Universitat Oberta de Catalunya
uoc.edu

PoC de un sistema Open Source, multiusuario y cooperativo de medición de consumo eléctrico

Xavier del Peso Ribera
Grado en Ingeniería Informática
Área: Redes Abiertas

Consultor: Amadeu Albós Raya
Profesor/a responsable: Joan Manuel Marqués Puig

08 de Junio de 2021

FICHA DEL TRABAJO FINAL

Título del trabajo:	*PoC de un sistema Open Source, multiusuario y cooperativo de medición de consumo eléctrico*
Nombre del autor:	*Xavier del Peso Ribera*
Nombre del consultor/a:	*Amadeu Albós Raya*
Nombre del PRA:	*Joan Manuel Marqués Puig*
Fecha de entrega (mm/aaaa):	06/2021
Titulación:	*Grado en Ingeniera Informática*
Área del Trabajo Final:	*Redes Abiertas*
Idioma del trabajo:	*Español*
Palabras clave	*Power Metering, IoT, Telemetría*

Resumen del Trabajo (máximo 250 palabras):

La medición del consumo eléctrico siempre ha sido un tipo de dato complicado de medir y desde el punto de vista de un usuario corriente, solamente era posible mediante la factura mensual. Pero con los tiempos que corren, donde cada vez preocupa más aprovechar eficientemente los recursos, empieza a ser necesario poder hacerlo.

Este trabajo explora, mediante una prueba de concepto, crear una plataforma que permita a usuarios y pequeñas empresas, sin grandes conocimientos técnicos, ni tampoco un gran presupuesto, medir con gran detalle el consumo de sus sistemas y aparatos eléctricos.

Para alcanzar este objetivo, el diseño de la plataforma se ha separado en dos partes claramente diferenciadas.

Por un lado, los usuarios deben poder adquirir sensores de bajo costo e incluso poder crear los suyos e integrarlos en la plataforma. En el PoC se han montado y estudiado dos tipos de clientes, uno que permite hacer mediciones agrupadas desde el cuadro eléctrico y otro que permite hacer mediciones de elementos individuales desde un enchufe.

Por otro lado, el diseño de la plataforma requería un servidor central donde almacenar y poder hacer reportes. Para no trasladar coste a los clientes, se ha diseñado en un formato cooperativo, para que usuarios con recursos y conocimientos, pongan a disposición del resto, el servidor donde dirigir los datos.

Con PoC, se ha validado que es posible hacer mediciones de consumo de gran detalle con clientes económicos, y que el servidor, es capaz de trabajar con varios usuarios simultáneamente, aislando completamente los datos de unos y otros.

Abstract (in English, 250 words or less):

The power metering has always been something hard to measure. This is because, from a user standpoint, it was only possible by checking the electric bill. But nowadays, with important topics arising like global warming and the efficient use of the energy, power metering is becoming a need.

This thesis explores, as a PoC, the creation of a platform that enables users and small companies, with neither specialized knowledge, technical skills nor budget, a way to measure their own electrical installations and devices.

To reach this goal, the architecture of the platform has been split into two much differentiated parts.

On one side, the users, in a client role, should be able to acquire low-cost sensors or even design their own to integrate them into the platform. In this PoC, two types of clients have been investigated. The first one, designed to measure from the electrical distribution panel, and the second, designed to be plugged into sockets and measure single electrical appliances.

On the other side, the platform required a server where host and store all the data from sensors. To avoid extra costs to the clients, the server has been proposed as a cooperative service, where a user or community would provide their own resources to others for free.

This PoC has validated that it is possible to measure any kind of device with high resolution, without implying high costs related to technology. Furthermore, the server is able to host all the data for all users at the same time, while keeping everything completely isolated.

Table of Contents

Lista de figuras

50. Regleta monitorizada por el sensor Sonoff Pow (en el recuadro)
51. Termo eléctrico monitorizado por un enchufe inteligente de la plataforma Tuya
52. Ejemplo de un dispositivo en el módulo Input. Se reciben 15 datos, pero se guardan solo los 8 seleccionados (RP1-7 y Vrms)
53. Ejemplo de un procesamiento para convertir a valor absoluto sin usar la operación absoluto predefinida
54. Es posible usar los dos lados del eje y para representar valores que trabajan en distinta escala o distinta unidad
55. Tabla de valores de cada feed representado con sus medias, máximos y mínimos, así como total de Wh acumulado en ese intervalo.
56. Consumo de la nevera en 24h en Julio ja sido de 3711Wh
57. Consumo de la misma nevera en Diciembre, ahora 1410Wh
58. El consumo de dos neveras contrastado
59. Figura 60. Artículo del periódico el ABC
60. Consumo total de un domicilio contrastado con una nevera antigua.
61. Comparación del consumo de una nevera de ocasión y el consumo general
62. Contraste del consumo general, el termo y una regleta donde hay un PC

1 Introducción

1.1 Contexto y justificación del TFG

En los últimos años está habiendo una creciente preocupación sobre cómo usamos la electricidad o, mejor dicho, la energía. Es de sobra conocido, que durante muchísimo tiempo hemos hecho un uso desmesurado de los recursos, pensando que son ilimitados y sin llegarnos a preocupar cómo se obtienen o si está teniendo efectos negativos para el planeta. Pero estamos en un punto donde se empiezan a notar las consecuencias de llevar ese ritmo. El agotamiento de los recursos naturales del planeta y el calentamiento global entre otros, son algunos de ellas.

Afortunadamente, en los últimos años, está surgiendo una concienciación general sobre cómo conseguir reducir nuestra huella. El uso de energía limpia y ganar eficiencia energética, consiguiendo que nuestros dispositivos, haciendo lo mismo que hacían antes, consuman menos, son ahora dos temas muy importantes de la actualidad.

Porque una cosa es muy clara, no podemos simplemente desconectar o dejar de depender de todos esos dispositivos. Eso significaría parar el progreso y volver al pasado, algo que no es posible.

La tecnología llega a nuestras vidas y penetra con mucha facilidad en el tejido social, lo que parece ser contraproducente respecto a lo que queremos conseguir, ya que, debido al boom que estamos experimentando en aumento de dispositivos eléctricos y digitales, se están disparando nuestras necesidades eléctricas

El IoT[1] y la interconexión masiva son un buen ejemplo de ello. Donde antes había herramientas manuales, ahora existen varios dispositivos para ello que además son eléctricos, y en la cocina encontramos un buen ejemplo, con el robot de cocina, la panificadora, la arrocera, etc. Además, son también inteligentes, generan información sin parar y se pueden controlar remotamente desde el último gran invento de la humanidad, el smartphone.

Para transportar y almacenar esta información, necesitamos mejores redes de comunicaciones y más centros de datos, que son monstruos devoradores de energía. Después, con las grandes corporaciones y la ciencia de datos, conseguimos transformar todo eso en información, que hace girar la rueda otro poco más, en busca del nuevo producto estrella mundial, y así ciclo a ciclo mientras la rueda es cada vez mayor.

Pues bien, este proyecto se enmarca precisamente en estas ideas y estas realidades. El proyecto consiste en crear una plataforma, que nos permita medir el consumo eléctrico de nuestra casa o empresa de forma detallada. Idealmente, de cada dispositivo individualmente y también del conjunto, para, mediante el análisis de esto, conocer cómo es nuestra huella digital y ver si la podemos mejorar. Actualmente, las compañías eléctricas nos enseñan lo que hemos consumido en un mes, pero no nos dan ningún detalle, de dónde viene ese consumo. Si la factura de luz sube en verano, podemos pensar que es del aire acondicionado, pero lo que no sabemos es que, la nevera puede llegar a consumir 2 o 3 veces más

[1] Del inglés Internet of Things (IoT)

en verano que en invierno. Saber estas cosas nos ayudará a entender nuestra factura eléctrica y nos ayudará a saber qué podemos mejorar.

Además, el proyecto viene acompañado de un magnifico momento, ya que tanto el IoT como la interconectividad masiva están en auge. Gracias a la masificación del IoT, estamos experimentando un abaratamiento de los costes muy grandes que propicia que cada elemento, por pequeño y barato que sea, se haga "smart"[2] y genere nuevos paradigmas.

La intercomunicación también se está viendo afectada enormemente por este nuevo status quo. Ahora podemos disponer de WiFi prácticamente en cada domicilio, y si no lo hay, existe cobertura de 4G en prácticamente todo el globo terráqueo y a precios asequibles. Y no solamente esto, sino que se están desarrollando e implantando nuevas tecnologías de comunicaciones específicas para el IoT, como LoRaWan[i], una red abierta y completamente gratuita de largo alcance y bajo consumo, que cada día se expande más y más y que hoy en día, ya cuenta con cobertura, incluso en entorno rural, en toda Europa y Estados Unidos.

Pues bien, como se ha dicho, el proyecto consiste en crear una plataforma/ecosistema abierto de medición de consumo en el que, de forma colaborativa, una persona u organización ofrecerá un servidor donde se guardará toda la información, y donde los usuarios, podrán adquirir sus dispositivos IoT de medición de consumo, instalarlos en sus casas, y enviar los datos al servidor, para poderlos visualizar a posteriori. Al final la tecnología también nos puede ayudar a cuidar de nuestro pequeño y único planeta.

1.2 Objetivo principal del proyecto

- El objetivo principal del proyecto es obtener una plataforma de medición de consumo eléctrico de carácter comunitario/colaborativo. En ella, cualquier persona debe poder ser usuario mediante el uso de los dispositivos descritos en este proyecto, como otros de fabricación propia, así como el uso de redes privadas WiFi/Ethernet/4G o comunitarias tales como LoraWan.
- El segundo objetivo principal del proyecto es mostrar cómo, gracias a la obtención de datos, es posible obtener información sobre nuestros hábitos y cómo se pueden aplicar mejoras sobre ellos para reducir nuestro consumo.

1.2.1 Objetivos específicos del proyecto

- Implementar un servidor central de carácter comunitario donde cada miembro pueda volcar datos para ser representados y contrastados.
- Mostrar cómo es la arquitectura y tecnología interna del servidor para que sea fácil de modificar o mejorar.
- Definir algunas de las tecnologías de transmisión compatibles con esta plataforma, tales como WiFi, Ethernet, 4/5G y LoraWan[3].
- Crear al menos dos sensores de medición de consumo eléctrico funcionales y generar mediciones reales de varios entornos, dispositivos y usuarios.
- Hacer una recopilación del consumo eléctrico de un domicilio y de algunos electrodomésticos y analizar cómo se podría optimizar su uso.

[2] Del inglés, inteligente en referencia a los dispositivos inteligentes.
[3] LoRaWan, del inglés, Long Range Wan. Es un tipo de protocolo de transmisión.

1.3 Enfoque y método seguido

El proyecto se desarrolla como una prueba de concepto, o PoC[4], de una plataforma de medición de consumo de ámbito cooperativo. El objetivo es explorar si posible montar una infraestructura viable técnicamente en el que distintos usuarios, independientemente de su situación y características particulares, puedan usar esta plataforma e incluso adaptarla a sus necesidades.

Para lograr un correcto desarrollo del PoC dentro de la temática general, se ha definido una serie de fases e hitos a alcanzar de una forma consecutiva:

Planteamiento de la necesidad y objetivos del proyecto: Se define porque puede ser útil o necesario disponer de un sistema de monitorización de consumo eléctrico y en que contextos puede ser vialmente posible instalarlo.

Revisión del estado del arte de la temática: En esta fase, se debe hacer una investigación profunda de las tecnologías o avances en la materia. Incluso, aunque sea un producto de creación nueva, es muy posible que el producto tome referencias o se vea influenciado por otros productos del mercado.

Definición de los requisitos y análisis de la infraestructura necesaria: Antes de empezar a diseñar e implementar el producto, es sumamente importante definir el alcance de este, los requisitos y las restricciones que se deben cumplir. Esta es una fase crítica de cara a tener un producto claramente definido o de otra forma, es posible empezar a difuminar los objetivos y que el producto empiece a crecer sin control. Una vez se han definido el alcance, los requisitos y las restricciones, es necesario hacer un análisis, de alto nivel, de la infraestructura o componentes necesarios para hacer el proyecto. Esto es precisamente la antesala a la fase de arquitectura y diseño.

Diseño y arquitectura del objeto del estudio: Se describe la arquitectura global del producto con todos sus componentes más relevantes. Además, se detallan todos los aspectos de bajo nivel en los que es importante profundizar para poder explicar adecuadamente aspectos clave de la estructura interna del desarrollo.

Implementación y confección de los escenarios de pruebas del PoC: Se detalla la implementación del producto, así como los escenarios donde se instalarán los dispositivos a modo de obtener datos utilizables. La fase de implementación debe describir los aspectos más importantes y particulares del producto. Los anexos se pueden incorporar al trabajo para añadir más información que pueda ser útil pero que no sea clave para la comprensión del proyecto.

Análisis de los resultados obtenidos y elaboración de conclusiones: Se hace un análisis de los datos obtenidos por el sistema. Finalmente se hace una valoración del PoC para saber si se han logrado los objetivos marcados y si ha habido cambios respecto al planteamiento inicial, así como si se han encontrado puntos que sería conveniente mejorar en una segunda fase.

1.4 Planificación del Trabajo

El proyecto está pensado para distribuirse entre Febrero y Junio. Durante estos meses, será posible ir paralelizando algunas tareas, pero siempre será necesario ir manteniendo una consistencia global. La

[4] Del inglés “Proof of Concept”, que significa prueba de concepto.

implementación puede ser que se empiece antes de terminar el diseño, pero en ningún caso empezará antes que este.

La documentación del proyecto se extenderá durante todo el proyecto y será paralela al resto de tareas.

	Name	Start	Finish
1	**Definición del proyecto**	**2/17/21 8:00 AM**	**3/11/21 1:00 PM**
2	Presentación del problema	2/17/21 8:00 AM	2/23/21 5:00 PM
3	Analisis de una solución	2/24/21 8:00 AM	3/1/21 5:00 PM
4	Definición de los objetivos	2/26/21 8:00 AM	3/4/21 9:00 AM
5	Definición del alcance	2/27/21 8:00 AM	3/5/21 3:00 PM
6	Definición de las pruebas	3/1/21 8:00 AM	3/8/21 5:00 PM
7	Planificación	3/6/21 8:00 AM	3/11/21 1:00 PM
8	Resultados esperados	3/1/21 8:00 AM	3/4/21 10:00 AM
9	**Fase de investigación**	**3/2/21 8:00 AM**	**3/11/21 5:00 PM**
10	Busqueda de una solución open so	3/2/21 8:00 AM	3/11/21 5:00 PM
11	Investigación sobre protocolos de	3/2/21 8:00 AM	3/6/21 5:00 PM
12	Comparativa con otros productos	3/3/21 8:00 AM	3/9/21 5:00 PM
13	Análsis de la plataforma	3/8/21 8:00 AM	3/13/21 5:00 PM
14	**Diseño de la Plataforma**	**3/15/21 8:00 AM**	**4/17/21 5:00 PM**
15	Flujo de comunicaciones	3/15/21 8:00 AM	3/23/21 1:00 PM
16	Diseño del servidor	3/16/21 8:00 AM	4/6/21 3:00 PM
17	Diseño de los clientes	3/19/21 11:00 AM	4/17/21 5:00 PM
18	**Implementación**	**3/16/21 8:00 AM**	**4/6/21 1:00 PM**
19	Instalación y parametrización del s	3/16/21 8:00 AM	3/22/21 3:00 PM
20	Configuración de los clientes	3/19/21 11:00 AM	4/3/21 5:00 PM
21	Instalación de los clientes en domi	3/20/21 8:00 AM	4/6/21 1:00 PM
22	**Recogida de datos**	**4/5/21 8:00 AM**	**5/10/21 4:00 PM**
23	Visualización de datos simples	4/5/21 8:00 AM	4/9/21 5:00 PM
24	Contraste de datos dentro de un d	4/5/21 8:00 AM	4/12/21 5:00 PM
25	Analsis de datos	4/9/21 8:00 AM	4/26/21 5:00 PM
26	Conclusiones del analsis	4/29/21 4:00 PM	5/10/21 4:00 PM
27	Elaboración de la Memoria	2/26/21 8:00 AM	5/24/21 5:00 PM
28	Revisión de la Memoria	5/13/21 8:00 AM	6/14/21 5:00 PM
29	Grabación de una presentación	6/14/21 8:00 AM	6/17/21 5:00 PM
30	Defensa	6/21/21 11:00 AM	6/25/21 11:00 AM

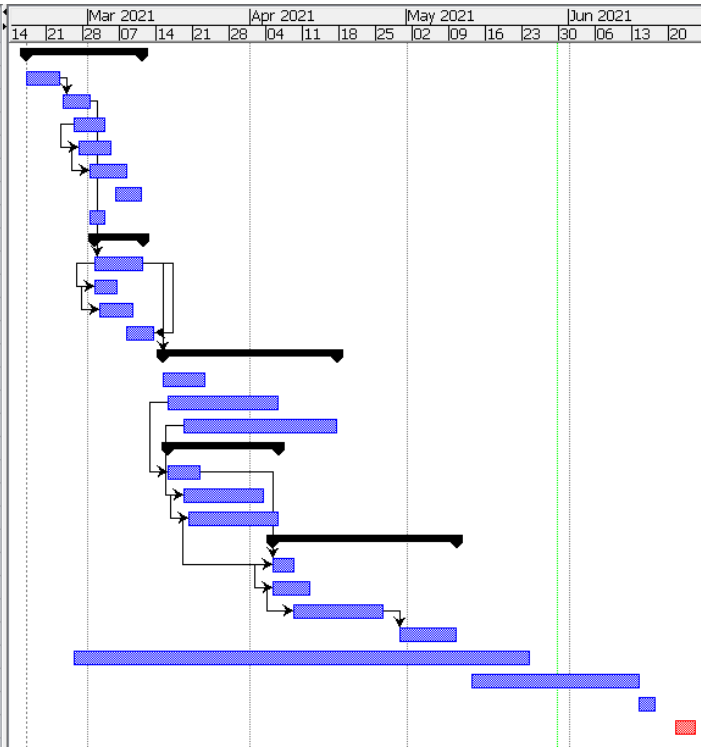


Figura 1. *Diagrama de Gantt que muestra la planificación del proyecto.*

1.5 Breve sumario de productos obtenidos

Este proyecto consta de 3 partes fundamentales, que, tras ser desarrolladas en este estudio, se convertirán en el resultado del mismo.

1.5.1 Servidor EmonCMS

El primer producto entregable es el servidor, es decir, la parte centralizada de la plataforma. Es aquí donde se reciben y almacenan todos los datos de la telemetría de los usuarios. Es alcanzable por internet y por varios protocolos de transmisión. El servidor es expandible y se pueden integrar varios tipos de clientes y sensores en esta plataforma de medición de consumo, lo que conforma un entorno muy heterogéneo (siempre y cuando estos reciban los datos en el formato adecuado).

En este proyecto, el servidor se entrega con todos los roles y funcionalidades de la plataforma en una sola máquina virtual. No obstante, dado que se utiliza software de código abierto, múltiples configuraciones son posibles, desde todos los roles del servidor en una Raspberry Pi[5] o similar, hasta una arquitectura distribuida en alta disponibilidad con máquinas virtuales, físicas y/o docker.

[5] Raspbery Pi, placa de desarrollo muy concida que es como un ordenador en pequeño formato

En concreto, este producto, si bien todos los usuarios lo usarán, no está enfocado para ser instalado o configurado para los que tienen un rol de usuario de la plataforma. Este producto, está orientado a personas, administradoras de la plataforma, que, sin tener un objetivo comercial o lucrativo sino cooperativo, quieran poner a disposición del mundo un servicio de medición de consumo donde cualquiera puede introducir datos.

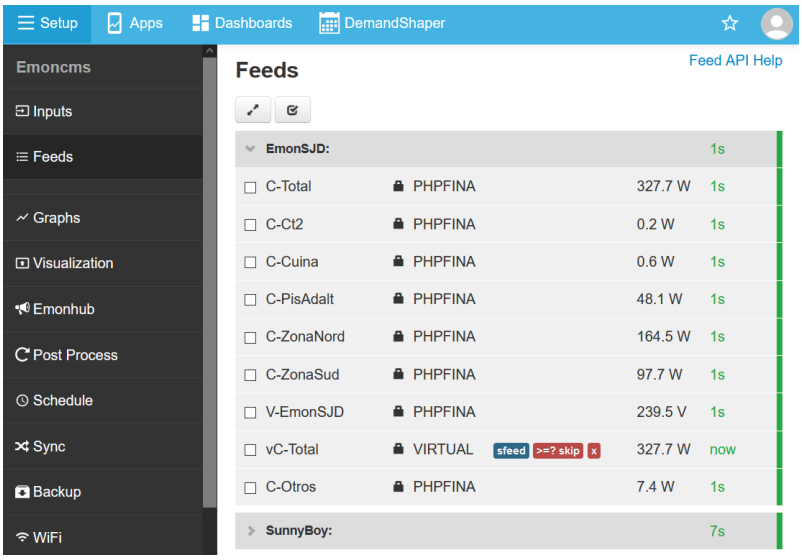


Figura 2. *Captura de pantalla del servidor EmonCMS*

1.5.2 Clientes

El segundo producto obtenido del TFG son, en realidad, dos dispositivos y están orientados a todas aquellas personas que tengan interés en conocer el consumo y patrón de consumo eléctrico de su domicilio o negocio y que, por tanto, serán clientes de la plataforma de medición de consumo.

El primero de ellos es una Raspberry Pi[6] con un Hat[7] específico que está pensado para monitorizar el consumo eléctrico directamente desde el cuadro eléctrico del domicilio y que, por tanto, debe ir instalado dentro del mismo cuadro. El objetivo es medir el consumo eléctrico total de la casa e incluso de las distintas ramificaciones de una instalación eléctrica común (línea de enchufes, línea de iluminación, línea de cocina, etc.).

El segundo, en cambio, está pensado para medir el consumo eléctrico de elementos simples que van conectados a un enchufe, tales como una nevera, un ordenador, o incluso una regleta de enchufes a la que se conectan varios dispositivos. Su diseño, en forma de enchufe inteligente, permite la fácil instalación y movilidad.

[6] Ordenador de bajo coste en formato de placa de desarrollo muy popular.
[7] Hat: módulos de expansión específicos para Raspberry Pi.

1.5.3 Breve análisis de los datos obtenidos

El último producto obtenido son los resultados de la monitorización de consumo de varios elementos, tales como una nevera o un calentador, así como un análisis de estos datos, para demostrar que tener este tipo de información puede llegar a ser muy últil.

1.6 Descripción de otros capítulos de la memoria

Los capítulos de la memoria se estructuran en los siguientes puntos:

- **Punto 7. Estado del Arte**: en este capítulo se hará un repaso del IoT y de la monitorización de consumo en un contexto global, mediante el cual el lector podrá conocer los distintos métodos y productos empleados en esta temática.
- **Punto 8. Definición de los requisitos**: En este capítulo se detallan los requisitos funcionales y no funcionales que debe cumplir el producto objeto de este trabajo
- **Punto 9. Selección de la plataforma**, Aquí se define, en líneas generales, los componentes que formarán el producto final, que deben permitir alcanzar los objetivos del proyecto, así como respetar todas las restricciones definidas.
- **Punto 10. Arquitectura y diseño**: En esta fase se incluyen todos los detalles de diseño necesarios para la comprensión de los distintos componentes, que los usuarios deberán conocer si los desean implementar. Dado que en el proyecto hay dos públicos objetivos, cada uno con su propio rol, es posible que encuentren que la información no les es necesaria. En cualquier caso, será de utilidad para adquirir un conocimiento más profundo.
- **Punto 11. Materiales usados**: Se hace un listado y descripción de los componentes usados en el proyecto así como una BOM[8].
- **Punto 12. Implementación**: En esta fase se detalla los aspectos más relevantes de la implementación. Igual que en la fase de arquitectura, hay dos tipos de implementación, la orientada al servidor que tiene un público objetivo y la orientada a cliente, con el otro público objetivo.
- **Punto 13. Análisis de los resultados**: Se hace un breve análisis de los datos obtenidos con la monitorización de consumo y realizan algunas conclusiones de estos análisis.
- **Punto 14. Conclusión**: Finalmente, se hace una valoración del proyecto, de los resultados obtenidos y de si el PoC cumple con los objetivos marcados.
- **Punto 15. Anexos**: Aquí añadimos información extra sobre otros usos del sistema y otros componentes que pueden usarse en conjunción con la plataforma, pero que no alteran ni son importantes para la comprensión del proyecto.

[8] Del inglés, bill of materials. Es una lista de todos los componentes que se deben comprar o adquirir.

2 Estado del arte

El estado del arte es un repaso sobre la actualidad de las tecnologías y conceptos presentes en la temática de este proyecto, especialmente aquellas más relevantes o de reciente aparición.

2.1 Internet of Things:

Internet of Things es un concepto que se traduce como Internet de las cosas, aunque su traducción no expresa la amplitud de lo que representa. Se calcula que cada persona está rodeada entre 1000 y 5000 objetos. El IoT pretende que cada uno de esos objetos sea inteligente y que no esté aislado, sino que sea operable remotamente y genere datos.

El mundo del IoT, en los últimos años, está experimentando un crecimiento exponencial gracias a la aparición de proyectos de diversa índole y el crecimiento de muchísimas empresas y profesionales especialistas en este sector. Esto propicia el desarrollo de nuevos chips y tecnologías enfocadas al IoT, lo que nuevamente retroalimenta nuevos proyectos y nuevas necesidades.

A continuación, se describirán algunos de estos elementos como los componentes electrónicos, tecnologías de transmisión y protocolos de red ampliamente usados y que se han convertido en una piedra angular en el mundo del IoT.

2.2 Tecnologías de transmisión en IoT:

Las tecnologías de transmisión permiten la integración entre distintos sistemas autónomos. Sin ellas, todo elemento sería una isla que solamente se podría operar localmente. Existen muchas tecnologías de comunicación y hay muchas capas en cada una de ellas. Las más usadas en el mundo del IoT tenemos:

2.2.1 WiFi o 802.11.

Este es posiblemente el protocolo de transmisión más extendido mundialmente, debido principalmente a que utiliza bandas de frecuencias de libre uso[9] (de 2.4Ghz y 5Ghz). Su norma determina que se utilizan 20Mhz para cada canal, tanto en 2.4 como en 5Ghz, hasta llegar al 13 o 14 canal en 2.4Ghz (momento en el que termina la concesión de frecuencia de libre acceso) o hasta el canal 190 en 5Ghz (en 5Ghz, el rango de frecuencia libre es mucho mayor y con nuevas revisiones se va añadiendo más rango, lo que permite usar más “canales”). Además, es posible, agrupar canales dentro de una misma comunicación, ocupando desde 20Mhz hasta 160Mhz.

