

LORA based wireless sensor networks: LORA things

Ivaylov Peychev, Bozhidar
Curso: 2020 - 2021

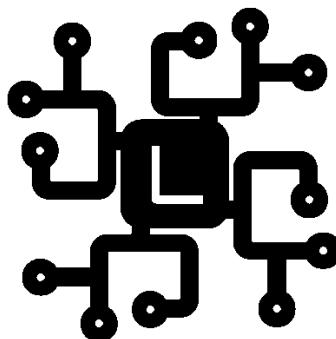
Tutores:
Toni Adame

Trabajo Final de Grado



Universitat
Pompeu Fabra
Barcelona

Escola
Superior Politècnica



Lora things
Connects the World

TREBALL FI DE GRAU DE

Bozhidar Ivaylov Peychev

Director: Toni Adame Vazquez

Grau en Enginyeria en Informàtica

Curs 2020-2021



Universitat
Pompeu Fabra
Barcelona

Escola
d'Enginyeria

Página en blanco intencional

Página en blanco intencional

DEDICATORIA

Dedico este trabajo a mi familia que han estado apoyándome continuamente con mis estudios y durante el desarrollo de este proyecto.

Página en blanco intencional

Agradecimientos

En primer lugar quisiera agradecer a Toni Adame, mi tutor que me ha guiado y ayudado durante el desarrollo de este proyecto. Gracias a él aprendí mucho y he podido llevar a cabo este proyecto. También agradezco a mis profesores de la facultad de ingeniería que me enseñaron tantas cosas durante estos años de carrera.

A mi familia que me ayudaba y apoyaba tanto durante mis estudios como durante el proceso de desarrollo de este proyecto. Ellos me daban las fuerzas y apoyó continuamente lo que me ayudó a terminar este trabajo.

Página en blanco intencional

Resumen

El Internet de las cosas(abreviado IOT) ha estado creciendo exponencialmente durante las últimas décadas. Esta incluye distintos dispositivos electrónicos ubicados en distintos lugares del planeta que se comunican entre sí, mediante distintos protocolos, redes de antenas, satelitales, etc. Hoy en día, el IOT se aplica en diferentes sectores como la domótica, smart cities, fábricas, agricultura y muchos más. Todos los dispositivos IOT tienen algo en común - los sensores. También conocidos como *redes de sensores inalámbricos*, estas tienen un amplio uso tanto por empresas grandes como por usuarios particulares o equipos de investigadores pequeños.

Sin embargo, una de las principales limitaciones de esta tecnología es la cobertura de la señal o el rango de control que puede haber entre el dispositivo IOT y la antena más cercana. La mayoría de los dispositivos IOT están usando tecnologías como LORA o SIGFOX que están limitadas no solo en la cobertura pero también en la cantidad de datos que se puede enviar durante un determinado tiempo.

El propósito de este proyecto es eliminar estas limitaciones y al mismo tiempo nos indica cómo construir una red de sensores inalámbricos de manera rápida, sencilla y que tenga cobertura mundial. Es decir, tenemos que ser capaces de acceder a nuestra red de sensores desde cualquier punto del planeta, utilizando únicamente un teléfono móvil, un ordenador o cualquier dispositivo que tenga acceso a Internet.

Para este propósito se han combinado varias tecnologías de comunicación inalámbrica incluyendo LORA y GSM. Estas tecnologías permiten cobertura mundial sin depender de otros dispositivos o aparatos adicionales como instrumentos de medición, antenas, etc. A parte, este proyecto incluye un protocolo de comunicación especialmente diseñado para este propósito, que facilita la comunicación entre los dispositivos.

El nombre que ha sido elegido para este proyecto es **LORA Things**, inspirado por la tecnología LORA y el *Internet of Things*. Para más comodidad en adelante usaremos este nombre ya que es más corto, aunque el nombre completo del proyecto es:

LORA based multipoint wireless sensor networks.

Summary

The Internet of Things (IOT for short) has been growing exponentially for the past few decades. This includes different electronic devices located in different places on the planet that communicate with each other, through different protocols, antenna networks, satellites, etc. Today, the IOT is applied in different sectors such as home automation, smart cities, factories, agriculture and many more. All IOT devices have one thing in common - sensors. Also known as wireless sensor networks, these are widely used by both large companies and private users or teams of small researchers.

However, one of the main limitations of this technology is the signal coverage or control range that may exist between the IOT device and the nearest antenna. Most of the IOT devices are using technologies like LORA or SIGFOX which are limited not only in coverage but also in the amount of data that can be sent during a certain time.

The purpose of this project is to eliminate these limitations and at the same time it shows us how to build a wireless sensor network quickly, easily and with worldwide coverage. In other words, we have to be able to access our sensor network from anywhere on the planet, using only a mobile phone, a computer or any device that has Internet access.

For this purpose various wireless communication technologies were combined including LORA and GSM. These technologies allow me worldwide coverage without depending on other devices or additional devices such as measuring instruments, antennas, etc. In addition, this project includes a communication protocol specially designed for this purpose, which facilitates communication between devices.

The name chosen for this project is LORA Things, inspired by LORA technology and the Internet of Things. For convenience from now on we will use this name as it is shorter, although the full name of the project is: LORA based wireless sensor networks.

Página en blanco intencional

Página en blanco intencional

Índice

1. INTRODUCCIÓN	19
1.1 Motivación	19
1.2 Objetivo del proyecto	20
2. ESTADO DEL ARTE	23
2.1. The Internet of Things	23
2.1.1. Introducción	23
2.1.2. Características	25
2.1.3. Arquitectura	25
2.1.4. Aplicaciones	27
Monitoreo de tráfico	28
Agricultura	29
Domótica	29
2.1.5. Problemas de seguridad	29
2.2. Smart Cities	30
2.2.1. Introducción	30
2.2.2. ¿Por qué las ciudades inteligentes son importantes?	30
2.2.3. Aplicaciones del IOT en las Smart Cities	31
2.3. Principales tecnologías de comunicación inalámbrica IoT para las Smart Cities	33
2.3.1. LORA(Long Range)	33
2.3.1.1. Introducción	33
2.3.1.2. Aplicaciones	34
2.3.1.3. Características y ventajas	35
2.3.1.4. Cobertura	36
2.3.2. SIGFOX	38
2.3.2.1. Introducción	38
2.3.2.2. Aplicaciones	38
2.3.2.3. Características	39
2.3.2.4. Cobertura	40
2.3.3. GSM	41
2.3.3.1. Introducción	41
2.3.3.2. Características y arquitectura	41
2.3.3.3. Aplicaciones	42
2.3.3.4. Cobertura	43
2.5. Análisis comparativo de tecnologías inalámbricas	44
2.5.1. Comparación	44
2.5.2. Integración de datos	45

2.5.3. Conclusión	45
3. DESARROLLO DEL PROYECTO	47
3.1 Concepto	47
3.2 Metodología	47
3.2.1. Fase 1: Investigación	48
3.2.2. Fase 2: Estructurar el proyecto	49
3.2.2.1. Arquitectura	49
3.2.2.2. Herramientas y componentes	51
3.2.2.3. Presupuesto	57
3.2.2.4. Competencia	59
3.2.3. Fase 3: Integración del proyecto	61
3.2.3.1. Configuración de los módulos Lora	61
3.2.3.2. Configurar el módulo GSM	68
3.2.3.3. Conectar los módulos Lora y GSM	71
3.2.3.4. Protocolo de comunicación	73
3.2.3.5. Configurar la plataforma IOT	78
3.2.3.6. Envío de datos a la plataforma IOT	84
3.2.4. Fase 4: Fase de pruebas y validación	88
3.2.4.1. Consumo de energía	88
3.2.4.2. Tiempo de ejecución	92
3.2.4.3. Prueba de colisión de datos	95
3.2.4.4. Prueba de comunicación en paralelo	96
3.2.4.5. Envío de datos a la plataforma IOT	98
3.2.4.6. Historial de datos	100
3.2.4.7. Sistema de notificación	101
4. Análisis de rendimiento y estadísticas	105
4.1. Prueba de consumo de energía	105
4.2. Tiempo de envío de los paquetes	108
5. Conclusiones	113
6.1. Conclusiones a partir de los resultados	113
6.2. Futuras mejoras	114
6. Referencias	117

Llista de figures

Figura 1	24
Figura 2	24
Figura 3	26
Figura 4	27
Figura 5	28
Figura 6	32
Figura 7	34
Figura 8	35
Figura 9	36
Figura 10	37
Figura 11	40
Figura 12	40
Figura 13	42
Figura 14	43
Figura 15	49
Figura 16	50
Figura 17	52
Figura 18	53
Figura 19	54
Figura 20	56
Figura 21	56
Figura 22	57
Figura 23	59
Figura 24	60
Figura 25	60
Figura 26	62
Figura 27	62
Figura 28	63
Figura 29	63
Figura 30	64
Figura 31	66
Figura 32	67
Figura 33	68
Figura 34	69
Figura 35	69
Figura 36	70
Figura 37	71
Figura 38	72
Figura 39	74
Figura 40	76
Figura 41	79
Figura 42	79
Figura 43	80
Figura 44	81
Figura 45	82
Figura 46	82

Figura 47	83
Figura 48	84
Figura 49	87
Figura 50	89
Figura 51	90
Figura 52	91
Figura 53	92
Figura 54	93
Figura 55	95
Figura 56	96
Figura 57	96
Figura 58	97
Figura 59	98
Figura 60	99
Figura 61	100
Figura 62	101
Figura 63	102
Figura 64	102
Figura 65	102
Figura 66	103
Figura 67	107
Figura 68	108

Llista de tablas

Tabla 1	44
Tabla 2	58
Tabla 3	75
Tabla 4	75
Tabla 5	85
Tabla 6	86
Tabla 7	106
Tabla 8	107
Tabla 9	109

Página en blanco intencional

1. INTRODUCCIÓN

1.1 Motivación

Desde siempre he tenido pasión por las tecnologías. Lo que más me llamaba la atención era la microelectrónica, sensores, comunicaciones inalámbricas, la programación. Por este motivo este proyecto incluye diferentes aspectos de la Ingeniería Informática como los que he mencionado.

Siempre he tenido curiosidad y ganas de aprender más sobre estas tecnologías pero no tuve la oportunidad hasta que entré en la universidad y ahí he podido profundizar y dedicarles más tiempo. Mi interés hacia las tecnologías IOT se despertó después de hacer una asignatura en la universidad llamada *IOT*. Fue entonces cuando aprendí sobre la existencia de este tipo de tecnologías y su enorme potencial y uso en el mundo actual.

Fue entonces cuando vi los diferentes usos de estas tecnologías, sus aplicaciones y posibles ventajas. Uno de los proyectos que me llamó la atención fue precisamente la red de sensores inalámbricos. Estos tienen un amplio uso en diferentes sectores de la industria, pero me he fijado que tienen algunas limitaciones. Tras profundizar más en estas tecnologías, me di cuenta que combinando algunas de estas, podría conseguir un nuevo dispositivo(o sistema de dispositivos) eliminando las limitaciones o restricciones que tienen algunas redes de sensores en la actualidad.

1.2 Objetivo del proyecto

El propósito de este proyecto es construir una red de sensores inalámbricos orientados al IOT, que permita habilitar un caso de uso como es la monitorización ambiental en cualquier parte del mundo. La estructura de este proyecto es relativamente sencilla y robusta lo que facilita su instalación, manejo y adaptación a diferentes casos de uso. Las tecnologías seleccionadas permiten a esta red un acceso a Internet desde cualquier lugar del mundo. El propósito de hacer esto es enviar los datos a un servidor o plataforma IOT una vez los sensores hayan leído los valores del medio ambiente.

En la actualidad existen productos similares como redes de sensores o instrumentos de medición del medio ambiente, sin embargo estos tienen ciertas limitaciones. Por ejemplo, los dispositivos carecen de ciertas funcionalidades, son difíciles de manejar o configurar, cumplen un solo propósito. Este proyecto combina algunas de las mejores características y funcionalidades de diferentes dispositivos y tecnologías de monitoreo medioambiental.

El proyecto no está pensado precisamente para usarlo dentro de una ciudad, ya que esta ya dispone diferentes antenas, repetidores, sensores y otros dispositivos, que ya nos ofrecen una buena conexión y hacen una gran variedad de mediciones. Además, unos de los principales obstáculos en una ciudad son los edificios y otras frecuencias que pueden debilitar la señal de nuestro dispositivo. Esto no quiere decir que no podamos usarlo en una ciudad como por ejemplo una *smart city*. El dispositivo funcionará perfectamente. Simplemente podríamos usar la infraestructura de antenas que ya están instaladas. Sin embargo, en una zona poco poblada o rural, como por ejemplo un bosque o un campo, las infraestructuras digitales y la conectividad no están tan desarrolladas o prácticamente no existen. Por este motivo, el proyecto dispone de todas las tecnologías y herramientas necesarias que le permiten acceso a Internet desde cualquier lugar del mundo. Esto lo hace independiente de cualquier otra infraestructura o sistema.

Si se desea monitorear el ambiente cerca de un lago o un bosque, sería necesario que una persona se acerque a este lugar y tome las mediciones necesarias. Sin embargo, para hacer mediciones frecuentes o durante un largo periodo de tiempo, hace que este proceso sea incómodo y poco práctico. Además, si la zona que se desea monitorear es difícil de acceder, tóxica o radioactiva, dificulta todavía más el proceso e incluso podría poner en peligro la vida de la persona.

Una posible solución sería usar un dispositivo que mediante sensores pueda hacer lecturas del medio ambiente y envíe los datos de forma inalámbrica. Las tecnologías que más se adaptan a esta situación son LORA, SIGFOX y GSM. La primera tiene una desventaja para cumplir los requisitos de este proyecto, ya que el máximo alcance de la señal es hasta 10km. Además, LORA depende de una red de antenas y repetidores especialmente diseñadas para este propósito. La mayoría de las ciudades tienen establecidas diferentes infraestructuras que disponen de una amplia variedad de antenas y repetidores que permiten retransmitir la señal enviada por un emisor. Sin embargo, en

un entorno rural no todos los países disponen de redes LoraWAN. Por este motivo, el uso solamente de esta tecnología no sería suficiente y no cumpliría con el objetivo del proyecto. Por otra parte, la tecnología SIGFOX tiene mayor rango de alcance de la señal, ya que utiliza la infraestructura de las operadoras telefónicas para la propagación de la señal. Sin embargo, esta tecnología todavía está en proceso de desarrollo y no llega a tener una cobertura mundial.

Una solución posible es combinar diferentes tecnologías de comunicación inalámbricas como LORA y el GSM. Hoy en día, la tecnología GSM tiene cobertura prácticamente a nivel mundial. En casi cualquier país o lugar se puede obtener señal aunque no siempre sea posible conseguir una conexión de buena calidad. Es decir, en zonas rurales o poco pobladas, las operadoras GSM instalan o bien pocas antenas base o bien antenas de baja potencia. Por lo tanto, las conexiones a Internet podrían ser o bien muy lentas o bien imposibles.

Sin embargo, los nodos finales de este proyecto pueden generar paquetes de un volumen relativamente pequeño, menos de 200 bytes. Además, el envío de datos a Internet que realizan los módulos GSM se hace con poca frecuencia, por ejemplo cada hora o hasta varias veces por día. Una conexión de un ancho de banda más bajo, implica una transmisión de datos más lenta, es decir, se transmiten pequeñas cantidades de datos en un determinado intervalo de tiempo. La potencia de transmisión del GSM también juega un papel muy importante en el proceso de comunicación. Utilizar una potencia más baja, puede tener como consecuencia la pérdida de algunos paquetes y el reenvío de los mismos. Todos estos factores pueden ralentizar el proceso de comunicación. Sin embargo este proyecto no se basa en conexiones de alta velocidad lo que nos permite poder conectarnos a Internet desde una zona rural o de baja conectividad.

El proyecto tiene una estructura bastante sencilla lo que facilita su instalación y el uso por parte del usuario. El prototipo de este proyecto es bastante básico, pero esto no tiene que ser una limitación ya que el usuario es libre de extenderlo, mejorarlo y adaptarlo a su gusto. Si lo que el usuario quiere es mejorar el rendimiento de este proyecto, simplemente tiene que cambiar el módulo correspondiente.

Página en blanco intencional

2. ESTADO DEL ARTE

2.1. The Internet of Things

2.1.1. Introducción

Internet of Things(abreviado IoT) o también conocido como Internet de las Cosas, es una red de dispositivos electrónicos de todo tipo que llevan integrados diferentes sensores, software y otras tecnologías con el fin de conectar e intercambiar datos con otros dispositivos y sistemas a través de Internet.

La red de IoT incluye dispositivos de cualquier tipo, desde el más sencillo como una bombilla, hasta los dispositivos más complejos como los televisores inteligentes. Las tecnologías del IoT nos permiten conectar no solo los ordenadores o los teléfonos móviles al Internet, sino otros dispositivos como televisores inteligentes, lavadoras, cámaras de video, incluso máquinas de café. Para ser más específicos, el término IoT hace referencia a los sistemas de dispositivos físicos que reciben y transfieren datos a través de redes inalámbricas sin la intervención humana. Estos dispositivos pueden incorporarse tanto en redes pequeñas o locales como en redes a gran escala y pueden comunicarse a través de la red de Internet sin importar su ubicación en el planeta.

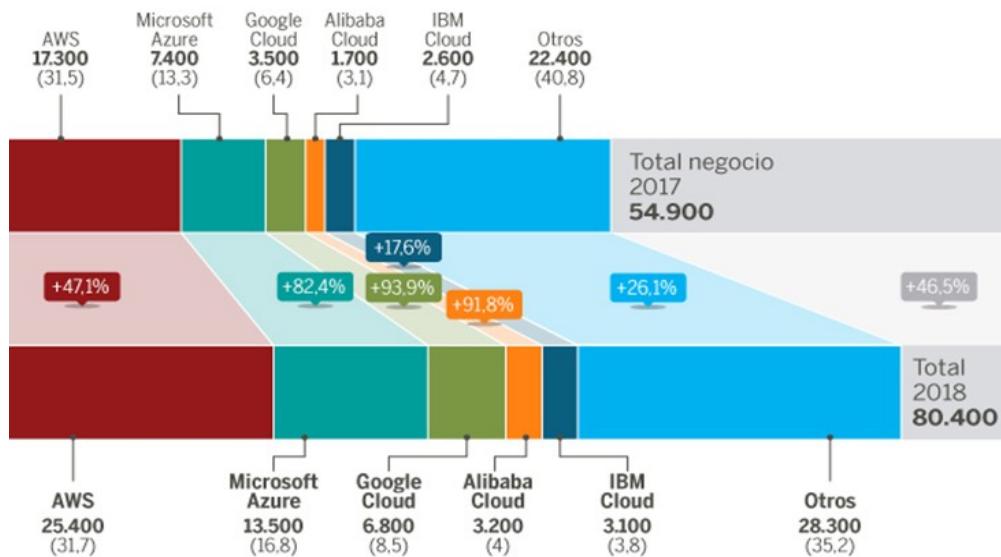
Un ejemplo de dispositivo IoT es un "termostato inteligente" ("inteligente", significa parte del "IoT") que puede recibir la ubicación del vehículo inteligente mientras el usuario conduzca. Si la persona se aproxima a su casa, el termostato puede ajustar la temperatura antes de que ésta llegue. También el vehículo puede enviar una notificación a la cafetera inteligente que uno tiene en su casa, de esta manera se prepararía el café justo antes de llegar. Todo esto se puede lograr sin la intervención del usuario.

A medida que el IoT iba creciendo, varias empresas como Microsoft, Amazon, y Google se dedicaron a ofrecer servicios **cloud**(o plataformas cloud) para aplicaciones IoT. Estos servicios incluyen una de las aplicaciones más destacadas - la plataforma de IoT. Esta permite a los usuarios gestionar de una manera óptima sus dispositivos y aplicaciones IoT. Algunos ejemplos de plataformas IoT son *Ubidots*, *Thingsboard*, *Microsoft Azure*.

En la siguiente imagen podemos ver cómo está dividido el servicio 'cloud' entre las empresas más destacadas como Amazon, Google y Microsoft.

RADIOGRAFÍA DE LA INDUSTRIA DE 'CLOUD' PÚBLICA

Negocio en millones de dólares. Entre paréntesis, cuota (%). Crecimiento en porcentaje.



Fuente: Canalys, estimaciones

Expansión

Figura 1: Estadística de la industria de cloud computing.

Fuente: <https://geoactio.com>

IoT está en un crecimiento constante. Actualmente con más de 35 mil millones de dispositivos conectados a Internet alrededor del mundo, en la última década su número se ha duplicado. Según las predicciones de algunos expertos, su número superará los 75 mil millones de dispositivos para el año 2025. A continuación se puede observar un gráfico que muestra cómo ha crecido el número de dispositivos IoT desde el año 2015 y qué números podemos esperar para el año 2025:

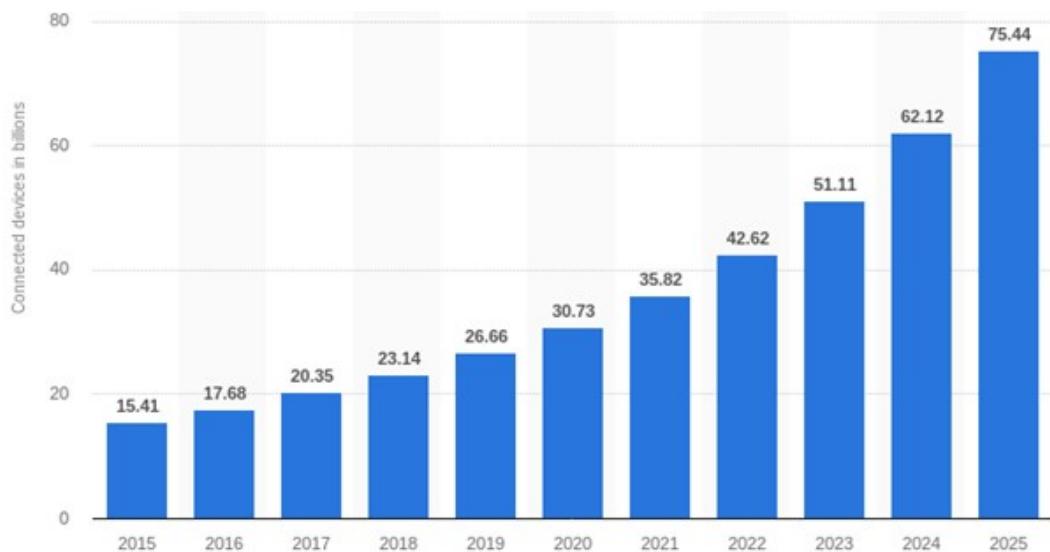


Figura 2: Estadística de la evolución de IoT.

Fuente: <https://www.iotworldonline.es>

2.1.2. Características

Como ya sabemos, IoT es una red de alta complejidad que interconecta objetos a Internet mediante diferentes protocolos de comunicación. Estos objetos proporcionan servicios sin la necesidad de intervención humana, tales como captura de datos, comunicación o capacidad de actuación a través de interfaces inteligentes, y tienen disponibilidad desde cualquier lugar. Las características^[1] principales que definen los objetos en un entorno IoT se muestran a continuación:

- **Sensibilidad.** Las máquinas necesitan sensores para poder interpretar el entorno a su alrededor. Estos dispositivos facilitan el monitoreo, la detección, la recopilación de datos y otras funciones básicas vinculadas al internet de las cosas.
- **Interacción.** El uso de la IoT ha hecho posible la comunicación entre realidad, sistemas y personas. Este avance se traduce en diferentes logros, como casas inteligentes, granjas automatizadas, robots y mucho más.
- **Conectividad.** Más allá de un simple módulo para tener señal, el mayor potencial de esta se refleja en la aumentar la compatibilidad y el acceso a la red sin importar el contexto.
- **Seguridad.** A mayor cantidad de equipos conectados, mayores las necesidades de protegerlos. Por ello, la arquitectura y la ciberseguridad son esenciales para estos proyectos.

2.1.3. Arquitectura

Debido a la cantidad de dispositivos y tecnologías que forman el IOT, no existe una arquitectura específica, lo cual puede llevar a una confusión en la definición de los sistemas IoT. Sin embargo, podemos definir un modelo genérico y dividir la arquitectura de IOT en los siguientes segmentos:

- **Sensores/actuadores**

Los sensores y los actuadores son los encargados de conectar el mundo físico y digital. Se encargan de convertir la información capturada en datos para su posterior análisis. Adicionalmente, los actuadores pueden interactuar con el mundo físico.

- **Capa de comunicación**

Para transportar la información capturada por los sensores se necesita una infraestructura de red que admite los protocolos de comunicación necesarios para las comunicaciones Machine to Machine(M2M) que cubren la amplia gama de aplicaciones y servicios que ofrece el IoT.

- **Capa de gestión de servicios**

Esta capa es la encargada del procesamiento de la información capturada a través del análisis, la gestión, el modelado de procesos y el control de seguridad de los dispositivos, adaptándose a las reglas definidas en la aplicación IoT a la cual sirve.

- **Capa de aplicación**

Cada aplicación IoT cubre entornos o espacios en dominios determinados por el objetivo de la propia aplicación.

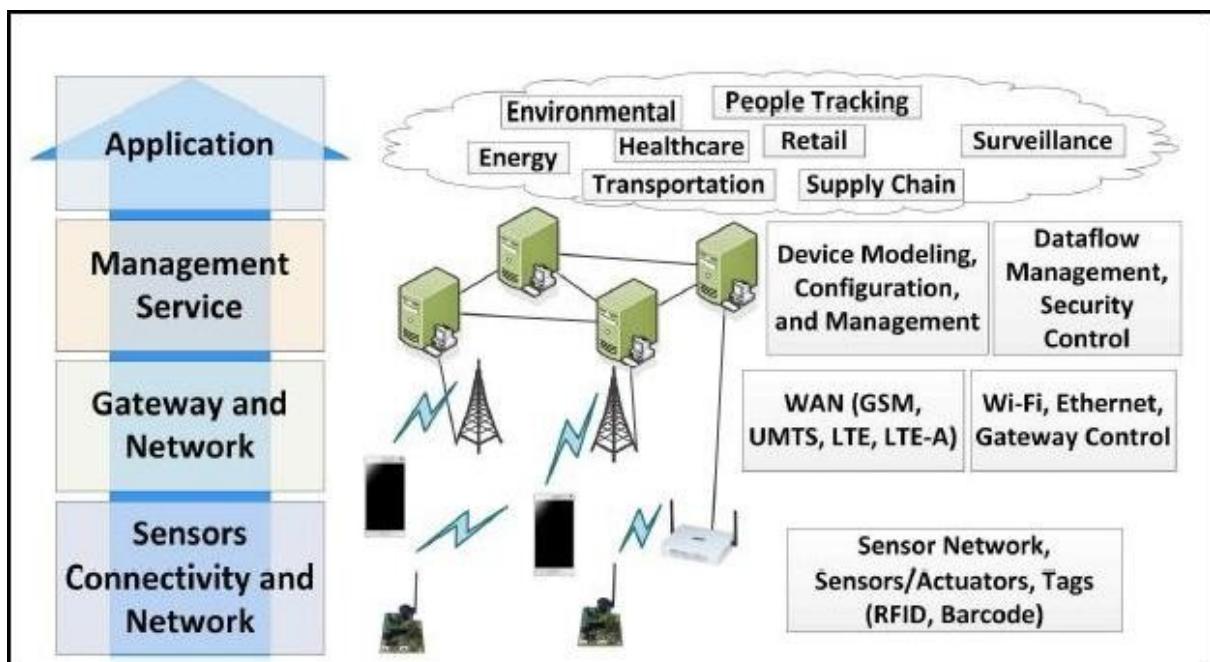


Figura 3: Esquema general de la arquitectura del IOT

Fuente: <https://aprendiendoarduino.wordpress.com/2018/11/11/arquitecturas-iot/>

2.1.4. Aplicaciones

IoT tiene múltiples aplicaciones en diferentes sectores incluyendo smart cities(o ciudades inteligentes), la agricultura, fábricas, diferentes comunidades científicas, la automoción y muchas más. Hoy en día, muchas empresas de diferentes sectores están adoptando esta tecnología para simplificar, mejorar, automatizar y controlar los diferentes procesos. Las tecnologías de IOT facilitan el control sobre diferentes máquinas, permiten automatizar diferentes procesos en las fábricas, las ciudades inteligentes(*smart city*)[2] o incluso en los hogares(*domótica*). El IoT también permite a los usuarios interactuar con el medioambiente tomando diferentes mediciones mediante una variedad de sensores.

En la imagen que aparece a continuación, representa una estadística de la división del IoT entre los diferentes sectores de la industria a nivel mundial. Según las últimas estadísticas del año 2020, los *smart cities* ocupan un 12%[3].

