



ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA

**GRADO EN INGENIERÍA INFORMÁTICA
INGENIERÍA DE COMPUTADORES/TECNOLOGÍAS
INFORMÁTICAS**

ARDUCONTROL

**Realizado por
FERNANDO MÉNDEZ REQUENA**

**Dirigido por
MANUEL JESÚS DOMÍNGUEZ MORALES
ALEJANDRO LINARES BARRANCO**

**Departamento
ARQUITECTURA Y TECNOLOGÍA DE COMPUTADORES**

Sevilla, MARZO, 2016

Agradecimientos:

Un agradecimiento singular al Profesor **D. Manuel Jesús Domínguez Morales** que, como tutor de este proyecto, me ha orientado y corregido mi labor con un interés y una entrega que han sobrepasado con mucho todas las expectativas que como alumno deposité en su persona.

Agradecer a mi familia, en especial a mis padres por el apoyo incondicional, moral y confianza que han depositado en mí durante mi formación académica.

RESUMEN

En la presente memoria, desarrollamos un proyecto basado en la medición de energía de uno o varios dispositivos de una vivienda, en tiempo real, haciendo uso de microcontroladores de bajo coste.

El proyecto actual se puede dividir en tres aspectos claramente diferenciados: en primer lugar, tenemos una aplicación web que hará el papel de monitor de consumo, la cual interactuará con el microcontrolador y en la que podremos visualizar los datos tomados del sistema. En segundo lugar, tenemos la implementación del software de control, para el sistema y su construcción. Por último, resaltar la elaboración de una PCB, en la cual se integran las conexiones de nuestro prototipo.

Tanto el control de los sensores, la captación de datos y su transmisión, se hizo a través de una placa ARDUINO, la cual se basa en microcontroladores de la familia ATMEGA. Gracias a este microcontrolador podremos realizar la medición de datos, almacenamiento y visualización a tiempo real, tanto en la web como en la pantalla LCD. Dichos datos quedan almacenados en una base de datos mySQL, ejecutada en un servidor Apache.

En cuanto a la implementación de la aplicación web, se ha usado; Html5, Php7.0, Javascript, Ajax y el framework Bootstrap. Para los gráficos de la web, se ha usado la librería gráfica que nos proporciona Google, llamada Google charts.

Por último, para el diseño de la PCB, se ha utilizado el programa Fritzing. Nos hemos decantado por su uso ya que es un programa gratuito de diseño, el cual destaca por su simplicidad y la rapidez con la que puedes comenzar a diseñar cualquier tipo de PCB.

ABSTRACT

This manuscript presents an energy-measurement project for a particular home, composed by one or more devices using low cost microcontrollers.

The current Project can be divided into three aspects: first, we have a web application that will monitor the energy consumption, which interacts with the microcontroller and can display the data from it. Next we have a firmware integrated inside the microcontroller used to control the correct function of. Finally, we did a PCB, for better connection and minimize read errors.

Both, control sensors, data and transmission, was made through with Arduino Board, which is based on ATMEGA microcontroller family. We can perform measurement, data storage and data visualization in real time, both on the web or in LCD screen. These data are stored in a mySQL database running in an Apache Web Server.

Regarding the implementation of the Web Application, I used; Html5, Php7.0, JavaScript, Ajax and Bootstrap framework. For graphics on the web, I used graphics library that provides Google, called Google Charts.

Finally, for the PCB layout, I used Fritzing. I opted to use it, because it's a free program design, which stands out simplicity and speed to design any type of PCB you want.

ÍNDICE

1. INTRODUCCIÓN	1
2. OBJETIVOS Y ANÁLISIS DE REQUISITOS.....	2
2.1. Objetivo general	2
2.2. Objetivos específicos.....	2
2.3. Requisitos del proyecto.....	2
3. ESTADO DEL ARTE	4
3.1. ¿Qué es una regleta inteligente? ¿para qué sirve?. .	4
3.2. Medidores de consumo individual	4
3.3. Otros medidores de consumo eléctrico (Self-Made) . .	7
4. EVALUACIÓN DE LAS ALTERNATIVAS	9
4.1. Sensores de corriente.....	10
4.1.1. Sensores de corriente invasivos	10
4.1.2. Sensores de corriente no invasivos	11
4.2. Microcontrolador.....	12
4.2.1. Arduino UNO	12
4.2.2. NetDuino	15
4.3. Transmisión y recepción de datos	17
4.3.1. Ethernet Shield	17
4.3.2. Módulo Bluetooth	18
4.4. Visualización de datos	19
4.4.1. Pantalla TFT	19
4.4.2. Lcd keypad shield	19
4.5. Relés	20
4.6. Resumen elección componentes	20
5. DISEÑO	21
5.1. Captación de corriente.....	22
5.1.1. Sensor de corriente	22
5.1.2. Firmware de captación de corriente	24
5.2. Transmisión de datos.....	27
5.2.1. Arduino Ethernet Shield	29
5.2.2. Firmware de transmisión de datos	29
5.3. Comunicación con microcontrolador	31
5.3.1. Dispositivo Bluetooth	31
5.3.2. Firmware de comunicación con microcontrolador	31
5.4. Diseño prototipo.....	31
5.4.1. Sensor de corriente	32
5.4.2. Bluetooth	35
5.4.3. Relés	37
5.5. Diseño PCB	38
5.5.1. Resultado PCB	39
5.6. Diseño Aplicación Android.....	40
5.6.1. Diagrama de flujo	41
5.6.2. Realización de Mockups	42

5.6.3. AppInventor	42
5.6.4. Desarrollo Interfaz Gráfica	44
5.6.5. Desarrollo Software	45
5.6.6. Android Studio	47
5.6.6.1. ¿Qué es Android Studio?	47
5.6.6.2. Interfaz Gráfica.....	48
5.6.6.3. Desarrollo Software aplicación	55
5.6.7. Firmware completo microcontrolador	63
6.DISEÑO APPLICACIÓN WEB	68
6.1 Diagrama de flujo	69
6.2 Realización de Mockups	69
6.3 Aplicación web	72
6.4 Código aplicación web	78
7.INSTALACIÓN	89
7.1 Mecanización de la caja	89
7.2 Resultado final	91
8.PRUEBAS	92
8.1 Pruebas Iniciales	92
8.2 Pruebas Finales	94
8.3 Comparativa de medidas, gráfica de respuesta	96
8.3.1 Cuantificación de errores	96
8.4 Prueba de comportamiento de la aplicación ..	99
9.COSTE DEL PROYECTO	99
9.1 Coste de personal	100
9.2 Coste de equipamiento	100
10.TEMPORALIZACIÓN DEL PROYECTO	101
10.1 Fases del proyecto	101
10.1.1 Búsqueda de información	101
10.1.2 Estudio del mercado	101
10.1.3 Desarrollo del proyecto	101
10.1.4 Pruebas y errores	102
10.1.5 Instalación	102
10.1.6 Documentación	102
10.2 Análisis Temporal	105
11.MANUAL DE USO	107
12.IMAGEN PROMOCIONAL DEL PRODUCTO	109
13.PROBLEMAS EN EL DESARROLLO DEL PROYECTO	109
14.CONCLUSIONES	111
15.BIBLIOGRAFÍA	112

ÍNDICE DE FIGURAS

Fig.3.2.1 Energy Monitoring Socket.....	5
Fig.3.2.1.1 Efergy Ego Smart Socket.....	6
Fig.3.2.1.2 Interruptor Wemo Insight.....	6
Fig.3.3 Open Energy Monitor.....	7
Fig.3.3.1 SEGMeter.....	8
Fig.3.3.2 Energy Visible.....	8
Fig.4 Diagrama en bloques	9
Fig.4.1.0 Sensor de corriente invasivo.....	10
Fig.4.1.1 Sensor de 30 Amp.....	11
Fig.4.2.1 ATmega328	12
Fig.4.2.1.1 IDE Arduino	13
Fig.4.2.1.2 Arduino Shield.....	14
Fig.4.2.1.3 Características Arduino UNO.....	14
Fig.4.2 Netduino Shield.....	15
Fig.4.3 Ethernet Shield.....	17
Fig.4.3.1 Modulo Bluetooth HC-06.....	18
Fig.4.4.1 LCD 4.3"	19
Fig.4.4.2 LCD Keypad Shield.....	19
Fig.4.5 Módulo Relé 5V.....	20
Fig.5.0 Diagrama bloques diseño	22
Fig.5.1.1 Sensor ACS712	22
Fig.5.1.1.1 Esquema sensor ACS712.....	23
Fig.5.1.1.2 Características Sensor ACS712	23
Fig.5.1.1.3 Diagrama de bloques funcional	24
Fig.5.1.1.4 Símbolo Arduino.....	24
Fig.5.1.1.5 Función para calcular la potencia.....	25
Fig.5.2.1 Ethernet Shield	26
Fig.5.2.2.1 Características Ethernet Shield.....	27
Fig.5.2.2.2 Diagrama bloques funcional	28
Fig.5.2.2.3 Esquema funcionamiento Software	28
Fig.5.2.2.4 Firmware transmisión de datos	29
Fig.5.2.2.5 Firmware Bluetooth.....	29
Fig.5.3.1 Esquema comunicación Bluetooth.....	30
Fig.5.3.1.1 Esquema Módulo Bluetooth HC-06.....	30
Fig.5.3.1.2 Firmware Bluetooth	31
Fig.5.3.1.3 Diagrama de flujo firmware Bluetooth.....	31
Fig.5.4 Símbolo Fritzing.....	32
Fig.5.4.1 Esquema sensor ACS712	32
Fig.5.4.1.1 Tabla conexiones sensor1	33
Fig.5.4.1.2 Tabla conexiones sensor2	33
Fig.5.4.1.3 Vista photoboard sensores.....	34
Fig.5.4.1.4 Vista circuito sensores.....	34
Fig.5.4.2 Esquema Módulo Bluetooth HC-06.....	35
Fig.5.4.2.1 Tabla conexión Bluetooth.....	35
Fig.5.4.2.2 Vista photoboard bluetooth.....	36
Fig.5.4.2.3 Vista circuito bluetooth.....	36
Fig.5.4.3 Conexión relés	37

Fig.5.4.3.1 Vista photoboard relés.....	37
Fig.5.4.3.2 Vista circuito relés.....	37
Fig.5.5 PCB.....	38
Fig.5.5.1 Resultado PCB.....	39
Fig.5.6.1 Diagrama de flujo Aplicación	41
Fig.5.6.2 Mockup Aplicación.....	42
Fig.5.6.3 Editor AppInventor diseño.....	43
Fig.5.6.3.1 Editor AppInventor lógico.....	44
Fig.5.6.4 Interfaz ArduControl.....	44
Fig.5.6.4.1 Vista aplicación Android Móvil.....	45
Fig.5.6.5 Lógica de la aplicación.....	46
Fig.5.6.6.1 Figura símbolo Android Studio.....	47
Fig.5.6.6.1.1 IDE Android Studio.....	48
Fig.5.6.6.2 Esquema Funcionamiento APP:	
Dispositivos.....	48
Fig.5.6.6.2.1 Esquema Funcionamiento APP:	
Panel de control.....	49
Fig.5.6.6.2.2 Creación Activity.....	49
Fig.5.6.6.2.3 TextView - Dispositivos Vinculados.....	50
Fig.5.6.6.2.4 Button - Dispositivos Vinculados.....	51
Fig.5.6.6.2.5 ListView.....	51
Fig.5.6.6.2.6 Resultado Parte I.....	51
Fig.5.6.6.2.7 RelativeLayout.....	52
Fig.5.6.6.2.8 TextView - Panel de control.....	52
Fig.5.6.6.2.9 TextView - Visualizar datos online.....	53
Fig.5.6.6.2.10 Button - Enchufe1 ON / OFF.....	53
Fig.5.6.6.2.11 Button - Enchufe2 ON / OFF.....	53
Fig.5.6.6.2.12 Button - Entrar.....	54
Fig.5.6.6.2.13 Button - Desconectar Bluetooth.....	54
Fig.5.6.6.2.14 Resultado Parte II.....	54
Fig.5.6.6.2.15 Diagrama Clases UML Aplicación.....	55
Fig.5.6.6.3 Declaración atributos.....	55
Fig.5.6.3.3.1 Método onCreate.....	56
Fig.5.6.3.3.2 Método listaDispositivosVinculados	56
Fig.5.6.3.3.3 MyListClickListener.....	57
Fig.5.6.3.3.4 Declaración atributos	59
Fig.5.6.3.3.5 Método onCreate.....	59
Fig.5.6.3.3.6 Método desconectarBluetooth.....	59
Fig.5.6.3.3.7 Método navegacionWeb.....	59
Fig.5.6.3.3.8 Método apagarEnchufe1 /	60
apagarEnchufe2	60
Fig.5.6.3.3.9 Método encenderEnchufe1 /	
encenderEnchufe2	60
Fig.5.6.3.3.10 Clase Interna ConnectBT.....	61
Fig.5.6.3.3.11 AndroidManifest.....	62
Fig.5.6.3.3.12 Diagrama de bloques Firmware Global.....	63
Fig.5.6.3.3.13 Firmware completo microcontrolador.....	64
Fig.6.6.1 Diagrama Flujo Aplicación Web.....	69
Fig.6.6.1.1 Cabecera web.....	69
Fig.6.6.2 Página Inicio Sesión.....	70
Fig.6.6.2.1 Página consumo tiempo real.....	70

Fig.6.6.2.2 Página consumo kW/h	71
Fig.6.6.2.3 Página consumo día actual.....	71
Fig.6.6.2.4 Página consumo mensual.....	72
Fig.6.6.2.5 Página error.....	72
Fig.6.6.3 Vista página Inicio Sesión.....	74
Fig.6.6.3.1 Vista página consumo tiempo real.....	75
Fig.6.6.3.2 Vista página consumo kW/h.....	75
Fig.6.6.3.3 Vista página consumo día actual.....	76
Fig.6.6.3.4 Vista página consumo mensual.....	76
Fig.6.6.3.5 Vista página error.....	77
Fig.6.6.4.1 Código Index.html	78
Fig.6.6.4.2 Código Index.php	79
Fig.6.6.4.3 Código Admin.html	81
Fig.6.6.4.4 Código Actual.php.....	82
Fig.6.6.4.5 Código Actual2.php.....	82
Fig.6.6.4.6 Código consumo kW/h.html	84
Fig.6.6.4.7 Código consumo kW/h.php	84
Fig.6.6.4.8 Código día actual.html	86
Fig.6.6.4.9 Código día actual.php.....	86
Fig.6.6.4.10 Código mensual.html	88
Fig.6.6.4.11 Código mensual.php	88
Fig.7. Vista Frontal	89
Fig.7.1 Vista lateral.....	89
Fig.7.1.2 Vista caja abierta.....	90
Fig.7.1.3 Vista Frontal caja abierta.....	90
Fig.7.2. Resultado final vista frontal	91
Fig.7.2.1 Resultado final vista perfil superior	91
Fig.7.2.2 Resultado final vista trasera.....	91
Fig.7.2.3 Resultado final vista lateral.....	91
Fig.8.1 Polímetro Marca Stanton UT57	92
Fig.8.1.2 Prueba bombilla 18mA	92
Fig.8.2 Prueba bombilla 50W.....	94
Fig.8.2.1 Foco consumo 120W.....	95
Fig.8.2.2 Prueba foco consumo 120W.....	95
Fig.8.3 Gráfica error relativo sensor 5 ^a	97
Fig.8.3.1 Gráfica error relativo sensor 20 ^a	98
Fig.9. Tabla coste del proyecto	99
Fig.10.1.1 Diagrama de fases del proyecto	103
Fig.10.1.2 Proceso de trabajo	104
Fig.10.2.1 Diagrama de Gant	106
Fig.11.1 Página de acceso IPAD	107
Fig.11.2 Página de inicio IPAD	108
Fig.12 Imagen promocional del producto	109
Fig.13.1 Vista Plotly	110
Fig.13.2 Mensaje Error memoria	110

1. INTRODUCCIÓN

A la hora de seleccionar mi trabajo, de entre todas las propuestas que se ofertaban, fueron varios los motivos que hicieron retomar ideas comunes, para un objetivo final, “*ArduControl*”.

Uno de los motivos principales era trabajar por primera vez con plataformas que usan microcontroladores: plataformas programables con un lenguaje fácil de comprender y que disponen de un entorno de desarrollo. Esas plataformas son compatibles con una gran cantidad de sensores; de luz, movimiento, temperatura...

Actualmente en la red se encuentra gran cantidad de proyectos realizados con este tipo de microcontrolador.

Otro de los motivos era trabajar con sistemas empotrados de bajo coste: Estos sistemas empotrados están basados en microprocesadores económicos que posibilitan la instalación de un sistema operativo de software libre como Linux.

Por otro lado, a pesar de nuestro interés por la Domótica, cada vez estamos más habituados a llevar el control de todo lo que hacemos diariamente, ya sean automatizar las tareas, controlar el gasto, en definitiva, hacer más cómoda la vida del usuario.

Es por ello por lo que se decidió a realizar este proyecto, es decir, en diseñar e implementar una regleta inteligente, la cual recopila datos de consumo a través de sus enchufes y poder obtener dichos datos en forma de histórico, a través de una aplicación web. En cuanto a la automatización, se podrán activar / desactivar los enchufes por medio de una aplicación diseñada para Android.

El reto para mí era grande, no conocía ninguna de las plataformas a utilizar ni incluso a diseñar aplicaciones Android, tampoco estaba seguro de poder encontrar ayuda, información y/o manuales suficientes para la materialización del proyecto.

2. OBJETIVOS Y ANÁLISIS DE REQUISITOS

En este apartado, pasaremos a detallar los requisitos y los objetivos que debe cumplir nuestro proyecto, a nivel educativo y profesional.

2.1 Objetivo general

El objetivo de este proyecto consiste en implementar mediante sistemas empotrados low cost, unos enchufes inteligentes, es decir, construir un sistema que nos permita medir el consumo de cualquier aparato eléctrico que esté conectado a nuestro sistema, visualizar en un panel LCD el consumo instantáneo, activar y desactivar a través de una aplicación Android las tomas de corriente de nuestro sistema y monitorizar en una aplicación de usuario, gráficos de consumo instantáneo, kw/h , diario, mensual y anual. Todo esto desarrollado con tecnología de microcontroladores low cost.

2.2 Objetivos específicos

- Medir mediante varios sensores la corriente que circula por cada aparato eléctrico que conectemos, obteniendo la potencia de consumo de cada elemento.
- Mostrar en un display el consumo actual de los elementos conectados.
- Hacer uso de microcontroladores para obtener dicho consumo y comunicación.
- Elaborar una aplicación para el control remoto; activar/desactivar enchufes.
- Almacenar los datos leídos por el sensor de forma remota, y visualizar en una aplicación web por medio de gráficas, el consumo real, acumulado, por hora, mensual, semanal y anual.

2.3 Requisitos del proyecto

Para construir la regleta inteligente, debemos contar con los siguientes requisitos:

- Tantos sensores de corriente, como enchufes que queramos manipular y puertos libres que disponga el microcontrolador.
- Microcontrolador, debe de ser capaz de obtener los datos, mostrarlos en un display LCD, comunicarse con el

dispositivo móvil a través del hardware bluetooth y almacenar los datos en la base de datos. Sin olvidar transformar previamente los datos en valores de corriente y potencia.

- Tantos relés como sensores de corriente. Dichos relés deberán ser manipulados a través de una aplicación Android.
- Transmisor y receptor de datos, realizará la conexión punto a punto. El primer transmisor tendrá comunicación exclusivamente con el usuario, usando tecnología bluetooth a través de la aplicación móvil. El segundo, se encarga el propio microcontrolador, a través de una shield ethernet, la cual se comunicará con la base de datos para almacenar los datos procesados.
- Almacén de datos. En nuestro caso será un disco duro. Su capacidad nos debe permitir almacenar gran cantidad de datos e instalar un software de gestión de datos, creación de tablas e instalación de servidor web.
- Monitor, display lcd y pantalla

En definitiva, nuestra regleta inteligente, estará formado por dos partes fundamentales:

- Estación base, formada por los sensores, relés, microcontrolador, display lcd y la parte transmisora de datos.
- Estación remota, formada por la parte receptora de datos, base de datos y servidor web.

3. ESTADO DEL ARTE: REGLETA INTELIGENTE

En este capítulo definiremos regleta inteligente como un monitor de energía individual. Para ello realizaremos una comparación de los distintos medidores existentes en el mercado, así como los diferentes proyectos relacionados.

3.1 ¿Qué es una regleta inteligente? ¿Para qué sirve?

Una regleta inteligente, definido como monitor de energía, son aparatos muy fáciles de usar con los que podemos conocer nuestro consumo instantáneo, el consumo real de la energía que nuestro dispositivo está utilizando y de donde proviene. Los monitores más sofisticados nos presentan gráficas del consumo histórico e incluso nos permite, obtener una descarga de datos.

Destacar que, tras realizar una búsqueda de los monitores de energía existentes en el mercado, se han podido encontrar muchos modelos de monitores de consumo eléctrico para el hogar, y no todos ellos son compatibles con la red eléctrica española (tipo de enchufe, frecuencia, etc..).

Se ha hecho una recopilación de los modelos disponibles y de sus características, en concreto, los medidores de consumo individual, es decir, aquellos que quieren conocer el consumo de algún aparato.

3.2 Medidores de consumo individual.

Los monitores de energía individual se conectan directamente al enchufe y nos permiten conectar los aparatos que queremos hacer el seguimiento en nuestro propio enchufe

Es muy útil si se quiere conocer el consumo particular de algún aparato y una forma de identificar cuáles son los electrodomésticos menos eficientes. También es la mejor forma de identificar los electrodomésticos en espera (cuando se encuentran en stand -by) .

Aparato eléctrico en Stand-by	Consumo
Ordenador fijo	12 W
TV 24" + TDT	4.6 W
Horno de cocina	2.9 W
Microondas	2.5 W
TV LCD 47"	4.2 W
Calentador	3.5 W
Total consumo	29.7 W

El consumo total, de una vivienda equipada con los aparatos de la tabla, es de 29.7W.

Resaltar que éstos medidores no permite saber el consumo de todo lo que está conectado a la red eléctrica de casa por lo que habrá un consumo que no podremos medir con estos aparatos. En resumen, estos instrumentos sólo miden consumos de electrodomésticos que tienen en un enchufe.

Ejemplo de uso:

- Medición del consumo del ordenador y periféricos y su consumo en stand-by.
- Seguimiento del consumo de nuestros aparatos de video y audio.
- Medición del consumo individual de cada electrodoméstico: Lavadora, frigorífico, horno de cocina, etc.

Ejemplos de algunos de estos monitores se muestran en la siguiente página:

Energy monitoring socket 2.0

Conecta nuestros aparatos y evalúan como de eficientes son, mostrando tanto la cantidad de energía que están consumiendo, (Fig.3.2.1) como también el coste de dicho consumo [1]. En una gran pantalla muestra los parámetros:

- Vatio (W)
- Consumo (kW/h)
- Tensión (V)
- Corriente (A)
- Frecuencia de red (Hz)



Fig.3.2.1 Energy Monitoring Socket

Características:

- Muestra información de día y hora, coste y parámetros de energía
- Posibilidad de seleccionar la tarifa horaria
- Bajo consumo
- Precio: 24.90€

Efergy Ego Smart Socket

(Fig.3.2.1.1) Además de evaluar los aparatos que conectemos, dispone de una app gratuita la cual cumple las siguientes funciones [2]:

- Controlar los electrodomésticos, en remoto
- Monitorizar consumo en tiempo real
- Analiza los costes históricos de electrodomésticos.
- Programar temporizadores de encendido / apagado
- Evitar el consumo stand - by.
- Precio: 47.90 €



Fig.3.2.1.1 Eergy Ego Smart Socket

Interruptor Wemo Insight

Básicamente, es un enchufe que cumple la función de interruptor (Fig.3.2.1.2). [3] No dispone de ningún display lcd, pero incorpora una aplicación móvil, Wemo App disponible para Android e IOS, gratuita, la cual cumple las características anteriormente comentadas a excepción de:

- Posibilidad de crear programas y recibir notificaciones.
- Sistema por módulos, posibilidad de añadir interruptores Wemo adicionales, de forma fácil.
- Funciona con IFTTT, red WI-FI doméstica y red móvil (3G/4G).
- Precio: 59.99€



Fig.3.2.1.2 Interruptor Wemo Insight

3.3 Otros medidores de consumo eléctrico (self-made)

En Internet nos podemos encontrar varios proyectos de construcción de medidores de consumo, basados en sistemas empotrados, plataformas de software libre y controladores asequibles.

Un ejemplo de algunos de ellos lo podemos ver a continuación:

Open Energy Monitor

Monitorización y control de la energía a través de Arduino (Fig.3.3) :

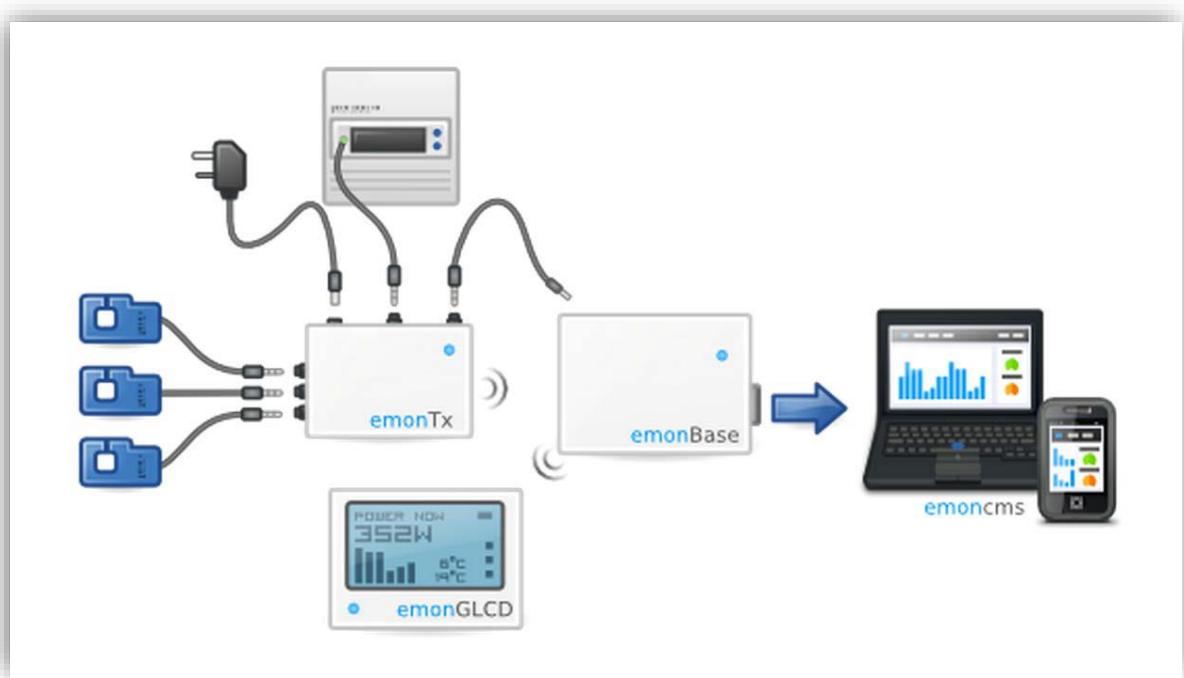


Fig.3.3 Open Energy Monitor

“Un proyecto para desarrollar herramientas de monitoreo de energía de fuente abierta que nos ayudan a relacionarnos con nuestro uso de la energía, los sistemas de energía y el desafío de la energía sostenible”. [4]

SEGMeter

También se trata de un diseño open-source basado en Arduino, aunque algo más elaborado que el anterior ya que además integra la conexión inalámbrica mediante ZigBee, la comunicación con Internet, la visualización de datos a través de un sitio web e incluye un sensor de temperatura. [5]



Fig.3.3.1 SEGMeter

Energy Visible

Por último, nos encontramos con un estudio experimental, desarrollado en el laboratorio Bits2Energy del instituto ETH de la universidad de Zurich. [6]

El proyecto consiste en enchufar medidores individuales de consumo eléctrico en los enchufes, los cuales están conectados a electrodomésticos de distinta índole. A continuación, se capturan los datos y se transfieren de forma inalámbrica la información a un servidor web, el cual, agrupa los datos para visualizarlos mediante una aplicación web.

En este proyecto los autores (*Dominique Guinard, Vlad Trifa, Matthias Kovatsch*), usaron los medidores Ploggs [7] de Energy Optimizers. Estos medidores están provistos de un trasmisor bluetooth que envía la información a un servidor, dónde está disponible para consultarse a través de un navegador.

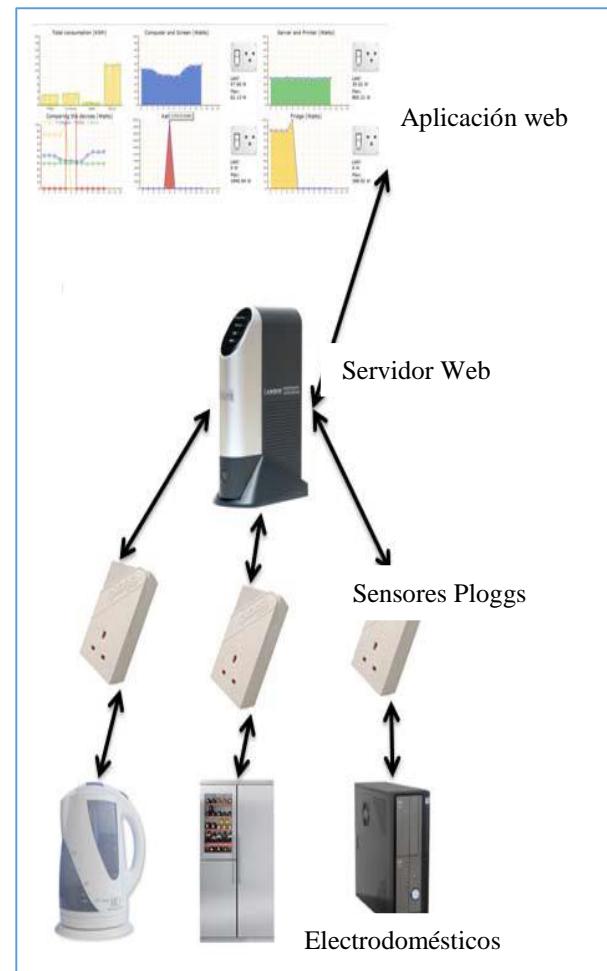


Fig.3.3.2 Energy Visible

4. EVALUACIÓN DE LAS ALTERNATIVAS

Uno de los principales requisitos para la evaluación de los componentes, que utilizaremos en nuestro proyecto, es el coste. Se deberá tener en cuenta su bajo coste, ya que el proyecto consiste en implementar una regleta inteligente, a un precio no muy elevado. Es por ello que decidimos utilizar módulos que estaban a nuestra disposición en el Departamento de Arquitectura y Tecnología de computadores de nuestra escuela.

A continuación, vamos a hacer un análisis de las distintas opciones disponibles en el mercado para cada una de las partes que conforman el sistema (Fig.4):

- Estación base, que realizará las medidas, mostrará la medida actual en un display y almacenará los datos procesados en una base de datos
- Estación remota, presentará los datos en una aplicación web.

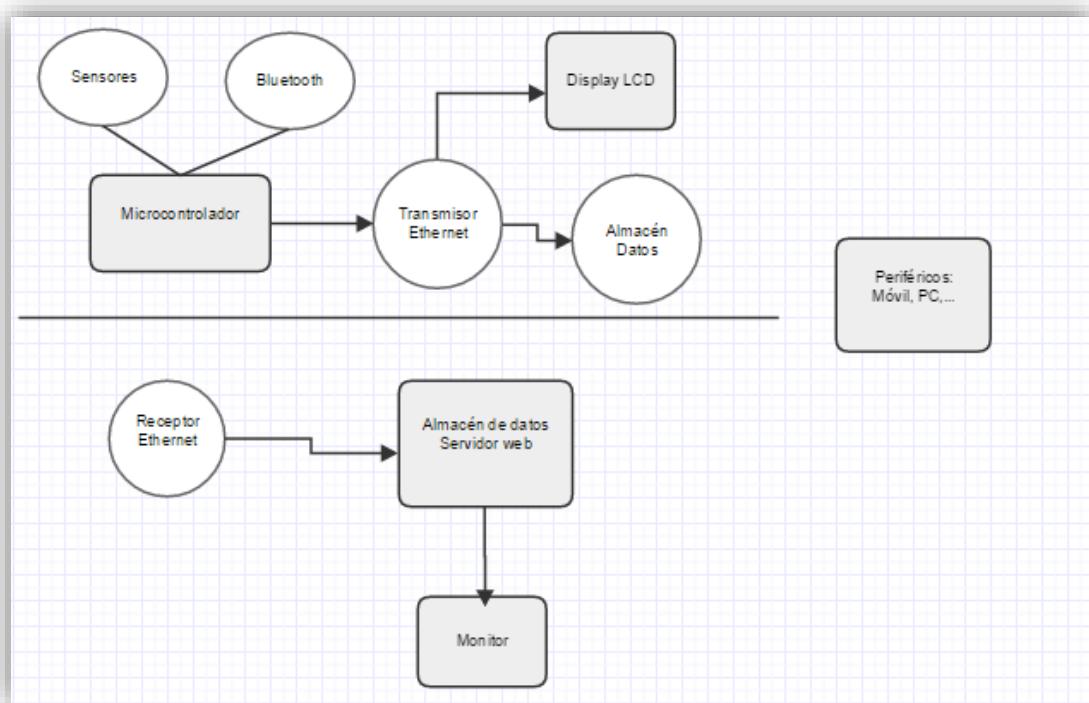


Fig.4 Diagrama en bloques

El diseño del proyecto está basado en 3 partes; la primera consiste en captar los datos, procesarlos y almacenarlos en una base de datos, las cuales tendrá una tabla de consumo. Por otro lado, tenemos una aplicación Android, la cual proporciona un control remoto sobre el sistema, activando y desactivando el estado de los enchufes.

Por último, una tercera parte que se encargará de la visión de los mismos en una aplicación web.

Elección de componentes:

Una vez definidos los requisitos a cumplir, vamos a realizar la evaluación de los posibles elementos/plataformas que pudieran usarse en el diseño.

4.1 Sensor de corriente

La primera y más importante elección que debemos tomar es el sensor de corriente. Dicho sensor, nos permitirá medir la corriente alterna.

Existen varios tipos de sensores; invasivos y no invasivos. En nuestro caso, nos da igual si es invasivo o no, ya que tendremos que construir y manipular los enchufes de nuestro sistema.

Los sensores de corriente son muy útiles para medir el consumo de la electricidad. Nuestro objetivo aquí es medir el consumo de electricidad de cualquier aparato electrónico que conectemos a nuestra regleta, es por ello que tenemos que buscar un sensor que soporte un rango grande de intensidades de corriente.

Estudiamos algunos de los sensores disponibles en el mercado, que se ajusten a nuestras necesidades y que estén en un precio admisible.

