Escuela Politécnica Superior de Linares

Grado en Ingeniería de Tecnologías de Telecomunicación

UNIVERSIDAD DE JAÉN
Escuela Politécnica Superior de Linares

Trabajo Fin de Grado

MONITORIZACIÓN DE PLAZAS DE PARKING EXTERIOR MEDIANTE SENSORES WASPMOTE Y TECNOLOGÍA LPWAN SIGFOX

Alumno: **Plácido Humberto Molina Rivillas**

Tutor: José Ángel Fernández Prieto
Depto.: Departamento de Ingeniería de Telecomunicación

Octubre, 2018

Índice General

1. RESUMEN

A día de hoy, un problema real al que no se le ha prestado la suficiente atención hasta ahora, es encontrar aparcamiento en ciudades y pueblos. Resulta ser una tarea, a veces, imposible de realizar debido al gran estancamiento producido por los numerosos vehículos que nos encontramos en la calzada. Este problema es más grave de lo que se piensa, ya que no solo se pierde tiempo, sino que aumenta el nivel de estrés en las personas, el gasto económico y la contaminación ambiental y acústica.

Este problema puede ser resuelto, ampliando el aparcamiento en las zonas de mayor tránsito, utilizando más el transporte público o ,en nuestro caso, instalando unos sensores en las plazas de aparcamiento que hay en las vías públicas para saber donde hay disponible aparcamiento y hacer esta tarea más sencilla.

Para ello, se utilizarán sensores de aparcamiento que enviarán mensajes a nuestro servidor informando del estado de la plaza, si está libre u ocupada. El trayecto del mensaje se refleja en la figura 1.1.

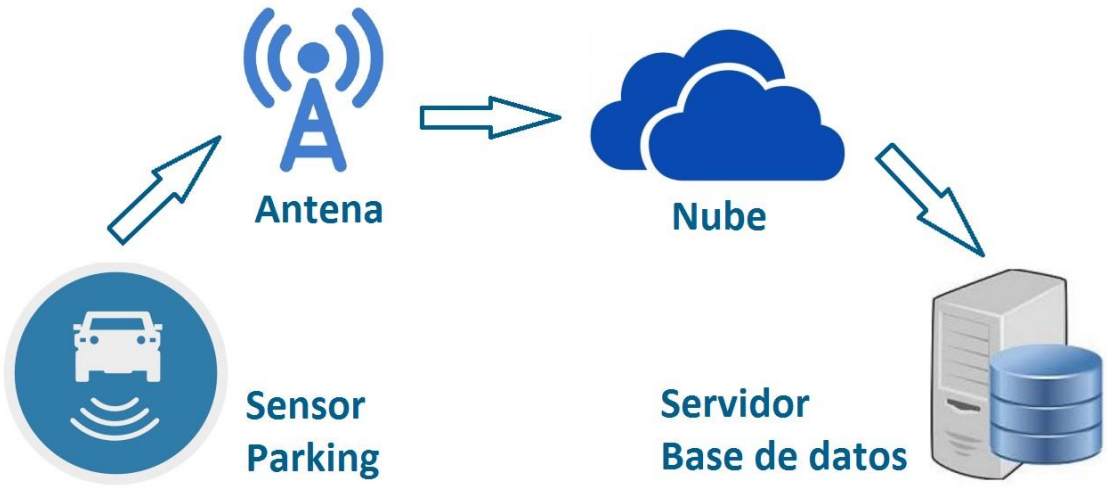


Figura 1.1: Envío del estado de la plaza de aparcamiento.

2. INTRODUCCIÓN

Como se ha mencionado anteriormente, encontrar aparcamiento en la vía pública a ciertas horas, como horas laborales, puede ser hasta perjudicial para la salud, llevando a cabo esta operación se puede tardar 30 minutos y en ciudades más grandes, como Barcelona, Madrid, Alicante, etc., incluso el doble de tiempo. A este problema de pérdida de tiempo y salud, hay que añadir la contaminación por el combustible de los coches, un coche diesel contamina más que los demás y en la actualidad hay más coches diesel que gasolina, híbridos y eléctricos juntos, por lo tanto, la contaminación es considerable si a la trayectoria de las personas en coche le añadimos entre 30 y 60 minutos más a diario.

Cada vez más ingenieros están buscando solución a este problema, y nuestra solución puede ser muy útil a la hora de encontrar aparcamiento en las grandes ciudades dentro de un horario comprometido. Este TFG se implementará de la siguiente forma:

- Se añadirá el equipo integrándolo en la calzada para que detecte a los vehículos desde abajo. Se cubrirá el equipo con un material transparente hasta no dejar imperfecciones de hueco en la vía.

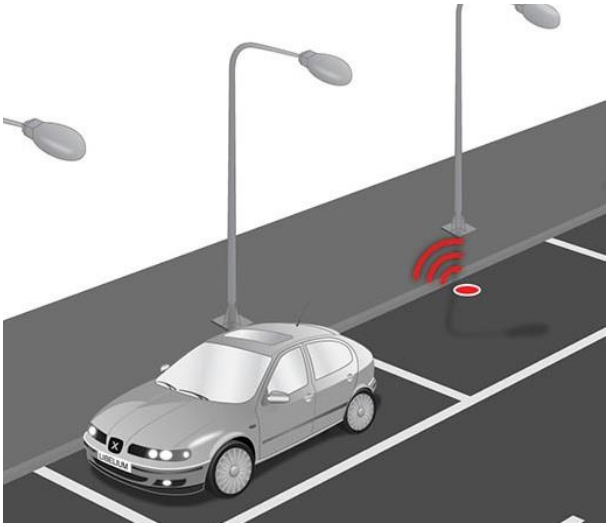


Figura 2.2 Sensor detectando plaza ocupada.

Fuente:http://www.libelium.com/

- El equipo constaría de una placa Waspmote pro 1.2, un sensor de parking (Smart Parking 1.2), un módulo de Sigfox para poder comunicarse y su respectiva antena de 4.5 dBi que es la que disponemos en nuestro caso.

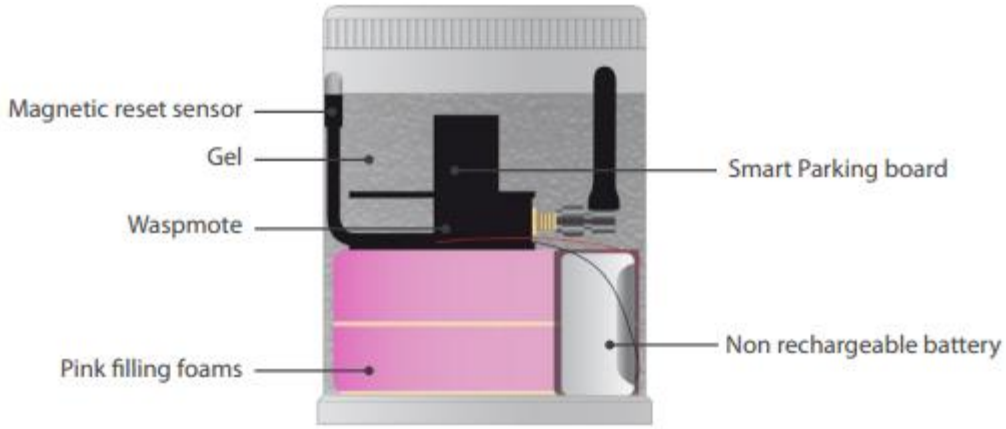


Figura 2.3 Recipiente preparado con el equipo completo.

Fuente: http://www.libelium.com/

- El equipo mandará la información de su ID y el estado de la plaza cada 5 minutos en horarios comprometidos (laborales o haciendo un estudio previo antes para ver cuando habría menos plazas disponibles), en nuestro caso se escogerá el horario de mañana desde las 8:00 a las 16:00 horas, cada 15 minutos en horarios flexibles (de 16:00 a 22:00 horas), y cada 30 minutos en horario de noche (de 22:00 a 8:00 horas), ya que no se pueden enviar al backend de sigfox más de 140 mensajes al día, y se priorizará las horas por congestión de tráfico. Este apartado se expondrá más adelante en la sección líneas de futuro como ampliación del TFG.
- Como se ha comentado, se mandarán los datos a través del módulo de Sigfox hasta el backend, donde se recibirán los datos, a través de una petición GET, se envían a una base de datos ya implementada en otro proyecto para poder acceder a ella desde el navegador viendo a tiempo real si las plazas están libres u ocupadas.

En este proyecto se implementará un grupo de sensores Libelium con módulo Sigfox a una frecuencia de 868 Mhz para la transmisión de datos que recibirá el backend y utilizando una base de datos para saber a tiempo real donde hay plazas de aparcamiento libres. Este modo de comunicación se implementa para la versión del sensor de parking 1.2 de Libelium con el Waspmote Pro 1.2. Se detalla a continuación el modo de comunicación de Sigfox.

2.1 Comunicación Sigfox

La empresa Libelium ha añadido un nuevo módulo de transmisión inalámbrico a los ya actuales, se trata del módulo Sigfox, donde se pueden usar dispositivos con rangos de transferencia grandes requiriendo poca energía. Esta tecnología fue inventada para el desarrollo de ciudades inteligentes y el Internet de las Cosas (IoT).

Actualmente la red de Sigfox cubre alrededor de 60 países, entre ellos España, Francia, Reino Unido, etc. En grandes ciudades, como EEUU, hay grandes proyectos realizados con esta tecnología en cada rincón de la ciudad, como la temperatura ambiente, ruido acústico de cada calle, la contaminación urbana o la recogida de las cosechas.

Figura 2.4 Esquema de Red Sigfox - Libelium

Fuente: http://www.libelium.com/

Como los datos se almacenan directamente en los servidores de Sigfox, se reduce el coste para las implementaciones de "IoT" y las "Smart Cities". Los dispositivos pueden dirigir los datos a una base de datos privada o a la nube.

2.2 Módulo Sigfox

Los módulos envían la información directamente a la puerta de enlace de Sigfox dando una respuesta rápida y una configuración de red inmediata. La red trabaja de manera similar a las redes de los operadores móviles sin necesidad de tarjeta SIM.

Además del modo Sigfox, los módulos pueden usar dos configuraciones más, el Modo P2P y el Modo Híbrido, aunque para nuestro proyecto no serán necesarios.

Figura 2.5Módulo de transmisión Sigfox con antena de 4.5 dBi

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

2.3 Backend de Sigfox

En toda comunicación hay dos extremos, el emisor y el receptor. El emisor serían los dispositivos Libelium y el receptor, en este caso, sería el backend de Sigfox y posteriormente la base de datos que recibe y procesa la información recibida.

El back-end de Sigfox proporciona una interfaz a la aplicación web para administrar los dispositivos, visualizar los mensajes transmitidos y la configuración de los datos obtenidos por los sensores.

Más información en el apartado 4.5 Backend Sigfox.

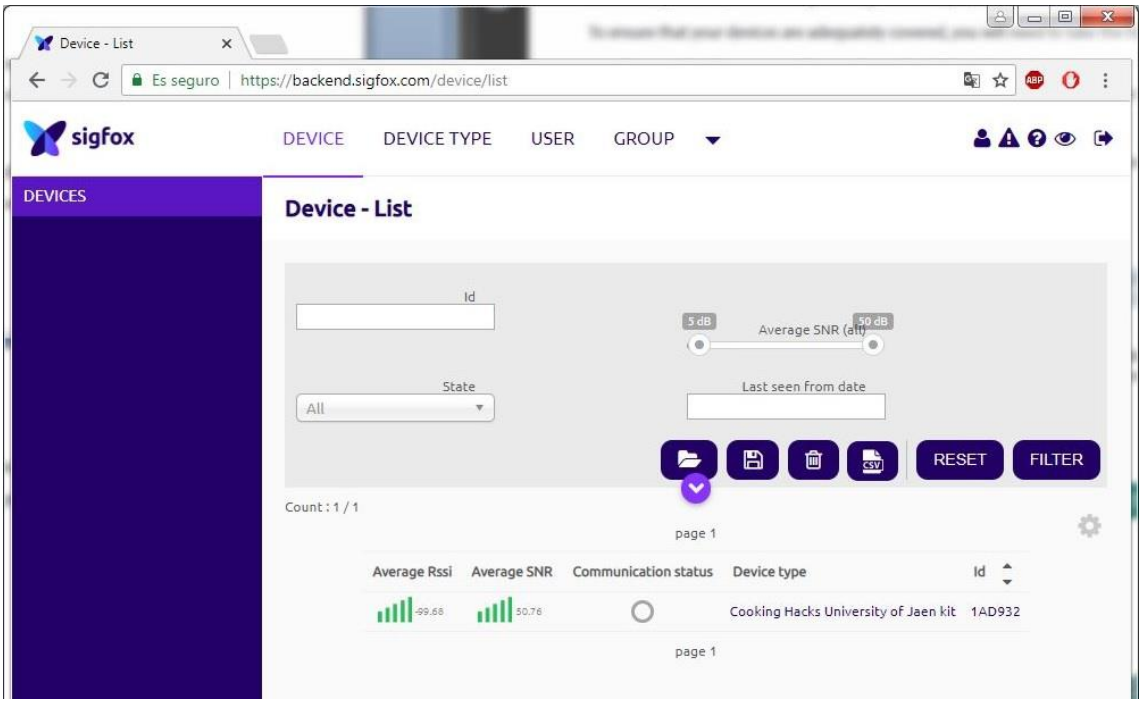


Figura 2.6: Backend de Sigfox

Fuente: https://backend.sigfox.com/device/list

3. OBJETIVOS

El objetivo principal de este proyecto es implementar una plataforma de parking en la vía pública a través de sensores Libelium inalámbricos.

Este objetivo se puede dividir en los siguientes sub-objetivos:

- Se programarán las placas Waspmote con sus sensores correspondientes de parking realizando las medidas cada cierto tiempo, enviando los datos al Backend de Sigfox. Tal programación se realizará con el programa "Waspmote Pro Ide v04" para windows facilitado por la marca del producto Libelium. Programación probada para nuestra versión Waspmote 1.2 Pro.
  Fuente del programa: http://downloads.libelium.com/waspmote-pro-ide-v04-windows.zip.
- Se estudiará el modo de comunicación Sigfox con su Backend donde se recibirán los datos de los sensores.
- Se enviarán los datos al Backend de Sigfox, comprobando que llega el mensaje correctamente.
- Se reenviarán los datos del Backend a una base de datos a través de una petición GET.
- Se accederá desde el navegador al servidor web para comprobar las plazas que hay disponibles.

4. MATERIALES Y METODOLOGÍA

4.1Waspmote versión 1.2 Pro

En este apartado se detallan las características de la placa Waspmote versión 1.2 Pro que se muestra en la figura 4.7, así como la configuración deseada para nuestro proyecto, la instalación tanto física como por software a manos de su programa Waspmote Pro Ide v04 para Windows (en nuestro caso), las librerías necesarias para su óptima programación y la integración del código de Sigfox para poder transmitir.