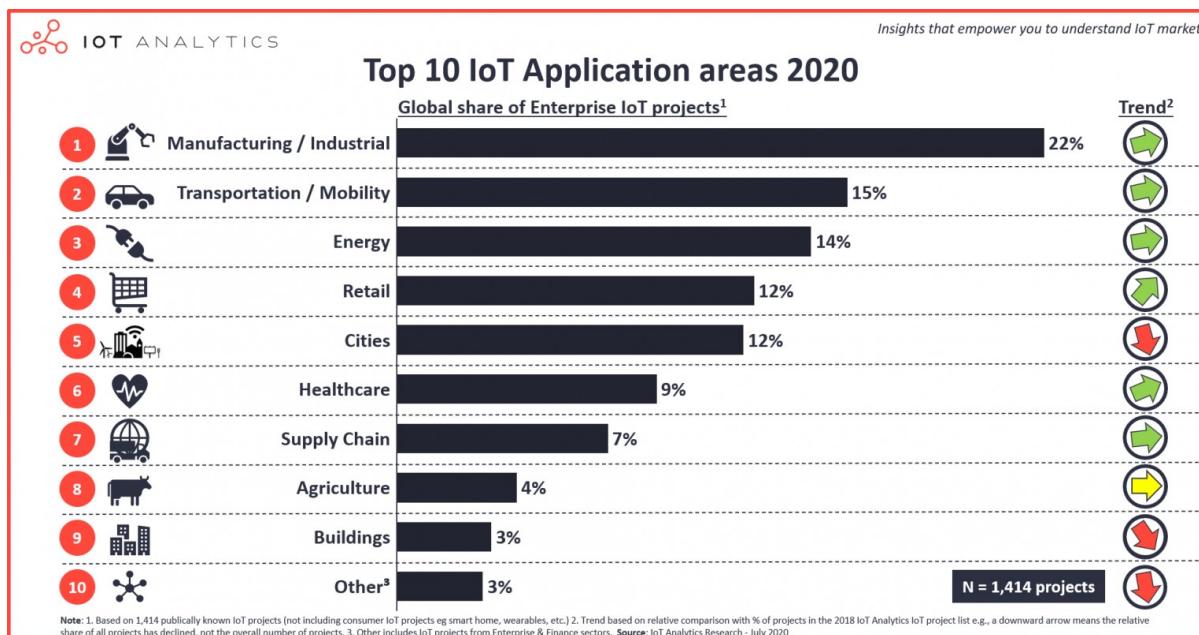


Figura 4: Estadística de las aplicaciones del IoT entre los diferentes sectores de la industria.
Fuente: <https://iot-analytics.com/top-10-iot-applications-in-2020/>

Durante los últimos años, IoT se ha convertido en una de las tecnologías más importantes y avanzadas del siglo XXI. Ahora que los usuarios pueden conectar a Internet casi cualquier dispositivo, existe una comunicación mucho más fluida entre personas, y máquinas. Gracias a la variedad de tecnologías que incluye IoT, entre las cuales, aplicaciones para ordenadores y móviles, sensores, dispositivos inteligentes, plataformas de desarrollo, los dispositivos IoT pueden interactuar entre sí, compartir y recopilar datos con una intervención humana mínima.

A continuación se verán con más detalle algunas de las aplicaciones más destacadas del IoT. Estas subcategorías ocupan un lugar importante en la industria de IoT.

- Monitoreo de tráfico

El IoT tiene amplio uso en el control y monitoreo del tráfico. No solo en las ciudades, sino también fuera de ellas como las autopistas por ejemplo. Las calles de las ciudades más grandes y modernas están equipadas con diferentes dispositivos como cámaras de vigilancia, sensores de velocidad y otros que nos ayudan a monitorear el tráfico. Los teléfonos móviles que los usuarios llevan encima, también juegan un papel importante en el monitoreo del tráfico. A través de aplicaciones como Waze o Google Maps, los móviles pueden compartir datos como la ubicación o velocidad. A partir de ahí, los servidores de Google hacen una estadística sobre el estado del tráfico y nos indican como de cargadas están las diferentes calles o autopistas.

La siguiente captura representa la intensidad del tráfico de la ciudad de Barcelona según las estadísticas de Google Maps. Se puede observar como las carreteras con una intensidad de tráfico diferente están coloreadas de diferentes colores:

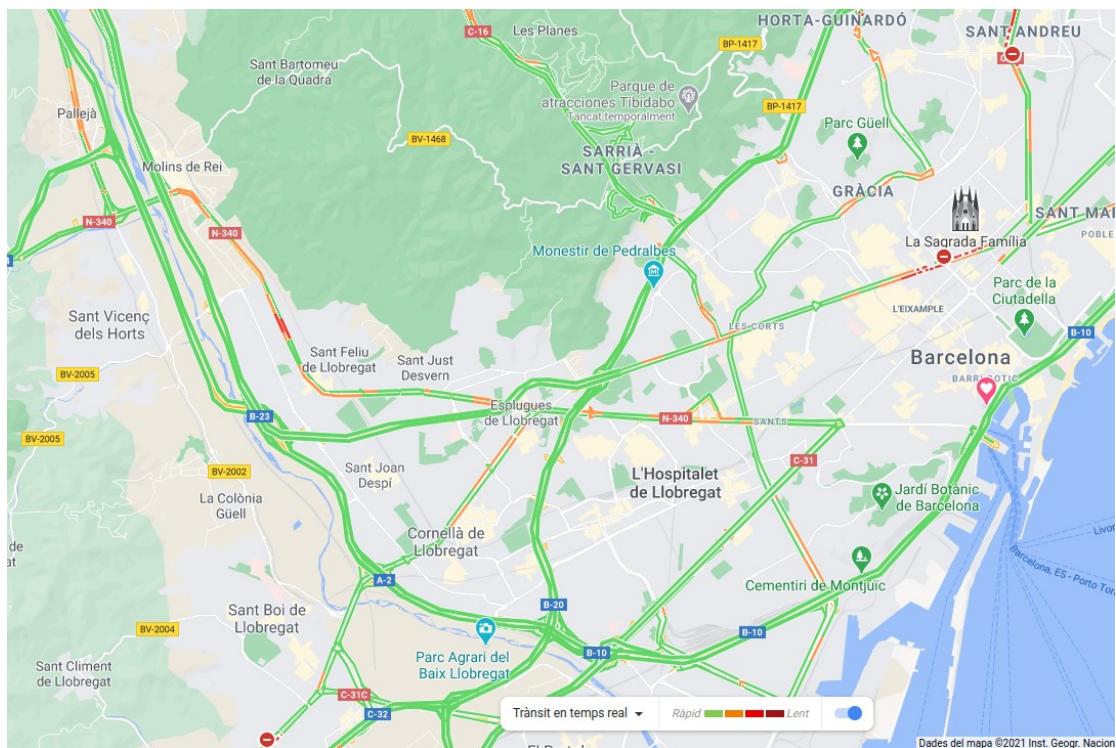


Figura 5: Mapa de la intensidad del tráfico de Barcelona
Fuente: Google Maps

- **Agricultura**

El IoT también tiene un uso amplio en las granjas y la agricultura. La calidad de los suelos es determinante para producir buenas cosechas, y el Internet de las Cosas ofrece a los agricultores la posibilidad de acceder al conocimiento detallado de sus condiciones. Mediante una variedad de sensores, se pueden obtener datos importantes sobre el estado del suelo, su nivel de acidez, la presencia de algunos nutrientes, temperatura del aire, humedad, y muchos más. Los datos recopilados por los sensores se pueden almacenar y procesar por los servidores de alguna plataforma IoT. Esto ayudará a los agricultores a controlar los riegos del campo, precisar los mejores momentos para iniciar la siembra, y hasta descubrir la presencia de enfermedades y/o plagas en las plantas y el suelo.

- **Domótica**

Con el crecimiento del IoT, este encontró múltiples aplicaciones en nuestras casas también. El término *domótica* se refiere al proceso de automatizar un edificio de cualquier tipo: una vivienda, oficina u otros. Mediante diversos aparatos y sensores que en los últimos años han ganado mucha popularidad en el mercado, podemos controlar y automatizar diferentes procesos de nuestra oficina o casa con una intervención humana mínima. Algunas de las aplicaciones del IoT en la domótica son: ajustar la climatización, controlar las persianas, gestión eléctrica, controlar la alarma u otro sistema de seguridad. En el año 2014, la empresa Amazon anunció un nuevo producto llamado Alexa. Este es un asistente virtual en forma de altavoz, también conocido como *Altavoz Inteligente Amazon Echo*[4].

2.1.5. Problemas de seguridad

Igual que muchas otras tecnologías, IoT también puede tener diferentes problemas de seguridad. Como ya sabemos, la red de IoT incluye gran variedad de dispositivos, aplicaciones y protocolos de comunicación. Todo esto forma un sistema bastante complejo a nivel global que requiere un constante mantenimiento y actualización. Sin embargo, si no se implementan las medidas de seguridad necesarias, los dispositivos IoT pueden llegar a tener un funcionamiento indebido o volverse vulnerables a los ataques de otras personas. Si esto ocurre, nuestra seguridad, datos personales o cualquier información sensible podría quedar expuesta y robada por otros. Muchas empresas tecnológicas como Telefónica están implementando diferentes estrategias y servicios de seguridad para IoT como la accesibilidad de red, la encriptación de datos o la autenticación de dispositivos.

2.2. Smart Cities

2.2.1. Introducción

Smart Cities (o ciudades inteligentes) son ciudades muy desarrolladas tecnológicamente que disponen de una amplia variedad de tecnologías, y sistemas de comunicación, algunas de las cuales orientadas al IoT. El término *ciudad inteligente*, se refiere a todo este conjunto de dispositivos, tecnologías o herramientas que juntos forman una gran infraestructura capaz de crear un bienestar para los ciudadanos. Como ya se sabe, el IoT es una red de dispositivos inteligentes capaces de funcionar con una intervención humana mínima. Aplicando el IoT a las ciudades modernas, nos da infinitas posibilidades para automatizar o mejorar diferentes procesos de nuestra vida cotidiana. Gracias a la evolución rápida de las tecnologías, las smart cities se vuelven cada vez más avanzadas y trabajan a nuestro favor.

2.2.2. ¿Por qué las ciudades inteligentes son importantes?

La urbanización es un fenómeno interminable que afecta a todos las personas de manera directa o indirecta, pudiendo reestructurar la vida social, económica y personal de todas las personas en una sociedad. Hoy en día, más de 55% de la población mundial vive en las ciudades con la expectativa de alcanzar los 65% para el año 2050[5]. El crecimiento de la población mundial y la urbanización requieren soluciones rápidas y eficientes para gestionar los diferentes procesos en las ciudades grandes o avanzadas. El objetivo es no solo conseguir un mejor control sobre una ciudad, sino también satisfacer las diferentes necesidades de los ciudadanos, sean particulares o empresas. La sostenibilidad ambiental, social y económica es imprescindible para seguir el ritmo de esta rápida expansión que está poniendo a prueba los recursos de nuestras ciudades. Las grandes ciudades también juegan un papel importante en el cambio climático. Las tecnologías usadas en las smart cities podrían ayudar a las personas a controlar o disminuir la contaminación medioambiental. Por este motivo, es muy importante tener en cuenta el avance tecnológico y poder aplicar estas tecnologías en nuestras ciudades para automatizar los diferentes procesos y facilitar nuestra vida cotidiana.

2.2.3. Aplicaciones del IOT en las Smart Cities

Las tecnologías de las smart cities tienen una amplia variedad de aplicaciones en diferentes sectores incluyendo casas(*domótica*), oficinas, tiendas, fábricas, automatización y control de tráfico, el transporte público y muchos más.

A nivel de desarrollo industrial, el Internet de las cosas en la Smart City ha impulsado el desarrollo de nuevos modelos de producción que recopilan datos de cómo se mueve una ciudad, y una sociedad en concreto, para establecer métodos eficientes con los cuales se consiga hacer un uso correcto de los recursos de la ciudad. La finalidad es desarrollar modelos de sostenibilidad de consumo y producción para impulsar la economía. En las ciudades inteligentes el IoT se implanta mediante sensores u otros dispositivos, dándole importancia a los servicios públicos como el transporte, la iluminación, sistemas de riego, recogida de residuos, etc. Esto permitirá un manejo más eficiente de los recursos, reducir el gasto público y la contaminación medioambiental. Las ciudades inteligentes deben apostar por modelos autosustentables en cuanto a energía y uso de los recursos, a través del IoT. Gracias a esto se mejorará la calidad de vida tanto de los ciudadanos como del entorno en el que viven. A continuación, veremos algunas aplicaciones concretas del IOT en las Smart Cities.

- **IoT en nuestros hogares(Domótica)**

Se consideran los asistentes del hogar como Google Home o Alexa de Amazon, a medida que se utilizan más estos dispositivos con IoT, estos llevarán a cabo una mayor cantidad de tareas automatizadas en nuestras vidas.

- **Blockchain**

Blockchain es un conjunto de tecnologías que combinadas hacen posible que ordenadores y otros dispositivos puedan gestionar su información compartiendo un registro distribuido, descentralizado y sincronizado entre todos ellos, en vez de hacer uso de las tradicionales bases de datos.

- **Transporte**

Los automóviles y el transporte público también están afectados por el IOT. Los coches se hacen más sofisticados e inteligentes, equipados con una variedad de sensores y dispositivos. Lo último en tecnología son los coches autónomos de Tesla. Según la empresa, estos coches pueden moverse libremente por la ciudad(o fuera de ella) sin intervención humana. Gracias al piloto automático, el coche puede acelerar, girar o frenar totalmente solo[6]. El IOT tiene un uso amplio en el transporte público también. En muchas ciudades, los vehículos del transporte público están equipados con

diferentes sensores y dispositivos de seguimiento que indican a los usuarios el estado o la ubicación de este. Gracias a estas tecnologías, el usuario puede consultar desde su teléfono móvil cuánto tardará el próximo autobús o tren en llegar. Otro ejemplo interesante son los trenes autónomos que se hacen cada vez más populares. Estos trenes pueden moverse de una estación a otra completamente solos y sin la intervención humana, es decir, dentro del tren no hay un maquinista. El tren está controlado por un conjunto de tecnologías avanzadas incluyendo una inteligencia artificial y una variedad de sensores. De momento los trenes recorren distancias cortas, pero son el primer paso hacia una automatización más avanzada del transporte público. Este tipo de transporte ya se está aplicando en ciudades como Barcelona[7]. En la imagen que aparece a continuación se puede observar un esquema de la conectividad IoT del transporte público. Los sensores que lleva el vehículo se comunican con la nube a través de Internet y los servicios de IoT.

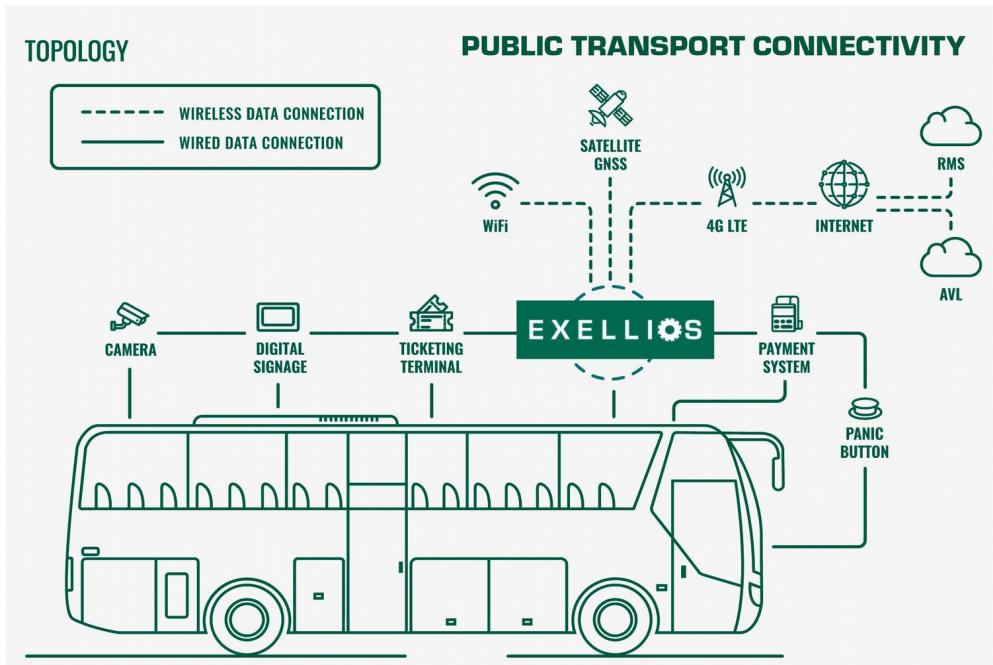


Figura 6: Esquema de la conectividad IoT del transporte público
Fuente: <https://www.exellios.com>

- **Big Data**

Gracias a la cobertura de los smartphones podemos identificar por colectivos cómo se mueven los ciudadanos y turistas, como el proyecto de LUCA[8] de Telefónica que identificó a los asistentes que pasaron la Nochevieja en la puerta del Sol de Madrid.

2.3. Principales tecnologías de comunicación inalámbrica IoT para las Smart Cities

En este apartado se profundizará en las principales tecnologías del IOT usadas en las *smart cities*. Puesto que IOT y las smart cities incluyen una amplia gama de diferentes tecnologías, solo voy a mencionar algunas de las tecnologías inalámbricas más importantes y usadas.

2.3.1. LORA(Long Range)

2.3.1.1. Introducción

LORA es una tecnología de comunicación inalámbrica basada en modulación de espectro ensanchado que deriva de la señal chirp. Esta tecnología incluye un protocolo de comunicación y un dispositivo físico basado en este protocolo. Igual que cualquier otra tecnología de transmisión inalámbrica, LORA también utiliza una modulación de la señal. Tecnologías como Wifi o Bluetooth utilizan frecuencias muy altas, 2,4 GHz para el Bluetooth y 2,4 GHz o 5 GHz para las señales de Wifi. Estas características permiten enviar un gran volumen de datos a una corta distancia(menos de 50 metros). La principal característica de la tecnología LORA es el bajo consumo(menos de 20 mW) y la gran distancia de transmisión(hasta 15 km). LORA utiliza frecuencias más bajas que el Wifi(433 MHz en Asia y China, 868 MHz en Europa, y 915 MHz en EEUU) lo que le permite enviar un pequeño volumen de datos pero a distancias mucho más grandes.

Con la configuración adecuada, un módulo LORA puede enviar la información a unos 10 km aproximadamente. Esto se puede conseguir si no hay obstáculos en el medio (edificios, montañas, etc). En una ciudad, existe una gran variedad de obstáculos que impiden a la señal que se propague a grandes distancias. Para conseguir una comunicación más estable dentro de un área, se tiene que establecer una red de diferentes módulos LORA llamados repetidores o Gateways[9]. Las ciudades inteligentes ya disponen de estas redes llamadas LORAWAN[10]. Esta es una tecnología de comunicación inalámbrica basada en LPWAN[11](redes de baja potencia y área amplia). Mientras que LPWAN engloba a un grupo más grande de dispositivos de bajo consumo, LORAWAN está explícitamente orientada a comunicaciones tipo LORA. Como el nombre indica, estas tecnologías están específicamente diseñadas para interconectar dispositivos de bajo consumo a un nivel local o global. Una conexión a nivel global permite al usuario acceder a los dispositivos o sensores desde cualquier lugar del planeta, ya que estos están conectados a Internet junto con otros dispositivos como teléfonos móviles, ordenadores, etc. Esto quiere decir que los dispositivos LoRa

pueden comunicarse con cualquier otro dispositivo(si este también está conectado a Internet) de manera autónoma, o recibir y enviar datos a petición del usuario.

En la siguiente imagen se puede apreciar la estructura básica de una red LORAWAN. Esta red se puede dividir en tres partes principales:

- Nodo final: Nodo equipado con uno o varios sensores y un transmisor LoRa. Este nodo envía la información al Gateway.
- Gateway: Este dispositivo recibe los datos del nodo final y los reenvía a un servidor usando una conexión Wifi, Ethernet o 3G.
- Servidor: Recibe y almacena los datos que envía el Gateway.

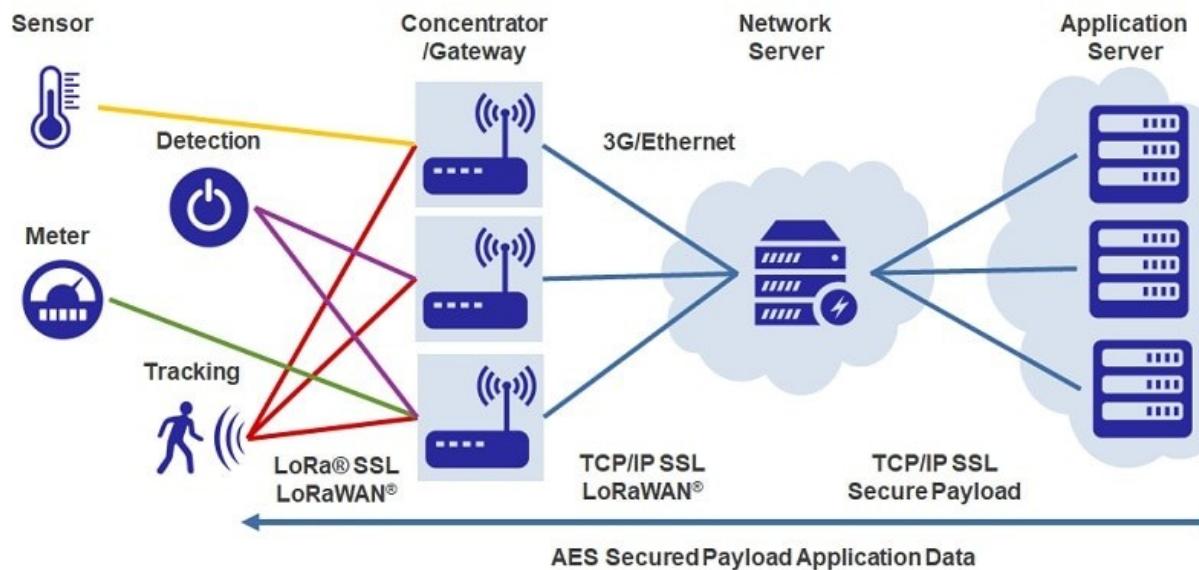


Figura 7: Esquema de la una red LORAWAN

Fuente: <https://www.renesas.com/tw/en/application/communications/lora-solutions>

2.3.1.2. Aplicaciones

La tecnología LORA tiene un uso amplio, no solo en las smart cities, sino también en la agricultura, domótica, fábricas, transporte y muchos más. Como se ha visto anteriormente, las principales ventajas de LORA son el bajo consumo y el largo alcance, lo que permite enviar datos a grandes distancias durante un largo periodo de tiempo. Con la configuración adecuada, un módulo LORA puede enviar datos a más de 10km y la batería podría durar años. La ventaja de las ciudades inteligentes es que ya disponen de redes tipo LPWAN, que permiten interconectar a diferentes dispositivos de bajo consumo incluyendo LORA, Sigfox, LTE, y otros. Gracias a estas tecnologías, se pueden monitorear las ciudades con el fin de obtener datos sobre el tráfico, contaminación, el tiempo, etc. También se pueden mejorar los servicios de algunos

sectores como el transporte público, autovías, fábricas u otras empresas, incluso podemos aplicarlas en nuestros hogares.

Sin embargo, en una zona rural, como por ejemplo un campo o una granja, el acceso a una red LORAWAN podría estar limitado. Esto quiere decir que sería necesario instalar una red de sensores propia. La ventaja en este caso es que se puede adaptar según la necesidad que uno tiene. Sin embargo, el usuario tiene que encargarse de la instalación desde cero y el mantenimiento de la misma. La falta de una fuente de alimentación en un campo o bosque es otro factor que se tiene que tener en cuenta. Por lo tanto, la única forma de alimentar los dispositivos es el uso de una batería. En una situación como esta, sería interesante el monitoreo de un área de varias hectáreas durante un largo periodo de tiempo. La naturaleza de la tecnología LORA la convierte en el candidato perfecto para esta situación debido a su bajo consumo y largo alcance.

A continuación se pueden observar algunas de las aplicaciones más destacadas de las redes LORAWAN. Tal y como se puede ver, estas no solo se aplican en las smart cities o la agricultura, también ocupan un lugar importante en la gestión energética, logística, transporte, fábricas y muchos sectores más.

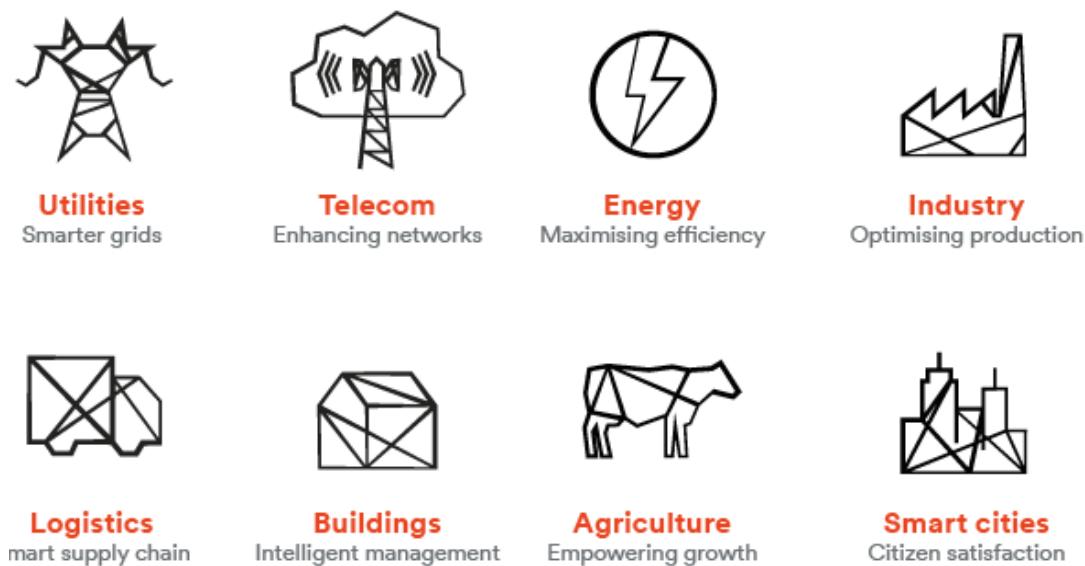


Figura 8: Las aplicaciones más destacadas de LORAWAN
Fuente: <http://texascom.co.id/lorawan/>

2.3.1.3. Características y ventajas

Como ya se mencionó anteriormente, las características más destacadas de Lora son el bajo consumo y el largo alcance. Se tiene que tener en cuenta que para enviar datos

a gran distancia, es necesario usar bajas frecuencias, es decir, cuanto más lejos se desea llegar, más bajas frecuencias se tienen que usar.

Sin embargo, LORA destaca con otras características[12] que compensan esta desventaja:

- Alta tolerancia a las interferencias
- Alta sensibilidad para recibir datos (-168 dBm)
- Basado en modulación “chirp”
- Bajo Consumo (hasta 10 años con una batería)
- Largo alcance: hasta 15 km
- Baja transferencia de datos (hasta 255 bytes) por mensaje
- Conexión punto a punto, o punto a multipunto en algunos módulos
- Frecuencias de trabajo: 868 MHz en Europa, 915 MHz en América, y 433 MHz en Asia.

LoRa opera en la banda de radio ISM (industrial, científica y médica) sin licencia que está disponible en todo el mundo. En toda Europa, se utiliza un plan de frecuencias sin licencia en torno a los 868 MHz, mientras que en Estados Unidos es 915 MHz y en Asia 433 MHz.

Podemos ver más detalles sobre las frecuencias a continuación:

Region	Frequency (MHz)
Asia	433
Europe, Russia, India, Africa (parts)	863-870
US	902-928

Region	Frequency (MHz)
Australia	915-928
Canada	779-787
China	779-787, 470-510

Figura 9: Frecuencias de LORA.
Fuente: <https://lora.readthedocs.io/en/latest/#id4>

2.3.1.4. Cobertura

Las redes de LORAWAN han ganado popularidad y han crecido exponencialmente en los últimos 5 años, tanto en número de dispositivos conectados como en número de redes alrededor del mundo. Actualmente hay más de 148 Redes Operadoras de LORAWAN en 162 países. Esto quiere decir que tenemos cobertura prácticamente en todo el planeta excepto algunos países y lugares muy remotos donde no hay antenas instaladas o la cobertura de estas no alcanza.

En la imagen siguiente podemos ver el mapa de los países que utilizan LORAWAN:

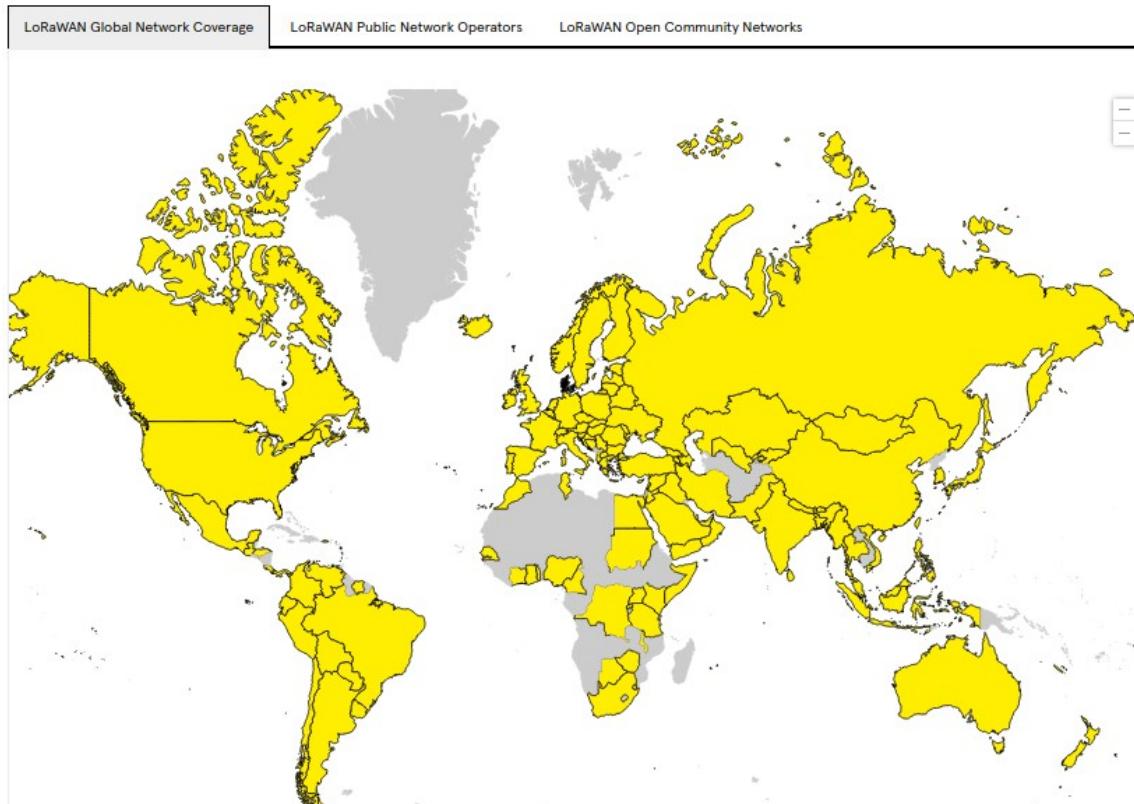


Figura 10: Países que utilizan la red LORAWAN
Fuente: <https://lora-alliance.org/>

2.3.2. SIGFOX

2.3.2.1. Introducción

Sigfox es una red de comunicaciones inalámbricas orientada al IOT y basada en el bajo consumo y largo alcance(LPWAN). El principal propósito de esta tecnología es conectar cualquier objeto de nuestro entorno a muy bajo coste y con un mínimo consumo energético. Es decir, Sigfox permite conectar una amplia variedad de dispositivos electrónicos(relojes, medidores eléctricos, alarmas y otros) al mundo de IoT. Al igual que LORA, Sigfox forma parte de la red LPWAN, y opera en varias frecuencias en diferentes continentes incluyendo Europa(868 MHz) y USA(902 MHz). La red se basa en una topología tipo estrella y requiere un operador móvil para gestionar el tráfico generado. La señal también puede ser usada fácilmente para cubrir grandes áreas y alcanzar objetos bajo tierra.