A diferencia de otras tecnologías, que también usan bandas de frecuencia libre (como el Bluetooth), el WiFi se integra en el modelo OSI[10], concretamente se encuentra en las capas 1 y 2, que son las capas de medio. Gracias a esto, cualquier dispositivo conectado a una red WiFi, es capaz de comunicarse mediante TCP/IP con todo el mundo.

Precisamente, los factores, licencia libre, modelo OSI integrado y bajo coste de componentes, son los pilares por los cuales surgen organizaciones cooperativas para hacer una red libre y abierta para todo el mundo, tales como Guifi.net

[9] Las frecuencias de libre uso son aquellas frecuencias del espectro, que cualquier ciudadano o empresa puede usar sin tener una licencia o permiso oficial para enviar o recibir información.

[10] Modelo OSI: Open Systems Interconnection, es un modelo de referencia para los protocolos de la red.

Guifi.net como Red IoT.

Guifi.net es una organización y una red abierta basada radioenlaces, que funcionan con el protocolo WiFi. Esta red abierta está principalmente extendida en Catalunya (ver figura 3) y creciendo en España, pero con cobertura muy limitada fuera de estos sitios. Su modelo es hacer de operador completamente gratuito gracias a la colaboración de sus miembros.

En una hipotética situación donde hubiera un gran apoyo, este proyecto podría convertirse en el mayor operador mundial gratuito que, incluso, podría ser usado como red IoT para llegar a lugares donde no existen otros operadores.

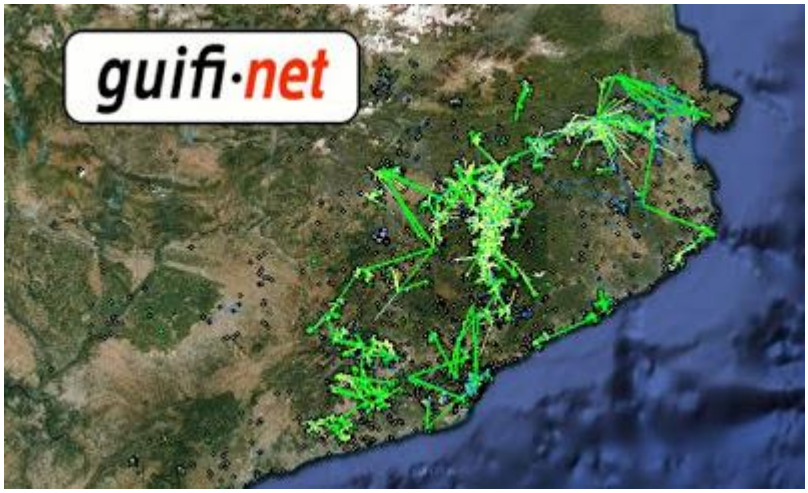


Figura 3. *Distribución y agrupación de canales en 5Ghz [Ref[ii]]*
http://biqfr.blogspot.com/2010/11/interconexion-de-usuarios-para-formar.html

No obstante, esta red, está más pensada para actuar a modo de operador, que como una red final de nodos IoT. Esto significa que difícilmente podríamos conectar un nodo Raspberry Pi u otro cliente directamente a la red de Guifi.net, algo que la organización califica como "conexiones oportunistas" y que son dañinas para la red (ver figura 4). En su lugar, sería necesario instalar un radio-enlace en la azotea dónde tengamos el cliente y ahí generar otra red Ethernet o WiFi donde conectar definitivamente nuestro cliente.

> **CUIDADO:** Las conexiones con portátiles, pda, etc, que llamamos conexiones oportunistas, se aceptan, pero ya que perjudican el buen funcionamiento de los puntos de acceso, no son aconsejables. Lo más aconsejable es conectar utilizando un dispositivo wireless fijo para tener una conexión estable, bien hecha y en condiciones. Hay un documento con más información: ¿Puedo acceder con un portátil o una PDA a la red y tener Internet? Los puntos de acceso pensados para conexiones esporádicas son aquellos que tienen por nombre de red (ssid) «guifi.net-AccesObert» o «guifi.net-AccesoAbierto».

Figura 4. *Definición en la web de Guifi.net respecto a las conexiones oportunistas [Ref][iii]*
https://guifi.net/trespasos

Algo que sí es posible por otro lado, es hacer un tándem de Guifi.net con antena LoRaWan. El tráfico LoRaWan es muy ligero, pero que necesita estar conectado a internet para enlazar las comunicaciones desde la red Lora a Internet. De hecho, ya existe un proyecto en marcha entre la organización Guifi.net y The Things Network Catalunya (TTNCat) en el que la intención es que la primera se comporte a modo de operador de la red LoRaWan. Artículos 1[iv] y 2[v].

2.2.2 3G/4G/5G:

Estas redes operan en frecuencias licenciadas. El uso de esta banda de frecuencias está regulado en cada país, ya que el Estado es el propietario del espectro radioeléctrico en todo su territorio. El estado hace subastas de fragmentos de las bandas, en los que los operadores pujan por estos fragmentos para poder usarlos para sus comunicaciones.

Como ocurre en WiFi, este protocolo de comunicaciones también permite el uso de redes TCPIP directamente desde el dispositivo cuando este monta un módem y puede registrarse en la red del operador, lo que dota a estos dispositivos, de la posibilidad de comunicarse con cualquier parte del mundo.

2.2.3 LoraWan, The Things Network (TTN), Sigfox

Lorawan

La red Lorawan es una implementación de comunicaciones para dispositivos IoT basado en el protocolo Lora y que forma parte de las redes de tipo LPWan[11], que son redes especialmente diseñadas para cubrir un gran alcance. Además, estas redes tienen un coste de energético muy bajo, lo que lo hace especialmente útil para un entorno IoT, ya que en ocasiones son dispositivos que funcionan gracias a baterías o pilas.

Los Gateways, que es como se llama a las antenas de la red LoRaWan, están instaladas en azoteas y torres, pero a diferencia de las antenas de 4/5G, que pertenecen a operadores, estas son comunitarias (ver figura 5), es decir, que es una red abierta. Por tanto, son los usuarios quienes voluntariamente adquieren y mantienen en servicio para que nodos clientes de otros usuarios o empresas se conecten. Además de aportar la antena, también proporcionan la conexión de internet para dar salida a la antena al "resto del mundo" y el resto de costes derivados del mantenimiento.

Figura 5. *LoRaWan Gateway [Ref[vi]]*
https://www.thethingsnetwork.org/marketplace/product/the-things-gateway

[11] LPWAN: Low Powered WAN

Una de las principales carencias que tiene LoRaWan, es la garantía de servicio, tanto en que una antena deje de funcionar, como que no haya cobertura en una zona por la ausencia de antenas. Esto es debido a la componente colaborativa y gratuidad de esta plataforma pero, mediante la proliferación de más y más antenas, llamados Gateways en el ecosistema TTN, esta carencia se está minimizando.

The Things Network

The Things Network[12] (TTN) no es una tecnología o protocolo, sino la organización y la plataforma de que hay detrás de la red LoRaWan y que posibilita la intercomunicación entre las redes Lora y el mundo TCP/IP. Esta organización tiene un servicio cloud[13] distribuido mundialmente, al cual, cada Gateway se registra y redirige todo el tráfico LoRaWan, para que salga a internet a través este cloud (ver figura 6). En este, cada usuario puede determinar qué hacer, cómo procesar y a dónde mandar los datos enviados por sus dispositivos (nodos en el ecosistema TTN).

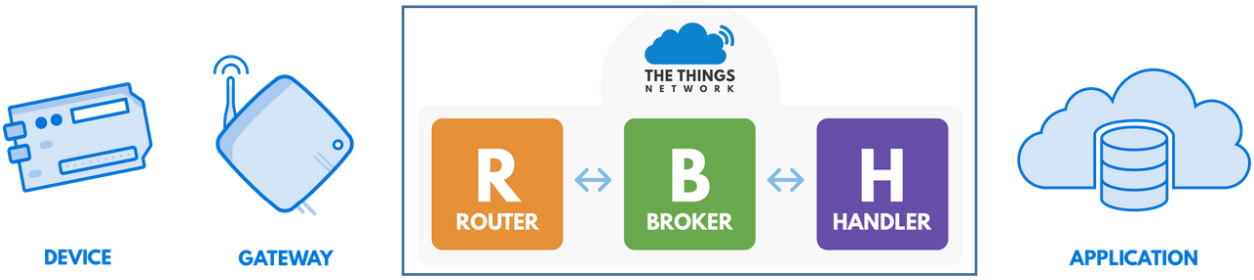


Figura 6. *Arquitectura LoRaWan y TTN en la parte central [Ref[vii]]*
https://www.thethingsnetwork.org/article/the-things-network-architecture-1

A diferencia de los gateways que son comunitarios, el servicio cloud sí que está completamente mantenido y cubierto por esta organización, con lo que la disponibilidad y el buen funcionamiento están garantizados.

Sigfox

Finalmente, Sigfox es el equivalente a la LoRaWan + The Things Network juntas (ver figura 7), pero en su vertiente privada. Por poner un símil con otras empresas, es una especie de operador o ISP[14] pero en una red LPWAN. Como partes positivas encontramos que, aquí sí que se garantiza cobertura en todo el territorio y, si una antena deja de funcionar, ellos son responsables de la reparación y restauración del servicio. También se encargan del mantenimiento y desarrollo de su plataforma cloud. Como es obvio, en contraprestación, tenemos que esta red sí tiene un coste, que se contabiliza anualmente por nodo conectado.

[12] The Things Network: organización que gestiona el protocolo y las comunicaciones LoRaWan
[13] Sistema de servidores que actúan como un servicio distribuido y alcanzable desde internet.
[14] Internet Service Provider. Del inglés, significa operador de internet.

Figura 7. *Arquitectura de Sigfox [Ref[viii]]*
https://www.researchgate.net/figure/SigFox-Network-Architecture_fig6_329936193

2.3 Dispositivos del mundo IoT:

A pesar de que los dispositivos IoT se cuentan por miles y que la configuración electrónica es única en cada uno de ellos, ya que suele hacerse a medida, en esta sección solo cubriremos aquellas placas de desarrollo más conocidas y que se utilizan a nivel de usuario.

2.3.1 Arduino:

Seguramente la placa de desarrollo más conocida mundialmente. Esta placa está pensada para construir o transformar objetos del mundo real en objetos que estarían dentro del paraguas de IoT. Existen múltiples versiones con diferentes características y tamaños. Las placas Arduino tienen un microcontrolador reprogramable con una serie de pines analógicos y digitales (ver figura 8) a través de los cuales interactúan con los distintos sensores o elementos externos con los que se integra.

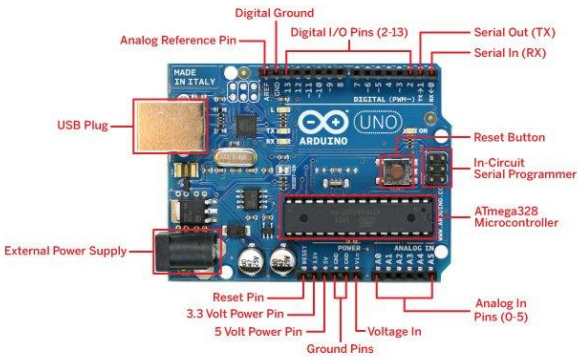


Figura 8. *Modelo UNO de Arduino [Ref][ix]*
https://es.wikipedia.org/wiki/Arduino_Uno

2.3.2 Raspberry Pi

La Raspberry Pi, es una placa de desarrollo más reciente que Arduino, aunque ya van por la cuarta versión de la misma. Su diferencia con Arduino radica en que, a diferencia de esta, que monta un microcontrolador, la Raspberry Pi monta un procesador de propósito general (SoC[15]) que le permite ser,

[15] Del inglés, System on a Chip.

en esencia, un ordenador completo, con un sistema operativo y que puede ejecutar tareas mucho más complejas.

La principal ventaja respecto a Arduino, es que todo el software actualmente desarrollado para PC, en caso de adaptarse para soportar la arquitectura ARM[16], es directamente compatible con Raspberry Pi, lo que amplia mucho sus posibilidades al disponer de un abanico de software muy grande. Algunos ejemplos de ello, son los Sistemas operativos como Ubuntu o software como MySQL o Apache[17] entre muchos otros. En la actualidad, una gran parte del software, especialmente los más conocidos, es ya compatible con ARM y Raspberry Pi.

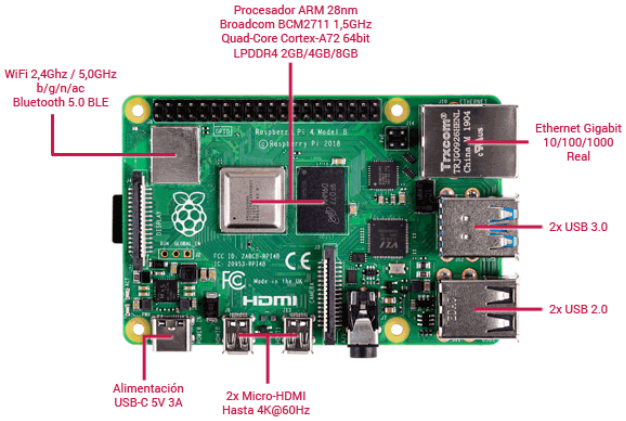


Figura 9. *Raspberry Pi 3B+[Ref[x]]*
https://es.wikipedia.org/wiki/Raspberry_Pi

2.3.3 SoC ESP8266 y NodeMCU

Por un lado, tenemos el SoC que es de 32bits, que es donde se realiza todo el procesamiento y cálculo. Además, este SoC ya monta una radio y capacidad para realizar comunicaciones TCP/IP de forma nativa, con lo que parte aventajada respecto Arduino, que depende de módulos o shields[18] adicionales.

En la placa de desarrollo NodeMCU encontramos la memoria flash que contiene el micro-código y dispone además del resto de entradas y salidas digitales y analógicas.

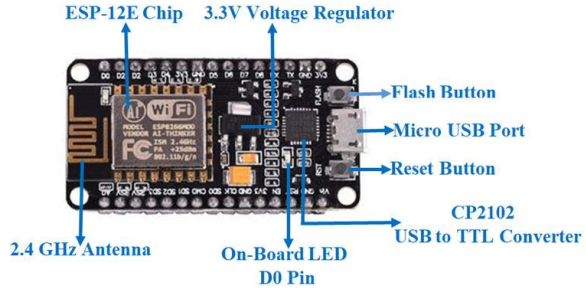


Figura 10. *NodeMCU compuesto por el chip ESP8266 soldado en la parte superior [Ref[xi]]*
https://es.wikipedia.org/wiki/NodeMCU

[16] Advanced RISC Machine. Es una arquitectura de procesadores.
[17] Software de tipo SGBD i de Servidor Web
[18] Del inglés, escudos, es como se conoce a las placas de expansión de Arduino.

2.4 Métodos para monitorizar el consumo eléctrico:

En la actualidad, el coste de la electricidad está calculada en kWh, que son kilo Watts por hora. Para calcular la potencia de un dispositivo o de un circuito, se realiza la operación Voltaje * Intensidad (o corriente) lo que da una Potencia determinada. Esa potencia es en un instante. Para determinar el coste del uso de esa potencia, se mide en el tiempo la cantidad de vatios consumidos en una hora. Así, un dispositivo que está consumiendo 500w durante 1h habrá consumido 500Wh o 0,5kWh. En España, el precio de la electricidad en vatios/h, sin considerar impuestos y partes fijas de acceso a la potencia, suele rondar los 0,13€ el kWh.

Si tomamos como referencia el ejemplo anterior, un electrodoméstico que ha tenido un consumo constante de 500W durante una hora, habrá incrementado nuestra factura en 0.065 céntimos. El problema es que los dispositivos, ni tienen una potencia constante, ni tampoco se utilizan en unidades de 1 hora, por lo que, conocer su consumo es muy complicado y siempre se suele estimar.

Para hacer un cálculo real de la potencia instantánea o la potencia consumida durante un intervalo de tiempo, tanto por un dispositivo en concreto como por un conjunto, se debe hacer con un sensor que mida el consumo eléctrico. Estos sensores pueden medir el consumo eléctrico con dos métodos distintos:

2.4.1 Invasivo:

El método invasivo consiste en un módulo que intercepta la fase del circuito eléctrico a monitorizar, y mide la corriente y voltaje que circula cuando el circuito está cerrado (ver figura 11). Este método se basa en el Efecto Hall[xii]. El chip en cuestión, produce una señal de salida auxiliar que es la información deseada. Esta salida debe ser capturada y procesada con un microcontrolador como Arduino o ESP8266. El gran inconveniente de este método es la capacidad máxima de corriente que soportan estos chips, que es aproximadamente de 30 Amperios. Esto representa, en un entorno de 220V una potencia máxima de 6600W. Para circuitos domésticos suele ser suficiente, pero para entornos industriales, esto es una gran limitación.

Algunos de los chips más conocidos que van integrados en medidores de corriente comerciales, pueden ser el ACS712[xiii]o el HLW8012[xiv].

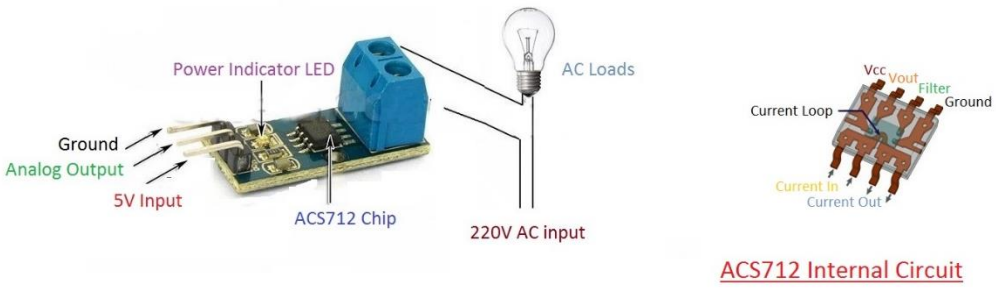


Figura 11. *ACS712 Current Sensor Module de 20Amps [Ref[xv]]*
https://www.theengineeringprojects.com/2019/03/introduction-to-acs712.html

2.4.2 No invasivo:

El método no invasivo significa que la monitorización del consumo eléctrico se hace sin llegar a interceptar, y por tanto sin cortar, la fase del circuito, sino que mediante el uso de un transformador de

corriente (CT[19]), es posible llegar a conocer la intensidad que circula por el circuito monitorizado. Esto es posible gracias a que el CT en su interior es una bobina toroidal que, en el momento en que por dentro tenemos un hilo conductor con corriente, este hilo induce un voltaje proporcional sobre el hilo de la bobina (ver figura 12):

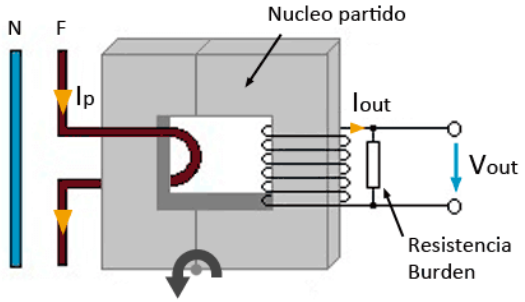


Figura 12. *Esquema transformador de corriente con el hilo principal y el hilo secundario [Ref[xvi]]*
https://www.luisllamas.es/arduino-sensor-corriente-sct-013/

Existen CT de muchas capacidades y relaciones que pueden llegar a medir hasta 2000 amperios, motivo por el cual, este método tiene aplicaciones más industriales que el anterior. El inconveniente de este tipo de sensores, es que, cuanto más alta es su capacidad, menor es su precisión para medir la corriente real. Un detalle importante de este método, es que el CT no puede rodear la fase y el neutro a la vez, ya que sus lecturas en este escenario dan 0.

En nuestro proyecto, usaremos uno de los CT más extendidos, el SCT-013-000 (ver figura 37).

2.5 Productos y soluciones de monitorización de energía:

En el siguiente apartado se describirán algunas de las soluciones más conocidas de medición de potencia de la actualidad, tanto a nivel particular como a nivel de empresa. Todas ellas disponen de un entorno completo similar en los conceptos estudiados en este PoC, es decir, mismos métodos de monitorización, servidores y clientes, uso de WiFi, etc.

2.5.1 OWL

OWL es una empresa inglesa especializada en la medición de consumo eléctrico orientada al mercado de consumo, que vende sus propios monitores y que dispone de su propia plataforma cloud donde almacena todos los datos y donde los usuarios pueden ver el consumo de los elementos monitorizados.

Los tipos de monitores que venden son de tipo no invasivo (es decir que están basados en un CT) y pueden monitorizar desde una línea concreta de una casa, la línea general (la que sale directamente del contador eléctrico), hasta la de generación de energía que viene de una fuente de energía como placas solares, un aerogenerador o una batería.

Debido a que son dispositivos completamente fabricados por ellos y diseñados exclusivamente para su plataforma, no es posible integrar dispositivos de terceros en ella, ni modificar sus sensores para que usen otros protocolos de comunicación distintos al WiFi.

[19] Del inglés Current Transformer, significa transformador de corriente.

Figura 13. *Dispositivos y plataforma de OWL Intuition [Ref[xvii]]*
https://myecohub.com/product/owl-intutition-lc-3-phase-electricity-monitor/

2.5.2 Efergy

Efergy es una compañía también inglesa especializada en el sector de la medición y análisis del consumo eléctrico que fue fundada por allá el 2006.

Como ocurre con OWL, también está orientada al mercado no empresarial y ofrece toda una solución para que el consumidor no deba preocuparse de nada tras adquirir e instalar el producto. En este caso, también se ofrece la medición de consumo exclusivamente de forma no invasiva por lo que no es posible, por ejemplo, hacer una medición de un electrodoméstico simple como la nevera.

También vende sus propios kits de dispositivos que automáticamente se conectan al WiFi y que no permiten variaciones como conectarlos por 4G u otras tecnologías. Tampoco es posible integrar dispositivos de terceros en esta solución.

Figura 14. *Dispositivos y plataforma de Efergy [Ref[xviii]]*
https://myecohub.com/product/owl-intutition-lc-3-phase-electricity-monitor/

2.5.3 Schneider Electric

Schneider Electric es una de las compañías líderes mundiales en fabricación de componentes e instalaciones eléctricas. Sus componentes están instalados en gran parte de los edificios del mundo, desde oficinas, viviendas, estadios, fábricas, etc. Además de la fabricación de componentes eléctricos tienen

otras líneas de negocio, tales como soluciones integrales dedicadas a la gestión eléctrica, por ejemplo, de entornos extremadamente críticos como hospitales, o incluso datacenters[20].

Entre los miles de productos de su porfolio, también tienen un sistema propio de medición de consumo eléctrico orientado exclusivamente a grandes empresas.

Sus productos son mucho más especialistas, robustos y completos que OWL y Efergy y, con toda probabilidad, se integran muy bien tanto con otros productos de Schneider y como aplicaciones de terceros, algo que es común y necesario en entornos empresariales.

Como es obvio, el coste de estas soluciones empresariales solo es asumible para empresas que con ello obtienen un beneficio mayor. Sin ir más lejos, estos dos clientes elegidos al azar en su porfolio de productos valen 800€ el primero (que es de la gama de entrada) y 6.000€ el segundo, aunque pueden subir hasta 15.000 y 18.000€ por cada uno de los monitores eléctricos (es decir, el rol de cliente).

Figura 15. *Dispositivos y plataforma de Schneider [Ref[xix]]*
https://www.se.com/us/en/product-subcategory/52530-basic-multifunction-power-and-energy-meters/?filter=business-4-low-voltage-products-and-systems

2.5.4 EmonCMS de Open Energy Monitor

OpenEnergyMonitor es un proyecto que nació en el año 2009 de la mano de Trystan Lea y Glyn Hudson, unos estudiantes de Física de la universidad de Cardiff en Inglaterra.

La idea principal del proyecto, que se ha mantenido hasta hoy en día, es ofrecer una plataforma de medición de consumo completamente basada en Open Source[21] en la que poder hacer monitorización del consumo eléctrico originalmente usando sensores de tipo no invasivo. La plataforma creada a raíz del proyecto OpenEnergyMonitor se llama EmonCMS y hoy en día ya van por la versión 10.2.27.

Gracias a que es completamente Open Source y que ha sido bien recibida por la comunidad, con los años, la plataforma se ha ido expandiendo mediante las subsiguientes versiones, así como la colaboración de usuarios, que han ido aportando propios desarrollos en forma de módulos. Hoy en día, esta solución es integrable con otro tipo de sensores, como enchufes inteligentes, sensores de humedad, temperatura, medición de flujo de agua, placas solares, baterías y un largo etc.

[20] Del inglés, significa Centro de datos. Son grandes emplazamientos de servidores donde se encuentran los servicios digitales que conocemos como cloud.
[21] del inglés, código abierto. Significa que el código original es completamente público para ser consultado y en ocasiones cambiado sin restricción.

Además, inicialmente fue un producto pensado para ser instalado por cada usuario exclusivamente en una Raspberry Pi, pero ahora se puede instalar tanto con el método inicial como en un equipo x86, en máquinas virtuales, en docker[22], etc.

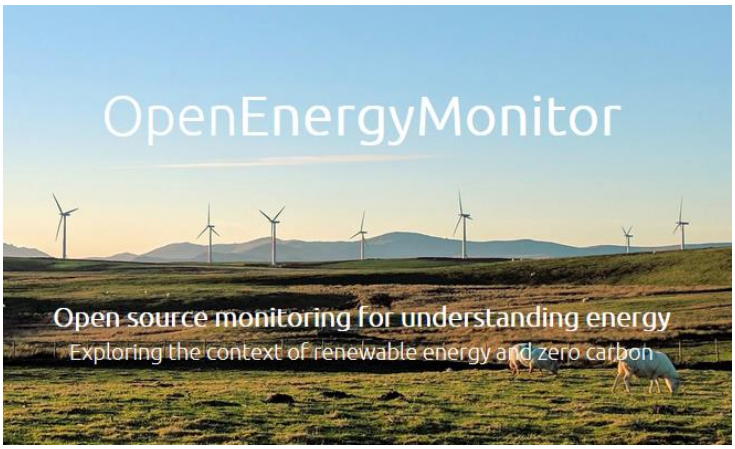


Figura 16. *Sistema de OpenEnergyMonitor [Ref[xx]]*
https://openenergymonitor.org/

2.5.5 Tuya

Tuya no es una plataforma de medición de consumo, sino una plataforma de IoT muy popular mundialmente. Cualquier empresa mediante la compra de unas licencias puede integrar sus dispositivos en esta plataforma.

