



ugr | Universidad
de **Granada**

TRABAJO FIN DE GRADO
INGENIERÍA INFORMÁTICA

**Diseño e implementación de un sistema
de monitorización de consumo eléctrico
doméstico inalámbrico y libre.**

Autor
Miguel Moral Llamas

Tutor
Sergio Alonso Burgos



ESCUELA TÉCNICA SUPERIOR DE INGENIERÍAS INFORMÁTICA Y DE
TELECOMUNICACIÓN

—
Granada, junio de 2017



ElectroMonitor

**Diseño e implementación de
un sistema de monitorización
de consumo eléctrico
doméstico inalámbrico y
libre.**

Autor

Miguel Moral Llamas

Tutor

Sergio Alonso Burgos

Monitorización del consumo eléctrico inalámbrico libre

Miguel Moral Llamas

Palabras clave: IoT, ESP8266, monitorización energía

Resumen

El objetivo del proyecto ha sido crear un dispositivo con el menor coste posible que nos permita monitorizar los consumos de un hogar o un dispositivo en concreto.

Este dispositivo es capaz de comunicarse con un servidor alojado en una raspberry pi donde se almacena la información que se recibe del dispositivo vía WiFi. En este servidor además se puede visualizar información básica de los consumos introduciendo las fechas que se desean consultar.

Para la realización del proyecto se ha utilizado una placa nodeMCU 1.0 Amica basada en el chip ESP8266 con un sensor de corriente no invasivo conectado a la misma. Esta placa envía mediante WiFi utilizando el protocolo HTTP los datos que recoge el sensor en formato JSON. Una vez se reciben dichos datos en el servidor estos se almacenan en una base de datos MongoDB. El usuario tendrá acceso a los datos mediante una web alojada en la raspberry pi.

Free wireless energy monitor

Miguel Moral Llamas

Keywords: IoT, ESP8266, Energy monitor

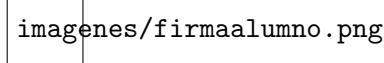
Abstract

The main idea of this project is to create a device as cheap as possible, which allows to check energy consumption from a whole house or only from a particular device.

This device can communicate with a server set up in a server(wich can be a Rapberry Pi). The server also stores all the data posted from the device using a WiFi connection. We also can check basic information about consumption in this server just introducing the specific dates we want to check.

We used nodeMCU 1.0 Amica board based on ESP8266 chip and a non invasive current sensor conected to this board. This board which is connected via WiFi and sends JSON data using HTTP protocol. We store all this data in a MongoDB data base. The user will be able to check all this data just visiting a web site hosted in the server.

Yo, **Miguel Moral Llamas**, alumno de la titulación **grado en ingeniería informática** de la **Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación** de la **Universidad de Granada**, con DNI XXXXXXXXX-X, autorizo la ubicación de la siguiente copia de mi Trabajo Fin de Grado en la biblioteca del centro para que pueda ser consultada por las personas que lo deseen.



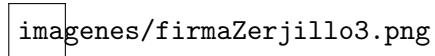
Granada a 7 de septiembre de 2017 .

D. **Sergio Alonso Burgos**, Profesor del Área de **Lenguajes y Sistemas Informáticos** del Departamento **Lenguajes y Sistemas Informáticos** de la Universidad de Granada.

Informa:

Que el presente trabajo, titulado ***Diseño e implementación de un sistema de monitorización de consumo eléctrico doméstico inalámbrico y libre***, ha sido realizado bajo su supervisión por **Miguel Moral Llamas**, y autorizo la defensa de dicho trabajo ante el tribunal que corresponda.

Y para que conste, expiden y firman el presente informe en Granada a 7 de Septiembre de 2017 .



Sergio Alonso Burgos

Agradecimientos

A todas aquellas personas que me han ayudado en esta etapa de mi vida. En especial a mi familia y amigos por apoyarme en los malos momentos y a mi tutor D. Sergio Alonso Burgos por su tiempo y dedicación durante la realización de este proyecto.

Índice general

1. Introducción y motivación	19
1.1. Software libre	20
1.2. Hardware principal utilizado en el proyecto	21
1.2.1. ESP8266	21
1.2.2. Raspberry Pi	22
1.3. Sensores de energía en el mercado actual.	24
1.4. Tabla comparativa de sensores de energía en el mercado.	32
1.5. Porqué hacer nuestro propio monitor	34
2. Objetivos	37
2.1. Diagrama objetivos	38
2.2. Conocimientos necesarios para alcanzar estos objetivos.	40
3. Planificación y presupuesto	41
3.1. Planificación del trabajo	41
3.2. Presupuesto del proyecto	44
4. Módulo hardware	45
4.1. Requisitos	45
4.1.1. Requisitos funcionales	45
4.1.2. Requisitos no funcionales	45
4.2. Planificación	46
4.3. Arquitectura del sistema	47
4.3.1. Módulo principal	47
4.3.2. Módulo sensor	47
4.3.3. Módulo conversor analógico digital	48
4.3.4. Módulo pantalla	49
4.4. Coste unitario	50
4.5. Prototipos	51
4.6. Versiones finales	54
4.7. Montaje de la caja	56

5. Módulo firmware	59
5.1. Requisitos	59
5.1.1. Requisitos funcionales	59
5.1.2. Requisitos no funcionales	59
5.2. Planificación	60
5.3. Arquitectura del firmware	61
5.3.1. Módulo de lectura de sensor y cálculo de irms y potencia	61
5.3.2. Módulo para mostrar información en la pantalla LCD	63
5.3.3. Módulo envío de información al servidor	63
6. Módulo servidor	65
6.1. Requisitos	65
6.1.1. Requisitos funcionales	65
6.1.2. Requisitos no funcionales	65
6.2. Backend	66
6.3. Frontend	66
6.4. Servidor	66
6.4.1. Sistema Operativo	67
6.4.2. Aprovisionamiento	67
6.4.3. Iniciar aplicación web	67
6.4.4. IP estática Raspberry	69
6.4.5. Aspecto final de la página	70
7. Pruebas	73
8. Experimentación	81
8.1. Comparativas con aparatos reales	81
8.2. Prueba consumo constante	82
8.3. Prueba con un electrodoméstico real	82
8.4. Posibilidades de explotación de un dispositivo como el presentado	85
9. Conclusiones y trabajos futuros	87
9.1. Trabajos futuros	88
A. Anexo	89
A.1. Como flashear firmware en el ESP8266	89
Bibliografía	94

Índice de figuras

1.1.	Resumen de la licencia GNU GPL v3	20
1.2.	NodeMCU.	22
1.3.	Raspberry.	24
1.4.	Efergy ENGAGE HUB 1.1	25
1.5.	Neurio W1-HEM energy monitor	26
1.6.	Efergy Elite Classic.	27
1.7.	Floureon Power Meter Energy Monitor.	28
1.8.	Belkin Conserve Insight.	29
1.9.	EmonTx V3.	30
1.10.	Sense.	31
1.11.	British Gas.	32
2.1.	Diagrama de objetivos	39
3.1.	Diagrama de Gantt	43
4.1.	YHDC.	48
4.2.	ADS1115.	49
4.3.	LCD 16x02 y módulo I2C	50
4.4.	Esquema v1	52
4.5.	Esquema v2	52
4.6.	Esquema v3	53
4.7.	Esquema v4	54
4.8.	Esquema PCB versión 1	55
4.9.	PCB soldada versión 1	55
4.10.	Esquema PCB versión 2	56
4.11.	PCB soldada versión 2	56
4.12.	Caja versión 1	57
4.13.	Caja versión 2	58
5.1.	Representación datos obtenidos del sensor.	62
5.2.	Ley de Ohm y Watt.	63
6.1.	rc.local Raspberry Pi	69

6.2. Configuración /etc/network/interfaces	70
6.3. Página principal	70
6.4. Tabla de datos	71
6.5. Gráfica mensual	71
6.6. Gráfica ultimos consumos	72
8.1. Monitor serial Arduino IDE.	82
8.2. Gráfico muestras recogidas.	82
8.3. Muestras recogidas congelador.	83
8.4. Muestras recogidas congelador.	84
A.1. Archivo/Preferencias	90
A.2. Herramientas/Placa/Gestor de tarjetas	90

Lista de Tablas

1.1.	Monitores de energía en el mercado.	33
4.1.	Listado de precios.	51
7.1.	Prueba 1.	73
7.2.	Prueba 2.	74
7.3.	Prueba 2.	75
7.4.	Prueba 4.	76
7.5.	Prueba 5.	77
7.6.	Prueba 6.	78
7.7.	Prueba 7.	79
7.8.	Prueba 8.	80
7.9.	Prueba 9.	80
8.1.	Comparación consumos.	81
8.2.	Tabla datos	84

Capítulo 1

Introducción y motivación

A medida que va avanzando la tecnología van apareciendo nuevos aparatos eléctricos en nuestros hogares con el gasto eléctrico que implica tener estos aparatos conectados a la electricidad. Si tenemos un control sobre el consumo eléctrico que tenemos en casa en todo momento nos permitirá recortar consumos innecesarios, reduciendo de esta forma la factura de luz y además contribuimos a la reducción de la emisión de gases contaminantes como el CO_2 provenientes de generar la electricidad.

A día de hoy la mayoría de hogares disponen de una conexión a Internet, permitiendo de esta forma mandar los datos recogidos por el sensor a un servidor ya sea local o no, de una manera cómoda y sencilla. Cada vez más dispositivos cuentan con conexión a internet lo que permite que dispositivos que no imaginariamos que se pudieran comunicar como pueden ser una lavadora, un frigorífico,... tengan la capacidad de mandar o recibir datos haciendo uso de esa conexión a internet.

Actualmente en el mercado existen algunas soluciones comerciales que permiten consultar el consumo eléctrico en todo momento. Uno de los problemas de la mayoría de estas soluciones es su elevado precio. Además estos dispositivos al estar fabricados por empresas las cuales no van a liberar el código fuente ni a especificar como funciona su aparato, no nos permiten modificar a nuestro antojo que valores queremos que se muestren, donde queremos mandar esta información, etc.

Otro problema que tienen algunas de las soluciones actuales que hay en el mercado es que mandan los datos a servidores privados de su propiedad. Esto implica tanto problemas de privacidad ya que la empresa no nos informa de los protocolos de seguridad que utilizan para proteger sus servidores ni que hacen con nuestros datos. Esta empresa podría vender nuestros datos, o una persona ajena a la empresa podría encontrar una debilidad en la web y conseguir todos nuestros datos. Hemos de tener en cuenta que gracias a los consumos eléctricos una persona podría saber perfectamente a que hora estamos en casa, a que hora la casa se encuentra vacía, si nos hemos

ido de vacaciones y la casa esta sola. Sin olvidarnos que si algún día esta empresa quiebra o simplemente con el tiempo decide dejar de dar soporte web y cierra estos servidores tendremos un aparato totalmente inservible ya que al no poder modificarlo no tendremos la posibilidad de mandar esta información a otro servidor o a un servidor que montemos nosotros mismos.

Pero ¿por qué es interesante monitorizar la energía que consumimos en nuestro hogar y porque podría resultar interesante? La idea principal para llevar a cabo la monitorización del consumo eléctrico de un hogar es reducir en la factura de la luz. Si somos conscientes de cuanto consume cada aparato eléctrico durante el tiempo que esta en funcionamiento y tener un seguimiento de estos consumos podremos ver de que manera podremos reducir los consumos. Los monitores de consumo por si solos no van a hacer que nuestro consumo disminuya, pero si que van a permitir que nos demos cuenta que tan solo cambiando algunos de nuestros hábitos podremos ahorrar bastante dinero a lo largo del año.

Con este proyecto se pretende además aprender algunas de las competencias que no se cursan en la rama específica que he estudiado como pueden ser competencias relacionadas con la rama de hardware o en general sobre nuevas tecnologías como el IoT.

Además puede servir como un pequeño experimento de lo que puede averiguar una empresa que esté interesada en este servicio. Pudiendo de esta forma obtener datos sobre consumos eléctricos para distintos usos como pueden ser conocer los hábitos de consumo eléctrico de una casa, etc.

1.1. Software libre

Este proyecto se encontrará publicado en su totalidad en GitHub <https://github.com/Miguelmora/ESP8266-energy-monitor> bajo la licencia GNU GPL v3 <https://www.gnu.org/licenses/gpl-3.0.html> [20]. Como podemos ver 1.1 los permisos que conceden a la persona que adquiera este software son los de un uso comercial, la modificación, la distribución, el uso de patentes y el uso privado siempre y cuando se cumplan las condiciones de el aviso de licencia y copyright, los cambios realizados en el código han de ser notificados, cuando el código se vuelva a distribuir este ha de ser público con la misma licencia GNU GPL con la que el autor de dicho software lo publicó.

Miguelmora/ESP8266-energy-monitor is licensed under the GNU General Public License v3.0	Permissions	Limitations	Conditions
Permissions of this strong copyleft license are conditioned on making available complete source code of licensed works and modifications, which include larger works using a licensed work, under the same license. Copyright and license notices must be preserved. Contributors provide an express grant of patent rights.	<ul style="list-style-type: none"> ✓ Commercial use ✓ Modification ✓ Distribution ✓ Patent use ✓ Private use 	<ul style="list-style-type: none"> ✗ Liability ✗ Warranty 	<ul style="list-style-type: none"> ⓘ License and copyright notice ⓘ State changes ⓘ Disclose source ⓘ Same license

Figura 1.1: Resumen de la licencia GNU GPL v3 Fuente: [19]

Todas las bibliotecas utilizadas para este proyecto tienen una licencia de software libre.

1.2. Hardware principal utilizado en el proyecto

En esta sección se hablará sobre el Hardware principal utilizado para realizar este proyecto.

1.2.1. ESP8266

Para la realización de este proyecto se ha decidido utilizar una Placa ESP8266 frente a otras soluciones más habituales como suele ser Arduino.

Uno de los problemas del Arduino es que carece de conexión a internet. Si necesitamos dotar a nuestro proyecto de conexión WIFI tendremos que utilizar un chip para proporcionar a nuestro Arduino de dicha conexión. Aquí es donde entra en juego el chip ESP8266 creado con la idea de ser utilizado en proyectos de IoT y poder dotar al Arduino de conexión WiFi.

En el mercado existen diferentes fabricantes de placas que integran el chip ESP8266, pero para este proyecto se va a utilizar la placa NodeMCU v2.

En 2014 salta al mercado el kit de desarrollo de código libre NodeMCU. Inicialmente se trataba únicamente de un firmware en lenguaje LUA. Pero dos meses después el proyecto se amplia y desarrollan una placa de hardware libre. Se trata de la primera versión conocida como v0.9 o V1. Estas placas cuentan con una memoria de 4 KBytes y un almacenamiento de 4 MBytes [42]. Existen actualmente tres versiones diferentes de esta placa. Todas estas versiones son muy similares de la primera a la segunda generación tan solo cambia la anchura de la placa ya que con la placa v1 se ocupaban los 10 pines que traen las protoboard de tamaño standard, la v2 soluciona este problema haciendo la placa un poco más estrecha dejando una fila de pines en cada lado de placa cuando colocamos esta en una protoboard, además de substituir el chip ESP12 para pasar a montar el nuevo chip mejorado ESP12E. Al tratarse de hardware libre cualquier compañía interesada puede fabricar las placas, en el caso de estas dos primeras versiones han sido producidas por la empresa Amica, sin embargo la v3 la ha sacado al mercado la empresa Lolin. Esta nueva versión v3 no aporta cambios sustanciales con respecto a la v1 y la v2, tan solo un puerto USB más robusto y ha utilizado uno de los pines reservados para alimentación USB para proporcionar un pin adicional de GND [27] .

Principalmente las ventajas que nos ofrece esta placa son su bajo precio, su simplicidad tanto a la hora de programar como a la hora de realizar las conexiones, la posibilidad de soportar distintos lenguajes de programación como son (LUA, MicrPython o Arduino) y sobretodo su conexión WiFi.

En la imagen (1.2) podemos ver todas las conexiones de las que dispone la versión v2 de la placa NodeMCU.