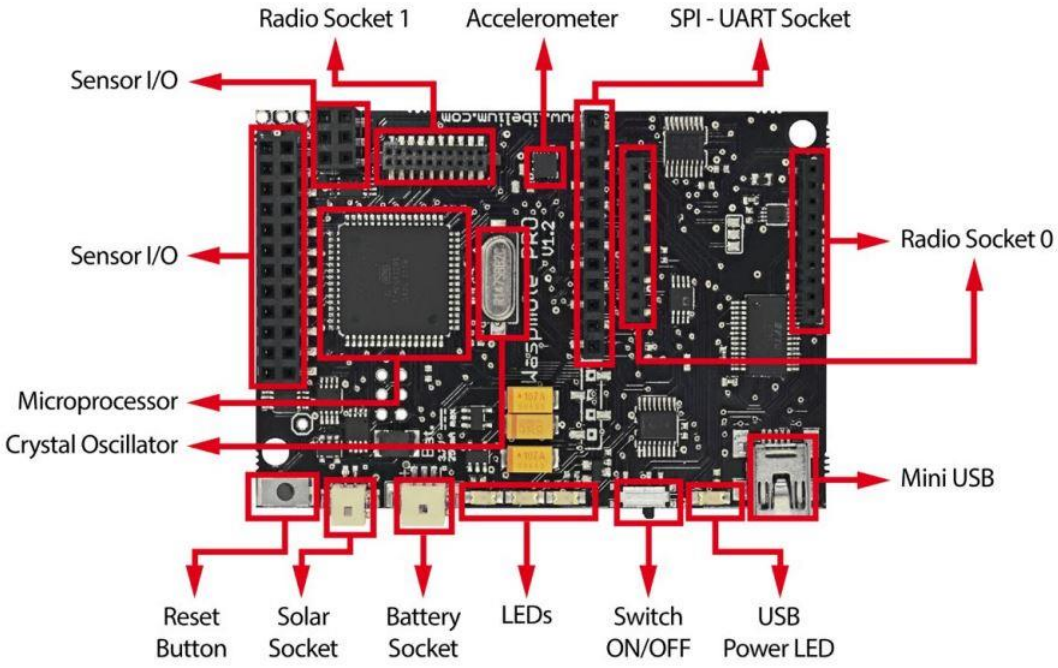


Figura 4.7: Waspmote versión 1.2 Pro

Fuente: http://www.libelium.com

4.1.1 Características

Los sensores Waspmote están pensados para trabajar consumiendo una cantidad de energía baja, junto a Sigfox, se consigue que su vida útil sea más larga.

Especificaciones:

- Microcontrolador: ATmega1281
- Frecuencia: 8Mhz
- SRAM: 8KB
- EEPROM: 4KB
- FLASH: 128KB
- Tarjeta SD: Hasta 2GB
- Peso: 20gr

- Dimensiones: 73.5 x 51 x 13mm
- Rango de Temperatura: [-20ºC, +65ºC]

Valores de funcionamiento:

- Tensión de batería mínima de funcionamiento 3.3V
- Tensión de batería máxima de funcionamiento 4.2V
- Tensión de carga USB 5V
- Tensión de carga placa solar entre 6 y 12V
- Corriente de carga de batería por USB 100mA (max)
- Corriente de carga de batería por placa solar 280mA (max)

Valores máximos absolutos:

- Tensión en cualquier pin [-0.5V, +3.8V]
- Corriente máxima por cualquier pin I/O digital 40mA
- Tensión de alimentación USB 7V
- Tensión de alimentación placa solar 18V
- Tensión de batería cargada 4.2V

4.1.2 Conexión con Sigfox

Para poder enviar nuestra información al Backend de Sigfox, se necesitará colocar en la placa un módulo de transmisión. Para ello, se introduce en los socket 0 o 1.

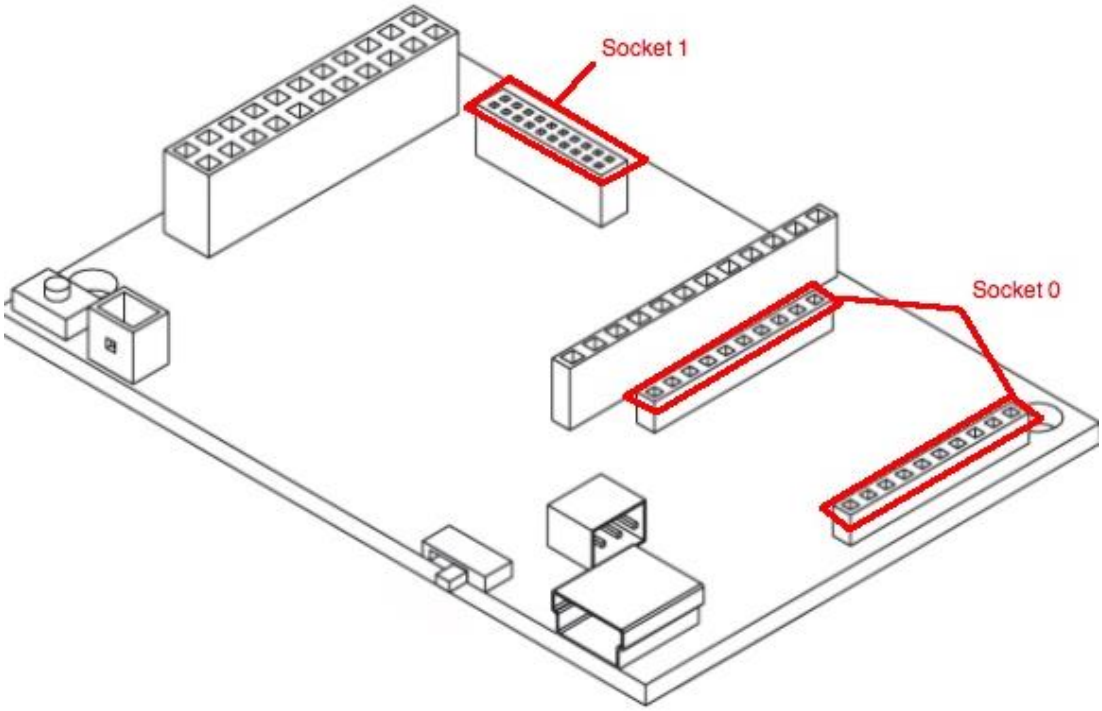


Figura 4.8: Sockets para poder conectar los módulos Sigfox

Así es como quedaría si se utilizaran ambos sockets, aunque en nuestro caso solo se utilizará un socket, el socket 0.

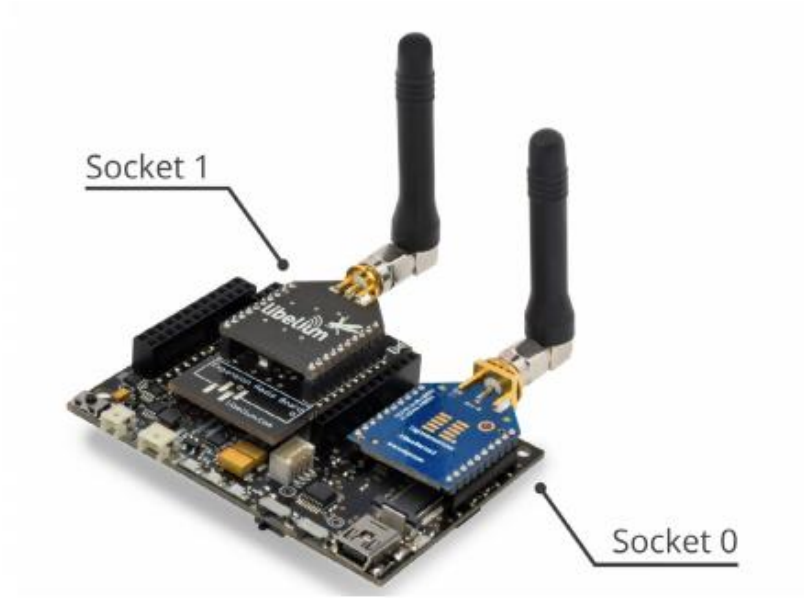


Figura 4.9: Sockets con los módulos de Sigfox

Fuente: http://www.libelium.com

4.1.3 Instalación del software Waspmote Pro Ide v04

Lo primero será descargar tanto el software como las librerías, para ello, se accede a la página de Libelium, se hace clic en la pestaña Development y por último en la opción "old Development Section" (figura 4.10).

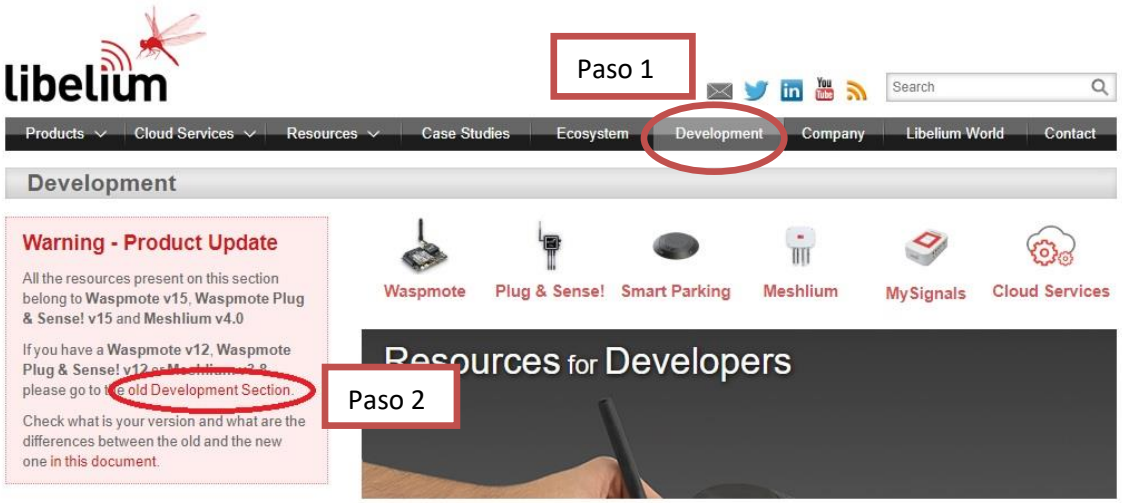


Figura 4.10:Instalación software Waspmote parte 1

Fuente: http://www.libelium.com/development/

Después se selecciona la pestaña "SDK and Applications", apareciendo los archivos que se pueden descargar. Se deberán descargarlas librerías y el programa, en nuestro caso, para Windows, como muestra la figura 4.11.

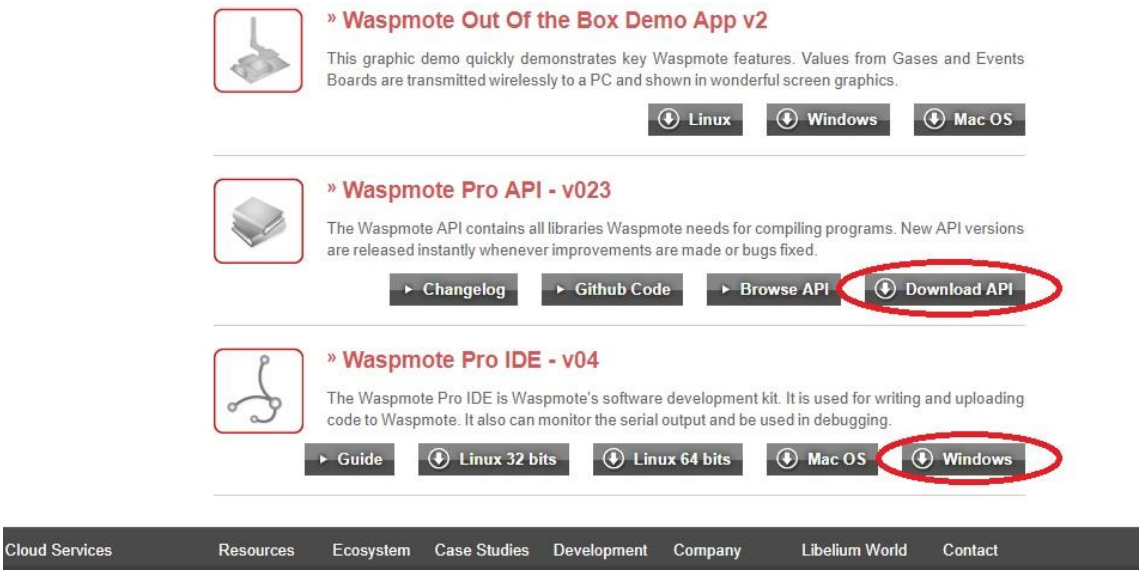


Figura 4.11: Instalación software Waspmote parte 2

Fuente: http://www.libelium.com/v12/development/plug-sense/sdk_applications/

Una vez descargados los archivos, se procede a descomprimirlos y a situarlos en un mismo directorio, para que la aplicación pueda acceder a las librerías.

Por último, se abre el programa accediendo a la carpeta waspmote-pro-ide-v04-windows y ejecutando el archivo .exe (waspmote.exe). Se abrirá la interfaz del programa como se puede ver en la figura 4.12.

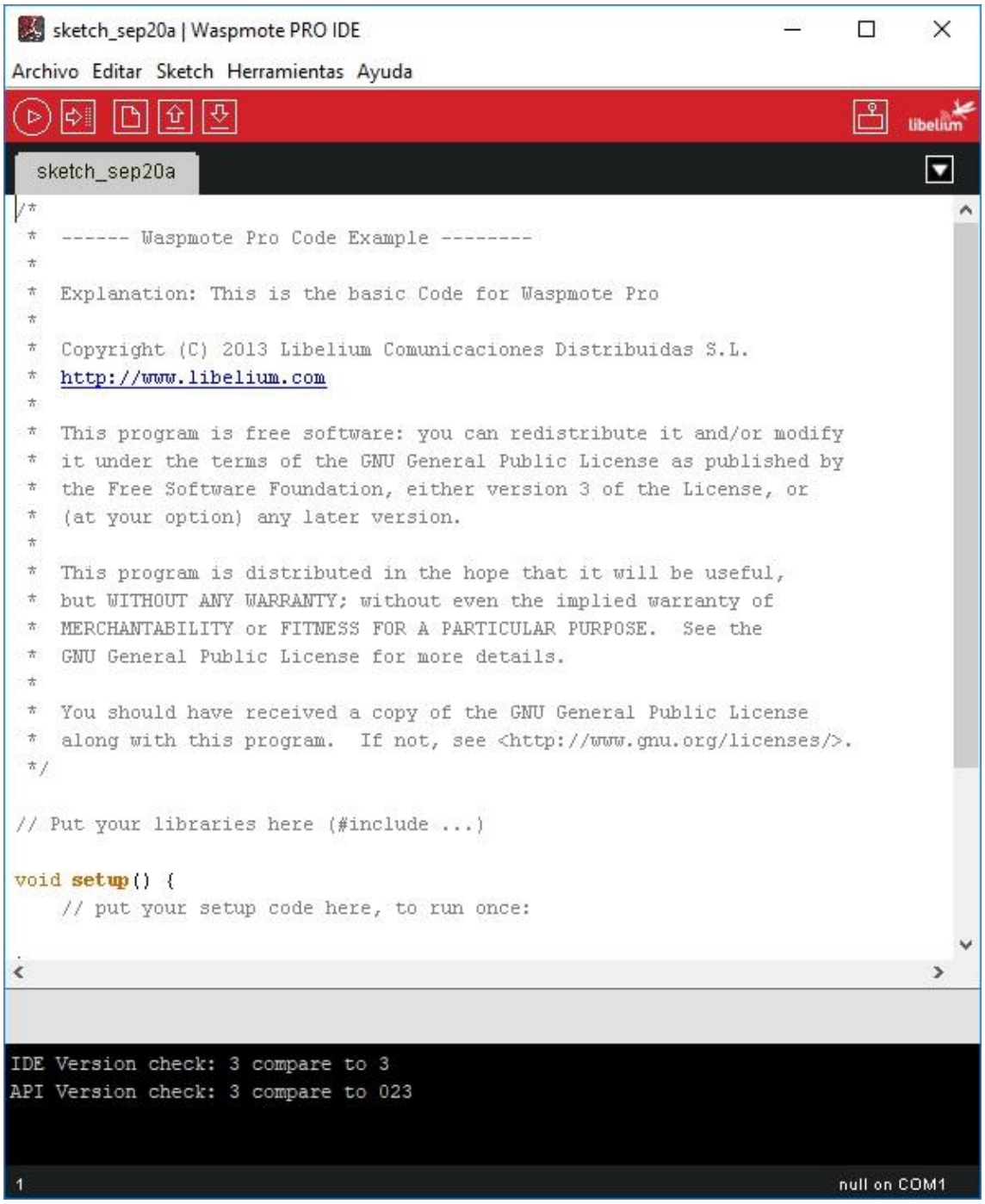


Figura 4.12: Software Waspmote Pro IDE v04 para Windows

Como se puede ver, se encuentran con las funciones del programa y un script abajo para escribir el código que se cargará en la placa Waspmote. Las funciones son la de verificar el script, cargar un script, nuevo script, abrir un script, guardar el script que está abierto y monitorear por el puerto serie el código que está ejecutando nuestra placa Waspmote.

