



Escola d'Enginyeria de Telecomunicació i
Aeroespacial de Castelldefels

UNIVERSITAT POLITÈCNICA DE CATALUNYA

TRABAJO FINAL DE GRADO

TÍTULO DEL TFG: Evaluación de LoRa/LoRaWAN para escenarios de Smart City

TITULACIÓN: Grado en Ingeniería Telemática

AUTOR: Rubén Pérez García

DIRECTORES: Carles Gómez Montenegro, Rafael Vidal Ferré

FECHA: 5 de Febrero del 2017

Título: Evaluación de LoRa/LoRaWAN para escenarios de Smart City

Autor: Rubén Pérez García

Directores: Carles Gómez Montenegro, Rafael Vidal Ferré

Fecha: 5 de Febrero del 2017

Resumen

El objetivo principal de un proyecto de smart city debe ser mejorar la economía, la sociedad, el entorno y la calidad de vida de sus ciudadanos y hacerla sostenible. Estos proyectos deben planificarse y ejecutarse con la idea de que sean duraderos y proporcionen servicio a largo plazo.

La Universidad Politécnica de Catalunya (UPC), en concreto la Escola d'Enginyeria de Telecomunicació i Aeroespacial de Castelldefels (EETAC), ha iniciado un proyecto de smart city en las inmediaciones del Campus del Baix Llobregat consistente en desplegar una red de sensores capaz de cubrir un amplio terreno como es el Campus y parte de la ciudad de Castelldefels.

En este proyecto se propone un estudio detallado sobre las prestaciones y limitaciones de los protocolos de comunicaciones inalámbricas para sensores LoRa/LoRaWAN, considerando herramientas analíticas y experimentales. El proyecto incluye la descripción del proceso de implementación de una red de sensores en un escenario real como es el Campus del Baix Llobregat de Castelldefels empleando esta tecnología y detallando las herramientas, el software y la configuración utilizadas para el correcto funcionamiento de una red de este tipo.

A lo largo de este documento se mostrarán los resultados obtenidos en las distintas pruebas realizadas en laboratorio junto a conclusiones con respecto a su correcto funcionamiento en un posible escenario real. Las pruebas abarcan la verificación de las distintas características que componen cada protocolo, la configuración de equipos y dispositivos para construir una red LoRa/LoRaWAN que permita almacenar los datos recogidos por los diferentes sensores que componen la red (ofreciendo la oportunidad de acceder a ellos de forma remota y con cualquier dispositivo validado con conexión a Internet) y un estudio de prestaciones en términos de consumo, throughput y cobertura.

Title: Evaluation of LoRaWAN for Smart City scenarios

Author: Rubén Pérez García

Directors: Carles Gómez Montenegro, Rafael Vidal Ferré

Date: February 5 th 2017

Overview

The main objective of a Smart City project should be to improve the quality life of citizens based on sustainability and energy efficiency. These projects are planned and implemented with the idea of being durable and provide a long-term service.

The Universidad Politécnica de Catalunya (UPC), specifically the Escuela de Ingeniería de Telecomunicaciones y Aeroespacial de Castelldefels (EETAC), has started a project of Smart City in the vicinity of the Campus of Baix Llobregat consisting of building a network of sensors capable of covering a large area such as the campus and part of the city of Castelldefels.

This project proposes a detailed study on the performance and limitations of the LoRa/LoRaWAN sensors wireless communication protocols considering analytical and experimental tools, as well as the description of the process of implementing a sensor network in a real scenario as is the Campus of the Baix Llobregat of Castelldefels using this technology detailing the tools, software and configuration used for the correct operation of a LoRa / LoRaWAN sensor network.

This document shows the results obtained in the concrete tests carried out in the laboratory together with conclusions in the context of a real scenario. The tests range from verifying the characteristics of each of the protocols to the configuration of equipment and devices to build a LoRa network that allows to store the data collected by the different sensors that make up the network, offering the opportunity to access them remotely and from any authorized device with Internet connection.

ÍNDICE

INTRODUCCIÓN	1
CAPÍTULO 1. LORA Y LORAWAN.....	3
1.1 LoRa.....	3
1.1.1 Red LoRa.....	4
1.1.2 Canales y rangos de frecuencias	4
1.2 LoRaWAN	5
1.2.1 Red LoRaWAN	6
1.2.2 Modos de acceso a una red LoRaWAN	7
1.2.3 Canales y velocidades de transmisión	7
1.2.4 Clases LoRaWAN.....	10
1.2.5 Seguridad en LoRaWAN	11
1.3 Características de una red LoRaWAN.....	11
1.3.1 Potencia de transmisión	12
1.3.2 Estructura de paquetes LoRaWAN	12
1.3.3 Comunicación entre nodos.....	14
CAPÍTULO 2. ENTORNO DE TRABAJO	17
2.1 Dispositivos LoRa	17
2.1.1 Transceptor LoRa.....	18
2.1.2 Wasp mote.....	19
2.1.3 Batería	20
2.1.4 Antena	20
2.2 Dispositivos LoRaWAN.....	20
2.2.1 Módulos LoRaWAN	21
2.2.2 Gateway LoRaWAN	21
2.3 Software.....	23
2.3.1 Wasp mote PRO.....	23
2.3.2 Raspbian.....	24
2.3.3 Python.....	25
2.3.4 Cutecom	26
2.4 Librería Wasp mote API	26
2.4.1 Potencia de transmisión	26
2.4.2 Modos de transmisión	27
2.4.3 Estructura de paquetes LoRa.....	27
2.4.4 Comunicación entre nodos.....	28
2.4.5 Seguridad	31
CAPÍTULO 3. EXPERIMENTOS Y RESULTADOS CON LORA/LORAWAN .	32
3.1 Consumo en LoRaWAN	32
3.2 Throughput en LoRa y en LoRaWAN	39

3.3 Cobertura de LoRa/LoRaWAN.....	42
CAPÍTULO 4. CONCLUSIONES	46
4.1 Líneas futuras	47
4.2 Estudio de ambientación	47
BIBLIOGRAFÍA	48
ANEXOS	50
I Puesta a punto del gateway LoRa	50
I.1 Configuración del LoRa Gateway.....	50
I.2 Configuración de una Raspberry Pi para controlar el LoRa Gateway	52
II Configuración de los módulos LoRa	58
II.1 Transmisión en módulos LoRa.....	60
II.2 Recepción en módulos LoRa	63
II.3 Pruebas de cobertura	66
II.4 Sniffing.....	69
III Puesta a punto del gateway LoRaWAN	71
IV Configuración de los módulos LoRaWAN	79
V Montaje para realizar las pruebas de consumo en LoRaWAN.....	82

INTRODUCCIÓN

Internet of Things (IoT) es un concepto prometedor en el presente gracias al enorme número de dispositivos que cuentan con la posibilidad de interconectarse y a la necesidad de crear redes capaces de gestionar estos dispositivos. A su vez, estas redes deberán ser capaces de gestionar el considerable tamaño de datos que puedan generar utilizando diferentes tecnologías de red a nivel físico y de enlace.

En este ámbito, las Low-Power Wide Area Networks (LPWANs) están emergiendo, y centralizando el interés de la industria, la academia y organismos de estandarización. Las LPWAN se caracterizan por ofrecer conectividad inalámbrica de largo alcance, baja potencia y bajo coste entre "things" u objetos conectados, tales como sensores alimentados mediante baterías. Asimismo, las LPWAN presentan limitaciones significativas en cuanto a ancho de banda, tasa de mensajes y tamaño de las tramas.

Las tecnologías LPWAN están diseñadas para entornos Machine to Machine (M2M). Estas redes pueden proporcionar conectividad a un gran número de dispositivos alimentados por baterías. Con menor consumo de energía, mayor alcance y menor coste que las redes móviles, se cree que las LPWAN conquistarán las comunicaciones M2M y el ámbito de la IoT.

Algunas de sus características y limitaciones descritas por numerosos estudios [1], [2] son:

- Ancho de banda reducido, que puede comprender desde los 50 bits/s hasta los 250 kbits/s con ciclos de trabajo de entre 0.1% y 10%.
- Tramas muy cortas, con un payload que puede comprender desde los 12 bytes hasta los 222 bytes.
- Alta probabilidad de pérdida de paquetes, por motivos de colisión o malas condiciones de transmisión.
- Soporte para centenares de miles de nodos por estación base.
- Bajo consumo, permitiendo a los dispositivos operar con baterías durante un largo periodo de tiempo.
- Capacidad de trabajar en distintas bandas de frecuencias, algunas de ellas sin licencia, con numerosos canales de radio.
- Comunicación unidireccional o bidireccional.
- Alcance de la comunicación en exteriores capaz de superar los 10 km de distancia en algunos casos con visibilidad directa y buena penetración en interiores.
- Infraestructuras de bajo coste, escalabilidad e implementación sencilla.

Algunos ejemplos de tecnologías LPWAN son LoRa/LoRaWAN, SigFox, IEEE 802.15.4k LECIM, IEEE 802.15.4g SUN, RPMA, DASH-7 o Weightless.

LoRa/LoRaWAN es una de las tecnologías LPWAN paradigmáticas que actualmente lidera el sector de las LPWAN en términos de despliegue de redes. Sus características hacen que tenga aplicación en escenarios relevantes

como las denominadas smart cities. Para ello es necesario conocer las prestaciones de esta tecnología, con el fin de conseguir que su despliegue y configuración se realicen con éxito.

A lo largo de este documento dividido en cuatro capítulos, se analizarán las características que ofrece la tecnología LoRa/LoRaWAN con tal de evaluar su uso en una red de sensores para smart cities, mostrando como ejemplo los datos obtenidos en pruebas de laboratorio y en un escenario real desplegado en el Campus del Baix Llobregat de Castelldefels.

En el primer capítulo se ofrecerá una introducción a la tecnología LoRa y al protocolo LoRaWAN, donde se expondrán su definición y características más relevantes, así como lo necesario para entender su funcionamiento y comprender el resto de capítulos.

En el segundo capítulo se presentará el software y hardware necesario para implementar una red LoRa y una red LoRaWAN. La explicación detallada de los programas utilizados se trasladará a los anexos y los detalles sobre los elementos físicos que conforman los escenarios mostrados en este capítulo se verán al final del capítulo especificando únicamente los datos necesarios para comprender este documento.

En el tres capítulo se mostrarán los resultados obtenidos en las distintas pruebas realizadas en el laboratorio y en los escenarios reales llevados a cabo en el campus con sus respectivas conclusiones, a la vez que se comparan los resultados prácticos con los esperados de forma teórica. La descripción del proceso de implementación de la red de sensores, desde la configuración del gateway hasta la configuración de cada nodo LoRa/LoRaWAN, se encuentra detallada en los anexos al final de este documento.

Para acabar, se dedicará un cuarto capítulo a exponer las conclusiones extraídas de la realización de este proyecto referentes al uso de esta tecnología en un despliegue real de una red de sensores en una smart city.

A continuación de este último capítulo, se encuentran los distintos anexos que se ocupan de los pasos y procesos para preparar y configurar cada uno de los dispositivos utilizados en ambas redes, LoRa y LoRaWAN, junto a las explicaciones pertinentes para que el lector sea capaz de reproducirlos y crear su propia red si cuenta con el material necesario.

CAPÍTULO 1. LORA Y LORAWAN

En el primer capítulo se darán a conocer las tecnologías principales que son objeto de este proyecto, LoRa y LoRaWAN. Así, este capítulo servirá para presentar conocimientos previos para lograr la comprensión de los capítulos que vendrán a continuación.

LoRa/LoRaWAN han logrado posicionarse como las principales tecnologías dentro de las LPWAN, gracias a algunos requisitos clave como los servicios de comunicación bidireccionales, la facilidad de interoperabilidad entre sensores sin necesidad de instalaciones complejas y la libertad que ofrece a usuarios, desarrolladores y empresas para el despliegue de IoT.

LoRa define una tecnología de capa física desarrollada por Cycleo en 2010, empresa que dos años más tarde fue adquirida por Semtech, mientras que LoRaWAN es una especificación de red propuesta por la LoRa Alliance en 2015 que ofrece una capa MAC basada en la modulación LoRa. A continuación se describen ambas tecnologías.

1.1 LoRa

LoRa, abreviatura de Long Range, es una técnica de modulación basada en técnicas de espectro ensanchado y una variación de *chirp spread spectrum* (CSS), que modula los datos sobre diferentes canales y velocidades, con corrección de errores *forward error correction* (FEC) integrada. LoRa mejora significativamente la sensibilidad del receptor y, al igual que con otras técnicas de modulación de espectro ensanchado, utiliza toda la anchura de banda del canal para transmitir una señal, haciéndola robusta al ruido del canal e insensible a las compensaciones de frecuencia causadas por el uso de cristales de bajo coste [3]. LoRa puede demodular señales 19.5 dB por debajo del nivel de ruido, mientras que la mayoría de los sistemas de desplazamiento de frecuencia (FSK) necesitan una potencia de señal de 8-10 dB por encima del nivel de ruido para demodular adecuadamente [4].

LoRa se describe como un *frequency modulated (FM) chirp*, basándose en una modulación de espectro ensanchado, que mantiene las mismas características de baja potencia como la modulación FSK pero aumentando significativamente el alcance de la comunicación. Utiliza ganancia de codificación para aumentar la sensibilidad del receptor y es capaz de alcanzar velocidades de transmisión comprendidas entre 0.3 kbps hasta 38.4 kbps. Por tanto, es adecuado para comunicaciones P2P entre nodos, ie. comunicaciones donde los nodos pueden conectarse directamente entre ellos.

1.1.1 Red LoRa

En modo P2P (ver **Fig. 1.1**) los nodos pueden conectarse directamente entre ellos sin costes, ya que no utilizan infraestructura de red. Este modo funciona sin la necesidad de una estación base o una cuenta Cloud, y por tanto no requiere comprar ninguna licencia. Este modo es el único en el que pueden trabajar los nodos LoRa.



Fig. 1.1 Esquema modo P2P (LoRa y LoRaWAN) [5]

1.1.2 Canales y rangos de frecuencias

LoRa puede trabajar en varios rangos de frecuencia en distintas regiones del mundo. La banda de frecuencias utilizada en Europa es la banda ISM de 863-870 MHz regulada por el ETSI. Utiliza 8 canales elegidos arbitrariamente, según la UN-111 vigente en el BOE-A-2013-4845 español, con un ancho de banda de 0.3 MHz por canal (ver **Fig. 1.2**). La banda de frecuencias utilizada en USA, Canadá, Australia, Singapur o Israel es la banda ISM de 902-928 MHz, utilizando 13 canales con un ancho de banda de 2.16 MHz por canal. Estos canales se escogieron arbitrariamente ajustándose a los canales utilizados para el uso de XBee¹ de 900 MHz [6].

¹ Xbee es un conjunto de dispositivos que hacen uso de transceptores Zigbee modificados. Zigbee es un protocolo de comunicaciones inalámbrico basado en el estándar IEEE_802.15.4, el cual define el nivel físico y MAC de las WPAN (Wireless Personal Area Network).

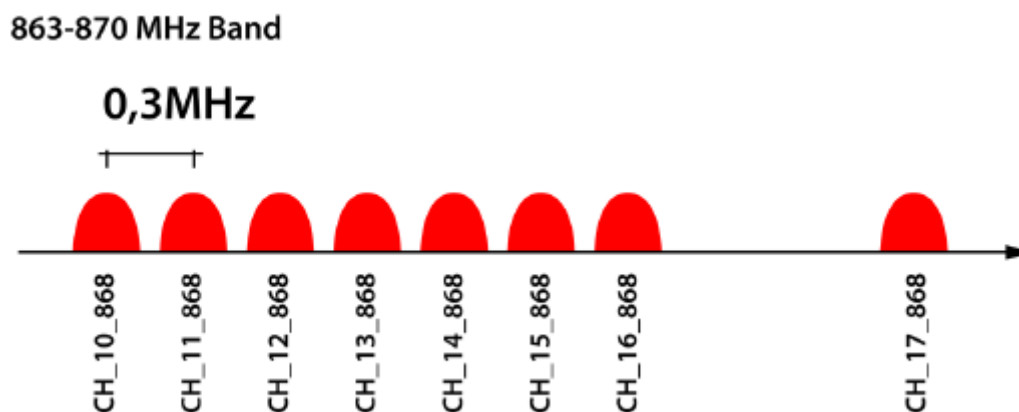


Fig. 1.2 Banda de frecuencias europea utilizada para LoRa [6]

La mayor virtud de LoRa está en la capacidad de alcance de esta tecnología y en su escalabilidad. Un gateway o una estación base puede cubrir ciudades enteras o áreas de cientos de kilómetros cuadrados.

Semtech es el fabricante de los módulos LoRa y ofrecen a disposición del usuario una librería programada que permite la comunicación entre nodos LoRa mediante un protocolo de enlace simple. Esta librería ha sido creada por Libelium, empresa con sede en Zaragoza que proporciona las herramientas y librerías necesarias para operar con LoRa o incluso para integrar seguridad en la red.

1.2 LoRaWAN

La modulación LoRa ofrece funcionalidad de capa física (PHY), mientras que LoRaWAN es un protocolo MAC para una red en estrella de gran capacidad y largo alcance que la LoRa Alliance ha estandarizado para las LPWAN [7]. LoRaWAN define el propio protocolo MAC y la arquitectura del sistema de la red, mientras que LoRa permite el enlace de comunicación de largo alcance a nivel físico. El protocolo y la arquitectura de la red de LoRaWAN determinan de forma decisiva la vida útil de la batería de un nodo, la capacidad de red, la calidad del servicio, la seguridad, y la variedad de aplicaciones de la red. A diferencia de LoRaWAN, LoRa no dispone de reglas o normas más allá de las limitaciones de capa física que ofrecen los dispositivos o las impuestas por la librería utilizada. LoRaWAN puede emplear LoRa o una modulación FSK a nivel físico.

El protocolo LoRaWAN está optimizado para sensores de bajo coste operados con batería e incluye diferentes clases de nodos para optimizar la compensación entre la latencia de la red y la duración de la batería. Es totalmente bidireccional y fue diseñado para garantizar la fiabilidad y la seguridad. La arquitectura de LoRaWAN también fue diseñada para localizar y

rastrear fácilmente objetos móviles. LoRaWAN está siendo desplegado para redes nacionales por los principales operadores de telecomunicaciones, y la LoRa Alliance está estandarizando LoRaWAN para asegurar que las diferentes redes nacionales sean interoperables [5].

1.2.1 Red LoRaWAN

Generalmente, las redes LoRaWAN se extienden en una topología de estrella-de-estrellas donde los gateways² retransmiten mensajes entre nodos³ y un servidor de red ubicado en el back-end. Los gateways se conectan al servidor mediante una conexión IP estándar mientras que los nodos y el gateway lo hacen mediante un enlace directo empleando LoRa o FSK a nivel físico. La comunicación puede ser unidireccional o bidireccional, aunque, en este último caso, el tráfico ascendente de un nodo al gateway y de éste al servidor es el predominante.

Los dispositivos LoRaWAN pueden operar de dos maneras: en modo P2P como en las redes LoRa, y en modo híbrido. En modo híbrido (ver **Fig. 1.3**), se utiliza una combinación de los modos LoRaWAN y P2P que permite enviar mensajes utilizando redes LoRaWAN. Este modo requiere licencia LoRaWAN. En este caso se utiliza un nodo central actuando como gateway trabajando en modo híbrido y el resto de nodos como P2P. Los nodos pueden utilizar una topología de estrella P2P para llegar al nodo central y hacer que este acceda a la red LoRaWAN para encaminar la información. La base de esta operación es que el gateway escucha paquetes P2P y los reenvía a la infraestructura LoRaWAN, tal y como se muestra en la siguiente imagen:



Fig. 1.3 Esquema modo híbrido (sólo LoRaWAN) [5]

² Un gateway es un dispositivo que permite interconectar redes con protocolos y arquitecturas diferentes. Su propósito es traducir la información del protocolo utilizado en una red al protocolo usado en la red de destino. Más información en el capítulo 2.

³ Un nodo es un dispositivo con la capacidad de procesar, reunir información gracias a un sensor y comunicarse con otros nodos o gateways de la red.

1.2.2 Modos de acceso a una red LoRaWAN

Hay dos métodos para que un módulo⁴ pase a formar parte de una red LoRaWAN.

- *Activation by Personalization (ABP)*: Las claves *Network Session Key* y *Application Session Key*, y la dirección del nodo (*DevAddr*) son conocidas tanto por el nodo como por el servidor. De este modo la transmisión de paquetes empieza desde el inicio.
- *Over-the-Air Activation (OTAA)*: El nodo y el servidor de red negocian las claves de cifrado en el momento que el nodo se une a la red. Para ello es necesario que el nodo envíe el identificador del dispositivo (*Device EUI*), el identificador de la aplicación (*Application EUI*), y la clave *Application Key* al servidor. Luego el servidor le envía la dirección del dispositivo y las claves *Network Session Key* y *Application Session Key*. En este modo, es necesario todo este procedimiento antes de que comience la transmisión de paquetes.

1.2.3 Canales y velocidades de transmisión

La comunicación entre nodos y gateways se realiza utilizando diferentes canales de frecuencia y velocidades de datos. Haciendo uso de la técnica de espectro ensanchado, las comunicaciones con diferentes velocidades de datos no interfieren entre sí, creando un conjunto de canales virtuales que aumentan la capacidad del gateway. Para maximizar el uso de las baterías de los nodos y la capacidad de la red, LoRa puede gestionar la velocidad de datos y la salida de RF para cada nodo individualmente por medio de un esquema adaptativo de velocidad de datos (ADR).

Así pues, los nodos pueden transmitir por cualquier canal disponible en cualquier momento y utilizando cualquier tasa de datos disponible siempre y cuando cumplan las siguientes normas:

- El nodo cambia de canal de forma pseudo-aleatoria para cada transmisión. La diversidad de frecuencia resultante hace que el sistema sea más robusto a las interferencias.
- El nodo respeta el máximo ciclo de trabajo de transmisión en relación a la sub-banda utilizada y a las regulaciones locales.
- El nodo respeta la duración máxima de transmisión en relación a la sub-banda utilizada y a las regulaciones locales.

⁴ Dispositivos capaces de recopilar datos de un sensor y transmitirlos al gateway. Más información en el capítulo 2.1.

Los módulos LoRaWAN disponen de 16 canales en las bandas de 433 MHz y 868 MHz (ver Tabla 1.1), y de 72 canales en la banda de 900 MHz. En Europa se utilizan las bandas de 433 MHz y 868 MHz y en este proyecto se trabaja sobre la banda de 868 MHz, la cual dispone de los siguientes parámetros para LoRaWAN:

Tabla 1.1 Canales de frecuencias LoRaWAN para las bandas de 868 MHz y 433 MHz [5]. Para más información sobre el Data Rate vease la **Tabla 1.2**

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

*Los primeros tres canales tienen frecuencias asignadas. El resto de canales se pueden configurar dentro del rango (868325 kHz - 869750 kHz) con un ancho de banda de 125 kHz para la banda de 868 MHz y (433050 kHz - 434790 kHz) para la banda de 433 MHz (bandas ISM Europeas). LoRaWAN puede usar la banda de frecuencias de 433 MHz siempre que el PIRE de los dispositivos sea inferior a 10mW (10dBm) y, al igual que en la banda de 868 MHz, el ciclo de trabajo sea inferior al 1%. La banda ISM de 779 MHz (China) dispone del mismo número de canales pero con distintas frecuencias centrales. En cambio la banda ISM de 915 MHz (US) cuenta con otro número de canales. La banda de 915 MHz está dividida en 64 canales ascendentes numerados del 0 al 63 que utilizan LoRa con un ancho de banda de 125 kHz empezando en la frecuencia 902.3 MHz e incrementando cada 200 kHz hasta 914.9 MHz, otros 8 canales ascendentes numerados del 64 al 71 utilizando LoRa con un ancho de banda de 500 kHz empezando en la frecuencia 902.3 MHz e incrementando cada 1.6 MHz hasta 914.9 MHz, y 8 canales descendentes numerados del 0 al 7 utilizando LoRa con un ancho de banda de 500 kHz empezando en la frecuencia 923.3 MHz e incrementando cada 600 kHz hasta 927.5 MHz.

