Trabajo de Fin de Grado en Ingeniería Informática

– Julio 2014 –

Diseño e implementación de un prototipo de enchufe inteligente mediante la placa Arduino

Alumno: *Óscar Antonio González de Chaves Pérez*

Tutor: *Dr. Alexis Quesada Arencibia / Escuela de Ingeniería Informática*

UNIVERSIDAD DE LAS PALMAS DE GRAN CANARIA
Escuela de Ingeniería Informática

A mi abuelo Paco.

Agradecimientos

"A hombros de gigantes" sea tal vez una de las frases más célebres de nuestra historia. Atribuida al gigante Isaac Newton, su origen real se cree en boca de Bernardo de Chartres. Él decía:

> "Somos enanos encaramados a hombros de gigantes. De esta manera, vemos más y más lejos que ellos, no porque nuestra vista sea más aguda sino porque ellos nos sostienen en el aire y nos elevan con toda su altura gigantesca."[1]

Esta frase, sin género de duda, tiene ganado su espacio en los anales de la historia. Nosotros no podemos más que compartir su significado y pensar en todos aquellos gigantes que nos han elevado.

Aprovecho esta oportunidad que se me ofrece para agradecer el tiempo, la dedicación y el esfuerzo entregado por cada persona que me ha acompañado a lo largo de mi vida. A quienes estuvieron conmigo en cada etapa, en el colegio, el instituto, la universidad y el trabajo.

Gracias de forma especial a mi tutor Alexis Quesada Arencibia, y al Instituto Universitario de Ciencias y Tecnologías Cibernéticas, sin quienes este trabajo no habría sido posible.

Gracias por supuesto a mis compañeros y amigos, que con su compañía han iluminado tantos momentos felices.

Agradezco el apoyo incondicional recibido siempre por parte de mi familia. Gracias a mi padre por haberme inspirado la honradez, a mi madre por haberme inculcado el carácter y la sensibilidad. Gracias a mi abuelo Paco por haber dejado en mí su huella indeleble, a mi querida abuela María Luisa, a mis tías y hermanas…

Por último, acabo recordando a mi persona más especial, mi compañera y amiga, la persona que se acuesta y amanece conmigo cada mañana, que me conoce y aún así me quiere. Gracias, Marta.

[1] Juan e Salisbury, siglo XII

Resumen

Este trabajo detalla las diferentes etapas del desarrollo de construcción del prototipo Clever Socket, abarcando desde su análisis, a su diseño e implementación.

Clever Socket es un conjunto hardware y software que facilita el control del encendido y el apagado de los aparatos electrónicos conectados a él a través de redes de comunicación LAN y WAN. Para lograr este propósito, el artefacto se apoya en la placa de desarrollo electrónico Arduino UNO y la placa Arduino WiFi Shield –que posibilita su conexión inalámbrica a la red–.

El dispositivo cuenta con cuatro bases de enchufe controladas a través de relés, que permiten la gestión de hasta cuatro dispositivos electrónicos de manera simultánea.

En el lado del software se presentan tres componentes:

Un servicio web de bajo nivel destinado a la gestión de la placa Arduino. Permite la comunicación con la placa, la lectura y escritura de sus pines.

Un servicio web específico del dispositivo Clever Socket. Facilita la comunicación con el prototipo hardware, permitiendo así la gestión de diferentes operaciones que se llevan a cabo sobre los aparatos conectados a sus enchufes.

Una aplicación web. Actúa como interfaz entre el usuario y el servicio web de Clever Socket, permitiendo la gestión del dispositivo. El diseño de la aplicación es *responsive*, lo que posibilita su correcta visualización en todo tipo de dispositivos, tanto móviles como de escritorio.

Abstract

This essay details the different building development stages of the Clever Socket prototype, covering different subjects, such as its analysis, design and implementation.

Clever Socket is a hardware and software system that makes possible the remote on and off control of all its electronic devices connected through, enabling the management via LAN and WAN communication networks. In order to achieve that goal, the device relies on the electronic development Arduino UNO board together with the Arduino WiFi Shield (which allows the network wireless communication).

The device has four sockets controlled by a group of relays which enable the management of up to four electronic devices in a simultaneous way.

On the software side, the system is divided into three components:

A low level web service focus on the Arduino board management. It allows the communication with the board and enables the reading and writing of its pins.

A web services oriented to the Clever Socket device. It provides a communication link with the hardware prototype, making possible the management of different operations pointing to the device's sockets.