Está en esta lista porque hay varios fabricantes que, entre otras cosas, venden enchufes inteligentes con capacidad de medición de consumo (gracias a que incorporan un módulo sensor de corriente en el interior). Esta medición de consumo puede ser leída mediante la aplicación móvil de la plataforma Tuya (ver figura 17).

Figura 17. *Monitorización consumo de un enchufe inteligente en la plataforma IoT Tuya*

Lógicamente, al no tratarse de un entorno diseñado específicamente para la medición de consumo eléctrico, las funcionalidades que ofrece son muy básicas. Solo se permite saber el consumo en tiempo real del dispositivo conectado al enchufe inteligente, el consumo diario de cada uno de los días del mes en curso y el total acumulado de este, así como el total acumulado de cada uno de los últimos 12 meses.

En esta solución se pueden obtener más reportes, ni utilizar sensores no invasivos, ni comparar mediciones de consumo entre sí, pero en comparación al resto de soluciones cerradas como OWL o Efergy, ofrece algo que estas no ofrecen, que es la medición de aparatos simples y a una fracción del coste de estas dos.

[22] es un sistema de abstracción que funciona a nivel de aplicación. Es un concepto similar a la virtualización.

3 Requisitos de la plataforma

Con tal de alcanzar los objetivos del proyecto, se deben cumplir una serie de requisitos especialmente en lo que se refiere a universalidad y facilidad de acceso. Dentro de los requisitos, encontramos los funcionales y los no funcionales:

3.1 Requisitos funcionales:

- El servidor debe tener la **capacidad de ser multitenant**[23]. Esto significa que el servidor podrá dar funcionalidad completa a todos los usuarios sin mezclar los datos de estos.
- **Accesible desde cualquier red**: El servidor debe poder ser accedido desde cualquier tipo de red, incluyendo redes propietarias como 4G o redes abiertas, redes WiFi comunitarias o LoRaWan.
- **Capacidad de visualización**: El servidor debe proporcionar las herramientas necesarias para visualizar, comparar y/o contrastar los datos de las distintas fuentes que cada usuario tiene almacenadas, sin ser necesarias herramientas externas.
- **Propiedad de los datos**: Los usuarios de la plataforma deben poder extraer los datos almacenados en el servidor (para manipularlos externamente, por ejemplo), así como borrarlos o modificarlos.
- **Seguridad del servidor**: El servidor debe poderse alcanzar usando métodos cifrados y seguros.
- **Varios métodos de entrada de datos**: El servidor debe aceptar varios métodos para introducir datos, para cubrir las distintas necesidades de los usuarios.

3.2 Requisitos no funcionales:

- **Software Open Source**: Todo el software (SW) usado debe ser de código abierto. La razón de ello, es que debe ser posible que cada miembro pueda implementar sus propios clientes, para lo cual debe saber cómo debe programarlos para que sean compatibles con el servidor.

  Aclaración: Respecto al hardware (HW), es deseable que sea también Open Source. No obstante, debido a las limitaciones de coste, es posible que sea necesario adquirir Hardware no Open Source. En particular, este proyecto se utilizará HW que no es Open Source con SW que si lo es.

- **Deslocalización del servidor**: El servidor puede estar ejecutándose en cualquier parte del mundo.
- **Administración del servidor**: El servidor debe poder contar con un administrador con permisos mayores para administrar y mantener el servidor.
- **Accesibilidad de la plataforma**:
  - El servicio debe poder ser usado sin necesidad de cada usuario tenga que montar un servidor propio.
  - El coste de los elementos necesarios para medir el consumo y volcarlos al servidor debe ser bajo, así como fáciles de encontrar.
  - No debe ser necesario tener conocimientos muy avanzados de programación y electrónica.

[23] del inglés significa multitenencia. Es un tipo de arquitectura de software que permite tener varias instancias dentro de un mismo sistema, sin que se mezclen entre sí.

4. Requisitos de la plataforma

Tras haber expuesto la necesidad a cubrir, los objetivos del proyecto, y sus restricciones más importantes, vamos a hacer un análisis a alto nivel de lo que se necesitaría para cubrir el escenario expuesto:

4.1 Servidor central

La plataforma debe tener un punto central donde los clientes introducirán los datos de telemetría y serán almacenados para ser visualizados a posteriori. Debe ser administrado en la medida de lo posible, por una comunidad bien organizada que pueda mantener los servicios funcionando y realizar administración del servidor.

El software del servidor central debe ser Open Source para que cada comunidad instale o configure el servidor como más conveniente le sea. Por ejemplo, eligiendo una configuración donde el servidor es una Raspberry Pi porque cubre una pequeña comunidad.

4.2 Conectividad

Para ser alcanzado mundialmente, el servidor aceptará el protocolo TCP/IP (ver datacenter en la figura 18). Esto automáticamente posibilita la conectividad mediante WiFi y 4/5G. Adicionalmente, también tiene que poder ser integrado con sistemas que usan LoRaWan. Un detalle importante a tener en cuenta en las comunicaciones, es el uso de mecanismos no sensibles al uso de NAT[24] en las redes.

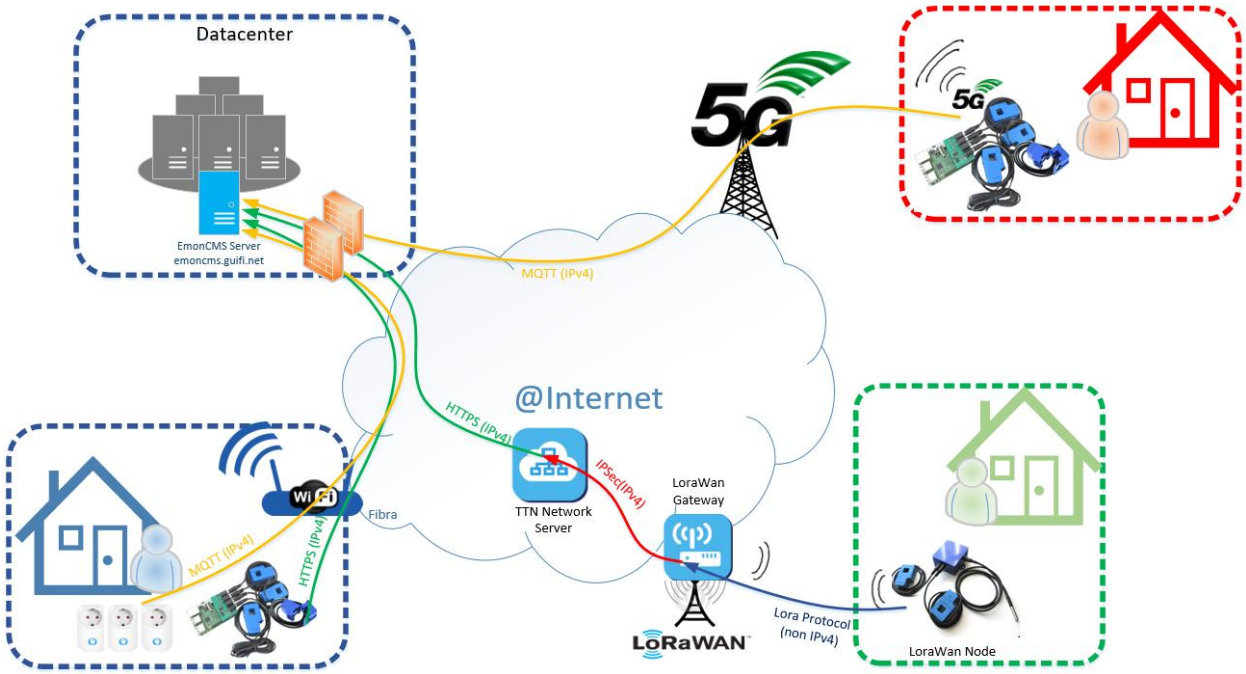


Figura 18. *Arquitectura de la plataforma de medición de consumo cooperativa y multiusuario.*

[24] Del inglés, network address translation. Es un mecanismo de redes usado para traducir y reemplazar las cabeceras de los paquetes por otras cabeceras, para no tener problemas de encaminamiento de paquetes.

4.3 Bases de datos

Las bases de datos deben estar diseñadas de forma que tengan la capacidad de manejar los potenciales miles de registros que cada usuario puede enviar cada año a la plataforma. Así mismo, debe ser posible modificar registros históricos y tolerar la concurrencia de escrituras en base de datos.

4.4 API

Es necesario contar con una API[25] para integrar los dispositivos de desarrollo propio. Además, dado que, en potenciales escenarios, el servidor no puede conectarse a los clientes para extraer información, este debe ofrecer una vía para que los clientes envíen el contenido al servidor. La API debe ser compatible con el multiusuario sin mezclar los datos entre ellos.

4.5 Visualización de la información

La plataforma debe disponer de una interfaz gráfica para visualizar la información de cada usuario. Dicha información debe poder ser visualizada por cualquier cliente de cualquier plataforma, por lo que debe usar un método accesible a todo el mundo, sin instalación de complicadas herramientas.

4.6 Clientes

Se necesitan dos tipos de clientes. El primero que se instala directamente en el cuadro eléctrico y que monitoriza la fase de un circuito eléctrico. En este caso, no es necesario que tenga un diseño muy específico, pues va oculto a la vista y no está pensado para moverse.

El segundo tipo de cliente, debe tener la característica de fácil uso y mantenimiento. El objetivo de este cliente es que un usuario pueda monitorizar cualquier dispositivo que se enchufa en un enchufe. Su factor de forma, por tanto, debe tener un enchufe macho por un lado (para enchufarlo en el enchufe valga la redundancia), y un enchufe hembra para conectar el dispositivo monitorizado.

4.7 Conclusiones

En base al análisis realizado sobre qué tipo de producto necesitaríamos para cumplir con los objetivos y restricciones del proyecto, el software más adecuado en este caso sería EmonCMS. Las razones son principalmente que el resto de plataformas, o son muy caras o son demasiado rígidas para adaptarse a las muy diferentes necesidades de los usuarios.

Por otro lado, EmonCMS, que es Open Source, no está afectado por estas limitaciones, ya que es modificable e integrable con otras plataformas, así como el precio, que es significativamente más reducido y eso juega mucho a su favor.

	Open Source	Compatible con terceros	Tipos de mediciones	Protocolos de comunicaciones	Medición enchufes	Medición fase	Modificable	API	Precio inicial
OWL	No	No	Consumo y SolarPV	WiFi	No	Si	No	No	250 €
Efergy	No	No	Consumo y SolarPV	WiFi	No	Si	No	No	300 €
Schneider	No	No	Consumo y SolarPV	WiFi, Ethernet	No	Si	Si	Sí	857 €
EmonCMS	Si	Si	Consumo, SolarPV, Temperatura, Humedad, Gas, Agua, etc	WiFi, Ethernet, LoRaWan, 4/5G, Bluetooth, etc	Si	Si	Sí	Sí	10 €
Tuya	No	Si	Consumo	WiFi	Si	Si	No	No	10 €

Figura 19. *Comparativa de productos para la medición de consumo eléctrico.*

[25] del inglés, Application Programming Interface. Es una interfaz de entrada a una aplicación para integrar aplicaciones o sistemas externos.

5. Arquitectura y diseño de la plataforma de medición de consumo.

5.1 Arquitectura de comunicaciones de la plataforma

5.1.1 Protocolos de transmisión

Para que el servidor sea accesible por cualquier cliente en todo el mundo, es necesario que sea alcanzable por Internet que, por hoy es el único tipo de red con cobertura mundial en todos los entornos. Esto es así porque la red TCP/IP se ha impuesto como el único protocolo de las redes de comunicaciones.

Para que todos los clientes sepan dónde deben conectarse, es necesario que el servidor tenga un nombre único en internet, llamado FQDN. Este nombre enmascara una IP Pública que es accesible desde todo el mundo.

Lo más habitual, es tener el servidor en un Datacenter o un CPD. En este PoC se plantea que el servidor sea una máquina virtual dentro de un hypervisor que estará detrás de un firewall. Los servicios que deben ser accesibles desde el exterior, son aquellos que son estrictamente necesarios para que los clientes se conecten. Esto es, el puerto de la API HTTP y de MQTT, es decir, el 443 y el 1883.

Los clientes son los que se conectarán al servidor por uno de los dos puertos, en función del protocolo de red que utilicen. Dado que los clientes estarán habitualmente en domicilios, empresas, en redes 4/5G o incluso LoRaWan, es muy posible que tengan IP's privadas no accesibles. Esto tampoco es un inconveniente porque el servidor no necesita conocer desde donde se conectan. Simplemente está a la espera que le lleguen los mensajes, identificados de alguna forma en el cuerpo del mensaje, para saber a quién corresponden (ver parte de la API sección 5.2.4).

La comunicación desde los clientes es posible ya que, en los routers[26] de cada domicilio, se realiza un NAT en origen o SNAT, mientras que la red destino, es inequívoca y es el servidor

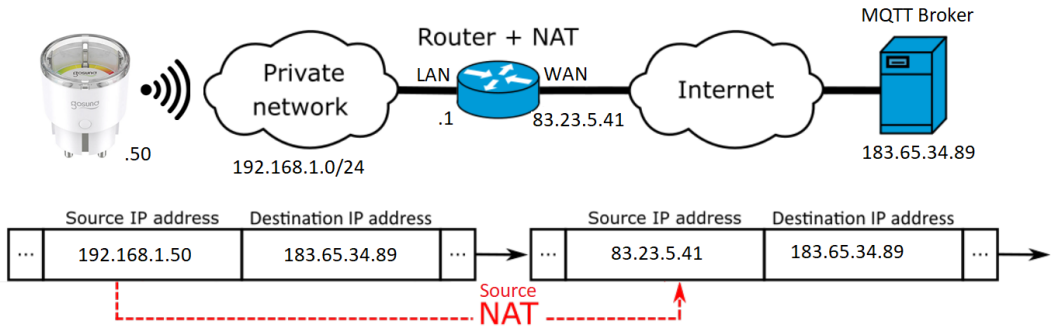


Figura 20. *Proceso de un Source NAT [Ref[xxi]]*
https://en.wikipedia.org/wiki/Network_address_translation

[26] Del inglés, encaminador. Es un dispositivo fundamental en las comunicaciones basadas en el modelo OSI.

Como se ve en la figura del SNAT (ver figura 20), toda la arquitectura está diseñada en modelo Cliente-Servidor, en el cual, es el cliente el que inicia la conexión y envía los datos al servidor y el servidor no sabe dónde están los clientes ni cuando le enviarán los datos, es decir que siempre está a la espera.

Esta no es la única forma que existe para que el servidor obtenga los datos, pues también sería posible que el servidor, por si mismo, fuera él el que activamente fuera a recoger los datos a los clientes. Esta casuística, aunque no se cubre en este proyecto sí que se puede dar en el mundo real cuando por ejemplo se quiere integrar un inversor o una batería de placas solares en la plataforma. Por ello es importante hacer una breve mención, ya que modifica en gran parte la arquitectura de comunicaciones.

Cuando el servidor debe conectarse al cliente, significa que el cliente es ahora el que espera y el servidor el que debe conocer dónde conectarse, por tanto, debe conocer la IP del cliente.

El problema se debe a que, en la mayoría de empresas y domicilios y en la conexiones 4/5G, los routers que dan acceso a internet, tienen toda la red nateada a una sola IP pública, es decir, que hay una relación de 1:n respecto la IP pública. El servidor solo puede conocer la IP pública, por lo que se convierte en requisito que el usuario reconfigure el router de su conexión a internet, para que haga DNAT a la IP que tiene que monitorizar y que dicha IP interna nunca varíe (ver figura 21). Cómo se puede ver, a diferencia de la conexión de cliente a servidor, en esta, es necesario considerar más factores y por tanto es necesario realizar más configuraciones.

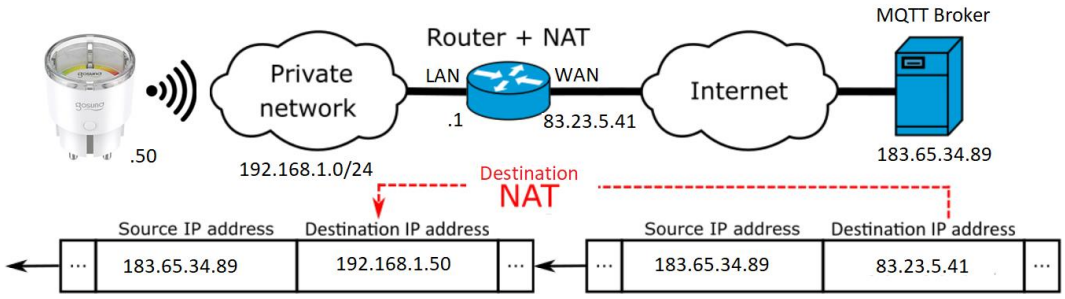


Figura 21. *Proceso de un Destination NAT*

En ocasiones, no solamente hay un nivel de NAT, sino que hay dos o incluso tres. Sin ir más lejos, en Guifi.net esta práctica se realiza a todos los usuarios. Esto obliga a que el usuario debe tener en cuenta varios niveles de NAT y ahora tampoco es administrador de los equipos que lo realizan, con lo que se complica todavía más.

Está práctica que hace Guifi.net, se denomina CGNAT[27] y se emplea para reducir la cantidad de IPv4 a distribuir a los clientes, agrupando unos cuantos clientes bajo una única IP Pública. Esta misma práctica es también realizada por otros operadores y por las redes 4/5G.

LoRaWan

En el caso de LoRaWan, también encontramos dificultades para realizar un flujo de datos de servidor a cliente, aunque son dificultades inherentes a la naturaleza de LoRaWan. De forma resumida, sería necesario realizar los siguientes pasos:

[27] Del inglés, Carrier Grade NAT. Es un tipo de NAT masivo realizado por el ISP.

1. Comunicar el servidor con los servidores de TTN. TTN tiene una API para enviar un mensaje a un cliente.
2. Cuando el servidor se comunica con dicha API, el mensaje es enviado al Gateway LoRaWan, que debe convertir el mensaje de protocolo TCP/IP a protocolo LoRa para mandarlo al nodo LoRa.
3. El mensaje se envía cuando el nodo LoRa abre una ventana de escucha. Debido a la importancia de la eficiencia, para ahorrar energía, los nodos de LoRa normalmente están siempre dormidos y solo despiertan para mandar datos. Es en ese momento que están unos instantes despiertos para recibir datos.
4. Tal como está diseñada la red LoRaWan, la cantidad de mensajes que se permiten desde los Gateways a los nodos, es mucho más reducida.

En conjunto, todas estas dificultades influyen en que la mejor forma de trabajar es usando el modelo cliente-servidor clásico, en el que el servidor nunca es la parte activa en la comunicación, sino el que siempre espera a ser contactado. Por este motivo, todas o casi todas las plataformas IoT, hoy en día disponen de un cloud donde los dispositivos IoT van registrados y por el cual, nosotros podemos interactuar con ellos a pesar de estar fuera de nuestro domicilio.

5.2 Arquitectura del software del servidor EmonCMS.

El sistema operativo del servidor EmonCMS es Ubuntu Server 20.04.2 LTS 64bits, sin ningún requerimiento particular en cuanto al sistema operativo. La elección de usar Ubuntu, en lugar de otras distribuciones[28], es debido a que los manuales oficiales[xxii] utilizan precisamente esta. No obstante, no hay ninguna contraindicación para usar otra.

5.2.1 Frontend[29] y Middleware[30]

EmonCMS, como muchas otras aplicaciones, está formado por un frontend, un middleware y un backend, aunque, al ser una aplicación web, parte del frontend y middleware están ciertamente algo entremezclados.

Disponemos de una interfaz web por donde el usuario interactúa con la plataforma para visualizarla y operarla. En esta parte encontramos los distintos módulos de EmonCMS. Los principales son el módulo de Input, que es por donde se reciben los datos de los sensores, el módulo Feed, que se encarga de guardarlos de forma permanente y el módulo Graph, que se encarga de leer datos de Feed y representar las gráficas, etc.

Dado que EmonCMS es de código abierto, cualquier persona puede añadir sus propios módulos o cambiar los existentes. Con el paso de los años, algunos módulos de terceros se han hecho tan populares, que han pasado a formar parte de la instalación básica de EmonCMS, como Admin, Visualitzation, Sync y Backup y Profile.

[28] distribución es como se llama a los distintos sistemas operativos que están basados en el kernel de Linux.
[29] del inglés, hace referencia a la parte de software más externa y por la que interactúa el usuario.
[30] del inglés, hace referencia a la capa de software que suele estar compuesta por el aplicativo en sí y que se sitúa entre el frontend y el backend.

Todos estos módulos están escritos en PHP[31] y son servidos por web, siendo Apache, el software de servidor web utilizado. Toda la plataforma se puede visualizar y operar por web. Además, por Apache también está accesible la API HTTP, para integrar dispositivos de terceros en esta plataforma.

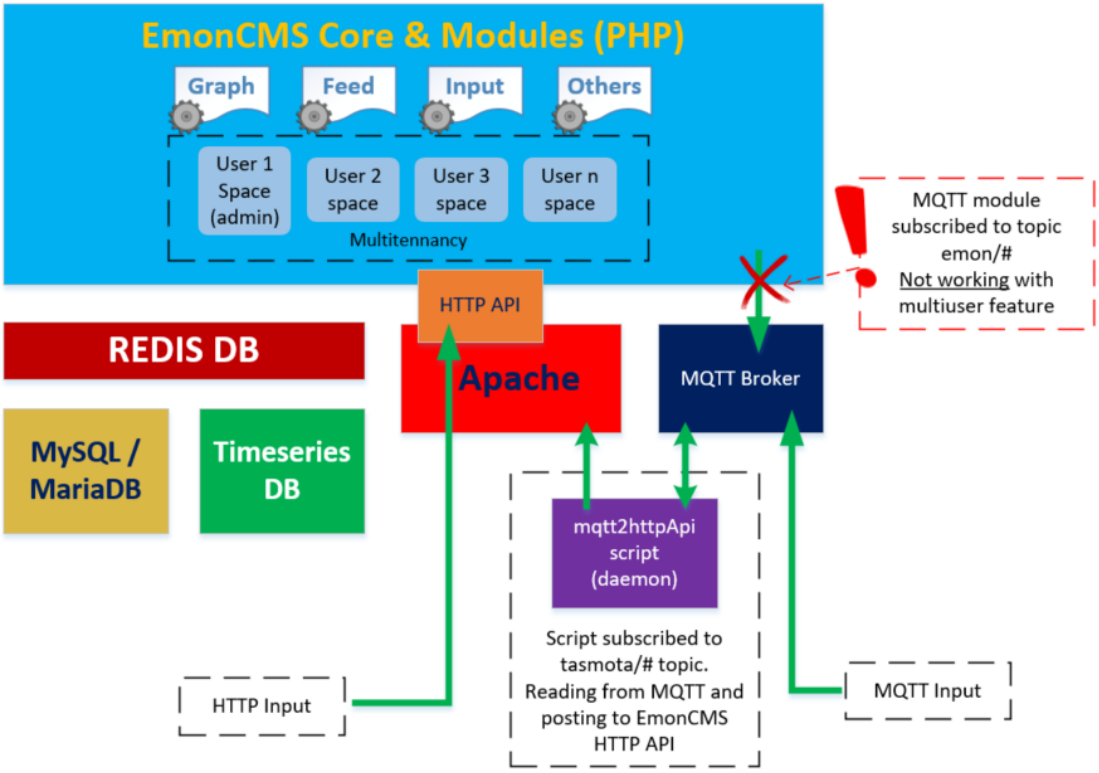


Figura 22. *Arquitectura del servidor EmonCMS.*

La API HTTP se utiliza para la integración de dispositivos externos. Estos envían sus datos de telemetría al servidor para que queden almacenados. Estos datos son recibidos por la API de EmonCMS, concretamente la API del módulo Input y, tras ser procesado con las correspondientes reglas, son almacenadas en base de datos con el módulo Feed. No necesariamente se configura un Feed para cada Input que existe, pues puede ocurrir que un dispositivo envíe más datos de telemetría de los que se desee almacenar.

MQTT también es otra forma oficialmente soportada para introducir datos desde los clientes al sistema, pero en las últimas versiones de EmonCMS, este método no es compatible con la funcionalidad de multiusuario de EmonCMS que, por defecto, viene desactivada. La razón de ello es que los datos de los dispositivos integrados por MQTT, en lugar de ir al usuario correspondiente, se agregarán al espacio del usuario administrador, independientemente del dispositivo[xxiii].

Para solucionar esta eventualidad, se ha creado un script a medida para poder utilizar MQTT como entrada de datos y que traduce los mensajes enviados por ese canal a HTTP, y así introducirlos al módulo de Input de EmonCMS mediante la API HTTP, que sí es compatible con la funcionalidad de multiusuario.

[31] PHP, es un lenguaje de programación.

5.2.2 Procesamiento interno

Antes de empezar a loguear [32] un Input a Feed [33], es posible manipular el dato, mediante operaciones y acciones. Por ejemplo, es posible multiplicar dos Inputs para obtener otro resultado (ver figura 23). Por Ejemplo: multiplicar la corriente (amperaje) por el voltaje, para obtener la potencia.

Otros, por ejemplo, son para eliminar valores negativos de potencia si los sensores CT se han puesto al revés (y por tanto la corriente inducida va en sentido contrario que da una potencia negativa). Como el consumo es siempre un valor absoluto, puede ser interesante configurar un procesado de los valores de entrada para multiplicar por -1, en caso de que el valor de potencia de entrada sea negativo.

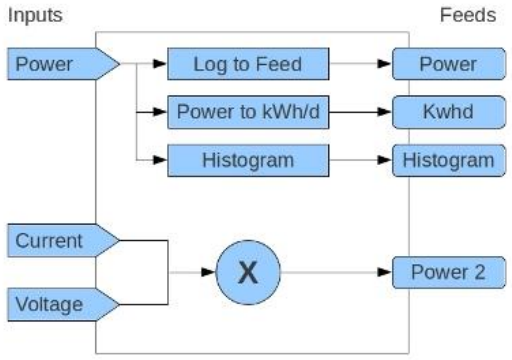


Figura 23. *Procesamiento entre Input y Feed [Ref[xxiv]]*
https://learn.openenergymonitor.org/electricity-monitoring/emoncms-internals/input-processing

Muchas otras operaciones son posibles, aunque al final, todas ellas deben terminar con guardar o realizar alguna acción sobre el dato , tal como enviar un mensaje MQTT o un email.

