UNIVERSIDAD DE OVIEDO

ESCUELA DE INGENIERÍA INFORMÁTICA

TRABAJO FIN DE GRADO

"CONSTRUYENDO UN PROYECTO DE IOT: SISTEMA DE MEDICIÓN DE CONSUMO ELÉCTRICO"

AUTOR: JUAN GRANDA MOLAGUERO

DIRECTOR: SECUNDINO JOSE GONZÁLEZ PÉREZ

Agradecimientos

Para todos los amigos, en especiales a los dokkanes, que tengo gracias a la universidad, y por todos los recuerdos que me llevo de este periodo tan importante.

También al tutor de mi proyecto, que apostó por este proyecto que se salía del ámbito de lo aprendido.

Y finalmente a todas las personas que han estado ahí apoyándome para acabar esta etapa de mi vida.

Resumen

Actualmente el consumo eléctrico es uno de los grandes problemas tanto en hogares como en empresas, es por ello por lo que en el mercado se puede encontrar un catálogo de enchufes inteligentes que muestra el consumo de nuestros dispositivos, o por otro lado packs para medir el consumo a través de sistemas invasivos o pinzas.

Algunas desventajas de estos productos son:

- Sistemas invasivos o de difícil instalación.
- Coste elevado, debido a que se necesitan múltiples dispositivos para su instalación.
- Amperaje bajo, suficiente para el hogar, pero no para las empresas.

Es por ello por lo que en este proyecto se va a diseñar y construir un prototipo de sistema consumo eléctrico tanto para el hogar como para las empresas, de bajo coste, con un solo dispositivo.

Para ello el proyecto se dividirá en las siguientes partes:

- Construcción del dispositivo de telemedida.
- Creación de software para el dispositivo de telemedida.
- Aplicación web para mostrar la información del dispositivo y la administración de estos.

La primera fase se diseñará y construirá un dispositivo de telemedida eléctrica, con un Arduino y una pinza de lectura de consumo de consumo. La cual se conectará de forma no invasiva al electrodoméstico o dispositivo que queramos medir su consumo, o a una regleta con varios elementos conectados para medir el total de su consumo.

En la segunda parte, se diseñará e implantará en el dispositivo el código que haga posible convertir la salida de la pinza de medición en potencia eléctrica consumida. A su vez, se realizará el código necesario para enviar la información obtenida al servidor que aloja la parte web del proyecto.

Finalmente, se diseñará e implementará un servicio web para observar las mediciones obtenidas y administrar nuestros dispositivos. Constará de dos partes:

- Una API REST para recibir la información obtenida.
- Una página web para mostrar la información obtenida a través del apartado anterior.

Palabras Clave

SCT-013, Controlador, Petición, Voltaje, IOT.

Abstract

Nowadays, electricity consumption is one of the major problems in homes and companies, this is the reason that you can find in the market a catalog of smart plugs that shows the consumption of devices, or on the other hand packs to measure the consumption through invasive systems or tweezers.

Some disadvantages of these products are:

- Invasive systems or difficult installation.
- High cost, because multiple devices are needed for installation.
- Low amperage, enough for home, but not for businesses.

This is the reason, this project is going to design and build a prototype system of electricity consumption for home and business, low cost and with a single device.

The project will be divided into the following parts:

- Construction of the telemetering device.
- Creation of software for the telemetering device.
- Web application to display the device information and its administration.

The first phase will be designed and built an electric telemetry device, with an Arduino devise and a consumer consumption reading clamp. Which will connect non-invasively to the appliance or device that we want to measure its consumption, or to a strip with several connected elements to measure the total consumption.

In the second part, the code that makes it possible to convert the output of the measuring clamp into electrical power consumed will be designed and implemented in the device. In turn, the necessary code will be made to send the information obtained to the server that hosts the web part of the project.

Finally, a web service will be designed and implemented to observe the measurements obtained and manage our devices. It is made up of two parts:

- A REST API to receive the information obtained.
- A web page to display the information obtained through the previous section.

Índice General

Contenido

CAPÍTULO 1. MEMORIA DEL PROYECTO 11

1.1 Resumen de la Motivación, Objetivos y Alcance del Proyecto 11
1.2 Resumen de todos los Aspectos 12

CAPÍTULO 2. INTRODUCCIÓN 13

2.1 Justificación del Proyecto 13
2.2 Objetivos del Proyecto 13
2.3 Estudio de la Situación Actual 14
2.3.1 Evaluación de Alternativas 17

CAPÍTULO 3. ASPECTOS TEÓRICOS 22

3.1 IOT o Internet de las cosas 22
3.2 Transformador de corriente 22
3.3 Ley de Ohm 23
3.4 Arduino 23
3.5 Arduino IDE 26
3.6 Emonlib 27
3.7 Wifinina 27
3.8 Métodos de petición HTTP 27
3.9 Señal Sinusoidal 27

CAPÍTULO 4. PLANIFICACIÓN DEL PROYECTO Y PRESUPUESTO INICIALES 29

4.1 Planificación Inicial 29
4.1.1 Resumen de tareas 30
4.2 Presupuesto Inicial 31
4.2.1 Desarrollo de Presupuesto Detallado (Empresa) 31
4.2.2 Desarrollo de Presupuesto Simplificado (Cliente) 34

CAPÍTULO 5. ANÁLISIS 35

5.1 Definición del Sistema 35
5.1.1 Determinación del Alcance del Sistema 35
5.2 Requisitos del Sistema 35
5.2.1 Requisitos funcionales 35
5.2.2 Requisitos no funcionales 39
5.2.3 Identificación de Actores del Sistema 39
5.3 Análisis de Casos de Uso 40
5.3.1 Diagrama de contexto 40
5.3.2 Casos de uso 40
5.4 Análisis de Interfaces de Usuario 49
5.4.1 Descripción de la Interfaz 49
5.4.2 Descripción del Comportamiento de la Interfaz 53
5.4.3 Diagrama de Navegabilidad 53

5.4.4 Diagrama de construcción del dispositivo 54

CAPÍTULO 6. DISEÑO DEL SISTEMA 55

6.1 Arquitectura del Sistema 55
6.1.1 Diagramas de Componentes 55
6.1.2 Diagrama de Despliegue 56
6.2 Diagrama de secuencia 57
6.2.1 Diagrama de secuencia "Envío de información de Arduino a API REST" 58
6.2.2 Diagrama de secuencia "Mostrar datos a usuario" 59
6.3 Diseño de la Base de Datos 59
6.3.1 Descripción del SGBD Usado 59
6.3.2 Integración del SGBD en Nuestro Sistema 59
6.3.3 Diagrama E-R 60
6.4 Diseño de la Interfaz 60
6.4.1 Barra de navegación 60
6.4.2 Pantalla de login 61
6.4.3 Pantalla de registro 62
6.4.4 Pantalla principal 62
6.4.5 Pantalla detalles de dispositivo 63
6.4.6 Pantalla editar dispositivo 64
6.4.7 Pantalla añadir dispositivo 64
6.5 Especificación Técnica del Plan de Pruebas 65
6.5.1 Pruebas unitarias 65
6.5.2 Pruebas de integración 69

CAPÍTULO 7. IMPLEMENTACIÓN DEL SISTEMA 72

7.1 Lenguajes de Programación 72
7.1.1 C/C++ Arduino 72
7.1.2 Ruby 73
7.1.3 Rails 73
7.1.4 HAML 74
7.2 Herramientas y Programas Usados para el Desarrollo 74
7.2.1 Herramientas para el desarrollo 74
7.2.2 Herramientas para la documentación 75
7.3 Creación del Sistema 77
7.3.1 Diagrama de construcción 77
7.3.2 Software del dispositivo 81
7.3.3 Software de la API REST 85
7.3.4 Software de Aplicación Web 87
7.4 Problemas Encontrados 90
7.4.1 Problema 1 90
7.4.2 Problema 2 90
7.4.3 Problema 3 91
7.4.4 Problema 4 91
7.4.5 Problema 5 91

CAPÍTULO 8. DESARROLLO DE LAS PRUEBAS 92

8.1 Pruebas Unitarias 92
8.2 Pruebas de Integración y del Sistema 97

CAPÍTULO 9. MANUALES DEL SISTEMA 98

9.1 MANUAL DE INSTALACIÓN 98
9.2 MANUAL DE EJECUCIÓN 98
9.3 MANUAL DE USUARIO 99
9.3.1 Registro de usuario 99
9.3.2 Inicio de sesión 100
9.3.3 Añadir un dispositivo 100
9.3.4 Ver detalles de un dispositivo 101
9.3.5 Editar datos de un dispositivo 102
9.3.6 Desvincular un dispositivo 103
9.3.7 Desconectarse de la sesión 104
9.4 MANUAL DEL PROGRAMADOR 105

CAPÍTULO 10. CONCLUSIONES Y AMPLIACIONES 106

10.1 CONCLUSIONES 106
10.2 AMPLIACIONES 106

CAPÍTULO 11. PRESUPUESTO 108

11.1 DESARROLLO DE PRESUPUESTO DETALLADO (CLIENTE) 108

CAPÍTULO 12. REFERENCIAS BIBLIOGRÁFICAS 110

CAPÍTULO 13. APÉNDICES 111

13.1 GLOSARIO Y DICCIONARIO DE DATOS 111
13.2 CONTENIDO ENTREGADO EN EL ARCHIVO ADJUNTO 111
13.2.1 Contenidos 111
13.2.2 Código Ejecutable e Instalación 113
13.2.3 Ficheros de Configuración 114

Índice de Figuras

Ilustración 1 Enchufe inteligente....14
Ilustración 2 Regleta inteligente....15
Ilustración 3 Pinza de medición....15
Ilustración 4 Medidor de cuadro eléctrico....16
Ilustración 5 Software de consumo....16
Ilustración 6 Panel de software de consumo....17
Ilustración 7 Placa Raspberry Zero....18
Ilustración 8 Placa Arduino Uno....19
Ilustración 9 Placa Arduino MKR1010 Wifi....19
Ilustración 10 Pinza SCT-013-030....20
Ilustración 11 Pinza SCT-013-100....20
Ilustración 12 Diagrama sensor de corriente....22
Ilustración 13 Primera placa Arduino....24
Ilustración 14 Evolución de las placas Arduino....25
Ilustración 15 Arduino IDE....26
Ilustración 16 Gráfica de una señal de corriente....28
Ilustración 17 Planificación parte 1....29
Ilustración 18 Planificación parte 2....29
Ilustración 19 Diagrama de contexto....40
Ilustración 20 Diagrama robustez: añadir dispositivo....42
Ilustración 21 Diagrama robustez: Registro en el sistema....43
Ilustración 22 Diagrama de robustez: Inicio de sesión....44
Ilustración 23 Diagrama de robustez: acceder a un dispositivo....45
Ilustración 24 Diagrama robustez: Editar dispositivo....46
Ilustración 25 Diagrama de robustez: Desvincular dispositivo....47
Ilustración 26 Interfaz de Inicio de sesión....50
Ilustración 27 Interfaz zonas de trabajo....50
Ilustración 28 Prototipo listado de dispositivos....51
Ilustración 29 Prototipo vista de información de dispositivo....52
Ilustración 30 Prototipo formulario de edición y añadir dispositivo....53
Ilustración 31 Diagrama de navegabilidad....53
Ilustración 32 Boceto de construcción del dispositivo....54
Ilustración 33 Diagrama de Componentes....55
Ilustración 34 Diagrama de Despliegue....56
Ilustración 35 Diagrama de envío de datos de Arduino a API REST....58
Ilustración 36 Diagrama mostrar datos de dispositivo a usuario....59
Ilustración 37 Diagrama E-R....60
Ilustración 38 Barra de navegación....61
Ilustración 39 Pantalla final: Inicio de sesión....61
Ilustración 40 Pantalla final: registro de usuario....62
Ilustración 41 Pantalla principal....62
Ilustración 42 Pantalla final: detalles de usuario....63
Ilustración 43 Pantalla final: editar dispositivo....64
Ilustración 44 Pantalla final: añadir dispositivo....64
Ilustración 45 Logo de C++....72

Ilustración 46 Logo Ruby **73**
Ilustración 47 Logo de Ruby on Rails **73**
Ilustración 48 Logo Haml **74**
Ilustración 49 Logo de Atom **74**
Ilustración 50 Logo de GitHub **75**
Ilustración 51 Logo Postman **75**
Ilustración 52 Logo Draw.io **76**
Ilustración 53 Logo Fritzing **76**
Ilustración 54 Diagrama de construcción del dispositivo **77**
Ilustración 55 Tabla de entrada y salida de los sensores SCT-013 ..... **78**
Ilustración 56 Construcción del dispositivo sin el sensor **80**
Ilustración 57 Construcción del dispositivo con el sensor **81**
Ilustración 58 Configuración de SSID y contraseña wifi **82**
Ilustración 59 DNS y puerto del servidor **82**
Ilustración 60 Petición GET Arduino **82**
Ilustración 61 Tratar información recibida. **84**
Ilustración 62 Inicializar objeto EnergyMonitor **85**
Ilustración 63 Archivo de configuración de rutas **86**
Ilustración 64 Añadir un nuevo consumo **87**
Ilustración 65 Paginación en controlador **88**
Ilustración 66 Configuración del modelo para las gráficas **89**
Ilustración 67 Gráfica registro diario **90**
Ilustración 68 Registro de usuario **99**
Ilustración 69 Inicio de sesión **100**
Ilustración 70 Listado de dispositivos de un usuario **101**
Ilustración 71 Detalles de un dispositivo **102**
Ilustración 72 Pulsar "Editar" en los detalles del dispositivo **102**
Ilustración 73 Editar dispositivo **103**
Ilustración 74 Desvincular dispositivo **103**
Ilustración 75 Desconectarse de la sesión **104**

Capítulo 1. Memoria del Proyecto

1.1 Resumen de la Motivación, Objetivos y Alcance del Proyecto

Hoy en día existen dispositivos de medición de consumo para el hogar o empresas, pero su precio es elevado o dependen de instalaciones avanzadas para poder ser utilizados. Además, los datos de las mediciones obtenidas pueden ser utilizadas por las empresas, debido a que los datos de medición de consumo de hogar es información de gran importancia y con un alto costo en el mercado.

Es por ello por lo que se pretende realizar un proyecto IOT donde el dispositivo sea de bajo coste, se escalable y su mantenimiento sea el menor. Por otra parte, se busca que el sistema no sea invasivo, es decir, que no sea preciso realizar modificaciones en los dispositivos a medir y no necesite de modificaciones en la instalación eléctrica. Y por último que la información de las mediciones obtenidas no sea tratada por una empresa, si no que las tenga el usuario y sea el quien trabaje con ellas. Para ello, se le proporcionará una manera sencilla de visualizar los datos obtenidos.

El proyecto utilizará por ello elementos de bajo coste como una Raspberry como servidor, la cual será configurada a través del manual de usuario, y el dispositivo será montado en una placa de Protoboard y no se instalará en una placa PCB así no necesitar de soldaduras.

Finalmente, el software para mostrar la información no tendrá gran complejidad ya que en este proyecto se encuentra centrado en la construcción del dispositivo y de la API REST que trata sus datos.

1.2 Resumen de todos los Aspectos

En este documento se tratará de todo lo relacionado con el desarrollo del proyecto. Por lo tanto se tratarán diversos temas entre los que se pueden encontrar:

- Objetivo del proyecto y la situación actual de mercado con otros dispositivos con funcionalidades parecidas.
- Planificación y presupuesto para desarrollar el proyecto al completo.
- Problemas encontrados a lo largo del proyecto, tanto en la construcción del dispositivo como en el desarrollo software de este.
- Casos de uso que se pueden dar en este proyecto, junto con pruebas que muestren el resultado esperado y obtenido.
- Manuales para el usuario de cómo instalar, funcionalidades y mantenimiento del dispositivo.
- Cómo está diseño el software del sistema y cómo interactúan las distintas partes del proyecto unas con otras.

Capítulo 2. Introducción

2.1 Justificación del Proyecto

Hoy en día el consumo eléctrico es algo importante tanto en el hogar como para la empresa, ya que permite ahorrar dinero conociendo los consumos de nuestros dispositivos. Muchos de los productos que existen en el mercado sólo permiten cortar o dejar pasar la corriente para que funcione un dispositivo o programarlos para que estén encendidos durante ciertos periodos. Pero tienen una carencia, ya que no muestran al usuario una forma de saber o contabilizar el consumo que tenemos en el hogar.

Otros dispositivos ofrecen esto último, pero su precio aumenta y tener varios dispositivos por el hogar haría que su precio sea muy elevado. Es por tanto que este proyecto pretende con un coste bajo una forma de tener un registro en tiempo real del consumo de electricidad de los electrodomésticos o demás aparatos eléctricos del hogar. Además permite que la información solo sea accesible por el usuario ya que cuenta con un servidor personal el cual tratará la información y no la compartirá con terceros, ya que será el propio usuario o a través de terceros el que lleve la instalación de este sistema.

Por otra parte al ser este proyecto un prototipo de lectura de corriente, algo que a un usuario medio puede parecerle no complicado y debido al uso de varias tecnologías y librerías, se puede utilizar como método para enseñar o iniciarse el mundo de proyectos de IOT con Arduino, ya que se utilizan tecnologías de comunicación con servidores mediante peticiones, tanto de envío cómo de recepción de datos, además de ver cómo se puede configurar un dispositivo Arduino a través de conexión Wifi.

Es por todo ello que se diseña y crea un prototipo de sensor de corriente junto con una aplicación web, los cuales pueden ser personalizados y ampliados por el usuario o tenerlo como base para otros proyectos.

2.2 Objetivos del Proyecto

A continuación se mostrará un listado con los objetivos que se quieren cumplir en este proyecto, diferenciado en dos partes. La primera referida al dispositivo Arduino y la API REST, y por otro lado la aplicación web.

- Dispositivo y API REST:
    - El dispositivo deberá tener un coste asequible y su construcción deberá ser sencilla sin poseer conocimientos avanzados de electrónica.
    - El dispositivo deberá poder medir corriente eléctrica por medio de una regleta, previamente personalizada, de cualquier aparato electrónico del hogar. Cómo restricción y al ser un sensor de corriente fásico, se limitará a dispositivos que necesiten sólo un cable fásico para su funcionamiento.

    - El dispositivo deberá poder conectarse a una red wifi, configurando los parámetros necesarios como el SSID y la contraseña wifi a la que el usuario desea conectarse.
    - El dispositivo deberá ser capaz de recibir datos de la API REST y a su vez ser capaz de enviar los datos necesarios a la misma.
    - La API REST deberá poder ser lanzada en máquinas que no requieran gran potencia y podrá ser escalable.
    - La API REST deberá ser capaz de enviar y recibir información del dispositivo, y dicha información enviarla a una base de datos.
- Aplicación web:
    - Deberá aportar un sistema de usuarios referido a inicio de sesión y registro de estos.
    - Mostrará el conjunto de dispositivos que tenga un usuario asociados.
    - Permitirá añadir, editar y desvincular dispositivos a sus usuarios.
    - Mostrará información del consumo de los dispositivos tanto en tiempo real, cómo un registro de los consumos previos.

Finalmente como objetivo de este proyecto a parte de lo anteriormente descrito, es el de tener un base con el que ampliar tanto el dispositivo como la funcionalidad de este, ya sea por un usuario o por una empresa especializada.

2.3 Estudio de la Situación Actual

Para entender bien el proyecto, primero se mostrará que es un dispositivo de telemedida de consumo eléctrico. Ya que se pueden dar dos casos:

1. Medidores de consumo de manera individual, en ellos se encuentran los enchufes eléctricos, también están incluidas las regletas para varios dispositivos o pinzas de medición. Algunos ejemplos son:
    1.1. Enchufe inteligente:

Ilustración 1 Enchufe inteligente

    1.2. Regleta inteligente:

Ilustración 2 Regleta inteligente

1.3. Pinza de medición:

Ilustración 3 Pinza de medición

2. Por otro lado, tenemos los medidores de consumo que van al cuadro eléctrico. A diferencia de los anteriores, éstos miden el consumo general, es por ello por lo que no sabremos el consumo específico de cada dispositivo. Algunos ejemplos son:
   2.1. Dispositivo a cuadro eléctrico:

Ilustración 4 Medidor de cuadro eléctrico

Todos los dispositivos anteriores pueden mostrar la información de dos formas, a través de Internet o a través de pantallas que incorpora el propio dispositivo. Esta última opción a veces no es práctica, ya que puede encontrarse oculto por tener elementos que obstaculicen su visión o que por ejemplo, al estar instalados en el contador eléctrico no sean de fácil acceso.

Por otro lado tenemos el software de ciertos dispositivos, ya sea web o de aplicación móvil, el cual muestra al usuario la información del medidor de consumo. En todas ellas podemos ver un patrón común, ya que suelen mostrar diversas gráficas para el consumo.

Ilustración 5 Software de consumo

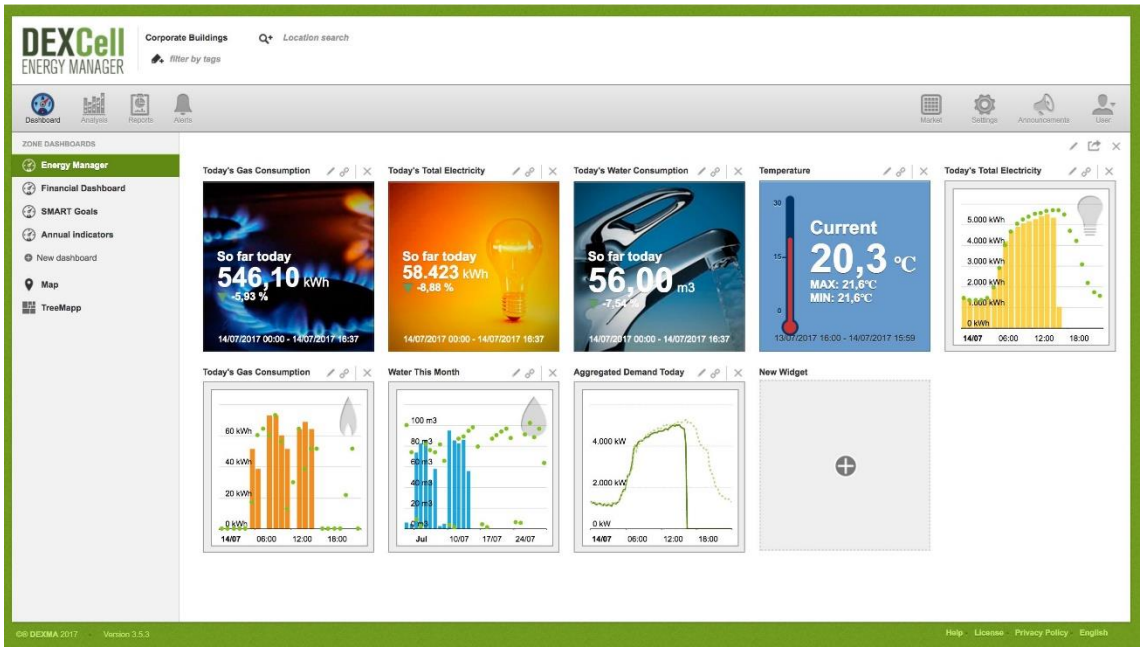


Ilustración 6 Panel de software de consumo

Finalmente, muchos de los dispositivos anteriores tienen un límite de medición de consumo muy bajo, suficiente para un dispositivo, pero no para varios o para maquinaría con consumo o amperajes altos. Además, otros necesitan de instalaciones complejas para el usuario medio.

Es por ello, que se diseñará y construirá un dispositivo de telemedida de consumo, de fácil instalación, bajo precio y con un servicio web que facilite la lectura de las mediciones.

2.3.1 Evaluación de Alternativas

En este apartado se mostrarán distintas opciones tanto a nivel de hardware como a nivel de software, éste último se decantará debido a los estudios del alumno.

2.3.1.1 Alternativa Hardware

Para hacer el proyecto con el menor coste posible, que sea capaz de transmitir los datos recogidos de la pinza de medición y que sea de pequeño tamaño para no hacer voluminoso el dispositivo, se barajan distintas opciones:

Raspberry Pi Zero:

Ilustración 7 Placa Raspberry Zero

Es un ordenador de tamaño reducido, el cual cuenta con su propio sistema operativo llamado Raspbian. Cuenta con Wifi incluido, lo que permite no tener que comprar un dispositivo adicional para añadir internet. Además posee los pines necesarios para realizar la instalación.

El problema de elegir esta opción es:

- No dispone de pines analógicos, los cuales son necesarios para la lectura de salida de la pinza de medición de consumo [2].

Arduino

Al igual que en el apartado anterior son ordenadores de pequeño tamaño, además existen diferentes modelos dependiendo de las necesidades del usuario. Para este proyecto se destacan dos soluciones:

- **Arduino UNO con chip ESP8266:** se basa en el Arduino UNO pero le añade el chip ESP8266, el cual descartamos debido a que tiene un precio elevado, además de ser bastante voluminoso respecto a lo que estamos buscando.

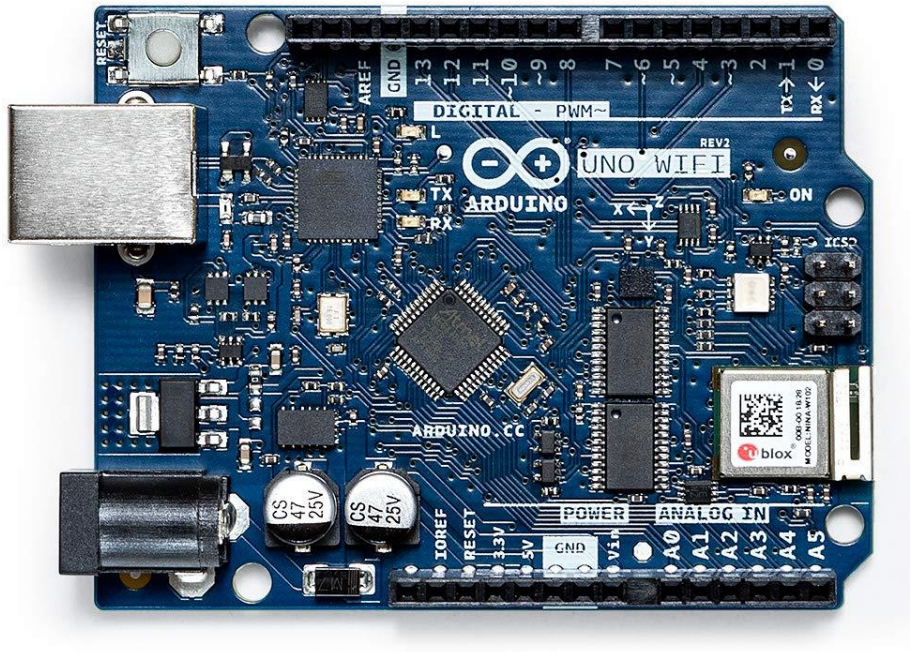


Ilustración 8 Placa Arduino Uno

- **Arduino MKR1010 Wifi:** Es la opción final escogida, debido a que presenta un tamaño que se ajusta con lo que se busca, posee pines analógicos para conectar la pinza de lectura de consumo y finalmente tiene integrado el chip necesario para tener Wifi en el dispositivo. Además su precio se ajusta de tal manera que podamos hacer el dispositivo barato.

Ilustración 9 Placa Arduino MKR1010 Wifi

Para la parte de lector de datos de consumo tenemos dos opciones:

- **SCT-013-030:** Su precio es barato, pero su principal característica y diferencia con el que veremos a continuación es que permite realizar medidas en un rango de 3 amperios, posee una resistencia de carga interna. Su relación es 30ª/1V.

Ilustración 10 Pinza SCT-013-030

- **SCT-013-100:** A diferencia del anterior con este podemos medir corrientes de hasta 100A, lo que da una salida de 50mA, por tanto su proporción es 100A/50mA

Ilustración 11 Pinza SCT-013-100

De estas dos opciones se ha escogido la segunda opción debido a que su coste es inferior y permite medir corrientes de mayor cantidad.

2.3.1.2 Alternativa Software

Para la parte software se han dispuesto dos opciones a la hora de configurar el Arduino, Python y el lenguaje propio de Arduino, Processing. Se optó por la segunda opción debido a las herramientas y librerías que proporciona Arduino y la comunidad para la transmisión de datos.

Para la parte de web se optó por Ruby on Rails, el cual es un framework de aplicaciones web basado en el lenguaje de programación Ruby, debido que ofrece librerías que facilitan el desarrollo de ciertas funcionalidades tanto en API REST como para la visualización de datos.

Capítulo 3. Aspectos Teóricos

3.1 IOT o Internet de las cosas

El término IOT, proviene del inglés *"Internet Of Things"*, que traducido al español sería el Internet de las cosas. El significado que se le puede dar viene dado por la conexiones que tienen los dispositivos entre sí a través de la red, tanto local como por Internet. En este proyecto el claro ejemplo es como conectamos varios dispositivos, en este caso los sensores de consumo eléctrico con los electrodomésticos, para enviar la información al usuario.

El objetivo final del IOT es conectar dispositivos entre sí para que, sin la necesidad de la ayuda humana, realicen operaciones entre ellos y se comuniquen entre sí para llevarlas a cabo de forma correcta.

3.2 Transformador de corriente

El dispositivo SCT-013 utilizado, son transformadores de corriente los cuales generan intensidad en el secundario proporcional a la intensidad que atraviesa el primario. A continuación, se muestra una imagen que describe el proceso.

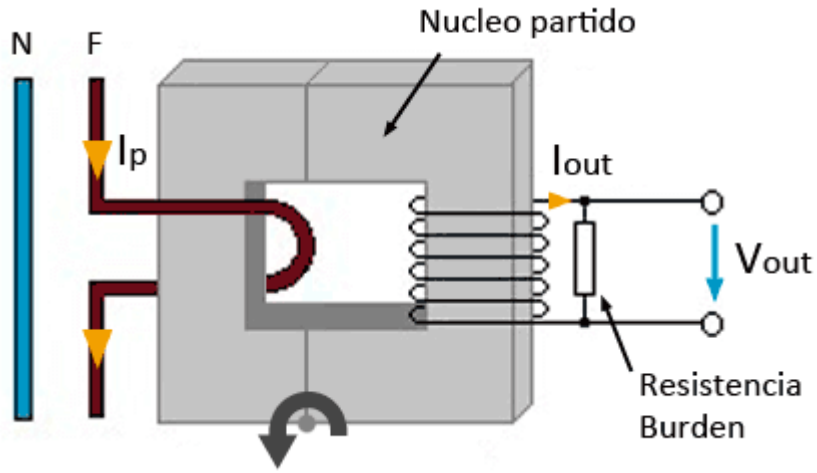


Ilustración 12 Diagrama sensor de corriente

Cómo se aprecia en la figura anterior el núcleo partido de esta forma se puede abrir y arrollar un conductor. Alrededor de este núcleo está un conjunto de espiras, de esta forma podemos obtener un transformador el cual el cable por el que circula la intensidad (cable F o marrón) constituye un devanado primario, nuestro dispositivo SCT-013 es el núcleo magnético y finalmente el devanado secundario está integrado como parte de la sonda.

