facilitan el desarrollo de proyectos.

Por otro lado, Visual Studio Code con la extensión de PlatformIO [42][43] proporciona un entorno de desarrollo más avanzado y versátil. Visual Studio Code es un editor de código abierto y altamente personalizable que ofrece características avanzadas como resaltado de sintaxis, depuración integrada y control de versiones. La extensión de PlatformIO agrega soporte para el desarrollo de firmware para una amplia variedad de plataformas, incluidas las placas Arduino, ESP8266, ESP32, entre otras. Además, PlatformIO ofrece características adicionales como la gestión de bibliotecas, la integración con herramientas de construcción avanzadas y la capacidad de trabajar en proyectos más grandes y complejos.

Después de evaluar las diferentes opciones disponibles para el desarrollo de firmware, se ha llegado a la conclusión de que la última opción, Visual Studio Code con la extensión de PlatformIO, es la más adecuada para este proyecto. Esta decisión se basa en la versatilidad y las características avanzadas que ofrece Visual Studio Code, junto con la extensión de PlatformIO, que permite trabajar con una amplia variedad de plataformas de desarrollo y proporciona herramientas adicionales para la gestión de proyectos y la integración con sistemas de control de versiones. Con esta combinación de herramientas, se espera facilitar el desarrollo y la depuración del firmware, así como mejorar la eficiencia y la productividad del equipo de desarrollo.

Además, como parte de la colaboración en este Proyecto Fin de Grado (PFG) con la empresa Nazaríes Intelligenia, se aprovechará la plataforma de IoT [1] proporcionada por dicha empresa. Nazaríes Intelligenia ofrece una plataforma completa y robusta para el desarrollo, despliegue y gestión de soluciones IoT. Esta plataforma proporciona herramientas para la recopilación, procesamiento y visualización de datos de sensores, así como funciones avanzadas de análisis y control remoto. Al integrar la plataforma de Nazaríes Intelligenia en el proyecto, se podrá aprovechar su experiencia y recursos para mejorar la implementación y la funcionalidad del sistema, así como facilitar la escalabilidad y el mantenimiento a largo plazo. Esta colaboración permite una coordinación entre el trabajo académico y la experiencia práctica de la empresa en el campo de la Internet de las Cosas (IoT), enriqueciendo así el desarrollo del proyecto y brindando una solución más completa y sólida.

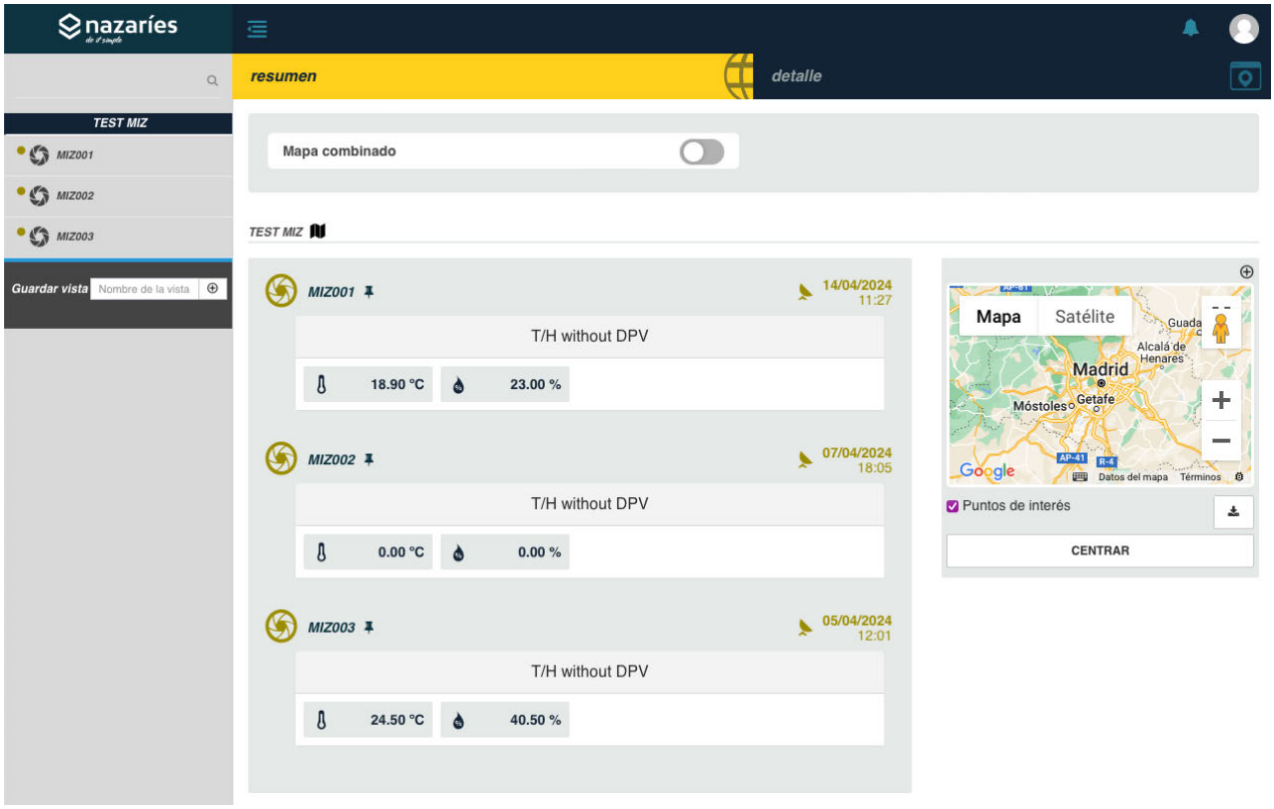


*Figura 29: Plataforma IoT de Nazaríes Intelligenia.*

## 4.2. Desarrollo del proyecto

En la siguiente subsección, se explica en detalle y el software implementado para el correcto funcionamiento de la red. Dado que se trata de dos dispositivos con funcionabilidades distintas, para facilitar la compresión del software, esta sección se dividirá en cinco partes.

Primero, se describe la configuración de los periféricos de ambos dispositivos. Segundo, se detalla el software específico para la puerta de enlace. En tercer lugar, se explica el software específico para los nodos transmisores. En el cuarto punto se aborda la planificación de la red y de las pruebas realizadas, que se describen en detalle en el capítulo 5. En el quinto y último, se presenta el diseño de los prototipos utilizados en las pruebas.

### 4.2.1. Configuración de los dispositivos

La configuración de los dispositivos dependerá de la función a la que estén destinados. En este proyecto se identifican dos tipos de dispositivos: puerta de enlace y nodo. Existen partes de la inicialización que se comparte en ambos dispositivos aunque no en su totalidad. En la Tabla 8 se muestran los módulos que intervienen en cada dispositivo como los periféricos que se configuran.

*Tabla 8: Módulos y periféricos software.*

| Puerta de enlace (Gateway) | Nodo |
|---|---|
| **Identificador único (ID)** | Identificador único (ID) |
| **Módulo LCD** | Módulo LCD |
| **Módulo LoRa** | Módulo LoRa |
| **Conexión WiFi** | - |
| **-** | Módulo Temporizadores |
| **-** | Sensor de DHT11 |
| **Tabla de enrutamiento** | Tabla de enrutamiento |

A continuación, se da una breve descripción de los módulos y periféricos que intervienen. Existen módulos comunes, como:

- Identificador único: cada dispositivo tiene un identificador que permite el reconocimiento del dispositivo dentro de la red.
- Módulo LCD: en este módulo se programa la visualización de información del dispositivo por el LCD, indicando si es la puerta de enlace o un nodo, el identificador correspondiente y, en el caso de los nodos, el enrutamiento hacía la puerta de enlace.
- Módulo LoRa: este módulo configura la comunicación de largo alcance entre la puerta de enlace y los nodos transmisores, utilizando la tecnología LoRa.

Por otro lado, la puerta de enlace está equipada con conectividad WiFi, lo que le permite enviar datos a una red local o a internet para el almacenamiento y análisis de los datos recibidos. Por el contrario, los nodos no tienen acceso a una conexión WiFi, pero disponen de un sensor de temperatura y humedad, el DHT11, conectado a uno de sus pines de entrada y salida (GPIO), que proporciona los datos ambientales que los nodos envían a la puerta de enlace. Además, los nodos también tienen un módulo con temporizadores para el envío de datos, lo que les permite realizar mediciones periódicas.

Por último, se genera una tabla de enrutamiento para el algoritmo de enrutamiento. Esta tabla se compone de cuatro datos principalmente, los cuales son:

- **Destino (*dest*).** Para el caso de estudio el destino siempre será la dirección única de la puerta de enlace o *gateway*. Para este caso, será siempre la dirección 0x8C en hexadecimal o 140 en número decimal.
- **Siguiente salto (*nextHop*).** Dirección única a la que se realiza el envío de información.
- **Contador de saltos (*countHop*).** Número de saltos hasta alcanzar el destino marcado. Él valor mínimo es “1”.

- **Secuencia de la ruta (*sequenceRoute*).** Secuencia de la ruta establecida, empezando siempre por el identificador único propio hasta el identificador único del destino. Cada identificador va separado por un punto y coma ";". Un ejemplo de cómo sería la tabla de enrutamiento es la que se muestra en la Tabla 9 (las direcciones se presentan como datos decimales).

*Tabla 9: Ejemplo de tabla de enrutamiento de un nodo.*

| *dest* | *nextHop* | *hopCount* | *sequenceRoute* |
|---|---|---|---|
| $\mathbf{140_D}$ | $140_D$ | 1 | "244;140;" |

El código generado para estas configuraciones se encuentra detallado en el Anexo I que se encuentra al final de esta memoria.

### 4.2.2. Estructura de mensajes de la Red

En la implementación de la red de sensores con LoRa se establece una estructura de mensajes específica para garantizar una comunicación organizada. La estructura del mensaje se divide en cinco campos principales: ID propio, ID vecino, ID destino, ID mensaje y Datos. Cada uno de estos campos tiene un tamaño definido, excepto el campo de Datos, que es de tamaño variable. En la Tabla 10 se detalla la estructura.

*Tabla 10: Estructura de mensajes.*

| ID propio | ID vecino | ID destino | ID mensaje | Datos |
|---|---|---|---|---|
| 1byte | 1 byte | 1 byte | 1 byte | variable |

- **ID propio:** identificador único del nodo que envía el mensaje.
- **ID vecino:** identificador único del nodo vecino. Este campo depende de dos situaciones:
  - **Situación 1:** el nodo que transmite tiene enlace directo con la puerta de enlace. En este caso el ID propio y el ID vecino es el mismo.
  - **Situación 2:** el nodo que transmite no tiene enlace directo con la puerta de enlace. Esto significa que para alcanzar la puerta de enlace el nodo pasa por un nodo intermedio que transmite la información del nodo principal. De esta forma, con este campo el *gateway* tiene conocimiento de que nodo provienen las medidas.
- **ID destino:** identificador del nodo destino, en este caso será siempre el identificador del *gateway*.
- **ID mensaje:** identificador del tipo de mensaje. Existen los siguientes tipos:

  - **RREQ (1):** mensaje de solicitud para el establecimiento de la ruta (*Route Request*).
  - **RREP (2):** mensaje de respuesta para el establecimiento de la ruta (*Route Reply*).
  - **ACK**[10] **(3):** mensaje de confirmación de recepción de datos.
  - **DATA (4):** mensaje de trasmisión de datos del sensor.
  - **RRER (5):** mensaje de error en la ruta (*Route Error*).
- Datos: información que se desea transmitir. Para el caso de los mensajes RREQ. RREP, la información a transmitir es la secuencia de la ruta que dispone el dispositivo; los tipos RERR y ACK, este espacio lo tienen vacío, y por último el tipo DATA transmite las mediciones de temperatura y humedad tomadas del sensor.

En las tablas Tabla 11 y Tabla 12 se muestra un ejemplo de la estructura trasmitida mediante LoRa para las situaciones 1 y 2 respectivamente (los identificadores se presentan como datos decimales).

*Tabla 11: Ejemplo de la estructura de un mensaje tipo DATA en situación 1.*

| ID propio | ID vecino | ID destino | ID mensaje | Datos[11] |
|---|---|---|---|---|
| $\mathbf{244_D}$ | $244_D$ | $140_D$ | 4 (DATA) | “263;360” |

*Tabla 12: Ejemplo de la estructura de un mensaje tipo DATA en situación 2.*

| ID propio | ID vecino | ID destino | ID mensaje | Datos |
|---|---|---|---|---|
| $\mathbf{112_D}$ | $244_D$ | $140_D$ | 4 (DATA) | “263;360” |

### 4.2.3. Funcionalidad del puerta de enlace (*gateway*)

Como se ha explicado en la sección 4.2.1 inicialmente se establece la configuración del dispositivo, una vez que esta configuración se ha completado se establece el modo de escucha de mensajes LoRa. Se hace una diferenciación de si el mensaje va destinado al dispositivo en cuestión. La razón de esto es evitar las interferencias en la comunicación debido a la posibilidad de existencia de redes con comunicación LoRa cercanas distintas a la del proyecto.

Una vez recibido el mensaje este se analiza para identificar el tipo. Los tipos de mensajes que puede recibir una puerta de enlace son los siguientes:

[10] Acknowledgement (ACK) traducción al español como “acuse de recibido” o “asentimiento”.

[11] Datos de temperatura y humedad. 26.3ºC y 36% de humedad

- **RREQ (*Route Request*):** si el mensaje recibido es una solicitud de ruta, la puerta de enlace contesta con un RREP que contiene su tabla de enrutamiento.
- **DATA:** mensaje que contiene los datos del sensor. La puerta de enlace es la encargada de procesarlos y reenviarlos al servidor. Una vez estos datos son transmitidos al servidor de forma correcta, este confirma la recepción enviando un mensaje tipo ACK.

En la Figura 65 del Anexo II: Diagramas de Flujo se presenta el diagrama de flujo del funcionamiento que se plantea para la puerta de enlace. A continuación, se explica qué protocolo se ha utilizado para conectar con el servidor web y reenviar la información de los sensores a la plataforma Web.

#### 4.2.3.1. Envío de información a la aplicación web

Una vez el *gateway* recibe los datos obtenidos por los nodos, este se encarga de trasmitirlos a la aplicación web mediante el Protocolo de Transferencia del Hipertexto, del inglés, *Hypertext Transfer Protocol*, HTTP.

HTTP se define como "un protocolo o conjunto de reglas de comunicación para la comunicación cliente-servidor", según lo describe AWS [44]. Es un protocolo de la capa de aplicación y se caracteriza por realizar solicitudes a un servidor, y el servidor contesta con un mensaje de respuesta. En las tablas Tabla 13 y Tabla 14 se presentan ejemplos de solicitudes y respuestas que maneja este protocolo.

*Tabla 13: Ejemplos peticiones al servidor por HTTP.*

| Solicitud | Descripción |
|---|---|
| **GET** | Obtener información del sitio Web. |
| **PUT o POST** | Enviar datos para crear o actualizar recursos. |

*Tabla 14: Ejemplos respuesta del servidor por HTTP.*

| Respuesta | Descripción |
|---|---|
| **200 - OK** | La solicitud ha tenido éxito |
| **201 - Created** | Solicitud con éxito y creación de un nuevo recurso (típica respuesta de una solicitud de tipo PUT) |
| **400 - Bad Resquest** | El servidor no puede interpretar la petición (mala sintaxis). |
| **404 - Not Found** | El servidor no encuentra el contenido solicitado. |

*Por motivos de confidencialidad con Nazaríes Intelligenia, no se permite la distribución del código que implementa la comunicación con el servidor web proporcionado por dicha empresa, ni el manejo de la información para su subida a la aplicación.*

### 4.2.4. Funcionalidad de los nodos

Al igual que con la puerta de enlace, inicialmente se realiza la configuración del dispositivo como indica en la sección 4.2.1. La programación que se plantea para este tipo de dispositivos se centra en la creación de dos hilos, uno para la escucha de paquetes recibidos por LoRa y otro encargado del proceso de envío de paquetes.

