Demostrador del Internet de las Cosas con la tecnología IEEE 802.15.4e utilizando la plataforma OpenMote, el sistema operativo OpenWSN y la plataforma theThings.io

Roberto Morago Martínez
Máster Universitario en Ingeniería de Telecomunicación UOC-URL

Jose López Vicario

Junio 2017

FICHA DEL TRABAJO FINAL

Título del trabajo:	*Demostrador del Internet de las Cosas con la tecnologia IEEE 802.15.4e utilizando la plataforma OpenMote, el sistema operativo OpenWSN y la plataforma theThings.io*
Nombre del autor:	*Roberto Morago Martínez*
Nombre del consultor:	*Jose López Vicario*
Fecha de entrega (mm/aaaa):	*06/2017*
Área del Trabajo Final:	*Telemática*
Titulación:	*Máster Universitario en Ingeniería de Telecomunicación*

Resumen del Trabajo:

En los últimos años, el paradigma del Internet de las Cosas (*Internet of Things – IoT*) se ha convertido en uno de los conceptos más importantes dentro del sector de las telecomunicaciones, debido principalmente al actual uso extensivo de Internet en prácticamente cualquier ámbito de nuestra sociedad.

El objetivo principal de este TFM es el desarrollo de un prototipo que sirva de demostrador de este paradigma, en combinación con la plataforma hardware OpenMote y la tecnología OpenWSN. Para ello, se ha creado un sistema capaz de registrar el consumo de corriente en una determinada toma conectada a la red, almacenando estos datos en la plataforma theThings.io, otra de las tecnologías protagonistas en este TFM.

El sistema constará de un bloque sensor, compuesto por una placa OpenMote y un sensor de corriente, el sensor ACS712, encargado de medir el consumo de corriente en la toma a monitorizar. La creación de una red OpenWSN permitirá que esta medida de corriente sea transferida desde este bloque hasta un bloque central de procesamiento, cuya labor será adecuar y procesar estos datos de consumo, y enviarlos a la plataforma theThings.io para su almacenamiento.

Las pruebas de verificación llevadas a cabo sobre el sistema han determinado que su fiabilidad y precisión es suficiente como para considerar que el prototipo cumple con los objetivos marcados al inicio del proyecto, y que por lo tanto la utilidad del demostrador queda probada.

Abstract:

In the last few years, the Internet of Things (IoT) paradigm has become one of the most important concepts in the telecommunications sector, mainly due to the current widespread use of the Internet in almost every area of our society.

The main goal of this Master's Thesis is the development of a prototype that serves as demonstrator of this paradigm, in combination with the OpenMote hardware platform and the OpenWSN technology. To achieve that, a system has been created to record current consumption in a certain socket connected to the network, and store this data in theThings.io platform, another of the technologies involved in this Master's Thesis.

The system will consist of a sensor block, composed of an OpenMote board and a current sensor, the ACS712 sensor, which is responsible of measuring the current consumption in the socket to be monitored. The creation of an OpenWSN network will allow this current measurement to be transferred from this block to a central processing block, whose work will be to adapt and process these consumption data, and send them to theThings.io platforms for storage.

Verification tests carried out on the system have determined that they reliability and precision are sufficient to consider that the prototype meets the goals set at the beginning of the project and therefore the usefulness of the demonstrator is proven.

Palabras clave :

Internet Of Things (IoT), OpenMote, OpenWSN, theThings.io, IEEE 802.15.4e

ÍNDICE DE CONTENIDOS

ÍNDICE DE FIGURAS

1. INTRODUCCIÓN

1.1. CONTEXTO Y JUSTIFICACIÓN DEL TRABAJO

Actualmente, el concepto del Internet de las cosas (IoT- Internet Of Things), es uno de los temas más en auge y a los que se considera con más recorrido en el sector de las telecomunicaciones, con la absoluta presencia de Internet en todos los ámbitos de nuestra sociedad y de nuestro día a día. Por lo tanto, parece bastante apropiado llevar a cabo un trabajo, tanto de investigación y documentación como de implementación, que gire en torno a este concepto en continuo desarrollo y crecimiento.

La posibilidad de que prácticamente cualquier dispositivo pueda conectarse a Internet es una gran oportunidad en el entorno doméstico, ya que permite la monitorización de diferentes aspectos de las vivienda, incluso sin la presencia del usuario en ella, permitiéndole así conocer e incluso gestionar estos aspectos de manera remota. Esta es la idea en la que se basa el prototipo desarrollado con este proyecto, permitir la monitorización del consumo eléctrico en una vivienda, pudiendo acceder a estos datos de consumo desde cualquier lugar y en cualquier momento, simplemente disponiendo de una conexión a Internet.

Evidentemente ya existen productos comerciales en el mercado que permiten el registro de este consumo, pero el prototipo que aquí se presenta pretende ser diferente en dos aspectos fundamentales. En primer lugar, se tratara de un sistema totalmente de código abierto, lo que permite total libertad e infinitas posibilidades de edición y modificación, para adaptar el sistema a las necesidades del usuario o para seguir mejorándolo o evolucionándolo. Y por otro, el empleo de la plataforma theThings.io permite el almacenamiento de los datos en la nube, con todas las posibilidades que eso permite en cuanto a aspectos de compartición y acceso a los datos desde cualquier dispositivo con acceso a internet.

Además, hoy en día hay cada vez existe una mayor preocupación y concienciación en torno a la sostenibilidad y ahorro energético, con lo cual el desarrollo de un sistema que permita en cierto modo tener un mayor control sobre el consumo de energía eléctrica pude resultar muy interesante en relación con estas preocupaciones medioambientales. Por ello se pretende implementar un sistema que suponga un primer paso en el empleo del IoT con este fin de contribuir a la sostenibilidad y al desarrollo sostenible en general. La idea es que la monitorización continua del consumo eléctrico en la vivienda contribuya a reducir al máximo el gasto no necesario de energía, que de otra manera puede pasar desapercibido, pero que con el empleo de esta tecnología podría ser identificado y analizado, permitiendo por lo tanto que se evite ese malgasto de dicha energía.

1.2. OBJETIVOS DEL TRABAJO

El objetivo final de este trabajo fin de Máster es el desarrollo e implementación de un prototipo que permita demostrar el IoT con la tecnologia IEEE 802.15.4e utilizando para ello la plataforma OpenMote como hardware, y OpenWSN y la plataforma theThings.io como elementos software. La función del sistema será la medición y monitorización del consumo eléctrico de una toma de

corriente de una vivienda. Para conseguir este objetivo global, se perseguirán los siguientes objetivos individuales:

- Realización de trabajo de documentación sobre el Internet de las cosas, que permita conocer el concepto en sí, sus características generales, y su estado de desarrollo actual.
- Realización de trabajo de documentación sobre los protocolos IEEE 802.15.4 e IEEE802.15.4e, así como de otros estándares relacionados con el IoT que permita adquirir un conocimiento básico sobre todos ellos y poder comparar sus características y prestaciones.
- Realización de trabajo de documentación sobre la plataforma OpenMote, Open WSM y plataforma theThings.io, a través del cual se consiga entender su funcionamiento y conocer lo suficiente estas tecnologías para poder aplicarlas en el TFM.
- Diseño y desarrollo de sistema prototipo hardware que represente la filosofía del IoT, usando para ello la plataforma OpenMote y un sensor de corriente conectado a esta plataforma que permita medir la corriente consumida en una determinada toma de corriente.
- Desarrollar software empleando plataforma OpenWSN que permita recoger los datos de un sensor de corriente eléctrica para su posterior procesamiento, lo que permitirá llevar a cabo un control sobre el consumo en la toma de corriente a monitorizar.
- Desarrollar software que permita almacenar los datos de consumo de corriente registrados en una toma de corriente en la plataforma theThings.io, donde podrán ser consultados.
- Desarrollo de aplicación software en lenguaje python, que ejecutada en un ordenador portátil, sea el núcleo de control de todo el sistema, permita al usuario arrancarlo y le muestre información básica sobre el estado del proceso de monitorización.
- Desarrollo de memoria que incluya todo el trabajo de documentación llevado a cabo y la descripción detallada del sistema desarrollado y su funcionamiento

1.3. ENFOQUE Y MÉTODO SEGUIDO

El desarrollo de este TFM puede dividirse en dos partes bien diferenciadas, y que por lo tanto han sido enfocadas de manera distinta. Por un lado está el trabajo de documentación llevado a cabo sobre los diferentes elementos técnicos y tecnológicos en los que se basa el trabajo, y por otro el trabajo de desarrollo del prototipo objetivo de este proyecto.

Respecto al trabajo de documentación, el método de trabajo ha consistido en realizar en primer lugar un guion sobre todos los temas técnicos a abordar, y que fuesen relevantes para el objetivo que se pretende en este TFM. No hay que perder de vista que uno de los principales objetivos de este trabajo de documentación es adquirir los conocimientos básicos necesarios para luego poder llevar a cabo el desarrollo del prototipo deseado, por lo que debía marcarse muy claramente cuáles eran los aspectos necesarios para este desarrollo, previamente al inicio de las tareas de investigación.

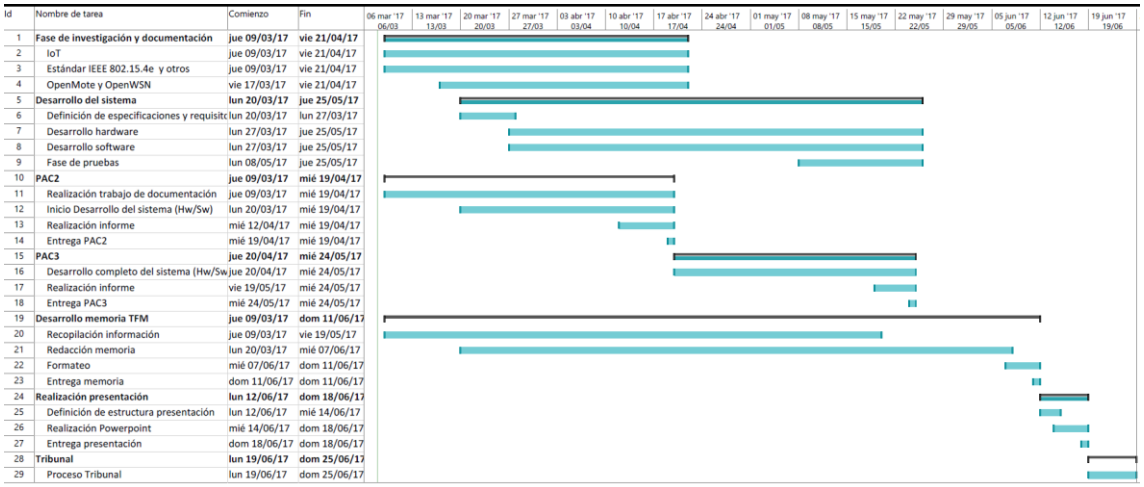
Una vez elaborado este guion previo, se ha realizado un trabajo de investigación y de recopilación de información de los diferentes temas, siempre teniendo en mente los puntos necesarios para el trabajo de desarrollo. El último paso consistió en recoger los aspectos fundamentales de la información recopilada y presentarlos en esta memoria.

En lo que respecta al trabajo de desarrollo del prototipo, el método seguido ha consistido en dividir el diseño completo en varias partes o subsistemas, abordando el desarrollo de cada uno de ellos por individual, y en un paso final proceder a la integración de todos para dar lugar al prototipo global completo.

Así, en primer lugar se ha llevado a cabo el diseño y desarrollo de la parte sensor del sistema, trabajando en paralelo en la comunicación con la plataforma theThings.io. Con estos dos aspectos ya implementados y probados, se ha procedido a trabajar en la configuración de la red OpenWSN en la que se basa el funcionamiento del prototipo, pasando a continuación a integrar todos estos elementos en un sistema único que los contenga y que permita las prestaciones deseadas.

1.4. PLANIFICACIÓN DEL TRABAJO

A continuación se recoge el diagrama de Gantt que se realizó al principio de este TFM estableciendo su planificación:



Id	Nombre de tarea	Comienzo	Fin
1	**Fase de investigación y documentación**	**jue 09/03/17**	**vie 21/04/17**
2	IoT	jue 09/03/17	vie 21/04/17
3	Estándar IEEE 802.15.4e y otros	jue 09/03/17	vie 21/04/17
4	OpenMote y OpenWSN	vie 17/03/17	vie 21/04/17
5	**Desarrollo del sistema**	**lun 20/03/17**	**jue 25/05/17**
6	Definición de especificaciones y requisitc	lun 20/03/17	lun 27/03/17
7	Desarrollo hardware	lun 27/03/17	jue 25/05/17
8	Desarrollo software	lun 27/03/17	jue 25/05/17
9	Fase de pruebas	lun 08/05/17	jue 25/05/17
10	**PAC2**	**jue 09/03/17**	**mié 19/04/17**
11	Realización trabajo de documentación	jue 09/03/17	mié 19/04/17
12	Inicio Desarrollo del sistema (Hw/Sw)	lun 20/03/17	mié 19/04/17
13	Realización informe	mié 12/04/17	mié 19/04/17
14	Entrega PAC2	mié 19/04/17	mié 19/04/17
15	**PAC3**	**jue 20/04/17**	**mié 24/05/17**
16	Desarrollo completo del sistema (Hw/Sw	jue 20/04/17	mié 24/05/17
17	Realización informe	vie 19/05/17	mié 24/05/17
18	Entrega PAC3	mié 24/05/17	mié 24/05/17
19	**Desarrollo memoria TFM**	**jue 09/03/17**	**dom 11/06/17**
20	Recopilación información	jue 09/03/17	vie 19/05/17
21	Redacción memoria	lun 20/03/17	mié 07/06/17
22	Formateo	mié 07/06/17	dom 11/06/17
23	Entrega memoria	dom 11/06/17	dom 11/06/17
24	**Realización presentación**	**lun 12/06/17**	**dom 18/06/17**
25	Definición de estructura presentación	lun 12/06/17	mié 14/06/17
26	Realización Powerpoint	mié 14/06/17	dom 18/06/17
27	Entrega presentación	dom 18/06/17	dom 18/06/17
28	**Tribunal**	**lun 19/06/17**	**dom 25/06/17**
29	Proceso Tribunal	lun 19/06/17	dom 25/06/17

Aunque en el capítulo de conclusiones se analizará también este aspecto, este apartado también es apropiado para comentar que esta planificación inicial se ha visto modificada con el transcurso del desarrollo de proyecto, retrasándose la entrega final aproximadamente dos semanas con respecto a la fecha estipulada inicialmente.

Este retraso ha venido motivado principalmente por la aparición de un funcionamiento defectuoso en una placa OpenMote que había sido facilitada para la realización de este TFM. Esto ha obligado a cambiar los planes existentes a la hora de integrar todas las placas OpenMote y formar la red OpenWSN, ya que sólo se disponía de un dispositivo. Así, se ha aprovechado ese periodo de espera por un nuevo hardware para probar el resto de elementos del sistema y así

no parar totalmente la evolución del trabajo, pero a pesar de ello se ha acumulado un retraso inevitable que ha provocado a su vez el retraso de la entrega final.

1.5. BREVE RESUMEN DE PRODUCTOS OBTENIDOS

Los productos que se han obtenido a través de la realización de este TFM son los siguientes:

- Prototipo básico hardware que permite tomar medidas de corriente de una toma determinada, y transmitir los datos a través de una red OpenWSN compuesta por hardware OpenMote.

- Firmware OpenWSN modificado para incluir aplicación capaz de muestrear la señal de tensión procedente de un sensor, mediante la utilización del convertidor analógico-digital de la placa CC2538, y su posterior transmisión a un nodo central mediante el protocolo CoAP.

- Aplicación python, que instalada en un ordenador portátil recoge los datos medidos por un sensor, los procesa y los transmite a la plataforma theThings.io.

- Plataforma theThings.io configurada para recoger los datos de medición enviados desde un nodo central, con la posibilidad de emitir avisos al usuario ante ciertas circunstancias.

1.6. BREVE DESCRIPCIÓN DEL RESTO DE CAPÍTULOS DE LA MEMORIA

La presente memoria está estructurada en los siguientes capítulos:

- **Capítulo 2. Marco del trabajo**. Este capítulo sirve para hacer una introducción y presentación de los diferentes aspectos tecnológicos que dan origen a este trabajo, destacando principalmente el Internet de las Cosas y el estándar IEEE 802.15.4. El objetivo es describir los elementos básicos fundamentales que permitan una mejor comprensión del prototipo desarrollado y sus fundamentos técnicos.

- **Capítulo 3. Descripción del sistema desarrollado**. En este capítulo se llevará a cabo una descripción minuciosa del prototipo desarrollado que es objetivo de este proyecto. Se comenzará con una descripción general de su estructura y funcionamiento, pasando a continuación a describir en detalle tanto la parte hardware como software de este

prototipo, definiendo los diferentes elementos que las componen y sus características y prestaciones.

- **Capítulo 4. Pruebas y verificación del sistema**. En este capítulo se recogen las diferentes pruebas y procesos de verificación que se han realizado para comprobar el funcionamiento y fiabilidad del sistema desarrollado. En un principio, se detallarán los procesos seguidos para probar cada elemento del sistema por separado, finalizando con una prueba sobre el sistema global que pretende ser muestra de los objetivos alcanzados en el trabajo.

- **Capítulo 5. Conclusiones y líneas futuras**. Este último capítulo sirve para presentar las diferentes conclusiones que se han extraído de la realización de este TFM y del prototipo asociado a su desarrollo, así como posibles líneas futuras que se abren tras este trabajo.

2. MARCO DEL TRABAJO

2.1. INTERNET DE LAS COSAS (IoT)

2.1.1. INTRODUCCIÓN

Aunque no existe una definición única o universal, se puede describir el internet de las cosas (IOT - Internet Of Things) como la extensión de las capacidades de conexión y procesamiento de cualquier objeto físico, incluyendo objetos cotidianos, de manera que cualquiera de ellos pueda convertirse en un terminal conectado a internet. Este término de "Internet of Things" fue utilizado por primera vez en 1999 por el británico Kevin Ashton, en relación a una red de objetos conectados mediante tecnología RFID (Radio-Frequency Identification), y en el contexto de la gestión de la cadena de abastecimiento de una compañía, permitiendo el recuento y registro de la mercancía sin la necesidad de intervención humana [1].

Pero antes de la primera aparición de este concepto hace relativamente poco tiempo, la idea de combinar redes y ordenadores para controlar y monitorizar dispositivos no es en absoluto nueva. Así, en la década de los 90 proliferó la comunicación M2M (Machine-To-Machine) y se hicieron de uso generalizado las soluciones industriales para la monitorización y la operación de dispositivos y equipamiento, aunque muchas de ellas estaban basadas en protocolos propietarios y no en redes IP ni protocolos de internet [2].

La utilización de IP para conectar equipos que no fueran ordenadores tampoco es una idea nueva, y ya en 1990 se presentó el primer objeto cotidiano conectado a internet, una tostadora que podía ser apagada y encendida a través de la red. Durante los siguientes años, la aparición de otras iniciativas de este tipo contribuyó al comienzo de sólidos estudios y ensayos en este campo, imprescindibles para el desarrollo del internet de las cosas de hoy en día. Las razones que explican el reciente auge del concepto del IoT son varias [2], [3]:

- **Conectividad ubicua:**

La conectividad de bajo coste y alta velocidad, principalmente a través de tecnologías inalámbricas hace que prácticamente cualquier objeto sea candidato para ser conectado a internet.

- **Abaratamiento de equipos:**

El abaratamiento de los equipos, principalmente de procesadores y memorias ha provocado que sea más abordable la fabricación de objetos inteligentes.

- **Miniaturización de dispositivos**

IoT emplea diferentes tecnologías para la conexión de muy diversos objetos a internet, pero un aspecto común a todas estas posibilidades es que el tamaño y el coste económico asociado a los diferentes componentes electrónicos necesarios para esta conexión y los procesos de

comunicación es un aspecto absolutamente crítico para el desarrollo del IoT en los diferentes campos de aplicación.

Los avances en la fabricación de componentes de reducido tamaño han permitido la incorporación de capacidad de comunicación e interconexión a objetos de también pequeño tamaño, destacando principalmente sensores y actuadores, que son elementos de gran importancia en la tecnología IoT.

En este aspecto, el desarrollo de la industria de fabricación de semiconductores ha sido espectacular, siguiendo el cumplimiento de la ley de Law que aseguraba que la densidad de transistores se doblaría cada dos años. Esto ha permitido que el número de transistores por chip haya crecido exponencialmente a lo largo de los años, posibilitando esa capacidad de computación a objetos de minúsculo tamaño, fundamental en algunas aplicaciones del IoT. Y así, al mismo tiempo que el tamaño de los chips ha ido haciéndose cada vez más reducido con el avance del tiempo, coste de los sensores también se ha hecho más económico, lo que contribuye a que un mayor número de estos elementos puedan ser incorporados en diferentes aplicaciones, favoreciendo así el despliegue del IoT.

- **Análisis de datos**

El aumento de la capacidad de procesamiento de los ordenadores, junto con el desarrollo de la capacidad de almacenamiento y los servicios en la nube, permiten la agregación y análisis de grandes cantidades de datos que suponen importantes fuentes de información y conocimiento.

- **Cloud computing:**

El rápido crecimiento de los servicios en la nube ha permitido que pequeños dispositivos distribuidos interactúen con potentes funcionalidades analíticas y de control back-end.

- **RFID (Radio Frequency Identificacion)**

La tecnología de identificación por radio frecuencia (RFID) se puede considerar de trascendental importancia en el desarrollo del IoT, ya que representa una de las primeras utilizaciones de la filosofía del IoT en aplicaciones industriales, siendo empleada para la monitorización y el registro de mercancías en el sector de producción y logístico, donde resulta de gran utilidad a la hora de identificación de mercancías gracias a su capacidad de recoger y procesar información constantemente de su entorno.

El rango de frecuencias de funcionamiento de la tecnología RFID se extiende desde las bajas frecuencias, 125 KHz, hasta las frecuencias super altas (SHF-Super High Frequency), 5,8 GHz, y su funcionamiento se va a basar en 3 elementos principales:

- Un chip o etiqueta (tag), que está unida a un objeto al cual identifica unívocamente y del que mantiene información, transfiriendo los datos a un lector de manera inalámbrica.
- Lector, que puede leer los datos de una etiqueta o modificar o añadir nueva información a la que contiene ésta.

- Antena, que permite la transmisión de datos hacia o desde un lector, ya sea para leer los datos de una etiqueta o para añadir a estos datos más información respectivamente

- **IPv6 (Internet protocol versión 6)**

A la hora de conectar dispositivos a internet resulta un aspecto fundamental la identificación de estos y la manera de asignarles una dirección que les represente y que permita que se comuniquen con el resto de dispositivos. Con el banco de direcciones IPv4 prácticamente agotado, IPv6 es el protocolo elegido para reemplazarlo y permitir nuevos direccionamientos.

Con IPv6 hay aproximadamente 3,4 x1038 direcciones únicas que poder asignar a dispositivos, con lo que se convierte en un elemento que puede resultar crucial a la hora del desarrollo del IoT, permitiendo la identificación unívoca de millones de objetos y dispositivos que a los que se pretende mantener conectados y comunicados, eliminando la necesidad de la traducción de direcciones de red (NAT- Network Address Translation).

2.1.2.ARQUITECTURA IoT

La arquitectura de un sistema IoT puede descomponerse en varias capas: Capa de información o sensores, capa de red, capa de gestión del servicio y capa de aplicaciones [3].

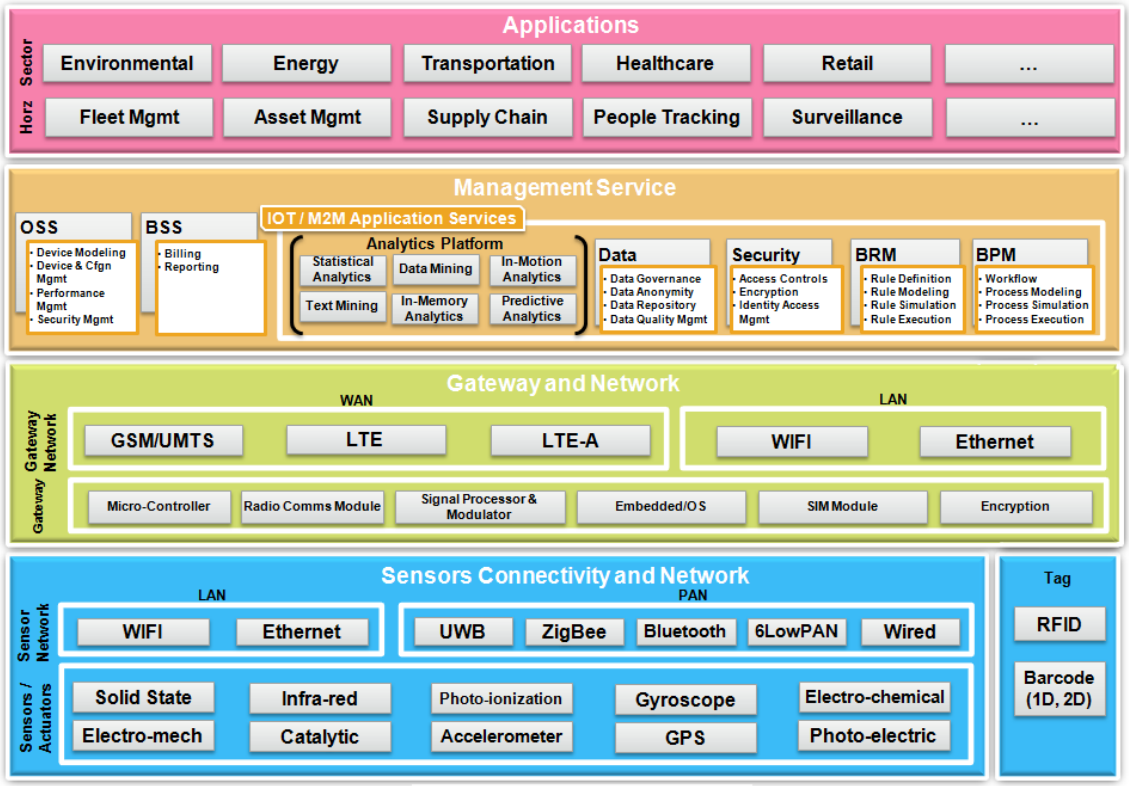


Figura 1. Arquitectura de los sistemas IoT [3]

Capa de información:

Esta capa, también denominada capa de sensores, es la primera capa de la arquitectura de un sistema IoT, y está compuesta por objetos físicos integrados con sensores. Así, esta capa tiene la misión de identificar los objetos y captar información del entorno. Como ya se comentó en un apartado anterior, la miniaturización de los dispositivos que ha permitido el avance en los procesos de fabricación, posibilita la creación de potentes sensores de pequeño tamaño, con lo que su integración con los objetos es mucho más sencilla.

Estos sensores miden una magnitud física y la convierten en una señal que pueda ser procesada por otro dispositivo, siendo estas magnitudes muy diversas, como por ejemplo temperatura, nivel de ruido, porcentaje de algún componente en el aire, movimiento, etc…y pueden ser clasificados de una manera general según su entorno o sector de utilización, como por ejemplo sensores de vehículos, sensores domésticos, sensores para el cuerpo humano…

En algunas ocasiones, además de esa capacidad de medición de magnitudes, los sensores precisan de disponer de otras características adicionales. Así, en ciertos casos se necesita que el sensor disponga de un cierto nivel de memoria, que le permita poder almacenar un determinado número de medidas anteriores. Otras características de los sensores, especialmente importantes en los sistemas IoT, son las relacionadas con la conectividad. Así, la mayoría de los sensores requerirán conectividad con los agregadores de sensores o puertas de enlace. Esta conexión puede llevarse a cabo de diferentes maneras, por ejemplo a través de una red de área local (LAN), ya sea basada en Ethernet o inalámbrica, o bien mediante una red de área personal (PAN). En este último caso, tendríamos como ejemplos los protocolos Bluetooth o ZigBee.

En el caso de sensores que no se comuniquen con agregadores de sensores, sino con servidores o aplicaciones, esta conexión se puede efectuar a través de redes de área amplia (WAN) mediante GPRS, GSM o LTE. Por último, por su especial importancia en los sistemas IoT, también hay que mencionar los sensores de baja potencia y bajas tasas de transmisión, que normalmente se agrupan formando redes denominadas redes de sensores inalámbricas (WSN).

Capa de red

Esta capa tendrá la misión de la transmisión, el enrutamiento y el control entre las otras dos capas de información y aplicación. La gran cantidad de datos generada por los sensores del sistema precisará de una infraestructura de red robusta y de alto rendimiento como medio de transporte, ya sea inalámbrica o cableada. La gran diversidad de aplicaciones y servicios IoT provoca que sea necesaria la colaboración entre múltiples redes de diferentes tecnologías y protocolos de acceso. Estas redes pueden estar en forma de modelos públicos, privados o híbridos, y tendrán que satisfacer los requisitos de latencia, ancho de banda y seguridad necesarios para las aplicaciones IoT.

Capa de gestión de servicio

Esta capa tiene la tarea de hacer posible el procesado de la información, gracias a elementos como el análisis, controles de seguridad y la gestión de los dispositivos y datos. Así, en lo referente al análisis, se emplean diferentes herramientas que permiten extraer la información

relevante de la gran cantidad de datos totales facilitados por los sensores, permitiendo así un procesado mucho más rápido. Una de estas herramientas sería el análisis en memoria interna, donde grandes cantidades de datos son almacenados temporalmente en memoria RAM en lugar de ser almacenados en discos físicos. Esto permite aumentar la velocidad de toma de decisiones gracias a que se reduce de manera significativa el tiempo de consulta de los datos. Otra herramienta de análisis es el análisis por streaming, donde los datos son analizados en tiempo real permitiendo la toma de decisiones en un tiempo muy bajo. Respecto al análisis de datos también es importante mencionar que puede llevarse a cabo también en parte en otras de las capas de la arquitectura, por ejemplo en la capa de información, lo que permite reducir la carga de la capa de red.