3.3 Ley de Ohm

El proyecto se basa en la ley de Ohm, ya que a través de la pinza SCT-013 obtenemos la intensidad de corriente del dispositivo a medir. Y aplicando la fórmula: $V = R \cdot I$, donde V es el voltaje a obtener, R la resistencia y finalmente I la intensidad medida.

En nuestro caso tenemos una corriente alterna, por lo tanto estamos empleando valores RMS (Raíz Media Cuadrática), esto hace que nuestra forma de obtener el voltaje varíe y apliquemos la fórmula siguiente: $V_{pico} = \sqrt{2} \cdot V_{RMS}$, donde $V_{RMS} = R \cdot I_{pico}$, obteniendo finalmente $V_{pico} = \sqrt{2} \cdot R \cdot I_{RMS}$. Finalmente tendremos que obtener el voltaje pico a pico, con esta fórmula: $V_{picoapico} = 2 \cdot V_{picoapico} = 2 \cdot \sqrt{2} \cdot R \cdot I_{RMS}$.

3.4 Arduino

Es una empresa que se dedica a la construcción de microcontroladores. Cuyo software y hardware son libres y de fácil uso. Fue creada en el año 2005 por el instituto IVRAE Massimo Banzi, en Italia, debido a la necesidad de aprender a construir y programar a bajo coste, ya que en aquella época se trabajaba con dispositivos llamados BASIC Stamp, los cuales tenían un precio demasiado elevado alrededor de 100 dólares [1]. Por lo tanto se crea el proyecto para generar herramientas de bajo coste y simples de programar. El lenguaje de programación será Processing, un lenguaje basado en C++, debido a que varios de sus fundadores trabajan con él en otros proyectos en los que se encontraban.

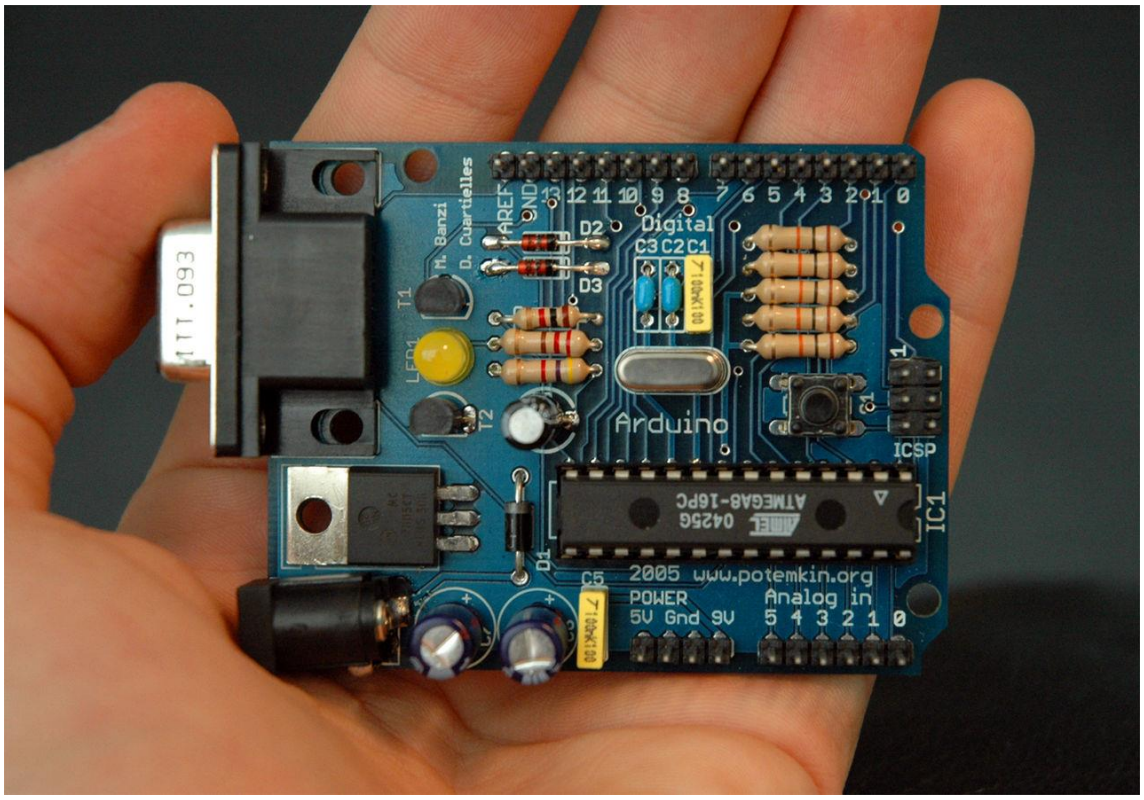


Ilustración 13 Primera placa Arduino

A lo largo de los años los microcontroladores de Arduino han ido modificándose y mejorando, generando distintos modelos, dependiendo de las necesidades que fueran surgiendo. Varían tanto de tamaño, como de componentes y voltajes. A continuación, se muestra una imagen de la evolución de los distintos Arduino, sacada de la revista MAKE.

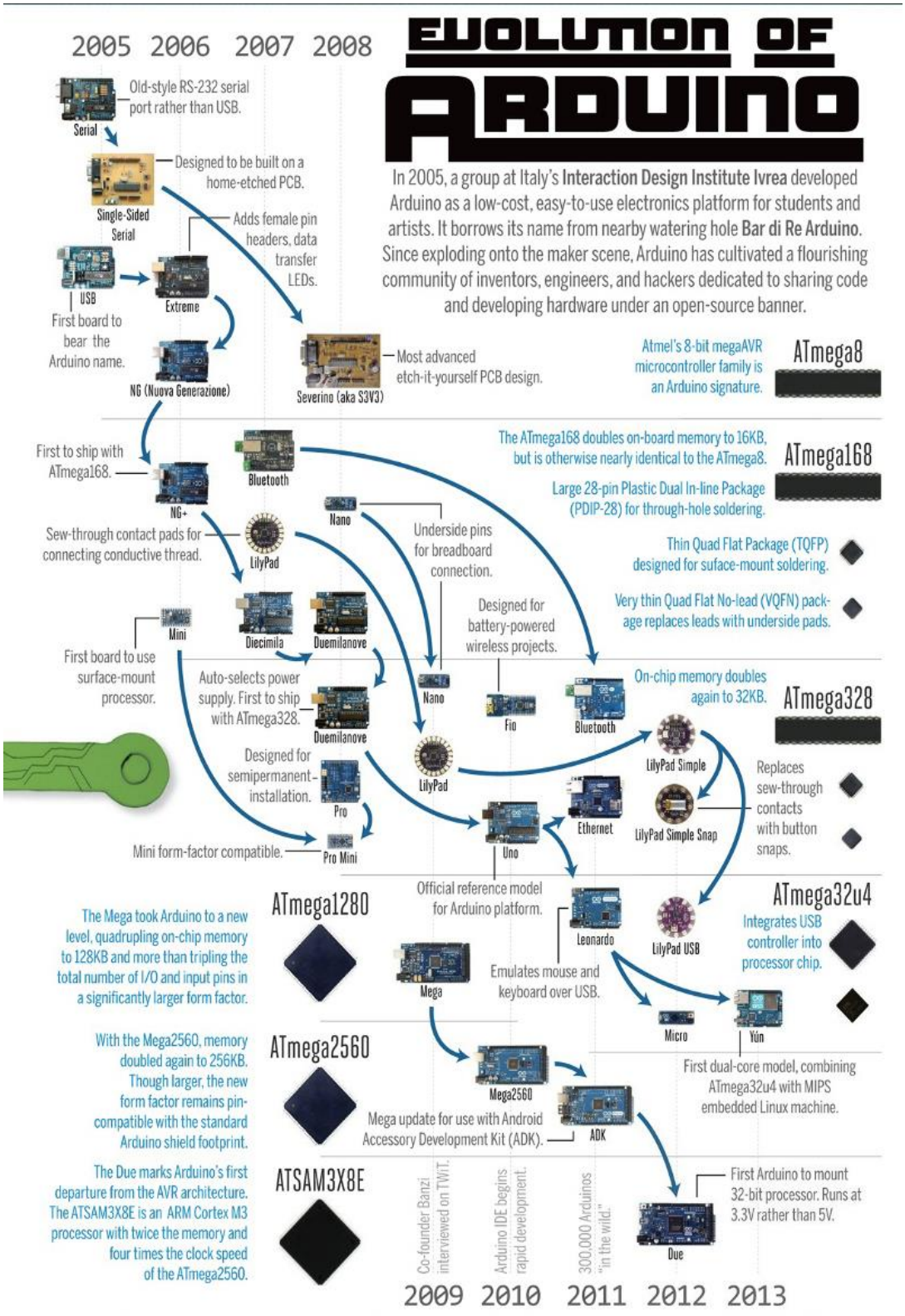


Ilustración 14 Evolución de las placas Arduino

3.5 Arduino IDE

Es un entorno de desarrollo especializado y creado para trabajar con placas Arduino. Es capaz de compilar programas escritos en lenguaje C++. Además aporta una serie de herramientas útiles para trabajar con dichas placas entre las que se puede encontrar:

- Administración de bibliotecas: gracias a esta herramienta podemos buscar e incluir librerías a nuestro proyecto de manera sencilla.
- Selección de puerto del dispositivo: esta herramienta es útil cuando tenemos varios dispositivos conectados a nuestro ordenador y queremos especificar a que dispositivo queremos cargar nuestro software.
- Monitor serie donde podemos observar la salida que obtenemos en nuestra placa y además podemos introducir datos a la misma.
- Finalmente podemos compilar y subir el software a nuestro dispositivo de manera sencilla por medio de dos botones.

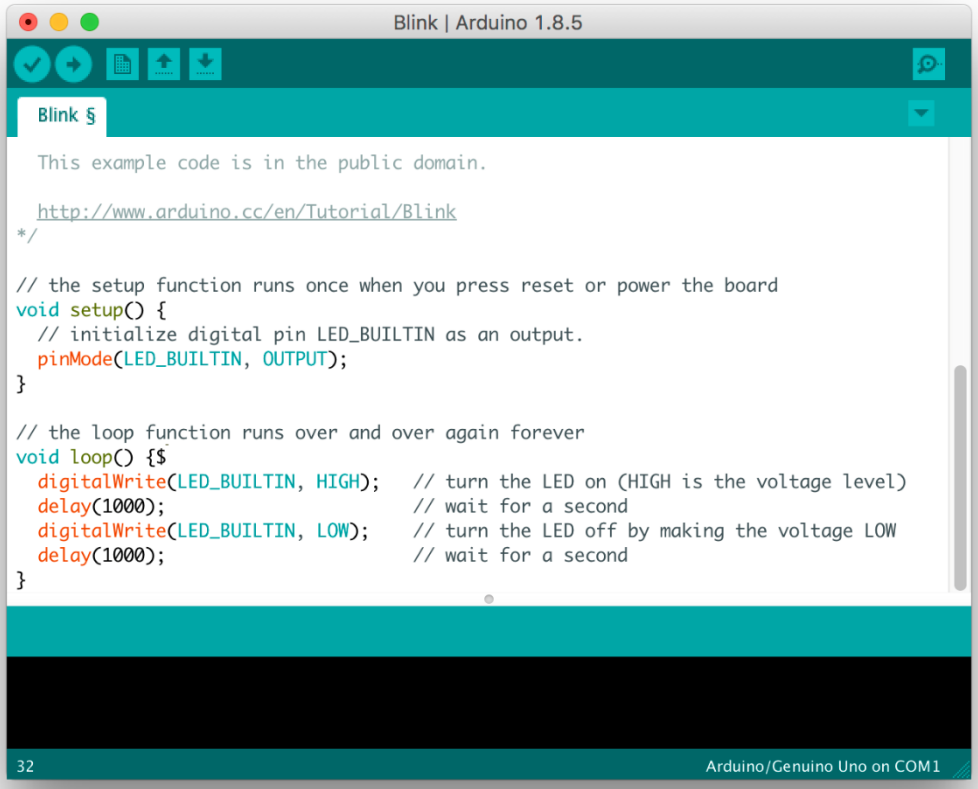


Ilustración 15 Arduino IDE

3.6 Emonlib

Esta es una librería diseñada en Processing para Arduino y fue diseñada en el proyecto Open Energy Monitor. Su objetivo es el de proporcionar de manera sencilla la medición desde un pin analógico del dispositivo la corriente que le llega, para ello tiene una función de inicialización, a la cual debemos pasarle el pin de entrada y el factor de calibración, éste último deberá ser calculado, ya que varía según el voltaje del dispositivo, el controlador Arduino y el voltaje de la propia red del hogar, además de la gama de SCT-013 que utilicemos.

3.7 Wifinina

Es una librería de Arduino que proporciona de una manera más sencilla habilitar el Wifi de nuestro dispositivo Arduino MKR 1010 – Wifi, ya que indicando a través de un fichero que incluiremos en el proyecto de nuestro código con el SSID de la wifi a conectarse y la contraseña podemos acceder ya a internet. Además, podemos crear un cliente de envío y recepción de información, para ello es necesario indicar el servidor o servidores que estaremos conectados, y a continuación crear las peticiones necesarias para dicho envío y recepción [5].

3.8 Métodos de petición HTTP

En este apartado describiré algunas de las distintas peticiones que se pueden hacer por HTTP, ya que son los utilizados en este proyecto. Estas peticiones son acciones que son enviadas o recibidas por el cliente o servidor, que según la configuración de cada software realizaran una serie de acciones. Entre las dos peticiones más comunes y que se utilizan en este proyecto son:

- Petición GET: son peticiones que se encargan de recibir datos enviados, tanto el encabezado cómo el cuerpo de la petición, ya que si solo se desea información en el encabezado podríamos usar peticiones HEAD.
- Petición POST: que envía información a un recurso específico y tras ello surgen una serie de modificaciones sobre ese recurso o acciones que deriven de este.
- Petición DELETE: permite eliminar un recurso específico del sistema.
- Petición PATCH: esta petición a diferencia de la petición PUT permite modificar sólo ciertos valores de un recurso o conjunto de recursos del sistema, a diferencia de la PUT que modifica todo el registro con la última información recibida.

3.9 Señal Sinusoidal

Debido a que la corriente que estamos midiendo es corriente de tipo alterna, su forma de representación se basa en una señal sinusoidal, esto quiere decir que sigue una gráfica de tipo seno.

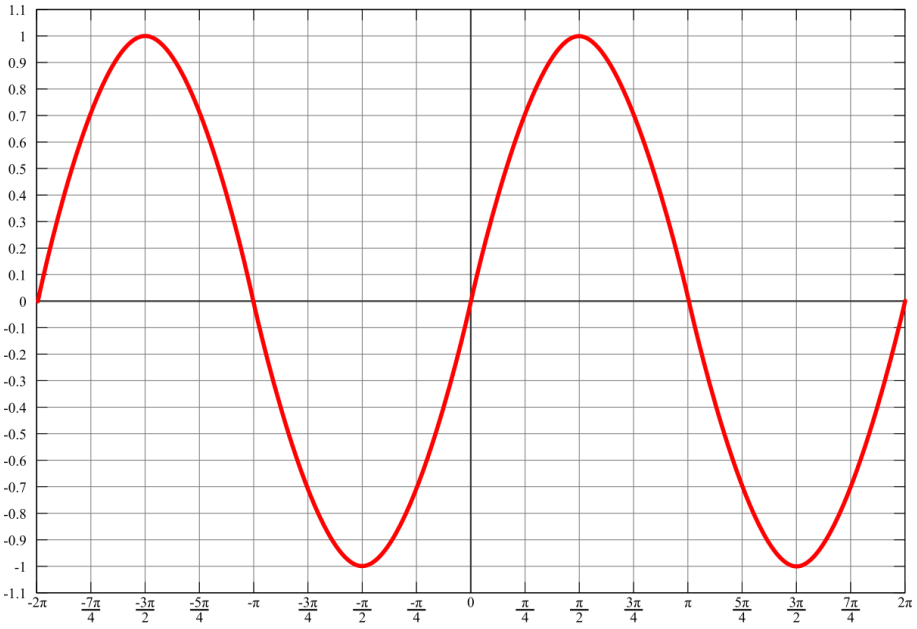


Ilustración 16 Gráfica de una señal de corriente

Cómo se puede apreciar en la ilustración el valor oscila entre un máximo y un mínimo. El problema que se encuentra en este proyecto es que las placas Arduino no son capaces de leer valores negativos y es por ello como se explicará más adelante que es necesario transformar dicha gráfica por otra que vaya desde el valor 0 hasta el máximo de cada placa Arduino, en nuestro caso hasta 3,3 V.

Capítulo 4. Planificación del Proyecto y Presupuesto Iniciales

4.1 Planificación Inicial

A continuación se mostrará el diagrama de Grant correspondiente a este proyecto en dos imágenes, ya que no entra toda la programación de tareas en una sola.

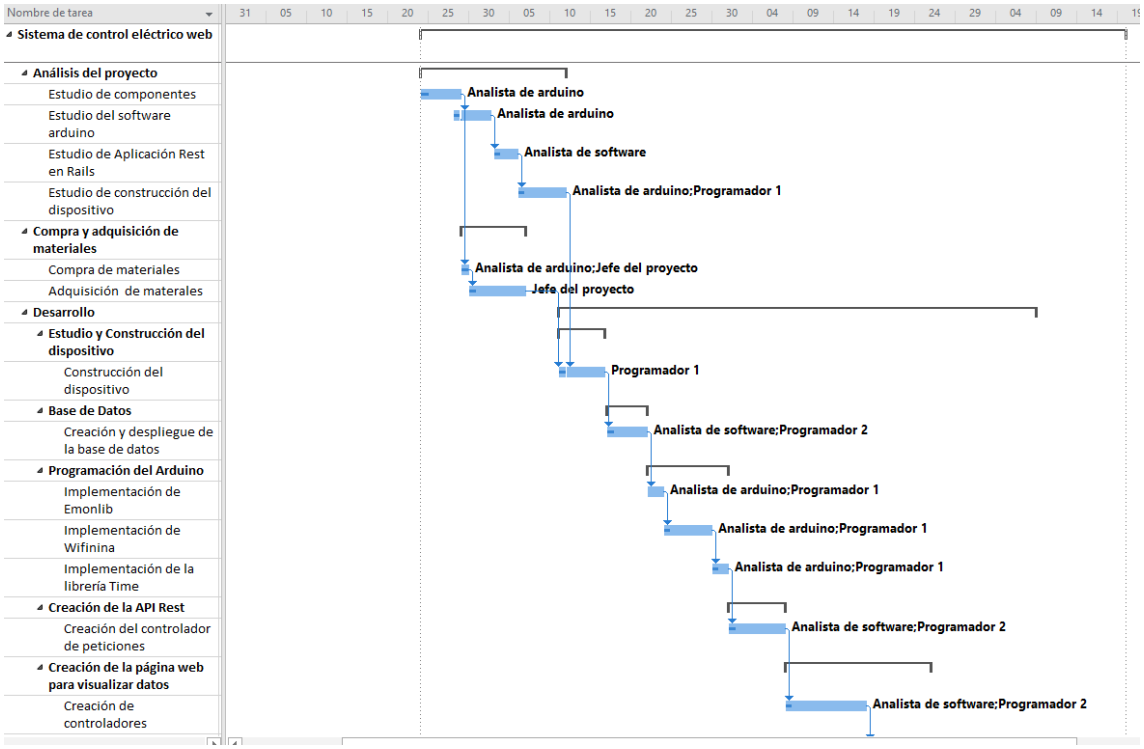


Ilustración 17 Planificación parte 1

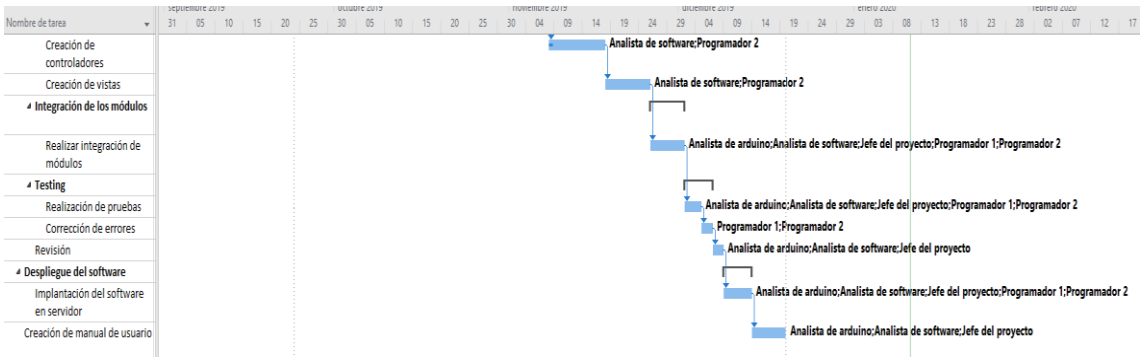


Ilustración 18 Planificación parte 2

El proyecto tiene una duración planeada de 87 días, siendo la jornada laboral de 4 horas y se trabajará todos los días de la semana. Por lo tanto la duración de este proyecto en horas es de 348 horas.

Para esta planificación se han creado 5 recursos, que en este caso son trabajadores, simulando un entorno real, para aproximar el proyecto a un entorno empresarial ficticio. Los trabajadores escogidos han sido un jefe de proyecto, un analista de software y otro de hardware y finalmente dos programadores cada uno especializado en realizar unas tareas específicas.

Aunque la planificación del proyecto sea con recursos ficticios, la duración se ha planificado para una sola persona, es por ello por lo que las tareas son secuenciales, es decir, hasta que no se acaba una no se empieza la otra, ya que el proyecto realmente solo ha sido realizado por una sola persona y por lo tanto no se puede llevar a cabo tareas en paralelo.

4.1.1 Resumen de tareas

Se pueden dividir las tareas en cuatro módulos:

1. Análisis: en este grupo de tareas se encuentra tanto el estudio previo necesario para poder realizar la tareas a posteriori de desarrollo y el estudio de los materiales necesarios. Y dentro de ese estudio se planifica y diseña tanto el software a desarrollar cómo el dispositivo a construir.
2. Compra y adquisición de materiales: tras el estudio de materiales y diseñando el dispositivo a crear se puede realizar una adquisición de materiales, la cual al tardar varios días en llegar es necesario incluirla, ya que debido a ello hay retrasos esperados en el proyecto.
3. Desarrollo: en esta parte se encuentran cuatro fases distintas:
   3.1. Construcción del dispositivo: en esta fase se construye el dispositivo.
   3.2. Creación de base de datos: se crea la base de datos según el diseño previamente creado y se despliega en el servidor.
   3.3. Programación del dispositivo Arduino: una vez creado el dispositivo es necesario añadir el software que es creado en esta fase y ver si funciona correctamente.
   3.4. Creación de la API REST: en esta fase se crea esta parte del proyecto y se prueba su funcionamiento con datos de prueba, sin llegar a implementarse junto con el dispositivo.
   3.5. Creación de la aplicación web: se desarrolla la parte final del proyecto, aunque no es la parte más importante del mismo y se realizan pruebas con datos de simulación.
   3.6. Integración de los módulos: se integran los cuatro módulos anteriores y se realiza una prueba de funcionamiento.
   3.7. Testing: finalmente se prueba todo en conjunto a través de las pruebas descritas y se corrigen los errores que hayan surgido.
4. Despliegue del software: en esta penúltima fase se despliega todo el proyecto en un servidor.
5. Manual de usuario: se crea un manual de usuario que podemos encontrar en este documento en el apartado 9. Manual del Sistema.

A continuación se muestra un listado con la duración por cada fase del proyecto:

Tarea	Horas dedicadas
Análisis	72
Adquisición de materiales	32
Desarrollo	236
Despliegue del software	20
Manual de usuario	24
Total	384

4.2 Presupuesto Inicial

Para el cálculo del proyecto se ha optado por realizar una simulación de una empresa con cinco trabajadores, entre los perfiles de empleados podemos encontrar: jefe de proyecto, Analista de Hardware, Analista de Software y dos programadores. Los sueldos de estos trabajadores varían entre los 45 €/hora a los 12 €/hora como se verá en los apartados siguientes.

4.2.1 Desarrollo de Presupuesto Detallado (Empresa)

En este apartado se mostrará los cotes que tiene la empresa en distintos apartados, tanto costes directos e indirectos, cómo los costes por cada fase en detalle.

4.2.1.1 Costes directos

4.2.1.1.1 Precios base

A continuación se mostrará una tabla con los precios de los trabajadores involucrados en el proyecto o que se han simulado, según una tarifa de una empresa ficticia. En esta tabla irá incluido el precio del trabajador cómo del equipo que necesitaría para realizar el proyecto.

Trabajador	Precio Hora
Jefe de proyecto	45,00 €
Analista de hardware	40,00 €
Analista de software	40,00 €
Programador 1	12,00 €
Programador 2	12,00 €

4.2.1.1.2 Análisis

En este apartado y con los costes por hora de cada uno de los empleados, junto con el número de horas trabajadas en este fase, tendremos el coste de la fase de análisis en este proyecto.

Análisis				
Código	Miembro	Hora(s)	Coste €/hora	Coste total €
EAW	Analista de Hardware	64	40,00 €	2.560,00 €
EAS	Analista de software	12	40,00 €	480,00 €
EJP	Jefe de proyecto	32	45,00 €	1.440,00 €
EP1	Programador 1	0	12,00 €	- €
EP2	Programador 2	0	12,00 €	- €
			Total	**4.480,00 €**

En esta fase ambos programadores no intervienen ya que se trata de una fase de análisis y diseño de la aplicación. El papel de los trabajadores será solo de desarrollo tanto de software cómo de hardware.

4.2.1.1.3 Desarrollo

Esta fase del proyecto es la más cara debido a su duración y además porque intervienen todos los empleados de la empresa en mayor o menor medida.

Desarrollo				
Código	Miembro	Hora(s)	Coste €/hora	Coste total €
EAW	Analista de Hardware	136	40,00 €	5.440,00 €
EAS	Analista de software	216	40,00 €	8.640,00 €
EJP	Jefe de proyecto	56	45,00 €	2.520,00 €
EP1	Programador 1	156	12,00 €	1.872,00 €
EP2	Programador 2	192	12,00 €	2.304,00 €
			Total	**20.776,00 €**

Esta fase del proyecto incluirá todas las tareas desde el desarrollo hasta la creación de manual de usuario del sistema.

4.2.1.1.4 Otros costes directos

En este apartado se incluirá todo el hardware necesario para la creación del dispositivo, cómo del servidor a desplegar la aplicación. Se ha simulado un pedido de 100 dispositivos y como servidor se utilizará el mismo que se utilizó durante el desarrollo real, el cual fue una Raspberry pi 3b, ya que se intenta acercar a un proyecto que un usuario medio pueda construir.

Código	Concepto	Precio por unidad	Unidades	Precio total
HW01	SCT-013	11,71 €	100	1.171,00 €
HW02	Arduino mkr-1010 wifi	27,90 €	100	2.790,00 €
HW03	Placa PCB	1,88 €	100	188,00 €
HW04	Condensador electrolítico de aluminio	0,08 €	100	8,00 €
HW05	Resistencias	0,01 €	1000	10,00 €
HW06	Cables de soldadura	- €	100	1,04 €
HW07	Raspberry (Servidor)	50,00 €	1	50,00 €
HW08	Regleta (personalizada)	4,00 €	100	400,00 €
HWT			**Total**	**4.618,04 €**

En el apartado de referencias se mostrarán todos los enlaces donde se pueden adquirir cada uno de los materiales previamente descritos.

4.2.1.2 Costes indirectos

Además de todos los costes previamente descritos, existen otros gastos derivados del desarrollo del proyecto. Ya que se ha optado por un desarrollo con una empresa ficticia se van a calcular unos costes indirectos, en base a datos reales de consumo los cuales se deben incluir ya que si no existirían perdidas en el proyecto.

Los costes indirectos han sido calculados para un año y luego se han dividido proporcionalmente al número de meses que ha abarcado el proyecto, en este caso ha sido de 4 meses, por lo tanto del total de cada uno de los gastos se ha divido entre 4. En la tabla que se muestra a continuación se muestra en detalle los gastos calculados.

Costes Indirectos				
Código	Material	Precio Anual	Meses utilizado	Precio Total
CI01	Licencia Microsoft Windows 10	276,00 €	4	69,00 €
CI02	Licencia Microsoft Project	172,80 €	4	43,20 €
CI03	Luz, agua y gas	1.200 €	4	300,00 €
CI04	Alquiler oficina	1.800,00 €	4	450,00 €
CI05	Internet y teléfono	480,00 €	4	120,00 €
CITO			**Total**	**982,20 €**

Estos costes indirectos se añadirán luego en el presupuesto del cliente de forma proporcional a las fases de análisis y desarrollo, de manera que el 30% del coste indirecto irá a la fase de análisis y el restante a la fase de desarrollo.

4.2.1.3 Resumen presupuesto interno

Finalmente se ha creado un presupuesto interno para la empresa donde se engloban todos los costes generados, descritos anteriormente pero de forma resumida. Se ha aplicado un beneficio del 23% del coste final para todos los apartados, esto incluye también el hardware ya que al dispositivo final hay que sacarle beneficio para así seguir adquiriendo nuevos.

Presupuesto interno		
Código	Concepto	Importe
PA	Análisis	4.480,00 €
PD	Desarrollo	20.776,00 €
PHDW	Hardware	4.618,04 €
PCI	Costes indirectos	982,20 €
TCPI	Total costes	30.856,24 €
BP	Beneficio (20%)	6.171,25 €
TCPIB	Total costes con Beneficio	37.027,49 €

4.2.2 Desarrollo de Presupuesto Simplificado (Cliente)

En este apartado se incluirán el presupuesto simplificado y el desarrollado por fase, que se le mostrará al cliente, en el cual se le mostrará el total con y sin IVA, se ha optado por un 21% de IVA sobre el total.