Antes de utilizar el programa, tenemos que seleccionar la tarjeta para poder utilizar las librerías. Para ello, se accede a la pestaña de "Herramientas", se selecciona la casilla "Tarjeta" y aparecerán las API's que se tengan en el mismo directorio, entonces se marca, en nuestro caso, la casilla de waspmote-API-v023 como aparece en la figura 4.13.

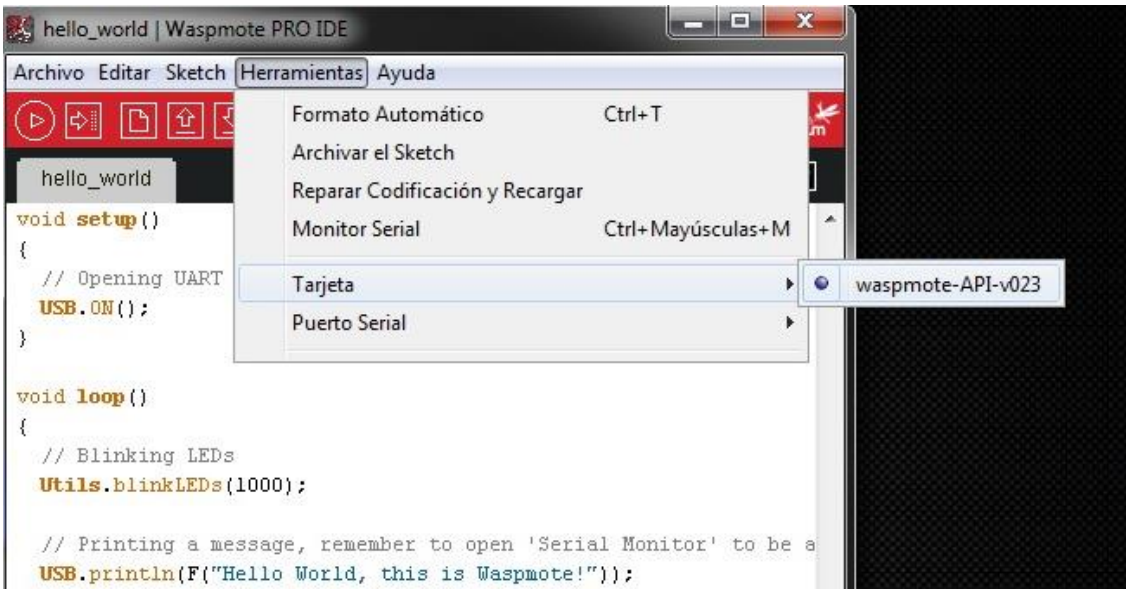


Figura 4.13: Seleccionando librerías.

Una vez que se haya escrito el código que se desea introducir en la placa Waspmote y lo se haya verificado, ya está listo para cargarlo en el sensor. Para ello, primero se selecciona el puerto serie al que está conectado el sensor accediendo a la pestaña de "Herramientas", se selecciona la casilla "Puerto Serial" y se marca, en nuestro caso, el COM 3, que es el puerto USB donde está conectado el sensor Waspmote como se puede ver en la figura 4.14.

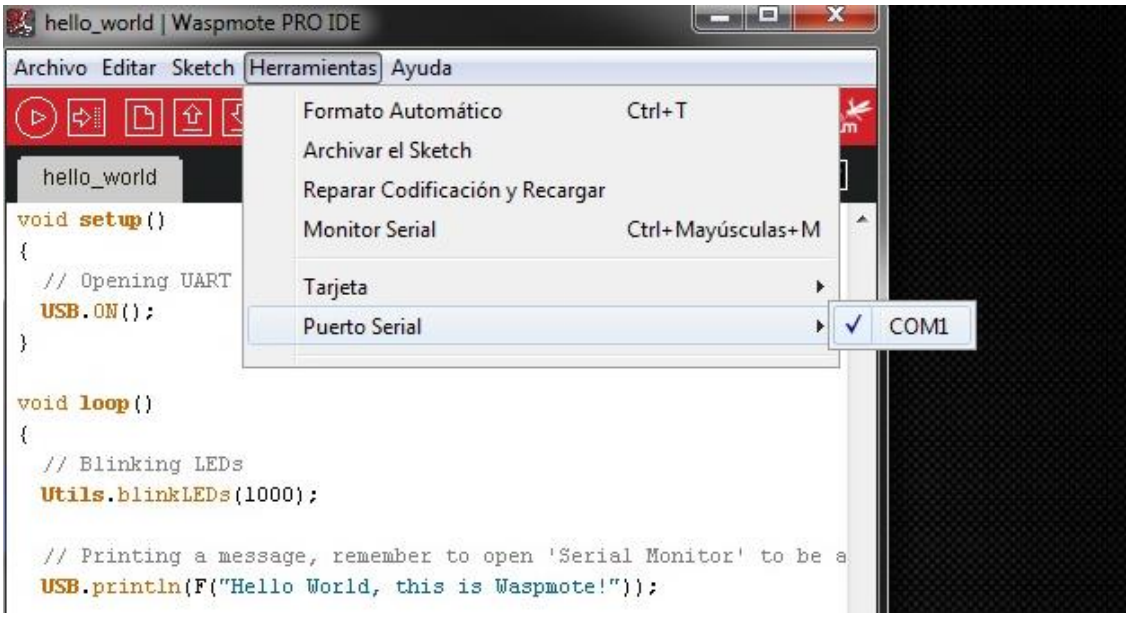


Figura 4.14: Seleccionando Puerto Serie.

4.1.4 Código implementado.

Se utilizará un código capaz de leer el sensor de parking de Libelium y de enviar a través del módulo de Sigfox la información, por eso al principio del script se deberá incorporar las librerías WaspSensorParking y WaspSigfox.

Sólo es necesario que envíe al Backend de Sigfox un 1 o un 0 dependiendo si está ocupada o libre, respectivamente.

Existen 3 modos de ahorrar energía cuando el sensor no necesita transmitir datos:

- Sleep: Se utiliza para ciclos de sueño de 32ms hasta 8 segundos.
- Deep Sleep: Se puede configurar a partir de 8 segundos a minutos, horas o días.
- Hibernate: Igual que el caso anterior, pero desactivando el microprocesador para un consumo menor.

Tanto el modo "Sleep" como el "Deel Sleep" consumen 62µA a diferencia del modo "Hibernate" que consume 0.7µA. Mientras esté despierto el sensor su consumo es de 9mA.

Para este TFG se utilizará el modo "Deep Sleep" como se muestra en el código implementado dentro del anexo1.

4.2 Sensor de Parking

Se utilizará un sensor de parking de la marca Libelium, también conocido como "Smart Parking". El modelo que se usará será el 1.2. Este sensor se montará en nuestra placa Waspmote y se colocará bajo el pavimento, ya que funciona con un sensor magnético detectando al vehículo por su chasis cuando está encima. El sensor funciona por cambios de campo magnético, este tipo de sensores se llaman sensores de campo magnético permalloy (MFS), contienen un material cuya resistencia eléctrica varía con el campo magnético informando al sistema de si detecta o no un vehículo.

4.2.1 Especificaciones

Placa sensor Smart Parking 1.2

- Peso: 20gr.
- Dimensiones: 73.5 x 51 x 1.3 mm
- Rango de temperatura: [-20ºC, +65ºC]
- Tensión de potencia de la placa: 5V
- Tensión de potencia del sensor: 5V
- Corriente máxima admitida de forma continua: 200mA
- Corriente máxima admitida de pico: 400mA

Sensor de campo magnético de tres ejes

- Tensión máxima de funcionamiento: 12V
- Temperatura de trabajo: [-40ºC, +125ºC]
- Resistencia del puente: 600 a 1200 Ohm.
- Tensión típica de salida: 3.5mV/V/gauss
- Consumo medio: 15mA
- Consumo máximo (de pico): 500mA

4.2.2 Partes del “SmartParking” de Libelium

El sensor (MFS) incluye otro módulo conectado de forma vertical para supervisar dicho eje del campo magnético. Cada MFS tiene asignado sólo un módulo vertical, por lo tanto, es importante etiquetarlos con la misma identificación para mantener la relación entre ellos vista en la figura 4.15.

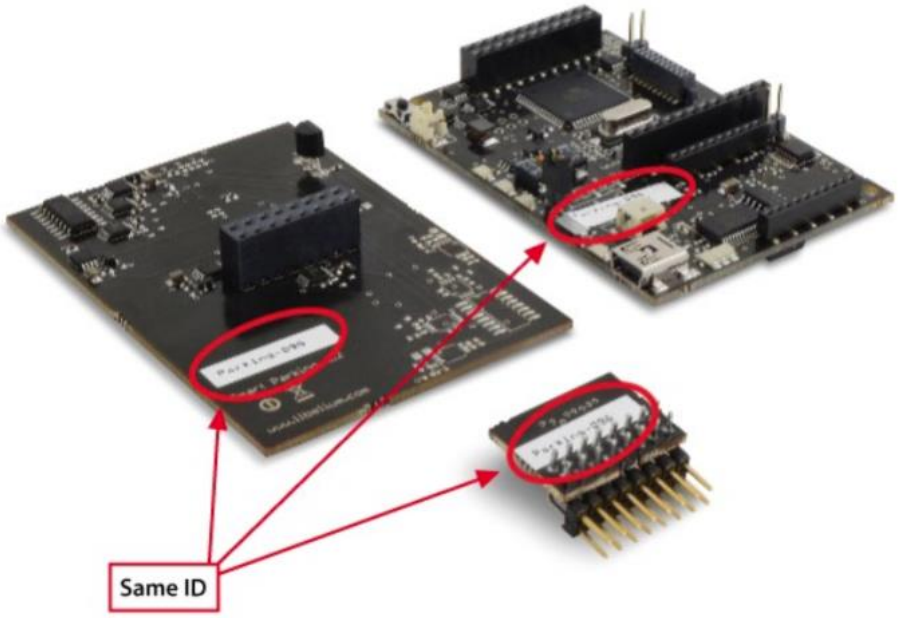


Figura 4.15: Misma identificación en la placa Waspmote, el sensor y su módulo vertical

Fuente: http://www.libelium.com/

El sensor (MFS) se introduce por el sensor I/O y el conector I2C - UART como se muestra en la figura 4.16.

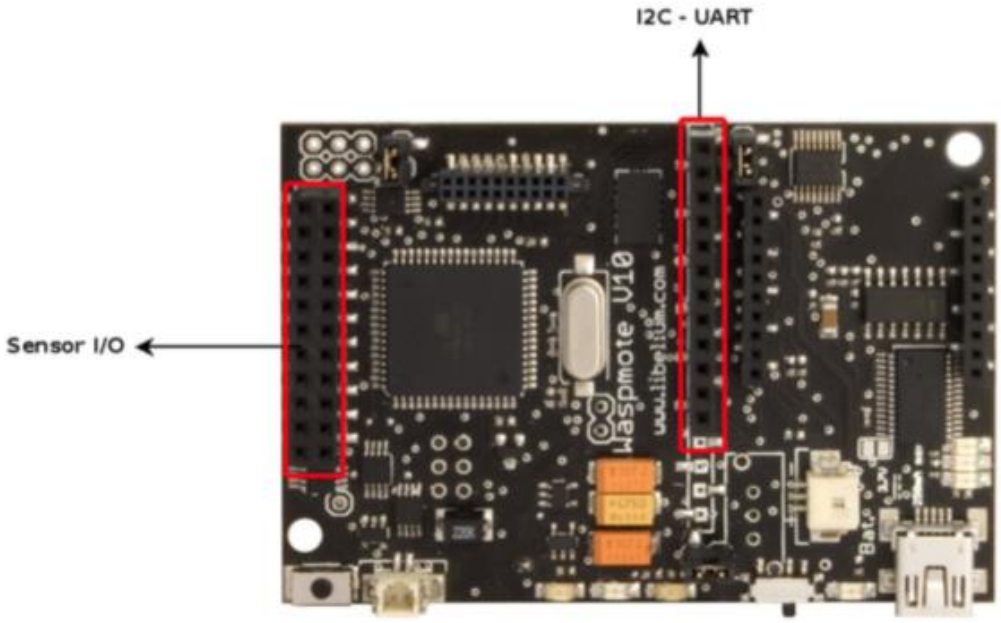


Figura 4.16: Conectores para colocar el sensor de parking

Fuente: http://www.libelium.com/

Una vez montado el módulo en la placa del sensor y este a su vez en la placa Waspmote el resultado sería tal y como se ve en la figura 4.17.

Figura 4.17: Sensor de Parking montado en placa Waspmote

Fuente: http://www.libelium.com

4.2.3 Medición

El sensor (MFS) es, de forma general, una película fina de permalloy obteniendo una resistencia en función del campo magnético. Esta resistencia oscila entre 600 y 1200 Ohm (por lo general 850 Ohm), por lo que entre los dos terminales de salida del sensor hay una tensión de 3.5mV/V/gauss con un voltaje de alimentación de 5V.

Gracias a que incorpora un módulo vertical, se puede monitorear el campo magnético en cualquier dirección. La salida de cada eje (x, y, z) se amplifica con un amplificador de instrumentación y se filtra para evitar problemas causados por otros campos magnéticos. Estas salidas se leen directamente con el convertidor del microprocesador Waspmote de analógico a digital en los pines de ANALOG (1, 2 y 5) de entrada analógica.

Una vez tomadas las lecturas de los ejes, para leerlas solo hay que introducir los comandos “readParking” o “readParkingSetReset” donde captura la tensión analógica en la entrada del convertidor analógico a digital. En la segunda función se aplica un pulso de corriente de 500mA (un microsegundo de ancho) a un circuito interno antes de

leer el sensor para que las moléculas de la película permalloy se realineen con el campo magnético, quitando efectos no deseados debido a la histéresis del material.