Sigfox también ha impulsado la creación de una tecnología llamada 0G. Esta es una red inalámbrica de bajo ancho de banda diseñada para conectar dispositivos IoT de bajo consumo y de baja potencia. La simplicidad de 0G permite a las ciudades evitar el uso de dispositivos que requieren tarjetas SIM. Según los expertos, la red 0G se puede considerar el futuro de las smart cities[13].

2.3.2.2. Aplicaciones

Al igual que LORA, Sigfox también tiene un amplio uso en el IOT. Se aplica en diferentes sectores de las smart cities incluyendo: domótica, fábricas, transporte, y muchos más. También ha encontrado aplicación en la industria agrícola donde esta tecnología se aprovecha para monitorear el clima y la humedad lo que permite optimizar muchos procesos y tomar unas decisiones más precisas, controladas e informadas. Los agricultores usan diferentes sensores, hardware automatizado y tecnología de tasa variable, junto con otras tecnologías, para sondear la variabilidad del suelo, optimizar la eficiencia del agua, monitorear el ganado y, en última instancia, mejorar los rendimientos[14]. Sigfox tiene otra aplicación interesante en las fábricas que han integrado esta tecnología para automatizar y monitorear los diferentes procesos. Se han instalado diferentes sensores en las máquinas y áreas de difícil acceso. Esto permite detectar fallos y facilitar el mantenimiento.

Una de las principales ventajas de Sigfox es que no se necesita implementar una infraestructura de antenas y repetidores, sino que depende de la infraestructura de alguna operadora telefónica. Esto quiere decir que nos podemos conectar a la red de

Sigfox solo con un emisor(o nodo final) que envíe los datos a la red y sin la necesidad de una tarjeta SIM. Esto le permite tener una cobertura prácticamente mundial. La desventaja aquí es que Sigfox tiene que llegar a un acuerdo con la compañía telefónica en un cierto área para tener cobertura. Por este motivo, Sigfox es una de las tecnologías preferidas en muchos sectores del IoT, sobre todo cuando se necesitan soluciones eficientes, simples y de bajo coste.

2.3.2.3. Características

Como ya se mencionó anteriormente, unas de las principales características de Sigfox son el bajo consumo y largo alcance. Además, los dispositivos son sencillos y fáciles de instalar y usar. Sigfox utiliza la tecnología Ultra Narrow Band(UNB) que está diseñada para funcionar a bajas velocidades con una transferencia de datos de 10 a 1.000 bits por segundo. El estándar existente para las comunicaciones Sigfox soportan hasta 140 mensajes "uplinks" al día, de los cuales cada uno de ellos tienen un "payload" de 12 bytes en un ratio de más de 100 bytes por segundo. La cantidad de datos que permite enviar Sigfox es relativamente pequeña, sin embargo otras características como el largo alcance, bajo consumo y alto nivel de seguridad pueden compensar esta desventaja.

A continuación se destacan algunas de las características más importantes de Sigfox[15]:

- Dispositivo de muy bajo coste.
- Dispositivos de muy fácil instalación y sin cables. Mano de obra mínima.
- Red inalámbrica independiente (no se utiliza la red interna del cliente).
- Sistema de bajo consumo que funciona con batería (10 años de autonomía).
- Máxima seguridad, se envía el mensaje con 3 frecuencias distintas.

Sigfox tiene una arquitectura bastante parecida a la de Lora. De hecho, se puede dividir en los mismos componentes principales:

- Nodo Final(nodo emisor equipado con sensores)
- Antena retransmisora
- Servidor(IOT Cloud)
- Cliente

En la siguiente imagen se puede apreciar con más detalle la estructura de una red Sigfox.

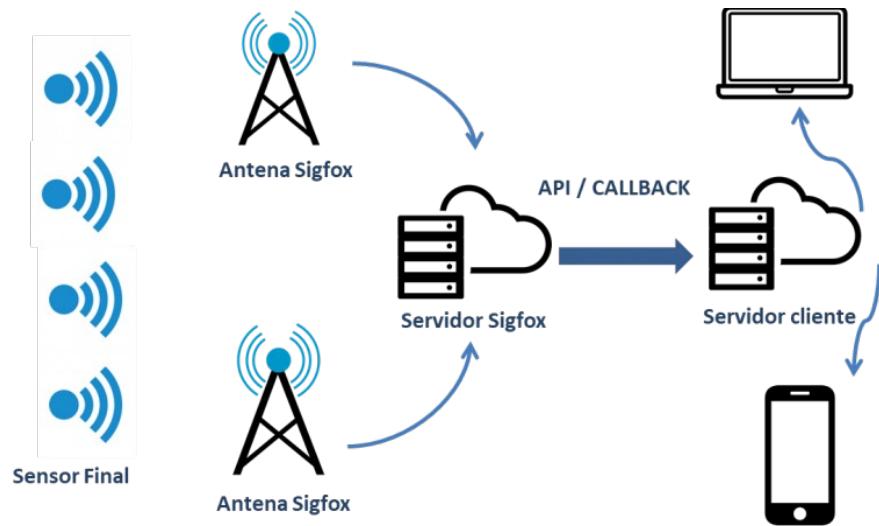


Figura 11: Estructura de la red Sigfox

Fuente: <http://www.dset-energy.com/2019/06/05/tecnologia-sigfox/>

2.3.2.4. Cobertura

Según SIGFOX, su red tiene una cobertura en 72 países alrededor del mundo con la expectativa de cubrir 100% del planeta en los próximos años. Sigfox no tiene una infraestructura propia sino que depende de la red de alguna compañía telefónica. Por una parte, esto le permite tener una cobertura mundial, ya que hoy en día las operadoras móviles tienen una cobertura por todo el planeta, pero para conseguirlo, es necesario el permiso de dicha operadora. Esto puede limitar la cobertura de Sigfox en algunas áreas. En la imagen que aparece a continuación, se puede ver la cobertura que tiene Sigfox a nivel mundial. En la mayor parte de Europa, SIGFOX ya tiene cobertura y en el resto de países esta tecnología está en desarrollo. En algunas zonas de los Estados Unidos, Sigfox tiene cobertura, sin embargo esta ha sido restringida en algunas regiones por lo que no ha podido ganar tanta popularidad ahí[16].

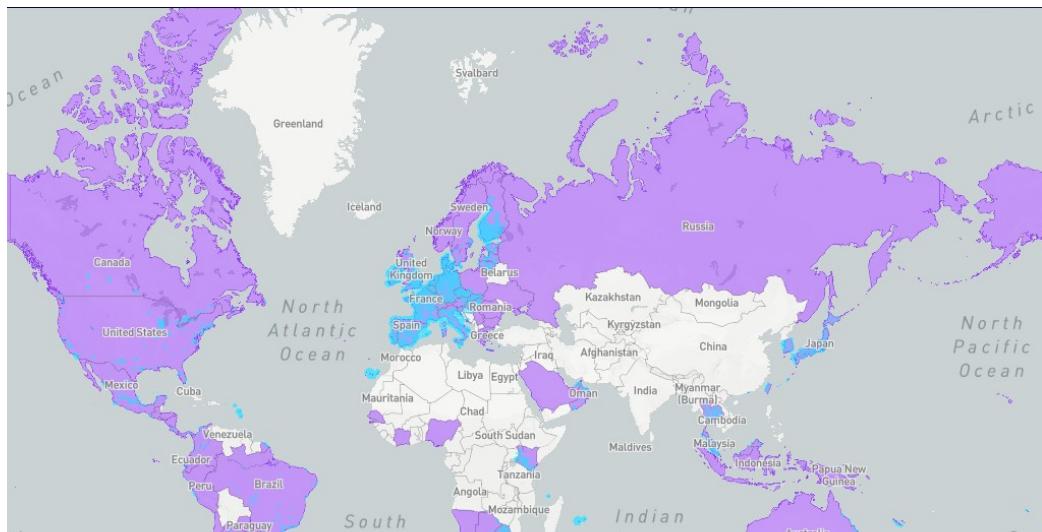


Figura 12: Cobertura mundial de SIGFOX

Fuente: <https://www.sigfox.com/en/coverage>

2.3.3. GSM

2.3.3.1. Introducción

El sistema global para las comunicaciones móviles o GSM es una red de comunicación de telefonía móvil digital. Hoy en día, GSM es una de las tecnologías de comunicación más usadas a nivel mundial. Gracias al GSM, se ha hecho posible la comunicación a grandes distancias, navegar en Internet, intercambiar datos y mucho más. GSM también tiene un uso amplio en el mundo del IOT. Muchos dispositivos utilizan esta tecnología para conseguir una cobertura a nivel mundial. Mediante la comunicación por GSM, el usuario puede acceder a un dispositivo o recibir alguna información de este mediante un SMS, o una petición HTTP/HTTPS que es ampliamente usada en la mayoría de las comunicaciones de Internet basadas en la arquitectura tipo cliente-servidor. Hoy con el avance de las tecnologías, y sobre todo del GSM, el acceso a Internet desde un teléfono móvil se ha hecho más fácil y además podemos navegar con una velocidad cada vez más alta. Tal y como se ha visto en el capítulo anterior, la tecnología Sigfox utiliza la infraestructura del GSM para interconectar sus dispositivos. Por estos motivos, sería muy razonable aplicar el GSM en los proyectos de IOT.

2.3.3.2. Características y arquitectura

Las redes de telefonía móvil se basan en el concepto de **celdas**, es decir zonas circulares que se superponen para cubrir un área geográfica. Las redes celulares se basan en el uso de un transmisor-receptor central en cada celda, denominado *estación base* (o *Estación base transceptora, BTS*). En zonas urbanas muy pobladas hay celdas con un radio de unos cientos de metros mientras que en zonas rurales hay celdas enormes de hasta 30 kilómetros que proporcionan cobertura[17].

El sistema GSM posee una serie de funcionalidades, que pueden ser implementadas por los operadores en sus redes. Las varias características incluyen:

- Posibilidad de usar el terminal y la tarjeta SIM en redes GSM de otros países (roaming).
- Servicio de mensajes cortos (SMS) a través del que pueden ser enviadas y recibidos mensajes con hasta 126 caracteres.
- Las llamadas son encriptadas, lo que impide que sean escuchadas por otros.
- Posibilidad de impedir la recepción / transmisión de ciertas llamadas.
- Navegar en Internet mediante el servicio GPRS.

- Opera en diferentes bandas de frecuencia:
 - GSM-900(880-915 MHz)
 - GSM-1800(1710-1785 MHz)
 - EGSM-900(880.0–914.8 MHz)
 - GSM-850(824.0–849.0 MHz)
 - GSM-1900(1850.0–1910.0 MHz)

La arquitectura GSM se compone básicamente de tres elementos:

- **MS:** Mobile Station - el teléfono móvil o el cliente
- **BSS:** Base Station Subsystem - la celda o la antena.
- **NSS:** Network & Switching Subsystem - red de servidores que procesan las peticiones de los clientes.

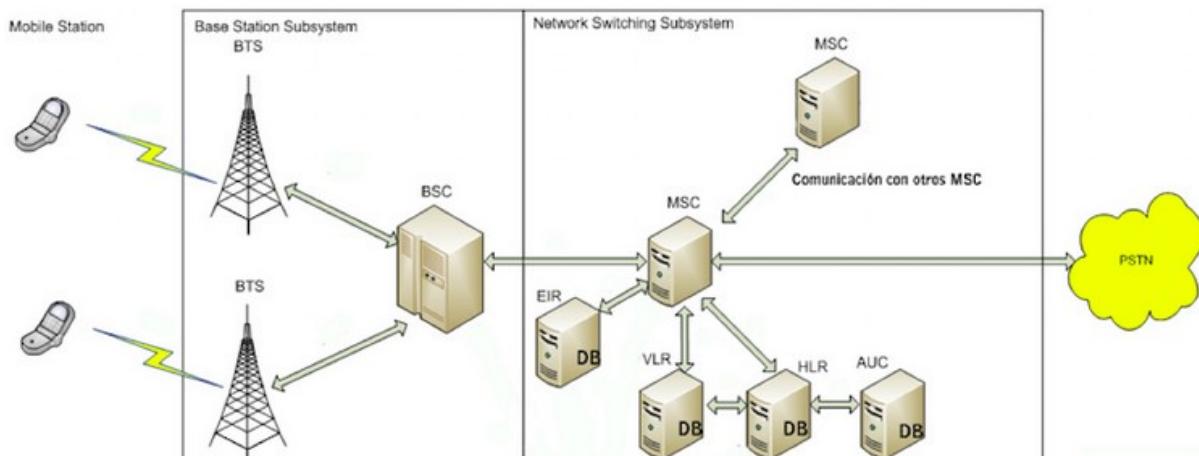


Figura 13: Arquitectura de la red GSM
Fuente: <https://mastermoviles.gitbook.io/>

2.3.3.3. Aplicaciones

Inicialmente, GSM fue creada para realizar llamadas de voz mediante los teléfonos móviles. Con la evolución de esta tecnología se añadían diferentes funcionalidades que permiten a los usuarios navegar en Internet, intercambiar datos de multimedia, hacer videollamadas y muchos más. Como ya mencionamos antes, GSM también ha encontrado su aplicación en el ámbito del IOT. Una de las grandes ventajas del GSM es la cobertura, que actualmente se extiende prácticamente a nivel mundial. Por otra parte, la naturaleza de IOT implica interconectar diferentes dispositivos electrónicos mediante la red de Internet y poder acceder a estos desde cualquier lugar del mundo.

Aunque GSM no es el único candidato que ofrece conexión a Internet, es uno de los mejores específicamente por su cobertura. En las zonas urbanas, disponemos de una

variedad de tecnologías de comunicación como Wifi, Bluetooth, Lora y muchas más. Sin embargo, las zonas remotas o interurbanas, carecen de este tipo de tecnologías lo que nos deja pocas opciones para poder comunicarnos. En las zonas remotas, la potencia de la señal GSM puede ser afectada, por lo cual la velocidad de transmisión de los datos sería más baja. En otras palabras, en vez de tener una conexión 4G o 5G, tendremos una de 2G. Sin embargo, los dispositivos IOT suelen enviar datos de un pequeño volumen, a menudo menos de 240 - 250 bytes por transmisión. En este caso una cobertura de 2G nos sirve perfectamente, ya que a parte de enviar paquetes de un pequeño volumen, también haremos pocas transmisiones diarias. La calidad de la señal y la cobertura dependen no solo de nuestra ubicación actual, sino también de la operadora GSM, la tarjeta SIM que usamos, y otros factores.

2.3.3.4. Cobertura

En la actualidad, la tecnología GSM tiene una cobertura mundial. Sin embargo, el tipo y calidad de la señal puede variar según la ubicación, operadora móvil u otros factores. La imagen que aparece a continuación, representa la cobertura del GSM en la actualidad.

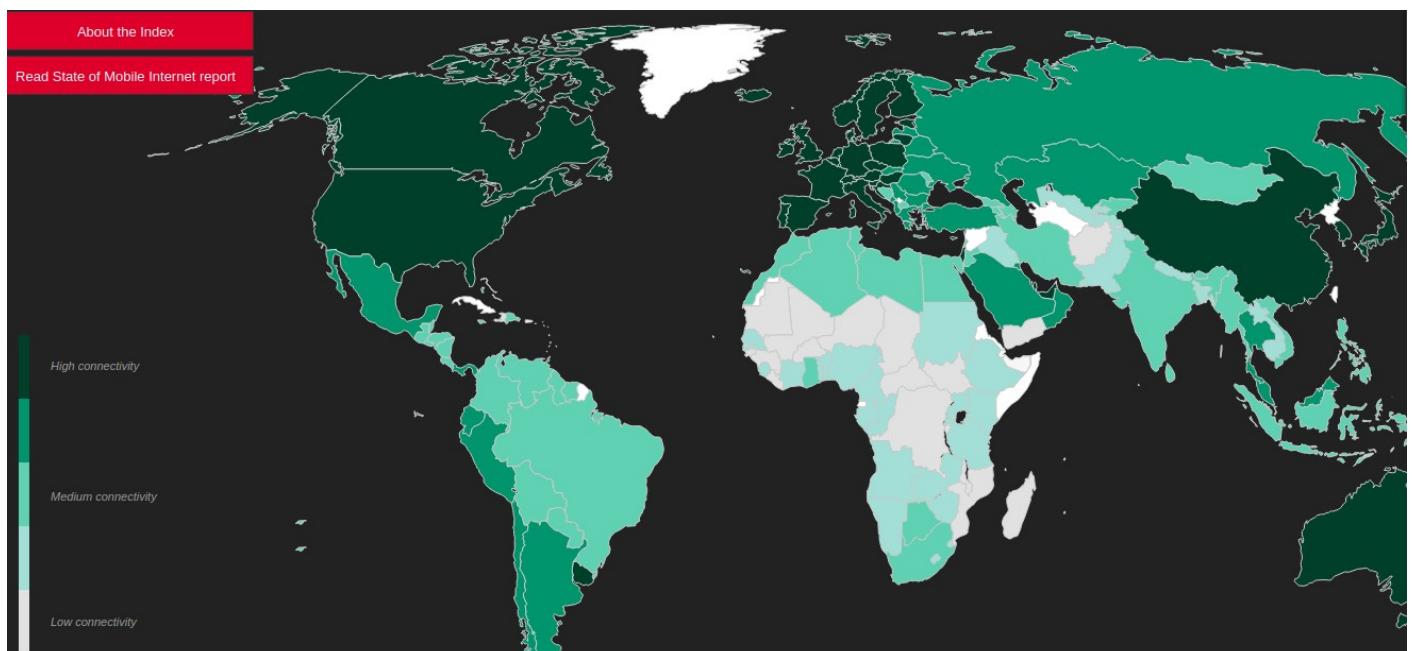


Figura 14: Cobertura mundial del GSM
Fuente: <https://www.mobileconnectivityindex.com/#year=2019>

2.5. Análisis comparativo de tecnologías inalámbricas

En conclusión a este capítulo, se realizará una comparación entre las tecnologías de comunicación que se han visto hasta ahora. Se destacarán las principales ventajas, desventajas y características de cada una. Además, se mencionarán las tecnologías seleccionadas para este proyecto y se justificará el motivo.

En la siguiente tabla se destacan las principales tecnologías de comunicación usadas en el mundo del IOT:

2.5.1. Comparación

Tecnología	Cobertura Mundial	Conectividad por países	Tamaño de paquete	Alcance(km)	Límite mensajes
LORA	No	167	255 bytes	10 - 20	30sec uplink/day
Sigfox	No	70	12 bytes	30 -50	4/día
GSM	Sí	mundial	ilimitado	<10km. Según la operadora móvil.	ilimitado

Tabla 1: Comparación de tecnologías IOT
Fuente: <https://bottrueactivities.com/comparativa-entre-sigfox-y-lorawan/>

Tal y como se puede observar, la tecnología Lora destaca con algunas características como el tamaño del paquete de datos, conectividad por número de países y el bajo consumo. Además, a diferencia de Sigfox, Lora no depende de la infraestructura de una operadora móvil. Es cierto que los transmisores Lora pueden depender de la infraestructura de LORAWAN, pero esto puede ocurrir solo en caso de que el usuario no instale su propia red de dispositivos. El objetivo de este proyecto es instalar y configurar una red propia por lo que se puede ignorar la infraestructura de LORAWAN.

Otra ventaja es la facilidad de instalación y uso. A nivel regional se pueden compartir datos entre los diferentes dispositivos con gran facilidad, incluso si estos están separados por varios kilómetros de distancia y sin depender de ninguna operadora móvil o infraestructura.

Otra tecnología interesante y con muchas ventajas es el GSM. Como ya se ha visto en el estudio previo de este capítulo, GSM tiene una cobertura mundial y permite enviar datos de diferentes tipos como SMS, MMS, llamadas de voz y video, pero lo más importante es la conexión a Internet. Gracias a estas características, no existe prácticamente ningún límite a la hora de compartir datos. Se puede enviar cualquier

tipo de información de cualquier tamaño. Uno de los puntos fuertes del GSM frente a las tecnologías mencionadas, es la cobertura mundial y por este motivo es una de las tecnologías seleccionadas para el proyecto.

2.5.2. Integración de datos

Sigfox ofrece una API muy simple para instalar el módulo de radio y conectarlo a la red de Sigfox. Es decir, no tenemos que instalar redes o repetidores propios, ya que Sigfox usa la infraestructura de alguna operadora móvil. Sin embargo, LoRa ofrece una API de bajo nivel altamente configurable, que posibilita diferentes optimizaciones. Por lo tanto, la integración del módulo de radio LoRa es más complicada que SigFox, pero mucho más configurable.

Por otra parte, GSM sí depende de alguna red móvil pero gracias a la tarjeta SIM, nos podemos conectar prácticamente a cualquier red y conseguir una cobertura mundial.

2.5.3. Conclusión

Tras realizar el estudio en este capítulo, se puede concluir que las tecnologías LORA y GSM son las más adecuadas para este proyecto ya que son las que mejor se adaptan. Por una parte Lora permite construir una red propia de transmisores y cubrir un área de varias hectáreas. Por otra parte, la tecnología GSM permite acceso a Internet con el fin de enviar los datos a un servidor o plataforma IOT.

Página en blanco intencional

3. DESARROLLO DEL PROYECTO

3.1 Concepto

La tecnología desarrollada en este proyecto permite establecer una red de transmisores basados en la tecnología LORA. Cada red está basada en la topología estrella, es decir los transmisores o *nodos finales* envían los paquetes a un nodo central o *Gateway*. Este nodo procesa los paquetes entrantes y mediante un módulo GSM los envía a un servidor o en este caso una plataforma IOT llamada *thingsboard*. Como ya se mencionó anteriormente, la idea de este proyecto es no solo el monitoreo de diferentes procesos medioambientales como la temperatura, presión atmosférica o humedad, sino también conseguir una cobertura mundial para cada red de transmisores instalado. En otras palabras, una red tiene que ser capaz de conectarse a Internet sin importar su ubicación geográfica. Esto se puede conseguir gracias a la tecnología GSM que como ya se ha visto en el capítulo anterior, hoy en día se ha extendido tanto que un teléfono móvil tiene cobertura prácticamente en cualquier lugar del planeta.

Es cierto que en la actualidad existen proyectos similares[18][19], sin embargo estos no cumplen con el propósito o la necesidad que pueden llegar a tener algunos usuarios. Por ejemplo, algunas ventajas clave de este proyecto son la simplicidad, escalabilidad y robustez. Esto permite configurar y programar los diferentes dispositivos según la necesidad del usuario. Otra ventaja muy importante es el bajo consumo que tienen los dispositivos lo que permite alimentar los circuitos con una pila de 5V, sin la necesidad de usar fuentes de alimentación o transformadores. En otras palabras, se puede instalar la red de sensores en cualquier lugar, tanto zonas urbanas como rurales. Esto incluye zonas remotas, montañas, campos, lugares de difícil acceso. La instalación se puede hacer tanto en el interior como en el exterior.

3.2 Metodología

El desarrollo del proyecto está dividido en 4 partes principales:

1. **Investigación:** Estudio del mercado y las tecnologías disponibles
2. **Estructurar el proyecto:** Decidir qué tecnologías y herramientas se van a usar
3. **Integración del proyecto:** Montar el circuito, instalar y aplicar las herramientas
4. **Fase de pruebas:** Lectura de los sensores, prueba de rendimiento, envío de paquetes a la plataforma IOT

3.2.1. Fase 1: Investigación

En esta fase se ha realizado un estudio sobre las tecnologías disponibles en el ámbito del IOT teniendo en cuenta las más usadas que tienen una tendencia a crecer en los próximos años. El estudio está centrado en las tecnologías de comunicación inalámbrica puesto que son de vital importancia para este proyecto. Entre las diferentes tecnologías disponibles, se han tenido que escoger las que mejor se adaptan a este proyecto y las necesidades que se quieren cubrir.

Para cumplir este propósito, se ha tenido que hacer un enfoque en tecnologías con unas determinadas características como rango de alcance, tamaño de paquete de datos, posibilidad de una cobertura regional o mundial, o conectividad a Internet. Las características más importantes de las diferentes tecnologías son la cobertura y la posibilidad de acceder a Internet, puesto que el proyecto está estrechamente relacionado con estas. En caso de que alguna tecnología o dispositivo no tengan una de estas dos características, o experimentan dificultades en cumplirlas, sería muy difícil o imposible la implementación del proyecto. Tras el estudio realizado, se ha llegado a la conclusión que la combinación de varias tecnologías llevarían a la integración de este proyecto en el mundo de IOT. Como ya se mencionó en el capítulo anterior, las tecnologías seleccionadas son Lora y GSM. Estas se combinan con una plataforma IOT donde se almacenan los datos enviados por los transmisores de Lora(o nodos finales).

Sin embargo, si se utilizan estas tecnologías por separado, no cumplirían con el propósito del proyecto. Por una parte, Lora no tiene acceso directo a Internet. Por otra parte, no es buena idea basarse solamente en comunicación por GSM ya que al acumular una cierta cantidad de clientes GSM en una determinada zona, se puede llegar a congestionar el tráfico de datos en este área.

Por este motivo, una de las mejores soluciones es implementar una red de la topología estrella, que puede tener hasta varios centenares de nodos finales conectados a un único nodo central(o Gateway). Este último se encargaría de transmitir la información de los terminales a Internet mediante un protocolo HTTP.

3.2.2. Fase 2: Estructurar el proyecto

En esta fase se explicará detalladamente los componentes que se van a usar en este proyecto. Las tecnologías ya están definidas, que en este caso serán: Lora, GSM y una plataforma IOT. También se va a especificar qué tipo o modelo de cada módulo se van a usar, las herramientas necesarias y cómo se relacionan entre sí. Además, se hará una visión genérica sobre el proyecto para entender mejor el concepto y su estructura.

3.2.2.1. Arquitectura

El proyecto está basado en la arquitectura tipo cliente - servidor. La parte del cliente es la red de sensores y transmisores Lora y la parte del servidor es la plataforma IOT que almacena los datos enviados por los terminales. A continuación se puede apreciar el diagrama de arquitectura que destaca los componentes principales del proyecto.

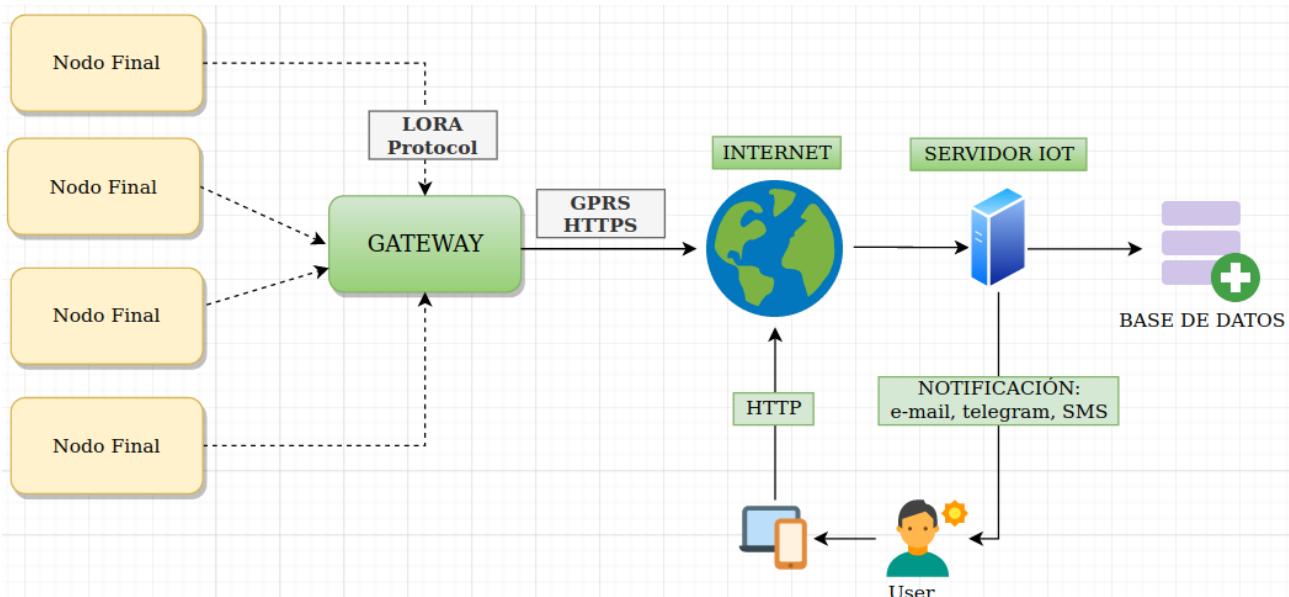


Figura 15: Diagrama de arquitectura
Fuente: Elaboración propia

En este diagrama se puede observar una clara división entre las partes del cliente y el servidor. La parte del cliente incluye el nodo final y el Gateway. Es decir, una red de topología estrella instalada y configurada por el usuario. Dicha red dispone de un solo gateway y uno o varios nodos finales. La cantidad de nodos puede variar según la necesidad del usuario. Más detalles sobre esta red y sus limitaciones se darán en los próximos apartados. Cada nodo final dispone de uno o varios sensores y un transmisor LORA. El gateway dispone de un módulo Lora que sirve como receptor y un módulo GSM que envía los paquetes que ha recibido de los nodos terminales a la plataforma

IOT. Tanto los nodos finales como el gateway son dispositivos de muy bajo consumo y utilizan una pila de 5V DC lo que nos permite portabilidad y la posibilidad de instalar los dispositivos en cualquier lugar, incluso los poco accesibles.