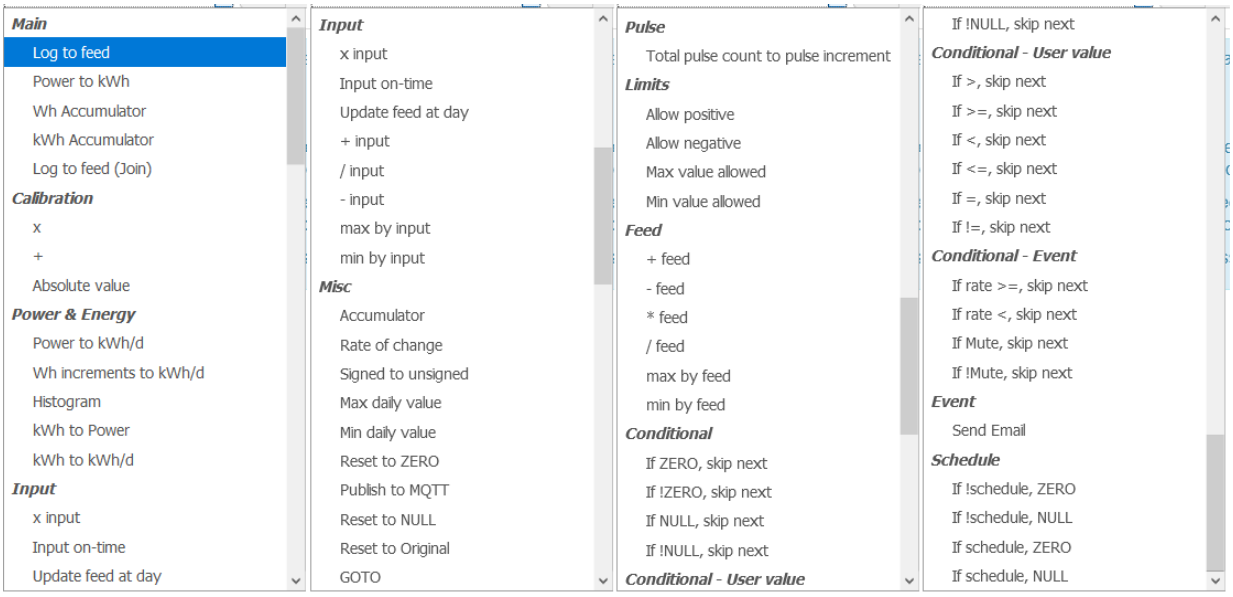


Figura 24. *Todos los tipos de operaciones permitidas para realizar durante el procesamiento interno.*

[32] del inglés, significa guardar, en este caso a base de datos.
[33] Feed, en este contexto vendría a ser una tabla de base de datos donde se almacenan los datos de algo atómico.

5.2.3 Backend[34]

La parte del backend está principalmente gestionada por el módulo Feed. Mediante este módulo se define qué métricas de qué sensores se guardarán en base de datos y también se utiliza recuperar datos de las bases de datos para poder construir gráficas o reportes solicitados por el usuario.

Podemos encontrar 3 tipos de bases de datos en función del tipo de dato que vamos a almacenar:

MariaDB/MySQL

En este SGBD se almacenan los datos que dan consistencia al perfil de todo usuario de EmonCMS, es decir, qué usuarios hay creados en la plataforma, la lista de nodos de cada usuario, cuáles se están guardando, los gráficos y reportes definidos por cada usuario, etc.

En el siguiente gráfico UML[35] se puede ver la estructura de datos principal de la plataforma. Existen otras tablas adicionales necesarias para funciones más secundarias, pero no están relacionadas entre ellas más allá del usuario que es el nexo común entre todas ellas.

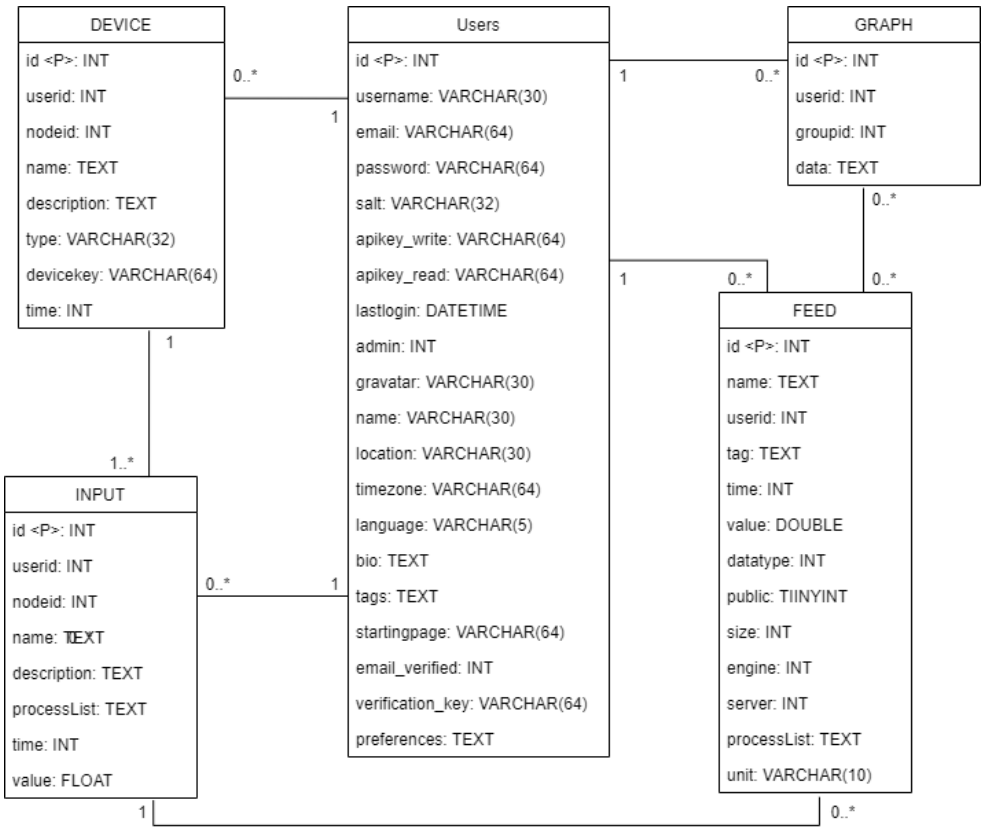


Figura 25. *Estructura interna de los datos (una parte)*

[34] Del inglés, significa la parte más atrás de las aplicaciones, donde se encuentran los datos almacenados, como las bases de datos.

[35] Del ingles, Unified Modeling Language. Es un lenguaje de modelado para la representación de sistemas de software.

Debido a que la telemetría produce una gran cantidad de datos que deben guardarse todos, en MySQL no se guarda ningúno de este tipo. Concretamente, en MySQL se almacenan los datos del usuario, los de inputs de cada usuario, así como los dispositivos (tabla DEVICE).

También se guardan en la tabla FEED, el nombre del Feed, su ID, el propietario de dicho Feed, y otra información importante, pero en ningún caso se guardan los datos de telemetría que van llegando del sensor. Estos se guardan en una base de datos de tipo fixed Timeseries.

Por ejemplo, si tenemos un sensor de temperatura y humedad que además tiene una batería integrada, los valores de telemetría que podría enviar sería: Valor de humedad, valor de temperatura, hora, nivel de batería. Solo queremos guardar la humedad y la temperatura. Esto en entradas en la base de datos MySQL se traduciría:

1. En la tabla Device se crearía un solo registro, que sería el sensor.
2. En la tabla Input se crearían 4 registros, uno de temperatura y otro de humedad, otro de hora y otro de nivel de batería, vinculados al registro de la tabla Device, pero recordemos que esto no significa que se almacenan los valores de estos tipos de datos, eso es parte de Feed.
3. En la tabla Feed, se crearían solamente dos registros, el de temperatura y el de humedad, vinculados a los respectivos registros de input. Y es en estos registros donde definimos que dónde guardaremos los valores en sí de Temperatura y de Humedad.

Fixed Timeseries DB:

Tal y como se venía explicando en el apartado anterior, en la tabla FEED de MySQL se crea una entrada para especificar que un determinado Input debe registrarse. En este punto, el módulo Feed de EmonCMS, creará y guardará los valores correspondientes a ese Input que le llegan del sensor, en una base de datos de tipo Timeseries.

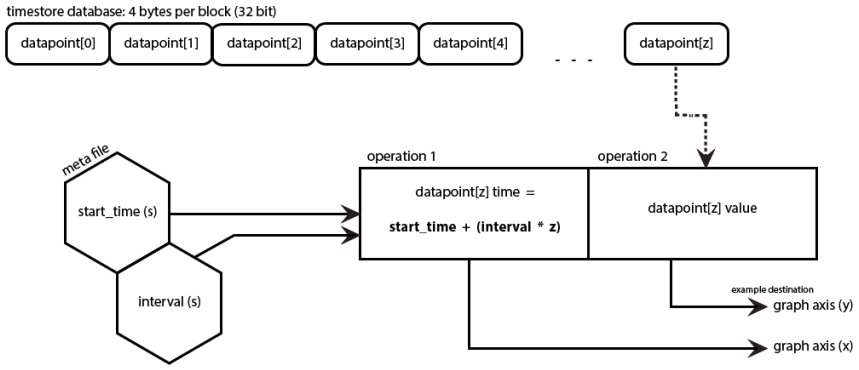


Figura 26. *Arquitectura de las bbdd de tipo Timeseries [Ref[xxv]]*
https://learn.openenergymonitor.org/electricity-monitoring/timeseries/Fixed-interval

Las bases de datos Timeseries[xxvi] son un tipo de bases de datos clave-valor especialmente diseñadas para este tipo de datos, ya que aportan mucha eficiencia en espacio y velocidad de lectura y escritura. Hay que tener en cuenta que la funcionalidad principal de EmonCMS es almacenar, consultar y visualizar los datos recibidos constantemente de los sensores volcados durante largos periodos. Por ejemplo, un sensor enviando datos cada 20 segundos, da lugar a 1.576.800 registros en un solo año.

Algunas de sus grandes ventajas son por ejemplo, su capacidad de compresión nativa, realización de medias automáticas, o que, para cada registro, no se guarda la fecha completa, sino el tiempo relativo desde la primera captura, guardada en "Epoch/Unix time[36]" [xxvii]. Para más información, el autor de EmonCMS expone, en su blog[xxviii], los resultados de varias pruebas de este tipo de bases de datos y explica los beneficios de usar este tipo de base de datos en contraste con MySQL.

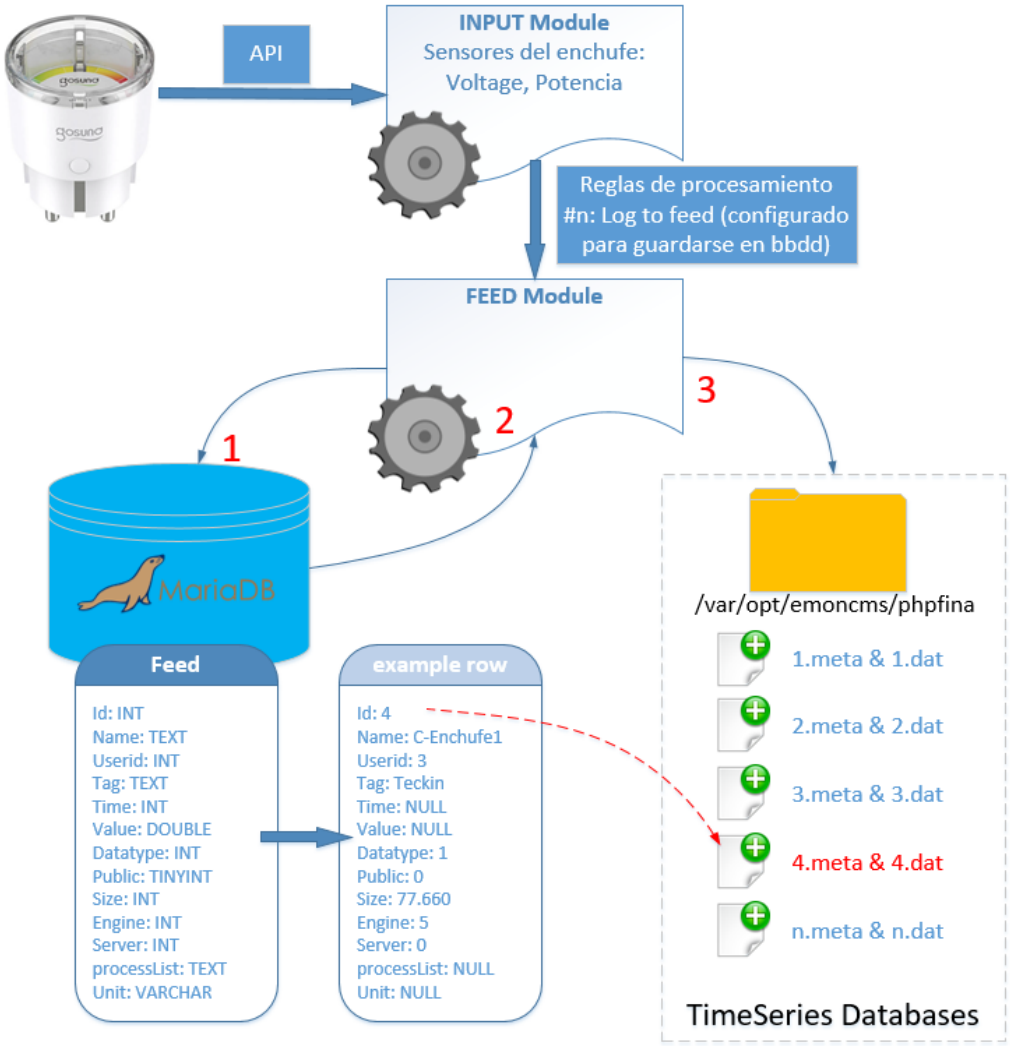


Figura 27. *Flujo de entrada y almacenamiento de datos en EmonCMS.*

Por tanto, cuando un usuario determina que quiere empezar a guardar un Input, se creará una tupla en la base de datos MySQL, que determinará qué ID tendrá ese Feed y posteriormente se crearán dos ficheros "*id.meta*" e "*id.dat*", que serán la base de datos de tipo Timeseries para ese feed. Estos dos ficheros serán exclusivos para volcar los datos de ese input en concreto. El fichero "*id.meta*", contendrá

[36] Epoch/Unix time es cómo funciona el conteo de tiempo en SO Linux. Se cuentan los segundos desde 1-Ene-1970.

dos valores, el tiempo de inicio del primer registro y el intervalo de cada registro. El fichero "*id.dat*", contendrá un contador como clave (ver z en la figura 26), y el valor en el segundo registro.

Cuando EmonCMS necesite leer los datos de un determinado Feed, consultará a la BBDD el ID del Feed y leerá el fichero "*id.meta*" e "*id.dat*" correspondiente.

RedisDB:

La base de datos Redis es opcional dentro del paquete de instalación. Su incorporación se debe al incremento del procesado de los datos en EmonCMS que obligan a realizar muchas lecturas y/o escrituras temporales y a que, originalmente, EmonCMS y sus bases de datos, estaban diseñadas para almacenarse en una MicroSD. Esto conllevaba una rápida degradación de estas memorias, al no estar diseñadas para estos propósitos.

Las bases de datos Redis, son un tipo de BD que reside enteramente en memoria, y que almacenan datos de tipo clave-valor, es decir, que sirven de apoyo a las bases de datos de tipo Timeseries y MySQL.

Si consideramos que un servidor EmonCMS puede tener centenares o miles de Inputs y que en cada uno de ellos puede haber varios procesamientos y varios Feed (como ocurre en la figura 28), es factible llegar a saturar la lectura y escritura de datos sobre MicroSD.

Id	Tag	Name	Process list
90	...	SobreConsum-400W	Source Feed + Allow positive
89	...	C-TotalElspins	Source Feed If schedule, NULL + source feed
92	...	A-TotalElspins	Source Feed + source feed
94	...	SobreConsum-50W	Source Feed + Allow positive
97	...	C-PisAbaix	Source Feed + source feed + source feed
98	...	A-PisAbaix	Source Feed + source feed + source feed
101	...	P-SolarPV	Source Feed + source feed x x
102	...	C-TotalPV	Source Feed x + source feed
107	...	C-TotalPVRealBat	Source Feed If ZERO, skip next If schedule, ZERO If !NULL, skip next Reset to ZERO
104	...	C-TotalPVReal	Source Feed Allow positive

Figura 28. *Ejemplos de Feed y sus procesamientos.*

Por ello, los distintos módulos de EmonCMS escritos en PHP que trabajan con las bases de datos, miran primero si está accesible la base de datos Redis para guardar o para leer los datos directamente de ahí. En la figura 29 vemos como está implementado en las funciones de PHP.

```
public function get_user_feeds($userid)
{
    $userid = (int) $userid;

    if ($this->redis) {
        $feeds = $this->redis_get_user_feeds($userid);
    } else {
        $feeds = $this->mysql_get_user_feeds($userid);
    }

    return $feeds;
}
```

Figura 29. *Función PHP para obtener el listado de un feeds de un usuario de la B.D Redis antes que MySQL*

5.2.4 API de EmonCMS

API HTTP

Las API de EmonCMS se utilizan para integrar los clientes con el servidor. Sin estas API, no sería posible enviar los datos de los distintos sensores, por lo que muchas de las posibilidades, tales como comparar distintas mediciones o integrar distintos productos y fabricantes no serían posible. Otro detalle importante de la API es que, gracias a esta, el servidor no ha de recoger los datos por sí mismo. Tener la API es lo que define este entorno como un modelo cliente-servidor[37].

Existen dos API. La primera es la del módulo Input, es decir la Input API. Se utiliza para introducir datos en la plataforma, que luego serán procesados tal y como indican las reglas de procesamiento y almacenados en la base de datos mediante el módulo Feed.

La API acepta datos en varios formatos. Pueden ser mediante HTTP Get o Post, y con datos en CSV o en JSON. Veamos un par de ejemplos usando GET y POST respectivamente:

GET:

```
curl
"http://localhost/emoncms/input/post?node=Enchufe1&json={power:34,volt:228}&api
key=aabbcc011b"
```

POST:

```
curl --data "node=Enchufe1&apikey=aabbcc011b&data={power:34,volt:228}"
"http://localhost/emoncms/input/post"
```

La segunda API es para el módulo Feed. El objetivo de esta API no es introducir datos a la base de datos, aunque sería posible. A diferencia de la Input API, cuando se utiliza la Feed API, las reglas de procesamiento no se ejecutan precisamente porque “están” en una instancia anterior. Esta API se utiliza para obtener datos históricos, como, por ejemplo, para dibujar gráficas o construir reportes. También se puede utilizar para exportar información o incluso importarla masivamente, con el objetivo por ejemplo de realizar una copia de seguridad o restauración.

[37] Es un modelo en el que, en una interacción entre 2 nodos, queda separado y bien identificado quien sirve recursos (servidor) y quien los demanda (clientes).

GET:

```
curl "http://localhost/feed/data.json?id=0&start=UNIXTIME&end=UNIXTIME&interval=5&apikey=aabbcc011b"
```

Se puede encontrar información de ambas API en: https://emoncms.org/site/api [xxix]

Una de las partes más importantes de las API, es la especificación del APIKEY, que corresponde a un identificador único de cada usuario de la plataforma. Al especificar el APIKEY dentro de la petición GET o POST, le indicamos al módulo Input o Feed, a qué usuario dentro de la plataforma, va dirigida esa petición, algo fundamental en la capacidad "multitenant" de la plataforma.

MQTT.

Aunque no es propiamente una API, se podría considerar como tal ya que sirve para introducir datos al servidor desde elementos externos. En este caso es un reemplazo únicamente de la Input API, ya que no puede ser usada para consultar datos y todos los datos introducidos mediante MQTT siempre pasarán las reglas de procesamiento configuradas.

Importante: Debido a una carencia de EmonCMS[xxx], actualmente no es posible utilizar este método en conjunción con la característica de multitenencia. Esto es debido porque a diferencia de la API HTTPS, donde sí es posible especificar el "APIKEY" del usuario, en MQTT esto no se ha llegado a implementar. Como consecuencia, todo mensaje publicado por MQTT al topic emon#[xxxi], va directamente al perfil del primer usuario, que es el administrador, lo que invalida completamente la forma oficial de MQTT como entrada de datos. En su lugar se ha creado una script que funciona en modo servicio y que permite la comunicación multiusuario por MQTT

MQTT Workaround

El script en cuestión va suscrito al topic[38] tasmota# del bróker MQTT (que está en el mismo servidor). El script va leyendo todos los mensajes que llegan al topic y extrae el primer subtopic, que corresponde al APIKEY del usuario, y el segundo subtopic que corresponde al dispositivo. De esta forma, cualquier usuario que desee mandar datos de telemetría al servidor usando MQTT debe usar el topic con la siguiente composición:

MQTT Topic → *"tasmota/$apikey/$dispositivo"*

Dentro del topic va el mensaje, en la jerga de MQTT llamado payload, que son los datos de los distintos sensores y sus valores. En concreto, el script toma el valor de aquellos datos que interesan y los pone en variables. Finalmente compone una petición HTTP de tipo POST y la envía a la API HTTP de EmonCMS, que al ser el mismo servidor, está en local.

Recordemos, que la API HTTP utiliza el APIKEY para conocer el usuario destinatario de los datos que recibe. Por tanto, gracias a que en el topic de MQTT, utilizamos ese APIKEY de escritura, se reutiliza en la petición HTTP POST para que vaya al lugar correcto.

En la figura 30 se observa el caso de un mensaje de ejemplo y el proceso del script.

[38] Del inglés, tema. Son canales de publicación/subscripción en el protocolo MQTT.

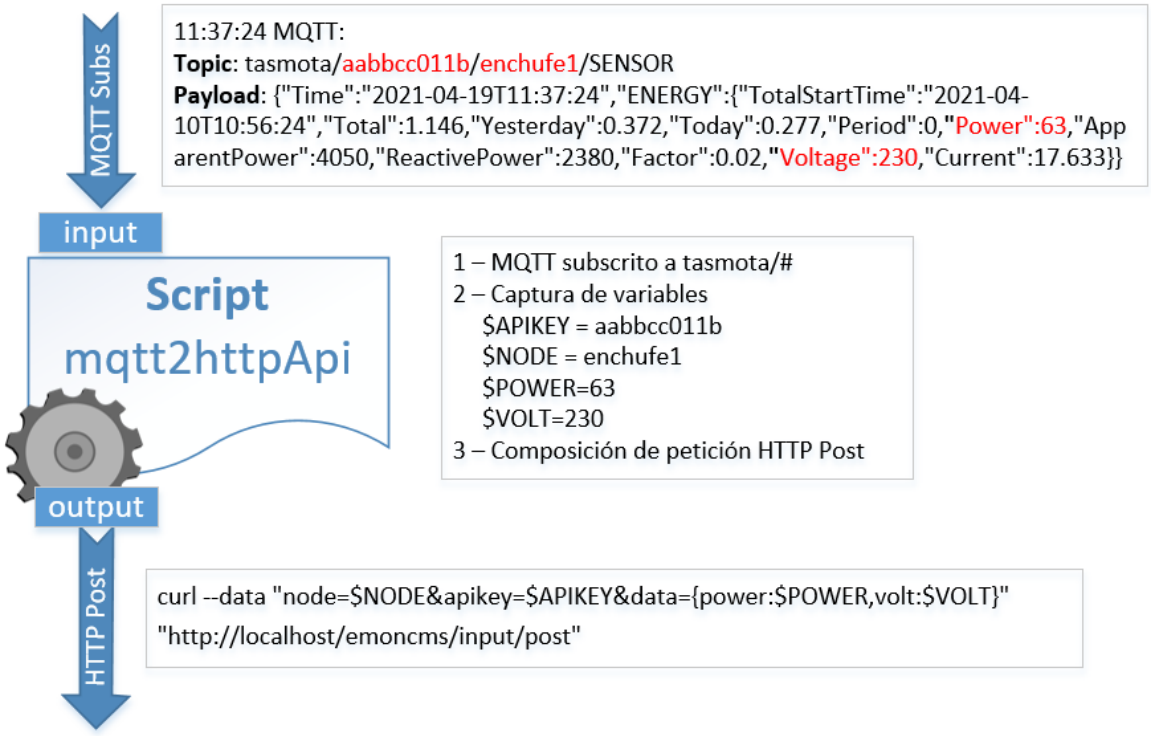


Figura 30. *Transformación de un mensaje de MQTT a HTTP*

5.3 Diseño de los clientes a nivel de hardware

En el siguiente apartado se describirán los dos tipos de clientes utilizados en este proyecto. Es importante recordar, que se han limitado a estos dos, pero que gracias a la API HTTP y a MQTT, es posible crear e integrar cualquier tipo de dispositivo.

5.3.1 Raspberry pi con el Hat para monitorización no invasiva.

Si recordamos la monitorización del consumo no invasiva, decíamos que esta consiste en "abrazar" con un CT, la fase del circuito eléctrico que se quiere monitorizar, para obtener una corriente inducida proporcional sobre el circuito secundario de la bobina que deberá ser leída por arduino o un HAT.

Usar esta metodología nos aporta las siguientes ventajas:

- Capacidad de monitorizar circuitos de gran potencia
- No se modifica ni se interrumpe el circuito original, con lo que no puede ser un punto de fallo.
- Capacidad de monitorizar varios circuitos en paralelo.

Hardware

Implementar este tipo de sensor es posible de muchas formas y, existen numerosos manuales en internet sobre cómo hacerlo con Arduino y similares. En los Anexos hay unos cuantos. Dado que explicar cómo se crea un sensor de este tipo, no es parte del scope del proyecto, se comprará uno en su lugar (de la tienda Lechacal[xxxii]) y explicaremos cómo integrarlo en la plataforma. Esto se ha elegido así, porque se quiere ofrecer a los posibles interesados que estén empezando en este tipo de proyectos, una forma rápida de obtener resultados.

Para ello, el diseño del cliente no invasivo que irá instalado en el cuadro eléctrico para monitorizar varias líneas, entre ellas la general, se compondrá por una Raspberry Pi y un Hat donde conectaremos los distintos transformadores de corriente.