Los nodos en su proceso de "escucha" llega a recibir los siguientes tipos de mensajes:

- **RREQ (*Route Request*):** si el mensaje recibido es una solicitud de ruta, el nodo contesta con un RREP que contiene su tabla de enrutamiento.
- **RREP (Route Replay):** mensaje de respuesta de los nodos vecinos. El algoritmo de enrutamiento es el que toma el control de estos mensajes para establecer la mejor ruta con destino a la puerta de enlace.
- **RERR (Route Error):** la recepción de un mensaje de error significa que un nodo vecino ha perdido conectividad.
- **DATA:** mensaje que contiene los datos del sensor. Los nodos se encargan de reenviar la información recibida. Una vez estos datos son transmitidos se confirma la recepción enviando un mensaje tipo ACK.
- **ACK:** mensaje de confirmación de recepción. En caso de no recibir este mensaje pasado un tiempo de la transmisión, el nodo se encarga de mandar de nuevo los datos obtenidos por el sensor. Además, se tiene un control del número de reenvíos, si este supera el límite establecido se da la ruta de envío por inaccesible y se manda un paquete RERR (*Route Error*).

Y en su proceso de estados el nodo transmite los siguientes tipos:

- **RREQ (*Route Request*):** solicitud de tablas de enrutamiento. Enviado a los vecinos para que el algoritmo de enrutamiento tome la decisión de cuál es la mejor ruta hacia la puerta de enlace.
- **RERR (Route Error):** pérdida de conexión con el nodo vecino que se establece como ruta.
- **DATA:** mensaje que contiene los datos del sensor.
- **ACK:** mensaje de confirmación de recepción. Enviado en el caso de ser un nodo intermedio.

En las figuras Figura 66, Figura 67 y Figura 68 del Anexo II: Diagramas de Flujo se presenta el diagrama de flujo del funcionamiento que se plantea para los nodos.

#### 4.2.4.1. Algoritmo de enrutamiento

El algoritmo de enrutamiento se basa en la implementación de un algoritmo AODV, explicado en la sección 2.4.4.1. Una vez se reciben los mensajes RREP, este los almacena para posteriormente la toma de decisión de la mejor ruta. Esta decisión depende de los siguientes parámetros:

- Enlace directo con la puerta de enlace.
- Calidad de la conexión (RSSI)
- En situación de no tener un enlace directo con la puerta de enlace, se considera el número de saltos que realiza la información hasta llegar al destino, con el objetivo de minimizar este número.

Los detalles del algoritmo de enrutamiento se muestran en las pruebas realizadas en campo del capítulo 5 sección 5.3.

### 4.2.5. Planificación de la red

Para las pruebas en campo hay que tener en cuenta cuando se implementa una red con comunicación LoRa. Uno de los factores más importantes es la distancia entre los dispositivos. Para conocer la distancia teórica de separación entre nodos se utiliza lo que se conoce en inglés como *Link Bucket* traducido como presupuesto de enlace [45] y se calcula mediante la siguiente ecuación:

$$P_{RX} = P_{TX} - G_{TX} - L_{TX} - L_{FS} - L_M + G_{RX} - L_{RX} \quad [1]$$

Donde:

- $P_{RX}\ (dBm)$: $potencia\ recibida.$
- $P_{TX}\ (dBm)$: $potencia\ de\ salida\ del\ trasnmisor.$
- $G_{TX}(dBi)$: $ganancia\ de\ la\ antena\ transmisora.$
- $L_{TX}\ (dB)$: $pérdidas\ del\ transmisor\ (conectores\ \dots).$
- $L_{FS}\ (dB)$: $pérdidas\ de\ trayectoria\ en\ espacio\ libre.$
- $L_M\ (dB)$: $pérdidas\ diversas.$
- $G_{RX}\ (dBi)$: $ganancia\ de\ la\ antena\ receptora.$
- $L_{TX}\ (dB)$: $pérdidas\ del\ receptor\ (conectores\ \dots).$

El *Link Bucket* depende a su vez de las pérdidas de trayectoria en espacio libre, es decir, las pérdidas durante la propagación entre las antenas transmisoras y receptoras. Esta pérdida se puede calcular mediante la siguiente fórmula:

$$L_{FS}(dB) = 20\log(4\pi\frac{d}{\lambda}) \quad [2]$$

Donde:

- $\lambda(m)$: $longitud\ de\ onda.$
- $d(m)$: $distancia\ entre\ antenas.$

Conociendo que $\lambda = \frac{c}{f}$ donde $c = 3 \cdot 10^8\ m/s$.

Para un cálculo aproximado de la distancia, en este caso, se va a considerar que no hay pérdidas en el trasmisor ni en el receptor, ni pérdidas diversas en campo.

$$L_{TX}(dB) = L_{RX}(dB) = L_M(dB) = 0$$

Quedándose la ecuación [1] de la siguiente forma:

$$P_{RX} = P_{TX} - G_{TX} - L_{FS} + G_{RX} \qquad [3]$$

Despejando las pérdidas de trayectoria ($L_{FS}\ (dB)$) de la ecuación [1] y despejando la distancia ($d(m)$) de la ecuación [2], obtenemos:

$$L_{FS} = P_{TX} - G_{TX} - P_{RX} + G_{RX} \qquad [4]$$

$$d = 10^{L_{FS} - 20\log\frac{4\pi}{c} - 20\log f} \qquad [5]$$

Por último, se sustituye [4] en [5]:

$$d = 10^{\frac{P_{TX} - G_{TX} - P_{RX} + G_{RX} - 20\log\frac{4\pi}{c} - 20\log f}{20}} \qquad [6]$$

En la Tabla 15 se presentan los valores tomados para el cálculo aproximado de la distancia. Los parámetros de potencia $P_{RX}\ (dBm)$ y $P_{TX}\ (dBm)$: se han obtenido de la información que nos proporciona el fabricante en su *datasheet [29]* del módulo LoRa SX127x. La ganancia de la antena corresponde a las antenas compradas, cuyas especificaciones técnicas son proporcionadas por el fabricante.

*Tabla 15: Parámetros para el cálculo de la distancia entre antenas.*

| Potencia recibida[12] | Potencia de salida del transmisor | Ganancia de la antena transmisora | Ganancia de la antena receptora | Frecuencia |
|---|---|---|---|---|
| $P_{RX}\ (dBm)$ | $P_{TX}\ (dBm)$ | $G_{TX}\ (dBi)$ | $G_{RX}\ (dBi)$ | $f$ (MHz) |
| -123 dBm | -14 dBm | 2 dBi | 2 dBi | 868 MHz |

[12] También conocida como sensibilidad del receptor.

Sustituyendo estos parámetros en la ecuación [6], se consigue obtener un valor teórico de la distancia entre dispositivos y, que serán tenidos en cuenta para la planificación de la red. En la Tabla 16 se muestra la distancia teórica calculada para las antenas disponibles.

*Tabla 16: Distancia de comunicación para diferentes combinaciones de antenas.*

| Antena Transmisora | Antena Receptora | Distancia [6] |
|---|---|---|
| $G_{TX}\ (dBi)$ | $G_{RX}\ (dBi)$ | $d\ (m)$ |
| 2 dBi | 2 dBi | 7751 m ≅ 7,8 km |

Además, existen factores que intervienen en las redes y que pueden ocasionar fallas en el sistema. En la realidad, las interferencias y los obstáculos pueden llegar a reducir la distancia teórica de manera significativa. Por ello, se estudia el comportamiento de la red en una zona con las siguientes características:

- Zona interurbana de la comunidad de Madrid.
- Zona que disponga de un área amplia con existencia de posibles obstáculos (estructuras edificadas, arboles, etc.).
- Zona accesible en un entorno seguro y de propiedad pública.

Teniendo en cuenta estas características se toma la decisión de realizar las pruebas en el campus universitario de Campus Sur debido a que cumple con las características anteriormente listadas. En la Figura 30 se muestra la ubicación seleccionada para colocación de los dispositivos.


*Figura 30: Universidad Politécnica de Madrid, Campus Sur.*

Se toma la decisión de que aunque el alcance de una red LoRa es de varios kilómetros, para este caso de estudio se opta por distancias más pequeñas y manejables. Se dispondrán los dispositivos en la ubicación tal y como se muestra en la Figura 31, donde adicionalmente se indican las distancias de separación entre ellos.

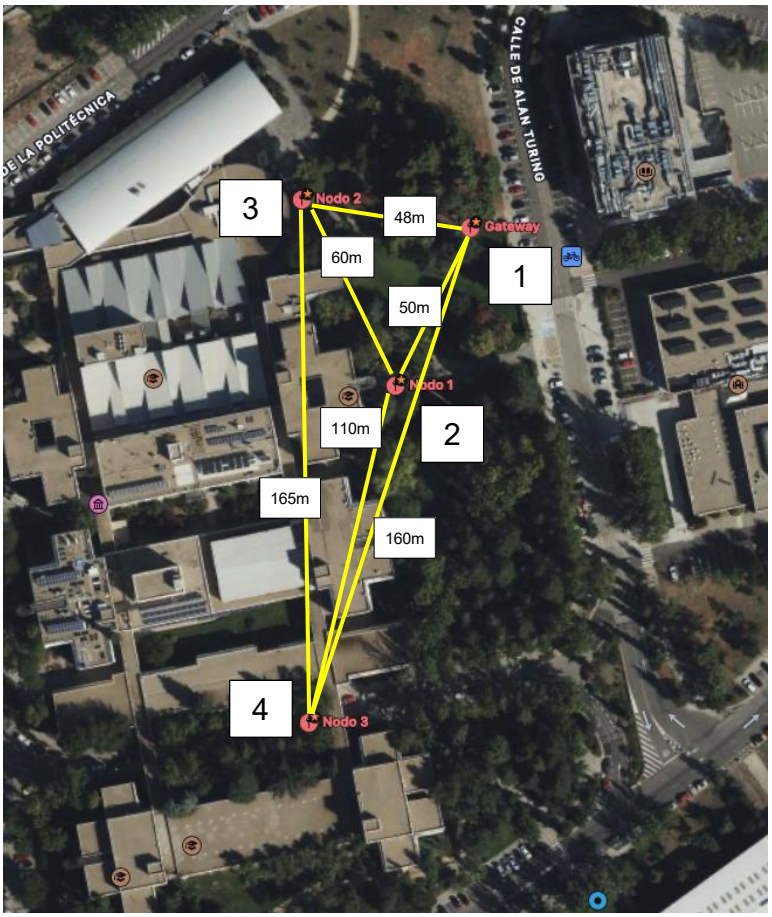


*Figura 31: Planificación de dispositivos en la UPM Campus Sur.*

A continuación se describe la ubicación de los distintos dispositivos:

- Puerta de enlace (ID $8C_H/140_D$ )[1]

Desde la puerta de enlace se monitorea las trazas a través del monitor serie, donde se ve la recepción de mensajes y la transmisión de mediciones al servidor. En la Figura 32 se visualiza la zona de trabajo.

*Figura 32: Pruebas en campo puerta de enlace.*

- Nodo 1 (ID $70_H/112_D$ )[2]

Se encuentra a poca distancia del *gateway* teniendo un enlace directo con este. En este caso, como en el de resto de nodos, se alimenta el prototipo con una batería de tipo LiPo, véase la Figura 33.


*Figura 33: Pruebas en campo Nodo 1.*

- Nodo 2 (ID $E8_H/232_D$ )[3]

Se coloca a poca distancia del *gateway* y con un obstáculo entre ellos para comprobar la conectividad en caso de que existan infraestructuras que puedan intervenir negativamente a la conexión (véase Figura 34).

*Figura 34: Pruebas en campo Nodo 2.*

- Nodo 3 (ID $F4_H/244_D$ )[4]

Colocado a una mayor distancia que los nodos 1 y 2, y entre edificios. Se busca que este nodo no tenga un enlace directo con la puerta de enlace y requiera de un nodo intermedio. En la Figura 35 se muestra su ubicación en campo.

*Figura 35: Pruebas en campo Nodo 3.*

Los resultados que ponen en práctica la planificación de la red se desarrollan en la sección 5.3.

### 4.2.6. Preparación de los prototipos

Para la realización de las pruebas en campo del proyecto se han preparado cuatro prototipos que llevan integrados una placa de desarrollo del fabricante LilyGo, un conector JST para poder conectar una batería de tipo LiPo y, en caso de los nodos, se integra el sensor de temperatura DHT11 con sus respectivas conexiones al microcontrolador.

Para la preparación de los prototipos, se han empleado placas de circuito impreso (PCB) de soldadura de doble cara, véase la Figura 36. Este tipo de placas ofrece varias ventajas, como una mayor durabilidad y la posibilidad de conexiones más seguras y estables, siendo muy útil en proyectos que requieren de robustez y confiabilidad. Además, las placas de doble cara permiten una distribución más eficiente de los componentes, optimizando el espacio y facilitando el proceso de ensamblaje.

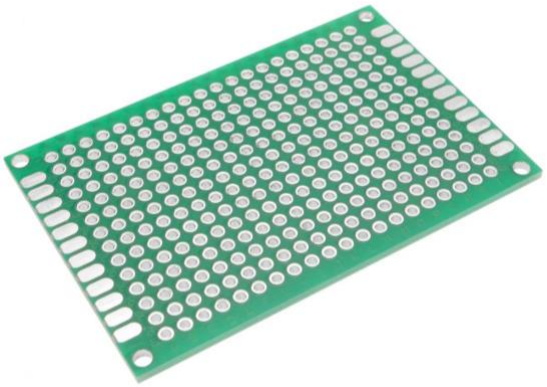

*Figura 36: Placa PCB de prototipo de circuito impreso de soldadura de doble cara.*

Para el montaje de los componentes en las placas, se han utilizado zócalos tanto para la placa principal como para el sensor de temperatura y humedad DHT11, véase Figura 37. Los zócalos son esenciales, ya que proporcionan una conexión segura y extraíble para los componentes. Esto no solo facilita el proceso de montaje y soldadura, sino que también permite sustituir los componentes en caso de fallo o necesidad de actualización, sin tener que desoldar y volver a soldar directamente sobre la PCB.

*Figura 37: Conector hembra de 40 pines PCB.*

El uso de zócalos para el DHT11 es particularmente ventajoso, ya que este sensor es sensible a daños por calor durante el proceso de soldadura. Al insertar el sensor en un zócalo previamente soldado, se minimiza el riesgo de dañarlo, asegurando su correcto funcionamiento y prolongando su vida útil. Además, los zócalos permiten una fácil instalación y extracción del sensor, lo que facilita el mantenimiento y posibles ajustes o reemplazos futuros.

En cuanto a la placa principal, los zócalos permiten el montaje del microprocesador TTGO LoRa32 de LilyGo de manera segura y sin riesgo de daño durante el proceso de ensamblaje. Esto es crucial para mantener la integridad de los componentes electrónicos y asegurar que el prototipo funcione de manera óptima.

Además de los zócalos, se ha incorporado un conector JST para la alimentación de las placas mediante una batería de tipo LiPo (polímero de litio), véase Figura 38. Los conectores JST son ampliamente utilizados en proyectos electrónicos debido a su capacidad para proporcionar conexiones fiables y seguras para fuentes de energía. La utilización de una batería LiPo de 3.7V y 1000mAh permite que el prototipo sea portátil, aumentando su versatilidad y autonomía. La elección del conector JST garantiza una fácil conexión y desconexión de la batería, simplificando el proceso de recarga y sustitución, así como minimiza el riesgo de conexiones inestables que podrían afectar el funcionamiento del prototipo.