A web application. It plays the user interface's role, offering a link with the Clever Socket web service and allowing the device control. The application has a responsive design, making possible an optimal viewing experience in all kind of devices, from moviles to desktops equipments.

Índice de contenidos

Índice de ilustraciones

Índice de tablas

1 Estructura del documento

En la redacción del presente documento se ha intentado seguir un orden lógico y lineal, que haga fácil su lectura y entendimiento.

A lo largo del trabajo se introducen términos que, en el caso de ser técnicos, intentan ser explicados para poder llegar a cualquier tipo de audiencia. Desde la aportación de referencias bibliográficas a la opción de incluir abundante número de ilustraciones, que permitan entender de forma rápida y directa el texto que acompañan.

Se aprecia una división en diferentes apartados y subapartados que intentan dar coherencia y marcar un camino por el que resulte agradable transitar.

A continuación resumimos los principales temas abordados y que encontrará el lector durante su lectura:

- **Presentación y estado del arte:** Introducimos el trabajo y ahondamos en el estado actual. Explicamos de forma breve el origen de la idea y mostramos algunos ejemplos de proyectos similares existentes en el mercado.
- **Justificación, objetivos y propuesta de valor:** En estas líneas vamos a justificar el motivo de nuestro trabajo. Hablaremos sobre los objetivos que nos marcamos, e intentaremos ofrecer nuestra visión sobre la propuesta de valor que vamos a ofrecer.
- **Competencias y metodología:** En el apartado de las competencias, vamos a detenernos en aquellas competencias que vamos a trabajar y desarrollar con nuestro trabajo. Explicaremos además, cuál ha sido nuestro plan de actuación y de desarrollo. Qué formas de actuar nos hemos propuesto y cómo pensamos llevarlas a cabo.
- **Análisis y pruebas de concepto:** Con el análisis pretendemos estudiar la viabilidad del proyecto, las diferentes opciones existentes y la mejor manera de llevar adelante nuestro trabajo. Además, introduciremos algunas pruebas que reforzarán el resultado del estudio.
- **Diseño y recursos:** En esta etapa, superada la fase del análisis, nos sumergimos en la proyección del trabajo, en la manera en que va a ser implementado. Aportando el mayor número posible de detalles sobre la forma en que la idea ha de convertirse en realidad. Además, introduciremos los recursos que vamos a necesitar.
- **Implementación:** La implementación mostrará de forma específica la puesta en práctica del trabajo anterior. La manera en que se va a implementar y desplegar el *software*. La forma en que vamos a proceder para realizar el montaje del prototipo *hardware*.
- **Normativa y legislación:** En estas líneas nos vamos a preocupar de atender a las normas y la legislación existente en relación con nuestro trabajo. De cómo afecta a

nuestro trabajo y cómo debemos proceder para estar seguros de que atendemos a su cumplimiento.

- **Proyección de futuro y conclusiones:** Después de haber transitado el camino que nos ha traído hasta aquí, estaremos en disposición de apuntar formas de mejora, y pensamientos sobre futuras modificaciones que puedan engrandecer nuestro trabajo. Además, reflexionaremos sobre qué ha significado para nosotros la tarea realizada.
- **Fuentes de información y anexos:** No nos olvidaremos de aportar las fuentes de información de las cuales hemos bebido durante nuestro camino. Incluyendo al final, aquellos contenidos que por su extensión, aun siendo relevantes, hemos considerado oportuno desplazar a este punto, intentando facilitar así la lectura del documento y reducir en lo posible su complejidad.

2 Estructura del CD

Este documento viene acompañado de un CD. La estructura y el contenido digital del disco son las siguientes:

- **EII-GII-2014-07-01-TFT-GonzalezDeChavesPerez_O-QuesadaArencibia_A-DIPEIMPA.pdf:** Respaldo digital en formato PDF de este documento.
- **resumen.txt:** Resumen del trabajo en español.
- **abstract.txt:** Resumen del trabajo en inglés.
- **source:**
    - **arduino-clever-socket:** Código fuente del Servicio Web de Arduino.
    - **api-clever-socket:** Código fuente del Servicio Web de Clever Socket.
    - **web-clever-socket:** Código fuente de la Aplicación Web de Clever Socket.

3 Introducción

Este trabajo de fin de grado (en adelante, "TFG") aborda el análisis, el diseño y la implementación de un prototipo de regleta de enchufes denominado Clever Socket. Su gobierno es llevado a cabo de forma remota a través de una red inalámbrica de tipo LAN o WAN y de un conjunto *software*.