** Hay limitaciones que establecen cuánto tiempo puede estar el transmisor activado o el tiempo máximo que puede estar transmitiendo. LoRaWAN impone una limitación del ciclo de trabajo por cada sub-banda **(1.1)**. Cada vez que una trama se transmite en una sub-banda dada, el tiempo de la emisión y la duración en el aire de la trama se registran para esta sub-banda. La misma sub-banda no puede ser utilizada de nuevo durante los próximos T_{off} segundos, donde:

$$T_{off} = \frac{TimeOnAir}{DutyCycle_{subband}} - TimeOnAir \quad (1.1)$$

*** La velocidad de datos (Data Rate) acepta valores del 0 al 5 para la banda de 868 MHz y del 0 al 7 para la banda de 433 MHz.

Tabla 1.2 Data Rates en LoRaWAN

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

El SF corresponde a una configuración de Spreading Factor (SF), que determina la cantidad de datos redundantes que se envían en la transmisión. La velocidad está relacionada con el SF de manera que si éste es alto se enviarán muchos datos redundantes y por tanto se logrará un máximo alcance con baja velocidad.

**** Los canales del 3 al 15 están en estado *Off* por defecto. Al activar nuevos canales, el administrador de la red deberá adaptar el ciclo de trabajo del resto de canales de manera que el total de canales no sobrepase el 1% máximo permitido en la banda de frecuencias. Además, a diferencia de los canales por defecto, los canales del 3 al 15 pueden ser configurados con los data rates 6 y 7 siempre y cuando sean compatibles con el módulo LoRaWAN utilizado.

1.2.4 Clases LoRaWAN

Existen tres clases de dispositivos LoRaWAN, denominadas Clase A, Clase B y Clase C (ver **Fig. 1.4**). Todos los dispositivos LoRaWAN implementan al menos la funcionalidad de clase A. Además, pueden implementar las opciones de Clase B o Clase C.

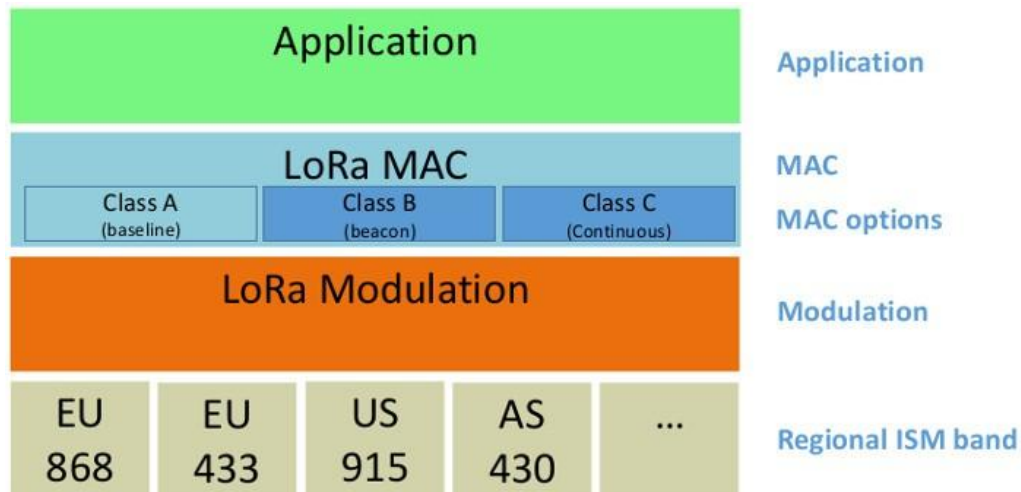


Fig. 1.4 Clases de dispositivos LoRaWAN [5]

Clase A. Los dispositivos de Clase A permiten una comunicación bidireccional, con la limitación de que sólo pueden recibir datos (canal *downlink*) si han enviado antes un paquete (canal *uplink*). Eso es porque cada vez que el dispositivo envía un paquete, se abren dos ventanas de recepción, la primera un segundo después de la transmisión y la segunda un segundo después de la primera ventana, con la oportunidad de recibir un paquete de vuelta. Este paquete de vuelta contiene el ACK del paquete enviado así como datos de la aplicación si es necesario.

Este tipo de dispositivos serán los de menor consumo energético de la especificación, puesto que estarán en modo dormido por defecto e iniciarán toda comunicación. Servirán para aplicaciones en las que los dispositivos no deban recibir datos habitualmente. Un dispositivo compatible con LoRaWAN debe implementar siempre esta clase básica.

Clase B. Los dispositivos de clase B añaden la capacidad de recibir datos (*downlink*) sin necesidad de enviar un paquete (*uplink*), de esta forma la aplicación puede enviar datos a los dispositivos de forma programada.

Esto se consigue mediante el envío periódico de *beacons* por parte del gateway. Estos *beacons* permiten a los dispositivos estar sincronizados con el gateway, y de esta forma pueden negociar tiempos de recepción de paquetes desde el gateway al dispositivo (*downlink*). Así, las ventanas de recepción se abren en momentos determinados. Esta clase de dispositivos tendrán un

consumo mayor de energía que los de clase A debido a la recepción periódica de los *beacons* desde el gateway.

Clase C. Los dispositivos de clase C están permanentemente escuchando (es decir, en modo de recepción), y por tanto pueden recibir datos (*downlink*) en cualquier momento (excepto cuando estén enviando datos (*uplink*)). En otras palabras, las ventanas de recepción se mantienen siempre abiertas excepto cuando transmiten.

Esta clase proporciona los mejores tiempos de respuesta y capacidad de envío desde el servidor a los dispositivos, a costa de un consumo energético mucho mayor respecto a las clases A y B.

1.2.5 Seguridad en LoRaWAN

En algunos escenarios la seguridad es indispensable y necesaria cuando se tratan con datos de toda una ciudad, los cuales pueden contener información que requiere ser protegida. Por ese motivo LoRaWAN incorpora varias capas de cifrado, con sus correspondientes claves, que hacen uso del algoritmo de cifrado AES128 para proteger las comunicaciones:

- *Network Session Key*: Clave de 128 bits que garantiza la seguridad a nivel de red.
- *Application Session Key*: Clave de 128 bits que garantiza la seguridad extremo a extremo a nivel de aplicación.
- *Application Key*: Clave de 128 bits que garantiza la seguridad extremo a extremo a nivel de aplicación, utilizada únicamente en OTAA.

LoRaWAN, al tratarse de un protocolo estandarizado, dispone de características establecidas en su especificación, permitiendo la interoperabilidad entre los dispositivos que implementen dicha tecnología, indiferentemente de las librerías externas utilizadas. A continuación se presentan la potencia de transmisión de los módulos RN2483, la estructura de los paquetes LoRaWAN y la comunicación entre nodos en una red LoRaWAN.

1.3 Características de una red LoRaWAN

LoRaWAN, al tratarse de un protocolo estandarizado, dispone de características establecidas en su especificación, permitiendo la interoperabilidad entre los dispositivos que implementen dicha tecnología, indiferentemente de las librerías externas utilizadas. A continuación, a modo de ejemplo, se presentan la potencia de transmisión de los módulos RN2483, los utilizados en este proyecto (ver **Fig.2.6**), la estructura de los paquetes LoRaWAN y la comunicación entre nodos en una red LoRaWAN.

1.3.1 Potencia de transmisión

El módulo RN2483 permite variar el nivel de potencia a la salida del transceptor. Los niveles de potencia que se pueden configurar en el módulo RN2483 son los mostrados en la **Tabla 1.3**.

Tabla 1.3 Niveles de potencia de salida del módulo RN2483 de LoRaWAN [8]

Parámetro	Potencia de salida (dBm)	Corriente a 3V (mA)
-3	-4	17,3
-2	-2,9	18
-1	-1,9	18,7
0	-1,7	20,2
1	-0,6	21,2
2	0,4	22,3
3	1,4	23,5
4	2,5	24,7
5	3,6	26,1
6	4,7	27,5

Parámetro	Potencia de salida (dBm)	Corriente a 3V (mA)
7	5,8	28,8
8	6,9	30
9	8,1	31,2
10	9,3	32,4
11	10,4	33,7
12	11,6	35,1
13	12,5	36,5
14	13,5	38
15	14,1	38,9

1.3.2 Estructura de paquetes LoRaWAN

Los paquetes LoRaWAN siguen una estructura definida en las especificaciones del protocolo LoRaWAN. Todos los mensajes LoRaWAN (ver **Tabla 1.4**) cuentan con un campo *preamble* de 8 bytes de longitud, una cabecera (PHDR) y el *Payload*, los dos últimos con su CRC⁵ [7].

Tabla 1.4 Capa Radio PHY

preamble	PHDR	PHDR_CRC	PHYPayload	CRC*
----------	------	----------	------------	------

*solo mensajes uplink

El Payload de capa física cuenta con una cabecera MAC, el *MAC Payload* y un Message Integrity Code, un código de cuatro bytes que se calcula a partir de la *Network session key (NwkSKey)* (ver **Tabla 1.5**).

⁵ Close-Range Correction: Código de detección de errores.

Tabla 1.5 PHY Payload

MHDR	MACPayload	MIC ⁶
1 byte	1-M bytes	4 bytes

La cabecera MAC especifica el tipo de mensaje y la versión del formato de la trama de la especificación de la capa LoRaWAN con la que ha sido codificada. Existen seis tipos de mensajes MAC (**Tabla 1.6**).

Tabla 1.6 Tipos de mensaje MAC

MType	Descripción
000	Join Request
001	Join Accept
010	Unconfirmed Data Up
011	Unconfirmed Data Down
100	Confirmed Data Up
101	Confirmed Data Down
110	RFU ⁷
111	Proprietary

El *MACPayload* incluye una cabecera de trama, un campo de puerto opcional y un campo de *Payload* de trama opcional (**Tabla 1.7**).

Tabla 1.7 MAC Payload

FHDR	FPort	FRMPayload
7-23 bytes	0-1 bytes	0-N bytes

La cabecera de trama contiene la dirección con la que se identifica el dispositivo dentro de la red, un campo *FCtrl* para habilitar el *Adaptive data rate*, un contador de tramas y un campo *FOpts* en el caso de que se desee transmitir un comando MAC. El campo *FPort* sirve para determinar si el campo *FRMPayload* contiene comandos MAC o datos de la aplicación (**Tabla 1.8**).

⁶ El Message Integrity Code (MIC) es una clave para autenticar el mensaje generada a partir de los campos MHDR y MACPayload.

⁷ Reserved for Future Usage

Tabla 1.8 Esquema FHDR

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

Los siete bits más significativos del campo *DevAddr* se utilizan para el identificador de red (*NwkID*) mientras que los veinticinco restantes corresponden a la dirección de red (*NwkAddr*), la cual puede ser asignada por el administrador de la red.

El tamaño máximo del campo *MACPayload* varía según la banda de frecuencia sobre la que se trabaja, el data rate y la ausencia del campo de control (*FOpts*), tal y como se puede ver en la **Tabla 1.9**:

Tabla 1.9 Tamaño máximo de las tramas en la banda de 868 MHz

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

1.3.3 Comunicación entre nodos

A diferencia de lo que ocurre en las redes LoRa donde el único intercambio de mensajes se produce entre los nodos y el gateway, con LoRaWAN el servidor de red se comunica con los sensores a través del gateway.

Aunque la comunicación entre módulos y gateway se efectúe de la misma forma que en las redes LoRa, la comunicación es distinta al otro lado del gateway ya que éste se encarga de convertir los paquetes LoRaWAN en paquetes UDP y viceversa. Además, están los mensajes que el gateway recibe desde los sensores y retransmite hasta el servidor de red (*Upstream*) y los mensajes de configuración de la red que envía el servidor de red hasta los sensores (*Downstream*).

En la **Fig. 1.5** vemos que el nodo puede transmitir tantos mensajes seguidos como números de canales tiene habilitados (*N*). Cuando el nodo transmite un mensaje con confirmación, este llega al gateway, que convierte el mensaje en un paquete UDP y lo reenvía al servidor de red. Aunque el mensaje no requiera confirmación, el gateway sí recibe respuesta del servidor de red. Una vez ha

transmitido por el último canal disponible, el nodo no podrá volver a transmitir hasta pasado el ciclo de trabajo de cada canal.

Es posible transmitir mensajes con confirmación, haciendo que el gateway envíe un ACK al nodo después de recibirlo por parte del servidor de red.

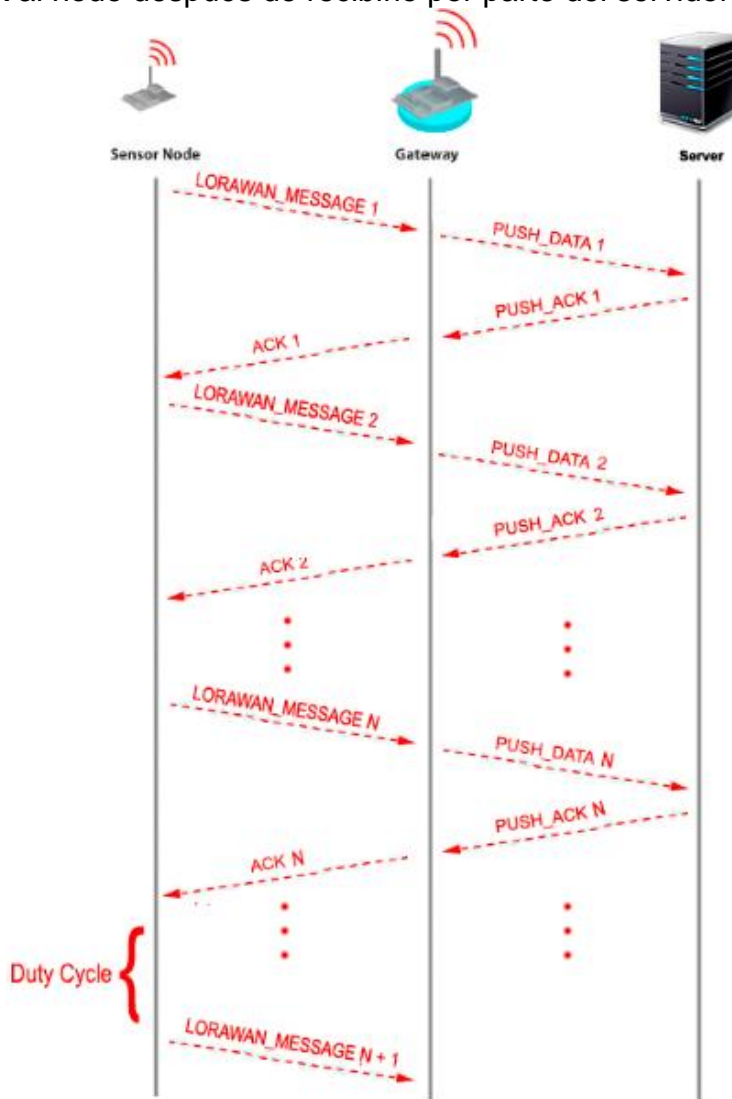


Fig. 1.5 Comunicación *Upstream* Nodo/Gateway/Servidor con confirmación

Los paquetes PUSH_DATA incluyen la información transmitida por los sensores hacia el gateway, un *token* generado aleatoriamente y la dirección MAC del gateway. El servidor devolverá un PUSH_ACK con el mismo *token* del PUSH_DATA y procesará la información proveniente del gateway. El servidor procesa los paquetes después de haber enviado el ACK correspondiente.

Por otro lado, cuando el servidor de red quiere enviar mensajes al gateway, la comunicación es un tanto peculiar (ver **Fig. 1.6**). A pesar de tratarse de la comunicación *downstream*, quien la inicia es el gateway porque lo contrario no sería posible si el gateway se encontrara detrás de un NAT. El gateway envía

paquetes *PULL_DATA* periódicamente para que el NAT quede siempre abierto y el servidor pueda enviar mensajes *PULL_RESP* a través del mismo puerto que los mensajes *PULL_DATA* dejan abierto.

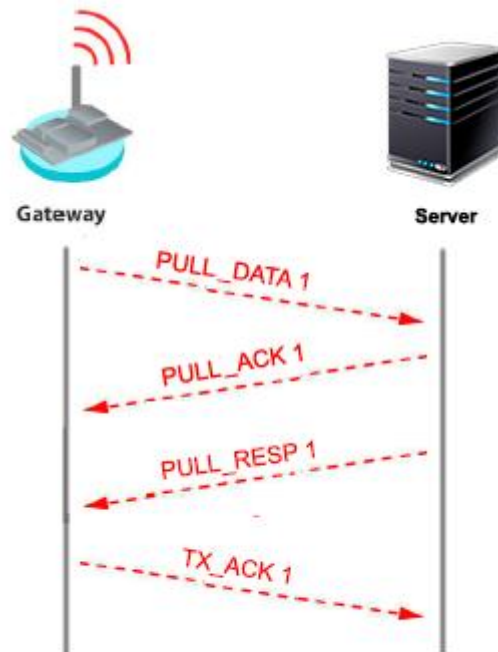


Fig. 1.6 Comunicación *Downstream* Servidor/Gateway

CAPÍTULO 2. ENTORNO DE TRABAJO

En este capítulo se describe el material empleado para utilizar las tecnologías expuestas en el capítulo anterior, mediante el cual se realizarán los montajes de los distintos escenarios de estudio que recogerá este documento.

2.1 Dispositivos LoRa

A la hora de desplegar una red LoRa, será necesario disponer de algunos dispositivos imprescindibles para cualquier red de sensores:

- Módulos LoRa (**Fig. 2.1**): Capaces de recoger los datos del sensor y transmitirlos al gateway. Este proyecto está pensado para crear una red de sensores LoRa indistintamente de la clase de sensor que se quiera utilizar. Cada módulo estará formado por un transceptor SX1272, un Wasp mote, una antena LoRa y una batería, además del sensor que se quiera utilizar.



Fig. 2.1 Nodo sensor LoRa [6]

- Gateway LoRa: El gateway LoRa no es otra cosa que un módulo LoRa con un transceptor SX1272, una placa Wasp mote Gateway y una antena LoRa con un adaptador USB (**Fig. 2.2**). Recibirá los datos de todos los módulos LoRa de la red de sensores. El adaptador USB permite conectar el módulo a una Raspberry Pi 3 gracias a la cual se podrán

consultar los datos obtenidos por los sensores desde cualquier PC con acceso mediante una sesión ssh.



Fig. 2.2 Gateway LoRa

A continuación se describen los distintos componentes de un módulo LoRaWAN.

2.1.1 Transceptor LoRa

El transceptor SX1272 (**Fig. 2.3**) es el corazón de un módulo LoRa. Este transceptor fabricado por Semtech proporciona una comunicación de largo alcance y alta inmunidad a interferencias a la vez que minimiza el consumo de energía.



Fig. 2.3 Transceptor LoRa SX1272 [9]

A continuación se detallarán sus principales características:

- Banda de frecuencia de 863 - 870 MHz
- Largo alcance: hasta 22 km.
- Link Budget máximo de 157 dBm.
- Salida RF constante de +20 dBm en 100 mW en comparación con la alimentación.
- PA de alta eficiencia de +14 dBm.
- Alta sensibilidad: hasta -137 dBm.
- Bajo consumo en recepción: 10 mA.
- Soporta varios tipos de modulación: FSK, GFSK, MSK, GMKS, LoRa y OOK.
- Detección de preámbulo.
- RSSI de rango dinámico.
- Sensor de temperatura integrado e indicador de batería baja.

2.1.2 Wasmote

Wasmote es una plataforma modular opensource diseñada por Libelium para construir redes de sensores inalámbricas de muy bajo consumo.



Fig. 2.4 Wasmote fabricado por Semtech

La placa Wasmote (**Fig 2.4**) incorpora un microcontrolador, memoria y sockets para añadir módulos. Utiliza el mismo entorno de desarrollo que Arduino, utilizando el mismo código para ambas plataformas con algunos ligeros cambios como el pinout y el esquema de E/S.

2.1.3 Batería

Los Waspmites recibirán alimentación a través de unas baterías de litio-ion recargables con un voltaje de 3.7 Voltios y capacidad nominal de 6600 mAh (Fig 2.5).



Fig. 2.5 Baterías utilizadas en los módulos LoRa

2.1.4 Antena

La antena utilizada en el módulo LoRa es un modelo especial de 4.5 dBi de ganancia.

2.2 Dispositivos LoRaWAN

Para los escenarios que son objeto de este proyecto, se desplegará una red LoRaWAN, de modo que se dispondrá de las ventajas de LoRaWAN, además de las características de LoRa.

- Módulos LoRaWAN: además de permitir la comunicación a nivel físico, estos dispositivos incorporarán a la red LoRa los beneficios que aporta el protocolo LoRaWAN.
- Gateway LoRaWAN: recibirá los datos de los módulos LoRaWAN de la red y los enviará al servidor de red.

2.2.1 Módulos LoRaWAN

Se escogieron los módulos RN2483-PICtail de Microchip (**Fig 2.6**), una placa preprogramada para proporcionar un puente serie USB-to-UART que permite una conexión en serie.



Fig. 2.6 Módulo LoRaWAN modelo RN2483-PICtail [10]

El módulo RN2483 [8] puede conectarse a una placa computadora como Arduino o directamente a un PC por USB. Ambas opciones son válidas para alimentar el módulo aunque la opción por USB permite al usuario comunicarse con la placa mediante la interfaz de comandos ASCII del RN2483 [11].

Algunas de sus características principales son:

- Cobertura de hasta 15 km en áreas suburbanas y hasta 5 km en áreas urbanas.
- Bajo consumo.
- Opera en dos bandas de frecuencia: 433 MHz y 868 MHz.
- Velocidad de bits en comunicaciones RF programable hasta 300 kbps con modulación FSK y hasta 11 kbps con modulación LoRa.
- Alta sensibilidad en recepción: hasta -146 dBm.
- PA de alta eficiencia de +14 dBm.
- Soporta varios tipos de modulación: FSK, GFSK y LoRa.

2.2.2 Gateway LoRaWAN

Una red LoRaWAN requiere de un gateway y un servidor de red. El modelo de gateway utilizado para hacer las pruebas es *Kerlink LoRa IoT Station* (**Fig. 2.7**).



Fig. 2.7 Gateway LoRaWAN Kerlink LoRa IoT Station [12]

El gateway Kerlink LoRa IoT Station es un gateway LoRaWAN capaz de operar en las bandas de 868 MHz y 915 MHz. Cuenta con protección IP67 para instalar en el exterior y ocho canales en los que operar con LoRaWAN.

El gateway cuenta con un sistema operativo Linux que le permite ejecutar programas y scripts con los que convertir los paquetes LoRaWAN en paquetes IP/UDP, y así transmitirlos al servidor de red LoRaWAN.

El programa llamado *packet forwarder* [13] se ejecuta en el Host del gateway LoRa y se encarga de retransmitir los paquetes recibidos por el concentrador en paquetes IP/UDP para transmitirlos al servidor de red vía Internet, a la vez que controla todo el tráfico entrante y saliente (ver **Fig. 2.8**). El concentrador es una radio LoRa multicanal basada en los transceptores de Semtech SX130x, SX135x y SX127x. En el host se ejecuta el reenvío de paquetes a través de un enlace SPI. El gateway debe disponer como mínimo de un concentrador, un host y conexión a internet o a una red privada. El servidor procesará los paquetes reenviados por el gateway y enviará otros a los distintos sensores conectados al gateway a través de él.