Presupuesto Cliente			
Código	Concepto	Total (€)	Total + IVA (€)
PEA	Análisis	5.729,59 €	6.932,80 €
PED	Desarrollo	25.756,25 €	31.165,06 €
PEH	Hardware	5.541,65 €	6.705,39 €
TO01	Total	37.027,49 €	44.803,26 €

El presupuesto final desarrollado del cliente se mostrará en el Capítulo 11. Presupuesto

Capítulo 5. Análisis

5.1 Definición del Sistema

5.1.1 Determinación del Alcance del Sistema

En este apartado se indica hasta donde se ha llegado en este proyecto, se puede hacer diferencia entre alcance a nivel de hardware y a nivel de software:

- Nivel de hardware:

Código	Concepto
ASH01	El sistema se montará sobre una Protoboard, para que no sea necesario el uso de soldaduras.
ASH02	El sistema usará una regleta personalizada, en la cual se realizará una modificación para dejar al descubierto el cable de fase, necesario para la medición del dispositivo.
ASH03	El servidor a utilizar se implementará en una Raspberry para que sea de fácil al usuario y puede configurarla con los parámetros necesarios.

- Nivel de software:

Código	Concepto
ASS01	El usuario deberá tener conocimientos básicos de informática, para modificar el archivo necesario de configuración de la red Wifi.
ASS02	El usuario deberá crear el servidor a través de la Raspberry, siguiendo las instrucciones del manual de usuario ofrecido.
ASS03	El código del dispositivo deberá ser modificado para añadir la dirección IP y puerto donde se encuentre el servidor montado.

5.2 Requisitos del Sistema

5.2.1 Requisitos funcionales

A continuación, se mostrarán los requisitos del sistema tanto del software del dispositivo, de la API REST y de la web de visualización de datos.

5.2.1.1 Hardware del dispositivo

Código	Nombre del Requisito	Descripción
RH1	Wifi	El dispositivo tendrá un módulo wifi (ESP32) para realizar las peticiones de envío y recepción de datos.
RH2	Sensor de corriente	El sensor de corriente deberá permanecer a la gama SCT-013, se puede cambiar por cualquiera de los modelos de su marca, pero esto lleva modificaciones en el coeficiente utilizado.
RH3	Regleta	La regleta deberá ser modificada para poder ajustar el sensor de corriente a esta. Mostrando su cable fásico.
RH4	Energía del dispositivo	El dispositivo deberá estar conectado a una fuente de energía de 5 V, ya sea mediante un cargador o una batería externa.
RH5	Tipo de dispositivo	El dispositivo debe ser Arduino, y debe contar con pines analógicos y un módulo Wifi.

5.2.1.2 Software del dispositivo

Código	Nombre del Requisito	Descripción
RSD1	Configuración wifi	La información wifi se incluirá en un archivo denominado “arduino_secrets.h” con la información de la red wifi.
RSD1.1	Datos del fichero	El fichero contendrá dos variables denominadas “SECRET_SSID” y “SECRET_PASS”.
RSD1.1.1	Completar los datos	La primera variable se completa con el SSID de la wifi que se vaya a conectar y el segundo con la contraseña de dicha wifi.
RSD2	Conexión al servidor	El sistema se conectará al servidor a través de su dirección IP o su DNS, además del puerto donde esté lanzada la API REST.
RSD3	Identificación del dispositivo	El dispositivo tendrá un identificador único, añadido por el usuario.
RSD4	Petición a la API REST	El sistema enviará una petición GET para obtener el coeficiente del dispositivo y la fecha actual.
RSD5	Configuración	El sistema se configurará con la fecha actual y el coeficiente recibido por la API REST.
RSD6	Lectura de corriente	El sistema será capaz de obtener una lectura lo más próxima a la del dispositivo a medir a través del sensor de corriente y tratando sus datos.
RSD7	Envío de información	El dispositivo será capaz de enviar la información a la API REST.
RSD7.1	Formato de información	Se enviará un JSON con la siguiente información, consumo: “consumo obtenido”, date_ard: “fecha actual”
RSD8	Envío fallido	En caso de que el envío sea fallido, el dispositivo almacenará las últimas NUM_REGISTROS

		lecturas de corriente.
RSD8.1	Número de peticiones guardadas	El valor de NUM_REGISTROS inicialmente será de 24. Pero será configurable por el usuario.
RSD9	Periodo de envío	El envío de información será cada TIEM_ENVIO.
RSD9.1	Configuración TIEM_ENVIO	El TIEM_ENVIO inicialmente será de una hora, pero puede ser configurado por el usuario.
RSD10	Periodo envío peticiones fallidas	El envío de las peticiones se realizará cada TIEM_FALLIDAS.
RSD10.1	Configuración TIEM_FALLIDAS	El valor de TIEM_FALLIDAS será inicialmente de treinta minutos. Puede ser configurado por el usuario.

5.2.1.3 API REST

Código	Nombre del Requisito	Descripción
RAR1	Petición de datos	El sistema deberá enviar al dispositivo el coeficiente correspondiente, junto con la fecha actual.
RAR1.1	Formato de petición	El sistema deberá enviar la información en formato JSON. Y de la siguiente manera: "{consumo: 5, now: {día: 25, mes: 12, year: 2019, hour: 13, minute: 13, second: 14}}
RAR2	Recepción de datos	El sistema recibirá dos datos, el consumo actual del sistema en el momento del envío junto con la fecha correspondiente, en el mismo formato que en RAR1.1
RAR3	Trato de datos	El sistema deberá tratar los datos para enviarlos correctamente a la base datos del sistema.
RAR4	Estado del tratado de datos	El sistema deberá devolver una respuesta dependiendo de si los datos han sido actualizados correctamente o no.
RAR4.1	Respuesta de la actualización	El sistema devolverá un JSON con "OK" y status 200, cuando la actualización sea correcta y un "No existe dispositivo" y status 404 cuando ocurra un fallo en su actualización.
RAR5	Añadir dispositivo al sistema	El sistema recibirá el identificador del dispositivo Arduino, si el dispositivo no existe en el sistema lo añadirá, en caso contrario no lo añadirá y devolverá un mensaje de error.

5.2.1.4 Web de visualización de datos

Código	Nombre del Requisito	Descripción
RWV1	Registro de usuario	El sistema permitirá registrar nuevos usuarios en el sistema.
RWV1.1	Información de registro	El sistema solicitará un email válido y una contraseña, junto con una confirmación de contraseña.
RWV1.1.1	Email	El email será válido si contiene el carácter "@".
RWV1.1.2	Contraseña	La contraseña tendrá un mínimo de caracteres que será de seis.
RWV2	Inicio de sesión	El sistema permitirá iniciar sesión a un usuario a través de su correo y su contraseña.
RWV3	Listado de dispositivos	El sistema mostrará el listado de dispositivos que tiene asociado el usuario actual.
RWV4	Detalles del dispositivo	El usuario podrá acceder a los detalles del dispositivo desde el listado.
RWV4.1	Información del dispositivo	El sistema mostrará al usuario el consumo actual, el coeficiente del dispositivo y tres gráficas.
RWV4.1.1	Gráficas de información	El sistema mostrará tres gráficas del consumo del dispositivo: la primera de los últimos veinticuatro registros, la siguiente de los últimos siete días y finalmente la última de los últimos doce meses.
RWV4.1.2	Calendario del consumo	El usuario podrá ver el consumo del dispositivo a través de un calendario, en el que se muestra el consumo total diario.
RWV4.2	Comprar usuario del dispositivo	El sistema comprobará que el usuario que solicita la información del dispositivo es el mismo que el usuario dueño de éste.
RWV4.2.1	No se corresponde con el usuario	Si el sistema detecta que el usuario que solicita la información de un dispositivo no es su dueño, el sistema mandará al usuario a su pantalla de inicio y mostrará un mensaje.
RWV5	Añadir dispositivo	El sistema permitirá añadir un nuevo dispositivo para el usuario.
RWV5.1	Formulario añadir dispositivo	El usuario deberá introducir el identificador del dispositivo, el nombre del dispositivo, los vatios a medir y el coeficiente teórico de su dispositivo.
RWV5.2	Dispositivo en Base de datos	El dispositivo a añadir deberá encontrarse previamente registrado en la base de datos.
RWV6	Desvincular dispositivo	El usuario podrá desvincular un dispositivo que tenga en posesión.
RWV6.1	Eliminación de información	Tras confirmar la desvinculación del dispositivo, éste pasará a tener los valores por defecto, pero sus registros de consumo guardados no serán eliminados.
RWV7	Edición del dispositivo	El usuario podrá modificar sus dispositivos.
RWV7.1	Formulario de edición del	El sistema solicitará la nueva etiqueta del

	dispositivo	dispositivo, los vatios del dispositivo y el coeficiente asociado del dispositivo.

5.2.2 Requisitos no funcionales

Código	Nombre del Requisito	Descripción
RNF1	Sistema operativo	La API REST y la Web de visualización podrán ser lanzados el cualquier sistema operativo.
RNF1.1	Instalación de Rails	Deberá ser necesario instalar Ruby on Rails para poder ser lanzadas ambas aplicaciones.

5.2.3 Identificación de Actores del Sistema

Para este caso podemos tener dos actores en el sistema, que serán los siguientes:

- Usuario de instalación y mantenimiento de dispositivos: todos los usuarios tendrán disponible el código del dispositivo como de la parte de API REST y Web para realizar el proyecto, junto con el manual de usuario para su instalación.

- Usuario del sistema web de visualización de datos: este tipo de usuario podrá, mediante el manual de usuario, ver cómo se añaden, desvinculan o ver información de los dispositivos que tenga asociados. Se puede entender este usuario separado del primero, ya que este último puede ser un usuario final el cual ha solicitado la instalación de este proyecto.

5.3 Análisis de Casos de Uso

5.3.1 Diagrama de contexto

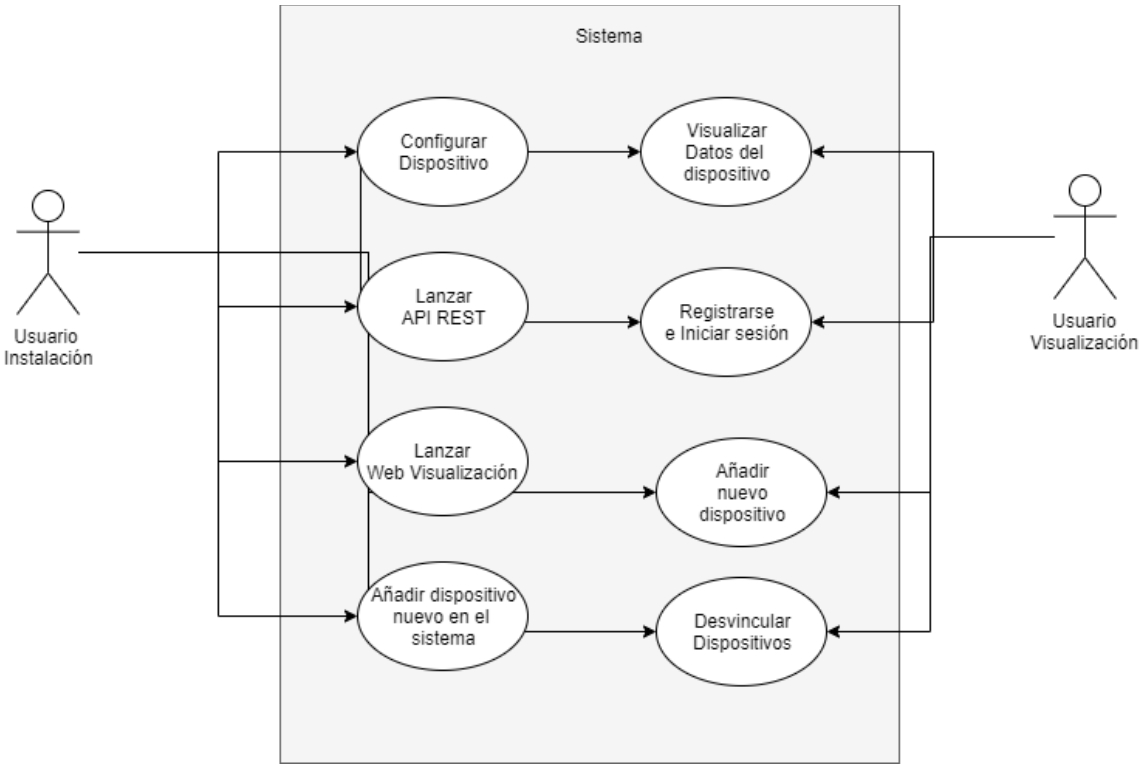


Ilustración 19 Diagrama de contexto

5.3.2 Casos de uso

5.3.2.1 Configuración del Arduino

Caso de Uso	Configuración del Arduino
Autor	Juan Granda Molaguero
Tipo	Primario
Requisito que cubre	RSD1, RSD2, RSD3, RSD4, RSD5, RSD6 y RSD7
Propósito	
Describe cómo un usuario de configuración puede configurar correctamente el Arduino	
Resumen de alto nivel	
El usuario se descargará el código necesario, conectará el Arduino al ordenador y complementará los datos necesarios para su correcto funcionamiento.	
Descripción detallada	
Flujo de acciones	
Actor	Sistema
1. El usuario descargará los archivos de la carpeta cew-arduino.	
2. El usuario conectará la placa por medio de	

USB al ordenador.	
3. El usuario completará los datos necesarios que se indican.	
4. El usuario cargará el software en la placa.	
Otros datos	
Frecuencia esperada	Alta, es necesario este proceso para que el proyecto arranque.
Relevancia en el sistema	Alta
Estabilidad	Alta
Urgencia	Alta

5.3.2.2 Envío de información a la API REST

Caso de Uso	Envío de información a la API REST
Autor	Juan Granda Molaguero
Tipo	Primario
Requisito que cubre	RAR2,RAR3,RAR4
Propósito	
Describir los procesos que ocurren cuando la API REST recibe información de un dispositivo.	
Resumen de alto nivel	
El dispositivo enviará información a la API REST, si el dispositivo existe en el sistema la información será tratada y se enviará un mensaje al dispositivo si la información ha sido tratada.	
Descripción detallada	
Flujo de acciones	
Actor	Sistema
1- El dispositivo enviará la información del consumo.	2- El sistema comprobará que el dispositivo existe dentro del sistema.
	3- Si el dispositivo existe y está vinculado a un usuario, añadirá un registro nuevo a los consumos hora y aumentará el consumo tanto diario como mensual del dispositivo.
	4- Si todos los datos han sido introducidos y actualizados, la API REST enviará un mensaje al dispositivo, OK y un estado 200, indicando que todo ha ido correctamente.
	5- De no existir el dispositivo u ocurrir un fallo durante la actualización y registro de datos, la API REST enviará un mensaje y un estado 404 al dispositivo.
Otros datos	
Frecuencia esperada	Continua, ya que los usuario irán registrando nuevos dispositivos a medida que adquieren más.
Relevancia en el sistema	Alta
Estabilidad	Alta
Urgencia	Alta

5.3.2.3 Añadir dispositivo nuevo al sistema

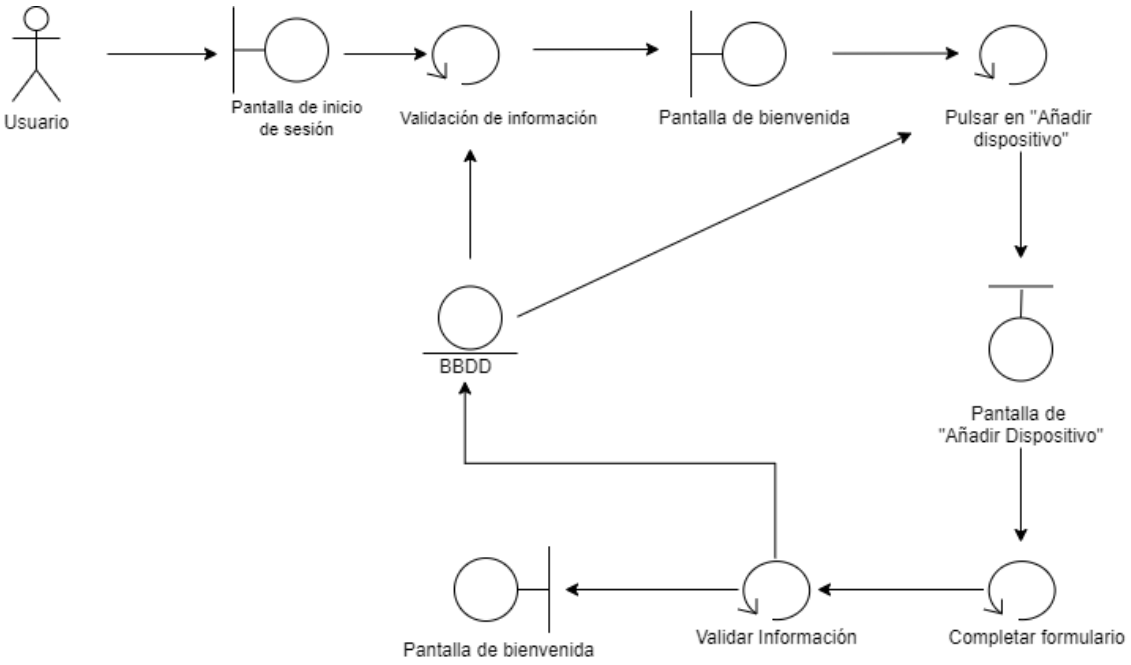


Ilustración 20 Diagrama robustez: añadir dispositivo

Caso de Uso	Añadir dispositivo nuevo al sistema
Autor	Juan Granda Molaguero
Tipo	Primario
Requisito que cubre	RW5
Propósito	
Describir cómo un usuario añade un nuevo dispositivo al sistema.	
Resumen de alto nivel	
El usuario añadirá un nuevo dispositivo al sistema, y así obtener mediciones y ver dichas mediciones a través del sistema web.	
Descripción detallada	
Flujo de acciones	
Actor	Sistema
1. El usuario accederá a su sesión en el sistema web.	2. El sistema comprueba que el usuario está correctamente registrado en el sistema.
4. El usuario accede a la sección “Añadir un nuevo dispositivo”, rellena el formulario y pincha en el botón “Añadir”.	3. El sistema le muestra sus dispositivos actuales.
	5. El sistema comprueba que el dispositivo existe en el sistema y no le pertenece a otro usuario y si es correcto lo añade al conjunto de dispositivos del usuario
Otros datos	
Frecuencia esperada	Continua, ya que los usuario irán registrando nuevos dispositivos a medida que adquieren más.
Relevancia en el sistema	Alta
Estabilidad	Alta

Urgencia	Alta

5.3.2.4 Registro en el sistema de visualización de datos

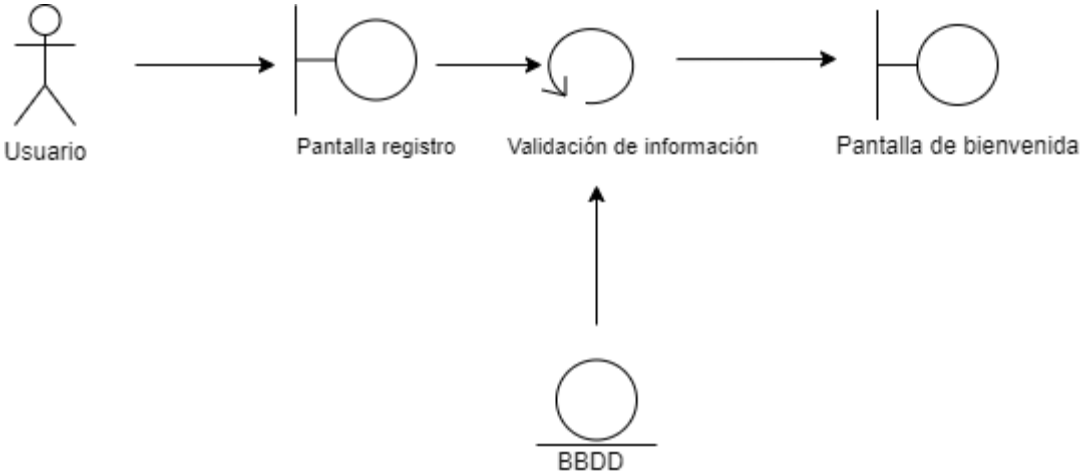


Ilustración 21 Diagrama robustez: Registro en el sistema

Caso de Uso	Registro de usuario en el sistema
Autor	Juan Granda Molaguero
Tipo	Primario
Requisito que cubre	RWV1, RWV1.1, RWV1.1.1, RWV1.1.2
Propósito	
El usuario se registrará en el sistema para poder visualizar los datos de sus dispositivos	
Resumen de alto nivel	
El usuario accederá a la aplicación de visualización de datos para registrarse y así poder acceder a su panel.	
Descripción detallada	
Flujo de acciones	
Actor	Sistema
1. El usuario accede a la aplicación web y pincha en el enlace “Registrarse”.	3. El sistema comprobará que los datos introducidos son correctos.
2. A continuación el usuario completará la información que se le solicita en el formulario.	4. Si los datos introducidos son correctos el sistema registrará al usuario. Y le enviará a su página de inicio
	5. Si los datos son erróneos se le mostrará un mensaje al usuario y se le solicitará de nuevos los datos del formulario.
Otros datos	
Frecuencia esperada	Alta, debido a que es necesario para poder visualizar los datos obtenidos.
Relevancia en el sistema	Media
Estabilidad	Alta
Urgencia	Media

5.3.2.5 Inicio de sesión en el sistema

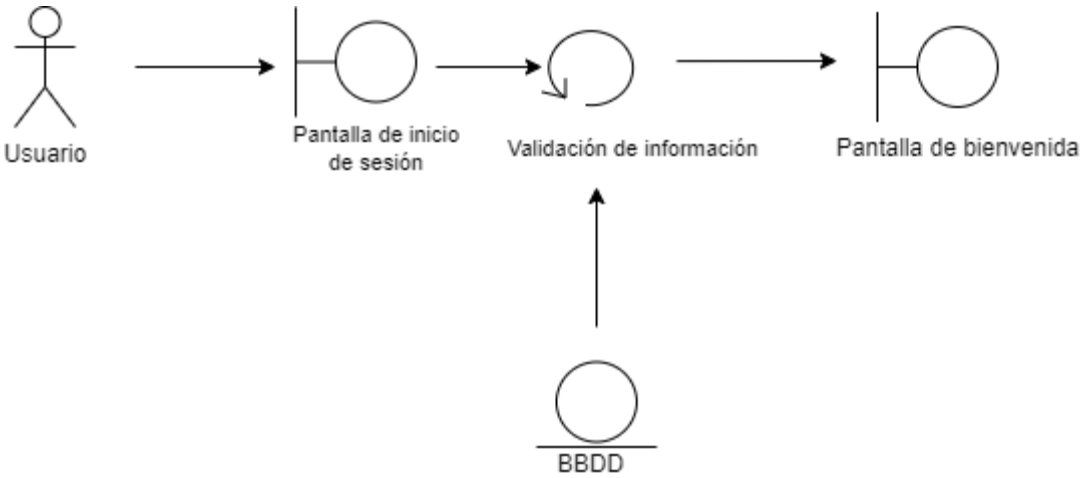


Ilustración 22 Diagrama de robustez: Inicio de sesión

Caso de Uso	Inicio de sesión en el sistema
Autor	Juan Granda Molaguero
Tipo	Primario
Requisito que cubre	RWV2, RWV3
Propósito	
Describir cómo un usuario accede a su sesión y la respuesta del sistema.	
Resumen de alto nivel	
El usuario completará los datos para acceder a su sesión y si son correctos el sistema le mostrará el listado de sus dispositivos asociados	
Descripción detallada	
Flujo de acciones	
Actor	Sistema
1. El usuario completará el formulario con su correo y contraseña.	2. El sistema comprobará que los datos introducidos son correctos.
	3. Si los datos son correctos el sistema le mostrará al usuario su listado de dispositivos asociados.
	4. Si los datos son erróneos el sistema le mostrará al usuario un mensaje y solicitará de nuevo la información necesaria.
Otros datos	
Frecuencia esperada	Continua, los usuarios deberán pasar por este proceso cada vez que quieran ver sus dispositivos y sus consumos.
Relevancia en el sistema	Alta
Estabilidad	Alta
Urgencia	Alta

5.3.2.6 Acceder a un dispositivo

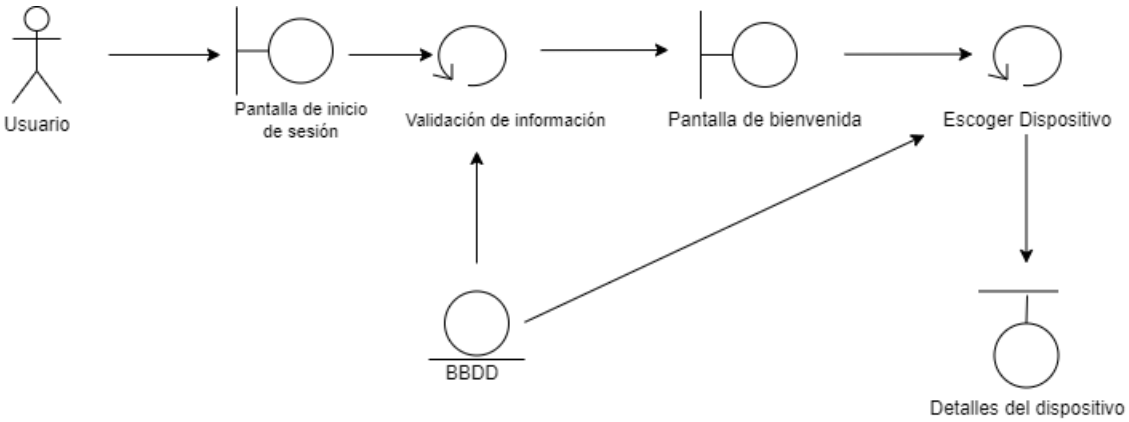


Ilustración 23 Diagrama de robustez: acceder a un dispositivo

Caso de Uso	Acceder a un dispositivo
Autor	Juan Granda Molaguero
Tipo	Primario
Requisito que cubre	RWV4, RWV4.1, RWV4.2
Propósito	
Describir qué sucede cuando un usuario accede a los detalles de un usuario	
Resumen de alto nivel	
El usuario intentará accede por medio de la URL a un dispositivo y el sistema comprobará si el usuario es dueño del dispositivo para mostrarle la información de éste.	
Descripción detallada	
Flujo de acciones	
Actor	Sistema
1. El usuario accederá al sistema web e indicará mediante URL al dispositivo que quiere acceder	2. El sistema comprobará que el dispositivo existe primero
	3. Si el dispositivo existe, comprobará que el usuario que solicita la información se corresponde con el usuario del dispositivo.
	4. Si el usuario es correcto, el sistema mostrará la información del dispositivo al usuario.
	5. Si el usuario no es correcto, el sistema devolverá al usuario a la página de inicio y le mostrará un mensaje de error.
Otros datos	
Frecuencia esperada	Continua, los usuarios deberán pasar por este proceso cada vez que quieran ver la información de consumo.
Relevancia en el sistema	Alta
Estabilidad	Alta
Urgencia	Alta

5.3.2.7 Editar un dispositivo

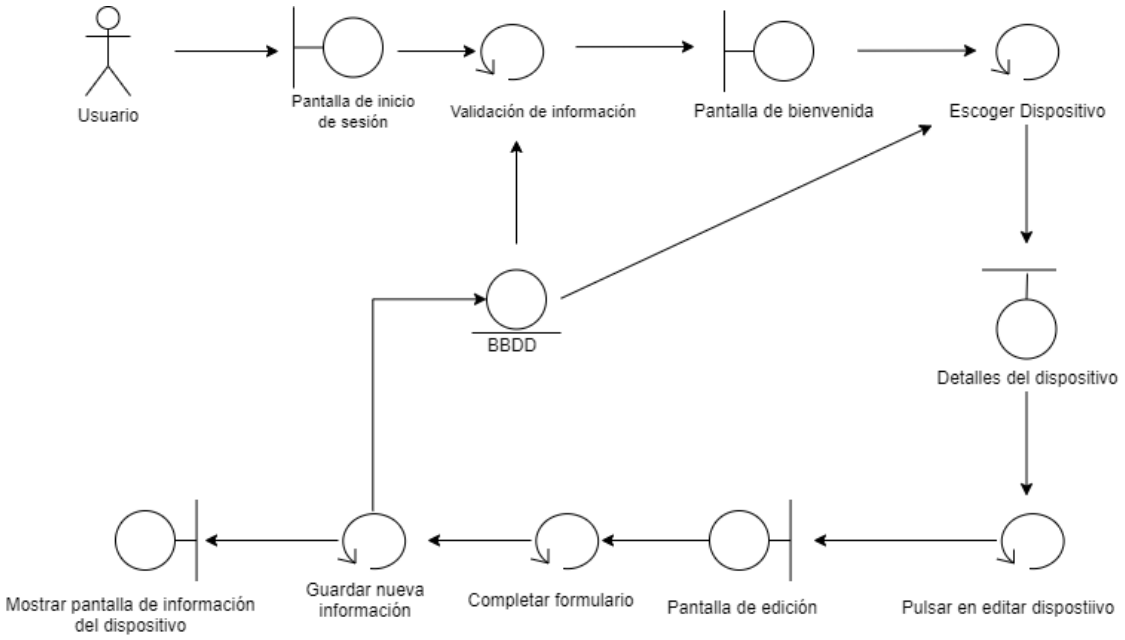


Ilustración 24 Diagrama robustez: Editar dispositivo

Caso de Uso	Editar un dispositivo
Autor	Juan Granda Molaguero
Tipo	Primario
Requisito que cubre	RWV7, RWV7.1
Propósito	
Describir cómo un usuario puede editar datos de un dispositivo asociado.	
Resumen de alto nivel	
El usuario deberá iniciar sesión con su cuenta y luego accederá a un dispositivo que tenga asociado, desde ahí podrá completar un formulario con los nuevos datos.	
Descripción detallada	
Flujo de acciones	
Actor	Sistema
1. El usuario iniciará sesión en el sistema.	2. El sistema comprobará que los datos introducidos son correctos.
4. El usuario accederá a unos de sus dispositivos.	3. Si los datos son correctos el sistema le mostrará al usuario su listado de dispositivos asociados.
5. El usuario pulsará en el botón “Editar”.	6. El sistema cargará el formulario de edición de un dispositivo.
7. El usuario completará los campos que desee editar y pulsará el botón editar.	8. El sistema comprobará que el dispositivo se encuentra en el sistema y si es correcto editará su información con los datos introducidos en el sistema.
Otros datos	
Frecuencia esperada	Baja, una vez que un usuario añada un dispositivo será poco frecuente que edite sus datos.
Relevancia en el sistema	Media

Estabilidad	Alta
Urgencia	Media.