Otra de las características de la capa de gestión de servicio es la gestión de los datos, que permite manejar el flujo de información de manera que se puede acceder a ésta y tener control sobre ella. Así, se puede evitar que la capa superior reciba información innecesaria, y además, mediante técnicas de filtrado, como la anonimización de los datos, integración de datos y sincronización de datos, se pueden ocultar los detalles de la información, facilitando a las aplicaciones sólo la información esencial que precisan.

Por último también hay que destacar la importancia de la seguridad, que debe asegurarse desde la capa de información hasta la capa de gestión de servicio, asegurando así a su vez la integridad de los datos a lo largo de todo el sistema IoT, lo que hará posible la toma de decisiones correctas y fiables. Además, asegurar la seguridad en el sistema evitará el acceso a los datos de personal no autorizado que pueda provocar riesgos en el sistema.

Capa de aplicación

Hay varias y diversas aplicaciones que pueden aprovechar el uso de un sistema IoT, tanto aplicaciones específicas de un sector como algunas que pueden ser empleadas en múltiples sectores.

2.1.3.MODELOS DE COMUNICACIÓN IoT

Desde el punto de vista en el que se comunican y conectan los dispositivos IoT, se puede hablar de 4 modelos diferentes de comunicación [2]:

- **Comunicación Dispositivo-Dispositivo**

En este modelo dos o más dispositivos se conectan y comunican directamente entre ellos, y aunque para ello pueden emplear diferentes redes, es típico el empleo de protocolos como Bluetooth o ZigBee.

Este tipo de modelo de comunicación es habitualmente utilizado en sistemas de automatización domésticos, donde los dispositivos se comunican a través de paquetes de datos de pequeño tamaño, necesitando tasas de transmisión relativamente pequeñas.

Otra de las características de este modelo de comunicación es que los dispositivos emplean generalmente formatos de datos y protocolos específicos en lugar de desarrollos de carácter abierto, lo que provoca que en muchas ocasiones dispositivos de diferente familia no sean compatibles entre sí, y por lo tanto el usuario vea limitada su capacidad de elección a los dispositivos de una misma familia compatibles y que pueden comunicarse entre sí.

- **Comunicación Dispositivo-Cloud**

En este modelo de comunicación el dispositivo IoT se conecta directamente con un servició en la nube, como un proveedor de servicios de aplicaciones, para el intercambio de datos y el control del tráfico de mensajes. Un ejemplo de este modelo de comunicaciones IoT sería el servicio Samsung Smart TV, donde la televisión emplea la conexión a internet para transmitir a Samsung información sobre lo que el usuario ve y para habilitar características de reconocimiento de voz, añadiendo así al dispositivo características adicionales que aportan valor añadido al usuario.

Similar a lo que ocurría con el modelo de comunicación Dispositivo-Dispositivo, en este modelo pueden aparecer problemas cuando se trata de integrar dispositivos de diferentes fabricantes, ocurriendo frecuentemente que el dispositivo y el servicio en la nube son del mismo proveedor, y así, al emplearse protocolos propietarios, el usuario de un dispositivo puede verse obligado a emplear un servicio en la nube concreto, no pudiendo utilizar proveedores de servicios alternativos.

- **Comunicación Dispositivo-Gateway**

En este caso, al igual que en el modelo Dispositivo-Cloud, existe una conexión entre un dispositivo IoT y un servicio en la nube, pero ahora esta comunicación no es directa, sino a través de una puerta de enlace, que proporciona algunas funcionalidades como las relacionadas con la seguridad o la interpretación de protocolos y datos.

Un ejemplo de este modelo viene a menudo representado por un smartphone que hace el papel de la puerta de enlace, ejecutando una aplicación que se comunica con un dispositivo IoT y entrega datos a la nube. El dispositivo no tiene la capacidad de comunicarse directamente con el servicio en la nube, por lo que emplea la aplicación del Smartphone como intermediario. Otro ejemplo de este modelo serían los dispositivos hub empleados en aplicaciones de automatización del hogar, que sirven tanto de puerta de enlace entre dispositivos IoT y un servicio en la nube, como de elemento de conexión entre estos dispositivos IoT mismos.

- **Comunicación Backend-Compartición de datos**

Este modelo de comunicaciones puede ser interpretado como una extensión del modelo Dispositivo-Cloud, que permite a los usuarios exportar y analizar conjuntos de datos de un servicio en la nube en combinación con datos de otras fuentes. Mediante este modelo de comunicación, los flujos de datos de dispositivos IoT individuales pueden ser agregados y analizados.

2.1.4.APLICACIONES IOT

Existen un gran número de sectores y ubicaciones donde los sistemas IoT pueden ser empleados con gran utilidad. A continuación se describen algunos ejemplos de estos casos de uso.

Cuerpo humano:

En este grupo se incluirían dos tipos principales de dispositivos IoT. Por un lado aquellos que se llevan sobre el cuerpo humano, y por otro aquéllos alojados dentro de él. Entre las principales utilidades de estos dispositivos están la monitorización y registro de la actividad física, control de parámetros relativos a la salud y enfermedades, o aumentar la productividad.

Un ejemplo de un sistema de este tipo sería una aplicación destinada a la monitorización de un paciente, empleando sensores para monitorizar parámetros vitales como por ejemplo el ritmo cardiaco, u otros como la temperatura o el nivel de azúcar en sangre. El sistema puede utilizarse para detectar posibles anomalías en el estado del paciente y a partir de ahí incluso predecir posibles problemas de salud que el paciente pueda experimentar. Otro ejemplo podrían ser las pastillas inteligentes, que serían sensores ingeribles que una vez dentro del cuerpo pueden tomar medidas fisiológicas.

Hogar:

En este caso se trataría de dispositivos IoT destinados al control de diferentes elementos de un edificio y a su automatización, como por ejemplo el encendido o apagado de la luz, o la apertura y cierre de puertas y ventanas, así como la monitorización y gestión de diferentes elementos de segurida. Uno de los ejemplos más comunes sería un sistema destinado a la gestión energética de una vivienda, cuyo objetivo sería optimizar el consumo y producción de la energía de la misma, incluyendo herramientas que analizan el uso de la energía y sensores que responden a precios variables de la electricidad.

Vehículos:

Dentro de este grupo se podrían encontrar dispositivos IoT destinados principalmente a la monitorización de parámetros para la ayuda al diagnóstico y mantenimiento de cualquier tipo de vehículo, desde coches a barcos, pasando por camiones y trenes.

Industria:

Dentro de un ambiente industrial de fabricación, los dispositivos IoT se pueden emplear para la monitorización de los procesos de fabricación y su optimización, junto con la gestión y mejora del uso del equipamiento y el control de materiales. Al igual que en las viviendas, otra posible uso de los sistemas IoT sería la gestión energética de las instalaciones.

En el caso de la industria logística, el objetivo que se busca con el uso de los sistemas IoT sería la optimización de los procesos de reparto, para lo cual el empleo de cámaras que facilite

información sobre las condiciones de congestión de las carreteras, y de sensores que proporcionen datos sobre las condiciones meteorológicas, resulta muy útil para hacer los ajustes necesarios sobre las rutas planificadas y así adaptarlas a las condiciones cambiantes.

Espacios comerciales:

En este caso los dispositivos IoT pueden emplearse por ejemplo en tareas de gestión y optimización de inventario, o de automatización de los procesos de compra y pago, haciendo más cómoda la experiencia del cliente y facilitando el trabajo del vendedor.

Ciudades inteligentes:

Una de las principales iniciativas IoT son las ciudades inteligentes o smart cities. En conjunto, se trata de diferentes acciones y estrategias destinadas a automatizar y aumentar la eficiencia de la gestión de las infraestructuras urbanas, con el objetivo de mejorar los servicios ofrecidos, facilitar la interacción de los ciudadanos con dichas infraestructuras, y la reducción de los gastos.

Entre los diferentes ejemplos de estas iniciativas se pueden destacar la mejora del tráfico, medidores inteligentes que permitan controlar los niveles de contaminación en aire y agua, mejora y agilización de los servicios de la administración que proporcione una mejor atención al ciudadano, o la optimización de la gestión energética.

Así, por ejemplo los sistemas de control de tráfico distribuidos registran la localización de los vehículos en tiempo real, lo que facilita la posibilidad de tomar las medidas necesarias para afrontar las condiciones del tráfico en un momento determinado, y así poder dotarle de mayor fluidez. Estos sistemas pueden emplearse también en caso de que se produzca una emergencia, permitiendo establecer caminos seguros y rápidos para vehículos prioritarios como ambulancias, camiones de bomberos o coches de policía.

2.1.5.RETOS PARA LOS SISTEMAS IoT

- **Soluciones IoT específicas**

Una gran cantidad de las soluciones IoT desarrolladas e implantadas hoy en día están dirigidas específicamente al dominio de la aplicación, de manera que estas soluciones carecen de compatibilidad o interoperabilidad entre sistemas y tecnologías, o de algún tipo de estandarización o enfoque en el mundo global de los sistemas IoT. Para que sea posible un mayor crecimiento o desarrollo de la filosofía y sistemas IoT es necesaria la estandarización de los sistemas, y que las soluciones sean interoperables entre ellas a varios niveles y a través de diferentes plataformas, facilitando la integración y la escalabilidad.

En este sentido, en los últimos tiempos diferentes organizaciones están tratando de formar alianzas abiertas para trabajar con miras al desarrollo de soluciones interoperables, tratando de

adoptar estándares que permitan esa compatibilidad y eviten estos problemas de las soluciones específicas.

- **Privacidad y seguridad**

Uno de los principales retos que tiene por delante el IoT para convencer a los usuarios de su total adopción es asegurarles la seguridad y privacidad de sus datos. En un sistema IoT hay una gran cantidad de datos que continuamente están siendo captados y comunicados entre dispositivos de manera automática y sin conocimiento ni intervención del usuario, y al ser algunos de estos datos personales, como la localización o cualquier preferencia del usuario, es aún más delicado y crítico la protección de su privacidad. Por tanto en un sistema IoT es fundamental que se decida y se conozca claramente quién controla los datos y por cuanto tiempo, así como disponer de protocolos de autorización que eviten un mal uso de los datos. Y por supuesto, estas soluciones relativas a la privacidad no se restringen únicamente al aspecto técnico, sino que es necesaria una regulación desde el punto de vista del mercado y las consideraciones éticas empresariales.

Algo similar ocurre respecto a los aspectos de seguridad. A la hora de que las implementaciones IoT puedan tener una amplia aceptación y recorrido, es imprescindible que los usuarios no tengan ningún tipo de duda o incertidumbre respecto a la seguridad de los datos y del sistema en sí. A medida que aumenta el número de dispositivos conectados a internet, también aumentan las oportunidades de que puedan ser explotadas posibles vulnerabilidades en la seguridad del sistema. Así, un dispositivo cuyo desarrollo sea pobre en términos de seguridad puede suponer un punto de entrada al sistema para ataques externos que puedan hacer funcionar el dispositivo incorrectamente o reprogramarlo para un funcionamiento diferente. También un dispositivo con falta de seguridad puede facilitar el robo de datos al no proteger correctamente el flujo de éstos en los diferentes intercambios que se llevan a cabo en el proceso de comunicación. Además, hay que tener en cuenta que un dispositivo pobremente diseñado en aspectos de seguridad no sólo resulta peligroso localmente, sino que puede provocar riesgos para toda la red, lo que demuestra de manera aún más clara la importancia de asegurar que un dispositivo cuenta con las medidas de seguridad adecuadas.

- **Limitaciones en la capacidad de la red**

Con el aumento del número de dispositivos que están constantemente captando e intercambiando información entre ellos, aumentan las exigencias sobre la infraestructura de red necesaria para soportar tales demandas. Muchas aplicaciones actuales requieren de frecuentes ráfagas de pequeños bloques de datos para tareas de actualización y sincronización, y la rapidez con la que se producen estas ráfagas puede influir en la latencia y ancho de banda de la red.

Este problema relativo a la capacidad limitada de la red ha llevado a muchos operadores globales a desarrollar iniciativas que aprovechan tecnologías en el espectro no licenciado y que aumentan el uso del WiFi para descargar el tráfico de datos.

2.1.6.ORGANIZACIONES Y ALIANZAS IoT

A continuación se recoge una lista con algunas de las organizaciones y consorcios más importantes relacionados con el desarrollo, promoción e implementación del IoT.

- **The Alliance for Internet of Things (AIOTI)**

Alianza creada por la comisión europea para desarrollar el diálogo e interacción entre los diferentes actores relacionados con el IoT en Europa. Su objetivo es la creación de un entorno dinámico del IoT en Europa que promueva el potencial del IoT, así como asistir a la comisión europea en la preparación de las futuras investigaciones acerca del IoT y de las políticas de innovación y estandarización.

https://ec.europa.eu/digital-single-market/alliance-internet-things-innovation-aioti

- **AllSeen Alliance**

Se trata de una organización sin fines lucrativos dedicada a conseguir y permitir la interoperabilidad entre dispositivos, productos y servicios IoT, empleando para ello un framework opensource creado por el propio consorcio, y que se denomina AllJoyn. Esta organización cuenta con más de 200 participantes, incluyendo fabricantes de componentes electrónicos, fabricantes de electrodomésticos, empresas tecnológicas, proveedores de servicios o desarrolladores de software entre otros.

https://allseenalliance.org/

- **European Telecommunications Standards Institute (ETSI)**

La iniciativa Connecting Things de la ETSI tiene como objetivo el desarrollo de estándares para la gestión, seguridad, transporte y procesamiento de los datos relacionados con los sistemas IoT. El objetivo final de la elaboración de estos estándares es posibilitar y facilitar la creación de aplicaciones y soluciones que garanticen la interoperabilidad y permitan al mercado IoT alcanzar su máximo potencial.

http://www.etsi.org/technologies-clusters/clusters/connecting-things

- **Institute of Electrical and Electronics Engineers (IEEE)**

El IEEE posee una iniciativa específica dedicada al IoT, y que sirve como centro de intercambio de información para la comunidad técnica que participa en la investigación, implementación, aplicación y uso de las tecnologías ioT.

http://iot.ieee.org/

- **European Reseach Cluster on the Internet of Things (IERC)**

Los objetivos del IERC es establecer una plataforma de cooperación para las actividades de IoT en Europa y definir una estrategia internacional para la cooperación en el área de la investigación y la innovación del IoT.

http://www.internet-of-things-research.eu/

- **Internet of Things Consortium (IoTC)**

Se trata de una organización compuesta por más de 60 empresas, cuyo objetivo es contribuir al crecimiento del mercado del IoT y el desarrollo de modelos de negocio sostenibles. El IoTC trata de transmitir el valor del IoT a compañías de diferentes sectores relacionados con el IoT, como empresas tecnológicas, distribuidores, compañías de seguros, etc…

http://iofthings.org/#home

2.2. ESTÁNDAR IEEE 802.15.4

2.2.1.INTRODUCCIÓN

El estándar IEEE 802.15.4 define la capa física y la subcapa de control de acceso al medio (MAC) para redes inalámbricas con tasas de transmisión bajas y requerimientos de muy bajo consumo. Son las denominadas redes LR-WPAN (Low-Rate Wireless Personal Area Network), cuyas características generales son facilidad de instalación, transferencia de datos fiable, corto alcance, y reducidos coste y consumo [4].

En una red de este tipo funcionando bajo el estándar IEEE 802.15.4 existirán dos tipos diferentes de dispositivos, los FFD (Full Function Device) y los RFD (Reducen Function Device). Un FFD es un dispositivo con una mayor funcionalidad y responsabilidad, que podrá operar en 3 modos diferentes, como coordinador de una red PAN, como coordinador, o simplemente como dispositivo, y que se puede comunicar tanto con otros FFD como con los RFD. Por su parte, un RFD es un dispositivo de mucha más simple funcionalidad, que sólo puede comunicarse con los FFD, y sólo puede estar enlazado a un único de estos dispositivos al tiempo. Los RFD están pensados para actuar en aplicaciones simples donde no sea necesario transferir grandes cantidades de datos, y donde su implementación requiera pocos recursos y memoria.

Los dispositivos anteriormente descritos van a representar los componentes más básicos de una red IEEE 802.15.4, de manera que al menos serán necesarios dos de ellos estableciendo comunicación por el mismo enlace físico para constituir la red, teniendo en cuenta que al menos debe haber un dispositivo FFD actuando como coordinador de la red PAN. La topología física que formen estos dispositivos para establecer la red puede ser de dos tipos diferentes: Topología en estrella y topología peer to peer.

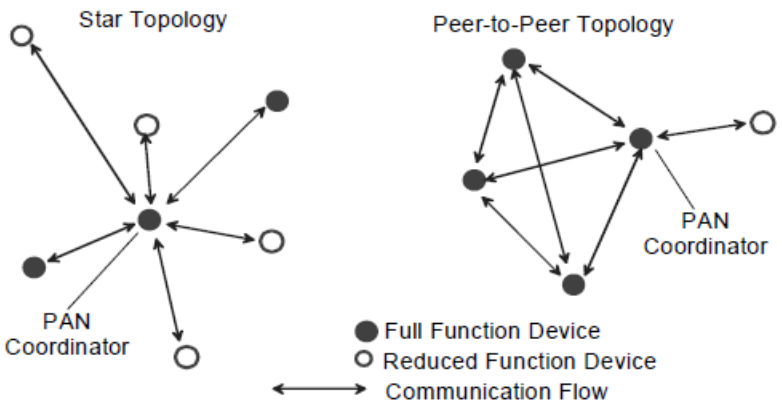


Figura 2. Topologías de redes IEEE 802.15.4 [4]

La topología en estrella se va a caracterizar por la existencia de un dispositivo controlador central, denominado coordinador PAN, con el cual establecerán la comunicación el resto de dispositivos de la red. La formación de una red en estrella consiste en la activación de un dispositivo FFD, que tras ser activado formará su propia red y se convertirá en coordinador PAN. Una vez que esta red formada disponga de un identificador único, el coordinador PAN permitirá a otros dispositivos, tanto FFD como RFD, unirse a la red.

Por su parte, una red con topología peer to peer se diferenciará de una red en estrella en las posibilidades de comunicación de los dispositivos entre ellos. En una red peer to peer seguirá existiendo un coordinador PAN, pero los dispositivos de la red pueden comunicarse unos con otros, no sólo con este coordinador. Esto permite estructuras de red más complejas que las redes con topología en estrella.

A pesar de las diferencias entre estas dos topologías, existirán algunas características que todas las redes PAN IEEE 802.15.4 van a poseer independientemente de esa topología. Así, todas las redes tendrán un identificador único, que va a permitir tanto la comunicación entre los dispositivos de la misma red empleando direcciones cortas, como entre dispositivos de redes independientes. También cada dispositivo dentro de una red va a estar identificado mediante una dirección única de 64 bits que será empleada para la comunicación con otros dispositivos, aunque en el caso de estar en la misma red, también puede utilizarse una dirección más corta que el coordinador PAN puede asociar a cada dispositivo.

2.2.2.ARQUITECTURA

La arquitectura del estándar IEEE 802.15.4 va a estar estructurada en varias capas basadas en el modelo de 7 capas OSI, cada una de las cuales tendrá unas responsabilidades diferentes y ofrecerá servicios a las capas superiores. En un dispositivo en una red LR-WPAN se podrán diferenciar la capa física y la subcapa MAC.

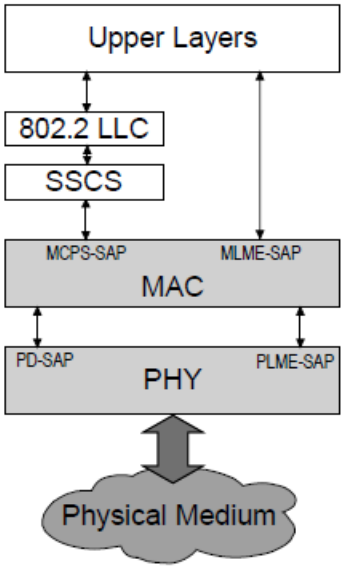


Figura 3. Arquitectura de sistema IEEE802.15.4 [4]

- **Capa física**

Esta capa ofrecerá dos servicios diferentes, el servicio de datos y el servicio de gestión, el primero de los cuales permite la transmisión y recepción de datos a través del canal físico de radio. Entre las tareas de esta capa física están la activación y desactivación del trasceptor de radio, la selección de canal, evaluación del estado del canal (CCA- Clear Channel Assesment) y la transmisión y recepción de paquetes a través del medio físico.

- **Subcapa MAC**

Las tareas de esta subcapa serán la gestión de beacons, el acceso al canal, la validación de tramas o el asentimiento de tramas recibidas.

2.2.3.FUNCIONAMIENTO DEL PROTOCOLO

2.2.3.1. MODELOS DE TRANSFERENCIA DE DATOS

El estándar IEEE 802.15.4 va a disponer de 3 modelos diferentes de transmisión de datos. Uno de ellos se corresponderá con el caso en el que un dispositivo transfiere datos a un coordinador, otro con el caso en el que es el coordinador el que transfiere datos a un dispositivo, y el tercero representará la situación en la que son dos dispositivos peer los que se comunican. En una topología peer to peer los 3 mecanismos son posibles, mientras que una en estrella sólo estarán presentes los dos primeros [4].

- Transferencia de datos a un coordinador

El mecanismo de transferencia de datos hacia un coordinador en una red PAN variará según esta red soporte o no la transferencia de beacons. Estos beacons serán unas tramas especiales empleadas para sincronizar a los dispositivos unidos y para identificar a la red PAN. Así, si la red soporta estos beacons, el primer paso que llevará a cabo un dispositivo que quiera transferir datos a un coordinador, será escuchar por el beacon de red. Cuando este beacon es encontrado, el dispositivo se sincronizará con la estructura de supertrama. Ésta es una estructura que deberán seguir todos los dispositivos para poder estar sincronizados, y que consiste en 16 ranuras temporales de igual tamaño, la primera de las cuales estará ocupada por la trama beacon que es generada por el coordinador. Para que un dispositivo pueda transferir sus datos una vez que ha escuchado el beacon, deberá competir con el resto de dispositivos de la red empleando acceso al medio CSMA-CA (Carrier sense multiple access with collision avoidance). Una vez llevada a cabo la transmisión de sus datos, el dispositivo puede esperar por una trama de asentimiento por parte del coordinador, en caso de que esta opción esté habilitada.

En el caso de que la red no soporte la transferencia de beacons, el dispositivo transmite sus datos empleando también acceso al medio CSMA-CA, pero esta vez no ranurado. También en este caso el coordinador puede emitir una trama de asentimiento de datos recibidos en caso de que la opción esté activada.

- Transferencia de datos por un coordinador

En este caso el mecanismo también variará según la red PAN soporte o no la transferencia de beacons. En el caso de que sí lo haga, lo primero que hará el coordinador que quiere transmitir los datos es indicar en el beacon de red que tiene datos pendientes por transmitir a un dispositivo. El dispositivo periódicamente escucha el beacon de red, y si descubre que hay datos pendientes para él, emite un comando MAC empleando acceso al medio CSMA-CA ranurado para solicitar la transferencia de esos datos pendientes. El coordinador confirmará que ha recibido esta solicitud de transferencia de datos, y los transmitirá empleando también acceso al medio CSMA-CA ranurado. Si la opción está habilitada, el dispositivo que recibe los datos puede emitir una trama de asentimiento para informar de este hecho. Por último, el coordinador retirará del beacon de red la indicación de que los datos están pendientes por enviar.

En el caso de una red PAN que no soporte el uso de beacons, el coordinador almacenará los datos destinados a un dispositivo determinado, a la espera de que éste contacte y requiera los datos. Este requerimiento se hace a través de un comando MAC enviado mediante acceso al medio CSMA-CA no ranurado. El coordinador informará de que ha recibido la petición del dispositivo, y en caso de que tenga datos pendientes para él, se los transmitirá empleando también para ello CSMA-CA no ranurado. En el caso de que el coordinador reciba la petición de transferencia de datos pero no disponga de datos pendientes por transmitir, puede informar al dispositivo de dos maneras. O bien indicando este hecho directamente en la confirmación del recibimiento de la petición de datos del dispositivo, o bien transmitiendo una trama sin datos útiles.

- Transferencia de datos entre peers

En el caso de una red PAN en la que los dispositivos puedan conectar y comunicarse unos con otros sin excepción, la transferencia de datos puede hacerse de dos maneras diferentes. Por un lado, un dispositivo puede transmitir directamente sus datos empleando para ellos acceso al medio CSMA-CA no ranurado. El otro mecanismo posible es de mayor complejidad, y requerirá diferentes procesos para conseguir la sincronización de los dispositivos antes de llevar a cano la transferencia de datos.

2.2.3.2. ESTRUCTURA DE TRAMA

El protocolo IEEE 802.14.5 define 4 tramas diferentes, cuyo diseño está orientado a buscar la máxima simplicidad pero asegurando la robustez necesaria para la transmisión en un canal con presencia de ruido. Estas 4 tramas serán la trama beacon, la trama de datos, la trama de asentimiento de datos recibidos, y la trama de comando MAC.

Simplemente a modo de información, a continuación se recogen las estructuras de estas 4 tramas cuya utilidad se ha comentado en el apartado anterior de modos de transmisión [4].

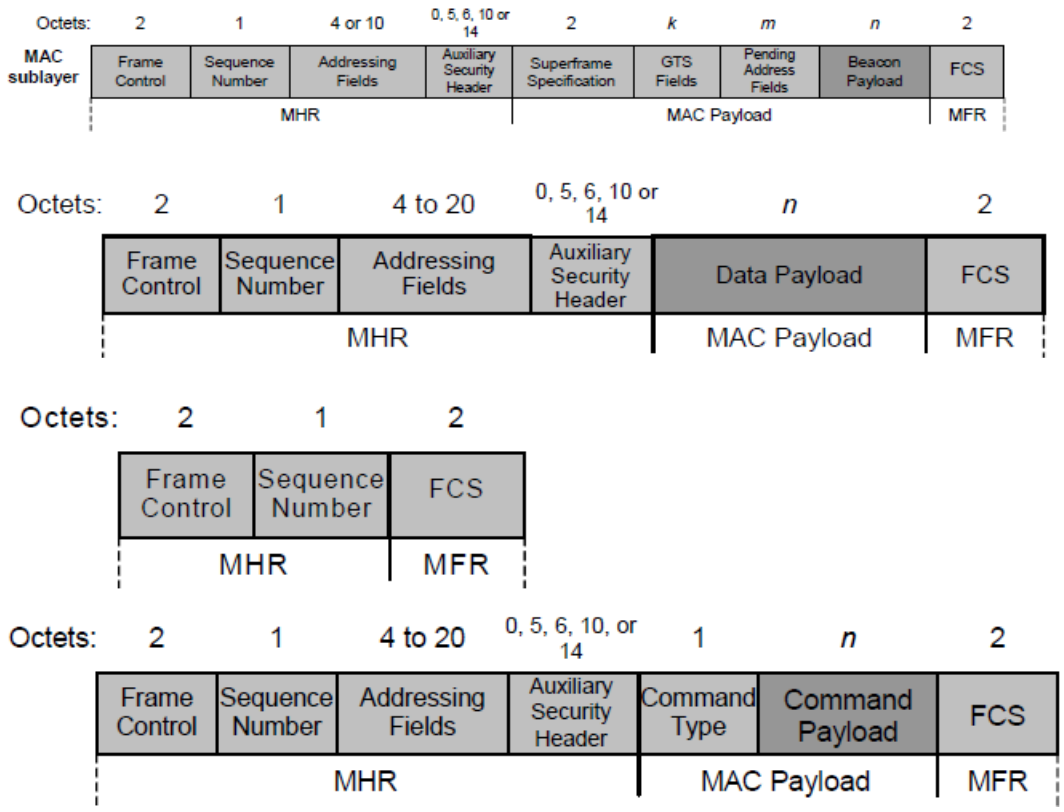


Figura 4. Diferentes tipos de trama del estándar IEEE802.15.4

2.2.3.3. MECANISMOS DE MEJORA DE LA TRANSMISIÓN DE DATOS

El protocolo IEEE 802.15.4 va a emplear una serie de mecanismo cuya finalidad es aumentar la probabilidad de que un proceso de transmisión de datos se efectúe de manera correcta. Estos mecanismos son el acceso al medio CSMA-CA, el asentimiento de tramas, y la verificación de datos.

CSMA-CA

Como ya se ha mencionado en apartados anteriores, a la hora de transmitir datos, el protocolo emplea dos mecanismos diferentes CSMA-CA, ranurado y no ranurado. En una red PAN donde no esté habilitado el uso de beacons, se empleará acceso al medio CSMA-CA no ranurado. En este caso, cuando un dispositivo quiere llevar a cabo la transmisión de una trama de datos o de una trama de comando MAC, lleva a cabo una espera por un tiempo aleatorio. Tras esta espera escucha el canal, y si no está ocupado lleva a cabo la transmisión. En caso contrario de que el canal sí esté ocupado, el dispositivo vuelve a esperar un tiempo aleatorio y lleva a cabo otra vez la escucha del canal. En el caso de estas redes, las tramas de asentimiento no se transmiten empleando un mecanismo CSMA-CA.

Por otro lado, si la red soporta la transmisión de beacons, el mecanismo de acceso al medio CSMA-CA que se empleará será ranurado, donde las ranuras se alinean y sincronizan con el comiendo de las transmisión de la trama beacon. Así, cuando un dispositivo quiere transmitir, localiza los límites de la ranura actual, y luego espera un número aleatorio de ranuras. Entonces, si el canal está libre, el dispositivo empieza a transmitir en la siguiente ranura. Si el canal está ocupado, espera otro número aleatorio de tiempos de ranura y vuelve a tratar de acceder al canal. En este tipo de redes, en la transmisión de las tramas de confirmación y las tramas beacon no se emplea el acceso al medio CSMA-CA ranurado.

Asentimiento de tramas

Este mecanismo se trata de una opción que se puede activar o no, y que si está habilitada supondrá que las tramas de datos y las tramas de comando MAC serán confirmadas por el receptor mediante el envío de una trama de asentimiento al transmisor de dichas tramas, siempre y cuando éstas sean recibidas correctamente.