En el siguiente diagrama se puede observar con más detalle la estructura de la parte del cliente, es decir el *Nodo Final* y el *Gateway*. En el nodo final, se pueden identificar claramente algunos componentes importantes como el microcontrolador, el emisor Lora y un conjunto de sensores. Componentes similares se pueden encontrar en el gateway, solo que este no dispone de sensores, sino de un módulo GSM. La comunicación entre los diferentes componentes dentro de cada módulo se realiza mediante diferentes protocolos de comunicación como el UART[20], I2C, SPI u otros. El tipo y modelo de los diferentes componentes no es de tal importancia, este diagrama solo es un ejemplo que representa los componentes que se han usado. Por lo tanto, se pueden usar la marca y el modelo que mejor se ajustan a la necesidad del usuario.

Los microcontroladores tanto de los nodos finales como de los gateways son programables. En este caso, se están usando unos modelos que se pueden programar con el compilador de Arduino, aunque también se puede usar otra plataforma o lenguaje de bajo nivel. El módulo del GSM ya viene programado previamente por el fabricante y dispone de un firmware que permite la comunicación con este mediante los comandos AT[21]

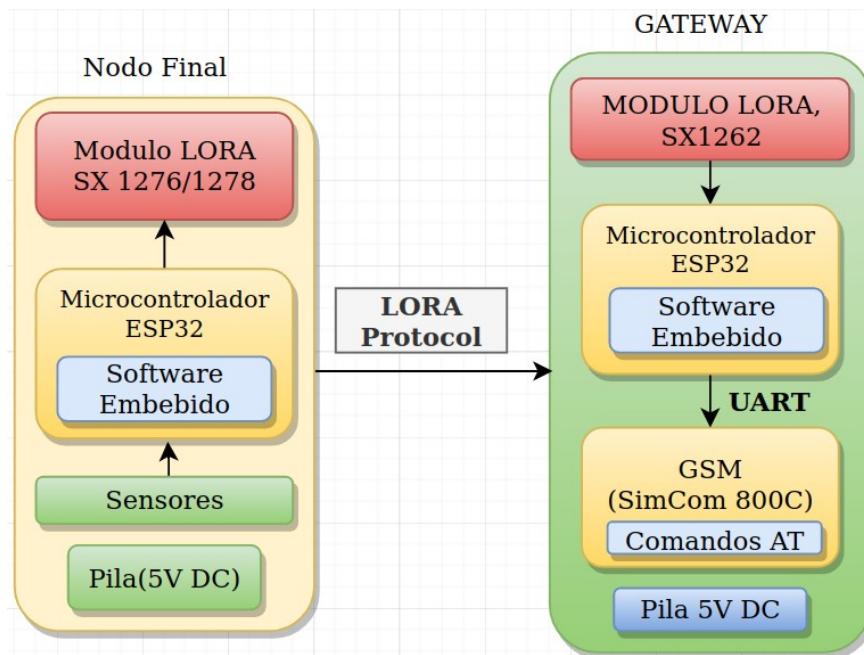


Figura 16: Diagrama de bloques
Fuente: Elaboración propia

En la parte del servidor se dispone de una plataforma IOT que mediante una API, recibe los mensajes enviados por el GSM y los almacena en una Base de Datos. En este proyecto se está usando la plataforma <https://demo.thingsboard.io>. Además, esta

plataforma ofrece una amplia variedad de funcionalidades como gestionar los datos entrantes escribiendo nuestro propio código, enviar notificaciones cuando suceda algún evento o visualizar los datos mediante diferentes tablas o gráficos. El motivo de escoger esta plataforma es que se adapta mejor al proyecto aunque cualquier plataforma similar podría servir.

3.2.2.2. Herramientas y componentes

A continuación se explicará detalladamente qué componentes y herramientas se han usado en el proyecto. En la Fase 3 se entrará en más detalle sobre cómo se configuran y comunican los diferentes módulos.

❖ Aplicaciones de Software:

- Arduino IDE
- Libelium Waspmove IDE
- Serial debugger: Moserial
- Sistema Operativo: Ubuntu
- Plataforma IOT: thingsboard.com
- Aplicación de mensajería: Telegram
- Multímetro USB: UM24C

● Serial debugger - Moserial

Esta aplicación se utiliza para comunicarse con sistemas embebidos a través del puerto UART. Se utiliza en sistemas de la familia UNIX y se puede instalar fácilmente ejecutando los siguientes comandos en la consola:

- `sudo apt-get update -y`
- `sudo apt-get install -y moserial`

Durante el desarrollo de este proyecto, el *moserial*[22] se ha utilizado para realizar las comunicaciones con las placas de desarrollo Lora y GSM. No es obligatorio el uso de esta aplicación en particular, el usuario puede aplicar otra cualquiera. Sin embargo, esta es la que mejor se adapta al desarrollo debido a su sencillez.

❖ **Materiales Hardware:**

- Nodo final/terminal: Heltec Lora Wifi Module(V2); Libelium Waspmotte
- Nodo receptor/Gateway: Heltec Lora, Wireless stick
- Módulo GSM: Simcom 800C
- USB to UART adapter
- Tarjeta SIM
- Fuente de alimentación - Adaptador o batería: 5V DC, 2A
- Cableado de pines
- Cables Micro USB
- Multímetro USB con Bluetooth

● **Nodo final**

En la siguiente imagen podemos ver el emisor Lora que se utiliza como nodo final en este proyecto. Aparte del módulo Lora, esta placa de desarrollo incluye el microcontrolador ESP32 que es fácil de programar a través del puerto USB utilizando plataformas como Arduino. Además, la placa tiene diferentes puertos como UART, I2C, SPI y otros que permiten conectar los diferentes sensores. Hay una gran variedad de placas similares así que no es obligatorio usar exactamente esta, sino que se puede elegir la que más se ajusta a una necesidad concreta.

❖ **Características:**

- Modelo: Heltec Lora ESP32 V2
- Comunicación inalámbrica: Lora(868MHz), Ble, Wifi
- LoraWAN Compatible
- Lora chip: SX1276/SX1278
- Buses de comunicación: UART, SPI, I2C, USB
- Microcontrolador: ESP32 200MHz
- Alimentación: 4.7V - 6V, 500mA
- Sitio web oficial: <https://heltec.org/project/wifi-lora-32/>

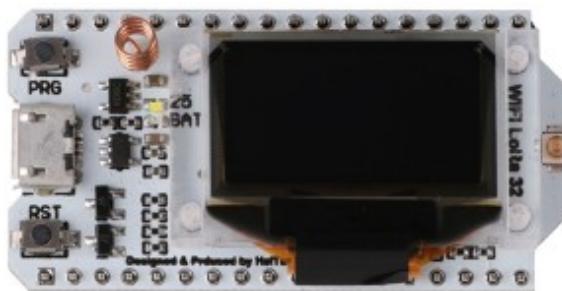


Figura 17: Fotografía del emisor Lora
Fuente: Elaboración propia

● **Gateway**

El Gateway está compuesto por dos módulos, el receptor Lora y el módulo GSM. En este caso, se ha usado una placa diferente para cada módulo aunque existen placas de desarrollo que incluyen ambos chips(GSM y Lora). Como ya se ha visto en el diagrama de bloques(figura 17) la placa de desarrollo de Lora dispone de un microcontrolador programable que permite la gestión de los paquetes que recibe este módulo. Programar el microcontrolador es muy fácil y se puede realizar mediante el compilador de Arduino u otro entorno de desarrollo similar. Además, este módulo es de muy bajo consumo por lo que se puede alimentar con una pila de 5V DC a través del puerto USB que lleva integrado. En la siguiente imagen, se puede ver el dispositivo Lora que se está usando como receptor en el Gateway:



Figura 18: Imagen del receptor Lora
Fuente: Elaboración propia

❖ Características:

- Modelo: Heltec Lora Wireless Stick
- Comunicación inalámbrica: Lora(868MHz), Ble, Wifi
- LoraWAN Compatible
- Lora chip: SX1276
- Buses de comunicación: UART, SPI, I2C, USB
- Microcontrolador: ESP32 240MHz
- Alimentación: 4.7V - 6V, 500mA
- Sitio web oficial: <https://heltec.org/project/wireless-stick/>

● GSM

El otro elemento importante que compone Gateway es el módulo GSM que aparece en la siguiente imagen:

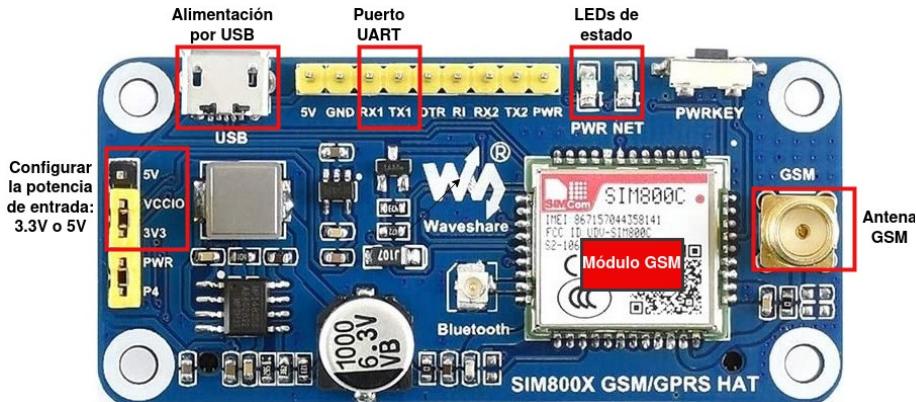


Figura 19: Fotografía del módulo GSM
Fuente: Elaboración propia

❖ Características

- Modelo de la placa: SIM800C Waveshare GPRS HAT
- Modelo chip GSM: Simcom 800C
- Bandas de trabajo: GSM 850/EGSM 900/DCS 1800/PCS 1900 MHz
- Conectividad: GPRS
- Bus de comunicación: UART(1200bps ~115200bps)
- Control via comandos AT
- Alimentación: 5V, 2A
- Sitio web oficial: https://www.waveshare.com/wiki/SIM800C_GSM/GPRS_HAT

Tal y como se ha mencionado anteriormente, el GSM se comunica con el receptor Lora a través del puerto UART. Algo muy importante que se tiene que tener en cuenta es que el módulo GSM aumenta su consumo de energía a la hora de buscar una red o enviar datos por el GPRS. Por este motivo, es imprescindible conectarlo a una fuente de alimentación distinta a la del módulo Lora. Una fuente del rango 3.4V - 4.4V es suficiente, aunque lo recomendable son 4V. Según la documentación oficial del módulo *Simcom800C*, si el voltaje no está dentro del rango recomendado el módulo se apagará automáticamente. Una transmisión podría causar caída de voltaje, así que la fuente de alimentación debería ser capaz de proporcionar suficiente corriente, hasta 2A. De lo contrario, el módulo se apagará debido a la caída de potencial[23].

Este rango del voltaje se aplicará si se trabaja directamente con el microchip de Simcom. Sin embargo, en este proyecto se utiliza una placa de desarrollo que a parte

del microchip del GSM, lleva un circuito integrado que controla el voltaje y tiene un consumo ligeramente elevado. Esta placa permite una alimentación de entrada de entre 3.3V - 5V, 2A.

Finalmente el GSM necesitará una tarjeta SIM para tener acceso a la red. Cualquier tarjeta SIM servirá siempre y cuando ésta permita el acceso a Internet y tenga cobertura en la zona donde se desea usarla.

- **Libelium Wasp mote**

Esta placa dispone de un módulo Lora y se puede usar como un nodo final. Este modelo lleva integrado un microcontrolador de *Atmel* lo que permite programar muy fácilmente mediante el entorno de desarrollo de Libelium. La estructura del proyecto y el protocolo de comunicación que se está usando, permiten incluir nodos terminales de cualquier tipo, incluyendo a este. Debido a que el modelo Wasp mote es ampliamente usado en el mundo del IOT, se ha decidido incluirlo en el prototipo. El uso de diferentes modelos de emisores Lora no debería influir de ninguna manera en el funcionamiento del Gateway ya que el protocolo de comunicación permite que cualquier dispositivo se conecte y comunique sin la necesidad de reconfigurar el receptor(o Gateway).

A continuación se destacan algunas de las características básicas de este módulo.

❖ **Características:**

- Modelo: Libelium Wasp mote
- Comunicación inalámbrica: Lora(868 - 900 MHz)
- LoraWAN Compatible
- Lora chip: SX1272
- Buses de comunicación: UART, SPI, USB
- Microcontrolador: ATmega1281, 14.74 MHz
- Alimentación: 5 V – 100 mA
- Sitio web oficial: <https://www.libelium.com/iot-products/wasp mote/>

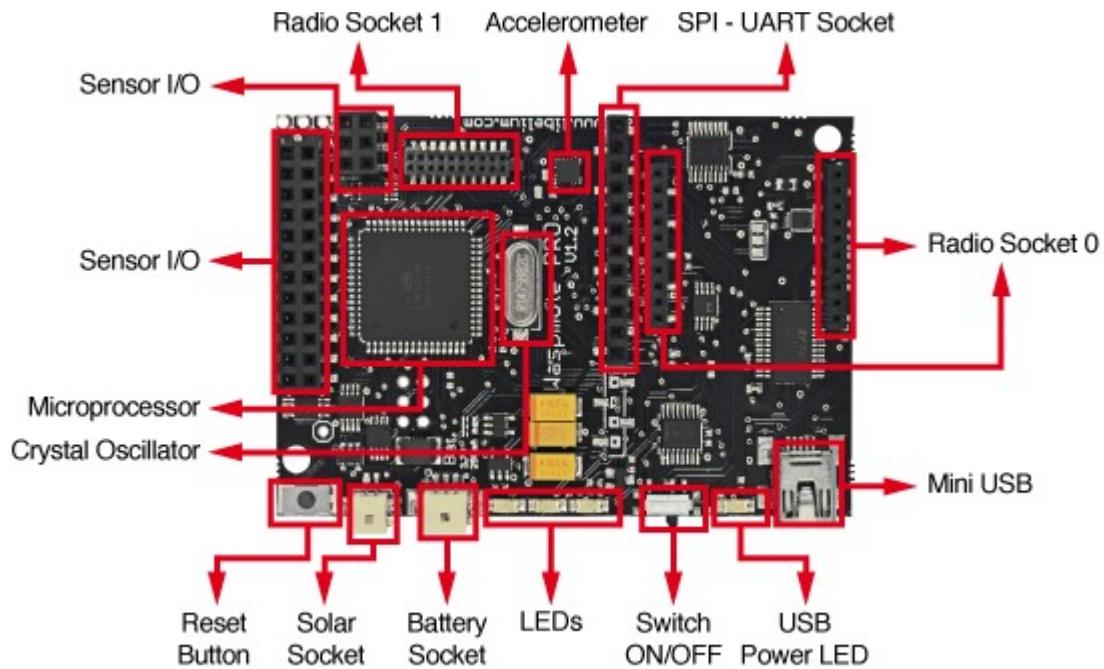


Figura 20: Fotografía del módulo Waspmovee

Fuente: <https://www.cooking-hacks.com>

• Serial debugger

Antes de conectar los módulos GSM y Lora, es necesario verificar que el GSM está configurado y funcionando correctamente. Es decir, se tiene que verificar su estado, configurar los Baudios(o frecuencia de trabajo), comprobar la conexión a Internet u otros parámetros. Para poder hacer esta prueba necesitaríamos un *serial debugger*, una aplicación que puede gestionar el tráfico de datos que pasa por el puerto UART. Primero debemos conectar el GSM al puerto USB del ordenador que en este caso se tiene que hacer mediante un dispositivo llamado conversor *UART a USB* que aparece en la imagen siguiente:



Figura 21: Fotografía del conversor UART - USB

Fuente: Elaboración propia

El propósito de este dispositivo es hacer posible la conexión entre un sistema embebido como un microcontrolador o en este caso el GSM y el ordenador a través del puerto USB. Los ordenadores modernos ya no disponen de un puerto serie y los dispositivos embebidos no siempre tienen puerto USB. Por otra parte, ambos puertos no son compatibles entre sí ya que utilizan protocolos de comunicación distintos. Así que la única forma de conectar un dispositivo embebido al ordenador es mediante este adaptador. También sería necesario usar algún software que nos permita enviar y recibir datos por la UART. En este caso se está usando la aplicación *moserial* aunque cualquier programa servirá perfectamente. En la Fase3 del desarrollo veremos con más detalle el proceso de depuración del GSM a través del puerto UART.

- **Multímetro - USB**

Para verificar el consumo de los diferentes dispositivos como el GSM o los módulos Lora, se utilizará un multímetro USB. La mayoría de los módulos permiten una alimentación a través del puerto USB así que lo más cómodo sería usar un multímetro de este tipo. Además, este modelo acepta conexión por Bluetooth desde un teléfono móvil lo que permite exportar y guardar las lecturas para poder hacer un posterior análisis. Más información sobre este dispositivo se puede encontrar en la documentación oficial[24].

En la siguiente captura aparece el multímetro usado en este proyecto:



Figura 22: USB multímetro

Fuente:

<https://files.banggood.com/2018/05/A3&A3-B.pdf>

3.2.2.3. Presupuesto

En la siguiente tabla se destaca el precio de cada uno de los elementos usados en el prototipo. Para este proyecto se han escogido componentes relativamente básicos y de un precio más bajo. Sin embargo, cualquiera de los componentes de hardware o software se puede sustituir por uno con más capacidad o características diferentes en función de la necesidad que se desea cubrir.

- **Componentes de Hardware:**

Componente	Precio €
Nofo final - Heltec	22
Nodo final: Libelium Wasp mote	153
Receptor Lora - Heltec	22
GSM - Simcom 800C	20
Tarjeta SIM	10
Serial Debugger	5
Powerbank	10
Micro USB cable	5
Multímetro USB	25
TOTAL €	272

Tabla 2: Presupuesto de los componentes usados

Fuente: Elaboración propia

- **Componentes de software**

Las aplicaciones de software utilizadas en este proyecto son gratuitas y no implican ningún coste adicional. El único servicio que puede tener algún coste es la plataforma IOT(*thingsboard*) que según el plan, tendrá un precio u otro. En este proyecto se está usando la versión demo que es gratuita pero limitada en algunos servicios.

La plataforma cumple con algunos de los requisitos más importantes. Por ejemplo, nos permite comunicarnos con esta a través de una API, enviar peticiones GET y POST mediante los protocolos HTTP/HTTPS, visualizar los datos mediante variedad de gráficos y gestionar los datos entrantes mediante algún script. Sin embargo, la versión demo de esta plataforma no permite la exportación de los datos a un fichero. Esto podría ser un obstáculo si queremos descargar los datos de un cierto periodo para poder analizarlos.

En la captura que aparece a continuación, se pueden ver más detalles sobre las diferencias entre el plan *demo* y *premium* de la plataforma *thingsboard*.

Customizable rule chains, widgets	✓		✓
MQTT, HTTP, CoAP, OPC-UA transport	✓		✓
Integrations with BigData systems	✓		✓
NB-IoT, SigFox, LoRaWAN support	Basic		Advanced
Rule Engine: Components <small>②</small>	Basic		Advanced
Entity groups <small>④</small>	✗		✓
Advanced RBAC for IoT <small>④</small>	✗		✓
Scheduler <small>④</small>	✗		✓
Reporting <small>④</small>	✗		✓
White-labeling <small>④</small>	✗		✓
CSV/XLS data export <small>④</small>	✗		✓

Figura 23: Tabla de comparación entre los planes de la plataforma thingsboard
Fuente: <https://thingsboard.io/products/thingsboard-pe/>

3.2.2.4. Competencia

En el mercado existen productos similares, tanto componentes individuales como kits de desarrollo. Cada uno de estos productos tiene unas características y precio distintos, sin embargo no todos se adaptan a este proyecto.

- **DockerPi, IOT Node**

Lo interesante de este módulo es que lleva los chips de Lora y GSM integrados en la misma placa de desarrollo. Sin embargo, esta placa no es programable sino que se conecta a otra placa programable como Arduino o Raspberry. Lo incomodo es que todas las comunicaciones se tienen que gestionar manualmente por UART, I2C o SPI, lo que implica más probabilidades de fallo.

En la siguiente imagen se puede apreciar de DockerPi placa de desarrollo:

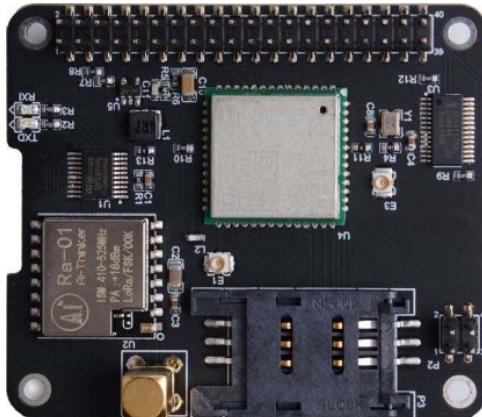


Figura 24: Placa de desarrollo DockerPI

Fuente: <https://media.digikey.com/>

- **LORAWAN Dragino Gateway**

Este gateway dispone de una gran variedad de funcionalidades avanzadas incluyendo conexión a la red de LORAWAN, Wifi, GSM, o Ethernet. Además, lleva un Firmware previamente grabado, sin embargo tiene algunas desventajas. La primera es que no se puede reprogramar con la misma facilidad que tienen los módulos Wasp mote o los microcontroladores ESP32. La segunda es que requiere una alimentación de 12V DC, que es un consumo relativamente alto comparado con los módulos usados en este proyecto. A parte, este gateway tiene un precio bastante elevado lo que no lo hace apto para ciertos proyectos.



Figura 25: LORAWAN Dragino Gateway

Fuente: <https://www.dragino.com>

3.2.3. Fase 3: Integración del proyecto

En este apartado se explicará detalladamente el proceso de instalación y configuración de los diferentes componentes del prototipo, tanto componentes de hardware como de software. Esta fase se divide en 6 partes principales cada una de las cuales describe la instalación o configuración de algún componente individual. Cada componente se configura y evalúa por separado con el fin de validar su correcto funcionamiento. Una vez configurados los componentes, se puede establecer la conexión entre ellos y automatizar el proceso de comunicación. Este proceso no se puede hacer antes de la configuración individual de cada componente ya que en caso de un fallo, sería más difícil para identificar la causa del error.

Los pasos que se van a seguir en esta fase son los siguientes:

1. Configurar los módulos Lora
2. [Configurar el módulo GSM](#)
3. [Conectar el receptor Lora y GSM](#)
4. [Protocolo de comunicación](#)
5. [Configurar la plataforma IOT](#)
6. [Envío de datos a la plataforma IOT](#)

3.2.3.1. Configuración de los módulos Lora

- **Configurar los Módulos Heltec**

Este es el primero y uno de los pasos más importante de la fase de integración del prototipo. A continuación se explicará cómo se puede establecer una conexión entre los nodos terminales y el receptor Lora. Primero de todo, hay que programar cada uno de los módulos lo que requiere la instalación previa de un entorno de desarrollo. El sistema operativo no es de tal importancia, las aplicaciones usadas son compatibles con Windows, Linux o Mac. El desarrollo de este prototipo está basado en Ubuntu, aunque algunas herramientas como el compilador del *Wasp mote* de Libelium requiere un entorno Windows.

Con el fin de programar los módulos Lora se necesita un entorno de desarrollo de Arduino. Este en si no puede reconocer las placas que de desarrollo que se están usando, así que se tienen que instalar algunas librerías adicionales. Se empezará con la configuración del nodo final. El primer paso es abrir el compilador de Arduino y seleccionar la opción de *Preferencias* tal y como se puede ver en la siguiente captura:

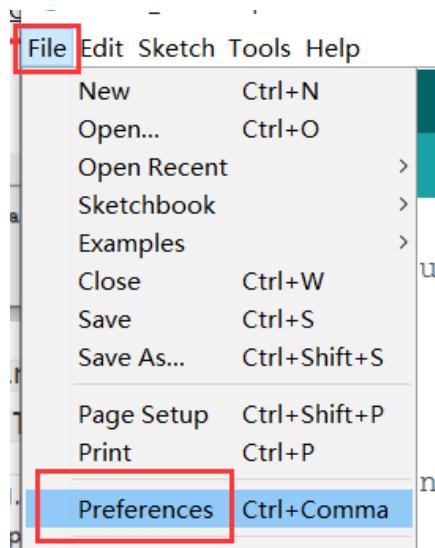


Figura 26: Menú de Arduino IDE

Fuente: Elaboración propia

Después, se tiene que seleccionar la opción *Additional Boards Manager*, tal y como está especificado en la siguiente captura:

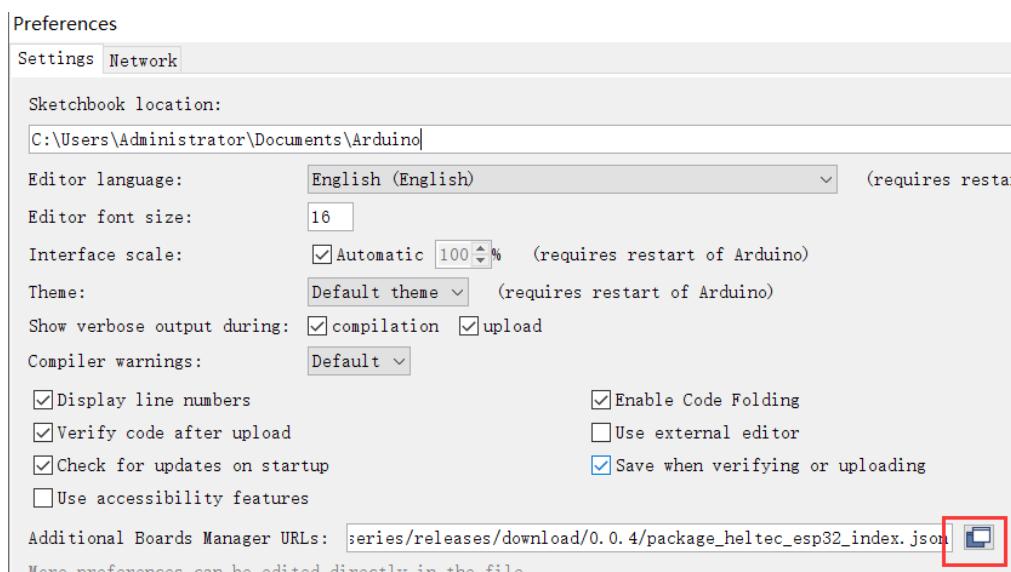


Figura 27: Menú de preferencias Arduino IDE

Fuente: [Heltec Automation](#)

En la ventana que aparece a continuación se tiene que añadir el siguiente link:

- https://resource.heltec.cn/download/package_heltec_esp32_index.json

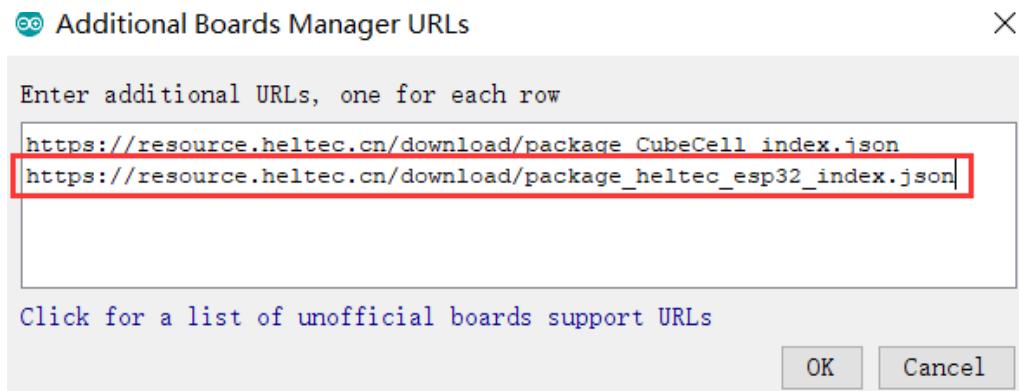


Figura 28: Repositorio de las placas Arduino

Fuente: [Heltec Automation](#)

Con estos pasos, se actualiza el repositorio responsable de gestionar las placas de desarrollo compatibles con Arduino. El siguiente paso es instalar los drivers de la placa Heltec. Para hacer esto se tiene que abrir nuevamente el menú y seleccionar la opción: *Tools → Board → Boards Manager*. En la ventana que aparece a continuación, se tiene que buscar la palabra clave: *Heltec ESP32* e instalar los drivers de esta placa de desarrollo.

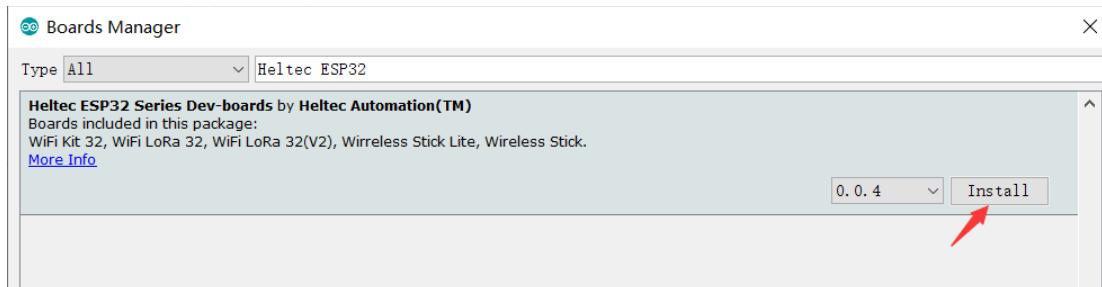


Figura 29: Arduino Board Manager

Fuente: [Heltec Automation](#)

Finalmente, se tiene que instalar la API con las librerías y código de ejemplo para esta placa. La instalación se puede hacer desde el menú: *sketch → include libraries → manage libraries*. En el buscador de la captura que aparece a continuación, se tiene que escribir el nombre de la placa, en este caso: *Heltec*, e instalar la librería.