El tándem hardware y software ofrece el control sobre las fases de encendido y apagado de un grupo de enchufes. Para ello, hace uso entre otros componentes electrónicos, de la placa de desarrollo electrónico Arduino UNO y su módulo de expansión Arduino WiFi Shield.

La gestión sobre el control del dispositivo es realizada a través de una aplicación web, que posibilita la selección del enchufe objeto de control y permite su encendido, apagado y conmutado.

El conjunto de funcionalidades y procedimientos ofrecidos por el dispositivo están gestionados a través de un servicio web *ad hoc* de tipo RESTful, disponible para cualquier sistema *software* de terceros que desee comunicarse con el aparato.

4 Estado del arte

En la actualidad, existen diversos dispositivos que nos permiten conectar y desconectar nuestros aparatos de la red de suministro eléctrico. Desde sencillas regletas de enchufes con interruptores para su encendido y apagados, enchufes con sistemas de programación, o artefactos provistos de control remoto que facilitan el encendido o apagado de nuestros equipos eléctricos.

Con el incremento de los precios en el consumo de la electricidad, cada vez son más los ciudadanos que se preocupan por ejercer un uso controlado del suministro eléctrico, así como de los distintos aparatos del hogar que se conectan a la red. Por este motivo, son diversos los fabricantes que han introducido en el mercado dispositivos para el control del consumo eléctrico, facilitando la tarea de ejercer un uso comedido y responsable de la electricidad.

En el campo de las regletas de enchufes, encontramos artefactos que además de ofrecer interruptores para permitir el encendido y apagado de los conectores, ofrecen también otro tipo de opciones y funcionalidades, como la conexión de conectores RJ45 y RJ11, USB, conectores para cables de antena de televisión, etc., así como protección contra picos de tensión eléctrica que los dispositivos enchufados a estos conectores pueden sufrir.

Se muestran a continuación algunas imágenes de diversos sistemas de regleta de enchufes existentes en el mercado:

Ilustración 1: Base de enchufe individual

Ilustración 2: Múltiples bases de enchufe

Ilustración 3: Múltiples bases de enchufe con protección contra picos de tensión

En el lado de los sistemas de enchufes programables también existe diversidad en la oferta. Desde conectores con temporizadores analógicos muy sencillos (ilustración 1), a sistemas digitales más sofisticados y que permiten una amplia variedad de programas para la temporización (ilustración 2) o lectura de sensores (ilustración 3).

Véanse a continuación algunas imágenes representativas de este tipo de enchufes:

Ilustración 4: Temporizador analógico

Ilustración 5: Temporizador digital

Ilustración 6: Temporizador digital con sensor de temperatura

Por último, el mercado ofrece también sistemas de enchufes que ofrecen su conmutación a través de dispositivos de control remoto. Habitualmente este tipo de aparatos operan a través de radio frecuencia. Pero también han aparecido recientemente algunos enchufes que pueden ser controlados a través de la conexión WiFi en el hogar. Algunos de estos dispositivos nos permiten incluso llevar un control sobre el consumo eléctrico de los aparatos conectados a ellos.

Algunos ejemplos de estos aparatos:

Ilustración 7: Enchufe controlado a través de radio frecuencia

Ilustración 8: Enchufe controlado a través de WiFi

Ilustración 9: Enchufe con control de consumo eléctrico

Los precios de todos estos dispositivos van en relación al nivel de complejidad y de funcionalidades ofrecidos. Desde precios muy asequibles para sistemas sencillos para el control del encendido y el apagado de sus enchufes, a precios bastante más elevados para otros sistemas de mayor complejidad tecnológica y con mayor número de funcionalidades y capacidades.

Debido al auge y a la popularización de la electrónica, los precios de los distintos componentes electrónicos ha ido disminuyendo. Cada vez resulta más habitual encontrar componentes y dispositivos electrónicos con mayores prestaciones y funcionalidades, sin que por ello se vea afectado el bolsillo del consumidor.

Si atendemos al campo de la electrónica de consumo, observamos cómo en los últimos años, han ido apareciendo componentes electrónicos dotados de cierta inteligencia que permiten a usuarios intermedios y *amateurs*, llevar a cabo proyectos de gestión y control electrónico cada vez más avanzados. De forma particular, hablamos de sistemas sofisticados como la popular placa Raspberry Pi (ilustración 10), o la más sencilla –pero también potente en prestaciones– placa Arduino UNO (ilustración 11).

La placa Raspberry Pi es un dispositivo que en sí mismo, es un pequeño ordenador. Decimos pequeño ordenador, aunque dentro de sí encierre la potencia de equipos informáticos mucho más aparatosos hace unos años atrás. Además, a esto hay que añadir la ridícula horquilla existente en su relación prestaciones/precio.