En el caso de que este mecanismo esté habilitado y el dispositivo transmisor no reciba la trama de asentimiento tras un tiempo de espera, éste asumirá que la transmisión no se ha llevado a cabo correctamente. En este caso, el transmisor volverá a llevar a cabo la transmisión de los datos. Si tras un determinado número de intentos sigue sin recibirse el asentimiento de los datos, el transmisor puede decidir dar por finalizada la transmisión. Si este mecanismo no está habilitado, el transmisor siempre asumirá que la transmisión se ha efectuado correctamente.

Verificación de datos

El protocolo IEEE 802.15.4 emplea un mecanismo de verificación de secuencia de trama (Frame Check Sequence FCS) para detectar posibles errores de bit en cada trama, utilizando para ello un mecanismo CRC (Cyclic Redundancy Check) de 16 bits.

2.2.4.ESTÁNDAR IEEE 802.15.4e

2.2.4.1. INTRODUCCIÓN

El objetivo con el que se diseñó el protocolo IEEE 802.15.4e fue superar las limitaciones del estándar IEEE 802.15.4, diseñando para ello un protocolo MAC multisalto y para aplicaciones de bajo consumo, que fuera capaz de afrontar las necesidades de las aplicaciones industriales embebidas.

Entre esas limitaciones del protocolo IEE 802.15.4 pueden destacarse [5]:

- Retardo no limitado

Ya que el protocolo IEE 802.15.4 emplea el mecanismo de acceso al medio CSMA-CA, tanto si la red soporta el uso de beacons como si no, no se puede garantizar un retardo máximo de transmisión para los datos, ya que los dispositivos tendrán que encontrar el canal libre antes de transmitir, y no se puede asegurar un número de intentos de escucha máximos antes de que eso ocurra. Por lo tanto, este protocolo no es adecuado para aplicaciones donde el retardo sea crítico, como las médicas o industriales.

- Fiabilidad en la comunicación limitada

El protocolo IEE 802.15.4 en una red PAN con habilitación de tramas beacon presenta una tasa de entrega de datos muy baja, debido a la aleatoriedad del mecanismo de acceso al medio empleado y la sincronización necesaria previa a la transmisión. Esto ocurre incluso aunque el número de nodos sea pequeño, mientras que si la red no emplea las tramas beacon, aunque se elimina la necesidad de la sincronización, el mecanismo de acceso al medio sigue siendo CSMA-CA, con lo que esta tasa de entrega seguirá siendo baja si el número de nodos es alto y un buen número de ellos comienzan la transmisión simultáneamente. Por lo tanto, esta característica también hace a este estándar inadecuado para escenarios críticos.

- Protección ante interferencias y multicamino

Los fenómenos de las interferencias y el desvanecimiento por multicamino son muy habituales en sistemas que cuentan con sensores y actuadores en la red. A diferencia de otras tecnologías inalámbricas, el estándar IEEE 802.15.4 emplea un planteamiento de un único canal, y no cuenta con mecanismos de salto frecuencial para conseguir protección frente a estos habituales fenómenos. Este hecho vuelve a poner de manifiesto que el protocolo no es adecuado para escenarios críticos.

- Consumo

Aunque teóricamente el estándar IEEE 802.15.4 permite la construcción de topologías donde el nodo central no precisa de estar activo todo el tiempo, en la práctica se ha demostrado que los

nodos intermedios en las redes IEEE 802.15.4 necesitan mantenerse activos todo el tiempo, con el gran consumo de energía que ello supone, lo cual lo hace inadecuado para aplicaciones industriales que precisan muy bajo consumo.

A la hora de mejorar el estándar IEEE 802.15.4 y conseguir superar estas limitaciones, el estándar IEEE 802.15.4e introduce dos diferentes tipos de mejoras MAC. Por un lado estarán las mejoras denominadas como funcionamientos MAC, destinadas a dominios de aplicación específicos, y por otro las mejoras de funcionamiento generales, que no estarán asociadas a ningún dominio de aplicación específico.

2.2.4.2. MODOS DE FUNCIONAMIENTO MAC

Una de las mejoras incorporadas por el protocolo IEEE 802.15.4e consiste en la inclusión de 5 nuevos modos de funcionamiento, cada uno de los cuales está dirigido a un dominio de aplicación específico. A continuación se recoge un resumen de estos diferentes modos [5].

- **Modo RFID Blink**

Este modo permite a un dispositivo transmitir su identificador, tanto el completo de 64 bits como el alternativo más corto, y otros datos adicionales a otro dispositivo, sin que sea necesario que haya ningún tipo de asociación previa entre estos dispositivos ni que el receptor tenga que asentir la correcta llegada al transmisor.

Este modo está dirigido a dominios de aplicación como la identificación de personas y objetos, su localización y seguimiento, por lo que está muy relacionado con el IoT. Este modo está basado en un trama de mínimo tamaño para conseguir que su transmisión requiera el menor gasto de energía posible, y que estará formada únicamente por los campos de la cabecera necesarios para el funcionamiento deseado.

- ***Asynchronous multi-channel Adaptation* (AMCA)**

Este modo de funcionamiento está ideado para redes de gran tamaño, donde grandes despliegues son necesarios, y donde sistema con un único canal común puede no ser suficiente para comunicar a todos los dispositivos en una PAN, pudiendo aparecer una gran variación en la calidad del canal o asimetrías entre dos dispositivos que provoquen que uno de ellos no pueda recibir lo que el otro transmite. Este modo sólo puede ser empleado en redes PAN donde no esté habilitado el uso de beacons.

- ***Deterministic and Synchronous Multi-channel Extension* (DSME)**

Este modo está destinado a aplicaciones en las cuales se requiera una latencia baja y determinística, gran fiabilidad, eficiencia energética, escalabilidad, flexibilidad y robustez. Por

ejemplo, este modo sería adecuado para aplicaciones industriales de automatización de procesos y automatización de fábrica, aplicaciones comerciales de automatización de viviendas y edificios inteligentes, o aplicaciones médicas de monitorización de pacientes.

Aunque el protocolo antecesor IEEE 802.15.4 contempla el uso de GTS (Guaranted Time Slots), porciones de una supertrama dedicadas en exclusiva a aplicaciones que requieran baja latencia o un ancho de banda específico, estos tienen una serie de características que limitan en gran medida su utilización, como por ejemplo que sólo se incluyen como máximo 7 ranuras, lo que impide que sean empleados en redes de gran tamaño, o que se emplea un canal de una única frecuencia.

DSME mejora estos GTS agrupando las supertramas en una supertrama múltiple y empleando un funcionamiento multicanal. Evidentemente, DSME sólo será utilizable en redes PAN donde este habilitada la opción de beacons.

- ***Low Latency Deterministic Networks (LLDN)***

Este modo está dirigido principalmente a aplicaciones comerciales e industriales que requieran una latencia baja y determinística, como por ejemplo automatización de fábricas, robots, empaquetamiento automatizado o logística de aeropuerto. Estas aplicaciones normalmente contarán con un gran número de sensores y actuadores, y tendrán unos requisitos estrictos de baja latencia y también de bajo tiempo de ida y vuelta (round-trip time).

Para cumplir estos requisitos de baja latencia, el modo de funcionamiento LLDD emplea una topología en estrella y utiliza supertramas, en las que las ranuras de tiempo serán de pequeña duración, lo que permitirá a su vez que la supertrama también sea corta. Como el número de dispositivos que pueden acceder al canal dependerá del número de ranuras de la supertrama, y las redes de las aplicaciones a las que va destinado este modo suelen ser de numerosos dispositivos, este modo permite que el coordinador PAN pueda hacer uso de varios transmisores en diferentes canales.

La supertrama anteriormente comentada estará formada por ranuras beacon, ranuras de gestión, que son opcionales, y un número determinado de ranuras base, que serán de igual tamaño. Estas ranuras base pueden ser de dos tipos diferentes, dedicadas y compartidas. Las ramuras dedicadas que estén asignadas a un único dispositivo, que por lo tanto será el único que podrá acceder a ellas. Por su parte, las ranuras compartidas, como su nombre indica, podrán ser accedidas por varios dispositivos, que tendrán que competir por ellas mediante el mecanismo de acceso al medio CSMA-CA ranurado.

- ***Time Slotted Channel Hopping (TSCH)***

Este modo esta principalmente dirigido a aplicaciones de automatización de procesos centradas especialmente en la monitorización de procesos y de equipamiento. Los sectores a los que está dirigido este modo de funcionamiento incluyen la industria del gas, sector químico, sector farmacéutico, tratamiento de agua, o el control del clima.

Este modo de funcionamiento va a combinar el acceso al medio por tiempo ranurado, que ya incluía el protocolo base IEEE 802.15.4, con dos novedades como capacidad multicanal y de salto de canal. Así, con el acceso por tiempo ranurado se puede conseguir un mayor rendimiento

gracias a la eliminación de las colisiones entre los dispositivos que compiten por el canal, y también una latencia determinística para las aplicaciones. Con la característica multicanal se puede aumentar la capacidad de la red, gracias a que más dispositivos podrán transmitir sus tramas al mismo tiempo, mientras que el salto de canal permite la protección contra interferencias y desvanecimiento por multicamino, uno de las debilidades del protocolo base, aumentando así la fiabilidad del sistema.

Por lo tanto, este modo permite aumentar la capacidad y la fiabilidad de la red, asegurando una baja predecible y manteniendo una alta eficiencia energética gracias a los cortos ciclos de trabajo que se consigue con el mecanismo de acceso al medio por tiempo ranurado.

2.2.4.3. MEJORAS DE FUNCIONAMIENTO GENERALES

La segunda categoría de mejoras introducidas por el protocolo IEEE 802.15.4e se corresponde con aquellas mejoras sobre el protocolo base que no están asociadas a ningún dominio específico de aplicación, sino que son generales. A continuación se muestra una breve descripción de estas seis mejoras generales [6].

- **Baja energía (Low Energy-LE)**

Este mecanismo está dirigido a aplicaciones que están dispuestas o pueden permitirse perder requisitos de latencia frente a mejorar sus características de eficiencia energética. Así, este mecanismo está basado en permitir a un dispositivo operar con un ciclo de trabajo muy bajo, de un 1% o incluso inferior, mientras que parecerá siempre conectado para las capas superiores, lo cual es muy importante ya que los protocolos de internet asumen que los equipos están siempre activados. Este mecanismo es utilizable tanto en redes PAN que tengan habilitado el uso de beacon y empleen supertramas, como en aquéllas que tengan deshabilitada esta opción del empleo de tramas beacon.

Existirán dos tipos diferentes de mecanismos LE. Por un lado estará el mecanismo CSL (Coordinated Sample Listening) y por otro el mecanismo RIT (Received Initiated Transmissions). CSL es aplicable en situaciones donde los requisitos de latencia sean bajos, menores de un segundo, mientras que RIT es aplicable en situaciones en las que la tolerancia respecto a la latencia es mayor, de decenas de segundos.

- **Elementos de información (Information Elements - IE)**

Los IE son mecanismos bien definidos y extensibles de intercambio de información en la subcapa MAC. Estos mecanismos van a proporcionar contenedores flexibles de información, que se pueden utilizar para encapsular diferente tipo de información como la relacionada con la sincronización o el estado de la red.

- **Beacons mejorados (Enhanced Beacons)**

Los beacons mejorados son una extensión de los beacon del protocolo base que cuentan con una mayor flexibilidad en su contenido que los originales. Estos beacons mejorados permiten la creación de beacons específicos de aplicación, incluyendo los IE´s relevantes para cada caso

concreto, que son empleados en los modos de funcionamiento DSME y TSCH anteriormente descritos.

- **Trama multipropósito (Multipropose Frame)**

Este mecanismo proporciona un formato flexible y extensible de trama que permite contener una gran variedad de operaciones MAC, utilizando para todas ellas un único valor de tipo de trama. Gracias a esto se puede conseguir que una misma estructura de trama pueda soportar múltiples operaciones MAC, sea extensible para soportar nuevos requerimientos MAC, y además posea compatibilidad hacia atrás.

Esta flexibilidad se consigue gracias a basar esta trama multipropósito en IE´s, lo que permite añadir nuevas funcionalidades sin tener que modificar la estructura de la trama o añadir nuevos tipos de trama.

- **Métricas de mejora MAC**

Este nuevo mecanismo añadido tiene como objetivo informar a las capas superiores sobre aspectos claves relacionados con la transmisión y recepción de tramas. Así, la recepción de la útil información sobre la calidad del canal permite tomar decisiones más correctas.

- **Asociación rápida (Fast Association- FastA)**

Este mecanismo tiene como objetivo mejorar el proceso de asociación empleado por el protocolo base IEEE 802.15.4, en el que dicho proceso está ideado de manera que se de prioridad a la eficiencia energética, lo cual introduce un importante retardo en este proceso de asociación. En aquellas aplicaciones donde aparecen requisitos temporales críticos, este retardo no es aceptable, por lo que el mecanismo FastA permite reducir este tiempo de asociación a cambio de una menor eficiencia energética en favor de una menor latencia.

2.2.5.OTROS ESTÁNDARES IoT

2.2.5.1. SIGFOX

SIGFOX se trata de una solución IoT que emplea una red alternativa independiente de las redes existentes, basada en una infraestructura de antenas y estaciones base. Se basa en la difusión de datos por parte de los objetos participantes en la red, sin que exista la necesidad de establecer y mantener conexiones, y sin que éstos por lo tanto tengan que permanecer únicos a la red. Así, un dispositivo no está enlazado con una estación base específica, sino que sus mensajes pueden ser recibidos por cualquiera de las estaciones bases que estén dentro de su rango, y no existe asentimiento para estos mensajes [7].

Emplea tecnología de radio UNB (Ultra Narrow Band), utilizando 200KHz de las bandas libres ISM (Industrial, Scientific and Medical ISM radio band), en las frecuencias 868-869MHz en Europa y 902-928MHz en Estados Unidos. El empleo de UNB permite concentrar la potencia de la señal en una banda muy estrecha, lo que proporciona una gran robustez contra las

interferencias, permitiendo así esta concentración de la energía que las estaciones base puedan demodular fácilmente las señales recibidas incluso en presencia de otras interferentes más potentes. Así, combinada con esta tecnología UNB, SIGFOX emplea modulación DBPSK (Differential Binary Phase Shift Keying) en el enlace ascendente y GFSK (Gaussian Frequency Shift Keying) en el enlace descendente. Cada mensaje tendrá un espectro de 100Hz con una tasa de transmisión de 100 ó 600 bits/s según la región. El empleo de modulación DBPSK se debe a su sencillez de implementación, a que la baja tasa de datos empleados permite utilizar componentes de bajo coste y a que su empleo posibilita una alta sensibilidad de las estaciones base, que podrán demodular señales muy próximas a la banda de ruido sin la necesidad de ningún tipo de codificación adicional. Esta sensibilidad de las estaciones base dependerá de la tasa de transmisión, siendo de -142dBm para la tasa de transmisión de 100 bps y de -134dBm para la tasa de 600 bps. Esta diferencia de sensibilidad será equilibrada por parte del dispositivo emisor, que empleará una mayor potencia radiada.

La transmisión entre los dispositivos SIGFOX y la red es asíncrona, de manera que el dispositivo difunde cada mensaje 3 veces a tres frecuencias diferentes, y las estaciones base rastrean el espectro buscando señales UNB, las cuales serán demoduladas.

SIGFOX utiliza mensajes de datos de pequeño tamaño contribuyendo a una gestión más sencilla y ligera de la red. Así, un mensaje de SIGFOX puede tener hasta un máximo de 12 bytes de carga útil, pudiendo tener la trama completa hasta 26 bytes en total. Así SIGFOX trata de reducir al máximo la cantidad de datos adicionales que no representen carga útil, minimizando para ello el tamaño de cabeceras de las tramas y evitando el uso de datos para la señalización y sincronización, que no son necesarios al tratarse de una red asíncrona. Mediante el empleo de este protocolo más ligero SIGFOX consigue por un lado un menor consumo de energía al necesitar la transmisión de menos datos, lo que permitirá una mayor duración de la batería de los dispositivos, y por otro, este uso más eficiente de los datos conlleva una mayor capacidad, al existir una mayor proporción de datos útiles en cada trama.

La pila de protocolos de SIGFOX cubre del 1 al 4 de los niveles del modelo OSI, y estará compuesto por 3 capas: nivel trama, control de acceso al medio y nivel físico. Esta pila de protocolos se sitúa entre los dispositivos conectados y la red, y está implementado en estos dispositivos para permitir llevar a cabo la modulación de la trama y la emisión de los mensajes.

El nivel trama es la primera capa de esta pila de protocolos, y su misión es recibir la carga de datos por parte de la capa de aplicación y generar a partir de ellos la trama que será transmitida via radio, añadiendo a estos datos un número de secuencia. En segundo lugar está la capa de control de acceso al medio, cuya función es añadir a la trama diferentes campos destinados a la identificación del dispositivo, así como otros parámetros como el código de detección de errores. Como ya se ha mencionado anteriormente, SIGFOX cuenta con la característica de que la transmisión es asíncrona, lo que conlleva a que este nivel de acceso al medio no incluye ningún tipo de información de señalización. Por último, en el último nivel de esta pila se encuentra la capa física, encargada de determinar cómo las señales SIGFOX son generadas. En este aspecto serán de suma importancia las técnicas de modulación empleadas, que como se indicó anteriormente se tratará de modulación GFSK en el enlace descendente y DBPSK en el ascendente.

La red SIGFOX contará con varios mecanismos para garantizar la seguridad. El primer elemento de seguridad aparecerá entre los dispositivos y la nube SIGFOX, ya que existirá un método de autentificación extremo a extremo basado en una clave privada. Esta clave estará almacenada

en una memoria no accesible, y será empleada en los mensajes enviados por los dispositivos para crear una firma única para cada mensaje que autentificará al emisor. Esta firma incluye un número de secuencia que se añade a cada trama, para evitar que se repitan. Otro aspecto de seguridad, ya relacionado con el medio radio, es que cada mensaje se envía 3 veces y a 3 frecuencias diferentes elegidas de manera aleatoria, consiguiéndose así que no se sepa de antemano en qué frecuencia va a transmitir un dispositivo, evitando que pueda ser escuchada dicha transmisión. Además, en lo que respecta a las estaciones base, la conexión de éstas con la nube SIGFOX se lleva a cabo a través de un enlace punto a punto empleando una VPN encriptada, consiguiendo así una comunicación robusta y segura.

2.2.5.2. LoRa

LoRa [8] se trata de una tecnología inalámbrica de radiofrecuencia diseñada para dispositivos de bajo consumo, operando en redes que pueden ser tanto de alcance local, regional, nacional o global. Además de este bajo consumo de energía, LoRa también cumple con otros de los requisitos y características del IoT, como comunicación bidireccional segura, movilidad y servicios de localización, bajas tasas de transmisión, largo alcance de las comunicaciones y baja frecuencia de transmisión.

Lora trabaja en espectro de radio no licenciado en las bandas ISM, empleando diferentes canales de frecuencia y tasas de transmisión en las comunicaciones. Así, en Europa LoRa emplea la banda de 858MHz, pudiendo ser los canales distribuidos libremente por el operador, aunque existen 3 canales que deben ser obligatoriamente recibidos por todas las puertas de enlace, cada uno de ellos de 125KHz de ancho: 868,1 MHz, 868,30 MHz y 868,5 MHz. Por otra parte, en Norte América LoRa opera en la banda de 915MHz, con 72 canales de subida y 8 canales de bajada.

La topología típica de LoRa consta de 4 elementos bien diferenciados: dispositivos, puertas de enlace, servidor de red y servidores de aplicación.

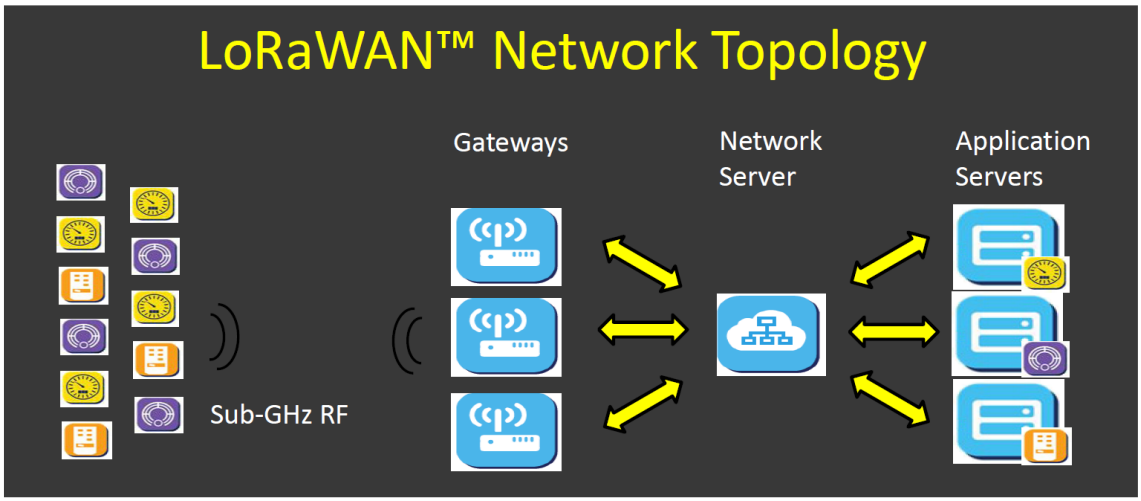


Figura 5. Topología de LoRa

Los dispositivos finales están comunicados con una o varias puertas de enlace mediante comunicación inalámbrica de un único salto, mientras que las puertas de enlace se comunican con el servidor de red mediante conexiones IP estándar. Como ya se ha comentado, existen

diferentes tasas de transmisión y canales de frecuencia mediante los cuales se comunican los dispositivos y las puertas de enlace, siendo normalmente la tasa de transmisión elegida cumpliendo un compromiso entre el alcance y la duración del mensaje. Estas tasas de transmisión pueden ir desde los 0.3kbps hasta los 50 kbps, y gracias al empleo de la tecnología de espectro ensanchado, LoRa consigue que las comunicaciones con diferentes tasas no interfieran unas con otras y se crea un conjunto de canales virtuales que aumenta la capacidad de la puerta de enlace. Además, LoRa también utiliza un esquema adaptativo de tasa de datos (ADR- Adaptive Data Rate), mediante el cual la tasa de datos y la frecuencia son gestionadas de manera individual para cada dispositivo final por el servidor de red, lo que permite maximizar la duración de las baterías y la capacidad de la red.

Los dispositivos finales van a estar divididos en 3 clases diferentes según su comportamiento y participación en la comunicación:

- **Clase A**: Estos dispositivos permiten comunicaciones bidireccionales, donde el dispositivo inicia la comunicación con el servidor (Enlace ascendente), y a continuación le siguen dos periodos o ventanas en las que el servidor se comunica con el dispositivo (enlace descendente). Cualquier otra comunicación del servidor debe esperar hasta otra comunicación del dispositivo. Este tipo de comunicación es la que representa menos consumo de energía.
- **Clase B**: Estos dispositivos permiten comunicaciones bidireccionales, en las que además de las ventanas aleatorias de los dispositivos de clase A, existirán también periodos de recepción de datos extra en momentos concretos. Para conocer estos momentos y poder recibir estas comunicaciones, existen unas balizas periódicas sincronizadas que envían las puertas de enlace y que permiten saber a los dispositivos que deben abrir su ventana de recepción, y al servidor conocer que el dispositivo está escuchando. En resumen, en este caso además de iniciar la comunicación el dispositivo, el servidor también puede iniciarla en momentos concretos.
- **Clase C**: En este caso los dispositivos finales permiten comunicación bidireccional total, de manera que el servidor puede también iniciar la comunicación cuando quiera, estando la ventana de recepción del dispositivo siempre abierta salvo si éste está transmitiendo.

Antes de que un dispositivo final pueda iniciar una comunicación, es necesario que lleve a cabo un proceso de activación. Este proceso puede realizarse siguiendo diferentes mecanismos, pero siempre hará falta una serie de información:

-**Dirección del dispositivo**: Identificador de 32 bits que es único dentro de cada red y que está presente en cada trama. Este identificador es compartido entre el dispositivo, el servidor de red y los servidores de aplicación.

-**Clave de sesión de red**: Clave encriptada (AES – Advanced Encrytion Standard) de 128 bits. Es única para cada dispositivo final, y es compartida por el dispositivo final y el servidor de red, proporcionado seguridad a la comunicación entre estos dos elementos e integridad a los mensajes de la comunicación.

-**Clave de sesión de aplicación**: Al igual que la clave de sesión de red, se trata de una clave encriptada (AES) de 128 bits, única para cada dispositivo final y compartida en este caso por el dispositivo y el servidor de aplicación. Esta clave se emplea para encriptar y desencriptar mensajes de aplicación, proporcionando así seguridad a los datos de aplicación.

El proceso de activación de un dispositivo que es necesario para que éste pueda comunicarse en una red, consistirá principalmente en la obtención por parte del dispositivo de las informaciones anteriores, proceso para el cual existen dos alternativas diferentes:

- Activación OTAA (*Over The Air Activation*): Este método está basado en identificador único global, y consiste en un intercambio de mensajes inalámbrico. Así, se inicia con la transmisión por parte del dispositivo final de una petición de unión, Join Request, hacia el servidor de aplicación. Esta petición contiene el identificador único global de dispositivo, el identificador de aplicación, y la autentificación con la clave de aplicación. A continuación, el dispositivo final recibirá por parte del servidor de aplicación la aceptación para unirse, Join Accept, la cual desencriptará para extraer su dirección de red que almacenará, y obtendrá las claves de sesión de red y de sesión de aplicación.

- Activación por personalización (ABP – *Activation By Personalization*): En este caso, todas las informaciones descritas anteriormente, dirección de red, clave de de sesión de red y clave de sesión de aplicación son configuradas en tiempo de producción, de manera que no hay ningún tipo de intercambio de mensajes como en el caso anterior ni es necesario ningún otro tipo de proceso adicional para que el dispositivo esté preparado para comunicarse en la red.

2.2.5.3. Ingenu

Ingenu [9] es el creador de la tecnología RPMA (*Random Phase Multiple Access*), tecnología designada específica y exclusivamente para la comunicación inalámbrica máquina a máquina (M2M – *Machine to machine*), y totalmente adecuada para la comunicación del IoT. Las bases en las que se sustenta esta tecnología son una mayor cobertura, más capacidad y escalabilidad de la red, mayor duración de las baterías y mayor longevidad de la red.

RPMA emplea la banda de frecuencias de 2,4GHz, un espectro global no licenciado de las bandas ISM, y utiliza la técnica de modulación DSSS (*Direct-Sequence Spread Spectrum*), una técnica de modulación de espectro ensanchado que permite reducir la interferencia media de la señal, ensanchando para ello la señal y consiguiendo así una mayor resistencia a las interferencias, ya sean intencionadas o no. El ancho de banda de señal utilizado será de 1MHz,

DSSS se caracteriza por un parámetro denominado por ganancia de procesado, que de alguna manera sirve para reflejar cuánto se ha ensanchado la señal. Cuanto mayor sea este parámetro, mayores podrán ser también la sensibilidad del receptor y el área cubierta, pero a cambio de que se vea reducida la tasa de transmisión. Como en el caso del IoT esta tasa no necesita ser alta, se puede tratar de maximizar esta ganancia de proceso sin miedo a esta contraprestación, pudiéndose llegar a una sensibilidad del receptor de hasta -145dBm

3. DESCRIPCIÓN DEL SISTEMA DESARROLLADO

3.1. DESCRIPCIÓN GENERAL

El objetivo final de este proyecto es el desarrollo tanto hardware como software de un prototipo que permita servir de demostrador del IoT con el estándar IEEE 802.15.4e, empleando para ello la plataforma OpenMote respecto al Hardware, y OpenWSN respecto al software.

El sistema que se pretende desarrollar será un sistema capaz de medir la corriente eléctrica alterna que se está consumiendo en una toma de corriente determinada, permitiendo así en primer lugar determinar si hay algún aparato eléctrico conectado en esa toma y activado, y registrando el consumo de corriente en caso afirmativo. Así, una primera utilidad del sistema sería detectar si existe algún aparato eléctrico que se haya dejado enchufado y encendido por descuido, mientras que otra posible función sería la monitorización y registro del consumo de corriente en esa toma monitorizada. En una versión más desarrollada a partir de este prototipo básico se podría implementar un sistema que monitorizara varias tomas de corriente simultáneamente, y por lo tanto permitiera el control de consumo de por ejemplo una vivienda completa.

Existen en el mercado diferentes sistemas registradores del consumo eléctrico [10], [11], y la ventaja que se pretende conseguir con este prototipo respecto a ellos es un menor coste y mayor simplicidad, al emplear el hardware OpenMote, y una mucho mayor capacidad de desarrollo y evolución, al tratarse de un sistema de código abierto en su totalidad, lo que permitirá que cualquiera pueda trabajar en su desarrollo futuro. Además, el empleo de la plataforma theThings.io proporciona al sistema características de trabajo en la nube, lo que le concede prestaciones de acceso totales a los datos en cualquier momento y desde cualquier lugar, característica no disponible en otros sistemas.

Para llevar a cabo la función de medición de corriente, el sistema contará con un sensor de corriente alterna, capaz de detectar el paso de corriente eléctrica a través de él. Este sensor, colocado en una toma de corriente de la vivienda, podrá detectar si circula corriente alterna por ella, y por lo tanto si existe algún aparato eléctrico conectado y encendido. La conexión de este sensor a internet, siguiendo el paradigma del IoT, permitirá que se informe de este hecho al usuario, y que se registren estos consumos.

Así, el sensor que se ha elegido para el desarrollo del sistema es el sensor de corriente ACS712, tanto por su sencillez de manejo e integración en el sistema, como por su precio económico, proporcionando las prestaciones necesarias que se requieren en el prototipo que se pretende desarrollar.

Figura 6. Sensor de corriente ACS712

Este sensor proporciona a su salida un voltaje que es proporcional a la corriente medida en su entrada, existiendo una relación lineal entre ambas, relación por supuesto conocida. La idea detrás del sistema a desarrollar es colocar este sensor en la toma a monitorizar, y conectar su salida con una placa OpenMote CC2538 sobre un módulo OpenBattery, con lo que se conseguiría la autonomía de este conjunto de medición, al emplear además comunicación inalámbrica con el bloque central del sistema.