4.1.0 Sensores de corriente invasivos

Este sensor (Fig 4.1.0) proporciona una medición casi exacta, según el aparato que estemos midiendo, es decir, deberemos usar el sensor adecuado respetando su amperaje máximo de 5A, 20A o 30A. [8]

Hay que tener en cuenta que se comete un error si usamos aparatos de poco consumo en chips que soportan gran amperaje. Este chip funciona tanto para AC y DC.

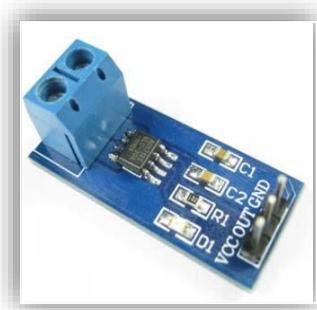


Fig.4.1.0 Sensor de corriente invasivo

Se conecta en serie con la carga a través de los dos bornes. El dispositivo se alimenta con una tensión de 5V. El cálculo de salida se realiza a través del llamado Efecto - Hall, es decir, la corriente que se desea medir, genera un campo magnético que el sensor convierte en un voltaje proporcional en su salida. Este voltaje a su vez será leído por el microcontrolador a través de un convertidor para obtener su valor pico y mediante operaciones aritméticas calcular el valor RMS de la corriente de carga.

4.1.1 Sensores de corriente no invasivos

Por otro lado (Fig.4.1.1) tenemos la categoría de sensores de corriente no invasivos. Este tipo de sensores son más adecuados para el uso de bricolaje, su facilidad de enganche a uno de los cables de entrada de tensión, sin tener que hacer ningún trabajo sobre el circuito.

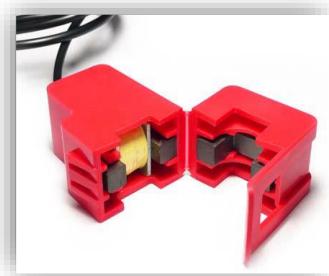


Fig.4.1.1 Sensor de 30 Amp

Aplicaciones:

- Adecuado para la medición de corriente.
- Vigilancia y protección de motores de corriente alterna
- Equipos de iluminación

Sensor Elegido:

Al ser un proyecto en construcción, y que no depende de la instalación eléctrica del domicilio, podríamos haber usado ambos tipos de sensores. El problema se observó en que la pinza amperimétrica necesita un cable con un grosor adecuado para que el valor medido sea exacto. Por otro lado, tampoco nos hace falta un sensor que nos proporcione un rango máximo de 100Amp. Es por ello que nos decantamos por el sensor del punto 4.1.1, llamado ACS-712, suficiente para medir el consumo de los electrodomésticos que estén conectados a nuestro sistema, modelos que soportan 5Amps y 20Amps.

4.2 Microcontrolador

Para la elección del microcontrolador usado en el proyecto, hemos visto algunas plataformas de hardware libre basada en el microcontrolador ATMEGA328 (Fig.4.2.1), microcontrolador asequible y de fácil programación.

Hemos estudiado y comparado algunas plataformas hardware libre programables en un lenguaje de programación no muy complejo de usar. Algunas de estas plataformas son:

4.2.1 Arduino UNO

Arduino es una plataforma de hardware libre, existe un modelo que usa el microcontrolador Atmega328, Arduino Uno, con varios puertos digitales y analógicos de entrada y salida, a través de los cuales podemos leer información detectada utilizando distintos sensores (de temperatura, proximidad, humedad, luminosidad, magnetismo, corriente, coordenadas de un GPS, etc.), mostrar información o notificaciones (usando LEDs, displays LCD, altavoz, etc.) o hasta interactuar con el usuario (botones, joysticks, finales de carrera etc.).

El entorno de desarrollo Arduino está constituido por un editor de texto para escribir el código, un área de mensajes, una consola de texto, una barra de herramientas con botones para las funciones comunes, y una serie de menús. Permite la conexión con el hardware de Arduino para cargar los programas y comunicarse con ellos. (Fig.4.2.2.1)



Fig.4.2.1
ATmega328

Arduino puede comunicarse con otro hardware a través del puerto serie integrado en la placa, que el ordenador, en el entorno de desarrollo, emula este puerto mostrando los datos en el monitor serie, (Pantalla de PC).

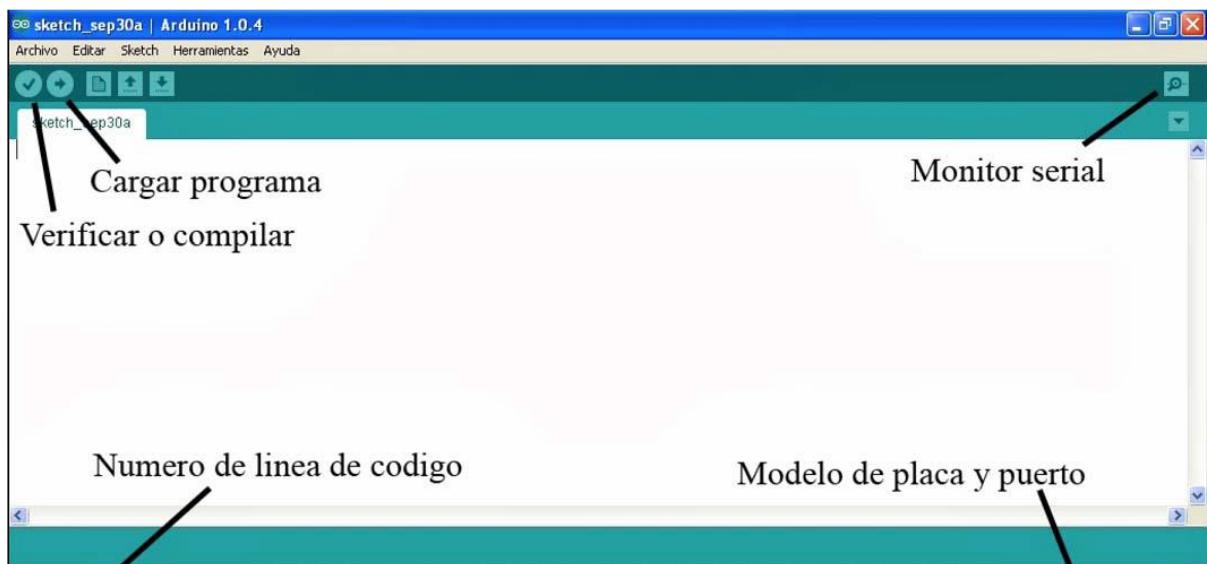


Fig.4.2.2.1 IDE Arduino

Arduino utiliza para desarrollar el software lo que denomina "sketch" (*firmware*). Estos firmwares son programados en el editor de texto. Existe la posibilidad de cortar/pegar y buscar/reemplazar texto.

En el área de mensajes se muestra información mientras se cargan los programas y también muestra errores. La consola muestra el texto de salida para el entorno de Arduino incluyendo los mensajes de error completos y otras informaciones. La barra de herramientas permite verificar el proceso de carga, creación, apertura y guardado de programas, la monitorización serie, que permite ver en la pantalla del ordenador conectado a través del USB, las órdenes que en programación se enviaron a imprimir en el puerto serie.

Existen varios modelos de Arduino, con diferentes prestaciones, el modelo Arduino Uno puede cumplir con las exigencias del proyecto. [9]

El microcontrolador en la placa Arduino Uno (Fig.4.2.1.2) se programa mediante el lenguaje de programación Arduino (basado en Wiring) y el entorno de desarrollo Arduino (basado en Processing).

Una vez cargado el programa, y este se ejecuta, no es necesario volverlo a cargar para su ejecución. El firmware se ejecuta cada vez que se inicia el arranque de la placa.

Una de las grandes ventajas de esta placa es que se le pueden acoplar una serie de placas de expansión denominadas "shields", que se conectan por la parte superior de Arduino para extender sus capacidades, como la conexión Ethernet, WiFi, GPRS o matrices de LEDs. Otra gran ventaja a destacar es la gran comunidad de usuarios que han trabajado con ella. (Fig 4.2.1.3) Precio: 22.00 €



Fig.4.2.1.2 Arduino Shield

Microcontrolador	ATmega328
Alimentación	5V
Voltios de entrada (recomendado)	7-12V
Voltios de entrada (límites)	6-20V
Entradas y/o salidas digitales	14 (6 tiene salidas PWM)
Entradas analógicas	6
Corriente máxima por entradas/salidas	40mA
Corriente máxima para la entrada 3.3V	50mA
Memoria Flash	32KB (0.5KB usados por el sector de arranque)
Memoria SRAM	2KB
Memoria EEPROM	1KB
Velocidad de reloj	16MHz

Fig.4.2.1.3 Características Arduino UNO

4.2.2 NetDuino

Es una placa de aspecto similar a la de Arduino Uno, aunque de mayores prestaciones y menor consumo, es superior tanto de memoria como de velocidad de reloj, en periféricos, buses. El ADC es mucho mejor pues incluye una resolución de 12 bits, a diferencia del ATMega que son de 8 bits. [\[10\]](#)

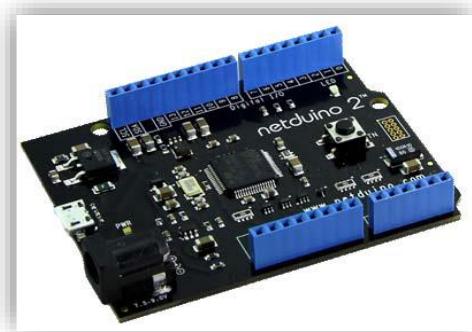


Fig. 4.2 Netduino Shield

Actualmente, se puede cambiar la resolución ACD de nuestro Arduino usando la librería "[erCaGuy NewAnalogRead](#)", desarrollada por Gabriel Staples [\[11\]](#)

Netduino (Fig. 4.2) está basada en la idea de Arduino o viceversa.

El principal inconveniente de esta placa es que el Microcontrolador está soldado a la placa, por lo que en caso de avería del mismo tendríamos problemas en la sustitución. Si lo comparamos con Arduino Uno, el microcontrolador ATMEGA 328 viene sobre zócalo, con lo que la sustitución del micro es muy fácil.

Otro de los inconvenientes o problemas que presenta la plataforma Netduino, es que tiene poca variedad de modelos, limitando así sus posibles aplicaciones en el mercado.

Tanto Netduino como Arduino son compatibles con casi todos los sensores y dispositivos que ya existen en el mercado para uno u otro. Hay algunos que no son compatibles, pero estaríamos hablando entre un 10-15%.

Aunque Netduino es más potente que la placa de Arduino, hay que tener en cuenta que, se programa en un lenguaje de alto nivel (como C# ó VB.NET de Netduino contra C ó C++ de Arduino) es por ello que el MicroFramework hay que integrarlo en la memoria flash.

Características de Netduino:

- Microcontrolador Atmel 32-bits
- Velocidad: 48Mhz
- Memoria para programa: 128 KB
- Memoria RAM: 60KB
- Alimentación: 7.5 - 12 VDC o USB
- Corriente por pin: 25 mA max.
- Pines I/O a 3.3V pero tolerantes a 5V, 14 digitales y 6 analógicos

Sobre esta placa encontramos menos información que sobre la placa Arduino Uno en la red, ya que hay menos comunidad trabajando con ella. Actualmente, podemos encontrar esta placa a un precio que ronda los 52 euros.

Actualmente con respecto a rendimiento el competidor por autonomía es la placa "ST Núcleo F411". Incorpora la misma disposición pinout que Arduino, con un micro de 32bits a 100 mHz, y con una unidad de punto flotante ARM CORTEX m4, lo cual añade una mejor precisión en comparación con las placas anteriormente comentadas. [\[11\]](#)

Microcontrolador elegido:

Nos hemos decidido por utilizar en nuestro monitor de energía la placa Arduino Uno por ser la plataforma de hardware libre más difundida, con más librerías desarrolladas de alto nivel para componentes (sensores, botones, displays, etc.) y más usada, por ser libre y por su sencillez. Otro motivo más a añadir es porque es la primera vez que nos enfrentamos a un proyecto el cual está orientado a trabajar con microcontroladores y con una gran comunidad detrás en la red

4.3 Transmisión y recepción de datos

En este apartado trataremos los elementos elegidos en nuestro proyecto y que permiten la comunicación entre Arduino y el resto de componentes que forma nuestro sistema. [11]

El hardware que hemos empleado para la trasmisión de datos capturados por el sensor y tratados por el microcontrolador ha sido una placa Ethernet (Fig.4.3), por su sencillez en la programación y coste. Para la comunicación con la app Android hemos utilizado un módulo Bluetooth HC-06.

Un aspecto a mejorar en el futuro en nuestro sistema, es la incorporación y sustitución de la conexión cableada por una inalámbrica.

Ethernet Shield

Esta placa permite conectar Arduino a una red LAN, utilizando la librería propia Ethernet incluida en el IDE de Arduino, ventaja frente al módulo WIFI.

Otra ventaja a frente a usar módulo WIFI es que su precio es más económico.

Es compatible con casi todos los modelos de Arduino, ya que se conecta por SPI. Se alimenta a 5V, incorpora un controlador Ethernet W5500 con 32K de buffer interno y trabaja con velocidades de conexión: 10/100Mb. Por último, decir que incorpora una ranura para tarjeta MICRO-SD, la cual podríamos almacenar de datos.



Fig.4.3 Ethernet Shield

4.3.1 Módulo bluetooth

El Bluetooth [12] es un estándar de comunicación inalámbrica que permite la transmisión de datos a través de radiofrecuencia en la banda de 2,4 GHz. Existen dos modelos de módulos Bluetooth: JY-MCU, el HC-05 que puede ser maestro/esclavo, y el HC-06 que solo puede actuar como esclavo. (Fig.4.3.1)

La diferencia entre maestro y esclavo es que en modo esclavo es el dispositivo quien se conecta al módulo, mientras que en modo maestro es el módulo quien se conecta con un dispositivo.

Físicamente, los dos módulos son muy parecidos, solo varían algunas conexiones. Los pines que encontraremos son los siguientes:

- **VCC:** Alimentación del módulo entre 3,6V y 6V.
- **GND:** La masa del módulo.
- **TXD:** Transmisión de datos.
- **RXD:** Recepción de datos a un voltaje de 3,3V.
- **KEY:** Poner a nivel alto para entrar en modo configuración del módulo (solo el modelo HC-05)
- **STATE:** Para conectar un led de salida para visualizar cuando se comuniquen datos.

Es posible configurar el módulo bluetooth gracias a los comandos AT. Los comandos AT son un tipo de comandos que sirven para configurar el módulo Bluetooth a través de un microcontrolador, un ordenador o con cualquier dispositivo que posea una comunicación serie (Tx/Rx). Son unas instrucciones que nos permiten cambiar los baudios del módulo, el PIN, el nombre, etc. Para usar los comandos AT el módulo Bluetooth no debe estar vinculado a ningún dispositivo (led rojo del módulo parpadeando). Según las especificaciones del módulo, el tiempo que se tiene que respetar entre el envío de un comando AT y otro tiene que ser de 1 segundo. Si se envía un comando AT y en menos de un segundo se envía otro, el módulo no devuelve respuesta.



Fig.4.3.1 Módulo Bluetooth HC-06

4.4 Visualización de datos

Para la elección del hardware de visualización, hemos recopilado los siguientes elementos:

4.4.1 Pantalla LCD 4.3"

Pantalla LCD, táctil con una resolución de 480 x 272 compatible con Arduino. Su interfaz LCD incorpora 24 bits en paralelo y 4 hilos para su panel táctil. (Fig.4.4.1) Entre las características más importantes destacan las siguientes:

- Luz de fondo LED
- Contraste 500:1
- Brillo (cd /m) 280
- Tamaño de la pantalla (mm) 95.04 (W) x 53.86 (H)
- Colores 16777216
- Consumo 56MW
- Luz de fondo actual mA
- Precio: 59.95€



Fig.4.4.1. LCD 4.3"

4.4.2 Pantalla lcd keypad shield

Se trata de un módulo de 16x2. Es mucho más simple que la anterior. (Fig.4.4.2) Incorpora una luz de fondo azul y las letras se muestran en color blanco. Utiliza 4 bits para los datos y 2 bits para VCC y GND. [13]

Precio: 5.28 €

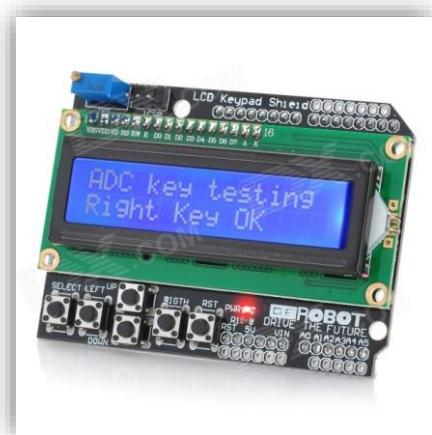


Fig.4.4.2 LCD Keypad Shield

4.5 Relés

En este apartado, haremos una breve descripción del relé elegido. En el apartado 4.1.1 seleccionamos los modelos de 5Amps y 20Amps. Es por ello, que nuestro relé deberá soportar valores máximos de 20Amps (Fig.4.5)

Dicho relé es capaz de trabajar tanto en DC como en AC, formado por 3 pines los cuales indican VCC, GND y pin digital para su control.



Fig.4.5 Módulo Relé 5V

Este elemento es imprescindible para poder ejercer la función de activar/desactivar los enchufes de nuestra regleta.

4.6 Resumen elección componentes

Tras la evaluación de las distintas opciones evaluadas para la fabricación del monitor de energía, nos hemos decidido por utilizar los siguientes componentes:

- Sensor de corriente: ACS-712 de 5Amps y 20Amps (Para 2 enchufes): Solución económica y precisa para medición de corriente en AC y DC.
- Microcontrolador: Arduino UNO: Microcontrolador de bajo coste y con una gran comunidad de usuarios en Arduino.
- Transmisión y recepción de datos: Ethernet Shield y módulo Bluetooth HC-06. Solución económica frente a módulo WIFI.
- Visualización de datos: LCD keypad shield 16 x 2.
- Relé: Relé 5V (Modelo 20A). El tamaño de la pantalla suficiente y su disposición de pines son suficientes.

5. DISEÑO

En este capítulo vamos a definir el diseño que hemos propuesto para la implementación de nuestro proyecto.

El objetivo principal es que podamos captar la potencia consumida de cada aparato que conectemos a nuestra regleta, poder visualizarla en tiempo real en un monitor, así como activar y desactivar los enchufes mediante control remoto y poder visualizar el consumo en una aplicación web. Para ello, necesitaremos captar la corriente mediante los sensores anteriores elegidos, obtener su respectivo valor de potencia, transmitir esos datos a un almacén de datos y servidor web, para poder así tener acceso a través de cualquier navegador que esté conectado a la misma red.

Para su realización, el proyecto lo vamos a dividir en dos partes. (Fig.5.0), una primera parte consistirá en la captación y procesado de los datos leídos por el sensor, desarrollo de una aplicación Android que nos sirve como control remoto del sistema, transmisión y almacenamiento de datos.

Una segunda parte, que se centrará en el desarrollo de una aplicación web, la cual, recogerá todos los datos procesados previamente por el microcontrolador y almacenados en una base de datos.

Como se expuso en el capítulo anterior, para alcanzar los objetivos propuestos en el proyecto deberemos hacer uso de los siguientes elementos hardware:

- Sensor de corriente: ACS-712 de 5Amps y 20Amps.
- Microcontrolador: Arduino UNO.
- Transmisión y recepción de datos: Ethernet Shield y módulo Bluetooth HC-06.
- Visualización de datos: LCD keypad shield 16 x 2.

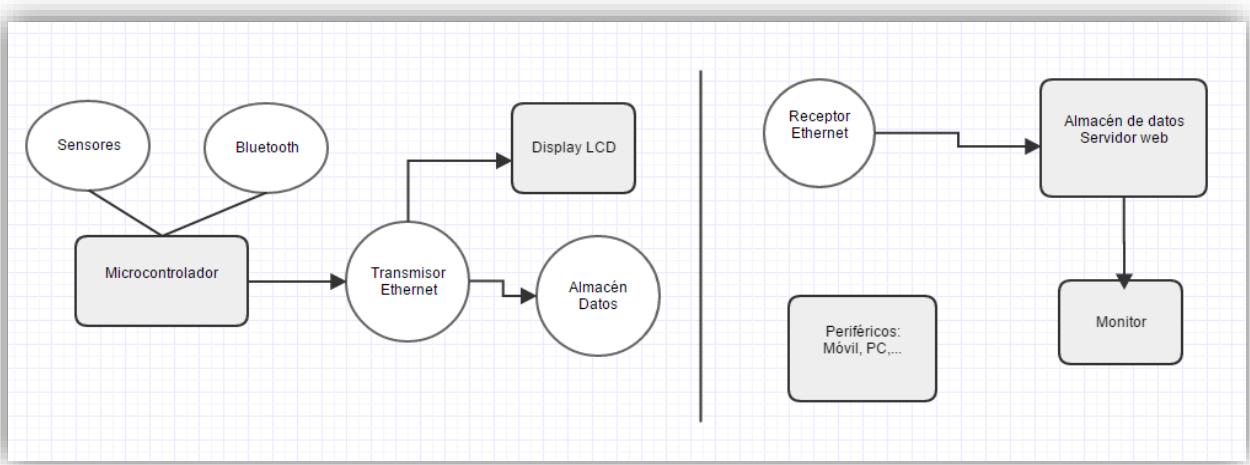


Fig.5.0 Diagrama de bloques

Captación de la corriente

En este capítulo explicaremos los elementos y firmware que intervienen en la captación de la corriente provocada por los aparatos enchufados en la regleta.

5.1.1 Sensor de corriente



Fig.5.1.1 Sensor
ACS712

Sensor de corriente por efecto hall. (Fig.5.1.1) Este sensor funciona transformando un campo magnético surgido en el paso del paso de la corriente por un alambre de cobre interno en el sensor, convirtiendo este campo en un voltaje variable. A mayor cantidad de corriente, mayor voltaje tendremos en el pin.

Especificaciones:

- Modelo ACS712ELCTR-05B-T
- Modelo ACS712ELCTR-20A-T

En nuestro proyecto usaremos los 2 modelos anteriores. El primero es la versión que mide hasta 5Amp, lo usaremos para medir electrodomésticos que no requieran de un consumo elevado. En caso contrario, usaremos el modelo que mide hasta 20A. En la siguiente figura (5.1.1.1) se puede ver el esquema del sensor.

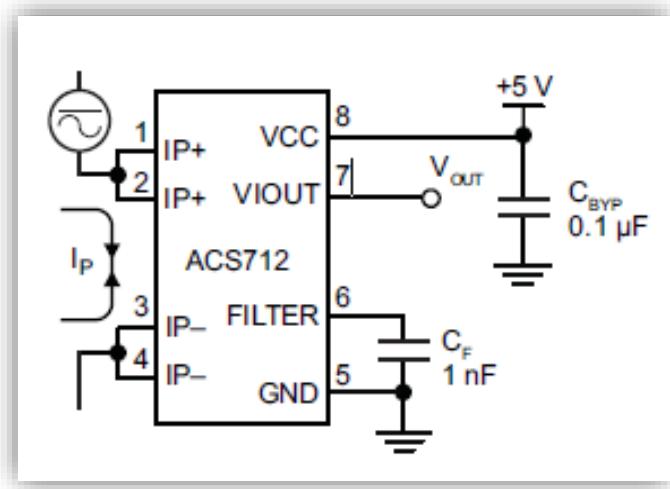


Fig.5.1.1.1 Esquema Sensor ACS712

Características Técnicas (Fig.5.1.1.2) :

Modelo	IP (A)	Sensibilidad (mV/A)
ACS712ELCTR-05B-T	±5A	185
ACS712ELCTR-20A-T	±20A	100
ACS712ELCTR-30A-T	±30A	66

Fig.5.1.1.2 Características Sensor ACS712

- Bajo ruido de la señal AC
- El ancho de banda del dispositivo se establece a través de un pin
- Tiempo de respuesta de 5 µs
- Temperatura de trabajo -65°C - 170°C
- Tensión de salida proporcional a AC o DC
- Error ±1.5%

Diagrama de bloques funcional

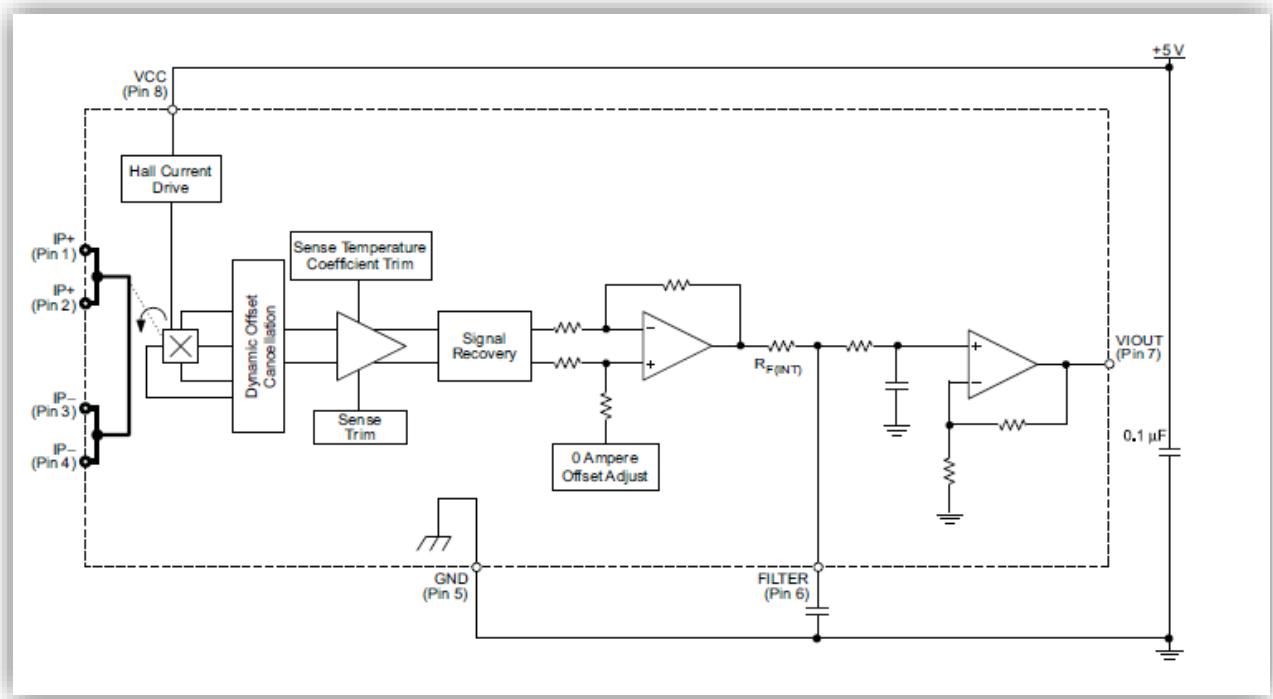


Fig.5.1.1.3 Diagrama de bloques funcional

5.1.2 Firmware de captación de corriente



Fig.5.1.1.4
Símbolo Arduino

Una vez que tenemos el sensor conectado a la placa Arduino, el siguiente paso es captar la intensidad que circula por el sensor.

Para realizar la captación de datos, necesitamos cargar en Arduino un programa que sea capaz de convertir la señal alterna producida por el sensor en valores reales de corriente.

La placa Arduino utiliza un entorno de programación fácil de usar por principiantes y a la vez, flexible para los usuarios avanzados (Fig. 5.1.1.4). El Software de Arduino, está publicado bajo una licencia libre, pudiéndose ampliar a través de librerías de C++.

El código fuente de las distintas librerías se encuentra liberado como dominio público.

El software de Arduino funciona en los sistemas operativos Windows, Macintosh OSX y Linux, mientras la mayoría de los entornos para controladores están limitados a Windows.

Nota: Mostraremos las implementaciones por separado, es decir; por cada requisito del proyecto, se ha implementado un desarrollo modular:

- Captación de corriente
- Transmisión y almacenamiento de datos
- Comunicación con microcontrolador
- Visualización de datos en pantalla LCD

El resultado final, será la implementación de un firmware en su totalidad partiendo de los anteriores módulos.

La señal que tenemos en la entrada analógica de Arduino, procedente del circuito adaptador, es convertida por el conversor analógico digital de 10 bits.

Esto significa que convertirá tensiones entre cero y 5 Voltios a un número entre 0 y 1023. (los números decimales que pueden ser representados con 10 bits) Esto proporciona una resolución en la lectura de 5 Voltios / de 1024 unidades, es decir, cada unidad vale 0,0049 Voltios (4,9 mV). Este valor es el considerado como precisión de las medidas, ya que no puede medir partes inferiores a 4,9 mV.

En nuestro caso, vamos a medir la corriente AC haciendo un promedio mediante el muestreo del voltaje de la componente AC que entrega el sensor, suponiendo un voltaje de la red de 230V.

Sabemos que:

$$V_{pp} = 2\sqrt{2V_{rms}} \text{ y } V_{pp} = \pi V_{avr} \text{ y por tanto: } 2\sqrt{2V_{rms}} = \pi V_{avr}$$

$$\text{De aquí que: } V_{rms} = V_{avr} \frac{\pi}{2\sqrt{2}} = 1.1107 \times V_{avr}$$

Si además se supone un factor de potencia cercano a la unidad(pensando que el sistema será aplicado en aparatos de consumo domésticos) se puede inferir de la anterior formula que: $I_{rms} = I_{avr} \frac{\pi}{2\sqrt{2}} = 1.1107 \times I_{avr}$

Para medir la corriente instantánea se usa la siguiente expresión, ya que tenemos un ADC con resolución de 10 bits, un voltaje de referencia de 5V y una sensibilidad de 185 mV/A para el sensor **ACS712ELCTR-05BT** de 5Amps y de 100mV/A para el sensor **ACS712ELCTR-20AT**:

$$I_{inst} = \frac{v_{ref}}{2^{n-1}} x \frac{1}{sensibilidad} = \frac{5 * 5.4054}{1023} x (ADC_{muestra} - 510) = \\ = 0.0254 x (ADC_{muestra} - 510)$$

Por otro lado, para ajustar la salida del sensor a valores cercanos a cero, debemos ajustar la ADC a un valor de 510.

Por otro lado, sabemos que la potencia P se obtiene mediante la siguiente fórmula:

$$P = V_{rms} x I_{rms} \text{ (W)}$$

El siguiente código (Fig.5.1.1.5), muestra la implementación en Arduino del consumo individual de un solo enchufe, el cual incorpora el modelo de sensor **ACS712ELCTR-05BT** que mide hasta 5Amps.

Código:

```

138 void consumoIndividual(int enchufe, int conectado){
139     //Enchufe 1 con sensor modelo 5A
140     if(enchufe==1){
141         //Consulta base de datos Plantal
142         lcd.setCursor(0,1);
143         lcd.print("Leyendo datos...");
144         for(int i=0; i<muestras; i++){
145             sensorValue_aux = sensibilidad5A * (analogRead(sensor1) - 510);
146             if(sensorValue_aux < 0 ) sensorValue_aux = - sensorValue_aux; //Rectificación de la componente - AC
147             valorSensor = valorSensor + sensorValue_aux / float(muestras); //Voltaje promedio de AC rectificada
148         }
149         valorCorriente = 1.1107 * valorSensor;
150         if(valorCorriente <= 0.05){ //+-0.05 aprox. valor experimental
151             valorCorriente = 0.000;
152         }
153         Serial.println(valorCorriente);
154         lcd.clear();
155         lcd.setCursor(0,0);
156         //Tratamos error
157         valorSensor = 0;
158         Serial.println(valorCorriente);
159         lcd.clear();
160         lcd.setCursor(0,0);
161         pot = valorCorriente * tension;

```

Fig.5.1.1.5 Función para calcular la potencia

5.2 Transmisión de datos

En este capítulo explicaremos los elementos y firmware que intervienen en la transmisión y almacenamiento de datos procesados por el microcontrolador.

5.2.1 Arduino Ethernet Shield

La placa Ethernet, (Fig.5.2.1) nos permite establecer una conexión a Internet. Para ello usamos la librería Ethernet, proporcionada por el IDE de Arduino. También incorpora una ranura para tarjetas micro-sd, la cual nos permite leer y escribir. Para hacer uso de la Ethernet, tan sólo debemos colocarla encima de nuestro Arduino. En cuanto a su configuración, le debemos asignar tanto una MAC como una IP. Es posible utilizar DHCP para asignar una IP dinámicamente y también se puede especificar una puerta de enlace de red y/o subred.

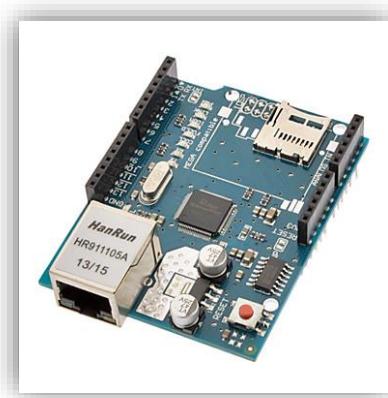


Fig.5.2.1 Ethernet Shield

Características Técnicas (Fig.5.2.2.1) :

- Alimentación: 5V
- Controlador Ethernet: W5100 con un buffer de memoria de 16K
- Velocidad: 10/100MB
- Conexión con Arduino a través de puerto SPI
- Conector estándar RJ-45

LED	INFORMACIÓN
PWR	Indica que la shield y Arduino están conectados
LINK	Indica la presencia de red y parpadea cuando hay transmisión de datos
FULLD	Conexión full dúplex
100M	Indica la presencia de conexiones 100MB/s
RX	Recibiendo datos
TX	Transmisión de datos
COLL	Indica si hay colisiones

Fig.5.2.2.1 Características Ethernet Shield

Diagrama de bloque funcional

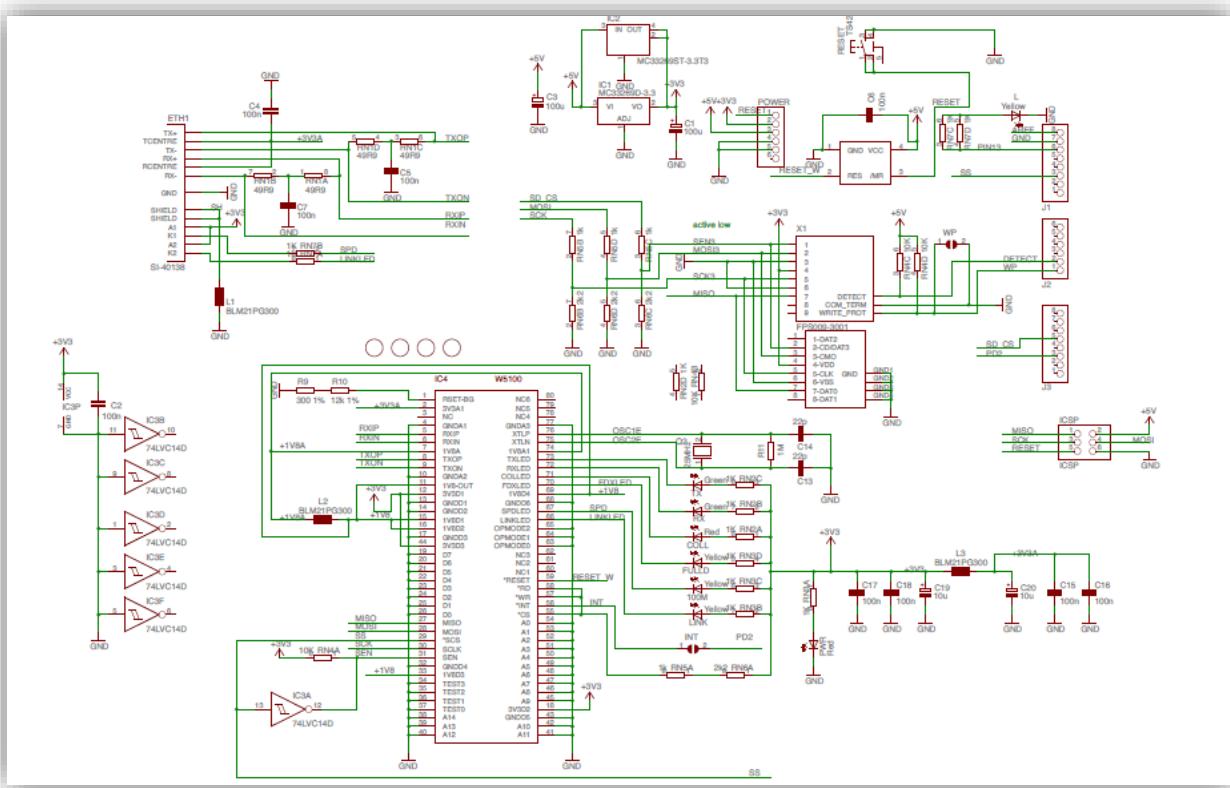


Fig.5.2.2.2 Diagrama bloques funcional

5.2.2 Firmware de transmisión de datos

Una vez que sabemos cómo obtener los datos, el siguiente paso es, trasmisitir dichos datos y almacenarlos en una base de datos.