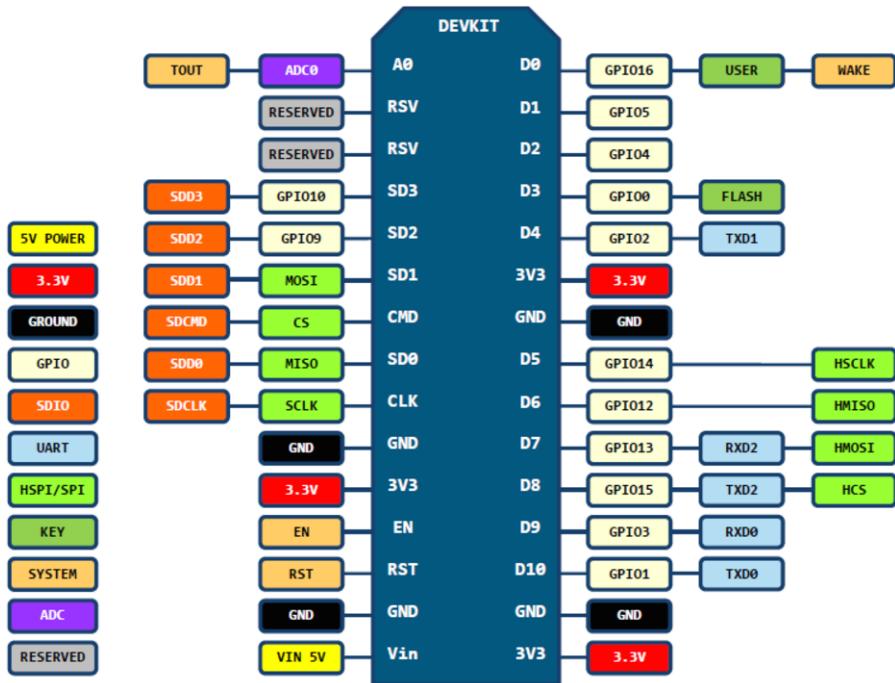


Figura 1.2: NodeMCU. Fuente: [27]

1.2.2. Raspberry Pi

Se trata de una pequeña placa base que monta un procesador del fabricante Broadcom, en concreto el modelo BCM2835. Se trata de un procesador con arquitectura ARM a 1GHz de velocidad, además cuenta con una GPU VideoCore IV y una memoria de 512 MB, el almacenamiento del dispositivo dependerá de la tarjeta SD que el cliente introduzca en la placa aunque no es muy recomendable introducir SD de gran capacidad ya que las SD soportan un número limitado de lecturas y escrituras, por lo que al estar manejando un sistema operativo estas SD no suelen tener un tiempo de vida muy largo debido a la cantidad de lecturas y escrituras que tienen que realizar. Además con lo que al almacenamiento se refiere la Raspberry soporta tanto discos duros externos como pendrives por lo que el almacenamiento en este dispositivo no es ningún problema [43].

Con respecto al software podemos escoger entre una gran cantidad de sistemas operativos los cuales están disponibles para esta placa. Los más populares están basados en GNU/Linux como pueden ser Raspbian tanto su versión Lite sin entorno gráfico más dedicada al uso como servidor o su versión de escritorio una versión específica para raspberry derivada de

Debian. Estos sistemas operativos se pueden descargar completamente gratis de la web oficial [33]. Una vez descargados tan solo tendremos que flashearlos en una tarjeta SD y tendremos el sistema operativo elegido funcionando sin ningún tipo de problema en nuestra raspberry. Existen además otros sistemas operativos para esta placa permitiendo utilizarla no solo como un pequeño ordenador o como un servidor, gracias a sistemas operativos como OpenELEC podremos utilizar este dispositivo como un centro multimedia, incluso existen sistemas operativos para emular videojuegos.

Además de su reducido precio para la gran potencia que nos ofrece, una de las mayores virtudes de esta placa es el gran número de conexiones que nos ofrece. Esta placa cuenta con las siguientes conexiones (modelo Pi 2) : [43]

- **Puertos USB:** Dispone de 4 puertos USB 2.0.
- **Salidas de video:** Conector RCA, HDMI e interfaz DSI para conectar un panel LCD.
- **Salidas de audio:** Cuenta con un conector Jack de 3,5mm y una salida HDMI.
- **Conectividad de red:** Dispone de una entrada Ethernet (RJ-45).
- **Alimentación:** Integra una entrada microUSB de 5V para poder alimentar la placa de manera sencilla.

En la imagen (1.3) podremos ver todas las conexiones de manera más clara.

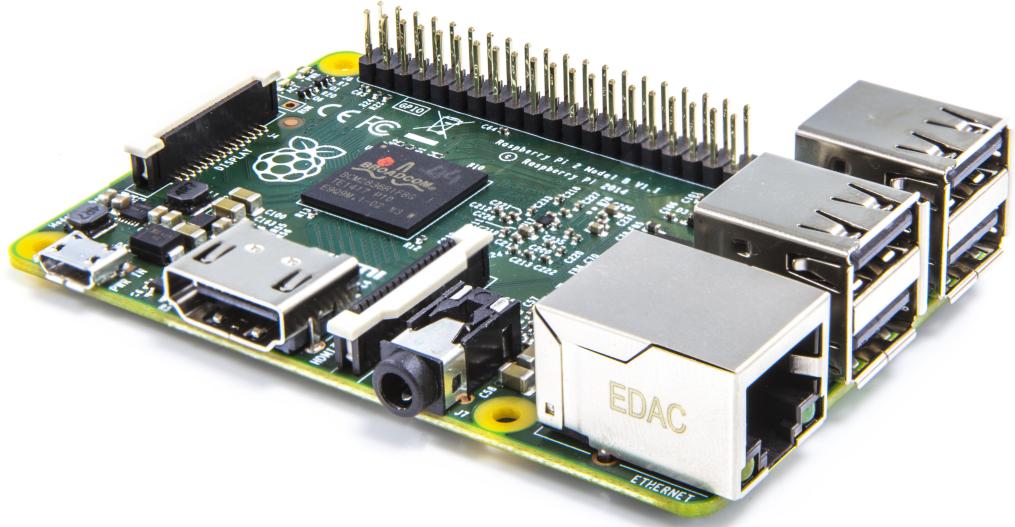


Figura 1.3: Raspberry. Fuente: [32]

1.3. Sensores de energía en el mercado actual.

Antes de empezar el proyecto es necesario analizar las soluciones que nos ofrece el mercado actualmente. A continuación vamos a repasar los puntos fuertes y los puntos débiles de estas soluciones.

- **Efergy Technologies ENGAGE HUB 1.1:** Se trata de un sensor que se conecta de forma inalámbrica con sus propios servidores donde se almacenará la información que se recoja en nuestro hogar. Pone a disposición del cliente un portal web gratuito que le permitirá consultar los datos recogidos, cuenta además con una aplicación para android y para IOS. Este dispositivo permite conectar una pantalla externa de la propia marca la cuál no está incluida en el precio. A un precio de 112,08 \$ (99,9€).[14]



Figura 1.4: Efergy ENGAGE HUB 1.1. Fuente: [14]

- **Neurio W1-HEM Home Energy Monitor:** Ofrece actualización de los datos segundo a segundo. Infraestructura basada en la nube permitiendo acceder a los datos desde cualquier lugar y aplicaciones web y para dispositivos android e IOS. Una de las ventajas de este dispositivo es que cuenta con una API pública que nos permitirá incorporar los datos que recojamos de forma sencilla. Precio en Amazon EEUU 219,99 \$. [29]



Figura 1.5: Neurio W1-HEM energy monitor. Fuente: [29]

- **Efergy Technologies Elite Classic:** Es el modelo más básico de la compañía Efergy. Cuenta con una pequeña pantalla en blanco y negro que permite consultar los datos que recoge del sensor. La información que se recoge del sensor se actualiza cada 10 segundos por defecto pudiendo configurarlo para que se produzca cada 15 o 20 segundos. Este modelo no permite enviar datos a la nube, a no ser que compremos el accesorio engage. Precio de 61,50 \$ (54,90 €).[13]



Figura 1.6: Efergy Elite Classic. Fuente: [13]

- **Floureon Power Meter Energy Monitor US TS-836A:** Se trata del modelo más básico de todos los que estamos analizando, siendo además el más económico de todos. Este dispositivo no se puede considerar directamente como IoT ya que tan solo muestra sus datos mediante una pantalla incorporada en el dispositivo, pero no existe la posibilidad de transmitir estos datos. Otro inconveniente de este dispositivo es que no permite monitorizar toda la casa ya que este dispositivo ha de conectarse en un enchufe. Su precio es de 20 \$ (17,80 €) más envío.[17]



Figura 1.7: Floureon Power Meter Energy Monitor. Fuente: [17]

- **Belkin Conserve Insight:** Se trata de un dispositivo muy similar al Floureon. La principal diferencia la encontramos en que el dispositivo de la marca Belkin nos muestra además la cantidad de dióxido de carbono que producimos según el consumo eléctrico que esté midiendo el aparato, aunque realmente se trata de una simple multiplicación. Cabe mencionar que algunos usuarios informan que la calidad de este aparato no es buena, llegando algunos usuarios incluso a comentar que el dispositivo se les ha quemado con el consiguiente riesgo de incendio. Al igual que pasaba con el dispositivo de la marca Floureon no se nos ofrece la posibilidad de almacenar los datos que recoge el sensor en ningún servidor ya que el aparato no dispone de conectividad para poder comunicarse con el servidor. El precio de este aparato es de 29,99 \$ (26,70 €). [7]



Figura 1.8: Belkin Conserve Insight. Fuente: [7]

- **EmonTx V3:** La propuesta opensource que nos trae OpenEnergy-Monitor nos permitirá conectar un total de 4 sensores de energía no invasivos. Conectividad vía RF 433/868 MHz o posibilidad de dotar al aparato con conectividad WiFi utilizando una placa ESP8266. Incorpora la opción de hacer medidas de temperatura además de hacer una monitorización de energía. Permite enviar los datos recogidos por el sensor mediante RF 433/868 MHz a una raspberry o a cualquier servidor en caso de utilizar la conexión WiFi del ESP8266. A un precio de 65 €.[15]



Figura 1.9: EmonTx V3. Fuente: [15]

- **Sense home energy system:** Este sensor tiene como idea que nuestra casa se comunique con nosotros. De modo que la casa nos dirá durante cuanto tiempo hemos estado viendo la televisión, o durante cuanto tiempo ha estado encendida una bombilla. Este sistema no tiene que aprender nuestros hábitos durante un periodo de tiempo para reconocer que aparato está consumiendo electricidad en cada momento si no que ya conoce el modelo de consumo eléctrico de ciertos electrodomésticos y aparatos eléctricos de la casa. Este aparato se comunica mediante WiFi para mandar la información que recogen sus sensores a un servidor de la propia empresa. Podremos visualizar nuestros consumos y demás datos a través de un portal web o mediante una app disponible para Android e IOS. El precio de este producto ronda los 265 €.[36]

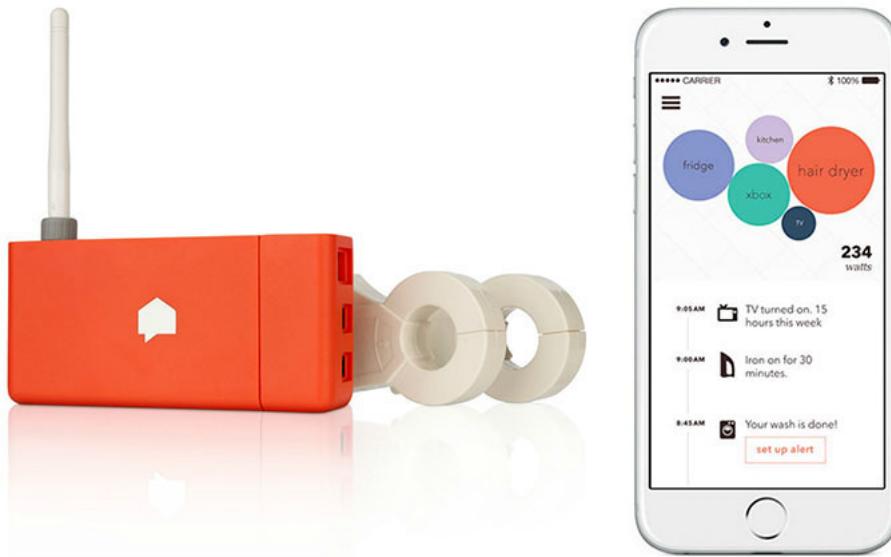


Figura 1.10: Sense. Fuente: [35]

- **British Gas energy monitor:** Debido a una iniciativa por parte del gobierno de gran Bretaña que busca reducir el consumo eléctrico de los hogares la compañía de luz y gas británica British Gas podrá pedir un monitor de energía y gas de forma gratuita incluyendo su instalación. Los datos recogidos por el sensor se mandan al monitor utilizando Zigbee y estos datos se mandan directamente vía GPRS a la compañía British Gas haciendo que no se tengan que revisar los consumos de cada casa ya que se actualizan automáticamente. Actualmente este dispositivo tan solo se puede conseguir siendo cliente de la compañía British Gas. [9]

32 1.4. Tabla comparativa de sensores de energía en el mercado.



Figura 1.11: British Gas. Fuente: [9]

1.4. Tabla comparativa de sensores de energía en el mercado.

Mediante la tabla (1.1) se pretende resumir de forma más clara y sencilla toda la información dada en la sección anterior de cada uno de los dispositivos que se nombraron.

Modelo	Dispone API	Tiempo refresco	Protocolo Comunicación	Monitorización solar	Pantalla para visualizar datos	Cloud	Precio €
Efergy Engage	✓	Selezionable 10, 15 o 20 sec	Wifi	Con accesorio	Con módulo Elite classic	✓	99.9
Efergy elite	✓	Selezionable 10, 15 o 20 sec	Sin especificar	Con accesorio	Con módulo Engage	✓	195.86
Neurio	✓	Instantáneo	Wifi 802.1 b/n/g	Con accesorio	✓	✓	54.90
Floureon	✗	Instantáneo	No transmite datos	✗	✗	✗	17.80
Belkin	✗	Instantáneo	No transmite datos	✗	✗	✗	26.70
EmonTx V3	✗	Instantáneo	RF 433/868 MHz	✗	✓	✓	65
Sense	✗	Instantáneo	Wifi, Bluetooth	Con accesorio	✗	✓	265
British Gas	✗	Instantáneo	GPRS, Zigbee	✗	✓	✓	-

Tabla 1.1: Monitores de energía en el mercado.

Teniendo en cuenta las diferencias entre modelos y los precios de cada aparato existen dos alternativas principales en función de la necesidad del cliente las mejores opciones serían :

- **Monitorizar el consumo de un solo electrodoméstico:** En caso de que el cliente necesite un dispositivo sencillo al menor precio posible, recomendaría el dispositivo de la marca Belkin (Conserve Insight). Si lo comparamos frente a su competidor el Floureon la diferencia de precio es de tan solo unos 9 euros permitiéndonos el dispositivo de la marca Belkin visualizar la cantidad de dióxido de carbono que producimos. Por lo demás ambos dispositivos son muy similares.
- **Monitorizar el consumo de toda la casa:** Como podemos observar en el caso de querer monitorizar la casa entera el precio de los dispositivos se dispara. En mi opinión el mejor dispositivo de los que se han presentado es el Eergy Technologies Elite Classic, además de ser el más económico de los tres modelos que permiten monitorizar el consumo de toda la casa podremos actualizar el dispositivo y comprar el módulo Engage para almacenar nuestros datos en los servidores que nos brinda la empresa Eergy y poder visualizar nuestros consumos desde su plataforma web o sus aplicaciones para dispositivos móviles. No considero que sea rentable pagar la diferencia de precio existente con el modelo que nos ofrece la compañía Neurio ya que no existen diferencias que justifiquen este sobreprecio. Sin embargo la característica que nos trae Sense que nos permite comunicarnos con nuestra casa viendo consumos exclusivos de cada aparato me parece muy interesante aunque en mi opinión no justifica el alto precio del producto. El EmonTx no me parece mala opción pero es necesario documentarse un poco para poder empezar a utilizarlo si no se tiene experiencia.

1.5. Porqué hacer nuestro propio monitor

Como podemos observar en el mercado existen diversas propuestas que se pueden adaptar a las necesidades que buscamos cubrir, pero en este proyecto se va a llevar a cabo el diseño y construcción de un monitor con características similares a los que existen. Algunos de los motivos que nos han llevado a realizar nuestro propio monitor son los siguientes:

- **Aprender:** Durante todo el proceso se van a adquirir conocimientos en diversas áreas como pueden ser hardware, electrónica, etc.
- **Precios:** Podremos además comprobar si el precio de realizar nuestro propio dispositivo es razonable con respecto a los precios que nos ofrecen aparatos comerciales.

- **No usar servicios externos:** A la hora de enviar los datos recogidos con nuestro dispositivo tendremos total libertad de escoger en que servidor se van a almacenar estos, pudiendo ser tanto un servidor local como cualquier otro servidor.

Capítulo 2

Objetivos

En este proyecto el objetivo principal **OBJ-G** es desarrollar un dispositivo libre que permita monitorizar el consumo eléctrico tanto del hogar como de un dispositivo específico y que permita transmitir estos datos de forma inalámbrica, todo ello al menor precio posible. Podemos dividir este objetivo principal en los siguientes objetivos principales:

- **OBJ-P1** Diseño hardware con una placa que permita la conexión inalámbrica WiFi para poder enviar los datos recogidos de un sensor no invasivo.
- **OBJ-P2** Implementación del firmware que permite capturar los datos que recoge el sensor y enviar estos datos vía WiFi.
- **OBJ-P3** Creación de un servidor web para poder almacenar los datos recibidos en una base de datos y que permitirá al usuario revisar consumos.
- **OBJ-P4** Documentación de todo el proyecto.