*Figura 38: Conector macho para PCB ángulo de 90º JST.*

En resumen, la utilización de placas con agujeros para soldar de doble cara, junto con zócalos para los componentes principales y un conector JST para la alimentación con batería LiPo, ha permitido un montaje eficiente, seguro y fácilmente mantenible de los prototipos. Este enfoque no solo optimiza el proceso de ensamblaje, sino que también garantiza la durabilidad y fiabilidad del diseño final.

#### 4.2.6.1. Montaje de los prototipos

Para proporcionar una visión más clara del proceso de montaje de los prototipos, a continuación, se presentan una serie de imágenes que ilustran cada paso del ensamblaje. Estas imágenes muestran detalladamente cómo se han montado los componentes en las placas PCB de doble cara, incluyendo la inserción de los zócalos, la conexión del sensor DHT11, y la integración del conector JST para la batería LiPo.

En primer lugar, se preparan los materiales que se utilizarán para el montaje de los prototipos. Esto incluye la selección de la placa PCB de doble cara con agujeros para soldar (1), los zócalos para los componentes (2), el conector JST (3), los modelos TTGO LoRa32 (4 y 5), la batería LiPo (6) y el sensor (7). Los materiales se muestran en las figuras Figura 39 y Figura 40.

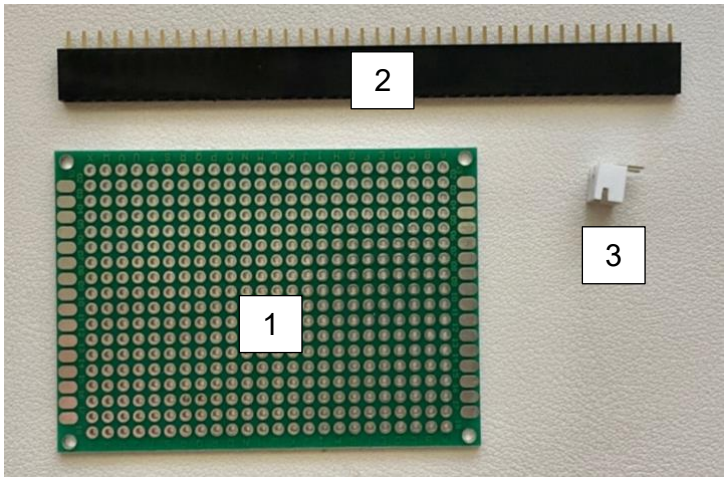


*Figura 39: Placa PCB (1), zócalos (2) y conector JST (3).*

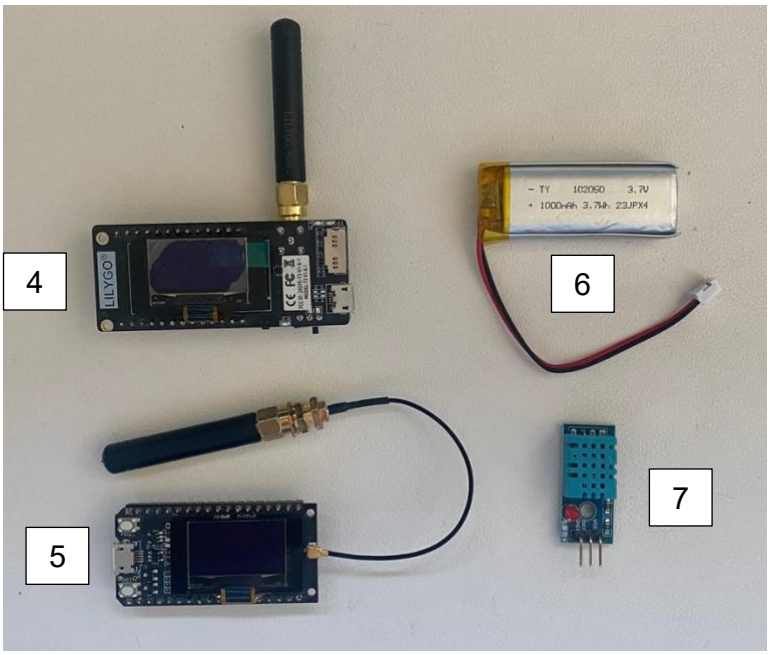

*Figura 40: Modelos TTGO LoRa32 (4 y 5), batería LiPo (6) y sensor DHT11 (7).*

A continuación, se presentan las tablas de conexiones que tomarán los prototipos diferenciando si se trata de una puerta de enlace o un nodo de transmisión de medidas. Además, se especifican las conexiones según el modelo de placa de desarrollo empleada.

- Prototipo de la puerta de enlace o *gateway*.

Para el prototipo de la puerta de enlace, que en adelante llamaremos *Gateway,* se utiliza la placa de desarrollo de LilyGo TTGO LoRa en su versión 1. En este caso, únicamente se conecta la alimentación mediante un cable que posee un conector hembra JST de 2 pines y 1.25mm a la PCB. A través de un camino de estaño, se lleva la conexión al conector macho de JST de 2 pines y 2mm. En las figuras Figura 41 y Figura 42 se muestra como ha quedado finalmente ensamblado el prototipo *Gateway*.

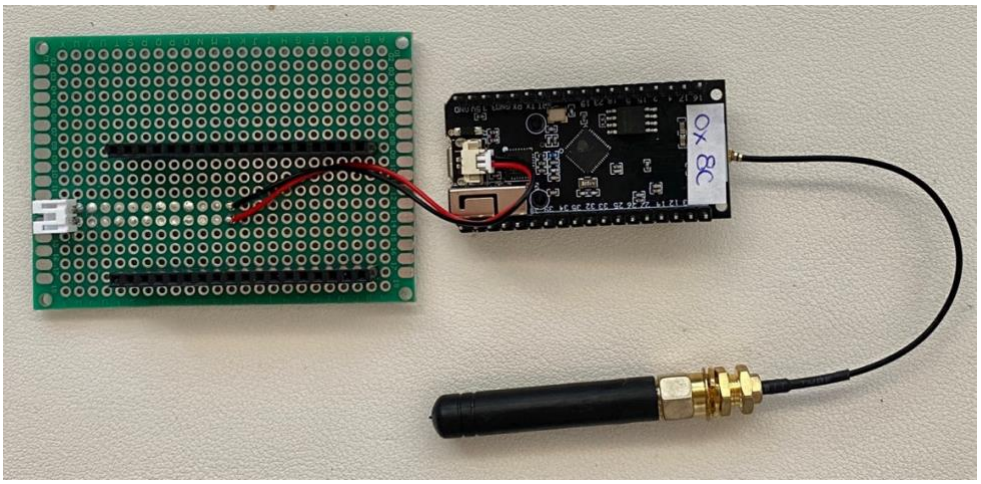

*Figura 41: Conexión de la alimentación.*

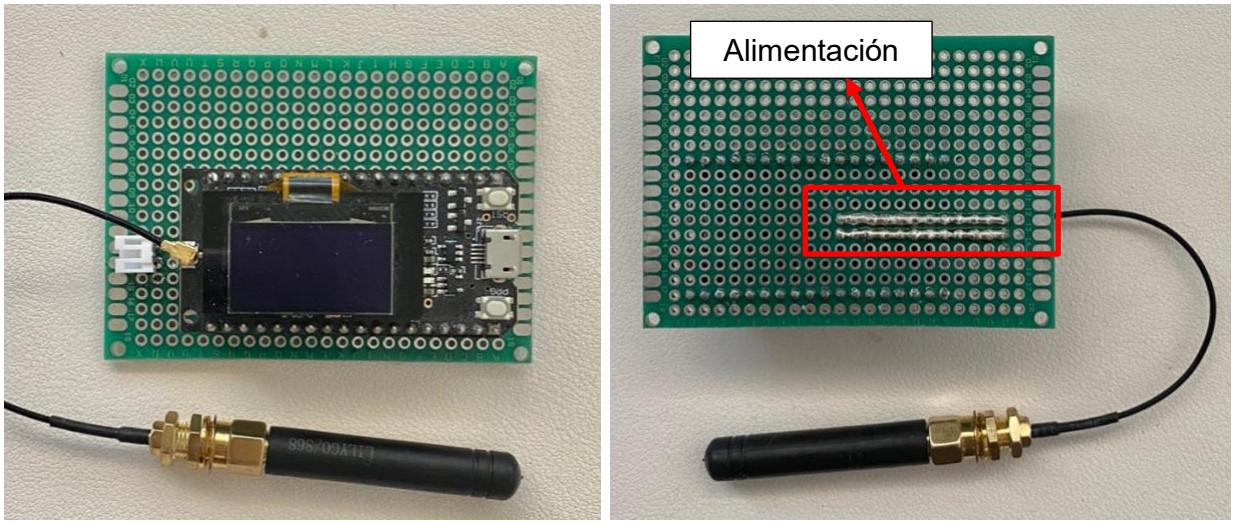


*Figura 42: Vista superior(izquierda) e inferior(derecha) del prototipo "Gateway".*

- Prototipo del nodo.

Para los tres prototipos que tienen la función de los nodos transmisores , que en adelante los llamaremos MIZ00X, donde X se refiere al número del nodo. Uno de ellos utiliza la placa de desarrollo de LilyGo TTGO LoRa32 en su versión 1, mientras que los otros dos prototipos emplean la placa de desarrollo de LilyGo TTGO LoRa32 en su versión 2.1-1.6. En ambas versiones, se realiza la misma conexión de alimentación que se ha descrito anteriormente en el *Gateway*. Esto se muestra en la Figura 43.

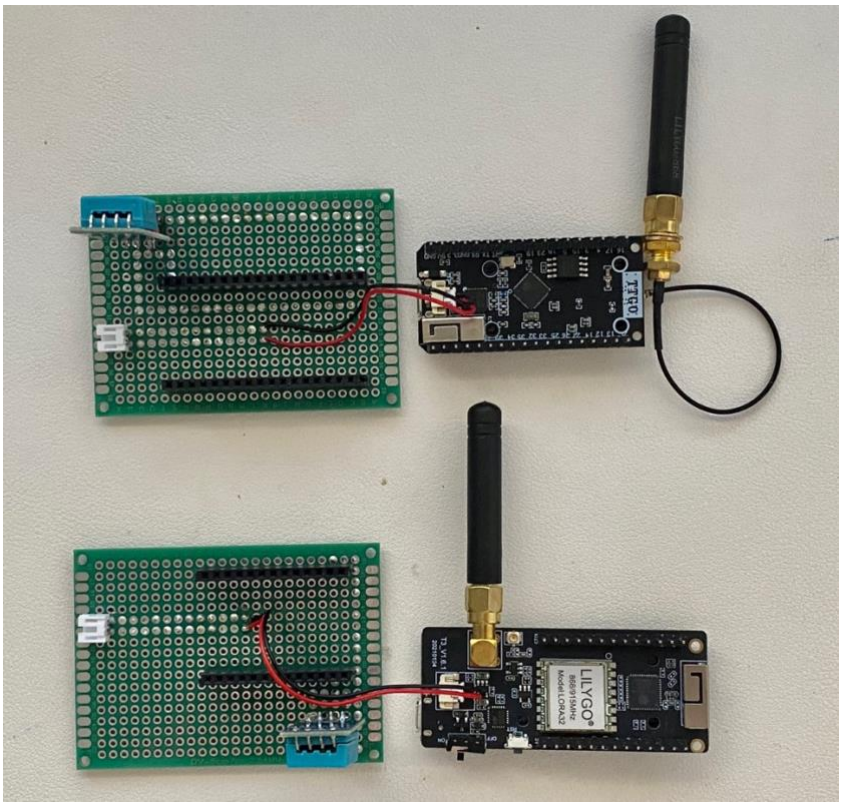

*Figura 43: Conexión de la alimentación TTGO LoRa32 versión 1 (arriba) y versión 2.1-1.6 (abajo).*

Además, a estos prototipos se les conecta un sensor DHT11 para la medición de temperatura y humedad. En la Tabla 17 se muestra las conexiones de DHT11 con la versión 1 de la placa de desarrollo, en las figuras Figura 44 y Figura 45 se muestra cómo queda finalmente ensamblado.

*Tabla 17: Conexionado prototipo MIZ001.*

| Pines TTGO LoRa v1 | Pines DHT11 |
|---|---|
| 5V | VCC |
| GND | GND |

| | |
|---|---|
| Pin 16 – GPIO16 | DATA |

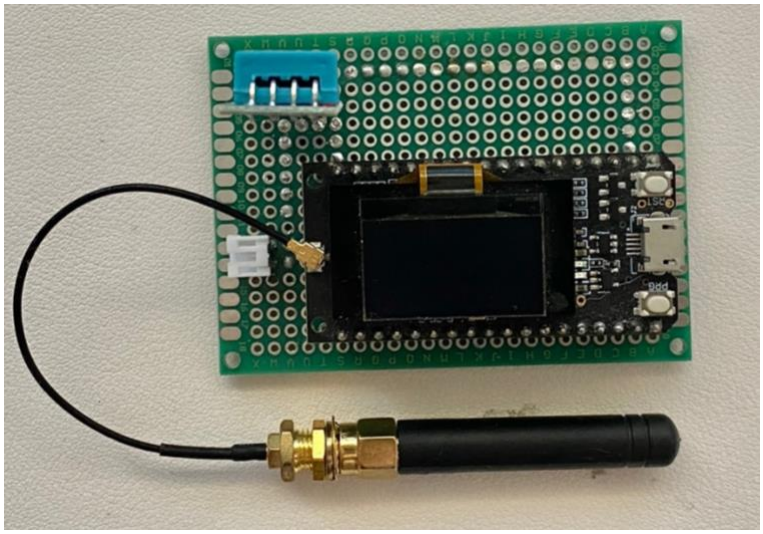

*Figura 44: Vista superior del prototipo "MIZ001".*

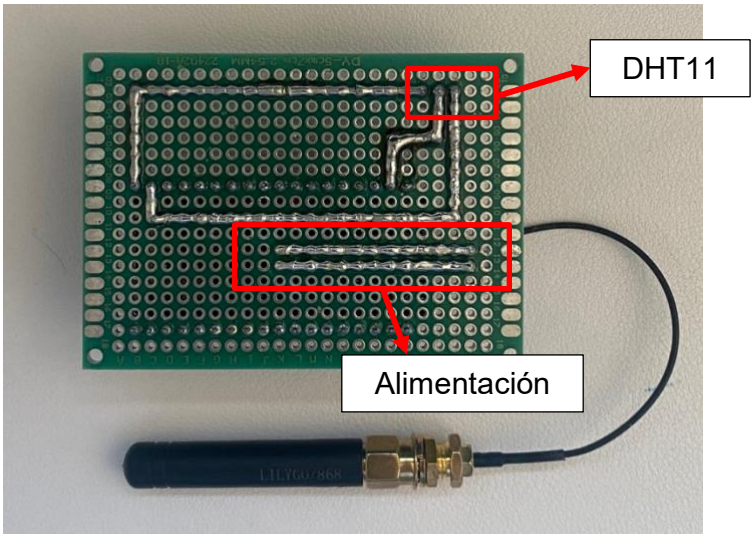


*Figura 45: Vista inferior del prototipo "MIZ001".*

En la Tabla 18 se muestra las conexiones de DHT11 con la versión 2.1-1.6 de la placa de desarrollo, en las figuras Figura 46 y Figura 47 se muestra cómo queda finalmente ensamblado.