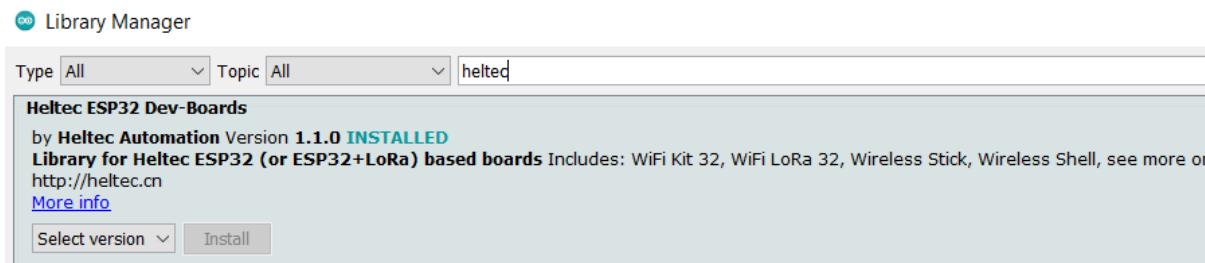


Figura 30: Arduino Library Manager

Fuente: [Heltec Automation](#)

Después de completar las instalación, nuestro entorno ya está preparado para trabajar con las placas de desarrollo Lora de Heltec. Ahora ya se puede proceder a la fase de programación de las placas, pero antes de hacerlo sería necesario indicar al compilador de Arduino qué tipo de placa se va a usar. En este caso, como nodo emisor se está usando el módulo *Lora Wifi 32(V2)*, y como receptor *Heltec Wireless Stick*, aunque el uso de cualquier otra placa es aceptable.

Para terminar con el proceso de configuración, se va a realizar una prueba de comunicación entre ambos módulos, emisor y receptor. Lo que se tiene que hacer es enviar una trama de datos del nodo terminal al gateway siguiendo el formato del protocolo de comunicación que se ha desarrollado. La estructura de este protocolo se comentará con más detalle en el apartado: **3.2.3.4.** El contenido de la trama que se enviará no es de tal importancia ya que el propósito de la prueba es verificar que la instalación y configuración de ambos módulos haya tenido éxito.

En esta primera prueba, se ha usado un código sencillo que simplemente consiste en habilitar cada módulo, aplicar los parámetros básicos de comunicación y enviar un mensaje del emisor al receptor. Los ejemplos del código fuente usados se pueden encontrar en el apartado de *File→ Examples* dentro del IDE de Arduino, o en el sitio web oficial de las placas que se han usado. No se entrará en muchos detalles explicando el código fuente de cada módulo ya que las librerías de los diferentes dispositivos usan una sintaxis y funciones distintas. Sin embargo, hay algunos parámetros comunes que se tienen que tener en cuenta. En el siguiente snippet se puede ver cómo se habilita el módulo Lora, y algunos parámetros básicos que se aplican.

```
//Instalar el servicio Lora
Heltec.begin(false /*DisplayEnable Enable*/,
             true /*Heltec.LoRa Disable*/,
             true /*Serial Enable*/,
             true /*PABOOST Enable*/,
             868 /*BAND*/);
```

//Configurar los parámetros de comunicación

```
LoRa.setSpreadingFactor(7);
LoRa.setSignalBandwidth(125E3);
LoRa.setCodingRate4(5);
LoRa.setPreambleLength(8);
LoRa.setSyncWord(0x12);
LoRa.setTransmissionPower(10);
```

A continuación veremos más detalladamente el propósito de cada uno de estos parámetros[25].

- **Spreading Factor:** Este parámetro se utiliza para determinar la cantidad de señales *chirp* por símbolo. Cuanto más grande sea el coeficiente del SF, más señales chirp se utilizan lo que alargaría el tiempo de modulación o demodulación de la señal. Esto permite que la señal tenga más alcance pero también aumenta el tiempo de envío del mensaje y el consumo de la batería.
- **Bandwidth:** Determina la cantidad de información que podemos enviar utilizando una cierta frecuencia en un intervalo de tiempo. Por ejemplo, un ancho de banda más grande, aumenta el *bitrate* o la velocidad de transmisión de bits por segundo utilizando el mismo SF y Coding Rate. Los microchips de Lora permiten tres tipos de ancho de banda: 125 kHz, 250 kHz, 500 kHz.
- **Coding Rate:** El proceso de modulación también incluye un sistema de corrección de errores en cada transmisión. Cuanto más alto sea el coeficiente, mejor corrección de errores tendremos pero esto ralentizará el tiempo de envío de los paquetes.
- **Sync Word:** Se utiliza tanto en el nodo emisor(*end node*) como en el receptor(*gateway*). La mayoría de chips Lora utilizan 1 byte para representar este parámetro que permite hasta 256 combinaciones. Se utiliza para definir un identificador de la red, de modo que todos los dispositivos con este ID puedan recibir el mensaje. Es como aplicar un filtro que impedirá a otras redes Lora cercanas interceptar nuestros mensajes. No afecta de ninguna manera la comunicación a menos que se utilicen valores distintos en ambos extremos(emisor y receptor), en este caso la comunicación no sería posible. Por defecto las redes Lora privadas utilizan el valor 0x12 y las públicas(LORAWAN) 0X34.
- **Transmission Power:** Este parámetro define la potencia de transmisión de la señal. La mayoría de emisores Lora permiten un rango de transmisión entre 0 dBm y 20 dBm. Es muy importante tener en cuenta que cada país o región aplica diferentes restricciones sobre esta potencia. En Europa el valor máximo permitido para las frecuencias sin licencia(como Lora) es de +14 dBm.

El protocolo Lora puede incluir muchísimos parámetros de comunicación, pero en este proyecto se han usado los más básicos e importantes. En una comunicación a corta distancia(<2 km), no es tan importante que valores se van a poner en cada parámetro, sin embargo, es muy importante que ambos módulos - emisor y receptor, utilicen los mismos valores, de lo contrario la comunicación no tendría éxito. Según la distancia entre el emisor y el receptor y la cantidad de datos que se quiere enviar, habría que aplicar diferentes valores a cada parámetro, sobre todo del *Spreading Factor*, *Bandwidth* y *Coding Rate*. Más adelante se explicará la importancia de estos parámetros y cual es el efecto al cambiar cada uno de ellos.

Una vez configurados y programados los módulos, ya se puede proceder a la prueba de comunicación entre ambos. En la captura siguiente se puede observar la trama de datos que ha recibido el gateway de uno de los nodos finales:

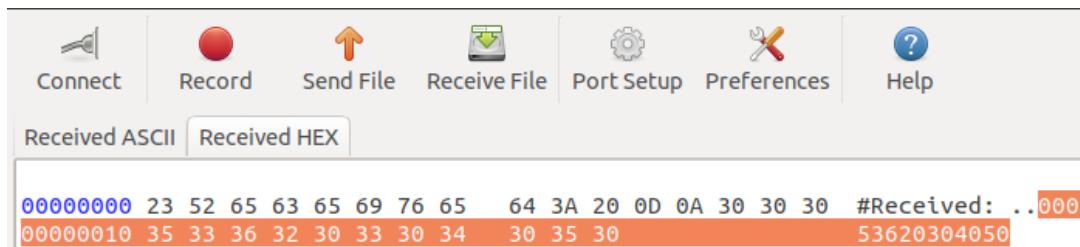


Figura 31: Fotografía del conversor UART - USB
Fuente: Elaboración propia

Se puede observar que el contenido de la trama es: **0, 0, 5, 36, 20, 30, 40, 50**.

Los primeros 2 dígitos representan el identificador y el estado del nodo final. El tercero, representa la longitud del payload que en este caso son 5 dígitos. Finalmente está ubicado el payload o los datos útiles que son los valores que hemos leído de los sensores.

Tras realizar esta prueba, se puede confirmar que el proceso de comunicación entre ambos módulos Lora ha tenido éxito. Ahora se puede proceder al siguiente paso de la implementación - la configuración del módulo GSM.

- **Configurar el módulo Libelium**

A continuación se explicará el proceso de instalación y configuración del módulo *Libelium*. En este caso se tiene que usar el Sistema Operativo Windows para poder hacer el desarrollo. Primero de todo, se tiene que descargar e instalar el IDE de Libelium desde el sitio web oficial[26].

Ahí también se puede encontrar una documentación completa de esta placa, el framework y los drivers para que el ordenador pueda reconocer el dispositivo. Una vez instalado el IDE, se tiene que abrir y configurar el puerto serie, tal y como se muestra en la siguiente imagen. Después de hacer esto, se puede proceder a la programación del módulo.

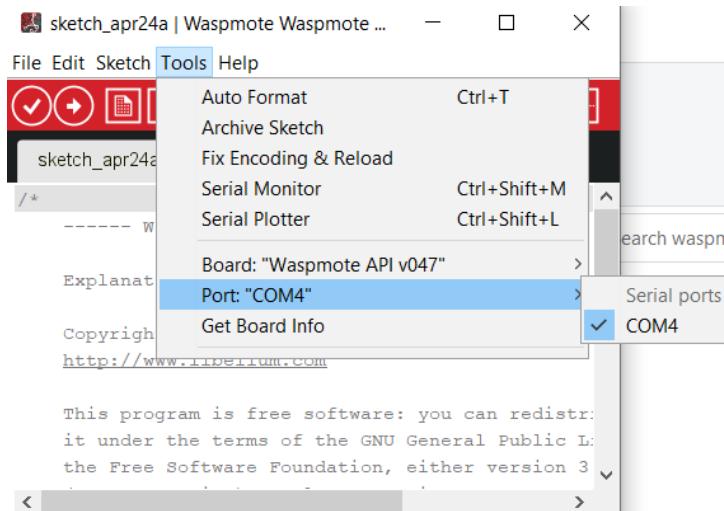


Figura 32: IDE del Libelium. Configuración del puerto serie
Fuente: Elaboración propia

❖ Posibles fallos

Es posible que el ordenador no pueda reconocer el módulo de *Libelium*. Una de las posibles causas es la falta de algún driver, probablemente para el USB. Para hacer esta validación, sería necesario abrir el panel de control y elegir la opción de *Device Manager*. Ahí, se tiene que verificar si aparece el dispositivo: **FT232_UART**. Si este no aparece o aparece con un símbolo de exclamación, significa que el Sistema Operativo no es capaz de reconocer el dispositivo. Por lo tanto, sería necesario instalar el driver: **CDM21228** que está disponible en el link que he dejado arriba.

En la siguiente captura, aparece el dispositivo FT232_UART mencionado arriba. Tal y como se puede ver, el SO no lo reconoce correctamente por lo tanto lo está marcando con un icono de exclamación.

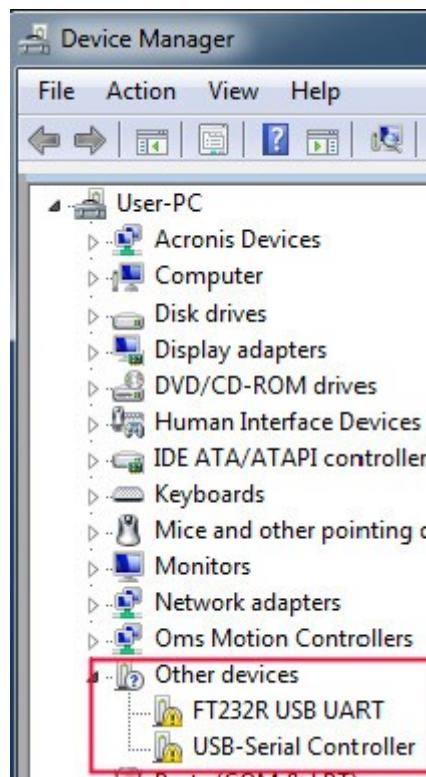


Figura 33: Captura - Device Manager
Fuente: Elaboración propia

3.2.3.2. Configurar el módulo GSM

Una vez configurados los módulos Lora, se tiene que proceder a la configuración del GSM. Como ya se ha mencionado en la Fase2, antes de establecer la conexión entre el receptor Lora y el GSM, se tiene que verificar que el GSM funciona correctamente. En este apartado solamente se explicará el proceso de establecer una conexión por UART con el módulo GSM, y enviar algunos comandos AT de prueba. Todavía no se van a realizar conexiones a ningún servidor, este paso se explicará detalladamente en el apartado: **3.2.3.4.**

Algunos parámetros muy importantes a tener en cuenta durante esta prueba son la fuente de energía, que debe proporcionar entre **3.3V - 5V** y **2A**, y los Baudios o la frecuencia de la UART. Por defecto, la placa de desarrollo del GSM está configurada a: **[115200 bps, 8 bits, no parity, 1 Stop bit, no flow control]**. Si es necesario, algunos parámetros se pueden cambiar mediante los comandos AT, o simplemente configurar el microcontrolador de la placa Lora que opere a la misma frecuencia. Es muy importante que ambos módulos se comuniquen por UART usando la misma frecuencia.

Lo primero que se tiene que hacer en esta prueba es conectar la placa del GSM al puerto USB del ordenador mediante el adaptador serial. En el siguiente esquema se puede observar cómo hacer las conexiones entre ambas placas:



Figura 34: Esquema de la conexión GSM - Ordenador
Fuente: Elaboración propia

En el siguiente paso se tiene que abrir un serial debugger y configurarlo para poder establecer la conexión con la placa del GSM. Si la prueba se realiza en un sistema como Linux, sería necesario asignar permisos de lectura/escritura al puerto USB. Esto se puede conseguir ejecutando el comando **chmod 777 /dev/ttyUSBX** desde el terminal. Cada vez que se conecta un dispositivo al puerto USB, Linux puede asignarle un nombre diferente, por este motivo, la 'X' al final del nombre representa un número entero asignado por el Sistema Operativo. Una vez asignados los permisos, se tiene que abrir el serial debugger que en este caso es el *moserial*. Antes de establecer la comunicación con el GSM, es necesario especificar algunos parámetros de comunicación como el puerto UART, frecuencia(baudios), y otros. En la siguiente captura se puede ver la ventana de configuración del puerto serie(o UART):

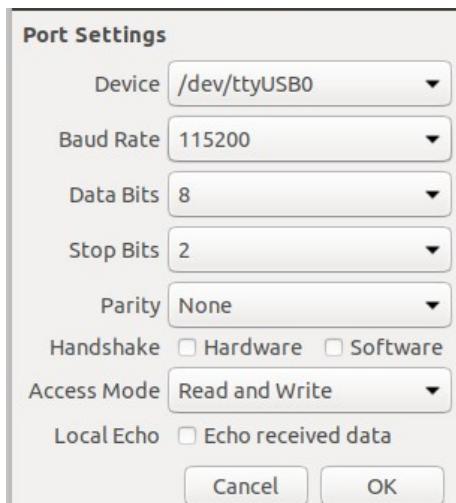


Figura 35: Ventana de configuración de la UART
Fuente: Elaboración propia

Después de realizar estos pasos, se puede establecer la conexión con el módulo GSM. Para saber si está funcionando y configurado correctamente, se tiene que enviar el comando **AT**. Si la configuración es correcta, como respuesta se recibirá la palabra **OK**. Si se recibe algún mensaje de error, ruido o no llega ninguna respuesta, esto significa que o bien el módulo no está configurado con los parámetros adecuados o bien tiene otro fallo. Algo importante a tener en cuenta es que la comunicación por UART requiere que los mensajes enviados terminen con los caracteres <CR><LF>(o \r\n). La mayoría de aplicaciones de comunicación por UART los añaden

automáticamente, pero igualmente se tiene que verificar que los comandos AT terminen con esta secuencia. En caso contrario se pueden experimentar problemas de comunicación.

En la siguiente captura se puede ver el resultado de la prueba de comunicación por UART con el GSM:

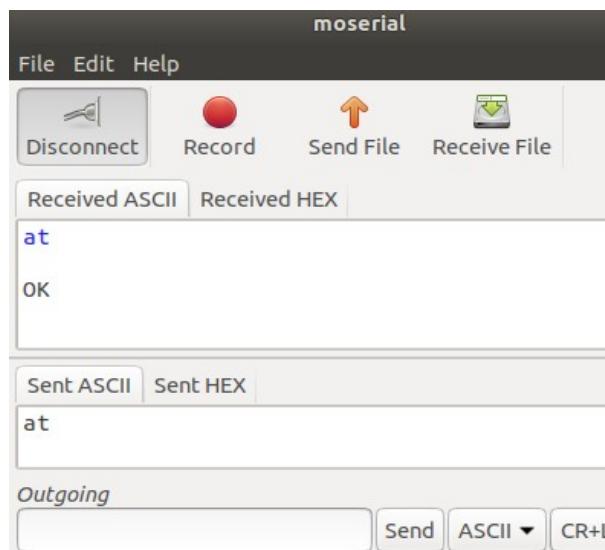


Figura 36: Ventana de comunicación por UART

Fuente: Elaboración propia

● Problemas de comunicación

Si se han seguido los pasos anteriores ya sería posible la comunicación con el módulo GSM. En algunos casos es posible tener problemas de comunicación. Por ejemplo no recibir ninguna respuesta del módulo o recibir ruido. Estos fallos podrían tener varias causas:

- **El GSM no responde:** Si el módulo no responde a los comandos AT, una de las causas podría ser la frecuencia de UART del mismo. Por defecto, este módulo opera a 115200 bps, sin embargo se tiene que verificar que la frecuencia del GSM sea la correcta. Esto se puede hacer ejecutando el comando **AT+IPR?**. Como respuesta, el módulo envía: **+IPR: <rate>**, donde **<rate>** es la frecuencia actual del GSM. Para cambiar los baudios, es necesario ejecutar el comando AT: **AT+IPR=<rate>**.
- **Recibir ruido:** A veces el GSM puede enviar ruido o caracteres ilegibles por la UART. Esto se debe o bien al uso de una frecuencia incorrecta o bien por mala conexión del cableado. También es posible que otro dispositivo conectado al ordenador cause interferencias. Si es el caso, se tienen que desconectar o intentar a utilizar un puerto USB distinto para comunicarnos con el GSM.

3.2.3.3. Conectar los módulos Lora y GSM

Después de configurar los módulos Lora y GSM, se tiene que establecer la conexión entre ambos. La comunicación entre estos módulos se realiza a través del puerto UART. Algo que debe tenerse en cuenta es que el receptor Lora utiliza dos puertos UART. El puerto UART configurado por defecto es el USB. Su principal función es no sólo alimentar la placa sino usarlo para depuración también. Es muy recomendable dejar este puerto libre para programar o depurar la placa y usar otro de los puertos UART disponibles para comunicarnos con el GSM. Si se usa el mismo puerto para varios propósitos, se puede llegar a tener colisiones de datos y no tener una conexión estable. En la siguiente imagen se puede ver como tienen que conectarse los módulos a través del puerto UART:

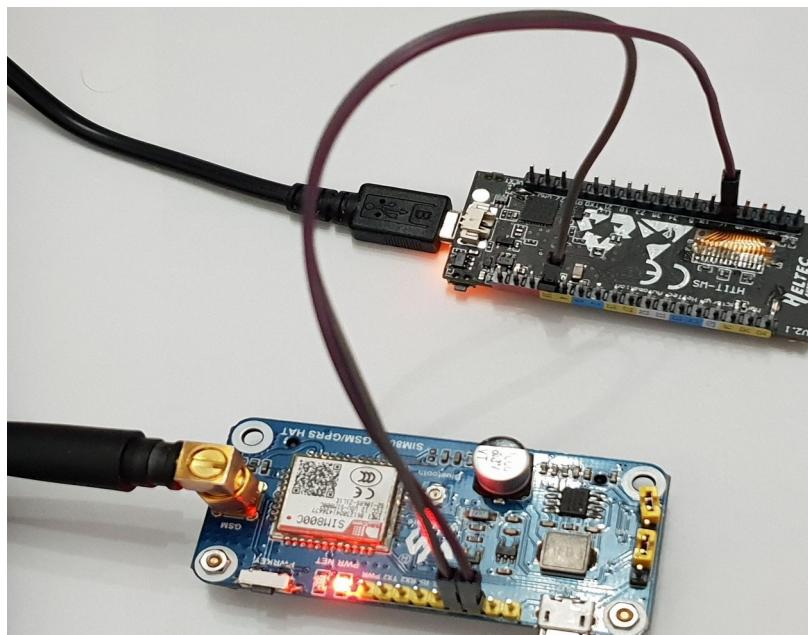


Figura 37: Conexión entre los módulos Lora y GSM
Fuente: Elaboración propia

El módulo de Lora, utiliza los pines **16** y **17** como segundo puerto UART. Para obtener más información se puede consultar el datasheet[27].

Al conectar los pines de la UART de cada módulo(Lora y GSM) estos tienen que cruzarse. Es decir, el pin de lectura o **RX** de un módulo, se conecta con el pin de escritura **TX** del otro, igual que en la figura 30. Otro punto importante que se tiene que tener en cuenta es el tema de la alimentación de cada módulo. Como ya se ha explicado en la Fase 2, cada placa se tiene que alimentar con una fuente de energía separada. En el caso del GSM, lo ideal sería usar una fuente de 5V, 2A. Finalmente se puede programar el módulo Lora para que este pueda recibir los paquetes del emisor(o nodo final) y enviarlos al GSM.

Algo muy importante que se tiene que tener en cuenta es que la comunicación por UART es **asíncrona**, es decir no hay ninguna confirmación si los datos enviados llegan correctamente o no. Por este motivo, hay que implementar algunos métodos de bloqueo o sincronización después de enviar un comando AT al GSM. Otro problema que puede aparecer es que tras ejecutar un comando AT, este puede tardar hasta varios segundos en enviar una respuesta. Así que no sería conveniente enviar todos los comandos uno tras otro sin ninguna pausa ya que el módulo se puede saturar o devolver una respuesta errónea. Lo que se puede hacer en este caso es enviar un comando AT y esperar hasta que el GSM devuelva una respuesta determinada. Según la respuesta que se haya recibido, se procederá a la ejecución del siguiente comando o no.

Al encender la placa Lora, lo primero que hará es configurar el módulo Lora y después el módulo GSM mediante los comandos AT. En la siguiente captura se puede ver un log del proceso de comunicación entre el módulo Lora y GSM:

```
/dev/ttyUSB0

entry 0x400806ac
Serial initial done
LoRa Initial success!
### Initializing GPRS...### Sending command: AT
### Answer: [] ###

### Sending command: AT+SAPBR=3,1, "Contype", "GPRS"
### Answer: [AT+SAPBR=3,1, "Contype", "GPRS"
OK
] ###

### Sending command: AT+SAPBR=3,1, "APN", "ac.vodafone.es"
### Answer: [AT+SAPBR=3,1, "APN", "ac.vodafone.es"
OK
] ###

### Sending command: AT+SAPBR=1,1
### Answer: [AT+SAPBR=1,1
OK
] ###

### Sending command: AT+SAPBR=2,1
### Answer: [AT+SAPBR=2,1
+SAPBR: 1,1,"172.25.20.222"

OK
] ###

### GPRS initialized...
```

Figura 38: Iniciar el servicio GPRS
Fuente: Elaboración propia

Como se puede ver, el servicio GPRS se ha iniciado con éxito. Ahora el módulo GSM ya se puede conectar a Internet. Durante el proceso de inicialización del GPRS se utilizan tres comandos principales: para iniciar el servicio GPRS, para conectar el módulo a un *Acces Point* y solicitar una IP. El último comando: *AT+SAPBR=2,1*, se utiliza para consultar la IP que ha asignado la operadora móvil al módulo GSM. Este comando no es obligatorio pero es recomendable durante el proceso de depuración. De esta manera se puede verificar que el GSM está conectado a un Punto de Acceso y el servicio del GPRS funciona correctamente. En el próximo apartado: *Protocolo de comunicación* se entrará en más detalle sobre cada uno de los comandos AT que se han utilizado.

En este punto, los módulos Lora y GSM ya están configurados y listos para operar. Para que el GSM pueda enviar datos a la plataforma IOT, el receptor Lora tiene que obtener una trama de datos de alguno de los nodos finales. Cuando esto ocurra, se activará el protocolo de envío de datos para transmitirlos a la plataforma IOT. Este proceso está explicado con más detalle en el siguiente apartado.

3.2.3.4. Protocolo de comunicación

En este apartado se explicará detalladamente la estructura del protocolo y la lógica de comunicación entre los nodos finales y el gateway. Para este prototipo se ha desarrollado un protocolo de comunicación específico utilizado por los diferentes módulos Lora.

El protocolo consiste en los siguientes pasos:

- 1. Lectura de los sensores:** Los nodos finales leen los sensores periódicamente. Según el tipo de sensor, la lectura puede ser más o menos frecuente.
- 2. Encapsular los datos:** Los datos que hemos leído, se encapsulan en una estructura, se añaden la longitud del payload y ID del nodo. Ver tablas 2 y 3.
- 3. Enviar la trama al Gateway:** La estructura de datos generada en el paso anterior se envía al Gateway usando el módulo LORA.
- 4. Añadir cabecera:** El Gateway recibe la trama y le añade su ID. Además, la trama se encapsula en formato JSON. No se hace ninguna validación de datos todavía. Finalmente el Gateway envía la trama al GSM a través del puerto UART.
- 5. Enviar la trama por HTTP al servidor IOT:** El GSM envía la estructura que recibe del módulo Lora a la plataforma IOT mediante una petición HTTP.
- 6. Parseo de los datos:** La plataforma IOT recibe los datos, los parsea y los valida

Para entender mejor el proceso de lectura de los sensores y envío de datos, se puede observar el diagrama de flujo siguiente:

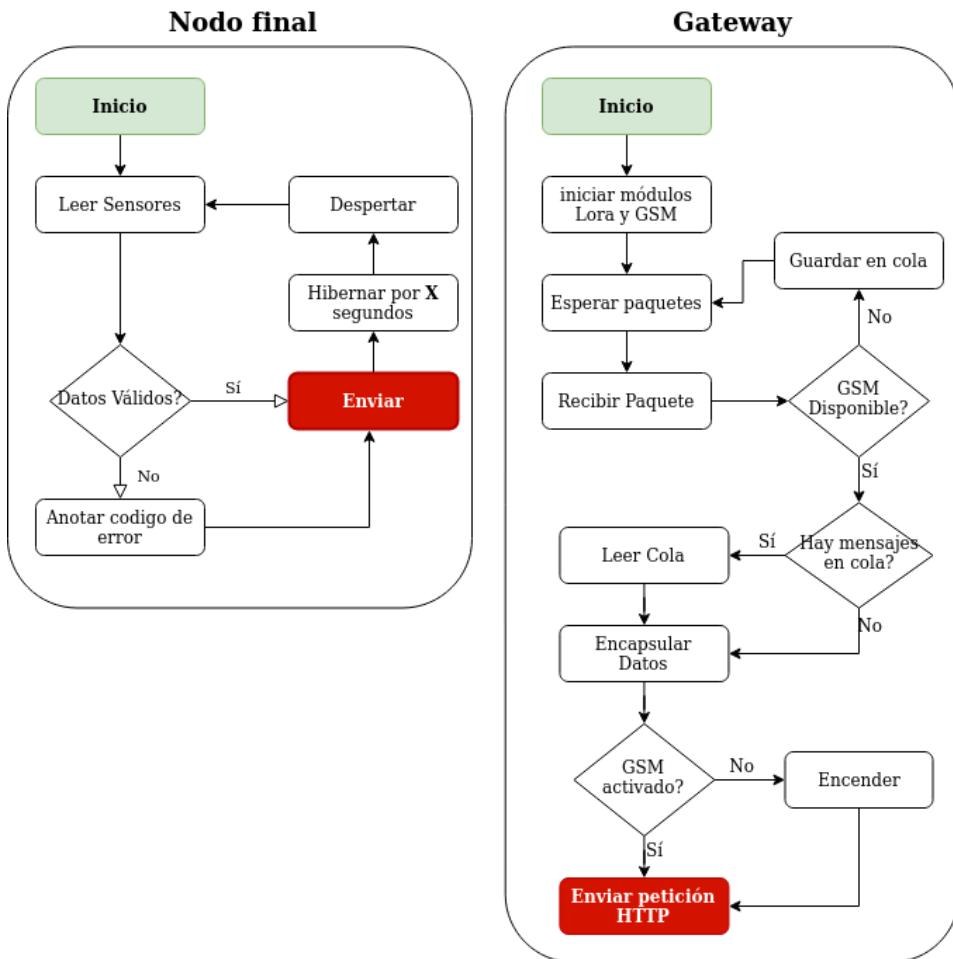


Figura 39: Diagrama de flujo de un nodo final y el Gateway
Fuente: Elaboración propia

La estructura de este diagrama es relativamente sencilla y solamente se han destacado los procesos más relevantes. El proceso **Enviar** del nodo final consiste en encapsular los datos según las reglas del protocolo de comunicación y enviarlos mediante el transmisor Lora. El proceso de **Encapsular Datos** del Gateway consiste en añadir el *ID* y estado del mismo y encapsular los datos en formato JSON.

Para encapsular la información, el protocolo utiliza la estructura de datos descrita en las tablas 2 y 3. Dicha estructura consiste en un array de bytes, es decir el valor de cada posición puede tener hasta 256 combinaciones.

- **Estructura de datos del protocolo**

0	1	2	3	4	5	6 - N
GAT_ID	G_STAT	DEV_ID	DEV_STAT	GR_ID	LEN	PAYLOAD

Tabla 3: Estructura de datos del protocolo de comunicación
Fuente: Elaboración propia

- **Tabla descriptiva de la estructura de datos**

Campo	Tipo de Datos	Descripción	Tamaño bytes
GAT_ID	byte	Id del gateway	1
G_STAT	byte	Estado del gateway	1
DEV_ID	byte	Id del nodo final	1
GR_ID	byte	Grupo del nodo	1
DEV_STAT	byte	El estado del nodo	1
LEN	byte	Longitud del payload	1
PAYLOAD	byte	Datos útiles	1 - N

Tabla 4: Tabla descriptiva de la estructura de datos
Fuente: Elaboración propia

Tal y como podemos observar, cada componente de la estructura ocupa 1 byte. Sin embargo, nos puede surgir la necesidad de almacenar valores superiores a 255. En este caso, deberíamos usar 2 bytes o más. Por ejemplo, haciendo lecturas de un sensor de presión atmosférica, este nos puede devolver valores superiores a 1000. En este caso, el valor se tiene que dividir entre 2 bytes con el fin de poder guardarlo en la estructura. Para poder hacer esto, debemos aplicar un desplazamiento de bits[28]. Suponiendo que el número que queremos guardar es 1000, necesitaremos 2 bytes para representarlo. Por lo tanto, el desplazamiento se haría de la siguiente manera:

```
payload[0] = pressure >> 8;
payload[1] = (pressure & 0xFF);
```

Después de realizar esta operación, el contenido del payload sería el siguiente:

```
payload[0] = 0x03;
payload[1] = 0xE8;
```

Como podemos ver, la variable anterior quedará guardada en el buffer del payload así que podemos proceder al envío de la trama.