Una vez tenemos todos los objetivos principales definidos vamos a dividirlos en objetivos más específicos. Dentro de estos subobjetivos existirán algunos de ellos obligatorios para que la funcionalidad del proyecto sea completa y otros opcionales que servirán para perfeccionar el proyecto, pero estos no serán vitales para su correcto funcionamiento:

- **OBJ-P1-OB1** Investigación de sensores de corriente no invasivos existentes en el mercado y como conectarlos a distintas placas.
- **OBJ-P1-OB2** Investigación para seleccionar la placa que mejor se adapte a este proyecto que cuente con conexión WiFi.
- **OBJ-P1-OB3** Montaje del primer prototipo con los componentes seleccionados anteriormente.

- **OBJ-P1-OP1** Mejora de este primer prototipo para añadir funcionalidades al proyecto.
- **OBJ-P1-OP2** Realizar todo el montaje de los componentes en un PCB para que el dispositivo tenga un buen acabado.
- **OBJ-P2-OB1** Decidir en que lenguaje programar la placa escogida y que IDE utilizar.
- **OBJ-P2-OB2** Implementar un primer firmware para asegurar que el hardware funciona adecuadamente.
- **OBJ-P2-OB3** Decidir en que formato y como se van a enviar las mediciones recogidas por la placa al servidor.
- **OBJ-P2-OP1** Mejorar el firmware para enviar más parámetros al servidor.
- **OBJ-P3-OB1** Selección de framework para crear la aplicación web.
- **OBJ-P3-OB2** Seleccionar el sistema de base de datos que vamos a utilizar para almacenar los datos recibidos por la placa.
- **OBJ-P3-OB3** Decidir donde alojar la base de datos y la aplicación web.
- **OBJ-P3-OB4** Implementar la aplicación web.
- **OBJ-P3-OP1** Mejorar la aplicación web para mostrar más información al usuario.
- **OBJ-P4-OB1** Subir a un repositorio de GitHub todo lo necesario para que las personas que estén interesadas puedan utilizar este proyecto por su cuenta de manera sencilla.
- **OBJ-P4-OB1** Realizar toda la documentación del proyecto en Latex siguiendo la plantilla ofrecida por la UGR.
- **OBJ-P4-OP1** Hacer un estudio de los consumos de algún electrodoméstico.

2.1. Diagrama objetivos

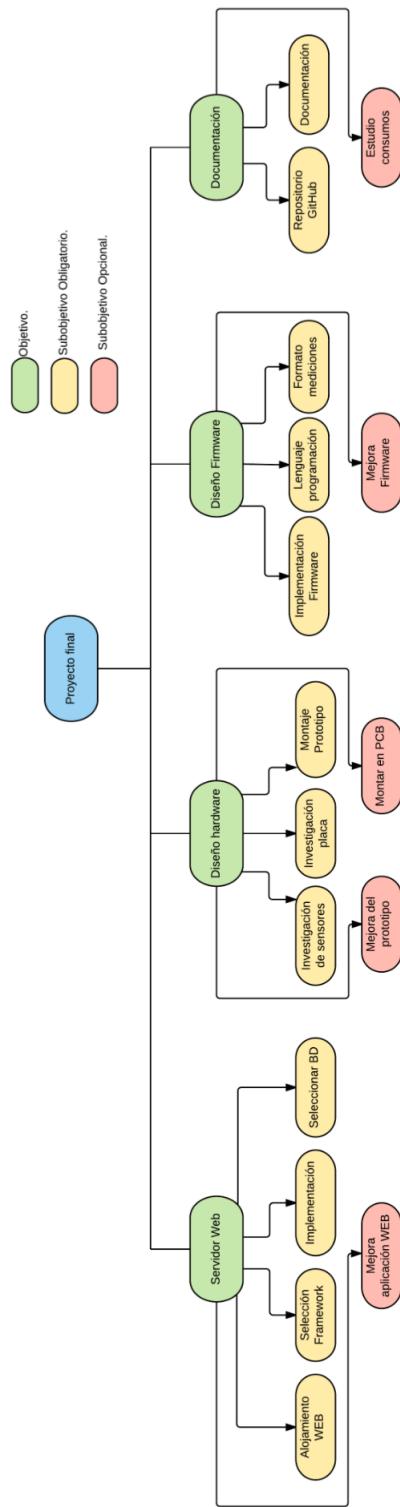


Figura 2.1: Diagrama de objetivos

2.2. Conocimientos necesarios para alcanzar estos objetivos.

Algunos de estos conocimientos han sido adquiridos tras cursar las asignaturas que se indicarán a continuación y otros de los conocimientos han sido adquiridos por mi propia cuenta al tener que realizar algunos de los objetivos de este proyecto:

- **Conocimientos adquiridos mediante asignaturas de la carrera:**
 - **Desarrollo de aplicaciones para internet:** Los conocimientos adquiridos tras cursar esta asignatura me han servido para poder realizar la web utilizando flask, además de conocer el lenguaje de programación python y las bases de datos NoSQL como MongoDB.
 - **Infraestructura virtual:** Se han puesto en uso los conocimientos a la hora de utilizar git y GitHub.
 - **Fundamentos de software:** Entre el temario de esta asignatura se encuentra una parte dedicada a la realización de scripts y algunos comandos básicos de Linux que se han puesto en práctica para la realización de este proyecto.
 - **Fundamentos de programación:** Donde se ven conceptos básicos sobre programación en c++ que en cierto modo han servido como base para realizar toda la parte de programación tanto web como a la hora de programar la placa.
- **Conocimientos adquiridos gracias a la realización de este proyecto:**
 - **LATEX:** He tenido que aprender Latex para poder realizar toda la documentación de este proyecto.
 - **Arduino IDE:** A la hora de programar la placa he tenido que aprender como usar este IDE.
 - **Electrónica básica:** Para poder llevar a cabo el montaje de todos los circuitos.
 - **Raspberry Pi:** He necesitado adquirir conocimientos básicos para poder montar un servidor en una RaspberryPi.
 - **Soldadura:** Conocimientos básicos de soldadura que nos permiten soldar pistas de estaño en la PCB para poder materializar los prototipos presentados en la protoboard en una placa real.
 - **Electricidad:** Necesario conocer un mínimo para poder calcular el Irms y la potencia ya que no tenía ningún tipo de conocimiento en este campo antes de realizar el proyecto.

Capítulo 3

Planificación y presupuesto

La metodología de trabajo utilizada a la hora de desarrollar este proyecto está basada en la creación y prueba de prototipos de manera iterativa, ya que al tratarse de un proyecto de desarrollo de hardware principalmente se ha considerado que esta es la mejor manera de comprobar si el prototipo funciona como se esperaba o existe algún tipo de fallo el cuál puede ser arreglado en el siguiente prototipo. Además dados los escasos conocimientos iniciales que se tenían en electrónica y electricidad este esquema de desarrollo ha permitido aprender de manera más gradual y práctica. Esta metodología de trabajo se verá reflejada en la planificación propuesta inicialmente que se especificará a continuación así como una estimación del coste total del proyecto.

3.1. Planificación del trabajo

Esta planificación fue realizada cuando aún no se había empezado a trabajar en el proyecto por lo tanto algunos de estos plazos finalmente no se han cumplido tal y como se esperaba.

▪ Módulo hardware:

- Reunión con la persona interesada en el proyecto 16/02/2017 - 16/02/2017.
- Investigación de todas las alternativas existentes para llevar a cabo esta parte del proyecto 16/02/2017- 22/02/2017.
- Adquirir todos los componentes necesarios para esta parte comparando los precios y el producto en si que ofrecía cada tienda 22/02/2017 - 23/02/2017.
- Diseño de prototipos 23/02/2017 - 17/03/2017.
- Montaje de los prototipos diseñados con realización de pruebas para comprobar su correcto funcionamiento 1/03/2017 - 14/04/2017.

- Diseño de las placas de las versiones finales 14/04/2017 - 21/04/2017.
- Soldar las versiones finales en una PCB 21/04/2017 - 28/04/2017.
- Probar las versiones finales 28/04/2017 - 03/05/2017.
- Diseño de cajas donde irá colocada la pcb y el resto de componentes de la versión final 03/05/2017 - 04/05/2017.
- Montaje de las cajas 04/05/2017 - 05/05/2017.

■ Módulo firmware:

- Reunión con la persona interesada en el proyecto 16/02/2017 - 16/02/2017.
- Investigación de todas las alternativas existentes para llevar a cabo esta parte del proyecto 16/02/2017- 22/02/2017.
- Desarrollo de toda la parte firmware que se subirá a la placa ESP8266 1/03/2017 - 02/06/2017.
- Pruebas 02/06/2017 - 09/06/2017.

■ Parte servidor:

- Reunión con la persona interesada en el proyecto 09/06/2017 - 09/06/2017.
- Investigación de todas las alternativas existentes para llevar a cabo esta parte del proyecto 09/06/2017 - 16/06/2017.
- Realización de una aplicación web para el proyecto 16/06/2017 - 30/06/2017.
- Alojar dicha aplicación web en un servidor (RaspberryPi) 19/06/2017 - 06/07/2017.
- Pruebas 01/07/2017 - 10/07/2017.

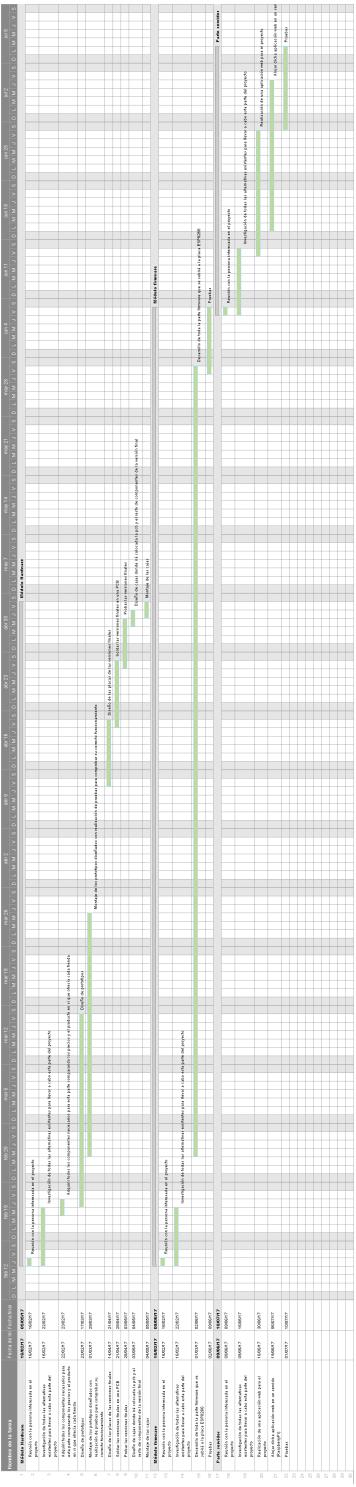


Figura 3.1: Diagrama de Gantt

3.2. Presupuesto del proyecto

A continuación se especificará un presupuesto con los costes del proyecto:

- **Hardware y componentes electrónicos :** En el apartado número 4 se especifica detalladamente el coste de todo el hardware adquirido el cuál se sitúa en los **40.47 €**, a este tenemos que sumarle otros 36.76 € ya que algunos de los componentes se compraron más de una vez ya que resultaron dañados mientras se realizaron las pruebas. Obteniendo por lo tanto un coste final de **77.23 €**
- **Parte servidor :** En esta parte el único coste ha sido la adquisición de una Raspberry Pi ya que el sistema operativo utilizado para la Raspberry tiene una licencia gratuita. Las herramientas de desarrollo utilizadas para esta parte son gratuitas como el editor de textos Atom [5]. El precio de la Raspberry está en torno a los **39.90 €**.
- **Parte firmware :** Para el desarrollo de esta parte se ha utilizado únicamente software libre en concreto Arduino IDE [4], además de bibliotecas con licencia de software libre, por lo tanto el coste ha sido **0 €**.
- **Horas de trabajo:** El proyecto en total ha tenido una duración total de 7 meses. Para contar las horas de trabajo vamos a suponer que los fines de semana no se ha trabajado por lo tanto si se cuentan los días laborables desde el 16/02/2017 hasta el 2/09/2017 obtenemos un total de 137 días laborables. Trabajando durante una media de 2 horas diarias para poder compatibilizar estudios y la realización de este proyecto obtenemos un total de 274 horas dedicadas a este proyecto. Suponiendo que cada hora se cobre a 12.5 € obtenemos un total de **3425 €**.
- **Coste del ordenador utilizado para el desarrollo de este proyecto :** Para calcular este coste calcularemos la depreciación [10] de este producto partiendo de que el precio base son 800 €, calculamos que el ordenador tendrá aproximadamente una vida útil de 5 años con un valor residual de 100 €. Calculamos la base de amortización $Baseamortizacion = valorinicial - valorresidual$ por lo que nos queda un valor residual de 700 €. Dividimos este valor entre la vida útil estimada para este ordenador y nos queda un total de 140 € al año. Durante los 7 meses que ha durado este proyecto hemos supuesto que el tiempo que el ordenador ha estado siendo usado son unos 2 meses, por lo tanto obtenemos un precio de **23.33 €**.
- **Total :** Sumando todos los costes tenemos un total de **3565.46 €**

Capítulo 4

Módulo hardware

La idea principal de este proyecto es utilizar todo el hardware libre posible, que entre otras ventajas nos permite abaratar los costes que es una de las ideas principales del proyecto.

En este capítulo se describirá detalladamente todo el hardware que se ha utilizado para llevar a cabo este proyecto.

4.1. Requisitos

A continuación se recogerán los requisitos tanto funcionales como no funcionales del módulo hardware:

4.1.1. Requisitos funcionales

- **RF1:** Recoger las mediciones mediante un sensor.
- **RF2:** Manejar la información obtenida para mostrar información útil para el usuario.
- **RF3:** Enviar dicha información a un servidor para poder tener todos los datos almacenados.
- **RF4 (Opcional):** Mostrar dicha información al usuario mediante una pantalla.

4.1.2. Requisitos no funcionales

- **RNF1:** La placa que se diseñe debe tener unas dimensiones reducidas para reducir costes entre otras cosas.
- **RNF2:** Las dimensiones de la caja donde se alojará la placa tendrá las dimensiones lo más reducidas posible.

- **RNF3:** El aparato debe permitir su instalación sin necesidad de contar con la ayuda de un electricista o de modificar cualquier elemento de la instalación eléctrica donde queramos medir los consumos.

4.2. Planificación

- **Fase 1:** Reunión con la persona interesada para concretar los detalles del proyecto que se quieren llevar a cabo.
 - **Recopilar la información** sobre los detalles que nos especifique la persona interesada.
 - **Estudiar** si es factible llevar a cabo las peticiones que se nos han hecho.
 - Tener una **segunda reunión** para concretar las cosas que se pueden llevar a cabo y las que no.
- **Fase 2:** Estudio de las tecnologías disponibles y compra del material.
 - **Revisar** que componentes de hardware libre se adecuan para llevar a cabo nuestro proyecto
 - **Estudiar** cuales de esos componentes son los más adecuados basándonos en aspectos como el precio, la comunidad o la disponibilidad que tengan estos componentes.
 - **Comparar** los distintos precios que nos ofrecen las tiendas para la compra de estos componentes y buscar la que nos ofrezca la mejor relación entre un tiempo de envío no demasiado alto y un precio adecuado.
- **Fase 3:** Diseño y montaje.
 - **Diseño** de los esquemas de montaje.
 - **Montaje** de dichos esquemas utilizando una protoboard[31].
 - **Montaje y pruebas** con distintos prototipos.
 - **Soldar** la PCB con los componentes correspondientes.
 - **Fabricación** de distintas carcasa para alojar la placa final
- **Fase 4:** Pruebas.
 - **Pruebas** de los distintos prototipos para comprobar su correcto funcionamiento.

4.3. Arquitectura del sistema

En este apartado se van a detallar los componentes que se han utilizado para desarrollar cada uno de los módulos hardware que componen este sistema.

4.3.1. Módulo principal

Podríamos haber optado por el uso de otra placa como puede ser un Arduino [3]. El principal inconveniente que presenta esta placa para su utilización en este proyecto es que carece de conexión inalámbrica WiFi por lo tanto para poder mandar los datos al servidor vía WiFi se necesitaría un módulo a parte para dotar al Arduino de conexión WiFi.

Para permitir que el Arduino disponga de conexión WiFi se utiliza el módulo WiFi ESP8266. En el caso de este proyecto únicamente la placa NodeMCU Amica cubre todas nuestras necesidades las cuales son:

- Conexión inalámbrica WiFi.
- Conexión I2C.
- Bajo coste.
- Tamaño de la placa muy reducido.

De esta manera conseguimos a parte de hacer un diseño más sencillo abaratar costes, ya que no será necesario adquirir además un Arduino.

Las especificaciones del ESP8266 se pueden consultar en el apartado 1.1.1.

4.3.2. Módulo sensor

En este proyecto es necesaria la utilización de un sensor de corriente no invasivo para permitir una fácil instalación del dispositivo sin necesidad de modificar ningún cable de la instalación eléctrica.