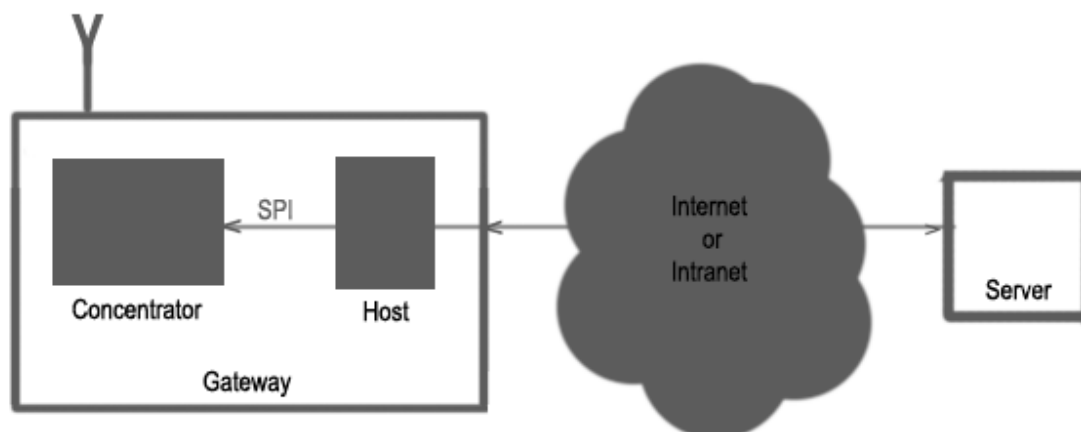


Fig. 2.8 Esquema Gateway LoRaWAN

2.3 Software

Tanto para programar los dispositivos (módulos LoRa/LoRaWAN, gateways y los servidores de red) como para diseñar la red, es necesario disponer del software necesario, desde programar los módulos LoRa/LoRaWAN hasta las librerías necesarias para el funcionamiento de los módulos LoRa.

2.3.1 Wasmote PRO

El transceptor SX1272 requiere de programación para ser configurado. Esta configuración puede efectuarse mediante Arduino, Wasmote o Raspberry Pi. En nuestro caso utilizaremos un Wasmote para cada módulo LoRa, por lo que será necesario el programa Wasmote PRO de Libelium (ver **Fig 2.9**) para configurar estos módulos.

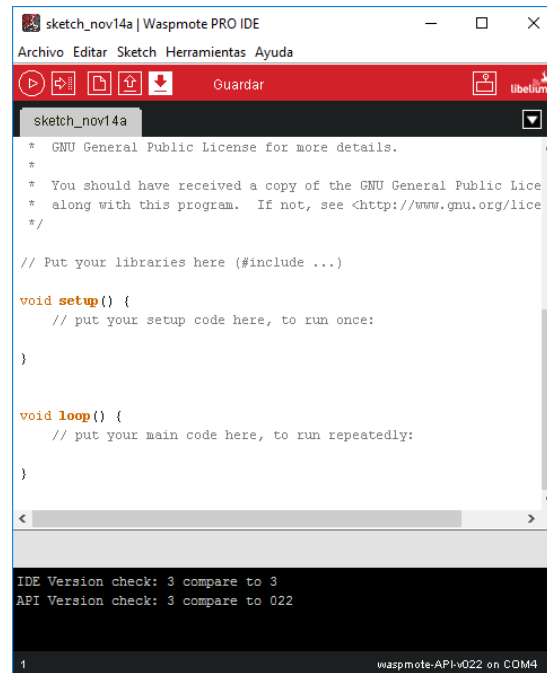


Fig. 2.9 Wasmote PRO [14]

Wasmote PRO ofrece una interfaz muy similar a la de Arduino. La programación de los módulos se divide en dos partes. Una primera parte para configurar los parámetros clave del módulo como la dirección, el modo de funcionamiento, la potencia de transmisión; y una segunda parte para añadir las funciones que queremos que ejecute dicho módulo, como transmitir periódicamente los datos recolectados por los sensores.

Para programar los módulos SX1272 estos deben encontrarse incorporados en la placa Wasmote. La librería de Libelium deberá estar importada en Wasmote PRO y el Wasmote conectado al PC por un cable USB. Cuando el programa reconozca la tarjeta Wasmote correspondiente a la versión de la librería y el puerto serie, podrá proceder a la configuración.

2.3.2 Raspbian

Raspbian (**Fig. 2.10**) es una distribución de Linux basada en Debian optimizada para las placas Raspberry Pi.

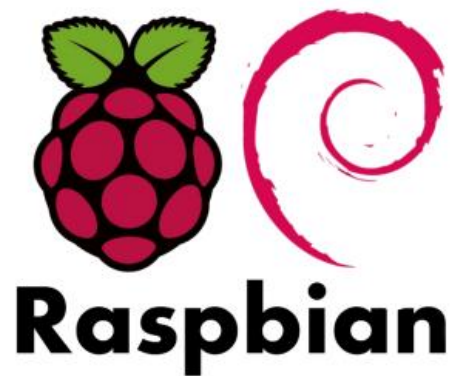


Fig. 2.10 Logo Raspbian SO [15]

Se utilizará Raspbian como sistema operativo para los gateways LoRa y LoRaWAN, permitiendo así una configuración más personalizada y la capacidad de incluir nuevas funcionalidades a través de scripts de Python en los mismos.

2.3.3 Python

El gateway LoRa requiere de un PC con un puerto USB. En este proyecto, como ya se ha comentado, se utilizó una Raspberry Pi 3. Ahora bien, el gateway únicamente recibiría los mensajes sin poder procesarlos o almacenarlos. Para gestionar los datos obtenidos por los sensores se creó un script de Python (**Fig 2.11**) que permite registrar cada mensaje que llegue al gateway en la memoria interna de la Raspberry Pi, proporcionando al administrador de la red la posibilidad de visualizarlos, y mandando una copia a una cuenta de Google Drive creada para esta red.



Fig. 2.11 Logo Python [16]

Puede verse más detalladamente la configuración del gateway en el Anexo I al final de este documento.

2.3.4 Cutecom

El gateway LoRa y los transceptores LoRaWAN requieren un programa de comunicaciones de puerto serie para recibir una configuración. Cutecom es un emulador de terminal que permite establecer comunicaciones serial [17]. Además, estos programas permiten monitorizar el tráfico entrante o saliente, por lo que pueden utilizarse para controlar el tráfico de la red desde el gateway.

Se pueden utilizar varias alternativas a Cutecom, como Minicom, Hyperterminal, Docklight, etc.

Puede verse más detalladamente la configuración del gateway y de los módulos LoRaWAN en los Anexos III y IV al final de este documento.

2.4 Librería Wasmote API

Para configurar los módulos LoRa es necesario disponer de una librería. En este proyecto se utilizaron Wasmotes y la librería de Libelium. En la realización de este proyecto se utilizó la versión v025 de la librería de Libelium [18].

Hay que recordar que LoRa es un tipo de modulación y no un protocolo propiamente, de modo que la mayoría de aspectos descritos en este capítulo no vienen definidos únicamente por el transceptor utilizado, sino también por la propia librería de Libelium que permite establecer los parámetros necesarios para desplegar una red LoRa.

A continuación se detallarán las características que posee la red LoRa desplegada en el Campus del Baix Llobretat, Campus en el que se encuentra la Escola d'Enginyeria de Telecomunicació i Aeroespacial de Castelldefels y sobre las que se han hecho las pruebas descritas en el capítulo 3.

2.4.1 Potencia de transmisión

La librería LoRa de Libelium permite configurar los módulos LoRa con distintos niveles de potencia de salida utilizando tres parámetros diferentes. Estos parámetros se corresponden a Low (L), High (H) y Max (M) (ver **Tabla 2.1**).

Tabla 2.1 Niveles de potencia de salida del módulo SX1272 de LoRa

Parámetro	Potencia de salida
L	0 dBm
H	7 dBm
M	14 dBm

2.4.2 Modos de transmisión

A la hora de transmitir, la modulación LoRa tiene 3 parámetros configurables: BW, coding rate y spreading factor.

La combinación de estos valores define el modo de transmisión. Es posible establecer un modo predefinido o configurar estos tres parámetros manualmente.

Hay diez modos predefinidos en la librería de Libelium (ver **Tabla 2.2**), incluyendo el modo que permite alcanzar la máxima distancia posible (modo 1) y el modo más rápido (modo 10). Estos modos pueden modificarse a través de la librería.

Tabla 2.2 Modos LoRa (Librería Libelium)

Modo	BW (kHz)	CR	SF	Sensibilidad (dB)
1	125	4/5	12	-134
2	250	4/5	12	-131
3	125	4/5	10	-129
4	500	4/5	12	-128
5	250	4/5	10	-126
6	500	4/5	11	-125.5
7	250	4/5	9	-123
8	500	4/5	9	-120
9	500	4/5	8	-117
10	500	4/5	7	-114

2.4.3 Estructura de paquetes LoRa

Los paquetes de la librería de Libelium utilizan una estructura llamada *pack* (ver **Tabla 2.3**). Esta estructura se compone de los siguientes parámetros:

Tabla 2.3 Esquema de una trama LoRa

dst	src	packnum	length	data	retry
1 byte	1 byte	1 byte	1 byte	Variable bytes	1 byte

- **dst:** Dirección destino. Este parámetro debe ser introducido por el usuario.
- **src:** Dirección origen. Este parámetro debe ser introducido por el usuario.
- **packnum:** Número del paquete. Este parámetro es establecido por la aplicación y se le asigna un valor a través de un contador. De esta manera el usuario puede ver si se pierde algún paquete.
- **length:** Longitud del paquete. Este parámetro es establecido por la aplicación.
- **data:** Es la información que se envía en el paquete. Esta información se encapsula y se envía a otros nodos o al Gateway. El tamaño máximo de este campo está definido en la librería como MAX_PAYLOAD, con valor de 250 bytes.
- **retry:** Contador de reintentos. Es una opción que permite reenviar un paquete si el nodo origen no recibe el ACK correspondiente a ese paquete. El valor de este campo es 0 por defecto. Para habilitar esta opción es obligatorio trabajar con ACKs.

Las direcciones serán únicas y de 8 bits. Las direcciones 0 y 1 están reservadas para comunicaciones broadcast y para el gateway, respectivamente, quedando disponibles las direcciones 2-255.

Los paquetes no pueden fragmentarse. Si algún mensaje supera la capacidad máxima, la librería no permitirá enviarlo.

La librería de Libelium para LoRa recorta algunos parámetros cuando superan cierta longitud. En el caso de añadir un nombre a un nodo, la librería permite añadir cualquier nombre sin restricciones pero cuando envía el paquete, este nombre aparece en el campo *data*, recortándolo a 16 caracteres (16 bytes).

Otro punto importante es que, aunque la longitud mínima de una trama LoRa sea de 5 bytes (campos de *dst*, *src*, *packnum*, *length* y *retry*), el mínimo permitido por la librería es de 27 bytes. Esto es así porque en el campo *data* agrega datos como la MAC del transceptor, el nombre o ID del módulo y el número de trama enviada. Estos datos suman 22 bytes a los 5 bytes mínimos destinados al resto de campos del paquete LoRa. Algo similar sucede con la longitud máxima de la trama. Aunque la longitud máxima sea de 255 bytes, la librería limita el parámetro *data* a 119 bytes sin contar los 22 bytes añadidos, recortando los bytes que le siguen y reservando ese espacio para datos específicos como la longitud de ID del módulo, entre otros.

2.4.4 Comunicación entre nodos

Los nodos SX1272 que se usaron para hacer las pruebas soportan dos tipos de transmisiones: unicast y broadcast. Hay varias opciones a nivel de fiabilidad para la comunicación entre dispositivos.

- Sin ACK: Los módulos LoRa, a diferencia de los LoRaWAN, no necesitan guardar información sobre las redes a las que pertenecen, por lo que no hay ningún intercambio de mensajes previo al envío de información entre ellos. La **Fig. 2.12** ilustra la comunicación entre dos nodos sin ACK.

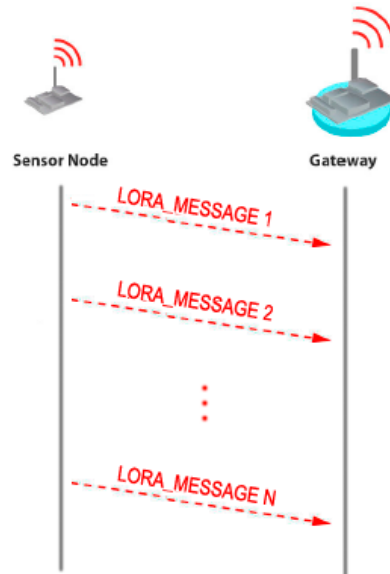


Fig. 2.12 Modo de transmisión LoRa sin ACK

- Con ACK: En el caso de habilitar el uso de ACKs, la comunicación sería la mostrada en la **Fig. 2.13**:

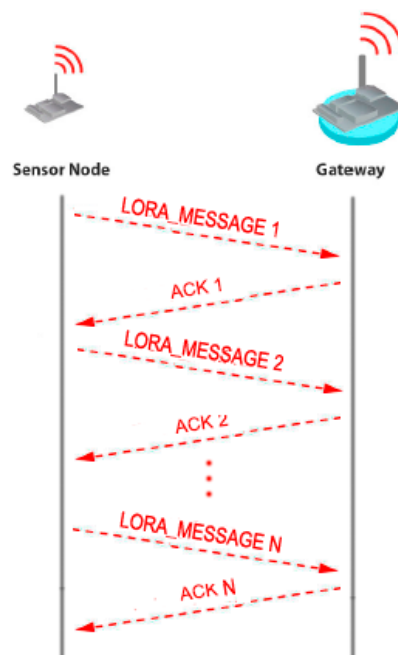


Fig. 2.13 Modo de transmisión LoRa con ACK

- Con ACK y retries: La opción de *retries* solo está disponible en caso de utilizar ACKs. Si el módulo transmisor no recibe el ACK, se volverá a enviar el paquete hasta un número configurado de veces o hasta que reciba el ACK, tal y como se muestra en la **Fig. 2.14**.

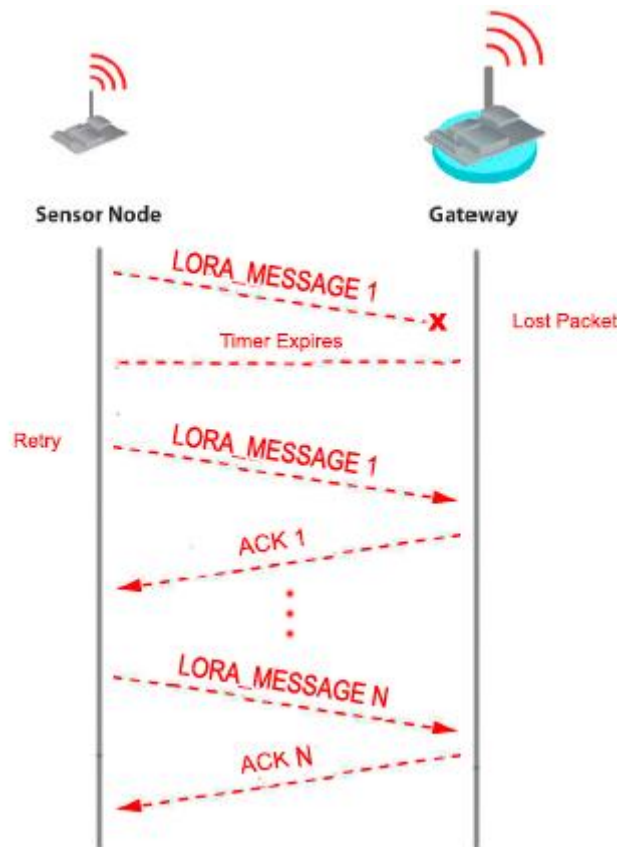


Fig. 2.14 Modo de transmisión LoRa con ACK y retries

Waspote PRO permite configurar el número máximo de retransmisiones, pero solo admite un máximo de 5 reintentos. Para comprobar el tiempo que lleva cada reintento se ha llevado a cabo una prueba en la que se han enviado tramas con cierto número de reintentos a un nodo que está apagado (ver **Tabla 2.4**). Los tiempos anotados corresponden al período desde que se envía la trama por primera vez hasta que el nodo desecha la trama y pasa a la siguiente.

Tabla 2.4 Tiempos de reintentos en LoRa

Reintentos	Tiempo
0	Tiempo de envío + 10 s
1	Tiempo de envío + 20 s
2	Tiempo de envío + 30 s
3	Tiempo de envío + 40 s
4	Tiempo de envío + 50 s
5	Tiempo de envío + 60 s

A falta de documentación oficial, se deduce que estos tiempos se componen de:

$$T_{total} = (T_{TX} \cdot n) + (10 \cdot n) \quad (2.1)$$

siendo T_{TX} el tiempo que el módulo LoRa tarda en transmitir un paquete, n el número de reintentos y T_{total} el tiempo total que tomaría desechar la trama. Las variables de tiempo están expresadas en segundos.

2.4.5 Seguridad

El módulo SX1272 no implementa ningún método de seguridad pero se puede añadir un cifrado a través de la librería Wasmote Encryption la cual utiliza un algoritmo AES con clave simétrica de 128, 192 o 256 bits [19]. Para poder implementar esta librería es necesario que el módulo LoRa opere sobre una placa Wasmote. Al no ser compatible con Arduino, Raspberry Pi o la placa sobre la que opera el gateway LoRa, solo es funcional en comunicaciones P2P entre módulos LoRa o entre módulos LoRa y el gateway Meshlium, el gateway de gama alta de Libelium.

Capítulo 3. Experimentos y resultados con LoRa/LoRaWAN

En los capítulos anteriores se habla sobre algunas características clave de LoRa y LoRaWAN. En este capítulo se realizarán algunos experimentos con el objetivo de verificar algunas de esas características. En el caso de LoRaWAN se realizará un estudio sobre el consumo de los módulos RN2483 y se efectuará un cálculo aproximado sobre el tiempo de vida de este módulo si se alimentara con una batería. A continuación se calculará el throughput en LoRa y LoRaWAN, mostrando una tabla comparativa con el throughput en cada modo de funcionamiento de cada tipo de módulo. En el mismo apartado se analizarán los diferentes estados por los que pasa un módulo LoRaWAN cuando transmite información. Para terminar, se verá el efecto que tienen algunos parámetros configurables como el canal utilizado, la potencia de transmisión o el modo de funcionamiento del transceptor, en la cobertura de los dispositivos LoRa.

3.1 Consumo en LoRaWAN

Uno de los parámetros de prestaciones más importantes de una tecnología de bajo consumo como LoRaWAN es el tiempo de vida de un nodo sensor que opera con batería. Para determinar dicho tiempo de vida, es necesario poder caracterizar el consumo de corriente del dispositivo. Para ello, es necesario poder caracterizar los diferentes estados por los que pasa un transceptor cuando transmite o recibe datos y cuando está inactivo. Por este motivo, se realizaron medidas de corriente del nodo sensor con LoRaWAN en distintas situaciones. Las pruebas fueron realizadas utilizando un analizador de potencia Agilent N6705A (**Fig. 3.1**).



Fig. 3.1 Analizador de potencia Agilent N6705A [20]

En la imagen que puede verse a continuación se observan los distintos estados de un transceptor al transmitir un mensaje *con un payload* de 5 bytes sin confirmación al gateway y transmitiendo con una potencia de 14 dBm (ver **Fig. 3.2**). El primer escalón corresponde a la transmisión, mientras que los otros dos se tratan de las dos ventanas de recepción descritas en el protocolo LoRaWAN. Dicha imagen se corresponde a un transceptor configurado para trabajar en modo Data Rate 5 (DR5), el modo más rápido que permite el protocolo LoRaWAN en los tres canales por defecto.

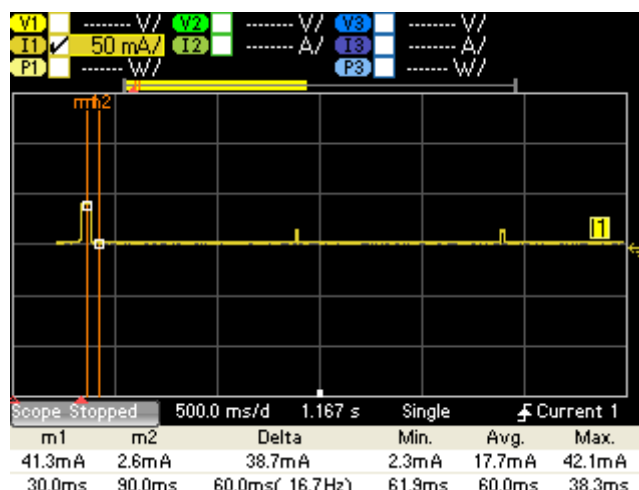


Fig. 3.2 Mensaje de 5 bytes desde un módulo LoRaWAN trabajando en modo DR5

El módulo, a pesar de haber enviado un mensaje que no requiere confirmación, abre las dos ventanas de recepción.

Un dato destacable de la Fig. 4.2 es el consumo de 2.6 mA cuando el transceptor no transmite ni recibe. Este consumo resulta elevado tratándose de un módulo que no hace nada pero en la práctica. Cuando el transceptor trabaje sobre un módulo completo, como por ejemplo sobre una Raspberry Pi, el consumo en modo sleep debería ser del orden de $140 \mu\text{A}$ ⁸.

Podemos comparar la imagen anterior con otra donde el transceptor trabaja en DR0, el más lento, y enviando esta vez un mensaje *de 5 bytes* con confirmación (**Fig. 3.3**).

⁸ El consumo puede variar dependiendo de la versión de firmware utilizada en los módulos LoRaWAN. Microchip, el fabricante del chip RN2483, informó en una hoja de erratas que el consumo real es de unos $140 \mu\text{A}$ y no el que viene el en datasheet: <http://ww1.microchip.com/downloads/en/DeviceDoc/80000689A.pdf>

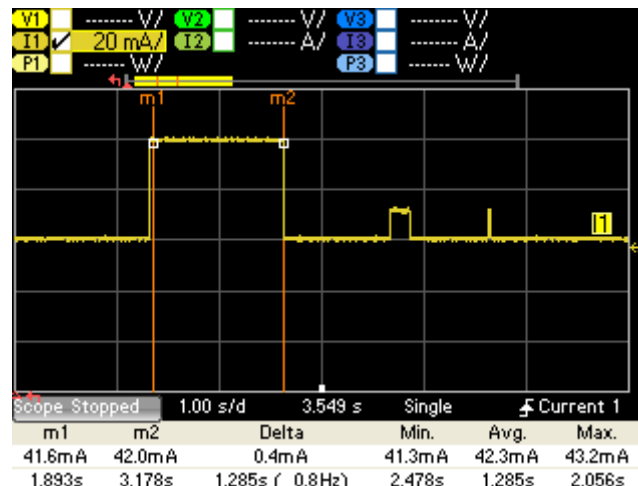


Fig. 3.3 Mensaje de 5 bytes desde un módulo LoRaWAN trabajando en modo DR0

Puede apreciarse como la transmisión dura más y la primera ventana, la cual es significativamente más ancha que la segunda, recibe el ACK correspondiente del gateway indicando que ha recibido el paquete. A pesar de recibir el ACK en la primera ventana de recepción, el módulo abre igualmente la segunda.

A continuación, comprobamos la duración y consumo de cada una de estas transmisiones (**Fig. 3.4** y **Fig. 3.5**).