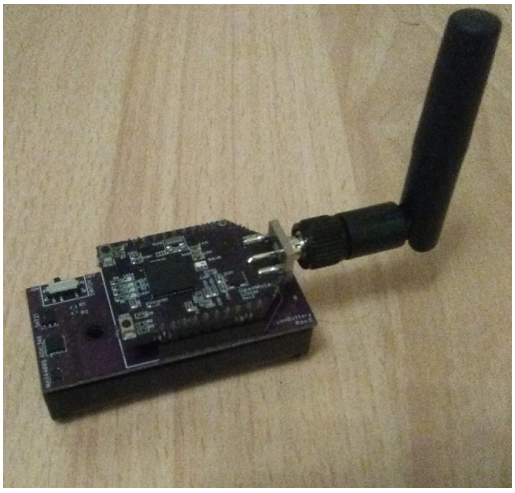


Figura 7. Módulo OpenBattery

Una vez obtenido este valor de voltaje que proporciona el sensor ACS712, sería necesario procesarlo para obtener el valor de corriente que representa, y así poder determinar el consumo que se está produciendo en esta toma de corriente. Para ello sería necesario emplear la relación entre el voltaje a la salida y la corriente a la entrada del sensor, y cualquier otra expresión o relación eléctrica necesaria, las cuales habría que incorporar en la parte software de procesado de datos. Esto se lleva a cabo en el bloque de procesamiento del sistema, el cual se comunica con el bloque sensor a través de la creación de una red inalámbrica OpenWSN, que permite la absoluta independencia de dicho bloque sensor, al no precisar ninguna conexión física con el bloque central de procesamiento.

Este bloque de procesamiento consiste en un circuito CC2538 montado sobre una placa OpenBase, estando el conjunto conectado a un ordenador portátil. En este ordenador se encontraría la aplicación central mediante la que el usuario arranca el sistema, y que se encarga de la gestión de las diferentes funcionalidades del mismo. Para este prototipo del sistema se ha empleado un portátil como contenedor del bloque de procesamiento, tanto por sencillez como por disponibilidad, y ya que el objetivo de este TFM es desarrollar simplemente un demostrador de una idea. Pero a la hora de desarrollar un sistema plenamente funcional, este bloque de procesamiento debería emplear algún dispositivo más económico y sencillo, como podría ser una raspberry Pi o incluso un Smartphone.

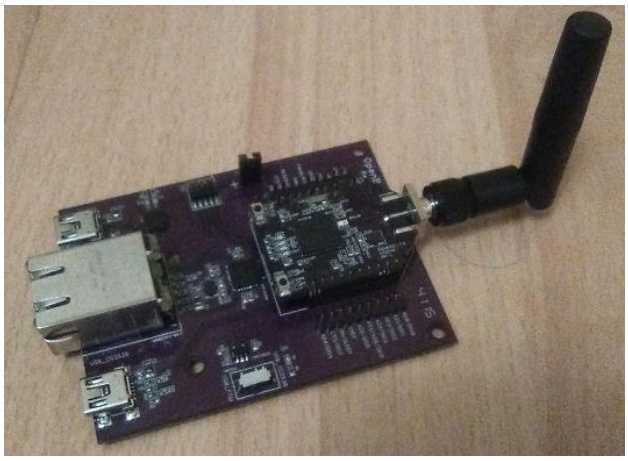


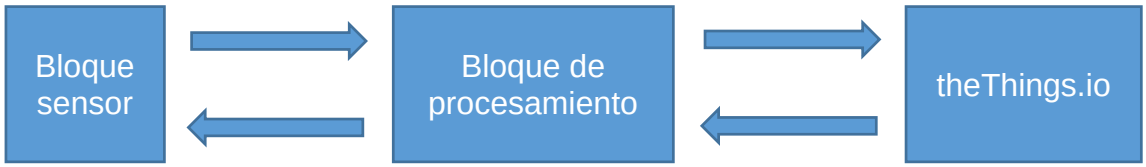
Figura 8. Módulo OpenBase

Por último, quedaría mencionar el último de los elementos que forman parte de este proyecto, la plataforma theThings.Io. Mediante esta plataforma se puede llevar a cabo el almacenamiento de los datos medidos por el sensor, así como su posterior consulta para mostrar la información recogida. Además, también se pueden emplear algunas de las funcionalidades de la plataforma para conseguir ciertas prestaciones, como los avisos mediante sms, para por ejemplo notificar al usuario de que hay algún aparato conectado a una toma cuando no hay nadie en la vivienda, o informarle de que el consumo ha superado cierto umbral.

Como ya se ha comentado, en un sistema completo sería preciso implantar los sensores en todos aquellos enchufes o tomas de corriente que se quiera monitorizar y controlar, para así poder detectar equipos y su consumo en todas ellas. En el prototipo que se pretende desarrollar en este proyecto, se limitará el trabajo a una única toma, ya que se considera que servirá de muestra más que suficiente para evaluar la utilidad de la idea propuesta. Así, con una toma de muestra ya se puede comprobar la correcta comunicación entre el módulo OpenBattery en el lado del sensor y la placa OpenBase conectada a portátil, así como generar datos de medida para ser almacenados y consultados en la plataforma theThings.Io. Añadir más tomas a monitorizar en el sistema sólo supondría el escalado de éste y la ampliación del equipamiento, empleando más sensores ACS712 y los módulos OpenMote CC2538 + OpenBattery necesarios.

3.2. ARQUITECTURA DEL SISTEMA

El sistema va a estar formado por 3 partes bien diferenciadas:



- Bloque sensor
- Bloque de procesamiento
- Plataforma de theThings.io

Bloque sensor

Será el encargado de llevar a cabo las medidas de la corriente que se consume en un enchufe, facilitando además estas medidas a la unidad central de procesamiento. Este bloque estará compuesto por el sensor ACS712 y la circuitería necesaria para su funcionamiento, y un módulo OpenBattery, que recibirá la salida del sensor en forma de voltaje. Este módulo OpenBattery realizará un primer pequeño procesamiento de estos datos recogidos por el sensor, empleando para ello el convertidor analógico-digital de la placa CC2538, y se los transmitirá a la unidad central de procesamiento.

Bloque de procesamiento

Este bloque está compuesto por un ordenador portátil, en el que se ejecutarán las herramientas software necesarias para la recepción, procesamiento y transmisión a la plataforma theThings.io de los datos procedentes del sensor de corriente. Además, conectada a este ordenador portátil habrá una placa openMote CC2538 montada sobre un módulo OpenBase, que será el enlace de la red OpenWSN con internet.

Plataforma theThings.io

Esta plataforma recogerá los datos de medición del sensor enviados por el bloque de procesamiento, y además permitirá opcionalmente configurar un mensaje de alerta al usuario en caso de que por ejemplo el consumo sobrepase cierto límite.

3.3. CONFIGURACIÓN HARDWARE

La arquitectura hardware del sistema estará formada por los diferentes elementos que permiten la medición de corriente, los elementos OpenMote que permiten la creación de la red OpenWSN en la que se basa el diseño, y por supuesto el ordenador portátil que sirve como unidad central de procesamiento y comunicación.

El bloque encargado de la medición de corriente estará compuesto por el sensor ACS712, circuitería necesaria para la alimentación y adecuación de este sensor, y una placa CC2538 montada sobre un módulo OpenBattery, que recibirá la salida del sensor.

Para facilitar la conexión del sensor ACS712 a la línea, se ha creado un montaje que permite esta conexión sin que sea necesario desmontar ni acceder a la toma en sí que se quiere monitorizar. Así, se permite que el acceso al sensor sea más sencillo, al igual que su conexión y montaje.

Figura 9. Montaje para la conexión del sensor a la línea.

Además de a la propia línea sobre la que se quiere medir, el sensor ACS712 también estará conectado a la placa OpenMote. Por un lado, la salida del sensor se conecta a uno de los pines de entrada de la placa OpenMote CC2538, que dispone de un convertidor analógico-digital (CAD), que le permitirá leer y convertir esta entrada. Pero además de la conexión con la línea y con la entrada de la placa OpenMote, el sensor debe ser alimentado, para lo cual se utiliza una pila y la propia tensión que genera la placa OpenMote. Así, se obtienen los aproximadamente 5V que necesita el sensor. En la siguiente imagen se muestra un esquema de la conexión de este bloque:

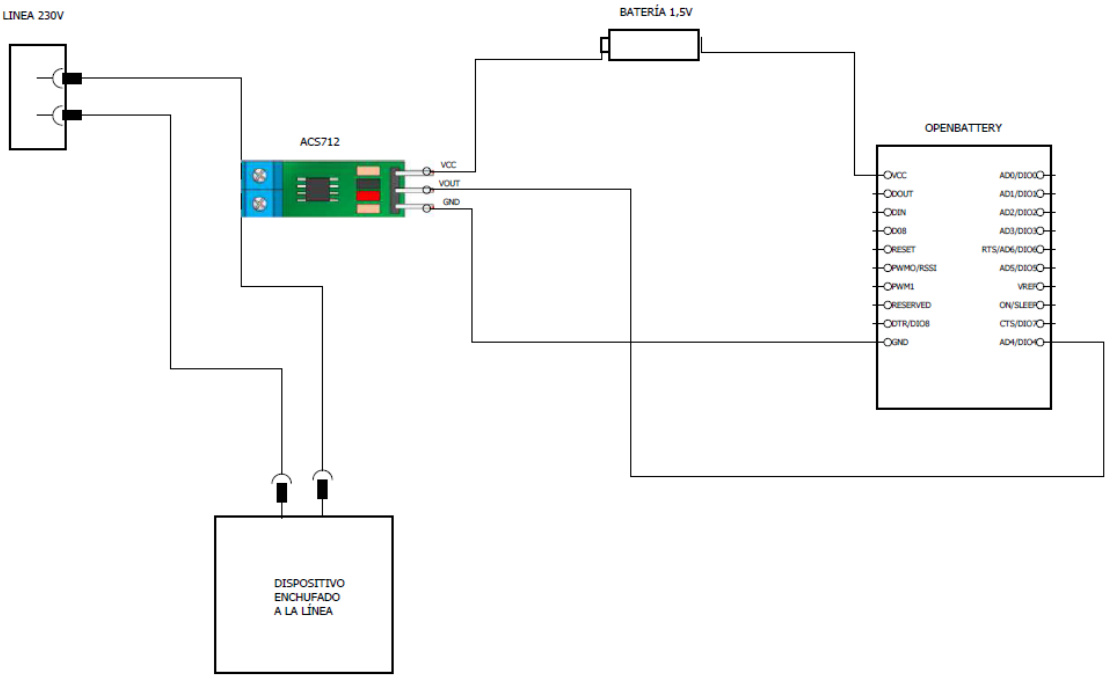


Figura 10. Esquema del conexionado eléctrico del sistema

Figura 11. Montaje de prueba del bloque sensor.

El resto de los elementos hardware del sistema, como ya se ha comentado anteriormente, consisten en un ordenador portátil y una placa OpenMote CC2538 montada en un módulo OpenBase conectado a este portátil. Esta placa no necesitará alimentación adicional, al estar

alimentada a través de su conexión al portátil. En la siguiente imagen se puede ver el montaje de prueba completo, donde la corriente que se está midiendo es la consumida por el ordenador portátil enchufado al montaje del bloque sensor.

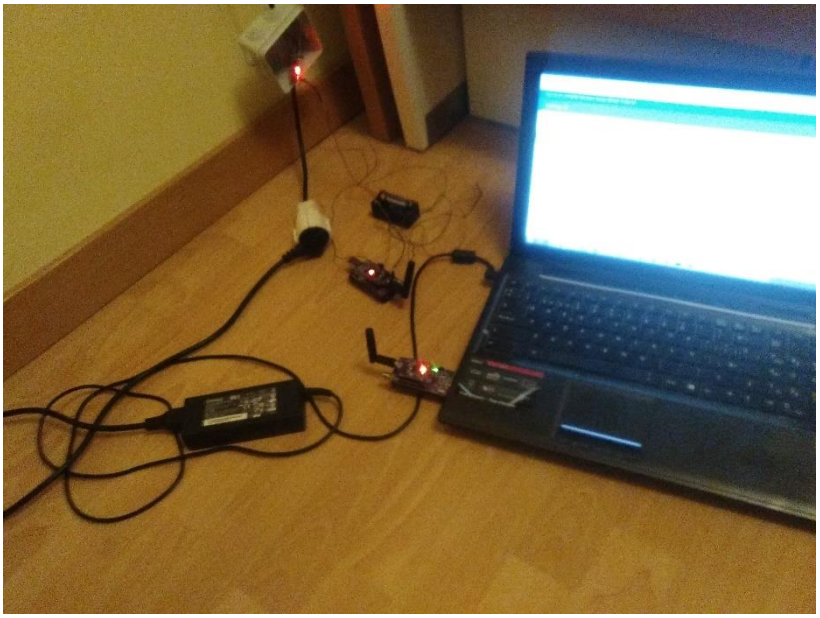


Figura 12. Montaje de prueba completo.

Como ya se ha mencionado previamente, el objetivo en este TFM es el desarrollo de un sistema prototipo, que sirva como demostrador de la idea propuesta de este medidor de consumo. Si a partir de este prototipo se decidiera desarrollar un sistema plenamente funcional, además del aspecto de poder monitorizar varias tomas de corriente mediante el empleo de varios bloques sensores, habría que emplear otro dispositivo diferente al ordenador portátil como bloque de procesamiento, en búsqueda de un menor coste y consumo, y una mayor portabilidad. Ejemplos de estos posibles dispositivos podrían ser una Raspberry Pi o un Smartphone.

También debería hacerse más compacto y de dimensiones más reducidas el módulo que se ha empleado en este prototipo para la conexión del sensor a la línea, totalmente funcional para esta fase de pruebas, pero quizás de tamaño excesivo para lo que sería necesario comparado con el tamaño del sensor.

3.4. CONFIGURACIÓN SOFTWARE

3.4.1.INTRODUCCIÓN

La configuración software del sistema va a ser algo más compleja que la hardware comentada anteriormente, ya que el empleo de diferentes plataformas en los diferentes elementos del sistema provocará que haya que emplear diferentes entornos de desarrollo y lenguajes de programación. Así, por ejemplo el firmware de las placas OpenMote estará programado en C, pero también se empleará el lenguaje python en la aplicación contenida en el ordenador portátil.

A nivel software, la arquitectura del sistema se puede dividir en dos bloques fundamentales:

- Firmware OpenWSN que se cargará en las placas OpenMote.
- Aplicación software del bloque de procesamiento central.

El funcionamiento del sistema consistirá en que desde el bloque de procesamiento se pedirá de manera periódica una medida al bloque sensor, que la llevará a cabo y se la transmitirá. Una vez recibida esta medida, el bloque de procesamiento la procesa adecuadamente y llevará a su vez su retransmisión a la plataforma theThings.io. En la siguiente figura se puede observar esquemáticamente este proceso:

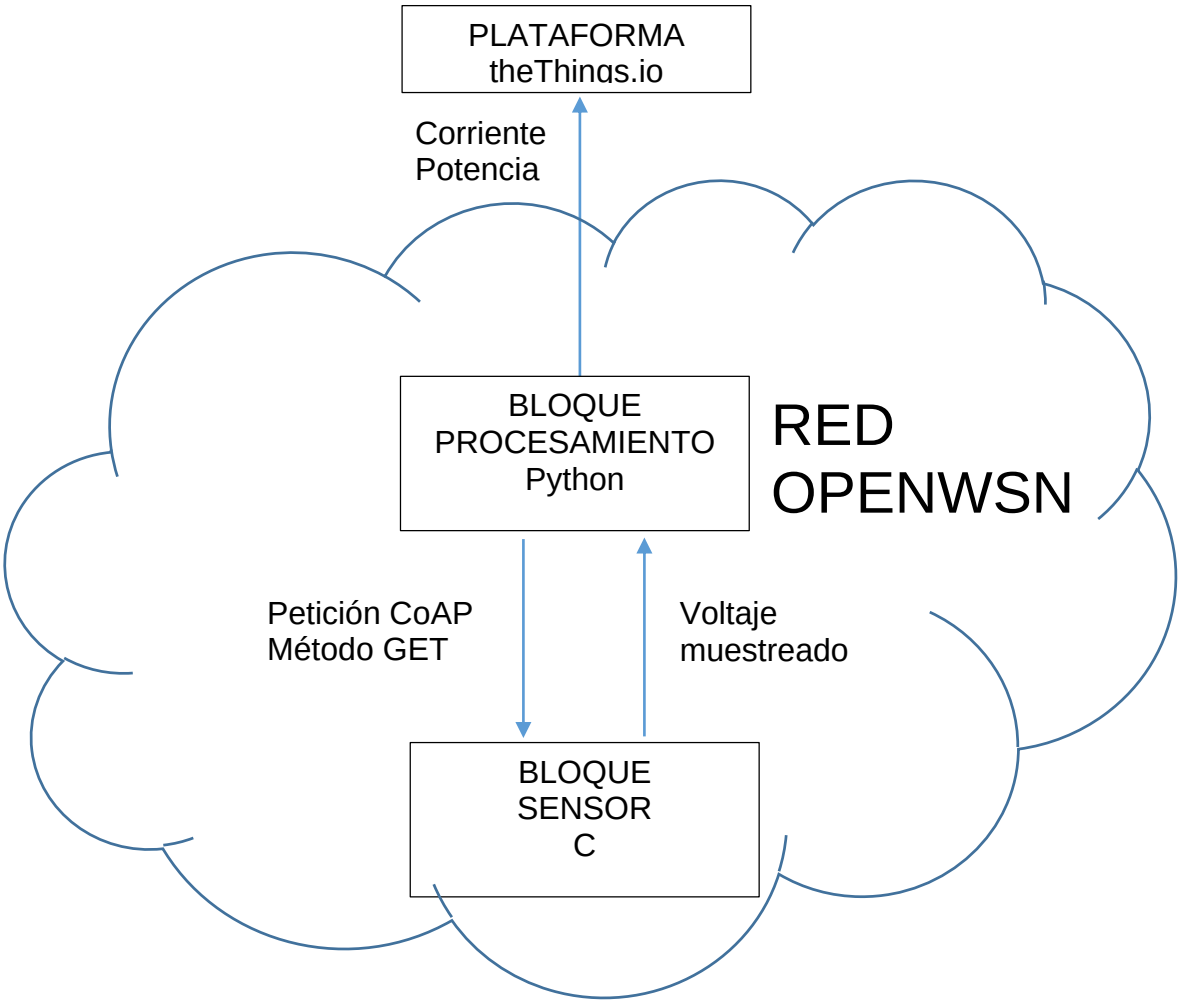


Figura 13. Arquitectura software del sistema desarrollado.

La comunicación entre los dos bloques se llevará a cabo mediante el procolo CoAP. Así, desde el bloque de procesamiento se enviará una petición CoAP hacia la OpenMote CC2538 del bloque sensor, con el método GET, y haciendo referencia al recurso que se quiere consultar. En este caso ese recurso es el valor de voltaje facilitado por el sensor ACS712 y adquirido por la placa OpenMote gracias a su convertidor analógico-digital. Para direccionar esta petición COAP se empleará la dirección ipv6 de la placa OpenMote del bloque sensor, que se supone conocida.

```
p = c.GET("coap://[bbbb::0012:4b00:060d:7ffb]/volt")
```

La placa OpenMote CC2538 del bloque sensor estará continuamente escuchando, y cuando reciba una petición por parte del bloque de procesamiento llevará a cabo la captura de las medidas procedentes del sensor ACS712, retransmitiéndoselas a continuación al bloque de procesamiento.

Una vez recibidos los datos del bloque sensor, éstos son procesados y transmitidos a la plataforma theThings.io, donde se almacenarán. Los datos enviados serán la corriente medida y la potencia consumida asociada a esta corriente.

3.4.2.HERRAMIENTAS TIC EMPLEADAS

En el proceso de desarrollo del prototipo objetivo de este TFM se han empleado diversas herramientas TIC, necesarias bien durante la propia fase de desarrollo o para el funcionamiento final del sistema. En este apartado se pretende recoger una lista de todas ellas.

- **Entorno de desarrollo Eclipse:**

http://www.eclipse.org/downloads/packages/eclipse-ide-cc-developers/mars2

A la hora de llevar a cabo la modificación del firmware de OpenWSN a cargar en las placas, se ha empleado este entorno para programar, compilar y cargar en las placas el código. Mediante esta herramienta se consigue de cierta manera facilitar y simplificar este proceso, sin necesidad de tener que acudir al uso de terminales ni comandos. Además, ya se tenía experiencia en su manejo anteriormente a este TFM, razón que también ha pesado en su elección.

Para llevar a cabo esta compilación del código y su carga en la placa, será preciso integrar en este entorno otras herramientas que permitan estas tareas, SCONS Y JLink.

- **SCONS**

http://scons.org/pages/download.html

Se trata de una herramienta de construcción de software, escrita totalmente en python, que permitirá llevar a cabo la compilación del código en la modificación del firmware de OpenWSN. Se dispone de tutorial e información de la herramienta suficiente en la web del proyecto OpenWSN [12], [13].

- **JLink**

https://www.segger.com/downloads/jlink

Esta herramienta permite la carga del firmware modificado de OpenWSN en las placas OpenMote, empleando además para ello la interfaz JTAG, que es la que se ha suministrado para el TFM y la razón de que se haya empleado esta herramienta.

Figura 14. Interfaz JTAG para la carga del código en las placas OpenMote

- **Tortoisegit**

https://tortoisegit.org/

Esta herramienta es un cliente GIT que permitirá la descarga desde Github de diferentes repositorios necesarios para el desarrollo de este sistema, como el firmware de OpenWSN por ejemplo. La elección de este cliente en concreto ha venido determinada por su utilización en el tutorial sobre OpenWSN contenido en la web del proyecto OpenWSN en atlassian.net [14].

- **Firmware OpenWSN**

https://github.com/openwsn-berkeley/openwsn-fw

Repositorio que contiene el firmware original de OpenWSN, sobre el que se trabajará para añadir las nuevas prestaciones que requiere el prototipo desarrollado en este TFM. Se descargará a través de la herramienta tortoisegit.

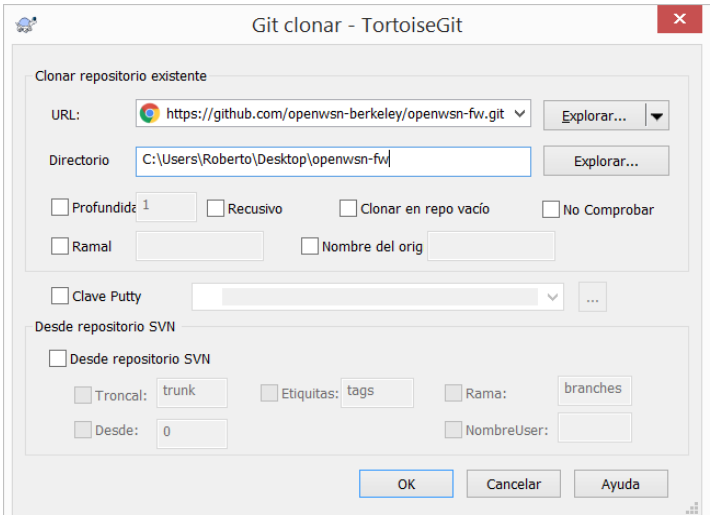


Figura 15. Descarga del repositorio del firmware OpenWSN a través de tortoisegit.

- **Software OpenWSN**

https://github.com/openwsn-berkeley/openwsn-sw

Repositorio que contiene diferentes herramientas de utilidad para el proyecto. Por ejemplo, aquí se encontrará la aplicación OpenVisualizer, descrita más adelante y que será necesaria para la creación de la red OpenWSN entre las placas OpenMote. Este repositorio también se descargará a través del cliente GIT tortoisegit.

- **Módulo python del protocolo CoAP**

https://github.com/openwsn-berkeley/coap

La aplicación software del bloque de procesamiento estará escrita en python, y una de las funciones de esta aplicación será la comunicación con el bloque sensor empleando el protocolo CoAP. Para ello, se empleará este módulo.

- **API theThings.io en python**

https://github.com/theThings/thethings.iO-python-library

Repositorio en GitHub que contiene una API en python para comunicarse con la plataforma theThings.io. Se empleará en la aplicación python del bloque de procesamiento para trasmitir los datos de consumo a dicha plataforma.

- **OpenVisualizer**

Como se ha comentado anteriormente, esta aplicación se puede encontrar dentro del repositorio software de OpenWSN descrito anteriormente, con lo que su descarga se realizará a través de la de este repositorio mediante tortoisegit. Esta aplicación puede ser empleada con diferentes interfaces: web, consola e interfaz gráfica.

La aplicación OpenVisualizer permite crear una red OpenWSN entre las placas OpenMote, lo cual será imprescindible en la implementación del prototipo que se pretende desarrollar en este TFM. Además de la creación, la herramienta permite la monitorización de esta red formada.

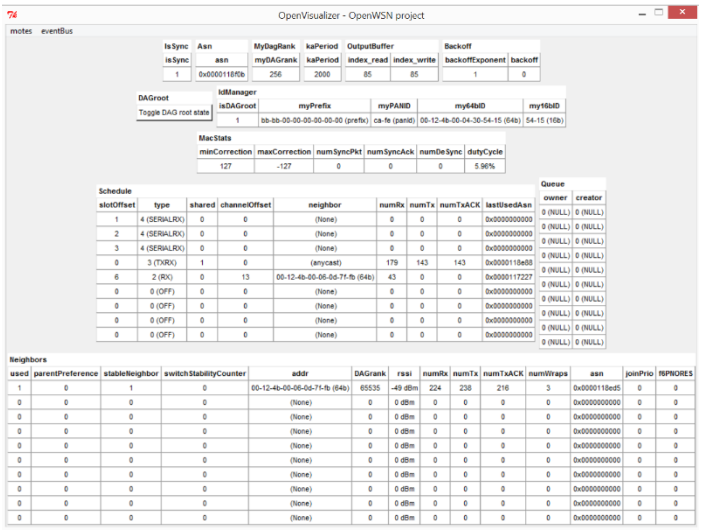


Figura 16. OpenVisualizer en su formato de interfaz gráfica.

- **PythonWin**

Entorno de desarrollo de python, que se ha empleado para la programación y depuración de la aplicación python del bloque de procesamiento. Se ha elegido esta herramienta por su facilidad tanto de instalación y configuración, como de empleo.

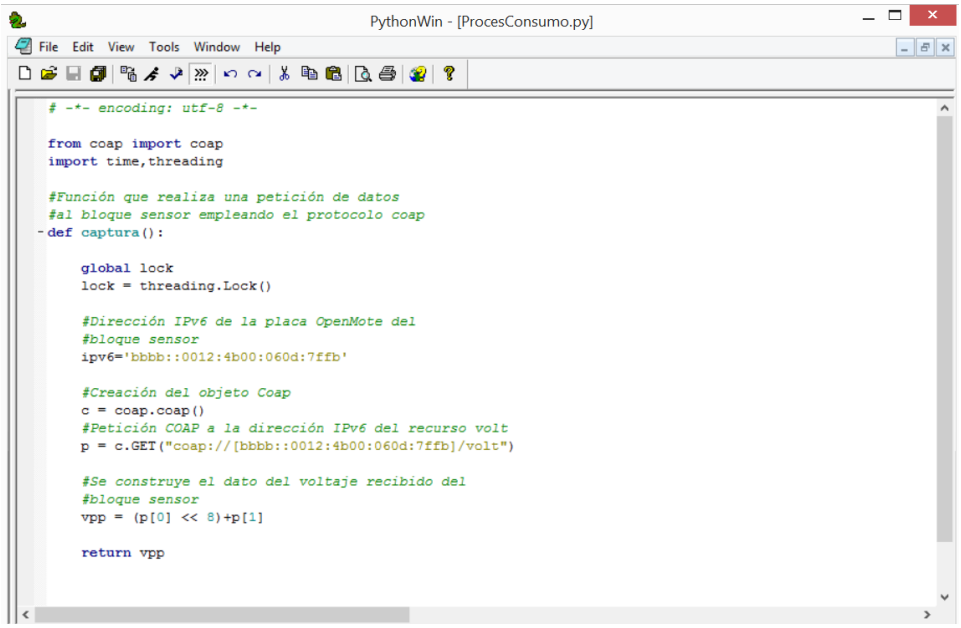


Figura 17. Interfaz de herramienta PythonWin

3.4.3.FIRMWARE OPENWSN

Como ya se ha comentado anteriormente, una de las partes de desarrollo software del sistema, consistirá en la modificación del firmware de OpenWSN que se cargará en las placas OpenMote. Este firmware está programado en lenguaje C, y para llevar a cabo esta modificación se ha elegido el entorno de desarrollo Eclipse.

El trabajo con el firmware de openWSN puede ser dividido en varias etapas:

- Descarga del repositorio del firmware de Openwsn de GitHub.
- Configuración del entorno de desarrollo eclipse para la correcta compilación y carga en las placas del firmware.
- Desarrollo, compilación y carga en las placas OpenMote del firmware modificado.

3.4.3.1. CONFIGURACIÓN PREVIA

Como ya se describió en el apartado dedicado a las herramientas TIC, para llevar a cabo la descarga de todos los respositorios de GitHub empleados en este proyecto se ha empleado el cliente GIT tortoisegit. Uno de estos repositorios es el firmware de OpenWSN.