5.3.2.8 Desvincular un dispositivo

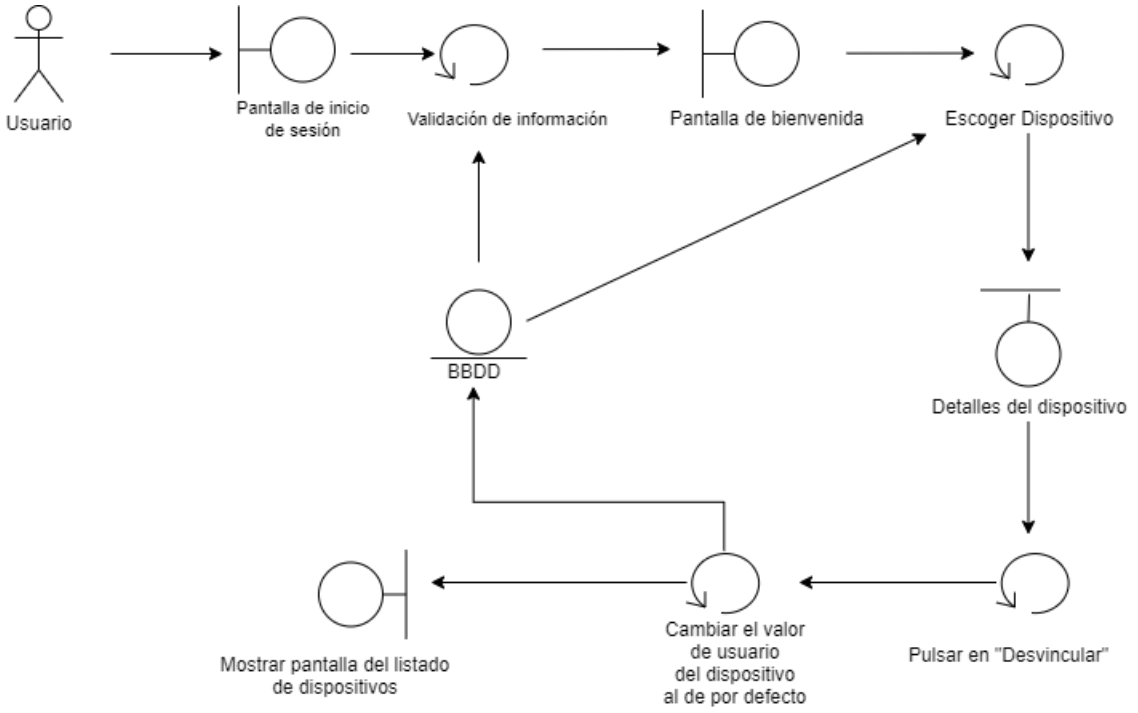


Ilustración 25 Diagrama de robustez: Desvincular dispositivo

Caso de Uso	Desvincular un dispositivo
Autor	Juan Granda Molaguero
Tipo	Primario
Requisito que cubre	RWV6, RWV6.1
Propósito	
Describir cómo un usuario desvincula uno de sus dispositivos asociados	
Resumen de alto nivel	
El usuario accederá al sistema, a continuación escogerá uno de sus dispositivos y al acceder a los detalles del éste podrá desvincular el dispositivo de su cuenta.	
Descripción detallada	
Flujo de acciones	
Actor	Sistema
1. El usuario completará el formulario con su correo y contraseña.	2. El sistema comprobará que los datos introducidos son correctos.
4. El usuario accederá a los detalles del dispositivo a desvincular y pulsará el botón “Desvincular”	3. Si los datos son correctos el sistema le mostrará al usuario su listado de dispositivos asociados.
	5. El sistema cambiará el usuario a uno por defecto, pondrá vacío el valor de la etiqueta, el de vatios del dispositivo, el coeficiente y el de consumo.
Otros datos	

Frecuencia esperada	Baja, ya que una vez añadido el dispositivo el usuario puede optar por editar el dispositivo y usarlo para medir otros aparatos eléctricos.
Relevancia en el sistema	Media
Estabilidad	Alta
Urgencia	Baja

5.3.2.9 Enviar información cuando el servidor esta caído

Caso de Uso	Envío de información del dispositivo al servidor con este último no operativo
Autor	Juan Granda Molaguero
Tipo	Primario
Requisito que cubre	RSD7, RSD8, RSD9, RSD10
Propósito	
Describir cómo un usuario desvincula uno de sus dispositivos asociados	
Resumen de alto nivel	
El usuario accederá al sistema, a continuación escogerá uno de sus dispositivos y al acceder a los detalles del éste podrá desvincular el dispositivo de su cuenta.	
Descripción detallada	
Flujo de acciones	
Actor	Sistema
	1. El dispositivo envía la información al servidor al cumplirse el tiempo configurado.
	2. El dispositivo comprueba que la conexión con el servidor no es posible y almacena la información.
	3. El dispositivo al cumplirse el tiempo de envío de información perdida, envía toda la información almacenada al sistema.
	4. Si el servidor se encuentra ya disponible irá recibiendo la información pendiente, de no ser así se volverá al paso 1.
Otros datos	
Frecuencia esperada	Baja, ya que se espera que las caídas del servidor sean las menores posibles.
Relevancia en el sistema	Alta
Estabilidad	Alta
Urgencia	Alta

5.3.2.10 Añadir nuevo dispositivo al sistema

Caso de Uso	Añadir nuevo dispositivo al sistema
Autor	Juan Granda Molaguero
Tipo	Primario
Requisito que cubre	RAR5
Propósito	

Describir qué ocurre cuando se añade un nuevo dispositivo al sistema	
Resumen de alto nivel	
Un usuario intentará añadir un nuevo dispositivo al sistema llamando a la API REST por medio de una interfaz o sistema externo, el sistema comprobará si puede añadir o no este nuevo dispositivo	
Descripción detallada	
Flujo de acciones	
Actor	Sistema
1. El usuario enviará una petición a la API REST con el identificador asignado al dispositivo Arduino	2. El sistema comprobará si el dispositivo existe ya en la base de datos.
	3. En caso de que no exista el dispositivo en la base de datos, el sistema creará un nuevo registro, y le pondrá como usuario el administrador.
	4. En caso de ya exista un registro con el mismo identificador en el sistema, el sistema enviará un mensaje indicando que no ha sido posible añadir el dispositivo.
	4. Si el servidor se encuentra ya disponible irá recibiendo la información pendiente, de no ser así se volverá al paso 1.
Otros datos	
Frecuencia esperada	Alta, se espera cada vez que salga una nueva remesa de dispositivos para el usuario.
Relevancia en el sistema	Alta
Estabilidad	Alta
Urgencia	Alta

5.4 Análisis de Interfaces de Usuario

5.4.1 Descripción de la Interfaz

En este proyecto solo una de las tres partes que lo componen tendrá una interfaz de usuario con la que interactuar. A continuación se mostrarán imágenes de las distintas pantallas del sistema y a que parte corresponde, dando una breve descripción de lo que se puede hacer en cada una de ellas.

La primera imagen es la de inicio de sesión y es un modelo de cómo será finalmente en el proyecto.

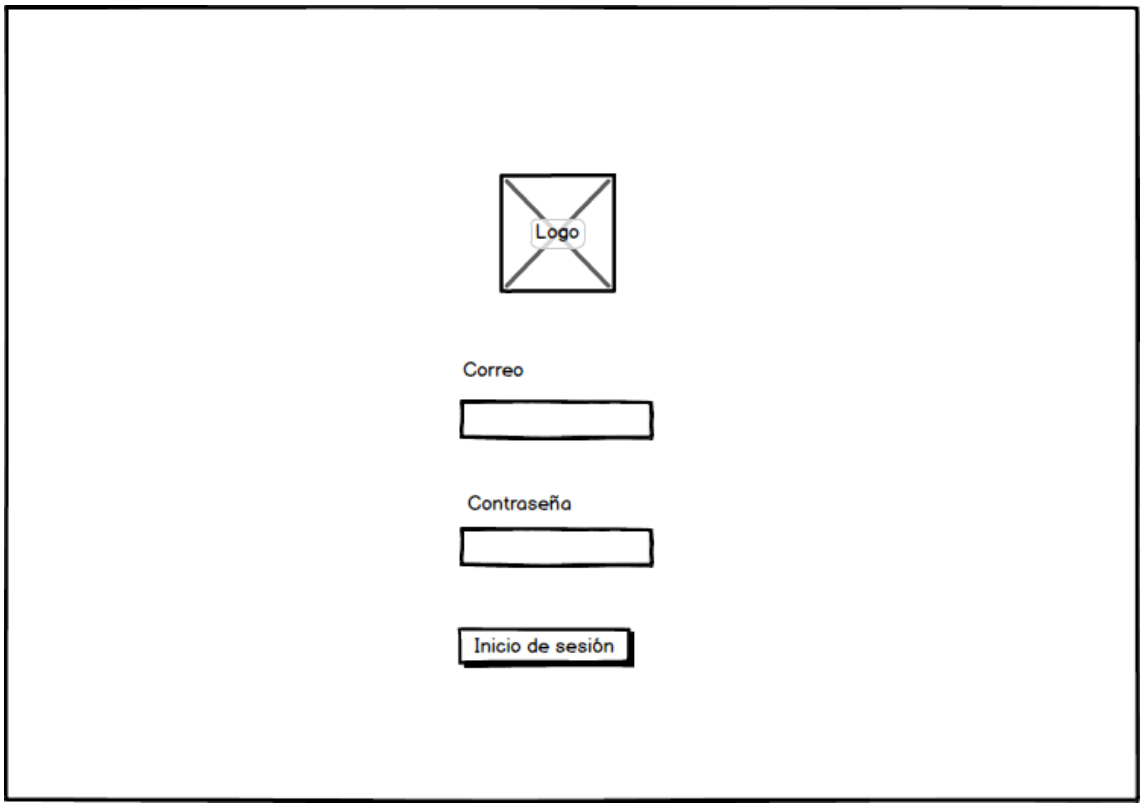


Ilustración 26 Interfaz de Inicio de sesión

Como se puede apreciar es un sistema de inicio de sesión básico, en el que es necesario introducir el correo electrónico con el que se haya registrado y la contraseña.

A continuación se mostrarán las ventanas ya dentro de la aplicación de visionado en sí, comenzando por las zonas de trabajo y luego las ventanas que se han diseñado.

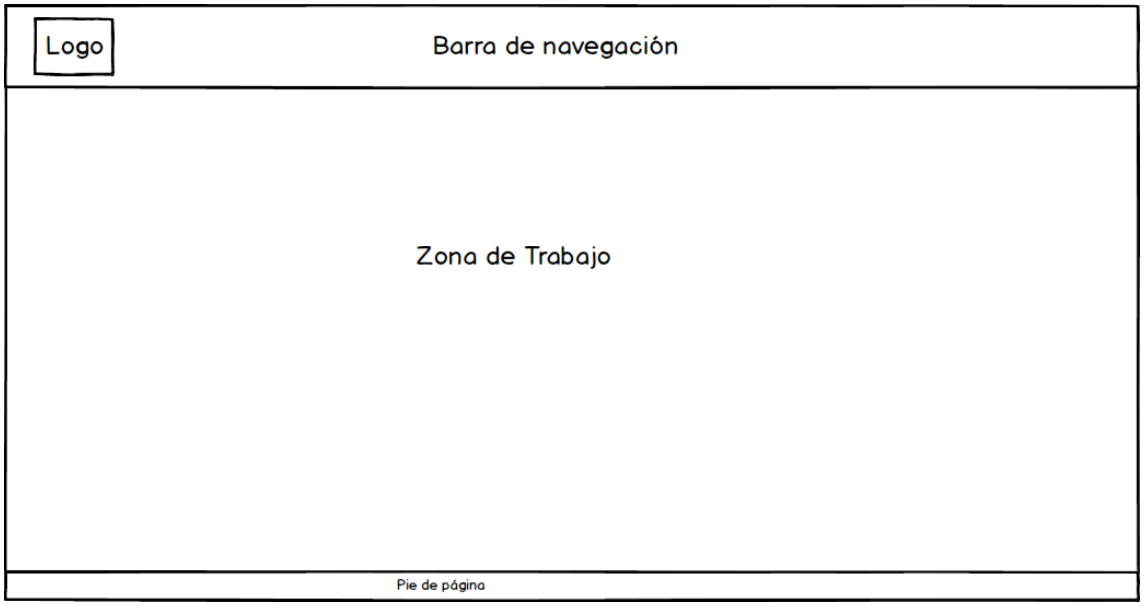


Ilustración 27 Interfaz zonas de trabajo

Podemos dividir la aplicación en dos zonas de trabajo una que es la barra de navegación donde se incluirán enlaces a ciertas partes de la aplicación, así como el botón para desconectarse y luego la zona de trabajo dónde se encuentran las ventanas que se mostrarán a continuación.

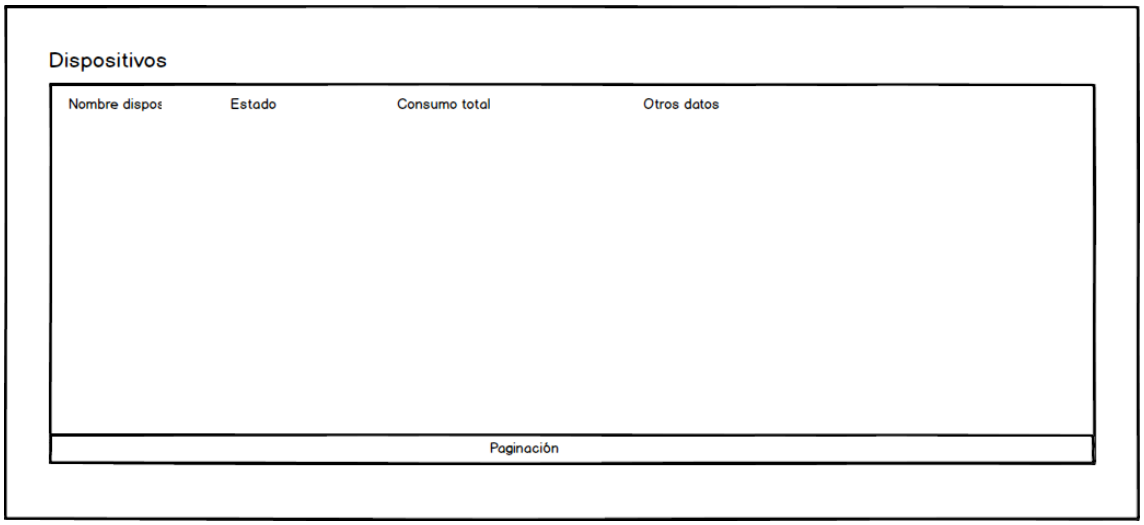


Ilustración 28 Prototipo listado de dispositivos

Esta primera venta mostrará el listado de dispositivos que tiene el usuario asociados y se mostrará información de relevancia para el usuario. Pulsando en uno de los dispositivos accederemos a la siguiente ventana de detalles del usuario.

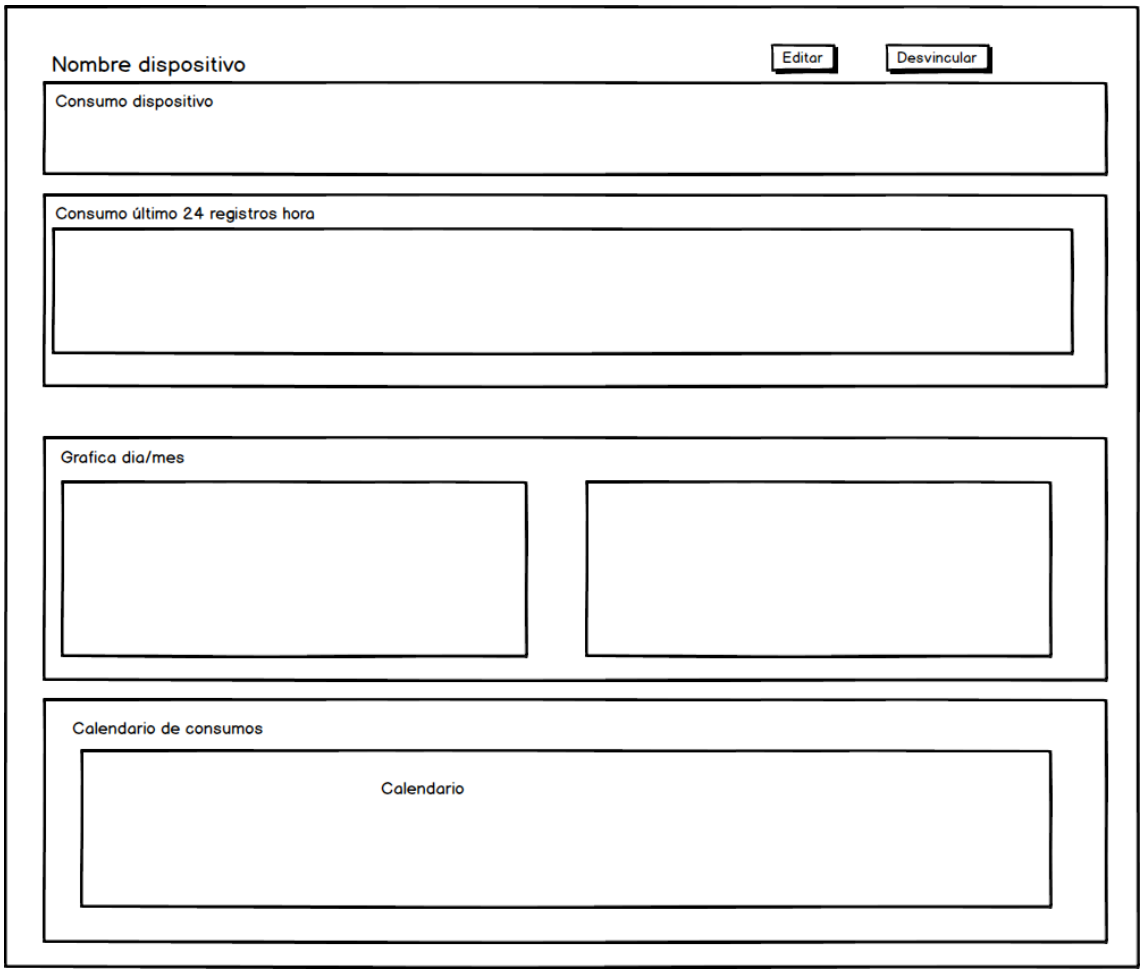


Ilustración 29 Prototipo vista de información de dispositivo

En esta ventana podemos observar que existen 4 zonas diferenciadas, la primera con datos importantes del dispositivo y con dos botones para editar o desvincular. A continuación, una gráfica con los últimos 24 registros por hora y tras ello dos gráficas con los registros de la última semana y de los últimos meses. Finalmente un calendario con los registros de los consumos por día.

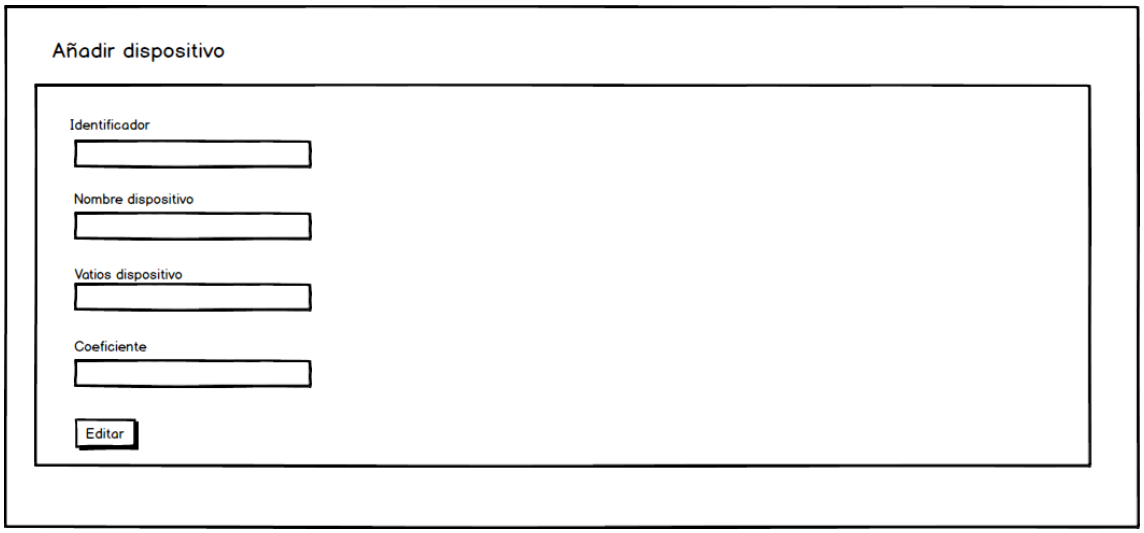


Ilustración 30 Prototipo formulario de edición y añadir dispositivo

Finalmente tendremos esta ventana con la que un usuario podrá añadir un nuevo dispositivo al sistema, esta ventana será igual cuando un usuario opte por editar los datos de uno de sus dispositivos.

5.4.2 Descripción del Comportamiento de la Interfaz

En este apartado se describirá tantos los mensajes de error que se le mostrarán al usuario y el tipo de valores que son necesarios en los distintos formularios del sistema.

5.4.3 Diagrama de Navegabilidad

A continuación se mostrará un diagrama de cómo se va de una ventana a otra a través de ciertas acciones realizadas por el usuario.

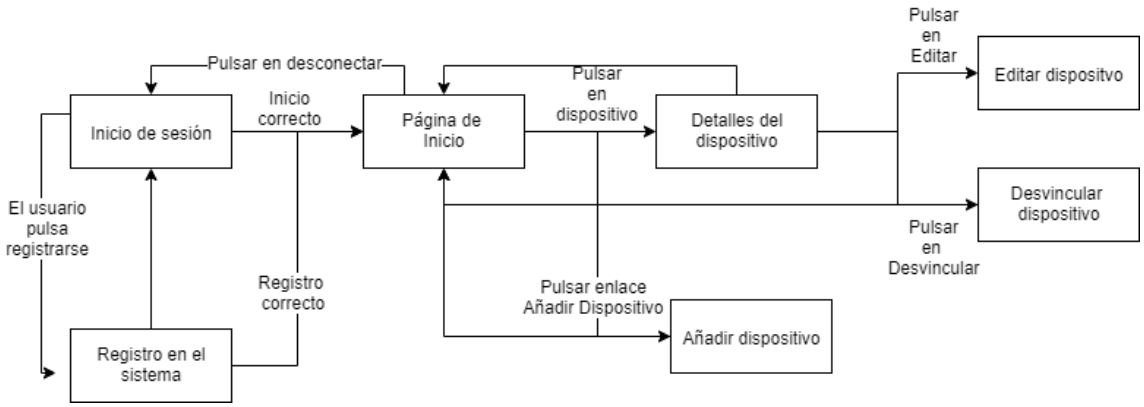


Ilustración 31 Diagrama de navegabilidad

5.4.4 Diagrama de construcción del dispositivo

A continuación se mostrará un boceto de cómo sería de forma abstracta el concepto del proyecto final.

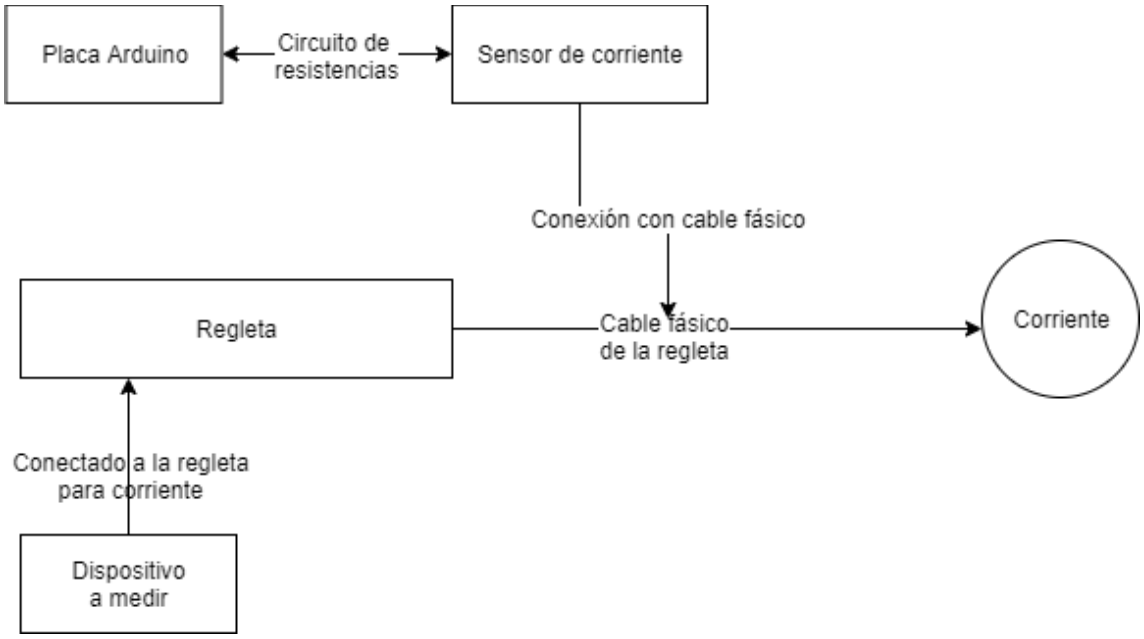


Ilustración 32 Boceto de construcción del dispositivo

Como se puede apreciar en el boceto el sistema consta de cuatro partes:

- Placa Arduino: que recibirá la señal del sensor de corriente y la transformará en datos válidos y la enviará al servidor. Dicha placa, podrá ir conectada a una batería externa de 5V o a un cable de corriente de 5V.
- Sensor de corriente: Irá conectado con la placa Arduino, y la pinza irá sobre el cable fásico de la regleta para medir el consumo de esta.
- Regleta: servirá para conectar el dispositivo, el cual queremos medir su consumo, de manera que no sea necesario manipular este, si no que solo es necesario manipular la regleta para que se puedan realizar las mediciones.
- Dispositivo a medir: cualquier dispositivo que se pueda conectar a la Regleta, será posible saber el consumo que tiene a lo largo del tiempo.

Capítulo 6. Diseño del Sistema

6.1 Arquitectura del Sistema

6.1.1 Diagramas de Componentes

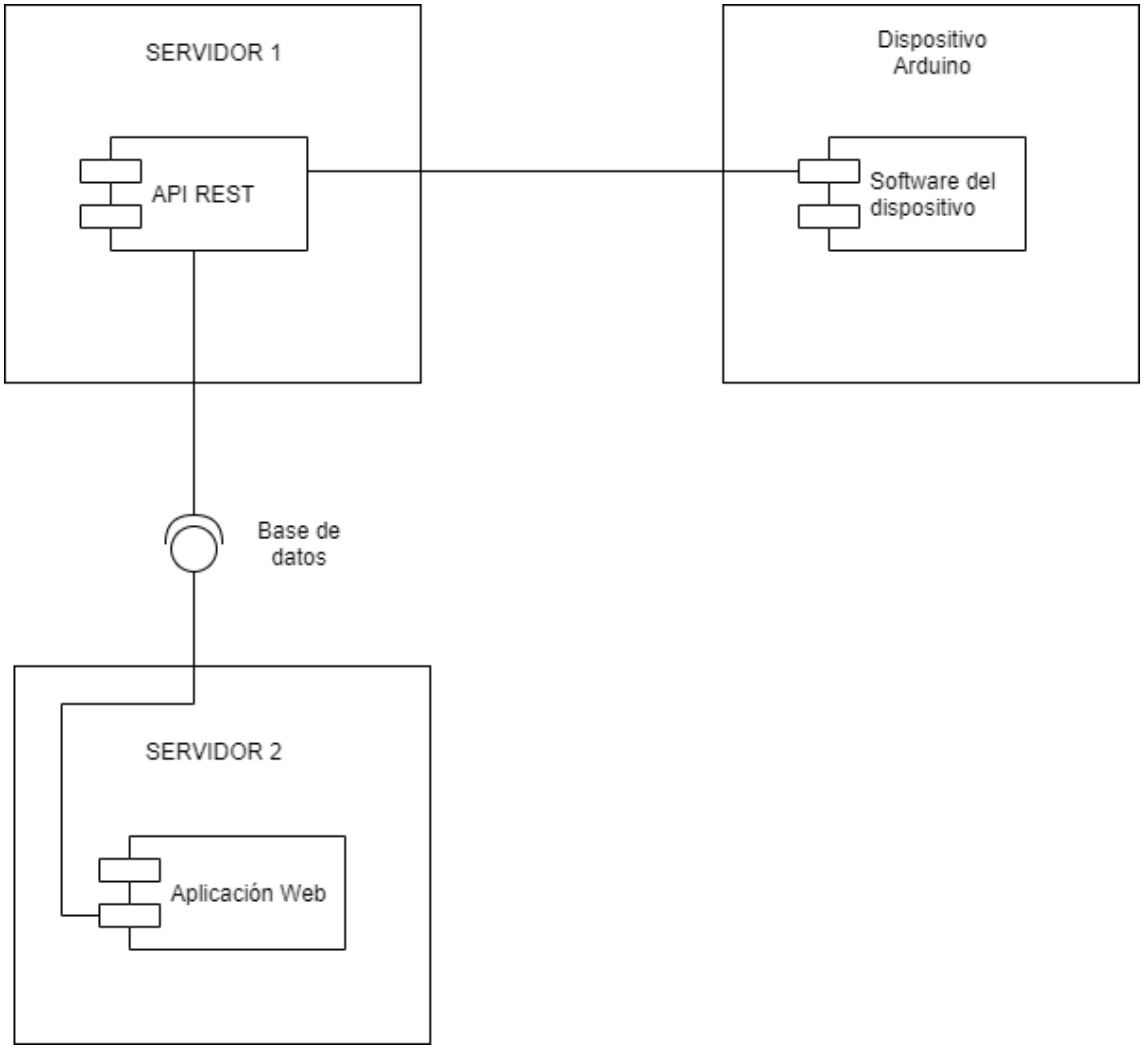


Ilustración 33 Diagrama de Componentes

Tal y como se aprecia en la ilustración anterior, se puede apreciar que existen tres componentes en este proyecto, por un lado tenemos el dispositivo y su software, el cual envía la información al servidor 1, el cual tiene desplegado la API REST, que se encarga de tratar los datos y enviarlos a la base de datos, la cual será consumida por el servidor 2, dónde se encuentra la aplicación web lanzada, que hará uso de los datos para mostrárselos al usuario final.