Ilustración 10: Raspberry Pi

Ilustración 11: Arduino UNO

Por otro lado, en los últimos tiempos, se ha ido incrementando el número de trabajos electrónicos llevados a cabo con un pequeño dispositivo, la placa Arduino. Arduino es un proyecto liderado por un pequeño equipo de desarrolladores italianos que actualmente es conocido a nivel mundial. Sus prestaciones, su reducido tamaño y su ridículo precio han hecho que se popularice en tiempo récord. Además, a todo ello, hay que añadirle el hecho de la existencia de *shields* o complementos, que acopladas a la placa, multiplican de forma exponencial las funcionalidades y prestaciones del equipo.

Hoy día cualquier usuario medio, puede llevar a cabo proyectos de gran complejidad técnica y envergadura, sobre todo atendiendo a dicha complejidad desde el escalón de la vanguardia tecnológica de hace unos pocos años atrás.

5 Justificación

Aunque existan ya, como hemos visto, una gran cantidad de dispositivos que nos ayudan a controlar el consumo y a ejercer un uso más sostenible y responsable de la electricidad, observamos todavía la oportunidad de ofrecer una propuesta capaz de mejorar la oferta actual, o al menos así lo pensamos. A continuación intentamos razonar y plantear los motivos por los cuales creemos justificable el desarrollo de este trabajo:

- Los equipos actuales de bajo costo son en la actualidad muy limitados en prestaciones.
- Aquellos dispositivos con múltiples funcionalidades todavía hoy continúan teniendo un precio elevado que los aleja del consumidor medio.
- Aunque se ha empezado a explorar el terreno del control de conectores de forma remota –incluso haciendo uso de la tecnología WiFi– todavía es poco convencional encontrar equipos que se conecten a través de redes locales o Internet.
- La opción que planteamos tiene un coste razonable, que podría reducirse de forma significativa una vez superada la etapa de prototipado.
- Vamos a ofrecer funcionalidades avanzadas de una manera que para el usuario sea fácil de gestionar.

- Al estar conectado a internet, ofrecemos la posibilidad de no cerrarse exclusivamente al control de los dispositivos desde el hogar, sino también a través de la distancia, desde cualquier punto conectado a través de Internet.
- Ofrecemos un API que permite el crecimiento futuro del proyecto, permitiendo que otros sistemas *software* se aprovechen de la pasarela para hacer uso de los procedimientos de encendido y apagado de los enchufes de la regleta.

6 Objetivos generales y soluciones propuestas

En un primer momento, a la hora de plantearnos la idea de este proyecto, pensamos en qué objetivos necesitaría cubrir nuestro trabajo para satisfacer las necesidades y demandas del usuario. El perfil de persona a quién va dirigida nuestra propuesta es el del ciudadano medio, comprometido con el uso eficiente de las fuentes de energía de consumo actuales, el del sujeto preocupado por el medio ambiente, por la economía doméstica y por la forma en cómo la tecnología pueda ayudar dando respuesta a estas demandas.

Pensando en lo anterior, definimos y nombramos a continuación, qué objetivos generales se impone este proyecto como propósitos y metas a conquistar:

- Ofrecer una herramienta de uso sencillo y accesible.
- Gestionar las funcionalidades del sistema desde equipos de sobremesa hasta terminales móviles.
- Desarrollar una propuesta económicamente viable.
- Definir y acotar el número de funcionalidades; plantear un producto en el que primen la eficacia y necesidad de las soluciones.
- Aprovechar la tecnología existente para reducir la complejidad y tiempo de desarrollo.
- Atender a la problemática de la seguridad en las comunicaciones.
- Maximizar la flexibilidad del producto, pensar en su escalabilidad, y dejar abierto el producto para una posible evolución.
- Conseguir tiempos de respuesta cortos en las peticiones de los usuarios.
- Diseñar un producto energéticamente eficiente.
- Entregar un prototipo funcional.

Definidas de forma concisa las metas, pasamos a la siguiente fase, pensar en las soluciones necesarios para alcanzar los objetivos.

Estas fueron las soluciones planteadas para dar respuesta a nuestros objetivos:

- Desarrollar una interfaz de usuario clara y sencilla.