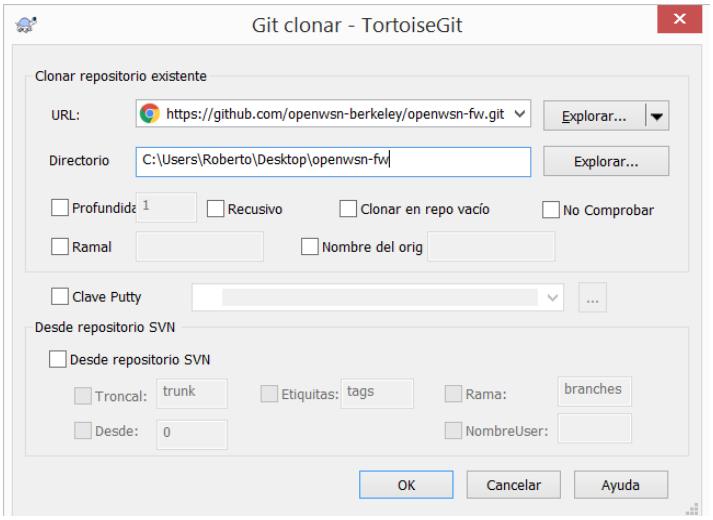


Figura 18. Configuración de la descarga en el cliente Git

Una vez descargado el firmware de OpenWSN e importado en Eclipse, el siguiente paso será configurar las diferentes herramientas de compilación y carga en las placas en este entorno de desarrollo. Como ya se ha mencionado anteriormente en esta memoria, como herramienta para la construcción/compilación del software se empleará SCONS, una herramienta escrita en python que permitirá llevar a cabo estas tareas. Así, en la pestaña Project>Properties, y en el menú C/C++ Build, se muestra:

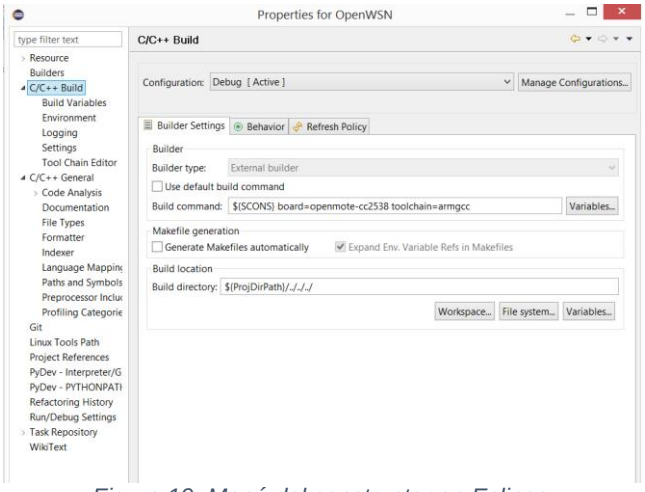


Figura 19. Menú del constructor en Eclipse.

Como se puede observar, el constructor hace referencia a una variable SCONS, que indicará la ruta hasta el ejecutable de SCONS.

Una vez compilado el software, ya se podrá llevar a cabo su carga y depuración. Para ello, se empleará JLink, utilizando la interfaz JTAG.

Así, en la pestaña Run>Debug Configurations, aparecerá un menú GDB SEGGER J-Link Debbuging, y en el una configuración para la carga y depuración de OpenWSN. Esta configuración será la que habrá que seleccionar a la hora de cargar el firmware de OpenWSN

modificado. En esta configuración, en el menú Main habrá que seleccionar el proyecto que se desea cargar, además del archivo ejecutable que se creó previamente en la compilación:

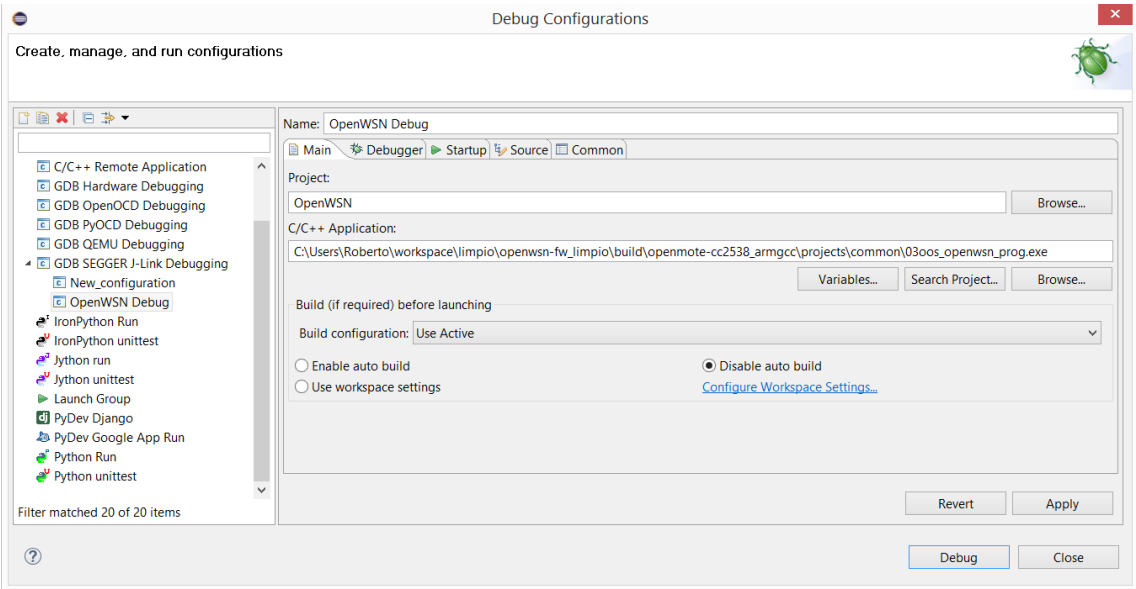


Figura 20. Configuración de la ruta de depuración.

En el menú Debugger, se pueden encontrar los parámetros empleados en la carga y depuración de esta configuración:

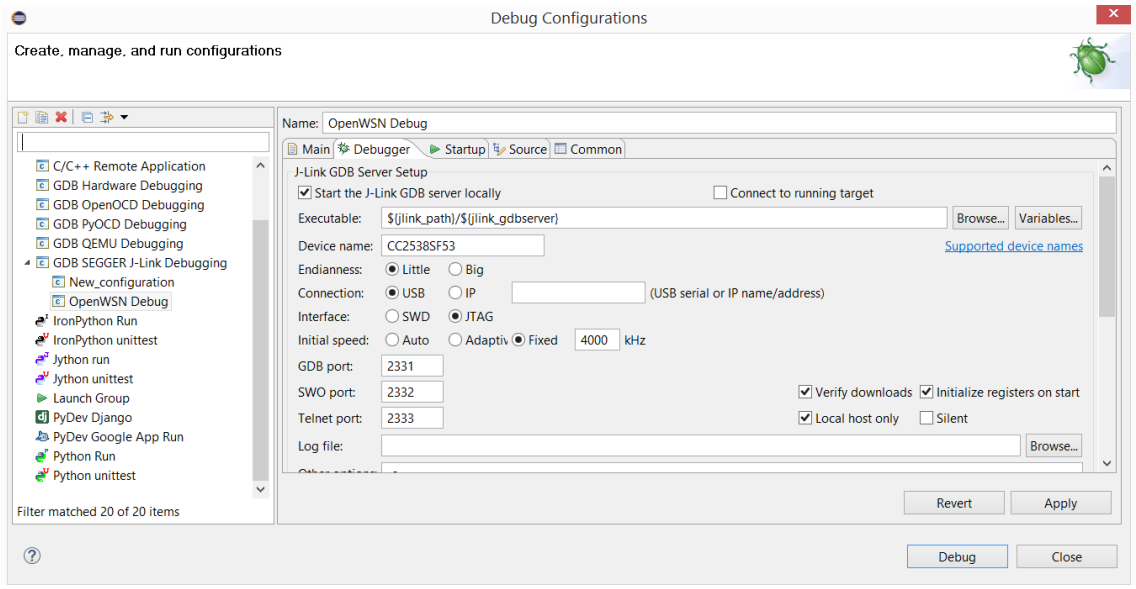


Figura 21. Parámetros del depurador.

Como se puede observar, está seleccionada la interfaz JTAG que se empleará para la carga, además de indicar el nombre específico del dispositivo en el que se efectúa la carga, la placa OpenMote CC2538.

3.4.3.2. MODIFICACIÓN *DEL FIRMWARE OPENWSN*

El objetivo de la modificación del firmware original de OpenWSN será proporcionar a la placa OpenMote CC2538 la capacidad de tomar lecturas de la medida de corriente realizadas por el sensor ACS712, y de transmitírselas a la aplicación instalada en el bloque de procesamiento central para que las procese y las envíe a la plataforma theThings.io. Para ello, lo que se va a hacer es añadir una aplicación adicional al firmware del OpenWSN que sea capaz de llevar a cabo estas tareas. Así, el trabajo a desarrollar se puede dividir en dos partes fundamentales. Por un lado estará la configuración y desarrollo del software necesario para la captura de los datos de medida por parte de la placa OpenMote CC2538, y por otro la implementación de la comunicación entre esta placa del bloque sensor y el bloque central de procesamiento, al que deberá transmitir estas medidas de corriente tomadas.

Para llevar a cabo esta comunicación se empleará el protocolo CoAP, ya implementado en el firmware de OpenWSN, por lo que la aplicación desarrollada que se pretende añadir al firmware, debe ser creada en torno a este protocolo. Para ello, además de la información contenida en la página **https://openwsn.atlassian.net/wiki/,** [15], [16], se ha utilizado como referencia las aplicaciones CoAP ya incluidas en el firmware OpenWSN, como por ejemplo la aplicación **cinfo**, que permite obtener información de una placa a través de comandos COAP.

Así, la nueva aplicación que se va a incorporar al firmware de OpenWSN recibirá el nombre de cvolt, y estará compuesta por un archivo **cvolt.c** y otro **cvolt.h**:

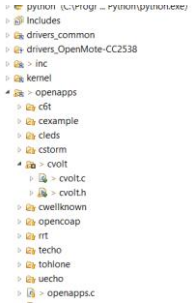


Figura 22. Aplicación cvolt añadida a la lista de aplicaciones de OpenWSN

La estructura que se ha seguido para construir el archivo **cvolt.c** será la misma que la del ejemplo **cinfo.c** que se ha tomado como referencia. Así, en primer lugar tendremos la definición de las diferentes librerías necesarias:

```
/**
cvolt.c  Aplicación que al recibir una petición COAT de tipo GET
con el recurso volt, toma muestra procedentes de la entrada analógica
y las trasmite como respuesta a esa petición
*/

#include "cvolt.h"
#include "opendefs.h"
#include "opencoap.h"
#include "openqueue.h"
#include "packetfunctions.h"
#include "openserial.h"
#include "openrandom.h"
#include "board.h"
#include "idmanager.h"
#include "adc.h"

```

Figura 23. Definición de librerías en cvolt.c

El principal aspecto a considerar aquí ha sido la inclusión de la librería **adc.h**, empleada para configurar los diferentes aspectos del CAD de la placa CC2538, que será empleado para tomar los valores medidos por el sensor de corriente.

A continuación, se incluyen las diferentes definiciones y declaraciones de variables y métodos necesarios para la ejecución del código del archivo:

```
18 //=========================== defines ==========================================
19
20 const uint8_t cvolt_path0[] = "volt";
21
22 //=========================== variables ========================================
23
24 cvolt_vars_t cvolt_vars;
25
26 //=========================== prototypes ======================================
27
28 owerror_t     cvolt_receive(
29    OpenQueueEntry_t* msg,
30    coap_header_iht*  coap_header,
31    coap_option_iht*  coap_options
32 );
33 void          cvolt_sendDone(
34    OpenQueueEntry_t* msg,
35    owerror_t error
36 );
37 uint16_t SampleADC(void);
38 double getVPP(void);
39
```

Figura 24. Definición de variables y métodos en cvolt.c

Así, se puede observar como en primer lugar se define la constante “volt”, que será el nombre abreviado del recurso que se utilizará desde el bloque de procesamiento para llamar a esta aplicación CoAP. Como se comentó anteriormente, esta aplicación será llamada desde el bloque de procesamiento, mediante una petición CoAP con el método GET, de la forma:

```
p = c.GET("coap://[bbbb::0012:4b00:060d:7ffb]/volt")
```

Como se puede ver, en la petición CoAP se incluye la dirección IPv6 de la placa a la que se consulta, y a continuación el nombre del recurso, en este caso “volt”, definido en el archivo **cvolt.c** como se ha mostrado anteriormente.

Además, en la imagen anterior se pueden ver también las declaraciones de dos funciones que serán imprescindibles para el funcionamiento de la aplicación, y que serán detalladas más adelante: **SampleADC**(), que define los parámetros y funcionamiento del CAD de la placa CC2538, y **getVPP()**, que toma muestras de la señal proporcionada por el sensor de corriente, llamando precisamente a **SampleADC()**, y calcula el valor pico a pico de la señal medida.

A continuación, una vez incluidas todas las declaraciones necesarias, en el archivo **cvolt.c** se van a definir todas las funciones que se emplean en la aplicación. En total van a ser 5 funciones:

➢ Función cvolt_init()

```
43 \brief Initialize this module.
45 void cvolt_init() {
46    // do not run if DAGroot
47    if(idmanager_getIsDAGroot()==TRUE) return;
48
49    // prepare the resource descriptor for the /i path
50    cvolt_vars.desc.path0len             = sizeof(cvolt_path0)-1;
51    cvolt_vars.desc.path0val             = (uint8_t*)(&cvolt_path0);
52    cvolt_vars.desc.path1len             = 0;
53    cvolt_vars.desc.path1val             = NULL;
54    cvolt_vars.desc.componentID          = COMPONENT_CVOLT;
55    cvolt_vars.desc.discoverable         = TRUE;
56    cvolt_vars.desc.callbackRx           = &cvolt_receive;
57    cvolt_vars.desc.callbackSendDone     = &cvolt_sendDone;
58
59    // register with the CoAP module
60    opencoap_register(&cvolt_vars.desc);
61 }
```

Figura 25. Definición de función cvolt_init en cvolt.c

Esta función es la que se va a llamar cuando se inicie OpenWSN, y que va a servir para definir una serie de propiedades y variables necesarias CoAP, además de apuntar a dos funciones que se describirán a continuación y que serán esenciales en el funcionamiento de la aplicación: **cvolt_receive()** y **cvolt_sendDone()**.

Para que esta función se llame al iniciarse OpenWSN, será necesario incluir su llamada en el archivo **openapps.c**, que efectivamente incluye las llamadas a las aplicaciones que se iniciarán con OpenWSN:

```
31
32 void openapps_init(void) {
33    //-- 04-TRAN
34    opencoap_init();     // initialize before any of the CoAP applications
35
36    // CoAP
37    c6t_init();
38    cvolt_init();
39    cleds__init();
40    cstorm_init();
41    cwellknown_init();
42
43    // TCP
44    techo_init();
45 }
```

Figura 26. Llamada a las funciones en el arranque de OpenWSN.

Además, también será necesario registrar el identificador unívoco **COMPONENT_CVOLT** en el archivo **opendefs.h**:

```
155   // applications
156      COMPONENT_C6T                    = 0x1a,
157      COMPONENT_CEXAMPLE               = 0x1b,
158      COMPONENT_CVOLT                  = 0x1c,
159      COMPONENT_CLEDS                  = 0x1d,
160      COMPONENT_CSTORM                 = 0x1e,
161      COMPONENT_CWELLKNOWN             = 0x1f,
162      COMPONENT_TECHO                  = 0x20,
163      COMPONENT_TOHLONE                = 0x21,
164      COMPONENT_UECHO                  = 0x22,
165      COMPONENT_RRT                    = 0x23,
166      COMPONENT_SECURITY               = 0x26,
167      COMPONENT_USERIALBRIDGE          = 0x27,
```

Figura 27. Registro de indentificador cvolt

Se puede comprobar como esta función sólo inicia sus componentes en caso de que la placa CC2538 no sea DAGroot, es decir, sólo si es la placa del bloque sensor.

- **Función cvolt_receive():**

```
owerror_t cvolt_receive(
      OpenQueueEntry_t* msg,
      coap_header_iht* coap_header,
      coap_option_iht* coap_options
   ) {

   owerror_t outcome;

   switch (coap_header->Code) {
      case COAP_CODE_REQ_GET:

         //=== reset packet payload (we will reuse this packetBuffer)
         msg->payload                     = &(msg->packet[127]);
         msg->length                      = 0;

         //=== prepare  CoAP response
         packetfunctions_reserveHeaderSize(msg, 3);
                msg->payload[0] = COAP_PAYLOAD_MARKER;
                uint16_t volt = 0x0;

         /*Se llama a la función getVPP, que devuelve un valor
         de tensión correspondiente a la corriente medida por
         el sensor*/
         volt = getVPP();

         // return as big endian
                msg->payload[1] = (uint8_t)(volt >> 8) ;
                msg->payload[2] = (uint8_t) (volt & 0xff);

         // set the CoAP header
         coap_header->Code                = COAP_CODE_RESP_CONTENT;

         outcome                          = E_SUCCESS;
         break;
      default:

          // return an error message
         outcome = E_FAIL;
   }

   return outcome;
}
```

Figura 28. Definición de función cvolt_receive en cvolt.c

Esta función va a ser la que se ejecute cuando se reciba una petición CoAP por parte del bloque de procesamiento. Se puede comprobar como sólo está prevista para manejar peticiones CoAP de tipo GET, que son precisamente las únicas que se esperan recibir por parte de aquel bloque. En ese caso, lo que se hace es preparar el paquete CoAP de respuesta, y llamar a la función **getVPP()**, que devolverá el valor pico a pico de la señal medida por el sensor de corriente.

- **Función getVPP():**

```
double getVPP()
{
    double result;


    int j=0;
    uint16_t readValue=0;
    long media=0;
    long  suma=0;
    uint16_t maxValue = 0;
    uint16_t minValue = 4096;


    //Se toman un determinado de muestras de CAD, y se almacenan el valor max y mín
    while(j < 1000)
        {
            readValue = SampleADC();
            // see if you have a new maxValue
            if ((readValue > maxValue)&&j!=0)
                {

                    maxValue = readValue;

                }

            if ((readValue < minValue)&&j!=0)
                {

                    minValue = readValue;
                }


            j++;
       }

    //Obtiene la diferencia entre el valor máximo y mínimo
    result = maxValue - minValue;

    return result;
}
```

Figura 29. Definición de función getVPP en cvolt.c

El objetivo de esta función será, a partir de medidas recibidas por el ADC, calcular el valor pico a pico de la señal recibida. La señal que mide el sensor ACS712 no se trata de una señal continua, sino la corriente alterna de la línea, que tendrá por lo tanto forma senoidal. Por lo tanto la señal de salida del sensor se comportará de la misma manera, y será necesario muestrearla para determinar sus valores máximo y mínimo que permitan calcular la amplitud pico a pico de esta señal ofrecida por el sensor. Esta es precisamente la tarea que lleva a cabo esta función **getVPP()**, que como se puede observar llama un determinado número de veces a la función **SampleADC()**, que devuelve un valor tomado del sensor, y almacena el valor máximo y mínimo de estas lecturas. A continuación los resta para obtener el valor pico a pico.

- **Función SampleADC()**

```
/*Función que configura el Convertidor Analógico Digital de la placa CC2538*/
uint16_t  SampleADC(void){
    uint16_t ui1Dummy;
    //
    // Configure ADC, External reference VDD5, 512 decimation rate (12bit)
    //
    SOCADCSingleConfigure(SOCADC_12_BIT, SOCADC_REF_AVDD5);
    //
    // Trigger single conversion on AIN2
    //
    SOCADCSingleStart(SOCADC_AIN2);
    //
    // Wait until conversion is completed
    //
    while(!SOCADCEndOfCOnversionGet())
    {
    }
    //
    // Get data and shift down based on decimation rate
    //
    ui1Dummy = SOCADCDataGet()>> SOCADC_12_BIT_RSHIFT;

    return ui1Dummy;
}
```

Figura 30. Definición de función SampleADC en cvolt.c

Esta función configura diferentes parámetros del convertidor analógico-digital con el que cuenta la placa CC2538, y realiza una conversión. Así, se puede observar como se configura la resolución del CAD a 12 bits, que como tensión de referencia se establece la tensión VDD de aproximadamente 3,3V, y además se establece como entrada analógica el pin AIN2, que se corresponde con el pin AD4/DIO4 de la OpenMote CC2538.

Además de configurar este pin de entrada al CAD, habrá que configurar este pin como entrada analógica en la placa OpenMote CC2538. Para ello, habrá que modificar de manera adecuada la función **gpio_init():**

```
142 static void gpio_init(void) {
143     
144 
145             GPIOPinWrite(GPIO_A_BASE, 0xFF, 0x00);
146             GPIOPinTypeGPIOInput(GPIO_A_BASE, 0xFF);
147             GPIODirModeSet(GPIO_A_BASE, GPIO_PIN_2, GPIO_DIR_MODE_IN);
148             OCPadConfigSet(GPIO_A_BASE, GPIO_PIN_2, IOC_OVERRIDE_ANA);
149 
150 
```

Figura 31. Configuración de entrada analógica para CAD

Como se puede ver, en primer lugar se define como entrada el puerto A, y después más específicamente el pin 2 de este puerto, que se corresponde con la entrada AD4/DIO4 de la placa OpenMote CC2538, pin que se utilizará como entrada de la señal enviada por el sensor ACS712. A continuación, se define este pin como entrada analógica.

Función **cvolt_sendDone()**

```
124 void cvolt_sendDone(OpenQueueEntry_t* msg, owerror_t error) {
125    openqueue_freePacketBuffer(msg);
126 }
127 
```

Figura 32. Definición de función sendDone en cvolt.c

Esta función es llamada cuando ha sido llevada a cabo la transmisión de un paquete, con la finalidad de liberar recursos.

3.4.4.APLICACIÓN PYTHON BLOQUE PROCESAMIENTO CENTRAL

Como se ha comentado anteriormente, el bloque de procesamiento central estará compuesto por un ordenador portátil con sistema operativo Windows, y una placa OpenMote CC2538 montada sobre un módulo OpenBase, que estará conectada a dicho ordenador.

En este ordenador portátil estará contenida la otra aplicación software que forma parte del sistema además del firmware de las placas OpenMote, y que consistirá en una aplicación programada en lenguaje Python. Se ha elegido este lenguaje ya que dispone de librería COAP, y demás existe suficiente documentación para poder llevar a cabo el desarrollo de la aplicación deseada **[15]**.

Esta aplicación va a cumplir varias funciones. En primer lugar, realizará consultas periódicas al bloque sensor para obtener valores medidos por éste, consultas efectuadas a través del protocolo CoAP con el método GET. Una vez recibidos los datos por parte del bloque sensor, los procesará para obtener valores reales de corriente y potencia consumida, que son los datos que realmente interesan al usuario. A continuación, y como última función de este bloque, transmitirá estos valores a la plataforma theThings.io, donde quedarán almacenados. En el siguiente diagrama se puede ver la composición de esta aplicación software:

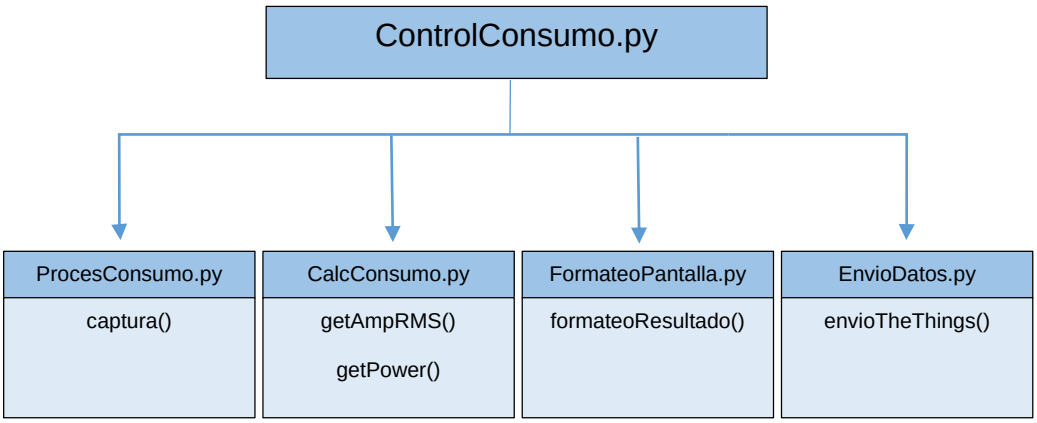


Figura 33. Estructura de la aplicación del bloque procesador.

La aplicación va a estar compuesto por 5 ficheros diferentes, todos ellos correspondientes a lenguaje python y por lo tanto con extensión .py. El fichero principal de la aplicación será **ControlConsumo.py**, desde el que se inicia y se gestiona el flujo de toda ella. Así, en este fichero se llevan a cabo las llamadas a todas las funciones empleadas en la aplicación y que estarán contenidas en otros ficheros, como se muestra en la figura anterior. A continuación se recoge el código de este fichero principal **ControlConsumo.py**:

```
import time,threading
import ProcesConsumo
from thethings import ThethingsAPI
import CalcConsumo
import FormateoPantalla
import EnvioDatos

i=0

#Tiempo entre capturas de datos
intervalo=30

#Creación de objeto de tipo ThethingsAPI que permitirá
#gestionar el envió de datos a la platafora theThings.io
thethings = ThethingsAPI("28sIAyQqH_7ZjlZxyvDBH-QfDxdVvKYAIbEpxrxWICY")

while True:
    i=i+1

    #Se define instante de inicio de captura de datos
    comienzoCaptura=time.time()

    #Se obtiene una medida del bloque sensor
    vpp=ProcesConsumo.captura()

    #Se obtienen los valores de corriente y potencia a partir
    #del valor proporcionado por el bloque sensor
    current=CalcConsumo.getAmpRMS(vpp)
    power=CalcConsumo.getPower(current)

    #Se formatean los datos para mostrarlos por la consola
    FormateoPantalla.formateoResultados(i, current, power)

    #Se envían los datos a la plataforma theThings.io
    EnvioDatos.envioTheThings(thethings, current, power)

    #Se define el instante final de la captura de datos
    finalCaptura= time.time()

    #Duración de la captura
    duracionCaptura= finalCaptura-comienzoCaptura


    #Si la captura dura menos que el tiempo definido se introduce
    #un tiempo de espera hasta completar el intervalo entre capturas
    if (duracionCaptura<intervalo):
        espera=intervalo-(duracionCaptura%30)
        time.sleep(espera)
```

Figura 34. Código de ControlConsumo.py

Como se puede observar, este fichero representa el núcleo de la aplicación del bloque procesador, gestionando todo el proceso desde que se piden los datos de una nueva captura al bloque sensor, hasta que esos datos, ya procesados, se envían a la plataforma theThings.io para su almacenamiento. Para ello, se realizarán varias llamadas a diferentes funciones que se encargarán de las tareas necesarias en todo este proceso. Todo ello estará implementado

mediante un bucle while que permitirá que este funcionamiento se ejecute de manera indefinida.

En el código, tras las definiciones necesarias que posibilitan la llamada a las diferentes funciones empleadas en el código, lo primero que aparece es la declaración de varias variables. Por un lado, se declara e inicializa una variable **i** de tipo entero que servirá para llevar la cuenta de las capturas realizadas, así como otra variable, también de tipo entero, que define la duración del intervalo entre capturas, la variable **intervalo.** En este caso está inicializada a un valor de 30 segundos, de manera que cada medio minuto se llevará a cabo una petición de medida al bloque sensor.

Por último, la última definición corresponde a la creación de un objeto de la clase **TheThingsApi**, el objeto **thethings**. Esta clase será la que permita y gestione la comunicación con la plataforma theThings.io, y se ha obtenido del repositorio en Github de dicha plataforma [17].

Una vez ya dentro del bucle while, lo primero que se lleva a cabo es el almacenamiento del instante en el que se inicia la captura de un nuevo dato de medida desde el bloque sensor, lo que servirá para monitorizar el intervalo de tiempo entre capturas. A continuación, se realiza la captura, es decir, la petición de una medida al bloque sensor. Para ello se llama a la función **captura()**, dentro del fichero **ProcesConsumo.py:**

```
from coap import coap
import time,threading

#Función que realiza una petición de datos
#al bloque sensor empleando el protocolo coap
def captura():

    global lock
    lock = threading.Lock()

    #Dirección IPv6 de la placa OpenMote del
    #bloque sensor
    ipv6='bbbb::0012:4b00:060d:7ffb'

    #Creación del objeto Coap
    c = coap.coap()
    #Petición COAP a la dirección IPv6 del recurso volt
    p = c.GET("coap://[bbbb::0012:4b00:060d:7ffb]/volt")

    #Se construye el dato del voltaje recibido del
    #bloque sensor
    vpp = (p[0] << 8)+p[1]

    return vpp
```

Figura 35. Código del fichero ProcesConsumo.py

Se puede comprobar como en esta función **captura(),** empleando la dirección IPv6 de la placa OpenMote del bloque sensor, se realiza una petición CoAP con el método GET al recurso volt. Antes de realizar esta petición, se ha creado el objeto de tipo CoAP que la contendrá. Por último,

a partir de los datos recibidos en la respuesta CoAP desde el bloque sensor, se construye el valor que ha proporcionado el CAD de la placa OpenMote CC2538 de dicho bloque.

Volviendo al código del fichero principal, **ControlConsumo.py,** tras recibir una medida del bloque sensor, se realiza la llamada a las funciones **getAmpsRMS()** y **getPower()**, cuya tarea será obtener los valores de corriente y potencia que representan el dato devuelto por el CAD del bloque sensor. Ambas estarán contenidas en el fichero **CalcConsumo.py**:

```
# -*- encoding: utf-8 -*-

#Función que calcula los amperios eficaces consumidos por el dispositivo
#a partir del valor de voltaje pico a pico enviado por la OpenMote
def getAmpRMS(vpp):

    volt=(vpp * 3.3)/2047.0;
    vrms = (volt/2.0) *0.707;
    ampsRMS = (vrms * 1000)/185;
    return round(ampsRMS,3)

#Función que calcula los vattios de consumo a partir de la intensidad
#consumida por el dispositivo
def getPower(arms):
    pow = arms * 230
    return round(pow,3)
```

Figura 36. Código del fichero CalcConsumo.py

En primer lugar, se realiza la llamada a la función **getAmpRMS()**, pasándole como parámetro el valor recibido desde el bloque sensor. Como se analizó en el apartado dedicado a la modificación del firmware de OpenWSN para la captura de datos mediante el CAD de la placa OpenMote, el valor que se enviará desde el bloque sensor a este bloque de procesamiento será el valor de salida del CAD que se corresponde con la amplitud pico a pico de la señal de tensión medida por el sensor ACS712. Por ello, lo primero que se hace en esta función **getAmpRMS()** es obtener ese valor de tensión pico a pico que corresponde al valor devuelto por el CAD. Para ello emplea la tensión de referencia del CAD del 3,3V y su número de niveles, 2047.

