ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA Y SISTEMAS DE TELECOMUNICACIÓN

PROYECTO FIN DE GRADO

TÍTULO: Diseño de una arquitectura de WSN basada en NB-IoT para el Campus Sur de la UPM

AUTOR: José María Gutiérrez García

TUTOR (o Director en su caso): Fernando Pescador del Oso

TITULACIÓN: Sistemas de Telecomunicación

DEPARTAMENTO: de Ingeniería Telemática y Electrónica.

VºBº

Miembros del Tribunal Calificador:

PRESIDENTA: Dª Bozena Wislocka Breit

VOCAL: D. Fernando Pescador del Oso

SECRETARIO: D. Hugo Alexer Parada Gélvez

Fecha de lectura: **de** julio **de 2022**

Calificación:

El Secretario,

Agradecimientos

Agradezco su ayuda a mis tutores Rubén y Fernando. Y agradezco a mi familia su apoyo y confianza.

Resumen

El presente PFG evalúa las características de la comunicación **NB-IoT** y propone una arquitectura de **WSN** (Wireless Sensor Network) basada en NB-IoT (Narrow Band-Internet of Things) para el Campus Sur de la **UPM** (Universidad Politécnica de Madrid). Los cambios en la legislación de algunas industrias como la telemetría, con los contadores de agua, o la geolocalización, con las balizas de emergencia, han disparado la necesidad en el mercado de dispositivos de comunicación de larga distancia y bajo consumo. Las *releases* 13, 14 y 15 del estándar 3GPP satisfacen dichas necesidades estandarizando dispositivos simplificados y mecanismos de ahorro de energía, mejorando cobertura en interiores y aumentando la potencia disponible, entre otras. En este ámbito, la tecnología NB-IoT se muestra como una tecnología de largo alcance y bajo consumo, que funciona en el espectro licenciado, asegurando seguridad y fiabilidad. La tecnología NB-IoT se encuentra en el porfolio de las grandes operadoras de telecomunicación, que han adherido esta tecnología a su red mediante hardware añadido a sus estaciones base servidoras de **LTE** por la parte de acceso y, en la mayoría de los casos, mediante la utilización de la banda de guarda de la señal LTE por la parte del espectro radioeléctrico. El prototipo propuesto contiene 2 modelos de motas sensorizadas con conectividad NB-IoT disponibles en el mercado: **SODAQ SARA** y **Libelium WASPMOTE**. Este proyecto diseña un prototipo de sistema de recogida de medidas para definir la propuesta. Para dar conectividad a las motas se utilizan SIM de Vodafone. Se proponen diferentes arquitecturas de software de gestión de datos mediante la utilización de datagramas **UDP** y conexiones **MQTT**. Los resultados muestran que las arquitecturas propuestas son viables y cada una se ajusta a unos requisitos de red determinados. Mediante el sistema de recogida de medidas se valida la arquitectura y las motas propuestas para el prototipo. Se visualizan los datos recogidos en el prototipo mediante una herramienta de visualización ubicada en el servidor de red **ThingSpeak**. Los resultados ilustran que el Campus sur de la UPM dispone de cobertura NB-IoT suficiente para realizar un despliegue de motas mediante la estructura propuesta. El prototipo propuesto es escalable a una solución que controle mayor número de motas sin perjuicio de la red.

Palabras clave: NB-IoT, WSN, UPM, SODAQ, SARA, LTE, Libelium, WASPMOTE, UDP, MQTT, ThingSpeak

Abstract

This PFG evaluates the characteristics of **NB-IoT** communication and proposes a **WSN** (Wireless Sensor Network) architecture based on NB-IoT (Narrow Band-Internet of Things) for the Campus Sur de la Universidad Politécnica de Madrid ("**UPM** Campus Sur"). The change in the legislation of some industries such as telemetry (in relation to water meters), or geolocation (with respect to emergency beacons) have raised the need in the market for long distance and low consumption communication devices. The releases 13, 14 and 15 of the 3GPP standard address these needs, among others, by standardizing simplified devices and energy saving mechanisms and improving indoor coverage and increasing available power. In this area, NB-IoT technology has proven to be a long-range, low-power technology, operating in licensed spectrum that is able to ensure security and reliability. NB-IoT technology is currently in the portfolio of large telecommunication operators, which have added this technology to their network by adding hardware to their **LTE** base stations on the access side and, in most cases, by using the LTE guard band on the radio spectrum side. The proposed prototype contains 2 models of sensor motes with NB-IoT connectivity available on the market: **SODAQ SARA** and **Libelium WASPMOTE**. A prototype measurement collection system is hereby designed to define the proposal. Vodafone SIMs are used to provide connectivity to the motes. Several software architectures for data management using **UDP** datagrams and **MQTT** connections are also proposed. The results show that the proposed architectures are feasible and each one of them has specific network requirements. The measurement collection system is used to validate the architecture and the proposed motes for the prototype. The data collected on the prototype is visualised using a visualisation tool located on the **ThingSpeak** network server. The results illustrate that the UPM Campus Sur has sufficient NB-IoT coverage to deploy motes using the proposed structure. The proposed prototype is scalable to a solution that controls a larger number of motes without affecting the network.

Keywords: **NB-IoT, WSN, UPM, SODAQ, SARA, LTE, Libelium, WASPMOTE, UDP, MQTT, ThingSpeak**

Tabla de contenidos

Listado de figuras

Listado de tablas

Listado de acrónimos

AES	Advanced Encryption Standard
AFF	Arduino Form Factor
API	Application Programming Interface
APN	Access Point Name
CDMA	Code Division Multiple Access
CSS	Chirp Spread Spectrum
DCI	Downlink Control Information
eDRX	Extended Discontinuous Reception
FDD	Frequency Duplex Division
FDMA	Frequency Division Multiple Access
GMSK	Gaussian Minimum Shift Keying
GSM	Global System for Mobile Communications
HARQ	Hybrid Automatic Repeat Request
H2H	Human to Human
IEEE	Institute of Electrical and Electronics Engineers
LoRa	Long Range
LoS	Line of Sight
LTE	Long Term Evolution
MAC	Medium Access Control
MCL	Maximum Coupling Loss
MME	Mobility Management Entity
MTC	Machine Type Communications
M2M	Machine to Machine
NAS	Non-Access Stratum
NCCE	Narrowband Control Channel Element
NIDD	Non IP Data Delivery
NPBCH	Narrowband Physical Broadcast Channel
NPDCCH	Narrowband Physical Downlink Control Channel
NPDSCH	Narrowband Physical Downlink Shared Channel
NPRACH	Narrowband Physical Random Access Channel
NPSS	Narrowband Secondary Synchronization Signal
NPUSCH	Narrowband Physical Uplink Shared Channel
NRS	Narrow Reference Signal
NSSS	Narrowband Secondary Synchronization Signal
PDCP	Packet Data Convergence Protocol
PHY	Referido a la capa física
PRACH	Physical Random Access Channel
PSD	Power Spectral Density
PSM	Power Saving Mode
RE	Resource Element

RLC	Radio Link Control
RRC	Radio Resource Control
RSRP	Reference Signal Received Power
RU	Resource Unit
SC-FDMA	Single Carrier Frequency Division Multiple Access
SFN	System Frame Number
TDMA	Time Division Multiple Access
ToA	Time of Arrival
UE	User Equipment

1 Introducción

Tras la publicación de las *releases* 13, 14 y 15 del 3GPP, se definen las tecnologías necesarias para el desarrollo de las redes LPWAN (Low Power Wide Area Network). Las redes LPWAN tienen un rango de varios kilómetros y gestionan un gran número de dispositivos de bajo consumo. La gran ventaja aportada a estas redes a través del estándar 3GPP es la posibilidad de un solo despliegue de dispositivos cada varios años debido a la larga duración de la batería de los dispositivos utilizados. Esta ventaja se obtiene mediante mecanismos de ahorro de energía estandarizados en las releases del 3GPP, como son el eDRX o el modo PSM. Una de las tecnologías de acceso radio aplicable a estas redes es la tecnología NB-IoT, que está orientada a las comunicaciones entre dispositivos de forma eficiente y hace uso del espectro licenciado, lo que asegura su vigencia a lo largo del tiempo.

Con todo, la tecnología NB-IoT constituye una buena alternativa para llevar a cabo el despliegue de grandes redes de dispositivos ahorrando en mantenimiento y en actualizaciones. Estas redes serían capaces de gestionar los dispositivos y la información recogida por los dispositivos de forma segura y duradera.

1.1 Alcance y objetivos del proyecto

Este PFG tiene como meta la demostración de una red LPWAN ubicada en el Campus Sur de la UPM. La red demostrada es capaz de recoger información a lo largo del campus mediante motas sensorizadas.

Se pretende hacer una valoración de las tecnologías NB-IoT que trabajan sobre el espectro licenciado, en concreto:

- Se diseñará una WSN (Wireless Sensor Network) en la que se habilitarán nodos y sensores que permitan hacer una propuesta de WSN.
- Se probarán funcionalidades aplicadas a WSN de dos placas de desarrollo disponibles en el mercado, la placa de desarrollo SODAQ SARA AFF y la placa de desarrollo Libelium WASPMOTE
- Se hará una comparativa entre las dos placas usando los servicios de un operador de red NB-IoT.

- Se diseñará una arquitectura software para la recogida de datos.

1.2 Estructura de la memoria

El trabajo realizado contiene una revisión de las tecnologías aplicables para las redes de control de dispositivos. Se comienza la memoria con la introducción a las redes LPWAN y sus posibilidades en combinación con la tecnología NB-IoT. Se sigue con la descripción en el Capítulo 2 de las tecnologías que anteceden a las utilizadas en la red propuesta, describiendo las mismas de forma cronológica en el apartado 2.1. Se describen los tipos de redes que se pueden implementar en función de su alcance y los dispositivos que integran haciendo hincapié en las redes LPWAN. Dentro de las tecnologías LPWAN, se realiza una diferenciación entre las tecnologías que usan el espectro licenciado y las que no lo usan, contraponiendo conceptos como la seguridad, la disponibilidad y la estandarización. Se describe con mayor nivel de detalle algunas tecnologías LPWAN.

Tras la contextualización de dichas tecnologías, se utiliza el apartado 2.2 para la descripción de la tecnología NB-IoT. Los epígrafes describen la arquitectura, los modos de operación, el Stack, la cobertura, el consumo, la topología y los dispositivos.

Para cerrar el contexto tecnológico, se analizan las aportaciones de las principales operadoras de telecomunicación en el campo del IoT y especialmente, en la tecnología NB-IoT, mostrando proyectos que desarrollan y mapas de cobertura en algún caso.

Luego, en el Capítulo 3 se definen las especificaciones del sistema propuesto detallando las placas de desarrollador, los servidores y el tipo de comunicación entre los elementos de la red. Así mismo, se definen las especificaciones del demostrador de red propuesto.

Tras definir las especificaciones, en el Capítulo 4 se detalla el diseño y la implementación de la propuesta de red. En el apartado de diseño se detalla la composición de la propuesta de prototipo de la WSN y las distintas alternativas que se analizan para desarrollarla. En el apartado de implementación se describen los pasos llevados a cabo para realizar la propuesta de diseño.

En el capítulo 5 se muestran los resultados obtenidos en las medidas de nivel de señal en el Campus Sur de la UPM.

En el capítulo 6 se desarrolla el presupuesto del proyecto.

En el capítulo 7 se presentan las conclusiones del proyecto y se proponen trabajos futuros para complementar el trabajo realizado.

Finalmente, en los anexos se incluye el código fuente utilizado en las motas y las distintas versiones de los servidores utilizados.

2 Marco tecnológico

2.1 Antecedentes tecnológicos

La primera generación de comunicaciones celulares comienza con sistemas de comunicación analógicos alrededor de 1980. Los sistemas utilizados se despliegan en bandas de frecuencia entre 450 MHz y 1GHz. El tamaño de las celdas oscila entre 2km y 40km. Además, dependiendo del sistema implementado, el ancho de banda ocupado es de 25kHz o 30kHz.

Seguidamente, la segunda generación [1] de comunicaciones celulares, caracterizada por la incorporación de servicios de voz basados en comunicaciones de datos a través de circuitos conmutados, ocupa las bandas de frecuencia entre 900MHz y 1.9GHz. Se incorporan las técnicas de acceso FDMA, TDMA, CDMA y FDD.

A partir del 2G se desarrollan 3 estándares: GSM, IS-54 e IS-95. Siendo GSM sobre el que se desarrolla la tecnología que analizaremos más adelante.

El estándar GSM está basado en FDMA/TDMA/FDD y modulación GMSK. El ancho de banda de cada canal es de 200 kHz dividido en intervalos de tiempo para 8 usuarios.

La generación 3G de comunicaciones define sus estándares en el 2000 a través de la norma IMT-2000. Trabaja en bandas comprendidas entre 450MHz y 2.1GHz, con un alcance de 50km en las bandas inferiores. En esta norma se integran 5 tipos de acceso radio: W-CDMA, CDMA 2000 1X, TD-SCDMA, EDGE y DECT.

La generación 3.9G se considera la siguiente generación y está compuesta por los estándares: Mobile WiMAX, 3GPP LTE, Ultra Mobile Broadband y IEEE 802.20 (Mobile-Fi).

En la *Release 8* del estándar 3GPP se instauran las bases del desarrollo de LTE [2]. Esta tecnología optimiza la conmutación de paquetes, se caracteriza por la alta velocidad de envío de paquetes y por su flexibilidad en frecuencia y ancho de banda. El direccionamiento de paquetes dentro de la red se realiza exclusivamente mediante direccionamiento IP.

La estructura de acceso LTE está compuesta por equipos de usuarios UE (User Equipment) y estaciones base dedicadas a la red LTE, llamadas eNB (enhanced Node

Base). La disposición de las estaciones permite un acceso descentralizado a la red que optimiza el establecimiento y el traspaso de información. Cada estación base, gestiona los UE mediante un organizador basado en solicitudes híbridas repetidas automáticamente (Hybrid Automatic Repeat Request, HARQ) enviadas por los UE. También, cada eNB identifica el estado de cada UE disponible en su rango de actuación, y planifica e informa los enlaces descendentes y ascendentes a cada UE.

La información transmitida a través de LTE se pauta mediante Intervalos Temporales de Transmisión (TTI) que duran 1 ms. Durante cada periodo de transmisión se establecen canales lógicos para el envío de información en los enlaces UL y DL organizados a través de cada eNB. Los recursos disponibles se fragmentan en bloques de datos como en la tecnología legada UMTS.

El ancho de banda ocupado por el acceso radio LTE varía desde 1.4 MHz hasta 20 MHz, destacando la eficiencia espectral. Los enlaces UL y DL transmiten en la misma banda mediante el uso de FDD y TDD. El enlace DL utiliza Divisiones en Banda Estrecha Ortogonales en Frecuencia (OFDM) que subdividen el espectro en subportadoras, identificadas como canales, que a su vez forman una estructura de datos. En el enlace DL se modula la señal mediante 16/64QAM o QPSK. Por otro lado, el enlace UL utiliza SC-FDMA, que genera una única portadora y satisface el requisito de menor potencia requerida al usuario y modula la señal mediante B/QPSK o 16/64QAM.

En la misma línea, LTE-Advanced (LTE-A) [3] se desarrolla sobre la tecnología LTE añadiendo mejoras que aumentan la capacidad de la tecnología legada. En la *Release 10* del estándar 3GPP se actualiza la tecnología mejorando los parámetros de: eficiencia espectral, tasas de subida y bajada de datos y rendimiento en los límites de las celdas.

Las tecnologías LTE/LTE-A inicialmente diseñadas para comunicaciones Humano a Humano (H2H) se han utilizado, y más tarde adaptado [4], para las comunicaciones Máquina a Máquina (M2M) llevadas a cabo por dispositivos pertenecientes a redes de la Internet de las Cosas (IoT).

En este ámbito, haciendo uso de las redes celulares, se encuentra el Internet de las Cosas (**IoT**). En el mismo, millones de dispositivos sensorizados transmiten información relativa a distintos sectores, como automoción, logística, o atención sanitaria. Para mejorar la comunicación entre dispositivos aparece el modo de Comunicaciones de Tipo Máquina Masivas (mMTCs).

En el marco de las comunicaciones IoT, la aplicación de la tecnología LTE/LTE-A supone un malgasto de recursos al tratarse de una comunicación que transmite más potencia de la necesaria en el ámbito IoT.

Para llevar a cabo la conexión de los dispositivos considerados del IoT, que son dispositivos sensorizados que transmiten una cantidad reducida de datos, se requieren tecnologías adaptadas a las mismas comunicaciones, como son las redes LPWA.

2.1.1 Tipos de redes celulares

Hay distintos tipos de redes celulares en función del rango de alcance, consumo de batería o tipo de transmisión de datos. Diferenciamos las siguientes tecnologías: LAN, WAN, PAN, LPWAN.

LAN

Las redes de área local LAN (Local Area Network) están compuestas por 2 o más dispositivos interconectados, que comparten el mismo ámbito en una red de medio alcance, con habitualmente al menos 1 acceso a internet. Los estándares de las redes LAN son 802.3 para enlaces físicos y 802.11 para enlaces inalámbricos.

El estándar 802.11 utiliza las bandas de 2,4GHz y 5 GHz alternativa o simultáneamente dependiendo de la versión del estándar. El ancho de banda ocupado es de 20MHz y sus múltiplos hasta 160MHz, utiliza modulaciones n-QAM, y se caracteriza por velocidades de transmisión de datos altas. Las sucesivas ediciones del estándar son: ‘b’, ‘a’, ‘g’, ‘n’, ‘ac’ y ‘ax’.

WAN

Las redes de área amplia WAN (Wide Area Network) están compuestas por varias redes LAN o por una red muy amplia, y tienen un tamaño indeterminado. Pueden ser de ámbito público o privado.

Las velocidades de transmisión de datos son menores y el tráfico en su interior se congestiona más que en las redes LAN.

PAN

Las redes de área personal PAN (Personal Area Network) se componen de 1 o varios dispositivos interconectados situados alrededor de una persona, hasta alrededor de 10m. Se define el estándar PAN en la IEEE 802.15.4. Algunas tecnologías inalámbricas de esta tecnología son: Bluetooth, THREAD, Zigbee, NFC, RFID.

Bluetooth [5] proporciona servicio en cortas distancias formando pequeñas celdas alrededor de cada usuario. Se trata de una tecnología de bajo consumo, que funciona en la banda de 2.4GHz, y tiene topología de estrella, sin comunicación entre dispositivos finales.

Zigbee [6] proporciona servicio a zonas comprendidas entre 10 y 100 metros. Se trata de una tecnología que utiliza módulos de pequeño tamaño, bajo coste y bajo consumo, que funciona en la banda de 868MHz en Europa, y puede trabajar con topología de estrella, malla o combinado.

NFC [7] es una tecnología derivada de RFID, que proporciona servicio hasta un rango máximo de 20cm. Se trata de una tecnología que proporciona alta velocidad de transmisión de datos, funciona en la banda de 13.56MHz, y comunica 2 dispositivos.

LPWAN

Las redes de área amplia y baja potencia **LPWAN** (Low Power Wide Area Network) se componen habitualmente de un gran número de dispositivos. Tiene un rango de varios kilómetros, sin embargo, desde el dispositivo emisor hasta el receptor puede no usarse el

mismo tipo de tecnología. En este sentido, el rango que alcanzan estas redes se debe en parte a la gran sensibilidad de hasta -130 dBm que disponen los receptores.

La velocidad de transmisión de datos es baja debido a que la tecnología se orienta a transmitir una reducida cantidad de datos, dado que los dispositivos emisores envían información relativa a sensores. Sin embargo, el volumen de información transmitido en la red puede ser muy alto debido a la gran cantidad de dispositivos que pueden componerla. Además, habitualmente la transmisión de datos se realiza de forma discontinua, lo que evita un establecimiento de la conexión permanente.

Las redes LPWAN están orientadas a que los dispositivos desplegados hagan uso de baterías portátiles. Debido a que los periodos de inactividad superan ampliamente los periodos de actividad, junto con diversas herramientas de ahorro de consumo, las baterías de los dispositivos duran varios años.

Los tipos de topología más utilizados son estrella y malla. La topología tipo estrella dispone la estación base en el centro de la red, y mediante una comunicación directa el dispositivo emisor envía los datos al dispositivo receptor. Por otro lado, la topología tipo malla dispone de una red de dispositivos emisores que, o bien, se comunican directamente con la estación base, o bien, se comunican con otros dispositivos que retransmiten la información recibida a la estación base u otros dispositivos. En este caso, la estación base se puede colocar en cualquier punto de la red.

Tomando la banda de radiofrecuencia como referencia, dentro de las tecnologías LPWAN se diferencian del **Espectro Licenciado** y del **Espectro no Licenciado**. Las tecnologías que trabajan en el espectro licenciado requieren pagos y licencias para transmitir a través del espectro radioeléctrico en una banda de frecuencias estipulada, por el contrario, las tecnologías que trabajan en el espectro no licenciado emiten en bandas de frecuencias determinadas, como las de 2,4GHz o 5GHz reconocidas mundialmente, con acceso gratuito, pero con limitaciones como la potencia de emisión o la duración de las señales.

2.1.2 Espectro no licenciado

Las tecnologías consideradas de espectro no licenciado usan en su mayoría la banda de radiofrecuencia ISM (Industrial S Medical), que trabaja en las bandas de 900 MHz, 2.4 GHz, y 5 GHz en el espectro radioeléctrico.

Las bandas ISM permiten su utilización sin el pago de licencias en caso de cumplir con especificaciones de esta. Las especificaciones requeridas incluyen la limitación de la potencia de emisión y la limitación del periodo de la señal, con el fin de limitar las interferencias que puedan ocasionar a otras señales. En esta línea, el principal inconveniente de dichas bandas son las interferencias producidas por el resto de las señales que funcionan en el mismo rango de frecuencias.

Habitualmente las tecnologías que hacen uso del espectro licenciado no están estandarizadas, y son desarrolladas por empresas independientes.

Algunas tecnologías que hacen uso del espectro no licenciado son: Weigthless, LoRaWAN y SigFox.

2.1.2.1 Weigthless

El estándar Weigthless [8] es la agrupación de varios estándares LPWAN orientados a diversos ámbitos del IoT. Está respaldado por la Weigthless Alliance. Opera en las bandas ISM, con despliegue en Europa en la banda de 868 MHz.

El estándar Weigthless proporciona un rango de aplicación de hasta 2 km en entornos urbanos. La comunicación se realiza a través de un ancho de banda 'ultra estrecho' de 12.5 kHz, y se modula mediante GMSK y QPSK acompañado de técnicas *Spread Spectrum*. Se aplican los accesos radio FDMA y TDMA. La velocidad de transmisión se adapta a los recursos de la red y varía entre 200 bps y 100 kbps.

Proporciona comunicación bidireccional con potencia adaptativa entre los dispositivos finales y el servidor central.

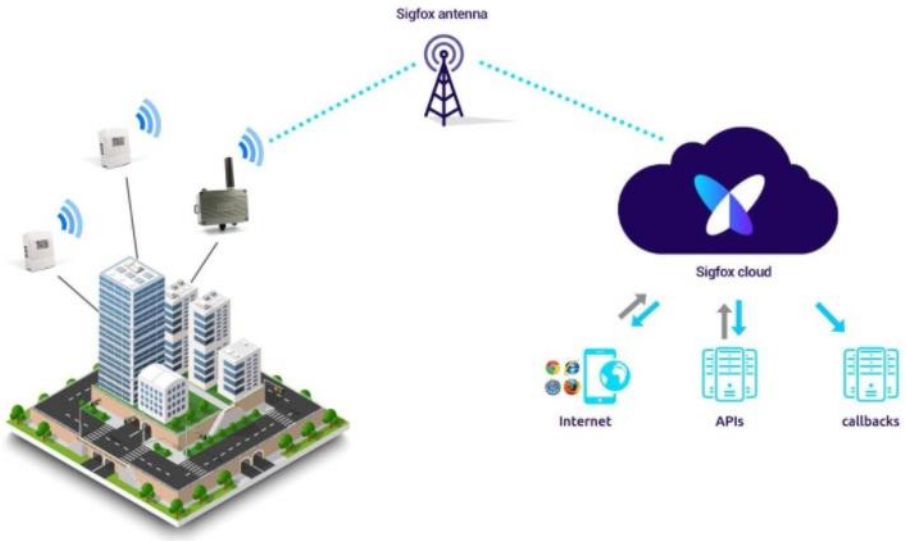
La seguridad en la plataforma se basa en encriptado AES128/256, y autenticación punto a punto.

La principal diferencia respecto a otras tecnologías LPWAN es que se trata de una tecnología orientada a una alta densidad de dispositivos finales y a una comunicación bidireccional más frecuente.

2.1.2.2 SigFox

La tecnología SigFox [9] irrumpe en el mercado como la primera tecnología LPWAN orientada al IoT. Opera en las bandas ISM, con despliegue en Europa en el rango de frecuencia entre 862 y 876 MHz.

La tecnología SigFox tiene un rango de aplicación de hasta 10 km por nodo en zonas urbanas y de 50 km por nodo en zonas rurales. La comunicación se realiza a través de un ancho de banda 'ultra estrecho', y se modula a través de GMSK y PSK. La velocidad de transmisión de datos está limitada 140 paquetes de 12 bytes enviados al día, lo que supone una velocidad de transmisión desde 0.3 a 50 kbps.



Fuente: https://www.sigfox.es

Figura 1.- Arquitectura de red de SigFox.

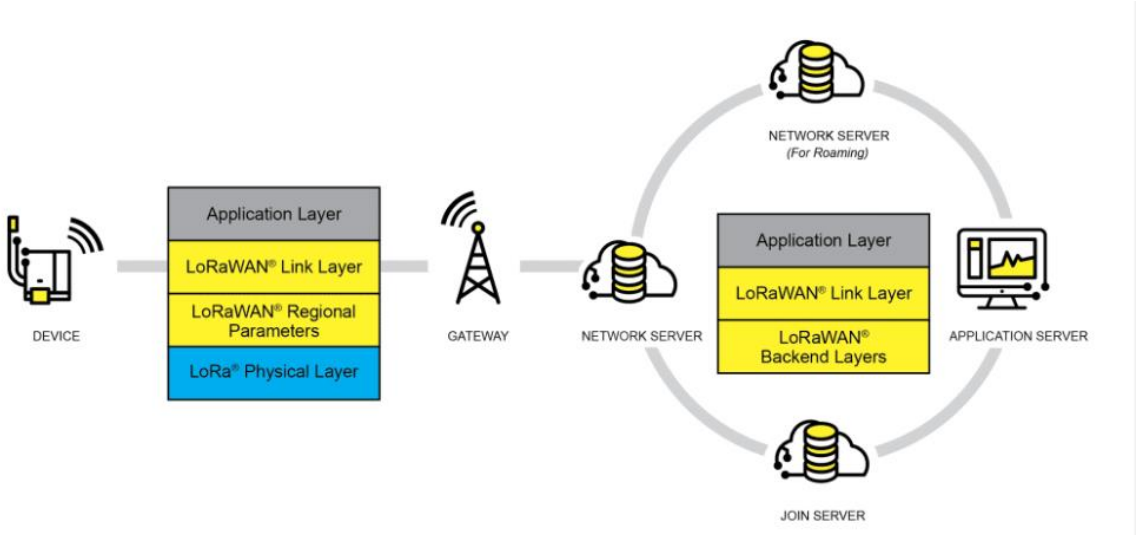
Tiene una comunicación bilateral y asíncrona entre dispositivo final y servidor central. La topología es con forma de estrella, de forma que los dispositivos finales se sitúan alrededor de una antena receptora que a su vez se conecta con la nube de SigFox, que proporciona datos al cliente final, como se muestra en la Figura 1. La nube de SigFox es una tecnología propietaria que requiere de pago para su uso.

La seguridad en la plataforma se basa en la autenticación mutua.

2.1.2.3 LoRaWAN

La especificación LoRaWAN [10] es un protocolo de red de bajo consumo y largo alcance (LPWAN), cuya capa física se desarrolla a través de LoRa, y cuya arquitectura se desarrolla a través de LoRa Alliance, como se muestra en la Figura 2.

La especificación LoRaWAN cubre las necesidades de los sistemas IoT [11] distribuidos a en amplias regiones, llegando a cubrir distancias comprendidas entre 10km y 20km. Ocupa el espacio de las tecnologías PAN en el ámbito de largo alcance, siendo compatible con algunas redes PAN como WiFi. Así mismo, la estructura típica de despliegue físico que maneja la especificación es una estrella de estrellas, en la cual los dispositivos finales se comunican con el servidor central a través de *gateways*, que transforman la señal RF recibida por los dispositivos en paquetes IP, y viceversa. Además, la retransmisión de los paquetes se realiza entre los dispositivos y entre los *gateways*.



Fuente: www.lora-alliance.org

Figura 2.- Arquitectura de red de LoRaWAN.

LoRa es la capa física para enlaces de larga distancia. LoRa utiliza una modulación de espectro ensanchado, similar a CSS (Chirp Spread Spectrum). Además, permite modular la cantidad de bits utilizados para transmitir cada símbolo, lo que modifica el tamaño del mensaje enviado, la velocidad de transmisión y la sensibilidad. Así mismo, el tamaño de la cabecera del datagrama varía entre 13 y 28 bytes, y el tamaño del *payload* del datagrama varía hasta los 255 bytes.

Los dispositivos disponen de distintos modos de servicio en función de su clase, sin embargo, todos tienen en común la comunicación bidireccional con el *gateway*, el envío de mensajes asíncrono, y 2 ventanas de escucha activa en un momento determinado.

La frecuencia de uso es la ISM variando en función de las distintas regiones.

En el aspecto de la seguridad, la transmisión de una señal a través de LoRa se realiza mediante una modulación que puede parecer aleatoria. Además, la potencia de transmisión se sitúa debajo del suelo de ruido mediante la utilización de redundancia, lo que convierte a LoRa en una transmisión segura. Por otro lado, se realiza una encriptación AES (*Advanced Encryption Standard*) de 128 bits.

Se despliega, o bien, mediante operadores públicos como Orange, o bien, mediante redes privadas usando servidores de red como “ResIoT Server”.

2.1.3 Espectro licenciado

Las tecnologías LPWAN que hacen uso del espectro licenciado habitualmente son desarrolladas por organizaciones internacionales que estandarizan las tecnologías previamente desplegadas en otro rango del espectro. También, son evoluciones de tecnologías del espectro licenciado que se adaptan a los requerimientos de las redes LPWAN.