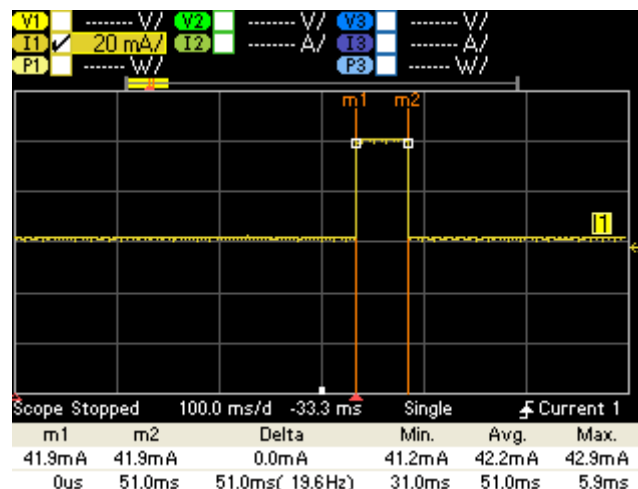


Fig. 3.4 Consumo medio de una transmisión DR5

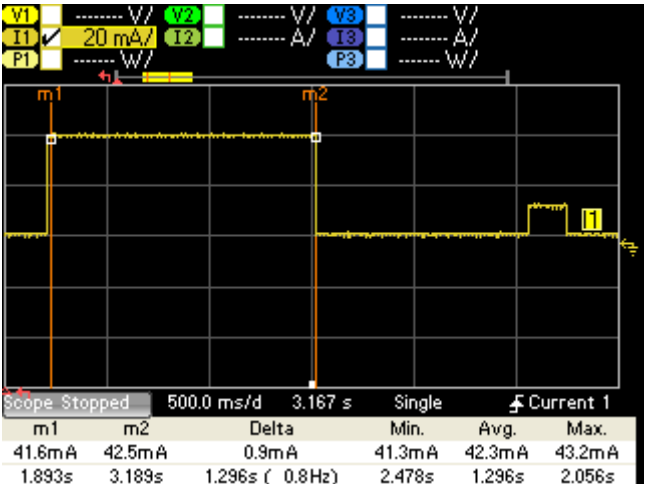


Fig. 3.5 Consumo medio de una transmisión DR0

Mientras que en modo DR5 el nodo sensor transmite durante 51 ms consumiendo 42,3 mA, en modo DR0 lo hace en 1.296 segundos, lo cual supone un mayor consumo de energía. Si recordamos la Tabla 3.11 con los niveles de señal de potencia de transmisión podemos comprobar que transmitiendo a 14 dBm deberíamos ver un consumo de 38.9 mA en lugar de los 42.3 mA que muestran las dos figuras anteriores.

A continuación caracterizamos los estados correspondientes a las ventanas de recepción dependiendo de si el nodo sensor ha recibido (Fig. 3.6) o no un ACK (Fig. 3.7).

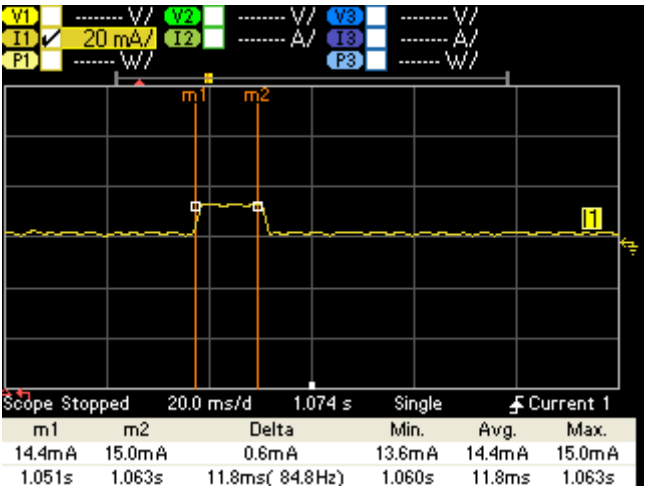


Fig. 3.6 Primera ventana de recepción sin ACK en modo DR5

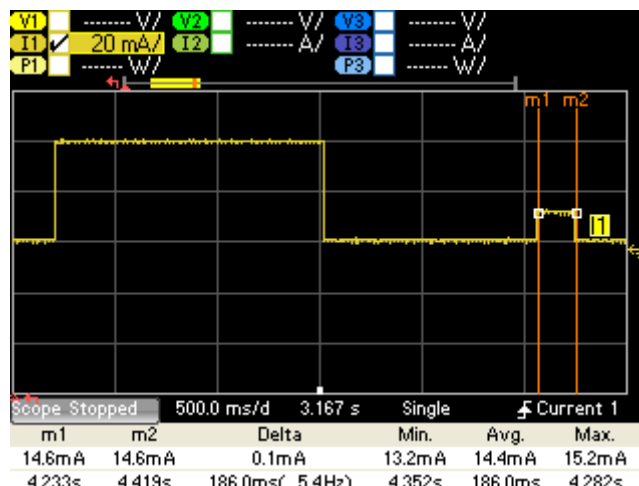


Fig. 3.7 Primera ventana de recepción con ACK en modo DR0

Sin ACK y transmitiendo en modo DR5, la ventana permanece abierta 11,8 ms consumiendo 14.4 mA mientras que con ACK su duración asciende a 186 ms.

Si comparamos con las imágenes anteriores podemos ver que desde que finaliza la transmisión hasta la primera ventana hay un lapso de tiempo de 1 segundo. En el caso DR5 la transmisión termina en el instante 0,051 segundos y la ventana se abre en el momento 1,051 segundos. En el caso DR0 ocurre de igual manera que la transmisión termina en el segundo 3,189 y la ventana se abre en el segundo 4,233.

En la **Fig. 3.5**, vemos que la primera ventana de recepción se cierra en el instante 1,063 segundos y en la **Fig. 3.8** la segunda ventana se abre en el momento 2,058 segundos, aproximadamente un segundo después.

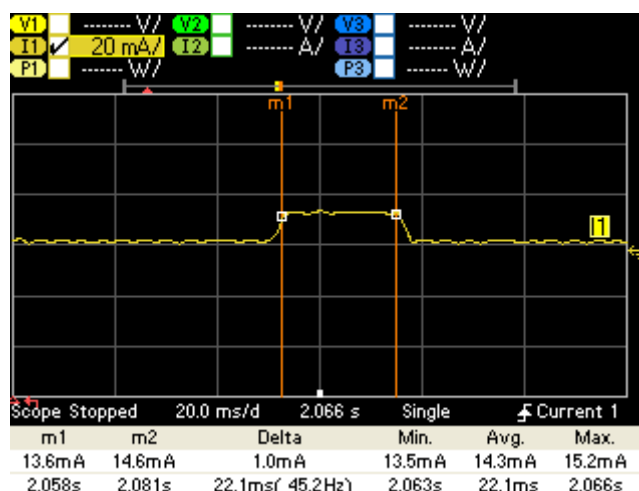


Fig. 3.8 Segunda ventana de recepción sin ACK en modo DR5

Estos tiempos entre ventanas vienen definidos en la propia especificación de LoRaWAN y se cumple para todos los módulos LoRaWAN funcionando a cualquier data rate.

Conociendo el consumo en cada uno de los estados por los que pasa el transceptor LoRa cuando transmite, se puede hacer una gráfica con la corriente media consumida en función de la tasa de envío de paquetes en los modos DR0 y DR5 con y sin ACK (**Fig. 3.9**).

Para calcular el consumo medio se ha empleado la siguiente fórmula:

$$I_{avg} = t_{Tx} * I_{Tx} + t_{1v} * I_{1v} + t_{2v} * I_{2v} + t_{idle} * I_{idle} + t_{sleep} * I_{sleep} \quad (3.1)$$

siendo I_{avg} el consumo medio, t_{Tx} el tiempo en porcentaje que el transceptor está transmitiendo, I_{Tx} el consumo medio mientras el transceptor está transmitiendo, t_{1v} el tiempo en porcentaje que está la primera ventana de recepción abierta, I_{1v} el consumo mientras la primera ventana está abierta, t_{2v} el tiempo en porcentaje que está la segunda ventana de recepción abierta, I_{2v} el consumo de la segunda ventana, t_{idle} el tiempo en porcentaje entre la transmisión del paquete y la primera ventana y la primera ventana y la segunda, I_{idle} el consumo en el tiempo descrito anteriormente, t_{sleep} el tiempo en porcentaje que el transceptor está "dormido" y I_{sleep} el consumo en modo sleep.

En la **Tabla 3.1** se pueden ver los resultados obtenidos de utilizar la formula anterior (3.1).

Tabla 3.1 Consumo medio según el tiempo que pasa entre transmisiones

Tiempo	DR0 sin ACK (mA)	DR0 con ACK (mA)	DR5 sin ACK (mA)	DR5 con ACK (mA)
5 min	0.33463	0.3426794	0.1653252	0.1733746
10 min	0.237315	0.24137867	0.15237467	0.15643833
15 min	0.20487667	0.20758578	0.14824978	0.15095889
30 min	0.17243833	0.17379289	0.14412489	0.14547944
1 h	0.15621917	0.15689644	0.14206244	0.14273972
5 h	0.14324103	0.14337649	0.14040969	0.14054514
1 día	0.14067522	0.14070344	0.14008535	0.14011357

Estos cálculos se ha hecho teniendo en cuenta que los paquetes transmitidos son los vistos anteriormente con un payload de 51 bytes.

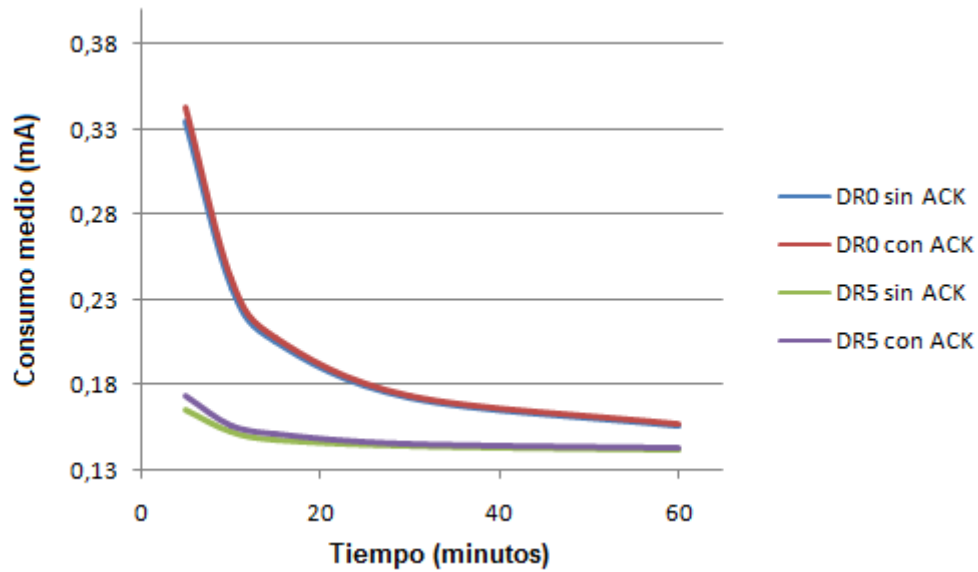


Fig. 3.9 Consumo medio de una transmisión de 51 bytes

En la gráfica se puede ver como, cuanto más tiempo se encuentre el nodo en modo sleep entre una transmisión y otra, más tendie este consumo al nivel de consumo en modo sleep.

Empleando una nueva fórmula (3.2) podemos estimar el tiempo de vida de la batería.

$$Lifetime = \frac{C(Ah)}{I_{avg}(A)} \quad (3.2)$$

siendo C la capacidad nominal de la batería.

Los resultados de emplear la fórmula (3.2) se pueden ver en la **Tabla 3.2**.

Tabla 3.2 Tiempo de vida de una batería con 6600 mAh de capacidad nominal

Tiempo	DR0 sin ACK (días)	DR0 con ACK (días)	DR5 sin ACK (días)	DR5 con ACK (días)
5 min	821.80	802.5	1663.39	1586.16
10 min	1158.8	1139.29	1804.76	1757.88
15 min	1342.27	1324.75	1854.98	1821.69
30 min	1594.77	1582.34	1908.07	1890.30
1 h	1760.35	1752.75	1935.77	1926.58
5 h	1919.84	1918.03	1958.55	1956.67
1 día	1954.86	1954.47	1963.09	1962.69

En el peor de los casos la batería duraría 802.5 días, el equivalente a 2.2 años con los nodos trabajando en modo DR0 con ACK. Cuanto mayor sea el período entre transmisiones más durará la batería y menor es la diferencia de duración entre los distintos modos. En el caso de transmitir un paquete diario en modo DR5 sin ACK se obtendría una duración de 5.23 años.

En la gráfica de la **Fig 3.10** se puede apreciar como la curva tiende a acercarse a los 2000 días de duración a medida que el tiempo entre transmisiones crece. El modo DR0 no es recomendable para transmisiones continuas ya que pasa más tiempo transmitiendo y, por tanto, consumiendo la carga de la batería.

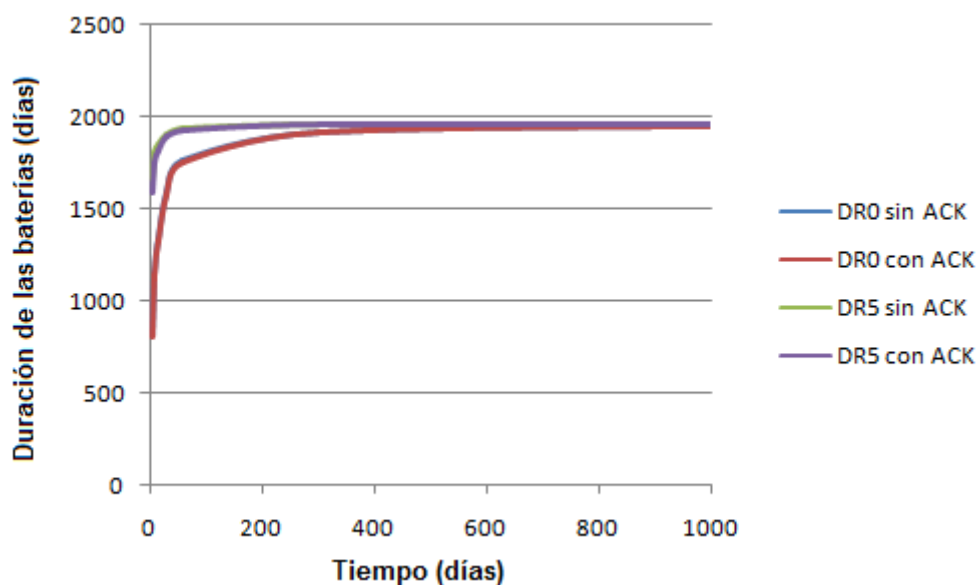


Fig. 3.10 Tiempo de vida de las baterías en función del periodo entre transmisiones

3.2 Throughput en LoRa y en LoRaWAN

Para medir el throughput en LoRa se ha utilizado una configuración con la cual el transmisor genera paquetes con un payload de 100 bytes cada segundo y comprobando mediante un timestamp a qué ritmo llegaban estos paquetes al receptor. El tamaño total de los paquetes será de los 100 bytes de payload más los 5 bytes del resto de campos que conforman un paquete LoRa y los 22 bytes que la librería de Libelium introduce en el campo *data*.

A continuación disponemos de una tabla (**Tabla 3.3**) comparativa con el throughput resultante en cada modo de los disponibles en la librería LoRa de Libelium.

Tabla 3.3 Tiempo de transmisión y Throughput en LoRa

Modo	Tiempo de transmisión de un paquete de 100 bytes (ms)	Tiempo de transmisión de un paquete de 100 bytes con ACK (ms)	Throughput sin ACK	Throughput con ACK
1	4727	6101	211,55 bps	163,91 bps
2	2510	3468	398,4 bps	288,35 bps
3	1379	2220	725,16 bps	450,45 bps
4	1330	2149	751,88 bps	465,33 bps
5	800	1509	1,250 kbps	662,69 bps
6	858	1584	1,156 kbps	631,31 bps
7	522	1183	1,915 kbps	845,31 bps
8	379	1012	2,639 kbps	988,14 bps
9	313	944	3,195 kbps	1,059 kbps
10	258	873	3,876 kbps	1,145 kbps

El modo 1 soporta la transmisión más lenta pero la que ofrece mayor rango de alcance. Al contar con ancho de banda menor y un spreading factor mayor, hace posible alcanzar una distancia mayor al resto de modos. Este modo es perfecto para distancias largas donde las comunicaciones no requieran una tasa de paquetes elevada, hecho que reduce la duración de las baterías más que cualquier otro modo de los disponibles en la librería de Libelium.

Por el contrario, el modo 10 alcanza una velocidad de transmisión muy superior sacrificando cobertura, ya que este modo es muy vulnerable a los muros y demás obstáculos que se pueda encontrar entre el módulo y el gateway.

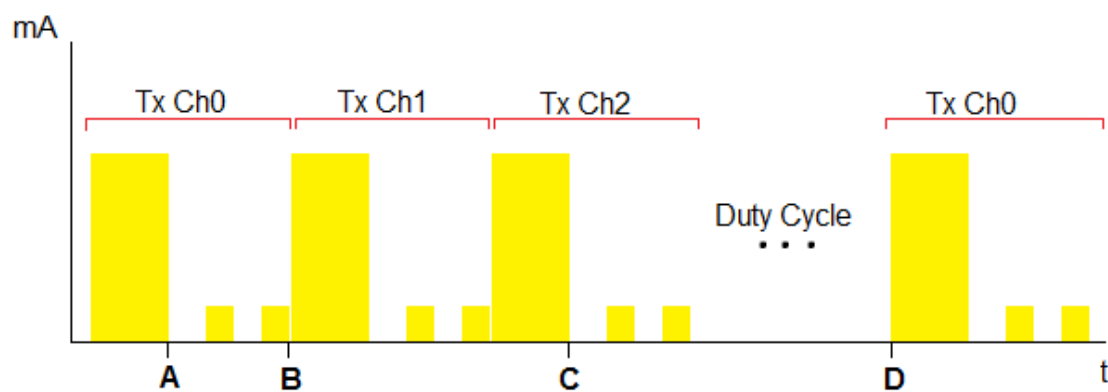
El estudio del throughput en LoRaWAN es más complejo, pues contamos con la opción de transmitir por varios canales y una normativa que obliga al usuario a no sobrepasar un ciclo de trabajo del 1% entre todos los canales habilitados.

Para medir el throughput, realizamos la transmisión de ráfagas de tres tramas consecutivas mediante un módulo LoRaWAN con los tres canales por defecto y varios puntos marcados que representan los siguientes instantes (ver la **Tabla 3.4**). Las capturas de tiempo se han llevado a cabo con el analizador de potencia, con el cual se controlaba el momento justo en el que el módulo transmitía y abría las ventanas de transmisión.

En esta ocasión hay varios detalles interesantes que medir. Utilizando el analizador de potencia para medir retardos de transmisión, se han asignado varios puntos de interés que ocurren en la transmisión de cada trama LoRaWAN (ver **Fig. 3.11**).

Tabla 3.4 Instantes de interés en una transmisión LoRaWAN según la Fig. 4.8

A	Finalizada una transmisión por un canal. Correspondería a una transmisión sin ACK.
B	Finalizada una transmisión y sus dos ventanas de recepción. Lo que correspondería a una transmisión con ACK.
C	Finalizada una ráfaga de paquetes con el número máximo de canales habilitados. Correspondería a la transmisión del mayor número de bits con mensajes consecutivos.
D	Finalizada una transmisión con un ciclo de trabajo del 0.33%, haciendo un total del 1% disponible entre todos los canales habilitados. Equivaldría a una transmisión completa y a través de la cual se puede calcular el throughput.

**Fig. 3.11** Esquema de una transmisión real en LoRaWAN con 3 canales

Para calcular el throughput se han enviado mensajes de 51 bytes, la máxima posible en modo DR0, de modo que todos los mensajes transmitidos puedan ser idénticos. Se han hecho experimentos para los DR comprendidos entre DR0 y DR5, ambos inclusive (ver **Tabla 3.5**).

Los canales habilitados son los tres por defecto con un ciclo de trabajo del 0,33% para cada uno. Si el administrador de la red lo prefiere, puede habilitar tantos canales como permita el gateway hasta un máximo teórico de 15 canales según la especificación LoRaWAN. La regulación del ciclo de trabajo corre a cargo del administrador, que deberá asegurarse de no superar el 1% con la suma de todos los canales.

Tabla 3.5 Tabla de tiempos y throughput en LoRaWAN

Datarate	A	B	C	D	Throughput
0	2,9 s	4,7 s	5,1 s	848,4 s	0,48 bps
1	1,7 s	3,7 s	5,3 s	462 s	0,88 bps
2	800 ms	2,7 s	5,8 s	214,2 s	1,9 bps
3	600 ms	2,6 s	6,7 s	118,8 s	3,43 bps
4	400 ms	2,5 s	9,2 s	66 s	6,18 bps
5	300 ms	2,4 s	5,1 s	37 s	11,03 bps

De la tabla anterior podemos comprobar que LoRaWAN es un protocolo para redes de sensores con un throughput muy reducido. De ello se deduce que el principal objetivo de sus desarrolladores era crear un protocolo de largo alcance enfocado especialmente en su cobertura, para ofrecer conectividad a un número elevado de dispositivos sensores sin requisitos elevados de throughput, reduciendo el coste de infraestructura.

Como medida adicional, se incluye una tabla con en la que los mensajes LoRaWAN pasan a transmitir paquetes de 220 bytes en modo DR5 (**Tabla 3.6**).

Tabla 3.6 Máximo throughput posible con tres canales

Datarate	A	B	C	D	Throughput
5	600 ms	3,3 s	6 s	121 s	14,55 bps

A pesar de contar con un throughput reducido, el protocolo LoRaWAN puede alcanzar altas velocidades instantáneas de transmisión para el número de bytes que es capaz de transportar. Aunque no sea un protocolo adecuado para transportar grandes volúmenes de datos, sí puede serlo para cubrir una amplia zona de sensores que no requieran de una transmisión continua.

3.3 Cobertura de LoRa/LoRaWAN

Una vez medido el bajo throughput de LoRa y LoRaWAN, a continuación procederemos a comprobar la principal ventaja de LoRa/LoRaWAN: su cobertura.

Realizaremos un despliegue real de una red de sensores LoRa en la Escola d'Enginyeria de Telecomunicació i Aeroespacial de Castelldefels que cubre parte de la ciudad de Castelldefels y analizaremos la cobertura de los nodos sensores. La prueba consiste en medir el nivel de señal en distintos puntos y ver cómo la ubicación de los nodos y el modo de transmisión afecta a la señal recibida.

La red LoRa desplegada cumple con las siguientes características:

- Se emplearon módulos LoRa en una comunicación P2P.
- Los módulos utilizan el modo 1, el cual permite una comunicación de mayor alcance.
- Potencia de transmisión de 14 dBm.
- Canal de frecuencia 12 en la banda de 868 MHz
- Mensajes con un payload de 25 bytes.
- Se han transmitido 100 mensajes desde cada ubicación para estimar las pérdidas.



Fig. 3.12 Mapa de cobertura en la ciudad de Castelldefels

En el centro del mapa de la **Fig. 3.12** tenemos el gateway marcado con una señal roja. El resto de puntos numerados del 1 al 5 se corresponden con cinco ubicaciones clave de la ciudad de Castelldefels.

- El punto 1 se corresponde con la entrada a la zona de huertos próxima al campus. Hay visibilidad directa desde ambos puntos, cosa que quedará reflejada en el éxito del 100% en la entrega de mensajes (ver **Tabla 3.7**).
- El punto 2 se encuentra en el paseo marítimo de la ciudad de Castelldefels. A pesar de la poca distancia con el gateway, la calidad de

la señal y el éxito de entrega de los mensajes son inferiores a la esperada por la cantidad de edificios que bloquean la visibilidad.

- El punto 3 se encuentra frente al centro comercial Anec Blau. La distancia entre este punto y el gateway no es muy extensa y el número de edificios que impiden la visibilidad directa es reducida.
- El punto 4 está situado en el Castell de Castelldefels. Nos encontramos con una comunicación a mayor distancia que las anteriores y con numerosos edificios bloqueando la visibilidad directa, incluido el propio castillo.
- El punto 5 está al lado del Canal Olímpic de Catalunya. Contamos con una comunicación casi directa, impedida únicamente por un edificio.