Con esta tensión pico a pico calculada, el siguiente paso es obtener la tensión eficaz equivalente, dividiéndola entre 2 para obtener la amplitud, y multiplicando por la raíz de dos para obtener este valor eficaz.

Finalmente se obtiene el valor de corriente correspondiente a este valor de tensión, según la relación entre estas dos magnitudes que ofrece el sensor ACS712, y que en este caso, teniendo en cuenta que el sensor empleado corresponde al modelo de 5A, es de 185mV por cada A [18]. Este valor se redondeará y será el que devuelva la función, que se corresponde con la corriente eficaz medida en la toma de corriente

A continuación, se realiza la llamada a la función **getPower()**, pasándole como parámetro precisamente el valor devuelto por la función **getAmpRMS()**, es decir, la corriente eficaz medida. A partir de esta corriente, se calcula la potencia equivalente teniendo en cuenta la tensión de la red, de aproximadamente 230V, y que multiplicada por esta corriente proporcionará el consumo

de potencia asociado. Finalmente se redondea el valor de potencia obtenido, y este es el valor devuelto por la función.

Una vez obtenidos estos valores de consumo, la aplicación software del bloque de procesamiento llevará a cabo dos tareas adicionales, mostrar estos valores por la pantalla y transmitirlos a la plataforma theThings.io, donde quedarán almacenados. Así, en primer lugar se llama a la función **formateoResultados()** del fichero **FormateoPantalla.py**:

```
# -*- encoding: utf-8 -*-

#Función que muestra por pantalla los valores
#de consumo medidos
def formateoResultados(i, current, power):

    print "Medida " + str(i)
    print "************************************ "
    currentStr="Corriente: "
    potenciaStr="Potencia: "
    currentStr+=str(current)
    currentStr+=" A"
    potenciaStr+=str(power)
    potenciaStr+=" W"

    print currentStr
    print potenciaStr
    print "************************************ "
    print "------------------------------------"
    print "                "
```

Figura 37. Código del fichero FormateoPantalla.py.

Esta función simplemente recibe como parámetros el número de medición, y los valores de corriente y potencia medidos, y los muestra por pantalla con un determinado formato.

Tras esta función, en el fichero **ControlConsumo.py** se llama a la función **envioTheThings()**, del fichero **EnvioDatos.py**:

```
# -*- encoding: utf-8 -*-

#Función que prepara los datos a enviar a plataforma theThings.io
#y lleva a cabo el envío
def envioTheThings(thethings, current, power):

    thethings.addVar("Current", current)
    thethings.addVar("Power", power)

    thethings.write()
```

Figura 38. Código del fichero EnvioDatos.py

Mediante esta función se consigue completar la última etapa y por lo tanto el proceso completo del sistema, ya que se encarga del envío de los datos medidos a la plataforma theThings.io. Esta función recibe como parámetros el objeto **ThethingsAPI** creado en el fichero principal

ControlConsumo.py, y los valores de corriente y potencia correspondientes a la medida. Mediante el método **addVar**() de la clase **ThethingsAPI** se crea la lista de datos que se va enviar, definiéndolos como un par clave-valor. Una vez creada la lista de datos, se envían a la plataforma theThings.io mediante el método **write(),** que lleva a cabo este envío y elimina de la lista de datos pendientes los ya enviados.

Con la finalización de este envío a la plataforma theThings.io se completaría la totalidad del proceso realizado por el sistema, pero como se puede ver en el código del fichero ControlConsumo.py, tras la llamada a esta función aún se llevan a cabo algunas tareas más. Así, tras efectuarse este envío a la plataforma, se almacena el instante de tiempo en el que se ha completado dicho envío, y se calcula la duración total del proceso mediante este valor y el de inicio de la captura, que también había quedado almacenado. Con esta duración, se establece un tiempo de espera hasta la siguiente captura, de manera que se cumpla el intervalo temporal entre capturas que se haya definido, en este caso de 30 segundos.

3.4.5. PLATAFORMA THETHINGS.IO

3.4.5.1. CONFIGURACIÓN PLATAFORMA

La plataforma theThings.io se empleará para almacenar los datos medidos por el sistema, y que le serán transmitidos a través de la aplicación python instalada en el bloque de procesamiento, como se ha explicado anteriormente, utilizando para ello el protocolo COAP. Para poder emplear esta plataforma será necesario en primer lugar crear una cuenta y configurarla adecuadamente, teniendo en cuenta que una cuenta de prueba tiene una caducidad de 14 días, tras los cuales los datos almacenados se pierden.

El registro es muy sencillo e inmediato, tan sólo es necesario rellenar algunos datos básicos:

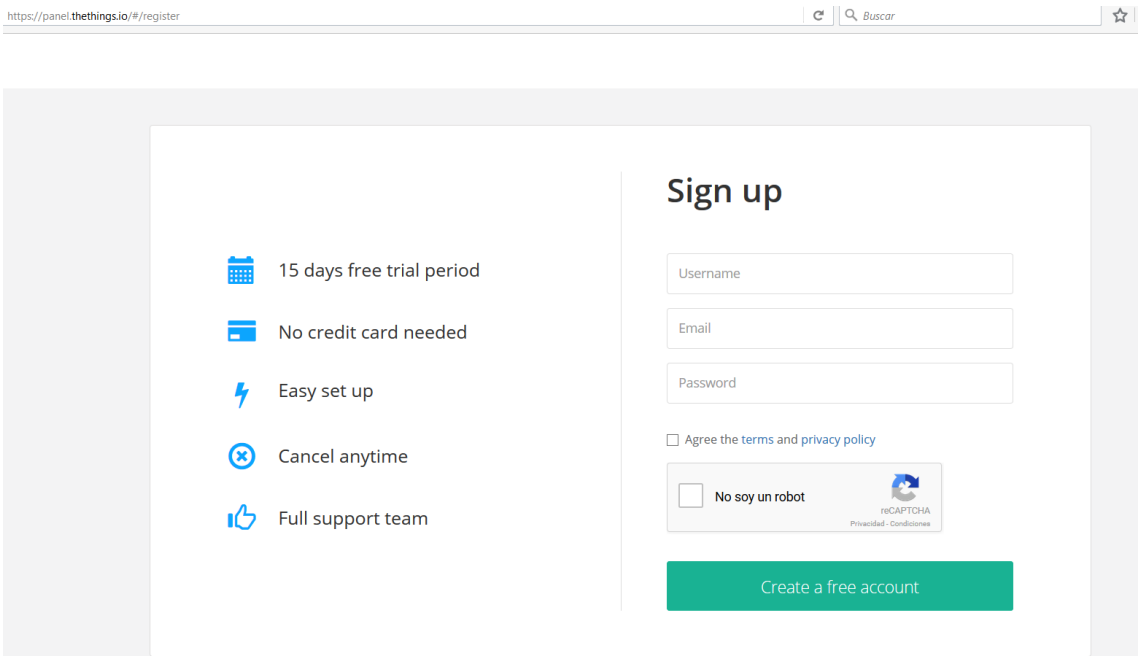


Figura 39. Pantalla de registro en theThings.io

Una vez confirmado el registro, ya se puede añadir un dispositivo o “cosa”, que podrá ser activado. La activación consiste en que este dispositivo podrá comunicarse con la plataforma para enviar y recibir datos, y con la cuenta de prueba se permiten hasta 3 activaciones.

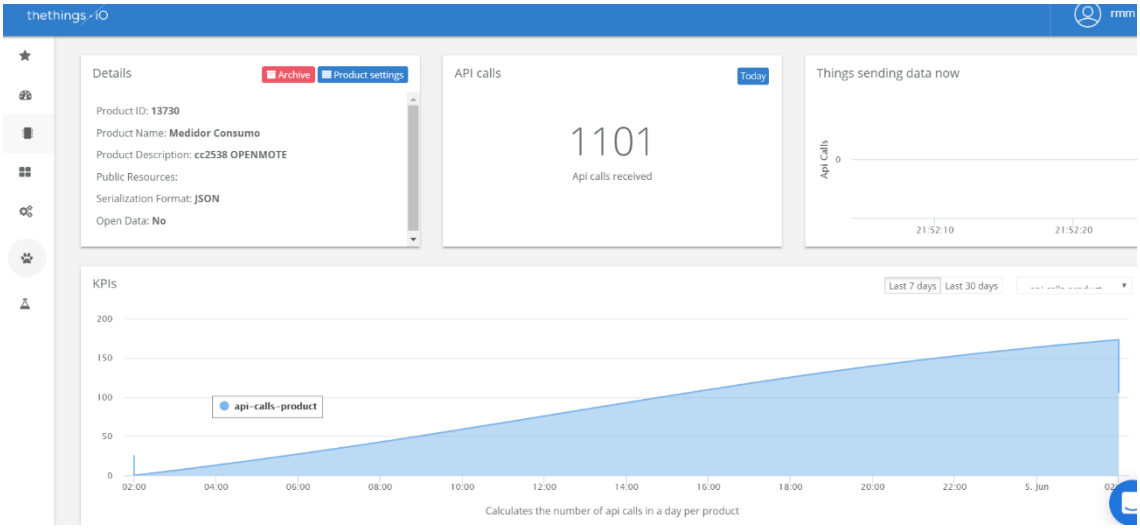


Figura 40. Pantalla de información de dispositivo,

Una vez activado un dispositivo ya se pueden enviar datos a la plataforma, y consultar los datos almacenados en ella.

Figura 41. Registro de datos en theThings.io

3.4.5.2. ALERTAS SMS

Una de las opciones que permite la plataforma theThings.io es establecer eventos que se disparen ante ciertas circunstancias. Así, en este TFM se ha probado la opción de que ante la

circunstancia de que el consumo de potencia supere un cierto umbral, se envíe al usuario un sms de aviso. Esto se puede conseguir mediante la configuración de un trigger en la plataforma:

```
1
2  /*
3    params: is an object with the keys:
4     - action: one of 'write' | 'read'
5     - thingToken: the thing that triggered the trigger
6     - values: only if action == 'write'. Is an array of values where each value is an object with:
7     - key: the key
8     - value: the data sent
9     - datetime: (can be null)
10
11   callback: is a function to be called when the trigger ends can contain a
12       parameter string "error" if the trigger needs to report an error.
13 */
14
15 function trigger(params, callback){
16   console.log('trigger triggered!!')
17   //ignore non write events
18   if(params.action !== 'write') return callback()
19
20   var values = params.values
21   var thingToken = params.thingToken
22
23   //iterate over the values of the write
24   for(var i=0; i<values.length; ++i){
25     if(values[i].key === 'Current' && values[i].value > 2){
26       console.log('omg too hot')
27       var telf = '+34617485764'
28
29       var message = 'Hay algo que está encendido! Dispositivo:' + thingToken
30
31       var twilio = new Twilio(
32         'ACe88217672e13ce30bd2da8caa41833b6',
33         '74c3390de9944e811b3688a096cca2d8'
34       )
35
36       twilio.sendMessage({
37           to: telf, // Any number Twilio can deliver to
38           from: '+34931071424', // A number you bought from Twilio and can use for outbound communication
39           body: message // body of the SMS message
40         },
41         callback
42       )
43     }
44   }
45   //end the trigger
```

Figura 42. Configuración de Trigger en la plataforma theThings.io

Como se puede ver en la figura anterior, en la línea 25 se define que para valores de corriente superiores a 2 amperios, se enviará un mensaje al teléfono indicado en la línea 27. El mensaje a enviar se define en la línea 29. Para conseguir este envío, es necesario tener un número de teléfono desde el que se envíe este sms. En este caso, se ha creado una cuenta en la herramienta Twilio [19], que con el registro ofrece gratuitamente un número de teléfono. Este número es indicado en la línea 38.

Cuando el consumo de corriente supere el valor marcado en el script anterior, se recibirá el sms en el teléfono indicado:

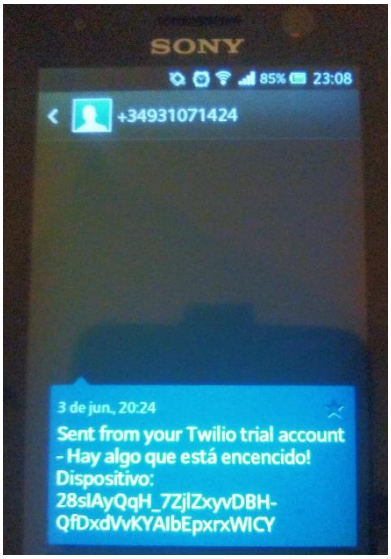


Figura 43. SMS enviado al dispararse el trigger en la plataforma theThings.io

3.5. PUESTA EN MARCHA Y FUNCIONAMIENTO DEL SISTEMA

A la hora de arrancar el prototipo será necesario tener en cuenta ciertas consideraciones y llevar cabo varias tareas para que el sistema funcione correctamente. A continuación se recoge el proceso que es necesario llevar a cabo para esta puesta en marcha:

➢ **Primer paso: Configuración red OpenWSN**

La red OpenWSN formada por las placas OpenMote es la base del funcionamiento del prototipo desarrollado. Por ello el primer paso para arrancar el sistema será asegurarse de que esta red está formada y los dispositivos sincronizados. Los pasos a seguir para ello serán:

- Conexión de la placa OpenBase al ordenador portátil
- Activación de la placa OpenBattery del bloque sensor. Esta activación, además de permitir alimentar la placa OpenMote, también sirve para alimentar el sensor ACS712 que llevará a cabo las medidas de corriente.
- Ejecución de la aplicación OpenVisualizer. Una vez abierta, habrá que seleccionar como DagRoot del sistema la placa OpenMote conectada al ordenador portátil.

En este punto, las placas OpenMote deben estar sincronizadas, lo que se puede ver a través del led naranja que poseen, o bien en la información que nos muestra la aplicación OpenVisualizer.

➢ **Segundo Paso: Ejecución de la aplicación python del bloque de procesamiento**.

Como ya se ha explicado en esta memoria, esta aplicación será la que controle el flujo de funcionamiento de todo el sistema desarrollado. Para comenzar su ejecución, sólo será necesario ejecutar el fichero python ControlConsumo.py, bien a través de hacer doble click sobre él, o bien a través del terminal.

Si todo es correcto, en el terminal se irá informando de cada medida que lleva cabo el sistema, medidas que a la vez irán transmitiéndose a la plataforma theThings.io.

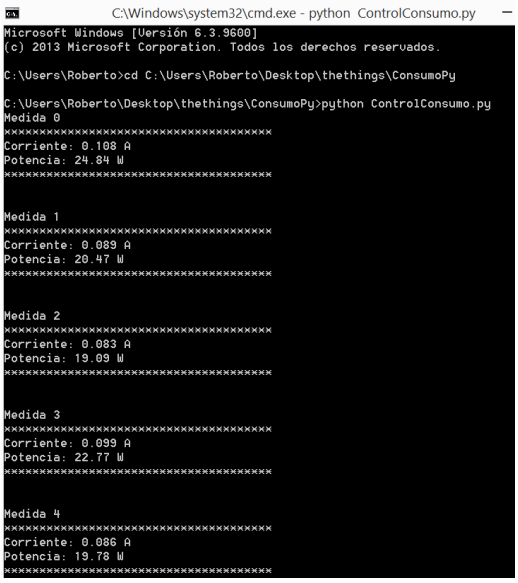


Figura 44. Sistema en ejecución.

4. PRUEBAS Y VERIFICACIÓN DEL SISTEMA

4.1. INTRODUCCIÓN

En este capítulo se recogen las diferentes pruebas y comprobaciones que se han llevado a cabo sobre el sistema para verificar su correcto funcionamiento y el cumplimiento de los objetivos para los que ha sido ideado. Para ello, además de comprobar el funcionamiento del sistema en global, también se han realizado ensayos sobre partes individuales del conjunto, de manera que se pudiera asegurar su eficiencia antes de ser integradas en sistema.

Así, en primer lugar se ha llevado a cabo la verificación del bloque sensor, tanto del sensor ACS712 como del CAD de la placa CC2538, intentando así probar que los datos que le llegan al bloque de procesamiento son realmente representativos del consumo que se esté produciendo en la toma de corriente a monitorizar.

Otra parte del sistema puesta bajo verificación ha sido la correcta comunicación con la plataforma theThings.io, asegurando así que los datos que se disponen procedentes de la etapa de medida quedan correctamente almacenados en dicha plataforma.

Por último, también se ha verificado que la comunicación entre el bloque sensor y el bloque de procesamiento se efectúa de manera correcta, validando así el buen uso y funcionamiento del protocolo COAP en esta tarea, y asegurando una parte muy importante del sistema global.

4.2. VERIFICACIÓN DEL BLOQUE SENSOR

Para que el prototipo que se ha diseñado y desarrollado con este TFM se pueda considerar válido desde el punto de vista de su funcionamiento, es parte vital que el bloque sensor posea un grado aceptable de fiabilidad y precisión a la hora de llevar a cabo las medidas sobre la toma que se desea monitorizar, y que además estas medidas sean correctamente recibidas por el CAD de la placa OpenMote CC2538 y éste realice correctamente la conversión, obteniendo un valor que sea fiel reflejo de la medida de corriente llevada a cabo en la toma de corriente. Para ello, este bloque sensor se ha intentado probar en dos etapas bien diferenciadas. Por un lado se ha verificado el correcto funcionamiento del sensor ACS712, y por otro el del CAD de la placa CC2538. Para ello, en primer lugar se ha verificado de manera individual el CAD de la placa CC2538, ya que éste es de mucha utilidad para comprobar a su vez a continuación el funcionamiento del sensor ACS712.

4.2.1. VERIFICACIÓN CAD PLACA OPENMOTE CC2538

La verificación de las correctas prestaciones del convertidor analógico-digital de la placa OpenMote CC2538 se ha llevado a cabo con esta placa de manera aislada, fuera del sistema general. La razón es que integrado en el sistema, las medidas que se llevan a cabo son de señales AC, que precisan de cierto procesamiento para luego obtener un valor eficaz, por lo que en primer lugar se preferido asegurar el buen funcionamiento del sistema con señales continuas, cuyo valor no varíe en el tiempo.

Para ello, el método de verificación seguido ha sido aplicar diferentes valores de tensión continua sobre el pin que se ha configurado como entrada analógica de la placa CC2538, y comprobar el valor que se obtiene a la salida de su convertidor analógico-digital. Estas señales de tensión continua de entrada han podido ser medidas mediante un multímetro, y por lo tanto son perfectamente conocidas. Para todas estas verificaciones hay que tener siempre presente la tensión de referencia elegida en la configuración del CAD, que es de unos 3,3V aproximadamente, siendo la tensión Vcc de la placa.

En primer lugar se le ha aplicado una tensión de 0V, para ello cortocircuitando la entrada analógica del CAD con el pin que representa GND de la placa. En esta situación, la salida convertida del CAD es de 0, y por lo tanto la salida esperada.

La siguiente prueba consiste en aplicar al CAD una tensión igual o superior a la tensión de referencia, de 3,3 V. Para ello, se le aplica directamente la tensión Vcc de la placa, de aproximadamente este valor. La salida que se obtiene es la siguiente:

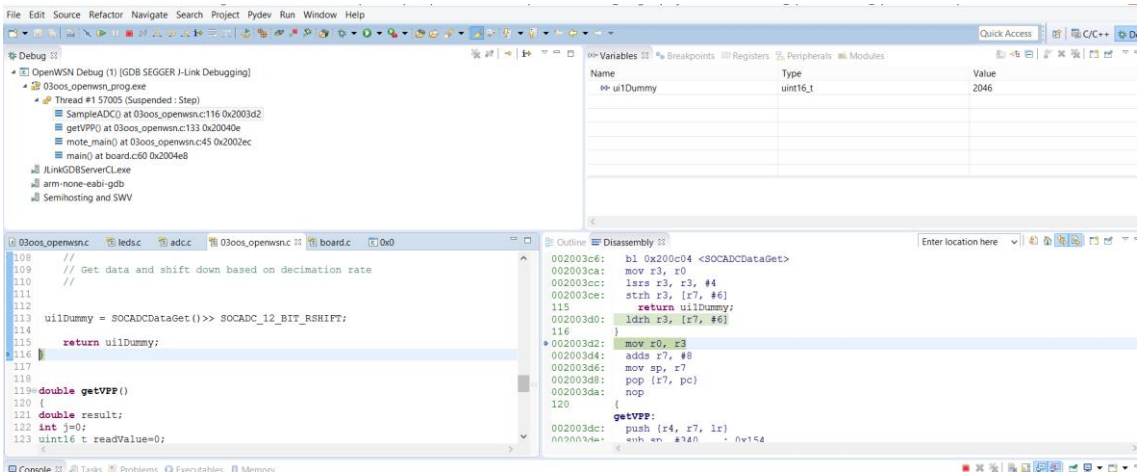


Figura 45. Salida del CAD para la entrada Vcc

Como se puede observar, el valor convertido que devuelve el CAD es 2046. Teniendo en cuenta que se emplean 12 bits de resolución en el CAD, efectivamente este valor se corresponde prácticamente con el fondo de escala:

$$2^{11} = 2048\ niveles\ del\ CAD$$

Como un nivel es el 0, el máximo valor por lo tanto corresponde a 2047.

Si se aplica una tensión mayor, concretamente de unos 4,5 V, se puede observar como el resultado sí es el máximo, habiéndose producido la saturación del CAD con una tensión superior a su máximo nivel de entrada:

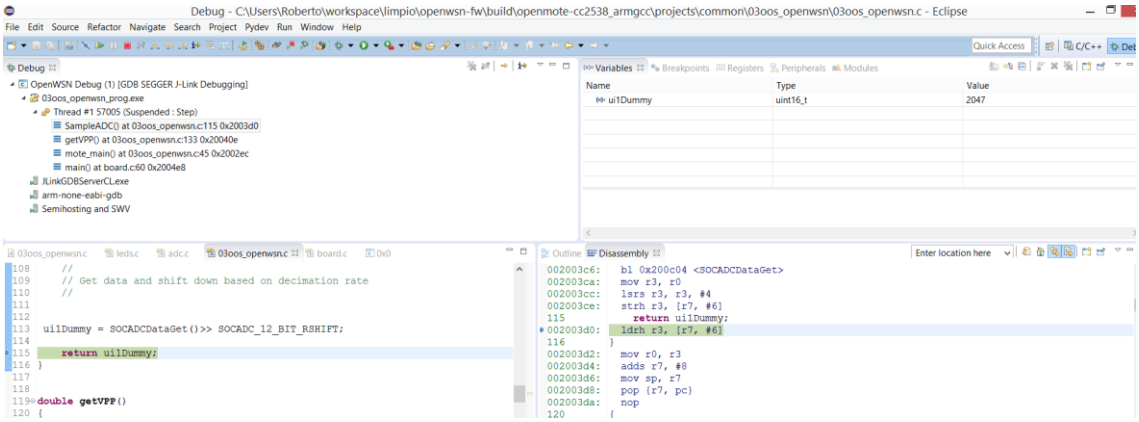


Figura 46. Salida del CAD en saturación

Una vez comprobados estos dos extremos de la conversión, el siguiente paso de la verificación consiste en aplicar diferentes tensiones dentro de este rango, y comprobar los valores convertidos devueltos por el CAD.

Así, si se aplica una tensión a través de una batería de 1,5V, se obtiene:

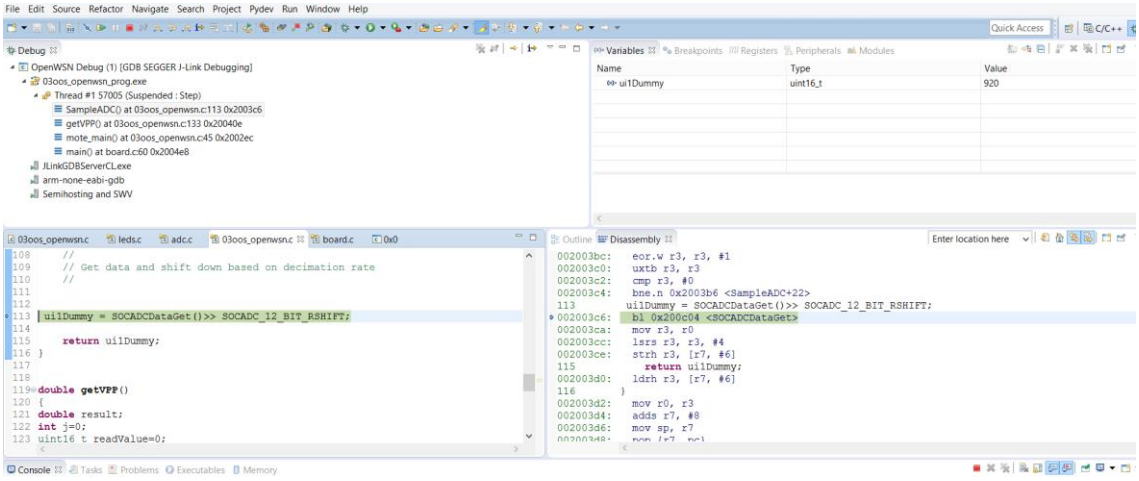


Figura 47. Salida del CAD para una entrada de 1,5V

Como se puede observar, el valor devuelto es de 920. Midiendo mediante polímetro la tensión real que proporciona la pila, se comprueba que ésta es de 1,48V. Si se mide ahora la tensión Vcc que proporciona la placa, que será igual a la tensión de referencia del CAD, se comprueba que es igual a 3,28V. Con estos valores, el valor teórico devuelto por el CAD debería ser:

$$\frac{1,48}{3,28} \cdot 2048 = 924$$

Vemos que efectivamente se aproxima mucho al valor devuelto.

4.2.2. VERIFICACIÓN SENSOR ACS712

Una vez comprobado que el funcionamiento del CAD se ajusta a sus prestaciones y a lo esperado para que su integración en el sistema sea útil, el siguiente paso es verificar el otro elemento del bloque sensor, que es precisamente el sensor encargado de las medidas de corriente de los dispositivos conectados a la toma de corriente que se monitoriza.

El sensor se va a emplear para medir consumos de corriente alterna de diferentes dispositivos, con lo que es muy complicado comparar estas medidas con medidas hechas mediante un polímetro de manera manual. Por ello, lo que se va a hacer para verificar el correcto funcionamiento del bloque, es realizar medidas sobre dispositivos de consumo conocido, comparando esas medidas con este consumo "teórico", y comprobando si realmente la medida empírica se ajusta al consumo esperado.

Para estas pruebas de verificación se han empleado tres dispositivos con un consumo diferente y conocido, como se ha comentado anteriormente. Se tratará de un exprimidor eléctrico, el cargador de un portátil, y un secador de pelo.

El primer paso de estas pruebas es llevar a cabo una serie de medidas sin ningún dispositivo conectado al sistema. Con esto se pretende obtener un valor de referencia del sistema, el cero del bloque sensor, que supondrá un offset en el resto de medidas que se hagan con él. Así, en la imagen siguiente se recogen los resultados de esta serie de 5 medidas espaciadas un minuto en el tiempo:

Figura 48. Obtención del offset del sistema.

Como se puede comprobar, la potencia medida no es 0 aunque no haya nada conectado al sistema, ya que siempre se ha medido una corriente residual en torno a los 90-100mA. Este valor de potencia oscilará en torno a los 19-20W, y por ello se tomará este valor aproximado como "offset" del sistema.

Una vez determinado este “offset”, se puede comenzar con las medidas sobre dispositivos reales. El primero de ellos, el exprimidor, será el que presente un consumo menor de todos ellos. Si se observa la información que incluye el propio exprimidor, se puede comprobar que éste presenta un consumo aproximado de 40W:

Figura 49. Características técnicas de exprimidor eléctrico.

Al conectar y activar el exprimidor, los resultados de medida que se obtienen desde el bloque sensor son los siguientes:

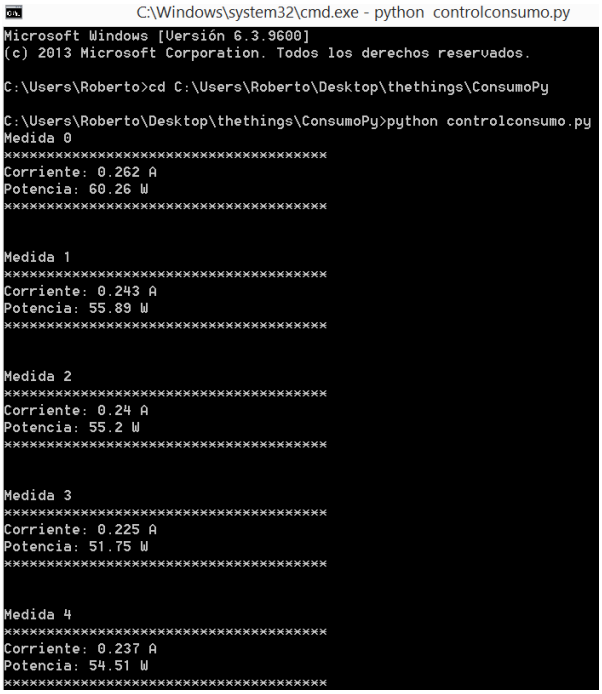


Figura 50. Medidas de consumo de exprimidor eléctrico.

Como se puede ver, si tenemos en cuenta ese “offset” determinado anteriormente, el consumo medido oscila entre los 30W y 40W, que efectivamente coincide con lo indicado en las especificaciones técnicas del exprimidor.

El siguiente dispositivo que se ha utilizado para probar las prestaciones del bloque sensor es un cargador de ordenador portátil. A continuación se muestran sus especificaciones técnicas:

Figura 51. Características técnicas de cargador de portátil.