*Tabla 18: Conexionado de los prototipos "MIZ002" y "MIZ003".*

| Pines TTGO LoRa v1 | Pines DHT11 |
|---|---|
| 5V | VCC |
| GND | GND |

| | |
|---|---|
| Pin 16 – GPIO16 | DATA |


*Figura 46: Vista superior del prototipo “MIZ002” y "MIZ003".*

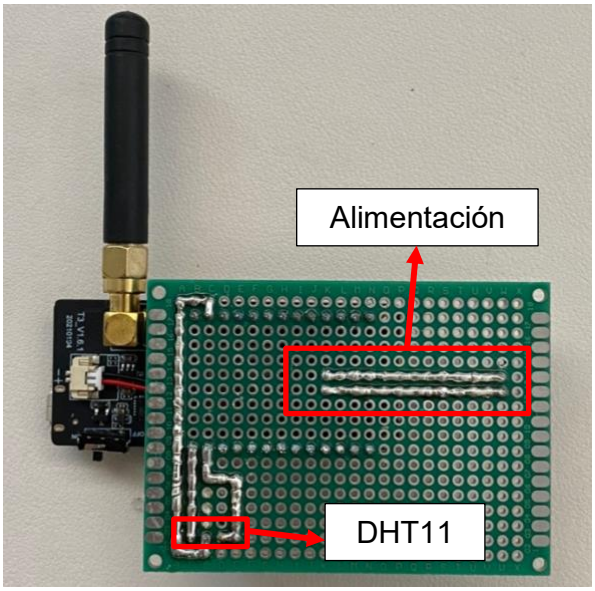


*Figura 47: Vista inferior del prototipo “MIZ002” y "MIZ003".*

# 5. Resultados

Para llevar a cabo el desarrollo del proyecto, se ha implementado una serie de pruebas. Inicialmente, se comenzó con un sistema simple compuesto por un único gateway y un nodo, los cuales se comunican de manera directa; a esta configuración se la denomina prueba punto a punto.

Posteriormente, con el fin de avanzar hacia el desarrollo de una red más extensa, se procederá a probar un algoritmo diseñado para establecer las tablas de enrutamiento. Esta prueba inicial de obtención de tablas de enrutamiento se llevará a cabo en el sistema simple descrito anteriormente para asegurar su precisión y eficacia en un entorno controlado.

Finalmente, se procederá a ampliar la red. En esta fase se utilizará un total de tres nodos y un gateway para evaluar el rendimiento del sistema y la correcta implementación de las tablas de enrutamiento en un entorno más complejo. Estas pruebas permitirán identificar y resolver posibles problemas durante la implementación del sistema, facilitando un control de errores más sencillo.

## 5.1. Prueba de una Red Punto a Punto

En esta sección, se detalla la prueba denominada punto a punto, cuyo propósito es validar la comunicación básica entre un gateway y un nodo. En la Figura 48, se presenta el diagrama secuencial que describe el procedimiento implementado para esta prueba.

El objetivo principal de esta prueba es verificar la integridad y funcionalidad de la comunicación directa entre los dispositivos involucrados. Esta etapa inicial es fundamental para asegurar que los componentes del sistema operan correctamente antes de proceder a pruebas más complejas y a la implementación de algoritmos de enrutamiento.

El diagrama secuencial proporciona una representación visual del flujo de comunicación y las interacciones específicas que ocurren entre el gateway y el nodo.

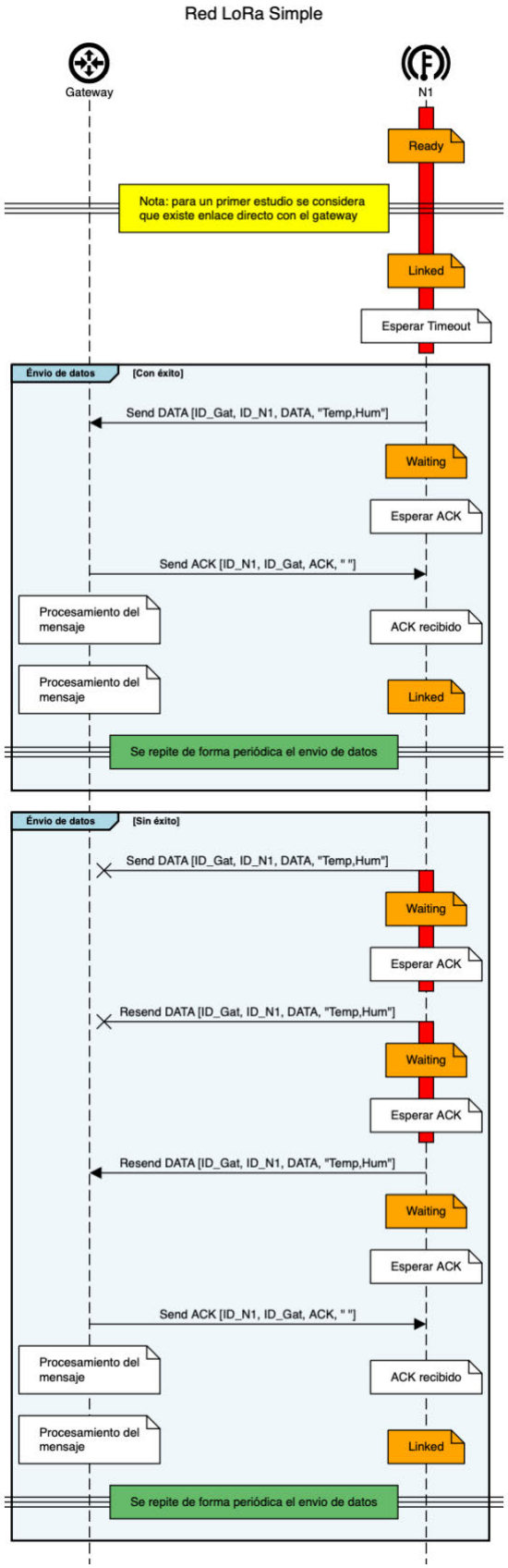


*Figura 48: Diagrama secuencial prueba punto a punto.*

La Figura 48 describe el proceso de envío de datos entre un nodo N1 y un *gateway* en una red LoRa simple. El diagrama se divide en dos escenarios hipotéticos que pueden ocurrir durante la ejecución: un envío con éxito y un envío sin éxito.

Una vez se inicia el sistema, el *gateway* se mantiene en constante escucha, mientras que el nodo (N1) se encuentra en el estado denominado *Ready*. Para este caso de estudio, se asume que existe un enlace directo con el *gateway*, por lo que no es necesario establecer una ruta para llegar a este. De este modo, el nodo pasa directamente al siguiente estado *Linked*. A continuación, se analizan los dos escenarios hipotéticos:

- **Envío de datos con éxito.** Este escenario representa el funcionamiento ideal del sistema, es decir, que no existan pérdidas de información durante la transmisión. El nodo N1, una vez toma los datos del sensor, los envía y permanece en el estado *Waiting* hasta que recibe el ACK de respuesta del *gateway*. Una vez recibido el ACK, el nodo retorna al estado *Linked*. Por su parte, el *gateway* envía el mensaje de confirmación de la recepción de los datos (ACK) y transmite estos datos a la plataforma Web.
- **Envío de datos sin éxito.** Este escenario refleja un funcionamiento más realista del sistema, ya que, debido a interferencias que pueden producirse durante el proceso de envío de datos, estos se pueden perder y no llegar a su destino. Este problema se resuelve mediante la comprobación del ACK. Si el ACK no se recibe en un periodo corto de tiempo, el último dato tomado se reenvía hasta que se reciba la confirmación de recepción (ACK).

El código desarrollado para este proyecto se encuentra disponible en un repositorio de GitHub, el cual puede acceder desde el siguiente enlace: https://github.com/mariaizuz98/LoRa-PointToPoint-Nazaries.

Para garantizar el correcto funcionamiento del código, se han utilizado sentencias ***print*** que permiten verificar la correcta ejecución de los procesos. A continuación, se presentan algunas imágenes que muestran los resultados obtenidos al ejecutar el código:

- Envío de datos con éxito.

Las figuras Figura 49 y Figura 50 muestran el monitor serie del nodo con identificador 0xF4 (244) y del *gateway* con identificador 0x8C (140), respectivamente. En estas figuras se presenta el proceso de obtención de información del sensor por parte del nodo y el envío de dicha información al *gateway* con identificador 0x8C (140). En la Figura 50 se muestra como el *gateway* recibe el mensaje con la información del sensor y confirma la recepción enviando un mensaje ACK (Message ID: 4), que es finalmente recibido por el nodo.

```
17:43:25:196 -> * Temperature (ºC):  26.10  |  Humidity:  36.00
17:43:25:196 ->  ··· Message LoRa send...
17:43:25:262 -> Sender: 0xF4 |  Destination: 0x8C  |  Message ID: 8  |  Message: 261;360
17:43:25:409 ->  ··· Message LoRa receive...
17:43:25:409 -> Sender: 0x8C |  Destination: 0xF4  |  Message ID: 4  |  Message:   |  RSSI:  -53  | SNR:  6.75
```

*Figura 49: Monitor serie nodo ID: 0xF4.*

```
17:43:25:235 ->  ··· Message LoRa receive...
17:43:25:235 -> Sender: 0xF4 |  Destination: 0x8C  |  Message ID: 8  |  Message: 261;360  |  RSSI:  -50  | SNR:  9.00
17:43:25:390 ->  ··· Message LoRa send...
17:43:25:390 -> Sender: 0x8C |  Destination: 0xF4  |  Message ID: 4  |  Message:
```

*Figura 50: Monitor serie gateway ID: 0x8C.*

- Envío de datos sin éxito.

En este caso, se busca comprobar la fiabilidad de la comunicación, asegurando que no se pierde información durante el proceso. En caso de que el *gateway* no reciba los datos del sensor, estos se reenvían hasta que se confirme su recepción. En la Figura 51 se muestra cómo el nodo reenvía los datos del sensor de manera repetitiva hasta que se recibe el mensaje ACK (Message ID: 4).

```
17:49:27:191 -> * Temperature (ºC):  26.30  |  Humidity:  36.00
17:49:27:191 ->  ··· Message LoRa send...
17:49:27:257 -> Sender: 0xF4 |  Destination: 0x8C  |  Message ID: 8  |  Message: 263;360
17:50:23:229 ->  --> Se envia de nuevo el paquete con los datos.
17:50:23:229 ->
17:50:23:229 ->  ··· Message LoRa send...
17:50:23:296 -> Sender: 0xF4 |  Destination: 0x8C  |  Message ID: 8  |  Message: 263;360
17:51:23:268 ->  --> Se envia de nuevo el paquete con los datos.
17:51:23:268 ->
17:51:23:268 ->  ··· Message LoRa send...
17:51:23:334 -> Sender: 0xF4 |  Destination: 0x8C  |  Message ID: 8  |  Message: 263;360
17:51:23:510 ->  ··· Message LoRa receive...
17:51:23:510 -> Sender: 0x8C |  Destination: 0xF4  |  Message ID: 4  |  Message:   |  RSSI:  -51  | SNR:  6.75
```

*Figura 51: Monitor serie nodo ID: 0xF4.*

```
17:51:23:320 ->  ··· Message LoRa receive...
17:51:23:320 -> Sender: 0xF4 |  Destination: 0x8C  |  Message ID: 8  |  Message: 263;360  |  RSSI:  -47  | SNR:  8.75
17:51:23:471 ->  ··· Message LoRa send...
17:51:23:471 -> Sender: 0x8C |  Destination: 0xF4  |  Message ID: 4  |  Message:
```

*Figura 52: Monitor serie gateway ID: 0x8C.*

Se ha verificado que la página web se actualiza correctamente y muestra los datos en una gráfica. A continuación, en las figuras Figura 53 y Figura 54 se incluye una imagen de la página web actualizada con los datos más recientes:

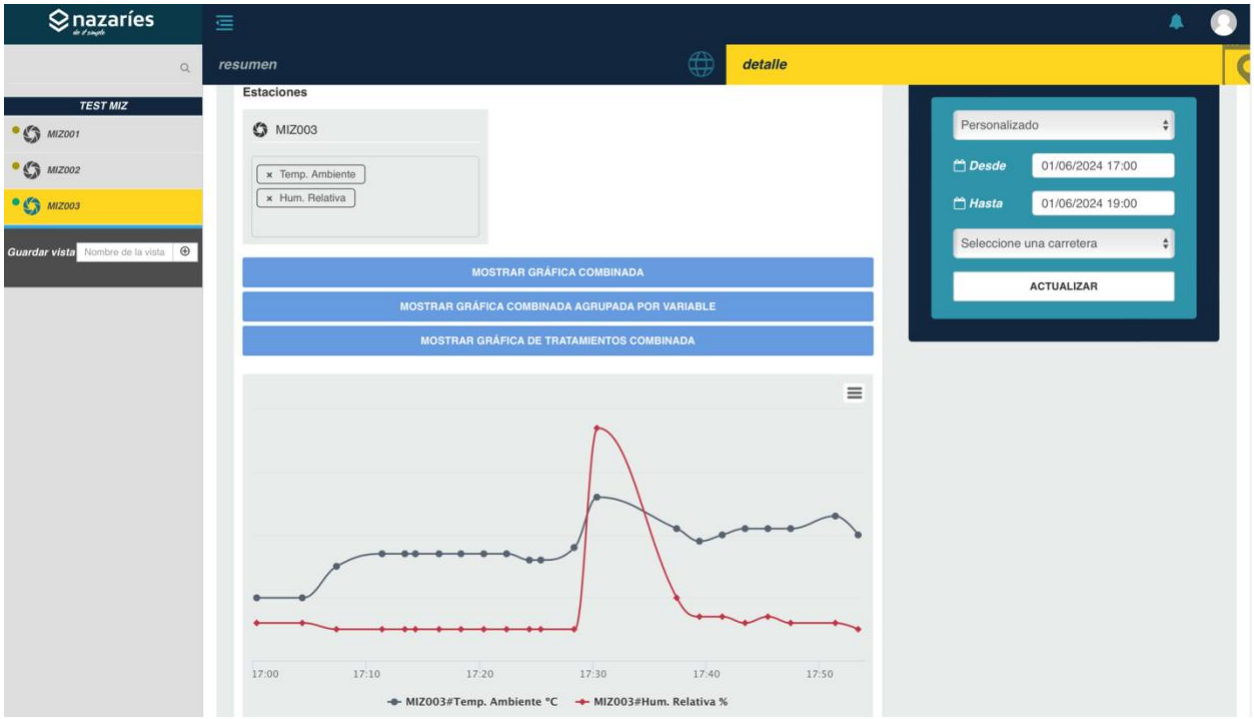


*Figura 53: Vista de la página Web.*

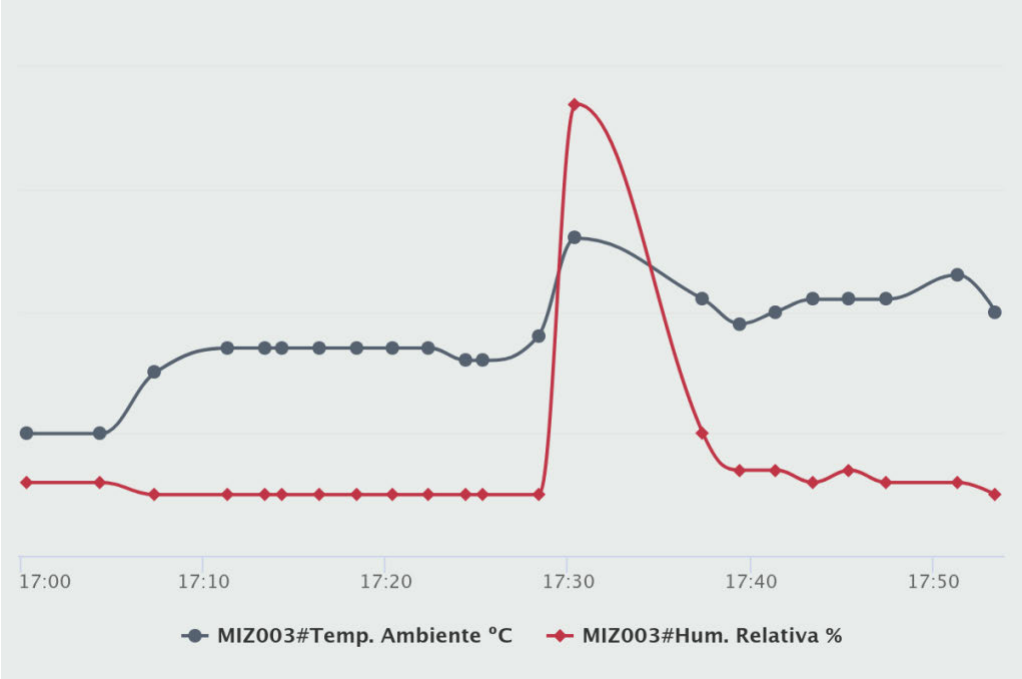


*Figura 54: Gráfica temperatura y humedad del nodo MIZ003.*

Estas pruebas aseguran que tanto el proceso de envío y recepción de datos, como la visualización en la página web funcionan según lo esperado.