Es muy importante tener en cuenta que una vez la plataforma IOT reciba la trama de datos, debemos juntar aquellos valores que están divididos entre varios bytes, tal y como hemos visto en el ejemplo anterior. Para conseguir esto, es necesario hacer la siguiente operación:

```
int pressure = (payload[0] << 8) | payload[1];
```

Lo que hace este código es aplicar un desplazamiento(offset) de 8 bits al primer byte del payload y sumar el valor de la siguiente posición. De esta manera se reconstruye el valor original de la lectura y a partir de ahí podemos proceder a su validación.

Las tramas enviadas por los nodos finales se validan por la plataforma IOT y no por el gateway. Este no hace ninguna validación o comprobación de los datos entrantes, tampoco hay un control sobre los dispositivos que se conectan o desconectan del mismo. Esto permite conectar o eliminar nodos finales de una red o enviar una trama de la longitud y contenido que deseamos sin tener que configurar o reprogramar el gateway. De esta manera se puede tener más flexibilidad y escalabilidad a la hora de modificar o extender una red de dispositivos.

En la imagen que aparece a continuación, podemos observar con más detalle la estructura de datos vista en la *Tabla 3*.

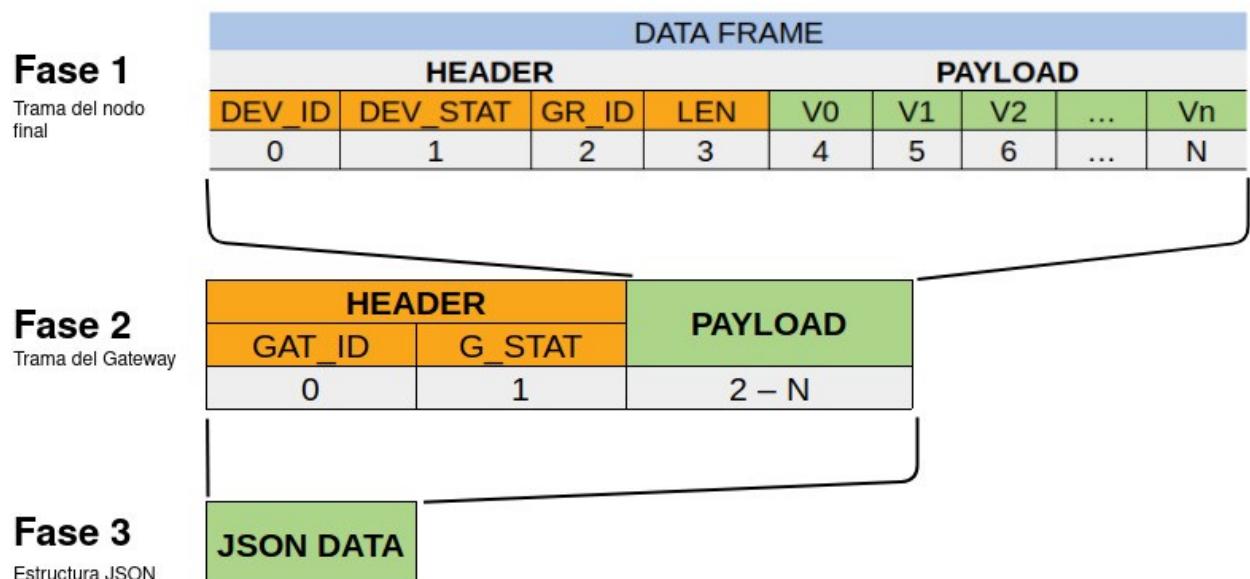


Figura 40: Estructura de la trama de datos
Fuente: Elaboración propia

Como podemos observar en este diagrama, el proceso de encapsulación de los datos consiste en tres fases principales. La primera se realiza por el nodo final, ahí es donde se agregan el identificador del sensor, su estado y los datos útiles(*payload*). A continuación, la trama se envía al gateway. Una vez este reciba los datos, se procede a la Fase 2, donde a la trama se añade el *Id* y estado del gateway. Como podemos ver, estos dos valores se convierten en la cabecera de la nueva trama y los datos que hemos recibido del nodo final, almacenan como *payload*. Finalmente, la trama se encapsula en formato JSON. Una vez hecho esto, el paquete de datos ya está preparado para ser enviado a la plataforma IOT.

El contenido típico de la estructura JSON puede tener el siguiente aspecto:

{“payload”:[0,0,0,0,0,2,10,20]}. Según el protocolo, esta trama fue enviada por el nodo final con *ID:0*, desde el gateway con *ID:0* y tiene un payload de una longitud de 2 bytes: 10 y 20 respectivamente.

Antes de enviar el mensaje al servidor, es importante verificar que este acepte peticiones por HTTP, HTTPS o ambas. La plataforma IOT que se está usando, acepta peticiones mediante ambos protocolos. Sin embargo algunos módulos GSM o bien no soportan este protocolo o bien utilizan una versión obsoleta. Debido a esto, han surgido problemas con el envío de datos por HTTPS, ya que el módulo de *simcom* soporta este protocolo pero con una versión de TLS obsoleta, es decir la 1.0 [29].

En este caso, la plataforma IOT rechazaba las peticiones enviadas por el módulo con un código de error: **606**. Según la documentación del módulo, este código representa un error del SSL, probablemente una versión antigua. Antes de enviar la petición, hay que verificar qué protocolos soporta el módulo GSM. Si se desea usar el HTTPS, es necesario que este tenga una versión de TLS 1.1 o 1.2. Las versiones más antiguas podrían ser incompatibles con los servidores modernos y por lo tanto las peticiones pueden ser rechazadas. Una de las posibles soluciones es actualizar el firmware del GSM lo que implicaría algunas complicaciones. Otra posible solución es descartar el uso de HTTPS y hacer las peticiones por HTTP. Finalmente se ha decidido usar el protocolo HTTP ya que se adapta mejor a este prototipo. Como ya se sabe, una de las ventajas del protocolo HTTPS es la conexión segura. Si el uso de una comunicación de este tipo no es imprescindible, se puede aplicar el protocolo HTTP sin ningún problema.

Los identificadores de cada nodo o gateway se asignan manualmente a la hora de grabar el firmware en un dispositivo. Como se puede ver en la tabla de arriba, cada *Id* es de tipo byte lo que permite tener hasta 256 combinaciones. No hay ninguna restricción a la hora de asignar los identificadores, mientras cada dispositivo del mismo gateway tenga un *Id* diferente. Cada identificador se interpreta por la plataforma IOT lo que permite hacer una mejor gestión de los datos entrantes como almacenarlos o visualizarlos mediante diferentes gráficos. Este proceso se explicará con más detalle en el apartado: *Configurar la plataforma IOT*.

Los campos de estado(*N_STAT* y *G_STAT*) de cada nodo sirven para notificar si algún dispositivo tiene un fallo como por ejemplo: fallo de algún sensor, baja batería, fallo en el circuito y muchos más. El campo de *Payload* representa los datos que se han leído de los sensores. Este puede tener una longitud distinta según la cantidad de sensores que se han conectado a un nodo. El contenido de la trama y el tipo de sensores que va a leer un nodo se definen al grabar el firmware en el nodo final. Los nodos no pueden reconocer automáticamente los sensores ni decidir qué tamaño de payload deben usar. En este caso es necesaria la intervención humana. Además por cada nodo es necesario indicar qué tipo o qué grupo de sensores se han aplicado, es decir asignar valor al parámetro **GR_ID**. Esta asignación se hace manualmente al programar el módulo. Cada grupo tiene un identificador propio, tal y como está especificado en la *Figura 28*. Por ejemplo, si un nodo está equipado con sensores de temperatura y humedad, se puede decir que son del grupo 1. Si al nodo final están conectados sensores de presión atmosférica y CO₂ por ejemplo, al campo de **GR_ID** se le asignaría el valor 2 por ejemplo. No hay ningún valor predeterminado por el momento, así que se puede asignar cualquier identificador de grupo. El usuario tiene la libertad de combinar los sensores y poner los IDs que desea, no hay ninguna restricción o regla. El propósito de este identificador es poder distinguir desde la plataforma los diferentes módulos y saber qué sensores lleva cada uno de ellos.

3.2.3.5. Configurar la plataforma IOT

En este apartado se hará la configuración de la plataforma IOT que se utiliza para almacenar los datos leídos por los sensores. Tal y como se ha mencionado anteriormente, se usará la plataforma: *demo.thingsboard.io*. Este servidor permite crear una cuenta gratuita aunque limitada en algunos servicios. Sin embargo, el plan *demo* cumple con todos los requisitos de este proyecto. Una de las características más importantes de esta plataforma es que permite comunicarnos con la misma mediante una API, lo que hace posible el envío de datos desde el GSM. Otra propiedad muy importante es la posibilidad de gestionar los mensajes/peticiones de entrada mediante un script hecho por el usuario. Esto permite una mejor gestión de los mensajes y datos y la posibilidad de ejecutar algunas rutinas en determinadas situaciones. Por ejemplo, si el valor que se recibe de un sensor está fuera de unos límites determinados, se puede enviar una notificación al usuario mediante email, SMS, Whatsapp u otros. Además, esta plataforma permite visualizar en tiempo real las últimas lecturas de los datos recibidos, ver el historial de datos de los sensores mediante diferentes gráficos y muchos más.

Para poder configurar la plataforma, lo primero que se tiene que hacer es registrarse si no se ha hecho todavía. El proceso de registro es totalmente gratuito. Una vez hecho esto es necesario entrar en la cuenta y desde el menú principal seleccionar la opción de *Rule Chain*. Esta permitirá gestionar los mensajes que recibe la plataforma de los

clientes/Gateways a través de la API. En la siguiente captura se puede ver el menú de la plataforma:

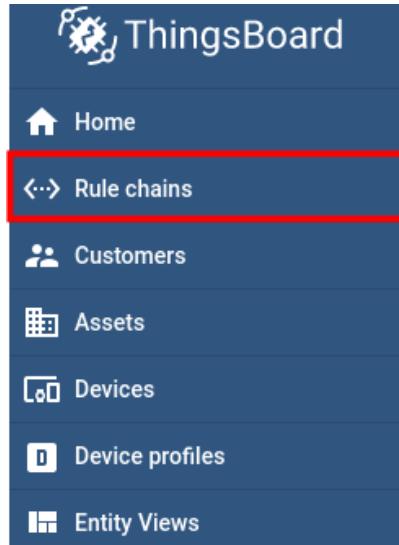


Figura 41: Menú principal de la Plataforma IOT
Fuente: Elaboración propia

Por defecto la plataforma ya tiene definidas unas reglas, pero estas no son suficientes así que es necesario añadir algunas adicionales. En la siguiente captura se puede ver la forma final del Rule Chain:

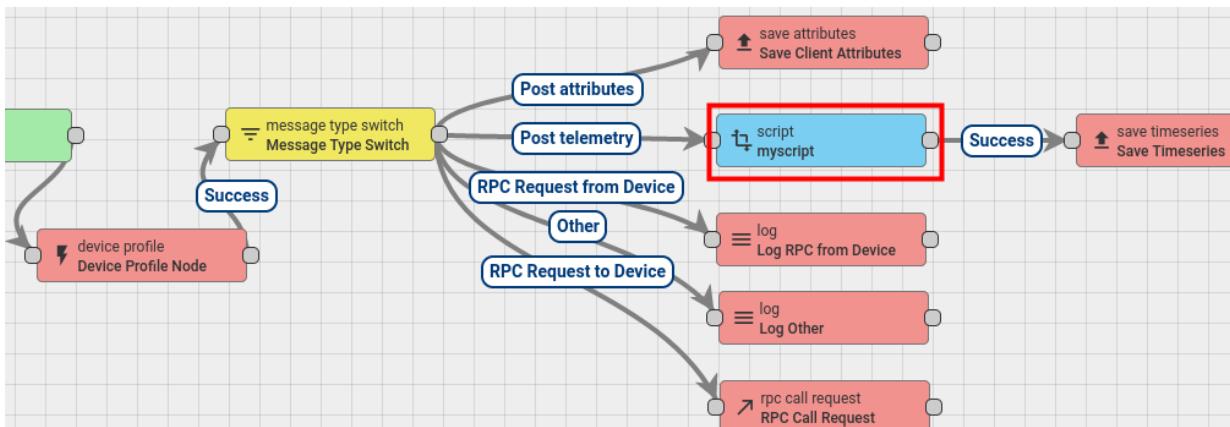


Figura 42: Rule Chain de la Plataforma IOT
Fuente: Elaboración propia

El elemento *script* permite gestionar las peticiones mediante un script hecho por el usuario. En este apartado se realiza una gestión y validación de los datos enviados por los Gateways. A continuación se explicará el proceso de parseo de los datos que recibe la plataforma.

- **Parseo de datos**

Al recibir un mensaje, el script procede al parseo de los datos siguiendo las reglas del protocolo de comunicación. Primero de todo se extraen los IDs del gateway y los nodos finales para determinar el origen del paquete. Después se revisan los códigos de estado de cada dispositivo y si se detecta algún fallo, se enviará una notificación al usuario por SMS, email, telegram u otro medio. Finalmente se tiene que verificar el grupo o tipo del dispositivo para poder interpretar los datos del payload correctamente. Tal y como se ha visto en la *tabla 3*, los diferentes dispositivos se agrupan según el tipo de sensores que tienen instalados. Por lo tanto, por cada grupo de dispositivos, se tiene que hacer un parseo diferente, ya que el contenido del payload tendrá un formato distinto. Además, la plataforma valida los datos del payload para determinar si ha habido algún fallo o anomalía en alguno de los módulos. Por ejemplo, si un nodo envía lecturas sobre la temperatura medioambiental, se pueden definir unos rangos aceptables, por ejemplo entre: -10°C y + 40°C. Si se recibe una lectura fuera de este rango, se puede interpretar como un valor crítico por lo que la plataforma debería enviar una notificación al usuario.

En la siguiente figura, podemos ver parte del código que se encarga del proceso de parseo de la trama recibida por la plataforma IOT.

```

1  parseDataFrame(msg);
2
3
4  function parseDataFrame(msg){
5
6      var GROUP_ID = msg.payload[4];
7
8      switch(GROUP_ID){
9          case 0:
10             deviceType_0(msg);
11             break;
12
13         case 1:
14             deviceType_1(msg);
15             break;
16
17         default:
18             badRequest(msg);
19     }
20 }
```

Figura 43: Snippet de parseo del data frame
Fuente: Elaboración propia

Tal y como se puede observar, lo primero que hace el script es extraer el *GROUP_ID* del mensaje con el fin de determinar el tipo de validación que se tiene que aplicar. A continuación se ejecuta la función correspondiente que se dedica a extraer los datos del payload y aplicar dicha validación. En la siguiente figura se puede apreciar un código simplificado que se encarga de hacer la validación de los datos:

```

2 function deviceType_0(msg){
3
4     var DEVICE_ID = msg.payload[2];
5
6     switch(DEVICE_ID){
7         case 0:
8             msg.temp0 = msg.payload[6];
9             break;
10
11        case 1:
12            msg.temp1 = msg.payload[6];
13            break;
14    }
15 }
```

Figura 44: Snippet de validación del data frame
Fuente: Elaboración propia

La variable *msg* contiene el payload que recibe la plataforma que en este caso es el mensaje JSON mencionado en el apartado del protocolo(3.2.3.4). Al declarar las variables: *msg.temp0* y *msg.temp1*, las hacemos accesibles en toda la plataforma. Es decir, podemos acceder a estas variables desde otros apartados de la plataforma. Por ejemplo, si se quieren mostrar por pantalla los valores de las variables, se tienen que usar unas herramientas llamadas *widgets* que podrán acceder a dichas variables sólo si están declaradas correctamente. A continuación veremos cómo podemos crear estos widgets y cómo vincularlos a las variables para que estos muestren sus valores correspondientes.

El siguiente paso requiere crear un dispositivo virtual en la plataforma. Para hacer esto, es necesario ir a la opción *Device* y añadir un dispositivo virtual. Después de guardarlo, es muy importante abrir el menú de configuración de este y copiar el *Access Token*. Este token se usará más tarde para enviar mensajes a la API de la plataforma.

En la siguiente captura se puede apreciar el menú del dispositivo virtual que se acaba de crear:

Device details

Details Attributes Latest telemetry Alarms Events Relations

Make device public **Assign to customer** **Manage credentials** **Delete device**

Copy device Id **Copy access token**

Name
HELTEC - ESP32

Device profile
default

Label

Is gateway

Figura 45: Ventana de configuración de los dispositivos virtuales
Fuente: Elaboración propia

Finalmente se tiene que crear un nuevo *Dashboard* y añadirle algunos widgets para poder visualizar los datos entrantes. Al seleccionar la opción para crear un nuevo widget, aparecerá la ventana siguiente:

Add alias

Alias name *

alias

Resolve as multiple entities

Filter type *

Single entity

Type *

Device *

Device

HELTEC - ESP32

Cancel Add

Figura 46: Ventana para añadir un alias
Fuente: Elaboración propia

Esta ventana permite crear un *alias* que hará de vínculo entre el dispositivo virtual y el widget. El nombre del alias no es importante en este caso, lo que sí importa es el tipo del alias - *Single Entity* y el nombre del dispositivo que en este caso sería el que acabamos de crear.

Después de crear el alias, se añadirá la variable definida en el apartado de Rule Chain. A continuación se puede ver la ventana de configuración del widget:

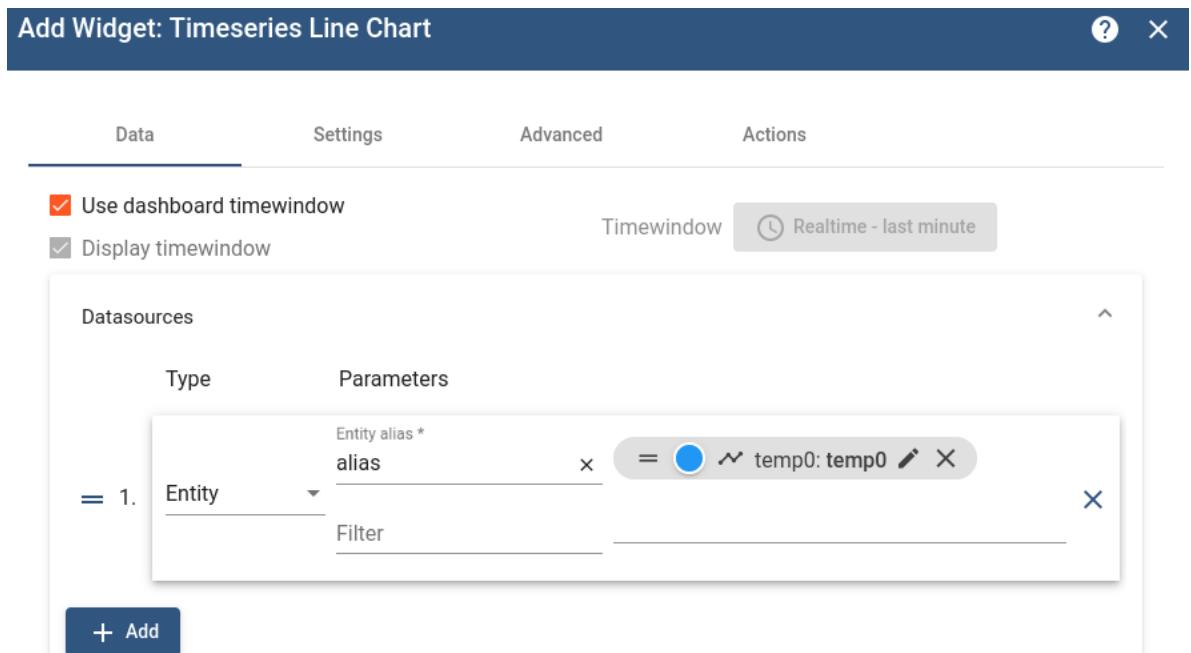


Figura 47 Ventana para añadir un widget
Fuente: Elaboración propia

Con este paso, la plataforma IOT ya está configurada. Para verificar que la configuración que se acaba de hacer es correcta, es necesario enviarle algunos mensajes. Esto se puede hacer desde un cliente REST o desde el sistema Linux enviar una petición POST utilizando la consola mediante el comando *curl*. El siguiente comando envía un array de números a la plataforma:

```
curl -v -X POST -d '{"payload": [0,0,0,0,0,1,10]}'
http://demo.thingsboard.io/api/v1/T4heef9dzbcuejT5qegC/telemetry --header
"Content-Type:application/json"
```

Si la conexión ha sido exitosa, la plataforma recibirá el mensaje y los resultados se podrán visualizar en el dashboard. En este caso, el número **10** del payload simula la temperatura. En la captura de abajo, aparece el widget que muestra el mismo valor que se ha enviado.



Figura 48: Widget analógico de la temperatura
Fuente: Elaboración propia

3.2.3.6. Envío de datos a la plataforma IOT

En este último apartado de la fase de integración se explicará detalladamente la comunicación entre el módulo GSM y la plataforma IOT.

Tal y como se ha explicado anteriormente, las tramas de datos se envían mediante el protocolo HTTP ejecutado por los comandos AT. El motivo de utilizar estas instrucciones es que nos permiten comunicarnos con el módulo GSM a un bajo nivel y tener mejor control sobre el dispositivo. En la actualidad existen librerías para el entorno de Arduino que facilitan la comunicación con los módulos GSM. El problema es que cada librería está adaptada a un modelo muy concreto de dispositivos GSM y cumple con unas ciertas funcionalidades. Si decidimos cambiar de modelo de GSM, probablemente sería necesario cambiar o adaptar la librería también.

Si utilizamos los comandos AT para comunicarnos con el GSM, al cambiar de módulo, es posible que tengamos que modificar algunos de los comandos utilizados. El motivo es que los diferentes fabricantes de módulos GSM pueden utilizar una sintaxis diferente. Por lo que la secuencia de comandos AT usada en un modelo de GSM, puede que no funcione en otro.

A continuación entraremos en más detalles sobre cada uno de estos comandos. Lo primero que se tiene que hacer es iniciar el servicio GPRS mediante los comandos AT que aparecen en la siguiente tabla.

- **Configurar el módulo GPRS**

Comando	Descripción	Respuesta
AT+SAPBR=3,1,"Contype","GPRS"	Seleccionar perfil - GPRS	OK
AT+SAPBR=3,1,"APN", "ac.vodafone.es"	Definir los datos del APN	OK
AT+SAPBR=1,1	Activar el contexto GPRS	OK
AT+SAPBR=2,1	Consultar el estado del GPRS	Devuelve la IP asignada

Tabla 5: Tabla con los comandos AT que habilitan el GPRS

Fuente: Elaboración propia

Una vez ejecutados los comandos, el servicio GPRS ya está iniciado y el módulo GSM está conectado al Access Point(**AP**) lo que nos permite acceder al Internet. En este caso, el nombre del AP es **ac.vodafone.es**. El usuario y la contraseña no son necesarios aunque algunos operadores sí requieren especificar las credenciales. Cabe destacar que el proceso de habilitar el GPRS se ejecuta solo una vez mientras el módulo GSM esté activo. Si se intenta instalar por segunda vez, al ejecutar el comando **AT+SAPBR=1,1** el GSM devuelve un error. No se trata de un error crítico, así que sería posible seguir ejecutando otros comandos o incluso hacer una conexión a Internet. Sin embargo, es recomendable evitar la ejecución múltiple de este comando para evitar comportamientos no deseados por parte del GSM.

Después de configurar el servicio GPRS correctamente, se puede proceder al envío de peticiones por HTTP. Por cada mensaje que se desea enviar al servidor, es necesario ejecutar todos los comandos que aparecen en la siguiente tabla. Si algún comando devuelve un mensaje de error, se tendrá que abortar el proceso de envío y empezar de nuevo.

- Enviar petición por HTTP

Comando	Descripción	Respuesta esperada
AT+HTTPINIT	Iniciar la sesión HTTP	OK
AT+HTTPPARA="CID",1	Asignar parámetros a la sesión HTTP	OK
AT+HTTPPARA="URL", "http://demo.thingsboard.io/api/v1/ API-KEI/telemetry"	Asignar la URL del servidor	OK
AT+HTTPPARA="CONTENT", "application/json"	Tipo del payload: JSON	OK
AT+HTTPDATA=29,5000	Parámetros del <i>payload</i> : 100 bytes de longitud, 10s de timeout	OK
{"payload": [10,0,0,0,0,1,20]}	El contenido del mensaje que queremos enviar	DOWNLOAD
AT+HTTPACTION=1	Tipo de la petición: POST	200
AT+HTTPTERM	Cerrar la sesión HTTP	OK

Tabla 6: Tabla con los comandos AT para enviar peticiones HTTP
Fuente: Elaboración propia

Uno de los comandos más importantes que aparecen en la tabla 6, es **AT+HTTPACTION=1**. Tras ejecutarlo, el GSM envía la respuesta **OK**, sin embargo esta no es la respuesta definitiva. Esto solo afirma que el comando fue ejecutado y la petición HTTP se ha enviado con éxito. Sin embargo, la respuesta del servidor puede tardar varios segundos dependiendo de varios factores como la calidad de conexión, disponibilidad del servidor, etc. Mientras tanto es importante quedar a la espera y no ejecutar ningún otro comando. Una vez se reciba la respuesta del servidor, es imprescindible cerrar la sesión HTTP independientemente del código que nos devuelva. Es muy importante hacer esto antes de enviar un nuevo mensaje. De lo contrario, el módulo GSM puede quedar bloqueado y no sería capaz de enviar nuevas peticiones.

Si el servidor envía como respuesta el código 200, significa que el envío tuvo éxito. Si se recibe otro código, sería necesario consultar el manual del protocolo HTTP o del módulo GSM. A pesar de que los comandos AT sean un estándar y los fabricantes intentan seguir unas reglas comunes, cada dispositivo es diferente y la sintaxis de sus comandos AT puede variar.

- **Problemas con las peticiones HTTP**

Uno de los procesos más sensibles es el envío de la petición HTTP al servidor. El éxito de esta operación depende de varios factores y un mínimo fallo puede cancelar el envío de los paquetes. Uno de los errores más comunes que puede devolver el módulo GSM es: **+HTTPACTION: 0,601,0**. Este código significa que el envío de la trama no se ha podido realizar. El error viene por parte del cliente, o el GSM en este caso. Este error puede tener varias causas:

- Expiración del contrato de la tarjeta SIM.
- Fallo de conexión a la Red telefónica
- Falta de crédito o de datos móviles

Para averiguar la causa exacta más fácilmente se puede insertar la tarjeta SIM en otro dispositivo móvil. De esta manera se puede verificar si el problema viene de la tarjeta SIM o del módulo GSM. Es recomendable realizar varias pruebas como conectarse a una red móvil, hacer una llamada, enviar un SMS o hacer una conexión a Internet.

En la siguiente captura se puede observar los resultados de uno de los intentos fallidos que han surgido al intentar enviar un paquete a la plataforma por GPRS. El último mensaje representa la respuesta por el GSM como resultado de la ejecución. El código 601 afirma un fallo de conexión.

```

AT+HTTPPARA="CID",1
OK
AT+HTTPPARA="URL","http://demo.thingsboard.telemetry"
OK
AT+HTTPPARA="CONTENT","application/json"
OK
AT+HTTPDATA=29,5000
DOWNLOAD
OK
AT+HTTPACTION=1
OK
+SAPBR 1: DEACT
+HTTPACTION: 1,601,0

```

Figura 49: Envío fallido de una petición HTTP desde el GSM.
Fuente: Elaboración propia

3.2.4. Fase 4: Fase de pruebas y validación

En esta última fase de integración del proyecto se van a realizar diferentes pruebas de rendimiento y validación del prototipo. Esta fase se puede subdividir en tres partes principales:

1. Prueba de consumo de energía
2. Prueba de eficiencia de los módulos Lora
3. Envío de datos a la plataforma IOT

3.2.4.1. Consumo de energía

- Nodo final

El propósito de esta primera prueba, es medir el consumo de uno de los nodos finales utilizando el multímetro mencionado en el apartado: 3.2.2.2. En esta prueba se van a combinar los parámetros de comunicación con el fin de conseguir el mayor consumo posible de energía, es decir crear las peores condiciones para la batería y el emisor y observar los resultados. El protocolo Lora permite el uso de 6 valores distintos para el Spreading Factor(SF), [SF7 - SF12]. Cuanto más grande sea el coeficiente, más lejos puede llegar la señal por lo que se necesitaría más energía. Por lo tanto, la batería se descarga en menos tiempo.

En esta prueba, se aplican los siguientes parámetros: [*Spreading Factor = 12, Band Width = 125 dBm, Coding Rate = 4/5, TX POWER = 14 dBm*]. Además, se realiza un envío de paquetes de 20 bytes cada segundo. Otro parámetro interesante que afecta al consumo de batería es la potencia de transmisión(*TX POWER*). En este caso se han asignado +14 dBm que es el máximo valor permitido en la mayoría de países de Europa. Estas condiciones causarían un consumo de energía bastante elevado. De esta manera se pueden comprender mejor los posibles efectos que pueden tener los diferentes parámetros de configuración de los módulos Lora.

En la siguiente captura podemos ver cómo están conectados el emisor Lora y el multímetro con el powerbank:



Figura 50: Medir el consumo del nodo LORA Cube Cell AB-01
Fuente: Elaboración propia

La placa del emisor Lora está conectada a un powerbank de 5V DC, con una capacidad de 1200 mAh . Esta placa consume aproximadamente 20 mA en un estado de reposo, es decir cuando el módulo no envía paquetes ni hace ningún otro cálculo adicional. Este estado no debe confundirse con la hibernación o *deepsleep* que es totalmente diferente.

Durante el envío de paquetes, el módulo puede alcanzar un consumo bastante elevado, hasta 150mA . La duración de esta prueba ha sido de aproximadamente 8 minutos por lo que fácilmente se puede calcular el valor medio del consumo que en este caso sería de 103.63 mA . Con este valor se puede calcular el tiempo de descarga de la batería según la fórmula: $t = C/I$ [30]. Donde **C** es la capacidad de la batería en $\text{Ah}(\text{Amperios Hora})$ e **I** es la corriente de descarga(en *Amperios*) o en este caso el consumo del módulo. Por lo tanto, el tiempo de descarga es: $t=1200/103.63$, **t=11.57h**. También cabe destacar que durante esta prueba, el emisor no entra en modo de ahorro de energía(*Deepsleep*) que es otra de las causas del consumo elevado. Existen otros factores que afectan la descarga como la temperatura, pero no se tendrán en cuenta en estas pruebas.