Para ello nos hemos basado en la librería `mysql.h`, la cual nos suministra un cursor en el cual emplearemos consultas SQL con el fin de almacenar los datos en nuestra BBDD. Un esquema del funcionamiento lo podemos ver en la figura (Fig.5.2.2.3).

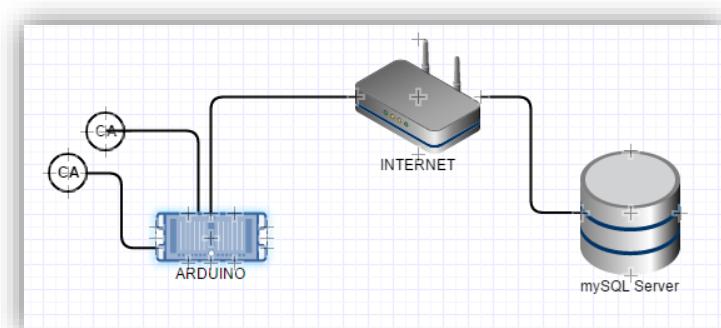


Fig.5.2.2.3 Esquema funcionamiento firmware

Código:

En la siguiente figura(Fig.5.2.2.4), se muestra la implementación. Ha sido necesario usar una librería externa (mySQL Connector), para transmitir los datos procesados por el microcontrolador, y con ello almacenar los datos directamente en nuestra base de datos. [18]

```
//Definicion conexion MySQL
IPAddress mysql_server(192,168,1,51); //Ip servidor
IPAddress ethernet_shield(192,168,1,10); //Ip arduino
char user[] = "usuario"; //Usuario Base de datos
char password[] = "1234"; //Contraseña
byte mac_addr[] = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED };
Connector arduino_conn; //Objeto conector

int Conexion(){
    Ethernet.begin(mac_addr, ethernet_shield);
    int res = 0;
    delay(1000);
    lcd.clear();
    lcd.setCursor(0,2);
    lcd.print("Conectando...");
    if(arduino_conn.mysql_connect(mysql_server,3306,user,password)){
        res= 1;
        lcd.clear();
        lcd.print("Conectado.");
        delay(1000);
    }
    else{
        res = 0;
        lcd.clear();
        lcd.print("Sin conexión");
        delay(1000);
    }
    return res;
}
if(conectado == 1){
    lcd.print("Guardando...");
    char * Query = "INSERT INTO sensor1 (INTENSIDAD,DATE,TIME) VALUE (%d,CURRENT_DATE,CURRENT_TIME)";
    sprintf(datos,Query,intensidadQuery);
    Serial.println("Datos");
    Serial.println(datos);
    arduino_conn.cmd_query("use arduino"); //Usamos tabla arduino
    arduino_conn.cmd_query(datos);
    lcd.clear();
}
```

Fig.5.2.2.4 Firmware transmisión de datos

5.3 Comunicación con microcontrolador

En este capítulo explicaremos los elementos y el firmware que intervienen en la comunicación con el microcontrolador. Para ello haremos uso de la tecnología Bluetooth

5.3.1 Dispositivo Bluetooth

Como ya comentamos en el apartado 4.3.3, haremos uso de un enlace punto a punto usando la tecnología bluetooth. En la siguiente figura podemos ver un esquema del funcionamiento. A través de la aplicación Android, la cual, hablaremos más adelante, mandaremos una serie de caracteres los cuales, el propio microcontrolador los interpretará y actuará, activando/desactivando los enchufes correspondientes que seleccionemos.

En la siguiente figura (Fig.5.3.1) podemos ver el circuito del funcionamiento del módulo bluetooth HC-06

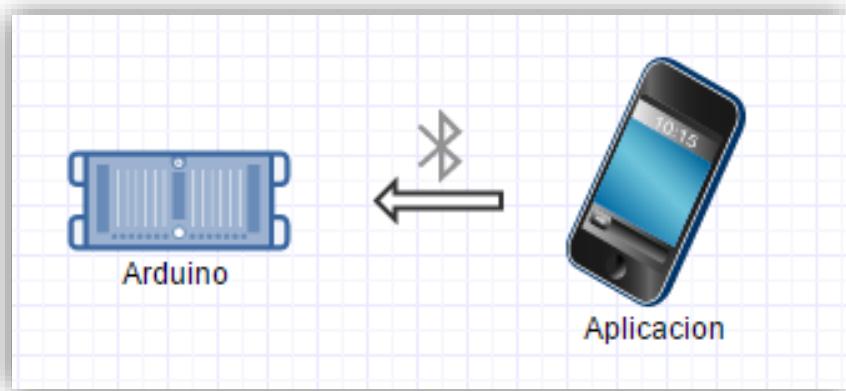
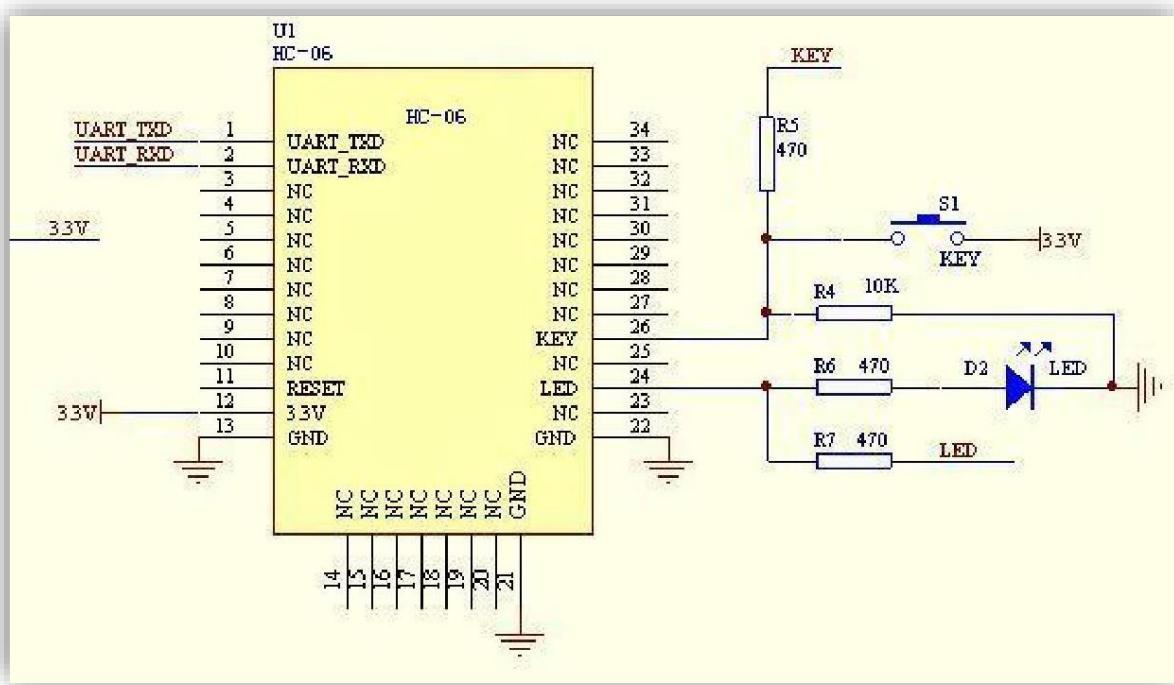


Fig.5.3.1 Esquema comunicación bluetooth



Esquema 5.3.1.1 Esquema módulo bluetooth HC-06

En principio, puede funcionar cuando los pines UART_TXD, UART_RXD, VCC y GND, estén conectados. Sin embargo, para mejores resultados, es necesario hacer uso de un divisor de tensión, puesto que la comunicación del pin TX funciona con una tensión de 3.3V.

5.3.2 Firmware de comunicación con microcontrolador

El dispositivo Bluetooth está configurado de forma que actúa como Slave. Es por ello recibirá ordenes desde la aplicación Android y el microcontrolador actuará en consecuencia.

A través de la aplicación móvil, enviamos una serie de caracteres según el botón que pulsemos. Dicho carácter se procesa en la función que se muestra en la siguiente figura. (Fig.5.3.1.2) [14]

```
int * bluetooth(){
    if(BT1.available()){
        estado = BT1.read();
    }
    if(estado == '1'){
        digitalWrite(pinRelay1,HIGH);
        res[0] = 1;
    }
    if(estado == '2' ){
        digitalWrite(pinRelay1,LOW);
        res[0] = 0;
        lcd.clear();
        lcd.setCursor(0,0);
        lcd.print("Enchufe 1 OFF");
        digitalWrite(led1,LOW);
        delay(1000);
        lcd.clear();
    }
    if(estado == '3'){
        digitalWrite(pinRelay2,HIGH);
        res[1]=1;
    }
    if(estado == '4'){
        digitalWrite(pinRelay2,LOW);
        res[1]= 0;
        lcd.clear();
        lcd.setCursor(0,0);
        lcd.print("Enchufe 2 OFF");
        digitalWrite(led2,LOW);
        delay(1000);
        lcd.clear();
    }
    return res;
}
```

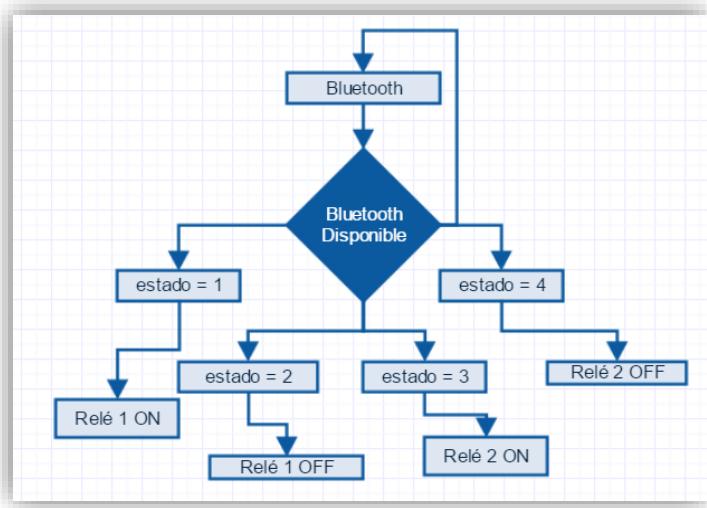


Fig.5.3.1.3
Diagrama de flujo firmware

Fig.5.3.1.2
Firmware Bluetooth

5.4 Diseño prototipo

En este apartado explicaremos la parte de diseño y maquetación que hemos tenido que llevar a cabo para el correcto funcionamiento del proyecto. Indicaremos también, todas las conexiones y el software utilizado.

Software Utilizado:

El software que hemos utilizado es **Fritzing**, (Fig.5.4) un programa de automatización de diseño electrónico libre que nos ayuda a diseñar para que poder pasar de prototipos (usando, por ejemplo, placas de pruebas) a productos finales. [15]



Fig.5.4 Símbolo
Fritzing

Fritzing fue creado bajo los principios de Processing y Arduino, y permite a los diseñadores, investigadores y aficionados documentar sus prototipos basados en Arduino y crear esquemas de circuitos impresos para su posterior fabricación. Además, cuenta con un sitio web complementario que ayuda a compartir y discutir bosquejos y experiencias y a reducir los costos de fabricación. Destacar que el editor cuenta con 3 vistas disponibles que nos son de gran utilidad. Vista Photoboard, vista esquema y vista PCB

5.4.1 Sensor de corriente

Antes de conectar debemos asegurarnos de que estamos conectando los valores de los pines correctamente tal y como nos marca el datasheet. En la siguiente figura (Fig.5.4.1), podemos ver la conexión.

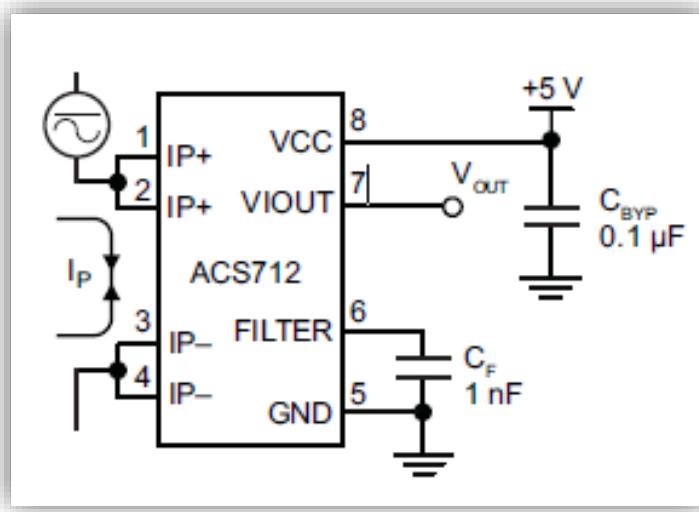


Fig.5.4.1 Esquema sensor ACS712

Conexiones:

A partir de la figura anterior, desarrollamos la siguiente tabla, para el sensor1 asignando el lugar correspondiente en el microcontrolador (Fig.5.4.1.1)

PIN SENSOR	MICROCONTROLADOR
VCC	VCC (5v)
VOUT	Analógico A5
GND	GND

Fig 5.4.1.1 Tabla Conexiones sensor1

Elaboramos una siguiente tabla para el sensor2 (Fig.5.4.1.2)

PIN SENSOR	MICROCONTROLADOR
VCC	VCC (5v)
VOUT	Analógico A4
GND	GND

Fig 5.4.1.2 Tabla Conexiones sensor2

Como podemos observar, las conexiones se repiten, a excepción el valor de entrada. Cada sensor tiene una entrada analógica al microprocesador.

Una vez que tenemos la asignación de las conexiones, para mayor comodidad y visualización, podemos pasar dichos resultados al programa anteriormente comentado. Como resultado obtenemos las siguientes figuras, (Fig.5.4.1.3) vista photoboard y (Fig.5.4.1.4) vista esquema circuito.

Vistas:

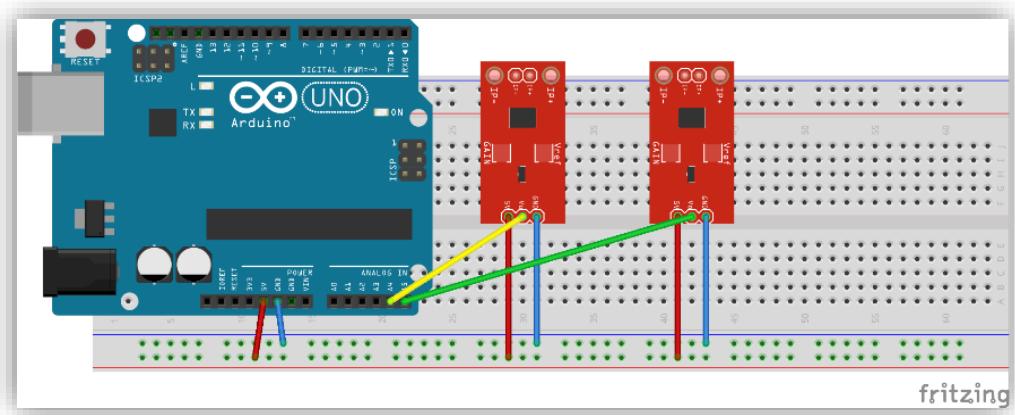


Fig.5.4.1.3 Vista photoboard sensores

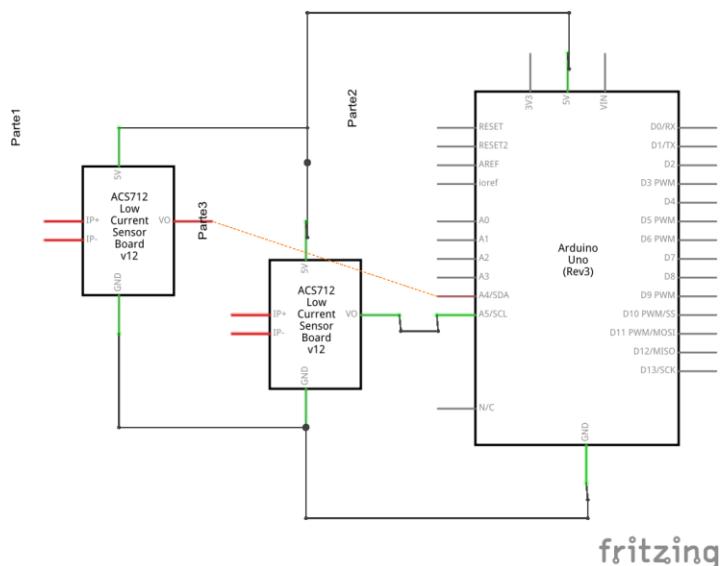


Fig.5.4.1.4 Vista circuito sensores

5.4.2 Bluetooth

En este apartado, realizaremos el mismo proceso que en el anterior. Mostraremos el funcionamiento del circuito en la siguiente figura (Fig.5.4.2) y la asignación de sus pines

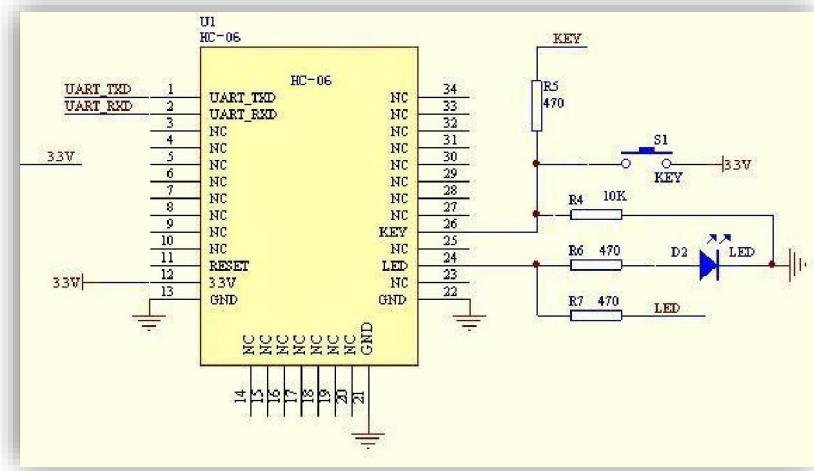


Fig.5.4.2 Esquema Módulo Bluetooth HC-06

A partir del esquema anterior, conectaremos y asignaremos adecuadamente los valores de los pines del módulo al microcontrolador. Para ello, para mayor comodidad haremos una tabla con las correspondientes asignaciones.

Conexión:

PIN BLUETOOTH	MICROCONTROLADOR
UART_RXD	Digital D1 VCC 3.3-6.0V
UART_TXD	Digital D0 5V
VCC	VCC (5V)
GND	GND

A continuación, se muestran las vistas: vistas photoboard (Fig.5.4.2.2) y vistas esquema (Fig.5.4.2.3) realizadas con Fritzing.

Hemos realizado un divisor de tensión, ya que el pin UART_RXD funciona a 3,3V. Para ello usamos dos resistencias de valores: 22kOhm y 10kOhm. A continuación, se muestran las vistas

Vistas:

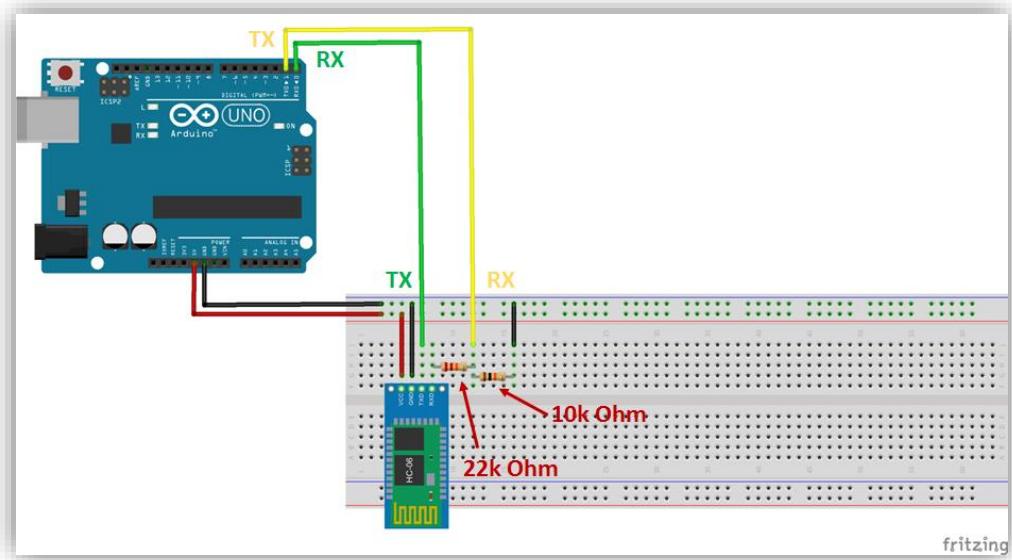


Fig.5.4.2.2 Vista photoboard bluetooth

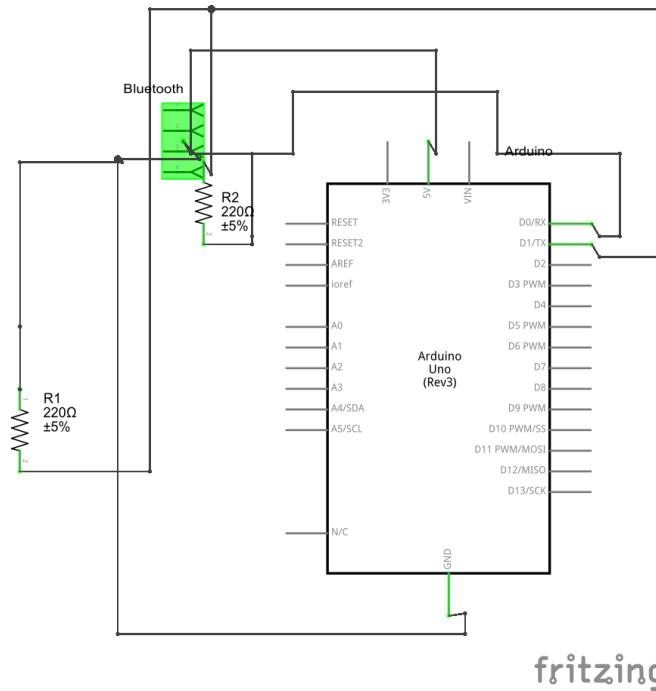


Fig.5.4.2.2 Vista photoboard bluetooth

5.4.3 Relés

Serán los encargados de activar o desactivar los enchufes de nuestra regleta, a través de la orden, transmitida por Bluetooth. En la siguiente tabla (Fig.5.4.3) se muestran las conexiones.

Conexiones:

PIN RELE1	PIN MICROCONTROLADOR
GND	GND
VCC	VCC (5V)
IN	DIGITAL D3
PIN RELE2	PIN MICROCONTROLADOR
GND	GND
VCC	VCC (5V)
IN	DIGITAL D2

Fig.5.4.3 Conexión relés

A continuación, se muestran las vistas: vistas photoboard (Fig.5.4.3.1) y vistas esquema (fig.5.4.3.2).

Vistas:

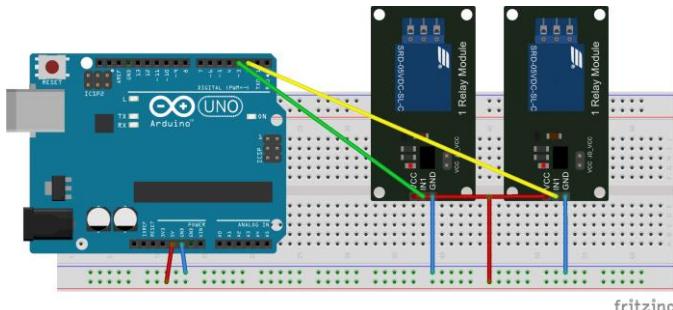


Fig.5.4.3.1 Vista photoboard
relés

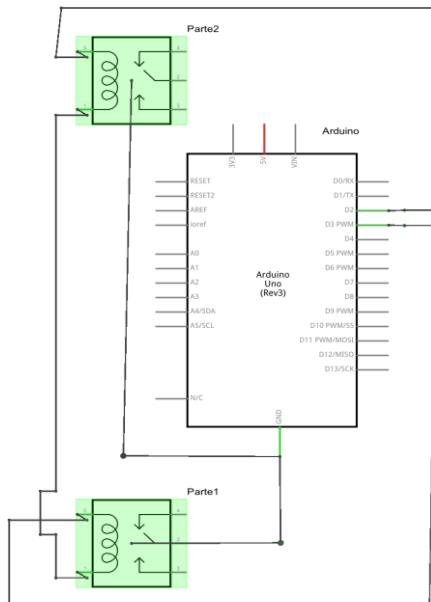


Fig.5.4.3.2 Vista esquema relés

5.5 Diseño PCB

Para el diseño de nuestro prototipo final, hemos utilizado la misma plataforma que se ha comentado en los puntos anteriores. [15]

Para una mejor conexión entre los elementos y mayor contacto entre ellos, hemos sustituido la photoboard, por un diseño sencillo de PCB. Dicho diseño cumple la función de shield. Para empezar a desarrollar, es necesario que conozcamos una serie de directrices y normas para aplicar.

A continuación, se muestra una figura con el diseño de la PCB (Fig.5.5).

El diseño está elaborado a doble capa, ya que así nos permite una mayor comodidad a la hora de enrutar las pistas, con un tamaño de 68.9 mm x 53.7mm de superficie. Por otro lado, tener presente que las capas adicionales aumentan los costos de producción.

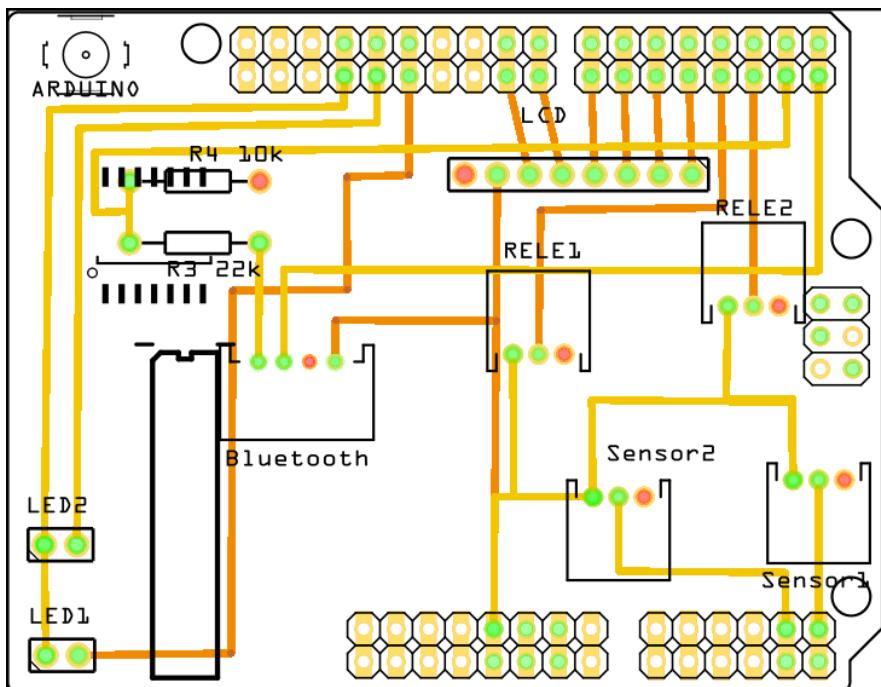


Fig.5.5 PCB

fritzing

Otro aspecto a tener en cuenta la forma de la placa. En este caso, en nuestra plataforma de diseño, podemos seleccionar un shield estándar con el mismo tamaño de Arduino.

Recomendable dibujar una visión general de dónde se ubicarán las diferentes zonas de circuito. Una de los primeros pasos del trazado del circuito es dibujar aproximadamente, el lugar donde se encontrará los elementos de nuestro circuito.

Como se puede apreciar en la figura, hemos utilizado conectores JST de 2.54 mm ya que aportan una mayor sujeción a las conexiones con el resto de elementos, usando el tamaño estándar de los pines que usa Arduino.

Hay que tener en cuenta el tamaño de la pista y la anchura que deben tener entre ellas las pistas, pues caso contrario podría provocar un corto, además si son demasiado grandes y están muy separados, puede restringir el número de pistas o incluso la refrigeración de los componentes. En este caso, el tamaño de las pistas usa un ancho de 2.4mm, tamaño estándar y suficiente, en proporción a la intensidad que circula.

Es muy recomendable que, en el diseño de las pistas, los giros y cruces entre ellas sean perpendiculares entre sí, ya que así minimizamos la autoinducción entre ambas y la captación de radiación en el ambiente.

Evitar las pistas que se desarrollen en paralelo, pues aumentará el nivel de interferencias con las señales de otra pista.

El resultado obtenido se muestra en la siguiente figura (Fig.5.5.1):

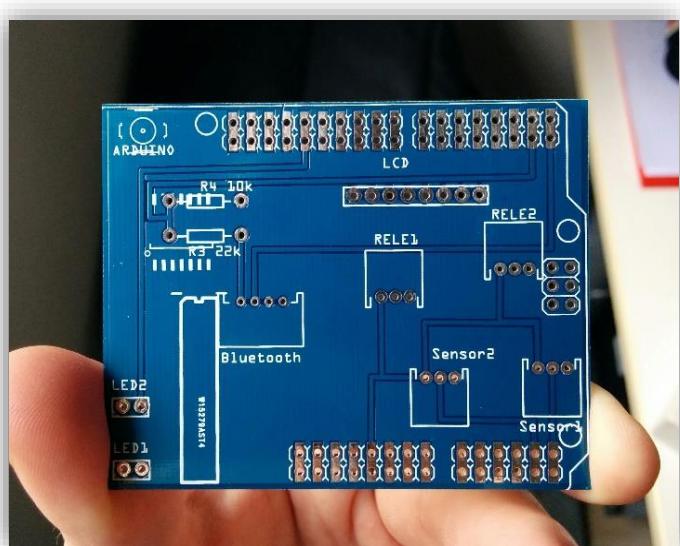


Fig.5.5.1 Resultado PCB

5.6 Diseño Aplicación Android

Una vez establecidos los objetivos y los requisitos que debe cumplir nuestro proyecto, junto con un diseño (CAPITULO 2 Y 5), hemos decidido el siguiente diseño para nuestra aplicación Android.

Optamos por un diseño móvil, al considerar que este diseño pueda tener aplicaciones futuras.

En este capítulo, hablaremos de dos prototipos de aplicación, ambos realizados de dos formas distintas:

- a) Framework Google Labs
- b) Android Studio

Framework Google Labs

Aplicación Android diseñada con framework de Google Labs, llamado *AppInventor*, el cual nos permite crear de forma sencilla una aplicación con sistema operativo Android. [\[16\]](#)

La aplicación constará de una interfaz en la cual disponemos un conjunto de botones, que serán los encargados de encender y apagar los enchufes de nuestra regleta.

También incorporará 2 botones, el primero será un acceso directo para activar la conexión Bluetooth, y el segundo botón, nos llevará a la web que monitoriza nuestros datos.

5.6.1 Diagrama de flujo

Para el diseño de nuestra aplicación hemos utilizado el diagrama de flujo de la siguiente figura. (Fig.5.6.1)

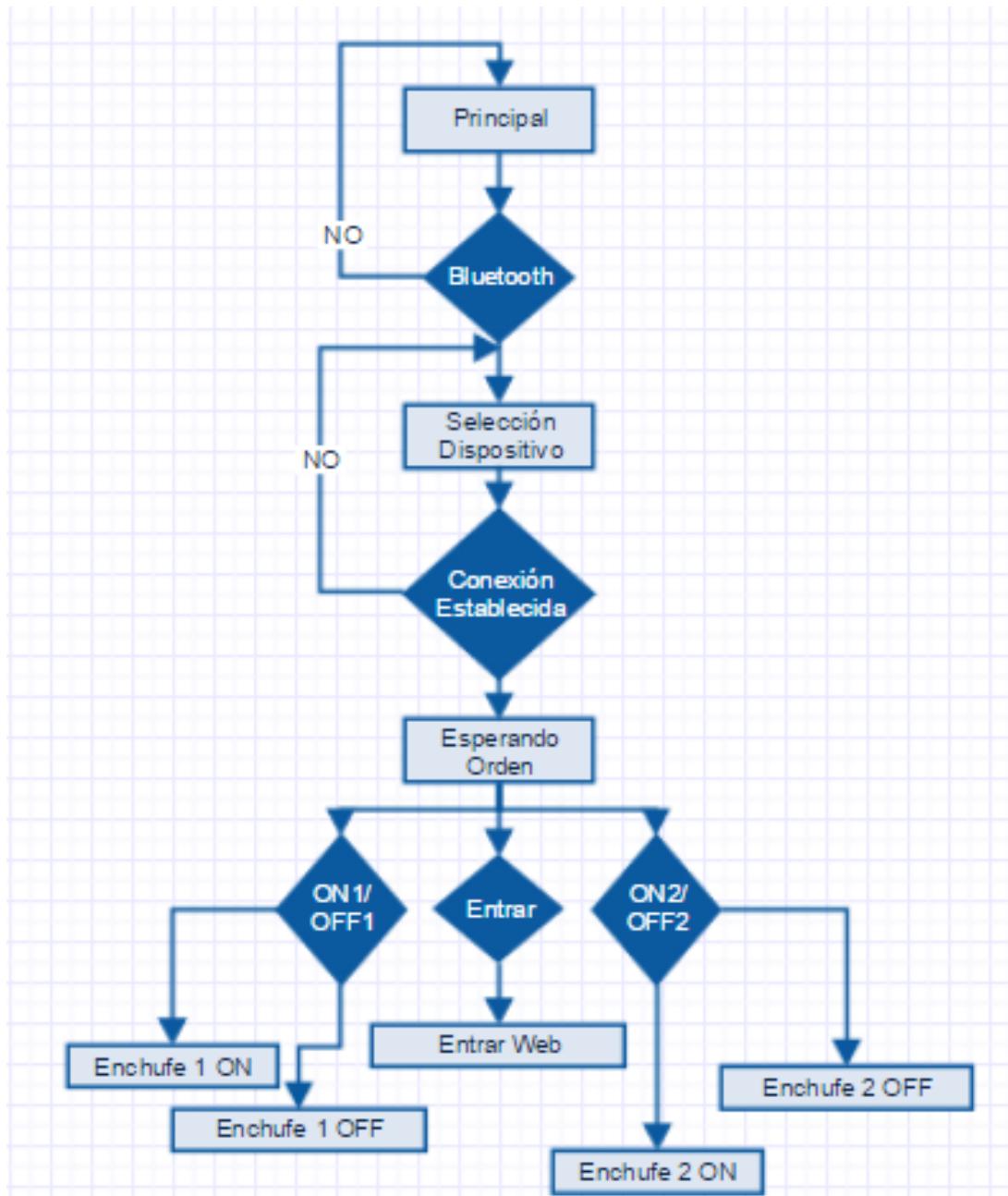


Fig.5.6.1 Diagrama de flujo Aplicación

5.6.2 Realización de Mockups

Tras realizar el diagrama de flujo, vamos a diseñar unos Mockups (simulación

interactiva de la misma), para fabricar un prototipo de nuestra aplicación para ello

hemos utilizado gliffy, una aplicación web gratuita.