- Ofrecer una solución que permita su ejecución desde cualquier sistema operativo con navegador web.
- Hacer uso de hardware económico, basando la idea en el uso de la infraestructura ofrecida por la plataforma Arduino.
- Plantear funcionalidades básicas, como encender o apagar un interruptor.
- Seguridad.
- Entregar un servicio web que permita la comunicación con terceros.
- Desarrollar un código limpio, que permita su comprensión y fácil modificación.
- Evitar funcionalidades innecesarias o redundantes que ralenticen la respuesta final del sistema.
- Diseñar un esquema electrónico con pocos componentes.
- Realizar un montaje sencillo que pueda ser llevado a cabo en poco tiempo.

7 Aportaciones

De la realización de este trabajo, han surgido en paralelo diversos desarrollos. Desde el pequeño servicio web para la tarjeta Arduino, al servicio web orientado al dispositivo Clever Socket, a la propia aplicación que explota este último servicio.

Hemos aportado valor con el desarrollo del servicio web para Arduino. Aún siendo conscientes de que queda mucho camino por recorrer en este apartado, creemos estar pisando un camino poco transitado hasta el momento. Durante nuestra etapa de análisis, no encontramos un solución ad hoc que permitiera establecer una comunicación para conocer y cambiar el estado de los pines de la tarjeta. Un servicio web de tipo RESTful que facilitara el desarrollo de aplicaciones en torno al ecosistema Arduino. Funcionalidad básica, que en nuestra opinión, no se ha explotado de forma suficiente hasta el momento.

En este sentido, gracias al servicio web específico de Clever Socket, abrimos una ventana a futuros desarrollos. Con este canal de comunicación, será posible en el futuro llevar a cabo nuevos proyectos que solamente tengan la preocupación de comunicarse con el servicio. Posibilitando que terceras partes hagan provecho de nuestro desarrollo.

Creemos que nuestro producto final, tal cual se entrega, ofrece valor. Ya en su etapa de prototipo es capaz de llevar a cabo su funcionalidad principal, permitir y cortar el suministro eléctrico de los aparatos conectados al dispositivo a través de una red LAN o WAN.

Gracias a nuestros dispositivo, se logra deslocalizar el interruptor de encendido y apagado de cualquier aparato eléctrico. Coloca este interruptor en nuestros bolsillos a través de los teléfonos móviles inteligentes, aprovecha en definitiva las ventajas que ofrecen los sistemas automáticos y la inteligencia del *software* desarrollado sobre ellos.

Con todo, creemos que nuestra principal aportación ha sido plantear la idea. El punto de inicio sobre el que continuar desarrollando futuros proyectos que lleguen mucho más lejos.

8 Competencias desarrolladas

Las tareas de investigación, análisis, diseño e implementación llevadas a cabo para la realización de este trabajo han permitido que hayamos desarrollado y cubierto las siguientes competencias propuestas a alcanzar por la asignatura de Trabajo de Fin de Grado para el curso 2013-2014:

- G1. Poseer y comprender conocimientos en un área de estudio (Ingeniería Informática) que parte de la base de la educación secundaria general, y se suele encontrar a un nivel que, si bien se apoya en libros de texto avanzados, incluye también algunos aspectos que implican conocimientos procedentes de la vanguardia de su campo de estudio.
- G2. Aplicar sus conocimientos a su trabajo o vocación de una forma profesional y posean las competencias que suelen demostrarse por medio de la elaboración y defensa de argumentos y la resolución de problemas dentro de su área de estudio.
    - Justificación G1. y G2.

      Se plasma en el documento y en nuestra labor, una metodología de trabajo propia de la Ingeniería Informática. Los pasos dados, así como las soluciones planteadas manifiestan la necesidad del uso de los conocimientos y capacidades propios de nuestro área de conocimiento.

- G3. Reunir e interpretar datos relevantes (normalmente dentro de su área de estudio) para emitir juicios que incluyan una reflexión sobre temas relevantes de índole social, científica o ética.
    - Justificación G3.

      Hemos realizado un análisis del estado del arte, así como de las necesidades reales a día de hoy y de la oferta existente.

- G4. Transmitir información, ideas, problemas y soluciones a un público tanto especializado como no especializado.
    - Justificación G4.

      Tanto en la elaboración de este documento, como en el resultado de nuestro trabajo, se ha hecho el esfuerzo por aunar la sencillez y la precisión en el lenguaje, utilizando los términos más adecuados en cada momento.

- G5. Desarrollar aquellas habilidades de aprendizaje necesarias para emprender estudios posteriores con un alto grado de autonomía.

- Justificación G5.

  El estudio de las herramientas y soluciones alcanzadas a lo largo de este trabajo nos han ayudado a adquirir conocimientos y habilidades que harán posible una mayor eficiencia y autonomía en tareas de grado similar o superior en el futuro.