Existen en el mercado distintos sensores de energía no invasivos como el que nos ofrece la marca SparkFun [37].

Para este proyecto vamos a utilizar el sensor de la marca YHDC ya que es el que más versiones nos ofrece y la disponibilidad del mismo es mayor que el de otros sensores de similares características.

En concreto se va a usar la versión de 100A que nos permitirá medir cargas de hasta 100A.

En esta imagen (4.1) podemos ver que la salida del sensor se produce a través de un jack de audio por el canal de la izquierda y el sleeve. En las demás versiones de pinzas de la marca con la de 30A contamos con una resistencia de carga que hace que la salida sea una señal de voltaje. Sin

embargo en el caso de la pinza de 100A como podemos ver en el esquema (4.1) en la figura de la izquierda la señal de salida que nos proporciona es continua ya que carece de una resistencia de carga por lo tanto tendremos que añadir a nuestro esquema eléctrico una resistencia de carga para convertir a una señal de voltaje.

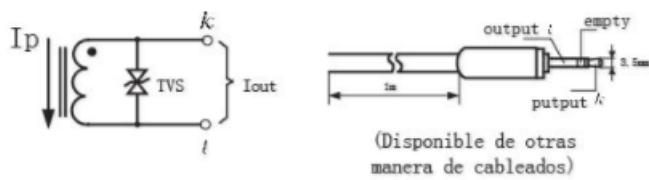


diagrama esquemático de
Esquema elemental estandar de 3 pines conectores

Figura 4.1: YHDC. Fuente: [45]

4.3.3. Módulo conversor analógico digital

El propio ESP8266 pueda realizar una conversión de una señal analógica a una digital mediante el PIN A0 con una resolución de 10 bits.

Sin embargo si utilizamos el conversor analógico digital ADS 1115 conseguiremos una resolución de 16 bits. Además utilizando la biblioteca Adafruit_ADS1015 [1] en concreto la función *readADC_Differential_0_1()* la cuál nos permite conocer la diferencia de voltaje entre el pin 0 y el pin 1 del ADS esto hace posible que no sea necesario el uso de dos resistencias en paralelo "bias resistor". Para llevar a cabo nuestro esquema eléctrico tendremos que hacer uso de los pines VDD y GND los cuales alimentaremos a 3,3 V, los pines SCL y SDA para conectarlo al ESP8266 haciendo uso de la conexión I2C y los pines A0 y A1.

Las ventajas que nos ofrece este conversor analógico digital son:

- Nos permite tener más precisión.
- Conexión I2C con la placa ESP8266.
- Elimina la necesidad de incorporar al circuito dos resistencias adicionales.

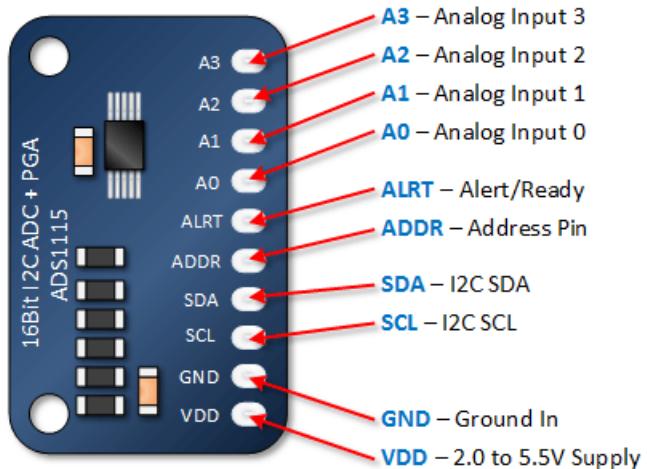


Figura 4.2: ADS1115. Fuente: [22]

4.3.4. Módulo pantalla

Como se especificó en el capítulo 2 de objetivos, el módulo de la pantalla corresponde a un objetivo opcional que finalmente se ha decidido incluir en el proyecto final.

A parte de ver la información detallada sobre los consumos en la web puede ser interesante para el usuario disponer de la información de forma inmediata y cómoda mediante una pantalla integrada en el propio dispositivo.

Debido a su bajo precio y que muchas de las soluciones para la medición de consumos eléctricos existentes en el mercado como se vio en el apartado 1.2 disponen se decidió que sería una buena idea incluirla en el diseño. Si la persona que deseé realizar este proyecto no ve necesaria la utilización de esta pantalla se podría quitar sin ningún tipo de problema teniendo de esta manera la misma versión del dispositivo con las mismas funcionalidades pero sin la pantalla.

Se trata de una pantalla LCD de 16x02 adquirida con un adaptador I2C directamente soldado a los pines que facilita de esta forma la conexión con el ESP8266. Gracias a la conexión I2C podemos conectar sin ningún tipo de problema a los puertos D1 y D2 de nuestro ESP8266 (puertos por defecto para la comunicación I2C en el ESP8266) especificando con qué puerto queremos comunicarnos no tendremos ningún tipo de problema a la hora de que tanto el conversor de señal analógico digital (ADS1115) y el LCD compartan la conexión con los puertos D1 y D2.

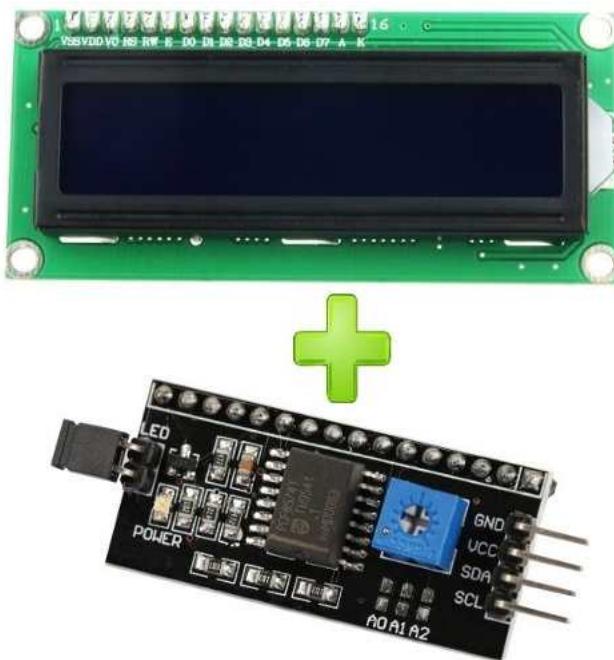


Figura 4.3: LCD 16x02 y módulo I2C

4.4. Coste unitario

Los precios mostrados a continuación (4.1) pueden variar en función de la tienda donde se adquieran los productos además pueden variar su precio a la alza o a la baja con el paso del tiempo. No están reflejados los gastos de envío de los productos ya que varían mucho en función de la web donde se adquieran.

Cabe destacar que los precios de algunos de los productos que aparecen en la tabla como puede ser la PCB están calculados restando el precio de la parte utilizada para llevar a cabo el proyecto al precio de la PCB completa.

Producto	Precio
NodeMcu ESP8266 CP2102	5.54 €
YHDC Non-Invasive AC Current Sensor [100A/30A]	5.41 €
ADS1115 16bits ADC+PGA	2.45 €
Pantalla LCD 16x02 (Opcional)	3.49 €
40x cable macho macho	1.80 €
Estaño	0.40 €
Tira pines macho	1 €
Tira pines hembra	3 €
Trozo de placa experimental prototipo (PCB)	2 €
Resistencias, jack hembra, conector entrada alimentación	4 €
Tornillos, tuercas	0.50 €
Total	29.59 €
Total sin Pantalla LCD	26.10 €

Tabla 4.1: Listado de precios.

4.5. Prototipos

Todos los diseños de estos prototipos se han realizado con el programa Fritzing antes de llevar a cabo pruebas en la protoboard. Todos estos prototipos se han montado en una protoboard para ahorrarnos de esta forma tener que soldar y gastar más material cada vez que se quisiera probar un nuevo diseño.

En este apartado se detallan las mejoras y fallos de cada uno de los 4 diseños que se analizaron para la construcción de este proyecto:

- **Prototipo 1:** Se trata de la versión inicial con la que comenzó este proyecto. Es un diseño básico que no cuenta ni con pantalla para mostrar información instantánea de los datos que está recogiendo el sensor en cada momento ni con una entrada para poder alimentarlo mediante un conector jack hembra de alimentación. La alimentación de este prototipo iba a través del micro-usb que incorpora la placa NodeMCU.

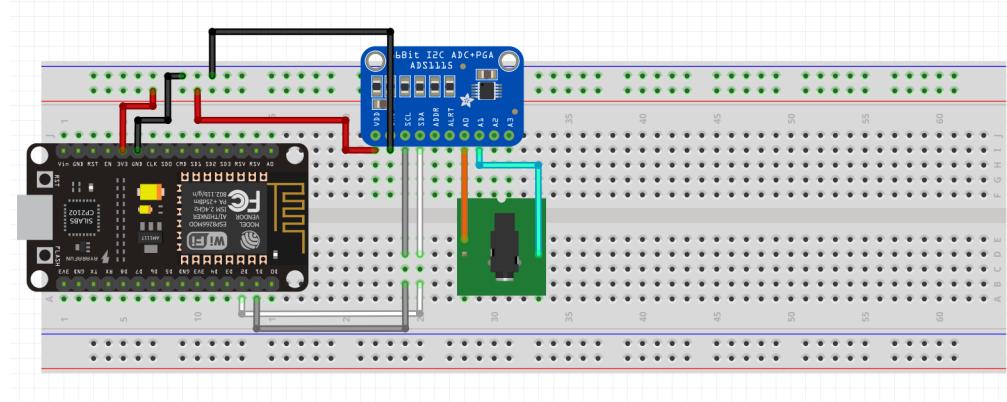


Figura 4.4: Esquema v1

- **Prototipo 2:** Este prototipo incorpora un jack hembra de alimentación que permite alimentar el prototipo ya sea mediante el micro-usb propio de la placa o mediante un cargador de los que suelen usarse normalmente para alimentar otras placas como Arduino con una salida de 5V, lo que se pretendía con este prototipo era usar un único cargador con una salida de 5V para evitar que el usuario cogiera cualquier otro cargador de móvil o de algún otro dispositivo que disponga de una conexión micro usb ya que estos tienen una salida muy diferente los unos de los otros dependiendo de cuál sea el modelo de cargador y la empresa que fabrica el mismo.

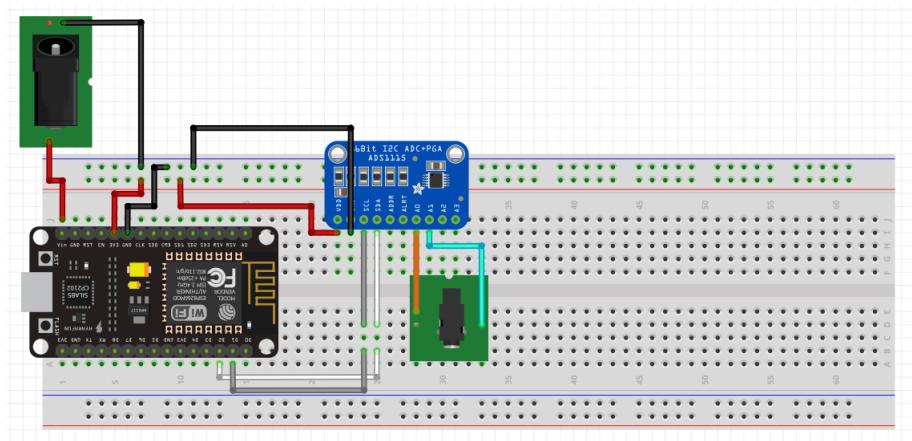


Figura 4.5: Esquema v2

- **Prototipo 3:** En este diseño se incorpora el amplificador LM358 [38] que intenta solucionar los malos resultados de los prototipos vistos anteriormente, pero realmente la solución a estos problemas se encon-

traban en la forma de poner el sensor de corriente ya que si el sensor rodea los dos cables la lectura no será correcta esta será cercana al 0 ya que los cables tienen corrientes opuestas, estas lecturas cercanas al 0 son el motivo principal por el que se intentó utilizar un amplificador de señal como solución al problema.

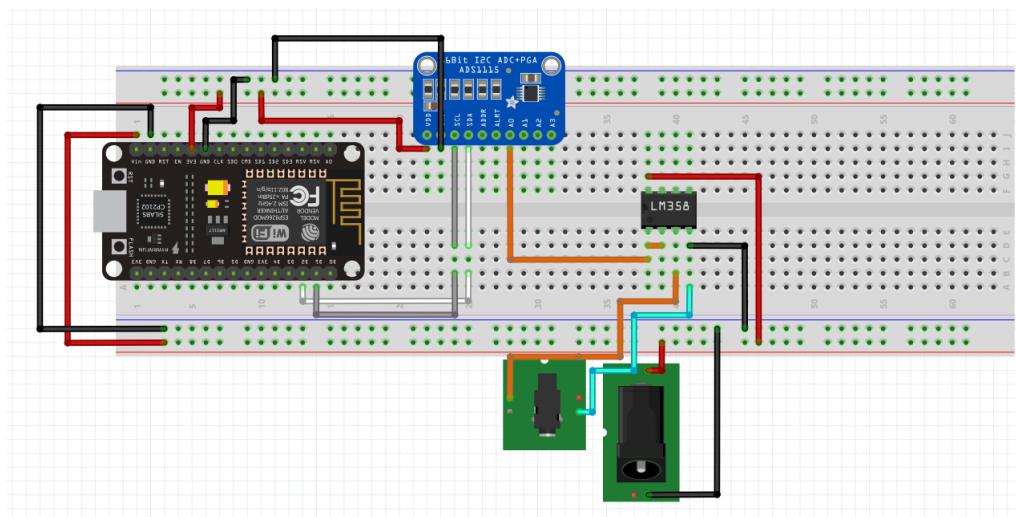


Figura 4.6: Esquema v3

- **Prototipo 4:** Se trata de la versión final del proyecto la cuál incluye una pantalla LCD conectada mediante I2C que permite al usuario ver de una manera cómoda la información que está recogiendo en ese momento el sensor.

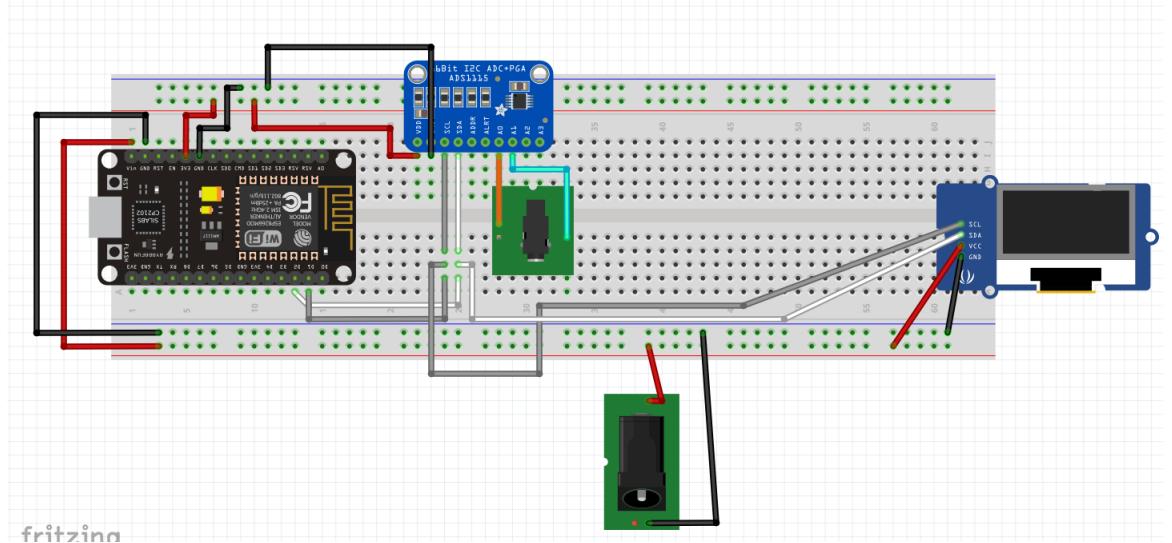


Figura 4.7: Esquema v4

4.6. Versiones finales

En este apartado se verán los diseños de las dos versiones finales soldando pines hembra a una PCB para facilitar de esta forma el fácil reemplazo de los componentes en caso de que alguno de estos se estropeara o simplemente se quisiera retirar para darle otro uso sin que sufra ningún desperfecto a la hora de retirarlo.

El diseño para soldar las pistas de estaño en la PCB se ha realizado mediante el programa fritzing para asegurar que ninguna de estas pistas se toquen entre ellas provocando de esta forma un mal funcionamiento y al mismo tiempo nos aseguramos de que el tamaño de la placa esta utilizado de la mejor forma posible haciendo la placa lo más pequeña posible por si en un futuro se decidiera producir una gran cantidad de estas placas ahorrar costes de producción además de conseguir un dispositivo de menor tamaño.