Figura 31. *Ejemplo del HAT instalado en la RPi [Ref[xxiii]]*
http://lechacal.com/wiki/index.php?title=RPICT7V1_Version_5

"Hat" es el nombre que recibe un addon o añadido de hardware que se acopla a la Raspberry Pi por los GPIO[39]. Esto es necesario ya que, por defecto, la Raspberry Pi es una placa que, en esencia, es un ordenador simple, pero carece de las entradas analógicas que le permitirían leer la información de un transformador de corriente.

El Hat, con todos sus componentes, incluyendo el microcontrolador, es el encargado de convertir e interpretar la señal analógica de los sensores y enviarla por puerto serie a la Raspberry Pi.

Software

La Raspberry puede funcionar con muchos SO, y en realidad, para este proyecto no hay ningún requisito concreto en cuanto a éste, más allá de ser Open Source pero, para simplificar, se usará Raspbian[40]. Cómo ya se ha explicado en la sección del hardware, el Hat instalado será el llamado RPICT7V1. Este Hat envía los datos por serie, y la Raspberry Pi los puede leer a través del puerto serie propio incluido en el GPIO. De esta forma, podemos recibir los datos en la RPi, y enviarlos por cualquier protocolo de transmisión, ya sea WiFi o 4/5G, utilizando las API MQTT o HTTP, o mediante la red abierta LoraWan, usando los mecanismos de envío de datos propio específicos del protocolo Lora.

La plataforma EmonCMS tiene un desarrollo propio, llamado EmonHub, que simplifica la tarea enormemente, ya que implementa la lectura de los datos desde el puerto serie y los envía al servidor EmonCMS usando el formato de la API. Solamente es necesario instalarlo en la RPi y configurar la dirección del servidor, la APIKEY del usuario y el nombre del sensor (ver figura 32).

A groso modo, EmonHub, actúa como intermediario entre distintos formatos y protocolos de entrada, (llamados interfacers[41]), y los protocolos de salida directamente a EmonCMS usando las API HTTP o MQTT. De esta forma no es necesario tener que programar ningún tipo de código o script entre estas entradas y la salida que es la API de EmonCMS. Además, EmonHub ya está desarrollado para que funcione en modo

[39] Del inglés, General Purpose Input Output. Son los pines de propósito general de la RPi.
[40] Raspbian, SO oficial de Raspberry Pi.
[41] Del inglés, interface. Son vías de comunicación compatibles con varios protocolos o implementaciones de software, como, UART, Modbus, Tesla API, etc

daemon[42] y es tolerante a fallos, por lo que, en caso de fallo de conectividad con el servidor, tiene mecanismos para guardar toda la información para enviarla después, en el momento que esté disponible.

En nuestro caso usaremos un Hat de la empresa Lechacal (ver figura 31). Un hat puede leer hasta 7 CT, aunque se pueden apilar hasta 8 hats (lo que equivaldría a 56 lecturas simultáneas).

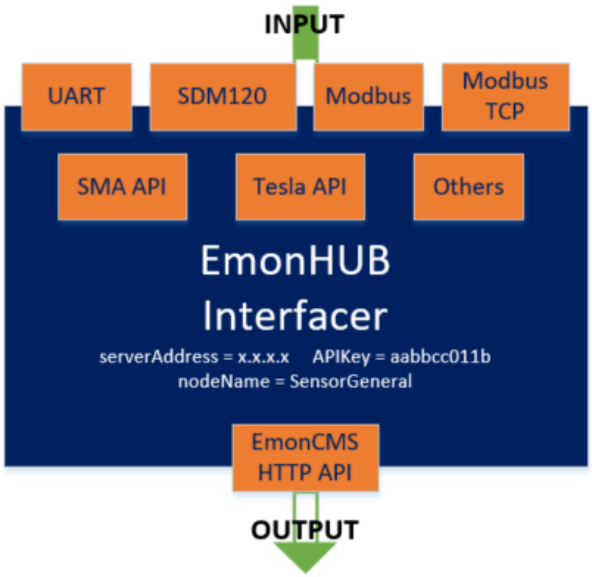


Figura 32. *EmonHub Interfacer [Ref[xxxiv]]*
https://guide.openenergymonitor.org/integrations/emonhub-interfacers/

5.3.2 Diseño de clientes pensados para enchufes.

El otro tipo de sensor es el invasivo, también monitoriza la corriente y, aunque sirve para casi lo mismo, la forma de implementarse es muy diferente, razón por la cual se usa en otro escenario. El factor de forma es de enchufe inteligente y se pone entre el enchufe de pared (hembra) y el enchufe (macho) del dispositivo que queremos monitorizar.

Figura 33. *Ejemplo de enchufe inteligente.*

[42] Del inglés, es un proceso que se gestiona automáticamente (sin intervención del usuario) en el SO.

Estudiar ambos y ofrecerlos como solución nos permite disponer de una plataforma que abarque más tipos de uso y por tanto, que sea más completa.

Con este sensor obtenemos simplificación y facilidad de uso, ya que nos permite integrarlo en el mecanismo de los enchufes de corriente. Además, con él es posible aislar el dispositivo monitorizado en caso de querer conocer un consumo individual.

Hardware & micro-código

A diferencia del anterior sensor, en el que tanto el HW como el SW son completamente accesibles para nosotros, en este caso, estamos usando enchufes inteligentes completamente cerrados de fábrica. En el otro podíamos o decidir programar un script o instalar un software que nos conviene, pero en este caso, estamos sujetos a cómo viene implementado de fábrica y sabemos que no es compatible con EmonCMS.

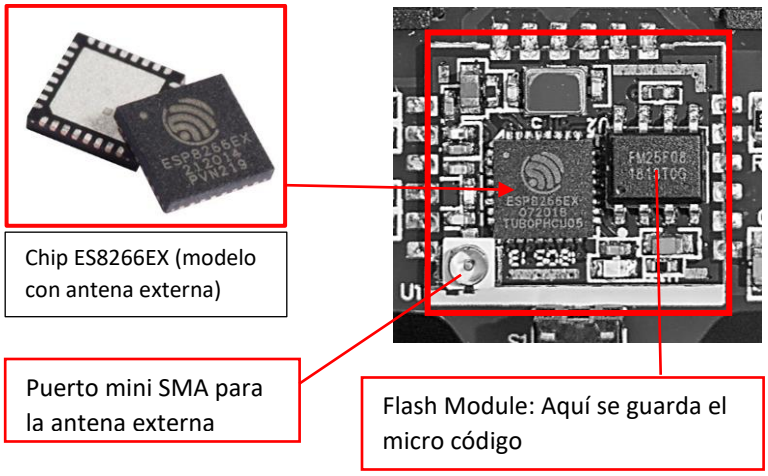


Figura 34. *Chip ESP8266EX de un enchufe inteligente Tuya [Ref[xxxv]]*
https://community.home-assistant.io/t/0-74-tuya-cloudflare-dns-push-camera-and-users-ui/60562/30

Si a pesar de ello, aun lo damos por bueno, es debido a que, hoy por hoy, no se fabrican enchufes inteligentes compatibles con EmonCMS o incluso, con el protocolo MQTT configurable (para configurarlo según EmonCMS requiere). Por otro lado, diseñar nuestros propios dispositivos a medida como prácticamente ocurre en el caso de los no Invasivos, sería lo ideal pero, debido a su factor de forma, supondría un aumento de coste y de conocimientos de electrónica muy avanzados, que serían una gran barrera de entrada.

Afortunadamente, gracias a una fuerte comunidad, se ha logrado convertir los enchufes inteligentes cerrados en enchufes inteligentes abiertos y configurables, con lo que ahora sí es posible integrarlos con EmonCMS. Esto es precisamente lo que se explicará en esta sección.

Para conseguir convertirlos, la comunidad, haciendo ingeniería inversa, ha visto que prácticamente todos los enchufes inteligentes de la plataforma IoT Tuya equipan el micro-controlador Expressif ESP8266. Gracias a ello, han podido instalar el Tasmota[xxxvi], que es un micro-código completamente Open Source y que tiene implementados nativamente el protocolo MQTT y muchas otras funcionalidades configurables.

Son precisamente el protocolo MQTT y la funcionalidad de telemetría, las piezas necesarias para integrarlas en EmonCMS. Por tanto, simplemente es necesario configurar el servicio MQTT como la API

MQTT de EmonCMS espera que le llegue, y la frecuencia, en segundos, con la que tiene que ir mandando toda la telemetría.

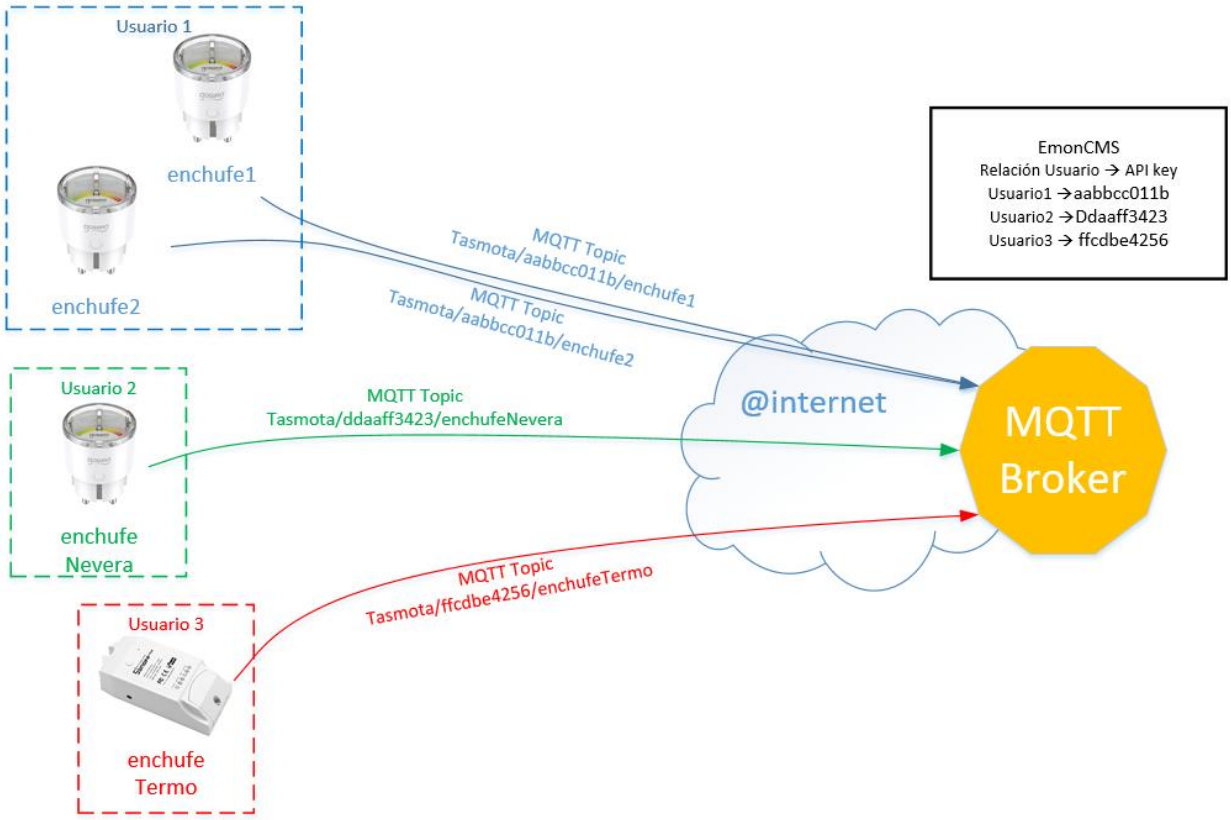


Figura 35. *Flujo de comunicación nodo-cliente mediante MQTT*

5.4 Materiales usados y BOM

En los siguientes puntos se detallarán todos los dispositivos usados para montar la solución. La parte de hardware del servidor se deja fuera de la sección del servidor EmonCMS ya que se ha decidido usar un sistema de virtualización para este PoC, pero que, en un entorno distinto, ya sea de producción o de PoC, bien podría usarse otro hardware, con o sin sistema de virtualización. En cualquier caso, se explicará ligeramente los componentes usados.

5.4.1 Hardware físico y máquina virtual

Como servidor físico se ha usado un ordenador en miniatura, llamado Intel NUC, de arquitectura x86. En una implementación real, lo más habitual sería usar un servidor en formato enrackable. Tampoco hay ningún requerimiento especial en cuanto a las conexiones de entrada. Para el PoC, se ha usado una fibra FTTH doméstica, pero como ya ocurre con el servidor, en una implementación real, lo habitual sería que estuviera tras una conexión empresarial que garantizase unos SLA adecuados y con una IPv4 pública fija, ya que debe conectarse mucha gente al servidor.

- El hardware usado es un Intel NUC modelo NUC7i7BNK[xxxvii] con 16Gb de RAM y 512 GB de SSD.

- El hipervisor[43] elegido es el VMware ESXi 6.0[44] licencia gratuita con licenciamiento para hasta 2 CPU físicas en el hardware.

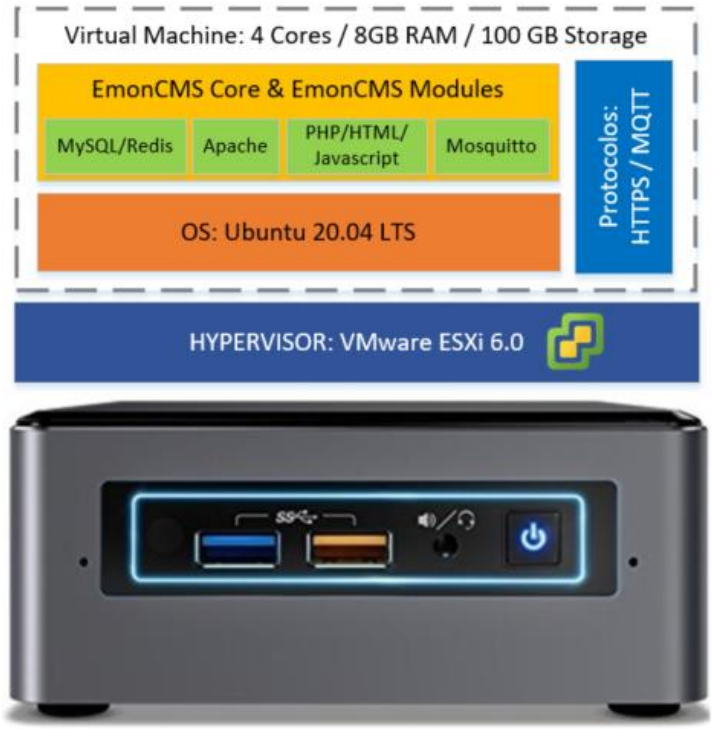


Figura 36. *Arquitectura del servidor*

Partiendo del escenario expuesto, la capacidad del servidor será una máquina virtual de 4 procesadores lógicos, 8 GB de Memoria RAM y 100 GB de disco duro. El servidor estará instalado con Ubuntu 20.04 LTS y con software EmonCMS[xxxviii] de OpenEnergyMonitor project.

Para este proyecto, vamos a suponer que existe una comunidad similar a Guifi.net, con recursos tecnológicos suficientes como para tener un servidor que dé servicio a 100 usuarios con una media de 10 sensores por usuario, volcando datos cada 15 segundos durante 5 años.

5.4.2 Servidor EmonCMS

El sistema operativo (SO) de la máquina virtual es Ubuntu Server 20.04 LTS[45]. Se ha tomado esta decisión en aras de simplificar lo máximo posible, ya que la documentación oficial, si bien no pone que sea un requisito, se realiza utilizando este SO.

Durante la instalación del software de EmonCMS v10, se instalarán los siguientes paquetes de software

- PHP 7.4.3
- MySQL / MariaDB 10.3.25
- Redis-Server 5.0.7

[43] Del inglés, hypervisor. Es un sistema que ejecuta máquinas virtuales en su interior.
[44] SO específico para servidor hipervisor.
[45] Del inglés: Long Term Support. Es un tipo de distribución que tiene un periodo de soporte más largo de lo habitual

- Apache2.4
- Mosquitto MQTT v3.1

Aclaración: Dado que el software se instala desde repositorios de software, es posible que las versiones de cada uno de ellos se incremente con el tiempo.

5.4.3 Cliente para cuadro eléctrico

Como ya se expuso en la parte de diseño, para la monitorización no invasiva es posible usar varios mecanismos, aunque en este PoC, con tal de cumplir con los objetivos de accesibilidad, usaremos una Raspberry Pi con un Hat que son económicos de adquirir y fáciles de configurar:

- Raspberry Pi 3B (o modelo inferior, pero con mínimo 1GB RAM)
  - 4+ GB MicroSD
  - Red Ethernet 100Mbps o WiFi 2,4Ghz o 5Ghz
- Hat Lechacal RPICT7V1 v3 para medición de consumo con varias entradas para CT.
- 5x Transformador de corriente SCT-013-000 y 1x EU AC/AC Sensor
- Sistema operativo Raspbian versión March 4th 2021
- Servicio o daemon: EmonHub.

Por la parte de software se usará Raspbian y EmonHub que son completamente gratuitos.

Figura 37. *Ejemplo de Raspberry Pi con Hat RPICT7V1 y 7 transformadores SCT-013-000 conectados*

5.4.4 Cliente para enchufe

Respecto al cliente invasivo para enchufe, utilizaremos dos dispositivos que realizarán la misma función, pero que se "flashean" con Tasmota de forma distinta. Lo haremos de esta forma para mostrar un par de procesos y dar más elección a los usuarios.

- Dos enchufes inteligentes basados en chip ESP8266 y sensor de corriente basado en el efecto Hall
- Antena WiFi 2.4Ghz b/g/n.
- Flasheados com Tasmota v8.5.1 y 9.2.0 respectivamente.

- Modelos usados:
    - Houzetek AWP07L
    - Sonoff Pow/Pow R2

Figura 38. *Houzetek AWP07L a la izquierda y Sonoff Pow a la derecha*

5.4.5 BOM

La Bom o Bill of Materials, es la lista de hardware usado en este PoC y que se considera el mínimo imprescindible para desarrollar los clientes vistos en los apartados anteriores. Únicamente se incluye la parte del cliente ya que, en este PoC, para la parte del servidor se decide usar una máquina virtual, que no tiene coste.

	Descripción	Unidades	Precio	Precio Final
Raspberry Pi 3B	Base para el sensor no invasivo	1	47 €	47 €
Hat Lechacal RPICT7VI	Hat para el sensor no invasivo	1	52 €	52 €
Carcasa 3D para RPi	Carcasa para Raspberry Pi con Hat	1	6 €	6 €
SCT-013-000	Pinzas transformador de corriente	7	9 €	63 €
EU AC/AC sensor	Transformador para medición de voltaje	1	15 €	15 €
Sonoff POW R2	Enchufe para sensor invasivo	1	18 €	18 €
Houzetek AWP07L	Enchufe para sensor invasivo	1	11 €	11 €
	Total			212 €

6 Implementación

6.1 Instalación del servidor EmonCMS.

Para esta sección solo se describirá brevemente factores a tener en cuenta en la instalación del software de EmonCMS, así como detalles importantes a definir del ecosistema, tales como parámetros de las bases de datos o cierta configuración de algunos módulos.

No se detallará como instalar el entorno virtual o hypervisor, ni tampoco el sistema operativo de la máquina donde correrá el EmonCMS, ya que se considera que el detalle de esos sistemas, no son partes fundamentales del proyecto. Además, tal y como se ha explicado en la parte del diseño, la elección de usar una máquina virtual, es completamente arbitraría ,pues docker o directamente sobre hardware físico serían opciones igualmente válidas.

Finalmente, es importante destacar, que la instalación se ha hecho en base a la versión actual del software y es válida para la fecha en la que se realiza el proyecto. Para evitar que esta información pueda no ser válida o perjudicar a instalaciones en el futuro, se adjuntarán en la referencia, los enlaces a la documentación oficial de las instalaciones de EmonCMS por parte de los autores, que sí se van actualizando con cada versión.

6.1.1 Preparativos

Estas son algunas de las tareas que se recomiendan hacer antes de instalar EmonCMS.

Crear un usuario bajo el que ejecutar EmonCMS y darle permisos de sudo. En este caso será el usuario “emon”.

```
adduser emon
usermod -aG sudo emon
```

Desactivar el prompt[46] del password al hacer sudo.

```
echo emon ALL=(ALL) NOPASSWD: ALL | sudo tee /etc/sudoers.d/emon && sudo chmod
0440 /etc/sudoers.d/emon
```

Permitir el puerto 80, 443 y el 1883 en el firewall interno de Ubuntu.

```
sudo ufw allow 80/tcp
sudo ufw allow 443/tcp
sudo ufw allow 1883/tcp
```

Tras esto, ya es posible descargar el script de instalación:

```
wget
https://raw.githubusercontent.com/openenergymonitor/EmonScripts/stable/install/
init.sh
chmod +x init.sh && ./init.sh
```

[46] Del inglés, es aquella situación en el que el sistema se queda esperando una respuesta del usuario.

Aquí encontraremos una ventana que nos recomienda revisar el archivo de configuración para revisar qué se va a instalar. Este punto es importante ya que, en un inicio, EmonCMS estuvo pensado para ser instalado en Rasperry Pi.

Por tanto, hay que revisar el archivo de configuración para quitar algunos detalles propios de la instalación en RPi.

```
# user - Change user to reflect your OS user
# emonSD_pi_env - Set emonSD_pi_env=0 if not a raspberrypi
user=emon
hostname=emoncms
emonSD_pi_env=0

# RaspberryPi emonSD applicable items - ignored if setting emonSD_pi_env=0
# set to false if not needed
# EmonHub: comment out line above "emoncms_modules[config]=stable" if
# you do not wish to have the emoncms emonhub config module installed
install_emonhub=false
install_firmware=false
install_emonpilcd=false
install_emonsd=false
install_wifiap=false

# Specific EmonPi / EmonBase Modules installed in $emoncms_www/Modules
# Configure branches as applicable
#emoncms_emonpi_modules[config]=stable
#emoncms_emonpi_modules[wifi]=stable
#emoncms_emonpi_modules[setup]=stable
```

Cambiar la contraseña de la base de datos y la contraseña del servicio MQTT.

```
# MySQL
mysql_user=emoncms
mysql_password=emonpiemoncmsmysql2016
mysql_database=emoncms

# MQTT - change settings.ini after install if server is not local
mqtt_user=emonpi
mqtt_password=emonpimqtt2016
```

6.1.2 Instalación del servidor

Finalmente deberemos guardar y ejecutar el script de instalación, que nos volverá a preguntar si queremos modificar el archivo de configuración de la instalación para cambiar algo. Como ya se ha hecho, ahora le diremos que no.

Tras la instalación ya podemos ir a http://localhost/, donde debemos registrarnos.

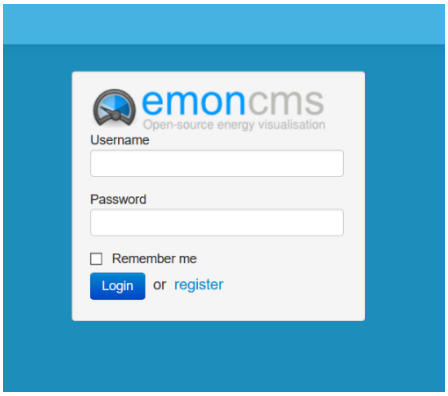


Figura 39. *Portal de bienvenida de EmonCMS*

Importante: el primer usuario de la plataforma será el administrador de la misma, con lo que es importante definirlo correctamente ya que no será posible cambiarlo en el futuro.

6.1.3 Habilitar la función multiusuario en el servidor

Esta funcionalidad o configuración, es la que nos dará la capacidad multitenencia, para poder registrar varios usuarios y que cada uno tenga su propio espacio.

Hay que editar el fichero:

```
nano /var/www/emoncms/settings.ini
```

Y añadir en la sección de [interface] del fichero:

```
enable_multi_user = true
```

```
[interface]
enable_admin_ui = true
feedviewpath = "graph/"
favicon = "favicon_emonpi.png"
enable_multi_user = true
```

Figura 40. *Configuración del servidor*

Ahora ya podemos refrescar la página de EmonCMS para registrar más usuarios.

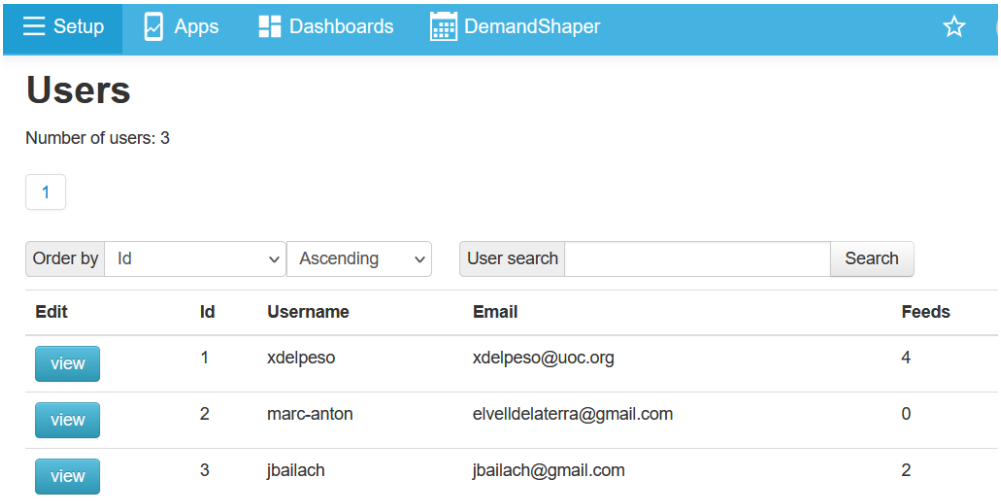


Figura 41. *Listado de usuarios del servidor*

6.1.4 Configurar el servidor MQTT

Como ya se ha explicado en la parte de diseño, una particularidad actual del sistema, es que no está correctamente implementada la API MQTT nativa hacia el módulo Input. En lugar de redirigir los datos de cada posible usuario de la plataforma hacia su propia instancia, los datos son redirigidos al usuario principal que es el administrador. Esto crea varias problemáticas, entre ellas, que se vulnera el aislamiento que deben tener los datos de cada usuario, así como que los usuarios tampoco obtienen la funcionalidad que esperaban ya que no pueden ver sus propios datos.

En un inicio estuvo correctamente implementado[xxxix] y aun se puede encontrar esa documentación que asegura que funciona. No obstante, en algún punto de las versiones recientes, esto ha dejado de funcionar y ha sido corroborado por los desarrolladores en los foros de soporte[xl].