6.1.2 Diagrama de Despliegue

En esta sección se mostrará un esquema de alto nivel, que muestra los elementos que son necesarios para que un usuario pueda visualizar los datos de un dispositivo eléctrico.

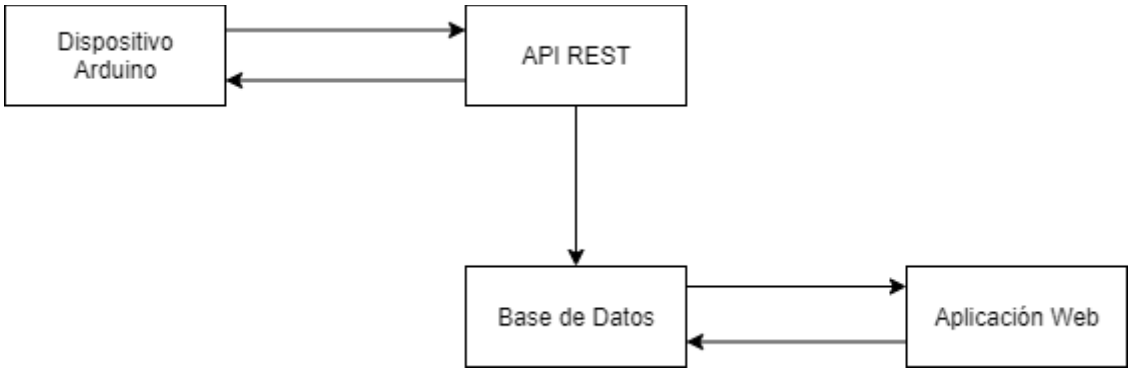


Ilustración 34 Diagrama de Despliegue

A continuación se describirá con mayor detalle cada uno de los elementos previamente vistos, indicando que es cada uno de ellos y su funcionalidad en el sistema global.

6.1.2.1 Dispositivo Arduino

Este elemento es el núcleo del sistema, ya que se encarga de recibir la corriente del dispositivo que estamos midiendo y tratando esos datos, enviarlos a la API REST. Este elemento es el más importante, ya que podríamos desprendernos de los demás elemento del sistema y añadir un sistema de visionado por una pantalla, y así el usuario podría ver el consumo en tiempo real del electrodoméstico a medir.

Pero debido a que se quiere mostrar un registro de los consumos del elemento que estamos midiendo, la información que recibe el dispositivo Arduino es enviado a la API REST, como se indica en los requisitos del sistema. Además para obtener los valores del coeficiente y la fecha actual. Es por ello por lo que mediante peticiones GET y POST ambos elementos se comunican entre ellos.

Finalmente si la conexión entre ellos no se produce el dispositivo Arduino guardará la información hasta ser posible el envío de ésta.

6.1.2.2 API REST

Esta aplicación se encarga de recibir los datos capturados por el Arduino y tratarlos, de manera que envía la información final a la base de datos en cada una de las tablas correspondientes. Además como se ha mencionado anteriormente, la API envía información necesaria a la placa Arduino para que pueda llevar a cabo de forma correcta la medición del consumo.

6.1.2.3 Base de datos

La base de datos servirá de puente entre la API REST y la aplicación web de visualización de datos, ya que por medio de ella y como se ha dicho en el apartado anterior, la primera envía la información obtenida por el dispositivo Arduino y por otra parte, la aplicación web coge esa información y se la muestra al usuario. Es por tanto la base de datos el medio por el cual un usuario final es capaz de ver el consumo de uno de sus dispositivos.

6.1.2.4 Aplicación web de visualización

Esta última parte del sistema se encarga de mostrar al usuario la información obtenida por el Arduino, a su vez se encarga de registrar nuevos usuarios en el sistema y de añadir nuevos dispositivos en el mismo.

Es por ello por lo que el sistema tiene dos funciones principales, conectarse con la base de datos para obtener la información de los dispositivos de un usuario, y por otro lado la gestión de los usuarios del sistema.

6.2 Diagrama de secuencia

En este apartado se mostrarán las secuencias más importante del sistema global, tanto de distintos sistemas entre sí, como dentro de un mismo sistema cómo interactúan ciertas partes con otras.

6.2.1 Diagrama de secuencia “Envío de información de Arduino a API REST”

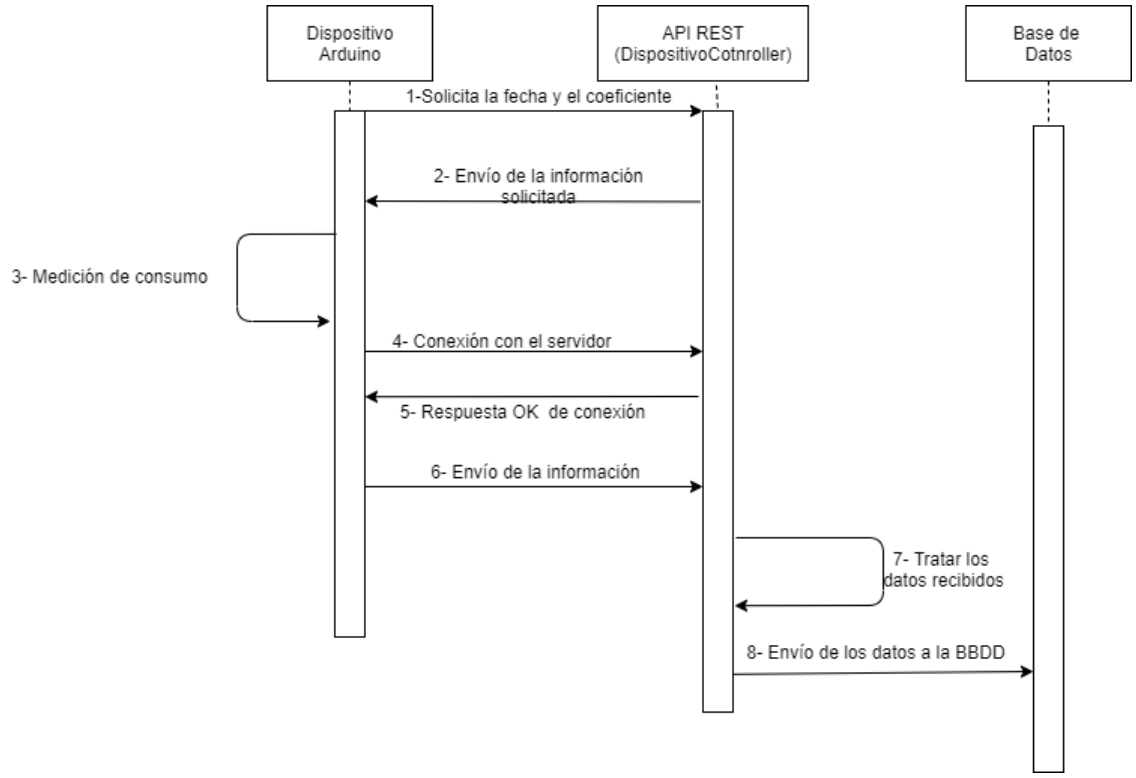


Ilustración 35 Diagrama de envío de datos de Arduino a API REST

Tras encenderse el dispositivo Arduino, éste solicita a la API REST la información necesaria para empezar a cargar los datos. Tras ello y cumplido el tiempo de envío de información, intenta conectarse con el servidor, si la respuesta es OK, envía la información recogida a la API REST, la cual recoge y tras tratar los datos la envía a la base de datos.

6.2.2 Diagrama de secuencia "Mostrar datos a usuario"

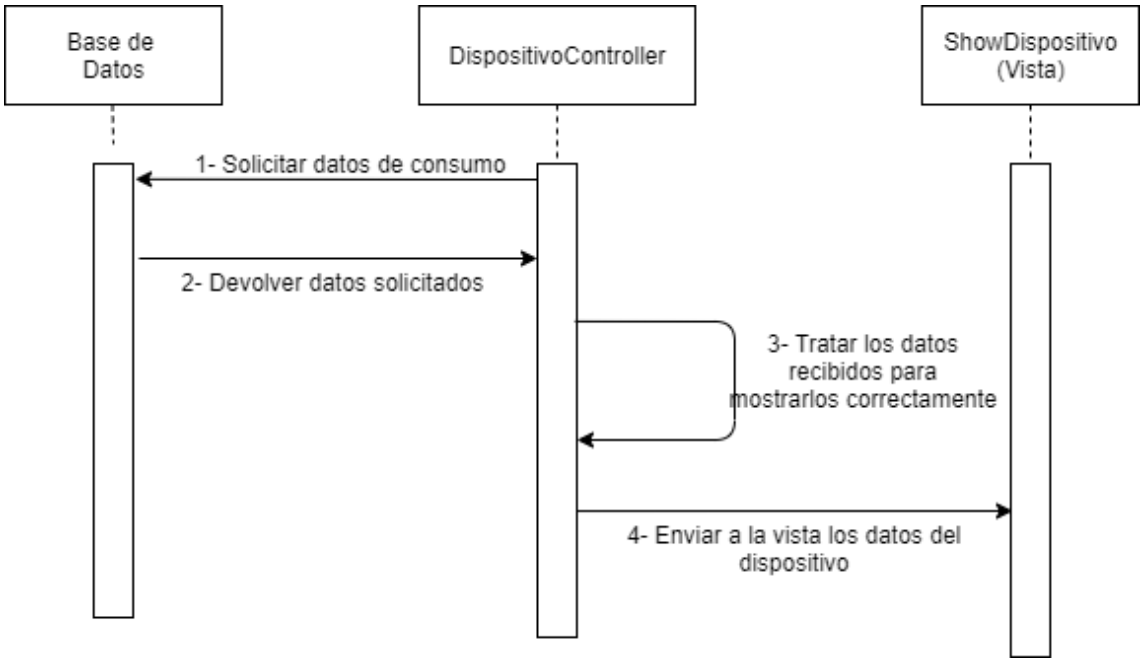


Ilustración 36 Diagrama mostrar datos de dispositivo a usuario

El usuario tras pulsar en uno de sus dispositivos, el DispositivoController solicita a la base de datos la información del dispositivo que se quiere mostrar la información, tras cargar los datos crea los datos para mostrar en la vista, tanto las gráficas, el calendario y demás datos que se encuentran en la vista. Finalmente, envía dicha información a la vista y el usuario puede visualizarla.

6.3 Diseño de la Base de Datos

6.3.1 Descripción del SGBD Usado

Para este proyecto se ha utilizado PostgreSQL, debido a que tiene una buena compatibilidad con el framework Ruby on Rails utilizado. Además incluye características de orientación a objetos, cómo puede ser herencia o los tipos de variables a utilizar. En este proyecto se ha utilizado la última versión de PostgreSQL (hasta la fecha de finalización del proyecto).

6.3.2 Integración del SGBD en Nuestro Sistema

Para integrar este SGBD a nuestro sistema ha sido necesario usar la librería o gema "pg", creada por Michael Granger y disponible en su repositorio https://github.com/ged/ruby-pg, dónde podemos encontrar cómo añadir correctamente esta gema a nuestro Gemfile. Es

necesario tener una versión superior a la PostgreSQL 9.2.X y una versión de Ruby superior a la 2.2 para su correcto funcionamiento.

Ha sido añadido tanto a la API REST del sistema como a la aplicación web de visualización de datos, esto quiere decir que ambas aplicaciones comparten base datos y la utilizan para escribir o leer datos necesarios para la funcionalidad del sistema.

6.3.3 Diagrama E-R

A continuación se muestra un diagrama de entidad relación de nuestro, el cual es el mismo para la API REST como para la aplicación web.

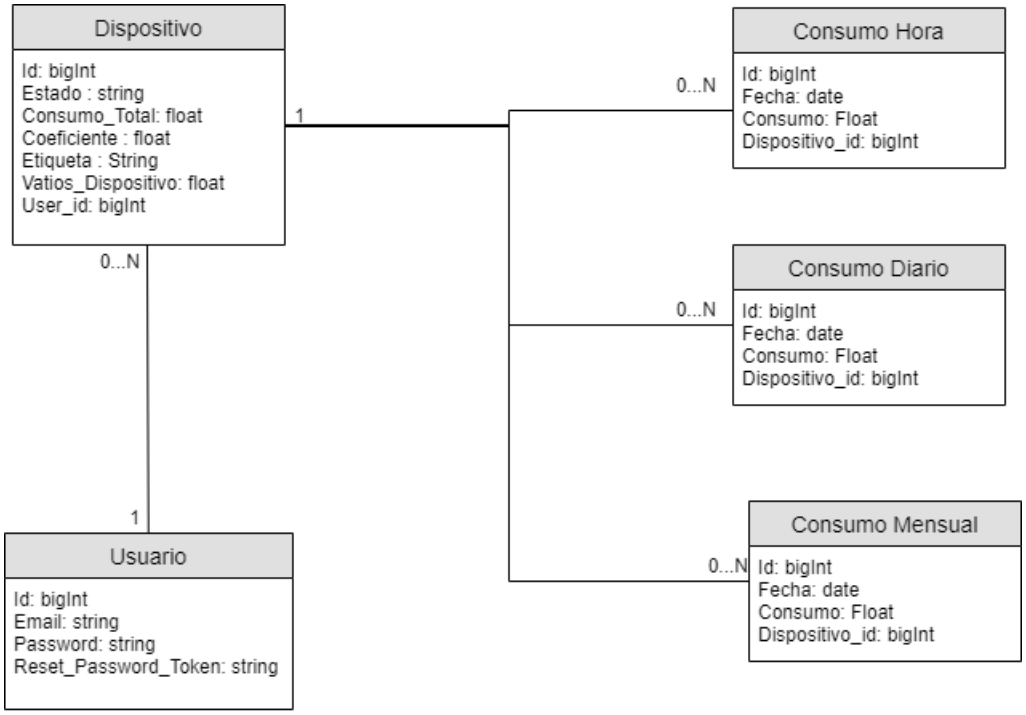


Ilustración 37 Diagrama E-R

6.4 Diseño de la Interfaz

En este apartado se complementará con las ventanas que no se hayan mostrado en el Manual de Usuario, para no duplicar información en el documento. Por lo tanto ciertas funcionalidades como puede ser la barra de navegación se mostrarán aquí.

6.4.1 Barra de navegación

La barra de navegación constará de 4 elementos, un logo y tras él un enlace a todos los dispositivos, añadir un nuevo dispositivo y finalmente desconectarse de la sesión.

Ilustración 38 Barra de navegación

Tanto el logo como “Dispositivos”, llevarán al usuario desde la ventana que se encuentre a la ventana del listado de sus dispositivos.

Al pinchar en “Añadir dispositivos”, se llevará al usuario al formulario de añadir un nuevo dispositivo a su listado, dónde deberá completar un formulario con los datos que se solicitan. Y finalmente el botón “Desconectar” que hará que finalice la sesión del usuario y se muestra la ventana de inicio de sesión.

6.4.2 Pantalla de login

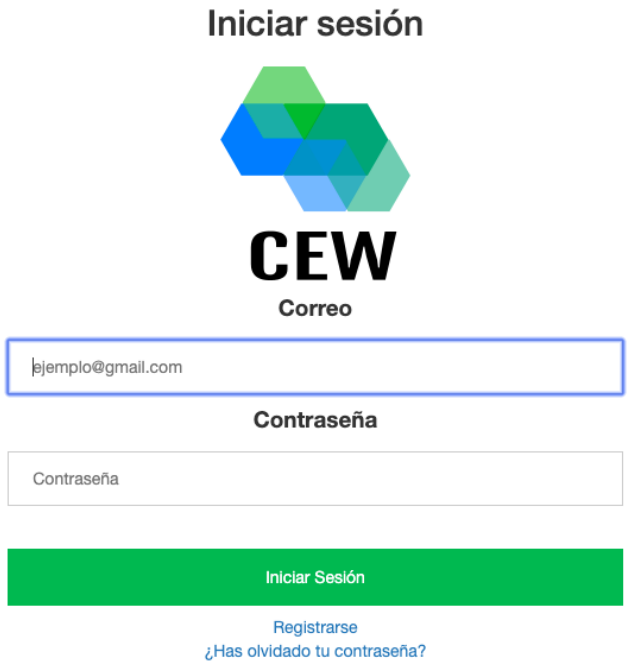


Ilustración 39 Pantalla final: Inicio de sesión

Desde esta pantalla podremos realizar dos acciones, la primera es completar los datos del formulario y acceder a nuestra pantalla principal de usuario, la siguiente es acceder a la pantalla de registrarse si todavía no tenemos una cuenta en el sistema.

Como se puede apreciar, en esta ventana y en la de registro son muy semejantes en cuanto estilo, con un logo, un formulario y finalmente un texto indicando la ventana que es.

6.4.3 Pantalla de registro

Ilustración 40 Pantalla final: registro de usuario

En esta interfaz al igual que en la anterior tendremos dos opciones posibles, la primera es registrarse en el sistema, para luego acceder a la pantalla de inicio de sesión. Y la segunda opción es acceder a iniciar sesión con un usuario que esté previamente registrado en el sistema.

6.4.4 Pantalla principal

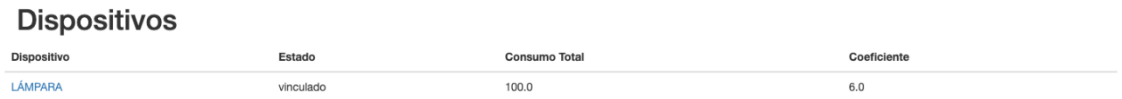


Ilustración 41 Pantalla principal

En esta pantalla podremos acceder a los detalles de los dispositivos que tengamos registrados, a su vez podemos realizar cualquier acción que se encuentre en la barra de navegación.

6.4.5 Pantalla detalles de dispositivo

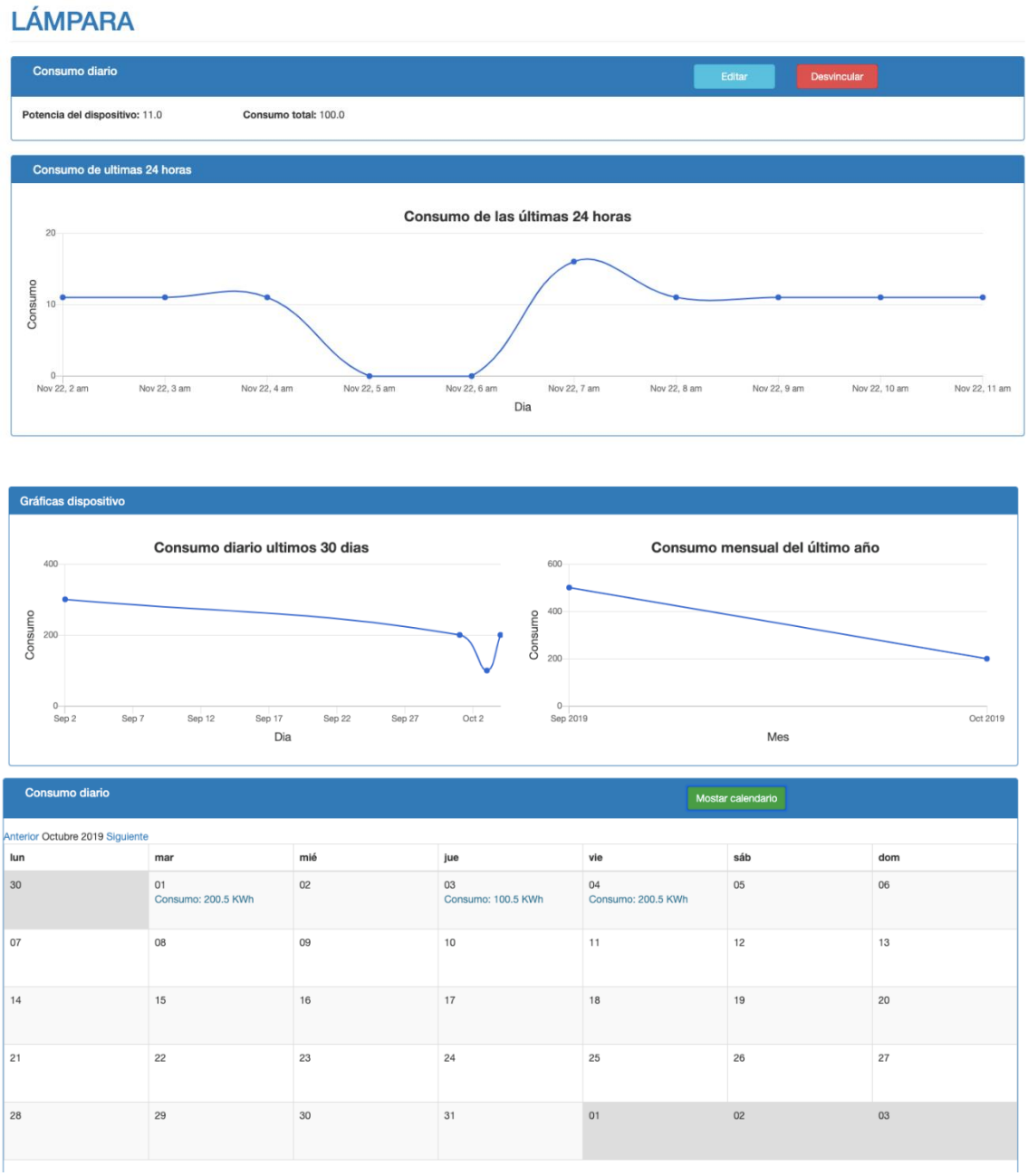


Ilustración 42 Pantalla final: detalles de usuario

En esta pantalla encontraremos los distintos datos de consumo de un dispositivo de distintas maneras. Cómo se puede apreciar en la ilustración anterior, la información se mostrará en forma de gráfica o en el calendario. Además se mostrará información extra al usuario, ya sea la potencia en vatios del dispositivo, cómo el coeficiente de este.

Tendremos a su vez dos funcionalidades en esta ventana, a partir de dos botones, uno de editar, por el cual podremos modificar cierta información del dispositivo, y por otra parte podremos desvincular el dispositivo.

6.4.6 Pantalla editar dispositivo

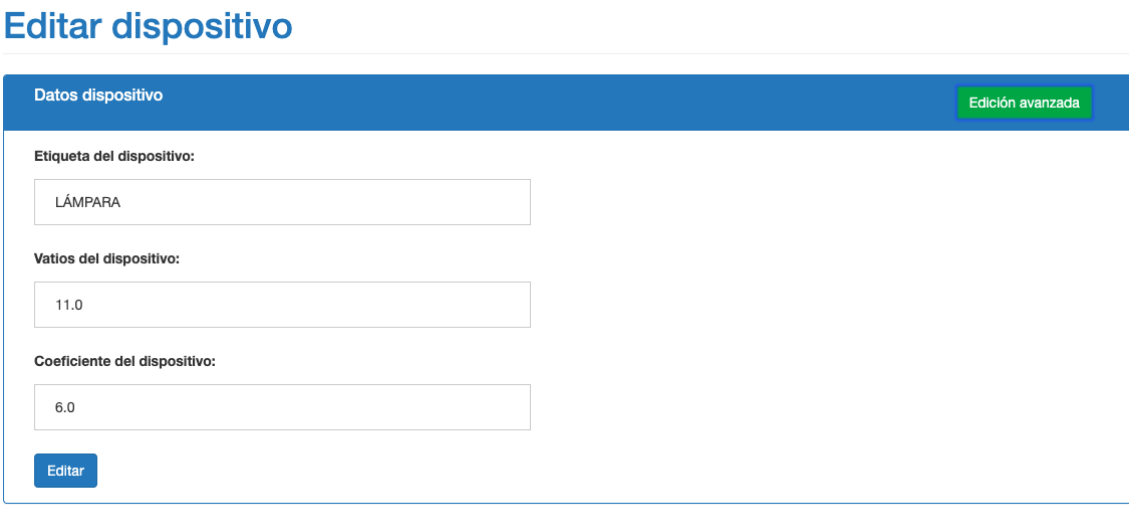


Ilustración 43 Pantalla final: editar dispositivo

En esta ventana podremos editar cierta información del dispositivo que hubiéramos elegido, además cuenta con un botón donde podremos mostrar cierta información extra, en este caso el coeficiente del dispositivo, por si no se hubiera calibrado correctamente.

6.4.7 Pantalla añadir dispositivo

Ilustración 44 Pantalla final: añadir dispositivo

En esta última ventana encontramos un formulario accesible a través de la barra de navegación pulsando en "Añadir Dispositivo". Completando la información que se solicita se puede añadir un nuevo dispositivo al sistema.

6.5 Especificación Técnica del Plan de Pruebas

A continuación se muestra una tabla que contiene los casos de uso previamente creados, los pasos seguido, el resultado esperado y el resultado final obtenido. Se trata tanto de pruebas del dispositivo, como de la API REST y finalmente de pruebas de alto nivel de la aplicación web.

Todas las pruebas se han realizado con los mismos elementos, que son:

- Placa Arduino MKR1010 Wifi.
- Raspberry pi 3b como servidor, con Raspbyan como sistema operativo.
- La versión de Ruby ha sido la 2.5 y la versión de Ruby on Rails 5.2.3
- El navegador web utilizado para mostrar los datos ha sido Google Chrome.
- Finalmente, el Arduino ha sido cargado desde un ordenador con Windows 10 como sistema operativo.

6.5.1 Pruebas unitarias

En este apartado se indicará, por medio de datos reales, las pruebas realizadas a cada uno de los módulos del sistema. Debido a las características del sistema y de los lenguajes de programación utilizados, las pruebas no serán realizados con JUnit (software de Java) o un sistema de éste.

Por lo tanto para estas pruebas se han utilizado tanto log de los distintos sistemas, cómo puntos de interrupción en ciertos test. A continuación se mostrarán tablas con los respectivos casos de uso e indicando la prueba junto con el resultado esperado.

Caso de uso 1: Configuración del Arduino	
Prueba	**Resultado Esperado**
El usuario configura los parámetros de forma correcta	*El dispositivo es capaz de conectarse al wifi correctamente, recibe la información de la API REST y envía tras el tiempo esperado la información correctamente*
Prueba	**Resultado Esperado**
El usuario configura erróneamente la wifi a la que se conecta	*El dispositivo intenta conectarse a la wifi y realiza varios intentos, tras superar el tiempo el sistema muestra un mensaje de que no es posible conectarse al sistema*
Prueba	**Resultado Esperado**
El servidor no está disponible	*El dispositivo intenta recibir la información necesaria para realizar los cálculos del consumo, al no conseguirlos*

	muestra un mensaje de error

Caso de uso 2: Envío de información a la API REST	
Prueba	**Resultado Esperado**
El dispositivo se encuentra en el sistema	*El dispositivo envía la información de consumo recogido y la envía a la API REST, ésta comprueba que el dispositivo existe en el sistema, registra la información y le envía un mensaje, el cual incluye que la información ha sido registrada.*
Prueba	**Resultado Esperado**
El dispositivo no se encuentra en el sistema	*El dispositivo envía la información a la API REST, la API comprueba que el dispositivo no existe en el sistema y tras ello le envía un mensaje de error.*
Prueba	**Resultado Esperado**
Error en el sistema	*El dispositivo, que existe en el sistema, envía información de consumo a la API REST, tras comprobar que existe en el sistema ocurre un fallo al registrar los datos, la API REST enviará un mensaje de error al dispositivo.*

Caso de uso 3: Añadir un dispositivo al sistema	
Prueba	**Resultado Esperado**
El usuario añade un dispositivo a su listado, existe en el sistema y no pertenece a otro usuario	*El sistema comprueba que el dispositivo existe en el sistema, no está vinculado con otro usuario y tras registrar los datos lo añade correctamente.*
Prueba	**Resultado Esperado**
El usuario intenta añadir un dispositivo que no se encuentra en el sistema	*Un usuario intenta añadir un dispositivo que no se encuentra en el sistema, tras ser comprobado se le muestra un mensaje al usuario para que contacte con el administrador para añadir este dispositivo.*
Prueba	**Resultado Esperado**
Un usuario intenta añadir un dispositivo que está vinculado con otro usuario	*El sistema muestra un mensaje al usuario diciendo que ese dispositivo se encuentra vinculado con otro usuario y que no es posible vincularlo con su cuenta.*

Caso de uso 4: Registro de usuario en el sistema de visualización de datos	
Prueba	**Resultado Esperado**
Un usuario que no está registrado en el sistema intenta registrarse	*El sistema recoge los datos, y si estos son correctos, registra al usuario en el sistema.*
Prueba	**Resultado Esperado**
Un usuario que no está registrado en el sistema intenta registrarse con datos erróneos	*El sistema comprueba los datos y al no ser correctos, le muestra un mensaje al usuario de error y le reenvía a la página de registro.*
Prueba	**Resultado Esperado**
Un usuario que se encuentra en el sistema intenta registrarse	*El sistema comprueba que el usuario se encuentra en el sistema y le muestra un mensaje de error, indicando que ya se ha registrado.*

Caso de uso 5: Inicio de sesión en el sistema	
Prueba	**Resultado Esperado**
Un usuario intenta iniciar sesión, estando en el sistema y con datos válidos	*El sistema comprueba que el usuario se encuentra registrado y le muestra su pantalla de inicio.*
Prueba	**Resultado Esperado**
Un usuario que no se encuentra registrado en el sistema intenta iniciar sesión	*El sistema comprueba que el usuario no existe en el sistema, tras ello le muestra un mensaje de error indicando que no existe el usuario.*
Prueba	**Resultado Esperado**
El usuario intenta iniciar sesión con datos erróneos	*El sistema comprueba el usuario y la contraseña, si la contraseña no es correcta le muestra un mensaje de error al usuario y le reenvía a la página de inicio de sesión.*

Caso de uso 6: Acceder a un dispositivo	
Prueba	**Resultado Esperado**
Un usuario intenta acceder a la información de uno de sus dispositivos	*El sistema comprueba que el usuario del dispositivo es el mismo que está en sesión y tras ello le muestra la página de detalles del dispositivo.*
Prueba	**Resultado Esperado**
Un usuario intenta acceder a un dispositivo que no está vinculado	*El sistema le muestra un mensaje de error, y le reenvía a la página de inicio, indicando que el dispositivo no está vinculado con él.*
Prueba	**Resultado Esperado**

Un usuario intenta acceder a un dispositivo que no se encuentra en el sistema	*El sistema mostrará un mensaje de error indicando que no existe el dispositivo en el sistema. Y tras ello, reenviará al usuario a su página de inicio*

Caso de uso 7: Editar un dispositivo	
Prueba	**Resultado Esperado**
Un usuario intenta editar un dispositivo con datos correctos	*El sistema comprueba que los datos son correctos y que el usuario está vinculado al dispositivo y tras ello le muestra un mensaje diciendo que el dispositivo ha sido editado correctamente.*
Prueba	**Resultado Esperado**
El usuario intenta editar un dispositivo con el cual no está vinculado	*El sistema le muestra un mensaje de error al usuario y le reenvía a su página de inicio.*

Caso de uso 8: Desvincular un dispositivo	
Prueba	**Resultado Esperado**
Un usuario intenta desvincular uno de sus dispositivos	*El sistema comprueba que el usuario está vinculado con el dispositivo y tras ello elimina la desvinculación entre usuario y dispositivo, a continuación, el sistema reenvía al usuario a su página de inicio.*
Prueba	**Resultado Esperado**
El usuario intenta desvincular un dispositivo que no se encuentra en el sistema	*El sistema comprueba que el dispositivo no existe y tras ello el sistema reenvía al usuario a su página de inicio, mostrando un mensaje de error.*
Prueba	**Resultado Esperado**
El usuario intenta desvincular un dispositivo, con el cual no está vinculado	*El sistema comprueba que el usuario del dispositivo no es el mismo que se encuentra en sesión y reenvía al usuario a su página de inicio mostrando un mensaje de error indicando que el dispositivo no está disponible.*

Caso de uso 9: Envío de información desde el dispositivo a API REST caída	
Prueba	**Resultado Esperado**
El dispositivo intenta enviar información al servidor que se encuentra caído	*Tras intentar enviar la información al servidor y ver que no ha sido posible el dispositivo almacena la información, para que cuando sea el tiempo de enviar información perdida vuelva a intentar lanzarse*
Prueba	**Resultado Esperado**
El dispositivo intenta	*El dispositivo intenta enviar información al servidor, que*

enviar información previamente intentada enviar	*previamente había sido enviada, tras comprobar que la API REST aún se encuentra sin conexión el sistema lo sigue intentando con otras, pero no la vuelve a almacenar y evita duplicados*

Caso de uso 10: Añadir un nuevo dispositivo al sistema	
Prueba	**Resultado Esperado**
El usuario intenta registrar un dispositivo que no se encuentre aún en el sistema	*El sistema comprobará que no hay registro en la base de datos del nuevo dispositivo, y tras comprobarlo, lo añadirá al sistema. Devolviendo un mensaje de confirmación al usuario.*
Prueba	**Resultado Esperado**
El usuario intenta añadir un dispositivo que ya se encuentra en el sistema	*El sistema comprueba que el dispositivo ya se encuentra en la base de datos, tras ello envía un mensaje al usuario informando que el registro ya se encuentra en la base de datos, no se duplica la información*

6.5.2 Pruebas de integración

En este apartado se listarán las pruebas de integración entre el dispositivo Arduino y la API REST, ya que son los módulos que están conectados entre sí. Tras ello se mostrarán las pruebas de integrar la API REST con la aplicación web. Así mismo se indicará a que caso de uso hace referencia y los datos de prueba necesarios.