Al igual que las tecnologías complementarias, ocupan poco ancho de banda, y habitualmente conectan a internet dispositivos que transmiten pocos datos.

Algunas tecnologías que hacen uso del espectro licenciado son: LTE-M, EC-GMS-IoT y NB-IoT.

2.1.3.1 LTE-M

Es un protocolo de comunicación estandarizado por el 3GPP, orientado a las comunicaciones con dispositivos que requieran baja potencia de transmisión, baja velocidad de transmisión y a dispositivos que puedan trabajar con altos tiempos de latencia [12].

La utilización de la tecnología LTE para el control de dispositivos IoT se refleja en sucesivas *releases* del estándar 3GPP, que se confirman en la R*elease* 13 en la cual se añaden especificaciones para definir dispositivos que permitan las comunicaciones MTC (*Machine Type Communicactions*), dando lugar a las eMTC (*enhanced Machine Type Communicactions*). Las principales características de los dispositivos definidos son la simplificación del hardware de estos, la capacidad de mantener la comunicación con pérdidas de MCL de 155.7 dB y la repetición en el envío de bloques en el UL con base en la RSRP (*Reference Signal Received Power*) medida en el UE.

Comparando las comunicaciones LTE-M y las comunicaciones LTE orientadas a IoT [13], las principales mejoras en la *Release 13* del 3GPP respecto a la anterior, son los dispositivos dedicados únicamente a las comunicaciones M2M y la definición del uso de la banda de 1.4 MHz para dichos dispositivos.

La baja latencia y el amplio ancho de banda de la tecnología LTE-M, tratándose de una tecnología orientada a las comunicaciones M2M, convierte a LTE-M en tecnología óptima

para aplicarse en el seguimiento de personas o cosas, telemetría o digitalización de servicios.

Se puede desabastecer con relativa rapidez dado que es una tecnología ligada a 4G.

2.1.3.2 EC-GMS-IoT

Es un protocolo orientado a dar cobertura a dispositivos IoT. Al igual que el protocolo LTE-M se basa en la tecnología 4G, en este caso, el protocolo EC-GMS-IoT se complementa con la tecnología GSM. Al tratarse de un protocolo complementario a GSM, también coexiste con las redes 2G, 3G y 4G.

2.2 NB-IoT

La investigación relacionada con la comunicación M2M a través de la red celular comienza en 2005 por medio del 3GPP. Esta investigación tiene su primer reflejo en las publicaciones del 3GPP en la *Release 8*, definiendo el modo de comunicación, la señalización y el direccionamiento para la gestión masiva de usuarios [14]. De nuevo, se reflejan avances en la *Release 12*, en la que se definen los dispositivos orientados a la comunicación MTC, se actualizan las especificaciones de seguridad, la señalización y la cobertura. Seguidamente, en la *Release 13*, se incorpora la tecnología eDRX, se simplifican los terminales M2M, se realizan mejoras de cobertura en interiores, se aumenta la potencia de transmisión y se añade una arquitectura de red. A partir de esta *release* se establece la tecnología NB-IoT.

2.2.1 NB-IoT. Arquitectura

La arquitectura habitual en sistemas LPWA se compone de nodos finales, *gateways* o concentradores, un servidor y una aplicación de servidor. Los nodos finales contienen el Equipamiento de Usuario. El UE normalmente emite de forma asíncrona en momentos puntuales, mientras que “escucha” de forma predefinida en ventanas temporales. Los *gateways* se consideran puntos intermedios que participan tanto en el UL (*Uplink* Enlace Ascendente) como en el DL (*Downlink* Enlace descendiente). Así, en DL retransmiten los paquetes recibidos y en UL procesan y controlan los paquetes recibidos y realizan envíos selectivos al servidor. El servidor almacena, controla y distribuye la información que recibe, tanto de los gateways como de la aplicación del servidor. Finalmente, la aplicación del servidor gestiona la información que recogen y transmiten los nodos finales para proveer servicios al usuario de la aplicación.

En distintos casos de uso de redes LPWAN, el segmento de la red comprendido entre los dispositivos finales y los *gateways* usa una tecnología propietaria, mientras que, en el segmento de la red comprendido entre los *gateways* y el servidor de datos, se encuentran la red de acceso radio y el núcleo de la red, que puede ser LTE o 5G, e internet, como se ilustra en la Figura 3.

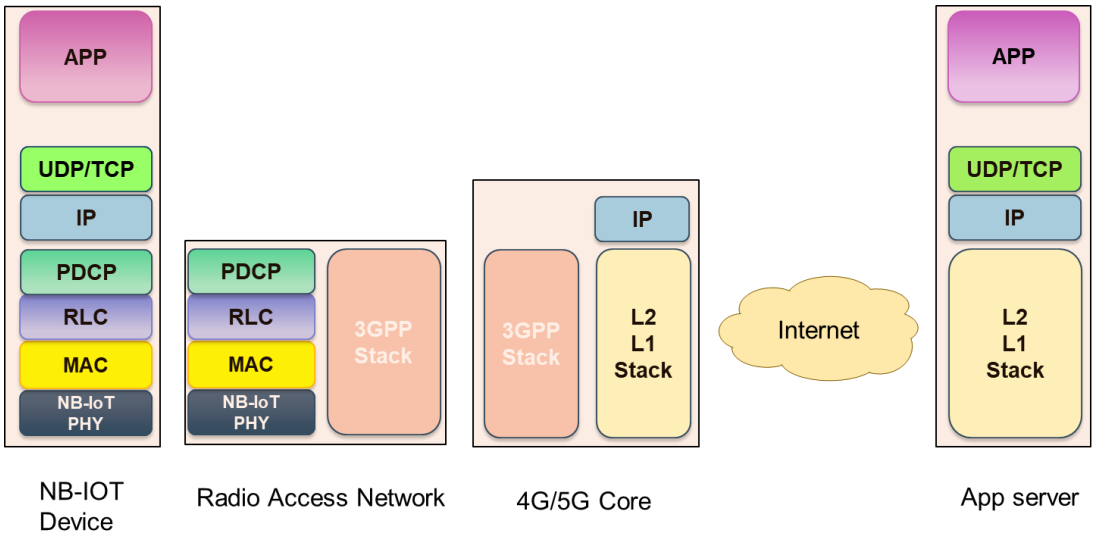
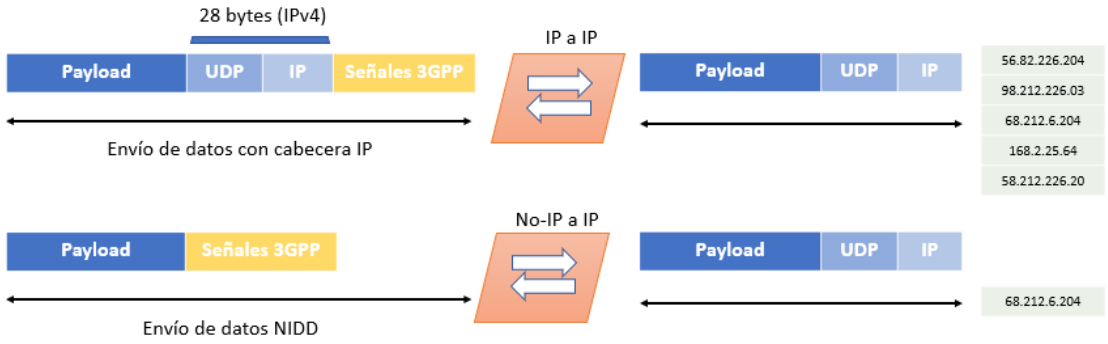


Figura 3.- Arquitectura de red de NB-IoT.

En el caso de la tecnología NB-IoT [15], al tratarse de una tecnología que hace uso de la red celular, las estaciones base de NB-IoT se incorporan en las mismas ubicaciones que las eNB que darían servicio a la red LTE. Además, los UE que hacen uso de la tecnología NB-IoT pueden emitir señales con un rango de varios kilómetros, lo que supone que las eNB darían servicio directo en la mayoría de los casos a los dispositivos finales y, por tanto, se elimina la necesidad de elementos intermedios como los *gateways*. En este sentido, los *gateways* pueden actuar como concentradores en caso de alta densidad de dispositivos.

Por otro lado, la transmisión de datos dentro de una red NB-IoT se puede realizar de 2 formas en función del formato de los paquetes de datos enviados: basado en IP y no basado en IP. La solución no basada en IP requiere una tecnología propietaria desde los módulos NB-IoT de los dispositivos hasta la plataforma de gestión, lo que dificulta su implementación.

Una solución mixta en la que paquetes de datos enviados sin cabecera IP se introducen en la red mediante una reestructuración de dichos paquetes se ilustra en la Figura 4. En esta solución ambos paquetes de datos tienen una estructura similar antes de llegar al otro extremo de la red, sin embargo, cuando el dispositivo final envía o recibe los paquetes, éstos difieren en su estructura.



Basada en:[15]

Figura 4.- Comunicación basada en IP y comunicación no basada en IP.

La transmisión no basada en IP (NIDD) estructura el paquete de datos enviado desde el UE tan solo con el *payload* e información de señalización en la red. Para enviar con éxito este tipo de paquetes se necesita un elemento intermedio en la red llamado *Packet Data Network Gateway,* que compone un nuevo paquete que podrá ser distribuido e identificado por la red LTE. Esta forma de componer paquetes de datos está prediseñada por el fabricante de módulos NB-IoT, lo que limita su implementación y, además, no todos los operadores soportan ese tipo de transmisión, lo que acota aún más este formato. Este tipo de transmisión tan solo puede transmitir a una dirección IP ya prefijada en el dispositivo o en el *Packet Data Network Gateway*.

Por otro lado, la transmisión de datos basada en IP estructura el paquete de datos enviado desde el UE con, además del *payload* e información de señalización en la red, una cabecera de datos IP y una cabecera de datos, o bien, UDP, o bien, TCP, en función del protocolo de transporte utilizado. Este tipo de transmisión permite transmitir los datos a la dirección IP que se requiera en cada momento.

Los protocolos de transporte utilizados dependen del tipo de comunicación que se pretenda establecer. En caso de utilizar el protocolo TCP, el dispositivo final establece una conexión permanente, que asegura la entrega de todos los paquetes de datos transmitidos en ambos sentidos, pero conlleva un consumo de batería constante para mantener establecida la comunicación.

En caso de utilizar el protocolo UDP, el dispositivo final envía paquetes no orientados a la conexión, lo que no asegura la entrega de todos los paquetes, pero conlleva un consumo de batería muy reducido al poder pasar los dispositivos finales a periodos de inactividad periódicamente.

La arquitectura de una red NB-IoT se define, entre otros, mediante 2 protocolos de internet: *Publish - Suscribe* y REST (REpresentational State Transfer).

La arquitectura Publicación – Suscripción comunica los dispositivos finales y el servidor mediante una comunicación bidireccional basada en mensajes predefinidos. La arquitectura requiere un bróker que haga de intermediario entre el servidor y los dispositivos finales.

La aplicación de este protocolo se puede llevar a cabo mediante un servidor MQTT [16] junto con el protocolo de transporte TCP. Los dispositivos se asocian a un bróker, a través del cual pueden realizar la función de publicar (“*Publish*”) o suscribirse (“*Suscribe*”) a un determinado tema o etiqueta. Mediante una publicación, un dispositivo actualiza la información asociada al tema que se dirige la publicación. Así mismo, mediante una suscripción, un dispositivo recibe la información asociada a un tema al que se suscribe junto con las sucesivas actualizaciones de información de dicho tema. Tanto las suscripciones como las publicaciones se realizan mediante una conexión TCP/IP.

Adicionalmente, con la necesidad intrínseca de ahorro de energía de la tecnología NB-IoT, la arquitectura *Publish - Suscribe* se puede llevar a cabo mediante un servidor MQTT-SN junto con el protocolo de transporte UDP. Comparando los 2 tipos de servidores [17] que se exponen, los servidores MQTT-SN permiten identificar temas mediante ID para ahorrar memoria y recursos, permiten la suscripción configurada previamente para ahorrar mensajes, permiten utilizar el protocolo UDP, y mantienen las suscripciones activas en dispositivos inactivos.

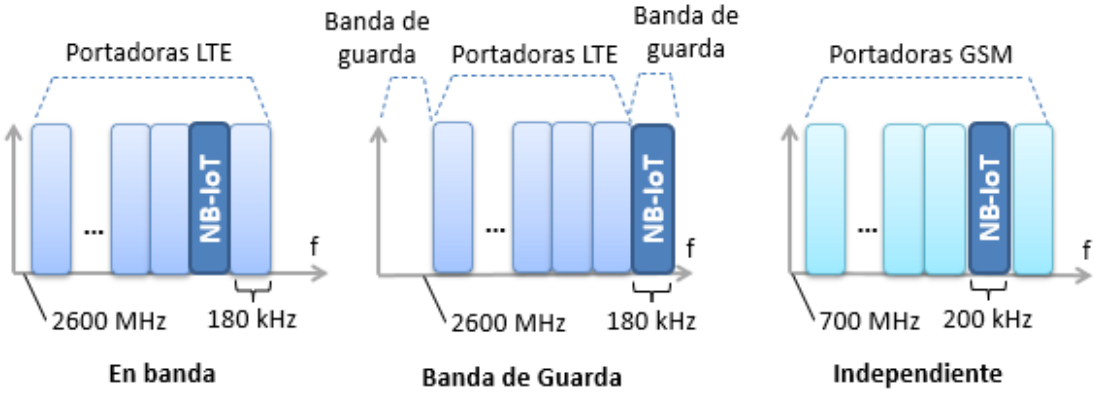
La arquitectura REST comunica los dispositivos finales y el servidor mediante un servidor central donde sitúa toda la información. La alternativa de utilización de un servidor HTTP y/o el protocolo de transporte TCP supone un consumo muy alto de recursos para la tecnología NB-IoT. Por otro lado, se puede implantar la arquitectura REST mediante un servidor CoAP y el protocolo de transporte UDP.

CoAP [18] funciona en modo solicitud / respuesta, en el cual el cliente inicia la solicitud al servidor y el servidor responde. Algunas señales que utilizan son: GET, POST, PUT y DELETE. No requiere una conexión persistente, por tanto, es adecuado para dispositivos que entran en periodos de inactividad. Dispone de mecanismos de retransmisión de mensajes para subsanar la fiabilidad del protocolo UDP. No funciona en tiempo real, debido a la latencia y a la inactividad de los dispositivos finales. Comparando con un servidor HTTP, CoAP optimiza el tamaño de paquete de datos.

2.2.2 NB-IoT. Modos de operación

Para realizar el despliegue de la señal NB-IoT y establecer una conexión entre el UE y la eNB, se establecen 3 tipos de acceso radio NB-IoT en función de los modos de operación [19] en el espectro radioeléctrico como se ilustra en la Figura 5. Los 3 tipos de acceso al espectro radioeléctrico mediante NB-IoT son:

1. *Independiente*, o *Stand-alone*, con la portadora dedicada a NB-IoT.
2. *En banda*, o *In-band*, con NB-IoT ocupando parte del ancho de banda dedicado a la portadora LTE.
3. En la *banda de guarda*, o *Guard-band*, ocupando el espacio dedicado a la banda de guarda de la portadora LTE.



Basada en:[20]

Figura 5.- Despliegues de la señal NB-IoT en el espectro.

En el modo de operación con *despliegue Independiente*, el ancho de banda ocupado es de 200 kHz, donde se dedican 180 kHz a la transmisión de la señal y 10 kHz a cada lado de la señal a bandas de guarda. La banda NB-IoT se sitúa en el espectro utilizado por la red de acceso inalámbrico de GSM, ocupando el espacio utilizado por 1 o más portadoras de GSM, cuyo apagado se prevé para el 2025.

Este tipo de despliegue es el menos complejo dado que reutiliza la infraestructura dispuesta para GSM. La transmisión tiene la limitación de uso de 1 sola antena en el UE.

En el modo de operación con *despliegue en banda*, el ancho de banda ocupado es de 180 kHz. La banda NB-IoT está integrada dentro de la portadora de LTE, y hace uso de uno o varios bloques de recursos físicos (PRB) de la estructura de bloques de LTE, sin incluir los bloques de señalización y control de dicha estructura. En este modo de operación, las tecnologías LTE y NB-IoT comparten eNB, por lo tanto, para llevar a cabo una incorporación progresiva de dispositivos NB-IoT a la red solo es necesario que la celda servidora de LTE que se utilice esté correctamente actualizada.

En el modo de operación con *despliegue en la banda de guarda*, el ancho de banda ocupado es de 180kHz. En este modo de operación, la banda NB-IoT se despliega en la banda de guarda de la portadora LTE, por lo tanto, al igual que en el despliegue *En banda*, las tecnologías LTE y NB-IoT comparten eNB.

Para situar la banda NB-IoT entre portadoras LTE se tiene en cuenta que cada portadora LTE tiene al menos 100 kHz de banda de guarda a cada lado, con lo que siempre habrá 200 kHz desocupados en la mínima separación que puede haber entre bandas LTE y, por tanto, las bandas NB-IoT y LTE no se solapan en funcionamiento simultáneo. Para evitar el solapamiento, la frecuencia central de la señal NB-IoT se aleja como máximo 7.5kHz de la posición esperada a 100kHz de la señal LTE y, además, se regula el factor roll-off de los filtros para LTE.

Respecto a la compatibilidad de la tecnología NB-IoT con la tecnología LTE, los modos de operación NB-IoT que utilizan las eNB desplegadas para LTE, generan la necesidad de procesado de ambas señales conjuntamente. Esto pone en riesgo la ortogonalidad de la portadora de LTE [20], lo que puede ocasionar interferencias entre subportadoras. En este ámbito, solo hay interferencias de los dispositivos NB-IoT a los dispositivos LTE y no en sentido contrario, además, las interferencias provenientes de la incompatibilidad entre el dispositivo NB-IoT y la eNB, dependen del tamaño de la banda de guarda utilizada y la frecuencia de muestreo utilizada.

En todos los modos de operación, la banda NB-IoT ocupa 180 kHz, equivalentes a 1 bloque de recursos de LTE, tanto en el enlace descendente como en el ascendente. Este bloque de recursos se transmite mediante el modo de transmisión FDD (*Frequency-Division Duplexing*) en ambos sentidos, lo que supone que los enlaces UL y DL se transmiten a través de bandas de frecuencia distintas. En el caso de la eNB, el funcionamiento es idéntico al utilizado en la tecnología LTE, transmitiendo y recibiendo al mismo tiempo. Sin embargo, en el caso del UE, dado que la tecnología NB-IoT se orienta al ahorro de energía y a la simplificación de los dispositivos, el modo de transmisión es HD-FDD (*Half Duplex Frequency-Division Duplexing*), o FDD tipo B, lo que supone que los enlaces UL y DL no concurren en el tiempo y existe una banda de guarda entre ambos enlaces.

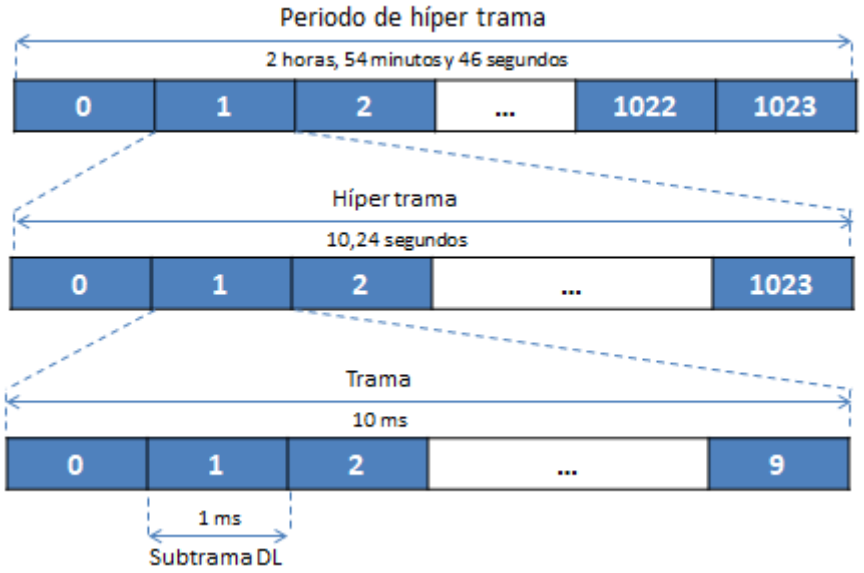
Durante la transmisión, la etapa de recepción de datos de un extremo del enlace coincide con la etapa de transmisión de datos del otro extremo del enlace, luego, se detiene la transmisión durante al menos una subtrama y seguidamente, las cadenas de transmisión y recepción de los correspondientes extremos del enlace se cambian los roles de transmisión y recepción.

Todos los enlaces se programan desde la eNB al inicio de la comunicación mediante la función de programación, o *scheduling*.

2.2.2.1 Trama del enlace descendente

La transmisión de la información se organiza mediante tramas en los enlaces UL y DL. Ambas tramas comparten la misma duración de 10 ms, sin embargo, dichas tramas tienen distinta estructura dependiendo de si se trata del enlace ascendente o descendente.

Para indexar cada trama se utiliza el sistema SFN (*System Frame Number*), que contabiliza hasta 1024 tramas (de 0 a 1023). La totalidad de tramas contenidas en un SFN constituye un periodo de SFN. Y a su vez, para contabilizar los periodos de SFN, se utiliza el sistema H-SFN (*Hyper-SFN*), que contabiliza hasta 1024 periodos de SFN. Estas divisiones, junto a la subdivisión de una trama DL, se ilustran en la Figura 6.

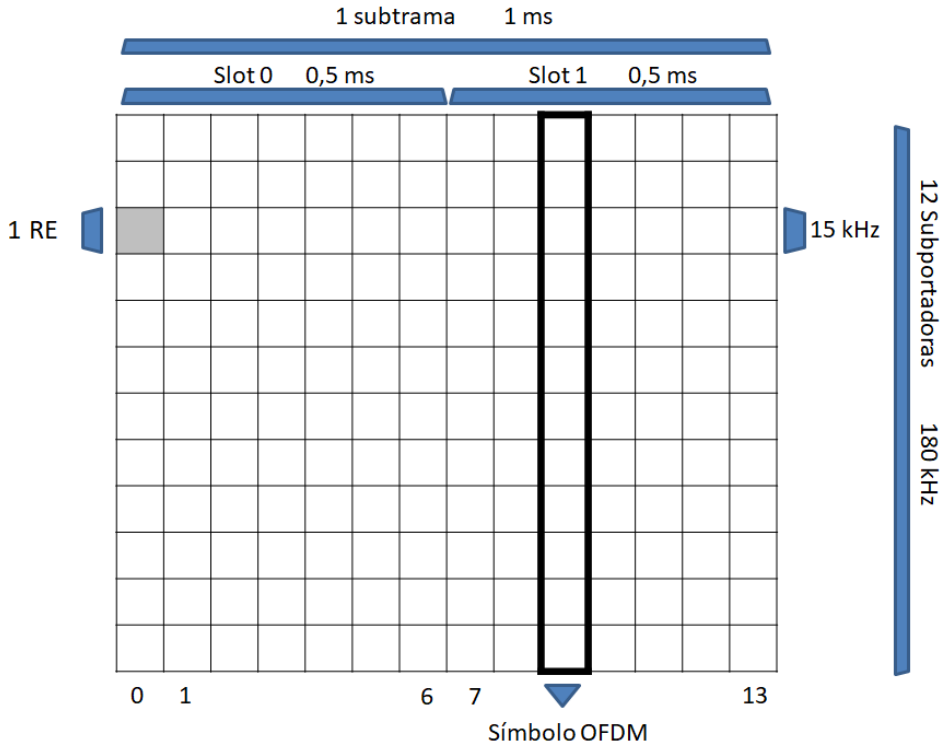


Basada en:[21]

Figura 6.- Sistema de división H-SFN en el enlace descendente.

Cada trama del enlace DL se subdivide en 10 subtramas de 1 ms de duración cada una y a su vez, cada subtrama se divide en 2 ranuras temporales, o slots, de 0.5 ms de duración. También se subdivide cada subtrama en 14 divisiones en el dominio del tiempo, correspondientes a la modulación OFDM, dando lugar a 14 símbolos OFDM por subtrama.

Continuando con la descomposición en el dominio de la frecuencia, en el caso del enlace descendente, cada subtrama se divide en 12 subportadoras separadas 15kHz entre sí. Tomando como referencia un símbolo OFDM, cada una de las 12 subdivisiones del símbolo OFDM es un RE (Resource Element), con un total de 168 RE por cada PRB, como se muestra en la Figura 7.



Basada en:[21]

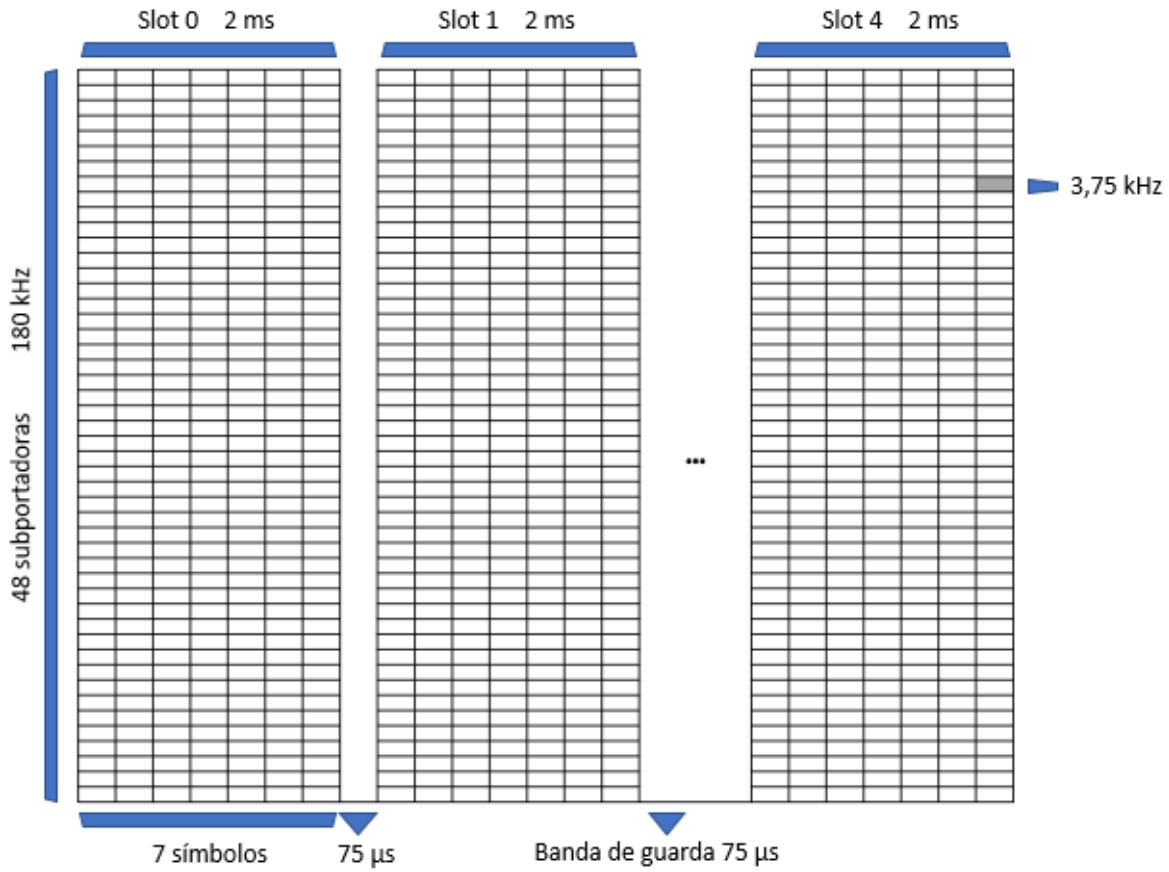
Figura 7.- Sistema de división de subtramas en el enlace descendente.

2.2.2.2 Trama del enlace ascendente

En el enlace UL, se incorpora una unidad de medida adicional correspondiente a la menor división de recursos disponible, llamada RU (*Resource Unit*).

El enlace UL dispone de una transmisión similar al enlace DL con subportadoras espaciadas 15 kHz y 3.75 kHz, emitiendo 12 o 48 subportadoras respectivamente, con una duración de 2 segundos del *slot* en caso de espaciado de 3.75 kHz. En caso de emitir 48 subportadoras, la cantidad de RU es igual a la cantidad de RE

El enlace UL también puede transmitirse mediante la modulación SC-FDMA, que supone la subdivisión temporal de las subportadoras. La modulación SC-FDMA solo puede usarse con un espaciado de 5 Hz y, en este caso, cada subportadora permite una subdivisión temporal de 1, 3, 6 y 12 subportadoras en el dominio del tiempo. Así mismo, la cantidad de RU se multiplica por 1,3,6, o 12 RE en función de la subdivisión elegida. En la Figura 8, se muestra una trama de enlace ascendente de 10 ms de duración con subportadoras espaciadas 3.75 kHz emitiendo en un único tono sin subdivisión de subportadoras.



Basada en:[21]

Figura 8.- Sistema de división de trama en el enlace ascendente.

2.2.3 NB-IoT. Canales y señales

La señalización y los canales utilizados en la transmisión NB-IoT dependen del sentido de la transmisión y del tipo de despliegue en la banda utilizado. Así, la señalización y los canales se analizan diferenciando el enlace descendente y el enlace descendiente.

2.2.3.1 Enlace descendente

En el enlace descendente se emplea la modulación OFDMA, con una separación entre subportadoras de 15kHz dentro de 180 kHz de ancho de banda ocupados por la portadora, lo que hace un total de 12 canales modulados. A su vez, cada canal se modula mediante QPSK.

El conjunto de las 12 subportadoras equivale a 1 bloque de recursos de LTE. Este bloque se situará en distintos lugares del espectro en función del tipo de despliegue, y así mismo, utilizará distintos recursos.

El posicionamiento de los canales de señalización y control varía ligeramente en función del modo de operación utilizado, siendo similares el resto de los canales en todos los despliegues.

En el enlace descendente se utilizan las señales de referencia y señalización NRS, NPSS y NSSS, y los canales NBPCH, NPDCCH, NPDSCH [20] [22], ilustrados en la Tabla 1.