En su lugar, se ha decidido desarrollar[xli] un script a medida que funciona como un daemon en el servidor Linux y que actúa de puente entre MQTT y la API, implementando la funcionalidad nativa perdida.

```
#!/bin/bash

# This script subscribes to the MQTT Tasmota topic using mosquitto_sub.
# On each message received, the content of the topic is splitted and assigned to its
respective variables to be reintroduced into the HTTP API by using a HTTP Post Method.

POWER=;
VOLT=;

while true  # Keep an infinite loop to reconnect when connection lost/broker unavailable
do
  mosquitto_sub -h "127.0.0.1" -u 'emonpi' -P 'emonpimqtt2016' -v -t "tasmota/#" | while
  read -r topic payload
  do
     if [[ $topic == *"SENSOR"* ]]; then
        APIKEY=`echo $topic | cut -f 2 -d "/"`;
        NODE=`echo $topic | cut -f 3 -d "/"`;
        POWER=`echo $payload | jq .ENERGY.Power`;
        VOLT=`echo $payload | jq .ENERGY.Voltage`;
         curl     --data     "node=$NODE&apikey=$APIKEY&data={power:$POWER,volt:$VOLT}"
"http://localhost/emoncms/input/post"
     fi
  done
  sleep 10
done
```

6.2 Instalación de los clientes.

En este proyecto se han limitado los clientes únicamente a 3, uno de tipo no invasivo y otros dos invasivos. Para cumplir con los objetivos de accesibilidad del proyecto, los clientes utilizados están los que cumplen un mejor equilibrio entre precio, facilidad de adquirir, de instalar y parametrizar.

No obstante, en el Anexo 2, se muestra un listado de otros clientes compatibles, que van desde un formato de comprar e instalar sin apenas configuración, hasta clientes que uno mismo puede fabricarse a medida.

6.2.1 Configuracion de la Raspberry Pi

Para la instalación del cliente de tipo no invasivo que irá instalado en un cuadro eléctrico, debemos instalar Raspbian. Raspbian es el SO oficial de Raspberry Pi y se instala "quemando[47]" una imagen de disco en una SD con un programa llamado Win32Img.

Tras haber quemado la imagen de disco en la SD de la RBP, ya podemos arrancarla, no sin antes, haber instalado el Hat de Lechacal.

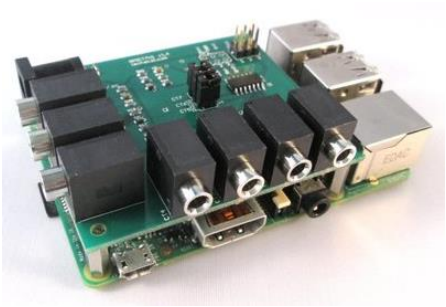


Figura 42. *Raspberry Pi con el Hat Lechacal RPICT7V1 que puede medir hasta 7 líneas eléctricas.*

En ella podemos configurar los parámetros que deseemos, tales como el hostname, configuración IP, interfaz gráfica, y actualizarla con "`apt update`". En principio, para el rol que debe tener este sensor, no son necesarias muchas parametrizaciones personalizadas, ni tampoco una interfaz gráfica, aunque tampoco afecta si finalmente se instala. Se recomienda que usemos la RPi solamente para el rol que tiene asignado. De esta forma, se mantiene todo lo más limpio y simple posible.

La instalación de EmonHub se realiza clonando el repositorio de Git del proyecto. Por ello, lo primero será instalar la herramienta Git, con "`apt install git`". Después ya podemos clonar y empezar la configuración especifica que corresponde a la instalación de EmonHub.

```
git clone https://github.com/openenergymonitor/emonhub.git
cd emonhub
git checkout stable
sudo ./install.sh
```

Podemos encontrar mucha información relativa a EmonHub en las páginas oficiales de la plataforma[xlii].

Después de instalarlo, con el siguiente comando podremos ver datos recogidos por el Hat y mandados a la RBP, en caso de tener la RPi con un CT conectado y rodeando la fase de un cable eléctrico por el que circula corriente.

```
journalctl -f -u emonhub
```

El objetivo, no obstante, es enviar esos datos a EmonCMS. Teniendo la APIKEY de escritura apuntada, debemos configurar el fichero de parámetros de EmonHub de la siguiente forma:

[47] Del inglés, burn. Significa grabar una imagen de disco en un soporte de almacenamiento como una SD o un CD.

```
[hub]
### loglevel must be one of DEBUG, INFO, WARNING, ERROR, and CRITICAL
### see here : http://docs.python.org/2/library/logging.html
loglevel = DEBUG #(default:WARNING)

[interfacers]

[[SerialDirect]]
  Type = EmonHubSerialInterfacer
  [[[init_settings]]]
      com_port = /dev/ttyAMA0
      com_baud = 38400
    [[[runtimesettings]]]
     pubchannels = ToEmonCMS,

[[emoncmsorg]]
   Type = EmonHubEmoncmsHTTPInterfacer
   [[[init_settings]]]
   [[[runtimesettings]]]
       subchannels = ToEmonCMS,
       url = https://emoncms.guifi.net/emoncms
       apikey = xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
       senddata = 1        # Enable sending data to Emoncms.org
       sendstatus = 1      # Enable to send public IP to EmonCMS
       sendinterval= 30    # Bulk send interval to Emoncms.org in seconds

[nodes]

[[11]]
  nodename = my_RPICT7V1
  [[[rx]]]
     names   =   RP1,   RP2,   RP3,   RP4,   RP5,   RP6,   RP7,   Irms1,   Irms2,   Irms3,
Irms4,Irms5,Irms6,Irms7,Vrms
     scales = 1,1,1,1,1,1,1,1,1,1,1,1,1,1,1
     units =W,W,W,W,W,W,W,mA,mA,mA,mA,mA,mA,mA,V
     datacode = 0
```

Después ya podemos reiniciar el servicio EmonHub, o reiniciar la RPi.

```
systemctl restart emonhub
```

En este punto, ya deberíamos estar mandando los datos de los distintos CT a EmonCMS, justamente en el usuario que pertenece la APIKEY usada.

6.2.2 Configuración de una Raspberry Pi con la red abierta LoRaWan

El uso de LoRaWan es algo distinto si lo hacemos con WiFi o 4/5G. Esto es debido a que las otras conexiónes pueden establecer una conexión TCP/IP desde el sensor hasta el servidor, o lo que sería punto a punto. LoRaWan en cambio, no “habla” TCP/IP, sino que es un protocolo de transporte completamente distinto.

Debido a esto, no es posible utilizar EmonHub, aunque está planteado poder adaptarlo en un futuro.

Para usar LoRaWan, debemos capturar las lecturas del HAT y mandarlas directamente a la pila LoRaWan mediante las librerías proporcionadas por la librería[xliii].

6.2.3 Instalación del sensor no invasivo en el cuadro eléctrico

El sensor no invasivo debe monitorizar solamente los cables que son la fase eléctrica de un circuito. La forma de monitorizarlos todos o los que nos interesa es justamente en el cuadro eléctrico, que es el lugar por donde se hace todas las interconexiones y la distribución de la electricidad en un domicilio o empresa.

La RPi puede instalarse dentro del cuadro eléctrico, siempre que haya sitio y habrá que tener en cuenta las conexiones que son necesarias para realizar la monitorización correctamente.

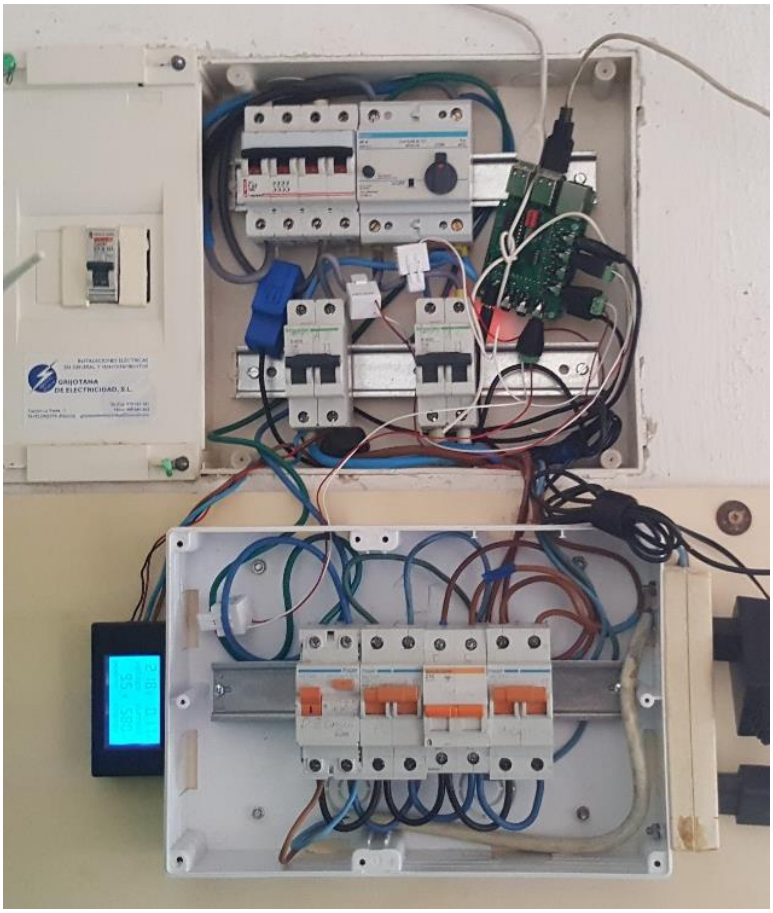


Figura 43. *Instalación del sensor no invasivo en un cuadro eléctrico.*

Por un lado, tenemos las pinzas (los CT SCT-013-000), que deben rodear la fase concreta que queremos monitorizar. Cada una de ellas debe monitorizar una fase diferente. Lo adecuado es monitorizar la fase general y luego una pinza por cada una de las ramificaciones internas (ver figura 43 y 44). También es necesario conectar un sensor de voltaje (AC/AC adapter), que leerá directamente de uno de los circuitos el voltaje en tiempo real para tener mediciones más exactas.

Por último, la RPi debe conectarse a la corriente para alimentarse y así poder funcionar, así como conectarse a la red, ya sea con WiFi, con 4G o con LoRaWan, si disponemos del Hat LoRa adecuado.

Si observamos la figura 43, podemos ver la RPi dentro del cuadro eléctrico y 3 pinzas, 1 azul y otras dos blancas de menor tamaño. A la derecha se aprecia el sensor de voltaje.

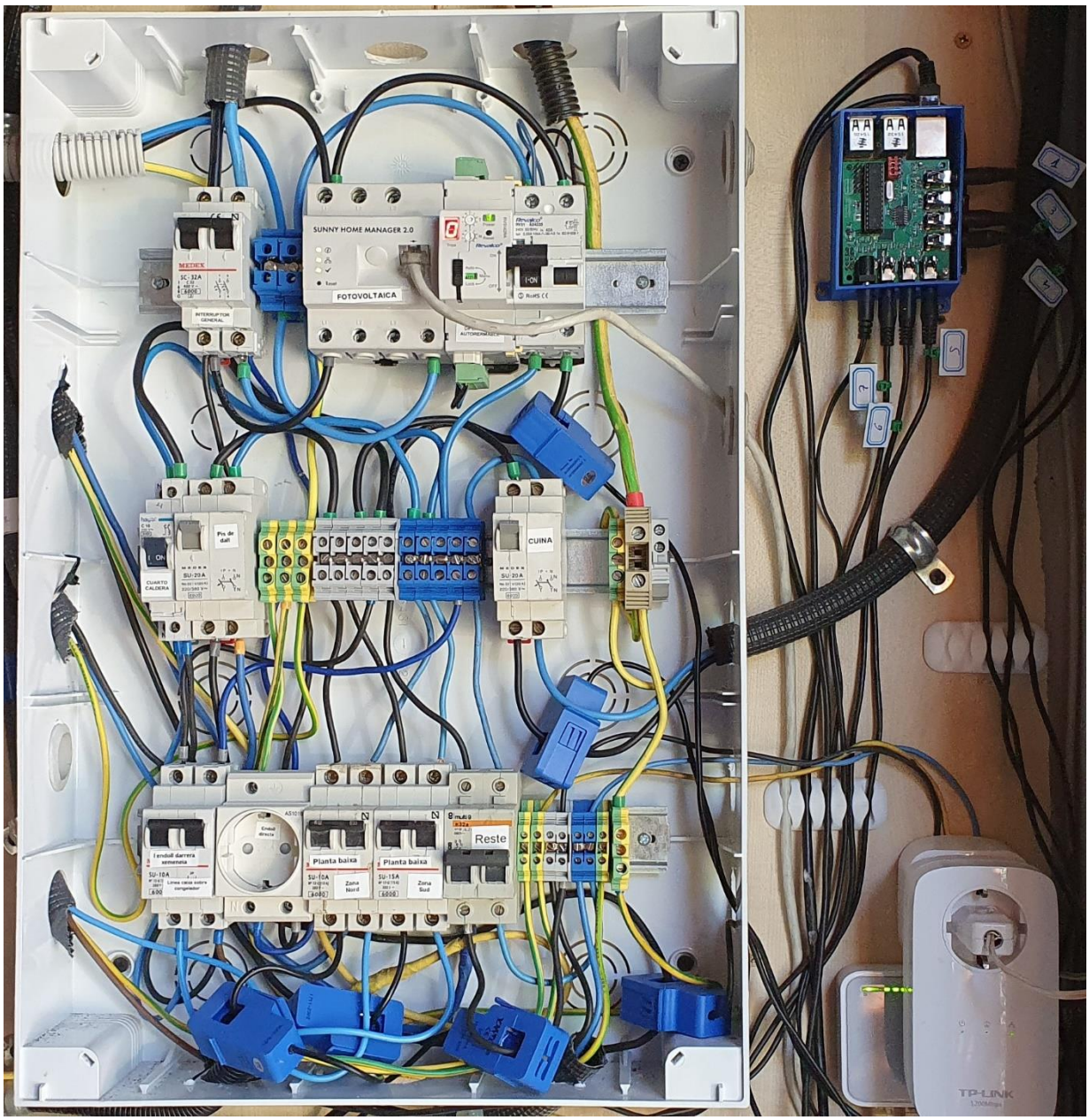


Figura 44. *Instalación de un sensor no invasivo con 6 CT en un cuadro eléctrico de mayor tamaño.*

En la figura 44, podemos ver otro cuadro eléctrico de otra instalación. En esta se puede apreciar que hay varias pinzas SCT-000-013 monitorizando varios circuitos además del general. Lo que no se llega a apreciar es la RPi, que está en otra caja externa por falta de espacio en la principal.

6.2.4 Conversión de un enchufe a Tasmota

Para convertir un enchufe inteligente ya fabricado a Tasmota, hemos adquirido dos modelos. El Sonoff Pow R2 y un enchufe compatible con la plataforma Tuya.

En resumen, el proceso consiste en reemplazar el firmware original de la flash de ambos dispositivos, basados en ESP8266, por Tasmota, un firmware Open Source que permite la integración con otras plataformas más allá que las del fabricante.

Importante: El proceso para flashearlos puede llegar a ser particular para cada dispositivo. Es precisamente aquí donde debemos buscar información y consultar la comunidad, ya que hay muchas personas que investigan cómo hacerlo con cada dispositivo y aportan información de gran valor para el resto.

Sonoff Pow R2

Para este dispositivo, debemos acceder a la parte interna de la circuitería ya que allí encontraremos unos conectores para acceder por serie al dispositivo.

Figura 45. *Conexiones serie del Sonoff POW*

Una vez conectado un conversor Serie a USB, podemos usar nuestro ordenador para “flashear” el firmware que puede ser descargado de la página oficial:

Enlace: https://github.com/arendst/Tasmota/releases/ y http://ota.tasmota.com/tasmota/

El firmware a descargar, en este caso es el básico, “tasmota.bin”. Existen otras variantes para otros idiomas, incluso, una versión lite para dispositivos cuya flash es demasiado pequeña para almacenar la versión completa de Tasmota.

Para flashear el firmware en el dispositivo, podemos usar varios métodos. De hecho, la comunidad del proyecto Tasmota, ha creado un software que simplifica el proceso, llamado Tasmotizer.

En nuestro caso, usaremos la herramienta oficial de Expressif para flashear el SoC ESP8266, llamada esptool[xliv].

```
esptool.py –port /dev/ttyUSB0 write_flash --erase-all --flash_mode dout --
flash_size 4MB 0x0 tasmota.bin
```

Tras esto, el cliente ya estará flasheado, por lo que debería estar creando una WiFi con nombre "***tasmota_XXXXXX-####***". Después de conectarnos a esa WiFi, ya podremos configurar el dispositivo desde un navegador.

Todo el proceso está extensamente explicado en: https://tasmota.github.io/docs/Getting-Started/

Enchufes inteligentes Tuya y variantes

Otro posible dispositivo son los enchufes inteligentes que funcionan con la plataforma Tuya. Como ya se ha comentado en apartados anteriores, la principal ventaja de estos dispositivos, es que ya tienen un factor de forma muy adecuado para lo que se les quiere usar.

La instalación del firmware Tasmota es, en este caso, extremadamente sencillo pero sensible. En la actualidad se utiliza un método llamado "Tuya-Convert"[xlv], que aprovecha una vulnerabilidad de estos dispositivos para lograr instalar el firmware alternativo sin tener que utilizar el acceso por serie al mismo. De esta forma, la instalación es mucho más limpia y rápida, aunque es de esperar que, en el futuro, este método deje de funcionar, por lo que, en dicho caso habría que utilizar el método de flashear mediante serie.

En cualquier caso, en la página de Tasmota se suele explicar el método válido en ese momento para realizar el flasheo.

6.2.5 Configuración del cliente tasmota: wifi, nombre y mqtt.

La configuración de Tasmota, solo consta de unos breves pasos:

Configuración de WiFi y nombre.

Después de flashear el dispositivo, lo primero será conectarnos a la WiFi que crea y configurar los parámetros de la WiFi definitiva.

Debemos indicar el nombre del SSID y el nombre del dispositivo (este nombre aparecerá en la red y lo usaremos para identificarlo en EmonCMS). También es posible indicar una red WiFi secundaria para que, en caso de que la primaria no se encuentre, se pueda conectar a la segunda.

Tras pulsar guardar, el dispositivo se reiniciará y se conectará a la red WiFi que le hayamos indicado. Por eso, nosotros debemos conectarnos también a esa red WiFi nuevamente para terminar la configuración.

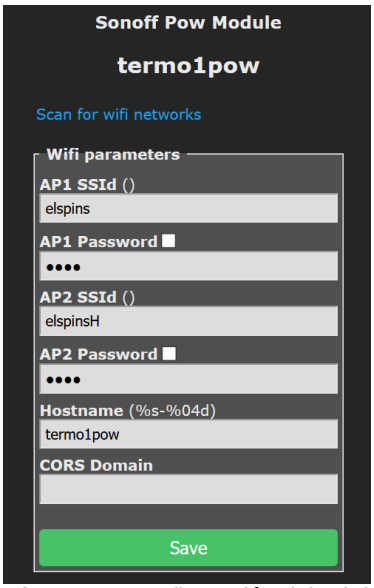


Figura 46. *Configuración del WiFi*

Configuración de MQTT

En este apartado debemos configurar el Broker de MQTT al que nos debemos conectar. Como ya hemos indicado en la parte de diseño, se ha diseñado un script que "traduce" los mensajes de MQTT a HTTP, para usar la API de EmonCMS y de esa forma, poder habilitar la funcionalidad Multitenant incluso en MQTT. Por ese motivo, nosotros, desde el dispositivo debemos mandar el APIKEY por MQTT.

Como se puede observar, el campo Topic, configura el topic de MQTT indicando: que se trata de un dispositivo tasmota (recordemos que el script en el servidor está suscrito a tasmota/#), también, la variable %topic% que es el APIKEY personal de escritura del usuario, y finalmente el nombre del dispositivo bajo el que se identificará en EmonCMS.

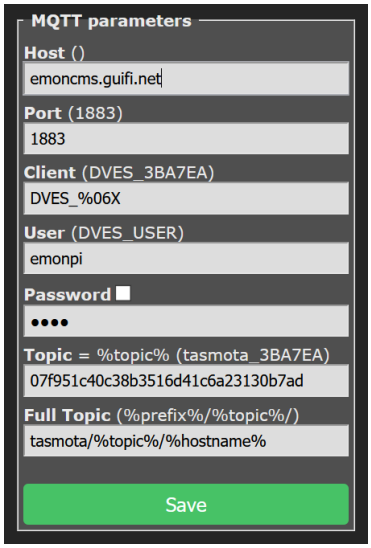


Figura 47. *Configuración de MQTT*

Configuración de tipo de dispositivo y frecuencia de la telemetría.

Hay que indicar el esquema de hardware a usar, internamente llamado "template[48]" (ver figura 48), para que Tasmota "conozca" el tipo de dispositivo en el que está corriendo y mapee correctamente las funciones del software al tipo de hardware bajo el que está instalado. Una vez el hecho, Tasmota sabrá leer correctamente la información de los distintos sensores internos, como el sensor de corriente. Con esto solamente quedaría configurar la frecuencia de la telemetría en el apartado "Logging" (ver figura 49)

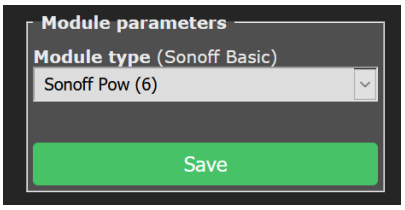


Figura 48. *Configuración de la template*

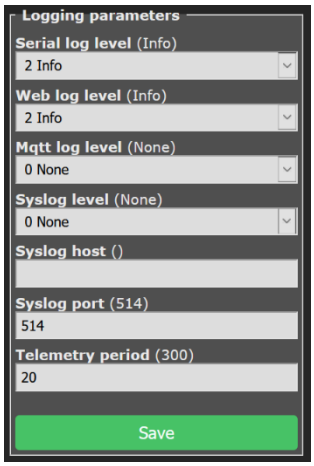


Figura 49. *Configuración de telemetría*

6.2.6 Instalación del enchufe inteligente o de un Sonoff Pow

La instalación de los enchufes es muy simple gracias a su diseño. Solamente es necesario poner el enchufe entre el dispositivo a monitorizar y el enchufe de la pared. De esta forma, toda la energía eléctrica que el dispositivo a monitorizar demande a la red eléctrica, pasará por el monitor, que la ira midiendo y enviando los valores al servidor por MQTT.

[48] Del inglés, plantilla. Es un término común en TI usado para configurar en base a un esquema predefinido.

Figura 50. *Toda una regleta monitorizada por el sensor Sonoff Pow (en el recuadro)*

Figura 51. *Termo eléctrico monitorizado por un enchufe inteligente de la plataforma Tuya transformado a Tasmota*

6.3 Configuración del servidor EmonCMS: inputs, feeds y gráficos.

Una vez se están recibiendo los datos en la parte de Input del servidor significa que la comunicación entre cliente y servidor está funciona correctamente.

Es muy posible que algunos monitores envíen más datos de los que se quieren medir como, por ejemplo, el monitor instalado en el cuadro eléctrico que, por defecto, mide 7 sensores. Puede ser que no todos los sensores estén monitorizando algo concreto, en cuyo caso sus lecturas serán 0, pero que igualmente se mandarán a la plataforma de monitorización. Por ese motivo, hay que diferenciar entre

Input, que es todos los datos que le llegan al servidor y Feed, que son los datos que decidimos guardar y cómo los decidimos guardar (procesado)

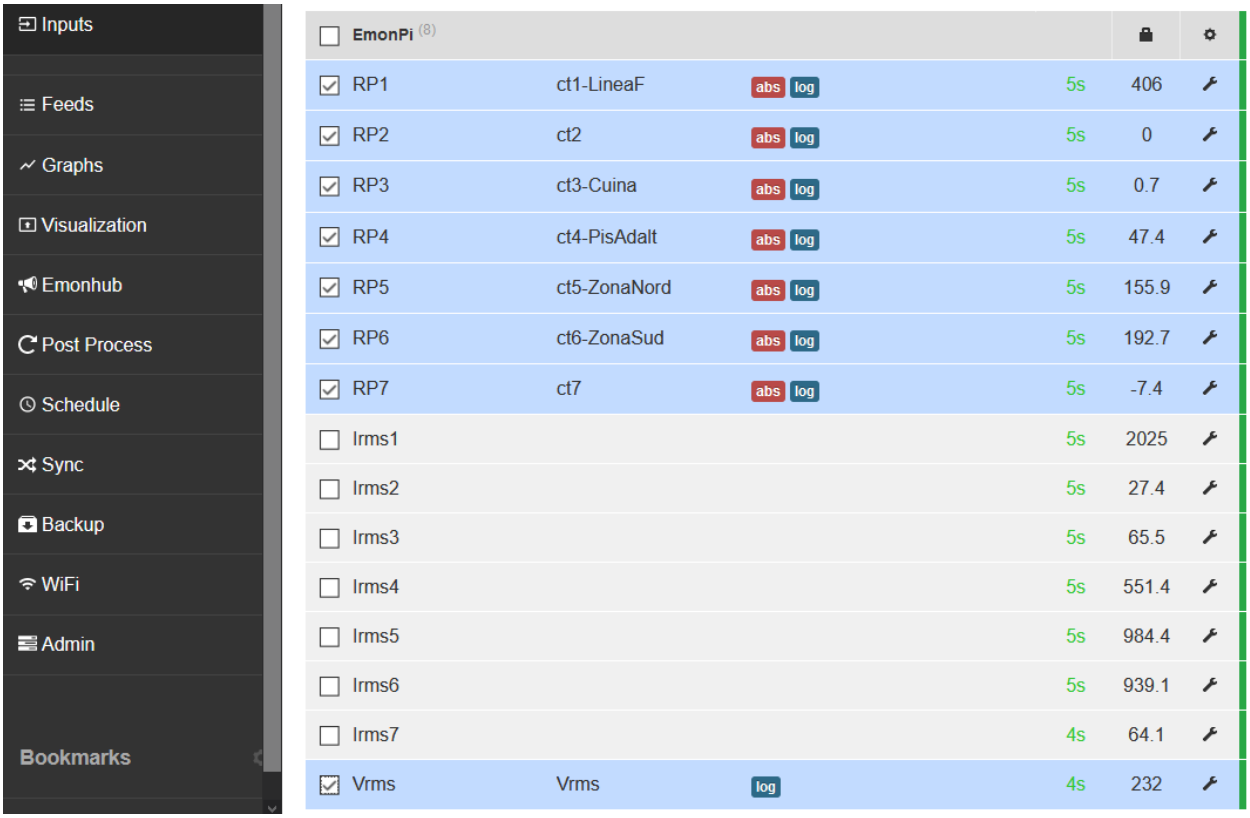


Figura 52. *Ejemplo de un dispositivo en el módulo Input. Se reciben 15 datos, pero se guardan solo los 8 seleccionados (RP1-7 y Vrms)*

Para guardar un input concreto, solamente hay que darle al icono de la llave inglesa a la derecha de ese dato. Se abrirá un panel (ver figura 53) en el que podremos realizar operaciones con el dato hasta guardarlo. Un dato además se puede guardar en varias ocasiones. Se puede por ejemplo recibir los valores, multiplicarlos por 10, guardarlos, luego multiplicarlos por 5 y guardarlos de nuevo. En este caso se habrán creado 2 Feed sobre el mismo input.