En la siguiente figura (Fig.5.6.2) podemos ver el resultado de la interfaz que usará nuestra aplicación Android.

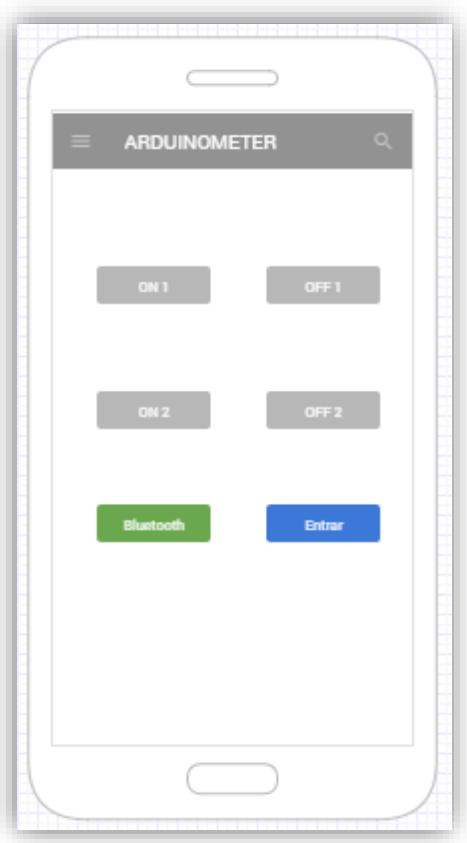


Fig.5.6.2 Mockup Aplicación

5.6.3 AppInventor

Una vez realizado nuestro diseño, vamos a proceder a explicar en este apartado, los aspectos básicos y fundamentales que necesitamos para hacer nuestra aplicación

¿Qué es AppInventor? ¿Cómo funciona?

Appinventor es un framework, desarrollado por Google Labs, el cual nos permite desarrollar de forma fácil, cómoda y rápida aplicaciones para dispositivos con sistema operativo Android. [\[16\]](#)

El funcionamiento global está formado por dos partes.

La primera parte consiste en el diseño de la interfaz de usuario: En ella podemos hacer uso de distintos elementos básicos que forman la interfaz. Nos podemos encontrar diferentes estilos, layouts, etc.

En la segunda parte nos encontramos el apartado de la programación. Podemos decir que resulta muy sencillo la

facilidad de uso en el desarrollo de código de programación, pues el estilo que utiliza está basado en “piezas de puzzle”. Tenemos diferentes estilos de piezas las cuales cada una de ellas cumplen una función determinada y encajan en el sitio adecuado.

En las siguientes figuras (Fig.5.6.3), y (Fig.5.6.3.1) se muestran las zonas en las que se divide el editor de diseño de interfaz, y el entorno de programación.

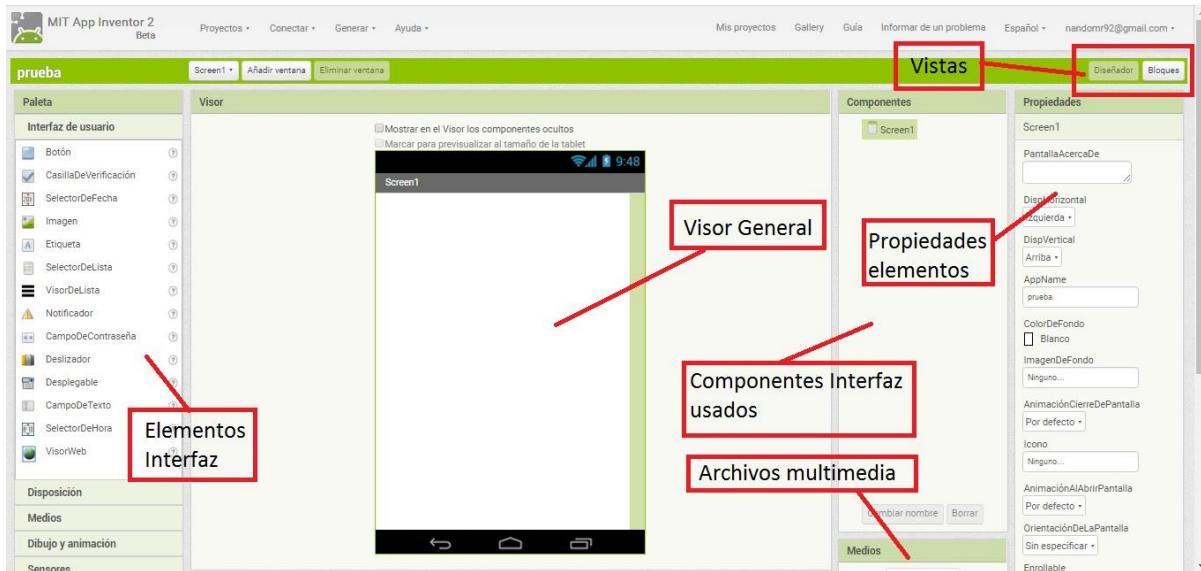


Fig.5.6.3 Editor AppInventor diseño

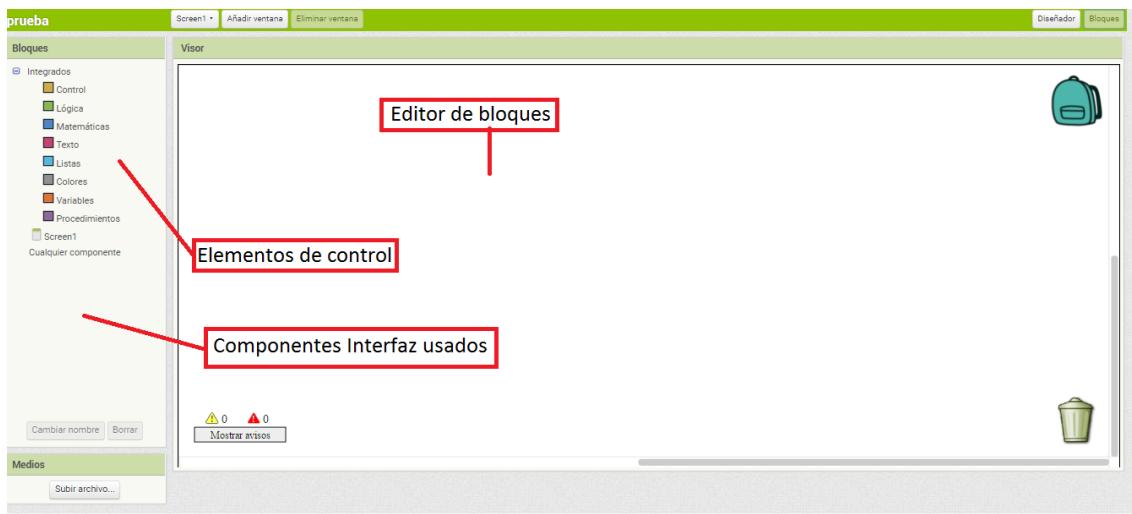


Fig.5.6.3.1 Editor AppInventor lógico

Un inconveniente a citar es que AppInventor, no nos ofrece ese potencial que nos brinda el entorno de desarrollo Android Studio. No se pueden realizar aplicaciones complejas y tampoco usar el estilo actual que lleva Android, material deseado.

5.6.4 Desarrollo de la interfaz gráfica

En este apartado, explicaremos y mostraremos el proceso de desarrollo de la aplicación.

En primer lugar, deberemos asignar un layout, o contenedor, el cual nos ofrece una estructura. Dicha estructura, nos permitirá colocar los elementos oportunos, según la configuración que ésta nos ofrece.

A continuación, ya podremos colocar los elementos que forman nuestra interfaz. Tan sólo debemos seleccionarlos y arrastrarlos hacia la posición adecuada.

En la siguiente figura, (Fig.5.6.4) podemos ver su diseño.



Fig.5.6.4 Interfaz ArduControl

Como podemos ver en la figura anterior, nos aparece titulado "Componentes no visibles". Los componentes no visibles son:

BluetoothClient: adaptador bluetooth, que hace función de cliente.

ActivityStarter, es un método que usa Android para lanzar otra actividad. En el siguiente apartado lo explicaremos con más detalle.

En la siguiente vista figura (Fig.5.6.4.1) podemos ver el resultado final

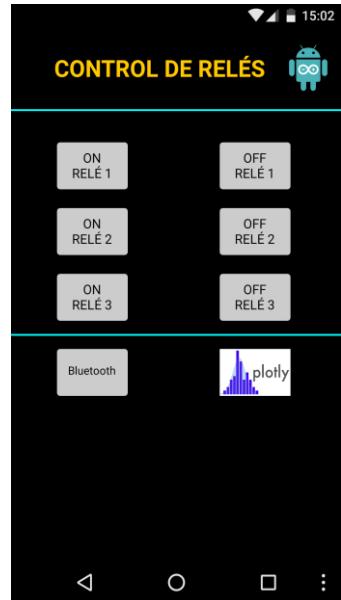


Fig.5.6.4.1 Vista aplicación Android en móvil

5.6.5 Desarrollo Software

En este apartado, mostraremos el código necesario que lleva nuestra aplicación.

Como hemos comentado anteriormente, disponemos de un entorno de programación gráfico, basados en piezas de puzzle, muy intuitivo y fácil para desarrollar.

Nuestra aplicación, (Fig.5.6.5) consta de 4 botones, que serán los encargados de dar la orden de activar / desactivar los enchufes. Comentamos que la comunicación se realiza a través de tecnología bluetooth.

Cada par de botones pertenece a un enchufe correspondiente, el cual, activa o desactiva el enchufe determinado. A través de los caracteres que enviamos al receptor bluetooth, realizamos la acción determinada.

El control del primer enchufe, lo realiza el tipo Button1 y Button2. El tipo Button3 y Button4 se encargan del segundo. Ambos, llaman a la función BluetoothClient, la cual manda un carácter que procesa el microcontrolador.

Disponemos de otro botón, Button7, el cual nos aporta un acceso directo a una web que hemos desarrollado que hace de monitor de energía. Hablaremos de su desarrollo en el siguiente capítulo.

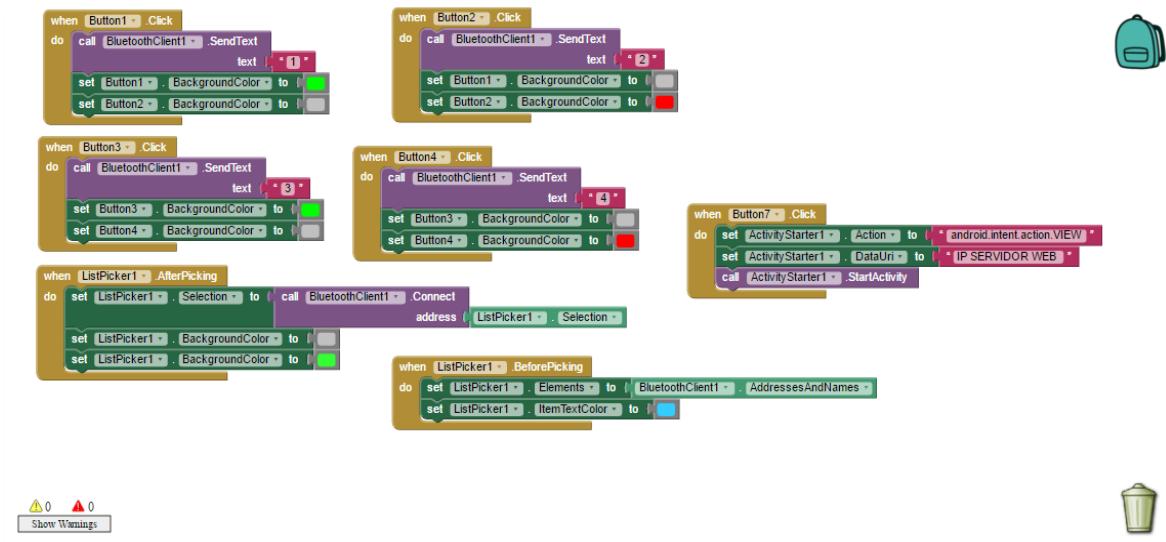


Fig.5.6.5 Lógica de la aplicación

Funciones:

- BluetoothClient.

Es una función que podemos encontrar en el apartado de conectividad, en la sección “Elementos de la interfaz”. Internamente, hace uso de un método llamado *connect*, el cual establece la conexión con el otro extremo. Si la conexión se ha establecido devuelve true, caso contrario false.

- ActivityStarter.

Dicha función nos permite lanzar otra actividad. Internamente hace uso del método *startActivity*, que usa un *Intent*. Un *Intent* es un elemento básico de comunicación, entre componentes Android, el cual, a través de él mostramos una actividad. Ejemplo de actividades es mostrar un servicio, enviar un mensaje, iniciar otra aplicación, etc.

Componentes:

-ListPicker

También llamado listas de selección. Son listas de texto en el cual los usuarios pueden seleccionar un elemento. Cuando el usuario selecciona dicho elemento, elige una acción determinada. En nuestro caso, la acción es "Selection", la cual llama a la función BluetoothClient que a partir del elemento que hemos seleccionado, establece conexión.

5.6.6 Android Studio

En los apartados anteriores, se habla del diseño y desarrollo de la aplicación que hará de interfaz de usuario para manejar nuestro sistema, desarrollada a través de AppInventor.

En este apartado, proponemos un desarrollo alternativo, más óptimo, más estable y con mejor rendimiento hacia el usuario.

5.6.6.1 ¿Qué es Android Studio?

Android Studio (Fig.5.6.6.1) es un entorno de desarrollo integrado para la plataforma Android. Fue anunciado el 16 de mayo de 2013 en la conferencia Google I/O, y reemplazó a Eclipse como el IDE oficial para el desarrollo de aplicaciones para Android.



Fig.5.6.6.1 Símbolo
Android Studio

Actualmente existe una versión 2.0 del IDE. En la siguiente figura (Fig.5.6.6.1.1) se muestra la ventana principal de nuestro IDE.] [\[17\]](#)

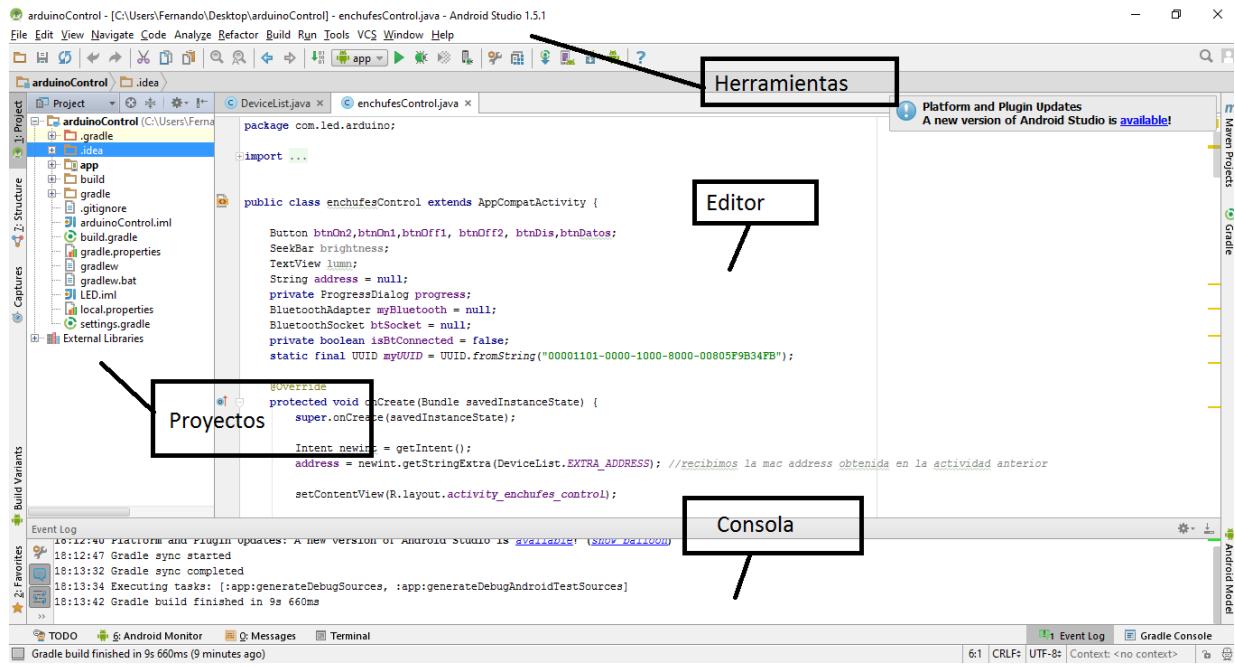
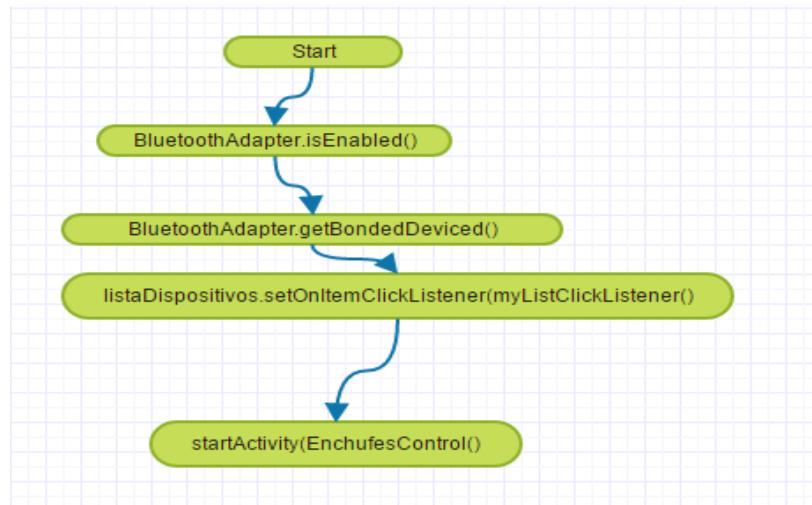


Fig.5.6.6.1.1 IDE Android Studio

5.6.6.2 Interfaz Gráfica

En este apartado mostraremos el desarrollo de la interfaz gráfica de uso nuestra aplicación. Dicha aplicación está formada por dos partes diferenciadas.

Parte I: Lita de dispositivos vinculados. Aquí mostraremos una lisa de dispositivos bluetooth los cuales hemos vinculado previamente. En la siguiente figura (Fig.5.6.6.2) se muestra un esquema del funcionamiento



5.6.6.2 Esquema Funcionamiento APP: Dispositivos

Parte II: Panel de control. En esta parte, nos encontramos con la capacidad de poder interactuar con el sistema. En la siguiente figura (Fig.5.6.6.2.1) se muestra un esquema del funcionamiento

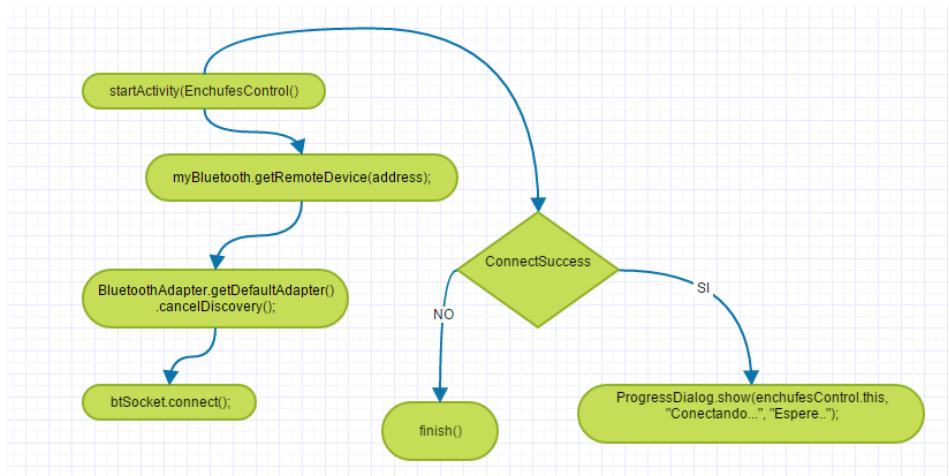


Fig.5.6.6.2.1 Esquema Funcionamiento APP: Panel de control

Diseño Parte I: El primer paso que debemos hacer, es crear una activity. Dicha activity está relacionada con nuestro layout. Para su creación, pulsamos con el botón derecho sobre la carpeta layout, situada en Android/app/layout/new/Activity/Blank Activity.

En la siguiente figura se puede mostrar el proceso (Fig.5.6.6.2.2)

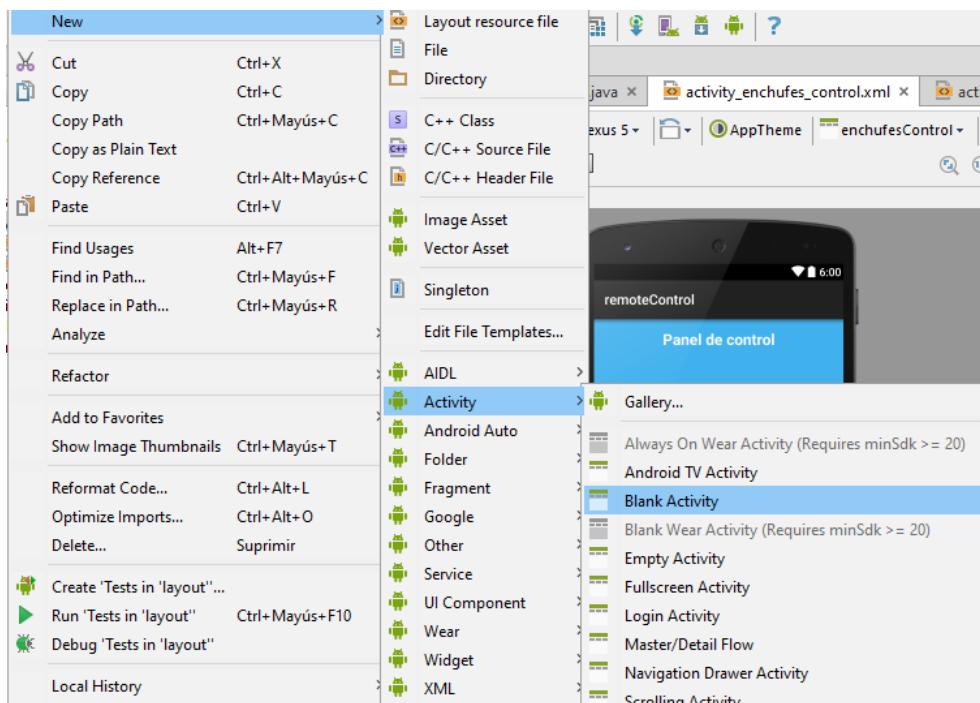


Fig.5.6.6.2.2 Creación Activity

Una vez creada nuestra Activity, debemos dirigirnos a la carpeta layout para su posterior diseño. En dicha carpeta nos encontramos tantos archivos XML, como Activity tengamos creadas.

En nuestro caso, nos encontramos con: `activity_device_list.xml`. En el editor, nos encontramos 2 pestañas las cuales, nos muestra dos tipos de vista, Design y Text. En la parte de Design nos encontramos todos los elementos que podemos usar en nuestra aplicación. En la parte text, nos encontramos la traducción de dichos elementos a lenguaje XML. En dicha traducción nos encontramos las propiedades que le podemos otorgar, ya sea el formato, color, tamaño, etc.

Hemos hecho uso del tipo `TextView`(Fig.5.6.6.2.3), un tipo `Button` (Fig.5.6.6.2.4) y por último el uso de un `ListView` (Fig.5.6.6.2.5). Se muestran en las siguientes figuras.

TextView:

```
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textAppearance="?android:attr/textAppearanceLarge"
    android:text="Dispositivos Vinculados"
    android:id="@+id/textView4"
    android:foregroundGravity="bottom|center"
    android:textStyle="normal|bold"
    android:textIsSelectable="false"
    android:textColor="#fefefe"
    android:layout_alignParentRight="true"
    android:layout_alignParentEnd="true"
    android:layout_alignParentTop="true"
    android:layout_alignParentLeft="true"
    android:layout_alignParentStart="true" />
```

Fig.5.6.6.2.3 TextView - Dispositivos Vinculados

Button:

```
<Button  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:text="Dispositivos Vinculados"  
    android:id="@+id/button"  
    android:layout_alignParentBottom="true"  
    android:layout_alignParentLeft="true"  
    android:layout_alignParentStart="true"  
    android:layout_alignRight="@+id/listView"  
    android:layout_alignEnd="@+id/listView" />
```

Fig.5.6.6.2.4 Button – Dispositivos Vinculados

ListView:

```
<ListView  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:id="@+id/listView"  
    android:background="#e5dfdf"  
    android:foregroundGravity="bottom"  
    android:layout_alignParentRight="true"  
    android:layout_alignParentEnd="true"  
    android:layout_above="@+id/button"  
    android:layout_below="@+id/textView4" />
```

Fig.5.6.6.2.5 ListView

En la siguiente figura (Fig.5.6.6.2.6) nos encontramos el resultado.



Fig.5.6.6.2.6 Resultado Parte I

Diseño Parte II

Como hemos comentado anteriormente, debemos crear una nueva *Activity*, *activity_enchufes_control.xml*, la cual se relaciona con el layout que mostraremos a continuación. Dicho layout, estará formado por los siguientes elementos:

RelativeLayout: su función es la de un contenedor. Nos permite distribuir por el panel los elementos y colocarlos de la forma que queramos. (Fig.5.6.6.2.7)

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools" android:layout_width="match_parent"
    android:layout_height="match_parent" android:paddingLeft="16dp"
    android:paddingRight="16dp"
    android:paddingTop="16dp"
    android:paddingBottom="16dp" tools:context=".enchufesControl"
    android:background="#41b1ed">
</RelativeLayout>
```

Fig.5.6.6.2.7 RelativeLayout

TextView: Objeto que nos sirve para mostrar texto. Tendremos un primer objeto titulado "Panel de control" (Fig.5.6.6.2.8) y un segundo titulado "Visualizar datos online" (Fig.5.6.6.2.9).

```
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textAppearance="?android:attr/textAppearanceLarge"
    android:text="Visualizar datos online"
    android:id="@+id/textView3"
    android:textStyle="bold"
    android:textColor="#fbfbfb"
    android:layout_below="@+id/button5"
    android:layout_centerHorizontal="true"
    android:layout_marginTop="37dp" />
```

Fig.5.6.6.2.8 TextView - Panel de control

```

<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textAppearance="?android:attr/textAppearanceLarge"
    android:text="Visualizar datos online"
    android:id="@+id/textView3"
    android:textStyle="bold"
    android:textColor="#fbfbfb"
    android:layout_below="@+id/button5"
    android:layout_centerHorizontal="true"
    android:layout_marginTop="37dp" />

```

Fig.5.6.6.2.9 TextView - Visualizar datos online

Button - Enchufes: el objeto botón. Tendremos 4 botones que harán referencia a los enchufes. Tendremos 2 pares de botones los cuales apagan y encienden los enchufes. Se muestran en las siguientes figuras.

```

<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Enchufe 1 ON"
    android:id="@+id/button2"
    android:layout_above="@+id/button5"
    android:layout_alignParentLeft="true"
    android:layout_alignParentStart="true"
    android:layout_marginBottom="60dp"
    />

<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Enchufe 1 OFF"
    android:id="@+id/button3"
    android:layout_alignTop="@+id/button2"
    android:layout_alignParentRight="true"
    android:layout_alignParentEnd="true" />

```

Fig.5.6.6.2.10 Button - Enchufe1 ON / OFF

```

<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Enchufe 2 ON"
    android:id="@+id/button5"
    android:layout_centerVertical="true"
    android:layout_alignParentLeft="true"
    android:layout_alignParentStart="true" />

<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="ENCHUFE 2 OFF"
    android:id="@+id/button6"
    android:layout_below="@+id/button2"
    android:layout_alignLeft="@+id/button3"
    android:layout_alignStart="@+id/button3" />

```

Fig.5.6.6.2.11 Button - Enchufe2 ON / OFF

En la siguiente figura (Fig. 5.6.6.2.12) se muestra otro objeto Button, cuya función nos permite acceder a la aplicación web. En el capítulo 6 se hablará del desarrollo de la misma.

```
<Button  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:text="ENTRAR"  
    android:id="@+id/button7"  
    android:layout_below="@+id/textView3"  
    android:layout_toLeftOf="@+id/button6"  
    android:layout_toStartOf="@+id/button6" />
```

Fig.5.6.6.2.12 Button - Entrar

Por último, nos encontramos un botón que sirve para desconectar la conexión Bluetooth establecida. Se muestra en la siguiente figura

```
<Button  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:text="DESCONECTAR BLUETOOTH"  
    android:id="@+id/button4"  
    android:layout_alignParentBottom="true"  
    android:layout_alignParentLeft="true"  
    android:layout_alignParentStart="true" />
```

Fig.5.6.6.2.13 Button - Desconectar Bluetooth

El resultado final se puede ver en la siguiente figura.



Fig.5.6.6.2.14 Resultado Parte II

5.6.6.3 Desarrollo Software aplicación

En este apartado, explicaremos el desarrollo software de nuestra aplicación. Como ya sabemos, tenemos 2 activities, las cuales forman nuestra aplicación. El objetivo se centra en programar su funcionamiento. Para saber dónde se encuentran, accedemos a la ruta app/java/com.arduControl.android

- a) DeviceList.java
- b) enchufesControl.java

A continuación, mostramos un diagrama de clases UML, el cual nos resume brevemente la estructura de nuestra aplicación. (Fig

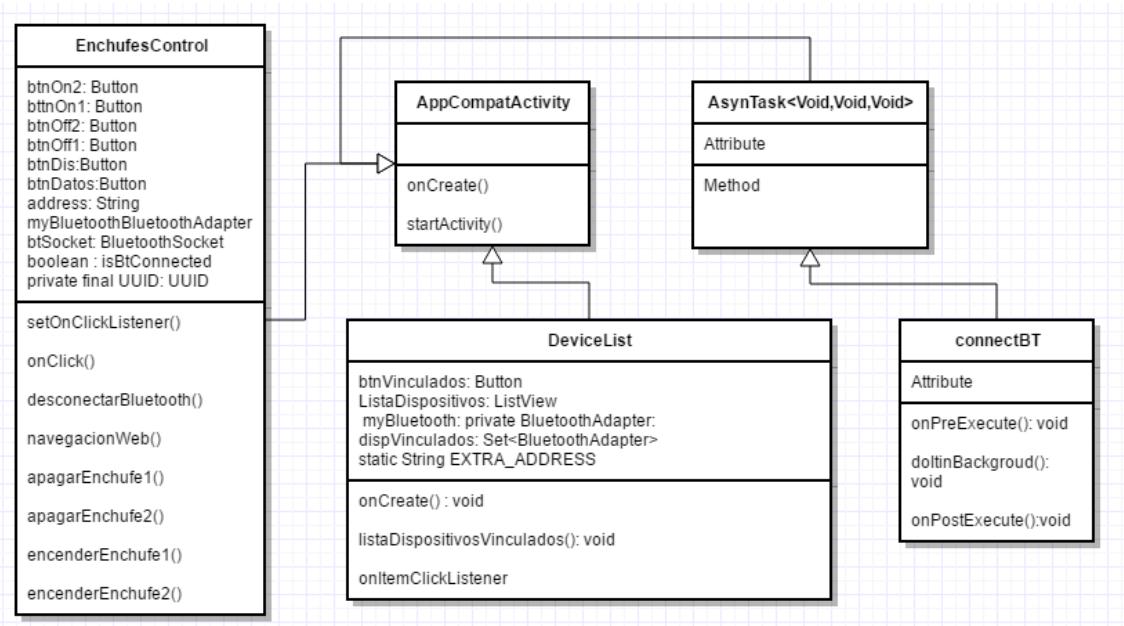


Fig.5.6.6.2.15 Diagrama Clases UML Aplicación

DeviceList.java.

Esta clase tiene como objetivo, mostrar los pares de dispositivos que tenemos vinculados en una lista. Para ello debemos hacer uso del tipo BluetoothAdapter, establecer una conexión y con ello buscar dichos dispositivos.

En la siguiente figura (Fig.5.6.6.3) vemos la declaración de los componentes. Dicha clase extiende a AppCompatActivity, de forma que se puede hacer uso de toda la funcionalidad de la action bar o barra.

```
public class DeviceList extends AppCompatActivity {
    //Declaramos Los Componentes
    Button btnVinculados;
    ListView listaDispositivos;
    //Bluetooth
    private BluetoothAdapter myBluetooth = null;
    private Set<BluetoothDevice> dispVinculados;
    public static String EXTRA_ADDRESS = "device_address";
```

Fig.5.6.6.3 Declaración atributos

El método onCreate (Fig.5.6.3.3.1) es invocado cuando la actividad se inicia. Es el lugar idóneo para inicializar la actividad; en nuestro caso preguntar el estado de nuestra conexión bluetooth y posteriormente saber si se ha hecho click en algún dispositivo de la lista.