6.5.2.1 Dispositivo Arduino y API REST

Esta prueba de integración es la más importante del sistema, ya que permitirá recibir la información que ha recogido el Arduino y enviarlo a la API REST, para que luego pueda ser usado por el usuario final.

Esta prueba se realizará en el momento que esté montado correctamente el dispositivo y ya se encuentre operativo, de la misma manera la API REST deberá encontrarse ya disponible para recibir y enviar información a los distintos dispositivos Arduino.

Los datos de entrada se dividirán en dos, los que envía el dispositivo y los que envía la API REST. A continuación se mostrará para cada uno de los casos datos que envía y cómo son tratados desde el otro módulo.

- Datos enviados desde la API REST: este módulo enviará un JSON en el cual se compone por los siguientes valores:
    - Coeficiente: con el valor de coeficiente del dispositivo que solicita la información, por lo tanto el sistema buscará el dispositivo y a continuación recogerá el valor solicitado.

    - Fecha: este valor es necesario enviarlo ya que el Arduino no tiene un sistema de fecha por el mismo, por lo tanto se inicializará una variable de tipo TIME y se seleccionará la fecha del momento de solicitud, tras ello se dividirá la fecha en día, mes, año, hora, minuto y segundo.

  Toda esta información se juntará en un JSON y será enviado al dispositivo Arduino, el cual recibirá los datos y los almacenará, por una parte en una variable coeficiente, y por otro la fecha la utilizará para inicializar una variable interna de tipo TIME para que así ya tengamos un reloj interno en él. Un ejemplo de datos de envío es: {coeficiente:20, now: {dia: 10, mes: 12, year: 2019, hour: 12, minute: 30, second: 22}}, esta sería la información que enviaría la API REST al sistema.
- Datos enviados desde el Arduino: el dispositivo irá almacenando el consumo del dispositivo durante el tiempo que haya sido previamente definido, tras ello enviará una media de la potencia obtenida junto con la fecha del envío en un JSON, además de enviar su identificador en la petición PATCH, la cual deberá ser creada a mano y deberá incluir correctamente la información que se quiere enviar. Tras ser enviado, la API REST comprueba primero que existe dicho dispositivo en el sistema, ya que de no ser así el sistema enviará una respuesta al otro módulo indicando que no ha sido posible tratar la información. Si el dispositivo existe, creará nuevos registros en la tabla de registros hora, tras ello añadirá al registro diario y mensual el nuevo consumo y actualizará finalmente el consumo total que lleva el dispositivo que ha enviado la petición. Un ejemplo de los datos que se envían es: {consumo: 16,date_ard: "16-12-2019 12:23:12"}, dónde date_ard es la fecha actual de envío de información.

6.5.2.2 Aplicación web

Para probar la aplicación web con los demás elementos del sistema se ha seguido los siguientes pasos, en relación con añadir un nuevo dispositivo al sistema, luego añadir un usuario nuevo, tras ello asignar ese dispositivo al usuario y comprobar que la información de consumo llega desde el dispositivo al usuario. Los datos utilizados han sido:

- Dispositivo: se ha añadido previamente un dispositivo con identificador = 1, un coeficiente de 6, una potencia de 15 V y finalmente el nombre de "Ordenador Portátil".
- Usuario: se ha registrado un usuario con email "test@test.com" y contraseña "123456".
- Tras ello se ha accedido a la sesión del usuario y se ha probado a añadir el primer dispositivo al usuario para visualizar la información. Cómo respuesta el sistema deberá añadir correctamente el dispositivo al sistema.

Para continuar con la prueba se ha seguido añadiendo información, la cual es:

- Dispositivo 2: este segundo dispositivo copia la información anterior a excepción del nombre, el cual ahora es "Lámpara".
- Usuario 2: se registra un segundo usuario con correo "test2@test.com" y de contraseña "123456". A continuación, se añade al listado de dispositivos del usuario 2 el dispositivo 2.

Tras ello, accedemos con el primer usuario al sistema e intentamos añadir el dispositivo 2, pero cómo se encuentra registrado con otro usuario no se permite el registro.

Finalmente y cómo última prueba de este apartado se desvincula del usuario 2 el dispositivo 2 correctamente. Tras ello se accede como el primer usuario al sistema, y se intenta añadir el dispositivo 2, cómo ahora no pertenece a ningún usuario, el dispositivo se añade correctamente al primer usuario.

Capítulo 7. Implementación del Sistema

7.1 Lenguajes de Programación

A continuación se muestran los lenguajes de programación que se han usado juntos con ciertas librerías importantes a destacar para el correcto funcionamiento de la ampliación.

7.1.1 C/C++ Arduino

Es un lenguaje de programación utilizado para programar las placas o dispositivos Arduino, aunque éstas también pueden ser programadas en Python, se ha escogido este lenguaje por las librerías que incorpora.

- Emonlib: es una librería orientada al procesamiento y cálculo de datos en relación con consumos eléctricos, como se describe en el apartado Emonlib.
- Wifinina: librería que facilita el uso de Wifi en las placas Arduino, para más información consultar el apartado Wifinina.

Ilustración 45 Logo de C++

7.1.2 Ruby

Es un lenguaje de programación creado por Yukihiro Matz Matsumoto y es una combinación de lenguaje de programación anteriores a éste. Es un lenguaje de programación libre [4].

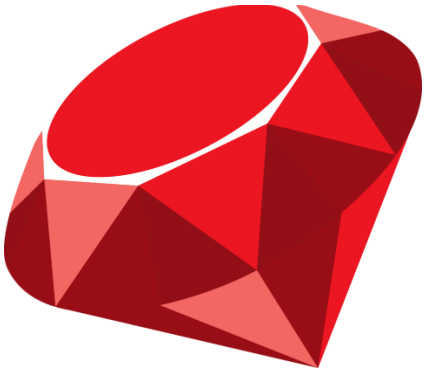


Ilustración 46 Logo Ruby

7.1.3 Rails

Es un framework web escrito en Ruby, se basa en un Modelo Vista Controlador, es decir, tendremos una vista o página web, la cual será controlada por un controlador y se basará en ciertos modelos. Dentro del proyecto se encuentran ciertas librerías, o en este caso gemas ya que así se denominan en este lenguaje, importantes para el desarrollo:

- Bcrypt: es una gema que facilita el encriptamiento de datos con la base de datos que utilicemos.
- Bootsnap: permite utilizar las plantillas de estilo de Bootstrap en nuestra aplicación.
- Devise: crea de forma automática un sistema básico de usuarios, ofreciéndonos un login y registro de usuarios con sus vistas y controladores [6].
- Kaminari: esta última gema proporciona un sistema de paginación de forma sencilla para los listados que tengamos en nuestro proyecto.
- Haml-rails: permite la compatibilidad de HAML con Rails.

Ilustración 47 Logo de Ruby on Rails

7.1.4 HAML

Está basado en HTML, pero facilita su escritura ya que elimina las etiquetas y no es necesario por ejemplo los cierres. En cambio, es necesario tener correctamente tabulado todo el documento, ya que de no ser así no conseguiremos el resultado deseado o generará errores.

Ilustración 48 Logo Haml

7.2 Herramientas y Programas Usados para el Desarrollo

7.2.1 Herramientas para el desarrollo

7.2.1.1 Arduino IDE

Arduino IDE es un software orientado a la programación de placas de Arduino, está basado en el entorno de Processing. Además facilita la importación de librerías a nuestro código, junto con funcionalidades como subir el software a la placa o compilar el código antes de ser subido.

7.2.1.2 Atom

Es un editor de código abierto, el cual soporta gran cantidad de lenguajes de programación, debido a que se pueden instalar plugins para que la creación de código sea más sencilla.

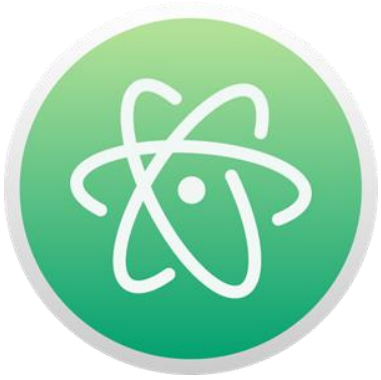


Ilustración 49 Logo de Atom

7.2.1.3 GitHub

Es una herramienta que permite tener un control de versiones de nuestro proyecto a través de Git, además sirve para guardar este proyecto de forma privada durante su desarrollo.

Ilustración 50 Logo de GitHub

7.2.1.4 Postman

Un software para probar de manera fácil y sencilla peticiones para la API REST de este proyecto. Con él se puede seleccionar el tipo de petición que queremos enviar, y la dirección a la que queremos enviarla. Dependiendo del tipo de petición que queramos enviar, se puede completar con información extra en la cabecera o en el cuerpo de la petición.

Ilustración 51 Logo Postman

7.2.2 Herramientas para la documentación

A continuación se muestran las herramientas que han sido utilizadas para realizar la documentación del proyecto.

7.2.2.1 Microsoft Word

Es un editor de texto de la compañía Microsoft para la redacción del documento del proyecto.

7.2.2.2 Microsoft Excel

Es un programa de Microsoft especializado en tareas contables o financieras, en el caso de este proyecto ha sido utilizado para todo lo relacionado con el presupuesto de este.

7.2.2.3 Microsoft Project

Herramienta para controlar y definir las tareas y la duración de estas, así mismo, para asignar la tareas a los trabajadores correspondientes y tener un visión general de la duración y recursos destinados para un proyecto.

7.2.2.4 Draw.io

Es un software online creado para diseñar diagramas de varios estilos, en este caso ha sido utilizado para crear los diagramas de base de datos.

Ilustración 52 Logo Draw.io

7.2.2.5 Fritzing

Es un software que permite realizar esquemas para circuitos, además incorpora distintas vistas como la de la Protoboard o la del PCB final. En este caso ha sido utilizado para mostrar el montaje del prototipo y además incluir un esquema de cómo quedaría la PCB final. Para este proyecto se ha utilizado una versión de prueba del sistema, ya que para utilizar este software es necesario pagar una cuota de 8 euros.

Ilustración 53 Logo Fritzing

7.3 Creación del Sistema

7.3.1 Diagrama de construcción

En este apartado se entrará en profundidad en la construcción y conexiones del dispositivo. Para empezar se mostrará un esquema del dispositivo, en este caso ya que la herramienta utilizada no poseía la placa que se utiliza en el proyecto, se utilizará la placa Arduino Uno, ya que a excepción de la salida de corriente del dispositivo, las entradas son las mismas.

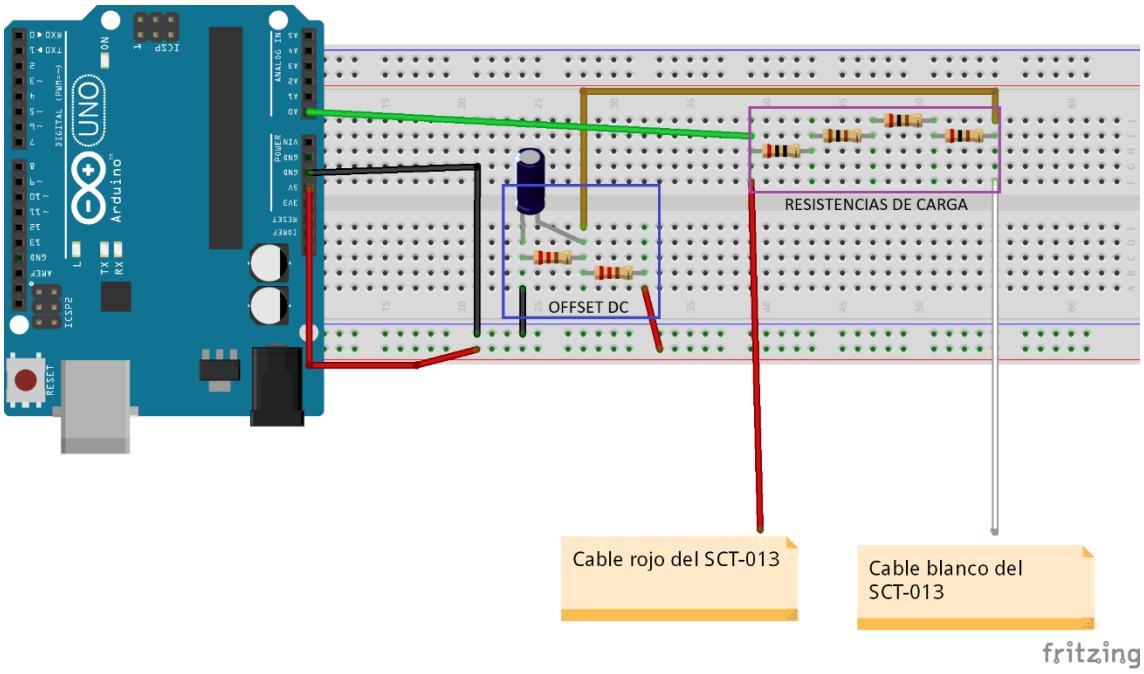


Ilustración 54 Diagrama de construcción del dispositivo

Como se puede observar dentro del diagrama ofrecido existen cuatro partes dentro de la construcción del dispositivo de medición como son el sensor SCT-013, la zona de resistencia de carga, el offset DC y finalmente la placa Arduino. A continuación se explicarán y se nombrarán los componentes utilizados.

7.3.1.1 Sensor SCT-013

Existen dos tipos de sensores SCT-013, unos que obtienen la corriente y otros que obtienen el voltaje del dispositivo, la gran diferencia entre ambos es que los que obtienen la corriente no tienen una resistencia de carga interna. En este proyecto se han utilizado el SCT-013-100 que pertenece al primer grupo.

Este sensor puede medir corrientes que van desde los 50 mA hasta los 100 A y transforman esa corriente en una corriente a otra que va desde los 0 A hasta los 50 mA. En cambio si el dispositivo tuviera esa resistencia interna la salida del sensor ya no sería corriente si no voltaje.

A continuación se muestra una tabla con la entrada y salida de los distintos sensores SCT-013 que existen en el mercado (la tabla es de la empresa YHDC, se encuentra en este *enlace)*

Model	SCT-013-000	SCT-013-005	SCT-013-010	SCT-013-015	SCT-013-020
Input current	0-100A	0-5A	0-10A	0-15A	0-20A
Output type	0-50mA	0-1V	0-1V	0-1V	0-1V
Model	SCT-013-025	SCT-013-030	SCT-013-050	SCT-013-060	SCT-013-000V
Input current	0-25A	0-30A	0-50A	0-60A	0-100A
Output type	0-1V	0-1V	0-1V	0-1V	0-1V

Ilustración 55 Tabla de entrada y salida de los sensores SCT-013

Como se puede apreciar el sensor SCT-013 puede ser cambiado por otro dentro de la gama dependiendo de las necesidades del usuario, es por tanto que este proyecto ofrece una gran compatibilidad dependiendo de las necesidades del usuario.

7.3.1.2 Resistencias de carga

Debido a que en nuestro proyecto se ha optado por un sensor que no tiene resistencias de carga interna se ha tenido que añadir al circuito, por lo tanto si se escoge otro sensor SCT-013 distinto el cual si la contenga, esta parte no se deberá introducir.

Por otro lado, hay que tener en cuenta el tipo de placa Arduino que vayamos a utilizar, en nuestro caso se ha optado por una placa MKR 1010 Wifi, a diferencia del Arduino Uno, esta placa tiene un voltaje máximo de 3,3 V que puede medir, mientras que el otro tiene un voltaje máximo de 5 V, este es un dato importante para tener en cuenta, ya que deberemos usar una resistencia de carga distinta para cada uno de ellos. Para calcular esto usamos la fórmula de resistencia de carga que es: $R_{carga} = \frac{\frac{AREF}{2}}{I_{pico}}$. Dónde AREF es la referencia interna de la entrada analógica, en nuestro caso de 3,3 V y la I pico es la corriente de pico, que se explicó previamente en Ley de Ohm.

Con los datos que calculamos, en este proyecto se obtiene un valor de 514Ω y debido a que los valores de resistencia que tenemos no nos dan para obtener ese valor exacto, se usará una resistencia de carga final de 520Ω.

7.3.1.3 Offset DC

Esta parte del circuito es un poco más compleja de entender, debido a que nuestro dispositivo tiene un voltaje máximo de 3,3 V, obtendremos una variación entre los 1,65 V a -1,65 V lo que sigue siendo una señal sinusoidal. El problema viene que las placas Arduino no leen voltajes negativos y por tanto tenemos que convertir esos valores a valores de 0 V a 3,3 V. Para ello es necesario el offset en DC para la señal que se recibe.

Para ello es necesario dos elementos en concreto un condensador que deberá ser de 10µF y por otro lado un divisor de tensión, el valor de este cambiará si el dispositivo va conectado a una batería o a la corriente, si optamos por la segunda opción el valor total de la resistencia deberá ser de 10kΩ y de no ser así bastará con un valor de unos 450 a 480kΩ [3].

7.3.1.4 Placa Arduino

Finalmente necesitamos una placa Arduino para tratar los datos de entrada y obtener un consumo real del dispositivo que estamos midiendo y por otro lado necesitamos que nuestro dispositivo tenga un sensor o una placa externa para enviar la información desde la placa Arduino al servidor donde está alojada la API REST.

En nuestro caso, hemos optado por una placa Arduino MKR 1010 Wifi, ya que viene con un sensor wifi incluido, en cambio sí optamos por un Arduino Uno será necesario incluir una placa extra que ofrezca el envío y recepción de información mediante conexión Wifi.

Las conexiones con el resto de los elementos previamente citados se basan en dar corriente al resto del circuito, conexión de 5 V, y de un punto de tierra, la conexión GND, además será necesario conectar la entrada del sensor SCT-013 a una de las entradas analógicas, ya que el valor de la corriente varía entre un máximo y un mínimo constantemente.

Finalmente y como se ha mencionado previamente, la placa deberá ir conectada a una fuente de energía ya sea mediante baterías o por medio de un cable a la corriente eléctrica.

7.3.1.5 Otros elementos

Debido a que el sensor SCT-013 viene con un cable Jack macho, se puede optar por dos opciones una es añadir a nuestro circuito un conector Jack hembra o por otro lado cortar dicho cable Jack y así obtener dos cables, uno rojo y otro blanco. En este proyecto se ha optado por la segunda opción, ya que tras probar con el conector Jack hembra no funcionaba correctamente las mediciones, y no se tenía más tiempo para cambiarlo, ya que suponía atrasar el proyecto y aumentar el gasto de este.

7.3.1.6 Construcción final del dispositivo

A continuación se mostrará el resultado final de la construcción del proyecto con todos los elementos previamente citados:

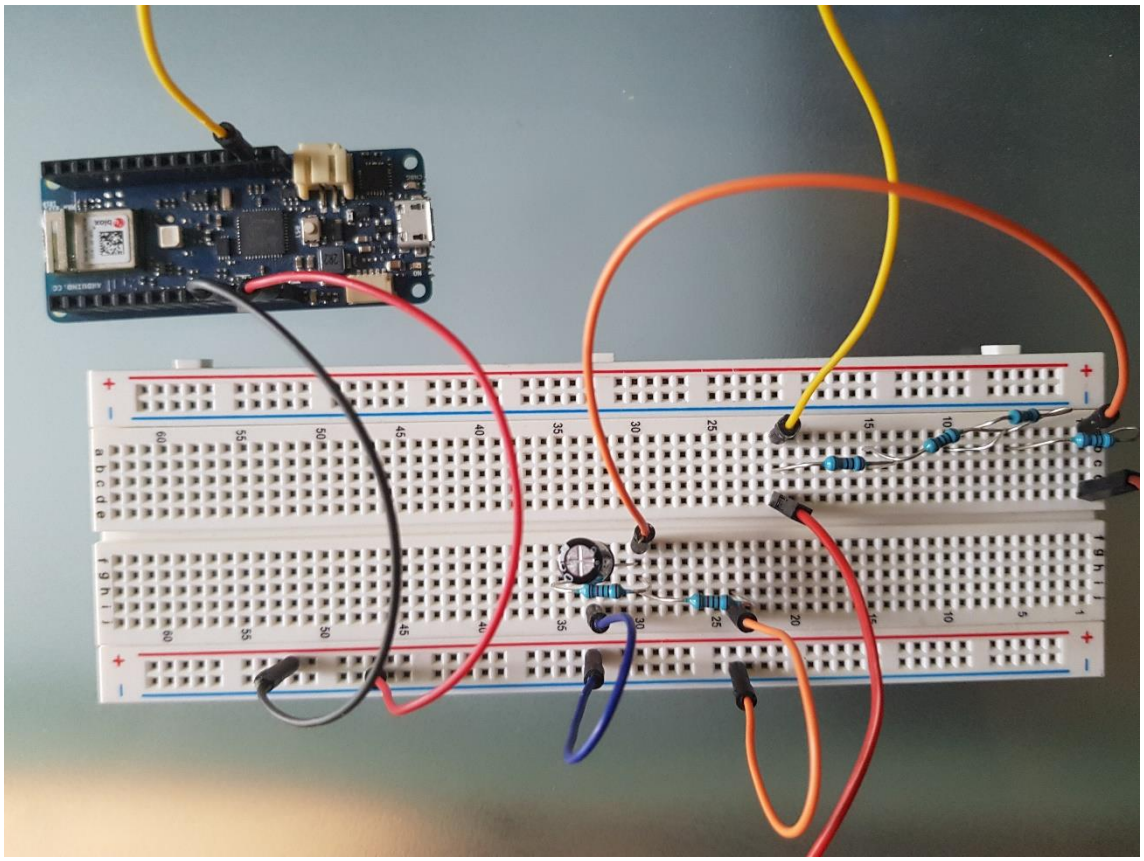


Ilustración 56 Construcción del dispositivo sin el sensor

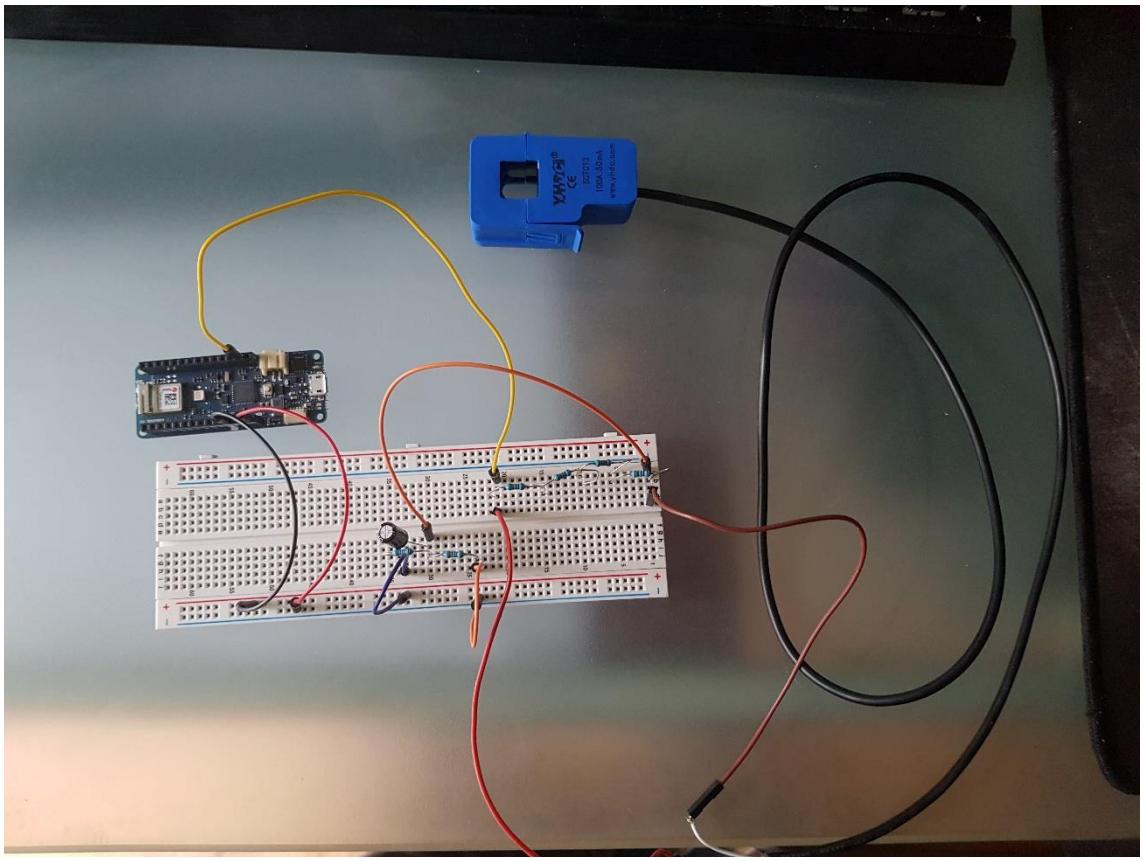


Ilustración 57 Construcción del dispositivo con el sensor

Como se puede apreciar se ha utilizado finalmente la Protoboard ya que el uso de placas PCB aumentaba el coste del proyecto y obtenerlas retrasaba la entrega de éste.

7.3.2 Software del dispositivo

En esta sección se indicarán las partes más importantes del script que ejecuta el dispositivo Arduino. Centrándose en la conexión wifi, envío y recepción del software.

7.3.2.1 Conexión wifi

Para que nuestro dispositivo pueda conectarse con la red wifi es necesario que éste conste de un módulo wifi. Tras ello podemos instalar la librería Wifinina en el código, para ello debemos incluir en el código la librería mediante: #include <WiFiNINA.h> al principio del código. Es importante recalcar que previamente debemos incluir la librería en el Arduino IDE.

Tras ello inicializamos el WifiClient, el cual es un objeto que ofrece la librería para conectarse a la wifi deseada y además de poder luego enviar o recibir peticiones, para ello es necesario indicar un SSID y una contraseña de la wifi a conectarse.

```
char ssid[] = SECRET_SSID;
char pass[] = SECRET_PASS;
```

Ilustración 58 Configuración de SSID y contraseña wifi

Finalmente, a través del método Wifi.begin(SSID,Contraseña), el objeto intenta conectarse a la wifi.

7.3.2.2 Enviar petición HTTP

Para enviar una petición, en este caso al servidor que aloja la API REST, es necesario establecer primero el DNS o IP del servidor y a continuación indicar el puerto donde está.

```
//IPAddress server[redacted];
char server[] = "raspberryjuantfg.hopto.org";

//Port of the API
int port = 3000;
```

Ilustración 59 DNS y puerto del servidor

A continuación se mostrarán las dos peticiones con las que trabaja el proyecto, la primera se tratará de una petición GET para solicitar la información del día actual y el coeficiente del dispositivo y la segunda de una petición POST enviando el consumo y la fecha, además se mostrará cómo crear el JSON que irá en el cuerpo de la petición.

```
if (client.connect(server, port)) {
  Serial.println("connected to server");
  // GET the coeficiente:
  client.println("GET /dispositivos/1 HTTP/1.1");
  client.println("Host: 192.168.1.116:3000");
  client.println("User-Agent: PostmanRuntime/7.17.1");
  client.println("Accept: */*");
  client.println("Cache-Control: no-cache");

  client.println("Connection: keep-alive");
  client.println();
}
```

Ilustración 60 Petición GET Arduino

Como se puede observar se indica primero la dirección y puerto del servidor, y si se consigue la conexión con él, se envía una petición con la petición a solicitar y la ruta para obtener dicha solicitud. Además en este caso se indica información adicional que puede ser utilizada en futuras ampliaciones.