Recordemos que cada feed será una entrada en la base de datos de feed y un fichero “id.meta” e “id.dat” en el filesystem donde se guardarán los datos en términos de clave-valor.

En nuestros ejemplos, cuando recibimos un input, le aplicamos la operación de valor Absoluto. Esto es debido a que, en ocasiones, debido a la manipulación física del sensor, las pinzas (o CT SCT-013-000) se colocan al revés, por lo cual, la corriente inducida es inversa y nos da un valor de potencia negativo. Si estos valores negativos se guardan como negativos, el cálculo del consumo acumulado se verá afectado, ya que, en lugar de sumar, restan, lo cual es incorrecto, ya que no es posible tener consumo negativo (significaría que tenemos una fuente dentro de casa y estamos generando electricidad).

Para evitar esos errores en los que la pinza se ha colocado mal, se configura una operación de valor absoluto, de forma que, llegue cómo llegue el resultado, siempre se guardará en positivo.

Sería lo mismo aplicar una operación de detectar si el valor es negativo, multiplicar por “-1” o no tocarlo si es positivo (ver figura 53).

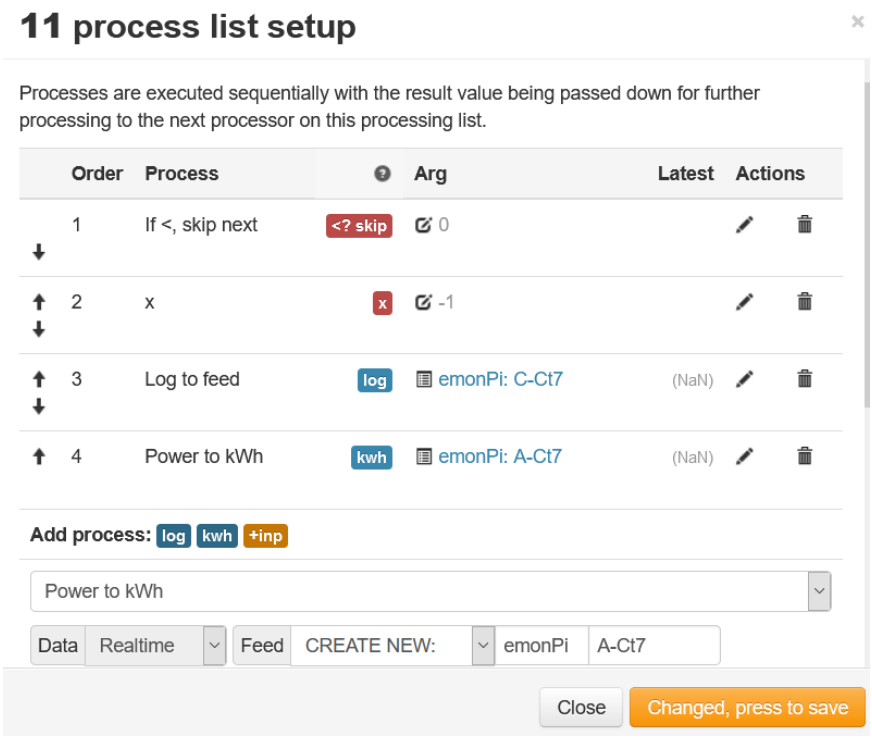


Figura 53. *Ejemplo de un procesamiento para convertir a valor absoluto sin usar la operación absoluto predefinida*

6.4 Visualización de los datos.

Para visualizar los datos, debemos ir a la pestaña Graphs, que internamente utiliza los módulos Graph.php para la representación de los datos y Feed.php para la obtención.

En el margen izquierdo habrá una lista de todos los feed de cada usuario y seleccionando cada uno mediante un checkbox, pasarán a representarse gráficamente en la caja central.

Figura 54. *Es posible usar los dos lados del eje y para representar valores que trabajan en distinta escala o distinta unidad.*

Es posible usar la escala izquierda y la escala derecha simultáneamente para representar dos tipos de datos distintos y que no se desean contrastar. En la parte inferior es posible ver la potencia que ha consumido cada elemento de la gráfica en el intervalo marcado:

Window: 15:26 11 May > 15:36 12 May, **Length:** 24h10 (87000 seconds)

Feeds in view — Show options

Feed	Quality	Min	Max	Diff	Mean	Stdev	Wh
1:termo1pow:C-Termo ()	100% (726/727)	0	1392	1392	143	415	3455
2:ikea1pow:C-Tv ()	100% (727/727)	9	58	49	27	20	661
3:tuya1mini:C-RegletaPC ()	82% (598/727)	6	58	52	28	21	665

Figura 55. *Tabla de valores de cada feed representado con sus medias, máximos y mínimos, así como total de Wh acumulado en ese intervalo.*

7 Análisis de datos de monitorización

En este apartado se expondrá la representación gráfica de algunos de los datos recogidos y se contrastarán varios para comparar consumos eléctricos de distintos tipos de electrodomésticos.

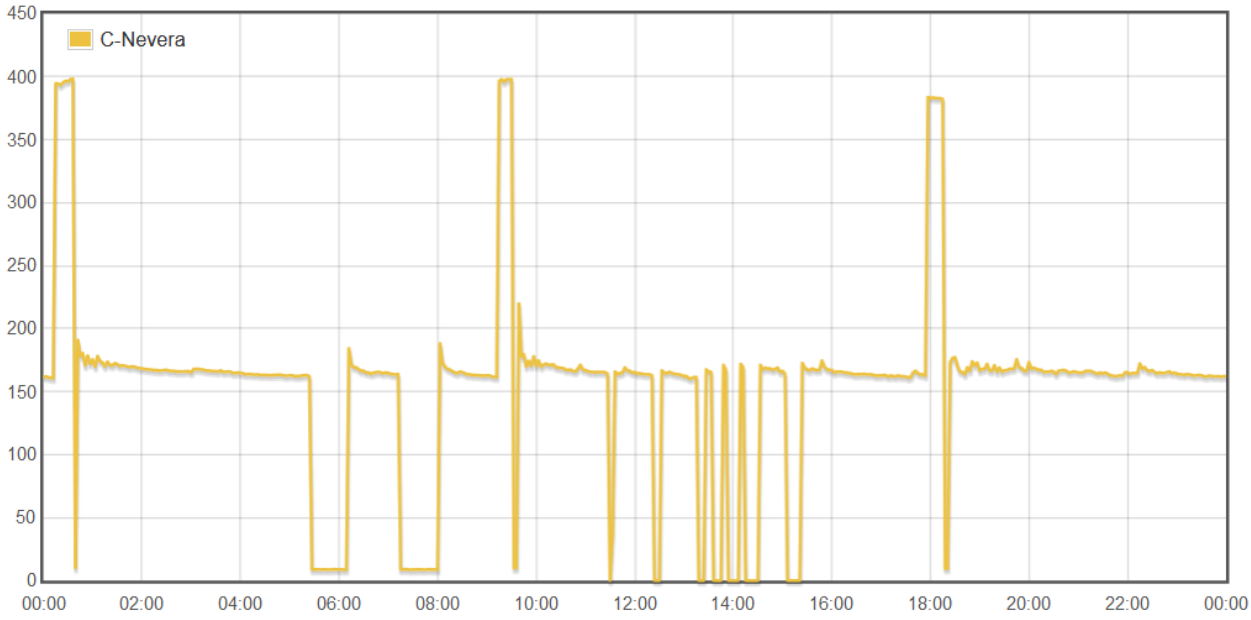
Objetivo es mostrar cómo, mediante el análisis de los datos, es posible llegar a conclusiones muy útiles sobre el comportamiento de nuestros electrodomésticos.

7.1 Consumo eléctrico de una nevera antigua en verano e invierno:

En este escenario estamos comparando el consumo de una nevera de unos 25 años a lo largo de un año. Es fácil pensar que la misma nevera funcionando de forma constante y sin cambiar la temperatura, no debe variar mucho su consumo a lo largo del año. Gracias a la medición del consumo, se puede observar que, el consumo en verano es más del doble del consumo en invierno.

En la gráfica de verano (ver figura 56), se puede observar como la nevera está consumiendo entre 150 y 200W casi constantemente durante todo el día, marcando un acumulado en 24h de 3711Wh.

Por el contrario, en invierno (ver figura 57), se observa un consumo entre 150 y 200W es solamente una parte del día y el resto es reposo. Esto se traduce en un consumo acumulado de 1410Wh en 24h, que es bastante menos de la mitad que en verano.

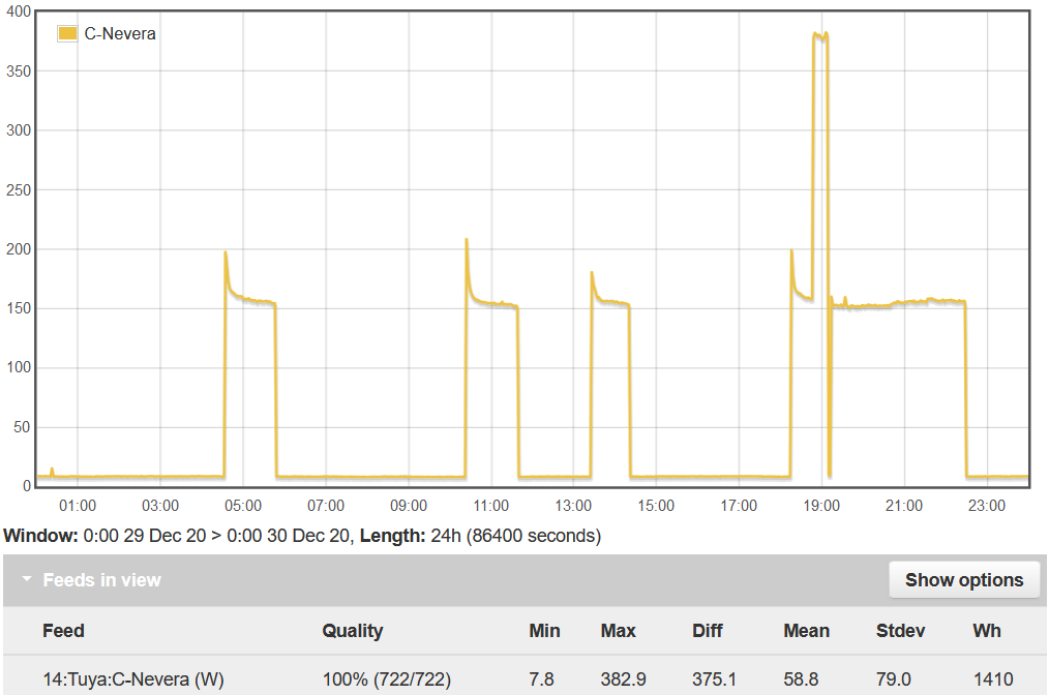


Window: 0:00 26 Jul 20 > 0:00 27 Jul 20, **Length:** 24h (86400 seconds)

Feeds in view — Show options

Feed	Quality	Min	Max	Diff	Mean	Stdev	Wh
14:Tuya:C-Nevera (W)	100% (482/482)	0.0	398.3	398.3	154.6	74.8	3711

Figura 56. *Consumo de la nevera en 24h en Julio ja sido de 3711Wh*



Window: 0:00 29 Dec 20 > 0:00 30 Dec 20, **Length:** 24h (86400 seconds)

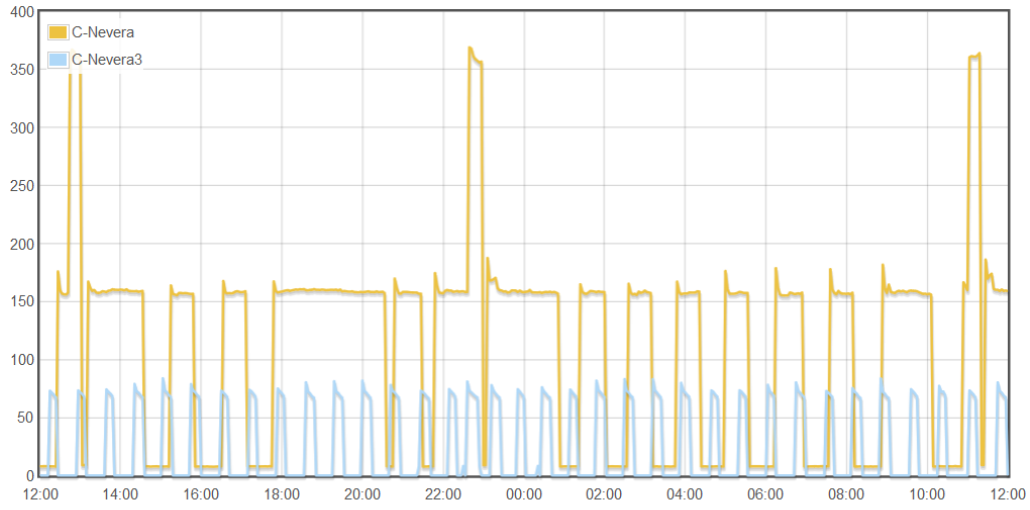
Feeds in view — Show options

Feed	Quality	Min	Max	Diff	Mean	Stdev	Wh
14:Tuya:C-Nevera (W)	100% (722/722)	7.8	382.9	375.1	58.8	79.0	1410

Figura 57. *Consumo de la misma nevera en Diciembre, ahora 1410Wh*

7.2 Comparación de consumo eléctrico de dos neveras

En el siguiente gráfico contrastamos el consumo de dos neveras distintas durante un día:



Window: 12:00 11 Jul 19 > 12:00 12 Jul 19, **Length:** 24h (86400 seconds)

Feeds in view — Show options

Feed	Quality	Min	Max	Diff	Mean	Stdev	Wh
178:Node SPlug: C-Nevera	100% (481/482)	7.5	369.3	361.8	115.2	87.3	2766
176:Node SPlug: C-Nevera3	100% (482/482)	0	84	84	24	34	573

Figura 58. *El consumo de dos neveras contrastado.*

Es fácil observar, como el trazo amarillo tiene un consumo más elevado y prolongado en cotas altas, así que, con toda seguridad, este tendrá un consumo más elevado, pero esto no es muy riguroso. Sin embargo, gracias al detalle de la tabla inferior, podemos apreciar como el consumo es de 2766Wh vs 573Wh, es decir, 5 veces mayor. Esta comparativa, que es de la nevera anterior, contrastada con otra de categoría A+, nos arroja un dato casi demoledor: la nevera de hace 25 años es muy muy poco eficiente.

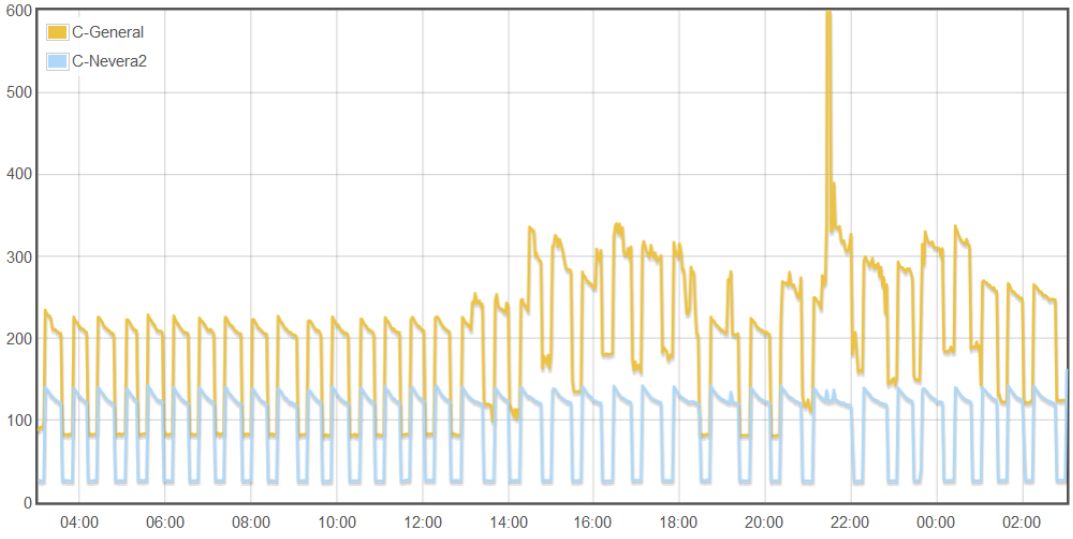
De hecho, si tomamos los datos de un año entero, la monitorización nos revela que esta nevera ha consumido 926.946Wh. Haciendo un cálculo rápido de este consumo a 0.15cent el kWh, sin incluir impuestos ni tampoco el coste de la parte fija de la factura eléctrico, obtenemos que este consumo de luz es aproximadamente 140€ en un año.

Los estudios estiman que un español tiene un gasto medio de 60€ al año en el frigorífico, y con seguridad, esto será una estimación en la que ya se toma en cuenta que una parte de la población puede tener neveras bastante antiguas.

Los españoles pagan **un total de 60,1 euros en electricidad frente a los 54 euros que abonan de media en el resto de países de la Unión Europea (UE)** por tener el frigorífico encendido durante un año.

Figura 59. *Artículo del periódico el ABC*
https://www.abc.es/economia/abci-cuanto-pagamos-espanoles-tener-frigorifico-encendido-durante-201912030206_noticia.html

7.3 Contraste nevera vieja en consumo total de la vivienda



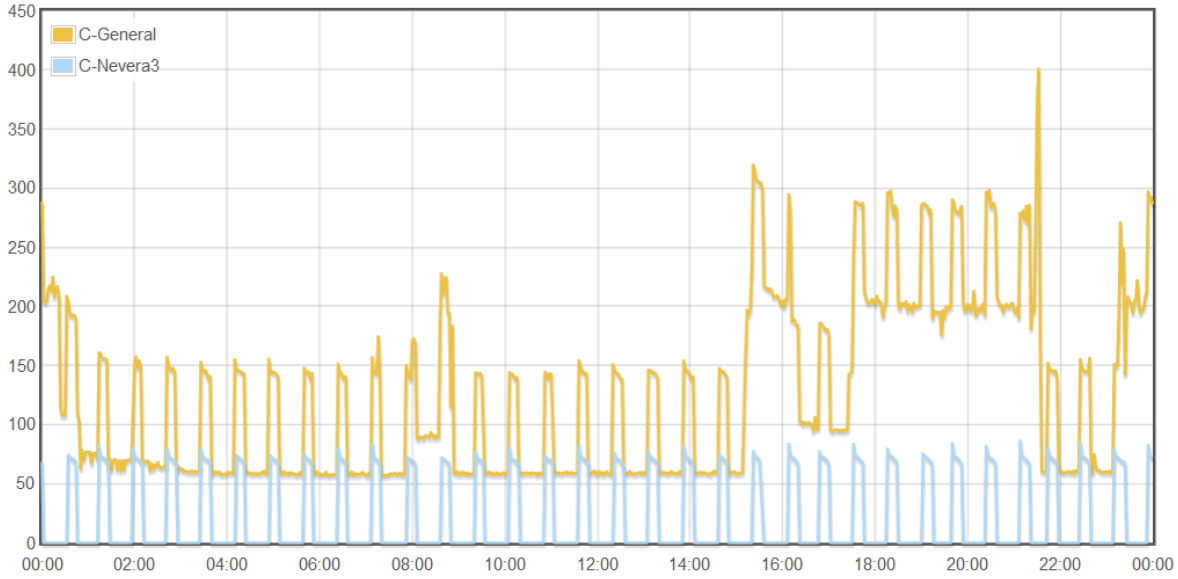
Window: 3:02 19 Aug 18 > 3:02 20 Aug 18, **Length:** 24h (86400 seconds)

Feeds in view							Show options
Feed	**Quality**	**Min**	**Max**	**Diff**	**Mean**	**Stdev**	**Wh**
151:Node 12: C-General	99% (712/722)	80.1	1193.4	1113.3	203.4	88.7	4882
95:Node SPlug: C-Nevera2	98% (711/722)	26	163	137	92	50	2215

Figura 60. *Consumo total de un domicilio contrastado con una nevera antigua.*

En la figura 61 se aprecia cómo el consumo de una nevera vieja impacta significativamente sobre el consumo total de una vivienda. En este intervalo de tiempo, casi corresponde a un 50% del consumo total del domicilio.

En el mismo domicilio, tras adquirir una nevera de segunda mano por 50€, que no es de gama alta obviamente pero sí mucho más nueva, el consumo se redujo significativamente, lo cual supuso en este caso, una muy buena inversión.



Window: 0:00 2 Aug 19 > 0:00 3 Aug 19, **Length:** 24h (86400 seconds)

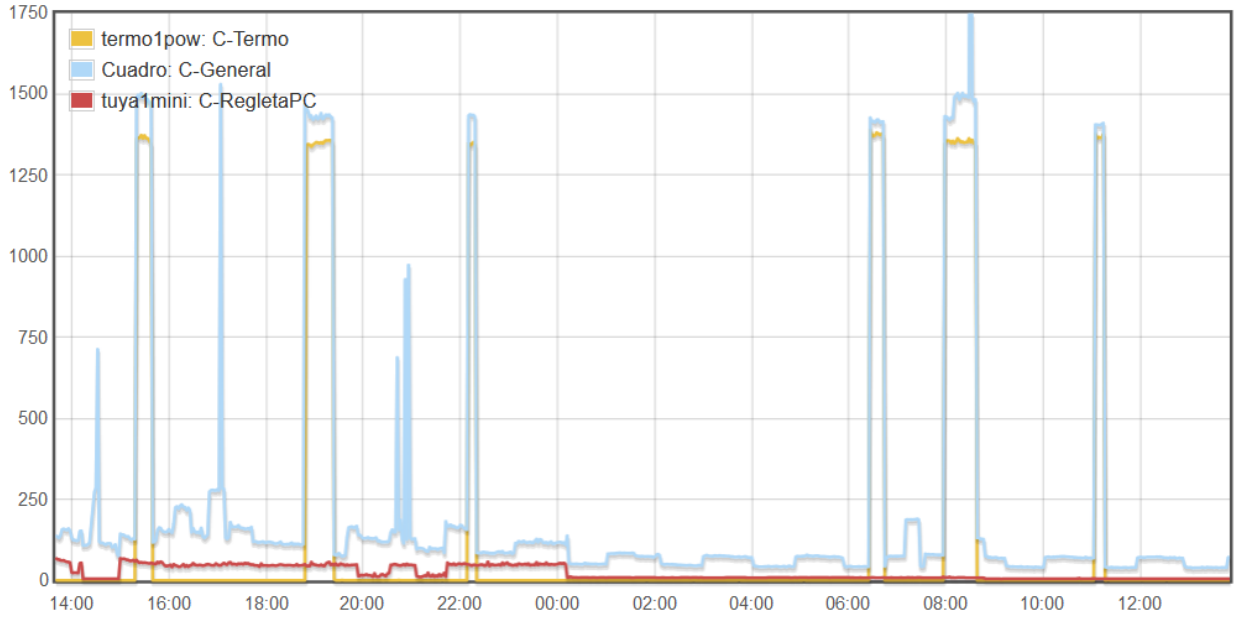
Feeds in view							Show options
Feed	**Quality**	**Min**	**Max**	**Diff**	**Mean**	**Stdev**	**Wh**
151:Node 12: C-General	99% (713/722)	56.5	400.9	344.4	128.0	76.0	3073
176:Node SPlug: C-Nevera3	100% (719/722)	0	86	86	22	33	525

Figura 61. *Comparación del consumo de una nevera de ocasión y el consumo general.*

Por otro lado, es de esperar que la diferencia entre una nevera A+ y una A++ no sea significativa, por lo que no compense adquirir una nevera nueva.

7.4 Consumo eléctrico de un termo eléctrico

En el siguiente gráfico observamos el consumo eléctrico de un termo eléctrico y, sobrepuesto sobre el mismo gráfico, el consumo total de un domicilio completo en el que vive una persona y también la regleta de enchufes donde está conectado un ordenador. Cómo se aprecia en el gráfico, el termo solo está unos intervalos del día, no obstante, si revisamos el consumo total de la vivienda en comparación al termo, vemos que el termo es algo más del 50% del consumo total. Con este análisis, vemos que se suele pensar que la nevera siempre es el gran motivo de consumo en un domicilio, pero mediante datos se demuestra que no siempre es así.



Window: 13:40 2 Jun > 13:50 3 Jun, **Length:** 24h10 (87000 seconds)

Feeds in view — Show options

Feed	Quality	Min	Max	Diff	Mean	Stdev	Wh
1:termo1pow:C-Termo (W)	100% (727/727)	0.0	1390.0	1390.0	125.2	392.9	3026
10:Cuadro:C-General (W)	100% (725/727)	38	2730	2692	229	411	5526
3:tuya1mini:C-RegletaPC (W)	100% (727/727)	4	70	66	22	20	540

Figura 62. *Contraste del consumo general, el termo y una regleta donde hay un PC*

8 Conclusiones del proyecto

Históricamente, desde una perspectiva de usuario doméstico o pequeña empresa, la medición o monitorización de ciertos parámetros como el consumo eléctrico, no ha sido ni muy relevante ni posible. Las dificultades técnicas asociadas a estos sistemas de medición, que necesitaban ciertas infraestructuras y altos conocimientos, se le sumaban altos costes, lo que limitaba el uso de estos sistemas al sector puramente empresarial.

No obstante, en los últimos 15 años hemos sido testigos de una gran revolución tecnológica que ha propiciado un absoluto cambio de paradigma. Las plataformas Cloud, el Big Data y el IoT, términos que antes eran casi desconocidos, han irrumpido con mucha fuerza y ahora, son la base de elementos que conforman nuestro día a día

Por eso, una vez esta revolución se ha normalizado en la sociedad, se plantea un buen momento para proponer un proyecto, en forma de prueba de concepto, que permita medir el consumo eléctrico a todos aquellos usuarios domésticos o empresas que hasta ahora no habían podido acometer un proyecto así por las dificultades que antes entrañaba.