Como se puede ver en la imagen anterior, el consumo indicado por la información técnica del cargador es de 90W. Una vez más, se conectó el cargador, tanto a la red como al propio portátil, y se realizaron varias medidas con el bloque sensor:

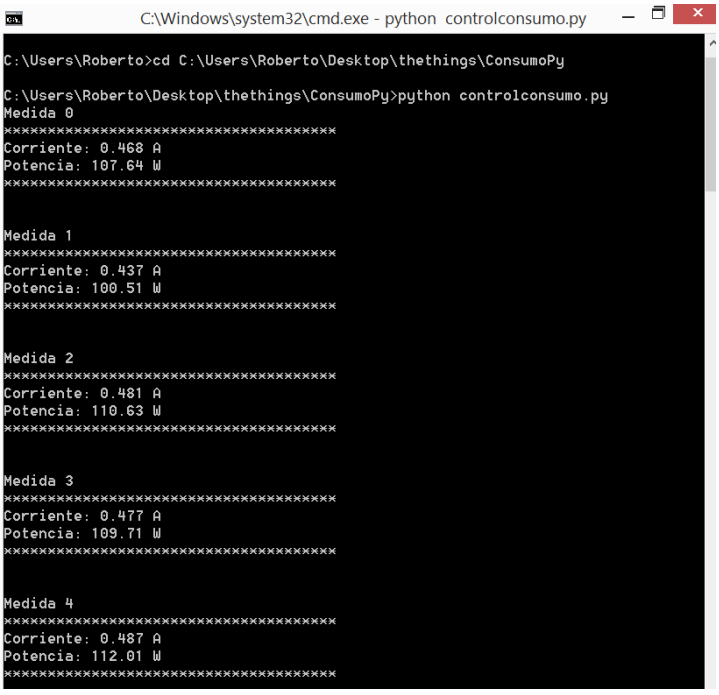


Figura 52. Medidas de consumo de cargador de portátil.

Se observa como el consumo medido oscila entre los 100W y los 110W. Si ahora se tiene en cuenta el valor de "offset" determinado, se llega a la conclusión de que el consumo real medido para este cargador está entre los 80W y los 90W, lo cual coincide con lo recogido en su información técnica.

Por último, para la prueba final sobre el bloque sensor se ha elegido un dispositivo que presente un consumo mayor que los anteriores. Se trata de un secador de pelo, que a su máxima potencia, ofrece unos datos de consumo de unos 500W. En la siguiente captura de pantalla se recogen la serie de medidas llevadas a cabo al enchufar el secador al sistema, y activar su funcionamiento a su máxima potencia:

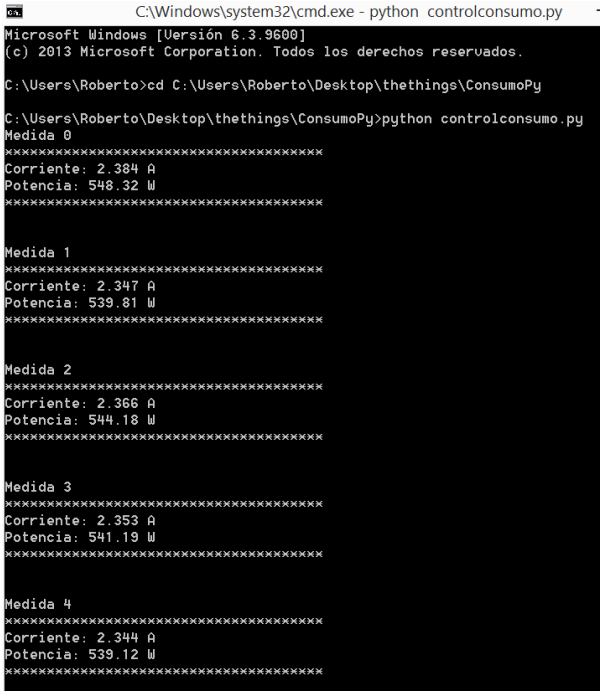


Figura 53. Medidas de consumo de secador eléctrico.

Como se observa, en este caso el consumo es mucho mayor que en los otros dos dispositivos, moviéndose en torno a valores de potencia de unos 530W y 540W, que teniendo en cuenta el "offset" del sistema se quedarían en un rango de entre 510W y 520W, lo cual se aproxima mucho al consumo teórico esperado del dispositivo, de unos 500W.

4.3. VERIFICACIÓN COMUNICACIONES

Una parte fundamental del sistema desarrollado comprende la comunicación e intercambio de datos entre sus diferentes elementos, aspecto crucial para su correcto funcionamiento. Por ello, en este apartado se van a recoger diferentes muestras y pruebas de esta correcta comunicación, que incumbirán a los diferentes bloques.

4.3.1. FORMACIÓN RED OPENWSN

El funcionamiento del sistema se basa en la creación de una red OpenWSN entre dos placas OpenMote CC2538, que permite la comunicación entre ellas y con Internet. Por ello, la comprobación de la creación de esta red es un aspecto fundamental a verificar. Para llevar a cabo esta verificación se han empleado varias alternativas de comprobación.

La primera de ellas será a través de la herramienta software OpenVisualizer, que es precisamente mediante la cual se crea esta red OpenWSN. Una vez que esta herramienta ha sido ejecutada, y se ha establecido una placa OpenMote como DAGroot, se puede comprobar como entre los vecinos de esta placa aparece la otra OpenMote, confirmando así el establecimiento de la red OpenWSN

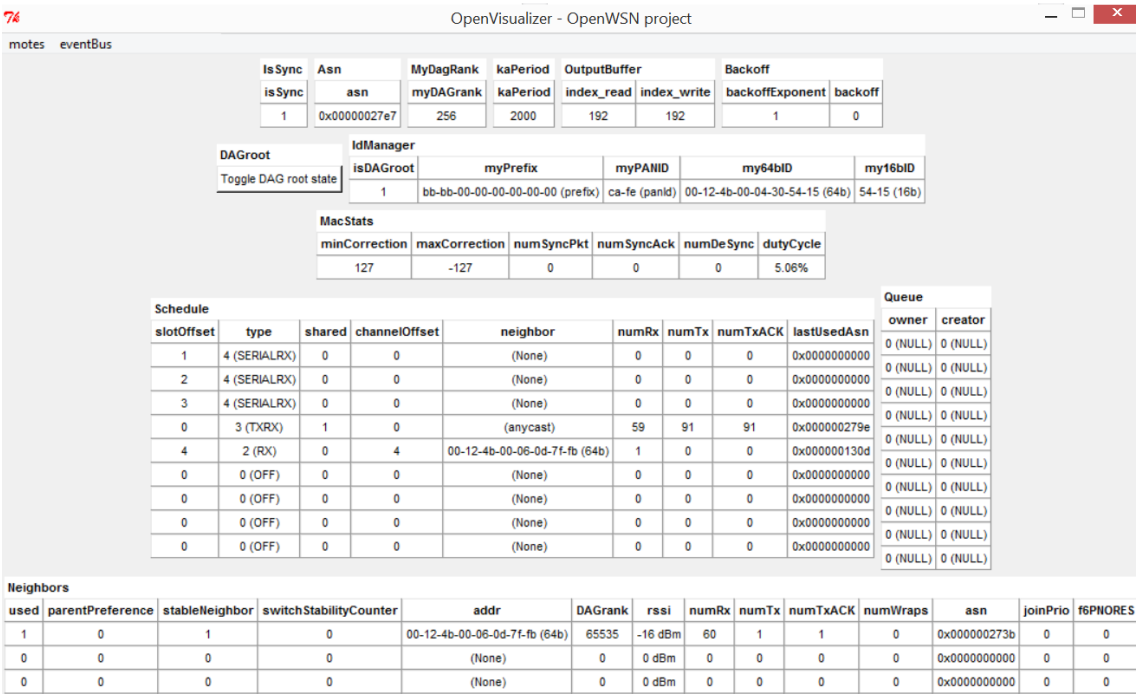


Figura 54. Interfaz de Openvisualizer con red sincronizada

Además, esto también puede verificarse mediante inspección física de las placas, ya que al estar sincronizadas, además de parpadear el led verde indicando que hay comunicación, también se enciende el led naranja, indicando dicha sincronización.

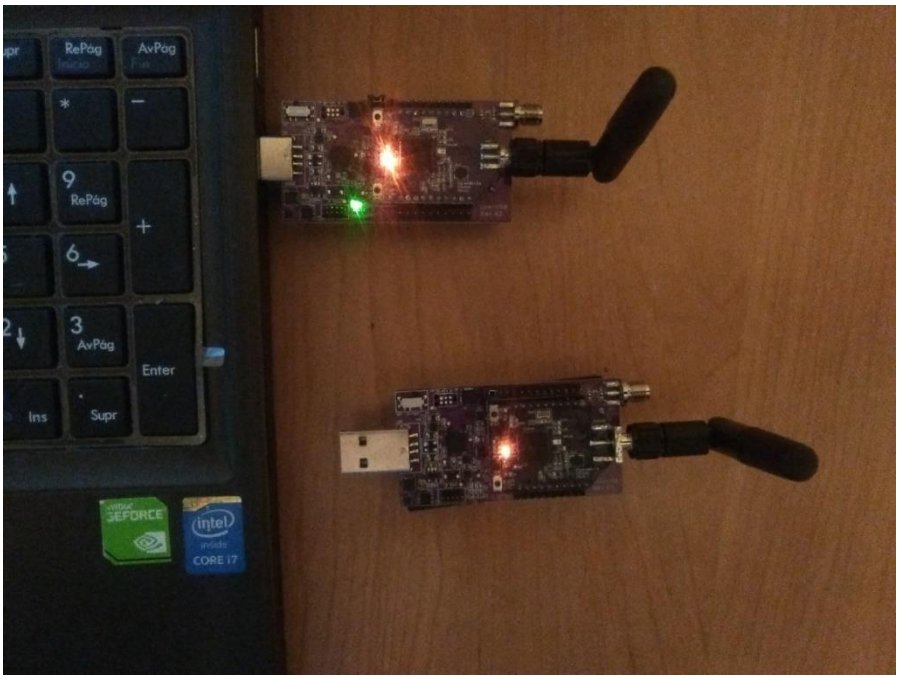


Figura 55. Placas OpenMote sincronizadas.

Por último, otra forma alternativa de comprobar que la red OpenWSN se ha establecido correctamente, es realizar un ping a la placa OpenMote del bloque sensor, que aparece como vecina de la marcada como DaGroot. Así, si una vez sincronizada la red, se lleva a cabo este ping a la dirección IPv6 de la placa vecina:

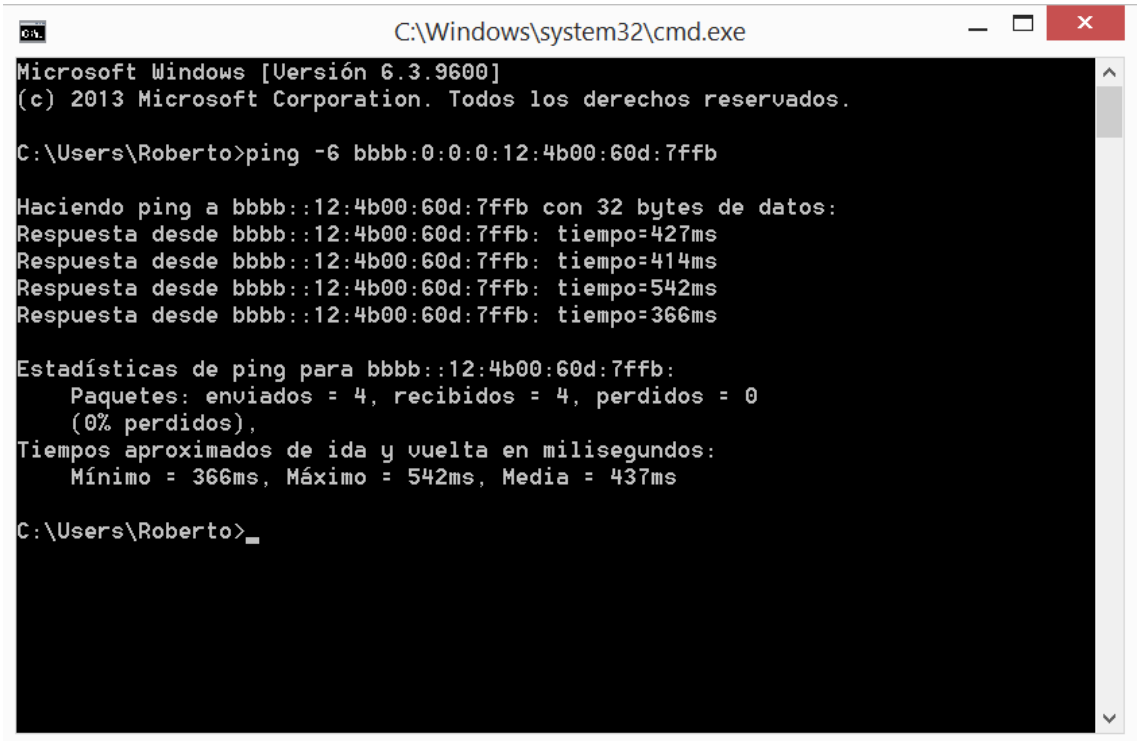


Figura 56. Pantalla del terminal con comando ping a la placa OpenMote

Como se observa en la imagen anterior, hay respuesta desde la placa OpenMote del bloque sensor, con lo que la red OpenWSN se ha establecido de manera correcta.

4.3.2. COMUNICACIÓN BLOQUE SENSOR Y BLOQUE PROCESAMIENTO

Como ya se ha recogido en esta memoria, la comunicación entre el bloque sensor y el bloque de procesamiento se lleva a cabo mediante el protocolo COAP, enviándose desde el bloque de procesamiento una petición COAP con el método GET. Al recibirla, el bloque sensor llevará a cabo un proceso de medida de corriente y responderá con los datos obtenidos.

Evidentemente, la manera más clara de verificar que este proceso de comunicación es correcto es comprobado que los datos llegan al bloque de procesamiento. Para aportar una prueba adicional que permita testear esta comunicación, se ha empleado la herramienta Wireshark, capturando el tráfico que se genera durante este proceso en la red OpenWSN .

En la siguiente imagen se muestra la captura realizada durante la comunicación:

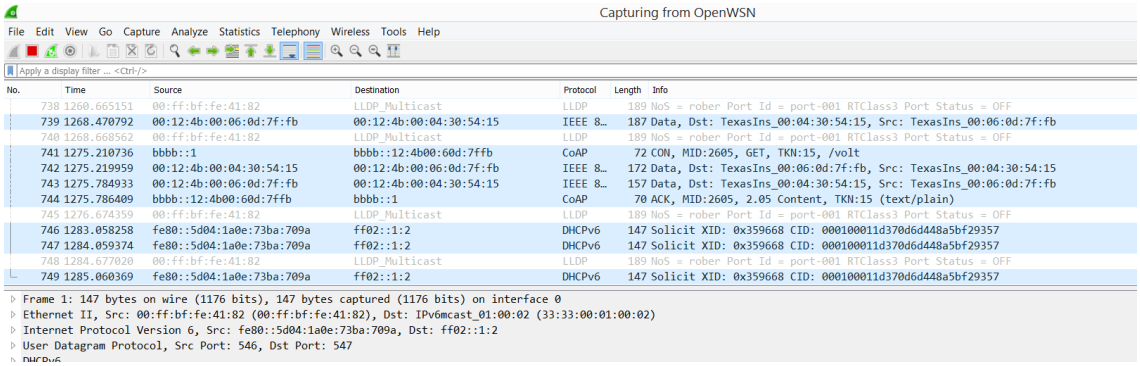


Figura 57. Pantalla de Wireshark con captura del tráfico de la red OpenWSN

Como se puede observar, desde el bloque de procesamiento (dirección bbbb::1) se emite una petición CoAP con el método GET hacia el bloque sensor (dirección bbbb:12:4b00:60d:7fffb), indicando el recurso volt, lo que permitirá que en este bloque sensor arranque la aplicación cvolt y tome una medida de corriente. A continuación, desde el bloque sensor se responde con un paquete CoAP de tipo ACK, en el que irá contenida la información requerida desde el bloque de procesamiento, es decir, el valor devuelto por el CAD de la placa OpenMote CC2538 de este bloque sensor.

4.3.3. COMUNICACIÓN CON PLATAFORMA THETHINGS.IO

La comunicación del sistema con la plataforma theThings.io será un aspecto fundamental a vigilar, ya que aquí se almacenan los datos registrados de consumo y por lo tanto es el mecanismo que permite al usuario consultarlos posteriormente. Por ello, una de las pruebas de verificación del sistema ha sido comprobar la fiabilidad de la comunicación y este almacenamiento de datos.

Para llevar a cabo esta verificación, lo que se ha hecho es realizar un envío de datos continuo hacia la plataforma, espaciados a intervalos constantes de tiempo. Así, se ha enviado a la

plataforma un número cada 30 segundos durante varias horas, incrementando este número en cada envío. Los resultados han sido satisfactorios, quedando almacenados todos los datos enviados.

Figura 58. Datos de corriente registrados en plataforma theThings.io

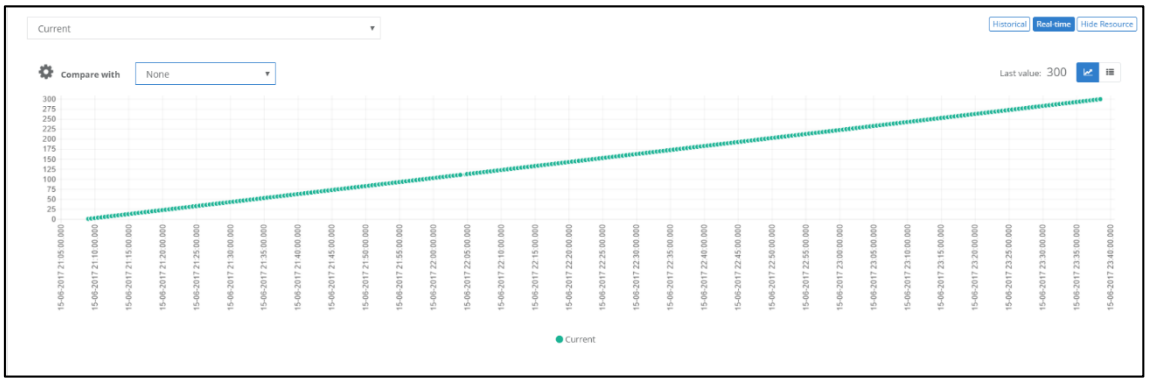


Figura 59. Gráfica con datos de corriente registrados en plataforma theThings.io

4.4. VERIFICACIÓN GLOBAL DEL SISTEMA

Todas las pruebas anteriores han servido para verificar que cada elemento del sistema funciona correctamente por separado, y con la fiabilidad y eficacia mínima necesarias para conseguir un sistema global de utilidad. Pero evidentemente es necesaria y totalmente imprescindible la verificación del sistema con todos sus elementos integrados, funcionando tal y como se ha diseñado el prototipo para alcanzar sus objetivos.

La verificación sobre el sistema global ha consistido en la monitorización de una toma concreta de corriente durante un periodo amplio de tiempo, en el cual el prototipo ha estado en continuo funcionamiento tomando medidas de corriente de dicha toma. Durante este periodo, la toma ha pasado por estados de total desconexión, sin tener ningún dispositivo conectado, y otros en los que diferentes dispositivos se han conectado durante un tiempo determinado.

El objetivo de esta verificación es comprobar si efectivamente estos diferentes estados de la toma de corriente son recogidos por el prototipo, empleando para ello los datos almacenados en la plataforma theThings.io, y analizándolos para determinar su coherencia con respecto a la evolución de la toma. Para ello, evidentemente, las actuaciones sobre la toma serán planificadas y conocidas, para poder así llevar a cabo esta comprobación con los datos recogidos.

El periodo de monitorización de la toma ha sido de 8 horas, durante el cual se han conectado a ella diferentes dispositivos, entre los que se encuentran los ya mencionados en el apartado de verificación del sensor ACS712. Así, la evolución de la conexión a la toma durante estas 8 horas ha sido la siguiente:

- **ETAPA 1 – 0h**: Ordenador portátil conectado
- **ETAPA 2 - 1h30min**: Ningún dispositivo conectado
- **ETAPA 3 – 3h15min**: Conexión de exprimidor durante 5 minutos
- **ETAPA 4 - 3h 20min**: Ningún dispositivo conectado
- **ETAPA 5 - 4h**: Lámpara conectada
- **ETAPA 6 - 5h**: Ningún dispositivo conectado
- **ETAPA 7 - 5h20min**: Conexión de secador durante 5 minutos
- **ETAPA 8 - 5h.35 min**: Conexión de secador durante 5 minutos
- **ETAPA 9 - 6h**: Conexión de exprimidor durante 5 minutos
- **ETAPA 10** - **6h.20min**: Conexión de cafetera
- **ETAPA 11 - 6h.45min**: Conexión de ordenador portátil

Los datos almacenados en la plataforma theThings.io durante este tiempo son los siguientes:

➢ **Potencia:**

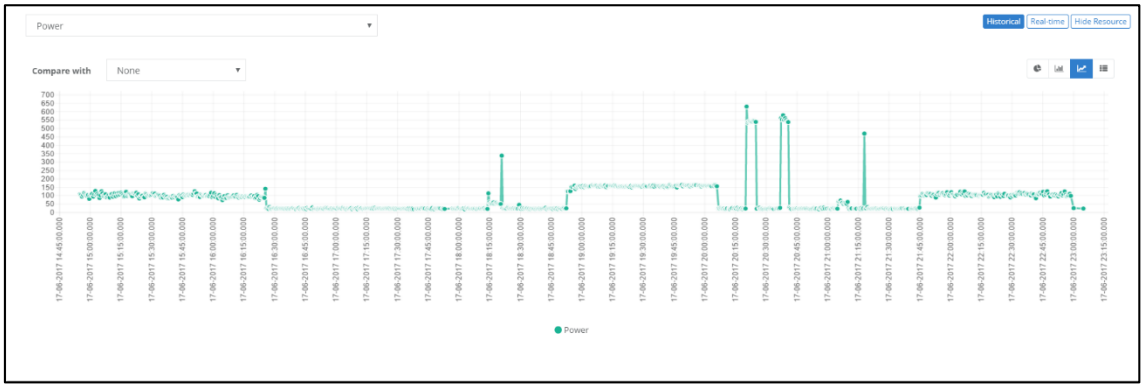


Figura 60. Gráfica de evolución de la potencia durante prueba de verificación.

➢ **Corriente:**

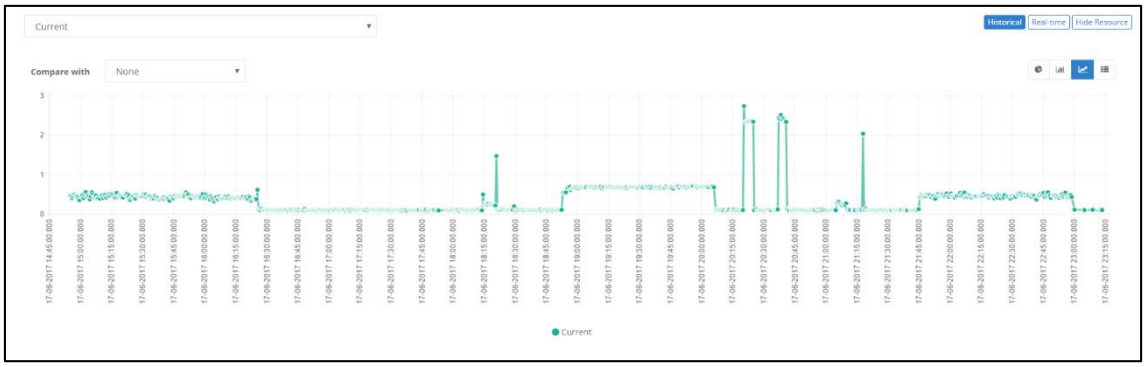


Figura 61. Gráfica de evolución de la corriente durante prueba de verificación.

Lo primero que se puede observar en estas dos gráficas es que ambas tienen la misma forma, lo cual es lógico teniendo en cuenta que la corriente y la potencia consumida en la toma están relacionadas mediante la tensión de la línea de 230V. Además, en ambas se pueden ver bien diferenciados los diferentes tramos correspondientes a la evolución de la toma durante el periodo de prueba.

Así, centrándose en la gráfica de la potencia, que es la medida más característica para representar el consumo de un dispositivo, se puede observar como ésta comienza con un tramo en el que su valor oscila aproximadamente entre 90 y 110W, que es precisamente el consumo esperado para el ordenador portátil, como también se vio en el apartado de verificación del sensor ACS712.

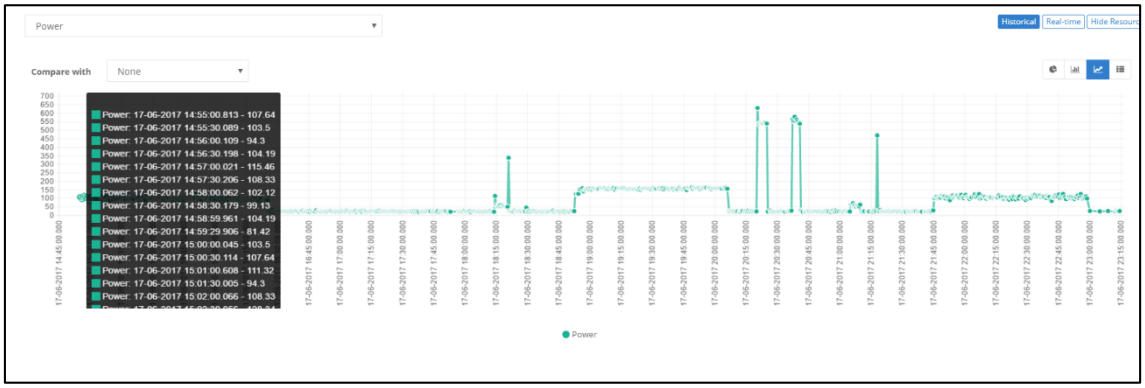


Figura 62. Detalle de consumo de potencia durante primera etapa de prueba (ordenador portàtil)

A continuación, se distingue que la potencia disminuye y existe un tramo constante de valor en torno a los 20W. Este tramo se corresponde con la segunda etapa de la planificación descrita anteriormente, en la que no hay ningún dispositivo conectado. No hay que olvidar que estos aproximadamente 20W se corresponden con el offset que presenta el sistema sin que haya presencia de dispositivo conectado.

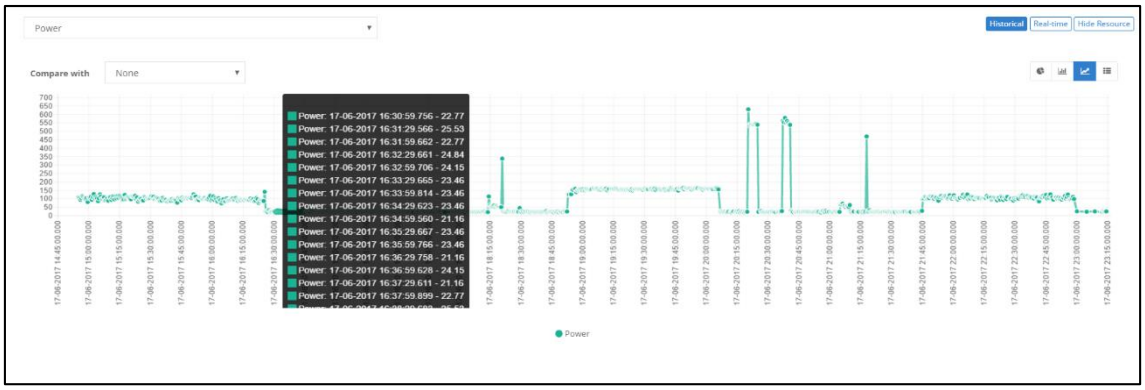


Figura 63. Detalle de consumo de potencia durante segunda etapa de prueba (Desconexión)

Tras este tramo de potencia mínima, se observa como hay un tramo en el que la potencia aumenta ligeramente durante un periodo corto, con un pico más pronunciado final, volviendo a continuación a su valor mínimo. Esta etapa se corresponde con la utilización del exprimidor, que tiene un consumo aproximado de unos 40W:

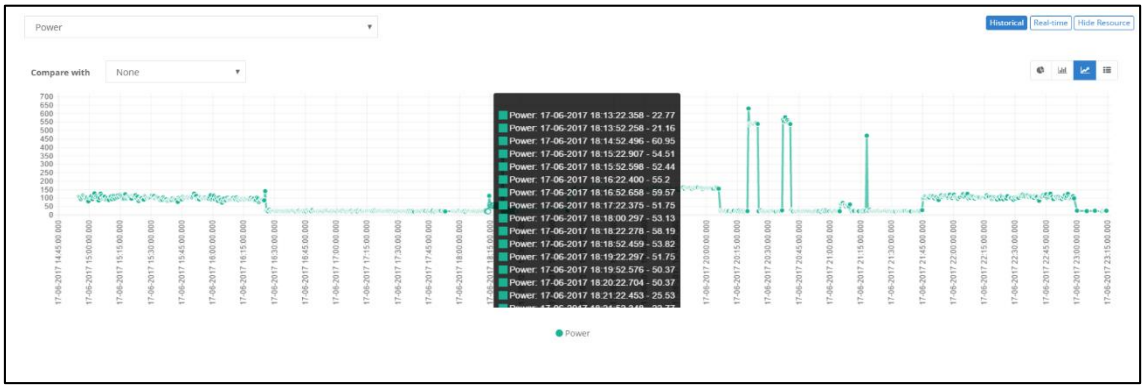


Figura 64. Detalle de consumo de potencia durante tercera etapa de prueba (Exprimidor)

Como se puede ver, el consumo registrado en la plataforma oscila en torno a los 50-60W, lo cual se corresponde con lo esperado para este dispositivo (No hay que olvidar el offset de unos 20W). Como se describió en la planificación, el exprimidor ha estado activado de manera continua durante 5 minutos. Este periodo es demasiado amplio para lo que está preparado este dispositivo, lo que ha provocado que antes de llegar a este tiempo se haya sobrecargado y haya tenido que ser desactivado para evitar dañarlo. De esta circunstancia proviene ese pico de potencia que puede observarse en la gráfica, antes de volver a los valores de potencia mínima tras ser desactivado.