- **Versión 1 :** una vez que la persona interesada dio el visto bueno a este diseño (4.8) realizado soldando pines hembra a una PCB para facilitar el cambio de componentes se pasa a realizar el primer modelo final soldado a una pcb como podemos ver en la imagen (4.9), observamos 8 conectores hembra donde van colocados los pines correspondientes a la alimentación y comunicación I2C de la pantalla LCD, los conectores para el jack del sensor de corriente y los conectores para conectar el jack de alimentación encargado de dar corriente al ESP8266.

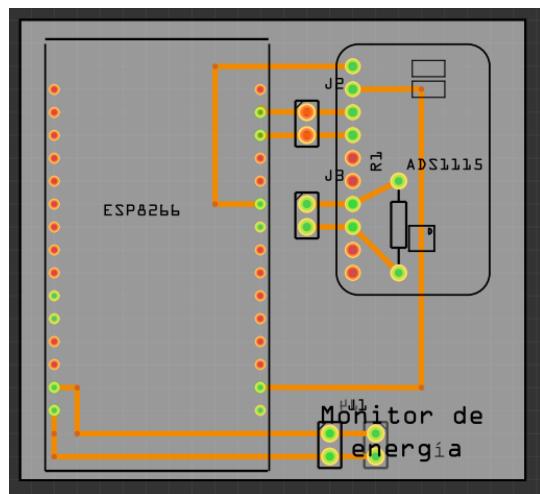


Figura 4.8: Esquema PCB versión 1

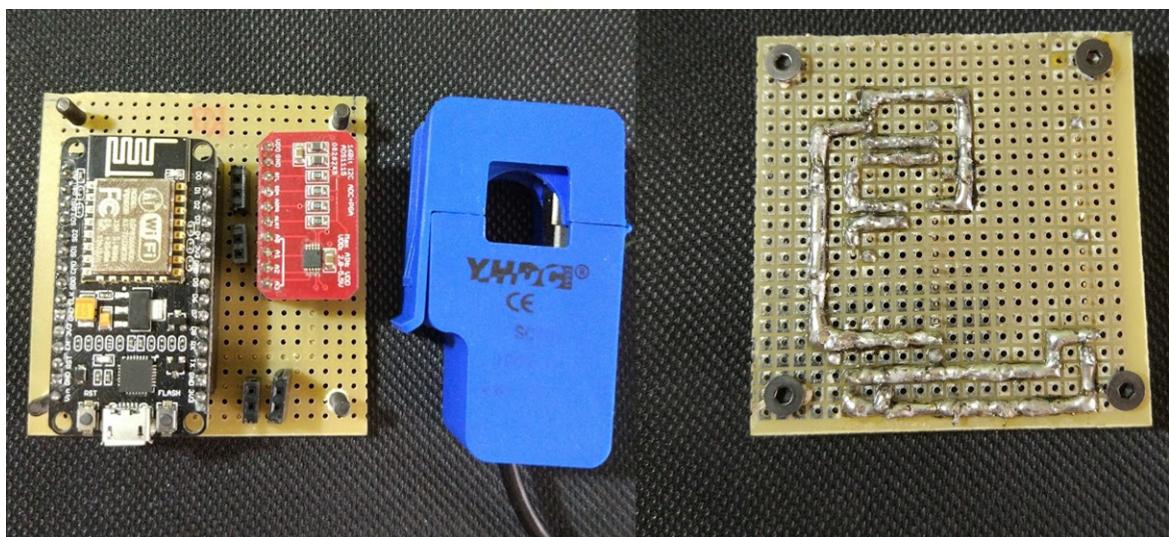


Figura 4.9: PCB soldada versión 1

- **Versión 2 :** Con la idea de ahorrar espacio y hacer una caja de dimensiones algo más reducidas se estudió un nuevo diseño que permitiera tener uno de los lados de la placa con más espacio, con la idea de hacer la placa más larga que es algo poco importante ya que la caja de todas formas ha de tener el largo suficiente para poder alojar la pantalla LCD, pero un poco más estrecha pudiendo de esta manera colocar la pantalla LCD en uno de los laterales de la caja. Consiguiendo de esta forma que la tapa de la caja pese menos al no tener que soportar el peso de la pantalla LCD.

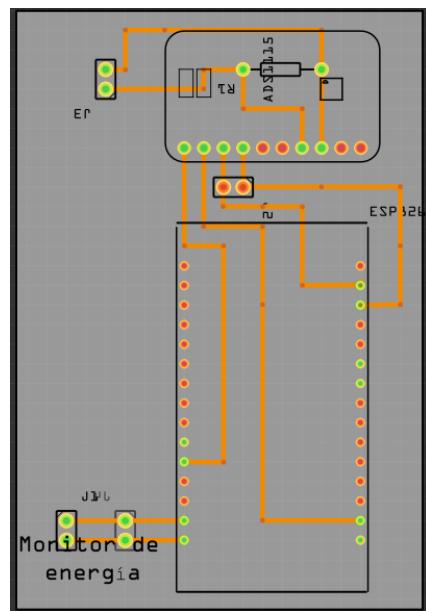


Figura 4.10: Esquema PCB versión 2

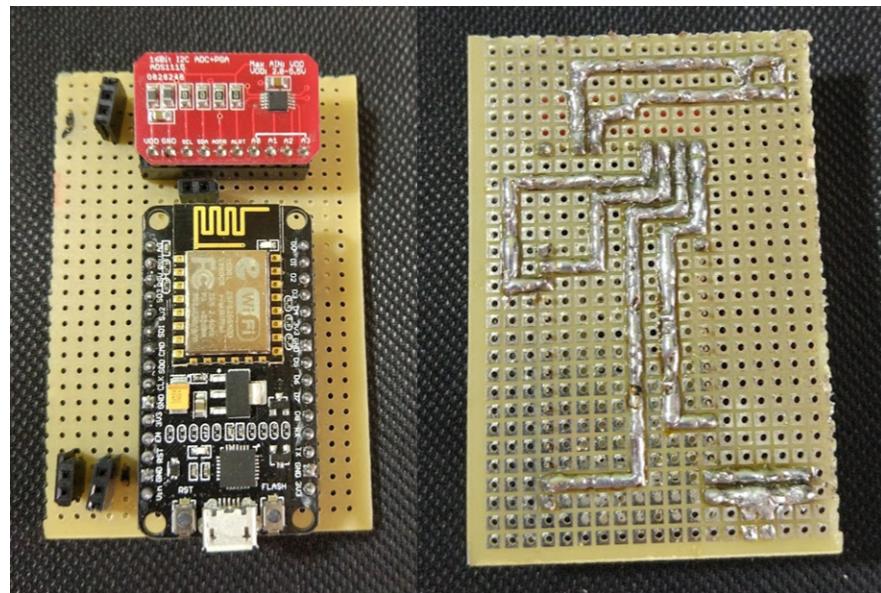


Figura 4.11: PCB soldada versión 2

4.7. Montaje de la caja

Se ha utilizado una cortadora láser para construir las piezas necesarias para alojar la placa y los componentes necesarios en una caja.

Los primeros diseños de la caja están elaborados con DM ya que debido al bajo precio de este material nos permite realizar todas las pruebas que queramos y en caso de que alguna pieza no encaje o el resultado final no sea el esperado siempre podremos repetir esta pieza sin necesidad de desperdiciar un material de precio superior.

- **Versión 1 :** En esta primera versión contamos con una caja básica que permite tener acceso tanto a la entrada micro usb de la placa ESP8266 por si fuera necesario cambiar el firmware de la misma, además de tener acceso a la entrada de jack del sensor de corriente y la entrada para la alimentación tal y como se puede apreciar en la imagen (4.12)



Figura 4.12: Caja versión 1

- **Versión 2 :** En esta segunda versión se mejora la accesibilidad a los componentes que se encuentran en el interior de la caja introduciendo dos pestañas que actúan como bisagras (visibles en la imagen 4.13) sujetando la tapa de arriba de la caja la cuál incluye la pantalla haciendo posible de esta forma la apertura de la caja para cambiar cualquier elemento.

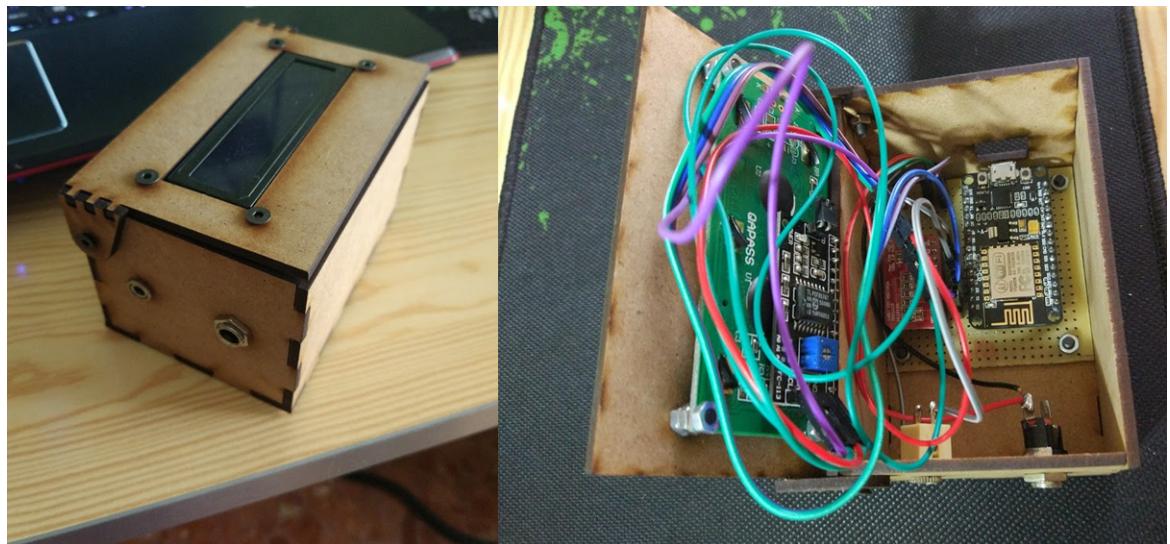


Figura 4.13: Caja versión 2

Capítulo 5

Módulo firmware

En este capítulo se especificarán todos los detalles relacionados con la programación que se ha llevado a cabo en la placa Node MCU ESP8266 [42]. Esta placa permite al usuario que se programe tanto en LUA [25], MicroPython [18] o Arduino [3].

Para la realización de este proyecto se ha elegido el lenguaje de programación de Arduino ya que está basado en C y me siento más cómodo con este lenguaje en comparación con micropython. Además los códigos en C son algo más rápidos y podrían ocupar algo menos de memoria.

5.1. Requisitos

5.1.1. Requisitos funcionales

- **RF1:** Introducir manualmente las credenciales de la conexión WiFi así como el tiempo que tardará en enviarse la información al servidor.
- **RF2:** Comprobar que el dispositivo está correctamente conectado a la red WiFi.
- **RF3:** Recoger la información de los sensores.
- **RF4:** Manejar la información obtenida para mostrar información útil para el usuario.
- **RF5:** Mostrar dicha información al usuario mediante una pantalla.
- **RF6:** Enviar dicha información a un servidor para poder tener todos los datos almacenados.

5.1.2. Requisitos no funcionales

- **RNF1:** La conexión WiFi del dispositivo ha de ser estable ya que si no los datos no podrían ser enviados al servidor.

- **RNF2:** La información que se muestra en la pantalla LCD se borrará antes de mostrar información nueva para evitar que la información se muestre en lugares de la pantalla inadecuados.

5.2. Planificación

- **Fase 1:** Reunión con la persona interesada para concretar los detalles del proyecto que se quieren llevar a cabo.
 - **Recopilar la información** sobre los detalles que nos especifique la persona interesada.
 - **Estudiar** si es factible llevar a cabo las peticiones que se nos han hecho.
 - Tener una **segunda reunión** para concretar las cosas que se pueden llevar a cabo y las que no.
- **Fase 2:** Estudio para decidir en qué lenguaje se va a llevar a cabo la programación y como se va proceder para enviar los datos al servidor.
 - **Investigar** qué lenguaje de programación de los 3 (LUA, MicroPython, Arduino) que permite el ESP8266 nos interesa más para llevar a cabo este proyecto.
 - **Estudiar** como interpretar los datos recogidos por el sensor para transformarlos en datos legibles por parte del usuario.
 - **Estudiar** cual es la mejor forma de mostrar la información en la pantalla LCD para mostrarla al usuario.
 - **Estudiar** cuál es la mejor forma de mandar los datos al servidor.
 - **Estudiar** cuales son las bibliotecas necesarias para realizar las labores citadas anteriormente.
- **Fase 3:** Programación de los distintos módulos.
 - **Programación** del módulo de cálculo de irms (valor eficaz) [44].
 - **Programación** del módulo para mostrar la información en la pantalla LCD.
 - **Programación** del módulo para enviar la información al servidor.
- **Fase 4:** Pruebas.
 - **Pruebas** de los distintos módulos que se han programado.

5.3. Arquitectura del firmware

En este apartado se va a detallar el desarrollo de los distintos módulos que han sido necesarios para la elaboración de este proyecto.

5.3.1. Módulo de lectura de sensor y cálculo de irms y potencia

- **Lectura del sensor:** La lectura de los valores del sensor se realizarán mediante el conversor analógico digital ADS1115. Para ello haremos uso de la biblioteca proporcionada por Adafruit en concreto la Adafruit_ADS1x15 [1] válida tanto para la versión 1015 de 12 bits como para la versión 1115 de 16 bits.

Esta biblioteca nos permite leer los valores de los pines desde el A0 hasta el A3 bien sea de forma única *singleended* como calculando la diferencia entre dos pines *differential* a parte de una tercera forma *comparator*. En nuestro caso utilizaremos el modo que calcula la diferencia entre dos pines, es decir la función :

```
int16_t readADC_Differential_0_1();
```

Esta biblioteca la podemos encontrar en el repositorio de GitHub de Adafruit [1].

- **Cálculo de irms y potencia:** Para este proyecto se van a calcular el irms o valor eficaz [44] y a partir de ahí la potencia real.

Tras ver los resultados obtenidos por el sensor tal cuál los leemos gracias a la función de la biblioteca de Adafruit mencionada anteriormente. Estamos midiendo corriente alterna, la cuál está cambiando constantemente, por lo tanto tendremos valores con una polaridad negativa y otros con polaridad positiva. Dependiendo del tipo de onda obtenida la fórmula que hay que aplicar para calcular el irms cambiará. En la gráfica (5.1) podemos ver que los valores que capta el sensor son tanto positivos como negativos obteniendo esta onda. Por lo tanto no nos interesa obtener ni la media de todos estos valores obtenidos por el sensor ni el valor pico de la onda lo que nos interesa calcular es el valor eficaz. En nuestro caso particular según la bibliografía [41] para calcular el irms de una onda sinusoidal generada al medir corriente alterna la solución es simplemente calcular la media cuadrática aplicando la fórmula de esta media [41] $\sqrt{\frac{x_1^2+x_2^2+\dots+x_n^2}{N}}$.

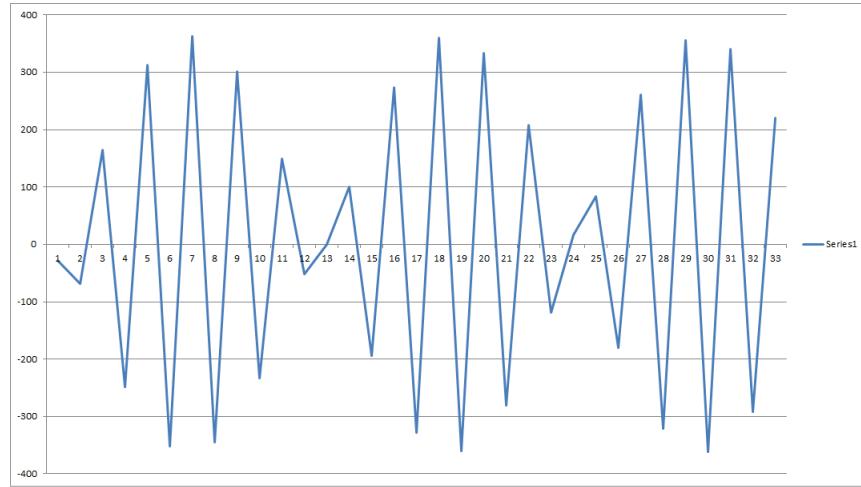


Figura 5.1: Representación datos obtenidos del sensor.

Pero antes de realizar este cálculo tendremos que calcular el offset mediante un filtro de paso bajo software al tratarse de un filtro paso bajo el offset corresponde a los valores más altos de la onda y tendremos que restar este valor de offset, de esta forma conseguimos dejar la onda lo más centrada posible atenuando los valores altos y dejando pasar los más bajos [40]. En nuestro caso el sistema se está alimentando con 3.3V por lo tanto el offset que se le resta a la muestra es el equivalente a una corriente de 1,65V.

Lo que conseguimos restándole este offset al valor obtenido mediante el sensor es eliminar ese voltaje que tiene de por si el sistema aunque no se esté midiendo nada es decir cuando está en reposo. De todas formas el sistema se calibrará gracias a la variable ICAL ya que dependiendo del sistema que montemos la calibración variará.