Los resultados de las pruebas de cobertura se pueden ver en la **Tabla 3.7**:

Tabla 3.7 Resultados de las pruebas de cobertura

Punto	Distancia	SNR (dB)	RSSI (dB)	RSSI packet (dBm)	Paquetes entregados (%)
1	500 m	-13	-112	-129	100
2	1,15 km	-5	-109	-122	49
3	714 m	-20	-112	-120	96
4	1,3 km	-12	-113	-129	95
5	945 m	-12	-112	-127	91

Como se verá a continuación (**Tabla 3.8**), si se cambiaran los nodos a otro modo de funcionamiento, el número de paquetes entregados con éxito se vería notablemente afectado. En modo 10, el cual cuenta con un throughput muy superior al modo 1, obtendríamos unas pérdidas de casi el 100% en cualquiera de los puntos del mapa anterior.

El modo 10 es adecuado si se quiere recibir un volumen de datos mayor dentro del mismo edificio en el que se encuentra el gateway pero es una mala opción cuando se pretende cubrir una distancia superior a los 100 metros por la mala penetración a los edificios que tiene, por lo que se deduce que el modo 10 está limitado a interiores o a comunicaciones con visibilidad directa.

Entre el gateway y los puntos 1, 3 y 4 apenas hay edificios altos que causen problemas de visibilidad y aunque la distancia entre el gateway y distintos puntos sea dispar, las pérdidas son mínimas con apenas diferencias entre ellas. Esto indica que la distancia no es la principal causa de las pérdidas de paquetes, sino el número de obstáculos que bloquean la visibilidad con el gateway.

A continuación se comprobará cómo afectan a la señal las distintas configuraciones de modo de transmisión, potencia de salida y canal empleado para la comunicación.

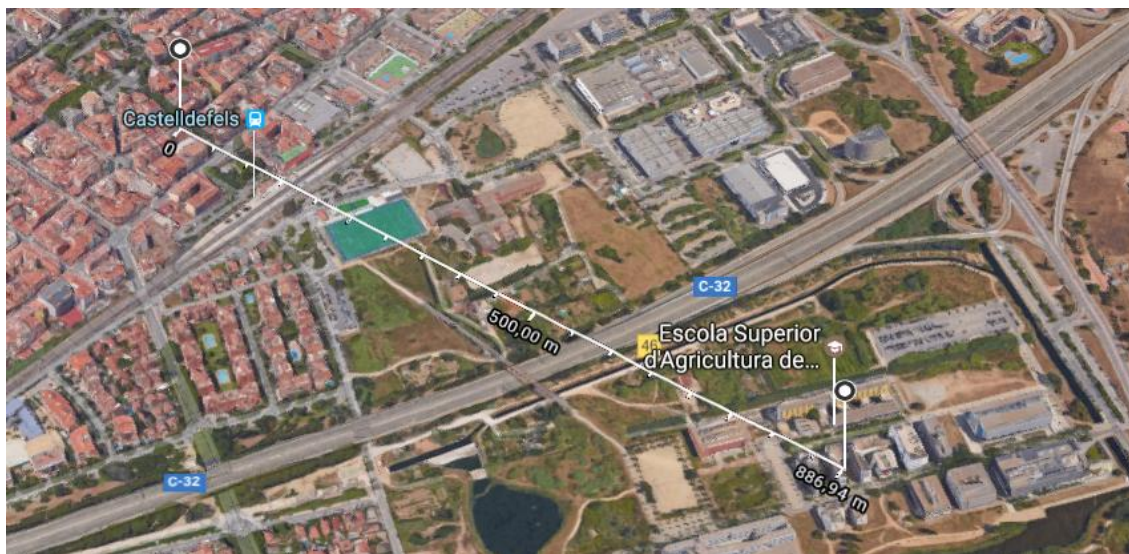


Fig. 3.13 Segundo mapa de cobertura

La distancia entre el gateway del campus y el módulo LoRa ubicado en Castelldefels (mostrado en la **Fig. 3.13**) es de 887 metros. A pesar de no encontrar obstáculos en la mayor parte del camino, varios edificios bajos impiden la visibilidad directa entre dispositivos.

Tabla 3.8 Resultados segunda prueba de cobertura

Modo	Potencia	SNR	RSSI	PacketRSSI	Paquetes entregados (%)
1	H	-19	-113	-129	87
	M	-13	-113	-123	96
5	H	-22	-111	-127	63
	M	-16	-111	-121	89
10	H	-	-	-	0
	M	-	-	-	0

Como es lógico, la potencia de salida de la señal afecta claramente a la calidad de la señal como al éxito de entrega de paquetes.

En el modo 10, el gateway no llega a recibir ningún mensaje del módulo. Como ya se comentó anteriormente, este modo no es apto para cubrir largas distancias sin visibilidad directa. A pesar de eso, el modo 5 consigue establecer una comunicación casi sin pérdidas empleando la máxima potencia de transmisión y una tasa de éxito del 63% con una potencia de transmisión alta. Recordemos que la diferencia entre ambas potencias de salida es de 7 dB.

CAPÍTULO 4. CONCLUSIONES

Después de haber dado por terminado este proyecto podemos dar por cumplido el objetivo de este, el cual era efectuar un estudio sobre LoRa y LoRaWAN y las características que componen una red de sensores que utilicen estas tecnologías.

A lo largo de este documento se han recopilado datos y efectuado pruebas con la intención de que el lector pueda extraer sus propias conclusiones y sopesar en qué tipos de escenarios puede ser una buena idea desplegar una red de sensores LoRa/LoRaWAN. Por ejemplo, un punto importante es la falta de mecanismos de seguridad en LoRa (que no en LoRaWAN) que impide el uso de estos dispositivos en el caso de que sea necesario transmitir información privada.

Junto a varios factores que pueden parecer destacables en LoRa/LoRaWAN ha quedado demostrado que la característica principal y su mayor virtud es el rango de cobertura que cubren las redes LoRa. Esta característica abre un abanico de posibilidades a la hora de pensar posibles usos para estas redes que no puedan cubrir otros protocolos.

La cantidad de información que puede transmitir por mensaje también puede ser una ventaja en comparación con otros protocolos que compiten en cobertura pero no consiguen igualar el tamaño de los datos que LoRa/LoRaWAN pueden transportar por mensaje.

También es necesario destacar el limitado throughput de LoRaWAN y la incapacidad para transmitir periódicamente a causa del cumplimiento del ciclo de trabajo que en todo momento, los módulos LoRaWAN, deberán cumplir con la normativa de que la suma del ciclo de trabajo de todos los canales en uso no supere el 1%. A pesar del gran inconveniente que supone esta limitación, puede no suponer ningún problema cuando se necesita transportar cierta cantidad de información en pocos mensajes que no requieran una actualización inmediata y que permita a los dispositivos cumplir el ciclo de trabajo.

El consumo puede parecer un tanto elevado, pero teniendo en cuenta la clase de uso para la que fue diseñada esta tecnología y el protocolo LoRaWAN y el cumplimiento del ciclo del trabajo ayudará también a reducir el consumo y lograr una mayor autonomía en las baterías.

Las redes desplegadas para realizar este proyecto constituyen un posible prototipo, pero nada impide hacer realidad un escenario de smart city con dispositivos LoRa/LoRaWAN más allá del coste de estos y la instalación, la cual no requiere ningún tipo de obra que pueda suponer un aumento de presupuesto.

4.1 Líneas futuras

De cara al futuro, para otros posibles proyectos o una ampliación de este, sería interesante ampliar el estudio sobre el consumo para distintos niveles de potencia de transmisión, un número mayor de canales LoRaWAN y variaciones en el ciclo de trabajo de cada uno de esos canales. Además de repetir las mismas pruebas si actualizan el firmware de los dispositivos LoRaWAN solucionando el problema de consumo mencionado en el capítulo 3.

Aunque LoRa sea la capa física de LoRaWAN y las pruebas de cobertura de este proyecto (realizadas con dispositivos LoRa) sean teóricamente iguales que en LoRaWAN, sería interesante hacer un estudio sobre la cobertura LoRaWAN si en un futuro se contara con un gateway LoRaWAN en el Campus del Baix Llobregat.

Dejando de lado las capacidades de LoRa/LoRaWAN, interesaría desplegar una red LoRa/LoRaWAN real y describir el proceso de selección de componentes (adecuados para el tipo de red que se quiera establecer), instalación y protección de los dispositivos, configuración para esa red concreta y presupuesto.

4.2 Estudio de ambientación

Existe la posibilidad de incluir unas placas solares que recarguen las baterías a lo largo del día. Únicamente sería necesario tener en cuenta el consumo en la zona del back-end donde se encuentra el servidor de red, el cual puede ser nulo para el usuario si utiliza servidores externos como el servidor de The Things Network visto en el Anexo III.

Si a lo anterior añadimos el impacto ambiental que supone el ahorro de personal que deba desplazarse hasta los puntos en los que se ubiquen cada uno de los módulos, a sabiendas de que la distancia sea del orden de centenares de metros o de unos pocos kilómetros, nos encontramos con un ahorro importante de dinero y recursos.

Los datos podrían utilizarse para distintas aplicaciones o estudios y a medida que se van ejecutando medidas contra la suciedad de los estanques o canales, se pueden mantener los valores deseados una vez se hayan cumplido los objetivos deseados y así evitar que los problemas vuelvan a suceder.

BIBLIOGRAFÍA

- [1] A. Minaburo and L. Toutain, *LPWAN GAP Analysis, draft-minaburo-lpwan-gap-analysis-01*, pp. 1-2, Network Working Group (2016). Recuperat de <https://tools.ietf.org/html/draft-minaburo-lpwan-gap-analysis-01>
- [2] C. Gomez, J. Paradells and J. Crowcoft, *Analysis of IPv6 over LPWAN design space and challenges, draft-gomez-lpwan-ipv6-analysis-00*, pp. 1-2, Network Working Group (2016). Recuperat de <https://tools.ietf.org/html/draft-gomez-lpwan-ipv6-analysis-00>
- [3] LoRa Alliance. LoRa Alliance — What is LoRa? Recuperat de <https://www.lora-alliance.org/What-Is-LoRa/Technology>
- [4] Semtech Corporation. Semtech — LoRa FAQs. Recuperat de <http://www.semtech.com/wireless-rf/lora/LoRa-FAQs.pdf>
- [5] Libelium. Waspote LoRaWAN — Networking Guide. Recuperat de <http://www.libelium.com/downloads/documentation/Waspote-lorawan-networking-guide.pdf>
- [6] Libelium. Waspote-LoRa-868MHz_915MHz-SX1272 — Networking Guide. Recuperat de http://www.libelium.com/downloads/documentation/Waspote_lora_868_mhz_915mhz_sx1272_networking_guide.pdf
- [7] N. Sornin, M. Luis, T. Eirich and T. Kramp, *LoRaWAN Specification*, LoRa Alliance (2015). Recuperat de <https://www.lora-alliance.org/portals/0/specs/LoRaWAN%20Specification%201R0.pdf>
- [8] Microchip. RN2483 — Low-Power Long Range LoRa Technology Transceiver Module. Recuperat de <http://ww1.microchip.com/downloads/en/DeviceDoc/50002346A.pdf>
- [9] Semtech Corporation. Semtech Products — Semtech SX1272. Recuperat de <http://www.semtech.com/wireless-rf/rf-transceivers/sx1272/>
- [10] Microchip. RN2483 LoRa Technology — PICtail/PICtail Plus Daughter Board User's Guide. Recuperat de <http://ww1.microchip.com/downloads/en/DeviceDoc/50002366A.pdf>
- [11] Microchip. RN2483 LoRa Technology Module — Command Reference User's Guide. Recuperat de <http://ww1.microchip.com/downloads/en/DeviceDoc/40001784B.pdf>
- [12] The Things Network. Documentation — Kerlink LoRa IoT Station. Recuperat de <https://www.thethingsnetwork.org/docs/gateways/kerlink/>

- [13] LoRa-net. Github — Packet_forwarder. Recuperat de https://github.com/Lora-net/packet_forwarder
- [14] Libelium. Waspote IDE — User Guide for Waspote PRO. Recuperat de http://www.libelium.com/downloads/documentation/v12/Waspote_ide_user_guide.pdf
- [15] Raspberry Pi Foundation. Raspian OS — Downloads. Recuperat de <https://www.raspberrypi.org/downloads/raspbian/>
- [16] Python. Python — Downloads and Documentation. Recuperat de <https://www.python.org/>
- [17] Sourceforge. Cütecom — Download and Documentation. Recuperat de <http://cütecom.sourceforge.net/>
- [18] Libelium. Libelium Devolpment — Waspote SDK Applications. Recuperat de http://www.libelium.com/development/Waspote/sdk_applications
- [19] Libelium. Libelium — Encryption Libraries for Waspote Sensor Network. Recuperat de <http://www.libelium.com/products/Waspote/encryption/>
- [20] Agilent Technologies. Agilent N675A DC Power Analyzer — Quick Fast Sheet. Recuperat de http://www.alliedelec.com/images/products/mkt/pb/agilent/quickfactsheets/analyzers/N6705A%205989-8615ENDI_Allied.pdf

ANEXOS

En los anexos se incluirán las partes del TFG que, a pesar de ser totalmente necesarios, quedan fuera del tema principal que trata sobre un estudio sobre los protocolos LoRa y LoRaWAN.

Los anexos se dividirán por tipo de red y por elemento de red. Empezando por la configuración del gateway LoRa y los módulos LoRa con transceptores SX1272. A continuación se hará lo propio con la configuración del gateway LoRaWAN y los módulos LoRaWAN RN-2483-PICTAIL de Microchip. Se dedicará un último apartado a explicar los pasos a seguir para efectuar las pruebas de consumo de los módulos LoRaWAN.

I Puesta a punto del gateway LoRa

Este apartado se compone de dos partes. En primer lugar la configuración del gateway LoRa y en segundo la instalación y configuración de una Raspberry Pi para controlar el gateway desde un ordenador remoto mediante una sesión SSH.

I.1 Configuración del LoRa Gateway

El gateway utilizado en nuestra red LoRa se trata de un módulo Wasmote Gateway que utiliza un transceptor SX1272 como radio LoRa y operará en la banda de 868 MHz. El transceptor dispondrá de una antena de 4.5 dBi, al igual que los módulos de la red.

A diferencia de los transceptores SX1272 utilizados en los módulos sensores, que se programan utilizando la aplicación Wasmote PRO (Anexo II), requiere de una configuración específica para este gateway.

Una vez tenemos el transceptor sobre el Wasmote Gateway y conectado a un ordenador a través de una conexión USB, será necesario conocer el puerto COM si se utiliza Windows o el puerto ttyUSB si se utiliza Linux.

A continuación se requerirá de algún programa de comunicación serie para pasar una configuración. En este caso se utilizó Cutecom.

La configuración del Cutecom debe ser la siguiente:

- Baud rate: 38400
- Data bits: 8
- Stop bits: 1
- Parity: None

A continuación se muestra una captura (**Fig. I.1**) de la configuración de CúteCom antes de iniciar la comunicación.

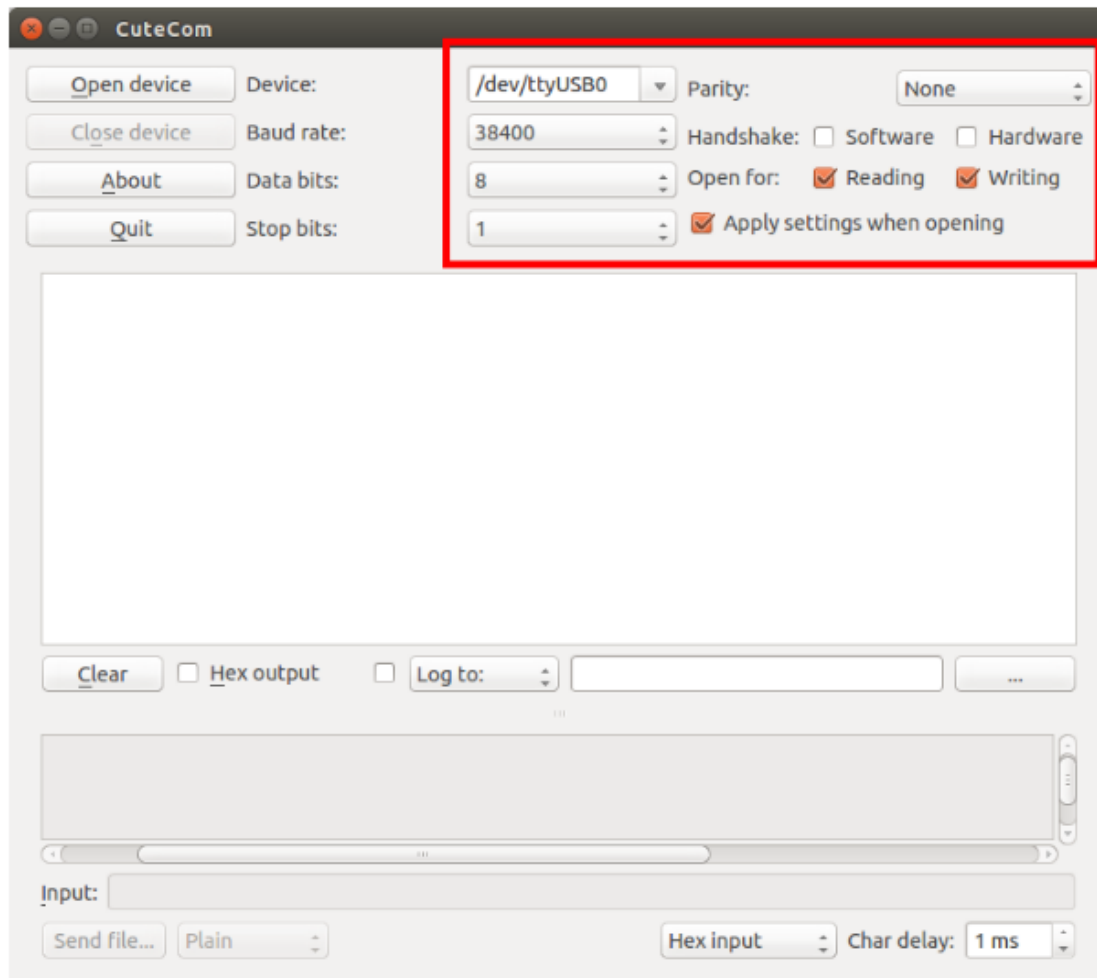


Fig. I.1 Configuración Cútecom

Para la comunicación entre un PC y el módulo se utiliza un protocolo diseñado específicamente para esto. Este protocolo usa algunos caracteres especiales.

Por ejemplo, para conocer la configuración actual del gateway se emplearía el siguiente comando:

01 52 45 41 44 0D 0A 32 41 33 31 04

Traducido quedaría como:

[SOH]READ[CR+LF]2A31[EOT]

El primer número hexadecimal corresponde al carácter especial SOH que indica el comienzo de un comando de configuración.

Los siguientes cuatro caracteres hexadecimales corresponden al comando READ y se encarga de obtener la configuración del módulo.

Los caracteres hexadecimales 0D y 0A corresponden a los caracteres especiales CR y LF, los cuales sirven para empezar una nueva línea.

Los caracteres 2A31 sirven para recuperar la configuración del módulo LoRa y se complementa con el comando READ.

Y para acabar, el carácter hexadecimal 04 se trata de otro carácter especial para indicar el fin de una transmisión.

Para pasar una configuración al gateway a través del programa Cútecom, se utiliza el mismo lenguaje y es necesario introducir el código en hexadecimal. Siempre se puede emplear un conversor ASCII - Hexadecimal y viceversa.

Nuestro gateway dispone de la siguiente configuración:

- Canal de frecuencia: 12
- Dirección: 1
- Ancho de banda: 125 kHz
- Coding Rate: 4/5
- Spreading Factor: 6

El comando para introducir esta configuración sería:

[SOH]INFO#FREQ:CH_12_868;ADDR:3;BW:BW_125;CR:CR_6;SF:SF_6;RSSI:32;SNR:21[CR+LF]FB05[EOT]

Y el resultado en hexadecimal:

01 49 4e 46 4f 23 46 52 45 43 3a 43 48 5f 31 32 5f 38 36 38 3b 41 44 44 52 3a 33 3b 42 57 3a 42 57 5f 31 32 35 3b 43 52 3a 43 52 5f 36 3b 53 46 3a 53 46 5f 36 3b 52 53 53 49 3a 33 32 3b 53 4e 52 3a 32 31 0D 0A 46 42 30 35 04

Ya solo requerirá de alimentación para funcionar. Para ello se conectará a una Raspberry Pi 3 mediante una conexión USB.

I.2 Configuración de una Raspberry Pi para controlar el LoRa Gateway

Como ya se comentó, se utilizará el modelo 3 de Raspberry Pi para controlar el LoRa Gateway y guardar los datos recogidos por los sensores. Permitiendo así al administrador de la red acceder a estos datos y, si lo desea, subirlos automáticamente a la nube.

Lo primero es seleccionar el sistema operativo que utilizará la placa Raspberry Pi 3. En este caso se optó por utilizar Raspbian por su similitud con Debian y la familiarización con Linux.

El sistema operativo puede descargarse gratuitamente de Internet. Para pasarlo a la Raspberry Pi será necesario disponer de una tarjeta microSD de

más de 2 Gb de memoria y, si se utiliza Windows en el PC, del programa Win32DiskImager (ver **Fig. I.2**).

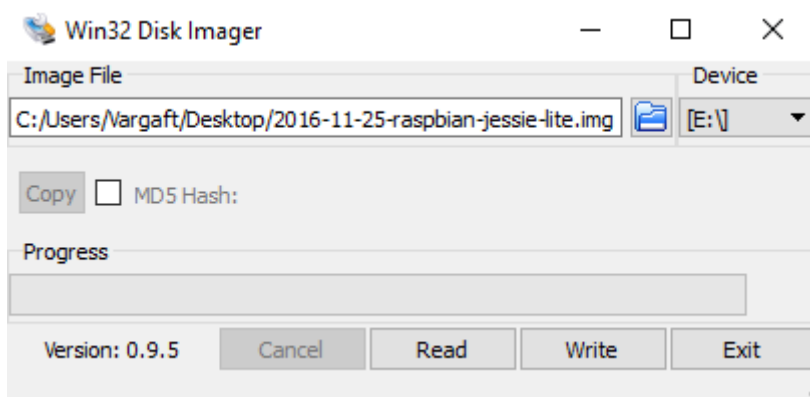


Fig. I.2 Ventana del software Win32DiskImager

Una vez seleccionada la imagen a grabar y la unidad sobre la cual volcar el sistema operativo, clicamos sobre el botón Write.

Es recomendable formatear la tarjeta antes de grabar el sistema operativo en la tarjeta microSD. Un buen programa para ello es el SDFormatter (**Fig.I.3**).