```
protected void onCreate(Bundle savedInstanceState)
{
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_device_list);
    //Declaramos nuestros componentes relacionandolos con los del layout
    btnVinculados = (Button)findViewById(R.id.button);
    listaDispositivos = (ListView)findViewById(R.id.listView);
    //Comprobamos que el dispositivo tiene bluetooth
    myBluetooth = BluetoothAdapter.getDefaultAdapter();

    if(myBluetooth == null)
    {
        //Mostramos un mensaje, indicando al usuario que no tiene conexión bluetooth disponible
        Toast.makeText(getApplicationContext(), "Bluetooth no disponible", Toast.LENGTH_LONG).show();
        //Finalizamos la aplicación
        finish();
    }
    else if(!myBluetooth.isEnabled())
    {
        //Preguntamos al usuario si desea encender el bluetooth
        Intent turnBTon = new Intent(BluetoothAdapter.ACTION_REQUEST_ENABLE);
        startActivityForResult(turnBTon,1);
    }
    btnVinculados.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v)
        {
            listaDispositivosVinculados();
        }
    });
}
```

Fig.5.6.3.3.1 Método onCreate

Para ello elaboramos un método, `listaDispositivosVinculados()`, el cual nos añade el dispositivo que tenemos vinculado en nuestro sistema, a nuestra lista. Acto seguido, para poder seleccionar el dispositivo, debemos hacer uso del método `myListClickListener`.

```
private void listaDispositivosVinculados()
{
    dispVinculados = myBluetooth.getBondedDevices();
    ArrayList list = new ArrayList();

    if (dispVinculados.size()>0)
    {
        for(BluetoothDevice bt : dispVinculados)
        {
            list.add(bt.getName() + "\n" + bt.getAddress()); //Obtenemos los nombres y direcciones MAC de los disp. vinculados
        }
    }
    else
    {
        Toast.makeText(getApplicationContext(), "No se han encontrado dispositivos vinculados", Toast.LENGTH_LONG).show();
    }

    final ArrayAdapter adapter = new ArrayAdapter(this, android.R.layout.simple_list_item_1, list);
    listaDispositivos.setAdapter(adapter);
    listaDispositivos.setOnItemClickListener(myListClickListener);
}
```

Fig.5.6.3.3.2 Método listaDispositivosVinculados

El desarrollo del método `myListClickListener` se puede ver en la siguiente figura. (Fig.5.6.3.3.3)

```
private AdapterView.OnItemClickListener myListClickListener = new AdapterView.OnItemClickListener()
{
    public void onItemClick(AdapterView<?> av, View v, int arg2, long arg3)
    {
        //Obtenemos dirección MAC
        String info = ((TextView) v).getText().toString();
        String address = info.substring(info.length() - 17);

        // Hacemos Intent para lanzar actividad
        Intent i = new Intent(DeviceList.this, enchufesControl.class);

        //Lanzamos actividad
        i.putExtra(EXTRA_ADDRESS, address);
        startActivity(i);
    }
};
```

Fig.5.6.3.3.3 myListClickListener()

Según la documentación que nos aporta Android, un Intent, es un objeto el cual nos permite lanzar acciones desde otro componente a través de diferentes formas. Existen 3 casos:

- Lanzar una actividad:
Una actividad representa una pantalla en nuestra actividad. Podemos lanzar una nueva instancia de una actividad pasando por parámetro un objeto *Intent* al método *startActivity()*.
- Lanzar un servicio:
Un servicio es un componente que realiza operaciones en segundo plano sin hacer uso de ninguna interfaz de usuario. Podemos hacer uso de dicho servicio, pero solo se podrá realizar una sola vez. Un ejemplo, abrir el servicio Bluetooth, cámara, micrófono, descarga de un archivo... .
- Para entregar un mensaje Broadcast:
El sistema nos permite enviar mensajes broadcast para nuestro sistema de eventos, por ejemplo, cuando el sistema arranca o cuando el sistema empieza a cargar batería... .

En nuestro caso, necesitamos saber cuándo se ha pulsado el botón “*dispositivos vinculados*”, para mostrar la lista de pares de dispositivos. Para ellos usamos el método *onItemClick()* haciendo uso de un *Intent*, el cual lanza una actividad, *enchufesControl()* .

Clase: *enchufesControl.java*

En esta sección mostraremos el desarrollo de nuestra segunda actividad. Dicha actividad hace referencia al panel de control comentado en el apartado 5.6.6.2. El objetivo consiste en establecer una conexión bluetooth con el dispositivo seleccionado.

En la siguiente figura(Fig.5.6.6.4) vemos la declaración de los elementos necesarios para utilizar nuestra actividad. Factor a destacar es el uso de una variable UUID. Dicha variable está formada por 128bits los cuales identifican de forma única un servicio en particular, en nuestro caso el receptor bluetooth

En el método `Oncreate()` desarrollamos la respuesta (Fig 5.6.3.3.5) que obtenemos al hacer click, a partir de los siguientes métodos. Para ello hacemos uso del método `setOnClickListener(View v)`.

```

25 public class enchufesControl extends AppCompatActivity {
26
27     Button btnOn2,btnOn1,btnOff1, btnOff2, btnDis,btnDatos;
28     SeekBar brightness;
29     TextView lumm;
30     String address = null;
31     private ProgressDialog progress;
32     BluetoothAdapter myBluetooth = null;
33     BluetoothSocket btSocket = null;
34     private boolean isBtConnected = false;
35     static final UUID myUUID = UUID.fromString("00001101-0000-1000-8000-00805F9B34FB");
36
37     @Override
38     protected void onCreate(Bundle savedInstanceState) {
39         super.onCreate(savedInstanceState);
40         Intent newint = getIntent();
41         address = newint.getStringExtra(DeviceList.EXTRA_ADDRESS); //recibimos la mac address obtenida en la actividad anterior
42         setContentView(R.layout.activity_enchufes_control);
43         btnOn1 = (Button) findViewById(R.id.button2);
44         btnOff1 = (Button) findViewById(R.id.button3);
45         btnOn2 = (Button) findViewById(R.id.button5);
46         btnOff2 = (Button) findViewById(R.id.button6);
47         btnDatos = (Button) findViewById(R.id.button7);
48         btnDis = (Button) findViewById(R.id.button4);
49
50         new ConnectBT().execute(); //Call the class to connect
51         btnDatos.setOnClickListener(new View.OnClickListener() {
52             @Override
53             public void onClick(View v) {
54                 navegacionWeb();
55             }
56         });
57         btnOn1.setOnClickListener(new View.OnClickListener() {
58             @Override
59             public void onClick(View v) {
60                 msg("Enchufe 1 activado");
61                 encenderEnchufe1();
62             }
63         });
64         btnOn2.setOnClickListener(new View.OnClickListener() {
65             @Override
66             public void onClick(View v) {
67                 msg("Enchufe 2 activado");
68                 encenderEnchufe2();
69             }
70         });
71         btnOff1.setOnClickListener(new View.OnClickListener() {
72             @Override
73             public void onClick(View v) {
74                 msg("Enchufe 1 desconectado");
75                 apagarEnchufe1();
76             }
77         });
78         btnOff2.setOnClickListener(new View.OnClickListener() {
79             @Override
80             public void onClick(View v) {
81                 msg("Enchufe 2 desconectado");
82                 apagarEnchufe2();
83             }
84         });
85         btnDis.setOnClickListener(new View.OnClickListener() {
86             @Override
87             public void onClick(View v) {
88                 desconectarBluetooth();
89             }
90         });
91     }
92 }
```

Fig.5.6.3.3.5 setOnClickListener

A continuación, mostramos el desarrollo de los métodos mostrados en la figura anterior:

Método *desconectarBluetooth()*

```
94     private void desconectarBluetooth()
95     {
96         if (btSocket!=null)
97         {
98             try
99             {
100                 btSocket.close();
101             }
102             catch (IOException e)
103             { msg("Error");}
104         }
105         finish();
106     }
107 }
```

Fig.5.6.3.3.6 Método *desconectarBluetooth*

Método *navegacionWeb()*

```
109     public void navegacionWeb(){
110         Intent browser = new Intent(Intent.ACTION_VIEW, Uri.parse("http://www.google.com"));
111         startActivity(browser);
112     }
113 }
```

Fig.5.6.3.3.7 Método *navegacionWeb*

ApagarEnchufe1 y ApagarEnchufe2: Mandamos los caracteres 2 y 4, los cuales serán tratados por nuestro microcontrolador y con ello responderán de una forma u otra. En este caso, abrir y cerrar el relé que usa el enchufe1.

```
114     private void apagarEnchufe1()
115     {
116         if (btSocket!=null)
117         {
118             try
119             {
120                 btSocket.getOutputStream().write("2".toString().getBytes());
121             }
122             catch (IOException e)
123             {
124                 msg("Error");
125             }
126         }
127     }
128     private void apagarEnchufe2()
129     {
130         if (btSocket!=null)
131         {
132             try
133             {
134                 btSocket.getOutputStream().write("4".toString().getBytes());
135             }
136             catch (IOException e)
137             {
138                 msg("Error");
139             }
140         }
141     }
142 }
```

Fig.5.6.3.3.8 Método *apagarEnchufe1 / apagarEnchufe2*

Encender enchufe1 y enchufe2. Mandamos los caracteres 1 y 3. Serán procesados por el microcontrolador y éste actuará en consecuencia.

```
143     private void encenderEnchufe1()
144     {
145         if (btSocket!=null)
146         {
147             try
148             {
149                 btSocket.getOutputStream().write("1".toString().getBytes());
150             }
151             catch (IOException e)
152             {
153                 msg("Error");
154             }
155         }
156     }
157     private void encenderEnchufe2()
158     {
159         if (btSocket!=null)
160         {
161             try
162             {
163                 btSocket.getOutputStream().write("3".toString().getBytes());
164             }
165             catch (IOException e)
166             {
167                 msg("Error");
168             }
169         }
170     }
171 }
```

Fig.5.6.3.3.9 Método encenderEnchufe1 / encenderEnchufe2

Por último, desarrollamos una clase interna, cuya función consiste en elaborar un socket, el cual establecerá la conexión con el dispositivo final. Dicha clase extiende a AsyncTask, la cual nos permite renderizar la interfaz de la aplicación y al mismo tiempo ejecutar en segundo plano otra actividad

Dicha clase se encuentra formada por tres métodos. En desarrollo, se encuentra en la siguiente figura (Fig. 5.6.3.3.10)

```
196     private class ConnectBT extends AsyncTask<Void, Void, Void> // UI thread
197     {
198         private boolean ConnectSuccess = true;
199
200         @Override
201         protected void onPreExecute()
202         {
203             progress = ProgressDialog.show(enchufesControl.this, "Conectando...", "Espere..");
204         }
205
206         @Override
207         protected Void doInBackground(Void... devices)
208         {
209             try
210             {
211                 if (btSocket == null || !isBtConnected)
212                 {
213                     myBluetooth = BluetoothAdapter.getDefaultAdapter();
214                     //conectamos al dispositivo y chequeamos si esta disponible
215                     BluetoothDevice dispositivo = myBluetooth.getRemoteDevice(address);
216                     btSocket = dispositivo.createInsecureRfcommSocketToServiceRecord(myUUID);
217                     BluetoothAdapter.getDefaultAdapter().cancelDiscovery();
218                     btSocket.connect();
219                 }
220             }
221             catch (IOException e)
222             {
223                 ConnectSuccess = false;
224             }
225             return null;
226         }
227         @Override
228         protected void onPostExecute(Void result)
229         {
230             super.onPostExecute(result);
231
232             if (!ConnectSuccess)
233             {
234                 msg("Conexión Fallida");
235                 finish();
236             }
237             else
238             {
239                 msg("Conectado");
240                 isBtConnected = true;
241             }
242             progress.dismiss();
243         }
244     }
```

Fig. 5.6.3.3.10 Clase Interna ConnectBT

AndroidManifest

Es un archivo de configuración situado en la raíz de nuestras aplicaciones, en donde podemos aplicar las configuraciones básicas de nuestra app. Se puede configurar gráficamente, pero es recomendable conocer su sintaxis ya que en muchas ocasiones será más rápido y fácil hacerlo desde el propio XML.

En este caso, deberemos añadir permisos para que podamos hacer uso de la tecnología Bluetooth y con ello usar el objeto *BluetoothAdapter*. Lo podemos ver en la siguiente figura (Fig.5.6.3.3.11)

```
<?xml version="1.0" encoding="utf-8" ?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.led.arduino" >

    <uses-permission android:name="android.permission.BLUETOOTH_ADMIN"/>
    <uses-permission android:name="android.permission.BLUETOOTH"/>

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >
        <activity
            android:name="com.led.arduino.DeviceList"
            android:label="@string/app_name" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity
            android:name="com.led.arduino.enchufesControl"
            android:label="remoteControl" >
        </activity>
    </application>

</manifest>
```

Fig.5.6.3.3.11 AndroidManifest

5.6.7 Firmware completo microcontrolador

En capítulos anteriores hemos visto por separado las partes de las que se compone nuestro proyecto, junto con su firmware asociado. A continuación, en esta sección se pondrán en común dichas secciones para cumplir nuestro objetivo final.

5.6.7.1 Diagrama de flujo

En la siguiente figura (Fig.5.6.3.3.12), se muestra un diagrama de bloques, el cual muestra el funcionamiento global de nuestro firmware.

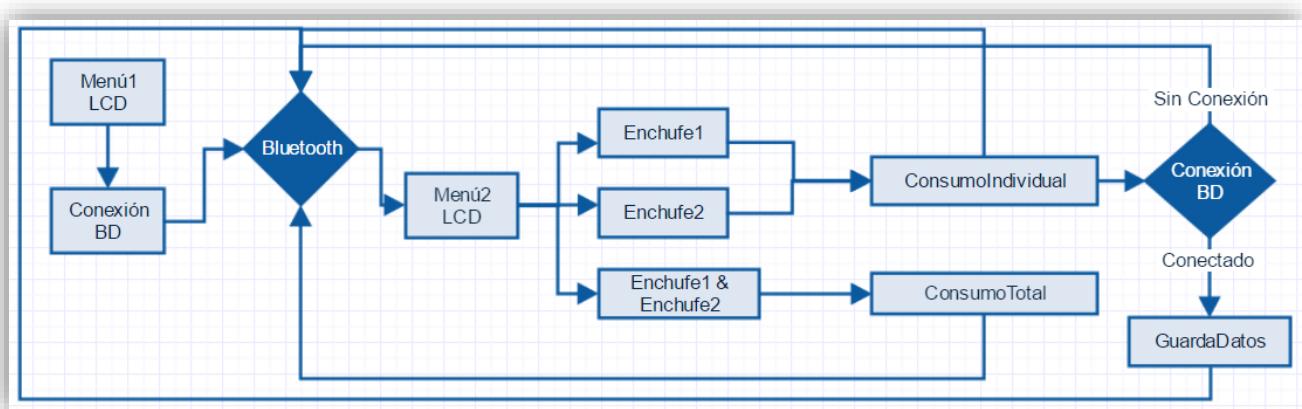


Fig.5.6.3.3.12 Diagrama bloques firmware global

A continuación, se muestra en la siguiente figura el firmware completo que usa nuestra regleta inteligente. (Fig.5.6.3.3.13)

Firmware:

```
potencia.ino      *
1 #include <Wire.h>
2 #include <SoftwareSerial.h>
3 #include <LiquidCrystal.h>
4 #include <SPI.h>
5 #include <Ethernet.h>
6 #include <sha1.h>
7 #include <mysql.h>
8 #include <stdlib.h>
9
10 //Pin sensores
11 int sensor1 = A5;
12 int sensor2 = A4;
13
14 //Consumo
15 int muestras = 1000;
16 char datos [2048];
17 float sensorValue_aux = 0.0;
18 float sensorValue_aux1 = 0.0;
19 float valorCorriente = 0.0;
20 float valorSensor = 0.0;
21 float voltsporUnidad = 0.004887586; //Conversión ADC 5/1023
22 float sensibilidad = voltsporUnidad * 10; //Sensibilidad para sensor 20A
23 float sensibilidad5A = voltsporUnidad * 5.4054; //Sensibilidad para sensor 5A usada para calculo de la potencia
24 float tension = 230.0; //Tensión de la zona. En este caso 230V
25 float pot = 0.0;
26 int intensidadQuery = 0;
27
28 //Bluetooth
29 int estado = 0;
30 int res[2];
31 int * value;
32 #define rxPin 11
33 #define txPin 12
34
35 SoftwareSerial BT1(rxPin,txPin);
36
37 //Definicion Pines de los Reles
38 const int pinRele1 = 3;
39 const int pinRele2 = 2;
40
41 //Estado de los Leds
42 const int led1 = 13;
43 const int led2 = 12;
44
45 //SoftwareSerial BT1(1,0); // RX, TX
46 //#define BT1 Serial
47 LiquidCrystal lcd(8, 9, 4, 5, 6, 7);
48 //Defincion conexion MySQL
```

```

49 IPAddress mysql_server(192,168,1,51); //Ip servidor
50 IPAddress ethernet_shield(192,168,1,10); //Ip arduino
51 char user[] = "usuario"; //Usuario Base de datos
52 char password[] = "1234"; //Contraseña
53 byte mac_addr[] = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED };
54 Connector arduino_conn; //Objeto conector
55 int estadoConexion = 0;
56 void setup() {
57     Serial.begin(9600);
58     pinMode(pinRele1,OUTPUT);
59     pinMode(pinRele2,OUTPUT);
60     pinMode(sensor1,INPUT);
61     pinMode(sensor2,INPUT);
62     BT1.begin(9600);
63     lcd.begin(16,2);
64     pinMode(led1,OUTPUT);
65     pinMode(led2,OUTPUT);
66     lcd.print("/** BIENVENIDO **");
67     digitalWrite(led1,HIGH);
68     digitalWrite(led2,HIGH);
69     delay(2000);
70     lcd.setCursor(0,1);
71     lcd.print("Cargando...");
72     estadoConexion = Conexion();
73     delay(2000);
74     digitalWrite(led1,LOW);
75     digitalWrite(led2,LOW);
76     lcd.clear();
77 }
78 }
79
80 void loop() {
81     int * value = bluetooth();
82     displayMenu(value);
83 }
84
85 void displayMenu(int value[]){
86     int enchufe1= value[0];
87     int enchufe2= value[1];
88     if(enchufe1==0){
89         if(enchufe2==0){
90             lcd.clear();
91             lcd.setCursor(0,1);
92             lcd.print("Esperando orden");
93             delay(1000);
94         }else{
95             lcd.clear();
96             lcd.setCursor(0,0);
97             lcd.print("Enchufe 2 ON");
98             lcd.setCursor(0,1);
99             lcd.print("Leyendo datos..");
100            digitalWrite(led1,HIGH);
101            delay(1000);
102        }
103    }else{
104        if(enchufe2==0){
105            Serial.println("Enchufe1 Activado");
106            lcd.clear();
107            lcd.setCursor(0,0);
108            lcd.print("Enchufe 1 ON");
109            lcd.setCursor(0,1);
110            lcd.print("Leyendo datos..");
111            digitalWrite(led2,HIGH);
112            delay(1000);
113        }
114    }else{
115        lcd.clear();
116        lcd.setCursor(0,0);
117        lcd.print("Enchufes 1-2 ON");
118        lcd.setCursor(0,1);
119        lcd.print("Leyendo datos..");
120        digitalWrite(led1,HIGH);
121        digitalWrite(led2,HIGH);
122        delay(1000);
123    }
124 }
125 }
```

```

126     if(enchufe1==1 and enchufe2==0){ //Enchufe 1 ON, Enchufe2 OFF
127         int valor = 1;
128         consumoIndividual(valor,estadoConexion);
129     }
130     if(enchufe2==1 and enchufe1==0){ //Enchufe 1 OFF, Enchufe2 ON
131         int valor = 2;
132         consumoIndividual(valor,estadoConexion);
133     }
134     if(enchufe1==1 and enchufe2==1){ //Enchufe 1 ON, Echufe 2 ON
135         consumoTotal(enchufe1,enchufe2);
136     }
137 }
138 void consumoIndividual(int enchufe, int conectado){
139     //Enchufe 1 con sensor modelo 5A
140     if(enchufe==1){
141         //Consulta base de datos Planta1
142         lcd.setCursor(0,1);
143         lcd.print("Leyendo datos...");
144         for(int i=0; i<muestras; i++){
145             sensorValue_aux = sensibilidad5A * (analogRead(sensor1) - 510);
146             if(sensorValue_aux < 0 ) sensorValue_aux = - sensorValue_aux; //Rectificacion de la componente - AC
147             valorSensor = valorSensor + sensorValue_aux / float(muestras); //Voltaje promedio de AC rectificada
148         }
149         valorCorriente = 1.1107 * valorSensor;
150         if(valorCorriente <= 0.05){ //+-0.05 aprox. valor experimental
151             valorCorriente = 0.000;
152         }
153         Serial.println(valorCorriente);
154         lcd.clear();
155         lcd.setCursor(0,0);
156         //Tratamos error
157         valorSensor = 0;
158         Serial.println(valorCorriente);
159         lcd.clear();
160         lcd.setCursor(0,0);
161         pot = valorCorriente * tension;
162         intensidadQuery = 0 + valorCorriente * 1000;
163         if(conectado == 1){
164             if(pot != 0.00 || intensidadQuery != 0){
165                 lcd.print("Guardando...");
166                 char * Query = "INSERT INTO sensor1 (INTENSIDAD,DATE,TIME) VALUE (%d,CURRENT_DATE,CURRENT_TIME)";
167                 sprintf(datos,Query,intensidadQuery);
168                 Serial.println("Datos");
169                 Serial.println(datos);
170                 arduino_conn.cmd_query("use arduino");//Usamos tabla arduino
171                 arduino_conn.cmd_query(datos);
172                 lcd.clear();
173             }
174         }
175         lcd.print("Corriente:");
176         lcd.print(valorCorriente,3);
177         lcd.print("A");
178         //Cambiemos de cursor
179         lcd.setCursor(0,2);
180         lcd.print("Consumo:");
181         lcd.print(pot);
182         lcd.print("W");
183         lcd.setCursor(0,3);
184         valorSensor = 0;
185         valorCorriente = 0;
186         delay(2500);
187     }
188     else{
189         //Enchufe 2 con sensor modelo 20
190         lcd.setCursor(0,1);
191         lcd.print("Leyendo datos...");
192         for(int i=0; i<muestras; i++){ //Sensor2 Rele 2
193             sensorValue_aux = sensibilidad * (analogRead(sensor2) - 510);
194             if(sensorValue_aux < 0 ) sensorValue_aux = - sensorValue_aux;
195             valorSensor = valorSensor + sensorValue_aux / float(muestras);
196         }
197         valorCorriente = 1.1107 * valorSensor;
198         if(valorCorriente <= 0.06){ //+-0.06 aprox. valor experimental
199             valorCorriente = 0.000;
200         }
201         lcd.clear();
202         lcd.setCursor(0,0);
203         pot = valorCorriente * tension;
204         intensidadQuery = valorCorriente * 1000;
205         if(conectado ==1){
206             if(pot != 0.00 || intensidadQuery != 0){
207                 lcd.print("Guardando...");

```

```

208     char * Query = "INSERT INTO sensor2 (INTENSIDAD,DATE,TIME) VALUE (%d,CURRENT_DATE,CURRENT_TIME)";
209     sprintf(datos,Query,intensidadQuery);
210     //Usamos tabla arduino
211     arduino_conn.cmd_query("use arduino");
212     arduino_conn.cmd_query(datos);
213     Serial.println("Inserting data...");
214     Serial.println(datos);
215     lcd.clear();
216     }
217   }
218   lcd.print("Corriente:");
219   lcd.print(valorCorriente,3);
220   lcd.print("A");
221   //Cambiamos de cursor
222   lcd.setCursor(0,2);
223   lcd.print("Consumo:");
224   lcd.print(pot);
225   lcd.print("W");
226   lcd.setCursor(0,3);
227   valorCorriente = 0;
228   valorSensor = 0;
229   delay(2500);
230   lcd.clear();
231   lcd.print("Guardando....");
232   delay(2500);
233   }
234 }
235 int * bluetooth(){
236
237 if(BT1.available()){
238   estado = BT1.read();
239 }
240 if(estado == '1'){
241   digitalWrite(pinRele1,HIGH);
242   res[0] = 1;
243 }
244 if(estado == '2'){
245   digitalWrite(pinRele1,LOW);
246   res[0] = 0;
247   lcd.clear();
248   lcd.setCursor(0,0);
249   lcd.print("Enchufe 1 OFF");
250   digitalWrite(led1,LOW);
251   delay(1000);
252   lcd.clear();
253 }
254 if(estado == '3'){
255   digitalWrite(pinRele2,HIGH);
256   res[1]=1;
257 }
258 if(estado == '4'){
259   digitalWrite(pinRele2,LOW);
260   res[1]= 0;
261   lcd.clear();
262   lcd.setCursor(0,0);
263   lcd.print("Enchufe 2 OFF");
264   digitalWrite(led2,LOW);
265   delay(1000);
266   lcd.clear();
267 }
268 return res;
269 }
270 //Calcula potencia total de los 2 enchufes
271 void consumoTotal( int enchufe1, int enchufe2){
272   //Consumo Enchufe1 y Enchufe2
273   if(enchufe1 == 1 and enchufe2 == 1){
274     for(int i=0; i<muestras; i++){
275       sensorValue_aux = (sensibilidad5A * (analogRead(sensor1) - 510)) + (sensibilidad * (analogRead(sensor2) - 510));
276       if(sensorValue_aux < 0 ) sensorValue_aux = - sensorValue_aux;
277       valorSensor = valorSensor + sensorValue_aux / double(muestras);
278     }
279     valorCorriente = 1.1107 * valorSensor;
280     if(valorCorriente <= 0.08){
281       valorCorriente = 0.00;
282     }
283     valorSensor = 0;
284     lcd.clear();
285     lcd.setCursor(0,0);
286     int pot = valorCorriente * tension;
287     lcd.print("Corriente:");
288     lcd.print(valorCorriente,3);
289     lcd.print("A");

```