Para la otra petición será necesario primero generar primero el JSON que queremos enviar y a continuación realizar una petición semejante a la anterior.

```
String JsonData = "{\"consumo\":";
JsonData = JsonData + "\"" + pt + "\",";
JsonData = JsonData + "\"date_ard\":";
JsonData = JsonData + "\"" + date + "\"}";

if (client.connect(server, port)) {
  Serial.println("Send info");
  client.println("PATCH /dispositivos/add_consumo/" + id_arduino + " HTTP/1.1");
  client.println("Host: 192.168.1.116:3000");
  client.println("Content-Type: application/json");
  client.println("User-Agent: Arduino/1.0");
  client.println("Connection: close");
  client.print("Content-Length: ");
  client.println(JsonData.length());
  client.println();
  client.println(JsonData);

}
```

Cómo podemos apreciar es necesario construir el JSON a mano para poder ser enviado con los parámetros que sean oportunos. Y al igual que en la anterior petición indicaremos la ruta y el host final.

7.3.2.3 Recibir petición desde la API REST

A continuación se mostrará cómo recibir la información tras enviar la petición GET al servidor, para ello será necesario crear un nuevo objeto de tipo JsonObject, proporcionado por la librería ArduinoJSON y que al igual que otras librerías deberemos añadirla previamente.

```
while (client.connected() || client.available())
{
  if (client.available())
  {
    String line = client.readStringUntil('\n');

    JsonObject& root = jsonBuffer.parseObject(line);
    if ( root["now"]["dia"] != 0) {
      day_r = root["now"]["dia"];
      month_r = root["now"]["mes"];
      year_r = root["now"]["year"];
      hour_r = root["now"]["hour"];
      minute_r = root["now"]["minute"];
      second_r = root["now"]["second"];
    }

    if (root["coeficiente"] != 0)
      coeficiente = root["coeficiente"];


  }
```

Ilustración 61 Tratar información recibida.

Cómo podemos observar se realiza un bucle while en la petición ya que existe un problema en recibir la información del servidor la cual puede ser vacía o en este caso igual a cero. Tras recibir la información se convierte a un JSON y luego utilizamos la información recibida para completar las variables que necesitamos.

7.3.2.4 Calcular consumo obtenido

Finalmente, para calcular el consumo leído por el sensor SCT-013 se ha utilizado la librería EmonLib, la cual es necesaria incluir previamente en el sistema. Tras ello en el setUp del script a través de la creación de un objeto de la clase EnergyMonitor y por medio el método current(PIN de entrada, coeficiente), podemos inicializar el sistema.

```
//El primer parámetro es el pin conectado
//El segundo es el valor del coeficiente devuelto de la API
energyMonitor.current(0, coeficiente);
```

Ilustración 62 Inicializar objeto EnergyMonitor

Tras ello debemos calcular en cada vuelta del script la potencia que hemos obtenido y para ello se ha creado un método que a partir de la entrada podemos calcular el consumo del dispositivo a medir.

```
double getIrms() {
  // Obtenemos el valor de la corriente eficaz
  // Pasamos el número de muestras que queremos tomar
  double Irms = energyMonitor.calcIrms(1484);

  // Calculamos la potencia aparente
  double potencia =  Irms * voltajeRed;
  return potencia;
}
```

El valor del método calcIrms es el número de muestras que queremos obtener. Tras calcular el IRMS y por medio de la ley de Ohm obtenemos la potencia y la devolvemos.

7.3.3 Software de la API REST

Esta API es la base para enviar y recibir la información obtenida en el Arduino y enviarla a la base de datos para que luego pueda ser visualizada. Este apartado se centra en dos de las partes más importantes del sistema, por un lado la configuración de rutas y por el otro el controlador de dichas peticiones.

7.3.3.1 Configuración de rutas

Al realizar este proyecto mediante Ruby on Rails, éste proporciona medios que facilitan el crear nuevas rutas en el sistema. Para ello primero debemos acceder a la carpeta de configuración y luego acceder al fichero “routes.rb”, el cual es creado por defecto gracias a la herramienta. En este fichero podremos configurar el tipo de petición por cada ruta y el controlador que tratará dicha petición. En nuestro caso solo existe un controlador de peticiones. A continuación se muestra una imagen del archivo previamente nombrado con las rutas configuradas.

```
Rails.application.routes.draw do
  resources :consumo_mensuals
  resources :consumo_diarios
  resources :dispositivos
  devise_for :users
  # For details on the DSL available within this file, see http://guides.rubyonrails.org/routing.html
  controller :dispositivos do
    patch '/dispositivos/add_consumo/:id' => :add_consumo
    post '/dispositivos/add_disp/:id' => :add_disp
  end
end
```

Ilustración 63 Archivo de configuración de rutas

Como podemos observar se pueden apreciar dos partes, las primeras líneas son creadas por defecto por el sistema. Pero en la parte que se menciona el controlador del dispositivo, esa parte es configurada, en ella se estable que por medio de una petición PATCH y a través de esa ruta el controlador del dispositivo llamará al método “add_consumo” para que trate la información recibida. Cómo apunte, la última parte de la ruta “:id”, ese valor será sustituido en el dispositivo Arduino por el identificador propio de éste.

7.3.3.2 Añadir nuevo dispositivo al sistema

A través de la API REST se añadirán nuevos dispositivos al sistema, esto se debe a que se pueden utilizar otros sistemas o interfaces distinta a la de visualización de datos para gestionar la inserción y eliminación de dispositivos en el sistema. Para ello se ha creado una nueva ruta en el archivo de rutas del sistema por el cual a través de la ruta “dispositivo/add_disp/:id” podemos añadir nuevos dispositivos Arduino que se vayan creando al sistema.

El sistema comprobará primero que el dispositivo no se encuentre ya añadido, ya que de ser así se enviará un mensaje al otro sistema indicando que el dispositivo ya ha sido añadido y no se volverá a añadir. De no existir en el sistema, se creará un nuevo registro en la base de datos, el usuario del dispositivo será el usuario administrador con correo “admin@admin.com”. Esto permitirá a usuarios de la aplicación de visualización de datos vincularse con estos usuarios.

7.3.3.3 Añadir un nuevo consumo al sistema

En la clase “DispositivoController” se encuentra el método del apartado anterior, el cual recibe la información de la petición y tras comprobar si el dispositivo existe o no. En caso de que exista realiza una serie de operaciones para añadir la información a las distintas tablas de la base de datos, cómo se muestra a continuación.

```
ConsumoHora.create(dispositivo_id: dispositivo.id, consumo: consumo, hora: today)


consumo_diario = ConsumoDiario.find_by(dispositivo_id: dispositivo.id, consume_date: today)

if consumo_diario.nil?
  consumo_diario = ConsumoDiario.create(dispositivo_id: dispositivo.id, consume_date: today, consume: consumo)
  consumo_diario_a = consumo_diario.consume
else
  consumo_diario_a = consumo_diario.consume + consumo
end
consumo_mensual = ConsumoMensual.find_by(dispositivo_id: dispositivo.id, mes_año: firs_day_of_month)
if consumo_mensual.nil?
  consumo_mensual = ConsumoMensual.create(dispositivo_id: dispositivo.id, mes_año: firs_day_of_month, consume: consumo);
  consumo_mensual_a = consumo_mensual.consume
else
  consumo_mensual_a = consumo_mensual.consume + consumo
end

consumo_total_a = dispositivo.total_consume + consumo
if consumo_diario.update(consume: consumo_diario_a) && consumo_mensual.update(consume: consumo_mensual_a) && dispositivo.update(total
  render json: "OK", status: 200
else
  render json: "No se ha podido actualizar el dispositivo", status: 404
end
```

Ilustración 64 Añadir un nuevo consumo

Cómo se puede apreciar para el registro por horas es necesario solamente crear un nuevo registro con la información obtenida. Para el resto de los tipos de consumo se debe comprobar primero que exista o no un registro de ellos, ya que se puede dar el caso en que se cambie de día o de mes y no haya registro en esas tablas. En el caso de que si exista, se debe aumentar el registro del consumo y la fecha última de cuando ha sido realizado.

7.3.4 Software de Aplicación Web

En este apartado no se comentarán de nuevo cosas que han sido mencionadas previamente como la configuración de rutas, se mostrará temas de paginación, medidas de seguridad y cierta funcionalidad extra debido al uso del calendario en los consumos.

7.3.4.1 Paginación

Para este proyecto no se ha optado por realizar una paginación desde cero, sino que se ha optado por usar una librería llamada Kaminari, la cual proporciona una manera sencilla de agregar a nuestra web un sistema configurable de paginación. Lo primero que es necesario es incluir la gema o librería en el Gemfile, mediante esta línea de código: “gem ‘kaminari’”. Tras ello vamos al método de indexación que genera automáticamente la herramienta en el controlador del dispositivo. Y finalmente añadimos la siguiente línea de código:

```
# GET /dispositivos
# GET /dispositivos.json
def index
  @dispositivos = Dispositivo.where(user_id: current_user.id).page(params[:page]).per(10)
end
```

Ilustración 65 Paginación en controlador

Dónde el último método “per” sirve para indicar el número de elementos que queremos por página. Tras modificar el controlador sólo será necesario añadir la paginación a la vista de la siguiente forma:

```
%tbody
  - @dispositivos.each do |dispositivo|
    %tr
      %td
        = link_to dispositivo.etiqueta, ("dispositivos/" + dispositivo.id.to_s)
      %td= dispositivo.state
      %td= dispositivo.total_consume
      %td= dispositivo.coeficiente
= paginate(@dispositivos)
```

Mediante ese método “paginate” y gracias la gema, el sistema generará automáticamente la formación de páginas según la configuración escogida.

7.3.4.2 Gestión de usuarios

En este apartado se indicará cómo se ha realizado la gestión de usuarios y cómo se relaciona esto con la edición, visualización y listado de dispositivos. Para empezar y como en el apartado anterior se ha optado por usar una gema para la gestión de usuarios llamada Devise, esta gema o librería proporciona una serie de funcionalidades como son: gestión de registro e inicio de sesión de usuarios, aportando controladores y vistas que pueden ser configuradas, además incluye una manera de obtener el usuario en cualquier momento desde cualquiera de los controladores del sistema, para así dar acceso o no a los usuarios según a los datos que quieran acceder o modificar. Para ello añade una variable denominada “current_user” que almacena la información del usuario actual, cómo su identificador y su email, por lo tanto podemos comprobar antes de mostrar cualquier información si el usuario actual es el que se encuentra vinculado con el dispositivo, del cual queremos obtener información.

7.3.4.3 Creación de gráficas

En este apartado se indicará cómo crear las gráficas del sistema, cómo en apartados anteriores se ha utilizado una gema para no crearlas desde cero, en este caso se trata de la gema “Chartkick” la cual proporciona distintas gráficas a generar dependiendo de los parámetros de

entrada que utilicemos. Pero antes de utilizarla es necesario al igual que las anteriores, añadirla al Gemfile.

Tras ella se seguirán una serie de pasos, en este ejemplo se muestra el registro de consumos diarios, aunque se realiza de la misma manera tanto para los registros por horas o mensuales. Los pasos a seguir son:

1. Accedemos al modelo del consumo diario y añadimos el siguiente código:

```
def self.last_month
  ConsumoDiario.group("consume_date")
  .order("consume_date").limit(30)
  .pluck("consume_date, sum(consume) as quantity_sum")
end
```

Ilustración 66 Configuración del modelo para las gráficas

1.1. Cómo podemos observar lo que realizamos en el modelo es lo siguiente:
   1.1.1. Agrupamos los usuarios por consumo
   1.1.2. A continuación los ordenamos por consumo
   1.1.3. Obtenemos los últimos 30 registros del sistema
   1.1.4. Finalmente a través del método "pluck" indicamos el valor de la x e y en la gráfica indicando además el consumo total como segundo parámetro.

2. Tras ello debemos configurar la vista, para ello seguiremos los siguientes pasos:

```
= line_chart @dispositivo.consumo_diarios.last_month,
  title: "Consumo diario ultimos 30 dias",
  label: "Consumo",
  xtitle: "Dia",
  ytitle: "Consumo"
```

2.1. Cómo podemos observar primero llamamos al método previamente creado en el modelo e indicamos el tipo de gráfica a utilizar, en nuestro caso la gráfica lineal.
2.2. Tras ello configuramos la gráfica:
   2.2.1. Indicamos el título de la gráfica
   2.2.2. El valor que mostraremos al pasar por encima de cada uno de los puntos
   2.2.3. El valor del eje x
   2.2.4. Finalmente el valor del eje y

Así podremos obtener una gráfica como la siguiente

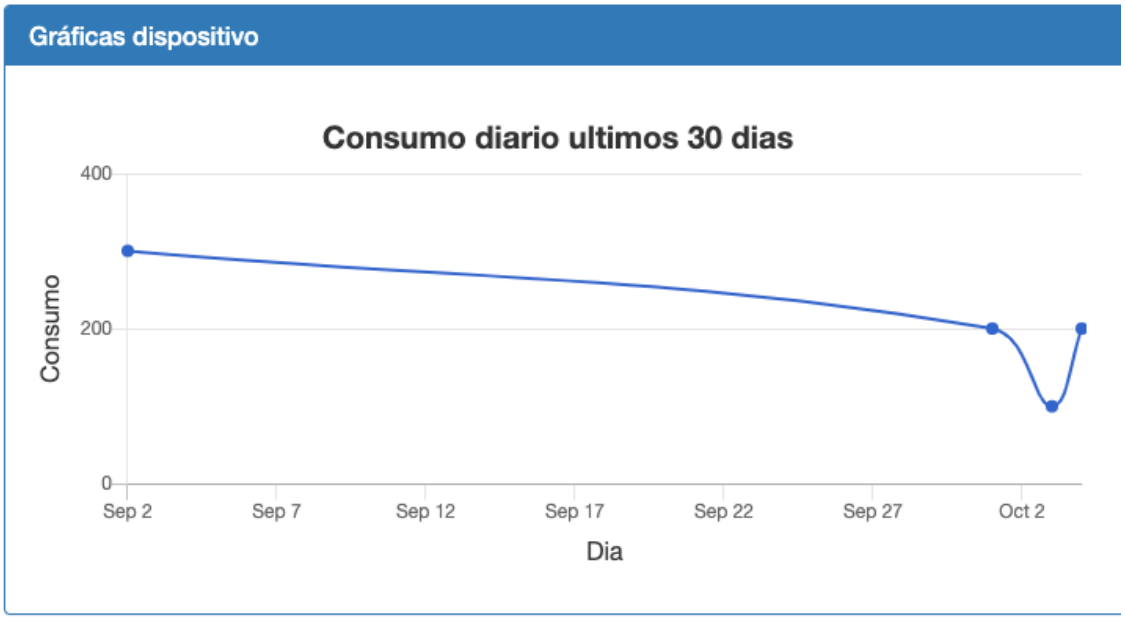


Ilustración 67 Gráfica registro diario

7.4 Problemas Encontrados

A continuación se irán nombrando todos los problemas encontrados a la hora de realizar el proyecto, todos ellos van relacionados con el dispositivo tanto a nivel de hardware cómo a nivel de software. Tras cada problema se proporcionará la acción llevada para solventarlo o cómo ha sido abordado.

7.4.1 Problema 1

Empezaremos a nivel de hardware con el problema a la hora de construir nuestro dispositivo en la placa, ya que dependiendo del voltaje de salida de nuestro dispositivo tendremos que modificar las resistencias en el sistema. Ya que no todos las placas Arduino tienen el mismo voltaje de salida. En nuestro caso, al trabajar con la placa MKR 1010 Wifi su salida de voltaje es de 3.3 V por lo tanto las resistencias a utilizar cambiarán si usamos una placa Arduino Uno cuya salida es de 5 V.

Para este problema ha sido necesario calcular el valor de las resistencias según su voltaje cómo se menciona en el documento en el apartado Resistencias de carga.

7.4.2 Problema 2

El primer problema que se encuentra a nivel de software va en relación con la librería Emonlib, ya que su configuración necesita un coeficiente para calcular y transformar correctamente la

lectura desde el sensor de corriente a la API REST. Dicho coeficiente varía con diversos factores como puede ser el voltaje de nuestra red, ya que el estándar en España es de 220 V, pero en otros países el voltaje en el hogar es de 230 V, además no se aplica el mismo coeficiente en dos dispositivos aunque estén conectados a la misma corriente, ya que para un dispositivo de 11 V su valor será distinto a uno de 650 V como puede ser una aspiradora.

Este problema ha generado crear un registro en la base de datos del sistema con el coeficiente específico para cada dispositivo que se añade. Además, se ha implementado una funcionalidad entre el dispositivo Arduino y la API REST por el cual, el dispositivo solicita la información correspondiente a su coeficiente cuando se enciende, y la API REST envía la información del coeficiente en base al dispositivo que la solicita.

7.4.3 Problema 3

El sensor de corriente no invasivo SCT-013, ya que aunque no sea invasivo es necesario modificar ya sea la regleta a la que va conectada el dispositivo a medir o el cable del propio dispositivo. Ya que este sensor mide el consumo en fase, esto quiere decir que si nosotros medimos directamente sobre el cable de un dispositivo, la medida obtenida será 0, ya que se anulan las corrientes entre ellas.

Para ello se ha modificado la regleta donde irá enchufado el dispositivo que queramos medir su consumo, de modo que el sensor SCT-013 podrá acceder al cable fásico de la regleta y medir el consumo.

7.4.4 Problema 4

La placa Arduino no consta de un reloj interno, por lo tanto es necesario añadir una librería, en este caso la librería Time, para que una vez conectado reciba una fecha y desde ahí empieza a contar. A su vez esto genera el problema de tener que implementar una manera de inicializar correctamente el tiempo cada vez que se reinicie o se conecte la placa a la corriente.

Al igual que en el problema 2, la solución ha sido implementar una funcionalidad en la API REST que envía la información del día actual junto la información del coeficiente del dispositivo, para que luego pueda ser tratada de forma correcta por el Arduino.

7.4.5 Problema 5

El software de Arduino no cuenta con pilas de almacenamiento, por lo que es necesario implementar un sistema similar a mano. Esto no es un gran problema, ya que su solución no es compleja, pero es algo que tener en cuenta.

Para este problema se ha creado un sistema de pila a mano, ya que la complejidad de este sistema de almacenamiento no es muy compleja de implementar.

Capítulo 8. Desarrollo de las Pruebas

8.1 Pruebas Unitarias

En esta sección indicaremos finalmente los datos utilizados para cada una de las pruebas descritas anteriormente, qué fallos fueron encontrados gracias a estas y que medidas fueron tomadas. Además debido a que el sistema se encuentra en tres módulos diferentes se realizará módulo a módulo.

Caso de uso 1: Configuración del dispositivo			
Prueba	**Resultado Esperado**	**Resultado obtenido**	**Corrección**
El usuario configura los parámetros de forma correcta	El dispositivo es capaz de conectarse al wifi correctamente, recibe la información de la API REST y envía tras el tiempo esperado la información correctamente	El dispositivo se conecta a la wifi deseada tras varios intentos.	Ninguna
El usuario configura erróneamente la wifi a la que se conecta	El dispositivo intenta conectarse a la wifi y realiza varios intentos, tras superar el tiempo el sistema muestra un mensaje de que no es posible conectarse al sistema	El dispositivo no se conecta a la wifi, y solicita reintroducir los datos en el sistema. El dispositivo continua con la ejecución del sistema erróneamente.	Corregir el código para que si no consigue conectarse a la red, no realice ninguna operación.
El servidor no se encuentra disponible	El dispositivo intenta recibir la información necesaria para realizar los cálculos del consumo, al no conseguirlos muestra un mensaje de error	El dispositivo no recibe la información, pero intenta calcular el consumo de igual manera. Lo que genera un error	Añadir un medio por el cual si no se ha recibido información se detenga el funcionamiento del dispositivo y se muestre un mensaje de error.

Caso de uso 2: Envío de información a la API REST			
Prueba	**Resultado Esperado**	**Resultado obtenido**	**Corrección**
El dispositivo se encuentra en el sistema	El dispositivo envía la información de consumo recogido y la envía a la API REST, ésta comprueba que el dispositivo existe en el sistema, registra la información y le envía un mensaje, el cual incluye que la información ha sido registrada.	El resultado obtenido es igual al esperado	Ninguna
El dispositivo no se encuentra en el sistema	El dispositivo envía la información a la API REST, la API comprueba que el dispositivo no existe en el sistema y tras ello le envía un mensaje de error	El resultado obtenido es el esperado	Ninguna
Error en el sistema	El dispositivo, que existe en el sistema, envía información de consumo a la API REST, tras comprobar que existe en el sistema ocurre un fallo al registrar los datos, la API REST enviará un mensaje de error al dispositivo.	El resultado obtenido es el esperado	Ninguna

Caso de uso 3: Añadir un dispositivo al sistema			
Prueba	**Resultado Esperado**	**Resultado obtenido**	**Corrección**
El usuario añade un dispositivo a su listado, existe en el sistema y no pertenece a otro usuario	El sistema comprueba que el dispositivo existe en el sistema, no está vinculado con otro usuario y tras registrar los datos lo añade correctamente.	El resultado obtenido es igual al esperado	Ninguna
El usuario intenta añadir un dispositivo que no se encuentra en el sistema	Un usuario intenta añadir un dispositivo que no se encuentra en el sistema, tras ser comprobado se le muestra un mensaje al usuario para que contacte con el administrador para añadir este dispositivo.	El resultado obtenido es el esperado	Ninguna
Un usuario intenta añadir un dispositivo que está vinculado con otro usuario	El sistema muestra un mensaje al usuario diciendo que ese dispositivo se encuentra	No se puede añadir el dispositivo, porque interpreta que el usuario por defecto es un usuario	Se configura, para que si el usuario del dispositivo es el usuario por defecto, se permita

	vinculado con otro usuario y que no es posible vincularlo con su cuenta.	más en el sistema	añadir dicho dispositivo al sistema.

Caso de uso 4: Registro de usuario en el sistema de visualización			
Prueba	**Resultado Esperado**	**Resultado obtenido**	**Corrección**
Un usuario que no está registrado en el sistema intenta registrarse	El sistema recoge los datos, y si estos son correctos, registra al usuario en el sistema.	El resultado obtenido es igual al esperado	Ninguna
Un usuario que no está registrado en el sistema intenta registrarse con datos erróneos	El sistema comprueba los datos y al no ser correctos, le muestra un mensaje al usuario de error y le reenvía a la página de registro.	El resultado obtenido es el esperado	Ninguna
Un usuario que se encuentra en el sistema intenta registrarse	El sistema comprueba que el usuario se encuentra en el sistema y le muestra un mensaje de error, indicando que ya se ha registrado.	El resultado obtenido es el esperado	Ninguna

Caso de uso 5: Inicio de sesión en el sistema			
Prueba	**Resultado Esperado**	**Resultado obtenido**	**Corrección**
Un usuario intenta iniciar sesión, estando en el sistema y con datos válidos	El sistema comprueba que el usuario se encuentra registrado y le muestra su pantalla de inicio.	El resultado obtenido es igual al esperado	Ninguna
Un usuario que no se encuentra registrado en el sistema intenta iniciar sesión	El sistema comprueba que el usuario no existe en el sistema, tras ello le muestra un mensaje de error indicando que no existe el usuario.	El resultado obtenido es el esperado	Ninguna
El usuario intenta iniciar sesión con datos erróneos	El sistema comprueba el usuario y la contraseña, si la contraseña no es correcta le muestra un mensaje de error al usuario y le reenvía a la página de inicio de sesión.	El resultado obtenido es el esperado	Ninguna

Caso de uso 6: Acceder a un dispositivo			
Prueba	**Resultado Esperado**	**Resultado obtenido**	**Corrección**
Un usuario intenta acceder a la información de uno de sus dispositivos	El sistema comprueba que el usuario del dispositivo es el mismo que está en sesión y tras ello le muestra la página de detalles del dispositivo.	El resultado obtenido es igual al esperado	Ninguna
Un usuario intenta acceder a un dispositivo que no está vinculado	El sistema le muestra un mensaje de error, y le reenvía a la página de inicio, indicando que el dispositivo no está vinculado con él.	El resultado obtenido es el esperado	Ninguna
Un usuario intenta acceder a un dispositivo que no se encuentra en el sistema	El sistema mostrará un mensaje de error indicando que no existe el dispositivo en el sistema. Y tras ello, reenviará al usuario a su página de inicio	El resultado obtenido es un fallo en el sistema.	Se elimina una condición en el sistema, que generaba que si un dispositivo no existía en el sistema mostrase la página de error de Rails. Ahora si el dispositivo no existe se reenvía al usuario a su página de inicio y se le muestra un mensaje de error.

Caso de uso 7: Editar un dispositivo			
Prueba	**Resultado Esperado**	**Resultado obtenido**	**Corrección**
Un usuario intenta editar un dispositivo con datos correctos	El sistema comprueba que los datos son correctos y que el usuario está vinculado al dispositivo y tras ello le muestra un mensaje diciendo que el dispositivo ha sido editado correctamente.	El resultado obtenido es igual al esperado	Ninguna
El usuario intenta editar un dispositivo con el cual no está vinculado	El sistema le muestra un mensaje de error al usuario y le reenvía a su página de inicio.	El resultado obtenido es el esperado	Ninguna

Caso de uso 8: Desvincular un dispositivo			
Prueba	**Resultado Esperado**	**Resultado obtenido**	**Corrección**
Un usuario intenta desvincular uno de sus dispositivos	El sistema comprueba que el usuario está vinculado con el dispositivo y tras ello elimina la desvinculación entre usuario y dispositivo, a	El resultado obtenido es igual al esperado	Ninguna

	continuación, el sistema reenvía al usuario a su página de inicio.		
El usuario intenta desvincular un dispositivo que no se encuentra en el sistema	El sistema comprueba que el dispositivo no existe y tras ello el sistema reenvía al usuario a su página de inicio, mostrando un mensaje de error	El sistema se para y se muestra la página de errores de Rails	Se elimina la condición para este método por el cual si no existe el objeto se interrumpa la ejecución del sistema. Y se corrige reenviando al usuario a su página principal e indicando que el dispositivo no existe.
El usuario intenta desvincular un dispositivo, con el cual no está vinculado	El sistema comprueba que el usuario del dispositivo no es el mismo que se encuentra en sesión y reenvía al usuario a su página de inicio mostrando un mensaje de error indicando que el dispositivo no está disponible.	Igual que el resultado esperado.	Ninguna

Caso de uso 9: Envío de información desde el dispositivo a la API REST caída			
Prueba	**Resultado Esperado**	**Resultado obtenido**	**Corrección**
El dispositivo intenta enviar información al servidor que se encuentra caído	Tras intentar enviar la información al servidor y ver que no ha sido posible el dispositivo almacena la información, para que cuando sea el tiempo de enviar información perdida vuelva a intentar lanzarse	El resultado obtenido es igual al esperado	Ninguna
El dispositivo intenta enviar información que previamente ha intentado ser enviada	El dispositivo intenta enviar información al servidor, que previamente había sido enviada, tras comprobar que la API REST aún se encuentra sin conexión el sistema lo sigue intentando con otras, pero no la vuelve a almacenar y evita duplicados	A excepción de la repetición de información el resultado obtenido es igual al esperado	Se amplía la funcionalidad, para que en el caso de que no se haya podido volver a enviar la información, ésta no se duplique en el listado de consumos no enviados

Caso de uso 9: Envío de información desde el dispositivo a la API REST caída			
Prueba	**Resultado Esperado**	**Resultado obtenido**	**Corrección**
El usuario intenta añadir un dispositivo nuevo al sistema que no se encuentra en la base de datos	El sistema comprueba que el dispositivo no se encuentra registrado en la base de datos y tras ello crea un nuevo registro en la base de datos.	El resultado obtenido es igual al esperado	Ninguna
El usuario intenta añadir un dispositivo nuevo al sistema que se encuentra en la base de datos	El sistema comprueba que ya hay un registro en la base de datos con esa información y envía una notificación al usuario, informando que el dispositivo ya se encuentra en la base de datos.	El resultado obtenido es igual al esperado.	Ninguna

8.2 Pruebas de Integración y del Sistema

Las pruebas de integración del sistema creadas anteriormente no generaron ningún problema a la hora de ponerlas en ejecución en el sistema y los valores obtenidos fueron iguales a los esperados. Por lo tanto no se ha tenido que modificar ninguna parte del código de los distintos módulos que generan el sistema.

Capítulo 9. Manuales del Sistema

9.1 Manual de Instalación

A continuación se mostrarán los pasos necesarios para la instalación del software necesario para el correcto funcionamiento del sistema. Estará dividido en tres fases: software del Arduino, la API REST y finalmente la aplicación web.

- Arduino: deberá descargarse desde el enlace de GitHub el archivo cew-arduino.ino, a continuación, conectamos la placa Arduino al ordenador donde tengamos el archivo descargado. Tras ello accedemos al Arduino IDE y subimos el fichero a la placa Arduino. Deberá ser necesario modificar ciertas variables en el sistema:
    - La SSID wifi y la contraseña de esta que se encuentra en el fichero arduino_secrets.h
    - La dirección IP o DNS donde se aloja la API REST y su puerto de entrada.
    - El identificador del dispositivo.
- API REST y aplicación Web: para la instalación de la API accederemos primero al servidor donde vayamos a lanzar dicho programa, tras ello instalaremos Ruby on Rails en el servidor, podemos encontrar la información en la página web oficial [8] a través de este enlace: http://rubyonrails.org.es/instala.html. Además, será necesario instalar PostgreSQL, a través de este enlace podemos obtener una guía de cómo instalar correctamente PostgreSQL[7].

9.2 Manual de Ejecución

Para este apartado no será necesario indicar nada del dispositivo de medición de consumo, ya que una vez conectado a la red, el dispositivo empieza a ejecutar el código. Por otra parte, la API REST y la aplicación web necesitan ser iniciadas de manera manual si el usuario decide usar un servidor personal, para ello debe seguir los siguientes pasos:

1. Accedemos a la carpeta CEW-WEB, e iniciamos el siguiente comando “rails s -b ‘IP o DNS del servidor -p ‘puerto donde queramos lanzar nuestra aplicación’”. Donde -b es el comando necesario para indicar la dirección IP donde está lanzada y -p el puerto a lanzar.
2. Accedemos a la carpeta CEW-API y realizamos los mismos pasos que en el paso anterior. Hay que tener en cuenta que el puerto en que se lancen ambas aplicaciones debe ser distinto, por ejemplo 3000 y 3001.

9.3 Manual de Usuario

En este apartado se indicará las acciones que puede hacer un usuario final en la aplicación web, ya que en las otras dos partes del proyecto este usuario no interviene.