- N1. Comunicarse de forma adecuada y respetuosa con diferentes audiencias (clientes, colaboradores, promotores, agentes sociales, etc.), utilizando los soportes y vías de comunicación más apropiados (especialmente las nuevas tecnologías de la información y la comunicación) de modo que pueda llegar a comprender los intereses, necesidades y preocupaciones de las personas y organizaciones, así como expresar claramente el sentido de la misión que tiene encomendada y la forma en que puede contribuir, con sus competencias y conocimientos profesionales, a la satisfacción de esos intereses, necesidades y preocupaciones.
  - Justificación N1.

    Tanto en la redacción de este documento, como en la preparación de la presentación oral de este trabajo, se ha desarrollado la competencia a través de la búsqueda de formas de comunicación y transmisión de ideas y conceptos cercanas y accesibles para el público objetivo, tanto profano como ilustrado en la materia.

- N2. Cooperar con otras personas y organizaciones en la realización eficaz de funciones y tareas propias de su perfil profesional, desarrollando una actitud reflexiva sobre sus propias competencias y conocimientos profesionales y una actitud comprensiva y empática hacia las competencias y conocimientos de otros profesionales.
  - Justificación N2.

    A la hora de reunir información y durante el proceso de elaboración de nuestro trabajo, hemos procurado consultar opinión experta en los momentos que estimamos oportuno, antes de tomar decisiones trascendentes que afectasen al desarrollo del trabajo.

- N3. Contribuir a la mejora continua de su profesión así como de las organizaciones en las que desarrolla sus prácticas a través de la participación activa en procesos de investigación, desarrollo e innovación.
  - Justificación N3.

    Hemos plasmado nuestro interés en mejorar el ejercicio de nuestra profesión y en favorecer la colaboración con otros grupos de investigación a través del trabajo conjunto llevado a cabo con algunos de los miembros del Instituto Universitario de Ciencias y Tecnologías Cibernéticas.

- N4. Comprometerse activamente en el desarrollo de prácticas profesionales respetuosas con los derechos humanos así como con las normas éticas propias de su ámbito profesional para generar confianza en los beneficiarios de su profesión y obtener la legitimidad y la autoridad que la sociedad le reconoce.
    - Justificación N4.

      A través del ejercicio profesional de nuestra actividad y de la realización de nuestra propuesta, hemos procurado reportar un beneficio comunitario, ofreciendo una solución que favorezca el uso comedido y razonado de un bien común, como es la energía eléctrica.

- T1. Capacidad para concebir, redactar, organizar, planificar, desarrollar y firmar proyectos en el ámbito de la ingeniería en informática que tengan por objeto, de acuerdo con los conocimientos adquiridos según lo establecido en apartado 5 de la resolución indicada, la concepción, el desarrollo o la explotación de sistemas, servicios y aplicaciones informáticas.
- T2. Capacidad para dirigir las actividades objeto de los proyectos del ámbito de la informática, de acuerdo con los conocimientos adquiridos según lo establecido en apartado 5 de la resolución indicada.
- T3. Capacidad para diseñar, desarrollar, evaluar y asegurar la accesibilidad, ergonomía, usabilidad y seguridad de los sistemas, servicios y aplicaciones informáticas, así como de la información que gestionan.
- T5. Capacidad para concebir, desarrollar y mantener sistemas, servicios y aplicaciones
- informáticas empleando los métodos de la ingeniería del software como instrumento para el aseguramiento de su calidad, de acuerdo con los conocimientos adquiridos según lo establecido en apartado 5 de la resolución indicada.
- T6. Capacidad para concebir y desarrollar sistemas o arquitecturas informáticas centralizadas o distribuidas integrando hardware, software y redes, de acuerdo con los conocimientos adquiridos según lo establecido en apartado 5 de la resolución indicada.
    - Justificación T1., T2., T3., T5. y T6.

      Véase la justificación de las competencias G1. y G2.

- T7. Capacidad para conocer, comprender y aplicar la legislación necesaria durante el desarrollo de la profesión de Ingeniero Técnico en Informática y manejar especificaciones, reglamentos y normas de obligado cumplimiento.
    - Justificación T7.

      Durante el desarrollo de este trabajo se ha procurado en todo momento atender a las recomendaciones y exigencias establecidas por los reglamentos, normas y legislación vigentes.

Véase también la justificación N4.