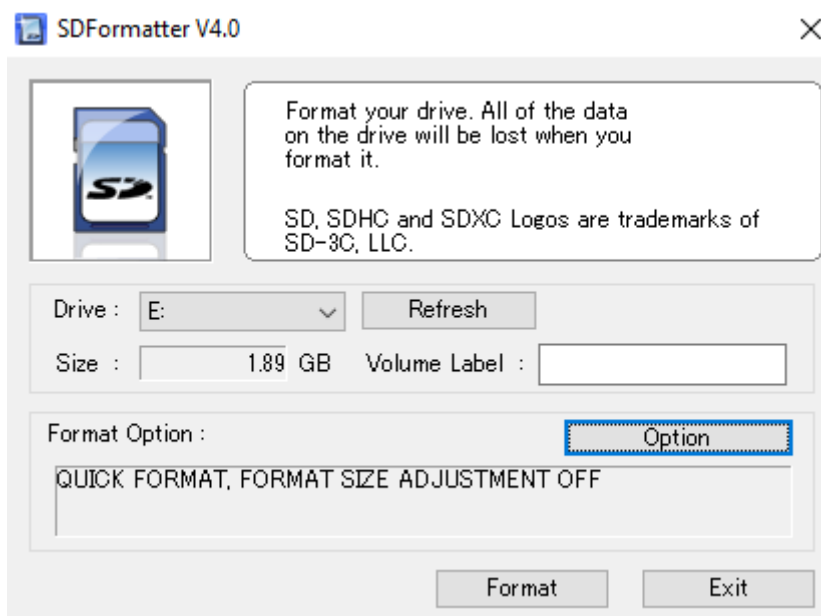


Fig. I.3 Ventana del SDFormatter

Una vez introducida la tarjeta microSD con el sistema operativo en la Raspberry Pi 3 solo hay que seguir intuitivamente los pasos para la primera puesta en marcha.

Cuando lleguemos al escritorio de Raspbian, será necesario habilitar una dirección IP estática para poder acceder a ella más tarde.

Para ello accedemos a la consola de comandos de Raspbian y utilizamos el siguiente comando:

```
sudo nano -w /etc/network/interfaces
```

En pantalla veremos las siguientes líneas:

'interfaces':

```
auto lo  
  
iface lo inet loopback  
iface eth0 inet dhcp
```

Deberán sustituirse o añadir debajo las siguientes::

Nuevo 'interfaces':

```
auto eth0  
iface eth0 inet static  
    address 147.83.113.103  
    netmask 255.255.255.0  
    gateway 147.83.113.1  
    dns-nameservers 8.8.8.8 8.8.4.4  
  
auto eth0:0  
iface eth0:0 inet static  
    address 10.10.10.10  
    netmask 255.255.255.0  
    dns-nameservers 8.8.8.8 8.8.4.4
```

Guardamos el documento y reiniciamos la máquina para comprobar que los cambios se han efectuado correctamente.

A continuación, para habilitar el protocolo SSH debemos acceder de nuevo a la consola de Raspbian para introducir el comando de configuración de Raspbian.

```
sudo raspi-config
```

En el monitor se mostrará el menú de configuración del sistema operativo y accederemos al menú 8 (**Fig. I.4**), opciones avanzadas, para activar el acceso por el protocolo SSH.

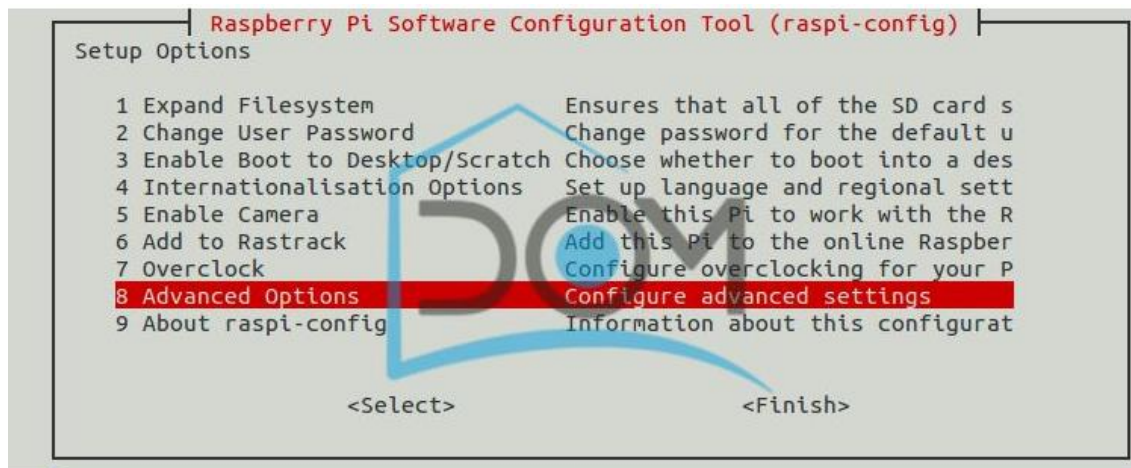


Fig. I.4 Menú de configuración del SO Raspbian

Para acceder a la Raspberry Pi desde un PC con Windows es necesario disponer de un software específico. Una opción es utilizar el programa es Putty (**Fig. I.5**).

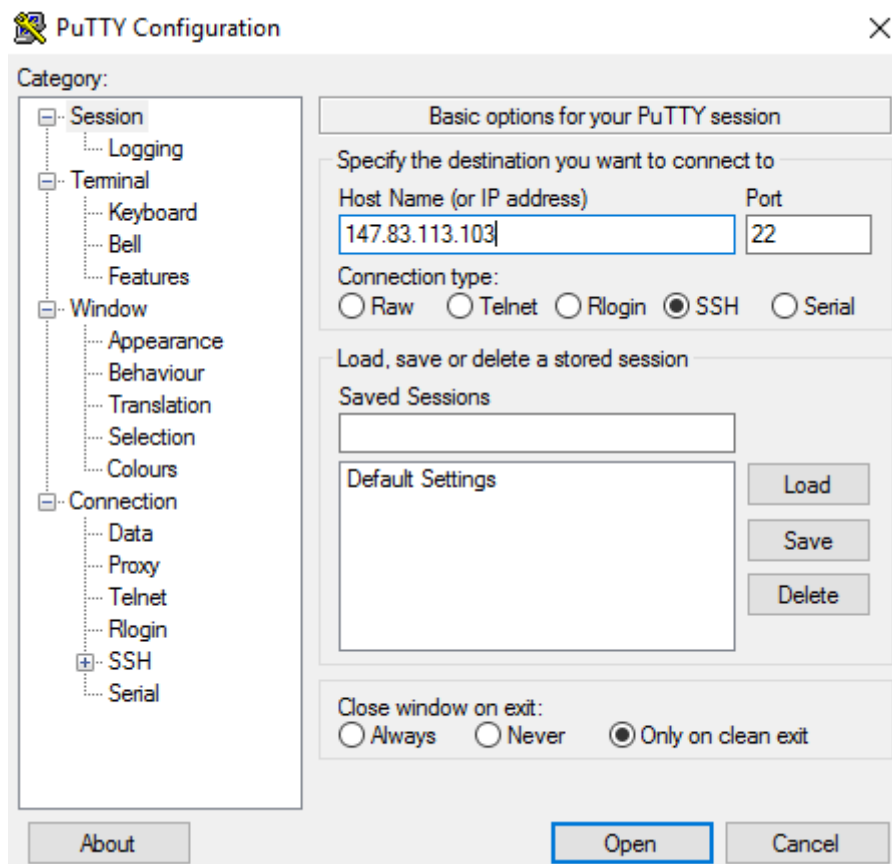


Fig. I.5 Ventana del software Putty

Solo es necesario escribir la dirección IP de la Raspberry Pi y el puerto 22 utilizado para acceder por SSH.

Ahora hay que impedir que cualquier persona ajena acceda a la Raspberry Pi para evitar problemas de seguridad. Para ello configuramos el NAT de la Raspberry Pi para que únicamente se pueda acceder a ella a través de un PC de la misma red ubicado en un laboratorio del campus.

'iptables':

```
sudo iptables -t filter -A INPUT -i eth0 -s 147.83.113.126 -p tcp --dport 22 -j ACCEPT
sudo iptables -t filter -A INPUT -i eth0 -s 147.83.113.104 -p tcp --dport 22 -j ACCEPT
sudo iptables -t filter -A INPUT -i eth0 -m conntrack --ctstate ESTABLISHED -j ACCEPT
sudo iptables -t filter -A INPUT -i eth0 -j DROP
```

Con la anterior configuración únicamente se podrá acceder al Lora Gateway desde las dos máquinas con dirección IP 147.83.113.126 y 147.83.113.104, ubicadas en el despacho 314 y el laboratorio 325 del edificio C3 del campus.

Para finalizar creamos un script de Python que guarde en un archivo de texto los datos de cada paquete LoRa que llegue al gateway. De manera que podamos acceder a ellos abriendo este documento. Otra forma de acceder a estos datos sería creando otro código en Python o ampliando el mismo código mencionado anteriormente para que almacene la información en la nube.

Lo primero de todo será instalar Python. Una vez instalado, descargamos las librerías necesarias para poder ejecutar el script de Python.

- gdata-python-client: Para escribir una aplicación haciendo uso de uno de los muchos servicios de datos de Google.
- gspread-0.3.0: Para conectarse a Google Spreadsheet y establecer un valor a una celda.
- oauth2client-2.2.0: Para poder autenticarte en Google Drive.
- pyserial-3.1.1: Para recoger los datos leídos por el puerto serie.

Para instalar las librerías deberán descomprimirse en alguna ubicación conocida y, desde el Terminal y dentro de la carpeta correspondiente a la librería descomprimida, introducir el comando:

```
sudo python setup.py install
```

El siguiente paso será crear el script que registre los datos de los sensores. El gateway enviará a la Raspberry Pi todos los paquetes entrantes a través del puerto serial.

'loradrive.py':

```
# -*- coding: utf-8 -*-
from __future__ import print_function
import serial
import os
import sys
import json
import gspread
from oauth2client.service_account import ServiceAccountCredentials
from datetime import datetime

scope = ['https://spreadsheets.google.com/feeds']
credentials =
ServiceAccountCredentials.from_json_keyfile_name('/home/entel/loraproject-
201bbbe0af00.json', scope)
gc = gspread.authorize(credentials)
wks = gc.open('test2').sheet1

os.chdir("/home/entel/datos")
ser = serial.Serial('/dev/ttyUSB0', 38400)

i = 2

while True:
    data = open("datos2.txt", "a")
    msg=ser.readline().decode('ascii', 'ignore').strip("\r\n")
    msgStr=str(msg)
    msgStr1=msgStr.split("#")[2]
    msgStr2=msgStr.split("#")[3]
    msgStr3=msgStr.split("#")[4]
    dts=datetime.utcnow().isoformat()
    dtsStr=str(dts)
    data.write(dtsStr + '\t' + msgStr1 + '\t' + msgStr2 + '\t' + msgStr3 + ('\n'))
    print(dtsStr + '\t' + msgStr1 + '\t' + msgStr2 + '\t' + msgStr3 + ('\n'))
    data.close()

    BatteryCell = 'D' + str(i)
    NumpacketCell = 'C' + str(i)
    NodeIDCell = 'B' + str(i)
    dateCell = 'A' + str(i)

    wks.update_acell(BatteryCell, msgStr3)
    wks.update_acell(NumpacketCell, msgStr2)
    wks.update_acell(NodeIDCell, msgStr1)
    wks.update_acell(dateCell, dtsStr)

    i = i + 1
    # Up to the next column

ser.close()
```

Para que el script funcione deberemos crear una carpeta llamada *datos* en la ruta descrita en el código para que se cree el archivo de texto en ella y así guardar los datos.

También deberemos crear una cuenta de Google para hacer uso de su servicio Google Drive, donde se guardarán los datos en una hoja de cálculo creada con anterioridad y llamada *test2*, como se ve en el código.

El siguiente paso será ir a la consola de desarrollador de Google (<https://console.developers.google.com/>) para crear un proyecto y habilitar una Google Drive API.

Deberemos clicar sobre *Credentials/OAuth consent screen* y crear una clave escribiendo una dirección de email y un nombre. Descargas el archivo (JSON) y lo guardamos en la misma ubicación donde esté el script de Python.

El archivo tiene esta forma:

'loraproject-201bbbe0af00.json':

```
{
  "type": "service_account",
  "project_id": "lora-project",
  "private_key_id": "201bbbe0af0077e685cf3ef7362d858d4d3a264a",
  "private_key":
    "\nMIIEvQIBADANBgkqhkiG9w0BAQEFAASCBCkwggSjAgEAAoIBAQC/fH1clriPHbse\nnK60EwxgO3yMaaUYqePqEcpFeO1wCRnxz30uiHaRWiF\nNbid2c6dt0aqhtAOnpYolGilZrUEgDp1zftiGaLPBg8BhBFPNCnx\nnKqyOecnNAGMBAAECggEBALEd6ipt5z5hprGKfpgYCMnAtYx5cbBZhlc8HaA\nvf9pV8cqPuRgdwHlce4u05rRKzo9QxPRwaS6fqYQBZ\nnrpHR/uuqInxp9dKQsZljiOYksyNnzt5rj7TjrUX8t7GECgYEA7taGMeVqnqjljypn\nnlQsHJBQvEE+3nOzim6z9HmZr/QjzTngkUqngkUdzM5Msesig6T4EyVKtKCl/6KRchqbxzQyJf5SRabEHVDrb0qJk+dG6Gj\nn/CGD80V8PaUCgYAxK1DtLM\nCk9\nnrxf9EZjLlpp1FenngUx4WY16U+LcidCRTeOWCMbvNgq3CQ7MFb/yWyLQy/zxcLLG\n\nnk+H4eR/vKolRCAbDekyYZNkbhrq+yKZp1vNe8Qdh/\n",
  "client_email": "1032243964509-compute@developer.gserviceaccount.com",
  "client_id": "102794483755333031470",
  "auth_uri": "https://accounts.google.com/o/oauth2/auth",
  "token_uri": "https://accounts.google.com/o/oauth2/token",
  "auth_provider_x509_cert_url": "https://www.googleapis.com/oauth2/v1/certs",
  "client_x509_cert_url": "https://www.googleapis.com/robot/v1/metadata/x509/1032243964509-compute%40developer.gserviceaccount.com"
}
```

Entramos en la hoja de cálculo *test2*, le damos al botón azul *Share* y compartir con el *client_email* del archivo JSON que se ha introducido anteriormente.

La hoja de cálculo deberá comenzar a registrar datos una vez se haya puesto en marcha el script de Python.

Un punto importante es que el reloj debe tener la hora correcta. Sino no se almacenarán los datos en la hoja de cálculo por no tener el reloj de la Raspberry Pi sincronizado con el de Google.

II Configuración de los módulos LoRa

Con el Lora Gateway en funcionamiento solo resta habilitar los módulos LoRa y que estos envíen datos al gateway.

Como ya comentamos en el capítulo dos, los módulos LoRa constan de un transceptor SX1272, una placa Wasmote y una antena LoRa para la banda de 868 MHz.

Para configurar los módulos es necesario conectar la placa Wasmote al PC a través de un cable USB y disponer del software Wasmote PRO, el cual ya mencionamos también en el segundo capítulo.

Lo primero será cargar las librerías de Libelium en Wasmote PRO (**Fig. II.1**) igual que se haría con Arduino y seleccionar en la pestaña Herramientas la tarjeta Wasmote-API-v0xx, correspondiente a la versión de la librería utilizada, y el puerto al cual está conectada la placa Wasmote.

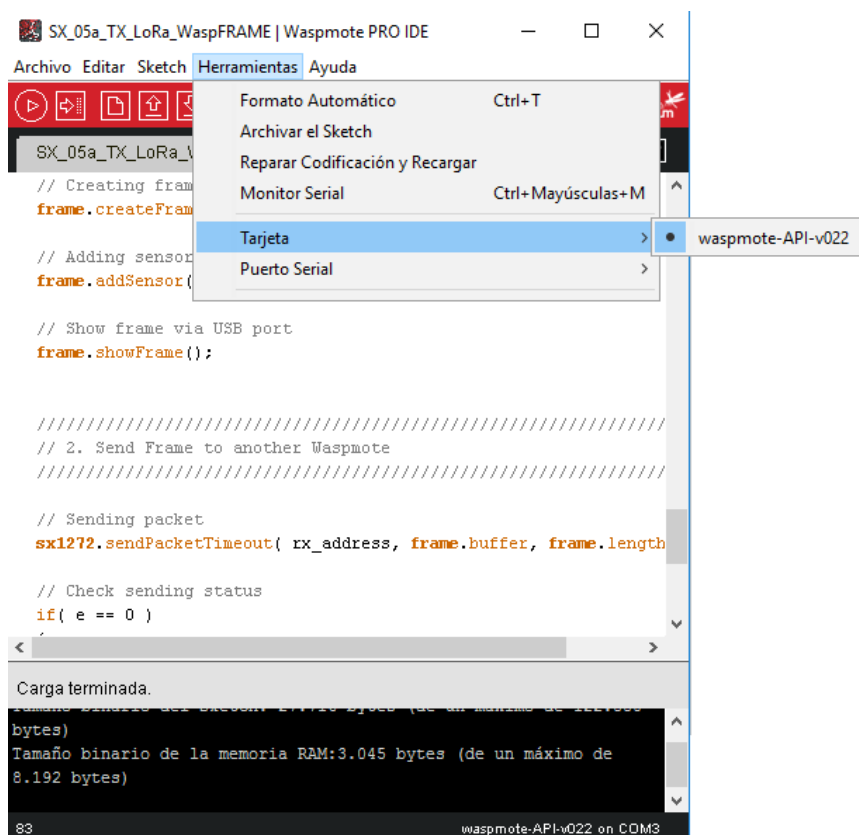


Fig. II.1 Selección de placa Wasmote

Lo siguiente es entrar en la programación de los módulos.

En primer lugar veremos el código que se introducirá en la programación de los módulos y las distintas opciones que ofrece la librería Libelium, o al menos las que utilizaremos. Como complemento adicional veremos cuál sería el resultado en recepción si se dispusiera de otro módulo LoRa el cual programaríamos para recibir paquetes en una comunicación punto a punto.

II.1 Transmisión en módulos LoRa

Los programas constan de dos partes, configuración del módulo LoRa y creación y envío de un paquete LoRa.

'SX_05a_TX_LoRa_WaspFRAME':

```
// Include these libraries to transmit frames with sx1272
#include <WaspSX1272.h>
#include <WaspFrame.h>

// Define private key to encrypt message
char nodeID[] = "node_001";

// define the destination address to send packets
uint8_t rx_address = 5;

// status variable
int8_t e;

void setup()
{
    // Init USB port
    USB.ON();
    USB.println(F("SX_05a example"));
    USB.println(F("Semtech SX1272 module. TX LoRa with Wasp mote Frame"));

    // set the node ID
    frame.setID(nodeID);

    USB.println(F("-----"));
    USB.println(F("Setting configuration:"));
    USB.println(F("-----"));

    // Init SX1272 module
    sx1272.ON();

    // Select frequency channel
    e = sx1272.setChannel(CH_10_868);
    USB.print(F("Setting Channel CH_10_868.\t state "));
    USB.println(e);

    // Select implicit (off) or explicit (on) header mode
    e = sx1272.setHeaderON();
    USB.print(F("Setting Header ON.\t\t state "));
    USB.println(e);

    // Select mode: from 1 to 10
    e = sx1272.setMode(1);
    USB.print(F("Setting Mode '1'.\t\t state "));
    USB.println(e);

    // select CRC on or off
    e = sx1272.setCRC_ON();
    USB.print(F("Setting CRC ON.\t\t\t state "));
```

```
USB.println(e);

// Select output power (Max, High or Low)
e = sx1272.setPower('M');
USB.print(F("Setting Power to 'M'.\t\t state "));
USB.println(e);

// Select the node address value: from 2 to 255
e = sx1272.setNodeAddress(4);
USB.print(F("Setting Node Address to '4'.\t state "));
USB.println(e);
USB.println();

delay(1000);

USB.println(F("-----"));
USB.println(F("Sending:"));
USB.println(F("-----"));
}

void loop()
{
    //////////////////////////////////////
    // 1. Create a new Frame with sensor fields
    //////////////////////////////////////

    USB.println(F("Create new Frame:"));

    // Creating frame to send
    frame.createFrame(ASCII);

    // Adding sensor battery
    frame.addSensor(SENSOR_BAT, (uint8_t) PWR.getBatteryLevel());

    // Adding timestamp
    frame.addTimestamp();

    // Show frame via USB port
    frame.showFrame();
    //////////////////////////////////////
    // 2. Send Frame to another Waspote
    //////////////////////////////////////

    // Sending packet
    sx1272.sendPacketTimeout( rx_address, frame.buffer, frame.length);

    // Check sending status
    if( e == 0 )
    {
        USB.println(F("Packet sent OK"));
    }
    else
    {
        USB.println(F("Error sending the packet"));
        USB.print(F("state: "));
    }
}
```

```

    USB.println(e, DEC);
}

USB.println();
delay(1000);

}

```

El resultado sería similar a lo siguiente:

```

E#
SX_05a example
Semtech SX1272 module. TX LoRa with Waspmote Frame
-----
Setting configuration:
-----
Setting Channel CH_10_868. state 0
Setting Header ON. state 0
Setting Mode '1'. state 0
Setting CRC ON. state 0
Setting Power to 'H'. state 0
Setting Node Address to '4'. state 0
-----
Sending:
-----
Create new Frame:
=====
Current ASCII Frame:
Length: 34
Frame Type: 128
frame (HEX):
3C3D3E800123333837323536303030236E6F64655F3030312330234241543A393823
frame (STR): <=>#387256000#node_001#0#BAT:98#TST:946692072#
=====
Packet sent OK

Create new Frame:
=====
Current ASCII Frame:
Length: 34
Frame Type: 128
frame (HEX):
3C3D3E800123333837323536303030236E6F64655F3030312331234241543A393923
frame (STR): <=>#387256000#node_001#1#BAT:99#TST:946692076#
=====
Packet sent OK

Create new Frame:
=====
Current ASCII Frame:
Length: 34
Frame Type: 128
frame (HEX):
3C3D3E800123333837323536303030236E6F64655F3030312332234241543A393923
frame (STR): <=>#387256000#node_001#2#BAT:99#TST:946692079#
=====
Packet sent OK
...

```

Si quisiéramos utilizar ACKs solamente deberíamos cambiar lo siguiente:

```
// Sending packet
.....
sx1272.sendPacketTimeoutACK( rx_address, frame.buffer, frame.length);
.....
```

Y si quisiéramos habilitar los reintentos de reenvío deberíamos añadir lo siguiente en el *void setup()*:

```
// Select the maximum number of retries: from '0' to '5'
e = sx1272.setRetries(3);
USB.print(F("Setting Max retries to '3'.\t state "));
USB.println(e);
.....
USB.println();
```

II.2 Recepción en módulos LoRa

De la misma manera, el módulo receptor deberá cargar una configuración que le permita recibir mensajes y mostrar los resultados.