```

290     //Cambiamos de cursor
291     lcd.setCursor(0,2);
292     lcd.print("Consumo:");
293     lcd.print(pot);
294     lcd.print("Watts");
295     lcd.setCursor(0,3);
296     delay(5000);
297   }
298 }
299 int Conexion(){
300   Ethernet.begin(mac_addr, ethernet_shield);
301   int res = 0;
302   delay(1000);
303   lcd.clear();
304   lcd.setCursor(0,2);
305   lcd.print("Conectando...");
306   if(arduino_conn.mysql_connect(mysql_server,3306,user,password)){
307     res= 1;
308     lcd.clear();
309     lcd.print("Conectado.");
310     delay(1000);
311   }
312   else{
313     res = 0;
314     lcd.clear();
315     lcd.print("Sin conexion");
316     delay(1000);
317   }
318   return res;
319 }

```

Fig.5.6.3.3.13 Firmware completo microcontrolador

6. DISEÑO APPLICACIÓN WEB

En este apartado, describiremos como estará constituida la aplicación web. Mostraremos su funcionamiento a través de un diagrama de flujo y el diseño con Mockups.

La función que le otorgamos al diseño de una aplicación web, es para el uso de un monitor de energía. La aplicación mostrará el consumo en tiempo real, los kW/hora, el total consumido del día actual, semanal y mensual.

La aplicación está constituida por una zona privada, la cual sólo se podrá acceder previamente si el usuario, está dado de alta en la base de datos. Por mayor simplicidad en la gestión de usuarios, hemos supuesto que el usuario final, a la hora de adquirir el producto, lleva un usuario único y contraseña. Un aspecto a mejorar es incorporar un formulario que permita el cambio de contraseña para acceder al sistema.

6.6.1 Diagrama de flujo

Para el diseño de nuestra aplicación web hemos utilizado el diagrama de flujo de la siguiente figura (Fig.6.6.1)

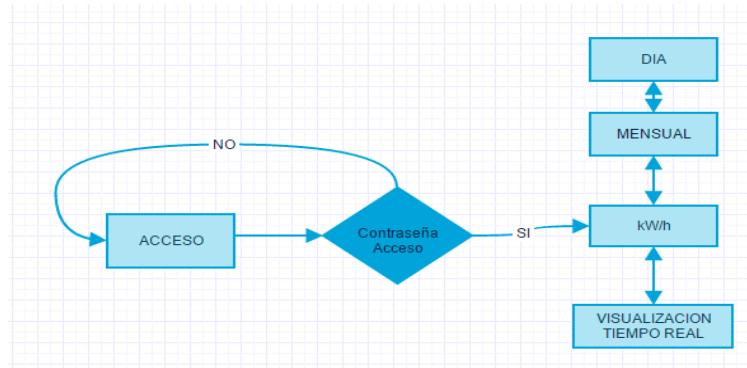


Fig.6.6.1 Diagrama Flujo Aplicación Web

Cuando accedemos mediante un navegador web, introduciendo la IP proporcionada (mostrada en el monitor), nos encontramos con la página de acceso.

En caso de introducir el usuario y contraseña de acceso correcta, nos mostrará el consumo en tiempo real y el menú siguiente de la figura (Fig.6.6.1.1).

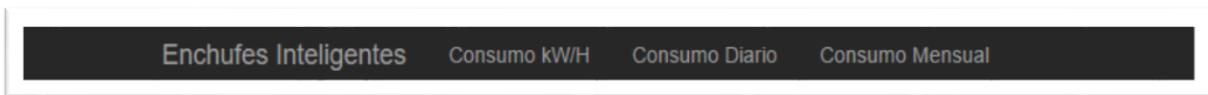


Fig.6.6.1.1 Cabecera web

6.6.2 Realización de Mockups

Tras realizar el diagrama de flujo, vamos a diseñar unos Mockups (simulación interactiva de la misma), para fabricar un prototipo de nuestra aplicación para ello hemos utilizado **mockups**, versión web de prueba.

- Página Inicio de sesión (Fig.6.6.2)
- Página consumo en tiempo real (Fig.6.6.2.1)
- Página consumo kW/h (Fig.6.6.2.2)
- Página consumo diario (Fig.6.6.2.3)
- Página consumo mensual (Fig.6.6.2.4)
- Fig.6.6.2 Página Inicio Sesión

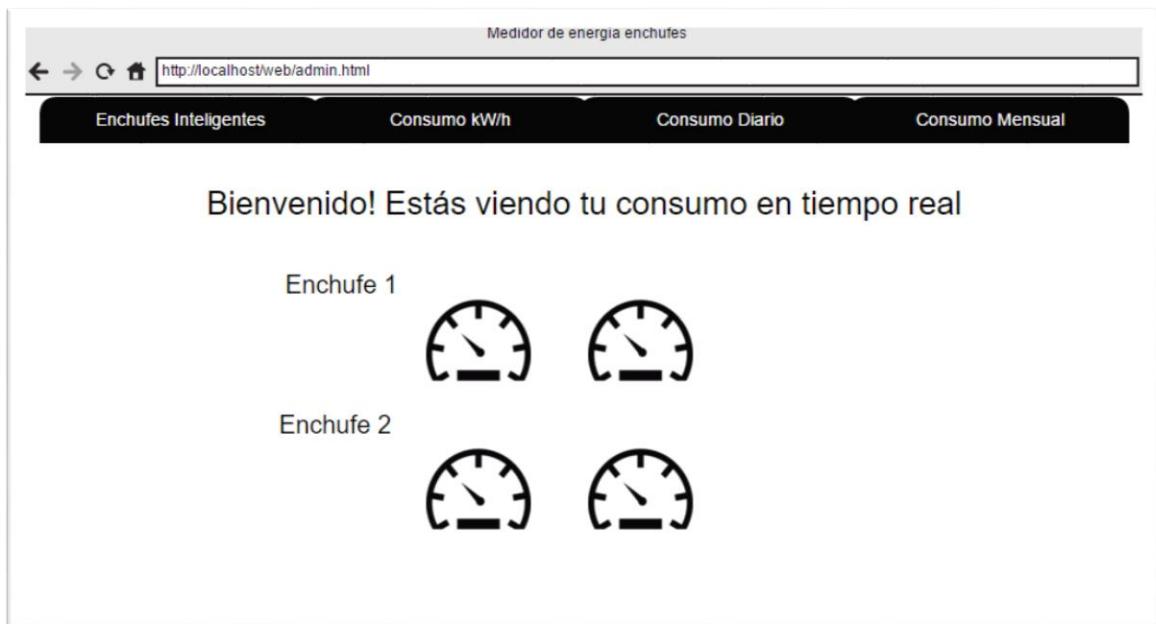


Fig.6.6.2.1 Página consumo tiempo real

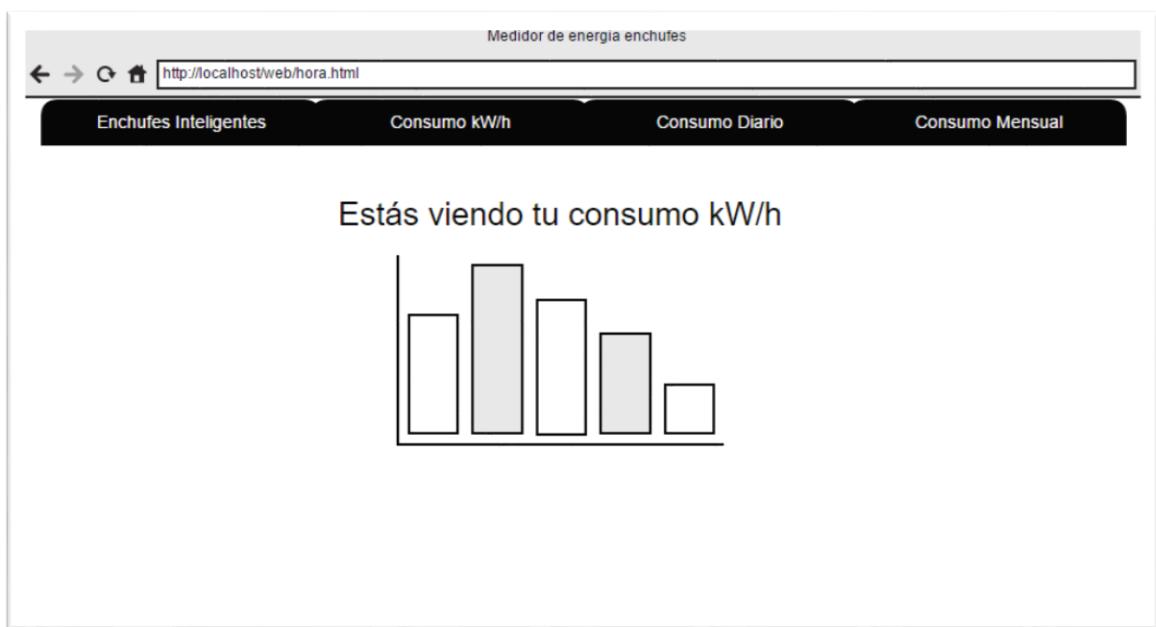


Fig.6.6.2.2 Página consumo kW/h



Fig.6.6.2.3 Página consumo diario

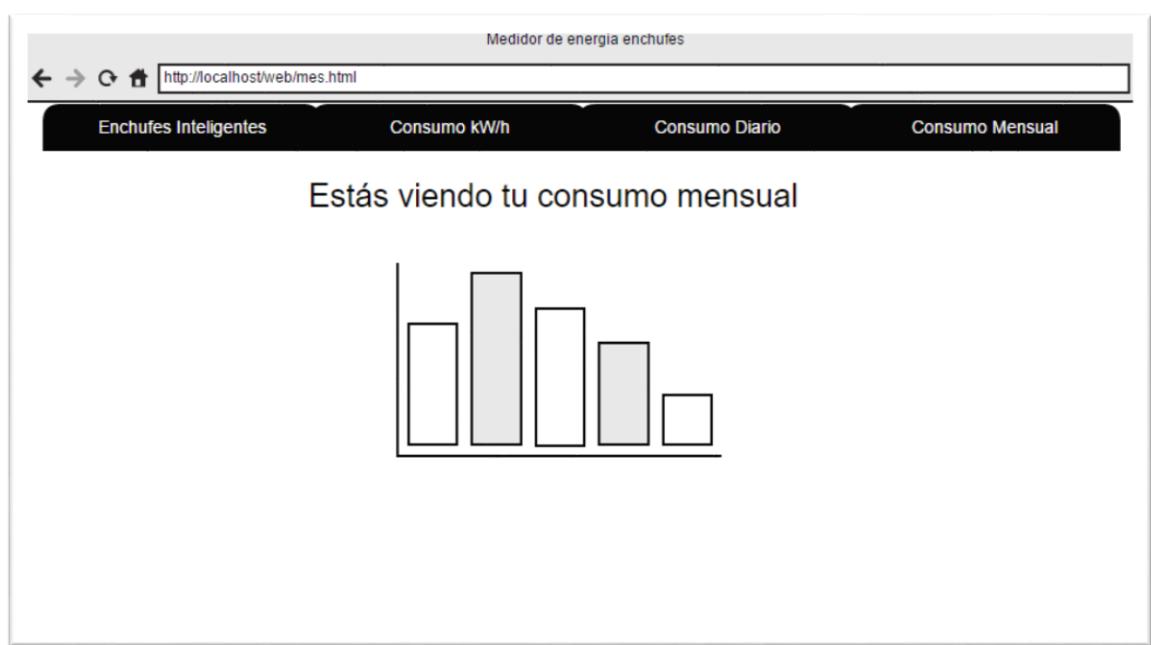


Fig.6.6.2.4 Página consumo mensual

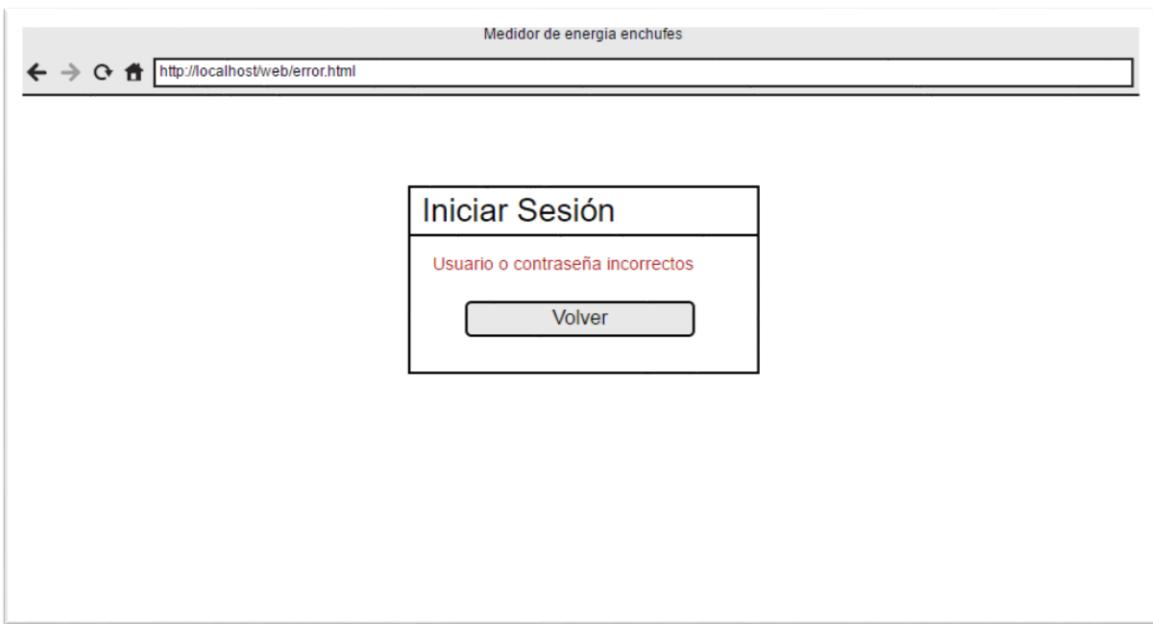


Fig.6.6.2.5 Página error

6.6.3 Aplicación Web

Una vez diseñado los Mockups pasamos al desarrollo de la aplicación web.

Para la implementación de las aplicaciones web nos hemos decidido por Sublime Text 2, un editor de texto pensado para escribir código en la mayoría de lenguajes de programación y formatos documentales de texto, utilizados en la actualidad:

Java, Python, Perl, HTML, JavaScript, CSS, XML, PHP, C, C++, etc.

Permite escribir todo tipo de documentos de código en formato de texto y es capaz de colorear el código, ayudarnos a la escritura, corregirnos mientras escribimos y personalizar hasta el último detalle.

Algunos de los motivos de la elección de sublime text 2 fueron:

- Fácil uso e interfaz agradable.
- Permite codificar en casi cualquier lenguaje.
- Es muy ligero. Ocupa apenas siete megabytes, por lo que no consume apenas recursos en el ordenador.
- Fácil uso e interfaz agradable.
- Es multiplataforma

Tecnologías Web

Para el desarrollo de la aplicación web hemos empleado 4 tecnologías distintas: PHP, JavaScript, HTML, CSS y un toque de AJAX. Todos ellos han sido estudiados durante la carrera.

PHP

PHP, ya que es un lenguaje de programación del lado del servidor, además es uno de los lenguajes para web más populares y dispone de una de las comunidades más grandes de desarrolladores.

JavaScript

JavaScript es un lenguaje de programación interpretado, definido como orientado a objetos, basado en prototipos, débilmente tipado (no controla los tipos de las variables) y dinámico (los tipos de las variables se deciden en tiempo de ejecución) que hemos utilizado para mejorar la apariencia del usuario haciendo la interfaz más dinámica y atractiva.**HTML y CSS**

También hemos utilizado HTML, que se trata del principal lenguaje a la hora de construir páginas web, y CSS que es un lenguaje que se encarga de definir los estilos usados en una página web.

Librerías utilizadas

Las librerías usadas para la realización de las gráficas y diseño de la aplicación son las siguientes:

Google Charts

Para la realización de las gráficas, usaremos Google Charts ya que cuenta con una magnífica API Javascript para desarrolladores en la cual podemos probar los distintos gráficos que nos ofrece antes de ponernos a implementar y ver un modelo aproximado de lo que será nuestro modelo final.

Bootstrap

Para el diseño de la aplicación, botones, texto..., hemos utilizado Bootstrap, ya que nos proporciona un conjunto de herramientas gratuitas para crear sitios web y aplicaciones web.

Desarrollo de la aplicación web

Una vez definido los prototipos de la aplicación web, así como elegidas las librerías y las tecnologías a usar, vamos a crear cada una de las páginas que compone la aplicación web.

Página de acceso:

Página de ingreso controlado (Fig.6.6.3), esta página está formada por un formulario que nos solicita un usuario y una contraseña. Por defecto el fabricante nos aporta una contraseña por defecto para el ingreso, tal y como se explicó en el diagrama de flujo.

Una vez enviado el formulario, la aplicación compara si estos valores coinciden con los valores de la tabla USUARIO, si coinciden, con usuario y contraseña de acceso (por defecto "admin", "admin"), se accede al sistema.

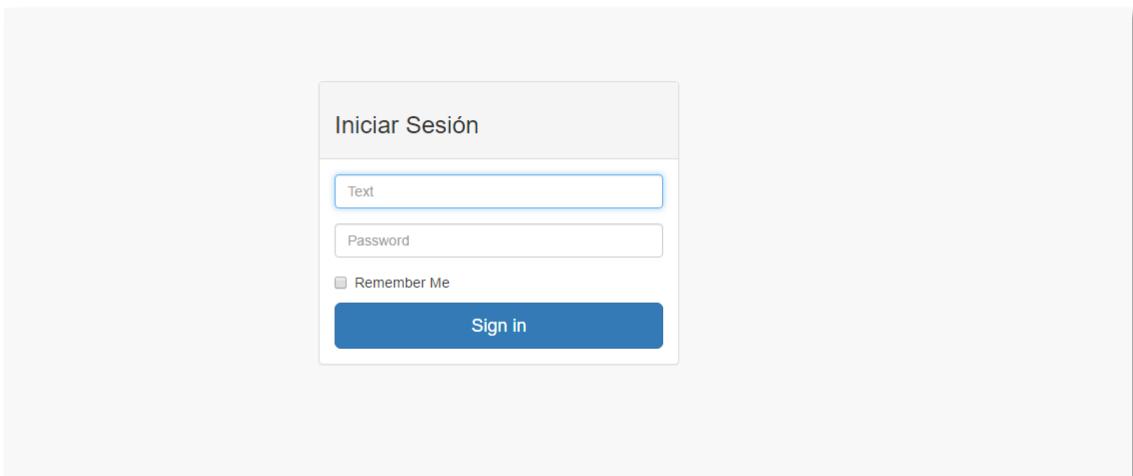


Fig.6.6.3 Vista página Inicio Sesión

Página consumo en tiempo real:

Una vez que accedemos, nos encontramos con la página que nos muestra el consumo en tiempo real que procesa el sistema.

Todos los datos que muestra pertenecen a la tabla Sensor1 y Sensor2. En ella aparecen como campos la intensidad y potencia asociada a cada sensor, así como también su fecha y hora.

Cada par del medidor gráfico, hace referencia a un enchufe, el cual nos muestra el consumo real e intensidad. Lo podemos ver en la siguiente figura. (Fig.6.6.3.1).



Fig.6.6.3.1 Vista página consumo tiempo real

Página consumo kW/h

Esta página (Fig.6.6.3.2) nos muestra el consumo total que obtenemos en una hora. Está formada por una gráfica al estilo de barras verticales, cuyas coordenadas representan los kW que se generan en la hora actual del sistema.



Fig.6.6.3.2 Vista página consumo kW/h

Página consumo diario

Dicha página nos muestra el consumo total que se da en el día actual. Representan los kW que se generan a lo largo del día. La siguiente figura(Fig.6.6.3.2) muestra lo anteriormente comentado.



Fig.6.6.3.2 Vista página consumo día actual

Página consumo mensual En esta página(Fig.6.6.3.3) podemos ver un histórico del consumo por mes. Al igual que las anteriores, el sistema de representación es el mismo; kW/h frente a mes correspondiente.



Fig.6.6.3.3 Vista página consumo mensual

Página de error

En la siguiente figura(Fig.6.6.3.5) se muestra la página de error. A dicha página se accede si el proceso de autenticación ha fallado. El botón volver, nos redirecciona a la página de acceso.

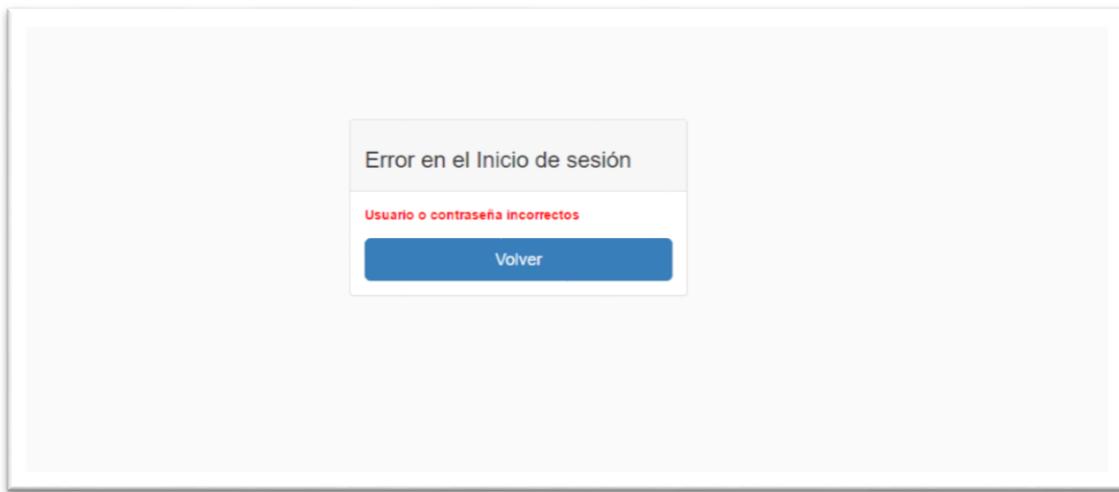


Fig.6.6.3.5 Vista página error

6.6.4 Código Aplicación Web

En este apartado, mostraremos el código implantado en nuestra página web. Para su programación hemos usado el editor sublime text 2

Código página de acceso:

Formada por una parte visual llamada Index.html (Fig.6.6.4.1) y una parte lógica llamada login.php (Fig.6.6.4.2)

```
index.html
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4     <meta charset="utf-8">
5     <meta http-equiv="X-UA-Compatible" content="IE=edge">
6     <meta name="viewport" content="width=device-width, initial-scale=1">
7     <meta name="description" content="">
8     <meta name="author" content="">
9     <title>INICIAR SESIÓN</title>
10    <!-- Bootstrap Core CSS -->
11    <link href="bower_components/bootstrap/dist/css/bootstrap.min.css" rel="stylesheet">
12    <!-- MetisMenu CSS -->
13    <link href="bower_components/metisMenu/dist/metisMenu.min.css" rel="stylesheet">
14    <!-- Custom CSS -->
15    <link href="dist/css/sb-admin-2.css" rel="stylesheet">
16    <!-- Custom Fonts -->
17    <link href="bower_components/font-awesome/css/font-awesome.min.css" rel="stylesheet" type="text/css">
18 </head>
19 <body>
20     <div class="container">
21         <div class="row">
22             <div class="col-md-4 col-md-offset-4">
23                 <div class="login-panel panel panel-default">
24                     <div class="panel-heading">
25                         <h3>Iniciar Sesión</h3>
26                     </div>
27                     <div class="panel-body">
28                         <form role="form" action="http://localhost/login.php" method="post" name="ingreso">
29                             <fieldset>
30                                 <div class="form-group">
31                                     <input class="form-control" placeholder="Text" name="email" type="text" autofocus>
32                                 </div>
33                                 <div class="form-group">
34                                     <input class="form-control" placeholder="Password" name="password" type="password" value="">
35                                 </div>
36                                 <div class="checkbox">
37                                     <label>
38                                         <input name="remember" type="checkbox" value="Remember Me">Remember Me
39                                     </label>
40                                 </div>
41                                 <!-- Change this to a button or input when using this as a form -->
42                                 <button class = "btn btn-lg btn-primary btn-block" name = "login" value="login" type="submit">
43                                     Sign in</button>
44                             </fieldset>
45                         </form>
46                     </div>
47                 </div>
48             </div>
49         </div>
50     </div>
51     <!-- jQuery -->
52     <script src="bower_components/jquery/dist/jquery.min.js"></script>
53     <!-- Bootstrap Core JavaScript -->
54     <script src="bower_components/bootstrap/dist/js/bootstrap.min.js"></script>
55     <!-- Metis Menu Plugin JavaScript -->
56     <script src="bower_components/metisMenu/dist/metisMenu.min.js"></script>
57     <!-- Custom Theme JavaScript -->
58     <script src="dist/js/sb-admin-2.js"></script>
59 </body>
60 </html>
```

Fig.6.6.4.1 Index.html

```
login.php
1  <head>
2    <title>Check Login</title>
3    <meta charset="UTF-8">
4  </head>
5  <body>
6
7  <?php
8  error_reporting(E_ERROR);
9  $host_db = "127.0.0.1";
10 $user_db = "prueba";
11 $password_db ="1234";
12 $db_name = "arduino";
13 $tbl_name = "usuario";
14 $conexion = mysqli_connect($host_db, $user_db, $password_db,$db_name) or die("Cannot Connect to DataBase");
15
16 if(mysqli_connect_errno()){
17   exit();
18 }
19
20 $username = mysqli_real_escape_string($conexion,$_POST["email"]);
21 $password = mysqli_real_escape_string($conexion,$_POST["password"]);
22 $query = "SELECT * FROM usuario where USUARIO ='$username'AND CONTRASENA ='$password'";
23 $resultado = mysqli_query($conexion,$query);
24 if(mysqli_num_rows($resultado) > 0){
25   header("Location: http://localhost/admin.html");
26 }
27 else{
28   header("Location: http://localhost/error.html");
29   echo "<p>Usuario o contraseña incorrecto</p>";
30 }
31 mysqli_close($conexion);
32 ?>
33 </body>
```

Fig.6.6.4.2 Login.php

Código página consumo en tiempo real

Llamada Admin.html (Fig.6.6.4.3) la cual hace uso de Actual.php (Fig.6.6.4.4) y Actual2.php (Fig.6.6.4.5).

Hacemos uso de 2 funciones Javascript que nos ofrece la librería Google Charts y agregamos las opciones de configuración necesarias.

```
admin.html
1  <!DOCTYPE html>
2  <html lang="es">
3  <head>
4      <meta charset="utf-8">
5      <meta http-equiv="X-UA-Compatible" content="IE=edge">
6      <meta name="viewport" content="width=device-width, initial-scale=1">
7      <meta name="description" content="">
8      <meta name="author" content="">
9      <title>Medidor energía enchufes</title>
10     <!-- Bootstrap Core CSS -->
11     <link href="pages/css/bootstrap.min.css" rel="stylesheet">
12     <script src="https://ajax.googleapis.com/ajax/libs/jquery/2.2.0/jquery.min.js"></script>
13     <script type="text/javascript" src="https://www.gstatic.com/charts/loader.js"></script>
14     <!-- Custom CSS -->
15     <style>
16     body {
17         padding-top: 60px;
18         /* Required padding for .navbar-fixed-top. Remove if using .navbar-static-top. Change if height of navigation changes. */
19     }
20     </style>
21     <script type="text/javascript">
22     google.charts.load('current', {'packages':['gauge']});
23     google.charts.setOnLoadCallback(drawChart1);
24     function drawChart1() {
25         var data = google.visualization.arrayToDataTable([
26             ['Label', 'Value'],
27             ['Intensidad', 0],
28             ['Potencia(W)', 0]
29         ]);
30         var options = {
31             width: 400, height: 400,
32             redFrom: 9000, redTo: 10000,
33             yellowFrom:7500, yellowTo: 9000,
34             minorTicks: 5,
35             min: 0,
36             max:10000
37         };
38         var data1 = google.visualization.arrayToDataTable([
39             ['Label', 'Value'],
40             ['Intensidad', 0],
41             ['Potencia(W)', 0]
42         ]);
43         var chart = new google.visualization.Gauge(document.getElementById('Medidores'));
44         var chart1 = new google.visualization.Gauge(document.getElementById('Medidores1'))
45         chart.draw(data, options);
46         chart1.draw(data1, options);
47
48         setInterval(function() {
49             var JSON=$.ajax({
50                 url:"http://localhost/actual.php",
51                 dataType: 'json',
52                 async: false}).responseText;
53             var Respuesta =jQuery.parseJSON(JSON);
54
55             data.setValue(0, 1,Respuesta[0].INTENSIDAD);
56             data.setValue(1, 1,Respuesta[0].POTENCIA);
57             chart.draw(data, options);
58         }, 1300);
59
60         setInterval(function() {
61             var JSON1=$.ajax({
62                 url:"http://localhost/actual2.php",
63                 dataType: 'json',
64                 async: false}).responseText;
65             var Respuesta1=jQuery.parseJSON(JSON1);
66
67             data1.setValue(0, 1,Respuesta1[0].INTENSIDAD);
68             data1.setValue(1, 1,Respuesta1[0].POTENCIA);
69             chart1.draw(data1, options);
70         }, 1300);
71     }
72     </script>
73 </head>
74 <body>
```

```

75      <!-- Navigation -->
76      <nav class="navbar navbar-inverse navbar-fixed-top" role="navigation">
77          <div class="container">
78              <!-- Brand and toggle get grouped for better mobile display -->
79              <div class="navbar-header">
80                  <button type="button" class="navbar-toggle" data-toggle="collapse" data-target="#bs-example-navbar-collapse-1">
81                      <span class="sr-only">Toggle navigation</span>
82                      <span class="icon-bar"></span>
83                      <span class="icon-bar"></span>
84                      <span class="icon-bar"></span>
85                  </button>
86                  <a class="navbar-brand" href="http://localhost/admin.html">Enchufes Inteligentes</a>
87              </div>
88              <!-- Collect the nav links, forms, and other content for toggling -->
89              <div class="collapse navbar-collapse" id="bs-example-navbar-collapse-1">
90                  <ul class="nav navbar-nav">
91                      <li>
92                          <a href="http://localhost/hora.html">Consumo kWh/H</a>
93                      </li>
94                      <li>
95                          <a href="http://localhost/dia.html">Consumo Diario</a>
96                      </li>
97                      <li>
98                          <a href="http://localhost/mes.html">Consumo Mensual</a>
99                      </li>
100                  </ul>
101              </div>
102          <!-- /.navbar-collapse -->
103      </div>
104      <!-- /.container -->
105  </nav>
106  <!-- Page Content -->
107  <div class="container">
108      <div class="row">
109          <div class="col-lg-12 text-center">
110              <h1>Bienvenido! Estás viendo tu consumo en tiempo real</h1>
111          </div>
112      </div>
113      <div class="col-md-3 col-md-offset-3">
114          <ul class="list-unstyled">
115              <li><h3>Enchufe 1</h3></li>
116          </ul>
117      </div>
118      <div class="col-md-3 col-md-offset-4" id="Medidores"></div>
119      <div class="col-md-3 col-md-offset-3">
120          <ul class="list-unstyled">
121              <li><h3>Enchufe 2</h3></li>
122          </ul>
123      </div>
124      <div class="col-md-3 col-md-offset-4" id="Medidores1"></div>
125  </div>
126  <!-- /.container -->
127  <!-- jquery Version 1.11.1 -->
128  <script src="js/jquery.js"></script>
129  <!-- Bootstrap Core JavaScript -->
130  <script src="js/bootstrap.min.js"></script>
131
132 </body>
133 </html>

```

Fig.6.6.4.3 Admin.html

A través de Actual.php (Fig.6.6.4.4) recogemos los datos obtenidos por el sensor1, caso contrario para Actual2.php (Fig.6.6.4.5).

```

actual.php * 
1 <?php
2 $host_db = "127.0.0.1";
3 $user_db = "prueba";
4 $password_db ="1234";
5 $db_name = "arduino";
6 $tbl_name = "usuario";
7 $data=array();
8 $conexion = mysqli_connect($host_db,$user_db, $password_db,$db_name) or die("Cannot Connect to DataBase");
9
10 try{
11     //query = "SELECT INTENSIDAD, POTENCIA FROM sensores ORDER BY TIME DESC LIMIT 1";
12     $query = "SELECT SENSOR1.INTENSIDAD AS INTENSIDAD from SENSOR1 ORDER BY SENSOR1.TIME DESC limit 1";
13     $resultado = mysqli_query($conexion,$query);
14
15     if(mysqli_num_rows($resultado) == 0){
16         $error = "error";
17         echo $error;
18     }
19     else{
20         while ($row= mysqli_fetch_array($resultado, 1)) {
21             $Intensidad = $row['INTENSIDAD'];
22             $Potencia = ($Intensidad / 1000) * 230.0;
23         }
24         $data[] = array('POTENCIA' =>($Potencia), 'INTENSIDAD' => ($Intensidad));
25         echo json_encode($data);
26     }
27 }
28 catch(Exception $e){
29     die($error);
30     $error = "error";
31     echo $error;
32 }
33 ?>
```

Fig.6.6.4.4 Actual.php

```

actual2.php *
1 <?php
2 $host_db = "127.0.0.1";
3 $user_db = "prueba";
4 $password_db ="1234";
5 $db_name = "arduino";
6 $tbl_name = "usuario";
7 $data=array();
8 $conexion = mysqli_connect($host_db, $user_db, $password_db,$db_name) or die("Cannot Connect to DataBase");
9
10 try{
11     //query = "SELECT INTENSIDAD1, POTENCIA1 FROM sensores ORDER BY TIME DESC LIMIT 1";
12     $query = "SELECT SENSOR2.INTENSIDAD AS INTENSIDAD2 from SENSOR2 ORDER BY SENSOR2.TIME DESC LIMIT 1";
13     $resultado = mysqli_query($conexion,$query);
14
15     if(mysqli_num_rows($resultado) == 0){
16         $error = "error";
17         echo $error;
18     }
19     else{
20         while ($row= mysqli_fetch_array($resultado, 1)) {
21             $Intensidad = $row['INTENSIDAD2'];
22             $Potencia = ($Intensidad /1000) * 230.0;
23         }
24         $data[] = array('POTENCIA2' =>($Potencia), 'INTENSIDAD2' => ($Intensidad));
25         echo json_encode($data);
26     }
27 }
28 catch(Exception $e){
29     die($error);
30     $error = "error";
31     echo $error;
32 }
33 ?>
```

Fig.6.6.4.5 Actual2.php

Código página consumo kW/h

```
hora.html      *
1  <!DOCTYPE html>
2  <html lang="es">
3  <head>
4      <meta charset="utf-8">
5      <meta http-equiv="X-UA-Compatible" content="IE=edge">
6      <meta name="viewport" content="width=device-width, initial-scale=1">
7      <meta name="description" content="">
8      <meta name="author" content="">
9      <title>Medidor energía enchufes</title>
10     <!-- Bootstrap Core CSS -->
11     <link href="pages/css/bootstrap.min.css" rel="stylesheet">
12     <script src="https://ajax.googleapis.com/ajax/libs/jquery/2.2.0/jquery.min.js"></script>
13     <script type="text/javascript" src="https://www.gstatic.com/charts/loader.js"></script>
14     <!-- Custom CSS -->
15     <style>
16     body {
17         padding-top: 60px;
18     }
19     </style>
20
21     <script type="text/javascript">
22     google.charts.load('current', {packages: ['corechart','bar']});
23     google.charts.setOnLoadCallback(drawBasic);
24
25     function drawBasic() {
26         var datosTabla = [];
27         var request = $.ajax({
28             type :"POST",
29             dataType :"json",
30             url : "http://localhost/hora.php",
31         }).done(function(data, textStatus, jqXHR) {
32             var i=0;
33
34             datosTabla = [ [ 'HORA', 'kW' ] ];
35             $.each(data, function(indice, objeto) {
36                 fila = [];
37                 i++;
38                 datosTabla[i] = [];
39                 $.each(objeto, function(CABECERA,VALOR) {
40                     fila.push(VALOR);
41                 });
42                 datosTabla[i] = fila;
43             });
44
45             var date=google.visualization.arrayToDataTable(datosTabla);
46             console.log(date);
47
48             var options = {
49                 title : 'Consumo kW/Hora',
50                 vAxis : {
51                     title : "kW/Hora"
52                 },
53                 seriesType : "bars",
54                 series : {
55                     1: {
56                         type : "line"
57                     }
58                 },
59             };
60
61             var chart = new google.visualization.ColumnChart(document.getElementById('barchart_values'));
62             chart.draw(date, options);
63         }).fail(function(jqXHR, textStatus, errorThrown){
64             console.log("La solicitud a fallado: " + textStatus);
65         });
66     }
67     </script>
68 </head>
69 <body>
70     <!-- Navigation -->
71     <nav class="navbar navbar-inverse navbar-fixed-top" role="navigation">
72         <div class="container">
73             <!-- Brand and toggle get grouped for better mobile display -->
74             <div class="navbar-header">
75                 <button type="button" class="navbar-toggle" data-toggle="collapse" data-target="#bs-example-navbar-collapse-1">
76                     <span class="sr-only">Toggle navigation</span>
77                     <span class="icon-bar"></span>
78                     <span class="icon-bar"></span>
79                     <span class="icon-bar"></span>
80                 </button>
81                 <a class="navbar-brand" href="http://localhost/admin.html">Enchufes Inteligentes</a>
82             </div>
83             <!-- Collect the nav links, forms, and other content for toggling -->
84             <div class="collapse navbar-collapse" id="bs-example-navbar-collapse-1">
85                 <ul class="nav navbar-nav">
86                     <li>
87                         <a href="http://localhost/hora.html">Consumo kW/H</a>
```

```

88     </li>
89     <li>
90         <a href="http://localhost/dia.html">Consumo Diario</a>
91     </li>
92     <li>
93         <a href="http://localhost/mes.html">Consumo Mensual</a>
94     </li>
95   </ul>
96 </div>
97 <!-- /.navbar-collapse -->
98 </div>
99 <!-- /.container -->
100 </nav>
101 <!-- Page Content -->
102 <div class="container">
103   <div class="row">
104     <div class="col-lg-12 text-center">
105       <h1>Estás viendo tu consumo kW/hora</h1>
106     </div>
107   </div>
108   <div class="col-md-4 col-md-offset-0" id="barchart_values" style="width: 900px; height: 500px;" position = "absolute"></div>
109 </div>
110 <!-- /.container -->
111 <!-- jQuery Version 1.11.1 -->
112 <script src="js/jquery.js"></script>
113
114 <!-- Bootstrap Core JavaScript -->
115 <script src="js/bootstrap.min.js"></script>
116
117 </body>
118 </html>

```

Fig.6.6.4.6 Consumo kW/h.html



```

hora.php * 
1 <?php
2   $host_db = "127.0.0.1";
3   $user_db = "prueba";
4   $password_db = "1234";
5   $db_name = "arduino";
6   $tbl_name = "usuario";
7   $conexion = mysqli_connect($host_db, $user_db, $password_db,$db_name) or die("Cannot Connect to DataBase");
8
9 if(mysql_connect_errno()){
10   exit();
11 }
12 date_default_timezone_set('Europe/Madrid');
13 $hora = date('G'); //Eliminar -1 para dia actual
14 $array[0] = $hora;
15 $array[1] = 0;
16 try {
17   //Query = "SELECT SUM(POTENCIA) as SENSOR1, SUM(POTENCIA1) AS SENSOR2 FROM sensores WHERE DATE = CURRENT_DATE AND HOUR(TIME) =".$hora;
18   $query = "SELECT IFNULL(SUM(sensor1.INTENSIDAD),0) AS INTENSIDAD_SENSOR1, (SELECT IFNULL(SUM(sensor2.INTENSIDAD),0)
19   FROM sensor2 WHERE HOUR(sensor2.TIME) =".$hora.") AS INTENSIDAD_SENSOR2 FROM sensor1 WHERE HOUR(sensor1.TIME)=".$hora;
20   $resultado = mysqli_query($conexion,$query);
21
22   if(!mysqli_num_rows($resultado)){
23     $html='<h2>No existen datos</h2>';
24     mysqli_close($conexion);
25   }
26   else{
27     $data=array();
28     $Potencia = 0;
29     while ($row= mysqli_fetch_array($resultado, 1)) {
30       $potenciaSensor1 = $row['INTENSIDAD_SENSOR1'] * 230.0;
31       $potenciaSensor2 = $row['INTENSIDAD_SENSOR2'] * 230.0;
32       $Potencia += $potenciaSensor1/1000 + $potenciaSensor2/1000;
33     }
34     $array[1]= ($Potencia/3600000);
35     $data[] = array('HORA' => ($array[0]).":00 horas",'POTENCIA' =>($array[1]));
36     echo json_encode($data);
37   }
38 }
39 catch(Exception $e){
40   die($error);
41   $error = "error";
42   echo $error;
43 }
44 ?>

```

Fig.6.6.4.7 Consumo kW/h.php

Código página consumo día actual

```
dia.html
1  <!DOCTYPE html>
2  <html lang="es">
3  <head>
4      <meta charset="utf-8">
5      <meta http-equiv="X-UA-Compatible" content="IE=edge">
6      <meta name="viewport" content="width=device-width, initial-scale=1">
7      <meta name="description" content="">
8      <meta name="author" content="">
9      <title>Medidor energía enchufes</title>
10     <!-- Bootstrap Core CSS -->
11     <link href="pages/css/bootstrap.min.css" rel="stylesheet">
12     <script src="https://ajax.googleapis.com/ajax/libs/jquery/2.2.0/jquery.min.js"></script>
13     <script type="text/javascript" src="https://www.gstatic.com/charts/loader.js"></script>
14     <!-- Custom CSS -->
15     <style>
16         body {
17             padding-top: 60px;
18         }
19     </style>
20     <script type="text/javascript">
21         google.charts.load('current', {packages: ['corechart','bar']});
22         google.charts.setOnLoadCallback(drawBasic);
23
24         function drawBasic() {
25             var datosTabla = [];
26             var request = $.ajax({
27                 type : "POST",
28                 dataType :"json",
29                 url : "http://localhost/dia.php",
30             }).done(function(data, textStatus, jqXHR) {
31                 var i=0;
32
33                 datosTabla = [ [ 'DIA','KW' ] ];
34                 $each(data, function(indice, objeto) {
35                     fila = [];
36                     i++;
37                     datosTabla[i] = [];
38                     $each(objeto, function(CABECERA,VALOR) {
39                         fila.push(VALOR);
40                     });
41                     datosTabla[i] = fila;
42                 });
43
44                 var date=google.visualization.arrayToDataTable(datosTabla);
45                 console.log(date);
46
47                 var options = {
48                     title : 'Consumo Diario',
49                     VAXIS : {
50                         title : "KW/día"
51                     },
52                     seriesType : "bars",
53                     series : {
54                         1: {
55                             type : "line"
56                         }
57                     },
58                 };
59                 var chart = new google.visualization.ColumnChart(document.getElementById('barchart_values'));
60                 chart.draw(date, options);
61             }).fail(function(jqXHR, textStatus, errorThrown){
62                 console.log("La solicitud ha fallado: " + textStatus);
63             });
64         }
65     </script>
66 </head>
67 <body>
68     <!-- Navigation -->
69     <nav class="navbar navbar-inverse navbar-fixed-top" role="navigation">
70         <div class="container">
71             <!-- Brand and toggle get grouped for better mobile display -->
72             <div class="navbar-header">
73                 <button type="button" class="navbar-toggle" data-toggle="collapse" data-target="#bs-example-navbar-collapse-1">
74                     <span class="sr-only">Toggle navigation</span>
75                     <span class="icon-bar"></span>
76                     <span class="icon-bar"></span>
77                     <span class="icon-bar"></span>
78                 </button>
79                 <a class="navbar-brand" href="http://localhost/admin.html">Enchufes Inteligentes</a>
80             </div>

```