En el gráfico siguiente se puede apreciar el consumo en *Amperios* que ha tenido el emisor Lora durante esta prueba en función del tiempo:

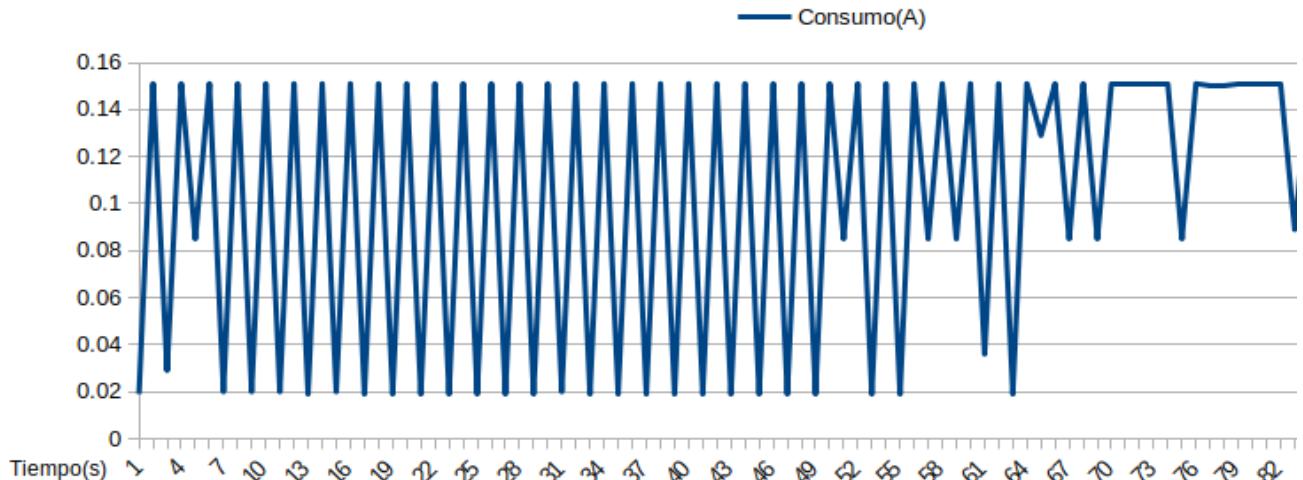


Figura 51: Gráfico del consumo del emisor Lora

Fuente: Elaboración propia

Como se puede observar, periódicamente aparecen picos de corriente que por una parte se deben al envío frecuente de datos y por otra a factores como la potencia de transmisión(14 dBm) o el *Spreading Factor*(12). Este comportamiento es algo completamente normal puesto que al usar coeficientes de *SF* y TX POWER más altos, la señal emitida puede llegar más lejos, sin embargo el consumo de batería aumenta.

- **Receptor Lora**

En la siguiente prueba vamos a medir el consumo que tiene el receptor Lora durante una transmisión frecuente de paquetes. Como ya se explicó en la prueba anterior, el envío se realiza cada segundo. Además, se aplican los mismos parámetros de comunicación. El objetivo de la prueba es observar el cambio de consumo del receptor en una situación como esta. El receptor Lora se conecta de la misma manera con el powerbank y el multímetro que en la prueba anterior. En el siguiente gráfico se puede apreciar el consumo que tiene el dispositivo durante el proceso de recepción de paquetes:

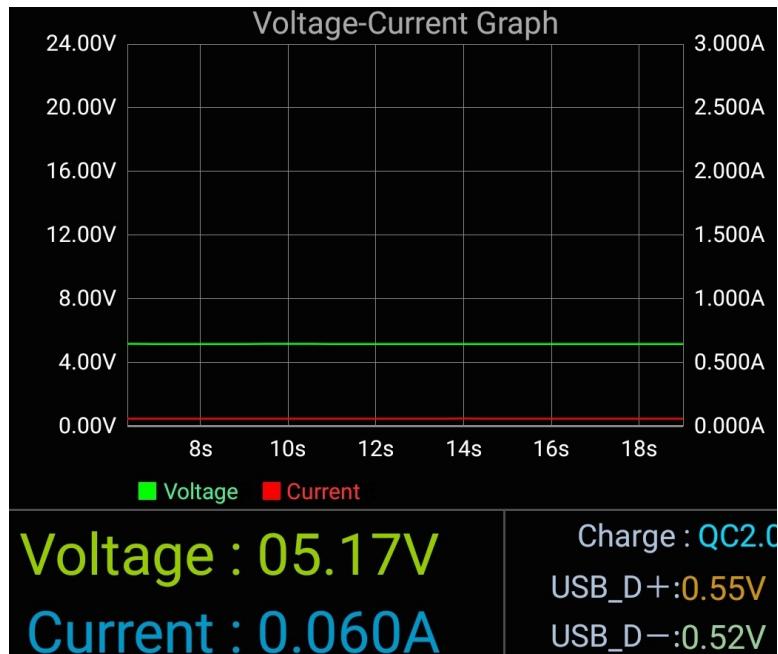


Figura 52: Gráfico del consumo del receptor Lora
Fuente: Elaboración propia

Durante esta prueba, el consumo del receptor llega hasta 60 mA aproximadamente, que es un valor bastante elevado. Se puede observar que este valor se mantiene fijo y no hay picos de consumo. Puesto que el gateway tiene un consumo bastante elevado, se va a necesitar una fuente de alimentación estable y capaz de mantener el dispositivo encendido durante largos periodos de tiempo. También es importante destacar que el gateway no puede entrar en modo de hibernación(*deepsleep*) ya que tiene que estar disponible para recibir paquetes en cada momento.

La batería utilizada para alimentar el receptor Lora durante esta prueba es de 1200mAh. El tiempo estimado de descarga sería el siguiente: $t = 1200/60$, $t = 20(h)$. Este tiempo es relativamente corto debido al alto consumo del módulo. Sin embargo, se puede alargar el tiempo de descarga de la batería cambiando algunos de los parámetros de comunicación.

- **Módulo GSM**

Tal y como se ha mencionado en el apartado 3.2.3.2 - *Configuración del GSM*, en determinadas ocasiones este módulo puede tener picos altos de consumo. Por ejemplo, cuando el GSM intenta conectarse a una red o comunicarse a través del GPRS, el consumo puede aumentar hasta más de 1A. Por lo tanto es muy importante tener una fuente de alimentación estable: 5V DC, 2A.

En esta prueba se han realizado varios envíos por GPRS con el fin de observar el consumo del GSM en cada momento. Como se puede observar en el siguiente gráfico,

aparecen varios picos de consumo. Los picos que aparecen entre los segundos **0 y 30**, ocurren al encender el dispositivo GSM por lo que este automáticamente empieza a buscar una red telefónica disponible. Este proceso es uno de los más costosos a nivel de consumo y ocurre en todos los dispositivos GSM que es algo completamente normal. El pico más grande es de aproximadamente 400 mA y ocurre poco antes de los 150 segundos. En este momento, el módulo envía los paquetes por GPRS al servidor y es cuando el consumo es el más elevado en toda la transmisión de datos.

Durante estas pruebas el GSM se ha apagado varias veces. El motivo es que el módulo se alimentaba desde el USB de un ordenador portátil y este puerto tiene una capacidad de 500mA. Como ya se puede deducir, esta capacidad no es suficiente y como consecuencia el módulo GSM se apagará.

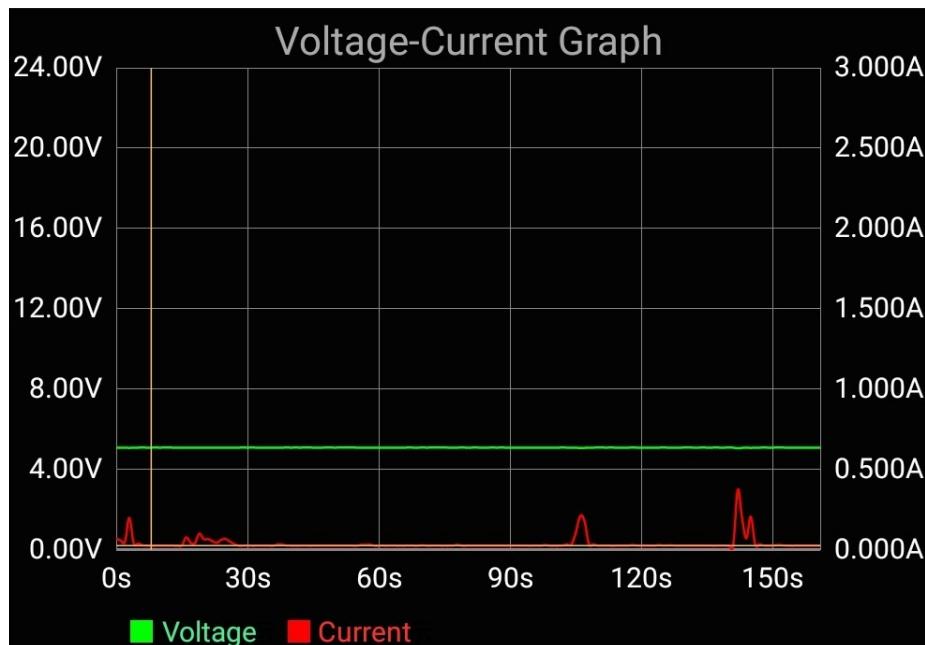


Figura 53: Gráfico del consumo del módulo GSM
Fuente: Elaboración propia

3.2.4.2. Tiempo de ejecución

En esta prueba se va a medir el tiempo de ejecución de los diferentes procesos del módulo GSM. El motivo de hacer las pruebas solamente con este módulo es que los procesos del mismo tienen un tiempo de ejecución más largo comparado con los demás módulos Lora. La prueba consistirá en dos partes, medir el tiempo que se tarda en iniciar el servicio GPRS y el tiempo de envío de un paquete a la plataforma IOT. El tiempo de respuesta del servidor no se tendrá en cuenta ya que este no depende del

módulo GSM, sino de otros factores como la cobertura, congestión de la red, carga del servidor y otros.

- Iniciar el servicio GPRS

En la siguiente captura aparecen los comandos AT que se han ejecutado con el fin de iniciar el servicio de GPRS. Después de cada comando, aparece la respuesta que nos devuelve el módulo GSM y el tiempo de ejecución de este. Cada uno de los comandos tarda un tiempo distinto en ejecutarse. Después de repetir la prueba cinco veces, se ha observado que los comandos tienen un tiempo de ejecución diferente en cada prueba. Esto se debe al hecho que el módulo GSM intenta conectarse a una operadora móvil y solicita una IP con el fin de conectarse al Internet. El tiempo de respuesta de dicha operadora puede variar según diferentes factores como calidad de la señal, congestión de la red, entre otros.

Se ha calculado el valor promedio y resulta que el servicio GPRS tarda en iniciarse aproximadamente 7 segundos.

Figura 54: Tiempo de inicio del servicio GPRS
Fuente: Elaboración propia

- Envío de paquetes por HTTP

En esta prueba se han realizado varios envíos de paquetes a la plataforma IOT. Los envíos se realizan mediante el protocolo HTTP desde el módulo GSM mediante los comandos AT. Cada comando AT tarda aproximadamente 1100 ms en ejecutarse. Durante cada una de las pruebas, un determinado comando AT tarda aproximadamente el mismo tiempo en ejecutarse aunque en ocasiones puede haber pequeñas variaciones. El tiempo total de envío es cerca de 9 segundos. Este tiempo no sería un problema ya que el propósito del gateway utilizado en este prototipo es enviar a la plataforma IOT pequeñas cantidades de datos con poca frecuencia. Por ejemplo, cada hora o varias veces al día.

Como ya se ha mencionado anteriormente, la idea principal del proyecto es enviar por GPRS los datos que obtienen los sensores del medio ambiente incluyendo temperatura, humedad, presión atmosférica, y otros. Para almacenar este tipo de información se requiere muy poca memoria, con uno o dos bytes sería suficiente. Además, estos valores no sufren variaciones bruscas a lo largo del día. Por este motivo, no son necesarios envíos frecuentes de datos.

Si queremos hacer envíos más frecuentes, por ejemplo cada segundo, o enviar paquetes de datos de varios MB, el tiempo de espera mencionado arriba haría que la comunicación sea ineficaz. Por ejemplo, utilizar este módulo GSM para hacer un live streaming, navegar por internet o manejar un vehículo en tiempo real, sería muy difícil o imposible.

En la siguiente captura aparecen los resultados de una de las pruebas de envío que se ha realizado:

```

--> Sending command: {"payload":[0,0,0,0,0,1,16]}

UART Response: [
OK
]

Execution time:
1100

--> Sending command: AT+HTTPACTION=1

UART Response: [
OK

+HTTPACTION: 1,601,0
]

Execution time:
1100

--> Sending command: AT+HTTPTERM

UART Response: [
OK
]

Execution time:
1100

### HTTPS Request sent ###
Total time: 9900

```

Figura 55: Tiempo de envío de una trama por GPRS
Fuente: Elaboración propia

3.2.4.3. Prueba de colisión de datos

Otro detalle muy importante a tener en cuenta es la gestión de paquetes recibidos por el Gateway. Como ya se ha mencionado en el apartado del protocolo de comunicación(3.2.3.4), el GSM puede enviar a la plataforma IOT solo un paquete a la vez. Por lo tanto, si durante el envío, el gateway recibe otro paquete de datos, no lo puede enviar al GSM, ya que este no está disponible. Para solucionar este problema, se podría utilizar algún método de bloqueo o guardar los paquetes entrantes en una cola de espera y hacer el envío cuando el GSM estuviera disponible de nuevo. Más detalles sobre esta funcionalidad se pueden ver en el apartado de *Futuras Mejoras*. En la captura siguiente se puede observar que el Gateway recibe varios paquetes pero solo se envía el primero al GSM, el resto quedará a la espera.

```

Received ASCII Received HEX
### Package received ###
0 : 0 : 0 : 1 : A :

### Package received ###
0 : 0 : 0 : 1 : A :

### Gateway Busy ###
### Package received ###
0 : 0 : 0 : 1 : A :

### Gateway Busy ###

```

Figura 56: Prueba de colisión de datos
Fuente: Elaboración propia

3.2.4.4. Prueba de comunicación en paralelo

Esta prueba consiste en observar el comportamiento del Gateway y el protocolo de comunicación durante una transmisión en paralelo. Es decir, nos interesa observar qué ocurre cuando varios nodos finales envían información al gateway al mismo tiempo. Puesto que un gateway puede tener varios clientes y los nodos finales pueden ser de diferentes fabricantes y modelos, no sabemos si la comunicación será fluida o habrá colisión y pérdida de datos. Para hacer esta validación, simplemente tenemos que conectar varios emisores Lora al Gateway al mismo tiempo y observar si todos los paquetes llegan correctamente. En la siguiente imagen podemos observar la estructura de la red de dispositivos Lora que se ha usado para esta prueba:

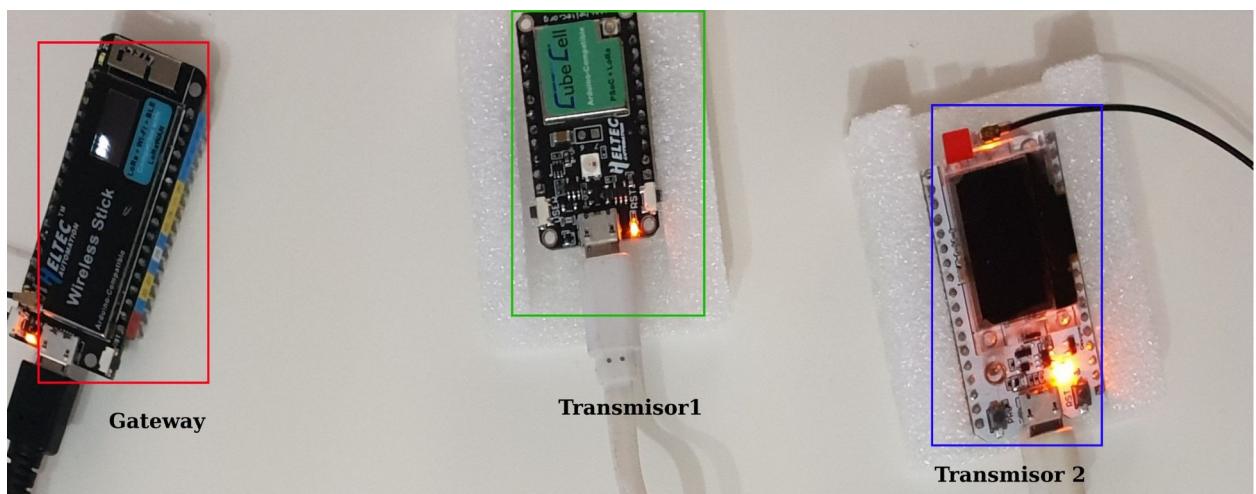


Figura 57: Fotografía de una red de dispositivos Lora
Fuente: Elaboración propia

Tal y como podemos ver, la red de dispositivos Lora está compuesta por dos emisores y un receptor. Los emisores envían paquetes con una frecuencia de 3 segundos. El envío frecuente de paquetes es importante para obtener los resultados deseados en esta prueba. El objetivo es intentar causar una congestión de paquetes en el gateway y observar su comportamiento.

En la siguiente captura se pueden ver los resultados de esta prueba:

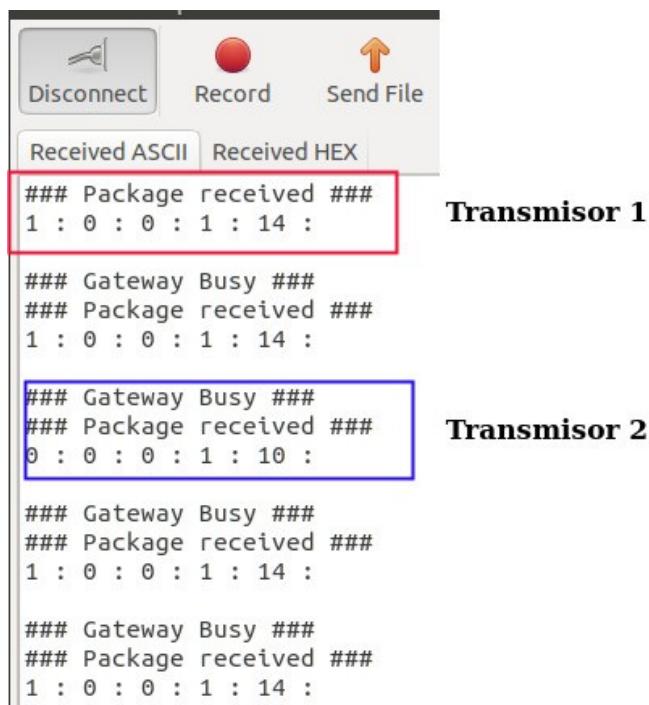


Figura 58: Comunicación en paralelo entre módulos Lora
Fuente: Elaboración propia

Podemos ver que el Gateway ha recibido 2 tipos de paquetes. Según el protocolo de comunicación que se está usando, el primer byte de cada trama corresponde al identificador del nodo final. Aquí claramente podemos distinguir entre los nodos **0** y **1** respectivamente. También podemos ver que el Gateway ha recibido varios paquetes de estos dos módulos sin ningún problema. Las tramas han llegado enteras, sin ruido o solapamiento. Sin embargo, al recibir paquetes con tanta frecuencia, el Gateway solamente enviará el primero en llegar al GSM, los demás como ya se ha explicado en el apartado anterior, quedarán guardados en una cola de espera.

3.2.4.5. Envío de datos a la plataforma IOT

En esta última fase de la implementación, se va a realizar una prueba de envío de datos desde el prototipo que se ha montado a la plataforma IOT. A pesar de haber realizado pruebas exhaustivas en cada una de las fases del desarrollo, es necesario hacer más pruebas de rendimiento con el prototipo entero instalado. Como ya sabemos, la red de dispositivos Lora, está compuesta por varios componentes. Cada uno de ellos tiene una estructura diferente y utiliza un protocolo de comunicación diferente por lo que es probable que surjan problemas de comunicación en un determinado momento.

Para poder realizar las pruebas de este apartado correctamente, es necesario tener la red de dispositivos Lora ya instalada. De esta manera se automatizará el proceso de comunicación y se podrá hacer envío de paquetes sin nuestra intervención.

En la siguiente imagen se puede observar el prototipo completo incluyendo los nodos finales y el gateway:

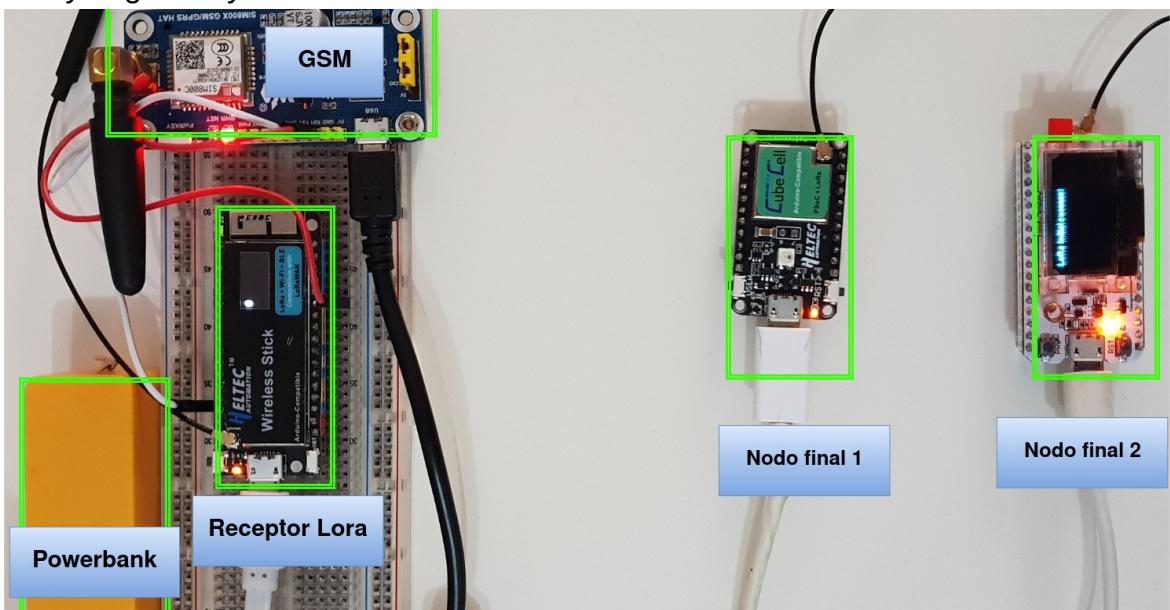


Figura 59: Fotografía del prototipo completo

Fuente: Elaboración propia

Tal y como podemos ver, esta red dispone de dos nodos finales y un gateway que consiste en un receptor Lora y un módulo GSM. El powerbank está alimentando al receptor Lora. El resto de dispositivos utilizan una fuente de alimentación distinta.

Suponiendo que los pasos y las pruebas anteriores se hayan cumplido con éxito, vamos a proceder al envío de los datos desde los nodos finales a la plataforma IOT. Primero de todo, tenemos que tener nuestra red de dispositivos montada con los módulos conectados a la fuente de alimentación tal y como indican los pasos anteriores. Como se puede ver en la figura 53, hay dos dispositivos conectados al gateway. Sin embargo, el protocolo de comunicación permite definir hasta 256 grupos

de dispositivos y hasta 256 IDs de nodo por cada grupo. Esto hace un total de 65535 dispositivos que se pueden conectar a un solo gateway. El receptor Lora utilizado en este prototipo probablemente no soportaría tantas conexiones simultáneas debido a su bajo potencial. Sin embargo, el protocolo de comunicación sí sería capaz de gestionar correctamente todas las conexiones.

Los nodos finales realizan una lectura de los sensores periódicamente. La frecuencia de estas lecturas puede variar según el tipo de sensor que se esté usando. Por ejemplo, valores como la temperatura ambiental, o la presión atmosférica pueden tener pequeñas variaciones a lo largo del día. Por lo tanto, hacer una lectura cada hora sería más que suficiente. Una vez hecha la lectura, los datos obtenidos se enviarán al Gateway y a continuación éste hará el reenvío a la plataforma IOT.

Para esta prueba, se ha programado al emisor Lora para enviar una trama con un determinado contenido. De este modo se puede saber si la trama llega correctamente a la plataforma o no. El contenido de la trama es el siguiente:

`{"payload": [0,0,0,0,0,1,22]}`. El dato que más nos interesa en este caso es el valor: **22**, que representa una supuesta lectura de un sensor térmico. En esta prueba los valores del payload son aleatorios, ya que no nos interesa tanto la lectura de un sensor o el origen de un mensaje, sino el envío correcto de los datos a la plataforma. Si esta prueba termina con éxito, el valor del sensor aparecerá en la plataforma IOT. En la captura de abajo podemos ver los comandos AT que ejecuta el GSM para enviar los datos a la plataforma IOT:

```

Received ASCII Received HEX
AT+HTTPINIT
OK
AT+HTTPPARA="CID",1
OK
AT+HTTPPARA="URL","http://demo.thingsboard.io/api/v1/T4heef9dzbcuejT5qegc,
telemetry"
OK
AT+HTTPPARA="CONTENT","application/json"
OK
AT+HTTPDATA=29,5000
DOWNLOAD
OK
AT+HTTPACTION=1
OK
+HTTPACTION: 1,200,0
AT+HTTPTERM
OK

```

Figura 60: Envío de una petición HTTP desde el GSM.
Fuente: Elaboración propia

Podemos observar que cada uno de los comandos vistos en la *tabla 5*, se ejecutaron con éxito. Uno de los pasos más importantes es la ejecución del comando: *AT+HTTPACTION*. Tal y como se ha explicado, este envía la petición HTTP al servidor y espera una respuesta. El código de esta respuesta es 200 lo que indica que el mensaje fue recibido por el servidor con éxito. Si el código del mensaje es distinto, tenemos que consultar el manual del protocolo HTTP para saber la causa del error.

En el dashboard de la plataforma IOT, veremos la última lectura que esta ha recibido del gateway. En la captura de abajo podemos ver los resultados de dicha lectura:

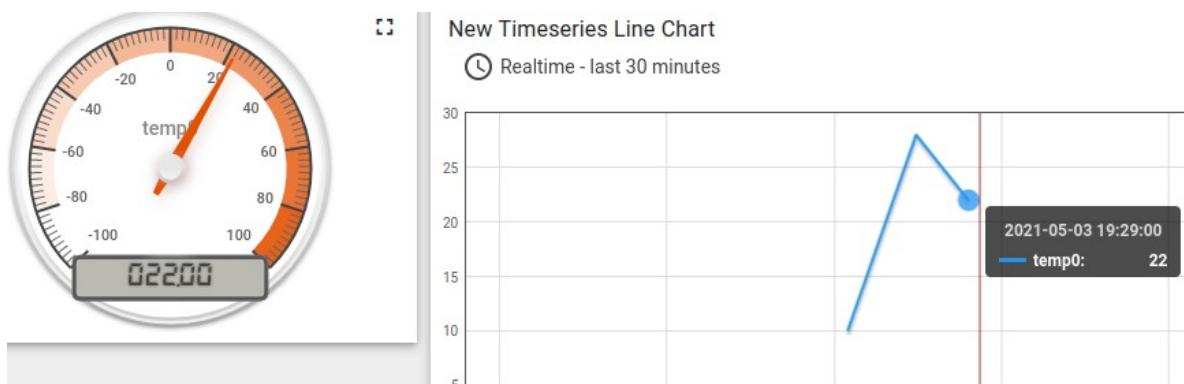


Figura 61: Widgets de la plataforma IOT.
Fuente: Elaboración propia

Claramente se puede ver que el valor enviado por el nodo final, aparece en la gráfica del dashboard. Con esto se puede concluir que la prueba ha tenido éxito. Los mensajes que se enviarán a partir de este momento deberían de llegar sin problemas, siempre y cuando se cumplan los pasos explicados anteriormente.

3.2.4.6. Historial de datos

Como ya sabemos, nuestra plataforma nos permite no solo visualizar los últimos datos que recibe en tiempo real, sino que también ver el historial de los envíos en un periodo de tiempo determinado. En la siguiente captura podemos ver el historial de datos que ha estado recibiendo la plataforma durante un tiempo determinado:

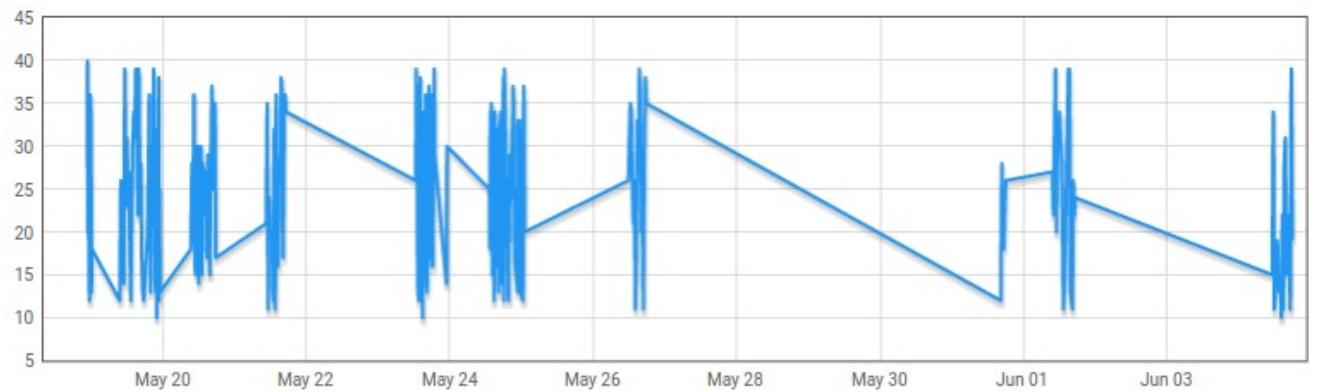


Figura 62: Historial de datos de la plataforma IOT

Fuente: Elaboración propia

3.2.4.7. Sistema de notificación

Una de las ventajas de este proyecto es que implementa un sistema de notificaciones que puede avisar a los usuarios sobre diferentes anomalías. Como vimos en el apartado 3.2.3.5, la plataforma realiza una validación de los datos recibidos por los gateways. Si ocurre algún fallo en algún módulo, la plataforma reciba datos no válidos o los valores están fuera de un determinado rango, la plataforma detectará el fallo y enviará una notificación al usuario vía email, SMS, Telegram, u otro medio.