9.3.1 Registro de usuario

El usuario accederá a la página web, tras ello verá un enlace que indica “Registrarse”, tras pinchar ahí aparece un formulario donde será necesario completar los datos email, contraseña y repetir la contraseña elegida.

Ilustración 68 Registro de usuario

9.3.2 Inicio de sesión

El usuario podrá iniciar sesión a través de su correo electrónico junto con la contraseña que hubiera puesto a la hora de registrarse. Tras ello podrá ver su panel con los dispositivos que tiene en su posesión.

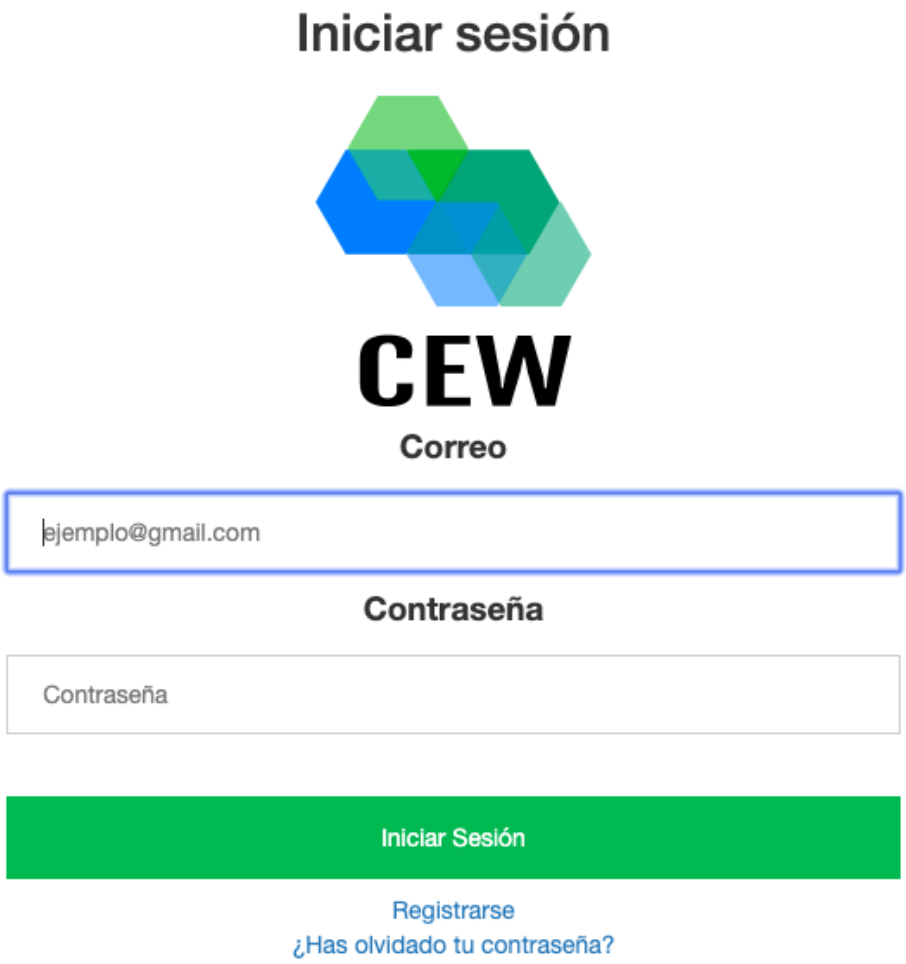
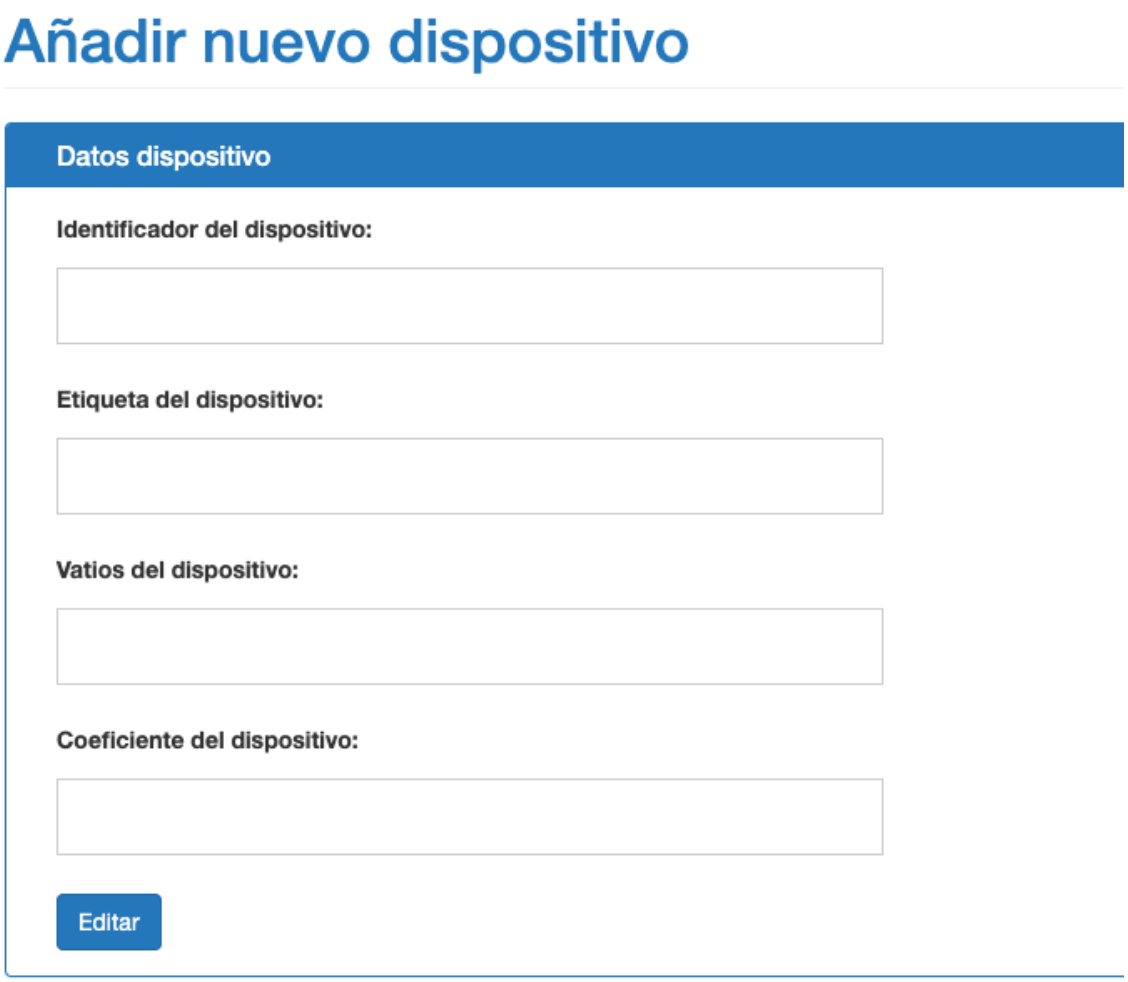


Ilustración 69 Inicio de sesión

9.3.3 Añadir un dispositivo

El usuario podrá añadir un nuevo dispositivo en el sistema, añadiendo los datos necesarios, los cuales son: identificador del dispositivo, etiqueta o nombre que le quiere dar al dispositivo, los vatios del dispositivo y finalmente el coeficiente necesario para la lectura de dicho dispositivo.



9.3.4 Ver detalles de un dispositivo

El usuario podrá ver los detalles de un dispositivo pinchando en uno de los que aparezcan en el listado que tiene en su posesión. Tras ello se le mostrará la información acerca del consumo actual del dispositivo y el consumo tanto del último mes y del último año.

Ilustración 70 Listado de dispositivos de un usuario

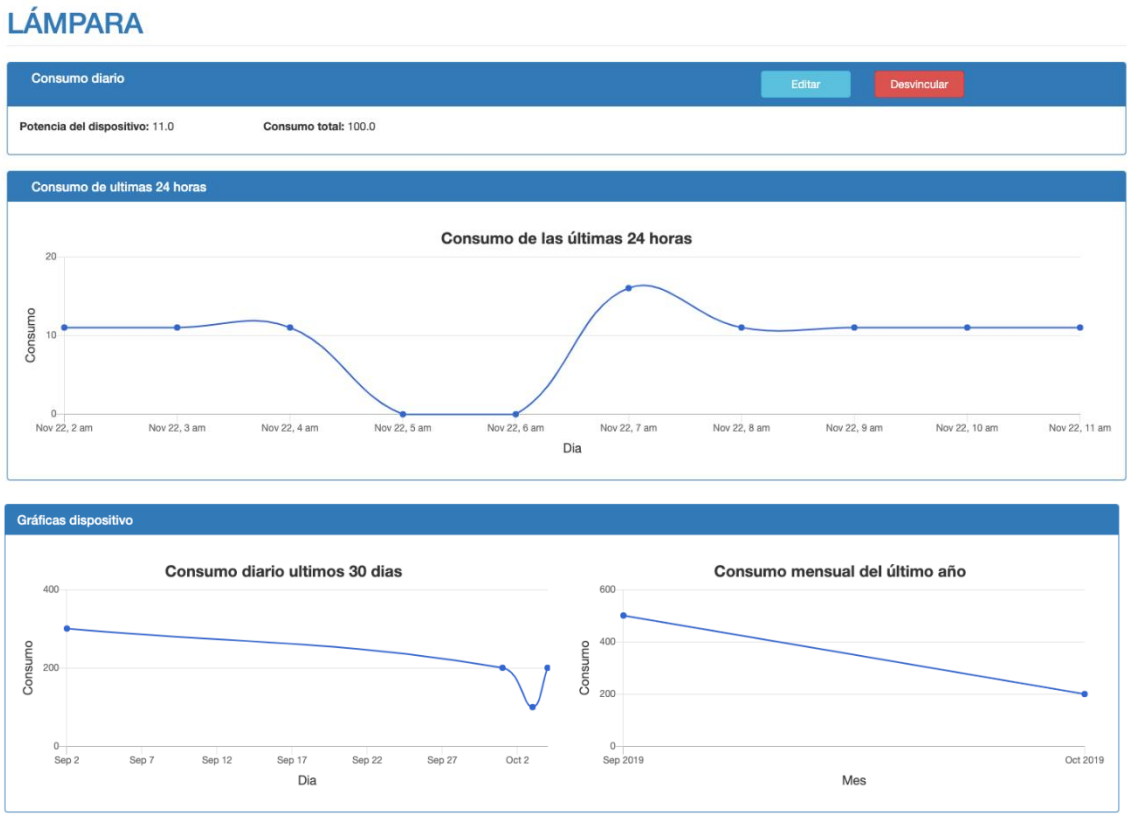


Ilustración 71 Detalles de un dispositivo

9.3.5 Editar datos de un dispositivo

En la ventana de detalles de un dispositivo, el usuario podrá editar cierta información pulsando el botón “Editar”.

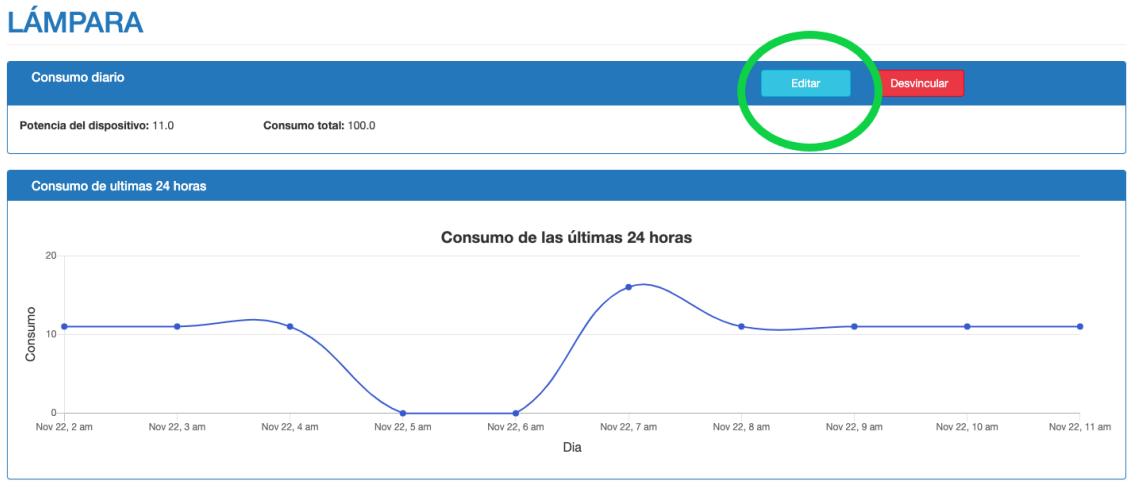


Ilustración 72 Pulsar “Editar” en los detalles del dispositivo

Tras ello aparecerá un formulario donde podrá modificar el nombre del dispositivo, los vatios y el coeficiente de dicho dispositivo.

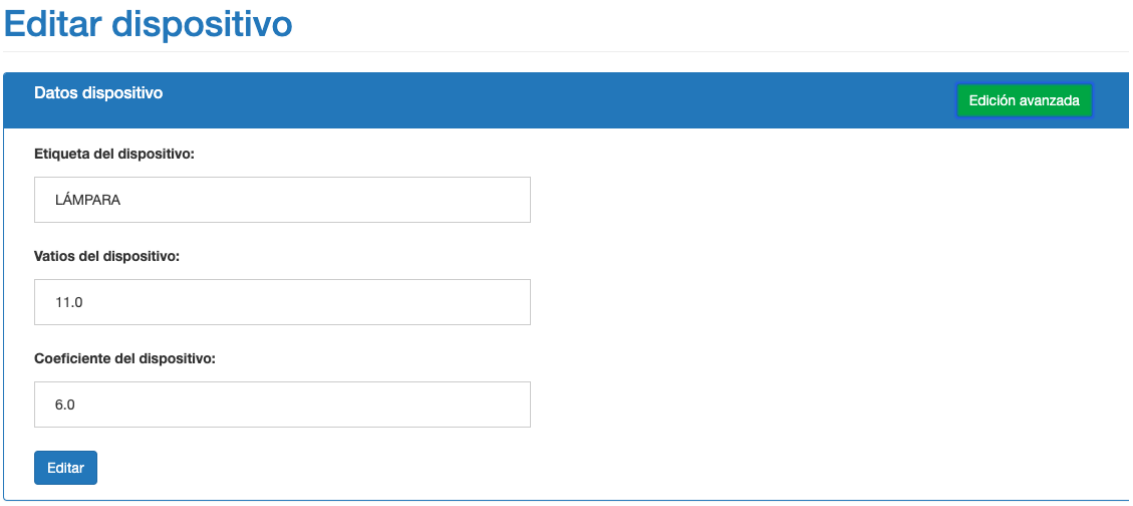


Ilustración 73 Editar dispositivo

Para mostrar el coeficiente del dispositivo deberá pulsarse en el botón "Edición avanzada", ya que puede ser que el usuario que lo está utilizando no necesite modificar este dato.

9.3.6 Desvincular un dispositivo

En la ventana de detalles de un dispositivo, pulsando el botón de "Desvincular", el usuario podrá desvincular un dispositivo que tenga en posesión, para ello se le mostrará una venta si desea o confirma la desvinculación con el dispositivo. Finalmente se le mostrará un mensaje que el dispositivo ha sido desvinculado y no se le mostrará en el listado de sus dispositivos.

Ilustración 74 Desvincular dispositivo

9.3.7 Desconectarse de la sesión

El usuario podrá desconectarse de la sesión si pulsa en la barra de navegación en "Desconectar". Esto hará que se finalice la sesión del usuario y se muestre la ventana de inicio de sesión.

Ilustración 75 Desconectarse de la sesión

9.4 Manual del Programador

En este apartado se incluirá información relevante para el programador, tanto para ampliar, modificar o editar ciertas partes del proyecto, en los distintos módulos del sistema.

- Ampliación de funcionalidad del dispositivo Arduino: a continuación se enumerarán una serie de ampliaciones o modificaciones que se pueden realizar al código de este módulo:
    - Aplicar encendido y apagado del dispositivo: se deberá incluir un nuevo sensor en el hardware del dispositivo para permitir pasar o no la corriente de la regleta. Pero a nivel de software se deberá incluir un sistema a través de una petición POST, la cual será enviada por el usuario a la API REST y está enviará la petición al dispositivo, por la cual el dispositivo envíe la opción de apagado o encendido al nuevo sensor. Para ello es necesario añadir un sistema de recepción de información en el bucle loop del software del Arduino para que una vez llegue la petición desde la API REST, éste realiza las operaciones necesarias.
- Ampliación de funcionalidad de la API REST:
    - Cómo en el primer apartado de ampliación de software del Arduino, será necesario crear el controlador oportuno para que reciba la petición desde la aplicación web de visualización de datos y envíe a continuación una petición al Arduino.
    - Se puede crear una nueva aplicación web o de escritorio que envíe peticiones a la API REST para añadir nuevos dispositivos que se vayan creados, y así puedan ser utilizados por lo usuarios en la aplicación de visualización de datos.

Capítulo 10. Conclusiones y Ampliaciones

10.1 Conclusiones

Tras realizar este proyecto, tanto en la parte hardware como en la parte software del sistema, me he dado cuenta la dificultad que supone por un lado que cómo de complicado es diseñar y construir un sistema IOT, ya que por un lado tenemos los componentes físicos, ligados al dispositivo Arduino, para lo cual es necesario tener unos conocimientos básicos de electrónica para entender el por qué se usan cada uno de los componentes y que función realizan. Por otro lado es necesario tener conocimientos de cómo se trabaja con una placa Arduino y las limitaciones que tienen estos dispositivos. A su vez, ha sido gratificante poder construir desde cero un dispositivo de estas características con la poca información que existe acerca de los sensores SCT-013 y cómo implementar esto en un hogar con poco coste.

Además del dispositivo Arduino, la construcción de la parte software cómo es la API REST y la aplicación web, han supuesto una complejidad añadida al implementarlo en un lenguaje que no se ha visto en la universidad. Ya que esto supone aprender un lenguaje de programación nuevo y cómo se pueden construir y utilizar los elementos que te proporciona. Pero debido a la extensa comunidad de Ruby on Rails que existe, junto con las librerías que ofrece este framework el aprendizaje y construcción del sistema ha sido buena.

Ha sido gratificante poder haber realizado el dispositivo físicamente y aumentar los conocimientos respecto a un nuevo lenguaje como es Ruby y en especial al framework Ruby on Rails y también cómo usar el dispositivo Arduino de manera más avanzada. Ya que durante la carrera no he dado por ejemplo el lenguaje de programación Ruby y tampoco vi ciertos aspectos del desarrollo del dispositivo Arduino.

Para concluir, realizando este proyecto me he dado cuenta de que existen muchas ramas dentro de la informática, más de la vistas en la universidad, pero que teniendo un buena base aprendida, ampliar los conocimientos para realizar este tipo de proyectos es mucho más sencillo y ameno.

10.2 Ampliaciones

Este proyecto tiene diversas ampliaciones que no se han podido implementar, ya sea por falta de recursos, tiempo o conocimiento, éste último necesitaría de más tiempo para aprender ciertos procesos que elevarían las horas del proyecto. A continuación, se muestra un listado de las ampliaciones:

- Identificación automática de placas Arduino: actualmente el identificador del dispositivo deberá se añadido a mano por el usuario. En una futura ampliación el dispositivo al ser cargado su software generará un identificador único.
- Disminución del espacio del dispositivo: por falta de recursos no fue posible crear en una placa PCB personalizada el dispositivo y fue montado sobre una placa Protoboard. Esto genera que el dispositivo sea de un tamaño mayor del que sería si se tuviera dicha placa.
- Una interfaz para añadir de manera más sencilla el wifi al dispositivo, sin tener que acceder al archivo de configuración de este apartado. Para ello habría que buscar la manera de realizar la interfaz, web o móvil, y cómo enviar dicha información al sistema.
- Calculo automático de coeficiente: respecto con la ampliación anterior al crear todos los dispositivos con los mismos materiales, sólo será necesario saber el voltaje del dispositivo a medir y mediante cálculos se cree un coeficiente válido para la correcta medición del dispositivo.
- Creación de una aplicación web o de escritorio que se comunique con la API REST para la gestión del sistema en los registros de los dispositivos que se vayan creando. Además se puede ampliar la funcionalidad de la API REST para que lleve a cabo más funcionalidades de gestión.
- Mejora en la interfaz web: actualmente el proyecto está centrado en el dispositivo y en la API REST, y la interfaz web sirve como visualización de datos básicos, por lo tanto, una ampliación es mejorar o ampliar funcionalidades de la web.
- Finalmente, cómo ampliación de este proyecto sería transformar la regleta para que añadiendo un sensor extra, el usuario puede cortar y dar corriente a la regleta para proporcionar energía a los dispositivos eléctricos conectados a ella.

Capítulo 11. Presupuesto

11.1 Desarrollo de Presupuesto detallado (Cliente)

A continuación, se muestra el presupuesto detallado, dónde encontramos el coste tanto del hardware, cómo de las fases de análisis y desarrollo desplegado en tareas. Hay que tener en cuenta que se ha simulado 5 trabajadores, cada cual con distintos precios, por lo tanto el importe por tarea es del conjunto de los trabajadores, además la columna del número de horas será el total de horas trabajadas en total que se ha dedicado a la tarea por los distintos trabajadores. Para ver en detalle los trabajadores en cada tarea se puede consultar el WBS, que se ofrece con el resto de documentación.

Presupuesto del cliente detallado					
Código	Concepto	Horas	Importe (€)	Total (€)	Total + IVA (€)
CDA0	Análisis			5.729,59 €	6.932,80 €
CDA1	Estudio de componentes	20	960,00 €		
CDA2	Estudio del software Arduino	16	768,00 €		
CDA3	Estudio de Aplicación Rest en Ralis	12	576,00 €		
CDA4	Estudio de construcción del dispositivo	24	768,00 €		
CDA5	Compra de materiales	4	408,00 €		
CDA6	Adquisición de materiales	28	1.512,00 €		
CDD0	Desarrollo			25.756,25 €	31.165,06 €
CDD1	Construcción del dispositivo	20	288,00 €		
CDD2	Creación y despliegue de la base de datos	20	1.248,00 €		
CDD3	Implementación de Emonlib	8	499,20 €		
CDD4	Implementación de Wifinina	24	1.497,60 €		
CDD5	Implementación de la librería Time	8	499,20 €		
CDD6	Creación del controlador de peticiones	28	1.747,20 €		
CDD7	Creación de controladores Web	40	2.496,00 €		
CDD8	Creación de vistas	32	1.996,80 €		
CDD9	Realizar integración de módulos	24	4.291,20 €		
CDD10	Realización de pruebas	12	1.742,40 €		
CDD11	Corrección de errores	8	230,40 €		
CDD12	Revisión	8	1.200,00 €		

CDD13	Implantación del software en servidor	20	3.576,00 €		
CDD14	Creación de manual de usuario	24	3.600,00 €		
HW00	Hardware			5.541,65 €	6.705,39 €
HW01	SCT-013		1.405,20 €		
HW02	Arduino mkr-1010 wifi		3.348,00 €		
HW03	Placa PCB		225,60 €		
HW04	Condensador electrolítico de aluminio		9,60 €		
HW05	Resistencias		12,00 €		
HW06	Cables de soldadura		1,25 €		
HW07	Raspberry (Servidor)		60,00 €		
HW08	Regleta (personalizada)		480,00 €		
			Total	37.027,49 €	44.803,26 €

Capítulo 12. Referencias Bibliográficas

[1]Equisoain, D. L. (2016). *Arduino práctico.* Madrid: Anaya Multimedia.

[2]Upton, E. (2016). *Raspberry Pi user guide.* Chichester: J. Wiley.

[3]Valle, L. d. (23 de Octubre de 2019). *SCT-013 mide el consumo eléctrico en tu casa con Arduino*. Obtenido de Programar fácil con Arduino: https://programarfacil.com/blog/arduino-blog/sct-013-consumo-electrico-arduino/

[4]Varios. (2013). *Ruby on Rails.* Mexico: Alfaomega Grupo Editor.

[5]Varios. (24 de 12 de 2019). *Arduino - WiFiNINA*. Obtenido de Arduino.cc: https://www.arduino.cc/en/Reference/WiFiNINA

[6]Varios. (29 de Diciembre de 2019). *heartcombo/devise*. Obtenido de Github: https://github.com/heartcombo/devise

[7]Varios. (10 de Octubre de 2019). *Install PostgresSQL*. Obtenido de Postgresqltutorial.com: https://www.postgresqltutorial.com/install-postgresql/

[8]Varios. (21 de Octubre de 2019). *Ruby on Rails - ¿Cómo instalar Ruby on Rails?* Obtenido de Rubyonrails.org.es: http://rubyonrails.org.es/instala.html

Capítulo 13. Apéndices

13.1 Glosario y Diccionario de Datos

- **Amperio:** es la unidad internacional para la medición de corriente eléctrica.
- **HTTP:** es un protocolo de transferencia de hipertextos.
- **IDE:** es una aplicación que genera un entorno de desarrollo software, en el cual se incluyen librerías o herramientas para facilitar dicho desarrollo.
- **IOT:** del inglés "Internet Of Things", traducido al español Internet de las cosas, son sistemas que se comunican entre sí e interaccionan entre ellos para cumplir un objetivo.
- **Onda senoidal:** curva que representa el valor de la tensión de la corriente alterna.
- **Placa PCB:** Placa de Circuito Impreso (en inglés "Printed Circuit Board") es un elemento utilizado en electrónica donde se instalan o sueldan los distintos componentes del dispositivo final.
- **Placa Protoboard:** es una placa de pruebas donde podemos insertar cables, resistencias, sensores y otros dispositivos para conectarlos entre ellos.
- **REST:** es un tipo de arquitectura utilizada para construir interfaces entre distintos sistemas que se basen en HTTP.
- **Sensor:** dispositivo con el cual se obtiene un resultado de una acción, temperatura, presión u otro tipo de datos y la transmite.
- **Voltio:** es la unidad internacional para la medición del potencial eléctrico.

13.2 Contenido Entregado en el Archivo adjunto

13.2.1 Contenidos

13.2.1.1.1 Introducción

En estos apartados se describirá el conjunto de directorios que forman el proyecto y el contenido de este que se encuentra en él. A partir del directorio raíz se accede a tres directorios importantes, que son el del software del dispositivo Arduino, el directorio de la API REST y finalmente el directorio de la aplicación web.

13.2.1.1.2 Recomendación estructura general directorios del Archivo adjunto

Directorio	Contenido
./ Directorio raíz del Archivo adjunto	Contiene un fichero REEDME en el cual se describe cómo esta divida la estructura de

	ficheros.
./cew-arduino	Contiene el conjunto de archivos necesarios para el dispositivo Arduino, que deberán ser cargados cómo se indica en el manual de instalación.
./CEW-API	Este fichero pertenece a la API REST del proyecto y dentro encontramos los archivos necesarios para su despliegue. Como es el Gemfile y el Gemfile.lock, necesarios para importar las librerías o gemas a utilizar.
./CEW-API/app	Contiene el código principal del programa, las vistas, los modelos y los controladores implementados del sistema.
./ ./CEW-API/app/assets	Contiene los fichero de JavaScript del proyecto.
./CEW-API/app/controllers	Se encuentran los controladores de la API REST.
./CEW-API/app/models	Contiene los modelos (objetos) de la aplicación que serán utilizados por los controladores
./CEW-API/config	Contiene ficheros de configuración del sistema cómo propiedades del servidor, el tipo de base de datos utilizados, idioma por defecto de la aplicación…
./CEW-API/db	Contiene las migraciones con la base de datos, la semilla o script para cargar datos por defecto y la estructura o esquema de la base de datos.
./CEW-WEB	Directorio donde se encuentran las carpetas y archivos necesarios para la ejecución de la aplicación web.
./CEW-WEB/app	Al igual que en la otra aplicación se encuentran las vistas, modelos y controladores del sistema.
./CEW-WEB/app/assets	En este proyecto a parte del JavaScript, se hayan las imágenes como el logo y los CSS utilizados, en este caso el CSS de Bootstrap
./CEW-WEB/app/controllers	Directorio donde se almacenan los controladores de la aplicación.
./CEW-WEB/app/models	En este directorio se encuentran los modelos u objetos del sistema.
./CEW-WEB/app/views	Finalmente encontramos en este directorio las vistas utilizadas en la aplicación.
./CEW-WEB/config	Directorio con información de configuración del sistema similar a la de CEW-API.
./CEW-WEB/db	Similar a CEW-API.

13.2.2 Código Ejecutable e Instalación

En esta sección se hará un breve resumen del manual de instalación que se encuentra previamente con los pasos para la puesta en funcionamiento del proyecto.

- Dispositivo Arduino: deberemos configurar con nuestros datos las variables que se indican en el fichero léeme.
- API REST y aplicación web: no es necesario modificar ningún fichero y para ejecutarlo deberemos escoger un puerto donde lanzar nuestra aplicación y decidir si lanzarlo en local o por medio de un DNS. Recordad que la dirección IP de ambas aplicaciones puede ser la misma, pero es necesario lanzar la aplicación en puertos distintos, ya que el propio sistema no permitirá lanzar ambas aplicaciones en el mismo.

13.2.3 Ficheros de Configuración

A continuación se mostrará un listado con los ficheros que son necesarios modificar, junto con las variables a modificar dentro de cada una de las partes.

Aplicación	Archivo	Variable	Valor
CEW-Arduino	arduino_secrets.h	SECRET_SSID	El nombre de la wifi a la que el usuario vaya a conectarse.
CEW-Arduino	arduino_secrets.h	SECRET_PASS	La contraseña del wifi a conectarse
CEW-Arduino	arduino_secrets.h	SERVER	Dirección DNS o IP dónde se encuentra lanzada la aplicación API REST
CEW-Arduino	arduino_secrets.h	PUERTO	Puerto dónde se encuentre la aplicación API REST
CEW-Arduino	arduino_secrets.h	VOLTAJE_RED	Voltaje que tenga nuestra red eléctrica. Por defecto será de 220, pero puede ser modificada a otro valor.
CEW-Arduino	arduino_secrets.h	ID_ARDUINO	Identificador que usará el dispositivo para identificarse con la API REST
CEW-Arduino	arduino_secrets.h	CONT_HORAS	Es la variable que nos permitirá configurar cada cuanto tiempo queremos enviar información, por defecto está en una hora
CEW-Arduino	arduino_secrets.h	CONT_HORAS_PERDIDAS	Es la variable que identifica cuándo queremos que se envíen los consumos que no han sido enviados por fallo en el servidor, por defecto está en media hora.