Enlace	Canal	
Enlace Descendente DL	NPDCCH	Narrowband Physical Downlink Control Channel
	NPDSCH	Narrowband Physical Downlink Shared Channel
	NPBCH	Narrowband Physical Downlink Broadcast Channel
	NPSS/NSSS	Narrowband Synchronization Signal
Enlace Ascendente UL	NPUSCH	Narrowband Physical Uplink Shared Channel
	NPRACH	Narrowband Physical Random Access
	NRS	Narrowband Reference Signal

Tabla 1.- Tabla de canales y señales NB-IoT

Señales de sincronización

NPSS es la señal de sincronización inicial utilizada en el enlace descendente.

NSSS es la señal de sincronización utilizada para confirmar la recepción de la toda la información referente a la celda correspondiente al UE.

Canales físicos

NBPCH

Este canal también se encuentra en la comunicación mediante LTE. Es el canal utilizado para transmitir información acerca de la celda y la configuración de la red. El bloque de información principal MIB-NB va en este canal. Para que el UE se sincronice, necesita decodificar el MIB, junto con los sistemas de bloques de información SIB1 y SIB2.

NPDCCH

Este canal también se encuentra en la comunicación mediante LTE. Es el canal utilizado para entregar las señales de control DCI (*Downlink Control Information*) desde la estación base (eNB) a la mota UE. Este canal dispone de 2 formatos (0 o 1), que determinan si el canal dispone de 1 o 2 Elementos de Control de Canal de Banda Estrecha (NCCE) respectivamente.

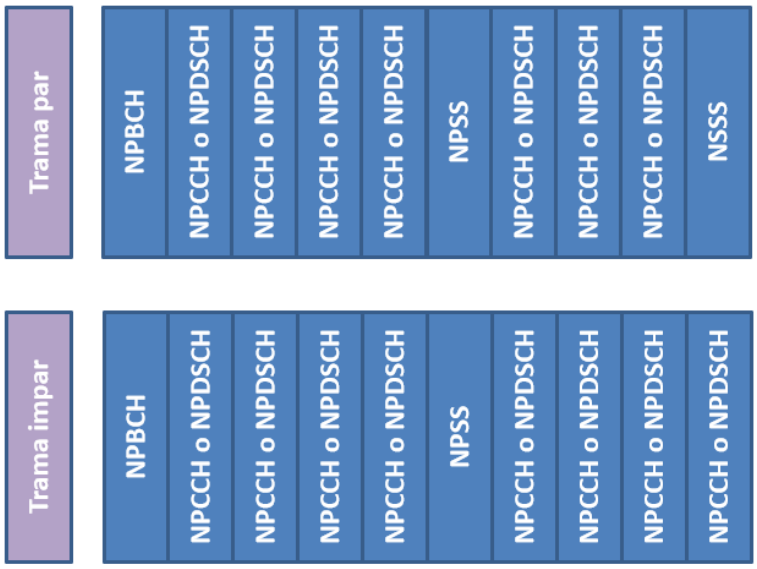
La información de control transmitida en este canal tiene 3 formatos distintos:

1. DCI N0: Información de control dirigida al UE para informar acerca de los recursos disponibles en el envío de datos en el enlace ascendente (NPUSCH)
2. DCI N1: Información de control dirigida al UE para informar acerca de la programación del enlace descendente, incluyendo cuando se transmitirán datos en el canal NPDSCH.
3. DCI N2: Información de control dirigida al UE para informar del *paging*.

NPDSCH

Este canal también se encuentra en la comunicación mediante LTE. Es el canal [23] utilizado para transmitir la señal que se usa para autenticar la llegada de paquetes, tanto de datos como de control, desde la estación base (eNB) a la mota (UE). El esquema de modulación del canal NPDSCH viene determinado por la información de control DCI N1. El espacio ocupado por cada subportadora en el canal NPDSCH es de 15 kHz, idéntico al utilizado en el DL en LTE.

El enlace descendente tiene una distinción entre las tramas pares e impares, que incluye la señal de sincronización NSSS al final de la trama impar. Y, por otro lado, un bloque adicional dedicado a los canales NPDCCH o NPDSCH en la trama par, como se ilustra en la Figura 9.



Basada en: [24]

Figura 9.- Trama par *even* y trama impar *odd* del enlace *descendente.*

La retransmisión de tramas a través de los canales NPCCH y NPDSCH puede variar desde la transmisión de 1 sola subtrama, con una duración de 1 ms, hasta la transmisión repetida de 2048 veces de una trama, con una duración total de 20,480 ms. Por otro lado, la retransmisión de tramas incrementa los tiempos de retardo en los periodos de ahorro de energía, que limitan el acceso radio a los dispositivos.

2.2.3.2 Enlace ascendente

En el enlace ascendente se utilizan 2 tipos distintos de modulación en función del número de subportadoras dedicadas a transmitir: SC-FDMA y FDMA.

Mediante la modulación SC-FDMA, se utilizan 1 o varias subportadoras para transmitir información a un UE.

Mediante la modulación FDMA, se utiliza una sola subportadora o canal para transmitir información a un UE.

La modulación utilizada para cada canal es la QPSK, con un espacio ocupado por cada subportadora en el canal NPUSCH de 3.75kHz, distinto al utilizado en el canal PUSCH de UL de LTE. La transmisión de la señal NB-IoT [25] a través del espectro se hace en 2 casos mediante la señal LTE, *en banda* y en la *banda de guarda*. Dada la diferencia de anchos de banda entre una tecnología de acceso y otra, se ocasionan interferencias al disponer de incompatibilidades en la decodificación del enlace. Se distingue el rendimiento en función del modo de despliegue utilizado, siendo el despliegue en la Banda de Guarda el que tiene menos interferencias, y por tanto, mayor rendimiento sobre el despliegue *en banda*, que sufre más interferencias.

En el enlace ascendente se utilizan los canales NPRACH y NPUSCH, ilustrados en la Tabla 1.

NPRACH

El canal NPRACH es utilizado para dar inicio al acceso radio [26], así, el dispositivo UE solicita la programación, o *scheduling*, a la eNB mediante el mismo. La eNB estima la posición del UE mediante el ToA (*Time of Arrival*) de la señal PRACH enviada a través del canal NPRACH.

El canal se puede subdividir en 12, 24, 36, o 48 subportadoras espaciadas 3.75 kHz, con periodicidad variable entre 40 ms y 2560 ms. El preámbulo enviado en este canal puede enviarse repetidamente hasta 128 veces. Las distintas opciones se adaptan al nivel de cobertura.

NPUSCH

El canal NPUSCH es utilizado para transmitir datos e información de control desde la mota (UE) a la estación base (eNB). Transmite la solicitud HARQ ACK. Ocupa un solo PRB. La información transmitida en este canal se repetirá un número determinado de veces en función del nivel de cobertura disponible en el UE [27]. La información contenida en el canal NPUSCH se ubicará utilizando como herramienta la organización de RU.

Los retardos en ambos sentidos de la comunicación se producen en su mayoría por la periodicidad de las señales [23], así en el enlace descendente la decodificación de los bloques información SIB y MIB, la periodicidad del canal NPRACH y la repetición en la transmisión del canal NPUSCH son factores que añaden retardo a la comunicación.

2.2.4 NB-IoT. Stack

El protocolo de *Stack* utilizado en NB-IoT tiene una estructura similar a la utilizada en LTE debido a que esta tecnología está diseñada para usar recursos de LTE, con la diferencia de que NB-IoT tiene menos complejidad con el objetivo de ser usada en dispositivos sencillos y de bajo coste.

El protocolo de *Stack* se clasifica en 2 puntos de vista: el plano de usuario y el plano de control.

El plano de usuario comprende la capa física (PHY), la capa MAC, la capa RLC y la capa PDCP.

El plano de control [28] comprende las capas PHY, MAC, RLC, PDCP, RRC y NAS como se ilustra en la Figura 10. La descripción de las capas es la siguiente:

La capa PHY, o capa física, similar a la que se encuentra en la tecnología LTE se adapta a las restricciones encontradas en NB-IoT. Está descrita en el apartado Modos de Operación.

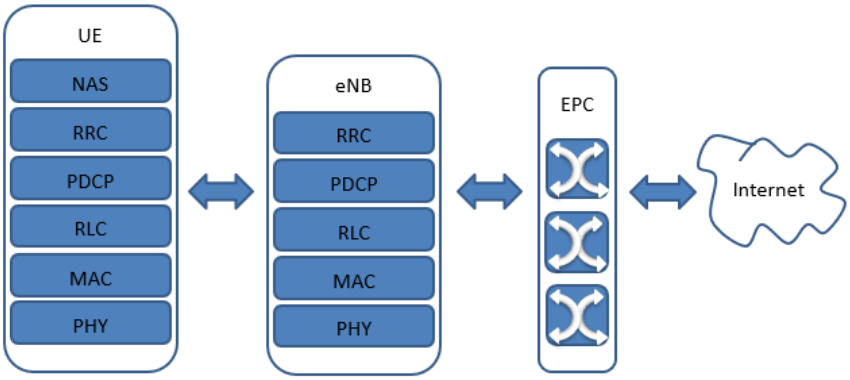
La capa MAC, *Medium Access Control*, se encarga de la programación de las señales y los canales, de distribuir los recursos en tiempo y frecuencia, y de la formación de paquetes de datos para su manejo en las distintas capas.

La capa RLC, *Radio Link Control*, se encarga de transferir unidades de datos de capas superiores.

La capa PDCP, *Packet Data Convergence Protocol*, se encarga del cifrado, la protección y la transferencia de datos dando servicio a capas superiores.

La capa RRC, *Radio Resource Control*, se encarga de programar los periodos de actividad y transmisión del dispositivo. Los módulos NB-IoT tienen los estados RRC conectado y RRC idle.

La capa NAS, *Non Access Stratum*, se encarga de transportar las señales que no sean radio entre el UE y la MME (*Mobility Management Entity*)



Basada en:[21]

Figura 10.- Protocolo de Stack en NB-IoT.

2.2.5 NB-IoT. Cobertura

La tecnología NB-IoT está orientada a proporcionar cobertura en un amplio rango de casos de uso, siendo el industrial, dentro de fábricas o edificios industriales, uno de los principales targets, y a la vez, uno de los más restrictivos. En este sentido, la tecnología NB-IoT satisface en la *Release 13* del estándar 3GPP una cobertura en interiores que disponga de una mejora una mejora de 20 dB sobre la tecnología GPRS, lo que supone un requerimiento de 164 dB de MCL a la tecnología NB-IoT.

Para satisfacer estos requerimientos de cobertura, como se muestra en el apartado Modos de Operación, se realizan 3 tipos de despliegue NB-IoT [29]: *en banda*, en *banda de guarda*, o *independiente*.

Cada tipo de despliegue supone una adición de hardware distinta en cada eNB y tiene, en consecuencia, unos valores distintos de velocidad de transmisión en cada tipo. En el caso del despliegue *en banda* se tienen las siguientes condiciones en la red:

La señal NB-IoT utiliza uno o varios bloques de recursos comprendidos dentro del ancho de banda de la portadora LTE. Como cita Mangalvedhe en [19], para una portadora LTE de 10MHz con una potencia asignada de 46 dBm, cada bloque de recursos dentro ésta tiene una potencia asociada de 35 dBm, teniendo también en este caso el PRB asignado a NB-IoT la misma posibilidad que los otros bloques de obtener un *PSD-boosting* de 6

dB, lo que sitúa el bloque de recursos asignado a NB-IoT con 41 dBm de potencia máxima asignada.

Dentro del PRB utilizado por parte de NB-IoT se distinguen el enlace ascendente y el enlace descendente. En el enlace descendente se utiliza la mayor parte del PRB, sin embargo, se reserva para LTE los espacios ocupados por las celdas dedicadas a la señalización o los *flags*.

En el despliegue en Banda de Guarda se dan las siguientes condiciones en la red:

Se utilizan bloques de recursos físicos que no estaban siendo utilizados previamente [29], y algunos recursos asignados a la tecnología LTE. Al no utilizar PRB previamente utilizados, la información dedicada a NB-IoT se puede desplegar en todo el PRB sin tener en cuenta los espacios de señalización o *flags* de LTE.

A pesar de no utilizar recursos LTE, la señal NB-IoT ocupa ancho de banda dedicado a la transmisión de la señal LTE, lo que quiere un ajuste en los filtros *roll-off* de la portadora LTE para que ambas señales se emitan sin solaparse y se mantenga la ortogonalidad.

Pueden aparecer problemas de potencia en el caso de que el dispositivo se encontrase muy alejado de la celda. Sin embargo, no hay problemas de interferencias. Estos problemas de potencia se pueden resolver [19] de la misma forma que se resuelve en la tecnología LTE, mediante *powerboosting*, donde se utiliza la potencia dedicada a otros PRB aumentando hasta 6dB la potencia del PRB deseado.

En el despliegue Independiente se dan las siguientes condiciones en la red:

Se utiliza un ancho de banda previamente utilizado por señales GSM, así, con 200MHz, el ancho de banda de la señal NB-IoT es ligeramente superior al ancho de banda de las subportadoras asignadas a los otros despliegues.

El PRB que utilice puede estar siendo también utilizado en otras celdas que utilicen otro tipo de despliegue (en banda).

Al tratarse de una portadora única, la señal NB-IoT se transmite con una potencia de 43 dBm desde la eNB. La señal transmitida mediante despliegue Independiente sufre menos interferencias que en los otros despliegues.

También en la *Release 13* del estándar 3GPP se instaura un objetivo de cobertura a un número determinado de dispositivos por celda servidora NB-IoT: 52547, determinado con base en la densidad de población de la ciudad de Londres.

En el caso de tan solo una parte de la estructura de celdas LTE soporte la red NB-IoT, la red NB-IoT tiene que adaptarse a las celdas disponibles, a pesar de que no sean las óptimas.

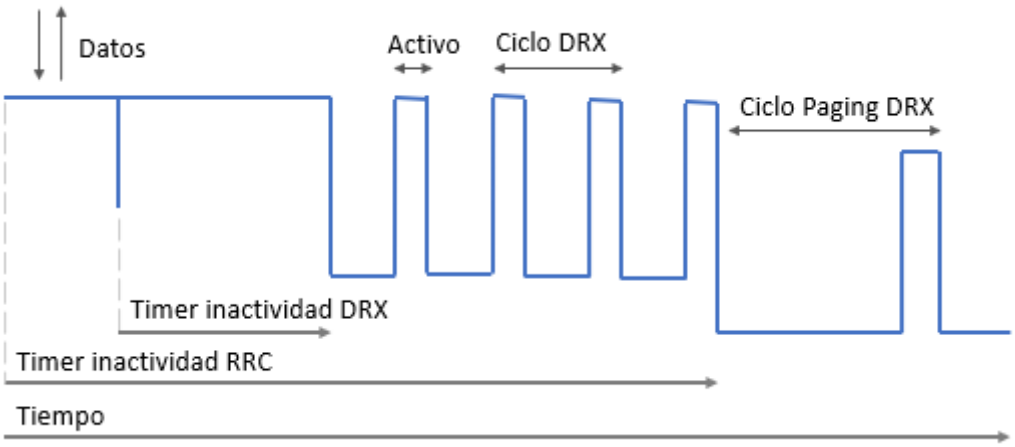
2.2.6 NB-IoT. Consumo

Uno de los puntos fuertes de la tecnología NB-IoT es el bajo consumo de batería que requieren los dispositivos para la comunicación por medio de NB-IoT, debido a las distintas formas de ahorro de consumo energético. Las principales formas de ahorro de energía [30] son Recepción Discontinua Extendida (eDRX) introducida en la *Release 13*, Modo de Ahorro de Batería (*Power Saving Mode*, PSM) introducido en la *Release 12*, baja potencia de emisión del UE, y reducido tiempo en el estado conectado RCC [31]. Estas formas de ahorro de batería permiten desplegar dispositivos que no necesiten recargar o sustituir su batería durante hasta 10 años. Las formas de ahorro de consumo de batería funcionan de la siguiente forma:

La potencia de emisión de las motas que utilizan la tecnología NB-IoT es de 14dBm (clase 6).

El periodo eDRX actúa sobre el modo IDLE, extendiendo el tiempo (o la periodicidad) en la que el modem se va a modo recepción. Se modifica ese tiempo desde 1 segundo hasta 2000 segundos. Se diferencia del modo PSM en que periódicamente atiende señales. Sin embargo, la mota que se encuentre en PSM y requiera atender señales, necesita realizar de nuevo el *setup* para encontrarse en disposición de hacerlo. Al igual que en el modo PSM, la mota (UE) solicita a la red si puede ampliar la periodicidad y la red ratifica o deniega la ampliación.

Una vez por cada ciclo *paging* DRX, se produce una ocasión de *paging*, donde la red notifica al UE la existencia de mensajes mediante un mensaje *paging*. En la Figura 11 se ilustran los ciclos de actividad e inactividad.



Basada en:[32]

Figura 11.- Ciclos de consumo de batería.

El modo PSM es un tipo de *Deep Sleep*. Una vez la mota entra en modo PSM se arranca un temporizador que va desde 2 segundos a 310 horas. Durante el periodo de PSM, el modem no es capaz de recibir ningún mensaje.

En el caso de una mota solicite ampliar su estado de PSM, la Entidad MME puede ratificar o denegar un periodo muy largo de PSM. En el caso de que la red acepte, lo notifica al dispositivo, y en caso contrario, la red propone unos valores con los que sí opera.

Por otro lado, se diferencia el ahorro de batería en el enlace ascendente y el enlace descendiente.

En el enlace descendente, una vez llega un paquete a la mota, se arranca un temporizador HARQ RTT que dictamina el tiempo que espera la mota antes de enviar una respuesta (3GPP TS 36.321 V8.0.0), durante ese intervalo de tiempo la mota no necesita monitorizar el canal NPDCCH, y se considera que comienza el periodo RTT, porque va a gestionar el paquete de datos actual. Seguidamente, una vez expira el timer HARQ RTT, finaliza el periodo de RTT y se arranca el periodo de inactividad de DRX en el cual no gestiona la llegada de paquetes sino el canal NPDCCH.

Paralelamente a la inclusión del periodo eDRX, se observa que reduciendo el periodo DRX en el cual la mota entra en un proceso de comprobaciones del canal NPDCCH, se reduce el consumo de la batería.

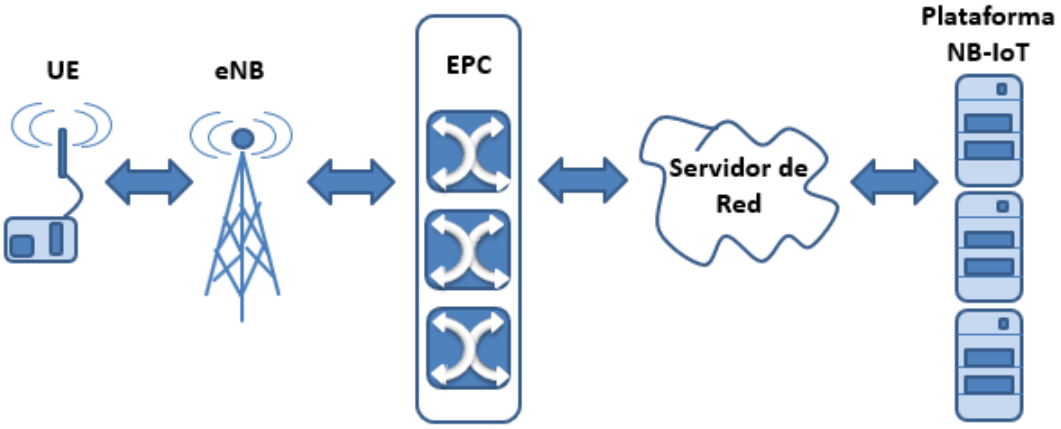
En el enlace ascendente, el dispositivo transmisor o mota no establece una comunicación permanente con el otro lado del enlace, sino que puntualmente y de forma asíncrona envía paquetes de datos a la red.

Para el control de los periodos de ahorro de energía se manejan 2 temporizadores principales: T3324 que finaliza el estado IDLE e introduce a la mota en estado PSM, y T3412 que finaliza el estado PSM e introduce a la mota en el estado TAU.

2.2.7 NB-IoT. Topología

La topología típica de las redes LPWA [33] se compone de dispositivos finales, varios *gateways*, y un servidor de red. El funcionamiento habitual del enlace ascendente incluye el envío de paquetes desde los dispositivos finales a varios nodos *gateways*, que a su vez reenvían dichos paquetes, tras procesarlos en algunos a casos, al servidor de red. Y, por otro lado, el funcionamiento del enlace descendente incluye el envío de paquetes desde el servidor de red, que selecciona un nodo *gateway* para el envío, y transmite el paquete mediante el mismo hasta el dispositivo final.

En el caso de las redes NB-IoT [15] la parte compuesta por los nodos *gateway*, o concentradores, es sustituida por las eNB, ya que los dispositivos finales se comunican directamente con las eNB. Una arquitectura típica de red NB-IoT [28] se compone del dispositivo final (UE), estaciones base adaptadas (eNB), un EPC adaptado, el servidor de red y finalmente, una aplicación de usuario, como se ilustra en la Figura 12.



Basada en:[28]

Figura 12.- Arquitectura de red NB-IoT.

Los dispositivos (UE) son motas sensorizadas que recopilan información de la ubicación en la que se encuentran. Éstas se comunican de forma inalámbrica con las estaciones base eNB mediante la tecnología NB-IoT.

Las estaciones base (eNB) se comunican con los dispositivos, por un lado, y por el otro lado, se comunican con el EPC adaptado a NB-IoT, con el que se comunican mediante el interfaz S1-lite.

El *Evolved Packet Core* (EPC) reenvía los paquetes recibidos a la plataforma NB-IoT. El EPC interactúa directamente con la capa NAS del UE.

El servidor de red gestiona y almacena la información recibida desde la red. Este reenvía la información a la plataforma NB-IoT con un formato determinado.

La plataforma NB-IoT es un servidor con conexión a internet diseñado ad-hoc para la recepción de paquetes enviados por las motas. Esta plataforma gestiona la información enviada por los dispositivos y en función de las especificaciones marcadas previamente, envía información a los dispositivos en consecuencia.

2.2.8 NB-IoT. Dispositivos

Los dispositivos que utilizan la tecnología NB-IoT pueden ser de 2 tipos en función del tipo de módulo NB-IoT utilizado. Este módulo se define por el tipo de arquitectura utilizada para desplegar la red, con lo que hay módulos diseñados para la comunicación NIDD (*Non IP Data Delivery*) y módulos diseñados para la comunicación basada en IP, y a su vez dichos tipos de dispositivo.

Los dispositivos NIDD trabajan con plataformas diseñadas específicamente para funcionar con dichos dispositivos, así se generan soluciones propietarias que no permiten la utilización de los dispositivos en otras plataformas.

Los dispositivos cuya comunicación está basada en IP se basan en el estándar 3GPP, por lo tanto, son extrapolables a distintas plataformas y usados universalmente. La limitación que tienen estos dispositivos se marca por las frecuencias asignadas a la

tecnología NB-IoT en cada país. Los dispositivos utilizados para el despliegue en el Campus Sur de la UPM están diseñados para la comunicación basada en IP.

2.2.8.1 NB-IoT. Dispositivos utilizados

Los dispositivos utilizados en el proyecto son SODAQ SARA AFF R412 [34] y Libelium WASPMOTE [35]. También he barajado la posibilidad de utilizar las placas de desarrollo akorIoT SensPro [36] y Portenta con shield Cat.M1/NB-IoT[37].

El dispositivo SODAQ SARA AFF (*Arduino Form Factor*) es una placa de desarrollo que incluye un microcontrolador de 32 bits, batería extraíble, 2 puertos JST para alimentación, 2 ranuras para colocar sensores, un acelerómetro, un módulo GPS y un módulo radio con tecnología NB-IoT, LTE-M y 2G.

Figura 13.- SODAQ SARA AFF con antena extraíble.

El dispositivo Libelium WASPMOTE es una placa de desarrollo que incluye el microcontrolador *ATmega12821*, batería extraíble, un módulo radio con tecnología 4G, NB-IoT, Cat-M y LoRaWAN entre otras. Además, se conecta al dispositivo la placa gestora de eventos *Waspmote Events Sensor Board v3* que permite tomar medidas de temperatura, presión atmosférica y humedad.

Figura 14.- Libelium WASPMOTE con módulo NB-IoT y 2 antenas.

El dispositivo akorIoT SensPro es una plaza de desarrollo que incluye un módulo *Quectel BG95-M3* con acceso radio a GSM, LTE-M y NB-IoT. Dispone de 7 sensores (temperatura, humedad presión, luz, aceleración, contacto y gas), 2 puertos I2C y extensores de sensores.

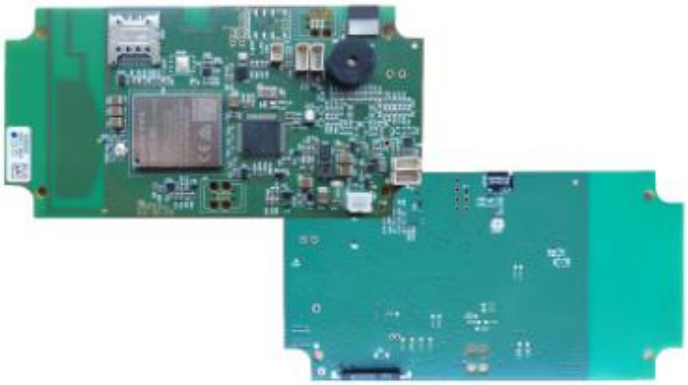


Figura 15.- AkorIoT SensPro.

El dispositivo Portenta H7 con Shield Cat.M1/NB-IoT dispone un microprocesador *STMicroelectronics dual-core STM32H747* que se combina con una placa adicional para proporcionar acceso radio de CAT-M y NB-IoT. Dispone de 2 conectores de alta densidad con 80 pines que permiten conectar variedad de aplicaciones a la placa.

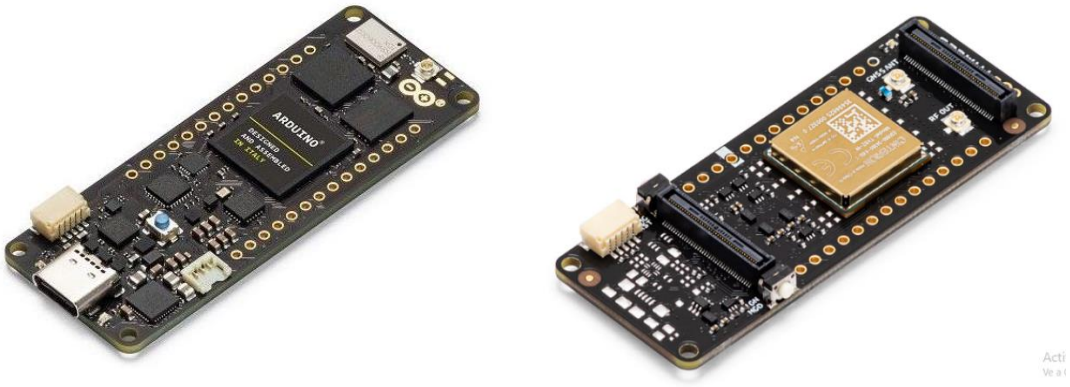


Figura 16.- Portenta H7 con Shield CAT.M1/NB-IoT.

2.3 Operadores

En el despliegue de puntos de acceso radio para la tecnología NB-IoT, hay varios operadores que han incorporado hardware adicional a sus estaciones base con el fin de proporcionar el servicio al cliente final en los mismos puntos de acceso que el cliente final accede a las redes 3G/4G/5G. Así, se puede observar que la cobertura NB-IoT se replica aproximadamente en las zonas con cobertura móvil. En la siguiente figura se muestra la posición de las antenas 4G desplegadas alrededor del Campus Sur de la UPM [38]:

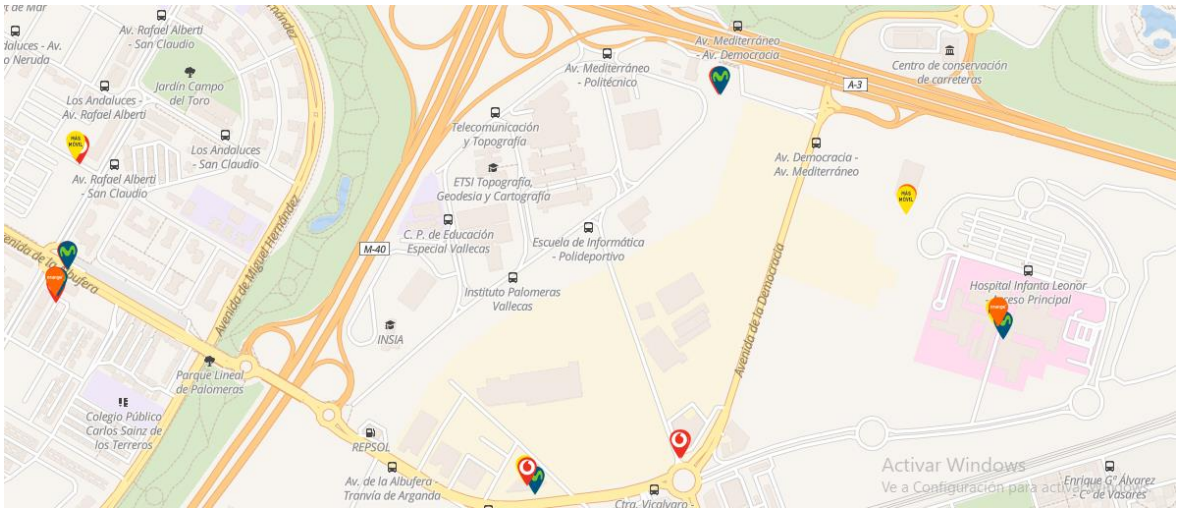


Figura 17.- Campus Sur de la UPM con antenas de operadoras.

En la Figura 17 se observa un mapa con las estaciones base próximas al Campus Sur de la UPM. Las estaciones base pertenecen a las operadoras de telecomunicación Telefónica, Vodafone, Orange y Más Móvil. Todas las estaciones base disponen de servicio LTE, el cual permite adaptarse para proporcionar servicio NB-IoT.

Los distintos operadores proporcionan en sus páginas web la información necesaria, o bien, para encontrar ubicaciones con cobertura, o bien, para encontrar aplicaciones que se desarrollen correctamente con la tecnología NB-IoT.