A continuación veremos detalladamente cómo debemos configurar la plataforma para que esta pueda enviar notificaciones por Telegram. La gran ventaja de esta aplicación es que permite el envío de mensajes a un determinado perfil a través de una API. Por otra parte, Telegram es compatible con la mayoría de dispositivos móviles y hoy en día gran parte de los usuarios disponen de un teléfono móvil. Esto facilita muchísimo la comunicación entre una aplicación o servicio web y el usuario.

Lo primero que debemos hacer es crear un Telegram Bot[31] y guardar los parámetros *API KEY* y *chat_id*. Estos son esenciales para que podamos enviar mensajes al Bot que acabamos de crear a través de la API de Telegram.

El siguiente paso consiste en configurar la plataforma *thingsboard* para que esta pueda enviar mensajes a una API. Lo que debemos hacer es abrir la opción de *Rule Chain* del menú y añadir la opción *rest api call*. Es necesario tener el *Rule Chain* configurado tal y como hemos visto en la Figura 42 del apartado: 3.2.3.5. En la siguiente figura se puede apreciar el aspecto final que debería tener el *Rule Chain*:

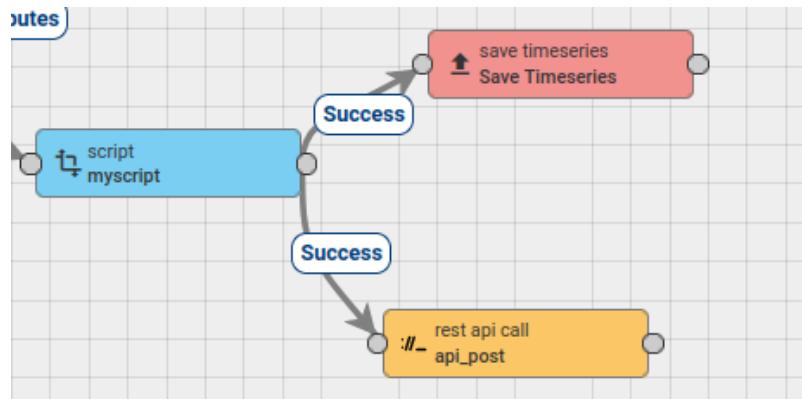


Figura 63: Historial de datos de la plataforma IOT

Fuente: Elaboración propia

El siguiente paso es configurar la opción *api call* aplicando los parámetros que hemos obtenido del nuestro Telegram Bot, es decir URL, API KEY y *chat_id*. En la siguiente captura podemos ver el menú de configuración de esta opción:



Figura 64: Historial de datos de la plataforma IOT

Fuente: Elaboración propia

El parámetro más importante es la URL de la API de Telegram que además contiene el API KEY. El menú también permite añadir otros parámetros. El *chat_id* lo podemos poner en este menú o asignarlo desde la opción del script.

En este último paso, tenemos que configurar el script que gestiona los mensajes entrantes. Lo único que se tiene que añadir es el mensaje que queremos enviar por Telegram y el *chat_id*. En la siguiente captura podemos observar el código utilizado para el envío de mensajes:

```

3 msg.text = "Warning! Temperature at node ["+NODE_ID+"] is exceeding safe levels: "+TEMP+" °C";
4
5 msg.chat_id = 227254225;
6
7 return {msg: msg, metadata: metadata, msgType: msgType};

```

Figura 65: Historial de datos de la plataforma IOT

Fuente: Elaboración propia

Los valores de las variables NODE_ID y TEMP ya se han asignado previamente. En el código que aparece en la *Figura 42* hemos visto con más detalle el proceso de asignación y validación de los datos. Tal y como está escrito el código de la *Figura 6*, se enviará un mensaje al perfil del Telegram Bot cada vez que la plataforma reciba una trama de datos.

Después de cumplir con los pasos de configuración, podemos pasar a la fase de pruebas. Para validar el sistema de notificaciones, podemos enviar un mensaje a la plataforma IOT desde la red de sensores que hemos instalado. En los apartados anteriores de validación, hemos confirmado que tanto la red de dispositivos Lora como la plataforma IOT están configuradas correctamente. Por lo tanto, lo único que queda por confirmar es el sistema de notificaciones.

En la siguiente captura se pueden apreciar los resultados de la prueba de envío de notificaciones:

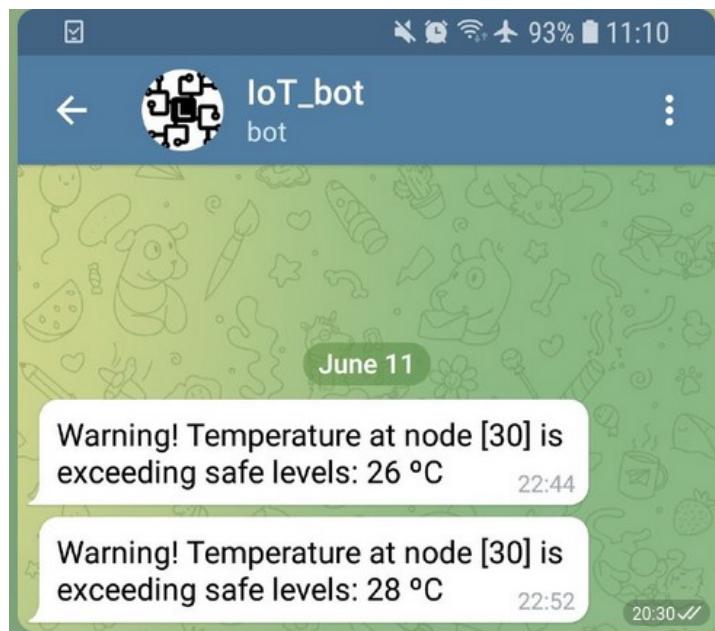


Figura 66: Telegram Bot
Fuente: Elaboración propia

Tal y como podemos ver, hemos recibido el mensaje definido en el script de la plataforma. Con esto podemos concluir que el sistema de notificaciones de la plataforma IOT funciona correctamente.

Página en blanco intencional

4. Análisis de rendimiento y estadísticas

En este capítulo se van a realizar pruebas de rendimiento con el fin de validar el comportamiento de los emisores Lora en diferentes situaciones. En el apartado 3.2.4. del capítulo anterior, se han realizado pruebas similares con los diferentes componentes incluyendo módulo GSM, emisor y receptor Lora. Sin embargo, el propósito de estas pruebas era validar el funcionamiento de los diferentes componentes y el prototipo en general. Ahora ya podemos asumir que todos los componentes funcionan correctamente y se comportan según lo esperado. Las pruebas que se van a realizar a continuación, analizan el rendimiento de los emisores Lora. Por ejemplo, nos interesa saber cómo cambia el consumo de energía o el tiempo de envío de los paquetes al aplicar diferentes parámetros de comunicación.

4.1. Prueba de consumo de energía

Para verificar el consumo de energía del emisor Lora, se han realizado 6 pruebas con una duración de 8 minutos cada. Las pruebas se llevaron a cabo utilizando el módulo *Cube Cell* visto en la *figura 46* del capítulo anterior. Como fuente de alimentación se ha utilizado un powerbank con una capacidad de 1200 mAh. Parámetros como *Bandwidth*, *Coding Rate* y frecuencia del envío de los paquetes se han mantenido fijos ya que estos no influyen en los resultados de ninguna manera. Sin embargo, se han utilizado diferentes valores de Spreading Factor y potencia de transmisión en cada una de las pruebas. En la tabla que aparece a continuación se pueden observar los resultados de las pruebas de consumo que se han realizado.

Es muy importante utilizar la frecuencia correcta en todos los módulos, de lo contrario se puede dañar el hardware. Otra cosa muy importante que se tiene que tener en cuenta es la potencia de transmisión. Las placas que se han usado en este proyecto tienen un rango entre 0 dBm y 20 dBm. Algunos países tienen restricciones sobre sobre algunas frecuencias y potencias de transmisión. Así que antes de configurar nuestra red Lora, debemos consultar las normas de la región en la que estamos trabajando[32].

Prueba Nº	SF	BW (kHz)	Potencia de Transmisión (dBm)	CR	Frecuencia de envío de paquetes(s)	Consumo (mA)
1	7	125	10	5	10	73
2	9	125	10	5	10	128
3	12	125	10	5	10	128
4	7	125	20	5	10	98
5	9	125	20	5	10	176
6	12	125	20	5	10	1014

Tabla 7: Resultados de las pruebas de consumo del emisor Lora
Fuente: Elaboración propia

Como podemos observar, el consumo del módulo aumenta a medida que aumenta el coeficiente del Spreading Factor. Los coeficientes de SF más altos permiten una mejor modulación con menos errores y alcanzar distancias más grandes. Sin embargo, este proceso requiere más energía lo que causaría una descarga más rápida de la batería.

Los valores que aparecen en la última columna son los picos de consumo que se producen durante el envío de paquetes en cada una de las pruebas. Estos valores no son exactos sino unas aproximaciones ya que al repetir una determinada prueba varias veces, se obtienen valores ligeramente diferentes. También se ha observado que la frecuencia de envío de paquetes no influye en los resultados de consumo. Es decir, el hecho de enviar paquetes cada minuto o cada 5 minutos causaría los mismos valores de consumo. Sin embargo, estos picos ocurrirían con menos frecuencia lo que aumentaría el tiempo de descarga de la batería.

Si observamos los resultados de las pruebas 1 y 4, veremos que aplicando el mismo coeficiente de SF pero con diferentes Potencias de Transmisión, obtenemos resultados diferentes, es decir 73 mA y 98 mA respectivamente. Esto quiere decir que la potencia también influye tanto en el proceso de transmisión como en el consumo y es independiente del SF.

En la gráfica que aparece a continuación, podemos observar los valores de consumo máximo, mínimo y medio en miliamperios obtenidos por cada una de las pruebas.

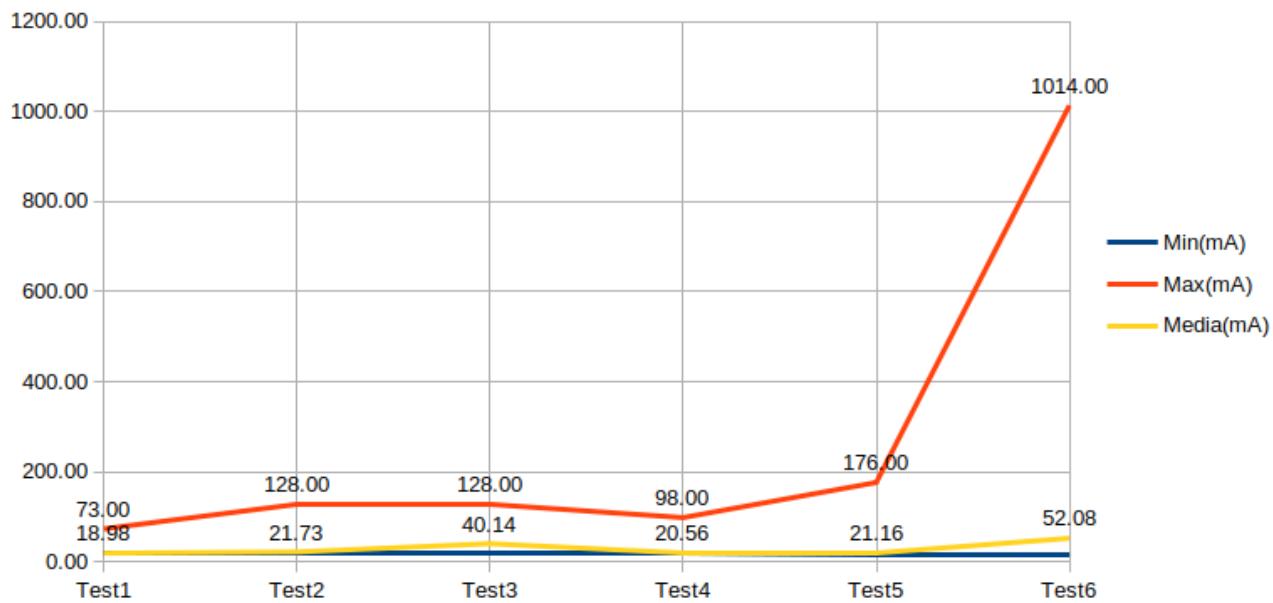


Figura 67: Gráfico del consumo del módulo Lora

Fuente: Elaboración propia

Tal y como podemos ver, las primeras 5 pruebas tienen unos valores máximos cercanos y la última tiene el valor más alto, 1014 mA. La causa son los parámetros que se están usando: SF12 y TX Power = 20 dBm(100 mW). Aplicar una configuración de este tipo no es recomendable debido al alto consumo de energía que ésta causaría en el dispositivo Lora. Si usamos una batería con la misma capacidad(1200 mAh), su duración sería de aproximadamente 23.04 horas.

En la siguiente tabla podemos ver información más detallada sobre los valores de consumo mínimos, máximos y medios obtenidos durante las pruebas. También está incluido el tiempo de duración de la batería.

Current(mA)	Prueba 1	Prueba 2	Prueba 3	Prueba 4	Prueba 5	Prueba 6
Min	18.00	19.00	18.00	18.00	16.00	17.00
Max	73.00	128.00	128.00	98.00	176.00	1014.00
Avg	18.98	21.73	40.14	20.55	21.16	52.08
Duration(h)	63.21	55.23	29.89	58.39	56.73	23.04

Tabla 8: Consumo y duración de la patería

Fuente: Elaboración propia

Tal y como se puede observar, los valores mínimos son casi idénticos en todas las pruebas. Esto ocurre porque durante su estado de reposo, el módulo no envía paquetes ni realiza ninguna otra operación, por lo que el consumo es mínimo. Los valores medios se utilizan para calcular la duración de la batería. Puesto que el

consumo de un dispositivo no es constante, sino que varía a lo largo de cada prueba, es necesario calcular la media de todas las muestras para obtener unos resultados más precisos.

El siguiente gráfico representa el tiempo de descarga de la batería según el consumo que tiene el dispositivo en cada una de las pruebas. Se puede ver claramente cómo disminuye el tiempo de duración de la batería a medida que aumenta el consumo.

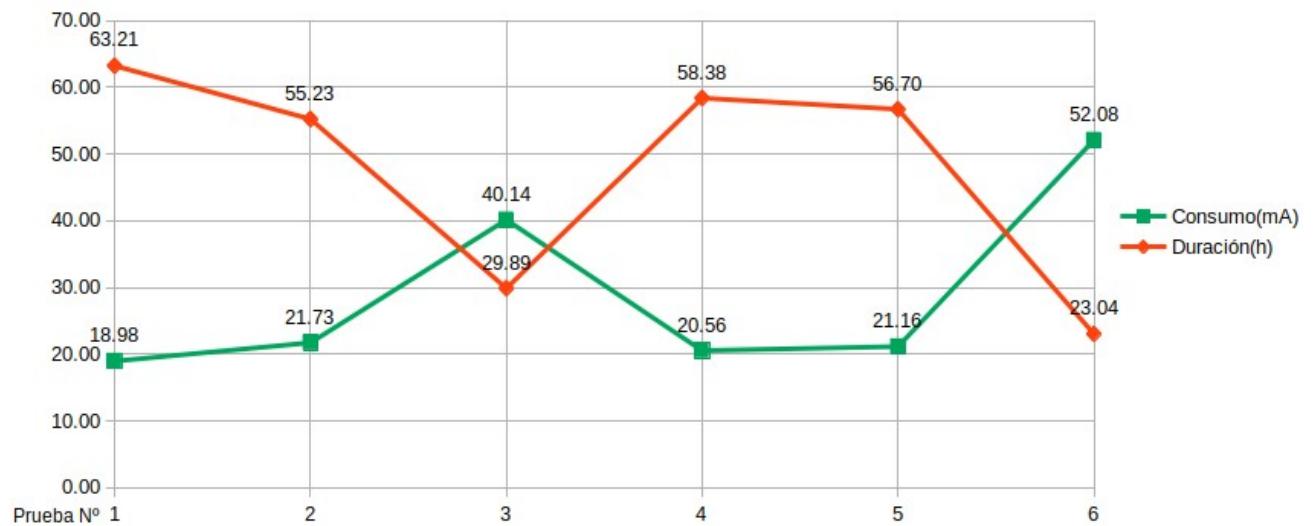


Figura 68: Gráfico de duración de la batería

Fuente: Elaboración propia

4.2. Tiempo de envío de los paquetes

En este apartado vamos a realizar una serie de pruebas con el fin de validar el tiempo de envío de los paquetes(*Time on Air*). Durante estas pruebas vamos a experimentar con tres parámetros: Spreading Factor, tamaño del paquete y potencia de transmisión. Nos interesa saber cómo varía el tiempo de envío de los paquetes y de qué factores o parámetros depende.

Los parámetros que aparecen a continuación también juegan un papel muy importante aunque se mantienen constantes durante las pruebas.

- Bandwidth: 125 kHz
- Coding Rate: 5
- Frecuencia de transmisión: 10 segundos

En la tabla que aparece a continuación podemos observar los resultados de las diferentes pruebas que se han realizado.

Prueba Nº	SF	Payload(bytes)	Time on Air(ms)	TX Power(dBm)
1	7	100	388	7
2	9	100	867	7
3	12	100	9251	7
4	7	100	388	14
5	9	100	867	14
6	12	100	9251	14
7	7	50	231	7
8	12	50	7560	7

Tabla 9: Tiempo de envío de los paquetes

Fuente: Elaboración propia

Como podemos observar, el tiempo en el aire de un paquete varía al cambiar el SF y el tamaño del payload. Al aplicar un SF7, el tiempo de envío es relativamente corto. Sin embargo, el SF12 hace que el paquete tarde hasta 9251 milisegundos en enviarse.

Tal y como hemos visto en el capítulo anterior, cuanto mayor sea el coeficiente del SF, más alcance tiene la señal. Sin embargo, el tiempo de propagación del paquete es mayor debido a una modulación y demodulación más lenta[33]. La distancia entre dos módulos Lora no influye en el tiempo de envío ya que la velocidad de propagación de las ondas electromagnéticas en el aire son cercanas a la velocidad de la luz 3×10^5 km/s, por lo que una distancia de varios kilómetros no marcaría una diferencia significativa.

Otro factor que influye en el tiempo de envío es el tamaño del payload, es decir los datos útiles que enviamos. Si hacemos una comparación de los resultados entre las pruebas Nº 1 y 7 de la *Tabla 9*, veremos cómo cambia el tiempo de envío de los paquetes al aplicar los mismos parámetros de SF y potencia de transmisión pero un tamaño del payload diferente. Podemos ver que hay una diferencia en el tiempo de envío. En una transmisión de paquetes, el protocolo Lora agrega datos adicionales en cada trama a parte del payload, es decir la cabecera y el footer[34]. Estos datos pueden aumentar ligeramente el tiempo de envío de un paquete.

Otro parámetro que juega un papel importante en el proceso de comunicación es la potencia de transmisión. Esta no afecta al tiempo de envío de los paquetes. Si comparamos los resultados de las pruebas 1 y 4 de la *Tabla 9*, veremos que al aplicar los mismos parámetros de comunicación pero una potencia diferente, el tiempo de envío de los paquetes no varía. Esto es así porque la potencia de transmisión afecta a

la calidad de la señal. Una potencia más alta implica más calidad y menos pérdidas de datos. La desventaja de una potencia más alta es el consumo de energía que aumenta exponencialmente tal y como hemos visto en las pruebas del apartado anterior. Otros parámetros como el ancho de banda y *Coding Rate* también pueden afectar al tiempo de envío, sin embargo no se han tenido en cuenta durante estas pruebas.

Página en blanco intencional

Página en blanco intencional

5. Conclusiones

6.1. Conclusiones a partir de los resultados

Basándonos en el estudio previo y los resultados obtenidos después de realizar las diferentes pruebas, podemos concluir que el prototipo que se ha construido es relativamente estable ya que cumple con el objetivo principal. Como ya se ha explicado anteriormente, el propósito de este proyecto es construir una red de sensores inalámbricos basados en las tecnologías Lora y GSM. De esta manera se puede conseguir una cobertura a nivel mundial.

El prototipo desarrollado en este proyecto es solo un producto de valor mínimo(MVP). A pesar de haber superado todas las pruebas y cumplir con los requisitos mínimos, por el momento dispone de unas funcionalidades muy básicas. Como ya se mencionó anteriormente, la estructura de este proyecto es relativamente sencilla lo que permite aplicar distintos cambios, mejoras y añadir nuevas funcionalidades fácilmente. Esto nos permite escalar el proyecto, y adaptarlo según las necesidades que tengamos.

Según los resultados obtenidos del capítulo anterior, podemos concluir que este producto es ágil y se adapta fácilmente a diferentes necesidades o condiciones. Como ya hemos visto en la fase de pruebas, el prototipo tiene un comportamiento diferente según los parámetros de configuración aplicados. La selección de los parámetros de transmisión tiene un impacto en el rendimiento de las comunicaciones. Más notablemente, la selección impacta en el rango de transmisión y la resistencia a las interferencias. Además, la selección tiene un gran impacto en el consumo de energía del dispositivo. En la mayoría de las situaciones, es deseable equilibrar el rendimiento de las comunicaciones y el consumo de energía, ya que los nodos funcionan con baterías y el objetivo es maximizar la vida útil. Obviamente, la potencia de transmisión tiene un impacto directo en el consumo tal y como hemos visto en el apartado anterior. Otro factor que determina el consumo de energía es el tiempo de aire requerido para transmitir un paquete, que depende del *bitrate* y el tamaño del paquete.

Como ya hemos visto, el rendimiento de una red Lora depende de varios parámetros de comunicación. Cada uno de estos parámetros tiene un impacto diferente como el tiempo en el aire, consumo de energía, bitrate, y otros. Por lo tanto, no existe una configuración óptima, sino que tenemos que configurar nuestra red Lora con los parámetros adecuados según la situación. Esto requiere realizar diferentes pruebas de comunicación entre los nodos finales y el gateway, evaluar los resultados y seleccionar los parámetros óptimos. De lo contrario, podríamos estar desaprovechando los recursos como disminuir el tiempo de duración de la batería. Este resultado no es deseado en una red Lora ya que el objetivo es tener dispositivos de bajo consumo tratando de maximizar su vida útil.

6.2. Futuras mejoras

Los componentes utilizados en este prototipo son relativamente básicos. Si se desea construir una red de sensores pequeña, estos componentes servirán perfectamente. Sin embargo, para proyectos a gran escala deberíamos usar dispositivos más sofisticados, con más capacidad de procesamiento. También se pueden aplicar varias mejoras en el software que podrían incrementar el rendimiento del proyecto en el futuro.

Una posible mejora sería implementar una aplicación para dispositivos móviles que nos permita leer los datos de la plataforma IOT. La aplicación también nos permitirá configurar los diferentes nodos finales o gateways remotamente.

Otra mejora interesante sería implementar unos métodos de comunicación más sofisticados. De esta manera los gateways podrían gestionar los paquetes de una manera más eficiente. Tal y como hemos visto en el apartado del *Protocolo de Comunicación*, el gateway puede gestionar sólo un paquete a la vez. El caso ideal sería poder guardar los paquetes que recibe el gateway en una cola de espera. De esta manera los datos no se pierden y serán enviados a la plataforma IOT cuando el módulo GSM estuviera disponible. Sobre las posibles pérdidas y reenvíos de paquetes entre el nodo final y el gateway no tenemos que preocuparnos ya que este proceso se gestiona por el protocolo de Lora.

La gestión de los sensores por parte del protocolo no sería un problema. Tal y como hemos visto en el apartado **3.2.3.4**, el payload del protocolo puede tener una longitud indeterminada, lo que nos permite gestionar una gran cantidad de sensores(más de 100). Un posible obstáculo que podemos encontrar es a nivel de hardware. Puesto que las diferentes placas de desarrollo o microcontroladores tienen tamaño y capacidad diferentes, es posible que una placa no tenga suficientes pines o conectores para poder gestionar un número elevado de sensores.

Página en blanco intencional

Página en blanco intencional

6. Referencias

- [1] "Características y usos del internet de las cosas". [En Línea]. Disponible en:

<https://uelectronics.com/caracteristicas-y-usos-del-internet-de-las-cosas-iot/>

[2] Jeff Desjardins (2019). "The anatomy of a smart city", [En Línea]. Disponible en: <https://www.visualcapitalist.com/anatomy-smart-city/>

[3] IoT analytics (2018). "The Top 10 IoT Segments", [En Línea]. Disponible en: <https://iot-analytics.com/top-10-iot-segments-2018-real-iot-projects/>

[4] "¿Qué es Amazon Alexa y qué puedes hacer con los dispositivos Echo?". [En Línea]. Disponible en: <https://tecnologiaencasa.com/amazon-alexa/>

[5] United Nations. "World Urbanization Prospects.". [En Línea]. United Nations, New York, 2014. Disponible en: <https://population.un.org/wup/publications/files/wup2014-highlights.pdf>

[6] "Piloto automático | Tesla". [En Línea]. Disponible en: https://www.tesla.com/es_ES/autopilot?redirect=no

[7] "¿Què és el metro automàtic?". [En Línea]. Disponible en: <https://www.tmb.cat/ca/sobre-tmb/millors-xarxa-transport/metro-automatic/que-es>

[8] "LUCA, la unidad de datos de Telefónica". [En Línea]. Disponible en: <https://www.telefonica.com/es/web/sala-de-prensa/-/luca-la-unidad-de-datos-de-telefonica-posicionada-como-lider-entre-proveedores-especializados-de-insights-de-big-data>

[9] "Wireless RF". Disponible en: <https://www.semtech.com/products/wireless-rf/lora-gateways>

[10] BoTrue White paper (2018). "Aplicación de la tecnología LoRaWAN en la agricultura". [En Línea], Disponible en: https://www.researchgate.net/publication/323883830_Aplicacion_de_la_tecnologia_LoRaWAN_en_la_agricultura

[11] "LPWAN: qué son y para qué se utilizan". [En Línea], Disponible en: <https://www.m2mlogitek.com/lpwan-que-son-y-para-que-se-utilizan/>

[12] "Lora - AlphaloT". [En Línea]. Disponible en: <https://alfaiot.com/tecnologias-iot/lora/>

[13] "¿Qué es Sigfox?". [En Línea]. Disponible en: <https://sigfox.com.py/que-es-sigfox/>

[14] "Las aplicaciones de la vida real del IoT y por qué la duración de la batería es algo crítico". [En Línea]. Disponible en: <https://www.sigfox.es/blogs/post/sigfox-las->

[aplicaciones-de-la-vida-real-del-iot-y-por-qu%C3%A9-la-duraci%C3%B3n-de-la-bater%C3%ADa-es-algo-cr%C3%ADtico](#)

[15] "Principales características de la tecnología Sigfox". [En Línea]. Disponible en: <http://www.slb-systems.com/sensores-sigfox/>

[16] "Failure To Launch: Why Sigfox Failed In The US". [En Línea]. Disponible en: <https://www.iotacommunications.com/blog/sigfox-usa/>

[17] "¿Qué es la red GSM y cómo funciona?". [En línea]. Disponible en: <https://es.ccm.net/contents/681-estandar-gsm-sistema-global-de-comunicaciones-moviles>

[18] "Sistemas de Monitoreo de Variables Ambientales". [En Línea]. Disponible en: <https://www.rosemlak.com/sistemas-de-monitoreo-ambiental>

[19] "ReliaSENS 19-15 - Environmental Monitoring System". [En Línea]. Disponible en: <https://www.eurotech.com/en/products/intelligent-sensors/environmental-monitoring-systems/reliasens-19-15>

[20] "UART communication protocol". [En línea]. Disponible en: <https://www.codrey.com/embedded-systems/uart-serial-communication-rs232/>

[21] "AT Commands". [En línea]. Disponible en: <https://doc.qt.io/archives/qtextended4.4/atcommands.html>

[22] "The moserial Project". [En línea]. Disponible en: <https://wiki.gnome.org/Apps/Moserial>

[23] Simcom. "SIM800_Hardware Design_V1.09". [En Línea]. China: Shanghai SIMCom Wireless Solutions Ltd. 2016. Disponible en: https://simcom.ee/documents/SIM800/SIM800_Hardware%20Design_V1.09.pdf

[24] "Multímetro - USB Color Tester Instructions". [En Línea]. Disponible en: <https://drive.google.com/file/d/11tz9vqYqnZbL5xqTpJCnOxriFgvAddAn>

[25] "Decoding Lora". [En Línea]. Disponible en: <https://revspace.nl/DecodingLora>

[26] "Libelium Wasp Mote". [En Línea]. Disponible en: <https://development.libelium.com/ide-user-guide/installation>

[27] "Wireless stick pinout diagram". [En línea]. Disponible en: https://resource.heltec.cn/download/Wireless_Stick/Wireless_Stick.pdf

[28] "Bitwise left and right shift operators". [En Línea]. Disponible en: <https://www.ibm.com/docs/en/i/7.2?topic=expressions-bitwise-left-right-shift-operators>

[29] Simcom. “SIM800 Series SSL Application Note”. [En Línea]. China: Shanghai SIMCom Wireless Solutions Ltd. 2016. Disponible en:
https://simcom.ee/documents/SIM800x/SIM800%20Series_SSL_Application%20Note_V1.02.pdf

[30] “Tiempo de descarga de la batería”. [En línea]. Disponible en:
<https://es.planetcalc.com/2283/>

[31] “Bots: An introduction for developers”. [En Línea]. Disponible en:
<https://core.telegram.org/bots>

[32] “Lora Documentation”. [En línea]. Disponible en:
<https://lora.readthedocs.io/en/latest/>

[33] “What are LoRa® and LoRaWAN®?”. [En Línea]. Disponible en: <https://lora-developers.semtech.com/library/tech-papers-and-guides/lora-and-lorawan/>

[34]. “LoRa- (Long Range) Network and Protocol Architecture & Frame Structure”. [En Línea]. Disponible en: <https://www.techplayon.com/lora-long-range-network-architecture-protocol-architecture-and-frame-formats/>