```

81     <!-- Collect the nav links, forms, and other content for toggling --&gt;
82     &lt;div class="collapse navbar-collapse" id="bs-example-navbar-collapse-1"&gt;
83         &lt;ul class="nav navbar-nav"&gt;
84             &lt;li&gt;|   &lt;a href="http://localhost/hora.html"&gt;Consumo kW/H&lt;/a&gt;
85             &lt;/li&gt;|
86             &lt;li&gt;|   &lt;a href="http://localhost/dia.html"&gt;Consumo Diario&lt;/a&gt;
87             &lt;/li&gt;|
88             &lt;li&gt;|   &lt;a href="http://localhost/mes.html"&gt;Consumo Mensual&lt;/a&gt;
89             &lt;/li&gt;
90         &lt;/ul&gt;
91     &lt;/div&gt;
92     &lt;!-- /.navbar-collapse --&gt;
93 &lt;/div&gt;
94 &lt;!-- /.container --&gt;
95 &lt;/nav&gt;
96 &lt;!-- Page Content --&gt;
97 &lt;div class="container"&gt;
98     &lt;div class="row"&gt;
99         &lt;div class="col-lg-12 text-center"&gt;
100            &lt;h1&gt;Estás viendo tu consumo dia actual&lt;/h1&gt;
101        &lt;/div&gt;
102    &lt;div class="col-md-4 col-md-offset-0" id="barchart_values" style="width: 900px; height: 500px;"&gt;&lt;/div&gt;
103 &lt;/div&gt;
104 &lt;!-- /.container --&gt;
105 &lt;!-- jQuery Version 1.11.1 --&gt;
106 &lt;script src="js/jquery.js"&gt;&lt;/script&gt;
107 &lt;!-- Bootstrap Core Javascript --&gt;
108 &lt;script src="js/bootstrap.min.js"&gt;&lt;/script&gt;
109 &lt;/body&gt;
110 &lt;/html&gt;
</pre>

```

Fig.6.6.4.8 Consumo día actual.html

```

1 <?php
2 $host_db = "127.0.0.1";
3 $user_db = "prueba";
4 $password_db ="1234";
5 $db_name = "arduino";
6 $tbl_name = "usuario";
7 $conexion = mysqli_connect($host_db, $user_db, $password_db,$db_name) or die("Cannot Connect to DataBase");
8
9 if(mysqli_connect_errno()){
10    exit();
11 }
12 date_default_timezone_set('Europe/Madrid');
13 $dia = date('j'); //Eliminar -1 para dia actual
14 $array[0] = $dia;
15 $array[1] = 0;
16
17 try {
18     //$query = "SELECT SUM(POTENCIA) as SENSOR1, SUM(POTENCIA) AS SENSOR2 FROM sensores WHERE DAY(DATE)=$dia";
19     $query = "SELECT IFNULL(SUM(sensor1.INTENSIDAD),0) AS INTENSIDAD_SENSOR1, (SELECT IFNULL(SUM(sensor2.INTENSIDAD),0)
20         FROM sensor2 WHERE DATE(sensor2.DATE) = CURRENT_DATE)
21         AS INTENSIDAD_SENSOR2 FROM sensor1 WHERE DATE(sensor1.DATE) = CURRENT_DATE";
22     $resultado = mysqli_query($conexion,$query);
23
24     if(!mysqli_num_rows($resultado)){
25         printf ("Tabla Vacía");
26         mysqli_close($conexion);
27     }
28     else{
29         $data=array();
30         $Potencia = 0;
31         while ($row= mysqli_fetch_array($resultado, 1)) {
32             $Potencia+= (($row['INTENSIDAD_SENSOR1']/1000) *230.0) + (($row['INTENSIDAD_SENSOR2']/1000) * 230.0);
33         }
34         $array[1]= $Potencia/3600000; //kW
35         $array[0]=($array[0]);
36         $data[] = array('DIA' => "Dia:".$array[0], 'POTENCIA' =>($array[1]));
37         echo json_encode($data);
38     }
39 }
40 catch(Exception $e){
41     die($error);
42     $error = "error";
43     echo $error;
44 }
45 ?>

```

Fig.6.6.4.9 Consumo día actual.php

Código página consumo mensual

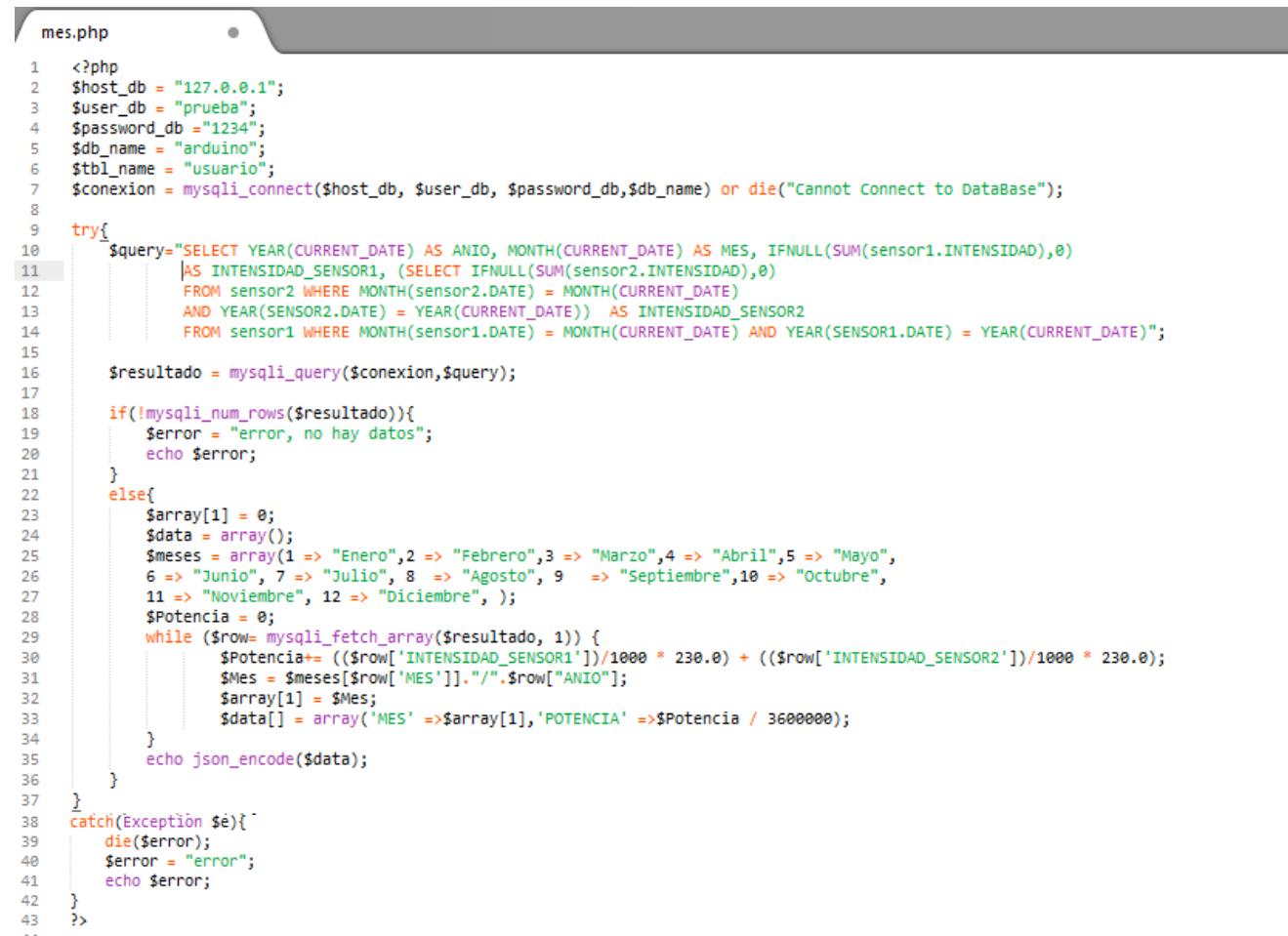
```
mes.html      x
1  <!DOCTYPE html>
2  <html lang="es">
3  <head>
4      <meta charset="utf-8">
5      <meta http-equiv="X-UA-Compatible" content="IE=edge">
6      <meta name="viewport" content="width=device-width, initial-scale=1">
7      <meta name="description" content="">
8      <meta name="author" content="">
9      <title>Medidor energía enchufes</title>
10     <!-- Bootstrap Core CSS -->
11     <link href="pages/css/bootstrap.min.css" rel="stylesheet">
12     <script src="https://ajax.googleapis.com/ajax/libs/jquery/2.2.0/jquery.min.js"></script>
13     <script type="text/javascript" src="https://www.gstatic.com/charts/loader.js"></script>
14     <!-- Custom CSS -->
15     <style>
16     body {
17         padding-top: 60px;
18     }
19     </style>
20
21     <script type="text/javascript">
22     google.charts.load('current', {packages: ['corechart','bar']});
23     google.charts.setOnLoadCallback(drawBasic);
24
25     function drawBasic() {
26         var datosTabla = [];
27         var request = $.ajax({
28             type :"POST",
29             dataType :"json",
30             url : "http://localhost/mes.php",
31         }).done(function(data, textStatus, jqXHR) {
32             var i=0;
33
34             datosTabla = [ [ 'MES','kW/mes' ] ];
35             $.each(data, function(indice, objeto) {
36                 fila = [];
37                 i++;
38                 datosTabla[i] = [];
39                 $.each(objeto, function(CABECERA,VALOR) {
40                     fila.push(VALOR);
41                 });
42                 datosTabla[i] = fila;
43             });
44             var date=google.visualization.arrayToDataTable(datosTabla);
45             console.log(date);
46
47             var options = {
48                 title : 'Consumo Mensual',
49                 vAxis : {
50                     title : "kW/mes"
51                 },
52                 hAxis : {
53                     title : "MES"
54                 },
55                 seriesType : "bars",
56                 series : {
57                     1: {
58                         type : "line"
59                     }
60                 },
61             };
62
63             var chart = new google.visualization.ColumnChart(document.getElementById('barchart_values'));
64             chart.draw(date, options);
65         }).fail(function(jqXHR, textStatus, errorThrown){
66             console.log("La solicitud a fallado: " + textStatus);
67         });
68     }
69     </script>
70     </script>
71 </head>
72 <body>
73     <!-- Navigation -->
74     <nav class="navbar navbar-inverse navbar-fixed-top" role="navigation">
75         <div class="container">
76             <!-- Brand and toggle get grouped for better mobile display -->
77             <div class="navbar-header">
78                 <button type="button" class="navbar-toggle" data-toggle="collapse" data-target="#bs-example-navbar-collapse-1">
79                     <span class="sr-only">Toggle navigation</span>
80                     <span class="icon-bar"></span>
```

```

81         <span class="icon-bar"></span>
82         <span class="icon-bar"></span>
83     </button>
84     <a class="navbar-brand" href="http://localhost/admin.html">Enchufes Inteligentes</a>
85 </div>
86     <!-- Collect the nav links, forms, and other content for toggling -->
87     <div class="collapse navbar-collapse" id="bs-example-navbar-collapse-1">
88         <ul class="nav navbar-nav">
89             <li>
90                 <a href="http://localhost/hora.html">Consumo kW/H</a>
91             </li>
92             <li>
93                 <a href="http://localhost/dia.html">Consumo Diario</a>
94             </li>
95             <li>
96                 <a href="http://localhost/mes.html">Consumo Mensual</a>
97             </li>
98         </ul>
99     </div>
100    <!-- /.navbar-collapse -->
101   <!-- /.container -->
102 </nav>
103     <!-- Page Content -->
104     <div class="container">
105         <div class="row">
106             <div class="col-lg-12 text-center">
107                 <h1>Estás viendo tu consumo mensual</h1>
108             </div>
109             <div class="col-md-4 col-md-offset-0" id="barchart_values" style="width: 900px; height: 500px;" position = "absolute"></div>
110         </div>
111     <!-- /.container -->
112     <!-- jQuery Version 1.11.1 -->
113     <script src="js/jquery.js"></script>
114     <!-- Bootstrap Core Javascript -->
115     <script src="js/bootstrap.min.js"></script>
116
117 </body>
118 </html>

```

Fig.6.6.4.10 Código Consumo mensual.html