Una vez calculado el irms (A) podremos calcular la potencia real (W) del aparato que estemos midiendo gracias a la ley de Watt y la ley de Ohm obteniendo la fórmula $P(W) = (irms(A) \cdot V(V))$

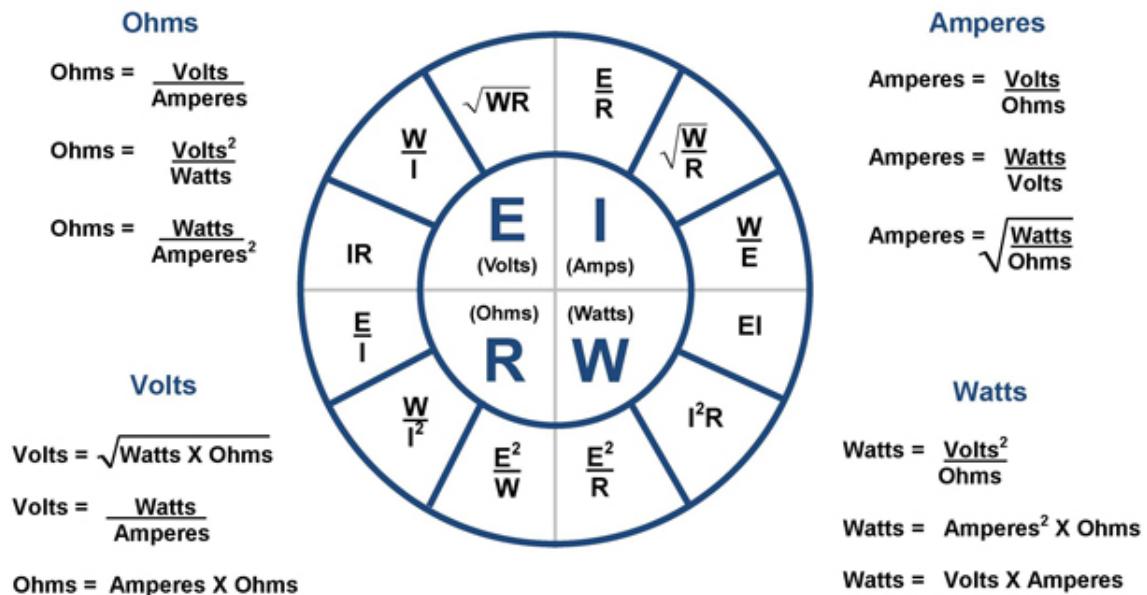


Figura 5.2: Ley de Ohm y Watt. Fuente [2].

5.3.2. Módulo para mostrar información en la pantalla LCD

Para mostrar información en una pantalla LCD no existen bibliotecas exclusivas diseñadas para el ESP8266 pero esta biblioteca *LiquidCrystal_I2C.h* diseñada para Arduino [24] nos permitirá mostrar la información que queramos sin ningún tipo de problema. Al estar compartiendo la conexión I2C con el conversor analógico digital tendremos que especificar en la creación del objeto la dirección de memoria de la pantalla LCD.

Una vez creado el objeto tan solo tendremos que llamar a la función creada para mostrar la información deseada la cuál se le pasará a la función como argumentos:

```
void imprimirLcd(double irms, double potencia){
    lcd.clear();
    lcd.print("irms: "); lcd.print(irms); lcd.print("A");
    lcd.setCursor(0,1);
    lcd.print("Potencia: "); lcd.print(potencia); lcd.print
        ("W");
}
```

5.3.3. Módulo envío de información al servidor

Para el envío de la información se ha decidido enviarla mediante HTTP (Hypertext Transfer Protocol), se trata de un protocolo de comunicación en

red que hace posible la comunicación entre un cliente y un servidor mediante peticiones por parte del cliente y respuestas por parte del servidor [26]. Para este proyecto vamos a utilizar el método de petición POST mediante el cuál seremos capaces de enviar la información que deseemos especificando la URL del servidor donde queramos enviar dicha información, cuando esa información sea recibida por el servidor se enviará una respuesta al cliente que generó la petición POST. Gracias a la biblioteca *ESP8266HTTPClient.h* [23] una biblioteca para dispositivos IoT es posible realizar (entre otros métodos implementados en esta biblioteca) una petición POST de una manera muy sencilla en la placa ESP8266 en concreto gracias a la función:

```
int POST(String payload);
```

A la hora de escoger en qué formato enviar la información al servidor se ha escogido JSON (JavaScript Object Notation) que no es más que un formato para realizar intercambio de datos. Haciendo uso de la biblioteca *ArduinoJson.h* [6] con la que podremos codificar arrays o incluso añadir valores al objeto creado directamente.

Todo esto se realizará llamando a la función que se ha creado para hacer el código más legible. Pero la parte correspondiente únicamente a la codificación del JSON es la siguiente

```
StaticJsonBuffer<300> JSONbuffer; //Declarar JSON buffer
JsonObject& JSONencoder = JSONbuffer.createObject();

JSONencoder["sensorType"] = "Power";
Serial.println("irms dentro de mandarjson: ");Serial.println(
    irms);
JSONArray& values = JSONencoder.createNestedArray("values");
//JSON array
values.add(irms); //Aniadir valor al array
values.add(potencia); //Aniadir valor al array

JSONArray& valuesInfo = JSONencoder.createNestedArray("valuesInfo");
//JSON array
valuesInfo.add("irmsnuevo"); //Aniadir valor al array
valuesInfo.add("potencianueva"); //Aniadir valor al array

char JSONmessageBuffer[300];
JSONencoder.prettyPrintTo(JSONmessageBuffer, sizeof(
    JSONmessageBuffer));
Serial.println(JSONmessageBuffer);
```

Capítulo 6

Módulo servidor

Para este proyecto se ha decidido crear un servidor propio con IP local. Es cierto que este servidor podría haberse alojado en algún hosting. Pero para asegurar la privacidad de los datos que publica el usuario de este monitor se ha decidido que lo mejor es alojar el servidor en una Raspberry Pi con IP local.

6.1. Requisitos

6.1.1. Requisitos funcionales

- **RF1:** Ha de permitir al usuario consultar los consumos en un intervalo de fechas.
- **RF2:** Mostrar mediante una tabla los consumos ordenados por fecha y hora.
- **RF3:** En caso de mostrar una tabla muy extensa dividirla por páginas mediante un paginador.
- **RF4:** Mostrar la media de consumo durante las fechas especificadas por el usuario.
- **RF5:** Mostrar mediante un gráfico consumos del mes anterior y posterior al consultado por el usuario.

6.1.2. Requisitos no funcionales

- **RNF1:** Hacer que el diseño de la web sea responsive, es decir que se adapte a tamaños de pantalla diferentes para ajustarse a dispositivos móviles como smartphones o tablets o a ordenadores.
- **RNF2:** Persistencia de datos mediante una base de datos NoSql.

- **RNF3:** Se utilizarán tecnologías open source.
- **RNF4:** Garantizar la privacidad de los datos del usuario.
- **RNF4:** Compatibilidad con los navegadores.

6.2. Backend

La parte del Backend de la aplicación web está realizada con Flask [16]. Se trata de un microframework para python que utiliza plantillas Jinja2 [30]. Para una aplicación web pequeña como es el caso de la que se va a realizar para este proyecto nos ha parecido más apropiado utilizar este microframework ya que al contrario que otros framework como puede ser Django cuenta con pocas características las cuales se pueden ir añadiendo mediante módulos los cuales en función de nuestras necesidades podremos ir añadiendo, sin embargo en un framework como Django [12] ya se incluyen todos estos componentes.

El motivo principal para escoger Flask frente a Django principalmente ha sido por el conocimiento previo que tenía adquirido en la asignatura desarrollo de aplicaciones para internet. También se había trabajado con Django en la asignatura infraestructura virtual pero en menor medida.

Para la persistencia de datos hemos utilizado bases de datos NoSQL más concretamente MongoDB, se trata de una base de datos orientada a documentos los datos se guardan en documentos BSON (una representación binaria de JSON) en lugar de en registros como ocurre en bases de datos convencionales. Esto permite introducir cualquier dato aunque se cambie el formato al contrario que en las bases de datos relacionales que es necesario seguir un esquema [34].

6.3. Frontend

Para realizar el frontend de la aplicación se han utilizado plantillas HTML 5 bootstrap para conseguir que el diseño de la web sea responsive. Al utilizar bootstrap conseguimos que el diseño de la web sea compatible con la mayoría de navegadores web [8]. Se necesita además una herramienta open source para realizar las gráficas de consumo. Para ello se ha utilizado la herramienta Highcharts la cual nos permite realizar gráficas utilizando JavaScript.

6.4. Servidor

Para alojar la aplicación web diseña para este proyecto no es necesario un servidor con demasiada potencia por lo tanto con alojar la web en una Raspberry Pi nos será más que suficiente.

6.4.1. Sistema Operativo

Como vamos a utilizar la Raspberry como servidor instalaremos un sistema operativo sin interfaz gráfica en concreto Raspbian Jessie Lite. Para instalar el sistema operativo en la tarjeta micro SD que introduciremos en la Raspberry tan solo tendremos que flashear la imagen que nos descargaremos de la web oficial de raspberry [33]. Para flashear la imagen en la SD existen múltiples programas pero en mi caso he elegido utilizar Win32 disk imager [21] un sencillo software open source disponible para múltiples plataformas que nos permitirá flashear la imagen en la tarjeta micro SD muy fácilmente. Hemos de tener en cuenta que previamente la tarjeta micro SD tendrá que estar formateada para lo cuál nos servirá cualquier software open source de los que están disponibles.

6.4.2. Aprovisionamiento

Para facilitar que cualquier persona interesada en el proyecto pueda montar de la forma más sencilla posible el servidor tan solo tendrá que descargarse mi repositorio de GitHub <https://github.com/Miguelmoral/ESP8266-energy-monitor>, los pasos a seguir para su descarga y puesta en funcionamiento son bastante sencillos [28]:

1. **Instalar git:** Para poder descargar el repositorio de GitHub será necesaria la instalación de git en nuestra máquina con el comando:

```
$ sudo apt-get update  
$ sudo apt-get install git
```

2. **Descargar repositorio:** Una vez tenemos instalado git podremos descargar el repositorio introduciendo:

```
$ sudo git clone https://github.com/Miguelmoral/ESP8266-  
energy-monitor
```

3. **Instalar requirements:** Una vez descargado este repositorio en el cuál se incluye todo lo necesario para el correcto funcionamiento del servidor el usuario tendrá que ejecutar el comando

```
$ cd ESP8266-energy-monitor  
$ sudo make install
```

6.4.3. Iniciar aplicación web

El *makefile* incluye además otro comando que permite directamente la ejecución de la aplicación web en el servidor, hay que tener en cuenta que

para que este comando funcione previamente tendremos que haber realizado los pasos explicados anteriormente.

```
$ make ejecutar
```

Al tratarse de una Raspberry Pi la cuál no dispone de pantalla sería interesante hacer que se ejecute automáticamente la aplicación web cada vez que arranquemos el sistema. Existen varias formas de conseguirlo [11]:

1. Modificando el archivo rc.local.
2. Modificando el archivo .bashrc.
3. Añadir el *script* o comandos que se quieran ejecutar al directorio /etc/init.d.
4. Utilizando los archivos systemd.
5. Mediante el demonio cron. Tendremos que añadir los script que queramos ejecutar en el fichero crontab y especificar cuando queramos que se ejecuten.

Para este proyecto se va a modificar el archivo rc.local de la máquina. Tan solo tendremos que abrir este archivo con un editor de textos con permisos root para poder editarla e incluir antes de exit 0 el comando que queramos que ejecute la máquina cada vez que se encienda en nuestro caso.

Abriremos un editor de textos para modificar el archivo rc.local con

```
$ sudo nano rc.local
```

Y añadiremos la orden

```
$ sudo python /home/pi/ESP8266-energy-monitor/web/main.py &
```

Quedándose el archivo rc.local de esta forma 6.1

```
GNU nano 2.2.6          File: /etc/rc.local

#
# By default this script does nothing.

# Print the IP address
_IP=$(hostname -I) || true
if [ "$_IP" ]; then
    printf "My IP address is %s\n" "$_IP"
fi

sudo python /home/pi/ESP8266-energy-monitor/web/main.py &

exit 0
```

Figura 6.1: rc.local Raspberry Pi

6.4.4. IP estática Raspberry

Establecer una IP estática en la Raspberry no es totalmente necesario para el proyecto ya que podría funcionar perfectamente con una IP dinámica, pero por comodidad es preferible establecer una IP estática.

Para que el ESP8266 pueda mandar la información siempre a la misma IP y no tener que modificar el código del firmare cada vez que la Raspberry cambie de IP vamos a hacer que la IP de la Raspberry que por defecto es dinámica pase a ser estática de esta forma esta IP será siempre la misma la cuál le asignaremos nosotros modificando el archivo.

Tendremos que consultar algunos valores sobre nuestra conexión antes de modificar el archivo /etc/network/interfaces [39]. Para ello utilizaremos el comando :

```
$ netstat -nr
```

Una vez sabemos cuál es nuestro broadcast range, subnet mask, gateway y destination podremos modificar el archivo /etc/network/interfaces de nuestra Raspberry para establecer una IP fija. Dependiendo de los valores que nos de dependiendo de la configuración de nuestra conexión tendremos que cambiar algunos datos a la hora de modificar este archivo pero en mi caso obtendríamos algo como lo que vemos en la imagen 6.2:

```

auto lo
iface lo inet loopback

iface eth0 inet static
address 192.168.0.106
netmask 255.255.255.0
network 192.168.0.0
broadcast 192.168.0.255
gateway 192.168.0.1

allow-hotplug wlan0
iface wlan0 inet manual

```

Figura 6.2: Configuración /etc/network/interfaces

6.4.5. Aspecto final de la página

Tendremos una simple página de inicio donde podremos seleccionar de que fecha a que fecha queremos visualizar los datos:

Figura 6.3: Página principal

Al introducir las fechas veremos una tabla con los datos correspondientes a las fechas introducidas, un apartado que nos muestra la media tanto del irms como de la potencia consumida y una gráfica correspondiente a las medias de el mes anterior al consultado y el siguiente. En la gráfica mostrada tanto el mes siguiente como el anterior sale a 0 ya que durante esos meses no se recogieron muestras. En la gráfica (6.4) podemos ver los resultados de medir una bombilla de 60 W durante un periodo de tiempo. El paginador nos mostrará 10 entradas por página de esta forma se podrá visualizar los

datos de una forma más clara que si se mostraran sin paginador. Además tendremos una gráfica que nos irá mostrando los últimos consumos que reciba el servidor (6.6)