'SX_05b_RX_LoRa_WaspFRAME':

```
// Include these libraries to transmit frames with sx1272
#include <WaspSX1272.h>
#include <WaspFrame.h>

// status variable
int8_t e;

void setup()
{
    // Init USB port
    USB.ON();
    USB.println(F("SX_05b example"));
    USB.println(F("Semtech SX1272 module. RX LoRa with Wasp mote Frame"));

    USB.println(F("-----"));
    USB.println(F("Setting configuration:"));
    USB.println(F("-----"));

    // Init SX1272 module
    sx1272.ON();

    // Select frequency channel
    e = sx1272.setChannel(CH_10_868);
    USB.print(F("Setting Channel CH_10_868.\t state "));
    USB.println(e);

    // Select implicit (off) or explicit (on) header mode
```

```

e = sx1272.setHeaderON();
USB.print(F("Setting Header ON.\t\t state "));
USB.println(e);

// Select mode: from 1 to 10
e = sx1272.setMode(1);
USB.print(F("Setting Mode '1'.\t\t state "));
USB.println(e);

// select CRC on or off
e = sx1272.setCRC_ON();
USB.print(F("Setting CRC ON.\t\t\t state "));
USB.println(e);

// Select output power (Max, High or Low)
e = sx1272.setPower('M');
USB.print(F("Setting Power to 'M'.\t\t state "));
USB.println(e);

// Select the node address value: from 2 to 255
e = sx1272.setNodeAddress(5);
USB.print(F("Setting Node Address to '5'.\t state "));
USB.println(e);
USB.println();

delay(1000);

USB.println(F("-----"));
USB.println(F("Receiving:"));
USB.println(F("-----"));
}

void loop()
{
  // Receiving a packet
  e = sx1272.receivePacketTimeout(10000);

  // check status
  if( e == 0 )
  {
    USB.println(F("\nShow packet received: "));

    // show frame received
    sx1272.showFramefromPacket();
  }
  else
  {
    USB.print(F("\nReceiving packet TIMEOUT, state "));
    USB.println(e, DEC);
  }
}

```

El resultado por pantalla será:

```

E#
SX_05b example
Semtech SX1272 module. RX LoRa with Waspmote Frame
-----
Setting configuration:
-----
Setting Channel CH_10_868. state 0
Setting Header ON. state 0
Setting Mode '1'. state 0
Setting CRC ON. state 0
Setting Power to 'H'. state 0
Setting Node Address to '5'. state 0

-----
Receiving:
-----

Show packet received:
=====
Current ASCII Frame:
Length: 36
Frame Type (decimal): 128
HEX: 3C3D3E800123333837323536303030236E6F64655F30303123313536234241543A393923
String: <=>[0]#387256000#node_001#156#BAT:99#
=====

Show packet received:
=====
Current ASCII Frame:
Length: 36
Frame Type (decimal): 128
HEX: 3C3D3E800123333837323536303030236E6F64655F30303123313537234241543A393923
String: <=>[0]#387256000#node_001#157#BAT:99#
=====

Show packet received:
=====
Current ASCII Frame:
Length: 36
Frame Type (decimal): 128
HEX: 3C3D3E800123333837323536303030236E6F64655F30303123313538234241543A393923
String: <=>[0]#387256000#node_001#158#BAT:99#
=====

Show packet received:
=====
Current ASCII Frame:
Length: 36
Frame Type (decimal): 128
HEX: 3C3D3E800123333837323536303030236E6F64655F30303123313539234241543A393923
String: <=>[0]#387256000#node_001#159#BAT:99#
=====
...

```

Si se quisieran utilizar ACKs, solo deberíamos cambiar esta parte del código:

```

void loop()
{
    // Receiving packet and sending an ACK response
    .....
    e = sx1272.receivePacketTimeoutACK(10000);
}

```

II.3 Pruebas de cobertura

A la hora de realizar una prueba de cobertura, es necesario conocer el nivel de la señal. De esta manera podemos cambiar algunos parámetros como la potencia de transmisión, modo o frecuencia, para optimizar el uso de las baterías. También cuenta con un contador de paquetes para calcular las pérdidas. Mientras que el emisor enviará paquetes con el mismo código visto en el apartado de emisión, el módulo en recepción deberá cargar con la siguiente configuración:

'SX_12_RSSI_LoRa':

```
// Put your libraries here (#include ...)
#include <WaspSX1272.h>

// status variable
int8_t e;

void setup()
{
    // init USB port
    USB.ON();
    USB.println(F("SX_12 example"));
    USB.println(F("Semtech SX1272 module RX in LoRa getting RSSI and SNR"));

    USB.println(F("-----"));
    USB.println(F("Setting configuration:"));
    USB.println(F("-----"));

    // init SX1272 module
    sx1272.ON();

    // Select frequency channel
    e = sx1272.setChannel(CH_10_868);
    USB.print(F("Setting Channel CH_10_868.\t state "));
    USB.println(e);

    // Select implicit (off) or explicit (on) header mode
    e = sx1272.setHeaderON();
    USB.print(F("Setting Header ON.\t\t state "));
    USB.println(e);

    // Select mode: from 1 to 10
    e = sx1272.setMode(1);
    USB.print(F("Setting Mode '1'.\t\t state "));
    USB.println(e);

    // Select CRC on or off
    e = sx1272.setCRC_ON();
    USB.print(F("Setting CRC ON.\t\t\t state "));
    USB.println(e);
```

```
// Select output power (Max, High or Low)
e = sx1272.setPower('M');
USB.print(F("Setting Power to 'M'.\t\t state "));
USB.println(e);

// Select the node address value: from 2 to 255
e = sx1272.setNodeAddress(5);
USB.print(F("Setting Node Address to '5'.\t state "));
USB.println(e);
USB.println();

delay(1000);

USB.println(F("-----"));
USB.println(F("Receiving:"));
USB.println(F("-----"));
}

void loop()
{
  // receive packet
  e = sx1272.receivePacketTimeout(10000);

  // check rx status
  if( e == 0 )
  {
    USB.println(F("\nShow packet received: "));

    // show packet received
    sx1272.showReceivedPacket();
  }
  else
  {
    USB.print(F("\nReceiving packet TIMEOUT, state "));
    USB.println(e, DEC);
  }

  //////////////////////////////////////
  // 1. Get SNR
  //////////////////////////////////////
  e = sx1272.getSNR();

  // check status
  if( e == 0 )
  {
    USB.print(F("Getting SNR \t\t--> OK. "));
    USB.print(F("SNR current value is: "));
    USB.println(sx1272._SNR);
  }
  else
  {
    USB.println(F("Getting SNR --> ERROR"));
  }

  //////////////////////////////////////
```



```

// 2. Get channel RSSI
////////////////////////////////////
e = sx1272.getRSSI();

// check status
if( e == 0 )
{
    USB.print(F("Getting RSSI \t\t--> OK. "));
    USB.print(F("RSSI current value is: "));
    USB.println(sx1272._RSSI);
}
else
{
    USB.println(F("Getting RSSI --> ERROR"));
}

////////////////////////////////////
// 3. Get last packet received RSSI
////////////////////////////////////
e = sx1272.getRSSIpacket();

// check status
if( e == 0 )
{
    USB.print(F("Getting RSSI packet \t--> OK. "));
    USB.print(F("Last packet RSSI value is: "));
    USB.println(sx1272._RSSIpacket);
}
else
{
    USB.println(F("Getting RSSI packet --> ERROR"));
}

USB.println();
}

```

Como resultado debería aparecer algo así:

```

E#
SX_12 example
Semtech SX1272 module RX in LoRa getting RSSI and SNR
-----
Setting configuration:
-----
Setting Channel CH_10_868. state 0
Setting Header ON. state 0
Setting Mode '1'. state 0
Setting CRC ON. state 0
Setting Power to 'M'. state 0
Setting Node Address to '5'. state 0

-----
Receiving:
-----

```

```

Show packet received:
=====
dest: 5
src: 2
packnum: 230
length: 26
retry: 0
payload (HEX): 546869735F69735F615F6E65775F6D657373616765
payload (string): This_is_a_new_message
=====
Getting SNR --> OK. SNR current value is: 6
Getting RSSI --> OK. RSSI current value is: -107
Getting RSSI packet --> OK. Last packet RSSI value is: -55

Show packet received:
=====
dest: 5
src: 2
packnum: 231
length: 26
retry: 0
payload (HEX): 546869735F69735F615F6E65775F6D657373616765
payload (string): This_is_a_new_message
=====
Getting SNR --> OK. SNR current value is: 8
Getting RSSI --> OK. RSSI current value is: -106
Getting RSSI packet --> OK. Last packet RSSI value is: -55

...

```

Cabe decir que las magnitudes para los distintos resultados son:

SNR	dB
RSSI	dBm
RSSI packet	dBm

II.4 Sniffing

También podemos hacer que un Wasp mote actúe como sniffing, de manera que pueda recibir todos los paquetes que sea capaz de captar.

'SX_10_receiveAll_LoRa':

```

// Put your libraries here (#include ...)
#include <WaspSX1272.h>

// status variable
int8_t e;

void setup()
{
    // init USB port
    USB.ON();
    USB.println(F("SX 10 example"));
}

```

```

USB.println(F("Semtech SX1272 module RX in LoRa from All Nodes"));

// Init SX1272 module
sx1272.ON();

// Select frequency channel
e = sx1272.setChannel(CH_10_868);
USB.print(F("Setting Channel CH_10_868.\t state "));
USB.println(e);

// Select implicit (off) or explicit (on) header mode
e = sx1272.setHeaderON();
USB.print(F("Setting Header ON.\t\t state "));
USB.println(e);

// Select mode (mode 1 is the faster)
e = sx1272.setMode(1);
USB.print(F("Setting Mode '1'.\t\t state "));
USB.println(e);

// select CRC on or off
e = sx1272.setCRC_ON();
USB.print(F("Setting CRC ON.\t\t\t state "));
USB.println(e);

// Select output power (Max, High or Low)
e = sx1272.setPower('M');
USB.print(F("Setting Power to 'M'.\t\t state "));
USB.println(e);

// Select the node address value: from 2 to 255
e = sx1272.setNodeAddress(9);
USB.print(F("Setting Node Address to '9'.\t state "));
USB.println(e);
USB.println();

delay(1000);

USB.println(F("-----"));
USB.println(F("Receiving from all nodes:"));
USB.println(F("-----"));
}

void loop()
{
  // Receive packets from any address as a sniffer
  sx1272.receiveAll(10000);

  // check rx status
  if( e == 0 )
  {
    USB.println(F("\nShow packet received: "));

    // show packet received
    sx1272.showReceivedPacket();
  }
}

```

```
}
else
{
    USB.print(F("\nReceiving packet TIMEOUT, state "));
    USB.println(e, DEC);
}
.....
}
```

Una unión de los dos últimos códigos permitiría hacer un sniffing más completo. Los paquetes cuentan con número de secuencia, dirección origen y destino, longitud en bytes, número de intentos y payload. Además, como se ve en el código del emisor, se puede incluir en el payload el porcentaje de batería del dispositivo y un timestamp. Sumando el hecho de que se puede obtener el RSSI, RSSI packet y SNR, se puede obtener mucha información sin necesidad de comprar ningún sniffer.

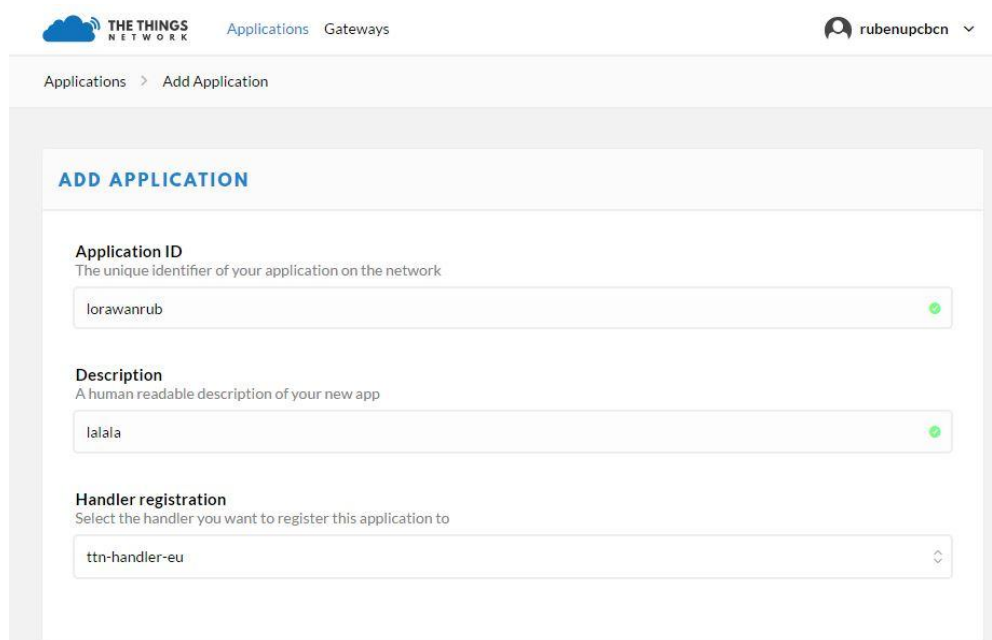
III Puesta a punto del gateway LoRaWAN

El gateway LoRaWAN utilizado es un modelo comercial, por lo que no requiere de una instalación completa como en el caso del gateway LoRa. Aún así explicaremos como configurar y ejecutar el programa *packet forwarder*.

El funcionamiento del gateway LoRaWAN es el siguiente. El gateway recibirá cualquier paquete que entre dentro de su radio de cobertura y lo retransmitirá al servidor que tenga habilitado. Si el servidor no dispone de una aplicación con las claves correspondientes, descarta el paquete. En caso contrario lo acepta y establece una comunicación normal.

En este caso se explicará cómo preparar una red LoRaWAN utilizando un servidor de The Things Network (TTN).

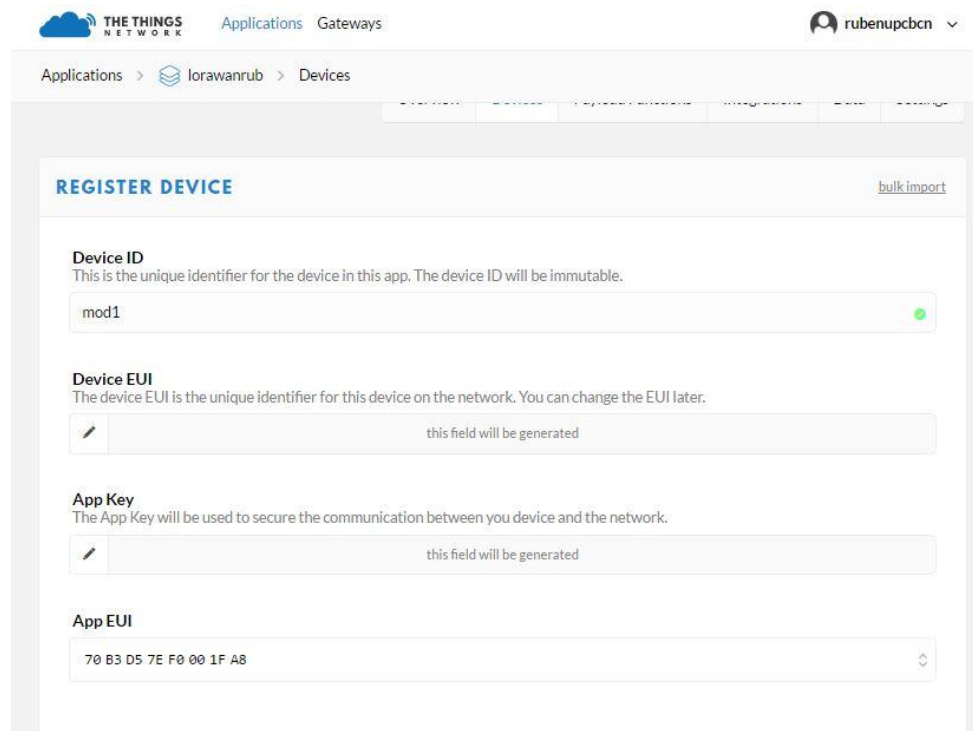
Lo primero es entrar en la web de TheThingsNetwork (<https://www.thethingsnetwork.org/>) y registrarnos en el caso de no disponer de una cuenta. Una vez efectuado el login o el registro, nos vamos a la pantalla de consola y creamos una nueva app (**Fig. III.1**).



The screenshot shows the 'ADD APPLICATION' form in the The Things Network web interface. The breadcrumb trail is 'Applications > Add Application'. The form has three main sections: 'Application ID' with a text input containing 'lorawanrub', 'Description' with a text input containing 'lalala', and 'Handler registration' with a dropdown menu showing 'ttn-handler-eu'. Each input field has a green checkmark icon on the right, indicating it is valid.

Fig. III.1 Configuración gateway LoRaWAN - Crear app

Únicamente debemos darle un identificador y una descripción. La aplicación se creará con este sencillo paso. Lo siguiente será registrar un dispositivo para esa aplicación (**Fig. III.2**).



The screenshot shows the 'REGISTER DEVICE' form in the The Things Network web interface. The breadcrumb trail is 'Applications > lorawanrub > Devices'. The form has four main sections: 'Device ID' with a text input containing 'mod1', 'Device EUI' with a text input containing 'this field will be generated', 'App Key' with a text input containing 'this field will be generated', and 'App EUI' with a text input containing '70 B3 D5 7E F0 00 1F A8'. Each input field has a green checkmark icon on the right, indicating it is valid. There is a 'bulk import' link in the top right corner of the form.

Fig. III.2 Configuración gateway LoRaWAN - Registrar dispositivo

De la misma manera solo debemos darle un identificador y el resto de parámetros se crearán automáticamente. Aunque también existe la posibilidad de escribir las claves manualmente.

Por defecto, The Things Network crea los dispositivos con método de activación OTAA (**Fig. III.3**). Esto se puede cambiar en la pestaña Settings que se encuentra en la zona superior de la pantalla, a la derecha.

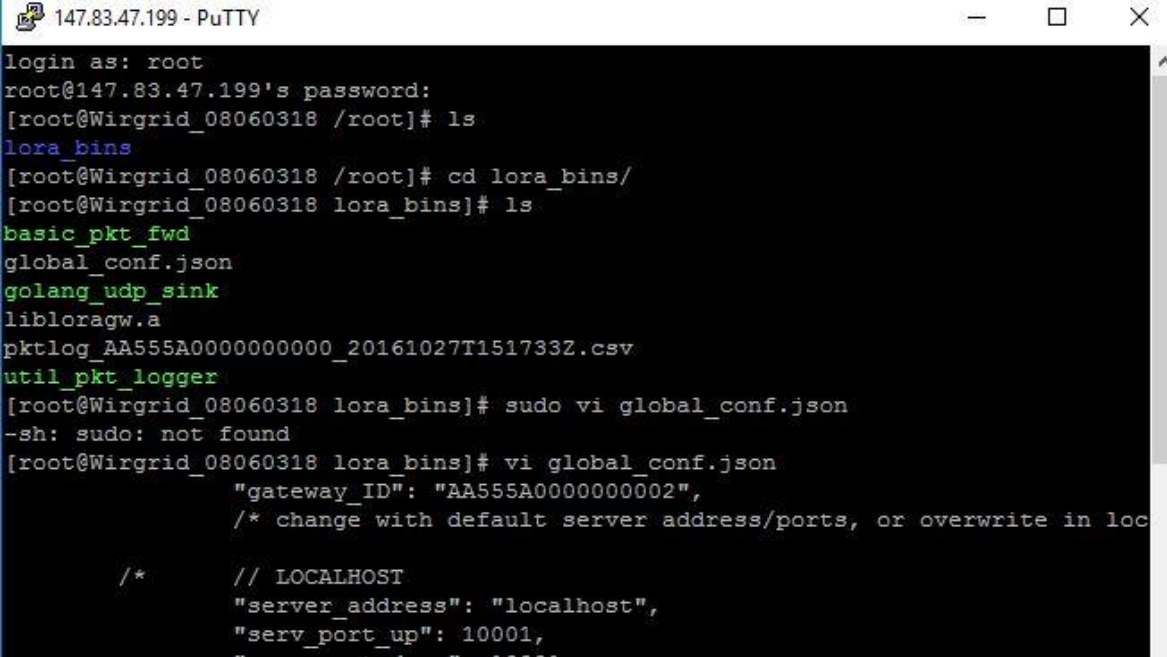
The screenshot shows the configuration page for a device in The Things Network. The breadcrumb navigation at the top reads: Applications > lorawanrub > Devices > lorawan2. The configuration fields are as follows:

- Application ID:** lorawanrub
- Device ID:** lorawan2
- Activation Method:** OTAA
- Device EUI:** 00 84 79 6B 54 8E 16 D2 (hex)
- Application EUI:** 70 B3 D5 7E F0 00 1F A8 (hex)
- App Key:** A field with 16 dots for manual entry, currently showing a masked view.
- Device Address:** 26 01 23 6C (hex)
- Status:** 2 minutes ago (indicated by a green dot)
- Frames up:** 1 (with a link to 'reset frame counters')
- Frames down:** 0

Fig. III.3 Configuración gateway LoRaWAN - Perfil de un dispositivo con sus claves

Las claves generadas se utilizarán en el Anexo IV, cuando configuremos los módulos LoRaWAN.

Mediante una sesión SSH, accedemos al gateway para ejecutar el packet forwarder (**Fig. III.4**), pero no sin habilitar antes el servidor que queremos utilizar (**Fig. III.5**).



```

147.83.47.199 - PuTTY
login as: root
root@147.83.47.199's password:
[root@Wirgrid_08060318 /root]# ls
lora_bins
[root@Wirgrid_08060318 /root]# cd lora_bins/
[root@Wirgrid_08060318 lora_bins]# ls
basic_pkt_fwd
global_conf.json
golang_udp_sink
libloragw.a
pktlog_AA555A0000000000_20161027T151733Z.csv
util_pkt_logger
[root@Wirgrid_08060318 lora_bins]# sudo vi global_conf.json
-sh: sudo: not found
[root@Wirgrid_08060318 lora_bins]# vi global_conf.json
    "gateway_ID": "AA555A0000000002",
    /* change with default server address/ports, or overwrite in loc

/*      // LOCALHOST
    "server_address": "localhost",
    "serv_port_up": 10001,
    "serv_port_down": 10002

```

Fig. III.4 Configuración gateway LoRaWAN - Configurar Packet_forwarder

'pkt_fwd.py':

```

" SX1301_conf": {
    "lorawan_public": true,
    "clksrc": 1, /* radio_1 provides clock to concentrator */
    "radio_0": {
        "enable": true,
        "type": "SX1257",
        "freq": 867500000,
        "rssi_offset": -166.0,
        "tx_enable": true
    },
    "radio_1": {
        "enable": true,
        "type": "SX1257",
        "freq": 868500000,
        "rssi_offset": -166.0,
        "tx_enable": false
    },
    "chan_multiSF_0": {
        /* Lora MAC channel, 125kHz, all SF, 868.1 MHz */
        "enable": true,
        "radio": 1,
        "if": -400000
    },
    "chan_multiSF_1": {
        /* Lora MAC channel, 125kHz, all SF, 868.3 MHz */
        "enable": true,
        "radio": 1,
        "if": -200000
    },
}

```

```
"chan_multiSF_2": {
    /* Lora MAC channel, 125kHz, all SF, 868.5 MHz */
    "enable": true,
    "radio": 1,
    "if": 0
},
"chan_multiSF_3": {
    /* Lora MAC channel, 125kHz, all SF, 867.1 MHz */
    "enable": true,
    "radio": 0,
    "if": -400000
},
"chan_multiSF_4": {
    /* Lora MAC channel, 125kHz, all SF, 867.3 MHz */
    "enable": true,
    "radio": 0,
    "if": -200000
},
"chan_multiSF_5": {
    /* Lora MAC channel, 125kHz, all SF, 867.5 MHz */
    "enable": true,
    "radio": 0,
    "if": 0
},
"chan_multiSF_6": {
    /* Lora MAC channel, 125kHz, all SF, 867.7 MHz */
    "enable": true,
    "radio": 0,
    "if": 200000
},
"chan_multiSF_7": {
    /* Lora MAC channel, 125kHz, all SF, 867.9 MHz */
    "enable": true,
    "radio": 0,
    "if": 400000
},
"chan_Lora_std": {
    /* Lora MAC channel, 250kHz, SF7, 868.3 MHz */
    "enable": true,
    "radio": 1,
    "if": -200000,
    "bandwidth": 250000,
    "spread_factor": 7
},
"chan_FSK": {
    /* FSK 50kbps channel, 868.8 MHz */
    "enable": true,
    "radio": 1,
    "if": 300000,
    "bandwidth": 125000,
    "datarate": 50000
},
"tx_lut_0": {
    /* TX gain table, index 0 */
    "pa_gain": 1,
```