Adicionalmente, en el Anexo III: Código prueba punto a punto se incluyen partes del código relevantes para este caso de prueba.

## 5.2. Prueba del Algoritmo de Enrutamiento con una Red Simple

En esta sección se presenta la prueba del algoritmo diseñado para establecer las tablas de enrutamiento, utilizando una red simple compuesta por un único gateway y un nodo. El objetivo de esta prueba es validar la precisión y eficacia del algoritmo en un entorno controlado y de baja complejidad antes de su implementación en una red más extensa. En las figuras Figura 55 y Figura 56 se presenta el diagrama secuencial que describe el procedimiento implementado para esta prueba.

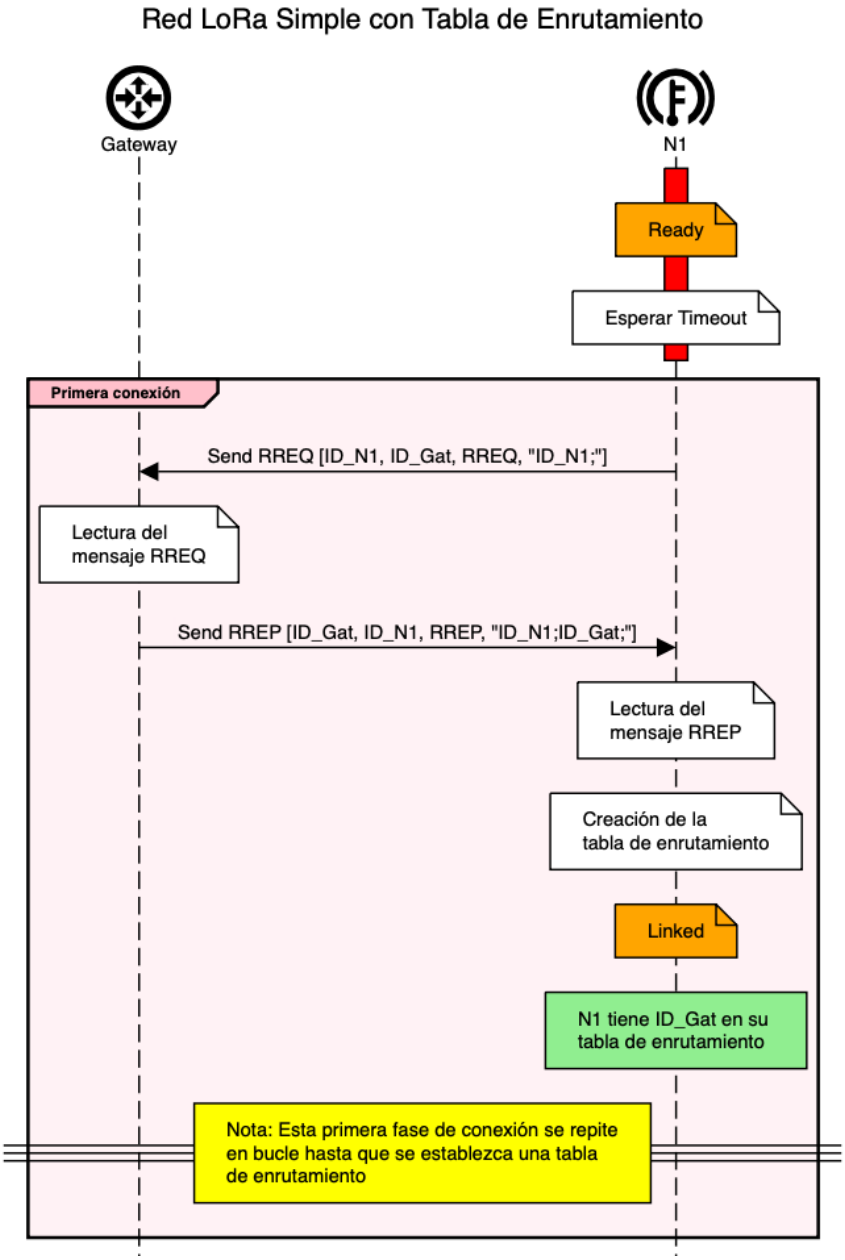


*Figura 55: Diagrama secuencial prueba Tabla de Enrutamiento con una Red Simple (Parte 1).*

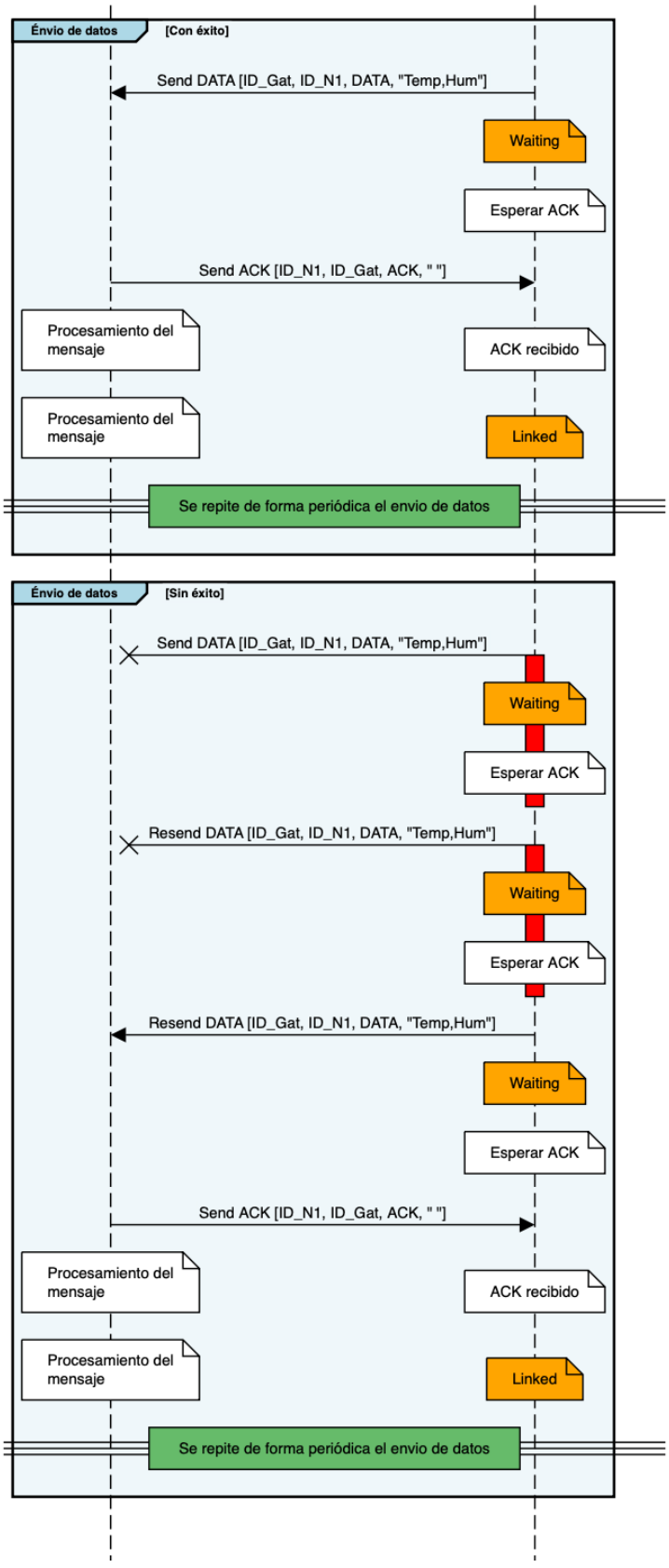


*Figura 56: Diagrama secuencial prueba Tabla de Enrutamiento con una Red Simple (Parte 2).*

En la Figura 55 se presenta el inicio del sistema, en el que durante la primera conexión, se establece la tabla de enrutamiento del nodo haciendo uso del algoritmo de enrutamiento AODV. Cuando el nodo N1 desea transmitir información, una vez pasado el tiempo de *time out*, primero verifica si dispone de una tabla de enrutamiento actualizada. En caso contrario, el nodo N1 envía un mensaje RREQ a todos los nodos vecinos y espera a recibir un mensaje RREP. Una vez recibido el RREP, el nodo analiza el indicador de intensidad de la señal recibida (RSSI, por sus siglas en inglés, *Received Signal Strength Indicator*) y determina la ruta con el objetivo de llegar al *gateway*, que para este caso de prueba tiene un enlace directo. Una vez se establece la tabla de enrutamiento, se cambia y pasa al siguiente estado, *Linked*.

Por otro lado, en la Figura 56 se repite el mismo proceso de envío de información entre en nodo y el *gateway* que se explica en el caso de prueba del apartado 5.1.

Como se explica en la sección 4.2.1, al inicio del sistema se crea una tabla de enrutamiento vacía, esto se muestra en la Figura 57 y se presenta en el terminal serie de la siguiente forma:

```
* Creating Device ID... Chip ID: 12290792... ID device: 0xE8
* Initializing Display... Display OK
* Initializing LoRa... LoRa OK
* Initializing Timers... Timers OK
* Initializing DHT11... DHT11 OK
*** Creating Route Table... Route Table(dest,nextHop,countHop,sequenceRoute):(  140  |  0  |  0  |     )
* LoRa Node... ID: 0xE8
```

*Figura 57: Creación de tabla de enrutamiento.*

Además, en las figuras Figura 58 y Figura 59 se muestra el proceso de establecimiento de la ruta entre el nodo y la puerta de enlace, y la actualización de la ruta en el nodo. Como se explica al inicio del apartado 5.2 se realiza el intercambio de mensajes RREQ y RREP entre el nodo que desea transmitir la información y el nodo vecino o puerta de enlace, hasta poder actualizar la tabla de enrutamiento.

```
19:34:14:490 ->  --- Message LoRa send RREQ ---
19:34:14:490 -> Sender: 0xE8 |  Destination: 0x8C  |  Message ID: RREQ  |  Message: 232;
19:34:14:722 ->  --- Message LoRa receive RREP ---
19:34:14:722 -> Sender: 0x8C |  Destination: 0xE8  |  Message ID: RREP  |  Message: 232;140;  |  RSSI:  -52  | SNR:  9.50
19:34:14:849 ->  *** Updating Route Table... Route Table(dest,nextHop,countHop,sequenceRoute):(  140  |  140  |  1  |  232;140;  )
```

*Figura 58: Intercambio de mensaje RREQ y RREP entre nodo y gateway (terminal serie del nodo).*

```
19:34:14:501 ->  --- Message LoRa receive RREQ ---
19:34:14:501 -> Sender: 0xE8 |  Destination: 0x8C  |  Message ID: RREQ  |  Message: 232;  |  RSSI:  -57  | SNR:  8.00
19:34:14:670 ->  --- Message LoRa send RREP ---
19:34:14:670 -> Sender: 0x8C |  Destination: 0xE8  |  Message ID: RREP  |  Message: 232;140;
```

*Figura 59: Intercambio de mensaje RREQ y RREP entre gateway y nodo (terminal serie del gateway).*

Una vez se actualiza la tabla se presenta la siguiente información por el terminal serie y adicionalmente se presenta en el Display del nodo la secuencia de la ruta, como se puede observar en la Figura 60.

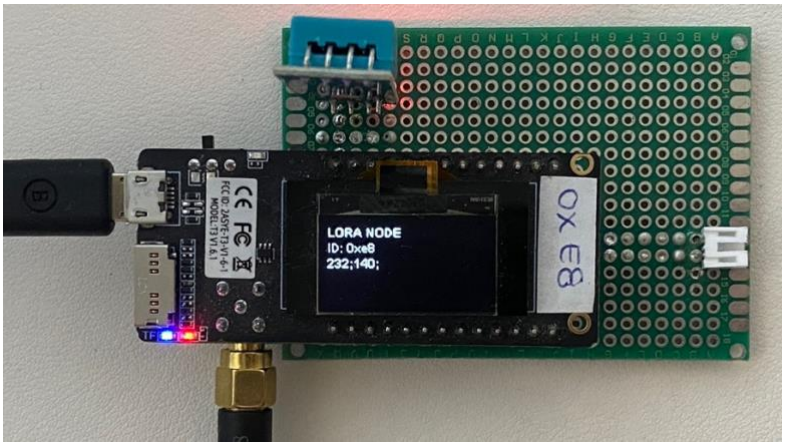


*Figura 60: Información presentada en el Display del nodo.*

La representación en la página web se mantiene igual al del apartado 5.2, ofreciendo gráficas similares a las de las figuras Figura 53 y Figura 54.

El código desarrollado para esta prueba se encuentra disponible en un repositorio de GitHub en su rama principal, a la cual se puede acceder desde el siguiente enlace: https://github.com/mariaizuz98/LoRa_RoutingMesh_Nazaries/tree/master. Adicionalmente, en el Anexo IV: Código prueba de algoritmo de enrutamiento se incluyen partes del código relevantes para este caso de prueba.

## 5.3. Pruebas en campo: despliegue de dispositivos

Esta última prueba lleva a cabo el despliegue de los dispositivos siguiendo la planificación documentada en el capítulo 4 en la sección 4.2.5. Esta prueba ha sido realizada en la Escuela Técnica Superior de Ingeniería y Sistemas de Telecomunicaciones (ETSIST[13]) y se han colocado los dispositivos tal y como se muestra en la Figura 31.

En la Tabla 19 se indican las tablas de enrutamiento de cada nodo durante la realización de la prueba.

*Tabla 19: Tablas de enrutamiento prueba en campo.*

| ID Nodo | Destino | Siguiente salto | Contador de saltos | Secuencia de la ruta |
|---|---|---|---|---|
| | *dest* | *nextHop* | *hopCount* | *sequenceRoute* |
| $\mathbf{112_D}$ | $140_D$ | $140_D$ | 1 | “112;140;” |
| $\mathbf{232_D}$ | $140_D$ | $140_D$ | 1 | “232;140;” |
| $\mathbf{244_D}$ | $140_D$ | $112_D$ | 2 | “244;112;140;” |

[13] Dirección: Nikola Tesla, s/n, 28031, Madrid.

Por software se ha dado la funcionalidad a los nodos intermedios de transmitir los datos recibidos de otros nodos, lejanos al *gateway*. Y que este tenga conocimiento de quien ha transmitido esta información. Esto se muestra como ejemplo en la Figura 61, donde se ve que el paquete DATA es enviado por el nodo 1 ($ID: 70_H/112_D$) en cambio el gateway envía los datos al servidor direccionándolos al espacio reservado con el nombre de MIZ003 que equivale al nodo 3 ($ID: F4_H/244_D$). Por motivos de confidencialidad con la empresa colaboradora de este proyecto se han suprimido partes de las trazas que se generan en el envío al servidor.