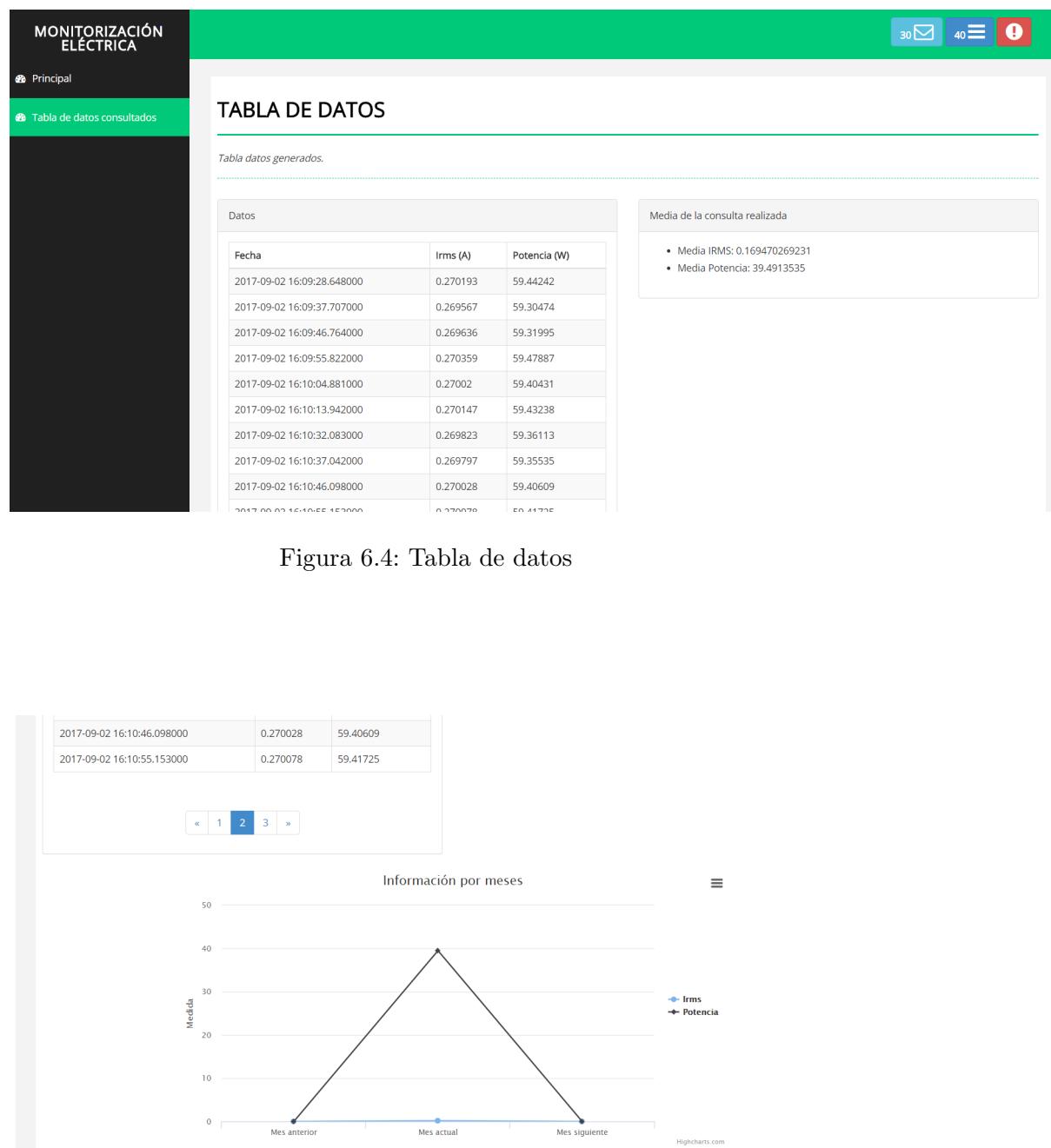


Figura 6.4: Tabla de datos

Figura 6.5: Gráfica mensual

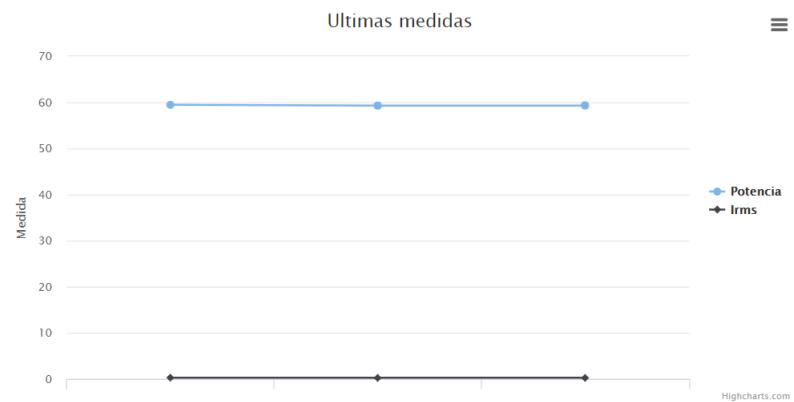


Figura 6.6: Gráfica ultimos consumos

Capítulo 7

Pruebas

En el siguiente capítulo se expondrán algunas de las pruebas realizadas hasta que se consiguió finalmente realizar un diseño funcional:

- **Prueba 1:** En este primera prueba se intentará hacer funcionar un circuito simple, en concreto el primer prototipo especificado en el capítulo 4 (4.4). Simplemente se quiere comprobar que todo funciona como debería por lo que en el firmware se cargará tan solo un programa simple que saque por la salida serial al ordenar los datos recogidos de el conversor analógico digital sin realizar ningún tipo de cálculo.

Número de prueba	1
Objetivo de la prueba	Comprobar funcionamiento del prototipo (4.4).
Procedimientos a llevar a cabo	<ul style="list-style-type: none">• Montar diseño del circuito en la protoboard.• Programar un firmware sencillo.• Flashear el firmware en la placa NodeMCU.• Visualizar resultados en la pantalla del ordenador.
Resultado deseado	Visualizar resultados lógicos dependiendo de que aparato esté conectado al sensor.
Resultado obtenido	Los resultados que vemos en pantalla son siempre 0 o demasiado cercanos al 0, además son siempre iguales independientemente de la electricidad que esté consumiendo el aparato en cada momento.
Estado de la prueba	Error.
Posibles soluciones	Diseñar otro prototipo con una entrada de energía diferente al micro USB.

Tabla 7.1: Prueba 1.

- **Prueba 2:** Para intentar solucionar el error de la prueba 1 se va a probar un segundo prototipo (4.5) alimentado a través del jack de corriente para ver si el problema era que le entraba un voltaje menor al necesario utilizando la entrada micro USB propia de la placa NodeMCU.

Número de prueba	2
Objetivo de la prueba	Comprobar funcionamiento del prototipo (4.5).
Procedimientos a llevar a cabo	<ul style="list-style-type: none"> • Montar diseño del circuito en la protoboard. • Programar un firmware que envíe los datos a un servidor. • Flashear el firmware en la placa NodeMCU. • Visualizar resultados que se muestran en el servidor.
Resultado deseado	Visualizar resultados lógicos dependiendo de que aparato esté conectado al sensor.
Resultado obtenido	La placa NodeMCU ha resultado dañada al utilizar un cargador cuya salida es corriente alterna.
Estado de la prueba	Error.
Posibles soluciones	Utilizar un cargador cuya salida sea corriente continua.

Tabla 7.2: Prueba 2.

- **Prueba 3:** Para intentar solucionar el error de la prueba 1 se va a probar un segundo prototipo (4.5) alimentado a través del jack de corriente para ver si el problema era que le entraba un voltaje menor al necesario utilizando la entrada micro USB propia de la placa NodeMCU.

Número de prueba	2
Objetivo de la prueba	Comprobar funcionamiento del prototipo (4.5)
Procedimientos a llevar a cabo	<ul style="list-style-type: none">• Montar diseño del circuito en la protoboard.• Programar un firmware que envíe los datos a un servidor.• Flashear el firmware en la placa NodeMCU.• Visualizar resultados que se muestran en el servidor.
Resultado deseado	Visualizar resultados lógicos dependiendo de que aparato esté conectado al sensor.
Resultado obtenido	Los resultados que vemos en pantalla son siempre 0 o demasiado cercanos al 0, además son siempre iguales independientemente de la electricidad que esté consumiendo el aparato en cada momento.
Estado de la prueba	Error.
Posibles soluciones	Comprobar que el sensor de corriente funcione correctamente.

Tabla 7.3: Prueba 2.

- **Prueba 4:** Se realizará una prueba simple conectando el sensor a un Arduino utilizando la comunicación serie para ver en el ordenador si el sensor está funcionando correctamente o tiene algún tipo de fallo y este es el motivo por el cuál las medidas que estamos tomando son tan cercanas al 0 y no varían en función de la energía que consume en cada momento el aparato al que está conectado el sensor.

Número de prueba	4
Objetivo de la prueba	Comprobar funcionamiento del sensor de corriente.
Procedimientos a llevar a cabo	<ul style="list-style-type: none"> • Montar diseño del circuito en el propio Arduino. • Programar un firmware sencillo. • Flashear el firmware en el Arduino. • Visualizar resultados en la pantalla del ordenador.
Resultado deseado	Visualizar resultados lógicos dependiendo de que aparato esté conectado al sensor.
Resultado obtenido	Los resultados que vemos en pantalla son siempre 0 o demasiado cercanos al 0, además son siempre iguales independientemente de la electricidad que esté consumiendo el aparato en cada momento.
Errores	Los resultados obtenidos de la medición no son correctos ya que son siempre iguales independientemente de la electricidad que esté consumiendo el aparato en cada momento.
Estado de la prueba	Error.
Posibles soluciones	Diseñar un prototipo con un amplificador para conseguir valores mayores.

Tabla 7.4: Prueba 4.

- **Prueba 5:** Para comprobar si el problema de las mediciones viene dado por que la señal que estamos midiendo es demasiado baja. El diseño de este circuito corresponde con el visto en el capítulo 4 (4.6).

Número de prueba	5
Objetivo de la prueba	Comprobar funcionamiento del prototipo (4.6).
Procedimientos a llevar a cabo	<ul style="list-style-type: none"> • Montar diseño del circuito en la protoboard. • Programar un firmware que envíe los datos a un servidor. • Flashear el firmware en la placa NodeMCU. • Visualizar resultados que se muestran en el servidor.
Resultado deseado	Visualizar resultados lógicos dependiendo de que aparato esté conectado al sensor.
Resultado obtenido	Los resultados obtenidos de la medición no son correctos ya que son siempre iguales independientemente de la electricidad que esté consumiendo el aparato en cada momento.
Estado de la prueba	Error
Posibles soluciones	Colocar el sensor de corriente rodeando tan solo uno de los cables (una fase) que tiene la manguera.

Tabla 7.5: Prueba 5.

- **Prueba 6:** Se probará de nuevo el prototipo (4.5) pero en esta ocasión el sensor de corriente se colocará al rededor de tan solo uno de los tres cables (una fase) de la manguera que contiene los tres cables.

Número de prueba	6
Objetivo de la prueba	Comprobar funcionamiento del prototipo (4.5).
Procedimientos a llevar a cabo	<ul style="list-style-type: none"> • Montar diseño del circuito en la protoboard. • Programar un firmware para visualizar los datos por pantalla. • Flashear el firmware en la placa NodeMCU. • Visualizar resultados que se muestran en la pantalla.
Resultado deseado	Visualizar resultados lógicos dependiendo de que aparato esté conectado al sensor.
Resultado obtenido	Los resultados son lógicos con respecto a los consumos que está haciendo el aparato al que está conectado el sensor.
Estado de la prueba	Correcto.
Posibles soluciones	

Tabla 7.6: Prueba 6.

- **Prueba 7:** Comprobación del correcto funcionamiento del prototipo (4.7) con un firmware que calcule el irms y que muestre la información a través de la pantalla LCD. Además se comparará la información obtenida utilizando el prototipo con la obtenida utilizando un sensor comercial.

Número de prueba	7
Objetivo de la prueba	Comprobar funcionamiento del prototipo (4.7).
Procedimientos a llevar a cabo	<ul style="list-style-type: none"> • Montar diseño del circuito en la protoboard. • Programar un firmware para visualizar los datos por pantalla. • Flashear el firmware en la placa NodeMCU. • Visualizar resultados que se muestran en la pantalla. • Comparar con los datos recogidos por el sensor comercial.
Resultado deseado	Visualizar los mismos resultados que los que obtendríamos utilizando el sensor comercial.
Resultado obtenido	Los resultados son lógicos con respecto a los consumos que está haciendo el aparato al que está conectado el sensor, pero no son iguales a los datos que visualizamos en el sensor comercial.
Estado de la prueba	Error.
Posibles soluciones	Calibrar el sensor.

Tabla 7.7: Prueba 7.

- **Prueba 8:** Comprobación del correcto funcionamiento del prototipo (4.7) con un firmware que calcule el irms y que muestre la información a través de la pantalla LCD. Además se comparará la información obtenida utilizando el prototipo con la obtenida utilizando un sensor comercial. Con la diferencia de que en esta ocasión se añadirá en el firmware una variable que nos permitirá calibrar el sensor.

Número de prueba	8
Objetivo de la prueba	Comprobar funcionamiento del prototipo (4.7).
Procedimientos a llevar a cabo	<ul style="list-style-type: none"> • Montar diseño del circuito en la protoboard. • Programar un firmware para visualizar los datos por pantalla. • Flashear el firmware en la placa NodeMCU. • Visualizar resultados que se muestran en la pantalla. • Comparar con los datos recogidos por el sensor comercial.
Resultado deseado	Visualizar los mismos resultados que los que obtendríamos utilizando el sensor comercial.
Resultado obtenido	La información visualizada utilizando el prototipo coincide con la que observamos en el sensor comercial.
Estado de la prueba	Correcto.
Posibles soluciones	

Tabla 7.8: Prueba 8.

- **Prueba 9:** Revisión del firmware para poder mandar los datos en formato JSON a un servidor.

Número de prueba	9
Objetivo de la prueba	Enviar la información a un servidor.
Procedimientos a llevar a cabo	<ul style="list-style-type: none"> • Adaptar el firmware para enviar datos especificando la IP del servidor. • Flashear el firmware en la placa NodeMCU. • Comprobar que los datos se reciben de forma correcta en el servidor.
Resultado deseado	Los datos se reciben en el servidor sin ningún tipo de problema.
Resultado obtenido	El servidor recibe los datos de la manera esperada.
Estado de la prueba	Correcto.
Posibles soluciones	

Tabla 7.9: Prueba 9.

Capítulo 8

Experimentación

En este capítulo se recogerán algunos experimentos realizados para cerciorarnos de que las medidas recogidas gracias al aparato diseñado se corresponde a medidas reales.

8.1. Comparativas con aparatos reales

Para poder calibrar el dispositivo y ver si las medidas recogidas son adecuadas lo que se ha hecho ha sido comparar las mediciones de distintos aparatos utilizando un medidor de consumos eléctricos del mercado, en concreto el modelo más simple de eFergy el cual se encuentra entre los dispositivos que podemos encontrar en el mercado analizados en el capítulo 1.

Una vez calibrado el dispositivo podemos ver en la siguiente tabla que los resultados obtenidos mediante el monitor de energía comercial y el que hemos diseñado nosotros las diferencias son mínimas:

Aparato a medir	Medida Eergy	Medida con nuestro sensor
Bombilla 40 W	41.2 W	41.14 W
Bombilla 60 W	59.8 W	59.76 W
Play Station	47 W	51.48 W
Ventilador	35.4 W	35.32 W
TV	16.7 W	17.52 W
Secador a plena potencia	1902 W	1910.52 W
Secador media potencia	983 W	984.58 W
Secador baja potencia	530 W	534.23 W

Tabla 8.1: Comparación consumos.

8.2. Prueba consumo constante

En este apartado comprobaremos que el aparato funciona correctamente midiendo el consumo constante durante un periodo de tiempo de una bombilla de 60 W este consumo se empezará a medir empezando con la bombilla apagada y finalmente con la bombilla apagada también por ese motivo la primera y última muestra son 0. Estas muestras se han recogido por comodidad conectando la placa directamente al puerto USB del ordenador y abriendo el monitor serie (8.1). Podremos ver estos resultados más claramente haciendo una gráfica en una hoja de cálculo (8.2).

```

COM3
*****
IRMS:
0.27
Potencia:
59.80
*****
IRMS:
0.27
Potencia:
59.69
*****
IRMS:
0.27
Potencia:
59.73

```

Figura 8.1: Monitor serial Arduino IDE.

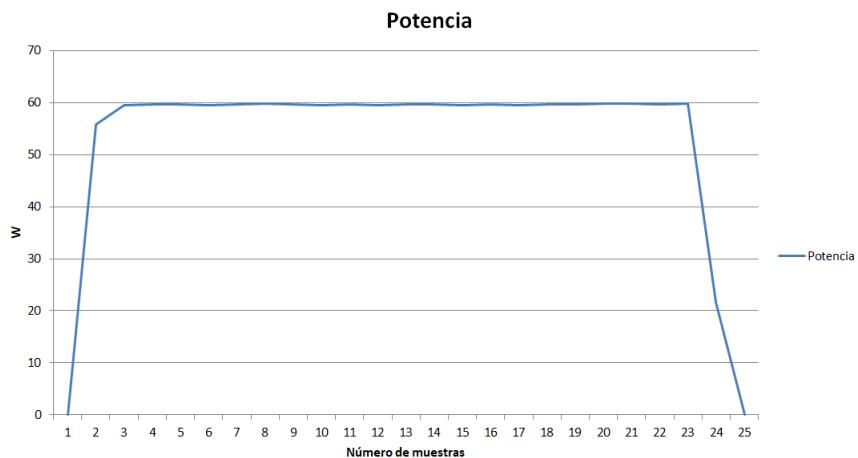


Figura 8.2: Gráfico muestras recogidas.

8.3. Prueba con un electrodoméstico real

Para esta prueba se ha conectado el sensor a un congelador y se han recogido medidas durante la noche obteniendo los siguientes resultados (8.3). Además podemos observar como justo en el momento en el que el compresor

del congelador vuelve a conectarse (Cuando detecta que ha perdido demasiada temperatura) el consumo se dispara ya que al volver a conectarse es cuando más energía consume.

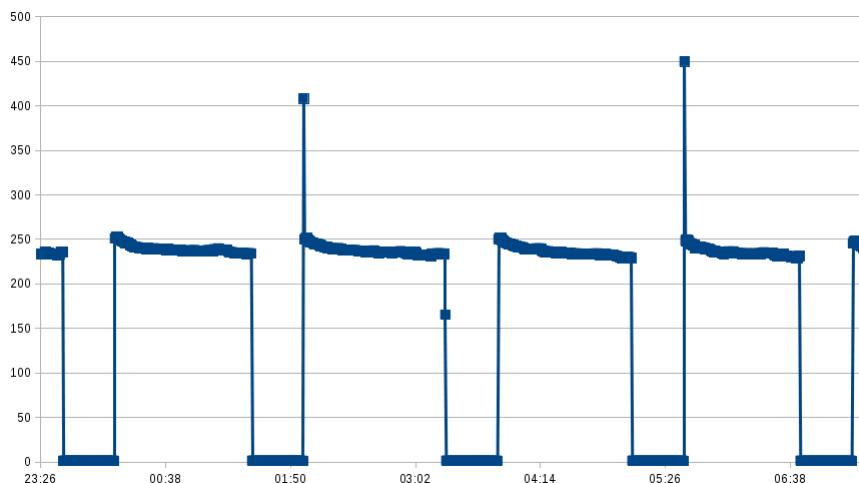


Figura 8.3: Muestras recogidas congelador.