La placa incorpora un sensor de temperatura para compensar los efectos que la temperatura tiene en el sensor. Se puede leer la temperatura con la función "readTemperature" en el pin ANALOG7 asignando el valor a una variable entera. Para la compensación de la temperatura, la empresa Libelium aporta los sensores calibrados entre 0 y 45ºC con los parámetro necesarios para calcular la referencia en función de la temperatura obtenida almacenada en la memoria EEPROM del sensor.

4.3 Sigfox: Una breve introducción

4.3.1 Cobertura y radiocomunicaciones

Sigfox es una solución económica y fiable para conectar sensores y dispositivos a baja potencia, transportando la información a través de su red inalámbrica.

Esta tecnología opera en la red LPWAN francés, trabaja como cualquier operador de móvil pero sin tener que utilizar tarjetas SIM. Es una de las redes con mayor despliegue en todo el mundo y opera en la banda de frecuencia pública 868 MHz en Europa y en la de 902 MHz en América. Se puede ver la cobertura que tiene esta red a nivel mundial en la figura 4.18.

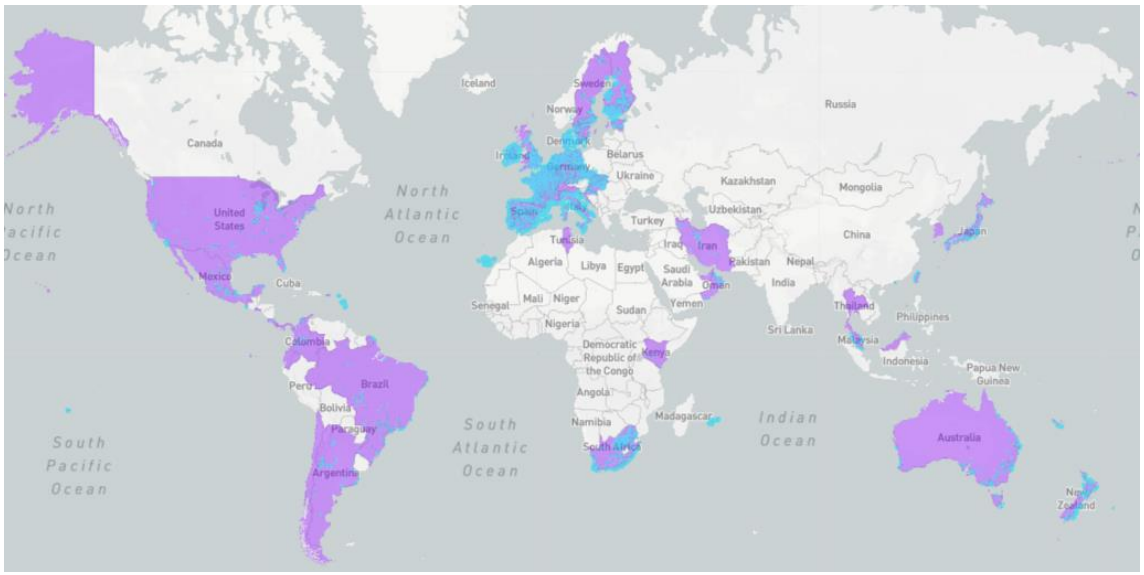


Figura 4.18: Cobertura de la red de Sigfox

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

Los operadores de Sigfox (SNO) despliegan sus servicios por todo el mundo. En color violeta vemos donde hay estaciones base pero sin operadores y en color cian los países con estaciones base y operadores disponibles.

La red Sigfox es inalámbrica, pudiendo dar cobertura a muchos países en un tiempo muy pequeño. Es similar a las redes móviles como el 4G pero con una mayor eficiencia energética, que se obtiene gracias a su tecnología para transmitir sobre un canal estrecho en el espectro de frecuencias (UNB). Utiliza una modulación DBPSK para la subida y GFSK para la bajada de información.

Como se ha comentado, la red opera en una banda de frecuencia pública, concretamente, en la banda ISM (industrial, Scientific and Medical), reservada para uso no comercial de radiofrecuencia, donde coexiste con otras redes sin temor a cancelarse entre ellas. Sigfox colabora con ETSI para estandarizar redes de bajo consumo.

4.3.2 Rasgos distintivos del protocolo Sigfox

Las características que diferencian al protocolo Sigfox de los demás son:

- *Autonomía*: Consumo de energía extremadamente bajo, lo que permite que las baterías duren años.
- *Sencillez*: Sin configuraciones, solicitud de conexión o señalización. Los dispositivos comienzan a estar operativos en cuestión de minutos.
- *Eficiencia de costo*: Se optimiza cada paso para que sea lo más rentable posible, desde los dispositivos hasta nuestra red.
- *Mensajes cortos*: Sin grandes activos o medios, solo pequeñas notificaciones.
- *Uso complementario*: Gracias a su bajo presupuesto y la facilidad de configuración, usted puede usar Sigfox como otro tipo de red, como Wi-Fi, Bluetooth, GPRS, etc.

4.3.3 Etapas del mensaje en Sigfox

Desde que el mensaje sale desde el sensor hasta la red existen diferentes etapas:

- Por lo general, los sensores mientras no están operativos, están en modo hibernación. El primer paso es que el dispositivo despierte para que el sensor haga su lectura y envíe la información.
- El mensaje se dirige hacia la estación base de Sigfox más cercana. Las estaciones cuentan con una antena para captar la información, un amplificador para filtrar la información y un punto de acceso para enviarla al Backend de Sigfox.
- Una vez en el Backend, se solicita una petición GET a nuestra base de datos para poder registrar la información.

Figura 4.19: Esquema del envío de datos.

4.3.4 Características de los mensajes de subida a Sigfox:

- El límite de mensajes al día hacia el Backend de Sigfox es de 140 al día y cada mensaje puede tener un tamaño máximo de 12 bytes.
- El mensaje enviado se recibe por las estaciones base más cercanas, ellas se encargan de enviarlo a la nube y de eliminar los mensajes que se han duplicado.
- La nube Sigfox reenvía la información al Backend, si este está configurado.

4.4 Módulo de transmisión Sigfox

Para poder transmitir por Sigfox y aprovechar todas sus ventajas, se debe adquirir un módulo de transmisión Sigfox que sea compatible con la placa Waspmote 1.2 Pro, en este proyecto se utilizará el módulo vendido por la empresa Libelium (figura 4.20).

Figura 4.20: Módulo Sigfox marca Libelium con antena de 4.5 dBi

Fuente: http://www.libelium.com/

El módulo Sigfox es gestionado por UART y se conecta por el SOCKET 0 y 1 (en nuestro caso se utilizará el SOCKET0).

4.4.1 Especificaciones del módulo de Sigfox

Las características principales del módulo son:

- Frecuencia: ISM 868 MHz
- Potencia de transmisión: 14dBm
- Consumo del chipset:
  - Transmisión: 49mA @ +14dBm
  - Recepción: 13mA
  - Apagado: 0mA
- Sensibilidad en recepción: -126dBm
- Limitación de ETSI: 140 mensajes al día de 12 bytes cada uno.
- Rango: Cada estación base cubre pocos kilómetros. Comprobar la red Sigfox.
- Certificado Sigfox: Clase 0u (el nivel más alto).
- Tensión de funcionamiento: 3.3V

4.4.2 Consumo de tiempo

Es el tiempo que tarda en completarse el proceso, durante este tiempo el módulo consumirá la corriente indicada anteriormente.

- Enviando datos con el módulo encendido: 6 segundos.
- Enviando datos con el módulo apagado: 12 segundos (6 para encenderse y 6 para enviar).
- Envío con ACK con el módulo encendido: 39 segundos (6 para enviar, 14 para entrar en estado de espera de datos y 19 para recibir la contestación).
- Envío con ACK con el módulo apagado: 45 segundos (6 para encenderse, 6 para enviar, 14 para entrar en estado de espera de datos y 19 para recibir la respuesta).

4.5 Backend de Sigfox

El Backend de Sigfox es la interfaz de la aplicación web de Sigfox, en ella se registran los dispositivos que envían su información la cual se puede visualizar a través de la aplicación web.

Para poder utilizarlo, se hará una petición al servicio de Sigfox, contratando el servicio de conectividad para los módulos que transmitirán la información a la red de Sigfox, en nuestro caso, el módulo Sigfox de Libelium con ID: 1AD132. Sigfox proporciona las instrucciones para hacerlo. Primero se crea una cuenta con una dirección de correo electrónico. Después se crea el grupo, usuario, tipo de dispositivo y unidades de dispositivo. Se necesita emparejar el número de PAC y el número de serie de cada módulo con una licencia.

Una vez completado el proceso de activación, tendrá las credenciales (correo electrónico y contraseña) para conectarse al sistema.

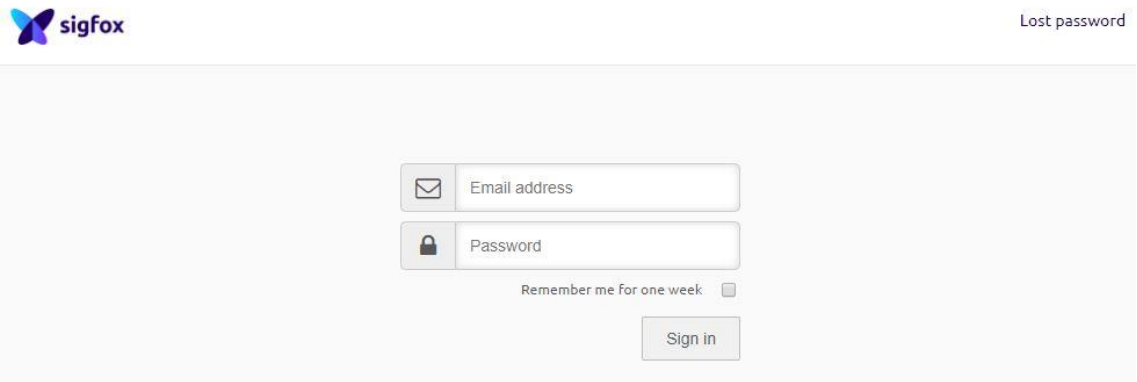
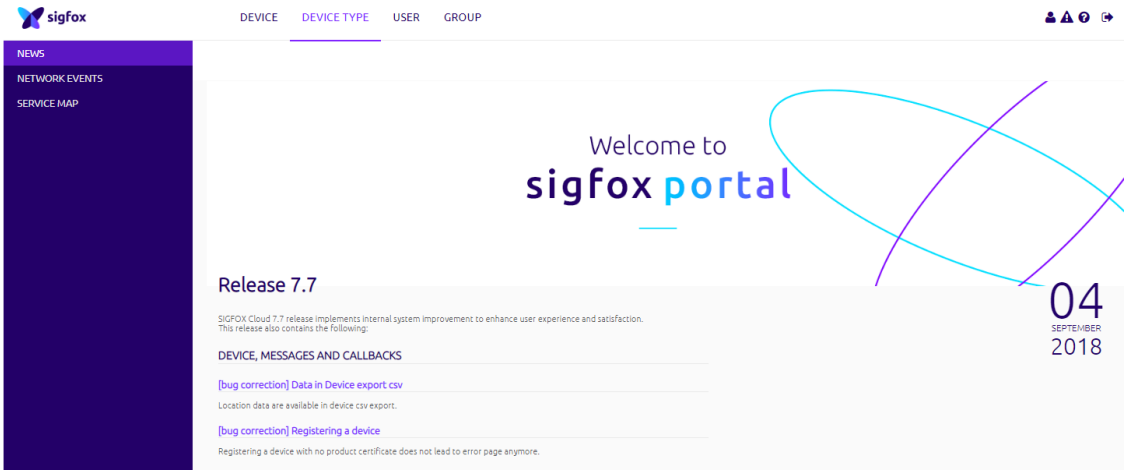


Figura 4.21: Login Backend de Sigfox

Fuente: https://backend.sigfox.com/auth/login

Una vez iniciada la sesión, se encuentra con la página principal, donde están las últimas noticias y actualizaciones, eventos y mapa de cobertura, cambiando de pestaña en el menú de la izquierda como se puede ver en la figura 4.22.



4.22: Página inicial del Backend de Sigfox

Fuente: https://backend.sigfox.com/welcome/news

4.5.1 Device

En la pestaña "Device", en el menú superior, se obtiene toda la información relacionada con los dispositivos asociados a su cuenta.

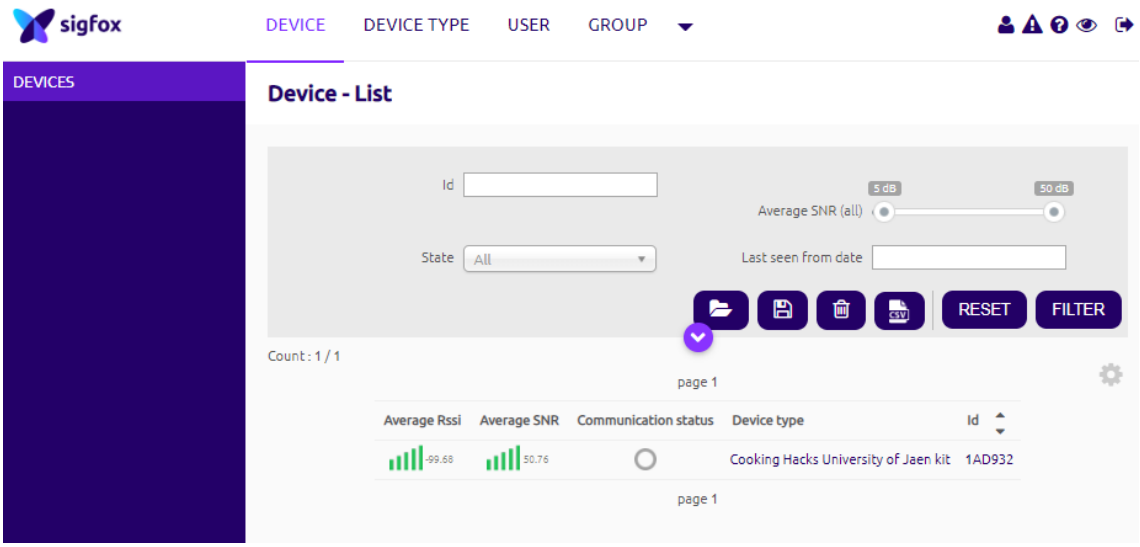


Figura 4.23: Menú Device Backend

Fuente: https://backend.sigfox.com/device/list

Si se pulsa sobre el "Id" de nuestro dispositivo, nos mostrará seis sub-categorías como se puede ver en la figura 4.24.

- Información: Muestra una detallada lista de información; nombre, localización en latitud y longitud, cobertura, etc.
- Ubicación: Muestra un mapa con la cobertura dentro de nuestra área.
- Mensajes: Se visualizan los mensajes enviados por nuestro dispositivo e informa de la hora en la que se recibió, información, localización del dispositivo, calidad de cobertura y callback.
- Eventos: Sirve para monitorear la actividad del dispositivo, notificando a los usuarios de la actividad irregular del dispositivo.
- Estadísticas: Se visualizan cuatro gráficas de mensajes/tiempo, bytes/tiempo, calidad del mensaje recibido/tiempo y RSSI/tiempo.
- Configuración de eventos: Se configuran alertas de dispositivos recibiendo notificaciones por correo electrónico.