```

mes.php

1  <?php
2  $host_db = "127.0.0.1";
3  $user_db = "prueba";
4  $password_db ="1234";
5  $db_name = "arduino";
6  $tbl_name = "usuario";
7  $conexion = mysqli_connect($host_db, $user_db, $password_db,$db_name) or die("Cannot Connect to DataBase");
8
9  try{
10      $query="SELECT YEAR(CURRENT_DATE) AS ANIO, MONTH(CURRENT_DATE) AS MES, IFNULL(SUM(sensor1.INTENSIDAD),0)
11          AS INTENSIDAD_SENSOR1, (SELECT IFNULL(SUM(sensor2.INTENSIDAD),0)
12          FROM sensor2 WHERE MONTH(sensor2.DATE) = MONTH(CURRENT_DATE)
13          AND YEAR(SENSOR2.DATE) = YEAR(CURRENT_DATE)) AS INTENSIDAD_SENSOR2
14          FROM sensor1 WHERE MONTH(sensor1.DATE) = MONTH(CURRENT_DATE) AND YEAR(SENSOR1.DATE) = YEAR(CURRENT_DATE)";
15
16      $resultado = mysqli_query($conexion,$query);
17
18      if(!mysqli_num_rows($resultado)){
19          $error = "error, no hay datos";
20          echo $error;
21      }
22      else{
23          $array[1] = 0;
24          $data = array();
25          $meses = array(1 => "Enero",2 => "Febrero",3 => "Marzo",4 => "Abril",5 => "Mayo",
26          6 => "Junio", 7 => "Julio", 8 => "Agosto", 9 => "Septiembre",10 => "Octubre",
27          11 => "Noviembre", 12 => "Diciembre", );
28          $Potencia = 0;
29          while ($row= mysqli_fetch_array($resultado, 1)) {
30              $Potencia+= (($row['INTENSIDAD_SENSOR1'])/1000 * 230.0) + ((($row['INTENSIDAD_SENSOR2'])/1000 * 230.0));
31              $Mes = $meses[$row['MES']]."/".$row["ANIO"];
32              $array[1] = $Mes;
33              $data[] = array('MES' =>$array[1],'POTENCIA' =>$Potencia / 3600000);
34          }
35          echo json_encode($data);
36      }
37  }
38  catch(Exception $e){
39      die($error);
40      $error = "error";
41      echo $error;
42  }
43 ?>

```

Fig.6.6.4.11 Código consumo mensual.php

7. INSTALACIÓN

En este capítulo vamos a tratar tanto la preparación de los módulos en sus cajas como su colocación.

No es necesario la adquisición de un monitor de energía para ver su consumo, pues como hemos comentado en capítulos anteriores, nuestro producto dispone de una pantalla LCD.

7.1 Mecanización de la caja instalación de los componentes

Para la protección de los elementos, hemos utilizado una caja de plástico duro. Para ello ha sido necesario crear los huecos necesarios para las conexiones y taladrar para colocar los componentes.

Las siguientes imágenes muestran el desarrollo del producto.



Fig.7 Vista Frontal

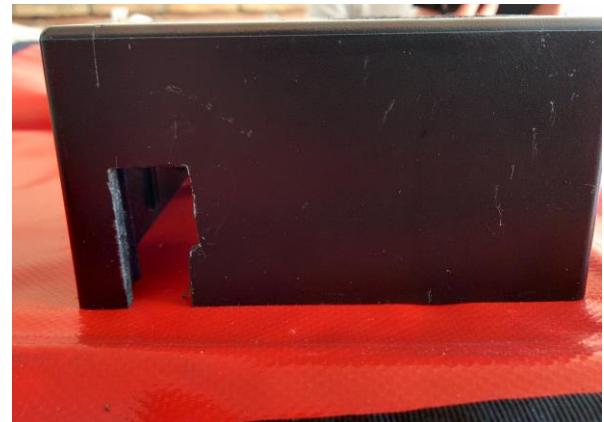


Fig.7.1 Vista Lateral

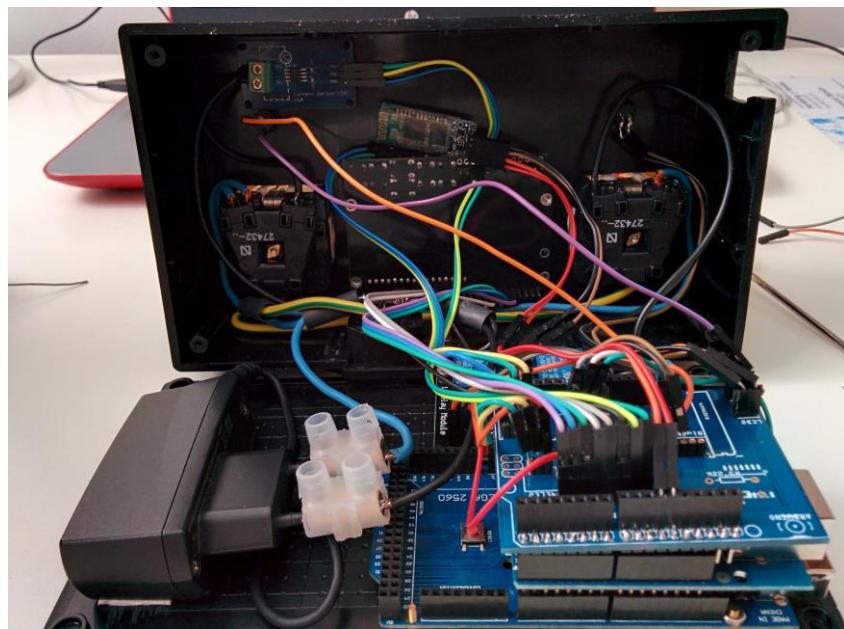


Fig.7.1.2 Vista caja abierta

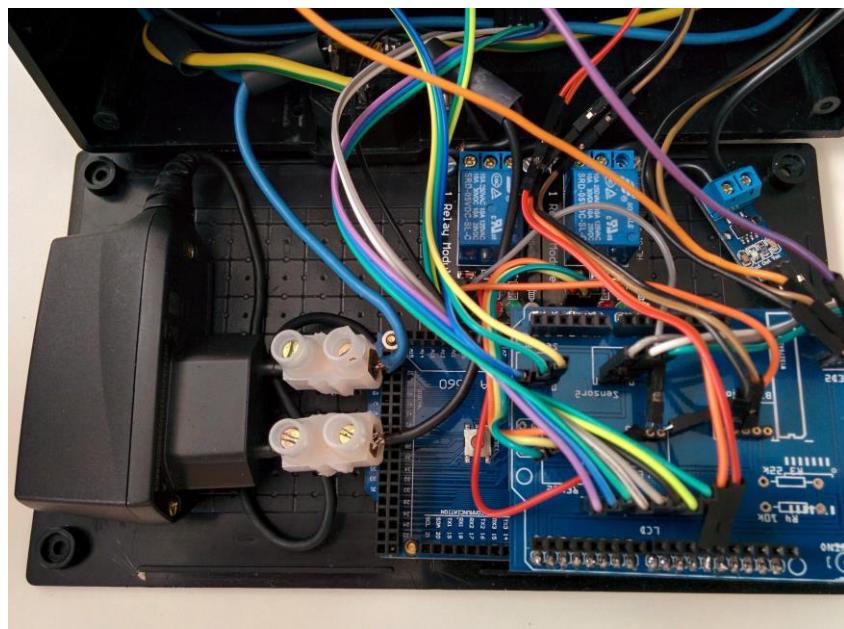


Fig.7.1.3 Vista frontal caja abierta

7.2 Resultado final

En las siguientes imágenes, se muestra el resultado final de nuestro proyecto.



Fig.7.2 Resultado final
vista frontal



Fig.7.2.1 Resultado
final vista perfil
superior

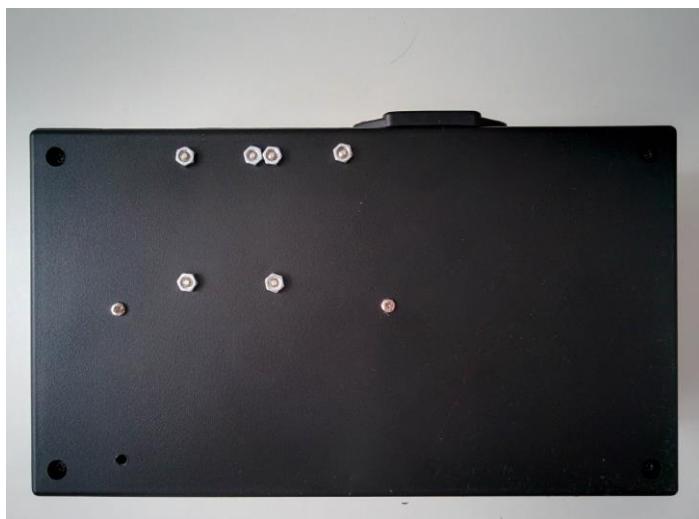


Fig.7.2.2 Resultado final
vista trasera

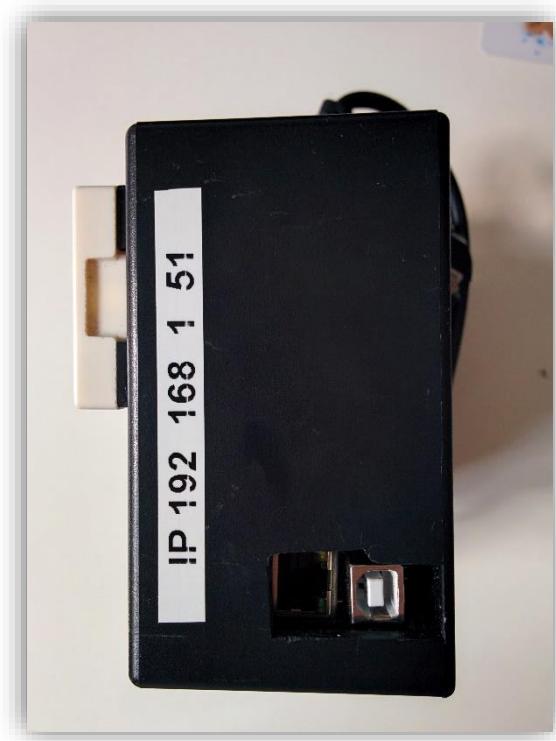


Fig.7.2.3 Resultado
final lateral

8. PRUEBAS

Para realizar las pruebas o ensayos se han utilizado los siguientes instrumentos de medidas.

Polímetro marca Stanton modelo UT57 (Fig.8.1) para las mediciones de resistencias, y voltios alterna.

Se realizaron distintas pruebas y medidas sobre el sensor para estudiar su comportamiento.



Fig.8.1 Polímetro Stanton UT57

8.1 Pruebas Iniciales

Se realizaron distintas pruebas y medidas sobre los sensores, antes de proceder a su embalaje final.

Prueba con sensor modelo ACS712ELCTR-05BT (5 Amperios) :

Se realizaron pruebas con una bombilla de intensidad de 18mA. El resultado se muestra en la siguiente figura

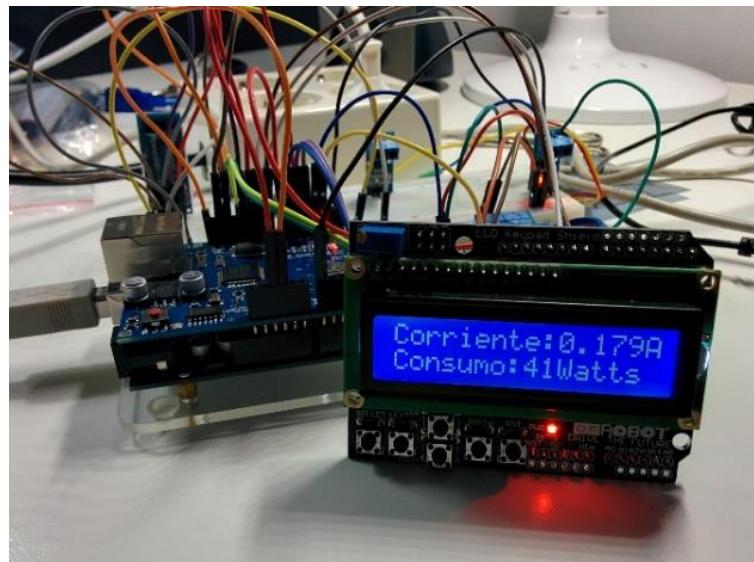


Fig.8.1.2 Prueba Bombilla 18mA

Por otro lado, al desconectar el dispositivo de la corriente, pudimos ver que el sensor comete un error en la medida.

En un principio pensamos que era el consumo interno del sensor, pero al usar el polímetro percatamos que realmente es un error.

En la siguiente figura se muestra el error en la medición cuando la bombilla está apagada.

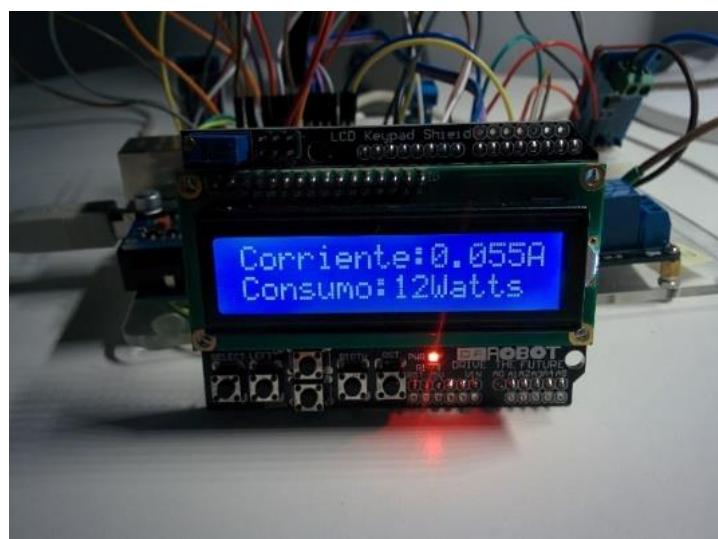


Fig.8.1.3 Prueba Bombilla 18mA
apagada

8.2. Pruebas Finales.

Todas las medidas realizadas en las pruebas de consumo han sido contrastadas con un medidor de consumo individual, que nos proporciona la Intensidad, en amperios, y/o el consumo, en vatios, así como el factor de potencia.

Se utilizó el medidor: **dale 601e safety analyzer**

Prueba de consumo realizada a una bombilla alógena de 50W.

En la siguiente figura se muestra el resultado obtenido y podemos ver que se comete un error de un $\pm 1\%$ aprox. Dicho error ha mejorado respecto la medición mostrada en el apartado anterior, ya que las conexiones de los componentes se encuentran soldadas.



Fig.8.2 Prueba Bombilla 50W

Prueba de consumo realizada a un foco con un consumo de 120W.

En las siguientes imágenes se muestran las medidas obtenidas y podemos ver que se comete un error de un $\pm 2\%$ aprox.

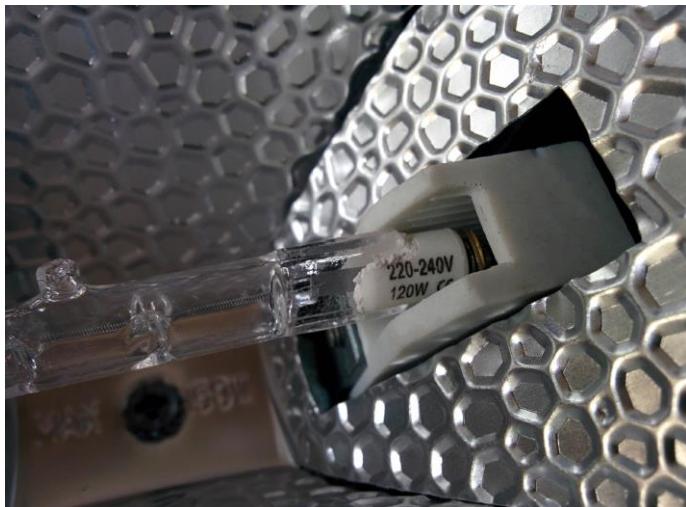


Fig.8.2.1 Foco 120W



Fig.8.2.2 Prueba foco 120W

8.3 Comparativa de medidas, gráfica de respuesta

8.3.1 Cuantificación de errores

Error absoluto

Obtenemos el error absoluto calculando la diferencia entre el valor indicado por nuestro patrón (X_v) y el obtenido por nuestro sistema (X_i)

$$\mathcal{E}_a = |x_i - x_v|$$

Error relativo

Se calcula dividiendo el error absoluto (Δx) por el valor real (X_v)

$$\mathcal{E}_r = \left| \frac{x_i - x_v}{x_v} \right| = \frac{|\Delta x|}{x_v}$$

Error porcentual

Es el error relativo multiplicado por cien, expresado en %

$$\mathcal{E}_r = \left| \frac{x_i - x_v}{x_v} \right| 100\%$$

El error relativo representa la fracción de imprecisión cometida en la medición y resulta útil para comparar la calidad de las mediciones.

En la siguiente tabla (Fig.) se muestra los cálculos del error absoluto, error relativo y porcentual de las medidas

realizadas, usando el sensor **ACS712ELCTR-05B-T** de la primera prueba realizada en el apartado anterior.

PATRON	SISTEMA	ERROR ABSOLUTO	ERROR RELATIVO	ERROR PORCENTUAL
0,21	0,214	0,004	0,018691589	1,869158879
0,21	0,215	0,005	0,023255814	2,325581395
0,21	0,21	0	0	0
0,21	0,217	0,007	0,032258065	3,225806452
0,21	0,211	0,001	0,004739336	0,473933649
0,21	0,218	0,008	0,036697248	3,669724771
0,21	0,215	0,005	0,023255814	2,325581395
0,21	0,216	0,006	0,027777778	2,777777778
0,21	0,215	0,005	0,023255814	2,325581395
0,21	0,21	0	0	0
0,21	0,215	0,005	0,023255814	2,325581395
0,21	0,216	0,006	0,027777778	2,777777778
0,21	0,212	0,002	0,009433962	0,943396226
0,21	0,217	0,007	0,032258065	3,225806452
0,21	0,21	0	0	0
		ERROR ABSOLUTO MEDIO	ERROR RELATIVO MEDIO	ERROR PORCENTUAL MEDIO
		0,003615385	0,016780073	1,678007282

La (Fig.8.3) muestra la gráfica del error relativo, obtenida con los valores obtenidos en la tabla anterior. Se aprecia una tendencia de error de un 1.67%

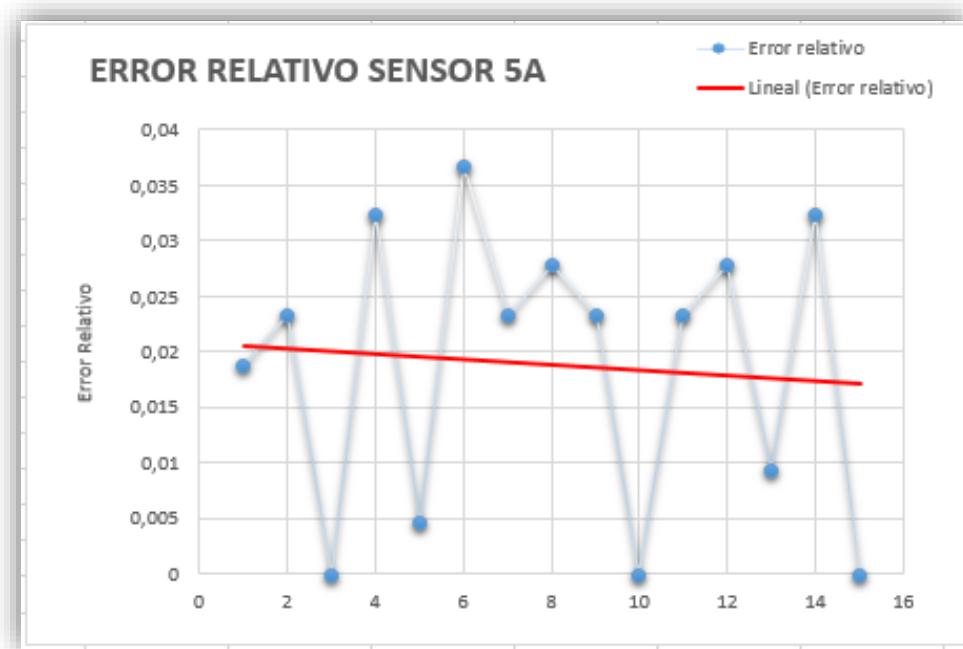


Fig.8.3 Gráfica error relativo
Sensor 5A

Para la segunda prueba, obtenemos los siguientes valores que se muestran en la tabla. Dichas mediciones, se han realizado a través del sensor **ACS712ELCTR-20A-T**

PATRON	SISTEMA	ERROR ABSOLUTO	ERROR RELATIVO	ERROR PORCENTUAL
0,52	0,56	0,04	0,071428571	7,142857143
0,51	0,55	0,04	0,072727273	7,272727273
0,51	0,56	0,05	0,089285714	8,928571429
0,52	0,562	0,042	0,074733096	7,473309609
0,52	0,557	0,037	0,066427289	6,642728905
0,51	0,556	0,046	0,082733813	8,273381295
0,52	0,551	0,031	0,056261343	5,626134301
0,52	0,555	0,035	0,063063063	6,306306306
0,52	0,55	0,03	0,054545455	5,454545455
0,52	0,55	0,03	0,054545455	5,454545455
0,52	0,557	0,037	0,066427289	6,642728905
0,52	0,557	0,037	0,066427289	6,642728905
0,52	0,554	0,034	0,061371841	6,137184116
0,52	0,55	0,03	0,054545455	5,454545455
0,52	0,549	0,029	0,052823315	5,282331512
ERROR ABSOLUTO MEDIO		0,036533333	0,065823084	6,582308404

La (Fig.8.3.1) muestra la gráfica del error relativo, obtenida con los valores obtenidos en la tabla anterior. Se aprecia una tendencia de error de un 6.58%

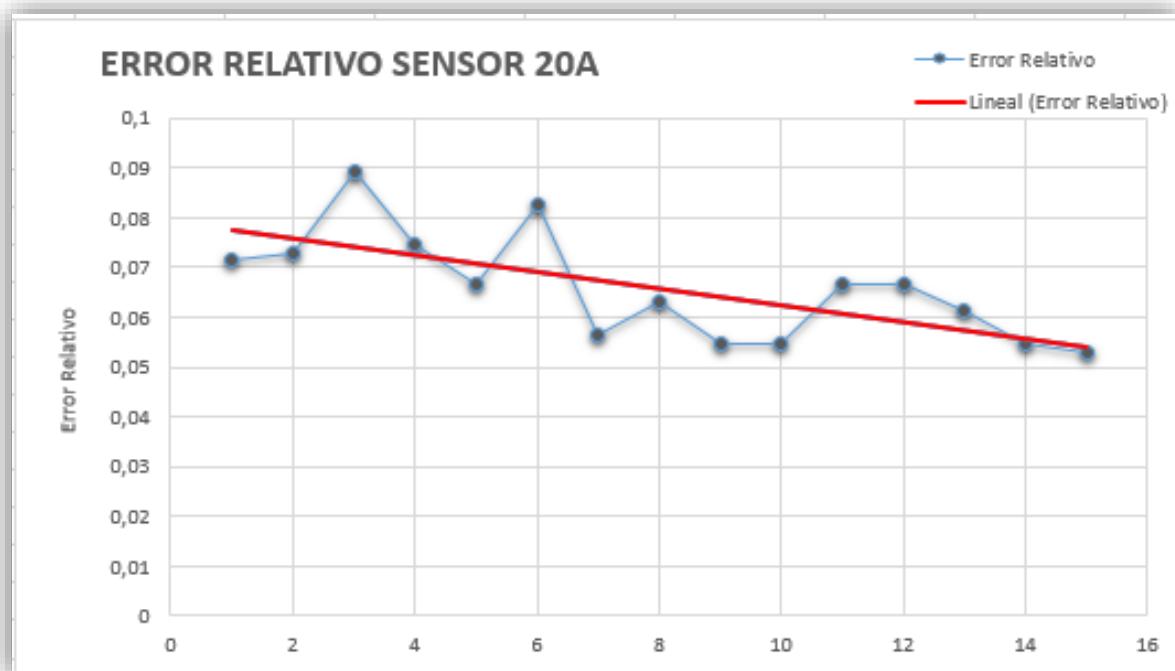


Fig.8.3.1 Gráfica error relativo Sensor 20A

8.4 Prueba de comportamiento de la aplicación

Se realizó la comprobación del funcionamiento de la aplicación, conectándonos a la estación base (servidor web ejecutando el sistema operativo UBUNTU 14.04 LTS, conectado a la red local del domicilio), a través de los distintos navegadores, se comprobó que funcionaban perfectamente con Mozilla Firefox, Google Chrome, Internet Explorer y Safari.

Comprobamos el acceso a la aplicación mediante la IP que viene impresa en la caja del sistema, nos solicita usuario y contraseña y tuvimos acceso. El resultado fue satisfactorio.

9. COSTE DEL PROYECTO

La siguiente tabla recoge el coste material del proyecto.
(Fig.9)

COSTE FABRICACIÓN DEL MEDIDOR DE ENERGÍA			
PRODUCTO	CANTIDAD	PRECIO UNIDAD	TOTAL
CONEXIONES - CABLES	1	3,52	3,52
MODULO BLUETOOTH HC-06	1	5,7	5,7
SENSOR ACS712	2	5,57	11,14
MODULO RELES	2	2,39	4,78
PANTALLA LCD	1	5,43	5,43
ARDUINO UNO	1	9,05	9,05
ARDUINO MEGA	1	24,19	24,19
ETHERNET SHIELD	1	7,36	7,36
PLATAFORMA	1	12,31	12,31
ENCHUFES	2	4,65	9,3
CABLES			0
CAJA PLASTICO	1	3,25	3,25
SOPORTE LEDS	2	1,25	2,5
PCB	1	26	26
TOTAL (en Euros, €)			124,53

Fig.9 Tabla coste del proyecto

9.1 Coste de mano de obra.

La realización de este proyecto, ha durado aproximadamente 5 meses, con una media de 30 días al mes, lo que hacen 160 días. Por tanto, asignando un salario de 20€/hora, el coste total estaría en (fig 9.1) :

Coste personal				
Trabajador	Días	Horas	Precio €/h	Coste Total €
Fernando Méndez Requena	160	8	20,00 €	25.600,00 €

Fig.9.1 Coste Personal

Nota: la planificación temporal se encuentra detallada en el siguiente capítulo.

9.2 Coste de equipamiento.

El coste destinado a los equipos informáticos es el siguiente (fig.9.2) :

Coste equipamiento informático		
Equipo	Antigüedad	Precio €
Portatil Hp Pavilion 15-ab105ns	6 meses	645

Fig.9.2 Coste de equipamiento

10. TEMPORALIZACIÓN DEL PROYECTO

En este capítulo describiremos las fases en las que hemos dividido el proyecto, así como un análisis del tiempo estimado y tiempo real empleado en el desarrollo del proyecto

10.1 Fases del proyecto

La implementación del proyecto la hemos dividido en 6 fases (Fig.10.1). A continuación, describiremos brevemente cada una de las partes que la componen:

10.1.1 Búsqueda de información

En esta primera fase, la hemos dedicado a la búsqueda de información de artículos de información sobre proyectos similares al nuestro. También hemos dedicado parte de este tiempo a la búsqueda y adquisición de los componentes necesarios para la fabricación de nuestro sistema.

10.1.2 Estudio de mercado

En esta parte del proyecto nos hemos dedicado al estudio de los distintos sistemas comunes al nuestro que existen en el mercado, y a las diferentes prestaciones que presenta cada uno de ellos a los usuarios.

10.1.3 Desarrollo del proyecto

En esta parte del proyecto se ha realizado el diseño e implementación de nuestros enchufes inteligentes, el cual se ha dividido en 5 partes:

- Captación de datos
 - o Desarrollo del firmware completo necesario para el funcionamiento correcto de nuestro sistema.
- Diseño e Implementación aplicación Android
 - o Creación de aplicación móvil la cual sirve como panel de control para interactuar con nuestro sistema
- Diseño PCB
 - o Creación y diseño de una placa PCB

- Aplicación web y Servidor
 - o Diseño e implementación de cada una de las páginas que forman nuestro sitio web, junto con la instalación y configuración del servidor web.
- Configuración del sistema
 - o Instalación de los componentes que forman parte de nuestra regleta inteligente.

10.1.4 Pruebas y Errores

En este apartado se realizan las pruebas y medidas necesarias que necesitamos para la implementación del proyecto, también se realizaran test para comprobar los porcentajes de error.

10.1.5 Instalación componentes

La instalación y montaje de todos los componentes que forman el proyecto, se llevó a cabo en mi propio domicilio.

10.1.6 Documentación

Contendrá toda la documentación necesaria para la implementación del proyecto, así como las conclusiones del mismo.

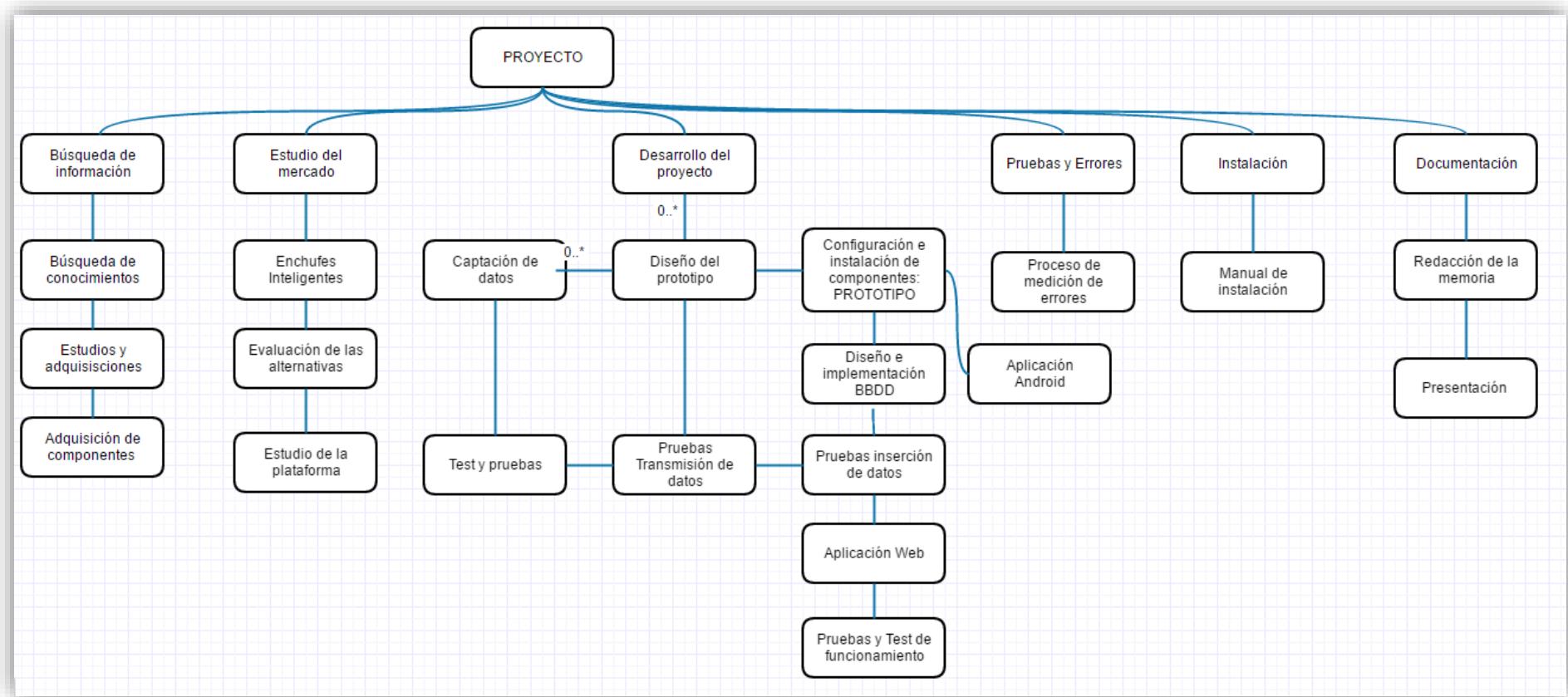


Fig.10.1.1 Diagrama de fases del Proyecto

El siguiente esquema (Fig.10.1.2) representa un diagrama del proceso de trabajo para la fabricación, prueba e instalación de la regleta inteligente.

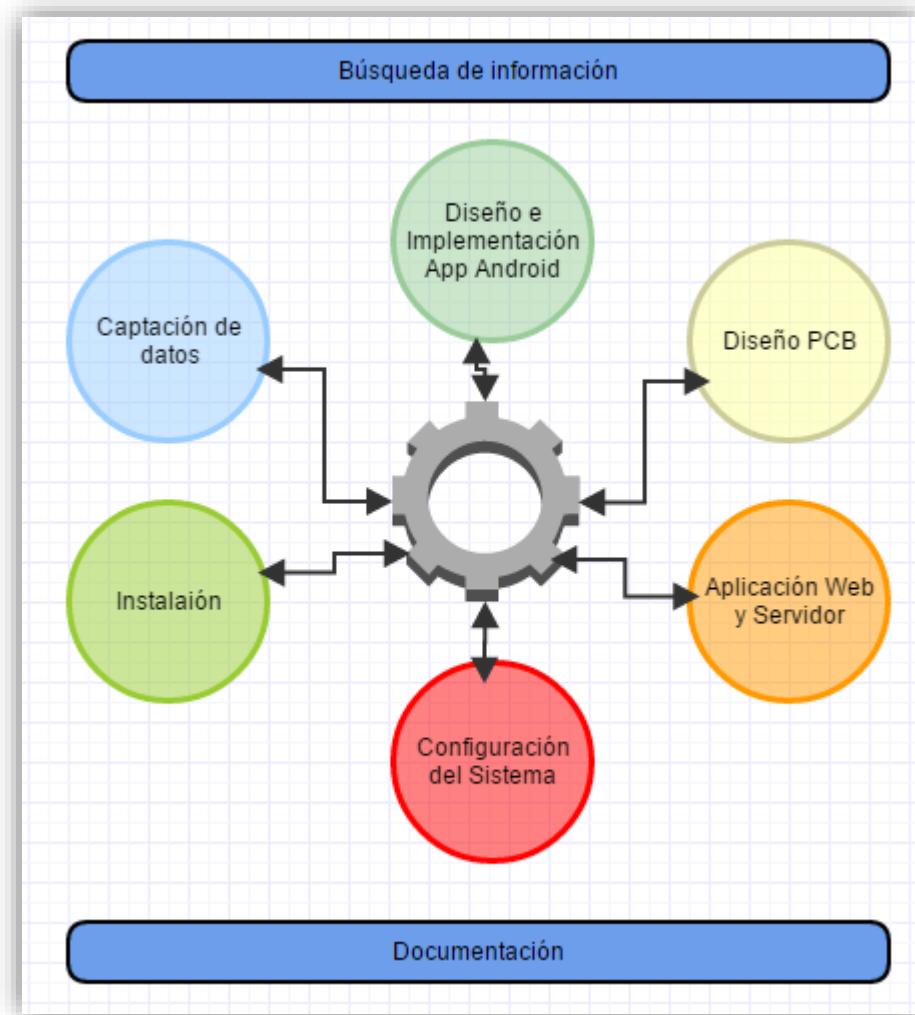
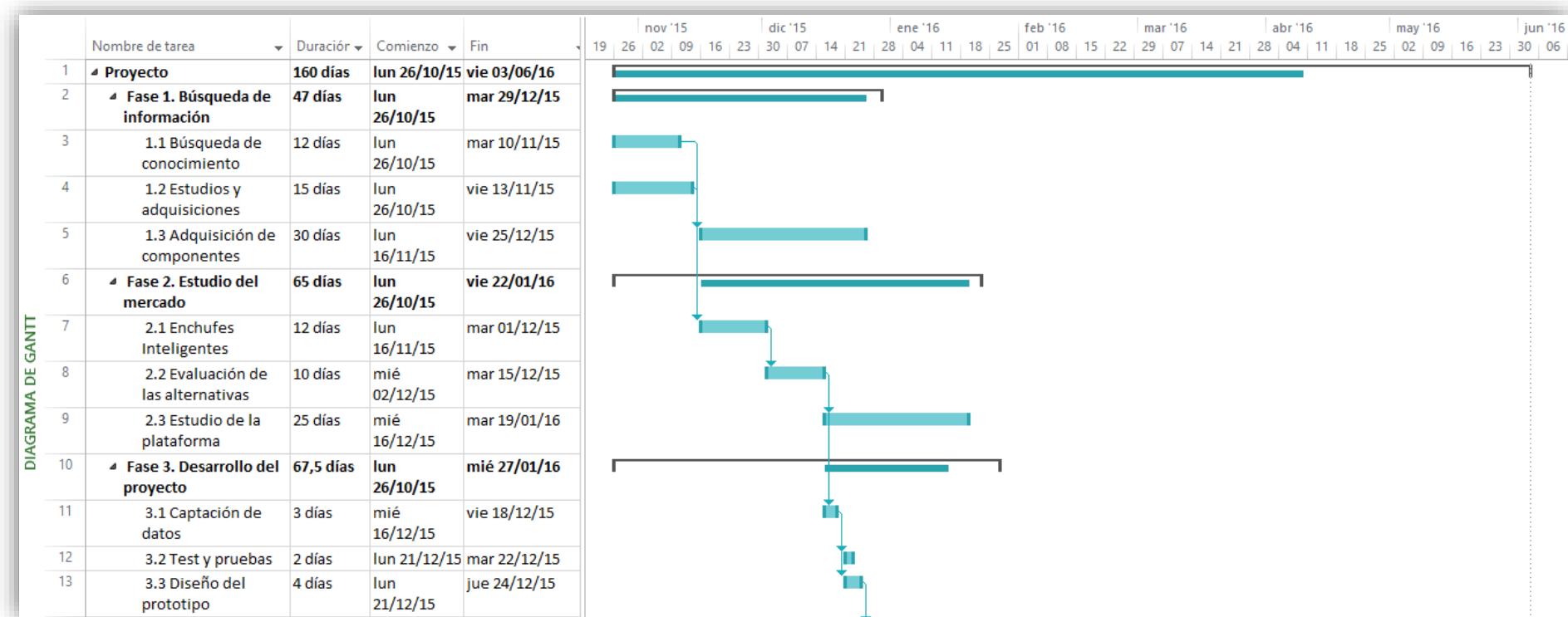


Fig.10.1.2 Proceso de trabajo

10.2 Análisis Temporal

Las siguientes figuras muestran una estimación aproximada del tiempo necesario para la realización de cada una de las fases del proyecto, así como la representación de un diagrama de Gant (Fig.10.2.1).



14	3.4 Pruebas transmisión de datos	2 días	vie 25/12/15	lun 28/12/15
15	3.5 Configuración e instalación de componentes	10 días	mar 29/12/15	lun 11/01/16
16	3.6 Diseño e implementación de BBDD	1 día	vie 25/12/15	vie 25/12/15
17	3.7 Pruebas de inserción de datos	2 días	mar 29/12/15	mié 30/12/15
18	3.8 Aplicación Web	3 días	lun 28/12/15	mié 30/12/15
19	3.9 Aplicación Android	3 días	mar 12/01/16	jue 14/01/16
20	4.0 Pruebas y test de funcionamiento	2 días	jue 31/12/15	vie 01/01/16
21	▫ Fase 4. Pruebas y Errores	30 días	lun 04/01/16	vie 12/02/16
22	Fase 4.1 Proceso de medición de errores	30 días	lun 04/01/16	vie 12/02/16
23	▫ Fase 5. Instalación	11 días	vie 12/02/16	vie 26/02/16
24	5.1 Manual de instalación	7 días	lun 15/02/16	mar 23/02/16
25	▫ Fase 6. Documentación	121 días	lun 26/10/15	sáb 09/04/16
26	Redacción de la memoria	30 días	mié 24/02/16	mar 05/04/16
27	Presentación PPT	2 días	mié 06/04/16	jue 07/04/16

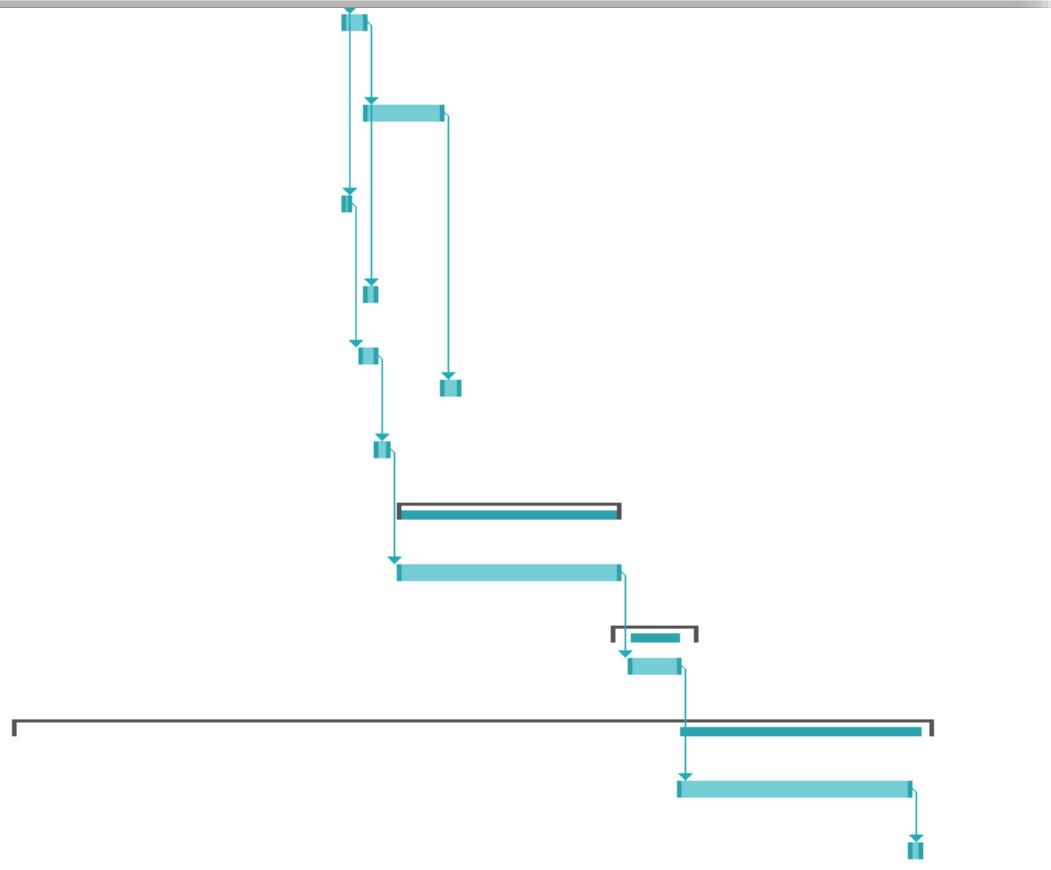


Fig.10.2.1 Diagrama de Gant

11. MANUAL DE USO

Usar nuestro sistema es muy fácil. La IP de nuestro servidor web se aloja en la parte lateral de nuestro sistema, en la zona del conector RJ45.

Nos conectaremos con cualquier navegador introduciendo en la barra de direcciones la dirección IP

La siguiente figura muestra en un Ipad(Fig.11.1), la página de acceso que se visualiza cuando accedes a través del navegador.

Notar que hacemos uso de la red interna de nuestro domicilio. Una mejora en una futura versión es disponer de alojamiento web en la nube, así como incorporar tecnología WIFI a nuestro sistema empotrado.

Los datos de acceso son los siguientes:

USUARIO:	CONTRASEÑA:
admin	admin

Una vez accedido correctamente, visualizaremos la página principal, que como ya comentamos en apartados anteriores, se titula Admin.html, la cual nos muestra el consumo instantáneo (Fig.11.2).

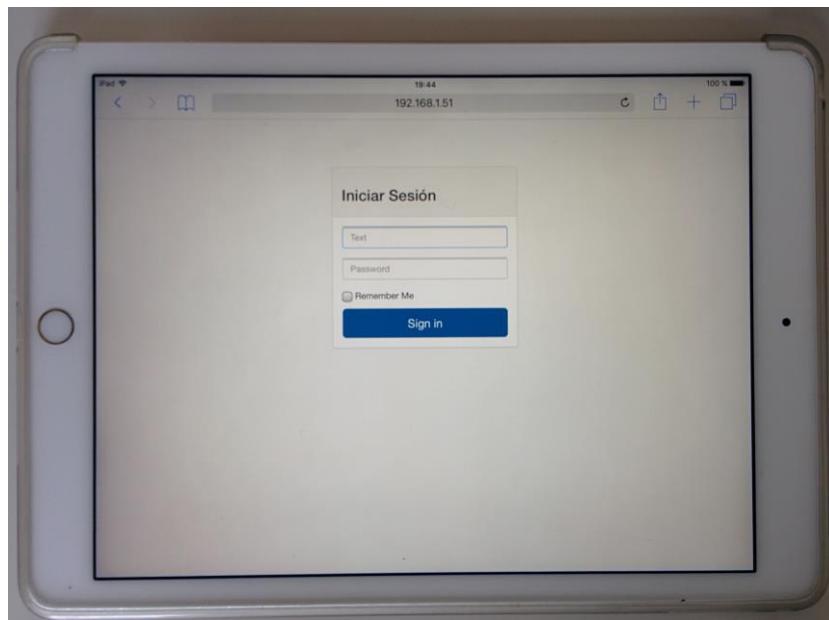


Fig.11.1 Página de acceso IPAD

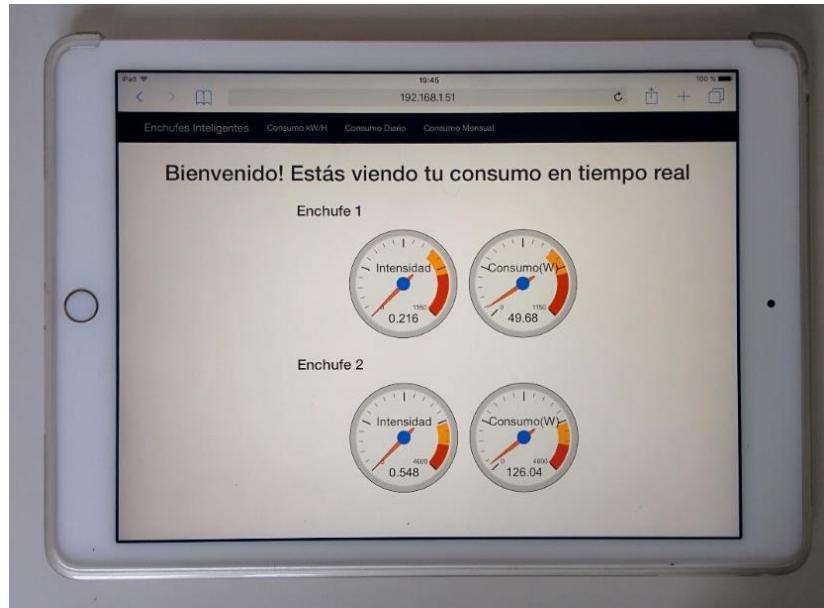


Fig.11.2 Página de inicio IPAD

En la parte superior de la página, se encuentra la cabecera, en ella podemos ver el contenido de las distintas páginas

Enchufes Inteligentes Consumo kW/H Consumo Diario Consumo Mensual

Si clicamos sobre "Enchufes Inteligentes", nos redirecciona a la página principal (Fig.11.2)

Por otro lado, el funcionamiento de nuestro medidor de energía también es muy sencillo. Como ya sabemos, disponemos de una aplicación Android para interactuar con el sistema.

Una vez que conectamos nuestro medidor de energía a la red eléctrica de nuestro domicilio, el siguiente paso es activar la función bluetooth de nuestro dispositivo móvil y con ello emparejarnos al receptor bluetooth de nuestro sistema. Una vez conectado, accedemos automáticamente al panel de control.

**Nota: Una futura mejora para el proyecto, es incorporar una función que permita al usuario modificar la contraseña. Así mismo, otro aspecto a mejorar, es implementar las conexiones inalámbricas a través de módulos de RT*

12. IMÁGEN PROMOCIONAL DEL PRODUCTO



Fig.12 Imagen promocional del producto

13. PROBLEMAS EN EL DESARROLLO PROYECTO

En este apartado, trataremos los problemas que tuvimos a la hora de desarrollar nuestro proyecto.

Como hemos comentado en la introducción, el reto fue grande, ya que no tenía suficientes conocimientos para su realización.

Los problemas que tuvimos fueron los siguientes.

a) Problemas orientados a la visualización de los datos procesados vía online

En un principio, optamos por utilizar el sitio web Plotly, [19] una plataforma web que nos permite analizar y visualizar los datos procesados por Arduino, en tiempo real.

Para ello, tuvimos que hacer uso de una librería externa, que nos proporciona dicha web. [20]

La siguiente imagen (Fig.12.1) muestra un ejemplo usando un potenciómetro. [21]

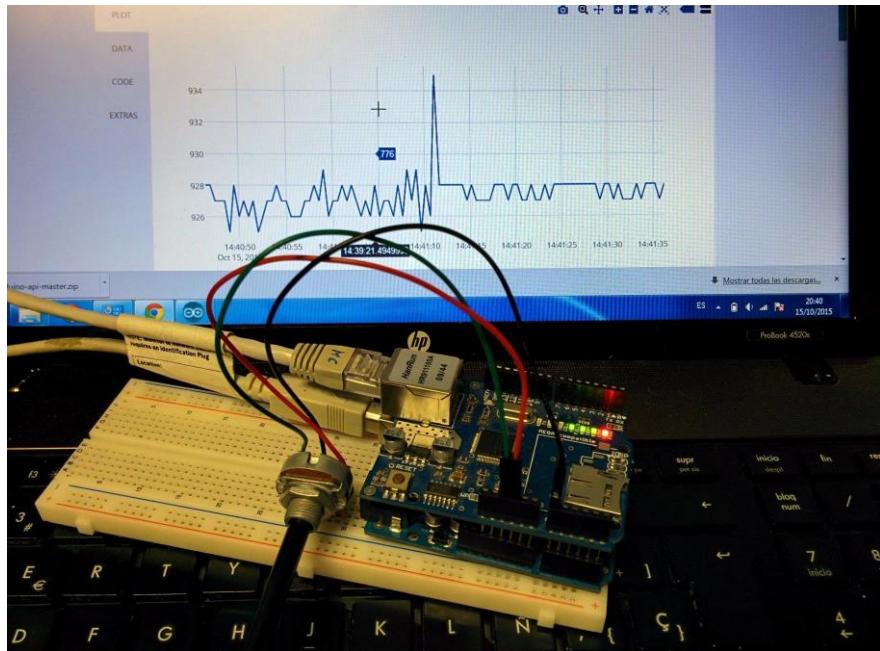


Fig.13.1 Vista Plotly

Con el paso del tiempo y haciendo varias pruebas, dejamos de obtener resultados y es por ello que descartamos el uso de dicha plataforma y añadimos un nuevo objetivo a nuestro proyecto, el cual consistió en realizar nuestra propia web para visualizar los datos.

b) Problema de rendimiento/memoria microcontrolador.

A medida que avanzaba el proyecto, la cantidad de código que íbamos implementando en nuestro procesador iba incrementándose, y con ello la memoria de nuestro microcontrolador iba agotándose.

Fue necesario, usar Arduino Mega, el cual nos proporciona más memoria.

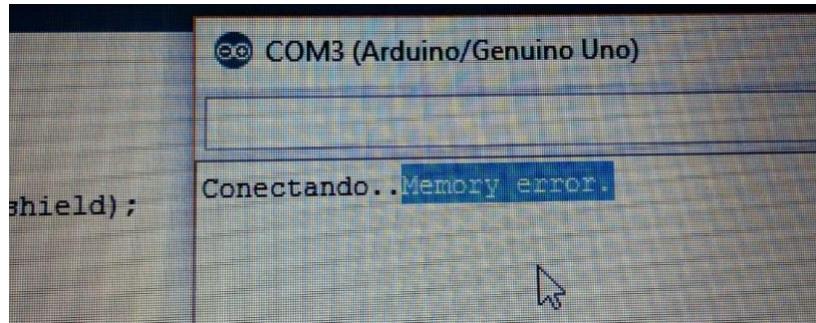


Fig.13.2 Mensaje Error Memoria

14. CONCLUSIONES

Se ha logrado diseñar e implementar un medidor de energía o “regleta inteligente” para instalaciones eléctricas domésticas mediante sistemas empotrados de bajo coste. Este sistema nos permite medir el consumo en tiempo real, de cualquier aparato electrónico que se conecte.

Además de poder interactuar con nuestro sistema de forma por medio de una app, podemos acceder a los datos leídos por nuestro sistema desde cualquier navegador web, visualizando el consumo actual, hora, diario y mensual.

Se añadió el acceso al sistema a través de usuario y contraseña.

Al principio presentaba una gran dificultad trabajar con herramientas totalmente desconocidas, como lo era el hardware, firmware de Arduino y librerías.

Otra dificultad que se presentó fue la construcción del circuito impreso PCB, pues era la primera.

Mejoras para una futura versión:

Mejora Hardware:

- Realización PCB: Se hará un diseño eficiente para construir
- Comunicación inalámbrica por radiofrecuencia: Para mayor comodidad, es recomendable usar tecnologías inalámbricas y en este caso más económico usar módulos RT.
- Uso de Raspberry Pi 3: Hará la función de servidor web

Mejora Software:

- Cambio de contraseña: Actualmente no hay posibilidad de cambiar la contraseña por medio de aplicación web.
- Poder controlar nuestro sistema por medio de la aplicación web

15. BIBLIOGRAFÍA

[1] Medidor de consumo individual de la marca efergy
<http://efergy.com/nz/products/energy-monitoring-socket#.V1XSa5GLTIU>

[2] Medidor de consumo individual de la marca efergy
<http://efergy.com/es/products/ego>

[3] Interruptor Wemo Insight
<http://www.belkin.com/es/p/P-F7C029/>

[4] Proyecto, Open Energy Monitor
<http://openenergymonitor.org/emon/>

[5] Proyecto, SEGMENTER
<http://smartenergygroups.com/categories/9-SEGmeter>

[6] Proyecto, Energy Visible
<http://www.visibleenergy.com/>

[7] Sensores ploggs
<http://www.treehugger.com/gadgets/remote-control-your-homes-energy-use-with-ploggs.html>

[8] DataSheet sensor ACS712 AC
<http://www.allegromicro.com/~/media/Files/Datasheets/ACS712-Datasheet.ashx>

[9] Página oficial de Arduino <http://www.arduino.cc/es/>

[10] Página oficial de Netduino <http://netduino.com/>

[11] Librería sobresampleo bits ADC: erCaGuy_NewAnalogdRead
<http://www.electricrcaircraftguy.com/2014/05/using-arduino-unos-built-in-16-bit-adc.html>

[12] Ethernet Shield Arduino
<https://www.arduino.cc/en/Main/ArduinoEthernetShield>

[13] DataSheet Bluetooth HC-06
http://www.tec.reutlingen-university.de/uploads/media/DatenblattHC-05_BT-Modul.pdf

[14] Lcd KeyPad Shield
<http://www.hobbytronics.co.uk/arduino-lcd-keypad-shield>

[15] Tutorial de Arduino usando un módulo bluetooth
<http://diymakers.es/arduinoblueooth/>

[16] Software Diseño PCB
<http://fritzing.org/home/>

[17] Framework Google Labs: AppInventor
<http://appinventor.mit.edu/explore/>

[18] Android Studio
<https://developer.android.com/studio/index.html?hl=es>

[19] Librería mySQL Connector
<https://github.com/ChuckBell/MySQL Connector Arduino>

[20] Plataforma web Plotly
<https://plot.ly/>

[21] Librería Plotly Arduino
<https://github.com/plotly/arduino-api>

[22] Tutorial Plotly Arduino
<https://www.youtube.com/watch?v=yqv6oj8CCEA>

[23] Tutorial Potenciómetro
<https://paruro.pe/aprende/arduino/b%C3%Alsico/arduino-b%C3%Alsico-lectura-anal%C3%B3gica-serial>

Tutoriales e información: Relés

Tutorial:
<http://www.hobbyist.co.nz/?q=android-control-arduino>

Información:
<http://www.mauroalfieri.it/elettronica/keyes-4-relay-shield-arduino.html>

Tutorial:
<https://www.youtube.com/watch?v=GQRJXm6ZD-k>

Tutorial:
<http://arduinobasics.blogspot.com.es/2014/09/relay-module.html>

Tutorial:
<https://geekytheory.com/arduino-ethernet-shield-relay/>

Tutoriales e información: Pantalla LCD-shield

Tutorial:
<http://elcajondeardu.blogspot.com.es/2013/12/tutorial-conectando-una-pantalla-lcd.html>

Tutorial:

<http://www.prometec.net/lcd-keypad-shield/>

Información:

<https://www.arduino.cc/en/Tutorial/LiquidCrystalDisplay>

Tutorial:

<https://www.youtube.com/watch?v=hdT0g7EJ3kc>

Tutorial:

<http://www.ajpdsoft.com/modules.php?name=News&file=article&sid=627>

Tutoriales e información: Visualización de datos

Tutorial:

<http://tronixstuff.com/2014/01/21/online-data-analysis-arduino-plotly/>

Tutorial:

<https://plot.ly/arduino/tmp36-temperature-tutorial/>

Tutorial:

<http://drmaker.es/plotly-el-servicio-web-para-visualizar-datos-en-tiempo-real-arduino-iot/>

Tutoriales e información: Sensor ACS712

Tutorial:

<http://www.leandroruiz.com/blog/amperimetro-con-arduino/>

Información:

<https://community.particle.io/t/current-sensor-acs712/3775/5>

Información:

<http://forum.arduino.cc/index.php?topic=281723.0>

Información:

<http://www.allegromicro.com/~/media/Files/Datasheets/ACS712-Datasheet.ashx>

Tutoriales e información: Bluetooth HC-06

Tutorial:

<http://www.prometec.net/bt-hc06/>

Tutorial:

<http://www.geekfactory.mx/tutoriales/bluetooth-hc-05-y-hc-06-tutorial-de-configuracion/>

Información:

http://www.tec.reutlingen-university.de/uploads/media/DatenblattHC-05_BT-Modul.pdf

Información: Diseño PCB

Guía de buenas prácticas:

<http://www.radio-electronics.com/info/electronics-design/pcb/pcb-design-layout-guidelines.php>

<http://www.electronics-project-design.com/PCB-Design.html>