Si analizamos los resultados más en detalle vemos que donde se produce un cambio significativo de la potencia registrada corresponde al tiempo que el compresor del congelador está activo y el tiempo que está apagado siempre es el mismo, como podemos ver con estos datos obtenidos directamente:

Hora	Medida	Estado	Duración
31/08/2017 23:39:12 AM	1.23	Off	0:29
01/09/2017 0:08	1.20	Off	
01/09/2017 0:08	250.90	On	1:35
01/09/2017 1:27	234.00	On	
01/09/2017 1:27	1.21	Off	0:30
01/09/2017 1:57	1.19	Off	
01/09/2017 1:57	408.01 W	On	1:22
01/09/2017 3:19	165.26 W	On	
01/09/2017 3:19	1.20 W	Off	0:30
01/09/2017 3:49	1.19 W	Off	
01/09/2017 3:49	250.04 W	On	1:17
01/09/2017 5:06	229.31 W	On	
01/09/2017 5:07	1.20 W	Off	0:29
01/09/2017 5:36	1.19 W	Off	
01/09/2017 5:37	449.66 W	On	1:06
01/09/2017 6:43	231.48 W	On	
01/09/2017 6:44	1.19 W	Off	0:29
01/09/2017 7:13	1.18 W	Off	
01/09/2017 7:14	245.68 W	On	

Tabla 8.2: Tabla datos

Todos estos resultados han sido obtenidos sin abrir en ningún momento la puerta del congelador. Pero en caso de que se abra la puerta del congelador se perderá frío y tendrá que conectar el compresor antes como podemos ver en esta otra gráfica (8.4) en la que justo en el momento en el que el compresor del congelador se apagó, se dejó abierta la puerta hasta que a los 3 minutos el congelador encendió de nuevo el compresor.



Figura 8.4: Muestras recogidas congelador.

8.4. Posibilidades de explotación de un dispositivo como el presentado

Como hemos visto gracias a la prueba anterior realizada en el congelador gracias a los patrones de consumo que observamos podemos determinar que electrodoméstico o dispositivo es el que está realizando esos consumos. Pero este estudio de los datos de consumo recogidos podría ir a más ya que como hemos observado es posible incluso determinar cuando el usuario ha abierto la puerta del congelador, esta idea puede ser aplicada en otros muchos electrodomésticos. De esta forma podrían extraerse patrones de comportamiento del usuario basados en los consumos de sus electrodomésticos permitiendo saber por ejemplo a que hora suele comer (basado en los cambios de consumo tanto de la vitrocerámica como el frigorífico) o si esta persona se encuentra o no en su hogar basándonos en el consumo eléctrico que se esté realizando en relación con la hora del día que sea.

Esto nos lleva a pensar que una empresa que estuviese interesada en reco-
pilar datos a cerca de los hábitos de consumo de la población como puede ser Google o cualquier otra empresa, podría viendo los datos recogidos detectar que electrodoméstico o aparato eléctrico está en ese momento consumiendo electricidad y podría informar al usuario por ejemplo de cuantas luces tiene encendidas en ese momento en su casa, si está encendido el televisor, etc. Se podrían reducir en gran medida los costes de este dispositivo fabricándolo en masa.

Capítulo 9

Conclusiones y trabajos futuros

Desde el principio la idea de este proyecto ha sido la de realizar algo similar a proyectos que se comercializan pero con el menor coste posible. Esto se consigue gracias al uso de todo hardware libre posible adquiriéndolo al menor coste posible y utilizando software libre. De esta forma además podemos ayudar a más usuarios que estén interesados en realizar este proyecto por su cuenta.

Este proyecto me ha ayudado a adquirir conocimientos tanto relacionados con el mundo de la informática como pueden ser conocimientos impartidos durante el grado, pero en otras ramas distintas a la que yo he cursado como otros conocimiento que no son específicos del campo de la informática los cuales se vieron en el capítulo 2. Además de los adquiridos gracias a la realización de un proyecto grande como este como pueden ser:

- **Cumplimiento de plazos establecidos:** En un primer momento la entrega de este proyecto estaba prevista para el mes de junio, pero por problemas durante algunas de las fases esos plazos no se cumplieron haciendo de esta forma imposible tener el proyecto finalizado para la fecha prevista por lo que se ha tenido que posponer para septiembre. Por lo tanto creo que esto no hubiera sucedido no tanto por establecer unos plazos poco realistas ya que ese aspecto creo que no se pensó del todo mal como por no seguir avanzando en otras tareas a realizar mientras no se avance en otra.
- **Documentarse bien antes de realizar diseños y pruebas:** En muchas ocasiones a la hora de realizar las pruebas de algún diseño no funciona de la manera correcta y en el caso del hardware puede deberse a cualquiera de los múltiples dispositivos hardware que forman el dispositivo completo por lo tanto es importante conocer perfectamente el funcionamiento y como se utiliza cada uno de ellos para poder

detectar errores de una forma sencilla.

- **Documentación al día :** Es importante ir documentando todo lo que se va haciendo en un periodo de tiempo corto ya que si la realización de la documentación se pospone demasiado ya no se tendrán tan frescos los conocimientos que queremos plasmar en la documentación y en muchas ocasiones será necesario volver a revisarlo con el gasto de tiempo extra que esto conlleva.

9.1. Trabajos futuros

Algunas de las mejoras que se pueden realizar en este proyecto son las siguientes:

- Mejorar el firmware para añadir nuevos parámetros como los kilowatios hora consumidos o la media de consumo.
- Introducir un método mediante el cuál se puedan observar los patrones de consumo de cada electrodoméstico permitiendo de esta forma al usuario conocer que dispositivos tiene conectados en cada momento en su hogar.
- Mejorar el servidor web para que le muestre al usuario más información así como incluir soporte para mostrar al usuario las mejoras anteriormente citadas.
- Añadir un sencillo sistema de recomendación de tarifas así como de potencia contratada que permitirá al usuario seleccionar la tarifa y potencia que mejor se adapte a sus necesidades.
- Monitorizar un año entero de consumo eléctrico de un hogar.
- Bigdata que nos permitirá aprender que consumos tiene cada aparto y en base a ese aprendizaje determinar que aparatos están consumiendo electricidad en cada momento.
- Establecer patrones de consumo de un usuario concreto para poder observar sus hábitos en lo que al consumo eléctrico se refiere.

Apéndice A

Anexo

A.1. Como flashear firmware en el ESP8266

En este apartado se explicará lo necesario para poder flashear el firmware alojado en mi github <https://github.com/Miguelmoral/ESP8266-energy-monitor/tree/master/firmware> en la placa ESP8266.

En primer lugar tendremos que descargar el programa Arduino IDE desde la web oficial [4].

Una vez instalada cualquiera de las versiones de Arduino IDE veremos que la placa ESP8266 no se encuentra disponible entre el listado de placas por defecto que nos permite seleccionar el IDE por lo que tendremos que instalar un plugin para utilizar dicha placa. En el menú Archivo/Preferencias veremos un apartado donde podremos escribir llamado Gestor de URLs Adicionales de Tarjetas donde introduciremos esta URL http://arduino.esp8266.com/stable/package_esp8266com_index.json.

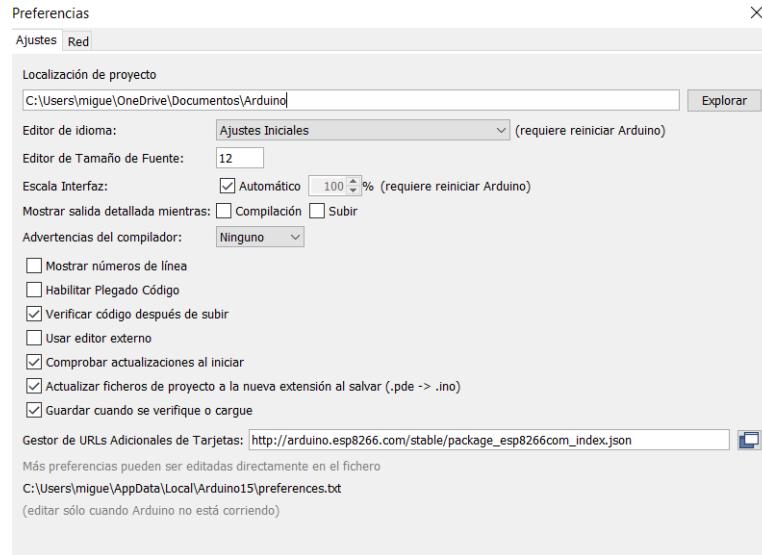


Figura A.1: Archivo/Preferencias

Una vez agregada la URL accedemos a Herramientas/Placa/Gestor de tarjetas y buscamos e instalamos esp8266 by ESP8266 community (A.2).



Figura A.2: Herramientas/Placa/Gestor de tarjetas

En este momento tan solo nos faltará abrir el archivo .ino descargado previamente de github el cuál contiene el firmware actualizando el servidor donde queremos que se manden los datos recogidos por el sensor y las credenciales de nuestra red Wifi para que la placa ESP8266 tenga acceso a la red y pulsar sobre el botón subir para flashear el firmware a la placa.

Bibliografía

- [1] adafruit. Biblioteca Adafruit. https://github.com/adafruit/Adafruit_ADS1X15. Online; accessed 20 agosto 2017.
- [2] Apcom. ohm's law calculator. <http://www.apcom-inc.com/resources/ohms-law-calculator>. Online; accessed 26 agosto 2017.
- [3] arduino. arduino. <https://es.wikipedia.org/wiki/Arduino>. Online; accessed 20 agosto 2017.
- [4] Arduino. Arduino ide. <https://www.arduino.cc/en/main/software>. Online; accessed 23 agosto 2017.
- [5] Atom. Atom. <https://atom.io/>. Online; accessed 23 agosto 2017.
- [6] bblanchon. C++ JSON library for IoT. <https://github.com/bblanchon/ArduinoJson>. Online; accessed 27 agosto 2017.
- [7] Belkin. Belkin Conserve Insight. <http://www.belkin.com/us/p/P-F7C005/>. Online; accessed 25 mayo 2017.
- [8] Bootstrap. Bootstrap. [https://es.wikipedia.org/wiki/Bootstrap_\(framework\)](https://es.wikipedia.org/wiki/Bootstrap_(framework)). Online; accessed 4 junio 2017.
- [9] British Gas. British Gas. <https://www.britishgas.co.uk/smart-home/smart-meters.html>. Online; accessed 2 junio 2017.
- [10] Carter McBride. Cómo calcular la amortización de una computadora portátil. http://www.ehowenespanol.com/calcular-amortizacion-computadora-portatil-como_440173/. Online; accessed 2 septiembre 2017.
- [11] dexterindustrie. Five Ways To Run a Program On Your Raspberry Pi At Startup. <https://www.dexterindustries.com/howto/run-a-program-on-your-raspberry-pi-at-startup/>. Online; accessed 5 junio 2017.
- [12] Django Girls. Django Girls. <https://tutorial.djangogirls.org/es/django/>. Online; accessed 4 junio 2017.

- [13] Efergy. Efergy: Elite Classic. <http://efergy.com/es/medidores/medidoresoffline/medidores/elitev1-monitor>. Online; accessed 25 mayo 2017.
- [14] Efergy. Efergy: Engage medidor de consumo eléctrico online. <http://efergy.com/es/engagekit-hub>. Online; accessed 25 mayo 2017.
- [15] EmonTx. Emontx openenergymonitor.
- [16] Flask. Flask. <http://flask.pocoo.org/>. Online; accessed 4 junio 2017.
- [17] Floureon. Floureon Power Meter Energy Monitor. http://www.floureon.com/product-g_142.html. Online; accessed 25 mayo 2017.
- [18] George Robotics Limited. Micropython. <https://micropython.org/>. Online; accessed 25 agosto 2017.
- [19] GitHub. GNU GPL github. <https://github.com/Miguelmoral/ESP8266-energy-monitor/blob/master/LICENSE>. Online; accessed 4 septiembre 2017.
- [20] GNU. GNU GPL. <https://www.gnu.org/licenses/gpl-3.0.html>. Online; accessed 4 septiembre 2017.
- [21] Gruemaster, tuxinator2009. Win32 disk imager. <https://sourceforge.net/projects/win32diskimager/>. Online; accessed 5 junio 2017.
- [22] Henry. Arduino ADS1115 Module Getting Started Tutorial. <http://henrysbench.capnfatz.com/henrys-bench/arduino-voltage-measurements/arduino-ads1115-module-getting-started-tutorial/>. Online; accessed 20 agosto 2017.
- [23] hreintke. HTTP client. <https://github.com/esp8266/Arduino/tree/master/libraries/ESP8266HTTPClient>. Online; accessed 26 agosto 2017.
- [24] joaopedrosgs. Library for the LiquidCrystal LCD display. https://github.com/fdebrabander/Arduino-LiquidCrystal-I2C-library/blob/master/LiquidCrystal_I2C.h. Online; accessed 27 agosto 2017.
- [25] LUA. Lua wikipedia. <https://es.wikipedia.org/wiki/Lua>. Online; accessed 23 agosto 2017.
- [26] Luis Castro. ¿Qué es HTTP y HTTPS? <https://github.com/esp8266/Arduino/tree/master/libraries/ESP8266HTTPClient>. Online; accessed 27 agosto 2017.

- [27] Marcel Stör. NodeMCU versiones. <https://frightanic.com/iot/comparison-of-esp8266-nodemcu-development-boards/>. Online; accessed 30 mayo 2017.
- [28] Miguel Moral. GitHub ESP8266-energy-monitor. <https://github.com/Miguelmoral/ESP8266-energy-monitor>. Online; accessed 5 junio 2017.
- [29] Neur. Neur solutions. <https://www.neur.io/>. Online; accessed 25 mayo 2017.
- [30] Pocoo. Welcome to Jinja2. <http://jinja.pocoo.org/docs/2.9/>. Online; accessed 7 septiembre 2017.
- [31] Protoboard. Protoboard. https://es.wikipedia.org/wiki/Placa_de_pruebas. Online; accessed 20 agosto 2017.
- [32] Raspberry. Raspberry official. <https://www.raspberrypi.org/blog/raspberry-pi-2-on-sale/>. Online; accessed 30 mayo 2017.
- [33] Raspberry. Web oficial Raspberry. <https://www.raspberrypi.org/downloads/raspbian/>. Online; accessed 30 mayo 2017.
- [34] Rubenfa. MongoDB: qué es, cómo funciona y cuándo podemos usarlo (o no). <https://www.genbeta.dev/bases-de-datos/mongodb-que-es-como-funciona-y-cuando-podemos-usarlo-o-no>. Online; accessed 5 junio 2017.
- [35] Sense. Sense app. <https://www.bakerelectricsolar.com/blog/2016-11-08/products-we-think-are-cool-%E2%80%93-sense-home-energy-monitor>. Online; accessed 2 junio 2017.
- [36] Sense. Sense official web. <https://sense.com/product.html>. Online; accessed 2 junio 2017.
- [37] SparkFun. SparkFun Datasheet. <https://cdn.sparkfun.com/datasheets/Sensors/Current/ECS1030-L72-SPEC.pdf>. Online; accessed 20 agosto 2017.
- [38] Texas instruments. LM358 DataSheet. <http://www.ti.com/lit/ds/symlink/lm158-n.pdf>. Online; accessed 20 agosto 2017.
- [39] ThePiHut. How to setup a static IP address on your Raspberry Pi. <https://thepihut.com/blogs/raspberry-pi-tutorials/16683276-how-to-setup-a-static-ip-address-on-your-raspberry-pi>. Online; accessed 5 junio 2017.
- [40] Wikipedia. Filtro paso bajo. https://es.wikipedia.org/wiki/Filtro_paso_bajo. Online; accessed 26 agosto 2017.

- [41] Wikipedia. Media cuadrática. https://es.wikipedia.org/wiki/Media_cuadr%C3%A1tica#Valor_eficaz_de_una_corriente_alterna. Online; accessed 26 agosto 2017.
- [42] Wikipedia. NodeMCU Wikipedia. <https://en.wikipedia.org/wiki/NodeMCU>. Online; accessed 30 mayo 2017.
- [43] Wikipedia. Raspberry Wikipedia. https://es.wikipedia.org/wiki/Raspberry_Pi. Online; accessed 30 mayo 2017.
- [44] Wikipedia. Valor eficaz. https://es.wikipedia.org/wiki/Valor_eficaz. Online; accessed 26 agosto 2017.
- [45] yhdc. yhdc current sensor. <http://www.yhdc.us/ES/index.asp>. Online; accessed 20 agosto 2017.