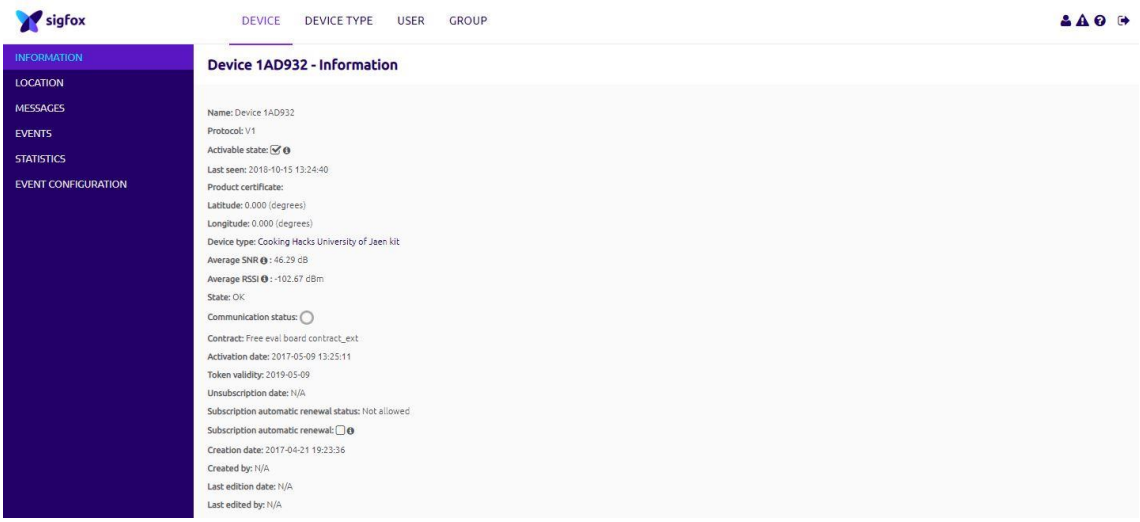


Figura 4.24: Sub-categorías de "Device"

Fuente: https://backend.sigfox.com/device/

4.5.2 Device Type

En la opción "Device Type" se encuentra toda la información del dispositivo que se haya registrado. Además se pueden crear nuevos tipos de dispositivos con el botón "New".

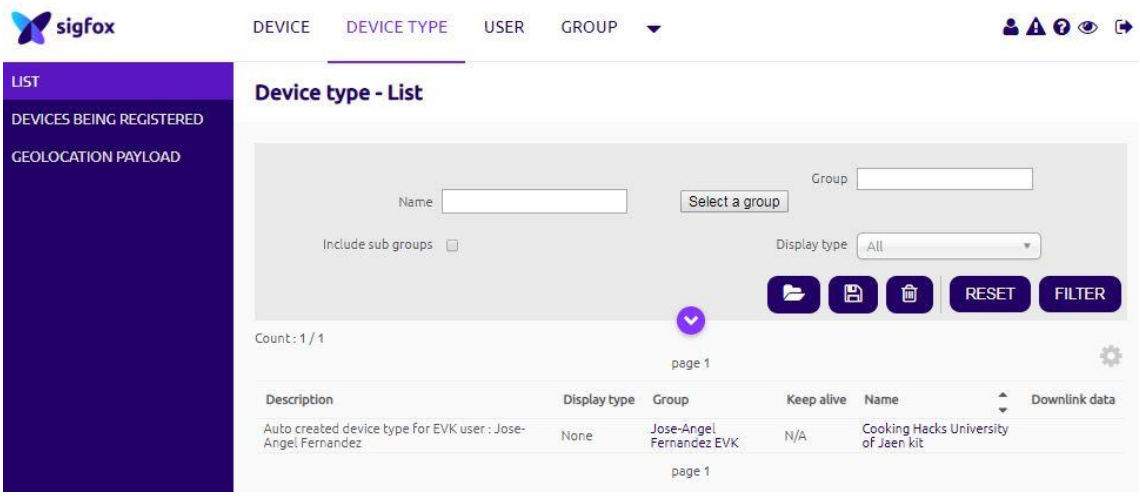


Figura 4.25: Menú "Device Type" del Backend.

Fuente: https://backend.sigfox.com/devicetype/list

También hay dos pestañas más en "Device Type" aparte de List, que es la pestaña por defecto:

- Devices Being Registered: Donde se encuentra una lista de los dispositivos que están transfiriendo.
- Geolocation Payload: Muestra algunas gráficas sobre la información transferida.

Cliqueando en nuestro tipo de dispositivo "Cooking Hacks University of Jaen Kit" de la figura 4.25 se muestran 7 sub-categorías como se ve en la figura 4.26:

- Información: Muestra toda la información del tipo de dispositivo como Id, nombre, grupo, fecha de creación, etc.
- Ubicación: Muestra en un mapa los módulos asignados a este tipo de dispositivo si tienen datos de longitud y latitud.
- Dispositivos asociados: Se muestran todos los dispositivos asociados a este tipo de dispositivo.
- Dispositivos siendo registrados: Muestra una lista de los dispositivos que se están transfiriendo.
- Estadísticas: Muestra 3 gráficas sobre la información transferida.
- Configuración de eventos: Configura alertas informando por el correo electrónico.
- Callbacks: Se programa un "callback" para que reenvíe la información que llega al backend hacia un tipo de dispositivo seleccionado, en nuestro caso a un servidor web.

Figura 4.26: Sub-categorías de "Device Type"

Fuente: https://backend.sigfox.com/devicetype/

4.5.3 User

En esta sección se pueden ver a todos los usuarios creados en un determinado grupo. Se pueden crear usuarios siempre asociándolos a un grupo. Para crear un usuario se cliquea en el botón "New" y se rellena el formulario.

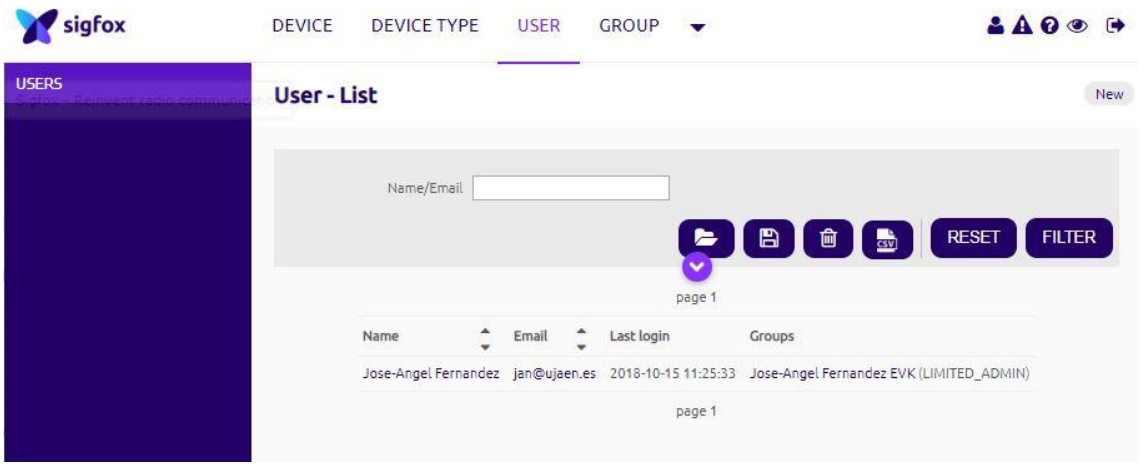


Figura 4.27: Menú "User" del Backend

Fuente: https://backend.sigfox.com/user/list

Si se pulsa sobre el nombre, en nuestro caso, "Jose-Angel Fernandez" se visualiza la información de dicho usuario como el IP y la fecha de las conexiones a esta cuenta, el email, ubicación, grupos, etc. (Figura 4.28).

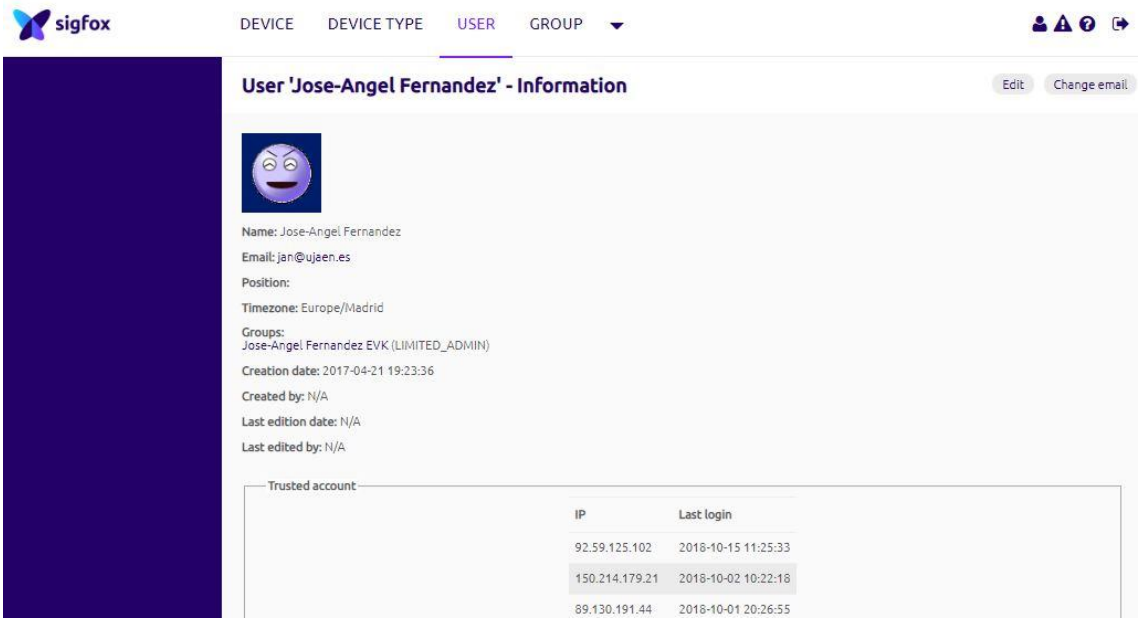


Figura 4.28: Información de usuario

Fuente: https://backend.sigfox.com/user/

4.5.4 Group

Este apartado muestra el grupo al que pertenece (figura 4.29). Se accede a la información del grupo y 4 sub-categorías más pinchando encima del grupo, en nuestro caso, "Jose-Angel Fernandez EVK" como muestra la figura 4.30.

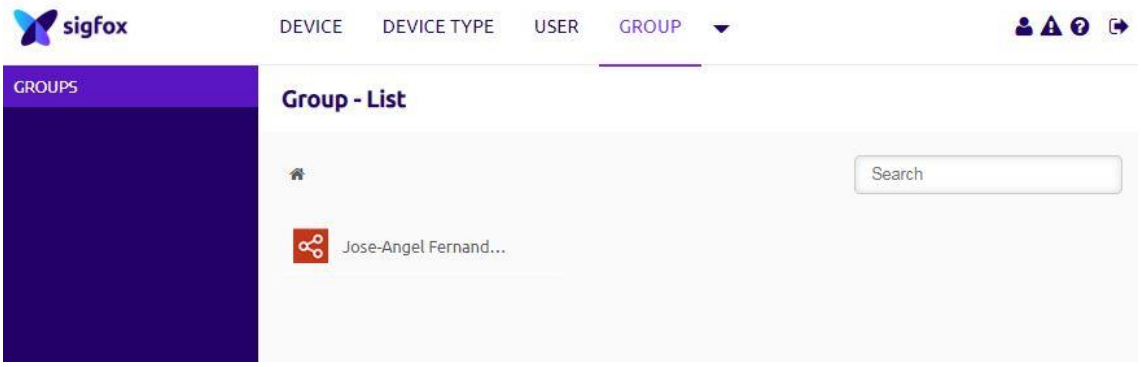


Figura 4.29: Menú Group del Backend

Fuente: https://backend.sigfox.com/group/list

Una vez dentro encontramos 5 sub-categorías:

- Información: Muestra la información del grupo como nombre, descripción, nombre del cliente, etc.
- Usuarios asociados: Ofrece una lista con los usuarios que existen en el grupo.

- Tipos de dispositivos asociados: Muestra una lista con los tipos de dispositivos asociados al grupo.
- Configuración de eventos: Información similar a la mostrada en las otras secciones de "Configuración de Eventos" pero dependiendo del tipo de grupo.
- Acceso API: Sirve para acceder a algunas funciones del Backend desde un servicio web.

Figura 4.30: Sub-categorías de "Group"

Fuente: https://backend.sigfox.com/group/

4.6 Callbacks

El Backend de Sigfox puede reenviar automáticamente mensajes usando el sistema "Callbacks". La opción "Callbacks" se encuentra dentro de la sección "Device Type" y permite crear una devolución de información a un tipo de dispositivo seleccionado.

Figura 4.31: Callbacks del Backend de Sigfox

Fuente: https://backend.sigfox.com/devicetype/list

A través del Callbacks se puede enviar la información que llega al backend hacia plataformas externas, en nuestro caso, se envía la información a una base de datos.

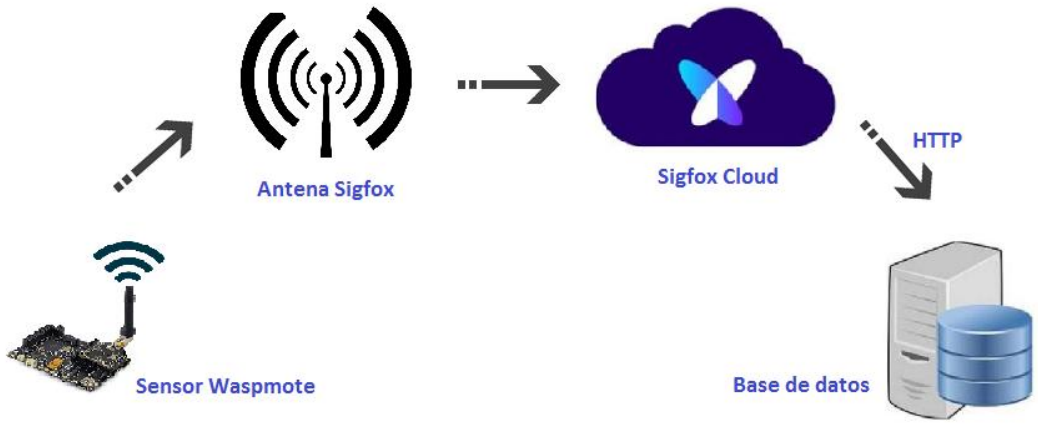


Figura 4.32: Esquema del envío de datos usando "Callbacks"

Una vez configurado el "Callbacks", la devolución de datos se activa cuando el backend recibe un mensaje o cuando se detecta la pérdida de comunicación del dispositivo.

Hay 3 tipos diferentes de "Callbacks":

- Datos
- Servicio
- Error

Se necesita un dispositivo y una cuenta registrada para configurar un Callback y el dispositivo debe estar configurado con un tipo de dispositivo y grupo, en nuestro caso el grupo "Jose-Angel Fernandez EVK" y nuestro tipo de dispositivo "Cooking Hacks Universitiy of Jaen kit".