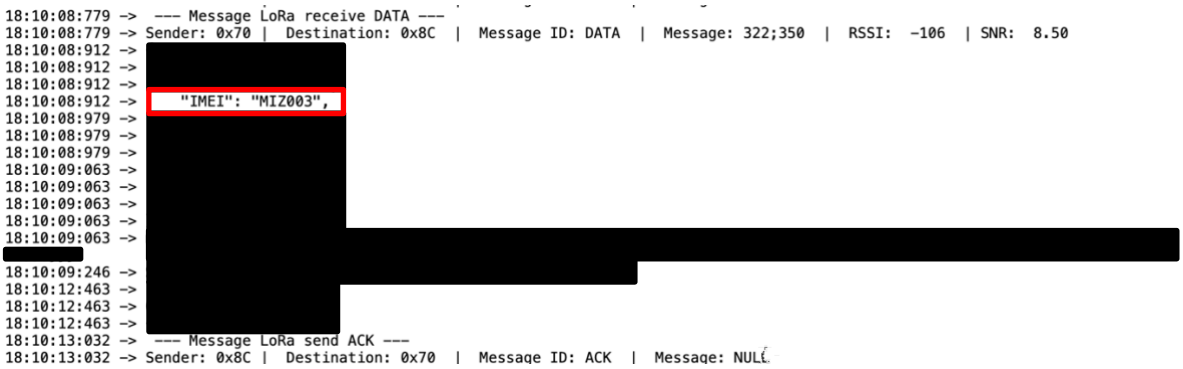


*Figura 61: Ejemplo envío a través de nodo intermedio.*

Adicionalmente, se implementa un proceso de recuperación de la conexión en caso de pérdida de un nodo intermedio. Esto significa que si se intentan realizar 5 envíos seguidos del paquete DATA y no se recibe confirmación (ACK), se considera la ruta actual como perdida. Esto tiene como resultado el reinicio de la tabla de enrutamiento y se genera una nueva solicitud de rutas (RREQ) en busca de un nuevo camino hacia la puerta de enlace. Además, el *gateway* envía a todos los dispositivos que se encuentren a su alcance un mensaje de error en la ruta (RERR). En caso de recibir este mensaje de error, quien lo recibe comprueba si el emisor es su enlace (*nextHop*) con el *gateway*, en caso afirmativo este también reinicia su tabla de enrutado y reenvía el error a los nodos vecinos.

En las figuras mostradas a continuación, Figura 62 y Figura 63, se presentan los gráficos que la plataforma IoT de Nazaríes Intelligenia genera con las muestras tomadas en esta prueba.

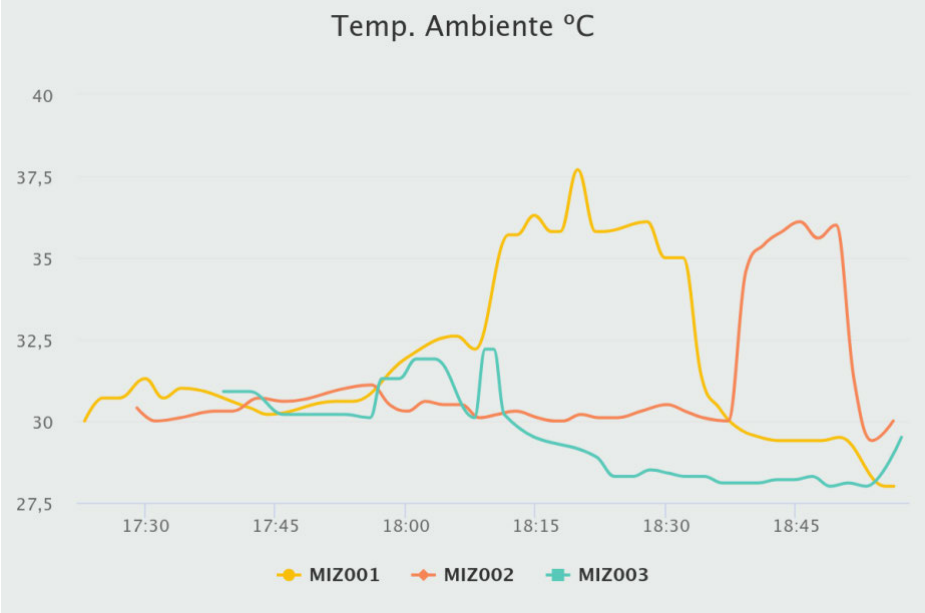


*Figura 62: Gráfica combinada Temperatura Ambiente (ºC).*

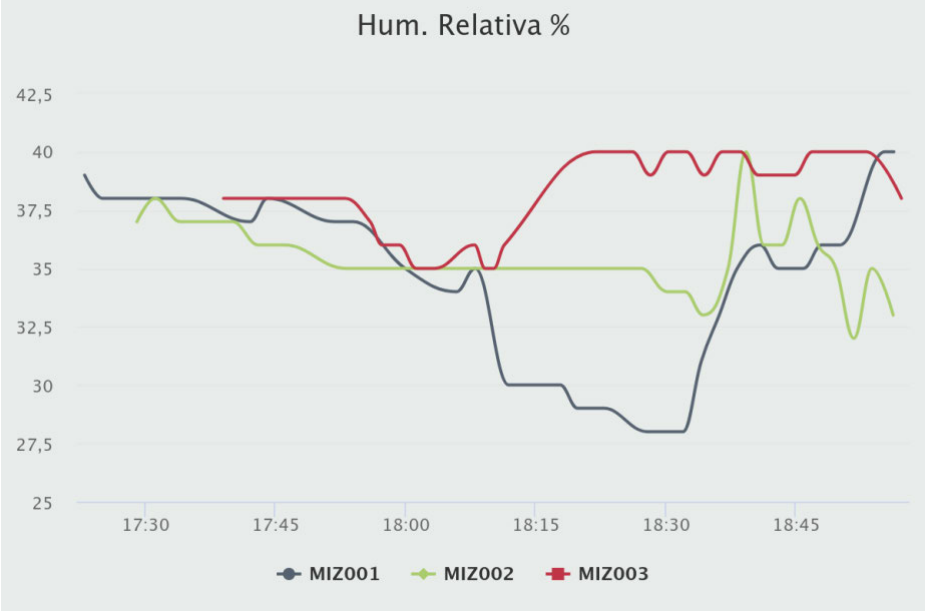


*Figura 63: Gráfica combina Humedad Relativa (%).*

El código desarrollado para esta prueba se encuentra disponible en un repositorio de GitHub en una rama secundaria, a la cual se puede acceder desde el siguiente enlace: https://github.com/mariaizuz98/LoRa_RoutingMesh_Nazaries/tree/v2_routing.

# 6. Presupuesto

En las Tabla 20 y Tabla 21 se indican los costes derivados de la mano de obra para el desarrollo del proyecto y los costes del material utilizado para la realización de este Proyecto de Fin de Grado. Los recursos necesarios para el proyecto han sido proporcionados por la empresa Nazaríes Intelligenia, exceptuando el ordenador portátil.

*Tabla 20: Coste mano de obra para el desarrollo del proyecto.*

| Descripción | N.º de horas | Precio por hora | Importe |
|---|---|---|---|
| **Estudiante** | 320 | 12,00 € | 3.840,00 € |
| **Ingeniero Senior** | 50 | 20 € | 1.000,00 € |
| | **TOTAL** | | 4.840,00 € |

*Tabla 21: Coste material para la implementación del proyecto.*

| Descripción | Cant. | Precio unitario | Importe |
|---|---|---|---|
| MacBook Pro de 13 pulgadas de 2020 | 1 | 1.729,00 € | 1.729,00 € |
| Placa TTGO LORA32 ESP32 con OLED 900 MHz (868 MHz y 915 MHz) | 2 | 24,14 € | 48,28 € |
| Placa TTGO LORA32 V2.1_1.6 (868 MHz) | 2 | 27,03 € | 54,06 € |
| Batería recargable de Polímero de Litio (LiPo) de 3.7V 1000 mAh | 4 | 3,12 € | 12,48 € |
| Kit de Placa PCB | 1 | 14,99 € | 14,99 € |
| Conector macho JST de 2 pines | 4 | 0,19 € | 0,76 € |
| Sensor de Temperatura y Humedad (DHT11) | 3 | 2,80 € | 8,39 € |
| | **TOTAL** | | 1.867,96 € |

Por último, en la Tabla 22 se presenta el presupuesto final del proyecto incluyendo los costes de mano de obra y el coste del material.

*Tabla 22: Presupuesto total del proyecto.*

| Descripción | Importe |
|---|---|
| Coste Mano de Obra | 4.840,00 € |
| Coste Material | 1,867,96 € |
| **TOTAL** | 6.707,96 € |

# 7. Impacto del proyecto

Con el auge de las tecnologías inalámbricas en la actualidad y teniendo en cuenta la importancia que hoy en día se le da a la sostenibilidad y a la optimización en la obtención de recursos, la idea de una red de sensores con tecnología LoRa se presenta como una oportunidad para mejorar el entorno. Además, en las ciudades, tanto grandes como pequeñas, se busca tener un control óptimo de la infraestructura con el objetivo de mejorar la calidad de vida de los ciudadanos.

Con el objetivo de concienciar a las personas con el cuidado de nuestro planeta, líderes mundiales establecieron el 25 de septiembre de 2015 [46] 17 objetivos llamados Objetivos de Desarrollo Sostenible. Estos objetivos tienen como fecha límite el año 2030, con la meta de ayudar al planeta. Este proyecto busca poder contribuir a la sociedad teniendo un vínculo directo con estos objetivos.


*Figura 64: Objetivos de la agenda 2030 [46].*

La implementación de esta idea establece una relación directa con varios de los propósitos que se observan en la Figura 64. Como se explica en el capítulo 4 sección 4.1.1.2, se trata de un proyecto flexible en cuanto a la elección de sensores. Esto significa que la red se mantiene fija, adaptando únicamente los datos de envío con una misma estructura, lo que permite el uso de la red en cualquier ámbito y su adaptación a los Objetivos de Desarrollo Sostenible. A continuación, se muestran algunos ejemplos de aplicaciones.

- La instalación de sensores que controlan la calidad del agua en ríos o sistemas de suministro de agua puede ayudar a la detección de sustancias contaminantes y garantizar el mantenimiento de los estándares de pureza. Esto previene ante futuros problemas de salud en la sociedad y asegura un suministro de agua potable. (ODS 6: Agua limpia y saneamiento).

- La implementación de sensores de detección de movimiento en los sistemas de alumbrado o la monitorización de la situación de tráfico en las ciudades aporta eficiencia energética y una mejora en la calidad de vida urbana. (ODS 11: Ciudades y comunidades sostenibles)
- La colocación de sensores ambientales en áreas urbanas y rurales capaces de medir la calidad del aire y los niveles de polución. Proporcionando estadísticas en tiempo real que facilitan la toma de decisiones para el control de los efectos negativos de la contaminación. Consiguiendo mejorar la salud pública y un entorno saludable. (ODS 13: Acción por el clima).

# 8. Conclusiones y propuesta de mejoras

En esta sección, una vez se han expuesto los resultados obtenidos en el proyecto, se discuten los resultados y se reflexiona sobre los desafíos superados durante el desarrollo. Además, se proponen posibles mejoras y futuras líneas de investigación.

## 8.1. Conclusiones

Con la finalización del proyecto titulado “Sensorización en Smart Cities usando comunicación LoRa” se confirma el cumplimiento de los objetivos marcados en el capítulo 1 sección 1.2. A continuación, se destacan los logros obtenidos durante la realización de este proyecto:

- Se ha llevado a cabo una investigación en profundidad sobre la tecnología IoT y el concepto de Smart City. Además, se ha adquirido conocimiento sobre las redes inalámbricas y todo lo que esto implica.
- Se llega a implementar una red de comunicación simple, entre un nodo y un gateway de forma satisfactoria.
- Se diseña una red con una topología de tipo malla, conocida como red *Mesh*, utilizando un total de cuatro dispositivos, desde cero. Logrado gracias a la implementación de un algoritmo de enrutamiento basado en tablas de enrutado.
- La red has sido diseñada con el objetivo de establecer una comunicación robusta y estable, capaz de adaptarse a posibles cambios en la topología de la red o fallos en los dispositivos.
- Se ha realizado una distribución de estos dispositivos en una zona de pruebas controlada y manejable para el estudio en campo.
- Las mediciones tomadas por los sensores son enviadas a un Servidor Web el cual almacena los datos y los presenta en la plataforma Web de Nazaríes Intelligenia. Con la posibilidad de generar gráficas o informes para su posterior análisis.

Un punto importante que no ha sido posible llevarse a cabo es el estudio del consumo energético de los dispositivos. A pesar de que no era uno de los objetivos establecidos al principio, se trata de un estudio valioso para la optimización de la eficiencia del diseño

## 8.2. Trabajos futuros

Aunque los resultados son satisfactorios y se ha cumplido con los objetivos marcados, existen mejoras o implementaciones futuras para este proyecto. Estas son las siguientes propuestas de mejoras:

- Mediante un estudio del consumo energético se llegará a obtener un conocimiento del consumo real de los dispositivos en la actualidad para poder conseguir a futuro un consumo más eficiente.

- Ampliación de la red realizando pruebas con distancias mayores y un mayor número de dispositivos.
- Dado que los dispositivos utilizados son prototipos para las pruebas de la red, existe la posibilidad de desarrollar dispositivos más robustos empleando materiales profesionales y dándoles una protección, haciendo uso de cajas adecuadas para su uso en el exterior. Dando lugar a un aumento de la durabilidad y la resistencia de los dispositivos.
- Establecimiento de comunicación con el servidor web mediante un protocolo de mensajería, como por ejemplo *Message Queuing Telemetry Transport*, MQTT. Esta mejora puede aportar al sistema una comunicación bidireccional entre los módulos y el servidor permitiendo a los usuarios tener un control a distancia de los dispositivos.

# 9. Referencias

[1] «Nazaríes IoT». Accedido: 28 de abril de 2024. [En línea]. Disponible en: https://iot.nazaries.cloud/

[2] B. Dorsemaine, J.-P. Gaulier, J.-P. Wary, N. Kheir, y P. Urien, «Internet of Things: A Definition & Taxonomy», en *2015 9th International Conference on Next Generation Mobile Applications, Services and Technologies*, Cambridge, United Kingdom: IEEE, sep. 2015, pp. 72-77. doi: 10.1109/NGMAST.2015.71.

[3] «Internet de las cosas», *Wikipedia, la enciclopedia libre*. 3 de octubre de 2023. Accedido: 26 de octubre de 2023. [En línea]. Disponible en: https://es.wikipedia.org/wiki/Internet_de_las_cosas

[4] L. Atzori, A. Iera, y G. Morabito, «The Internet of Things: A survey», *Comput. Netw.*, vol. 54, n.º 15, pp. 2787-2805, oct. 2010, doi: 10.1016/j.comnet.2010.05.010.

[5] I. D. B. Machado, «PROPUESTA DE ARQUITECTURA PARA INTERNET DE LAS COSAS».

[6] P. Sethi y S. R. Sarangi, «Internet of Things: Architectures, Protocols, and Applications», *J. Electr. Comput. Eng.*, vol. 2017, pp. 1-25, 2017, doi: 10.1155/2017/9324035.

[7] A. Vishnivetskaya y E. Alexandrova, «"Smart city" concept. Implementation practice», *IOP Conf. Ser. Mater. Sci. Eng.*, vol. 497, p. 012019, abr. 2019, doi: 10.1088/1757-899X/497/1/012019.

[8] A. S. Syed, D. Sierra-Sosa, A. Kumar, y A. Elmaghraby, «IoT in Smart Cities: A Survey of Technologies, Practices and Challenges», *Smart Cities*, vol. 4, n.º 2, pp. 429-475, mar. 2021, doi: 10.3390/smartcities4020024.

[9] N. Aakvaag y J. Frey, «Redes de sensores inalámbricos», *Rev. ABB ISSN 1013-3135 Nº 2 2006 Pags 39-42*, ene. 2006.