- T8. Conocimiento de las materias básicas y tecnologías, que capaciten para el aprendizaje y desarrollo de nuevos métodos y tecnologías, así como las que les doten de una gran versatilidad para adaptarse a nuevas situaciones.
    - Justificación T8.
      Véase justificación G1. y G2.

- T11. Capacidad para analizar y valorar el impacto social y medioambiental de las soluciones técnicas, comprendiendo la responsabilidad ética y profesional de la actividad del Ingeniero Técnico en Informática.
    - Justificación T11.
      Véase justificación N4.

- CII01. Capacidad para diseñar, desarrollar, seleccionar y evaluar aplicaciones y sistemas informáticos, asegurando su fiabilidad, seguridad y calidad, conforme a principios éticos y a la legislación y normativa vigente.
- CII02. Capacidad para planificar, concebir, desplegar y dirigir proyectos, servicios y sistemas informáticos en todos los ámbitos, liderando su puesta en marcha y su mejora continua y valorando su impacto económico y social.
- CII04. Capacidad para elaborar el pliego de condiciones técnicas de una instalación informática que cumpla los estándares y normativas vigentes.
- CII18. Conocimiento de la normativa y la regulación de la informática en los ámbitos nacional, europeo e internacional.
    - Justificación CII01., CII02., CII04. y CII18.
      Véase justificación G1., G2., N4. y T7.

- TFG01. Ejercicio original a realizar individualmente y presentar y defender ante un tribunal universitario, consistente en un proyecto en el ámbito de las tecnologías específicas de la Ingeniería en Informática de naturaleza profesional en el que se sinteticen e integren las competencias adquiridas en las enseñanzas.
    - Justificación TFG01.
      Véase justificación N1.

9 Metodología

En el desarrollo de este trabajo, hemos tenido presente la importancia de la correcta identificación y elección de la metodología.

Previo a la introducción sobre el tipo de filosofía elegida para la planificación y ejecución de las diferentes etapas de este proyecto, nos hemos preguntado acerca de qué entendemos por metodología.

Una definición muy general, nos dice que una metodología es un conjunto de métodos que se siguen en una investigación científica, un estudio o una exposición doctrinal.

Por método entendemos el procedimiento llevado a cabo con el propósito de alcanzar un fin.

Hemos de distinguir que la metodología parte de un origen teórico y que trae consigo la elección de métodos procedimentales para la realización del conjunto de tareas asociadas a la fase de investigación y ejecución de un trabajo.

En lo que concierne a este proyecto, hemos de distinguir dos aspectos fundamentales a valorar: por un lado lo que se refiere al *software,* y por el otro, al *hardware*.

La visión metodológica clásica del desarrollo de *software*, ha intentado acercar el proceso de desarrollo, a los métodos utilizados de manera tradicional en el mundo de la ingeniería; de ahí el auge y la acuñación del término "Ingeniería del Software".

Durante muchos años se han llevado a cabo procesos de desarrollo de *software* aplicando métodos propios de las ingenierías. Sin embargo, de un tiempo a esta parte, y motivado fundamentalmente por los altos índices de fracaso obtenidos, ha ido surgiendo una nueva corriente metodológica en el desarrollo de *software*, el agilismo.

9.1 Metodología ágil

Son múltiples los motivos esgrimidos y los indicadores que apuntan a las causas por las cuales la aplicación de las técnicas y procesos propios de las metodologías clásicas no funcionan de manera eficiente y eficaz en el desarrollo de *software*.

No es propósito de este trabajo ahondar en las causas y motivos de este problema. Pero sí queremos justificar el hecho por el cuál hemos decidido apuntar nuestro proceso metodológico hacia el agilismo, tanto en el desarrollo del *software*, como en el del *hardware*.

El germen y la causa de nuestra elección, se encuentra justificado fundamentalmente en los principios mencionados en el texto *Manifesto for Agile Software Development* firmado por Kent Beck y varios críticos de modelos de mejora de desarrollo de software basados en procesos.

A continuación valoramos y analizamos algunas de sus doctrinas y razonamos su apropiación a nuestro trabajo:

“Software funcionando sobre documentación extensiva”

Durante el desarrollo de nuestro trabajo, hemos valorado la opción de priorizar la entrega de valor y funcionalidad con respecto al desarrollo de documentación muy elaborada. Ello nos ha permitido realizar ciclos cortos de desarrollo en los que se ha ido aportando valor de manera continuada y sostenible, empezando por lo concreto y creciendo hacia lo general.

Además, hemos tenido la oportunidad de analizar de forma temprana la adecuación del producto obtenido, así como también de reajustar objetivos y restricciones iniciales en las distintas etapas.