La tecnología NB-IoT se desarrolla dentro de las operadoras con las empresas como cliente objetivo sobre el cliente final. Por este motivo, no se encuentra mucha publicidad o información de los operadores en el ámbito de consumo más allá de la cobertura NB-IoT proporcionada por cada operadora o algún caso de éxito implantado en una empresa.

Regulaciones en sectores como la industria, con la obligación de la digitalización de los medidores de agua o gas, o como en el sector del automóvil, con la obligación de incorporar balizas de emergencia en los vehículos, aceleran el desarrollo de la conectividad de dispositivos con una duración de batería prolongada en el tiempo y sin requerimientos de baja latencia. En este mercado las operadoras ofrecen conectividad NB-IoT sobre los dispositivos que se desarrollen para cubrir estas necesidades. Otras industrias que encajan con la conectividad NB-IoT son el mantenimiento y el control remoto o la logística. De esta forma, las operadoras utilizan la tecnología NB-IoT para proporcionar conectividad a la empresa.

Siguiendo la línea de los casos de uso que requieren la tecnología NB-IoT, la conectividad que requieren los dispositivos NB-IoT, la obtienen mediante tarjetas SIM proporcionadas por las operadoras. Estas tarjetas SIM proporcionan acceso a la red mediante el APN correspondiente a cada operadora.

Dentro de los modos de despliegue radioeléctrico, debido a la poca disponibilidad de canales GSM o LTE, los modos de despliegue independiente o en banda no se usan habitualmente en España. Con lo que, el modo de despliegue de NB-IoT predominante en los operadores es en la *banda de guarda*.

Los operadores de telecomunicación que operan en España son Telefónica, Vodafone, Orange y Más Móvil.

2.3.1 Telefónica

Los servicios proporcionados por Telefónica a través de los dispositivos M2M [39] abarcan los siguientes usos: máquinas de vending, *Smart Cities*, *E-Health*, servicios financieros, *utilities y smart metering*, gestión de flotas, y seguridad.

Dentro de los servicios enumerados, Telefónica proporciona conectividad a estos a través del despliegue de las redes LPWA. Estas redes se basan en las tecnologías LTE-M y NB-IoT. La tecnología LTE-M se utiliza en escenarios que requieran mayor capacidad de transmisión o menores de tiempos de latencia, y, de forma complementaria, la tecnología NB-IoT se utiliza en escenarios con bajo consumo de datos y difícil acceso a cobertura.

Para que un dispositivo NB-IoT acceda a la red de Telefónica se tiene que dirigir al APN “m2m.movistar.es” o al APN “movistar.es”.

Telefónica tiene desplegadas miles de tarjetas SIM configuradas para NB-IoT destinadas a dar conectividad a dispositivos de telemetría y sensorización. Un ejemplo de proyecto NB-IoT de grandes dimensiones llevado a cabo por Telefónica es [40] el despliegue de 150.000 tarjetas SIM destinadas a proporcionar conectividad a los dispositivos de control de 450.000 contadores de agua.

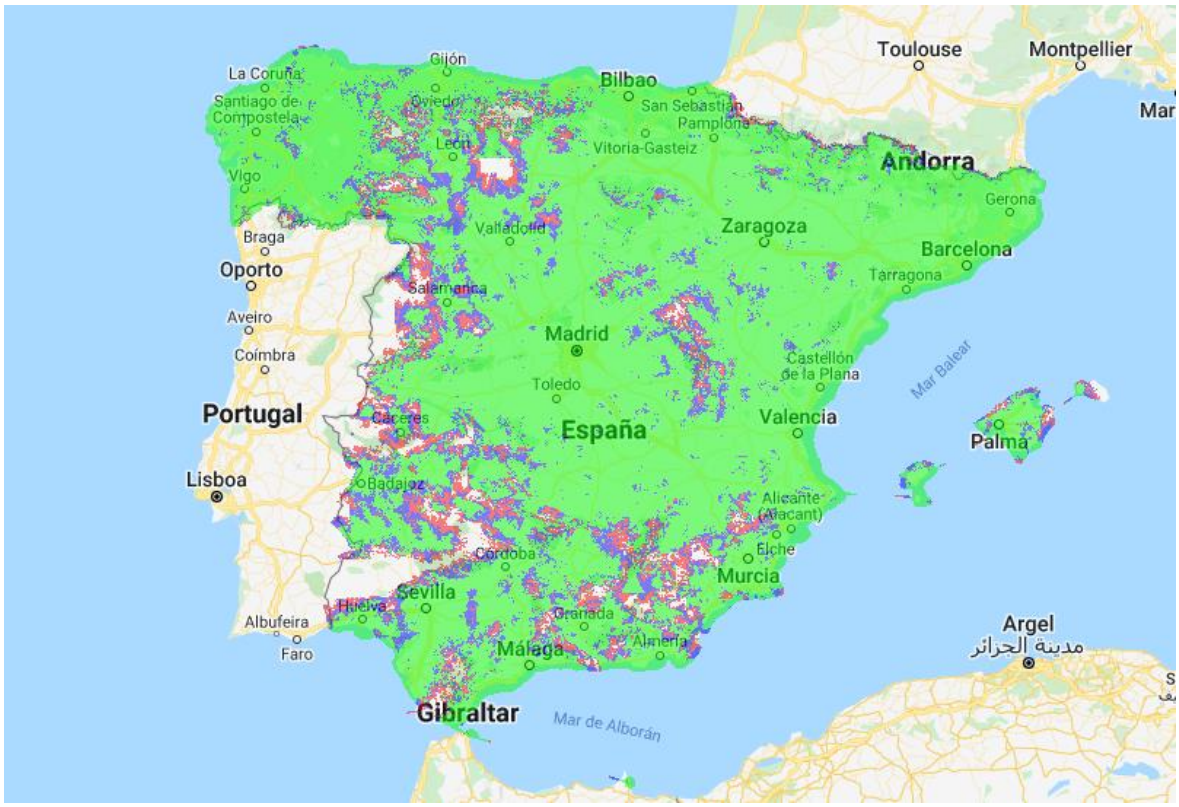
Además de proporcionar servicio a las ciudades (*Smart cities*) e industrias, Telefónica comienza la implantación de las soluciones IoT en el entorno rural (*Smart Region*) [41] mediante la sensorización de la recogida de residuos, la gestión del alumbrado y la gestión del agua.

2.3.2 Vodafone

Los servicios proporcionados por Vodafone a través de dispositivos M2M son similares a los proporcionados por Telefónica. En el caso de Vodafone, la información relativa a los precios y a la cobertura disponible en su red está más detallada que en los otros operadores.

La página web de la operadora [42] dispone de configuradores de tarifas online en función de la cantidad de dispositivos con tarjeta SIM desplegados y el tráfico esperado en estos. Además, hay descripciones de casos de uso y descripciones de las fases de operación.

Vodafone proporciona un mapa de cobertura 3G, 4G, 5G y NB-IoT en España. En la siguiente Figura se muestra el mapa de cobertura NB-IoT proporcionada por Vodafone en España.



Fuente: https://www.vodafone.es/c/conocenos/es/vodafone-espana/mapa-cobertura-movil/

Figura 18.- Mapa de cobertura NB-IoT de la red de Vodafone en España.

En la Figura 18 se observa que la mayoría del territorio español dispone de acceso a la red NB-IoT con un nivel de cobertura muy alto. En los mapas de cobertura de Vodafone, el color verde corresponde a un nivel de cobertura “muy alto”, el color azul a un nivel de cobertura “alto”, y el color rojo a un nivel de cobertura “bajo”.

Ampliando el mapa de la Figura 18, en el Sureste de Madrid, se muestra en detalle el mapa de cobertura NB-IoT en el Campus Sur de la UPM.

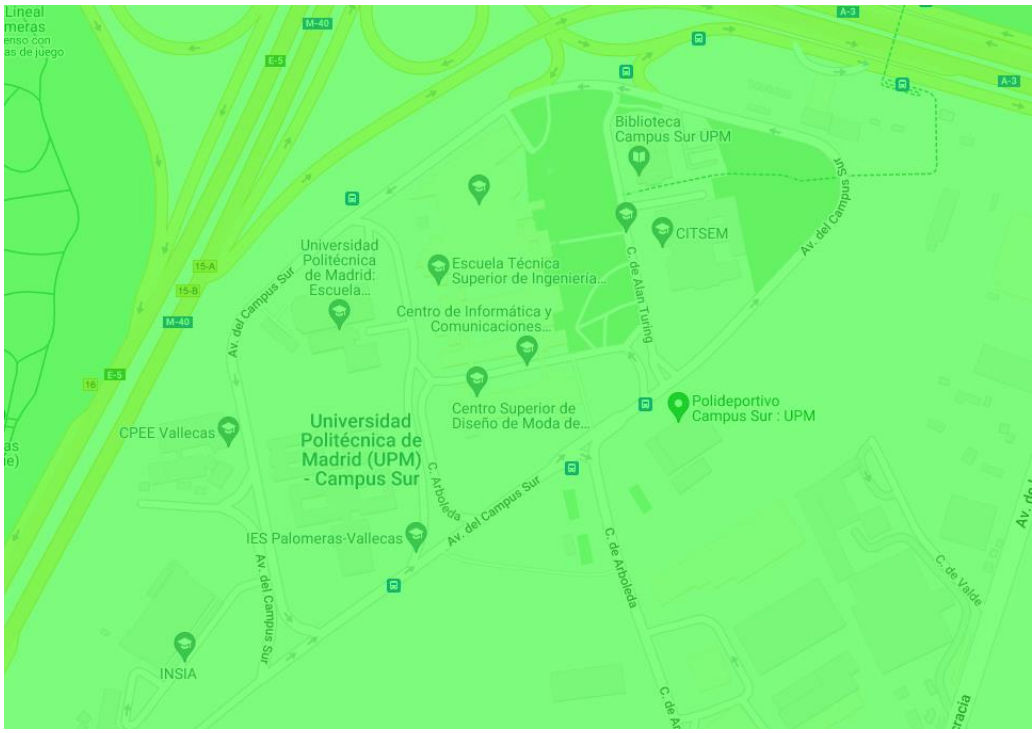


Figura 19.- Mapa de cobertura NB-IoT de la red de Vodafone en el Campus Sur de la UPM.

En la Figura 19 se observa que la cobertura que proporciona Vodafone en todo el Campus Sur de la UPM es “muy alta”.

Para que un dispositivo NB-IoT de terceros se homologue en Vodafone, requiere un testeo en los laboratorios de Vodafone donde se examina por un lado la conectividad NB-IoT, y, por otro lado, la connivencia con el resto de las tecnologías y los estándares actuales. El periodo de homologación de un dispositivo NB-IoT en Vodafone es reducido comparado con las homologaciones de la industria, durando de media 1 mes.

Para que un dispositivo NB-IoT acceda a la red de Vodafone se tiene que dirigir al APN “lpwa.vodafone.iot”. Además, por parte de Vodafone, se tiene que incluir el IMSI del dispositivo en una lista blanca para conceder la posibilidad de acceso a la red.

2.3.3 Orange y Más Móvil

Orange y Más Móvil no tienen servicio para redes NB-IoT. Tan solo Orange ofrece servicio de conectividad para la comunicación M2M, y esta comunicación está enteramente basada en LTE-M.

3 Especificación del Sistema.

Se especifica y diseña una arquitectura genérica de WSN que hace uso de una tecnología de espectro licenciado, la tecnología NB-IoT. Se implementa un prototipo demostrador de la arquitectura para la propuesta WSN. Para llevar a cabo el demostrador se proponen 2 placas de desarrollador y distintos servidores para recoger y reenviar la información recogida por los dispositivos que componen el demostrador

3.1 Requisitos del sistema

Utilizar las placas de desarrollador SARA AFF R412M y Libelium Waspmote. Ambas placas disponen de un módulo de comunicación NB-IoT, batería extraíble, y sensores de temperatura y acelerómetro incorporados. Realizar una comparación de las características de ambas placas con base en los módulos incorporados, puertos disponibles y demás características.

Son requisitos del prototipo los siguientes:

Recoger datos mediante sensores dispuestos en el Campus Sur de la UPM y seguidamente, transmitir los datos mediante paquetes UDP a un servidor central, que a su vez reenvíe los datos para su procesado a un servidor de datos.

Implementar un servidor de datos y aplicaciones en la red a través de la plataforma ThingSpeak.

Realizar la comunicación mediante NB-IoT a través del operador Vodafone. Los equipos desplegados disponen de tarjetas SIM con conectividad NB-IoT.

Determinar el método más adecuado para dotar de semántica a los datos de los sensores transmitidos por las placas.

Identificar cada envío de datos con la mota, el tipo de mota, y el nivel de voltaje de la batería. Además, dependiendo de las características de cada mota, proporcionar la hora de envío, la ubicación o la temperatura de la mota en los datos enviados.

Analizar las capacidades de la red NB-IoT interactuando con las motas utilizadas midiendo parámetros relacionados con la calidad de servicio. Medir parámetros como: la

posibilidad de conexión, la cobertura, la potencia de señal NB-IoT recibida en la mota, y la velocidad de transmisión, en distintas ubicaciones como exteriores e interiores de edificios, sótanos, y lugares de alta ocupación, dando como resultado valores de calidad de servicio.

Identificar aplicaciones para la WSN propuesta que puedan ser aplicadas en la mejora de las instalaciones del Campus Sur de la UPM con el objetivo de realizar una transición a un campus sostenible.

Realizar una demostración de comunicación entre las motas SODAQ SARA y un servidor UDP en la que se compruebe en cada envío de datos el nivel de señal en la mota. Realizar un servidor que compruebe el nivel de señal rssi, y si este valor es inferior a -70dB, que el servidor envíe un mensaje a la mota solicitando una inactividad de 10 minutos a la misma. Una vez pasado este tiempo, la mota volvería a su funcionamiento habitual.

De forma análoga, realizar una demostración de una comunicación entre las motas Libelium WASPMOTE y un servidor UDP en la que se compruebe en cada envío la temperatura ambiente. Realizar un servidor que compruebe la temperatura, y si este valor es superior a 25ºC, que el servidor envíe un mensaje a la mota solicitando una inactividad de 10 minutos a la misma. Una vez pasado este tiempo, la mota volvería a su funcionamiento habitual.

Establecer comunicaciones entre las motas y servidores de red y servidores propios, donde ambos extremos de la comunicación respondan a mensajes ad hoc. Usar un modelo de publicación-suscripción donde se publiquen mensajes en un servidor, y este servidor muestre gráficamente los datos recibidos por las motas.

Realizar un mapa de cobertura NB-IoT en el Campus Sur de la UPM.

4 Diseño e implementación

4.1 Diseño

Se ha diseñado una arquitectura de red de sensores inalámbricos dispuestos en el Campus Sur de la UPM mediante la utilización de motas sensorizadas con conexión NB-IoT.

La arquitectura de la red diseñada para el Campus Sur de la UPM se compone de 2 ámbitos diferenciados: la zona de infraestructura NB-IoT y la zona infraestructura de gestión de los datos obtenidos mediante las motas sensorizadas.

La zona de infraestructura NB-IoT comprende los dispositivos necesarios para la recogida de datos y los recursos necesarios para el transporte de los datos mediante el uso de la red NB-IoT hasta el repositorio de datos utilizado. Por otro lado, la infraestructura de gestión de los datos obtenidos mediante las motas comprende recursos utilizados para el almacenamiento, procesado y visualización de los datos. En la Figura 20 se ilustran ambas zonas de la infraestructura.

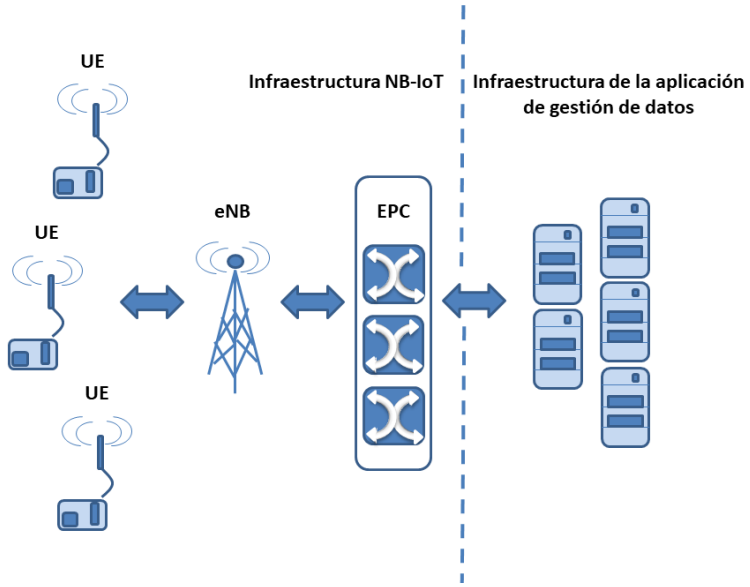


Figura 20.- Mapa de infraestructura NB-IoT e infrestructura de la aplicación de gestión de datos.

4.1.1 Diseño de la infraestructura NB-IoT

La zona de infraestructura NB-IoT comienza con las motas. Las motas son dispositivos sensorizados conectados a la red NB-IoT mediante tarjetas SIM. Las motas provienen de 2 placas de desarrollador distintas: Libelium Waspmote y SODAQ SARA. Ambas placas de desarrollador permiten introducir un código fuente codificado y compilado en C++, o Arduino, para desarrollar distintas funcionalidades. Para este proyecto, se ha desarrollado código fuente para que ambas placas lleven a cabo las tareas de: medir la potencia de señal NB-IoT recibida en las motas, medir la temperatura amiente que rodea las motas, medir su posición respecto a la tierra, además de funcionalidades descritas más adelante.

La mota SODAQ SARA está sensorizada mediante un acelerómetro y un módulo GPS, y la mota Libelium Waspmote se sensoriza mediante una placa adicional llamada “Waspmote Events Sensor Board v3.0”. Esta placa adicional tiene varios puertos para la conexión de sensores, y se elige conectar en el puerto 5 un sensor que mide la temperatura ambiente, la presión atmosférica y la humedad.

Las motas se complementan debido a que se alternan los sensores disponibles en cada una. Así, en las medidas realizadas en la comprobación del prototipo, las motas se ubican en parejas compuestas por 1 mota Libelium y una mota SODAQ, para recoger la mayor cantidad de información posible en un solo punto. Además, mediante los parámetros que ambas motas miden, se realiza una comparación entre ambas motas.

Las motas, junto a los sensores conectados a las mismas, se distribuyen a lo largo del Campus Sur situándose en ubicaciones que puedan aportar la mayor variedad de información posible. Así, se sitúan las motas en las lindes del Campus Sur, en el interior de los edificios en varias plantas, y en los exteriores, atendiendo a que se conoce la ubicación de la eNB a la que emiten las motas y, por tanto, se conoce la posibilidad de establecer Línea de Visión con la misma.

Los módulos NB-IoT que disponen las motas permiten a estas transmitir directamente la información a la eNB disponible más cercana sin la necesidad de elementos intermedios. En el caso del Campus Sur de la UPM, hay una estación base servidora de LTE disponible a decenas de metros de este, exactamente en 40.390708 lat / -3.624514 long, como se ilustra en la Figura 21, con lo cual, dado que la tecnología NB-IoT puede transmitir la señal hasta a 10km de distancia, todas las motas que se encuentran dentro del campus transmiten a la estación base mencionada. En la Figura 16, se rodea con una línea verde el Campus Sur de la UPM y con un círculo azul la estación base más cercana.

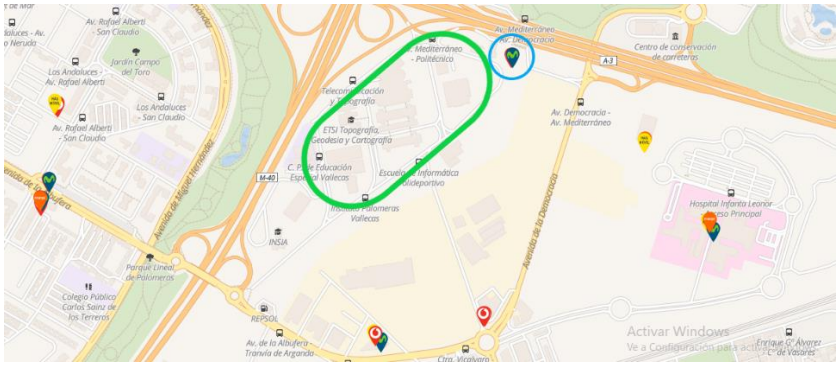


Figura 21.- Ubicación antena LTE respecto al Campus Sur de la UPM.

Los paquetes de datos generados en las motas, independientemente del protocolo de transporte utilizado, están asociados a un dominio que resuelve una dirección IP concreta, donde se ubican nuestros servidores. Así, la información transmitida desde la estación base hasta su destino a través de Internet tiene el identificador de la dirección IP destino en la capa física.

Los paquetes de datos se transportan desde el nodo eNB al EPC optimizado para NB-IoT. Y seguidamente, el EPC entrega los paquetes de datos a su destino, fuera de la estructura NB-IoT.

Para elegir la ubicación de las motas en el interior de los edificios se tiene en cuenta los espacios abiertos y la ubicación de las ventanas, tratando de ubicar las motas donde puedan recibir mayor nivel de señal. Para ubicar las motas en los exteriores se tiene en cuenta la posición de la antena y la posibilidad de bloqueo de señal debida a los edificios. Se propone colocar las motas separadas varios metros entre sí para aumentar la variedad de la información recogida.

4.1.2 Diseño de la arquitectura de gestión de datos

Por otro lado, para llevar a cabo la elección de la propuesta de arquitectura necesaria para almacenar, procesar y mostrar los datos recogidos en los sensores, se plantean 3 arquitecturas que incluyen servidores UDP, servidores MQTT y servidores de datos. De este modo, el protocolo de transporte utilizado en el interior de la infraestructura NB-IoT depende de la posición elegida para los servidores dispuestos en la arquitectura.

Las 3 arquitecturas propuestas son alternativas viables para el desarrollo de prototipos de gestión de datos. Tras un análisis de los beneficios y desventajas de cada alternativa, se propone una de ellas para el prototipo.

Las 3 alternativas se evalúan con la misma disposición de motas en el interior de la infraestructura NB-IoT.

4.1.2.1 Diseño de la aplicación gestión de datos con comunicación UDP

La primera alternativa analizada contempla el envío de paquetes UDP desde las motas hasta un servidor *ad hoc* para paquetes UDP, y la consiguiente extracción de datos del servidor. La arquitectura se ilustra en la Figura 22.

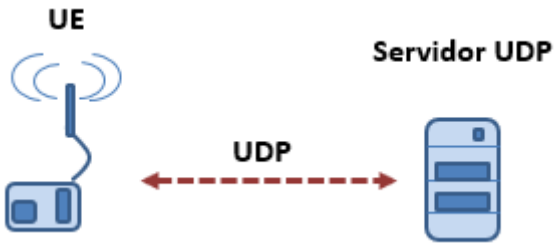


Figura 22.- Arquitectura con conexión UDP entre las motas y el servidor.

En esta arquitectura, los dispositivos finales, o motas, recogen la información mediante los sensores incorporados y los sensores conectados a las motas. Seguidamente, las motas envían la información mediante paquetes UDP a un servidor. Los paquetes UDP

están configurados con una semántica determinada para ser procesados por un servidor UDP que identifica los valores transmitidos.

El servidor UDP tiene una dirección IP y un puerto conocido por la mota emisora. Por su lado, el servidor identifica la dirección IP y el mensaje enviado por la mota al recibir un datagrama UDP. Al recibir el primer datagrama UDP, el servidor crea un fichero ".csv" en el que almacenará todos los datagramas recibidos desde el primero.

El servidor analiza cada parámetro recibido, validando los datos en función del nombre del parámetro recibido. El servidor puede analizar los parámetros y enviar una respuesta a la mota en función de los valores de los parámetros. El servidor analiza parámetros que identifican el ID de la mota, el número de envío, el nivel de señal rssi, y diversas medidas realizadas mediante los sensores de las motas.

La transmisión a través de esta arquitectura de red no está orientada al establecimiento de la conexión, de forma que esta alternativa presenta las ventajas de un consumo reducido de batería, una simplificación de la red dispuesta, y un reducido consumo de datos, por el contrario, las desventajas incluyen la posible pérdida de paquetes, así como las características inherentes a una comunicación no orientada al establecimiento.

4.1.2.2 Diseño de la aplicación de gestión de datos con comunicación MQTT

La segunda alternativa analizada contempla el establecimiento de una conexión MQTT entre los dispositivos finales y un servidor de red, como se ilustra en la Figura 23. Como servidor de red se utiliza un servidor integrado en la plataforma ThingSpeak.

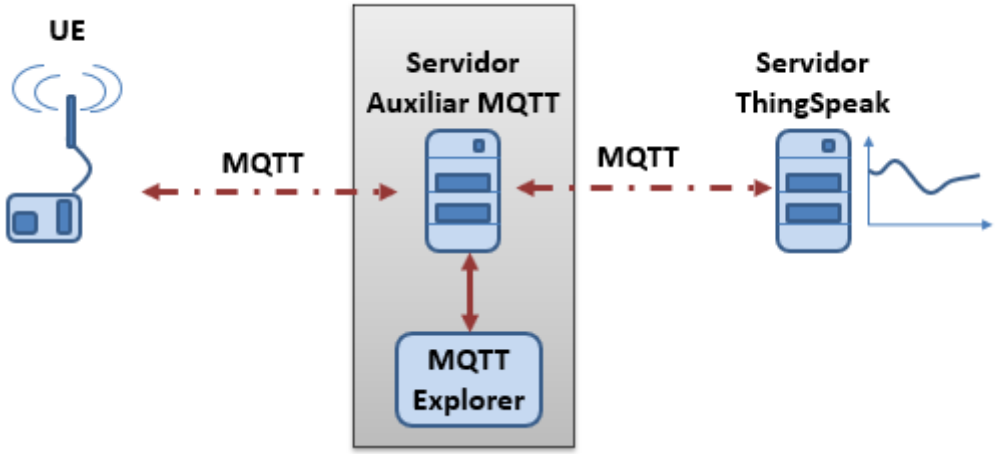


Figura 23.- Arquitectura con conexión MQTT entre motas y servidor de red ThingSpeak.

Los dispositivos finales, de nuevo, se sitúan a lo largo del Campus Sur de la UPM atendiendo a los distintos lugares en los que pueden variar las mediciones debido al medio. Además, la zona correspondiente a la infraestructura NB-IoT se mantiene, al igual que las motas, con la misma estructura hasta la entrega de los paquetes de datos al servidor.

La transmisión de datos mediante las motas, en este caso, se realiza mediante el establecimiento de una comunicación MQTT entre las motas y un servidor MQTT establecido en el servidor de red ThingSpeak. La comunicación con el servidor está

orientada al establecimiento de la conexión mediante el protocolo de transporte TCP, y el tipo de comunicación entre las motas y servidor sigue el modelo *Publish – Suscribe*.

El mismo servidor de datos dispone de herramientas para mostrar los datos recibidos mediante gráficas.

Dentro de la arquitectura de comunicación MQTT se incorporan elementos intermedios para facilitar la implementación del prototipo y aumentar el conocimiento sobre la red propuesta. Los elementos incorporados son un servidor MQTT y una herramienta de análisis de flujo de datos ilustrados en la franja gris de la Figura 23. El servidor auxiliar MQTT se sitúa entre las motas y el servidor MQTT establecido en el servidor de red ThingSpeak. Este servidor establece una comunicación MQTT tanto con las motas como con el servidor de red ThingSpeak. El servidor auxiliar reenvía la información recibida desde ambos extremos de la red al extremo opuesto en cada caso. La herramienta de medida añadida es MQTT Explorer. Esta herramienta permite analizar el flujo de información que atraviesa al servidor auxiliar proporcionando información referente a la hora, el formato y los datos del mensaje enviado a través del servidor analizado. La utilización de MQTT Explorer no supone ninguna alteración en el funcionamiento de la red.

Esta arquitectura tiene como ventajas la fiabilidad, debido a que los paquetes de datos se transmiten a través de una comunicación establecida y la redundancia, que permite recoger los datos enviados por las motas en 2 servidores distintos. Sin embargo, esta arquitectura impone un establecimiento de la conexión permanente a las motas, lo que supondría un consumo elevado de energía que se contrapone con los objetivos de la propuesta deseada.

4.1.2.3 Diseño de la aplicación de gestión de datos con comunicación mixta de UDP y MQTT

La tercera alternativa analizada contempla el envío de paquetes UDP desde las motas hasta un servidor intermedio, simultáneamente dicho servidor establece una conexión MQTT con el servidor de red ThingSpeak, y finalmente los datos transmitidos se muestran mediante gráficas en el servidor ThingSpeak, como se ilustra en la Figura 24. De esta forma, la arquitectura de red propuesta dispone de elementos situados en las 2 alternativas previas.

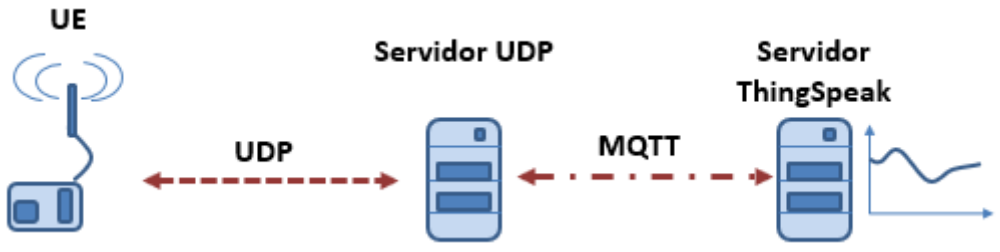


Figura 24.- Arquitectura con conexión UDP entre las motas y el servidor intermedio, y con conexión MQTT entre el servidor intermedio y el servidor de red ThingSpeak.

Los dispositivos finales, de nuevo, se sitúan a lo largo del Campus Sur de la UPM atendiendo a los distintos lugares en los que pueden variar las mediciones debido a las

condiciones de ubicación. Además, la zona correspondiente a la infraestructura NB-IoT se mantiene, al igual que las motas, con la misma estructura hasta la entrega de los paquetes de datos al servidor.

Al igual que en la primera alternativa, las motas se comunican mediante paquetes UDP con un servidor. Los paquetes UDP están configurados con una semántica determinada que permite al servidor identificar la información contenida en dichos paquetes.