Para crear un "callback" se pulsa el botón "New" (arriba a la derecha) como se muestra en la figura 4.31. En nuestro caso se seleccionará la opción "Custom callback" mostrada en la figura 4.33.

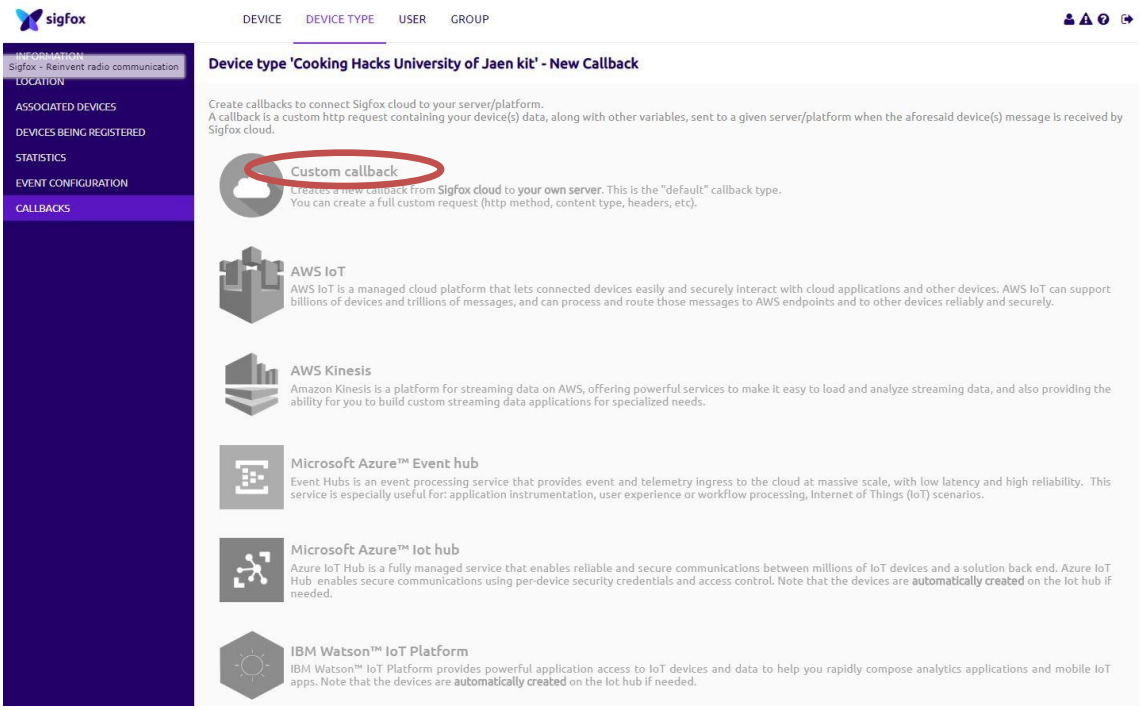


Figura 4.33: Distintas opciones de "Callbacks"

Fuente: https://backend.sigfox.com/devicetype/

Una vez seleccionada la opción elegida, se debe rellenar el siguiente apartado como muestra la figura 4.34.

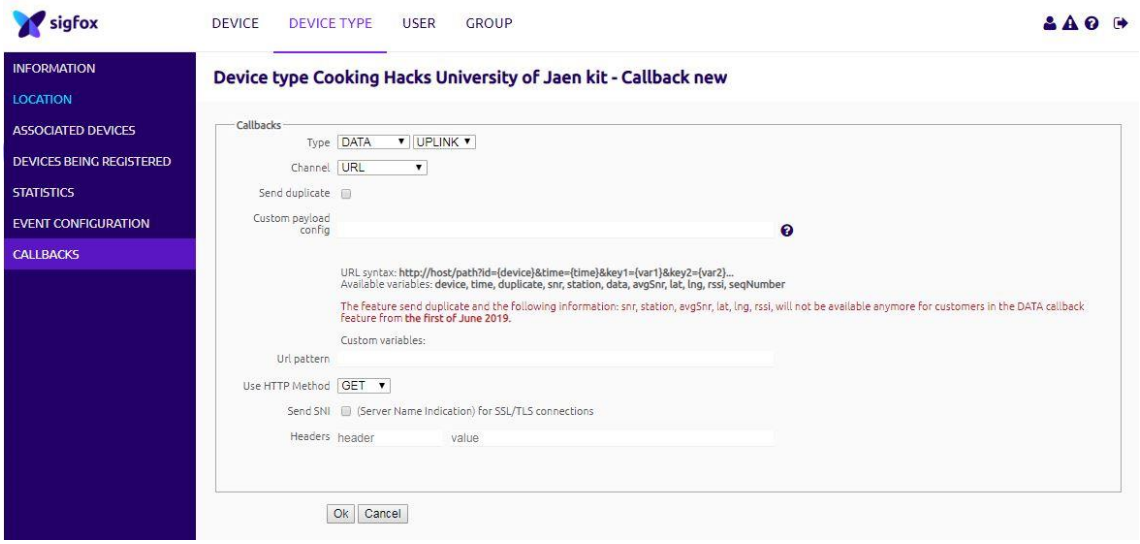


Figura 4.34: Nuevo Callback

Fuente: https://backend.sigfox.com/devicetype/

En esta sección lo más importante es rellenar la casilla "Urlpattern" con la dirección de nuestro servidor e introduciendo las variables al final. En "Custom payload

config" se configura la variable "parking" como entero. Y por último se utiliza el método HTTP "GET" terminando de configurar nuestro nuevo "Callback" como muestra la figura 4.35.

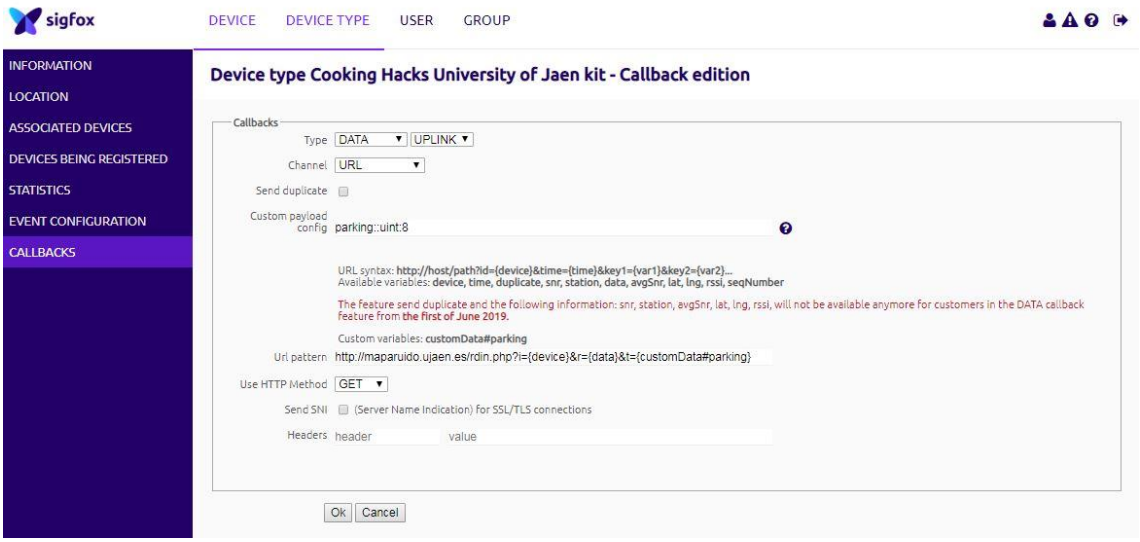


Figura 4.35: Callback configurado.

Fuente: https://backend.sigfox.com/devicetype/

4.7 Servidor Web

El Callbacks de Sigfox enviará la información a un servidor web. En este TFG, se utilizará el servidor del grupo de investigación "Ingeniería de sistemas telemáticos". Esta plataforma ya dispone de una implementación para la monitorización de la contaminación acústica, la calidad de aire y también para el aparcamiento en exterior, que es el que se utilizará en este TFG como se observa en la figura 4.36 en el menú de la izquierda.

A día de hoy no se puede acceder al servidor del grupo de investigación por motivos de seguridad, ya que se está llevando a cabo la migración a un servidor seguro que soporte https. La prueba para ver si se modificaba el estado del parking se ha realizado en local en un servidor web. La siguiente figura muestra una captura del servidor (figura 4.36).

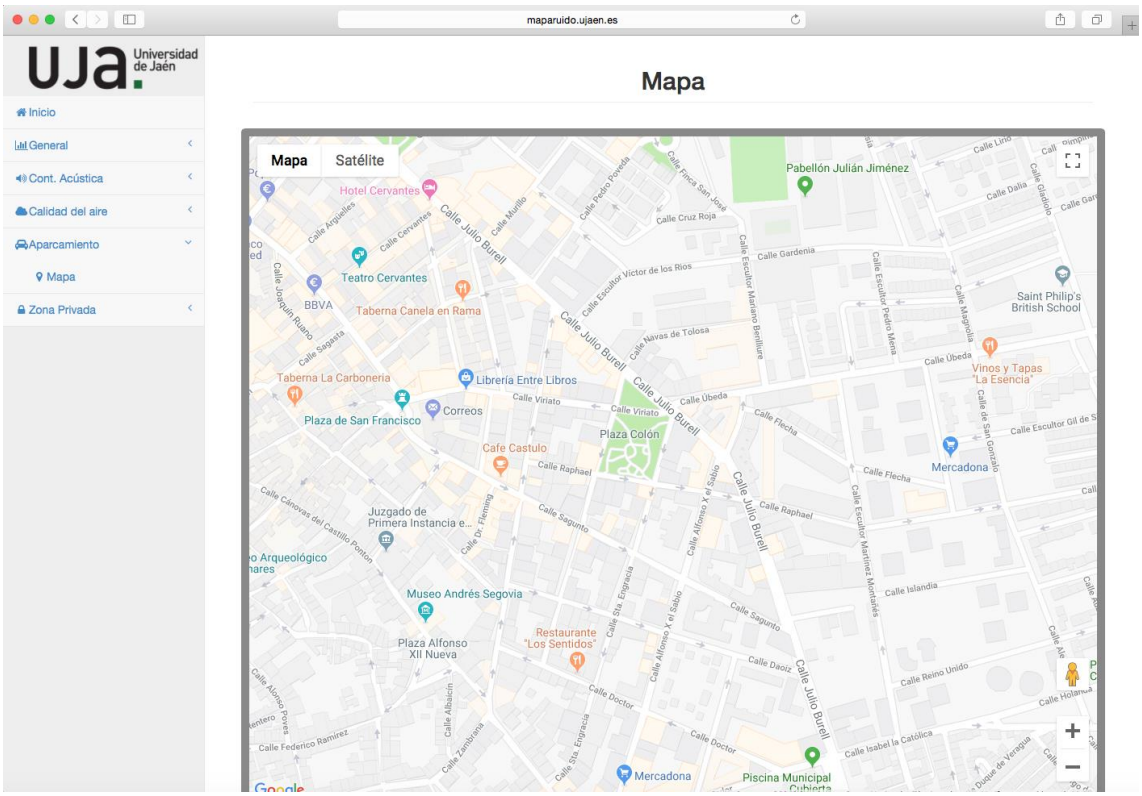
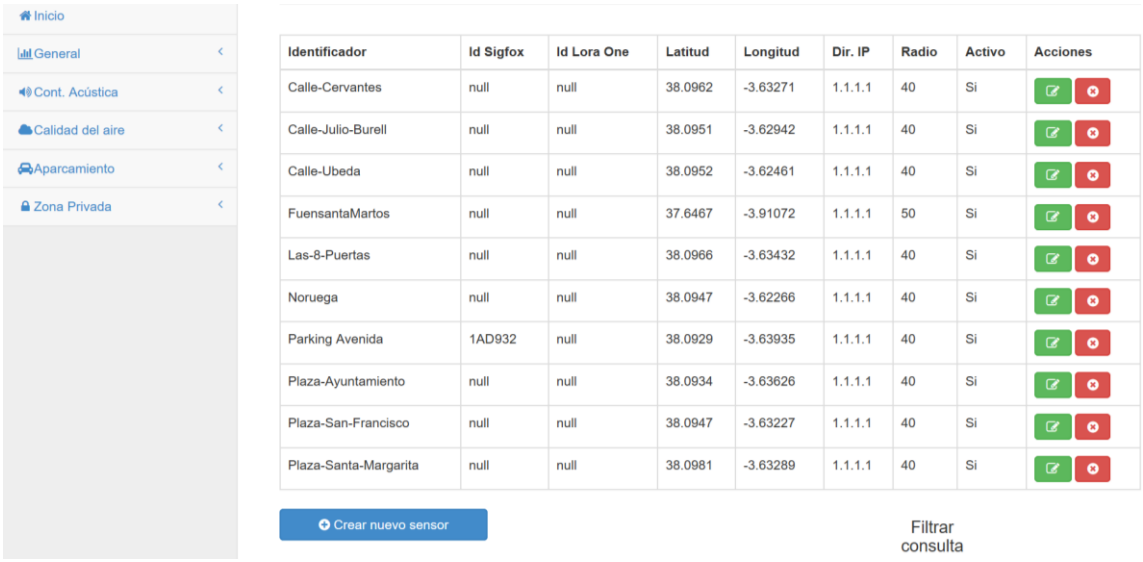


Figura 4.36: Servidor web

Como la base de datos ya está creada, sólo habría que dar de alta al sensor para incorporarlo, como se puede apreciar en la figura 4.37. Cuando se crea el sensor se debe de indicar su denominación, su id de Sigfox (1AD932) y las coordenadas donde se encuentra.



Identificador	Id Sigfox	Id Lora One	Latitud	Longitud	Dir. IP	Radio	Activo	Acciones
Calle-Cervantes	null	null	38.0962	-3.63271	1.1.1.1	40	Si	
Calle-Julio-Burell	null	null	38.0951	-3.62942	1.1.1.1	40	Si	
Calle-Ubeda	null	null	38.0952	-3.62461	1.1.1.1	40	Si	
FuensantaMartos	null	null	37.6467	-3.91072	1.1.1.1	50	Si	
Las-8-Puertas	null	null	38.0966	-3.63432	1.1.1.1	40	Si	
Noruega	null	null	38.0947	-3.62266	1.1.1.1	40	Si	
Parking Avenida	1AD932	null	38.0929	-3.63935	1.1.1.1	40	Si	
Plaza-Ayuntamiento	null	null	38.0934	-3.63626	1.1.1.1	40	Si	
Plaza-San-Francisco	null	null	38.0947	-3.63227	1.1.1.1	40	Si	
Plaza-Santa-Margarita	null	null	38.0981	-3.63289	1.1.1.1	40	Si	

Figura 4.37: Sensor en Base de datos

En la figura 4.38 se observa que nuestro sensor está dado de alta bajo la denominación de "Parking Avenida".

Las-8-Puertas	null	null	38.0966	-3.63432	1.1.1.1	40
Noruega	null	null	38.0947	-3.62266	1.1.1.1	40
Parking Avenida	1AD932	null	38.0929	-3.63935	1.1.1.1	40
Plaza-Ayuntamiento	null	null	38.0934	-3.63626	1.1.1.1	40
Plaza-San-Francisco	null	null	38.0947	-3.63227	1.1.1.1	40
Plaza-Santa-Margarita	null	null	38.0981	-3.63289	1.1.1.1	40

Figura 4.38: Sensor "Parking Avenida" dado de alta en el servidor.