Para alcanzar el objetivo en el que todos los usuarios puedan monitorizar los dispositivos de sus domicilios sin grandes costes ni grandes conocimientos, se ha optado por montar una plataforma de medición de consumo siguiendo un modelo cooperativo y Open Source.

Por un lado, el diseño de la plataforma contemplaba una parte cloud o de servidor central y deslocalizado. Para ello, el PoC ha explorado con éxito la viabilidad una persona u organización, con unos conocimientos avanzados, puedan poner a disposición de otros, como ya ocurre con los supernodos de la red Guifi.net, una plataforma de medición de consumo sin incurrir en grandes costes. Se ha decidido usar software Open Source, que permite una gran comprensión y adaptación, así como tecnologías muy conocidas, tales como máquinas virtuales, MySQL, Apache y lenguaje PHP.

Las comunicaciones eran otro de los grandes escollos de la parte del servidor, que también se han podido validar. Primero, era importante que no estuviera restringido a ningún protocolo de comunicación concreto que limitase el producto. Era necesario que pudiera ser expandido o personalizado por la comunidad y, gracias a la API HTTP y MQTT del servidor, es posible integrar cualquier tipo de dispositivo. En segundo lugar, la disponibilidad actual de líneas fibra óptica de gran capacidad a un precio muy asequible, permite ejercer el rol de servidor sin grandes impedimentos.

El segundo objetivo explorado con éxito era que, el usuario, en un rol de cliente y sin la necesidad de tener grandes conocimientos técnicos, pudiera monitorizar cualquier tipo de dispositivo con un gran detalle. En este punto el PoC ha contemplado estudiar dos tipos de dispositivos, uno que se utiliza para medir el consumo en el cuadro eléctrico y otro para monitorizar dispositivos simples como una nevera o un termo eléctrico.

En primer lugar, para la monitorización en el cuadro eléctrico, se ha usado una Raspberry Pi, que es una placa de desarrollo ampliamente conocida. Gracias a que es ampliamente expandible, hemos comprado un Hat para monitorizar varias líneas eléctricas simultáneamente. En segundo lugar, para la monitorización de dispositivos individuales, era primordial utilizar un dispositivo simple de usar y de mover, para poder cambiarlo de lugar con gran facilidad y así poder monitorizar varias cosas. Para ello, en

el PoC se ha usado un interruptor inteligente que se vende completamente fabricado, y se ha adaptado para usar protocolos abiertos y de terceros como este proyecto. En conclusión, gracias a la arquitectura abierta, ha sido bastante fácil encontrar, tanto dispositivos compatibles a la venta, como manuales sobre cómo construir o adaptar uno por uno mismo.

Por último y no menos importante, el motivo principal del producto en sí, es saber, por medio del análisis del consumo eléctrico, si nuestra huella eléctrica es eficiente o si la podemos mejorar. A parte de que podemos ahorrarnos dinero en nuestra factura eléctrica, lo importante (y cada vez pesa más en la conciencia de la sociedad) es el uso sostenible de los recursos.

Gracias a unos breves análisis hemos visto que, efectivamente, existen algunos dispositivos que pueden llegar a consumir mucho más de lo que se piensa y que, gracias al estudio, es fácil de comprobar que una renovación de algún electrodoméstico, puede acabar teniendo un retorno económico más rápido de lo que parece.

Debido a la temática definida y los objetivos del proyecto, un servidor cooperativo para la medición de consumo y unos clientes accesibles para todo el mundo, no se ha explorado otro tipo de integraciones y mediciones. No obstante, gracias a su arquitectura completamente abierta, tanto en el lado del cliente como del servidor, queda pendiente de validar otros usos, clientes, como por ejemplo de fabricación propia, y con uso de redes abiertas como LoRaWan.

De hecho, aunque ha quedado fuera de la memoria del trabajo, se ha integrado con éxito los datos de generación de energía fotovoltaica y de un sensor anemómetro. Con la energía solar como parte de los datos, se podría dar un vuelco a los resultados de los análisis y con el anemómetro sería posible hacer un estudio de viabilidad sobre un aerogenerador.

No obstante, también se han detectado algunas carencias en la plataforma durante su implementación en el PoC. La primera de ellas, es relativa a la seguridad. Se han detectado ciertas partes de la plataforma que no son del todo seguras, como que cada persona puede registrar tantos usuarios como desee, o incluso, alguna que otra función que debería estar únicamente disponible a administradores. Tampoco está preparada por defecto para encriptar los mensajes MQTT, lo cual es especialmente grave, porque debido a una carencia de la plataforma y a su workaround en el PoC, se envía un dato confidencial en los mensajes. A pesar de la gravedad, esto se podría llegar a corregir con un poco de investigación por parte de los interesados.

Por otro lado, las características multiusuario y multitenencia son características poco maduras dentro de la plataforma y, a pesar de que cumplen con lo mínimo, se han detectado algunos puntos en los que hay carencia de funcionalidades en estos aspectos. En cierta parte es normal porque el target principal de esta aplicación es el uso individual en cada domicilio y, de hecho, en la práctica es así ya que hay muy poca información sobre el uso que le hemos dado en este PoC.

Finalmente, el transcurso y desarrollo del proyecto se ha realizado en gran parte siguiendo la planificación. A pesar de ello, durante la realización del mismo, ha habido que acotar una parte, ya que la temática de la medición eléctrica y telemetría es tan amplia, que muchos sistemas eran interesantes de integrar, pero que podían difuminar los objetivos principales del proyecto.

Anexo 1: otros usos útiles de la plataforma

Como ya se indica en el nombre de la sección, aquí se expondrán algunos otros usos de la plataforma. Hay que recordar que esta plataforma de monitorización de consumo en realidad es extensible a otros posibles usos. Como ya se ha comentado en capítulos anteriores, la disponibilidad de una API y varios protocolos, así como una arquitectura abierta por parte de los clientes, permite integrar muchos otros tipos de sensores.

Sensor o volcado de datos fotovoltaicos de un inversor.

Si disponemos de un inversor de corriente que transforma la energía continua de las placas solares, en alterna de 220-240V de la red normal, esos datos se pueden exportar hacia EmonCMS. Para ello, necesitaremos un dispositivo que se pueda conectar a la API del inversor (casi todos tienen una) extraer los datos en tiempo real y, al mismo tiempo, enviarlos a EmonCMS usando la API HTTP o MQTT.

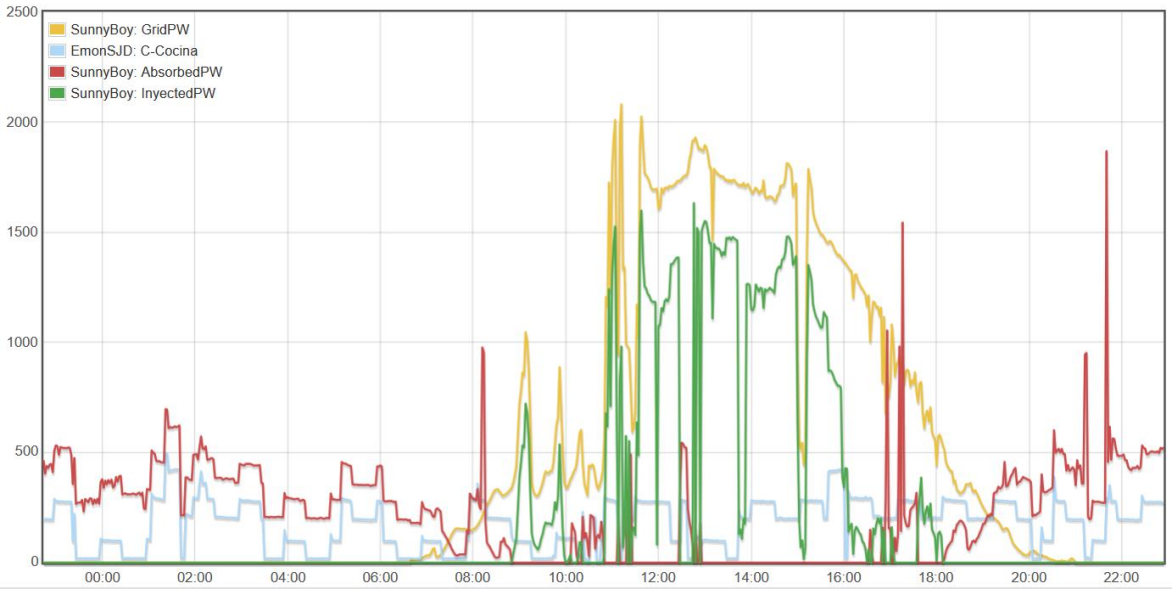


Figura 63. *Ejemplo de un procesamiento para convertir a valor absoluto sin usar la operación absoluto*

En la figura 58, estamos viendo en EmonCMS, la potencia generada por placas en amarillo, en azul el consumo de la cocina, que incluye la nevera que consume tanto del análisis de la sección 7. En rojo la energía absorbida de la red pública, es decir, energía de pago y en verde, la energía excedente de las placas vertida a la red pública.

Sensor o volcado de datos de nivel de carga eléctrica de una batería

Las baterías son una tecnología de apoyo al autoconsumo que todavía no está demasiado extendido. No obstante, en caso de disponer de un sistema de baterías en el que es posible leer la carga, estas lecturas es posible capturarlas y llevarlas a EmonCMS.

Sensor de viento o anemómetro.

Con un anemómetro y una RPi se pueden hacer lecturas de la velocidad del viento. Algo así nos puede permitir hacer un estudio de viabilidad de un aerogenerador que, por ejemplo, complemente las placas fotovoltaicas para momentos sin sol o con pocas horas solares al día. Normalmente, para que sea viable instalar un aerogenerador, debe haber vientos de forma sostenida sobre los 12-15m/s. En la figura X, ya se puede ver, como efectivamente no es viable.

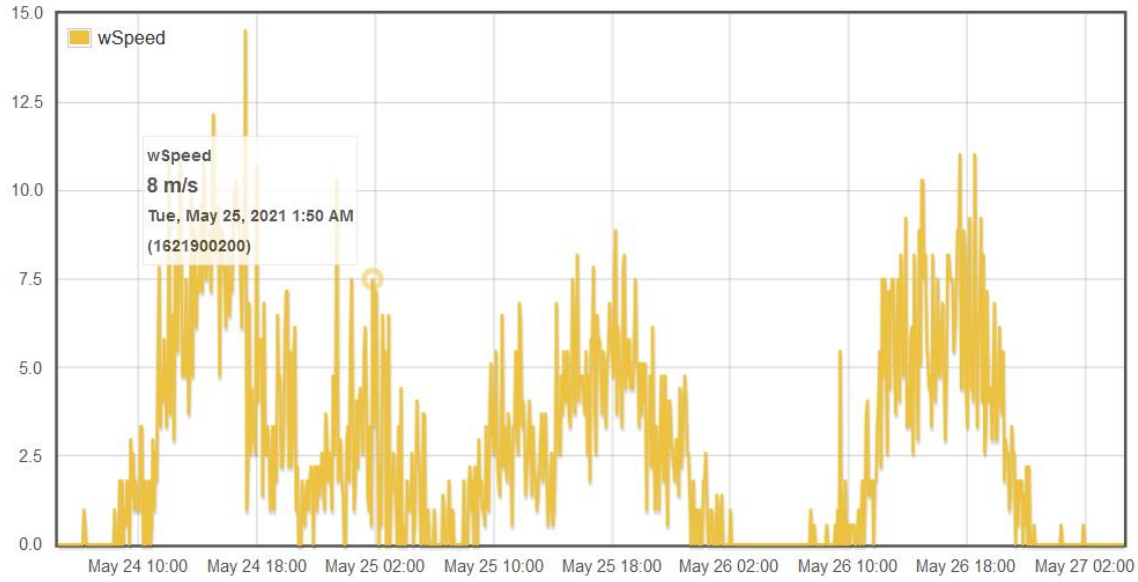


Figura 64. *Gráfica de la velocidad del viento durante unos 2 días.*

Sensor de flujo para medición de agua.

Con sensores de flujo se puede medir el caudal y leer la señal con una RPi o una Arduino.

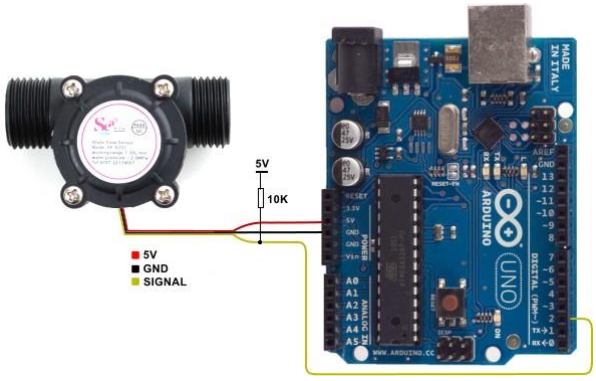


Figura 65. *Arduino con un sensor de flujo.*

Sensor de temperatura y de humedad.

Este sensor es otro de los clásicos y es extremadamente económico. Es posible incorporar estos datos a EmonCMS y contrastar la evolución de distintas estancias para ver cuáles de ellas sufren más variación.

Anexo 2: otros clientes compatibles

En este proyecto se han acotado los tipos de clientes a solamente dos tipos, el invasivo con una Raspberry Pi y a dos enchufes inteligentes con Tasmota integrado. No obstante, tal como ocurre con el otro anexo, gracias a la API, muchos otros tipos de cliente son posibles.

Sensor de corriente simple e invasivo con Arduino

Posiblemente éste sea el caso más simple de sensor. Consiste en una Arduino con una pinza CT (el transformador de corriente) rodeando la fase de un circuito.

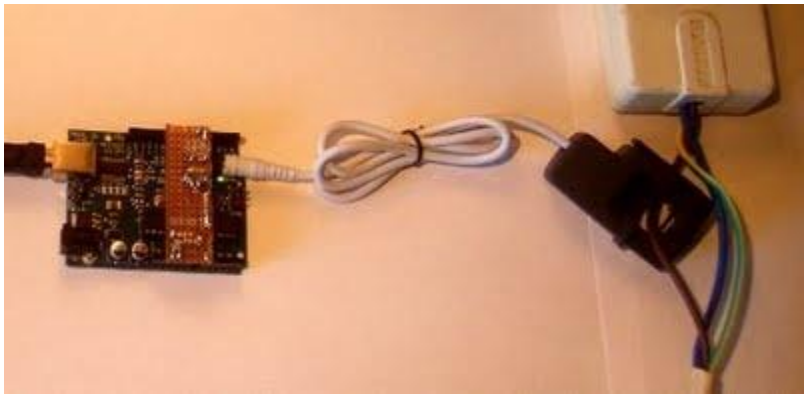


Figura 66. *Arduino con una pinza CT*

Enlace del proyecto: http://openenergymonitor.blogspot.com/2009/08/measuring-ac-mains-energy-use-non.html

Sensor de corriente integrado en Carril Din

El siguiente proyecto tiene una implementación mucho más elaborada y puede integrarse en Carril Din, lo que permite una integración mucho más natural en un cuadro eléctrico.

Figura 67. *Arduino con una pinza CT*

Enlace del proyecto: https://industruino.com/blog/our-news-1/post/proto-power-meter-20

Sensor de corriente de tipo invasivo

Este proyecto consiste en construir un sensor invasivo utilizando tecnología de comunicaciones Zigbee.

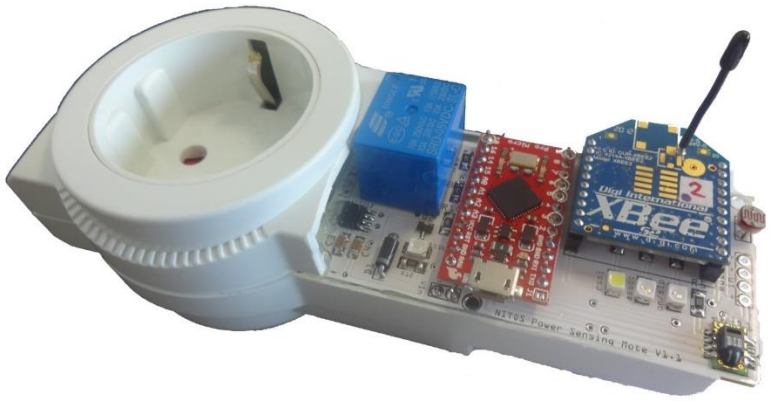


Figura 68. *Arduino con una pinza CT*

Enlace del proyecto: http://nitlab.inf.uth.gr/NITlab_old/index.php/component/content/article/12-research/publications/282-control-and-energy-monitoring-of-electrical-appliances

EmonTx de OpenEnergyMonitor

La plataforma también tiene sus propios productos. Gracias a que son sus propios productos, la utilización e instalación es extremadamente sencilla y además tienen un formato muy adecuado. En particular, EmonTx, además de medir el consumo, también tiene un sensor de temperatura.

Figura 69. *Arduino con una pinza CT*

Enlace del proyecto: https://guide.openenergymonitor.org/technical/emontx/

Sensor de corriente masivo Open Source.

Iotawatt es otro proyecto activo y en desarrollo que ofrece un sensor de corriente con hasta 16 líneas independientes. Entre otros, se integra con EmonCMS, con lo que tiene la API implementada internamente. Esto hace que, junto con EmonTx, su instalación y uso sean extremadamente simples.

Figura 70. *Arduino con una pinza CT*

Enlace del proyecto: https://iotawatt.com/ y https://blog.openenergymonitor.org/2017/10/iotawatt/

Dispositivos con Tasmota preflasheado de fábrica.

En ocasiones, flashear un enchufe inteligente puede llegar a ser complicado por varias razones. Frecuentemente, los enchufes inteligentes se venden sin tornillos ni una forma fácil para abrirlos. En la comunidad se pueden encontrar muchos foros con indicaciones y recomendaciones. En el siguiente proyecto hay una recopilación de enchufes inteligentes y otros dispositivos con notas sobre como abrirlos, flashearlos y otras recomendaciones.

También tienen una sección de productos que se venden directamente preflasheados con Tasmota para simplificar el proceso al usuario.

Enlace del proyecto: https://templates.blakadder.com/eu.html#Plug y https://templates.blakadder.com/preflashed.html

Hat para monitorización no invasiva con Raspberry Pi.

El siguiente producto es para integrarlo con la Raspberry Pi. Su formato de forma, permite instalarlo de una forma muy ordenada siempre y cuando los cables discurran en paralelo. En ocasiones, mantener un cuadro eléctrico muy estructurado es la única opción, y este tipo de Hats lo permiten.

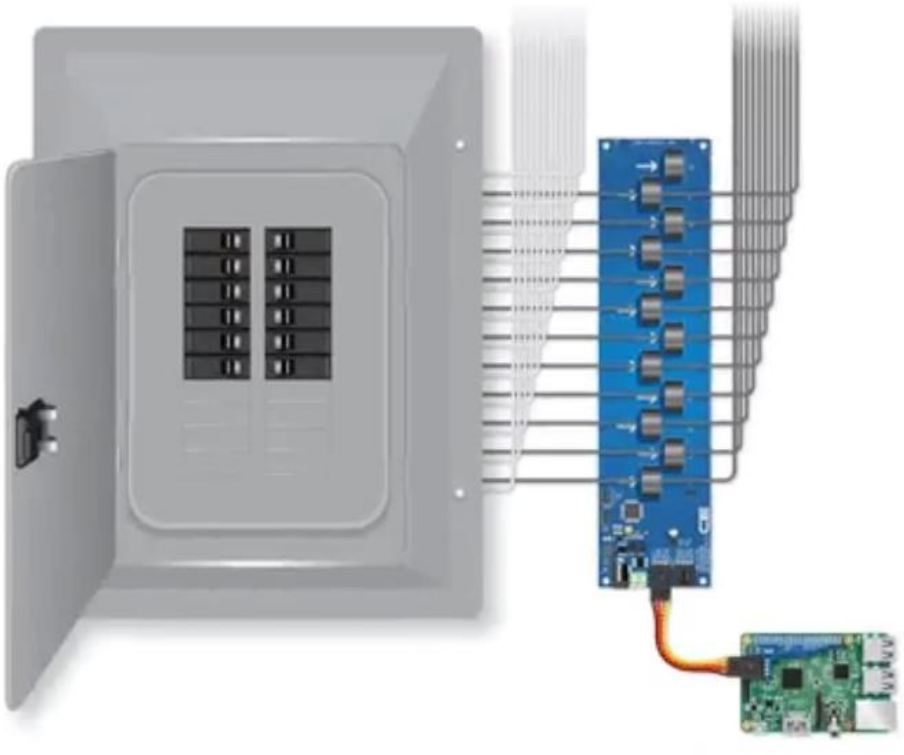


Figura 71. *Arduino con una pinza CT*

Enlace del proyecto: https://www.hackster.io/ControlEverything/energy-monitoring-through-a-raspberry-pi-190a2a

Bibliografía

[i] LoRaWan: Long Range Wide Area Network https://lorawan.es/
https://www.catsensors.com/es/lorawan/tecnologia-lora-y-lorawan

[ii] Mapa Guifi.net: http://biqfr.blogspot.com/2010/11/interconexion-de-usuarios-para-formar.html
[iii] Conexiones Oportunistas Guifi.net: https://guifi.net/trespasos
[iv] Articulo 1 alianza Guifi.net y TTNCat: https://media.guifi.net/media/guifilab-mataro-sense-fils-integracio-lorawan
[v] Articulo 2 alianza Guifi.net y TTNCat: https://femprocomuns.coop/construccion-de-la-red-abierta-de-internet-de-las-cosas-xoic-en-cataluna/?lang=es

[vi] Gateway LoraWan: https://www.thethingsnetwork.org/marketplace/product/the-things-gateway
[vii] Arquitectura TTN: https://www.thethingsnetwork.org/article/the-things-network-architecture-1
[viii] Arquitectura Sigfox: https://www.researchgate.net/figure/SigFox-Network-Architecture_fig6_329936193
[ix] Arduino: https://es.wikipedia.org/wiki/Arduino_Uno
[x] Raspberry Pi: https://es.wikipedia.org/wiki/Raspberry_Pi
[xi] NodeMCU: https://es.wikipedia.org/wiki/NodeMCU
[xii] Efecto Hall: https://es.wikipedia.org/wiki/Efecto_Hall
[xiii] Sensor de ACS712: https://www.theengineeringprojects.com/2019/03/introduction-to-acs712.html
https://www.cnx-software.com/2016/01/23/acs712-module-measures-currents-30a-1-dollar/
https://www.briandorey.com/post/esp8266-mains-energy-monitor http://electrobist.com/product/acs712-20a-range-current-sensor-module/

[xiv] Sensor de corriente HLW8012: https://tinkerman.cat/post/hlw8012-ic-new-sonoff-pow/
[xv] Sensor de corriente: https://www.theengineeringprojects.com/2019/03/introduction-to-acs712.html
[xvi] Transformador de corriente: https://www.luisllamas.es/arduino-sensor-corriente-sct-013/
[xvii] OWL Intuition: https://myecohub.com/product/owl-intutition-lc-3-phase-electricity-monitor/
[xviii] Efergy: https://es.efergy.com/emax-kit/
[xix] Schneider: https://www.se.com/us/en/product-range/1717-powerlogic-energy-meters/?parent-subcategory-id=87229&filter=business-4-low-voltage-products-and-systems

[xx] OpenEnergyMonitor: https://openenergymonitor.org/
[xxi] NAT: https://en.wikipedia.org/wiki/Network_address_translation
[xxii] Manuales de instalación:
https://github.com/openenergymonitor/EmonScripts/blob/master/install/readme.md
[xxiii] MQTT multiuser: (https://community.openenergymonitor.org/t/mqtt-to-specific-user/17445/9).
[xxiv] Internal Processing: https://learn.openenergymonitor.org/electricity-monitoring/emoncms-internals/input-processing

[xxv] Timeseries DB: https://learn.openenergymonitor.org/electricity-monitoring/timeseries/Fixed-interval
[xxvi] Timeseries: https://en.wikipedia.org/wiki/Time_series_database
[xxvii] Epoch Time: https://en.wikipedia.org/wiki/Unix_time
[xxviii] Estudio rendimento: http://openenergymonitor.blogspot.com/2013/11/improving-emoncms-performance-with_8.html

[xxix] API EmonCMS: https://emoncms.org/site/api
[xxx] MQTT multiuser: https://community.openenergymonitor.org/t/mqtt-to-specific-user/17445/3

[xxxi] Recordemos que según la documentación de EmonCMS, para enviar datos usando MQTT hay que usar el topic emon/: https://guide.openenergymonitor.org/technical/mqtt/
[xxxii] Lachacal Shop i wiki: http://lechacalshop.com/gb/ y http://lechacal.com/wiki/index.php

[xxxiii] HAT RPICT7V1: http://lechacal.com/wiki/index.php?title=RPICT7V1_Version_5
[xxxiv] EmonHub: https://guide.openenergymonitor.org/integrations/emonhub-interfacers/
[xxxv] Chip ESP: https://community.home-assistant.io/t/0-74-tuya-cloudflare-dns-push-camera-and-users-ui/60562/30
[xxxvi] Tasmota: https://tasmota.github.io/docs/

[xxxvii] Intel NUC NUC7i7BNH: https://ark.intel.com/content/www/es/es/ark/products/95065/intel-nuc-kit-nuc7i7bnh.html
[xxxviii] EmonCMS de OpenEnergyProject: https://github.com/emoncms/emoncms
https://guide.openenergymonitor.org/

[xxxix] Documentación indicando que es posible especificar el usuario en MQTT de forma nativa:
https://github.com/emoncms/emoncms/blob/master/docs/RaspberryPi/MQTT.md#node-format

[xl] Hilo donde se especifica que no funciona por ahora:
https://community.openenergymonitor.org/t/mqtt-to-specific-user/17445
[xli] Modelo: https://gist.github.com/David-Lor/63fb0be80b67359c8de6230c6b1dafa2
[xlii] EmonHub: https://github.com/openenergymonitor/emonhub
https://github.com/openenergymonitor/emonhub/blob/master/configuration.md
http://lechacal.com/wiki/index.php/Use_Emonhub_with_RPICT

[xliii] LoraWan send: https://learn.pi-supply.com/make/getting-started-with-the-raspberry-pi-lora-node-phat/

[xliv] Esptool: https://github.com/espressif/esptool
[xlv] Tuya-Convert: https://tasmota.github.io/docs/Tuya-Convert/ y https://github.com/ct-Open-Source/tuya-convert