Seguidamente, el mismo servidor que se comunica con las motas, también establece una comunicación MQTT con el servidor de datos ThingSpeak. La comunicación establecida con el servidor MQTT sigue el modelo *Publish – Suscribe*. También, la comunicación entre los servidores está orientada al establecimiento de la conexión, por tanto, el protocolo de transporte utilizado es TCP.

En este caso, al introducir un servidor intermedio entre las motas y el servidor de datos, no se incluye la herramienta de análisis MQTT Explorer, sino que la información se analiza almacenando los paquetes recibidos en el servidor intermedio.

El servidor UDP dispone de todas las posibilidades de análisis de la primera alternativa. También analiza la comunicación mediante la depuración de tráfico de mensajes que sucede en el servidor. La depuración del programa genera ficheros automáticamente para almacenar la información obtenida. Al igual que en la segunda alternativa, los paquetes de datos recibidos en el servidor proporcionan información referente a la hora, el formato y los datos del mensaje enviado a través de este.

El servidor dispone de opciones configurables para decidir si reenviar la información recibida, para decidir si procesar los datagramas UDP recibidos y para realizar una depuración de los mensajes que atraviesan el servidor. Estas opciones se muestran al administrador del servidor al iniciar su funcionamiento.

En este caso, se dispone de las ventajas inherentes a transmisión de la información desde las motas con un protocolo no orientado a la conexión, lo que conlleva un consumo reducido de batería, y un reducido consumo de datos. Además, dado que existe un establecimiento de la conexión entre los servidores mediante MQTT, existe un paso más de control de datos que se envían al servidor de red ThingSpeak.

4.2 Implementación

Se implementa la infraestructura NB-IoT propuesta y uno de los prototipos del diseño propuesto. La arquitectura implementada utiliza el diseño de la aplicación de gestión de datos con comunicación mixta de UDP y MQTT.

Para llevar a cabo la implementación de una arquitectura de red de sensores inalámbricos dispuestos en el campus sur de la UPM se divide la propuesta en tres partes diferenciadas: la configuración de las motas, la configuración de los servidores y las medidas realizadas.

En la infraestructura NB-IoT se diferencian 2 vías de actuación sobre las motas. En el caso de la mota Libelium WASPMOTE, se recibe asesoramiento por parte del tutor acerca del funcionamiento básico, la conectividad NB-IoT y la plataforma *Waspmote PRO IDE.* Por otro lado, el trabajo sobre la mota SODAQ SARA se realiza de forma

independiente con el objetivo realizar su puesta en marcha, analizar sus capacidades y finalmente, igual que a la mota Libelium, configurar su conectividad e integrarla en la infraestructura de NB-IoT del prototipo.

4.2.1 Implementación del código fuente de las motas

Para el desarrollo del código fuente necesario para las aplicaciones de las motas se usa la plataforma Arduino para la placa SODAQ SARA, la plataforma *Waspmote PRO IDE* para la placa Waspmote y Visual Studio para los servidores.

4.2.1.1 Implementación del software

Para el desarrollo del código fuente aplicado a las motas, se toma como base los scripts que facilitan los fabricantes de las placas de desarrollo. Seguidamente, se adaptan a las necesidades del proyecto, hasta que, finalmente, se orientan todos los scripts a la recogida y transmisión de datos desde las motas a un servidor.

El procedimiento llevado a cabo para el desarrollo ha seguido el modelo incremental, donde en primer lugar se ha testado la funcionalidad y conectividad de cada mota y seguidamente, se han incorporado funcionalidades a las mismas.

En primer lugar, se han realizado envíos de datagramas UDP a un servidor UDP realizado en Java. Para realizar el script de paquetes UDP hay que tener las siguientes consideraciones:

Aunque los datagramas UDP se envíen en las alternativas propuestas 1 y 3, el código fuente implementado se orienta a la definición de la propuesta de gestión de datos mixta.

Las librerías utilizadas se obtienen de distintas fuentes dependiendo de la mota. Para la mota Libelium WASPMOTE, la principal fuente de información es: *https://development.libelium.com/*, mientras que para la mota SODAQ SARA la principal fuente de información es: *https://github.com/* .

Al generar el socket, es necesario proporcionar el APN, la dirección IP pública del servidor, el código de red de operador y la banda utilizada. En el caso de las tarjetas SIM proporcionadas por Vodafone, no es necesario proporcionar el APN y el código de red, puesto que la tarjeta SIM tiene estos valores predefinidos.

Para acceder al servidor ubicado en la dirección IP pública se utiliza el puerto 37415.

Una vez comprobada la comunicación UDP, se define la estructura del formato utilizado en los paquetes UDP. Se utiliza un formato de clave y valor, con una separación de “:” entre la clave y el valor, y una separación de un espacio “ “ para separar los distintos campos.

4.2.1.2 Implementación de las tramas

El datagrama UDP generado por las motas varía en cada tipo de mota, pues cada una dispone de sensores distintos. El datagrama UDP generado por la mota SODAQ SARA incluye los siguientes campos: ID de la mota; ID del canal SODAQ en la plataforma ThingSpeak; Número de datagrama; Voltaje de la batería; Nivel de señal RSSI; Latitud;

Longitud. Cada campo se corresponde con una abreviación con el objetivo de optimizar el envío de datos. Así los campos se corresponden de la siguiente forma:

- ID de la mota idMota
- ID del canal SODAQ en la plataforma ThingSpeak id
- Número de datagrama cont
- Voltaje de la batería volt
- Nivel de señal RSSI rssi
- Latitud lat
- Longitud long

Estos campos generan el siguiente datagrama ejemplo:

idMota:7 id:1 cont:22 volt:3995 rssi:-71 lat:40.432293 long:-3.711822

La mota SODAQ SARA incorpora un módulo GPS que requiere varios segundos o minutos para determinar su ubicación en función de la exposición de la mota al cielo abierto, con su consiguiente consumo de energía adicional. Para limitar el tiempo que la mota dedica al cálculo de su posición se codifica un *timeout* para cada ciclo de medidas. Los parámetros actualizados mediante el módulo GPS son latitud y longitud.

Por otro lado, el datagrama generado por la mota Libelium WASPMOTE incluye los siguientes campos: ID de la mota; ID del canal Libelium en la plataforma ThingSpeak; Hora EPOCH; Número de datagrama; Voltaje de la batería; Nivel de señal RSSI; Temperatura; Humedad; Presión atmosférica. Cada campo se corresponde con una abreviación con el objetivo de optimizar el envío de datos. Así los campos se corresponden de la siguiente forma:

- ID de la mota idMota
- ID del canal SODAQ en la plataforma ThingSpeak id
- Hora EPOCH hora
- Número de datagrama cont
- Voltaje de la batería volt
- Nivel de señal RSSI rssi
- Temperatura temp
- Humedad humd
- Presión atmosférica pres

Estos campos generan el siguiente datagrama ejemplo:

idMota:4 id:2 created_at:946686800 cont:14 volt:4.31 rssi:-79 temp:24.75 humd:34.76 pres:94126.7

Al inicio de cada ciclo de medidas se extrae la hora en formato EPOCH.

En ambos casos las tramas identifican los parámetros que transmiten seguido del valor de cada parámetro. El número de datagrama se regula mediante un contador y la actualización de los valores transmitidos se realiza mediante medidas programadas antes de la transmisión de cada datagrama.

4.2.2 Implementación de los servidores en una arquitectura WSN

Para la implementación de los servidores UDP a los que transmiten las motas, se usan 2 alternativas, la codificación en Java y la codificación en Python. Ambos servidores corresponden con un mensaje de vuelta a la mota. La implementación de la versión final del servidor del Servidor UDP se realiza en Python, y tiene los siguientes pasos:

1. Servidor UDP con recepción y respuesta.

Servidor UDP que permite la apertura de un socket, a través del cual recibe el mensaje y envía la confirmación de recepción.

2. Servidor UDP con recepción, respuesta y almacenamiento de datos.

Para almacenar los datos recibidos en el servidor, tras la apertura del socket se crea un archivo ".csv" que genera una nueva línea por cada datagrama que recibe el servidor, y a su vez, clasifica cada dato atendiendo a la separación de un espacio " " en el interior del datagrama. Además, incluye un mecanismo para evitar el bloqueo del archivo ".csv".

3. Servidor UDP con recepción, respuesta, almacenamiento de datos y reenvío de datos a servidor de red ThingSpeak.

Antes de implementar el paso 3, que incluye un reenvío de datos al servidor ThingSpeak desde el servidor UDP, se implementa la comunicación MQTT ThingSpeak entre mota y servidor que se detalla más adelante. Así, se emula el funcionamiento MQTT entre la mota y el servidor MQTT en la comunicación del servidor UDP con el servidor MQTT, ya que establece la comunicación, al igual que la mota, introduciendo el ID del canal deseado y el *API key* (clave API) correspondiente.

Tras recibir cada datagrama UDP en el servidor y para que el servidor de red ThingSpeak procese correctamente los datos, se procesa en el servidor cada datagrama incorporando cada tipo de dato a un diccionario. El diccionario se compone de una lista de parámetros que tienen un valor asociado. Este parámetro actualiza su valor asociado si la mota transmite ese parámetro en el datagrama UDP enviado al servidor. Una vez actualizados los valores en el diccionario con el último datagrama recibido, se envían los datos respetando el formato:

field1=valor1&field2=valor2&field3=valor3

4. Servidor UDP con recepción, respuesta, almacenamiento de datos en función del tipo de mota y tipo de dato recibido, y reenvío de datos a servidor de red ThingSpeak.

El servidor UDP gestiona datos generados por 2 tipos de motas que generan un conjunto de datos diferentes entre sí. Por tanto, se incluye la clasificación y almacenamiento de datos en función del tipo de mota utilizada para transmitir. Se clasifican las motas SODAQ SARA con el identificador "1" y las motas Libelium WASPMOTE con el identificador "2". Mediante los diccionarios, se almacena la información correspondiente a cada tipo de dato. Cada identificador tiene asignado un ID de canal y una API key correspondiente en el servidor de datos ThingSpeak. Una vez asignado el canal, el mensaje reenviado se compone a través del diccionario, seleccionando los parámetros que se envían en cada tipo de mota. Así, para el siguiente datagrama UDP enviado desde la mota SODAQ SARA:

idMota:7 id:1 cont:22 volt:3995 rssi:-71 lat:40.432293 long:-3.711822

El servidor UDP reenvía el siguiente mensaje al canal dedicado a la mota SODAQ SARA en el servidor de red ThingSpeak:

field1=22&field2=3995&field3=-71&lat=40.432293&long=-3.711822

En el extremo final, el servidor de red ThingSpeak tiene esquemas de representación de datos distintos en cada canal para adaptarse a la información transmitida por cada tipo de mota.

5. Servidor UDP con recepción, respuesta atendiendo a la información recibida, almacenamiento de datos en función del tipo de mota y tipo de dato recibido, y reenvío de datos a servidor de red ThingSpeak.

Se incluye el análisis de 2 parámetros incluidos en los diccionarios generados, el nivel de señal rssi en la mota SODAQ SARA, representado como “rssi” en el diccionario, y la temperatura en la mota Libelium WASPMOTE, representada como “temp” en el diccionario.

El servidor incluye comprobaciones de formato de datagrama UDP enviado por las motas, entre las que se encuentran: la comprobación del ID del servidor al que se desea retransmitir la información y los parámetros “cont”, “volt”, ”rssi”, ”temp”, ”humd” y ”pres”.

Para llevar a cabo una propuesta de arquitectura más detallada, el servidor puede iniciarse en los siguientes modos:

- help Mediante el comando -h, muestra las opciones de ejecución.
- DEBUG Mediante el comando -d, ejecuta el programa en modo DEBUG.
- LISTEN Mediante el comando -l, procesa los datagramas UDP recibidos.
- THINGSPEAK Mediante el comando -ts, reenvía la información a ThingSpeak.

El dominio público utilizado para albergar el servidor UDP se establece mediante un servicio web de la página www.no-ip.com. El nombre del dominio utilizado para albergar los servidores utilizados es: “nbiot-csupm.ddns.net" Este dominio corresponde a la dirección IP pública en la que se encuentra el servidor. Para hacer llegar la información al servidor se realiza un enrutado en el router usando como referencia el puerto 37415.

4.2.3 Implementación de la comunicación MQTT

Para la implementación de la comunicación MQTT entre las motas y el servidor de red ThingSpeak, de forma análoga al establecimiento de la conexión con el servidor UDP, se utilizan los scripts que proporcionan los fabricantes, y progresivamente se les añaden funcionalidades.

Para establecer la comunicación es necesario proporcionar: el APN, el perfil URAT, el perfil de la SIM, el operador, el dominio web, y un puerto de entrada.

El dominio web mediante el que se accede a la comunicación MQTT con el servidor de red ThingSpeak es “mqtt.thingspeak.com”.

El puerto de entrada utilizado es el puerto común a los servidores MQTT que no disponen de encriptación, el 1883.

Para el envío de datos desde las motas al servidor de red, en primer lugar, se establece la comunicación de la mota con la red NB-IoT, seguidamente, se establece la conexión MQTT con el servidor ThingSpeak. Luego, periódicamente y en función del tipo de mota se realizan medidas de intensidad de señal, temperatura, presión, humedad o ubicación. En el bucle, tras la obtención de los datos, se genera el mensaje que se envía al servidor MQTT. El mensaje se compone con el *topic* y el *payload*, con la siguiente estructura:

- El *topic* incluye el canal al que se envía el mensaje con su correspondiente clave de escritura, o *API key*, y la selección de "*publish*" para publicar en dicho canal. En el mensaje *"idMota:7 id:1 cont:22 volt:3995 rssi:-71 lat:40.432293 long:-3.711822"*, el siguiente fragmento corresponde al *topic*

  channels/161036/publish/

- El *payload* incluye los resultados de las medidas realizadas previamente. Al igual que la comunicación entre el servidor UDP y el servidor de red, el *payload* sigue un formato de asignación de campos. En el mensaje *"idMota:7 id:1 cont:22 volt:3995 rssi:-71 lat:40.432293 long:-3.711822",* el siguiente fragmento corresponde al *payload*:

  field1=22&field2=3995&field3=-71&lat=40.432293&long=-3.711822

Los datos transmitidos en la comunicación MQTT entre las motas SODAQ SARA y el servidor de red son: nº de envío, voltaje de la batería, nivel de señal rssi, latitud y longitud.

Los datos transmitidos en la comunicación MQTT entre las motas Libelium WASPMOTE y el servidor de red son: nº de envío, voltaje de la batería, nivel de señal rssi, temperatura, humedad y presión atmosférica.

Para realizar el demostrador de comunicación mota-servidor se crea un sistema de comprobación de valores en el servidor mediante el cual, cuando se cumple una regla predefinida, el servidor solicita a la mota que entre en modo Deep Sleep durante un tiempo determinado. Así, se comprueba que en el sistema propuesto hay una comunicación bidireccional.

El sistema propuesto para la mota SODAQ SARA solicita a la mota entrar en Deep Sleep durante 600 segundos en caso de que la mota mida y comparta un paquete de datos que indique un nivel de señal rssi inferior a -70 dBm. Tras probar asignando distintos valores de rssi a la condición, se valida que el sistema funciona correctamente.

El sistema propuesto para la mota Libelium WASPMOTE, aprovechando la incorporación de la placa de eventos, solicita a la mota entrar en estado Deep Sleep durante 600 segundos en caso de que la mota mida y comparta un paquete de datos que indique una temperatura ambiente superior a 25ºC. Tras probar asignando distintos valores de temperatura ambiente a la condición, se valida que el sistema funciona correctamente.

Para la implementación del código necesario para que los 2 tipos de motas establezcan una comunicación MQTT con el servidor de datos ThingSpeak, se utiliza un servidor

auxiliar por el que pasan las comunicaciones en ambos sentidos. Este servidor auxiliar está implementado mediante la herramienta *mosquitto* y funciona como un servidor MQTT.

La particularidad de uso que añade la utilización de este servidor intermedio consiste en la adición de el fragmento de mensaje “gx/ts/” como prefijo del mensaje, donde x es un número entre 1 y 9. Los mensajes que recibe el servidor intermedio con este prefijo, se reenvían al dominio del servidor ThingSpeak eliminando dicho sufijo del mensaje enviado.

En el servidor auxiliar se utiliza la herramienta MQTT Explorer. Mediante la herramienta de análisis MQTT Explorer, se puede obtener información acerca del mensaje enviado como se ilustra en la Figura 25.

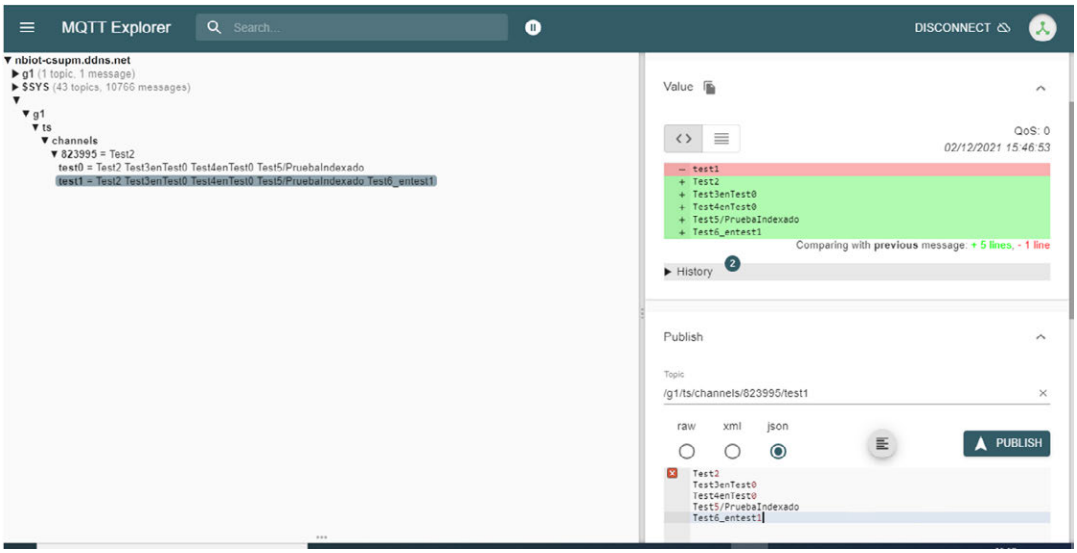


Figura 25.- Ventana de análisis de datos de MQTT Explorer.

En la parte izquierda de la Figura 25 se observa el nombre del host del servidor MQTT que se analiza: “nbiot-csupm.ddns.net”. Tras el nombre del dominio, se observa un árbol de entradas formado por los temas, o *topic*. En el caso del servidor intermedio, como se ha comentado previamente, las entradas “gx/ts/” corresponden a mensajes redirigidos al servidor de red ThingSpeak.

En la parte de derecha de la Figura 25 se observa el mensaje publicado en el servidor MQTT. En este mensaje se distingue, de izquierda a derecha, la referencia a los canales del servidor, el identificador del canal, el campo que se actualiza, y el valor utilizado para actualizar el campo.

Por otro lado, esta herramienta también permite escribir mensajes en el servidor MQTT analizado. De esta forma, al usar un servidor auxiliar al servidor de red, se pueden enviar datos al servidor auxiliar para testar el comportamiento del servidor de red.

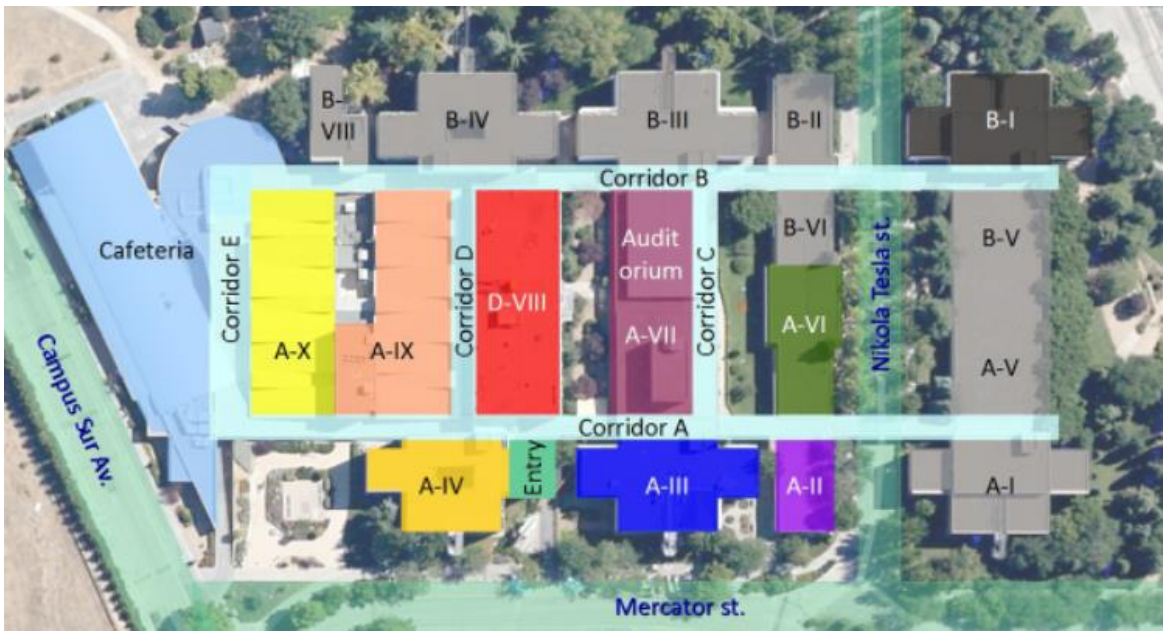
En las 3 alternativas analizadas se han enviado paquetes UDP a un servidor UDP, se ha establecido una comunicación MQTT con un servidor MQTT, y se ha realizado una mezcla de ambos sistemas. Tras analizar las alternativas, se decide que el prototipo propuesto utilice un servidor UDP para comunicarse con las motas y un servidor MQTT para comunicarse con el servidor de red ThingSpeak. Así, la propuesta de WSN se realizará siguiendo la tercera alternativa.

4.2.4 Implementación de las medidas

La distribución de las motas en el prototipo propuesto para el Campus Sur de la UPM se basará en la información obtenida tras realizar un estudio de cobertura. La planificación de la topología de la red se ha realizado con base a una forma de estrella impuesta por NB-IoT, en la cual los dispositivos finales se comunican directamente con la estación base, que a su vez se comunica mediante internet con el servidor central.

Para la exposición de los datos finales se ha usado la herramienta de visualización de datos ubicada en el servidor de datos ThingSpeak. La presentación de los datos se ha hecho mediante plantillas proporcionadas por la plataforma.

Seguidamente, se analizan las ubicaciones en las que se realizan las medidas de calidad de servicio. En la siguiente figura se ilustran los edificios que componen la Escuela Técnica Superior de Ingeniería y Sistemas de Telecomunicación (ETSIST), la Escuela Técnica Superior de Ingeniería de Sistemas Informáticos (ETSISI), y el Centro Superior de Diseño de Moda de Madrid (CSDMM). Los edificios que componen la ETSIST se colorean en la Figura 26.



Fuente: https://www.etsist.upm.es/escuela/instalaciones

Figura 26.- Bloques de edificios del Campus Sur.

Se realizan medidas el Aula Reina Sofía (A-II), en la entrada principal a la ETSIST (Entry), en los pasillos principales (Corridor A, B, C, D y E), en la zona común frente a cafetería y en todas las plantas de los bloques en los que hay aulas de la ETSIST (B-I, A-III, A-IV, A-VIII)

Los bloques de la ETSIST A-VI, A-X, y A-IX corresponden al edificio de administración y las aulas de examen, y no se toman medidas en esos bloques.

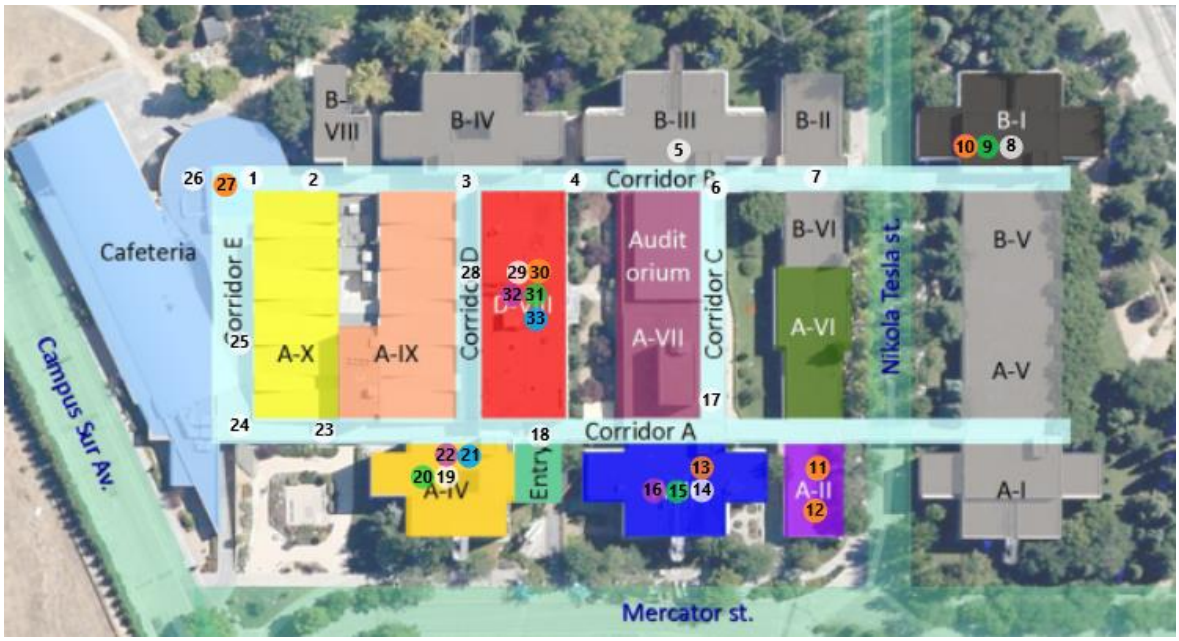


Figura 27.- Bloques de edificios del Campus Sur con los puntos de medida.

En los bloques con aulas de la ETSIST se analizan todas las plantas, de forma que se establece un código de colores para identificar la medida de cada planta. En la siguiente tabla se identifica el color de cada planta:

Número de planta	Código de colores
0	Naranja
1	Gris
2	Verde
3	Morado
4	Azul

Tabla 2 .- Tabla de código de colores para número de plantas

Otro edificio ubicado en el Campus Sur que es objeto de estudio es el Centro de Investigación en Tecnologías Software y Sistemas Multimedia para la Sostenibilidad (CITSEM). Se realizan medidas en los pasillos y zonas comunes del mismo. En la siguiente Figura se ilustran las ubicaciones del CITSEM en las que se realizan medidas:

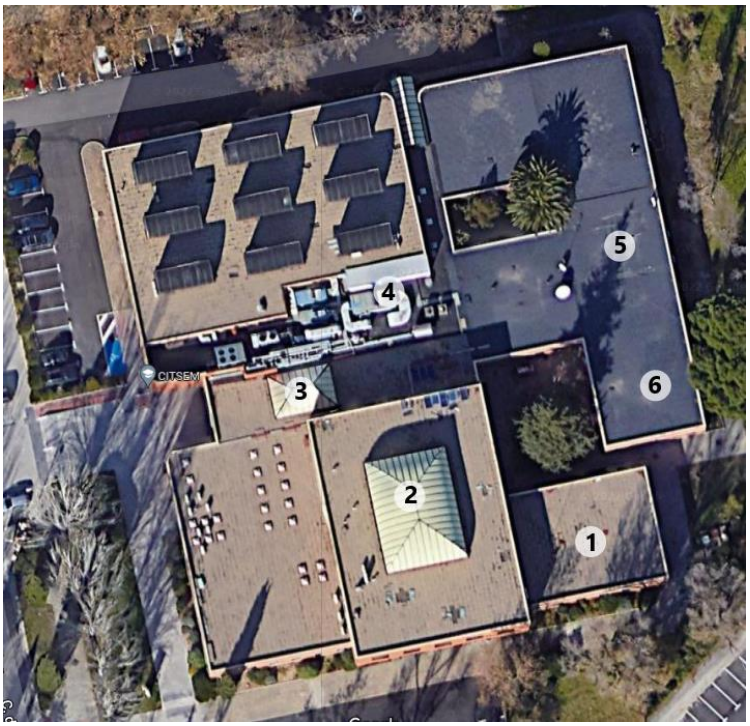


Figura 28.- Instalaciones CITSEM con los puntos de medida.

Una vez analizados los edificios, se lleva a cabo el análisis de los exteriores del Campus Sur. Se toman medidas en las carreteras que recorren el Campus, debajo de los pasillos A y B en zonas de paso, y en los alrededores de la biblioteca. En la Figura 29 se ilustran los puntos de medida numerados.

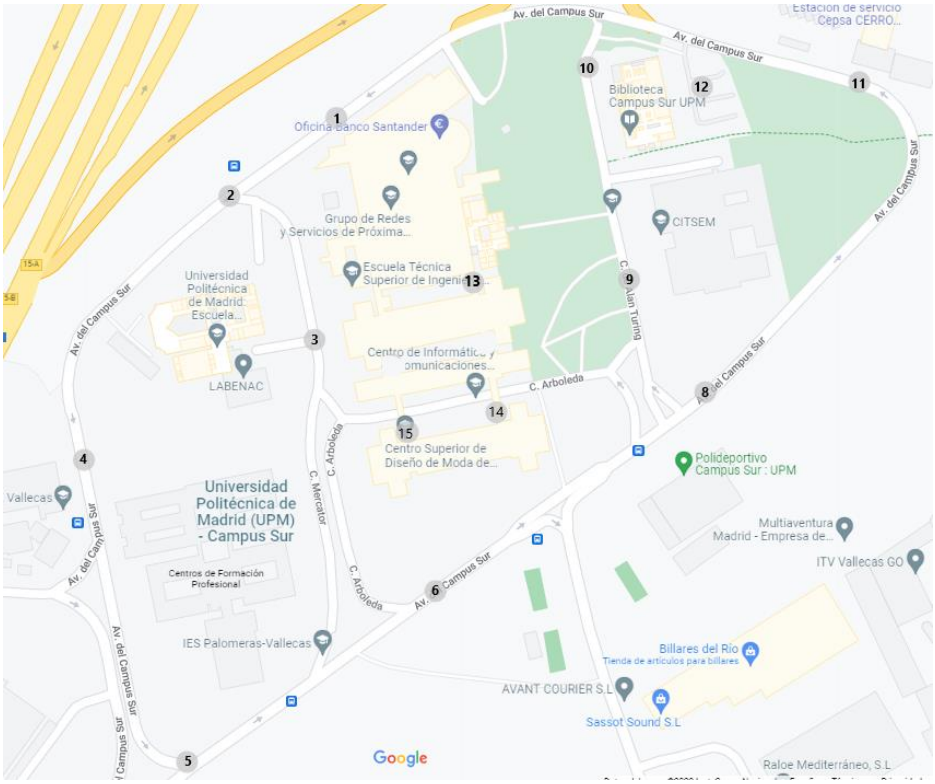


Figura 29.- Campus Sur con los puntos de medida en el exterior.