[10] «Telefónica, Ferrovial Servicios y Tellink digitalizan el alumbrado público», El Plural. Accedido: 27 de julio de 2024. [En línea]. Disponible en: https://www.elplural.com/economia/empresas/telefonica-ferrovial-servicios-tellink-digitalizan-alumbrado-publico_247741102

[11] «Topología de red», *Wikipedia, la enciclopedia libre*. 10 de abril de 2024. Accedido: 20 de abril de 2024. [En línea]. Disponible en: https://es.wikipedia.org/w/index.php?title=Topolog%C3%ADa_de_red&oldid=159348293

[12] C. A. Suescún y G. A. M. López, «REVISIÓN DEL ESTADO DEL ARTE DE REDES DE SENSORES INALÁMBRICOS», 2009.

[13] J. Soparia y N. Bhatt, «A Survey on Comparative Study of Wireless Sensor Network Topologies», *Int. J. Comput. Appl.*, vol. 87, n.º 1, pp. 40-43, feb. 2014, doi: 10.5120/15175-3255.

[14] M. I.Gaber, I. I.Mahmoud, O. Seddik, y A. Zekry, «Comparison of Routing Protocols in Wireless Sensor Networks for Monitoring Applications», *Int. J. Comput. Appl.*, vol. 113, n.º 12, pp. 1-7, mar. 2015, doi: 10.5120/19875-1879.

[15] A. Ramasamy Rajeswari, «A Mobile Ad Hoc Network Routing Protocols: A Comparative Study», en *Recent Trends in Communication Networks*, P. Mitra, Ed., IntechOpen, 2020. doi: 10.5772/intechopen.92550.

[16] I. D. Chakeres y E. M. Belding-Royer, «AODV Implementation Design and Performance Evaluation».

[17] «Optimized Link State Routing Protocol», *Wikipedia*. 18 de enero de 2024. Accedido: 3 de julio de 2024. [En línea]. Disponible en: https://en.wikipedia.org/w/index.php?title=Optimized_Link_State_Routing_Protocol&oldid=1196884090

[18] B. S. Chaudhari, M. Zennaro, y S. Borkar, «LPWAN Technologies: Emerging Application Characteristics, Requirements, and Design Considerations», *Future Internet*, vol. 12, n.º 3, p. 46, mar. 2020, doi: 10.3390/fi12030046.

[19] «futureinternet-12-00046-v2.pdf».

[20] M. Islam, H. Jamil, S. Pranto, R. Das, A. Amin, y A. Khan, «Future Industrial Applications: Exploring LPWAN-Driven IoT Protocols», *Sensors*, vol. 24, n.º 8, p. 2509, abr. 2024, doi: 10.3390/s24082509.

[21] «Comunicaciones inalámbricas de largo alcance frente a las de corto alcance: ¿Qué es lo mejor para su proyecto? | Digi International». Accedido: 27 de julio de 2024. [En línea]. Disponible en: https://es.digi.com/blog/post/long-range-vs-short-range-wireless-communications

[22] H. J. Albeyboni y I. A. Ali, «LPWAN TECHNOLOGIES FOR IOT APPLICATIONS: A REVIEW», *J. Duhok Univ.*, vol. 26, n.º 1, Art. n.º 1, abr. 2023, doi: 10.26682/sjuod.2023.26.1.4.

[23] K. Mekki, E. Bajic, F. Chaxel, y F. Meyer, «Overview of Cellular LPWAN Technologies for IoT Deployment: Sigfox, LoRaWAN, and NB-IoT», en *2018 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)*, Athens: IEEE, mar. 2018, pp. 197-202. doi: 10.1109/PERCOMW.2018.8480255.

[24] P. arvindpdmn, «Low-Power Wide-Area Network», Devopedia. Accedido: 9 de julio de 2024. [En línea]. Disponible en: https://devopedia.org/low-power-wide-area-network

[25] «3GPP – The Mobile Broadband Standard», 3GPP. Accedido: 11 de julio de 2024. [En línea]. Disponible en: https://www.3gpp.org/

[26] «Home», Sigfox 0G Technology. Accedido: 11 de julio de 2024. [En línea]. Disponible en: https://www.sigfox.com/

[27] «LoRa Alliance - Homepage», LoRa Alliance®. Accedido: 11 de julio de 2024. [En línea]. Disponible en: https://lora-alliance.org/

[28] «LoRaWAN y su aportación a las tecnologías IIoT | INCIBE-CERT | INCIBE». Accedido: 11 de julio de 2024. [En línea]. Disponible en: https://www.incibe.es/incibe-cert/blog/lorawan-y-su-aportacion-las-tecnologias-iiot

[29] «SX1276». Accedido: 8 de julio de 2024. [En línea]. Disponible en: https://www.semtech.com/products/wireless-rf/lora-connect/sx1276

[30] «LoRa32 V2.1_1.6», LILYGO®. Accedido: 6 de noviembre de 2023. [En línea]. Disponible en: https://www.lilygo.cc/products/lora3

[31] «MKR WAN 1300 | Arduino Documentation». Accedido: 8 de julio de 2024. [En línea]. Disponible en: https://docs.arduino.cc/hardware/mkr-wan-1300/

[32] «WiFi LoRa 32(V3)», Heltec Automation. Accedido: 8 de julio de 2024. [En línea].

Disponible en: https://heltec.org/project/wifi-lora-32-v3/

[33] «Dragino :: Open Source WiFi, Linux Appliance». Accedido: 8 de julio de 2024. [En línea]. Disponible en: https://www.dragino.com/

[34] «Kerlink, IoT connectivity networks, software and services», Kerlink. Accedido: 8 de julio de 2024. [En línea]. Disponible en: https://www.kerlink.com/

[35] «MS48-LR -- LoRaWAN To Modbus Gateway». Accedido: 8 de julio de 2024. [En línea]. Disponible en: https://www.dragino.com/products/lora-lorawan-gateway/item/320-ms48-lr.html

[36] «LoRa/LoRaWAN IoT Kit v3». Accedido: 8 de julio de 2024. [En línea]. Disponible en: https://www.dragino.com/products/lora/item/237-lora-lorawan-iot-kit-v3.html

[37] «Product - Kerlink - Wirnet iFemtoCell-evolution - IoT - LoRa», Kerlink. Accedido: 8 de julio de 2024. [En línea]. Disponible en: https://www.kerlink.com/product/wirnet-ifemtocell-evolution/

[38] «Wirnet iBTS», Kerlink. Accedido: 8 de julio de 2024. [En línea]. Disponible en: https://www.kerlink.com/product/wirnet-ibts/

[39] «LoRa32 V1.0», LILYGO®. Accedido: 6 de noviembre de 2023. [En línea]. Disponible en: https://www.lilygo.cc/products/lora32-v1-0

[40] «Módulo de ruptura DHT11 con sensor de temperatura de placa y cable y sensor de humedad copatible con Arduino y Raspberry Pi», AZ-Delivery. Accedido: 28 de abril de 2024. [En línea]. Disponible en: https://www.az-delivery.de/es/products/dht11-temperatursensor-modul

[41] «Software». Accedido: 28 de abril de 2024. [En línea]. Disponible en: https://www.arduino.cc/en/software

[42] «Visual Studio Code - Code Editing. Redefined». Accedido: 28 de abril de 2024. [En línea]. Disponible en: https://code.visualstudio.com/

[43] PlatformIO, «PlatformIO: Your Gateway to Embedded Software Development Excellence», PlatformIO. Accedido: 28 de abril de 2024. [En línea]. Disponible en: https://platformio.org

[44] «HTTP y HTTPS: diferencia entre los protocolos de transferencia. AWS», Amazon Web Services, Inc. Accedido: 9 de julio de 2024. [En línea]. Disponible en: https://aws.amazon.com/es/compare/the-difference-between-https-and-http/

[45] «Link budget», *Wikipedia*. 31 de diciembre de 2023. Accedido: 14 de julio de 2024. [En línea]. Disponible en: https://en.wikipedia.org/w/index.php?title=Link_budget&oldid=1192861660

[46] M. J. Gamez, «Objetivos y metas de desarrollo sostenible», Desarrollo Sostenible. Accedido: 10 de julio de 2024. [En línea]. Disponible en: https://www.un.org/sustainabledevelopment/es/objetivos-de-desarrollo-sostenible/

# Anexo I

En el Anexo I se presenta el código utilizado para la configuración de los dispositivos. Adicionalmente, en la Tabla 23 se detalla el lenguaje de programación, el entorno de programación y una estimación de las líneas utilizadas en el código.

*Tabla 23: Detalles del entorno de trabajo.*

| Descripción | Detalle |
|---|---|
| Lenguaje de Programación | C, C++ |
| Entorno de Programación | Visual Studio Code (VSCode) versión 1.90.0<br>PlatformIO (PIO) versión 6.1.1 |
| Líneas de Código | 1000 líneas |

- Identificador único (config.cpp)

```
void setupID(void){
  Serial.print("* Creating Device ID...");
  for(int i=0; i<17; i=i+8) {
    chipID |= ((ESP.getEfuseMac() >> (40 - i)) & 0xff) << i;
  }
  myBoardID = (byte) chipID;
  Serial.printf(" Chip ID: %d... ID device: 0x%2X \r\n", chipID, myBoardID);
  delay(1000);
}
```

- Módulo LCD (config.cpp)

```
int setupDisplay (void){
  if (OLED_RST != NOT_A_PIN){
    pinMode(OLED_RST, OUTPUT);
    digitalWrite(OLED_RST, LOW);
    delay(20);
    digitalWrite(OLED_RST, HIGH);
  }
  Serial.print("* Initializing Display...");
  // initialize OLED
  Wire.begin(OLED_SDA, OLED_SCL);
  if (!display.init()){ // Address 0x3C for 128x32
```

```
    Serial.println((" --- Error: SSD1306 allocation failed"));
    return -1;
  }
  Serial.println(" Display OK");
  return 0;
}
```

- Módulo LoRa (config.cpp)

```
int setupLORA (void){
  Serial.print("* Initializing LoRa...");
  //SPI LoRa pins
  SPI.begin(LORA_SCK, LORA_MISO, LORA_MOSI, LORA_CS);
  //setup LoRa transceiver module
  LoRa.setPins(LORA_CS, LORA_RST, LORA_DIO0);
  if (!LoRa.begin(BAND)) {
    Serial.println(" --- Error: Starting LoRa failed!");
    return -1;
  }
  LoRa.setTxPower(TXPOWER);
  // Text serial monitor and display
  Serial.println(" LoRa OK");
  return 0;
}
```

- Conexión WiFi (config.cpp)

```
void setupWiFi(void){
  #ifdef GATEWAY_LORA
    Serial.print("* Initializing WiFi...");
    /*Connection to the WiFi Network*/
    WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
    while(WiFi.status() != WL_CONNECTED){
      delay(500);
      Serial.print(".");
    }
    // Text serial monitor and display
    Serial.println(" WiFi OK");
  #endif
}
```

- Sensor DHT11 (sensor_DHT.cpp)

```
void setupDHT(void){
    dht.begin();
    Serial.println("* Initializing DHT11... DHT11 OK");
}
```

- Formato de la tabla de enrutamiento (routing.cpp)

```
typedef struct{
    byte destinationAddress;
    byte nextHop;
    int hopCount;
    char sequenceRoute[20];
} routeTableEntry;
routeTableEntry routeTable;

void setupRoutingTable(void){
    Serial.print("*** Creating Route Table...");
    routeTable.destinationAddress = GATEWAY_ID;
    routeTable.nextHop = 0;
    routeTable.hopCount = 0;
    strcpy(routeTable.sequenceRoute, " ");

    Serial.printf("Route Table(dest,nextHop,countHop,sequenceRoute):( %d | %d | %d | %s )\r\n",
routeTable.destinationAddress, routeTable.nextHop, routeTable.hopCount, routeTable.sequenceRoute);
}
```