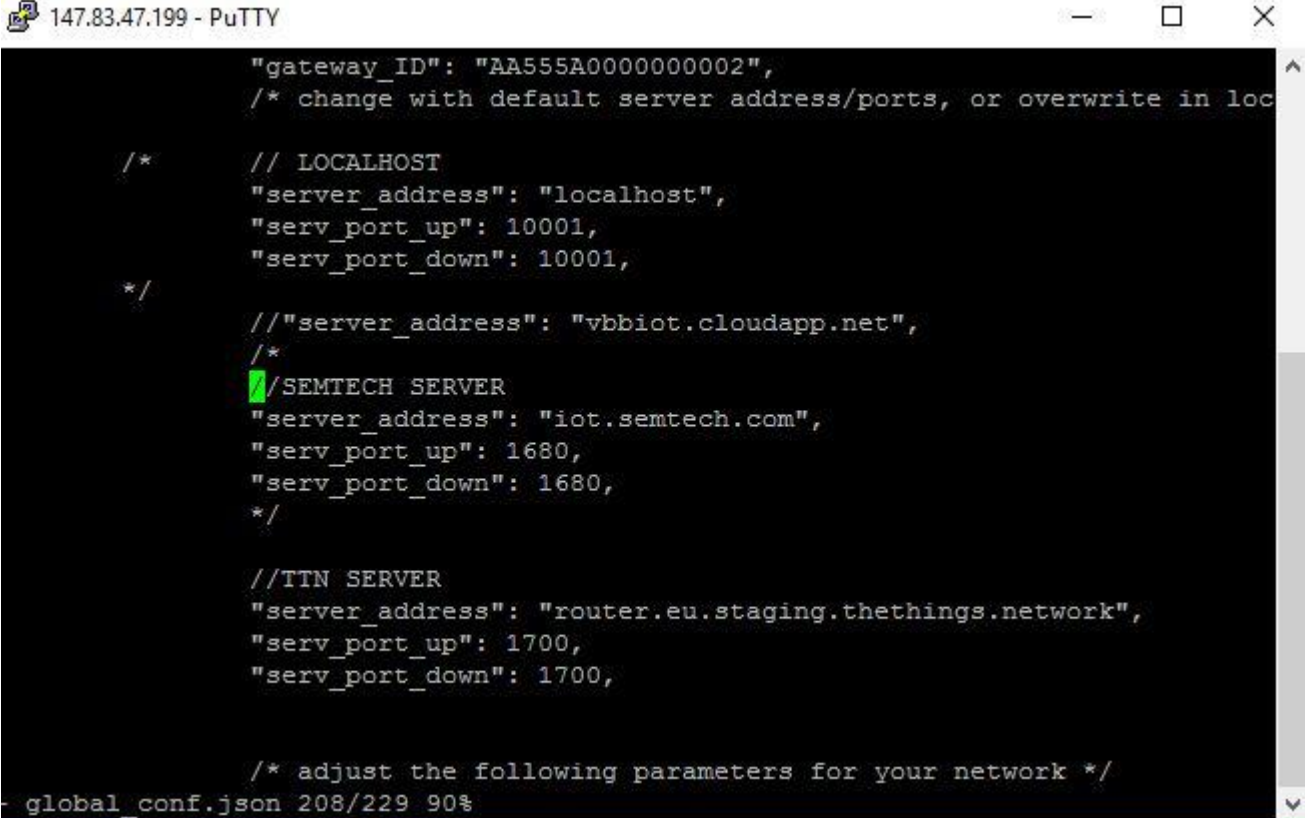


```
        "mix_gain": 8,  
        "rf_power": 0,  
        "dig_gain": 3  
    },  
    "tx_lut_1": {  
        /* TX gain table, index 1 */  
        "pa_gain": 1,  
        "mix_gain": 9,  
        "rf_power": 1,  
        "dig_gain": 3  
    },  
    "tx_lut_2": {  
        /* TX gain table, index 2 */  
        "pa_gain": 1,  
        "mix_gain": 10,  
        "rf_power": 3,  
        "dig_gain": 3  
    },  
    "tx_lut_3": {  
        /* TX gain table, index 3 */  
        "pa_gain": 1,  
        "mix_gain": 11,  
        "rf_power": 5,  
        "dig_gain": 3  
    },  
    "tx_lut_4": {  
        /* TX gain table, index 4 */  
        "pa_gain": 1,  
        "mix_gain": 13,  
        "rf_power": 8,  
        "dig_gain": 3  
    },  
    "tx_lut_5": {  
        /* TX gain table, index 5 */  
        "pa_gain": 2,  
        "mix_gain": 9,  
        "rf_power": 10,  
        "dig_gain": 3  
    },  
    "tx_lut_6": {  
        /* TX gain table, index 6 */  
        "pa_gain": 2,  
        "mix_gain": 10,  
        "rf_power": 12,  
        "dig_gain": 3  
    },  
    "tx_lut_7": {  
        /* TX gain table, index 7 */  
        "pa_gain": 2,  
        "mix_gain": 11,  
        "rf_power": 14,  
        "dig_gain": 3  
    },  
    "tx_lut_8": {  
        /* TX gain table, index 8 */
```

```
        "pa_gain": 2,  
        "mix_gain": 13,  
        "rf_power": 17,  
        "dig_gain": 3  
    },  
    "tx_lut_9": {  
        /* TX gain table, index 9 */  
        "pa_gain": 3,  
        "mix_gain": 9,  
        "rf_power": 19,  
        "dig_gain": 3  
    },  
    "tx_lut_10": {  
        /* TX gain table, index 10 */  
        "pa_gain": 3,  
        "mix_gain": 10,  
        "rf_power": 21,  
        "dig_gain": 3  
    },  
    "tx_lut_11": {  
        /* TX gain table, index 11 */  
        "pa_gain": 3,  
        "mix_gain": 11,  
        "rf_power": 23,  
        "dig_gain": 3  
    },  
    "tx_lut_12": {  
        /* TX gain table, index 12 */  
        "pa_gain": 3,  
        "mix_gain": 12,  
        "rf_power": 25,  
        "dig_gain": 3  
    },  
    "tx_lut_13": {  
        /* TX gain table, index 13 */  
        "pa_gain": 3,  
        "mix_gain": 13,  
        "rf_power": 26,  
        "dig_gain": 3  
    },  
    "tx_lut_14": {  
        /* TX gain table, index 14 */  
        "pa_gain": 3,  
        "mix_gain": 14,  
        "rf_power": 28,  
        "dig_gain": 3  
    },  
    "tx_lut_15": {  
        /* TX gain table, index 15 */  
        "pa_gain": 3,  
        "mix_gain": 15,  
        "rf_power": 29,  
        "dig_gain": 3  
    }  
},
```

```
"gateway_conf": {  
  "gateway_ID": "AA555A0000000002",  
  /* change with default server address/ports, or overwrite in loc  
  
  // LOCALHOST  
  "server_address": "localhost",  
  "serv_port_up": 10001,  
  "serv_port_down": 10001,  
  
  /*  
  // "server_address": "vbbiot.cloudapp.net",  
  
  /*  
  // SEMTECH SERVER  
  "server_address": "iot.semtech.com",  
  "serv_port_up": 1680,  
  "serv_port_down": 1680,  
  */  
  
  /*  
  // TTN SERVER  
  "server_address": "router.eu.staging.thethings.network",  
  "serv_port_up": 1700,  
  "serv_port_down": 1700,  
  */  
  
  /* adjust the following parameters for your network */  
  "keepalive_interval": 10,  
  "stat_interval": 30,  
  "push_timeout_ms": 100,  
  /* forward only valid packets */  
  "forward_crc_valid": true,  
  "forward_crc_error": true,  
  "forward_crc_disabled": false  
}
```

Anteriormente ya se mencionó que este gateway no llegaba a soportar los 15 posibles canales LoRaWAN, sino que solo disponía de 8. Aún así, podemos ver en el archivo `pkt_fwd` que solo hay 7 habilitados con sus respectivas configuraciones.



```
"gateway_ID": "AA555A0000000002",
/* change with default server address/ports, or overwrite in loc

/* // LOCALHOST
"server_address": "localhost",
"serv_port_up": 10001,
"serv_port_down": 10001,
*/

/* // "server_address": "vbbiot.cloudapp.net",
/*
//SEMTECH SERVER
"server_address": "iot.semtech.com",
"serv_port_up": 1680,
"serv_port_down": 1680,
*/

//TTN SERVER
"server_address": "router.eu.staging.thethings.network",
"serv_port_up": 1700,
"serv_port_down": 1700,

/* adjust the following parameters for your network */
global_conf.json 208/229 90%
```

Fig. III.5 Configuración gateway LoRaWAN - Habilitar servidor para programa packet_forwarder

Como se puede ver al final del script contamos con tres servidores aunque solo funciona si utilizamos uno, en este caso el de The Things Network.

Lo siguiente será configurar los módulos LoRaWAN y comprobar que hay comunicación con el gateway.

IV Configuración de los módulos LoRaWAN

La configuración de los módulos LoRaWAN se efectúa enviándole comandos UART a través del puerto serie. Nosotros lo haremos utilizando un cable USB y el programa Docklight.

Para establecer una comunicación serie con un módulo RN2483 debemos seguir la siguiente configuración:

Tabla IV.1 Configuración de la comunicación RN2483 - PC por cable serial

Especificación	Descripción
Baud Rate	57600 bps
Packet Length	8 bit
Parity Bit	No
Stop Bits	1 bit
Hardware Flow Control	No

Podemos configurar el módulo de dos formas dependiendo del modo de activación.

Configuración ABP:

```

sys reset
mac reset 868
mac set rx2 3 869525000
mac set nwkskey [NETWORK SESSION KEY]
mac set appskey [APP SESSION KEY]
mac set devaddr [DEVICE ADDRESS]
mac set adr off
mac set ar off
mac set pwridx 1 (14 dBm)
mac set dr [X] (datarate entre 0 y 5)
mac save
mac join abp
mac tx uncnf/cnf port data (cnf si queremos confirmación o uncnf en caso contrario)

```

Configuración OTAA:

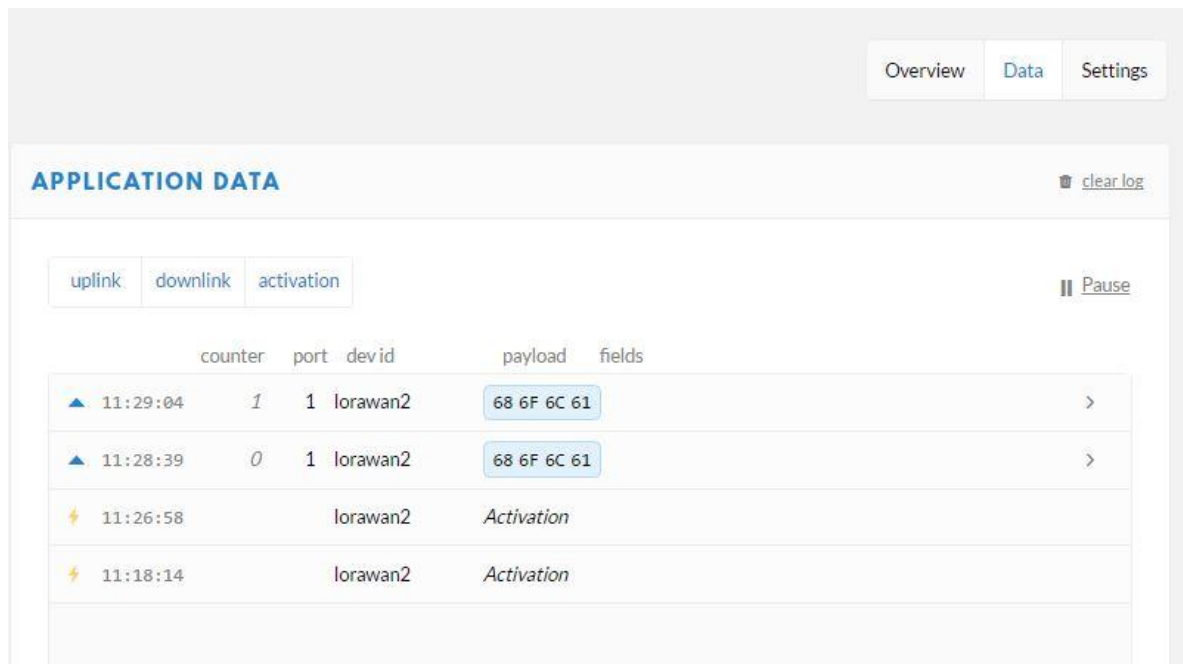
```

sys reset
mac reset 868
mac set rx2 3 869525000 (for TTN server)
mac set appeui [APPLICATION EUI]
mac set appkey [APPLICATION KEY]
mac set deveui [DEVICE EUI]
mac set adr off
mac set pwridx 1
mac save
mac join otaa
mac tx uncnf/cnf [PORT] [DATA] (puerto 1, el campo data debe ir en hexadecimal)

```

Los parámetros *nwkskey*, *appskey* y *devaddr* deberán ser los mismos que los proporcionados por la aplicación del servidor en caso de registrar un dispositivo ABP. En caso de registrar el dispositivo como OTAA se requerirán los parámetros *appeui*, *appkey* y *deveui*.

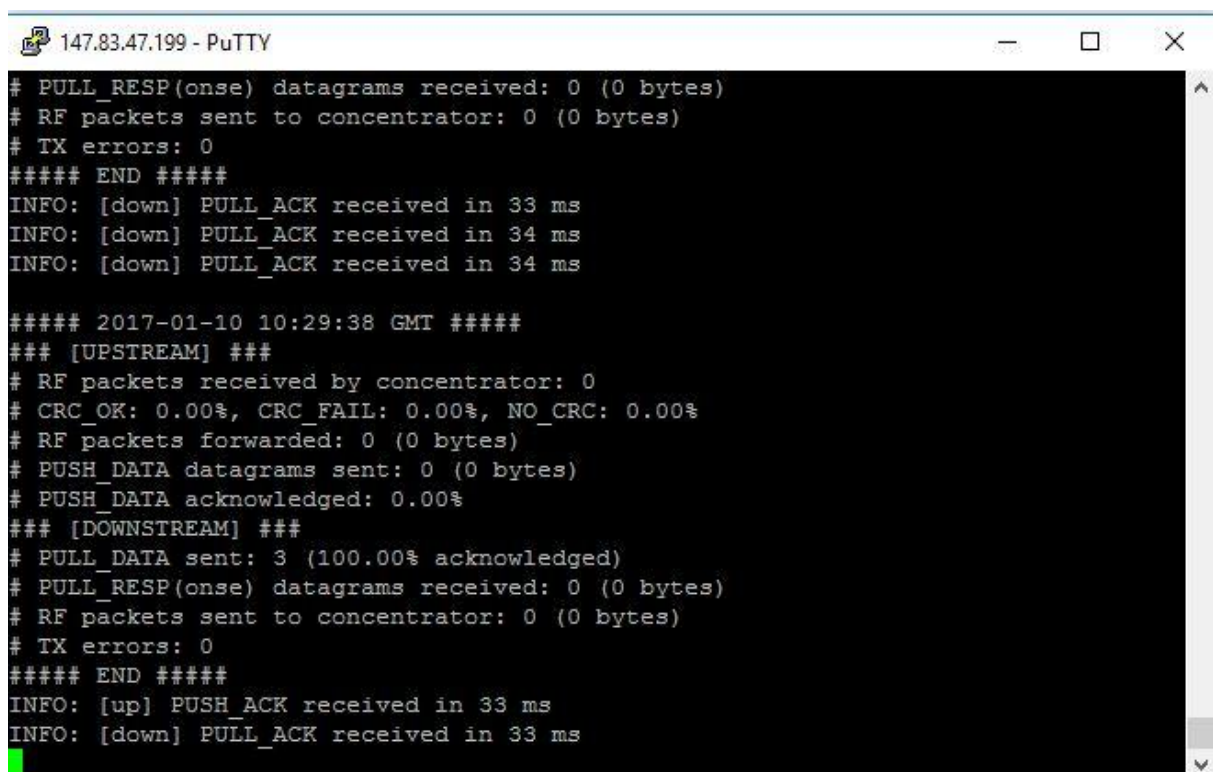
La activación de los dispositivos y los mensajes recibidos por el servidor podrán verse en la pestaña DATA de la consola del servidor utilizado (**Fig. IV.1**).



	counter	port	devid	payload	fields
▲ 11:29:04	1	1	lorawan2	68 6F 6C 61	>
▲ 11:28:39	0	1	lorawan2	68 6F 6C 61	>
⚡ 11:26:58			lorawan2	Activation	
⚡ 11:18:14			lorawan2	Activation	

Fig. IV.1 Paquetes LoRaWAN entrantes al servidor de red

Y también podremos verlos en la consola del gateway mientras ejecutamos el packet forwarder (**Fig. IV.2**).



```
# PULL_RESP(onse) datagrams received: 0 (0 bytes)
# RF packets sent to concentrator: 0 (0 bytes)
# TX errors: 0
##### END #####
INFO: [down] PULL_ACK received in 33 ms
INFO: [down] PULL_ACK received in 34 ms
INFO: [down] PULL_ACK received in 34 ms

##### 2017-01-10 10:29:38 GMT #####
### [UPSTREAM] ###
# RF packets received by concentrator: 0
# CRC_OK: 0.00%, CRC_FAIL: 0.00%, NO_CRC: 0.00%
# RF packets forwarded: 0 (0 bytes)
# PUSH_DATA datagrams sent: 0 (0 bytes)
# PUSH_DATA acknowledged: 0.00%
### [DOWNSTREAM] ###
# PULL_DATA sent: 3 (100.00% acknowledged)
# PULL_RESP(onse) datagrams received: 0 (0 bytes)
# RF packets sent to concentrator: 0 (0 bytes)
# TX errors: 0
##### END #####
INFO: [up] PUSH_ACK received in 33 ms
INFO: [down] PULL_ACK received in 33 ms
```

Fig. IV.2 Paquetes LoRaWAN entrantes al gateway

V Montaje para realizar las pruebas de consumo en LoRaWAN

En el capítulo 3 se realizaron unas pruebas de consumo para los módulos LoRaWAN. Los módulos utilizados en estas pruebas son los RN2483-PICtail vistos en el capítulo 2.

El módulo requiere de conexión USB para configurarlo, enviando comandos a través del puerto serial. El problema de medir el consumo sin más, es que los datos se verían contaminados por la alimentación del propio PC por el cable USB. Para solucionar el tema de la alimentación se optó por alimentarlo por las patas del PICtail desde el analizador de potencia y se utilizó un adaptador Serial a TTL (**Fig. V.1**) para conectar el módulo al PC sin que este lo alimente.



Fig. V.1 Adaptador Serial a TTL

Primero de todo se conectaron los pins del módulo al adaptador Serial a TTL. Los pins necesarios para alimentar el dispositivo y permitir el uso de comandos desde el PC son el PT Module RX (17) para recibir los comandos que le envía el PC, Module TX (21) para enviar respuestas al PC, +3V3 (26) para alimentar el módulo desde el analizador de potencia y el GND (28) que se corresponde con el pin de tierra. La ubicación de estos pins en el módulo RN2483-PICtail se puede comprobar en la **Fig. V.2**.

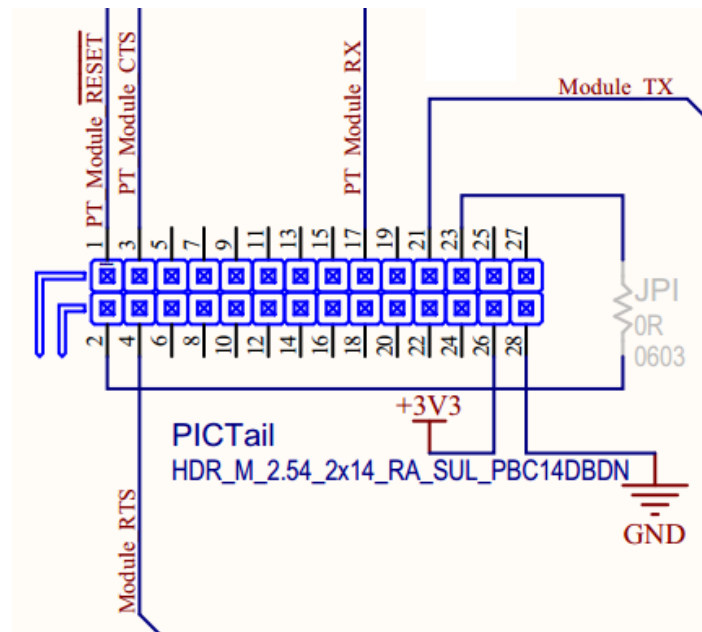


Fig. V.2 Pinout del módulo RN2483-PICtail

Desde el analizador de potencia se conecta el cable de alimentación hasta el conector Serial a TTL para que este alimente al módulo.

Antes de realizar la última conexión con el analizador de potencia es necesario comprobar la configuración de la fuente de alimentación (**Fig. V.3**). Al poner en marcha el analizador de potencia seleccionamos una de las cuatro fuentes de alimentación de las que dispone. Ajustamos la fuente seleccionada a una salida de 3.3V y limitamos la corriente a 0.5A.

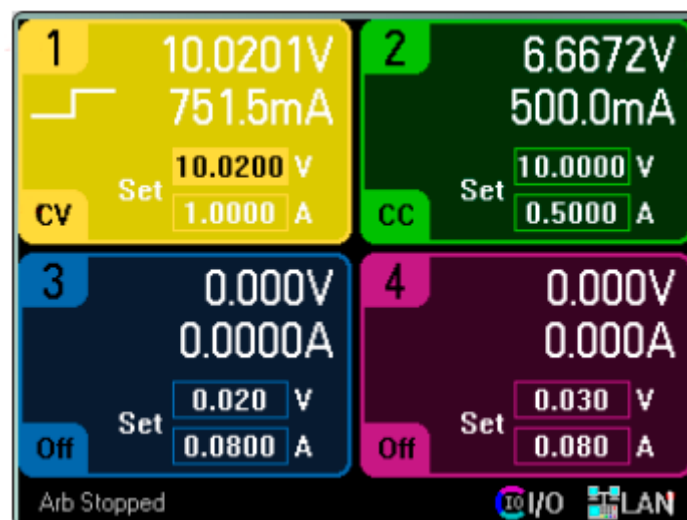


Fig. V.3 Configuración de la fuente de alimentación

Con la fuente configurada correctamente podemos conectar el módulo LoRaWAN y comprobar con el PC que recibe comandos haciendo uso del Cútecom.

El siguiente paso es seleccionar los datos que se quieren capturar con el analizador de potencia. Para ello entramos en el menú pulsando el botón Menu y seleccionamos la opción Datalogger (**Fig. V.4**). El Datalogger permite medir el consumo en un periodo de tiempo y analizar cualquier momento de esa captura con dos punteros que indican en pantalla el valor sobre el que se sitúan.

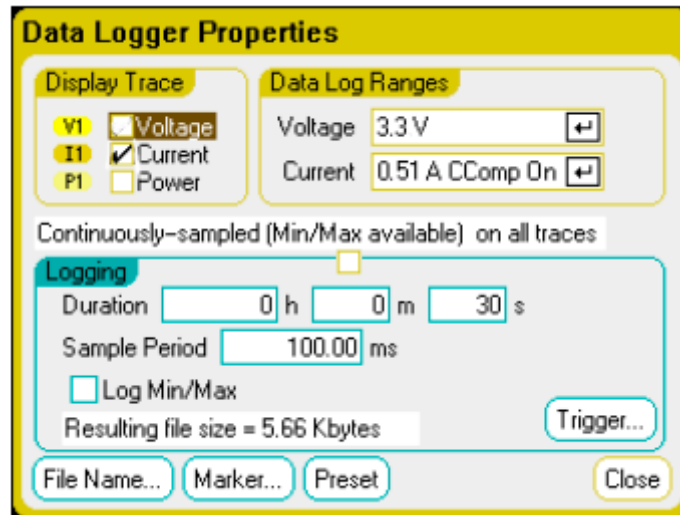


Fig. V.4 Ventana de configuración del datalogger

Para iniciar la captura se debe activar el botón Data Logger seguido del botón Run/Stop para comenzar o detener la captura. Con la captura pausada, se pueden utilizar los cursores para navegar por la pantalla sobre la cual aún se visualizarán los datos capturados.

El analizador de potencia Agilent N6705A tiene la opción de guardar capturas de pantalla en un dispositivo de memoria USB.