5 Resultados

Para comprobar la idoneidad de la arquitectura propuesta, se toman medidas en distintas ubicaciones del Campus Sur de la UPM utilizando simultáneamente las 2 motas sensorizadas descritas anteriormente. Los parámetros que se utilizan para comprobar la validez de la arquitectura son la medida del nivel de señal rssi y el porcentaje de éxito en el envío de paquetes al servidor ThingSpeak, es decir, el éxito de envío de información entre los 2 extremos de la arquitectura.

Para representar gráficamente los niveles de señal medidos se utiliza el código de colores que muestra en la siguiente tabla.

Nivel de señal rssi	Código de colores
De -51 a -57	Verde oscuro
De -58 a -64	Verde claro
De -65 a -71	Amarillo
De -72 a -77	Naranja
De -78 a -85	Rojo
No hay señal	Negro

Tabla 3 .- Tabla de código de colores para nivel de señal

5.1 Medidas en el edificio de las Escuelas de Telecomunicación, Informática y Diseño

Las medidas en el Campus Sur de la UPM se realizan mediante 2 tipos de motas: SODAQ SARA y Libelium WASPMOTE. A continuación, se muestran las medidas tomadas en los edificios del Campus Sur de la UPM:

Figura 30.- Medidas nivel de señal rssi de NB-IoT de la mota SODAQ SARA en los edificios de la ETSIST

Figura 31.- Medidas nivel de señal rssi de NB-IoT de la mota Libelium WASPMOTE en los edificios de la ETSIST.

En líneas generales, ambas motas funcionan correctamente en los edificios del Campus Sur. Sin embargo, hay 2 zonas en las que el nivel de señal rssi imposibilita la comunicación de las motas. Estas zonas son el bloque B-I, donde la mota SODAQ y Libelium pierden la señal en la primera y segunda planta del edificio respectivamente, y el bloque D-VIII, donde en la planta cero y alguna planta intermedia hay señal muy baja o no hay señal para sendas motas.

Además, en los puntos de medida 22 y 23, en los que se observa que el nivel de señal rssi es nulo para la mota SODAQ SARA, finalmente, tras realizar una segunda ronda de medias en dicha zona, se concluye que el nivel medio de señal rssi para la mota SODAQ en los puntos 22 y 23 es de -71, por lo tanto, tan solo los bloques B-1 y D-VIII tienen deficiencias de cobertura.

Para la obtención de los valores de nivel de rssi en cada ubicación de medida se toman entre 5 y 10 medidas en sendas motas que más tarde se procesan para conseguir el valor mostrado en el mapa. Se retiran las medidas próximas al desplazamiento de una ubicación de medida a otra y, además, se realiza una selección de los 5 mejores valores de medida en cada ubicación y se realiza una media aritmética de estas medidas.

El otro parámetro usado para determinar la calidad de servicio en una ubicación es el porcentaje de éxito en el envío de paquetes de datos. Se calcula desde que las motas están estables en un punto y comienzan a transimtir hasta que las motas cambian de ubicación de medida. En la siguiente tabla se ilustra en porcentaje de éxito de las motas en el envío de entre 5 y 10 paquetes UDP al servidor:

Ubicación	SODAQ SARA	
	Nivel de señal rssi	% Envíos completados
1	-67	100,00
2	-57.8	100,00
3	-71.4	100,00
4	-58.2	100,00
5	-65.8	100,00
6	-68.6	100,00
7	-68.2	100,00
8	-70.6	88,89
9	-67	66,67
10	-	-
11	-79.8	90,91
12	-77.4	94,12
13	-72.6	100,00
14	-78.2	100,00
15	-70.6	100,00
16	-59.4	100,00
17	-70.6	100,00
18	-75.8	100,00
19	-73.8	100,00
20	-63.4	100,00
21	-54.6	100,00
22	-56.2	90,91
23	-71	71.42
24	-71	71.42
25	-71.4	83,33
26	-61	100,00
27	-77.4	83,33
28	-67	100,00
29	-85	100,00
30	-	-
31	-73	77,78
32	-71	83,33
33	-67	33,33

Ubicación	Libelium WASPMOTE	
	Nivel de señal rssi	% Envíos completados
1	-64.3	100,00
2	-68	100,00
3	-75	66,00
4	-61	100,00
5	-73.6	100,00
6	-76	100,00
7	-71	100,00
8	-66.3	100,00
9		-
10	-73.8	71,43
11	-83	100,00
12	-85	50,00
13	-78.6	100,00
14	-78.3	100,00
15	-73.8	75,00
16	-65.8	100,00
17	-74.6	100,00
18	-73	100,00
19	-80.2	100,00
20	-71	66,00
21	-67	100,00
22	-60.6	100,00
23	-71	50,00
24	-75	50,00
25	-69.8	83,33
26	71.6	75,00
27	-75	100,00
28	-81	100,00
29	-81.5	100,00
30	-	-
31	-75	40,00
32	-	-
33	-65.6	100,00

Tabla 4 .- Tabla de medidas en los envíos en los edificios

Se observa que el porcentaje de éxito en los envíos de datos disminuye en el bloque B-I, el bloque D-VIII y en el pasillo A frente al bloque A-X. La mayoría de las ubicaciones permiten a las motas tener éxito en todos los envíos realizados.

Se reafirma que la comunicación en los puntos analizados y, por consiguiente, en los edificios del Campus Sur de la UPM es viable y de calidad. Sin embargo, se observa que, a pesar de que el mapa de cobertura proporcionado por Vodafone indica que la señal NB-

IoT en el Campus Sur de la UPM es muy alta, hay algunas zonas en el Campus Sur de la UPM que debido a la topografía y las edificaciones no tiene un nivel de cobertura muy alto.

5.2 Medidas en el CITSEM

El otro edificio analizado situado en el Campus Sur es el CITSEM. Las medidas realizadas en los pasillos y zonas comunes del mismo indican que la arquitectura propuesta funciona correctamente en el mismo. A continuación, se muestran las medidas de nivel de señal rssi realizadas en el CITSEM:

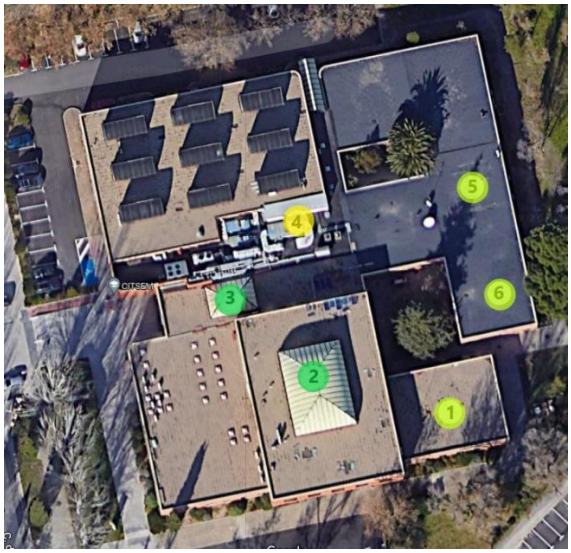


Figura 32.- Medidas nivel de señal rssi de NB-IoT de la mota SODAQ SARA en el CITSEM.

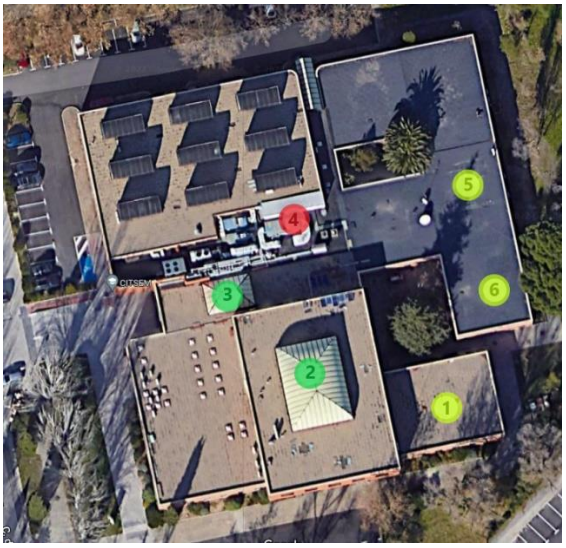


Figura 33.- Medidas nivel de señal rssi de NB-IoT de la mota Libelium WASPMOTE en el CITSEM.

Se observa que los niveles de señal permiten desarrollar la comunicación entre las motas y el servidor. Además, las motas tienen niveles de intensidad de señal similares. De modo que se valida la arquitectura propuesta para el CITSEM del Campus Sur.

5.3 Medidas en los exteriores de los edificios del campus

Por último, para finalizar con el análisis de calidad de servicio en el campus, se realizan medidas en los exteriores del Campus Sur de la UPM. A continuación, se muestran las medidas realizadas en los exteriores del Campus Sur.

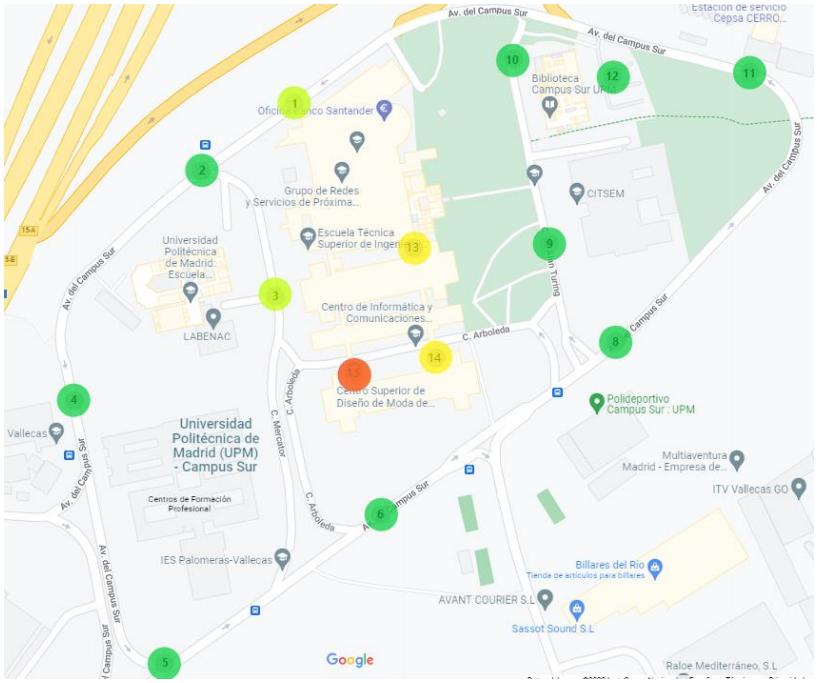


Figura 34.- Medidas nivel de señal rssi de NB-IoT en la mota SODAQ SARA en el Campus Sur de la UPM.

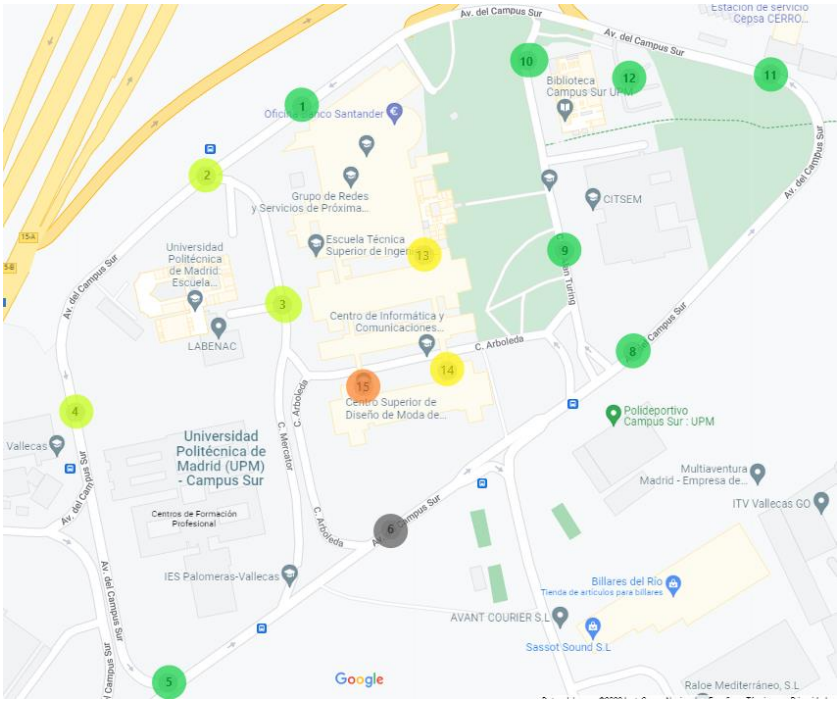


Figura 35.- Medidas nivel de señal rssi de NB-IoT en la mota Libelium WASPMOTE en el Campus Sur de la UPM.

Se observa que los peores valores de calidad de servicio se dan debajo de los pasillos que conectan los edificios de la ETSIST, aunque siguen siendo valores aceptables. Por otro lado, hay un servicio nulo en el punto 6 para la mota WASPMOTE. Sin embargo, la mota SODAQ dispone de muy buena señal en ese punto.

Para completar el análisis de la calidad de servicio en los exteriores del Campus Sur de la UPM, se calcula el porcentaje de éxito en el envío de paquetes desde las motas al servidor de datos. En la siguiente tabla se ilustra el nivel de señal rssi y el porcentaje de éxito de las motas en el envío de entre 5 y 10 datagramas UDP al servidor.

Ubicación	SODAQ SARA	
	Nivel de señal rssi	% Envíos completados
1	-57	100,00
2	-54.6	100,00
3	-57.8	91,67
4	-56.2	100,00
5	-51	100,00
6	-53	100,00
7	-51	100,00
8	-51	100,00
9	-51	100,00
10	-51	100,00
11	-51	100,00
12	-51	100,00
13	-66.6	87,5
14	-71.8	42,86
15	-73	88,23

Ubicación	Libelium WASPMOTE	
	Nivel de señal rssi	% Envíos completados
1	-57.6	75,00
2	-63	87,50
3	-63.2	100,00
4	-58.6	100,00
5	-51	100,00
6	-51	-
7	-51.3	100,00
8	-56.7	100,00
9	-51.2	100,00
10	-51	100,00
11	-51	100,00
12	-51	100,00
13	-71.5	100,00
14	-70.2	100,00
15	-75	66,66

Tabla 5 .- Tabla de medidas en los envíos en los exteriores

En los exteriores del Campus sur, los 2 modelos de mota tienen un nivel de señal rssi similar, y se alternan en el 100% de éxito en los envíos.

Dada la complementación de las motas Libelium y WASPMOTE parámetros de calidad de servicio, el diseño propuesto puede funcionar con los 2 tipos de mota trabajando de forma paralela asegurando conectividad en casi todas las ubicaciones del Campus Sur de la UPM.

Unido a la calidad de servicio, sendas motas proporcionan distinta información más allá del nivel de señal rssi, proporcionando la ubicación por parte de la mota SODAQ, y proporcionando medidas de temperatura, presión y humedad por parte de la mota Libelium mediante la placa de sensores. Debido a esto, el diseño propuesto cuenta con los 2 tipos de mota distribuidos en el Campus Sur de la UPM.

Además, ambas motas permiten el tratamiento de datos simultáneo por parte del servidor mediante la composición de un formato único para la transmisión de datos por parte de ambas motas. De esta forma, ambas motas pueden funcionar conjuntamente en la misma plataforma como se ha expuesto.

Para la propuesta de red, una pareja de motas sensorizadas mantiene una comunicación con muy buena calidad de señal en el Campus Sur de la UPM.

6 Presupuesto

En el presupuesto se desarrollan los costes de implementación del prototipo (recursos de hardware y software), así como el coste de los recursos humanos empleados en el desarrollo de este. El coste de los productos mostrados no incluye IVA.

6.1 Presupuesto de los materiales

La siguiente tabla describe los recursos de hardware utilizados en el proyecto:

Ítem	Unidades	Coste
SODAQ SARA AFF	1	118.52€
Libelium WASPMOTE	1	190.00€
Placa de sensores Libelium	1	60.00€
Módulo NB-IoT WASPMOTE	1	199.00€
Alquiler portátil ASUS K55V	1	200.00€

Tabla 6 .- Tabla de coste de recursos hardware

En el cálculo del presupuesto de los materiales no se tiene en cuenta el equipamiento del que se dispone previamente, como son el espacio de trabajo, los ordenadores de los que se dispone y el monitor utilizado. Tampoco se incluyen los costes de luz asociados al desarrollo del proyecto.

Por otro lado, en la siguiente tabla se describen los recursos de software utilizados en el proyecto:

Ítem	Unidades	Coste
Waspmote PRO IDE	1	0.00€
Arduino	1	0.00€
Visual Studio Code	1	0.00€
Eclipse Java IDE	1	0.00€

Tabla 7 .- Tabla de coste de recursos software

Al tratarse de software libre, el software utilizado para desarrollar el prototipo no tiene un coste adicional sobre el proyecto.

6.2 Presupuesto de los recursos humanos

El coste correspondiente a los recursos humanos corresponde con el coste de un ingeniero de desarrollo durante el tiempo requerido para realizar el proyecto. Se considera que el ingeniero desarrolla el proyecto a tiempo completo durante 320 horas. A continuación, se estima el coste de hora de trabajo:

- Salario bruto anual de un ingeniero de desarrollo: 33000.00 €
- $33000.00\ \frac{€}{año} * \frac{\text{año}}{52\ semanas} * \frac{\text{semana}}{40\ horas} * \frac{€}{año} = 15{,}86\ \frac{€}{hora}$

En la siguiente tabla se muestra el coste de los recursos humanos:

Descripción	Coste
Salario bruto por hora de trabajo	15.86 €
Duración del proyecto	320 horas
Coste estimado de desarrollo de proyecto	**5076.92€**

Tabla 8 .- Tabla de coste de recursos humanos

6.3 Coste total

EL coste total del proyecto se obtiene sumando los costes parciales mostrados anteriormente. En la siguiente tabla se muestra el coste total del proyecto:

Descripción	Coste
Recursos hardware	767.52€
Recursos software	0.00€
Recursos humanos	5076.92€
TOTAL	**5844.44€**

Tabla 9 .- Tabla de coste total de los recursos

7 Conclusiones y trabajos futuros

7.1 Conclusiones

La tecnología NB-IoT es útil para aplicaciones con bajo consumo de datos como la telemetría de valores como la temperatura, luminosidad, movimiento, o humedad, que requieren tan solo de algunos bytes para transmitirse.

La tecnología NB-IoT se encuentra fuera del ámbito de consumo y se centra en el sector empresarial. En este sector encaja institución educativa UPM.

En el aspecto de durabilidad de los proyectos que utilicen esta tecnología, por un lado, se tiene testada la duración de las baterías que usan los dispositivos mediante mecanismos de ahorros de energía y, por otro lado, la integración de dicha tecnología con el estándar actual de comunicaciones móviles, que asegura el presente y futuro del espectro radioeléctrico utilizado.

La tecnología está en las primeras fases de implantación. En este sentido, los fabricantes de placas de desarrollo facilitan a sus clientes scripts con los que establecer comunicaciones NB-IoT disponiendo SIM con conectividad NB-IoT y su placa.

He desarrollado un diseño de WSN para el Campus Sur de la UPM que se compone de una infraestructura NB-IoT y 3 alternativas para la gestión de datos mediante software. Para demostrar la viabilidad de los 3 prototipos propuestos se realizan pruebas de comunicación. Además, el prototipo que contempla la comunicación mixta de UDP y MQTT también valida mediante la toma de medidas en el Campus Sur de la UPM. Este prototipo también se usa para demostrar la comunicación bidireccional mediante una DEMO para cada mota.

Hay una antena de telecomunicaciones que da servicio LTE y NB-IoT muy próxima al Campus Sur de la UPM. Esta antena proporciona un nivel de señal NB-IoT muy alto a la mayoría de las ubicaciones donde se realizan medidas, pero debido a la topografía y a los edificios del campus, el nivel de cobertura es inferior al esperado en algunas zonas. Con todo, se puede llevar a cabo la implantación de la infraestructura NB-IoT en el Campus Sur de la UPM.

Dada la cobertura existente, se puede ubicar las motas en la mayor parte del campus. Dado que la tecnología NB-IoT está orientada a la comunicación de centenares de

dispositivos en una sola red, se podría situar uno o varios dispositivos en cada aula de los edificios sin perjuicio de que la red sufriese deterioro.

El prototipo propuesto dispone de un servicio de nube que permite la visualización de los datos mediante la plataforma del servidor de red ThingSpeak, que además de permitir configurar la visualización, permite compartir los datos recogidos en tiempo real.

7.2 Trabajos futuros

Realizar las medidas de ahorro de batería de las motas para prolongar el periodo de funcionamiento sin reemplazo o recarga de batería.

Calcular el tiempo de descarga de las motas dependiendo de la frecuencia de envío de mensajes y de la hibernación de las motas.

Mejorar la propuesta de codificación de las tramas para optimizar el envío de información. Estudiar la posibilidad de dotar de semántica los datos transmitidos.

El servidor de red ThingSpeak que se encuentra en el prototipo propuesto dispone de funcionalidades no explotadas en este proyecto. Dispone de herramientas de análisis de datos a través de Matlab y de herramientas de comunicación con otras plataformas o bases de datos. Así, en trabajos futuros se puede llevar a cabo un procesamiento de la información recogida en el Campus Sur de la UPM.

Bibliografía

[1] F. Hillebrand, «The creation of standards for global mobile communication: GSM and UMTS standardization from 1982 to 2000», *IEEE Wirel. Commun.*, vol. 20, n.º 5, Art. n.º 5, oct. 2013, doi: 10.1109/MWC.2013.6664470.

[2] «LTE». https://www.3gpp.org/technologies/keywords-acronyms/98-lte (accedido 21 de noviembre de 2021).

[3] «LTE-Advanced». https://www.3gpp.org/technologies/keywords-acronyms/97-lte-advanced (accedido 24 de enero de 2022).

[4] H. Althumali y M. Othman, «A Survey of Random Access Control Techniques for Machine-to-Machine Communications in LTE/LTE-A Networks», *IEEE Access*, vol. 6, pp. 74961-74983, 2018, doi: 10.1109/ACCESS.2018.2883440.

[5] R. A. Rashid y R. Yusoff, «Bluetooth Performance Analysis in Personal Area Network (PAN)», en *2006 International RF and Microwave Conference*, sep. 2006, pp. 393-397. doi: 10.1109/RFM.2006.331112.

[6] T. Elarabi, V. Deep, y C. K. Rai, «Design and simulation of state-of-art ZigBee transmitter for IoT wireless devices», en *2015 IEEE International Symposium on Signal Processing and Information Technology (ISSPIT)*, dic. 2015, pp. 297-300. doi: 10.1109/ISSPIT.2015.7394347.

[7] «Near Field Communication Technology Standards – NearFieldCommunication.org». http://nearfieldcommunication.org/technology.html (accedido 17 de febrero de 2022).

[8] «Weightless SIG | Weightless LPWAN», *Weightless SIG*. https://www.weightless-alliance.org (accedido 30 de noviembre de 2021).

[9] «Sigfox España». https://www.sigfox.es/ (accedido 20 de febrero de 2022).

[10] «What is LoRaWAN® Specification», *LoRa Alliance®*. https://lora-alliance.org/about-lorawan/ (accedido 30 de noviembre de 2021).

[11] A. Zourmand, A. L. Kun Hing, C. Wai Hung, y M. AbdulRehman, «Internet of Things (IoT) using LoRa technology», en *2019 IEEE International Conference on Automatic Control and Intelligent Systems (I2CACIS)*, jun. 2019, pp. 324-330. doi: 10.1109/I2CACIS.2019.8825008.

[12] P.-C. Hsieh, Y. Jia, D. Parra, y P. Aithal, «An Experimental Study on Coverage Enhancement of LTE Cat-M1 for Machine-Type Communication», en *2018 IEEE International Conference on Communications (ICC)*, may 2018, pp. 1-5. doi: 10.1109/ICC.2018.8422888.

[13] S. Dawaliby, A. Bradai, y Y. Pousset, «In depth performance evaluation of LTE-M for M2M communications», en *2016 IEEE 12th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*, oct. 2016, pp. 1-8. doi: 10.1109/WiMOB.2016.7763264.

[14] M. Chen, Y. Miao, Y. Hao, y K. Hwang, «Narrow Band Internet of Things», *IEEE Access*, vol. 5, pp. 20557-20577, 2017, doi: 10.1109/ACCESS.2017.2751586.

[15] M. Lukić, Ž. Mihajlović, y I. Mezei, «Data Flow in Low-Power Wide-Area IoT Applications», en *2018 26th Telecommunications Forum (TELFOR)*, nov. 2018, pp. 1-4. doi: 10.1109/TELFOR.2018.8611848.

[16] «MQTT - The Standard for IoT Messaging». https://mqtt.org/ (accedido 26 de febrero de 2022).

[17] A. Stanford-Clark y H. L. Truong, «MQTT For Sensor Networks (MQTT-SN) Protocol Specification», p. 28, 2013.

[18] «CoAP — Constrained Application Protocol | Overview», 26 de febrero de 2022. https://coap.technology/ (accedido 26 de febrero de 2022).

[19] N. Mangalvedhe, R. Ratasuk, y A. Ghosh, «NB-IoT deployment study for low power wide area cellular IoT», en *2016 IEEE 27th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC)*, sep. 2016, pp. 1-6. doi: 10.1109/PIMRC.2016.7794567.

[20] B. Yang, L. Zhang, D. Qiao, G. Zhao, y M. A. Imran, «Narrowband Internet of Things (NB-IoT) and LTE Systems Co-Existence Analysis», en *2018 IEEE Global Communications Conference (GLOBECOM)*, dic. 2018, pp. 1-6. doi: 10.1109/GLOCOM.2018.8647484.

[21] M. Kanj, V. Savaux, y M. Le Guen, «A Tutorial on NB-IoT Physical Layer Design», *IEEE Commun. Surv. Tutor.*, vol. 22, n.º 4, Art. n.º 4, 2020, doi: 10.1109/COMST.2020.3022751.

[22] B. Schulz, «NB-IoT Measurements», p. 62.

[23] L. Feltrin *et al.*, «Narrowband IoT: A Survey on Downlink and Uplink Perspectives», *IEEE Wirel. Commun.*, vol. 26, n.º 1, pp. 78-86, feb. 2019, doi: 10.1109/MWC.2019.1800020.

[24] G. Tsoukaneri, M. Condoluci, T. Mahmoodi, M. Dohler, y M. K. Marina, «Group Communications in Narrowband-IoT: Architecture, Procedures, and Evaluation», *IEEE Internet Things J.*, vol. 5, n.º 3, pp. 1539-1549, jun. 2018, doi: 10.1109/JIOT.2018.2807619.

[25] H. Kim, S. C. Cho, Y. Lee, y J. Shin, «Performance Analysis of NB-IoT System According to Operation Mode», en *2019 International Conference on Information and Communication Technology Convergence (ICTC)*, oct. 2019, pp. 876-878. doi: 10.1109/ICTC46691.2019.8939718.

[26] Y.-J. Yu y J.-K. Wang, «NPRACH-Aware Link Adaptation and Uplink Resource Allocation in NB-IoT Cellular Networks», *IEEE Trans. Veh. Technol.*, vol. 70, n.º 5, pp. 4894-4906, may 2021, doi: 10.1109/TVT.2021.3069272.

[27] Micro-/Nano-Electronic System Integration R&D Center (MESIC), Department of Electronic Science and Technology, University of Science and Technology of China (USTC), Hefei, Anhui, 230027, PR China, M. S. Ali, Y. Li, S. Chen, y F. Lin, «Narrowband Internet of Things: Repetition-Based Coverage Performance Analysis of Uplink Systems», *J. Commun.*, pp. 293-302, 2018, doi: 10.12720/jcm.13.6.293-302.

[28] K. K. Nair, A. M. Abu-Mahfouz, y S. Lefophane, «Analysis of the Narrow Band Internet of Things (NB-IoT) Technology», en *2019 Conference on Information Communications Technology and Society (ICTAS)*, mar. 2019, pp. 1-6. doi: 10.1109/ICTAS.2019.8703630.

[29] R. Ratasuk, B. Vejlgaard, N. Mangalvedhe, y A. Ghosh, «NB-IoT system for M2M communication», en *2016 IEEE Wireless Communications and Networking Conference*, abr. 2016, pp. 1-5. doi: 10.1109/WCNC.2016.7564708.

[30] F. Michelinakis, A. S. Al-Selwi, M. Capuzzo, A. Zanella, K. Mahmood, y A. Elmokashfi, «Dissecting Energy Consumption of NB-IoT Devices Empirically», *IEEE Internet Things J.*, vol. 8, n.º 2, Art. n.º 2, ene. 2021, doi: 10.1109/JIOT.2020.3013949.

[31] S.-M. Oh, K.-R. Jung, M. Bae, y J. Shin, «Performance analysis for the battery consumption of the 3GPP NB-IoT device», en *2017 International Conference on Information and Communication Technology Convergence (ICTC)*, oct. 2017, pp. 981-983. doi: 10.1109/ICTC.2017.8190831.

[32] «How to Enable Power Saving Modes of NB-IoT and Cat-M and the Energy Consumption Expected», *Digi-Key Electronics*.

https://www.digikey.com/en/articles/how-to-enable-power-saving-modes-of-nb-iot-and-cat-m (accedido 25 de junio de 2022).