Tras desconectar el exprimidor, se puede comprobar como existe un periodo de potencia mínima en el que no hay nada conectado al enchufe, y a continuación comienza un tramo con un consumo constante y en torno a los 150-160W. Esta franja se corresponde con la quinta etapa de la planificación, que consiste en la conexión a la toma de una lámpara, con un consumo aproximado de unos 150W.

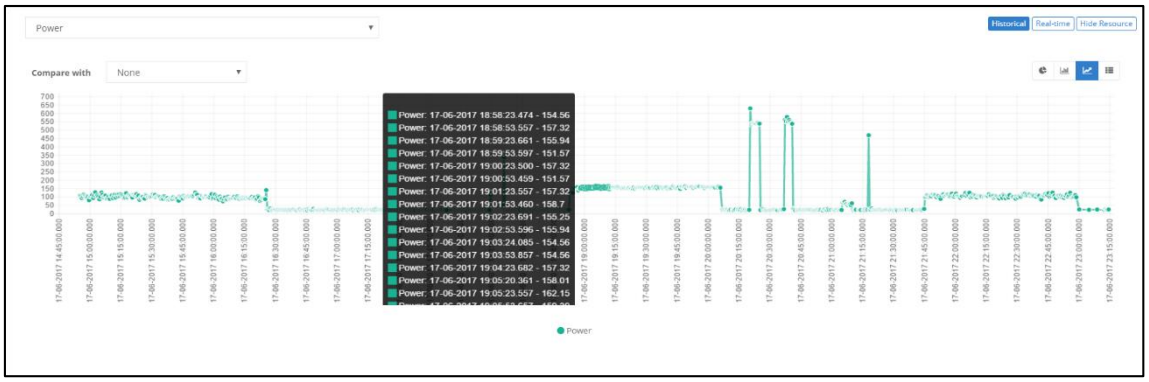


Figura 65. Detalle de consumo de potencia durante quinta etapa de prueba (lámpara)

Tras la desconexión de esta lámpara y la vuelta a los valores mínimos de potencia, se puede comprobar como existen dos picos con los valores más altos de potencia alcanzados durante este periodo de prueba. Estos dos picos se corresponden con la conexión del secador de pelo durante 5 minutos, las etapas 7 y 8 de la planificación.

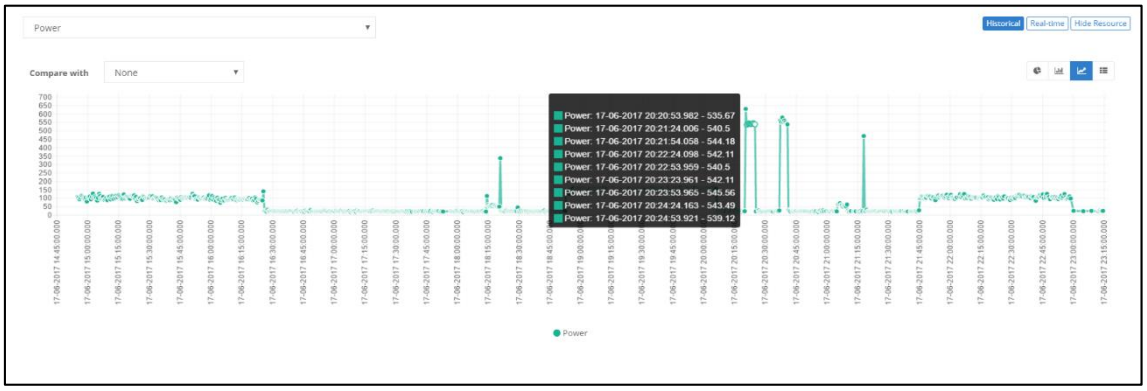


Figura 66. Detalle de consumo de potencia durante séptima etapa de prueba (secador de pelo)

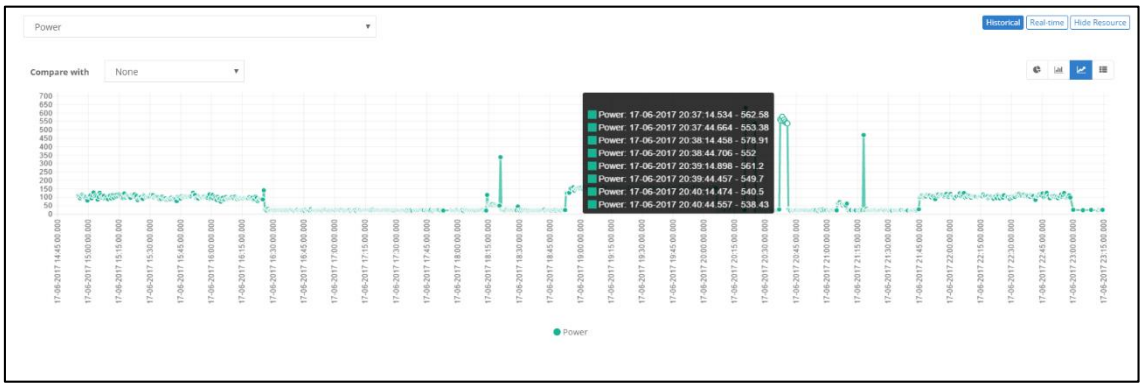


Figura 67. Detalle de consumo de potencia durante octava etapa de prueba (secador de pelo)

Como se observa en las capturas anteriores, el consumo durante estos dos tramos oscila entre los 540-560W, lo cual se corresponde con lo esperado para este dispositivo teniendo en cuenta el offset del sistema, ya que su consumo aproximado es de unos 500W.

Siguiendo con la gráfica de la prueba, se observa que tras un periodo de potencia mínima existe un tramo en que la potencia aumenta hasta un pequeño valor, de unos 60W, que se corresponde con la segunda conexión del exprimidor a la toma, etapa 9 de la planificación. Después, aparece un pico de potencia de casi 500W, que se corresponde con la conexión de la cafetera, etapa 10. Este tramo tiene tan sólo una medida porque la cafetera consigue alcanzar la temperatura configurada antes de un minuto, y por lo tanto al hacer las medidas cada 30 segundos, sólo se ha captado una muestra de su estado de activación.

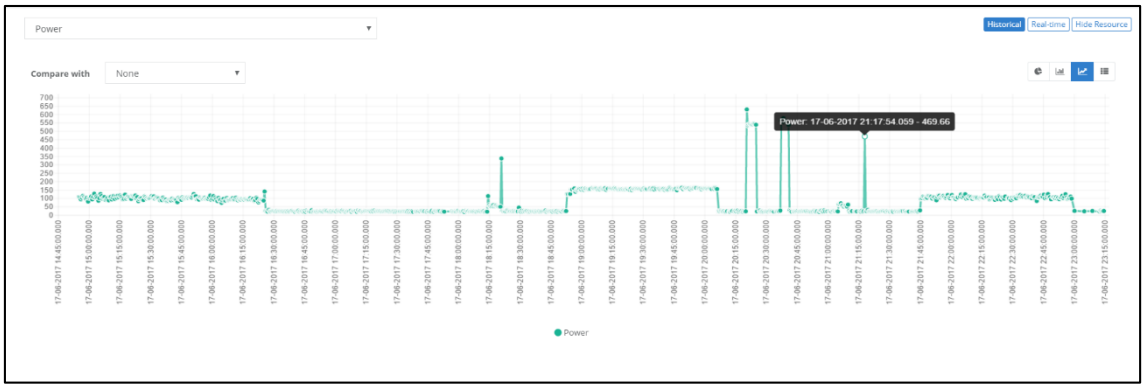


Figura 68. Detalle de consumo de potencia durante décima etapa de prueba (cafetera)

La última etapa de esta prueba de verificación, antes de dar por finalizado el proceso, consiste en volver a enchufar el ordenador portátil a la toma. Si se observan los resultados almacenados en la plataforma theThings.io, se ve como efectivamente aparece un tramo final con un valor de potencia en torno a los 100-110W, que efectivamente se corresponden con el consumo esperado para este dispositivo.

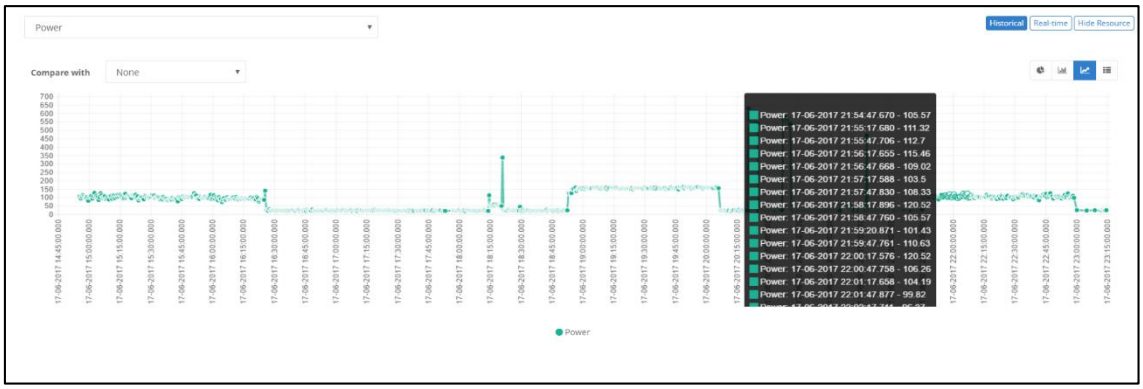


Figura 69. Detalle de consumo de potencia durante última etapa de prueba (ordenador portátil)

Una vez analizado con detalle todo este proceso de verificación sobre el sistema global, se puede concluir que los resultados obtenidos reflejan con una fiabilidad suficiente el estado de la toma de corriente, y que por lo tanto el prototipo cumple con los objetivos para los que ha sido diseñado, que eran principalmente tener la capacidad de identificar si existe algún dispositivo activado a la toma de corriente, y en caso afirmativo poder monitorizar su consumo con cierta precisión. Evidentemente esto no es óbice para posibles futuras mejoras y ampliaciones de características que se pueden llevar a cabo sobre este diseño, y que serán objeto de discusión en el capítulo de esta memoria dedicado a posibles líneas futuras de trabajo.

5. CONCLUSIONES Y LÍNEAS FUTURAS

5.1. CONCLUSIONES

El objetivo inicial de este TFM era el desarrollo de un prototipo que haciendo uso de la tecnología OpenMote y OpenWSN sirviera de demostrador del paradigma del internet de las cosas. Así, se ha diseñado y desarrollado un prototipo capaz de tomar medidas de corriente de una toma conectada a la red, y comunicar y almacenar estas medidas en la plataforma theThings.io, permitiendo por lo tanto una monitorización del consumo eléctrico llevado a cabo en dicha toma, con un grado de precisión suficiente para que sean datos de utilidad.

Además de este objetivo global, se habían marcado otros hitos a llevar a cabo y que también se considera que se han cumplido satisfactoriamente. Así, el primer paso en el desarrollo de este TFM ha consistido en una etapa de investigación y recopilación de información acerca de las tecnologías y técnicas útiles para el diseño del prototipo deseado. Por ello se ha realizado un trabajo de documentación acerca de diferentes aspectos del Internet de las cosas, elemento sobre el cual se construye la idea principal de este proyecto. En primer lugar definiendo y describiendo sus características genéricas y aspectos básicos, y después entrando más en detalle en un estándar de este paradigma, el estándar IEE 802.15.4, que será precisamente el que se emplee en este proyecto al emplear la tecnología OpenWSN. Todo este trabajo de documentación, además de como introducción al trabajo técnico llevado a cabo en este TFM, ha servido para adquirir los conocimientos básicos acerca de estos aspectos fundamentales en los que se basa la tecnología empleada por el prototipo diseñado.

El sistema desarrollado puede descomponerse de manera muy clara en su parte hardware y su parte software, en las cuales se ha trabajado en paralelo durante gran parte del desarrollo del proyecto, y que en ambos casos podrán ser divididas en un bloque sensor y un bloque central de procesamiento. La parte hardware del sistema está basada en OpenMote, plataforma caracterizada por su sencillez, bajo coste y gran autonomía. Junto con esta plataforma se ha integrado un sensor de corriente, el sensor ACS712, que también cumple con estas premisas, y proporciona las prestaciones suficientes para alcanzar los objetivos de precisión y fiabilidad deseados para este prototipo. La comunicación entre los diferentes bloques del sistema será inalámbrica, siendo una de las características deseadas del sistema la movilidad, y basada en una red OpenWSN, tecnología ideada para el internet de las cosas.

Por su parte, el lado software contara con diversas plataformas de programación, dependiendo del bloque del sistema. Así, el bloque sensor, encargado de capturar medidas de la toma de corriente, basará su parte software en la carga del firmware de openWSN en las placas OpenMote, una de las cuales estará específicamente encargada de muestrear y convertir las medidas proporcionadas por el sensor de corriente. Este firmware ha sido modificado adecuadamente para incluir una aplicación capaz de recibir peticiones desde el bloque central del sistema, tomar una medida procedente del sensor, y enviar los datos hacia ese bloque como respuesta a su petición. Este firmware estará programado en lenguaje C, y el protocolo de comunicación con el bloque central será el protocolo CoAP.

Por su parte, el bloque central de procesamiento contará con una aplicación software programada en python, que será la que se encargue de arrancar y llevar el flujo de funcionamiento del sistema, iniciando las tareas necesarias para su ejecución. Así, realizará la

petición de medidas al bloque sensor, medidas que procesará adecuadamente tras recibirlas, y finalmente trasmitirá los datos ya procesados a la plataforma theThings.io para su almacenamiento.

Para llevar a cabo la validación del prototipo diseñado se han realizado diversas pruebas de verificación cuyo resultado se considera aceptable para que esta validación se determine positiva. Estas pruebas han consistido en testear el prototipo con diversos dispositivos cuyo consumo es conocido, comprobando que los resultados obtenidos por el sistema son coherentes en comparación con este consumo teórico. Así, una primera fase de estas pruebas se ha ceñido a esta comparación de consumo, sin tener en cuenta ningún otro aspecto o parte del prototipo, mientras que la prueba de validación final ha tenido en cuenta todo el sistema global, incluyendo por lo tanto la comunicación con la plataforma theThings.io donde se almacenan estos consumos. En ambos casos los resultados de estas pruebas se consideran satisfactorios, y un punto de partida sólido para posibles mejoras o ampliaciones futuras del sistema.

También se quiere aprovechar este capítulo para hacer un breve análisis sobre la evolución del desarrollo del proyecto, teniendo en cuenta la planificación establecida al inicio y los objetivos marcados en esa etapa inicial. Así, hay que mencionar que ha habido un cierto cambio con respecto a esta planificación inicial, significando además un retraso en los tiempos marcados. La aparición de algunos problemas relacionados con un hardware defectuoso, ha provocado que no se pudieran cumplir los plazos previstos en la integración de este hardware en el sistema, e impidiendo concretamente que la etapa de configuración e integración de la red OpenWSN en el sistema se hiciera en el tiempo previsto.

Esto ha obligado a cambiar la planificación prevista, empleando el tiempo de espera de sustitución de este hardware en la configuración y pruebas del bloque sensor y de la plataforma theThings.io, cuyo funcionamiento podía ser validado sin la necesidad de tener formada esa red OpenWSN y sincronizadas las diferentes placas OpenMote. Así, una vez repuesto este hardware defectuoso, se pasó a configurar esta red y a hacer pruebas con el sistema ya completo, y aunque con un retraso de un par de semanas con respecto a la fecha inicial, se ha conseguido terminar el trabajo técnico de manera satisfactoria.

Además de este imprevisto técnico, también se han llevado a cabo algunos cambios respecto a la planificación inicial, con el objetivo principal de simplificar algunos aspectos del prototipo que, aunque se consideran útiles, han debido ser pospuestos para garantizar el cumplimiento de otros mucho más esenciales en los objetivos del TFM. Así, al inicio del proyecto se pretendía que la aplicación software del bloque de procesamiento tuviera una interfaz gráfica que permitiera al usuario arrancar el sistema y consultar el estado del mismo, pero ante la falta de tiempo, esta interfaz gráfica ha sido sustituida por un terminal de consola, con mucha menos usabilidad pero suficiente para los objetivos de este prototipo. Además, otra opción que se pretendía incluir era la posibilidad de importar los datos almacenados en la plataforma theThings.io para poder mostrarlos en esta interfaz gráfica, pero esta prestación también ha debido ser excluida del prototipo por los mismos motivos comentados anteriormente.

5.2. LÍNEAS FUTURAS

Mediante este apartado se pretende recoger una serie de aspectos que podrían ser considerados y estudiados para ser añadidos en una futura versión mejorada del prototipo desarrollado en este sistema. Algunos de ellos ya han sido comentados en el capítulo anterior de conclusiones, ya que eran prestaciones que se pretendía incluir en este prototipo, pero que finalmente no han sido incluidos por cuestiones de tiempo y cumplimiento de plazos.

- **Dispositivo móvil como unidad central**. El prototipo desarrollado cuenta con un ordenador portátil como elemento central del sistema, y en el que reside la aplicación central de procesamiento basada en el lenguaje python. En un sistema plenamente funcional, sería totalmente recomendable sustituir este ordenador portátil por un dispositivo de menor coste y consumo, y más manejable en cuanto a tamaño y movilidad, por ejemplo un Smartphone o una Raspberry Pi.

- **Interfaz gráfica en la aplicación central de procesamiento**. Actualmente, la aplicación central cuenta con una interfaz en modo terminal, en el que se muestran informaciones básicas acerca del estado del sistema y de las medidas tomadas. En un desarrollo futuro del sistema sería preciso incorporar una interfaz gráfica que mejorara la experiencia de usuario y además incorporara opciones adicionales de funcionamiento

- **Mayor aprovechamiento de plataforma theThings.io**. En el prototipo actual, la comunicación con la plataforma theThings.io es en un solo sentido, únicamente recibiendo los datos desde el bloque de procesamiento central para su almacenamiento. En una futura versión, se podría implementar la opción de que desde el sistema se pueda consultar a esta plataforma para recuperar datos almacenados y mostrárselos al usuario, sin que éste tenga la necesidad de tener que conectarse a la plataforma vía web. Esta prestación pretendía ser incluida en este prototipo desarrollado, pero por cuestiones de tiempo ha debido ser descartada.

- **Bloque sensor más compacto**. Para este prototipo, se ha diseñado un bloque sensor que incluye un módulo que facilita la conexión del sensor ACS712 a la red, evitando tener que manipular la toma de corriente que se pretende monitorizar. Aunque de mucha utilidad para los objetivos que se pretendían en este TFM, este módulo provoca que el tamaño del bloque sea demasiado grande y poco manejable, por lo que para un sistema más desarrollado sería preciso diseñar algo más compacto y de mayor movilidad, además de con una mejor presentación de la circuitería existente, aspecto que en este proyecto no se ha cuidado tanto al tratarse de una fase de pruebas y prototipo.

- **Escalado del sistema**. El prototipo desarrollado en este TFM posibilita la monitorización de una toma de corriente, lo cual permite obtener resultados y conclusiones suficientes en cuanto a los objetivos previstos para este proyecto. Sin embargo, para un sistema

futuro, un objetivo más real y útil sería poder monitorizar el consumo de una vivienda completa, o al menos una parte más amplia de ésta que una simple toma. Para ello, sería necesario diseñar un sistema con varios bloques sensor, o bien implementar este bloque en el cuadro general de la vivienda, para así monitorizar la corriente suministrada a toda la casa.

6. GLOSARIO

ADR: Adaptive Data Rate

AES: Advanced Encrytion Standard

AIOTI: Alliance for Internet of Things

AMCA: Asynchronous multi-channel Adaptation

CoAP: Constrained Application Procotol

CRC: Cyclic Redundancy Check

CSMA-CA: Carrier sense multiple access with collision avoidance

DBPSK: Differential Binary Phase Shift Keying

DSME: Deterministic and Synchronous Multi-channel Extension

DSSS: Direct Sequence Spread Spectrum

EB: Enhanced Beacon

ETSI: European Telecommunications Standards Institute

FCS: Frame Check Sequence

FFD: Full Function Device

GFSK: Gaussian Frequency Shift Keying

GPIO: General-purpose input/output

IE: Information Element

IEEE: Institue of Electrical and Electronics Engineers

IERC: European Reseach Cluster on the Internet of Things

IoT: Internet of Things

IoTC: Internet of Things Consortium

IP: Internet Protocol

IPv6: Internet Protocol versión 6

ISM: Industrial, Scientific and Medical ISM radio band

JTAG: Join Test Action Group

LAN: Local Area Network

LE: Low Energy

LLDN: Low Latency Deterministic Networks

MAC: Medium Access Control

M2M: Machine-to-machine

NAT: Network Address Translation

OTAA: Over The Air Activation

PAN: Personal Area Network

RFD: Reducen Function Device

RFID: Radio-Frequency Identification

RPMA: Random Phase Multiple Access

SHF: Super High Frequency

TFM: Trabajo Final de Máster

TSCH: Time Slotted Channel Hopping

UNB: Ultra Narrow Band

WPAN: Wireless Personal Area Networks

WSN: Wireless Sensor Network

6LoWPAN: IPv6 over Low power Wireless Personal Area Networks

7. REFERENCIAS

[1] Gubbi, J. and others, «Internet of Things (IoT): A vision, architectural elements, and future directions,» Future Generation Computer Systems, nº 29, pp. 1645-1660, 2013.

[2] Rose, K., Eldridge, S. and Chapin, L., «The Internet Of Things: An Overview,» ISOC, 2015.

[3] «The Internet of Things (IOT),» de Infocomm Media Development Authority.

[4] «Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (WPANs),» de Information technology—Telecommunications and information exchange between systems—Local and metropolitan area networks—Specific requirements—, IEEE, 2006.

[5] De Guglielmo, D., Anastasi, G. and Seghetti, A., «From IEEE 802.15.4 to IEEE 802.15.4e: A Step Towards the Internet of Things,» Springer International Publishing, 2014.

[6] IEEE Std 802.15.4e™-2012, «Part 15.4: Low-Rate Wireless Personal Area Networks (LR-WPANs) - Amendment 1: MAC sublayer,» IEEE Computer Society, 2012.

[7] «Sitio Web SIGFOX,» [En línea]. Available: https://www.sigfox.com.

[8] «Sitio Web de LoRa,» [En línea]. Available: https://www.lora-alliance.org/.

[9] «Sitio Web de la tecnología Ingenu,» [En línea]. Available: https://www.ingenu.com/.

[10] «Sitio Web de OWL,» [En línea]. Available: http://www.theowl.com/.

[11] «Registrador de consumo ENVI,» [En línea]. Available: http://www.cliensol.es/index.php/detallesenvi.

[12] «Tutorial SCONS,» Proyecto OpenWSN en atlassian.net, [En línea]. Available: https://openwsn.atlassian.net/wiki/display/OW/SCons.

[13] «SCONS integrado en Eclipse,» Proyecto OpenWSN en atlassian.net, [En línea]. Available: https://openwsn.atlassian.net/wiki/display/OW/Using+Scons+as+Eclipse+Builder.

[14] «Tutorial de OpenWSN en Windows,» Proyecto OpenWSN en atlassian.net, [En línea]. Available: https://openwsn.atlassian.net/wiki/display/OW/Kickstart+Windows.

[15] «Tutorial de COAP con OpenWSN,» openwsn.atlassian.net, [En línea]. Available: https://openwsn.atlassian.net/wiki/display/OW/CoAP+Interaction.

[16] «Añadir aplicación COAP al OpenWSN,» Proyecto OpenWSN en atlassian.net, [En línea]. Available: https://openwsn.atlassian.net/wiki/display/OW/Adding+a+CoAP+app.

[17] «Repositorio Github librería Python plataforma theThings.io,» [En línea]. Available: https://github.com/theThings/thethings.iO-python-library.

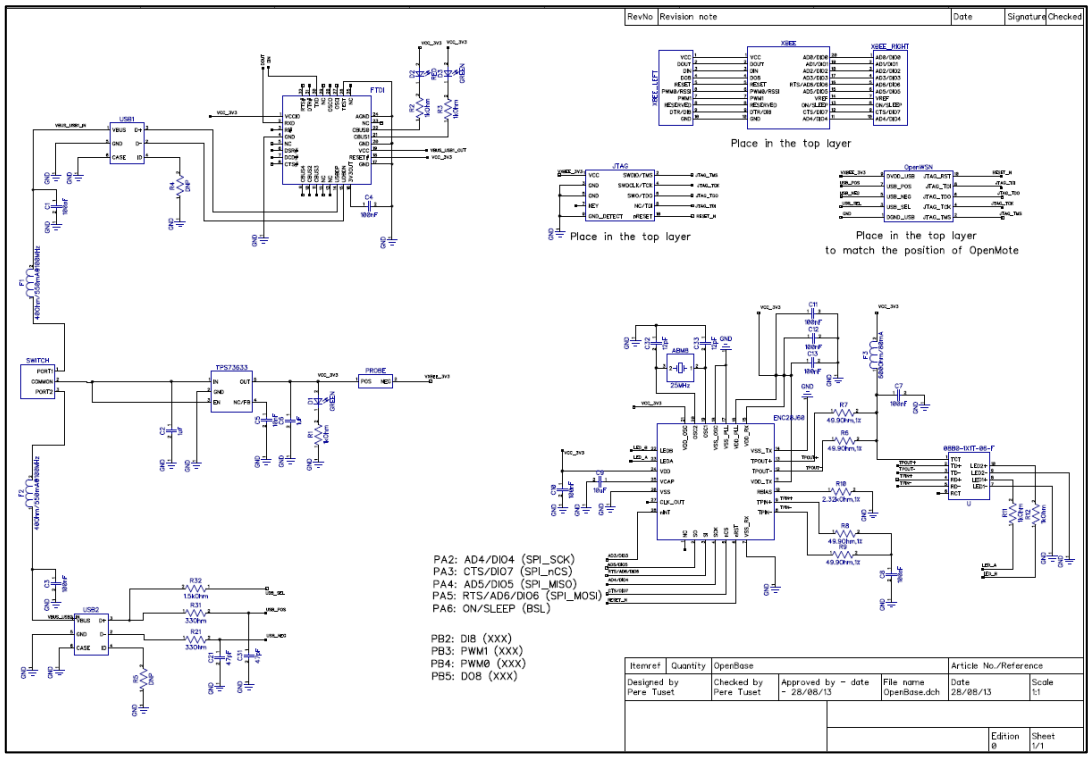
[18] «Datasheet Sensor ACS712,» [En línea]. Available: https://www.allegromicro.com/~/media/files/datasheets/acs712-datasheet.ashx.

[19] «Sitio Web de plataforma Twilio,» [En línea]. Available: https://www.twilio.com/?lang=null.

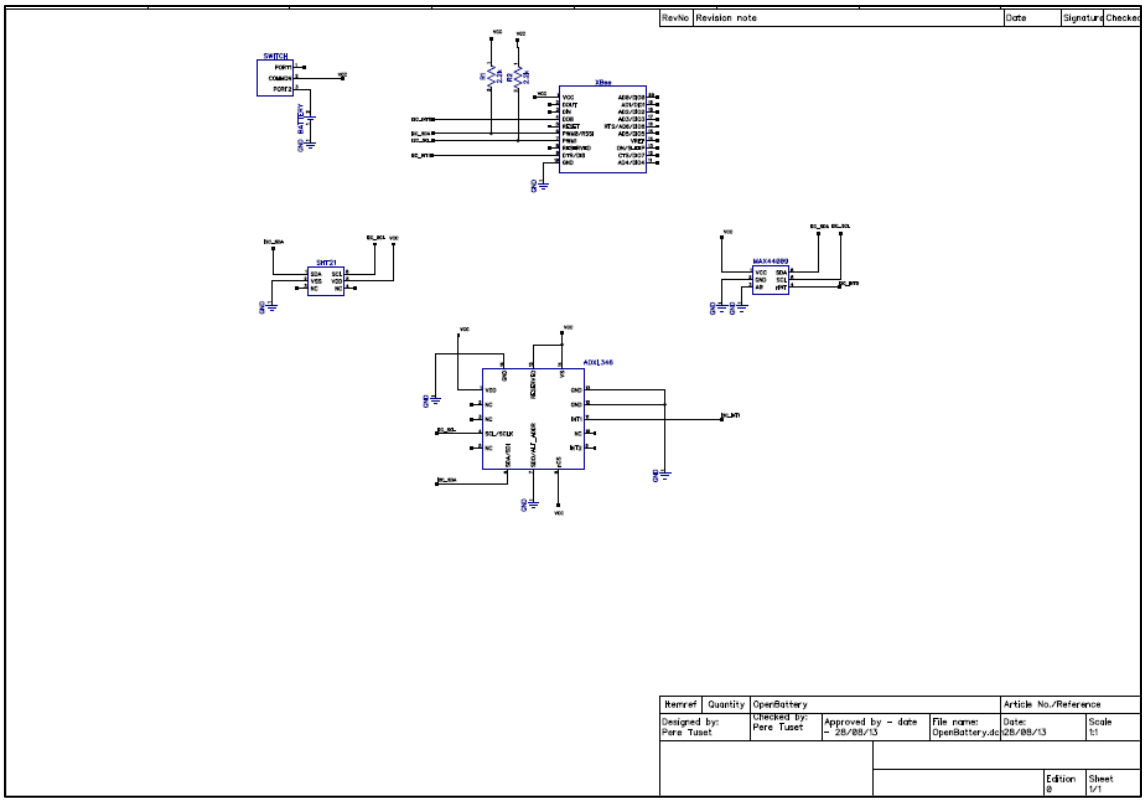
[20] «Tutorial para añadir aplicaci,» [En línea]. Available: https://openwsn.atlassian.net/wiki/display/OW/Adding+a+CoAP+app.

8. ANEXOS

8.1. ESQUEMA OPENBASE



8.2. ESQUEMA OPENBATTERY



8.3. ESQUEMA OPENMOTE-CC2538