5. RESULTADOS

En este apartado se mostrarán las pruebas que se han llevado a cabo y los resultados obtenidos en el TFG.

En las figuras 5.39 y 5.40 se visualiza el material utilizado. En la figura 5.39 se muestra el módulo de Sigfox, el sensor de parking, la placa Waspmote y la antena utilizada.

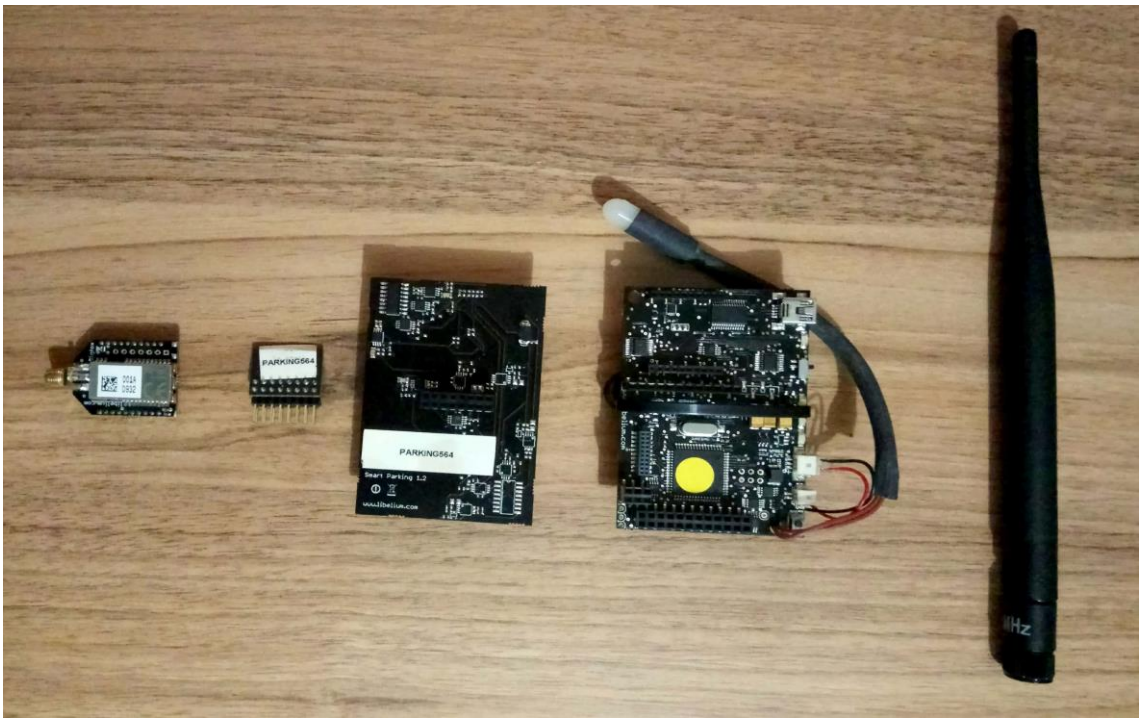


Figura 5.39: Módulos que componen el sensor de Parking.

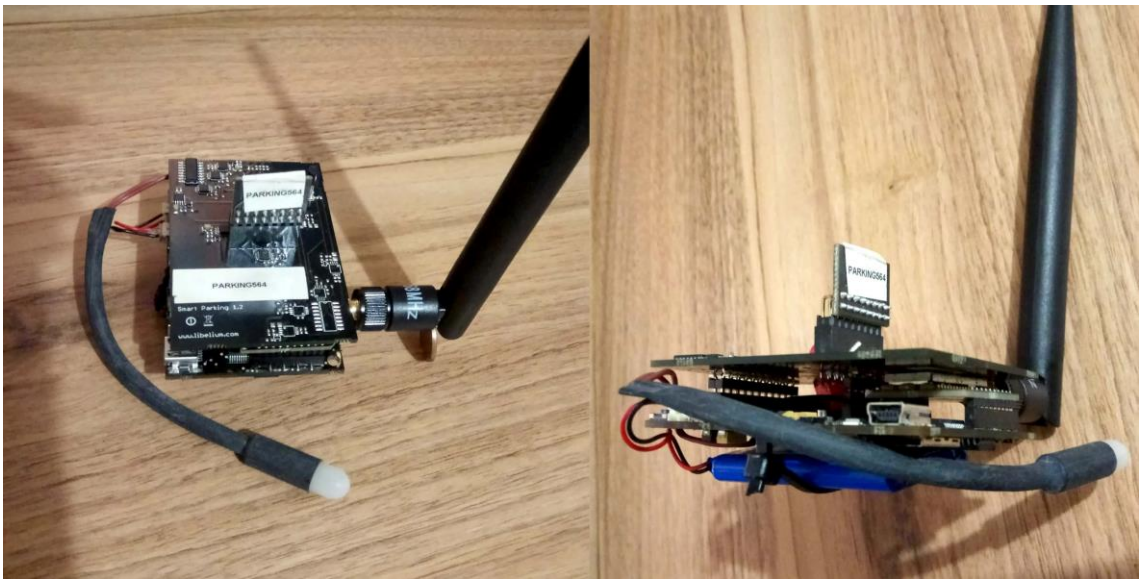


Figura 5.40: Sensor montado captado desde dos perspectivas.

Como se comentó en el apartado 5.2.2, los identificadores del sensor Smart Parking 1.2 junto al sensor vertical y la placa Waspmote Pro v1.2 coinciden correspondiendo al nombre de "PARKING564" como se observa en la figura 5.41. Además se puede comprobar que el ID mencionado del módulo Sigfox coincide con el que aparece en la imagen.

Figura 5.41: Partes del sensor con el mismo ID y módulo Sigfox correspondiente al visualizado en las imágenes del Backend de Sigfox.

A continuación, se muestra en la figura 5.42 el recipiente utilizado para hacer de caparazón al sensor de parking. También se puede ver la batería y una esponja rosa que hace de molde para que el equipo quede más compacto.

Figura 5.42: Equipo completo preparado para la inserción en la calzada.

5.1 Pruebas realizadas

La primera prueba fue realizada después de implementar el código que se carga al sensor. Para ver que funciona correctamente el sensor de parking se han llevado a cabo las siguientes operaciones:

- Implementar el código para que indique el sensor parking cuando está ocupada la plaza.
- Cargar el código en el sensor con la ayuda del programa Waspmote Pro Ide V04.
- Se utiliza un objeto metálico posicionándolo encima del sensor (figura 5.43), como si se tratase de un vehículo.

Figura 5.43: Se utiliza un objeto metálico en sustitución de un automóvil.

- Por último, se observa por el monitor serial del programa si la plaza está libre u ocupada y si se ha enviado el estado de la plaza al backend de Sigfox como se puede ver en las figuras 5.44 y 5.45 respectivamente.

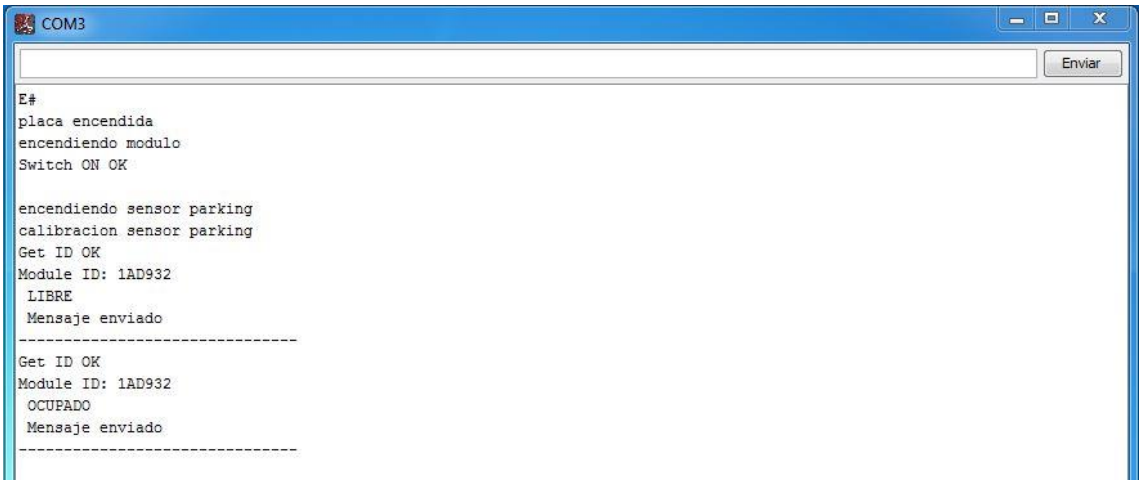


Figura 5.44: Se visualiza por monitor serial el resultado del código ejecutado por el sensor.

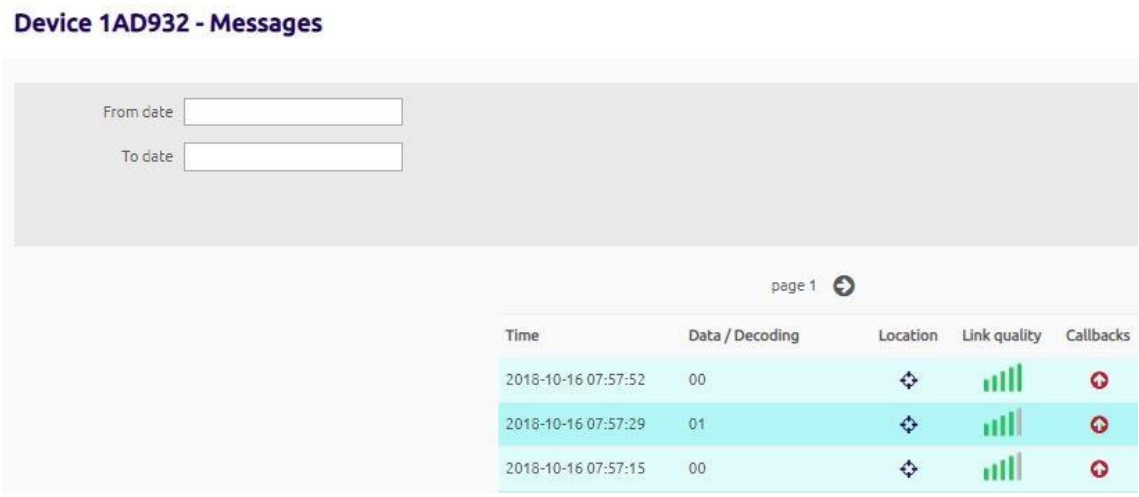


Figura 5.45: Mensajes recibidos al Backend de la primera prueba.

Fuente: https://backend.sigfox.com/device/

Una vez que se han realizado todas las partes de la primera prueba con éxito podemos pasar a la siguiente.

La segunda prueba ha sido realizada en el parking subterráneo de la Escuela Politécnica Superior de Linares. Para ello se ha colocado el sensor en una plaza del parking, primero con la plaza libre y después aparcando un automóvil para que la plaza quede ocupada. Se ha configurado el sensor para que tome valores cada 5 minutos y la prueba ha sido satisfactoria como podemos ver en la figura 5.46 donde aparece el backend de Sigfox mostrando los mensajes por el sensor, 00 para la plaza libre y 01 para la plaza ocupada.

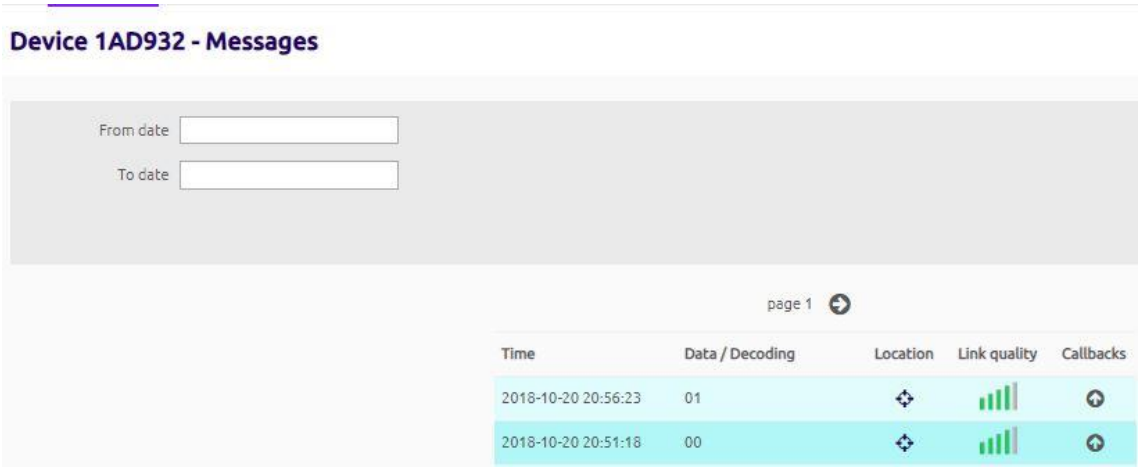


Figura 5.46: Mensajes recibidos al Backend desde el parking de la EPSL.

Fuente: https://backend.sigfox.com/device/

5.2 Cobertura

Se ha podido comprobar, mientras se realizaban las pruebas, que todos los mensajes enviados por el sensor llegaron satisfactoriamente al backend de Sigfox. Desde nuestra experiencia en este TFG el sensor ha sido probado en varias localidades de Jaén y Cádiz como Marmolejo, Arcos de la Frontera, Linares y la ciudad de Jaén y siempre se ha obtenido buena cobertura. Esto es un punto positivo desde la opinión del usuario ya que tanto los dispositivos de Libelium como la red de Sigfox dan confianza y seguridad de que su funcionamiento será óptimo y no presentará problemas en el futuro mientras que la batería del dispositivo perdure (tema expuesto en el siguiente apartado).

Como se visualiza en la figura 5.47, a día de hoy, la red Sigfox cubre el 99% (casi toda la totalidad) del territorio español, queda marcada la localidad de Linares, donde se ha realizado sobre el terreno la segunda prueba con el equipo completo de forma satisfactoria.