[33] M. Taneja, «LTE-LPWA networks for IoT applications», en *2016 International Conference on Information and Communication Technology Convergence (ICTC)*, oct. 2016, pp. 396-399. doi: 10.1109/ICTC.2016.7763505.

[34] «Overview - SODAQ Support pages», 20 de octubre de 2021. https://support.sodaq.com/Boards/Sara_AFF/ (accedido 20 de octubre de 2021).

[35] «Waspmote, la plataforma de sensores para desarrollar proyectos de IoT», *Libelium ES*, 20 de octubre de 2021. https://www.libelium.com/es/productos-iot/waspmote/ (accedido 20 de octubre de 2021).

[36] «akorIoT SensPRO the reference design for your NB-IoT / LTE-M application», *akorIoT*. https://www.akoriot.com/akoriot_senspro/ (accedido 25 de junio de 2022).

[37] «Portenta CAT.M1/NB IoT GNSS Shield», *Arduino Official Store*. https://store.arduino.cc/products/portenta-catm1 (accedido 25 de junio de 2022).

[38] «AntenasGSM.com», *AntenasGSM.com*. https://antenasgsm.com/ (accedido 16 de mayo de 2022).

[39] «Conectividad Gestionada M2M. Grandes Clientes - Movistar». https://www.movistar.es/grandes-empresas/soluciones/fichas/machine-to-machine-m2m/ (accedido 16 de mayo de 2022).

[40] L. Sacristán, «Telefónica instalará en España 450.000 contadores inteligentes de agua con tecnología NB-IoT», *Xataka Móvil*, 30 de junio de 2021. https://www.xatakamovil.com/movistar/telefonica-instalara-espana-450-000-contadores-inteligentes-agua-tecnologia-nb-iot (accedido 21 de mayo de 2022).

[41] «Telefónica España inicia el despliegue del Proyecto Territorio Rural Inteligente en Castilla y León basado en tecnologías NB-IoT», *Telefónica*, 11 de octubre de 2018. https://www.telefonica.com/es/sala-comunicacion/telefonica-espana-inicia-el-despliegue-del-proyecto-territorio-rural-inteligente-en-castilla-y-leon-basado-en-tecnologias-nb-iot/ (accedido 21 de mayo de 2022).

[42] «Vodafone NarrowBand IoT». https://www.vodafone.es/c/empresas/es/narrowband-iot/ (accedido 22 de noviembre de 2021).

Anexos

Anexo A Código de las motas

En este anexo se definen los programas usados por las motas SODAQ SARA AFF y Libelium WASPMOTE en el desarrollo del prototipo de WSN.

A.1 Ficheros de SODAQ SARA AFF

Los ficheros incluidos en este apartado contienen el código utilizado en la mota SODAQ SARA AFF.

Los ficheros son el demostrador de comunicación UDP, el demostrador de comunicación MQTT y el demostrador de comunicación bidireccional.

A.1.1 Fichero NB_IoT_udp_ThingSpeak.ino

Este es el fichero utilizado para transmitir datagramas UDP desde la mota SODAQ SARA AFF.

```
/*****************************************************************
 *  NB_IoT_13_udp_ThingSpeak.ino
 *  Descripción:
 *  Este script permite a la mota enviar paquetes UDP al servidor "nbiot-csupm".
 *  Los paquetes enviados incluyen la siguiente info:
 *  ID de la mota; Tipo de mota; Nº de paquete; Voltaje de la batería; Nivel se señal RSSI; Latitud; Longitud.
 *  Autor:      José María Gutiérrez García
 *  Versión:    1.0
 *  Fecha:      2022/03/15
 *****************************************************************/
//Librerias
#include "Sodaq_nbIOT.h"
#include "Sodaq_wdt.h"
#include <Arduino.h>
#include <Sodaq_UBlox_GPS.h>

#define R4XX

#define DEBUG_STREAM SerialUSB
#define MODEM_STREAM Serial1
#define powerPin SARA_ENABLE
#define enablePin SARA_TX_ENABLE
#define gpsEnablePin GPS_ENABLE

#define DEBUG_STREAM SerialUSB
#define DEBUG_STREAM_BAUD 115200

#define ADC_AREF 3.3f
#define BATVOLT_R1 4.7f
#define BATVOLT_R2 10.0f
#define BATVOLT_PIN BAT_VOLT

#ifndef enablePin
    #define enablePin -1
#endif

#define STARTUP_DELAY 10000
#define LONG_DELAY    1000000
//Medida del voltaje de la batería
static inline bool is_timedout(uint32_t from, uint32_t nr_ms) __attribute__((always_inline));
static inline bool is_timedout(uint32_t from, uint32_t nr_ms){return (millis() - from) > nr_ms;}
```

```
uint16_t getBatteryVoltage(){
    uint16_t voltage = (uint16_t)((ADC_AREF / 1.023) * (BATVOLT_R1 + BATVOLT_R2) / BATVOLT_R2 * (float)analogRead(BATVOLT_PIN));
    return voltage;
}
//Configuración NB-IoT
const char* apn = "lpwa.vodafone.iot";
const char* ip = "nbiot-csupm.ddns.net";
uint8_t     cid = 1;
const uint8_t band = 20;
const char* forceOperator = "";
//Variables
char    strBuffer[500];
char    dataDecode[100] = "";
const uint32_t timeout = 20;
bool signale = false;
int8_t cont;
int8_t    rssi_val;
uint8_t   ber_val;
uint16_t  volts;
int8_t    volts_val;
float lat_val;
float long_val;

Sodaq_nbIOT nbiot;

void sendMessageThroughUDP()
{   //Medida calidad de señal
    cont = cont + 1;
    signale = nbiot.getRSSIAndBER(&rssi_val,&ber_val);
    volts     = getBatteryVoltage();
    DEBUG_STREAM.println("captura señal extendida");
    DEBUG_STREAM.println("rssi:");
    DEBUG_STREAM.println(rssi_val);
    DEBUG_STREAM.println("Voltios:");
    DEBUG_STREAM.print(volts);
    DEBUG_STREAM.println(volts_val);

    DEBUG_STREAM.println("6. Sending message through UDP");

    int localPort = 37415;
    int socketID = nbiot.createSocket(localPort);
```

```
    if (socketID >= 7 || socketID < 0) {
        DEBUG_STREAM.println("Failed to create socket");
        return;
    }

    DEBUG_STREAM.println("7. Created socket!");
    snprintf(strBuffer,sizeof(strBuffer),"idMota:7 id:1 cont:%d volt:%d rssi:%d lat:%f long:%f",cont,volts,rssi_val, lat_val, long_val);
    size_t size = strlen(strBuffer);

    int lengthSent = nbiot.socketSend(socketID,ip, 37415, strBuffer);
    DEBUG_STREAM.println("8. Envía en el socket");

    DEBUG_STREAM.print("9. String length vs sent: ");
    DEBUG_STREAM.print(size);
    DEBUG_STREAM.print(" vs ");
    DEBUG_STREAM.println(lengthSent);
    nbiot.closeSocket(socketID);
    DEBUG_STREAM.println("12. Cerramos Socket");
    DEBUG_STREAM.println();
}

void setup()
{   //Configuración comunicación NB-IoT
    sodaq_wdt_safe_delay(STARTUP_DELAY);
    cont = 0;
    lat_val = 0;
    long_val = 0;
    DEBUG_STREAM.begin(115200);
    DEBUG_STREAM.println("1. Initializing and connecting... ");

#ifdef R4XX
    MODEM_STREAM.begin(115200);
    DEBUG_STREAM.println("2. post getbaudrate");
    nbiot.setDiag(DEBUG_STREAM);
    nbiot.init(MODEM_STREAM, powerPin, enablePin, SARA_R4XX_TOGGLE, cid);
    sodaq_gps.init(gpsEnablePin);
    DEBUG_STREAM.println("3. Post init()");
#else
    MODEM_STREAM.begin(nbiot.getDefaultBaudrate());
    DEBUG_STREAM.println("2.b. No está definido R4XX");
```

```
    nbiot.setDiag(DEBUG_STREAM);
    nbiot.init(MODEM_STREAM, powerPin, enablePin, -1, cid);
#endif

if(SARA_R4XX_TOGGLE != -1){
    DEBUG_STREAM.println("4.a Sabemos que entra en _isSaraR4XX ");

}

#ifdef SARA_RESET
    DEBUG_STREAM.println("4.b. Entra al RESET");
    pinMode(SARA_RESET, OUTPUT);
    digitalWrite(SARA_RESET, HIGH);
#endif

#ifndef R4XX
    DEBUG_STREAM.println("4. Entra a Override (y a R4XX)");
    nbiot.overrideNconfigParam("CR_0354_0338_SCRAMBLING", true);
#endif

    if (!nbiot.connect("",ip,"", 20)) {
        DEBUG_STREAM.println("Failed to connect to the modem!, primer fallo");
    }
    DEBUG_STREAM.println("5. Mandamos UDP desde setup");
    sendMessageThroughUDP();
}

void loop()
{   //Medida GPS
    sodaq_wdt_safe_delay(STARTUP_DELAY);
    if(sodaq_gps.scan(false,40L * 1000)){
      lat_val = sodaq_gps.getLat();
      long_val = sodaq_gps.getLon();
      DEBUG_STREAM.println(lat_val);
      DEBUG_STREAM.println(long_val);
    }else{
      DEBUG_STREAM.println("No hay datos de GPS");
    }
    if (!nbiot.isConnected()) {
        if (!nbiot.connect("",ip, "", 20)) {
            DEBUG_STREAM.println("Failed to connect to the modem!, esperamos 1 minuto");
```

```
        }
    }
    else {
        DEBUG_STREAM.println("Mandamos UDP desde el loop");
        sendMessageThroughUDP();
    }
}
```

A.1.2 Fichero mqtt_thingSpeak.ino

Este es el fichero utilizado para establecer una comunicación MQTT desde la mota SODAQ SARA AFF.

```
/*************************************************************
 *  mqtt_thingSpeak.ino
 *  Descripción:
 *  Este script permite a la mora establecer una comunicación MQTT el  servidor "ThingSpeak"
 *  En las publicaciones hechas en el servidor ThingSpeak se transmiten los siguientes parámetros:
 *  ID de la mota; Tipo de mota; Nº de paquete; Voltaje de la batería;
 *  Nivel se señal RSSI, RSRQ, RSRP, Latitud; Longitud
 *  Los mesajes enviados por la mota tienen el formato clave y valor que requiere la plataforma ThingSpeak
 *  Autor:      José María Gutiérrez García
 *  Versión:    1.0
 *  Fecha:      2022/03/15
 *************************************************************/
//Librerias
#include <Sodaq_R4X.h>
#include <Sodaq_wdt.h>
#include <Arduino.h>
#include <Sodaq_UBlox_GPS.h>

#define CONSOLE_STREAM   SerialUSB
#define MODEM_STREAM     Serial1
#define CONSOLE_BAUDRATE 115200
#define MODEM_BAUDRATE   115200
//Configuración NB-IoT
#define CURRENT_APN         "lpwa.vodafone.iot"
#define CURRENT_OPERATOR    SODAQ_R4X_AUTOMATIC_OPERATOR    //"0"
#define CURRENT_URAT        SODAQ_R4X_NBIOT_URAT            //8
#define CURRENT_MNO_PROFILE MNOProfile::VODAFONE            //19
#define NBIOT_BANDMASK      "524288"                        //Banda 20
//Configuración Servidor ThingSpeak
#define CLIENT_ID         1606136                           //Canal SODAQ SARA AFF
#define DEVICE_ID         "DRQMJTAUKig4IgAPLj0iLCM"         //MQTT Client ID
#define USER              "DRQMJTAUKig4IgAPLj0iLCM"
#define PASS              "bGRiVoiJlJ2v+84UfZ4HLFCe"
#define MQTT_SERVER_NAME  "mqtt.thingspeak.com"             //Servidor de ThingSpeak
#define MQTT_SERVER_PORT  1883
```

```
#ifndef NBIOT_BANDMASK
#define NBIOT_BANDMASK    BAND_MASK_UNCHANGED
#endif

#ifndef enablePin
    #define enablePin -1
#endif

#define ADC_AREF 3.3f
#define BATVOLT_R1 4.7f
#define BATVOLT_R2 10.0f
#define BATVOLT_PIN BAT_VOLT

static Sodaq_R4X r4x;
static Sodaq_SARA_R4XX_OnOff saraR4xxOnOff;
static bool isReady;
//Medida del voltaje de la batería
static inline bool is_timedout(uint32_t from, uint32_t nr_ms) __attribute__((always_inline));
static inline bool is_timedout(uint32_t from, uint32_t nr_ms){return (millis() - from) > nr_ms;}
uint16_t getBatteryVoltage(){
    uint16_t voltage = (uint16_t)((ADC_AREF / 1.023) * (BATVOLT_R1 + BATVOLT_R2) / BATVOLT_R2 * (float)analogRead(BATVOLT_PIN));
    return voltage;
}
//Variables
ext_sig_qual_t es_quality;
bool signale;
bool signale_b;
int8_t    rsrq_val;
int8_t    rsrp_val;
int8_t    rssi_val;
uint8_t   ber_val;
uint16_t  volts;
int8_t    volts_val;
int cont;
float lat_val;
float long_val;

void setup()
{
    es_quality.rsrq = 0;
    es_quality.rsrp = 0;
```

```
    lat_val = 0;
    long_val = 0;
    cont =0 ;
    while ((!CONSOLE_STREAM) && (millis() < 10000)){

    }
    //Establacimiento comunicación MQTT
    CONSOLE_STREAM.begin(CONSOLE_BAUDRATE);

    r4x.setDiag(CONSOLE_STREAM);
    r4x.init(&saraR4xxOnOff, MODEM_STREAM, MODEM_BAUDRATE);
    sodaq_gps.init(GPS_ENABLE);

    isReady = r4x.connect(CURRENT_APN, CURRENT_URAT, CURRENT_MNO_PROFILE, CURRENT_OPERATOR, BAND_MASK_UNCHANGED, NBIOT_BANDMASK);

    CONSOLE_STREAM.println(isReady ? "Network connected" : "Network connection failed");

    if (isReady) {
        isReady = r4x.mqttSetServer("mqtt.thingspeak.com",1883 ) && r4x.mqttLogin();
        CONSOLE_STREAM.println(isReady ? "MQTT connected" : "MQTT failed");
    }

    CONSOLE_STREAM.println("Setup done");
}

void loop()
{   //Medida calidad de señal
    cont++;
    signale   = r4x.getExtendedSignalQuality(&es_quality);
    rsrq_val  = (int8_t)  es_quality.rsrq;
    rsrp_val  = (int8_t) es_quality.rsrp;
    signale_b = r4x.getRSSIAndBER(&rssi_val,&ber_val);
    volts     = getBatteryVoltage();

    CONSOLE_STREAM.println("captura señal extendida");
    CONSOLE_STREAM.println("rssi:");
    CONSOLE_STREAM.println(rssi_val);
    CONSOLE_STREAM.println("Voltios:");
    CONSOLE_STREAM.println(volts);
    CONSOLE_STREAM.println("rsrq:");
    CONSOLE_STREAM.println(rsrq_val);
```

```
    CONSOLE_STREAM.println("rsrp:");
    CONSOLE_STREAM.println(rsrp_val);
    //Medida GPS
    if(sodaq_gps.scan(false,400L * 1000)){
      CONSOLE_STREAM.println("Post GPS scan");
      lat_val = sodaq_gps.getLat();
      long_val = sodaq_gps.getLon();
      CONSOLE_STREAM.println(lat_val);
      CONSOLE_STREAM.println(long_val);
    }else{
      CONSOLE_STREAM.println("No hay datos de GPS");
    }
    //Transmisión
    char buf1 [150];
    snprintf(buf1,sizeof(buf1),"field1=%d&field2=%d&field3=%d&field4=%d&field5=%d&lat=%f&long=%f",cont, volts, rssi_val, rsrq_val,
rsrp_val,lat_val,long_val);
    r4x.mqttPublish("channels/1606136/publish/3OW0CCAOHYTEA3BV", (const uint8_t*)buf1,strlen(buf1) ,0,0,0);
    delay(30000);
}
```

A.1.3 Fichero NB_IoT_udp_thingSpeak.ino con DEMO

Este es el fichero utilizado para transmitir datagramas UDP desde la mota SODAQ SARA AFF y demostrar la comunicación bidireccional entre la mota SODAQ SARA AFF y el servidor UDP.

```
/*****************************************************************
 *  NB_IoT_udp_ThingSpeak.ino
 *  Descripción:
 *  Este script permite a la mota enviar paquetes UDP al servidor "nbiot-csupm".
 *  Los paquetes enviados incluyen la siguiente info:
 *  ID de la mota; Tipo de mota; Nº de paquete; Voltaje de la batería; Nivel se señal RSSI; Latitud; Longitud.
 *  Este script contiene una DEMO de comunicación bidireccional de paquetes UDP con el servidor.
 *  La DEMO introduce una pausa en la ejecución del cógido de 1000 segundos en caso de que el mensaje recibido contenga "NOK".
 *  Autor:      José María Gutiérrez García
 *  Versión:    1.0
 *  Fecha:      2022/03/15
 *****************************************************************/
//Librerias
#include "Sodaq_nbIOT.h"
```

```
#include "Sodaq_wdt.h"
#include <Arduino.h>
#include <Sodaq_UBlox_GPS.h>

#define R4XX

#define DEBUG_STREAM SerialUSB
#define MODEM_STREAM Serial1
#define powerPin SARA_ENABLE
#define enablePin SARA_TX_ENABLE
#define gpsEnablePin GPS_ENABLE

#define DEBUG_STREAM SerialUSB
#define DEBUG_STREAM_BAUD 115200

#define ADC_AREF 3.3f
#define BATVOLT_R1 4.7f
#define BATVOLT_R2 10.0f
#define BATVOLT_PIN BAT_VOLT

#ifndef enablePin
    #define enablePin -1
#endif

#define STARTUP_DELAY 10000
#define LONG_DELAY    1000000
//Medida del voltaje de la batería
static inline bool is_timedout(uint32_t from, uint32_t nr_ms) __attribute__((always_inline));
static inline bool is_timedout(uint32_t from, uint32_t nr_ms){return (millis() - from) > nr_ms;}
uint16_t getBatteryVoltage(){
    uint16_t voltage = (uint16_t)((ADC_AREF / 1.023) * (BATVOLT_R1 + BATVOLT_R2) / BATVOLT_R2 * (float)analogRead(BATVOLT_PIN));
    return voltage;
}
//Configuración NB-IoT
const char* apn = "lpwa.vodafone.iot";
const char* ip = "nbiot-csupm.ddns.net";
uint8_t     cid = 1;
const uint8_t band = 20;
const char* forceOperator = "";
//Variables
char    strBuffer[500];
```

```
char    dataDecode[100] = "";
const uint32_t timeout = 20;
bool signale = false;
int8_t cont;
int8_t    rssi_val;
uint8_t   ber_val;
uint16_t  volts;
int8_t    volts_val;
float lat_val;
float long_val;

Sodaq_nbIOT nbiot;

void sendMessageThroughUDP()
{   //Medida calidad de señal
    cont = cont + 1;
    signale = nbiot.getRSSIAndBER(&rssi_val,&ber_val);
    volts     = getBatteryVoltage();
    DEBUG_STREAM.println("captura señal extendida");
    DEBUG_STREAM.println("rssi:");
    DEBUG_STREAM.println(rssi_val);
    DEBUG_STREAM.println("Voltios:");
    DEBUG_STREAM.print(volts);
    DEBUG_STREAM.println(volts_val);

    DEBUG_STREAM.println("6. Sending message through UDP");

    int localPort = 37415;
    int socketID = nbiot.createSocket(localPort);

    if (socketID >= 7 || socketID < 0) {
        DEBUG_STREAM.println("Failed to create socket");
        return;
    }

    DEBUG_STREAM.println("7. Created socket!");
    snprintf(strBuffer,sizeof(strBuffer),"idMota:7 id:1 cont:%d volt:%d rssi:%d lat:%f long:%f",cont,volts,rssi_val, lat_val, long_val);
    size_t size = strlen(strBuffer);

    int lengthSent = nbiot.socketSend(socketID,ip, 37415, strBuffer);
    DEBUG_STREAM.println("8. Envía en el socket");
```

```
DEBUG_STREAM.print("9. String length vs sent: ");
DEBUG_STREAM.print(size);
DEBUG_STREAM.print(" vs ");
DEBUG_STREAM.println(lengthSent);

// DEMO.
if (nbiot.waitForUDPResponse()) {
    DEBUG_STREAM.println("10. Received response!");

    while (nbiot.hasPendingUDPBytes()) {
        DEBUG_STREAM.println("hasPendingUDPBytes");
        char data[200];
        SaraN2UDPPacketMetadata p;
        int size = nbiot.socketReceiveHex(data, 1, &p);

        if (size) {
            DEBUG_STREAM.println(data);
            char part = (char) (int)strtol(data, nullptr, 16);
            size_t len = strlen(dataDecode);
            snprintf(dataDecode + len, sizeof dataDecode - len, "%c", part);
            DEBUG_STREAM.println(part);
            DEBUG_STREAM.println(dataDecode);
            DEBUG_STREAM.println(p.socketID);
            DEBUG_STREAM.println(p.ip);
            DEBUG_STREAM.println(p.port);
            DEBUG_STREAM.println(p.length);
            DEBUG_STREAM.println(p.remainingLength);
        }
        else {
            DEBUG_STREAM.println("Receive failed!");
        }
    }
    if (strstr (dataDecode, "NOK"))
    {
        DEBUG_STREAM.println("Alerta recibida. Enter deep sleep 10 minutos");
        sodaq_wdt_safe_delay(LONG_DELAY);
    }
    snprintf(dataDecode, 1, "");
}
else {
```

```
        DEBUG_STREAM.println("Timed-out!");
    }
    //Fin de la DEMO.
    nbiot.closeSocket(socketID);
    DEBUG_STREAM.println("12. Cerramos Socket");
    DEBUG_STREAM.println();
}

void setup()
{   //Configuración comunicación NB-IoT
    sodaq_wdt_safe_delay(STARTUP_DELAY);
    cont = 0;
    lat_val = 0;
    long_val = 0;
    DEBUG_STREAM.begin(115200);
    DEBUG_STREAM.println("1. Initializing and connecting... ");

#ifdef R4XX
    MODEM_STREAM.begin(115200);
    DEBUG_STREAM.println("2. post getbaudrate");
    nbiot.setDiag(DEBUG_STREAM);
    nbiot.init(MODEM_STREAM, powerPin, enablePin, SARA_R4XX_TOGGLE, cid);
    sodaq_gps.init(gpsEnablePin);
    DEBUG_STREAM.println("3. Post init()");
#else
    MODEM_STREAM.begin(nbiot.getDefaultBaudrate());
    DEBUG_STREAM.println("2.b. No está definido R4XX");
    nbiot.setDiag(DEBUG_STREAM);
    nbiot.init(MODEM_STREAM, powerPin, enablePin, -1, cid);
#endif

if(SARA_R4XX_TOGGLE != -1){
    DEBUG_STREAM.println("4.a Sabemos que entra en _isSaraR4XX ");

}

#ifdef SARA_RESET
    DEBUG_STREAM.println("4.b. Entra al RESET");
    pinMode(SARA_RESET, OUTPUT);
    digitalWrite(SARA_RESET, HIGH);
#endif
```

```
#ifndef R4XX
    DEBUG_STREAM.println("4. Entra a Override (y a R4XX)");
    nbiot.overrideNconfigParam("CR_0354_0338_SCRAMBLING", true);
#endif

    if (!nbiot.connect("",ip,"", 20)) {
        DEBUG_STREAM.println("Failed to connect to the modem!, primer fallo");
    }
    DEBUG_STREAM.println("5. Mandamos UDP desde setup");
    sendMessageThroughUDP();
}

void loop()
{   //Medida GPS
    sodaq_wdt_safe_delay(STARTUP_DELAY);
    if(sodaq_gps.scan(false,40L * 1000)){
      lat_val = sodaq_gps.getLat();
      long_val = sodaq_gps.getLon();
      DEBUG_STREAM.println(lat_val);
      DEBUG_STREAM.println(long_val);
    }else{
      DEBUG_STREAM.println("No hay datos de GPS");
    }
    if (!nbiot.isConnected()) {
        if (!nbiot.connect("",ip, "", 20)) {
            DEBUG_STREAM.println("Failed to connect to the modem!, esperamos 1 minuto");
        }
    }
    else {
        DEBUG_STREAM.println("Mandamos UDP desde el loop");
        sendMessageThroughUDP();
    }
}
```

A.2 Ficheros de Libelium WASPMOTE

Los ficheros incluidos en este apartado contienen el código utilizado en la mota Libelium WASPMOTE.

Los ficheros son el demostrador de comunicación UDP, el demostrador de comunicación MQTT y el demostrador de comunicación bidireccional.

A.2.1 Fichero NB_IoT_udp_ThingSpeak.pde

Este es el fichero utilizado para transmitir datagramas UDP desde la mota SODAQ SARA AFF.

```
/*****************************************************************
 *  NB_IoT_udp_ThingSpeak.pde
 *  Descripción:
 *  Este script permite a la mota Waspmote enviar paquetes UDP al servidor "nbiot-csupm"
 *  Estos paquetes enviados incluyen los siguientes parámetros:
 *  ID de la mota; Tipo de mota; Nº de paquete; nivel se señal RSSI; Voltaje de la batería; Temperatura; Presión; Humedad
 *  Autor:      José María Gutiérrez García
 *  Versión:    1.0
 *  Fecha:      2022/03/15
 *****************************************************************/
//Librerias
#include <WaspBG96.h>
#include <WaspSensorEvent_v30.h>
#include <WaspFrame.h>
//Variables
uint8_t   error;
uint32_t  previous;
unsigned      contador_intentos = 0;
unsigned long epoch;
int   rssi_val;
float volts;
float temp;
float humd;
float pres;
//Variables String
char data[500];
char message[150];
char message_received[150];
char vlts_str[8];
```

```
char rssi_str[8];
char epoch_str[13];
char temp_str[8];
char humd_str[8];
char pres_str[8];
// Configuración APN
char apn[] = "lpwa.vodafone.iot";
char login[] = "";
char password[] = "";
// selección operador
char network_operator[] = "21401";
uint8_t operator_type = NUMERIC_OPERATOR;
//Banda
char band[] = B20;
// Configuración SERVER
char host[] = " nbiot-csupm.ddns.net";
uint16_t remote_port = 37415;
// Definición formato
char http_format[] =
  "GET /getpost_frame_parser.php?frame=%s HTTP/1.1\r\n"\
  "Host: test.libelium.com\r\n"\
  "Content-Length: 0\r\n\r\n";

// ConfiguraciónSocket ID
uint8_t connId = WaspBG96::CONNECTION_1;

void setup()
{// Configuración APN
  USB.ON();
  USB.println(F("Start program\n"));
  BG96.set_APN(apn, login, password);
  BG96.show_APN();
}

void loop()
{
  RTC.ON();
  RTC.getTime();
  RTC.OFF();
  // Configuración del buffer
  frame.setID("NB_IOT_TCP");
```

```
frame.createFrame(ASCII);
frame.addSensor(SENSOR_TIME, RTC.hour, RTC.minute, RTC.second);
frame.showFrame();
char frame_string[frame.length*2 + 1];
memset(frame_string, 0x00, sizeof(frame_string));
Utils.hex2str((uint8_t*)frame.buffer, (char*)frame_string, frame.length);
snprintf( data, sizeof(data), http_format, frame_string);
// Encendido módulo NB-IoT
error = BG96.ON();

if (error == 0)
{
  USB.println(F("2. NB-IoT module ready"));

  // Comprobación conexión NB-IoT
  error = BG96.nbiotConnection(apn, band, network_operator, operator_type);
  if (error == 0)
  {
    USB.println(F("2.1. NB-IoT connection: OK "));

    error = BG96.contextActivation(1,5);
    if (error == 0)
    {
      USB.println(F("2.1. NB-IoT context connection: OK "));
      USB.print(F("IP: ")); USB.println(BG96._ip);

      // Configuración servidor DNS
      error = BG96.setDNSServer("8.8.8.8","8.8.4.4");
      if (error == 0)
      {
        USB.println(F("3. Setting DNS server: OK "));
      }
      else
      {
        USB.println(F("3. DNS error: ")); USB.println(error,DEC);
        USB.println(BG96._buffer, BG96._length);
      }

      // Apertura socket UDP
      error = BG96.openSocketClient(connId, "UDP", host, remote_port);
```

```
        if (error == 0)
        {
          USB.println(F("4. Opening a socket... done!"));

          USB.print(F("IP BG96 address:"));
          USB.println(BG96._ip);
        }
        // Medidas
        // EPOCH
        epoch = RTC.getEpochTime();
        snprintf(epoch_str, 12, "%lu", epoch);
        // RSSI
        error = BG96.getRSSI();
        rssi_val = BG96._rssi;
        // Voltaje
        volts = PWR.getBatteryVolts();
        dtostrf(volts,1,2,vlts_str);
        // Temperatura
        temp = Events.getTemperature();
        USB.println(temp);
        dtostrf(temp,1,2,temp_str);
        USB.println(temp_str);
        // Presión
        pres = Events.getPressure();
        USB.println(pres);
        dtostrf(pres,1,2,pres_str);
        USB.println(pres_str);
        snprintf(pres_str, 8, pres_str);
        // Humedad
        humd = Events.getHumidity();
        USB.println(humd);
        dtostrf(humd,1,2,humd_str);
        USB.println(humd_str);
        // Paquete UDP
        snprintf(message, sizeof(message), "idMota:4 id:2 created_at:%s cont:%d volt:%s rssi:%d temp:%s humd:%s pres:%s", epoch_str,
contador_intentos, vlts_str, rssi_val,temp_str,humd_str,pres_str);
        contador_intentos++;
        USB.print("Mensaje: ");
        USB.println(message);
        error = BG96.send(connId, message);
        if (error == 0)
```

```
      {
        USB.println(F("4.1. Sending a frame... done!"));
      }
      else
      {
        USB.print(F("4.1. Error sending a frame. Code: "));
        USB.println(error, DEC);
      }

      // Recepción de paquetes
      USB.print(F("4.2. Waiting to receive data..."));

      error = BG96.receive(connId, 60000);

      // 4.3. Close socket
      ////////////////////////////////////////////
      error = BG96.closeSocketClient(connId);

      if (error == 0)
      {
        USB.println(F("4.3. Socket closed OK"));
      }
      else
      {
        USB.print(F("4.3. Error closing socket. Error code: "));
        USB.println(error, DEC);
      }
    }
  }
}
else
{
  // In case problem with the communication with the NB-IoT module
  USB.println(F("NB-IoT module not started"));
  USB.print(F("Error code: "));
  USB.println(error, DEC);
}


// 5. Switch OFF the NB-IoT module
////////////////////////////////////////////////
```

```
  error = BG96.OFF();
   if (error == 0)
    {
      USB.println(F("5. Module is power off"));
    }
    else
    {
      USB.println(F("5. Power off ERROR"));
    }



  // 6. Sleep
  ///////////////////////////////////////////////
  USB.println(F("6. Enter deep sleep..."));
  PWR.deepSleep("00:00:00:15", RTC_OFFSET, RTC_ALM1_MODE1, ALL_OFF);

  USB.ON();
  USB.println(F("7. Wake up!!\n\n"));

}
```