# Anexo II: Diagramas de Flujo

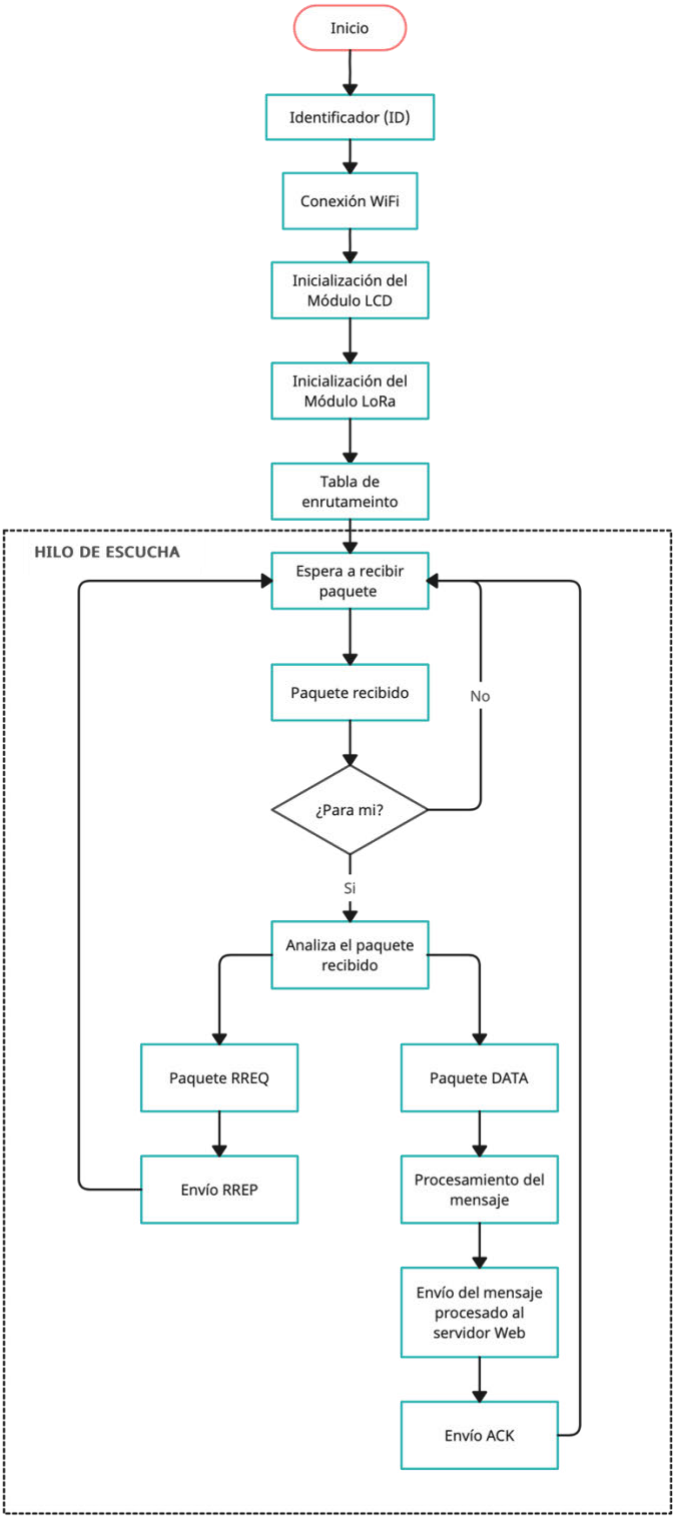


*Figura 65: Diagrama de flujo de la puerta de enlace (gateway).*

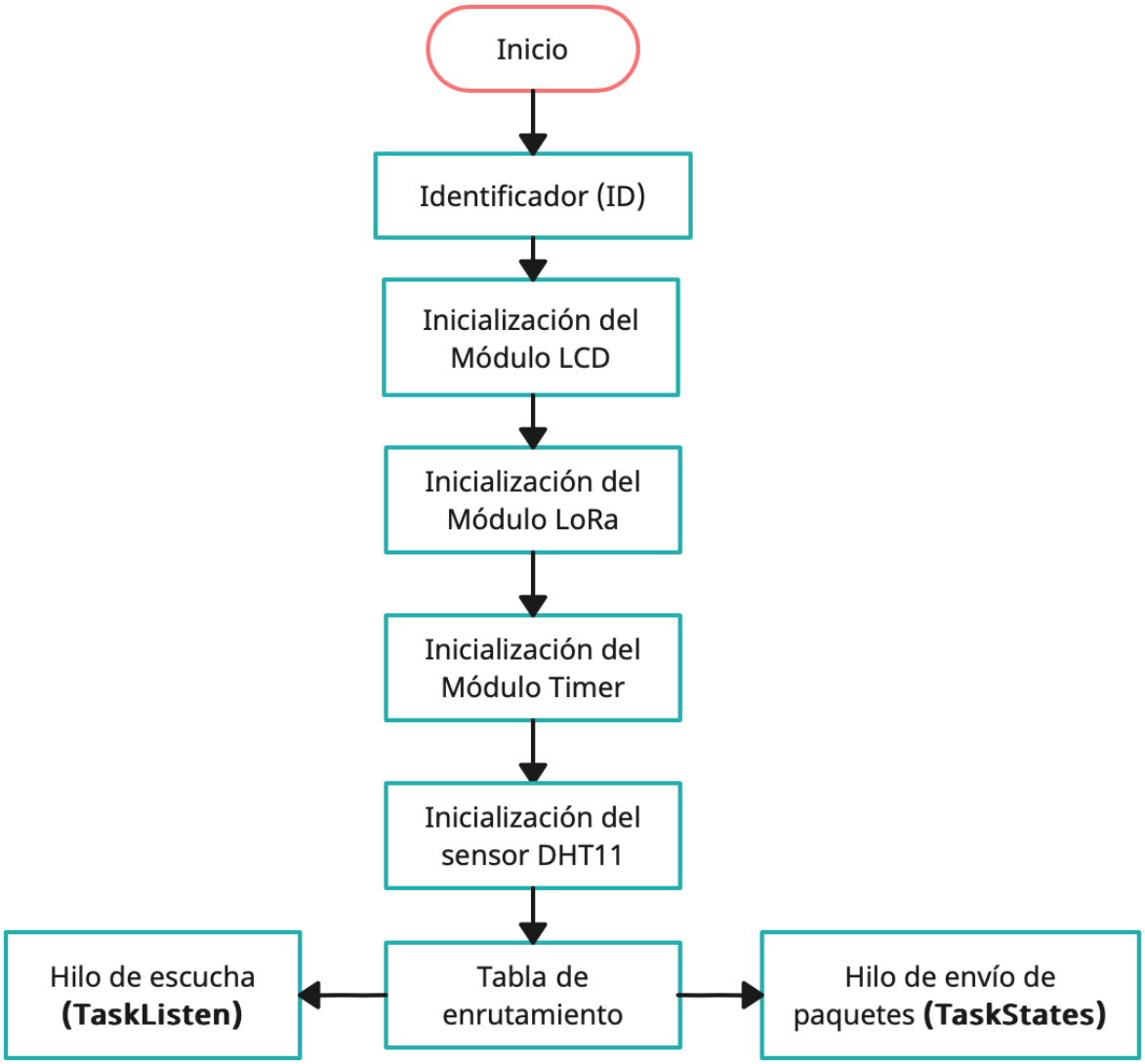


*Figura 66: Diagrama de flujo de los nodos (parte 1: visión general).*

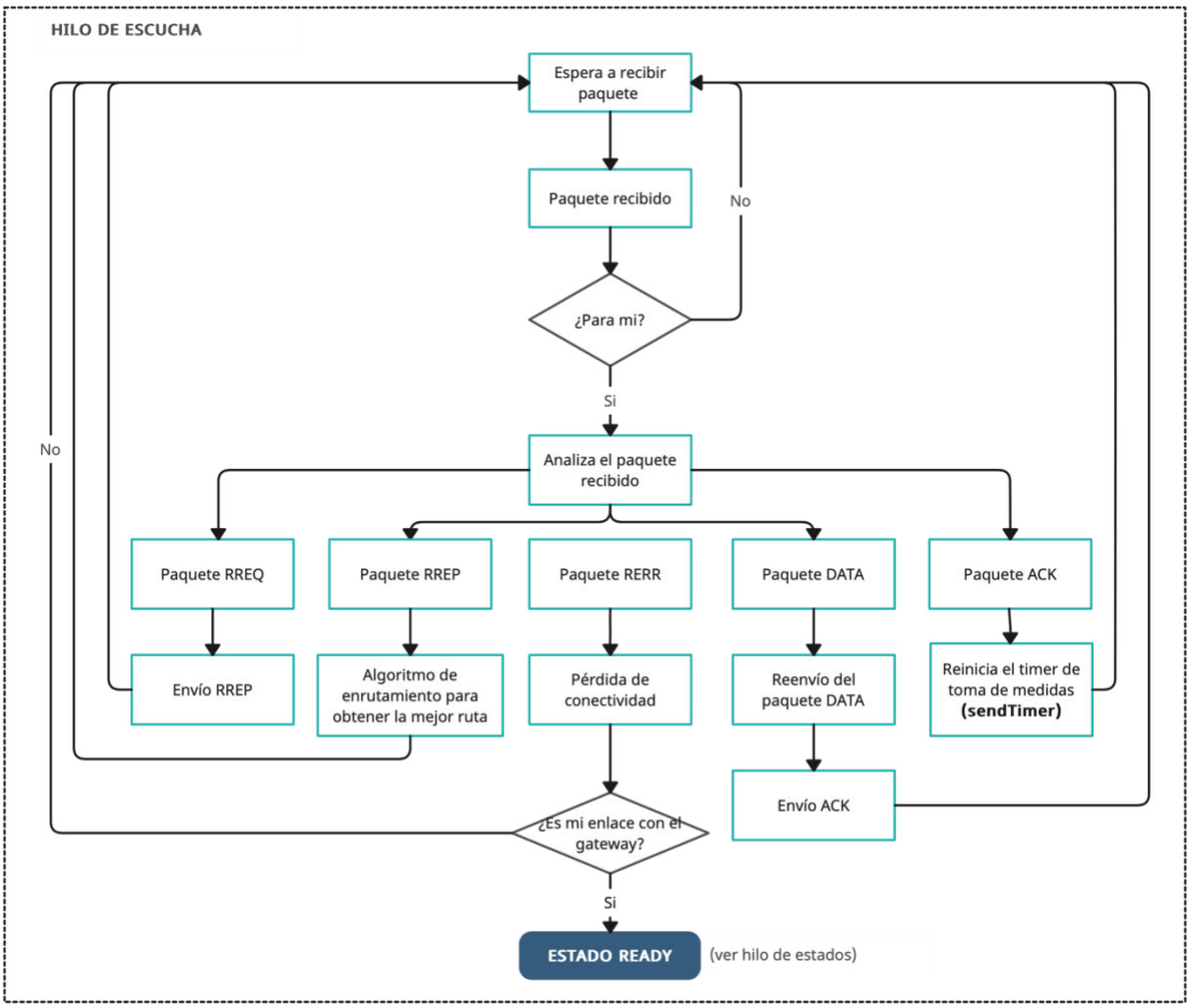


*Figura 67: Diagrama de flujo de los nodos (parte 2: hilo de escucha).*

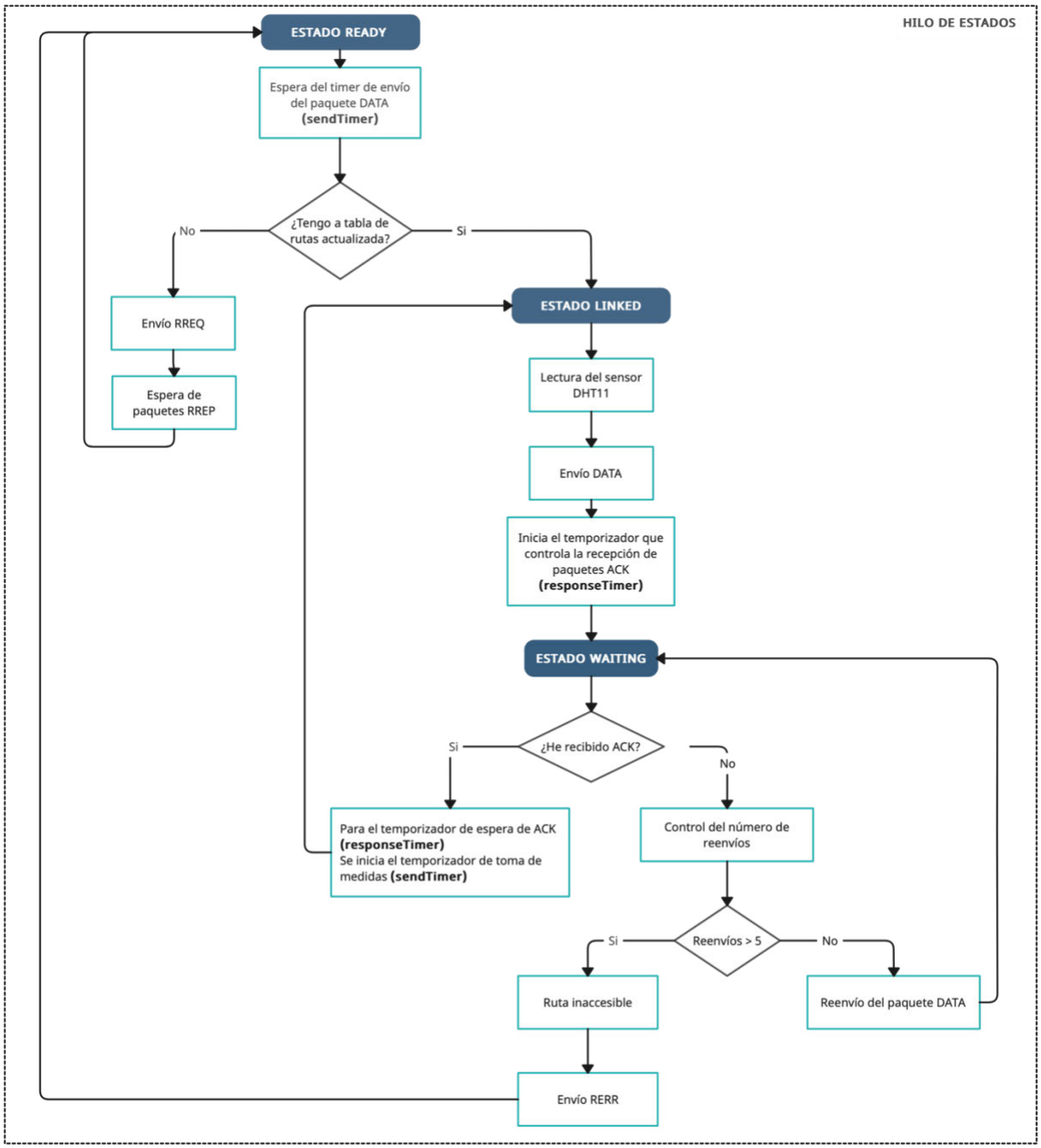


*Figura 68: Diagrama de flujo de los nodos (parte 3: hilo de estados).*

# Anexo III: Código prueba punto a punto

En el Anexo II se muestran partes de la programación que pertenece a las pruebas de conexión punto a punto del apartado 5.1.

- Estados del nodo (state.cpp)

```
void switchStates(void){
  switch (state) {
    case READY:
      state = LINKED;
      break;
    case LINKED:
      if(sendLoRa){
        char* dataMeasure = readSensorDHT();
        strncpy(measure, dataMeasure, sizeof(measure) - 1);
        measure[sizeof(measure) - 1] = '\0';
        sendPackage(GATEWAY, DATA, measure);
        sendLoRa = false;
        state = WAITING;
      }
      break;
    case WAITING:
      if(sendLoRaAgain){
        Serial.println(" --> Se envia de nuevo el paquete con los datos.\n");
        sendPackage(GATEWAY, DATA, measure);
        sendLoRaAgain = false;
      } else if(recieveACK){
        timerStop(responseTimer);
        timerStart(sendTimer);
        recieveACK = false;
        state = LINKED;
      }
      break;
    default:
      break;
  }
}
```

- Envío del paquete de datos (package.cpp)

```
extern byte myBoardID;
void sendPackage(byte destID, byte msgID, char* msg){
   // send packet
   LoRa.beginPacket();
   LoRa.write(myBoardID);
   LoRa.write(destID);
   LoRa.write(msgID);
   LoRa.print(msg);  // send the high byte
   LoRa.endPacket();

   Serial.printf(" ··· Message LoRa send... \r\n");
   Serial.printf("Sender: 0x%2X |  Destination: 0x%2X  |  Message ID: %d  |  Message: %s  \r\n", myBoardID, destID,
msgID, msg);
}
```

- Lectura del paquete de datos e identificación de mensajes (package.cpp)

```
byte senderID;         // sender address
byte receiverID;       // recipient address
byte incomingMsgID;    // incoming msg ID
String incomingMsg;    // data sensor
void readPackage(void){
   // read packet header bytes:
   senderID = LoRa.read();                  // sender address
   receiverID = LoRa.read();                // recipient/gateway address
   incomingMsgID = LoRa.read();                 // incoming msg ID
   incomingMsg = "";

   while (LoRa.available()) {
      incomingMsg += (char)LoRa.read();
   }
   if(receiverID == myBoardID){
      Serial.println(" ··· Message LoRa receive...");
      Serial.printf("Sender: 0x%2X |  Destination: 0x%2X  |  Message ID: %d  |  Message: %s  |  RSSI:  %d  | SNR:
%.2f  \r\n", senderID, receiverID, incomingMsgID, incomingMsg, LoRa.packetRssi(),
LoRa.packetSnr());identifyActionLoRa(incomingMsgID);
   }
}
void identifyActionLoRa(byte msgID){
```

```
    switch (msgID){
      case DATA:
        if(receiverID == localID){
          sendPackage(senderID, ACK, NULL);
          #ifdef GATEWAY_LORA
            sendDataToCloud();
          #endif
        }
        break;
      case ACK:
        if(receiverID == localID){
          recieveACK = true;
        }
        break;
      default:
        break;
    }
}
```

# Anexo IV: Código prueba de algoritmo de enrutamiento

En el Anexo III se muestran partes de la programación que pertenece a las pruebas de conexión punto a punto del apartado 5.2.

- Envío de mensajes RREQ y RREP (routing.cpp)

```
void sendRREQ(byte destinationId){
   char sequenceRREQ[50];
   sprintf(sequenceRREQ,"%d;",localID);
   strcpy(routeTable.sequenceRoute, sequenceRREQ);
   sendPackage(destinationId, RREQ, routeTable.sequenceRoute);
}

void sendRREP(byte destinationId, char* incomingSequence){
   char sequenceRREP[50];
   sprintf(sequenceRREP,"%s%d;", incomingSequence, localID);
   strcpy(routeTable.sequenceRoute,  sequenceRREP);
   sendPackage(destinationId,RREP,routeTable.sequenceRoute);
}
```

- Actualización de la tabla de enrutamiento (routing.cpp)

```
void updateRouteTable(const char *sequenceRoute){

   byte byte_nextHop;

   Serial.print(" *** Updating Route Table...");
   char* char_nextHop = getNextHop(sequenceRoute);
   //sscanf(char_nextHop,"%hhx", &byte_nextHop);

   if (sscanf(char_nextHop, "%d", &byte_nextHop) != 1) {
      Serial.println("Error: Failed to parse next hop.");
      return;
   }
   routeTable.nextHop = byte_nextHop;
   strcpy(routeTable.sequenceRoute, sequenceRoute);

   Serial.printf(" Route Table(dest,nextHop,countHop,sequenceRoute):(  %d  |  %d  |  %d  |  %s  )\r\n",
```

```
            routeTable.destinationAddress, routeTable.nextHop, routeTable.hopCount, routeTable.sequenceRoute);
    representLCD_Node();

}
```