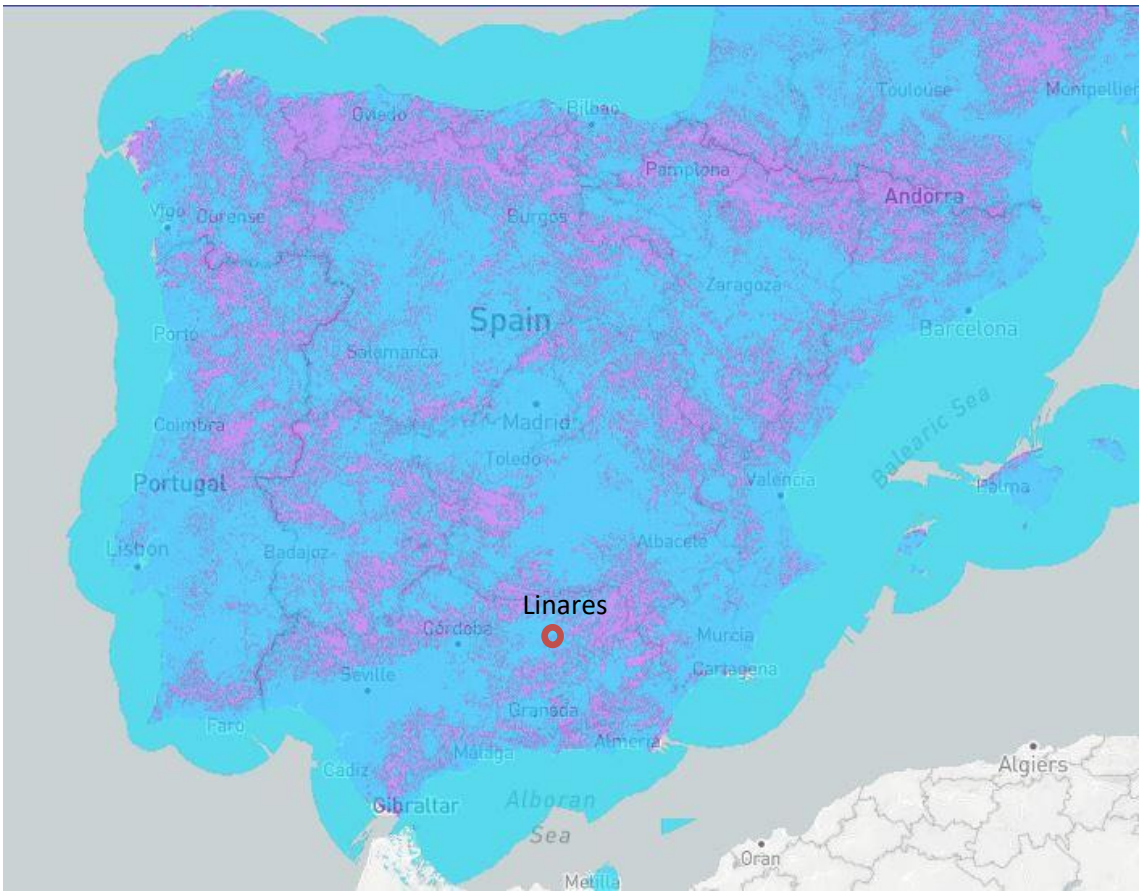


Figura 5.47: Cobertura en España de la red de Sigfox.

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

Para comprobar el funcionamiento del servidor, se hizo una captura visualizando el icono en rojo de la plaza ocupada dentro del mapa, en la avenida de Andalucía (Linares, Jaén), como muestra la figura 5.48.

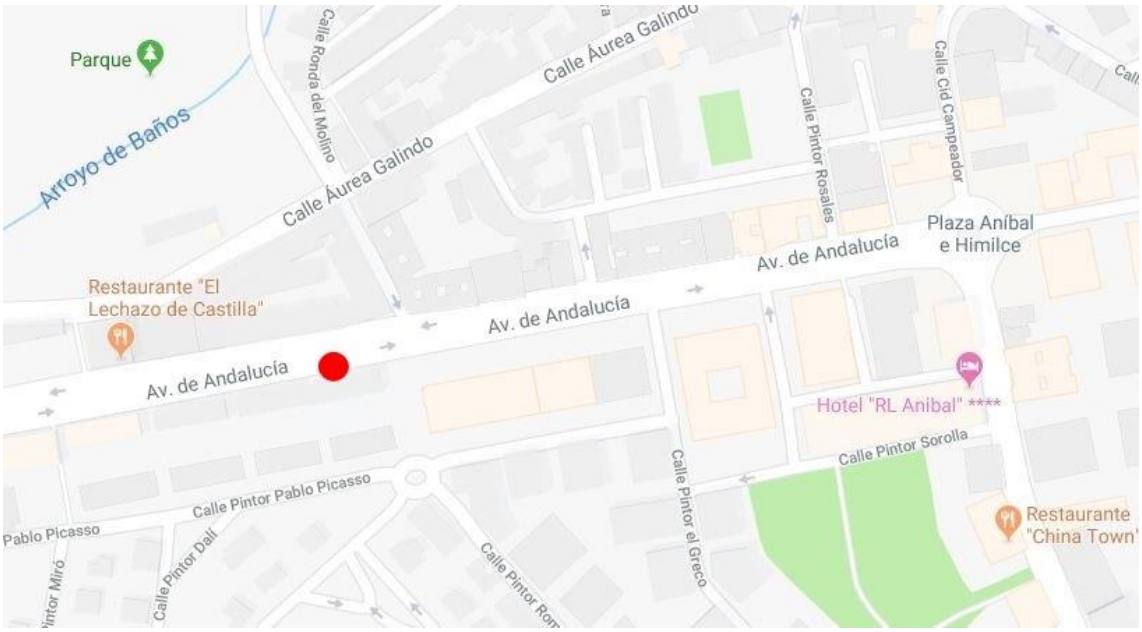


Figura 5.48: Mapa del Servidor Web mostrando la plaza ocupada.

5.3 Batería

Para la primera prueba, el sensor ha funcionado con una batería de 1150mAh, como se muestra en la figura 5.49.

Figura 5.49: Batería de 1150mAh.

Para la segunda prueba, se ha utilizado una batería acorde a las necesidades del TFG, una batería de amperaje más grande que la anterior, 26Ah, para que su vida útil sea más larga y haya que cambiarla menos veces (figura 5.50).

Figura 5.50: Batería para la prueba en vía y utilizada en el TFG.

5.3.1 Duración de la batería

Para saber cuánto tiempo se debe utilizar la batería antes de cambiarla se procede a realizar los siguientes cálculos.

La capacidad de la batería que se utiliza en este TFG es de 26Ah, pero para no dañarla se tendrá que utilizar el 80% de su capacidad, ya que si su capacidad baja del 20% aguantará menos ciclos de carga y descarga, ya que cuanto más profunda sea la descarga de la batería menos ciclos de carga y descarga aguantará. Por lo tanto, se realizarán los cálculos con 20.8Ah de capacidad.

Se ha implementado un código que introduce al sensor en un estado de sueño profundo, optimizando el consumo cuando no está enviando el estado de la plaza. En la siguiente lista se exponen los consumos dados por las diferentes partes del sensor cuando está encendido:

- La placa Waspmote 1.2 Pro v1.2 consume 9mA.
- El sensor Parking 15mA de consumo medio y 500mA de pico.
- El módulo de transmisión Sigfox 49mA mientras envía.

Se ha medido el tiempo que está despierto el sensor, desde que se despierta y manda el estado de la plaza hasta que entra en sueño profundo, el tiempo es de 17

segundos. Por lo tanto, se consumirá durante 17 segundos los 9mA de la placa y 15mA del sensor. Durante 6 de esos 17 segundos el módulo Sigfox consume 49mA. Durante 1 microsegundo de los 17 segundos el sensor de parking consume 500mA en realinear las moléculas de la película permalloy con el campo magnético antes de la lectura para quitar efectos no deseados debido a la histéresis del material (algo que se podrá despreciar debido a su bajo consumo).

Mientras el sensor esté en modo "Deep sleep" el único consumo es de 62µA gracias a este estado de sueño profundo.

En este TFG se han utilizado 120 mensajes de los 140 disponibles, por lo tanto, los 17 segundos se convierten en 2040 y los 6 de transmisión de datos en 720 segundos. Se pasarán los miliamperios consumidos en 2040 segundos a 3600 segundos y los miliamperios consumidos en 720 segundos a 3600 segundos que tiene una hora para ver los miliamperios que se consumirían de forma continua:

- 15 + 9mA = 24mA cada 2040 segundos que serían 13.6mA cada 3600 segundos.
- 49mA cada 720 segundos serían 9.8mA cada 3600 segundos.
- 500mA cada 120 microsegundos serían 16.7pA que se desprecian.
- A esto, se añaden 62µA que consume la placa en estado de sueño profundo cada 3515 segundos de 3600 que serían 0.061mA.

Con las anteriores medidas se obtendría 23.461mA de consumo continuo. Para saber las horas que duraría la batería se divide la capacidad de la batería entre el consumo medio dando 886.85 horas que son (36.94 días redondeando a la baja) 36 días que puede estar funcionando la batería sin tener que cambiarla.

A continuación, se realiza una prueba visualizando los datos entregados al backend con el código implementado en el anexo 1.

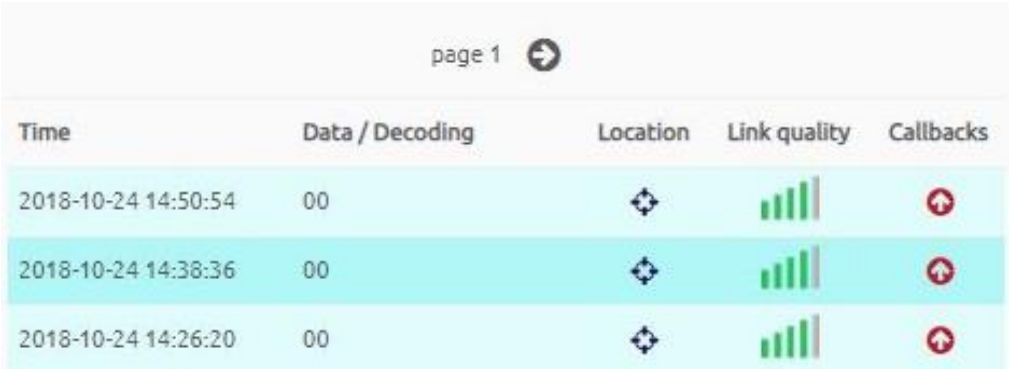


Figura 5.51: Entrada de mensajes cada 12 minutos con el modo "Deep Sleep".

Fuente: https://backend.sigfox.com/

6. CONCLUSIONES

Una vez alcanzados los objetivos, se evaluarán los resultados obtenidos en este apartado.

La programación del código no ha sido muy complicada, ya que la empresa distribuidora de los sensores utilizados en este TFG, Libelium, proporciona muchas ayudas en su página web, además de las librerías necesarias.

Los módulos que componen este sensor de la marca Libelium han transmitido, durante la realización del TFG, seguridad, debido a su buena fabricación y robustez a la hora de manipularlos.

La red Sigfox es una gran elección ya que la cobertura en España es muy buena y cubre todo el territorio español (en este TFG se han enviado mensajes a la red desde 4 municipios sin tener ningún problema). Hay que añadir que el precio para beneficiarse del servicio del backend de Sigfox es económico. Dos puntos positivos a tener en cuenta con la posibilidad de reenviar la información a un servidor web.

Durante la realización del TFG no se ha encontrado ningún punto negativo a los sensores utilizados ni a la red Sigfox.

Una vez realizadas las pruebas, tanto en interior como en exterior, han sido todo un éxito, no se han encontrado fallos de software ni de hardware.

La solución propuesta de sensores resulta ser muy útil debido a que se conoce la posición y la disposición de las plazas de aparcamiento que dispongan de esta tecnología. De esta forma la tarea de aparcar se puede convertir en un trabajo sencillo y agradable, algo difícil de imaginar en la actualidad.

A nota personal, se pensaba que la batería de 20.8Ah (el 80% de 26Ah) duraría más de 36 días debido a que se utiliza una tecnología de bajo consumo, pero también se entiende el resultado, ya que se utiliza casi el límite de mensajes propuestos por Sigfox. Utilizando la mitad de mensajes duraría algo menos de 72 días y con 30 mensajes más de tres meses.

7. LINEAS DE FUTURO

Para mejorar este TFG se proponen los siguientes objetivos:

- Se podría implementar un código para que tomara las mediciones con diferentes intervalos de tiempo, minimizando la separación de medidas entre las horas más conflictivas en las ciudades, como son de 8:00 a 16:00 horas y maximizando la separación en horas con mejor fluidez de tráfico, de 16:00 a 22:00 horas; y maximizando aún más para el horario nocturno de 22:00 a 8:00 horas.
- Hace poco Libelium ha incorporado nuevos sensores de parking donde vienen todos los módulos que se usan en este TFG integrados en una placa, haciendo su tamaño más pequeño. A la hora de incorporarlo a la vía pública, se puede montar sobre ella sin tener que perforar la calzada. Además garantiza un consumo menor de energía al utilizado en este TFG.

Figura 7.52: Nuevo sensor Parking

Fuente: http://www.libelium.com/products/smart-parking/

8. BIBLIOGRAFÍA

https://www.aprendiendoarduino.com/tag/backend-sigfox/

https://backend.sigfox.com/

http://www.libelium.com/

https://build.sigfox.com/steps/sigfox

https://www.sigfox.com/

https://www.cooking-hacks.com

9. ANEXOS

9.1 Anexo 1: Código implementado

```
#include <WaspSigfox.h>
#include <WaspSensorParking.h>
uint8_t socket = SOCKET0;              //variable para el socket donde está el modulo
int temperature;
uint8_t error;
uint8_t data[1];
uint8_t size;
boolean status; //variable para el estado del sensor del parking ocupado o libre
void setup()
{
  USB.ON();                            //encendemos la placa
  USB.println(F("placa encendida"));
  error = Sigfox.ON(socket);            //  encendemos el módulo
  USB.println(F("encendiendo modulo"));
  // Vemos si está el módulo conectado
  if( error == 0 )
  {
    USB.println(F("Switch ON OK"));
  }
  else
  {
    USB.println(F("Switch ON ERROR"));
  }
  USB.println();
```

```
  SensorParking.loadReference();
  SensorParking.ON();
  USB.println(F("encendiendo sensor parking"));
  SensorParking.calibration();
  USB.println(F("calibracion sensor parking"));
}
void loop()
{
    error = Sigfox.getID();
  SensorParking.readParkingSetReset();
  temperature = SensorParking.readTemperature();
  SensorParking.calculateReference(temperature);
  status = SensorParking.estimateState(); ////// guarda la lectura en status
  if( error == 0 )
  {
    USB.println(F("Get ID OK"));
    USB.print(F("Module ID: "));
    USB.println(Sigfox._id, HEX);

    if(status == PARKING_OCCUPIED){
    USB.println(" OCUPADO ");
          if(error==0){
            data[0]=1;  size=1;              //variables para el envio en ocupado
            error = Sigfox.send(data,size);//comando para enviar

          USB.println(F(" Mensaje enviado "));
          }else{
```

```
        USB.println(F(" Mensaje no enviado "));
       }
  }
   else{
   USB.println(" LIBRE ");
  if(error==0){
    data[0]=0;  size=1;                  //variables para el envio en ocupado
    error = Sigfox.send(data,size);     //comando para enviar

  USB.println(F(" Mensaje enviado "));
  }else{
    USB.println(F(" Mensaje no enviado "));
  }
  }
}
else
{
  USB.println(F("Get ID ERROR"));
}

USB.println(F("------------ENTRANDO EN DEEP SLEEP----------------"));

PWR.deepSleep("00:00:12:00", RTC_OFFSET, RTC_ALM1_MODE2, ALL_OFF);
USB.ON();
USB.println(F("WASPMOTE DESPERTADO, BUENOS DIAS"));
  error = Sigfox.ON(socket);
USB.println(F("modulo despertado"));
```

```
    SensorParking.loadReference();
    SensorParking.ON();
  USB.println(F("sensor despertado"));
    SensorParking.calibration();
  USB.println(F("calibracion sensor parking"));
}
```