A.2.2 Fichero NB_IoT_csupm_mqtt_ThingSpeak.pde

```
/*****************************************************************
 *  NB_IoT_csupm_mqtt_ThingSpeak.pde
 *  Descripción:
 *  Este ejemplo permite a la mota Waspmote establecer una conexión MQTT con el servidor ThingSpeak
 *  En la conexión MQTT se establece una transmisión de los siguientes parámetros:
 *  ID de la mota; Tipo de mota; Nº de paquete; Nivel se señal RSSI; Voltaje de la batería; Temperatura; Presión; Humedad.
 *
 *  Autor:      José María Gutiérrez García
 *  Versión:    1.0
 *  Fecha:      2022/03/15
 *****************************************************************/
```

```
//Librerias
/////////////////////////////////////
#include <WaspSensorEvent_v30.h>
#include <WaspBG96.h>
//Variables
/////////////////////////////////////
float battery;
float volts;
int rssi_val;
float temp;
float humd;
float pres;
int error;
uint32_t previous;
int8_t info;
unsigned contador_intentos;
bool data_retention = false;
uint8_t qos;
uint16_t msgID;
//Variables String
/////////////////////////////////////
char bttr_str[8];
char vlts_str[8];
char rssi_str[8];
char temp_str[8];
char humd_str[8];
char pres_str[8];
char mqtt_csq[150];
char mqtt_payload[150];
// APN settings
char apn[] = "lpwa.vodafone.iot";
char login[] = "";
char password[] = "";
// Operator selection
/////////////////////////////////////
char network_operator[] = "21401";
/////////////////////////////////////
uint8_t operator_type = NUMERIC_OPERATOR;
//Band
/////////////////////////////////////
char band[] = B20;
```

```
// SERVER MQTT settings
//////////////////////////////////////

char mqtt_server[] = "nbiot-csupm.ddns.net";
uint16_t mqtt_port = 1883;
char mqtt_user[] = "";
char mqtt_pass[] = "";
char mqtt_clientID[] = "Hola1234";
char mqtt_topic[] = "g4/ts/channels/1671999/publish/KELIUY34ST6OGYEI";
//Without Auxiliar server
//char mqtt_server[] = "mqtt.thingspeak.com";
//char mqtt_topic[] = "channels/1671999/publish/KELIUY34ST6OGYEI";

data_retention = false;
qos = 0;
msgID = 1;
contador_intentos = 0;

void setup()
{
  USB.ON();
  USB.println(F("Setup.-  Start program"));


  // 1. sets operator parameters
  ////////////////////////////////////////////////
  BG96.set_APN(apn, login, password);



  // 2. Show APN settings via USB port
  ////////////////////////////////////////////////
  BG96.show_APN();

}

void loop()
{

  // 0.- Preparing message
  ////////////////////////////////////////////////
```

```
contador_intentos++;
USB.print("0.- Payload previo: ");
USB.println(mqtt_payload);


// 1. Switch ON
//////////////////////////////////////////////
error = BG96.ON();


if (error == 0)
{
  USB.println(F("1. NB-IoT module ready..."));


  // 2.1. Check connection to network and continue
  //////////////////////////////////////////////
  error = BG96.nbiotConnection(apn, band, network_operator, operator_type);
  if (error == 0)
  {
    USB.println(F("2.1. NB-IoT connection: OK "));


    // 2.2.1 NB-IoT Context Activation
    //////////////////////////////////////////////
    error = BG96.contextActivation(1,5);
    if (error == 0)
    {
      USB.println(F("2.2.1. NB-IoT context connection: OK "));
      USB.print(F("IP: ")); USB.println(BG96._ip);


      // 3. Setting DNS server
      //////////////////////////////////////////////
      error = BG96.setDNSServer("8.8.8.8","8.8.4.4");
      if (error == 0)
      {
        USB.println(F("3. Setting DNS server: OK "));
      }
      else
```

```
{
  USB.println(F("3. DNS error: ")); USB.println(error,DEC);
  USB.println(BG96._buffer, BG96._length);
}

//RSSI y medidas varias
////////////////////////////////////////////

info = BG96.getInfo(WaspBG96::INFO_CSQ);
USB.print(F("CSQ: "));
USB.println(BG96._buffer, BG96._length);

info = BG96.getInfo(WaspBG96::INFO_QCSQ);
USB.print(F("QCSQ: "));
USB.println(BG96._buffer, BG96._length);

info = BG96.getInfo(WaspBG96::INFO_QSPN);
USB.print(F("QSPN: "));
USB.println(BG96._buffer, BG96._length);

info = BG96.getInfo(WaspBG96::INFO_ICCID);
USB.print(F("ICCID: "));
USB.println(BG96._buffer, BG96._length);

info = BG96.getInfo(WaspBG96::INFO_IMSI);
USB.print(F("IMSI: "));
USB.println(BG96._buffer, BG96._length);
//VOLTS
////////////////////////////////////////////
volts = PWR.getBatteryVolts();
USB.print(F(" | Battery (Volts): "));
USB.print(volts, 2);
USB.print(F(" V"));
dtostrf(volts,1,2,vlts_str);
//RSSI
////////////////////////////////////////////
error = BG96.getRSSI();
USB.print(F(" RSSI (dB): "));
USB.print(BG96._rssi);
USB.println(F(" dB"));
rssi_val = BG96._rssi;
```

```
        dtostrf(BG96._rssi,1,0,rssi_str);
        //Temperature
        ////////////////////////////////////////////
        temp = Events.getTemperature();
        USB.println(temp);
        dtostrf(temp,1,2,temp_str);
        USB.println(temp_str);
        //Pressure
        ////////////////////////////////////////////
        pres = Events.getPressure();
        USB.println(pres);
        dtostrf(pres,1,2,pres_str);
        USB.println(pres_str);
        snprintf(pres_str, 8, pres_str);
        //Humidity
        ////////////////////////////////////////////
        humd = Events.getHumidity();
        USB.println(humd);
        dtostrf(humd,1,2,humd_str);
        USB.println(humd_str);
        //Payload
        ////////////////////////////////////////////
        snprintf(mqtt_payload, sizeof(mqtt_payload),
"field1=%u&field3=%s&field2=%s&field4=%s&field5=%s&field6=%s",contador_intentos,rssi_str,vlts_str,temp_str,humd_str,pres_str);
        USB.print("0.- Payload enviado: ");
        USB.println(mqtt_payload);


        // 4.1. MQTT open session
        ////////////////////////////////////////////
        error = BG96.mqttOpenSession(mqtt_server, mqtt_port, mqtt_clientID);

        // check answer
        if (error == 0)
        {
          USB.println(F("4.1. MQTT open session OK"));

          previous = millis();


        // 4.2. MQTT session status
```

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

error = BG96.mqttSessionStatus();

if (error == 0)
 {
   USB.print(F("4.2. MQTT session status: "));
   USB.println(BG96._mqtt_status, DEC);
 }
 else
 {
   USB.print(F("4.2. Error calling 'mqttSessionStatus' function. error: "));
   USB.println(error, DEC);
 }


 // 4.3. MQTT publish message
 //////////////////////////////////////////////

 error = BG96.mqttPublish(msgID, qos, data_retention, mqtt_topic, mqtt_payload);

 if (error == 0)
 {
   USB.println(F("4.3. MQTT publish message OK"));
   USB.print(F("Topic: ")); USB.println(mqtt_topic);
   USB.print(F("Payload: ")); USB.println(mqtt_payload);
   msgID++;
 }
 else
 {
   USB.print(F("4.3. Error calling 'mqttPublish' function. error: "));
   USB.println(error, DEC);
 }


 // 4.4. MQTT close session
 //////////////////////////////////////////////

 error = BG96.mqttCloseSession();

 if (error == 0)
```

```
        {
          USB.println(F("4.4. MQTT close session OK"));
        }
        else
        {
          USB.print(F("4.4. Error calling 'mqttCloseSession' function. error: "));
          USB.println(error, DEC);
        }
       }
       else
       {
         USB.print(F( "4.1. MQTT open session ERROR: "));
         USB.println(error, DEC);
       }

USB.print(F("IP: ")); USB.println(BG96._ip);
USB.println(F("short delay period"));
delay(5000);

      }

  }
  else
  {
    // Problem with the communication with the NB-IoT module
    USB.println(F("NB-IoT module not started"));
    USB.print(F("Error code: "));
    USB.println(error, DEC);
  }

  }


  // 5. Powers off the 4G module
  ////////////////////////////////////////////
  USB.println(F("5. Switch OFF NB-IoT module"));

  error = BG96.OFF();
   if (error == 0)
    {
      USB.println(F("5. Module is power off"));
```

```
    }
    else
    {
      USB.println(F("5. Power off ERROR"));
    }



  // 6. Sleep
  ///////////////////////////////////////////
  USB.println(F("6. Enter deep sleep..."));
  PWR.deepSleep("00:00:00:10", RTC_OFFSET, RTC_ALM1_MODE1, ALL_OFF);

  USB.ON();
  USB.println(F("7. Wake up!!\n\n"));

}
```

A.2.3 Fichero NB_IoT_udp_ThingSpeak.pde con DEMO

```
/*****************************************************************
 *  NB_IoT_udp_ThingSpeak.pde
 *  Descripción:
 *  Este script permite a la mota Waspmote enviar paquetes UDP al servidor "nbiot-csupm"
 *  Estos paquetes enviados incluyen los siguientes parámetros:
 *  ID de la mota; Tipo de mota; Nº de paquete; nivel se señal RSSI; Voltaje de la batería; Temperatura; Presión; Humedad
 *  El script contiene una DEMO de comunicación bidireccional de paquetes UDP entre la mota y el servidor UDP.
 *  La DEMO introduce una pausa en la ejecución del cógido de 10 minutos en caso de que el mensaje recibido contenga "NOK".
 *  Autor:      José María Gutiérrez García
 *  Versión:    1.0
 *  Fecha:      2022/03/15
 *****************************************************************/
//Librerias
#include <WaspBG96.h>
#include <WaspSensorEvent_v30.h>
#include <WaspFrame.h>
//Variables
uint8_t   error;
uint32_t  previous;
```

```
unsigned      contador_intentos = 0;
unsigned long epoch;
int   rssi_val;
float volts;
float temp;
float humd;
float pres;
//Variables String
char data[500];
char message[150];
char message_received[150];
char vlts_str[8];
char rssi_str[8];
char epoch_str[13];
char temp_str[8];
char humd_str[8];
char pres_str[8];
// Configuración APN
char apn[] = "lpwa.vodafone.iot";
char login[] = "";
char password[] = "";
// selección operador
char network_operator[] = "21401";
uint8_t operator_type = NUMERIC_OPERATOR;
//Banda
char band[] = B20;
// Configuración SERVER
char host[] = " nbiot-csupm ";
uint16_t remote_port = 37415;
// Definición formato
char http_format[] =
  "GET /getpost_frame_parser.php?frame=%s HTTP/1.1\r\n"\
  "Host: test.libelium.com\r\n"\
  "Content-Length: 0\r\n\r\n";

// ConfiguraciónSocket ID
uint8_t connId = WaspBG96::CONNECTION_1;

void setup()
{// Configuración APN
  USB.ON();
```

```
  USB.println(F("Start program\n"));
  BG96.set_APN(apn, login, password);
  BG96.show_APN();
}

void loop()
{
  RTC.ON();
  RTC.getTime();
  RTC.OFF();
  // Configuración del buffer
  frame.setID("NB_IOT_TCP");
  frame.createFrame(ASCII);
  frame.addSensor(SENSOR_TIME, RTC.hour, RTC.minute, RTC.second);
  frame.showFrame();
  char frame_string[frame.length*2 + 1];
  memset(frame_string, 0x00, sizeof(frame_string));
  Utils.hex2str((uint8_t*)frame.buffer, (char*)frame_string, frame.length);
  snprintf( data, sizeof(data), http_format, frame_string);
  // Encendido módulo NB-IoT
  error = BG96.ON();

  if (error == 0)
  {
    USB.println(F("2. NB-IoT module ready"));

    // Comprobación conexión NB-IoT
    error = BG96.nbiotConnection(apn, band, network_operator, operator_type);
    if (error == 0)
    {
      USB.println(F("2.1. NB-IoT connection: OK "));

      error = BG96.contextActivation(1,5);
      if (error == 0)
      {
        USB.println(F("2.1. NB-IoT context connection: OK "));
        USB.print(F("IP: ")); USB.println(BG96._ip);

        // Configuración servidor DNS
        error = BG96.setDNSServer("8.8.8.8","8.8.4.4");
        if (error == 0)
```

```
{
  USB.println(F("3. Setting DNS server: OK "));
}
else
{
  USB.println(F("3. DNS error: ")); USB.println(error,DEC);
  USB.println(BG96._buffer, BG96._length);
}

// Apertura socket UDP
error = BG96.openSocketClient(connId, "UDP", host, remote_port);

if (error == 0)
{
  USB.println(F("4. Opening a socket... done!"));

  USB.print(F("IP BG96 address:"));
  USB.println(BG96._ip);
}
// Medidas
// EPOCH
epoch = RTC.getEpochTime();
snprintf(epoch_str, 12, "%lu", epoch);
// RSSI
error = BG96.getRSSI();
rssi_val = BG96._rssi;
// Voltaje
volts = PWR.getBatteryVolts();
dtostrf(volts,1,2,vlts_str);
// Temperatura
temp = Events.getTemperature();
USB.println(temp);
dtostrf(temp,1,2,temp_str);
USB.println(temp_str);
// Presión
pres = Events.getPressure();
USB.println(pres);
dtostrf(pres,1,2,pres_str);
USB.println(pres_str);
snprintf(pres_str, 8, pres_str);
// Humedad
```

```
        humd = Events.getHumidity();
        USB.println(humd);
        dtostrf(humd,1,2,humd_str);
        USB.println(humd_str);
        // Paquete UDP
        snprintf(message, sizeof(message), "idMota:4 id:2 created_at:%s cont:%d volt:%s rssi:%d temp:%s humd:%s pres:%s", epoch_str,
contador_intentos, vlts_str, rssi_val,temp_str,humd_str,pres_str);
        contador_intentos++;
        USB.print("Mensaje: ");
        USB.println(message);
        error = BG96.send(connId, message);
        if (error == 0)
        {
          USB.println(F("4.1. Sending a frame... done!"));
        }
        else
        {
          USB.print(F("4.1. Error sending a frame. Code: "));
          USB.println(error, DEC);
        }

        // Recepción de paquetes
        USB.print(F("4.2. Waiting to receive data..."));

        error = BG96.receive(connId, 60000);
        //DEMO. Lectura de paquete UDP enviado por el servidor. Si "NOK", ir a estado "Deep Sleep".
        ////////////////////////////////////////////
        if (error == 0)
        {
          if (BG96.socketInfo[connId].received > 0)
          {
            USB.println(F("\n-----------------------------------"));
            USB.print(F("Data received:"));
            USB.println(BG96._buffer, BG96._length);
            snprintf(message_received,sizeof(message_received)/*BG96._length*/,(const char*) BG96._buffer);
            if (strstr (message_received, "NOK"))
            {
              USB.println("Alerta recibida. Enter deep sleep 10 minutos");
              PWR.deepSleep("00:00:10:00", RTC_OFFSET, RTC_ALM1_MODE1, ALL_OFF);
              USB.ON();
              USB.println(F("7. Wake up!!\n\n"));
```

```
            }
            USB.println(F("-----------------------------------"));
          }
          else
          {
            USB.println(F("NO data received"));
          }
        }
        else
        {
          USB.println(F("No data received."));
          USB.println(error, DEC);
        }
        //END DEMO. Read UDP packet from UDP server. If "NOK", go to sleep.
        ////////////////////////////////////////////


        // 4.3. Close socket
        ////////////////////////////////////////////
        error = BG96.closeSocketClient(connId);

        if (error == 0)
        {
          USB.println(F("4.3. Socket closed OK"));
        }
        else
        {
          USB.print(F("4.3. Error closing socket. Error code: "));
          USB.println(error, DEC);
        }
      }
    }
  }
  else
  {
    // In case problem with the communication with the NB-IoT module
    USB.println(F("NB-IoT module not started"));
    USB.print(F("Error code: "));
    USB.println(error, DEC);
  }
```

```
  // 5. Switch OFF the NB-IoT module
  ////////////////////////////////////////////////
  error = BG96.OFF();
   if (error == 0)
    {
      USB.println(F("5. Module is power off"));
    }
    else
    {
      USB.println(F("5. Power off ERROR"));
    }



  // 6. Sleep
  ////////////////////////////////////////////////
  USB.println(F("6. Enter deep sleep..."));
  PWR.deepSleep("00:00:00:15", RTC_OFFSET, RTC_ALM1_MODE1, ALL_OFF);

  USB.ON();
  USB.println(F("7. Wake up!!\n\n"));

}
```

Anexo B Código de los servidores

En este anexo se muestran las distintas versiones de los servidores. Los servidores codificados son un servidor de comunicación UDP, un servidor de comunicación MQTT y un servidor de comunicación mixta UDP y MQTT.

B.1 Servidor UDP

Este es el servidor utilizado para recibir los datagramas UDP de las motas.

```
#############################################################
#   ServidorUDP.py
#   Descripción:
#   Almacena los datagramas recibidos en el servidor y responde con el mensaje "Temp:OK"
#   Autor:      José María Gutiérrez García
#   Versión:    1.0
#   Fecha:      2022/01/30
#############################################################

import socket
import csv

### Variables de entorno ###
# Dirección IP local del servidor (la misma del ordenador)
localIP =   "192.168.0.15"

# Dirección puerto entrada donde escucha el servidor 37415
localPort = 37415

# Tamaño del mensaje
bufferSize = 1024

# Información a enviar a la mota. Se modifica si la temperatura es superior a la recomendada
msgFromServer = "Temp:OK"
# Información codificada
bytesToSend = str.encode(msgFromServer)

# Se crea datagrama socket
UDPServerSocket = socket.socket(family=socket.AF_INET, type=socket.SOCK_DGRAM)
```

```
# Bind to address and ip
UDPServerSocket.bind((localIP, localPort))

#DEBUG solo si se indica mediante comandos
print("UDP server up and listening")

########
# Listen for incoming datagrams #Escuchar datagramas solo si se indica mediante comandos
while(True):

    bytesAddressPair = UDPServerSocket.recvfrom(bufferSize)

    message = bytesAddressPair[0]

    address = bytesAddressPair[1]

    clientMsg = f"Message from Client:{message}"
    clientIP = f"Client IP Address:{address}"

    #DEBUG solo si se indica mediante comandos
    print(clientMsg)
    print(clientIP)
    print(message)

    # write csv
    # string_original = "0.123123 1 2 3 4 5"
    #Evitar bloqueo de archivo
    with open('C:/Users/Fisherman/desktop/Servidor/Ficheros/test1503.csv', 'a', newline='') as csvfile:
        #Creamos archivo .csv
        writer = csv.writer(csvfile)
```

```
        #Write line con info separada por ' ' #utf-8 coge el byte en ese formato
        writer.writerow(message.decode("utf-8").split(' '))
    # Enviamos respuesta a la mota
    UDPServerSocket.sendto(bytesToSend, address)
```

B.2 Servidor MQTT

Este es el servidor utilizado para establecer una comunicación MQTT desde nuestro servidor y el servidor de red ThingSpeak . Este servidor se codifica para comprobar la comunicación de la plataforma ThingSpeak.

```
############################################################
#   ThingSpeak.py
#   Descripción:
#   Test de ThingSpeak
#   Autor:      José María Gutiérrez García
#   Versión:    1.0
#   Fecha:      2022/01/30
############################################################
import thingspeak

# Inicializo el canal
ch = thingspeak.Channel(id=1606136,api_key="3OW0CCAOHYTEA3BV")

ch.update({
    "field1": 0
})
```

B.3 Servidor de comunicación mixta UDP y MQTT

A continuación, se muestra el servidor de comunicación mixta UDP y MQTT.

Este servidor Almacena los datagramas UDP recibidos en el servidor, procesa la información recibida, comprueba si hay errores de coincidencia en los parámetros recibidos, analiza que tipo de mota ha enviado el datagrama y envía la información al servidor ThingSpeak correspondiente a cada mota mediante una comunicación MQTT.

```
############################################################
#   ServidorUDPOptions.py
#   Descripción:
#   Almacena los datagramas UDP recibidos en el servidor, procesa la información recibida, comprueba si hay errores de
coincidencia en los parámetros recibidos,
#   analiza que tipo de mota ha enviado el datagrama y envía la información al servidor ThingSpeak correspondiente a cada
mota mediante una comunicación MQTT.
#   El servidor puede iniciarse en los siguientes modos:
#   help    Mediante el comando -h, muestra las opciones de ejecución.
#   DEBUG   Mediante el comando -d, ejecuta el programa en modo DEBUG.
#   LISTEN  Mediante el comando -l, procesa los datagramas UDP recibidos.
#   THINGSPEAK    Mediante el comando -ts, reenvía la información a ThingSpeak
#   Se arranca con: python.\servidorUDPOptions.py -d f -l t -ts t
#   Autor:      José María Gutiérrez García
#   Versión:    1.0
#   Fecha:      2022/03/20
############################################################

from ast import Pass
import socket
import csv
import thingspeak
import logging
from logging.handlers import RotatingFileHandler
# Python program to demonstrate
# command line arguments
```

```
import argparse

# Initialize parser
parser = argparse.ArgumentParser()

def str2bool(v):
    if isinstance(v, bool):
        return v
    if v.lower() in ('yes', 'true', 't', 'y', '1'):
        return True
    elif v.lower() in ('no', 'false', 'f', 'n', '0'):
        return False
    else:
        raise argparse.ArgumentTypeError('Boolean value expected.')

# Adding optional argument
parser.add_argument("-d", "--Debug", default=False, type=str2bool, help = "In case of debug")
parser.add_argument("-l", "--Listen", default=True, type=str2bool, help = "Check if you want to listen UDP requests or
not")
parser.add_argument("-ts", "--ThingSpeak", default=True, type=str2bool, help = "Send info to ThingSpeak server")
args = parser.parse_args()

### Logging
if args.Debug:
    log_level = logging.DEBUG
else:
    log_level = logging.INFO
```

```
#Formato DEBUG
logger = logging.getLogger('Servidor_Mota')
logger.setLevel(log_level)
formatter = logging.Formatter('%(asctime)s - %(name)s - %(levelname)s - %(message)s')

### Configurar logging al fichero
fileHandler = RotatingFileHandler('server.log', maxBytes=2000, backupCount=5)
fileHandler.setFormatter(formatter)

### Configurar logging por pantalla
consoleHandler = logging.StreamHandler()
consoleHandler.setFormatter(formatter)

logger.addHandler(fileHandler)
logger.addHandler(consoleHandler)

### Variables de entorno ###
# Dirección IP local del servidor (la misma del ordenador)
localIP = "192.168.0.15"

# Dirección puerto entrada donde escucha el servidor 37415
localPort = 37415

# Tamaño del mensaje
bufferSize = 1024

# Temperatura límite de funcionamiento recomendado
tempLimite = 30
señalLimite = -80
# Información a enviar a la mota
```

```
msgFromServer = "Datos enviados a ThingSpeak"
# Información codificada
bytesToSend = str.encode(msgFromServer)

# Información de error de respuesta a la mota
errorUploadMsgFromServer = "ID required, data not uploaded"
bytesToSendErrorUpload = str.encode(errorUploadMsgFromServer)

# Información de error de datos a la mota
errorDataMsgFromServer = "An error occured, ID or Field exception"
bytesToSendErrorData = str.encode(errorDataMsgFromServer)

## Channel data
# Datos de la apikey en función del canal
api_key = {
    "1":{ "id":"1606136", "api_key": "3OW0CCAOHYTEA3BV"},
    "2": { "id": "1671999", "api_key": "KELIUY34ST6OGYEI"}
}

type_data = {
    "cont": "field1",
    "volt": "field2",
    "rssi": "field3",
    "temp": "field4",
    "humd": "field5",
    "pres": "field6",
    "lat":"lat",
    "long":"long"
    }
```

```
### Código del servidor
if __name__ == '__main__':
    # Inicializo el canal
    ch = thingspeak.Channel(id=id,api_key=api_key)

    # Abres un socket
    UDPServerSocket = socket.socket(family=socket.AF_INET, type=socket.SOCK_DGRAM)

    # Estableces la ip y puerto del socet
    UDPServerSocket.bind((localIP, localPort))

    #DEBUG solo si se indica mediante comandos
    logger.info("UDP server up and listening")


    # Esperando a recibir peticiones
    while(args.Listen):

        # Escribe la información recibida en una variable
        bytesAddressPair = UDPServerSocket.recvfrom(bufferSize)

        # Primera parte de la informacion es el mensaje recibido y la segunda la dirección IP de la mota
        message = bytesAddressPair[0]
        address = bytesAddressPair[1]

        clientMsg = f"Message from Client:{message}"
        clientIP = f"Client IP Address:{address}"

        #DEBUG solo si se indica mediante comandos
        logger.debug(clientMsg)
```

```
        logger.debug(clientIP)
        logger.debug(message)

        ### Escribir la información en un csv

        ## Decodificamos la información a utf-8 y generamos un array separado por ' '
        array_info = message.decode("utf-8").split(' ')

        # Abrimos el fichero para escribir y se usa with open para evitar que el fichero se quede abierto si se pierde el
servicio
        with open('C:/Users/Fisherman/desktop/Servidor/Ficheros/test1503.csv', 'a', newline='') as csvfile:
            # Inicialia la escritura del archivo .csv y lo crea si no existe
            writer = csv.writer(csvfile)
            # Escribe la información en la siguiente linea del archivo
            writer.writerow(array_info)

        # Inicializamos el diccionario donde se va a guardar la información
        data = dict()

        # Inicializamos la variable ID del canal
        channel_id = None

        # Recorremos el array de información donde vamos a extraer los fieldX e identificar el canal ID
        for i in array_info:
            # Separamos el string por : para sacar la clave (primera parte 0) y el valor (segunda parte 1)
            info_split = i.split(':')
            # Sacar la clave
            key = info_split[0]
            # Sacar el valor
            value = info_split[1]
```

```
            # En el caso de que la clave sea "id" implica que se le está pasando el id del canal al que enviar la info
            # Usamos esta variable para enviar la información a ThingSpeak
            if key == "id":
                logger.debug(key)
                channel_id = value
                logger.debug(id)
            # Si la clave no es "id" será el fieldX que se guardará en el diccionario
            elif key == "idMota":
                # Pendiente de tratar la información en función de la mota que lo envía
                pass
            else:
                try:
                    data[type_data[key]] = value
                except Exception as e:
                    logging.error(f"No está considerado ese valor {i} dentro de los conocidos, ha de ser uno de los
siguientes:")
                    logging.error(f"cont|volt|rssi|temp|humd|pres")
        logger.debug(data)

        ###DEMO --
        # Se compara la potencia de señal recibida con el límite recomendado
        #if api_key == 1:
         #   if float(data[type_data["rssi"]]) < señalLimite:
          #      logger.debug(f'Señal baja: {data[type_data["rssi"]]}')
           #     msgFromServer = "Datos enviados a ThingSpeak. Señal NOK->Enter deep sleep"
            #else:
             #   msgFromServer = "Datos enviados a ThingSpeak"
        # Se compara la temperatura recibida por el sensor con el límite recomendado
        #elif api_key == 2:
         #   if float(data[type_data["temp"]]) > tempLimite:
```

```
          #      logger.debug(f'Temp alta: {data[type_data["temp"]]}')
           #     msgFromServer = "Datos enviados a ThingSpeak. Temperatura NOK->Enter deep sleep"
            #else:
             #   msgFromServer = "Datos enviados a ThingSpeak"
        #else:
         #   logger.error(errorUploadMsgFromServer)
            # Para enviar a la mota el error
          #  UDPServerSocket.sendto(bytesToSendErrorUpload, address)
        ###FIN de la DEMO --

        # Información codificada
        bytesToSend = str.encode(msgFromServer)

        # Comprueba si quiere enviar la información al ThingSpeak por el parametro ejecutado, en caso de no serlo vuelvo
al inicio del While
        if not args.ThingSpeak:
            logger.debug("Ignoring ThingSpeak's step")
            continue

        # Si existe id del canal se procede a enviar la información a la API de ThingSpeak
        if channel_id:
            logger.info("Connecting to ThingSpeak...")
            # Se usa este try-catch por si los datos o el id son incorrectos y que no pierda servicio el servidor
            try:
                # Inicializamos el canal en función de la id y usamos la api_key del diccionario declarado anteriormente
                ch = thingspeak.Channel(id=api_key[channel_id]["id"],api_key=api_key[channel_id]["api_key"])
                logger.debug(f'id: {api_key[channel_id]["id"]}, api_key: {api_key[channel_id]["api_key"]}')
                # Utilizamos este metodo para enviar la información a ThingSpeak
                ch.update(data)
                # Enviamos una respuesta a la mota
```

```
            UDPServerSocket.sendto(bytesToSend, address)
        # Para capturar la excepción e informar a la mota
        except Exception as e:
            logger.error(errorDataMsgFromServer)
            # Para enviar a la mota el error
            UDPServerSocket.sendto(bytesToSendErrorData, address)
# En el caso de que no se le ha pasado el id del canal se informará a la mota
else:
    logger.error(errorUploadMsgFromServer)
    # Para enviar a la mota el error
    UDPServerSocket.sendto(bytesToSendErrorUpload, address)
```