“Colaboración con el cliente sobre negociación contractual”

Este principio centra el foco en la atención a la relación con el cliente, prevaleciendo sobre las negociaciones contractuales. Creemos que esto es un acierto, debido a que la condiciones son cambiantes, los puntos de vista del cliente pueden ser unos al comienzo del proyecto, y conforme se va realizando, pueden cambiar. No solo esto, además es importante ratificar que se están interpretando de manera correcta los deseos del cliente.

En nuestro caso, y debido a la singularidad del trabajo, nosotros somos nuestro propio cliente y ello nos ha facilitado en gran medida nuestra tarea. No únicamente porque conozcamos y entendamos bien los requisitos del proyecto, sino también porque podemos eliminar completamente la barrera existente entre cliente y desarrollador, y lo que es también importante, entre responsable de negocio y desarrollador.

“Respuesta ante el cambio sobre seguir un plan”

La experiencia nos ha demostrado que gracias en buena parte al seguimiento de este principio, hemos podido comprobar que el trabajo cumple con las expectativas, y en los casos en que no se han cumplido, hemos tenido la oportunidad de realizar las correcciones necesarias en tiempo.

Ello no quiere decir, que no hayamos seguido un plan, o que no hayamos trazado una línea maestra por donde andar, pero sí quiere decir que nuestro norte ha ido cambiando durante el recorrido, y sobre todo se ha ido adaptando a los cambios que se han ido produciendo a lo largo de nuestro camino.

Como mencionábamos anteriormente, es propio de las metodologías ágiles poner el foco del desarrollo en ir de lo concreto a lo general. En este sentido, tanto en el apartado concerniente al *hardware* como al *software* hemos intentado seguir la máxima en cada momento. Para ello nos hemos asegurado de realizar componentes nucleares que se han ido convirtiendo en partes de elementos más complejos, tanto en lo estructural como en lo funcional.

9.2 Buenas prácticas de programación

Durante la fase de programación, se han querido seguir algunos consejos y buenas prácticas identificados por expertos en el desarrollo de *software*.

En el desarrollo de nuestro código hemos aplicado los fundamentos de la programación orientada a objetos. En este sentido, se han intentado respetar 5 principios clave –identificados por el estudioso en métodos de desarrollo de *software* Robert C. Martin– que han sido popularizados bajo el acrónimo SOLID:

1. Principio de responsabilidad.
2. Abierto a la ampliación pero cerrado a la modificación.
3. Principio de sustitución de Liskov.
4. Principio de segregación de interfaces.
5. El principio de inversión de la dependencia.

Otro aspecto muy relevante en el desarrollo de software que hemos tenido en cuenta, es el principio SoC (*separation of concerns*), por el que se respeta la separación de las partes en base a su funcionalidad. Como veremos más adelante –en el apartado del desarrollo– se ha hecho uso de este principio de forma muy clara con el uso del *framework* de PHP Laravel en la parte de la aplicación web.

Por último, destacar el interés que hemos mostrado en la aplicación de los consejos sugeridos en el libro "*Clean Code*"[2] del autor citado anteriormente Robert C. Martin. Desde su publicación, esta obra se ha popularizado entre los miembros de la comunidad de programadores, hecho que ha traído consigo la generalización de términos como "*clean code*" –código que refleja las buenas cualidades descritas por el autor–, o "*code smell*" –por el contrario, código que incumple las cualidades anteriores–.

Mencionamos algunas de las cualidades descritas en el libro y que nos han guiado a lo largo del desarrollo de nuestro código:

"El código limpio no hace demasiadas cosas, el código limpio es enfocado"

"El código limpio no usa rodeos ni soluciones ofuscadas"

"El código limpio no es redundante"

---

[2] Robert C. Martin, 2008

“El código limpio es placentero de leer”

“El código limpio puede ser modificado fácilmente por cualquier otro desarrollador”

“El código limpio debe tener dependencias mínimas”

“El código limpio es pequeño”

“El código limpio es expresivo”

“El código limpio tiene módulos de prueba”

Existen otras muchas recomendaciones que a lo largo del tiempo se han identificando como buenas prácticas y que también hemos tenido en cuenta durante nuestro trabajo, como por ejemplo el uso del versionado del código a través de la utilización del software de control de versiones GIT.

10 Recursos

10.1 Materiales

Para la construcción de nuestro prototipo, hemos hecho uso y necesitado de los siguientes materiales:

- 1 x Arduino UNO Rev3
- 1 x Arduino WiFi Shield SD
- 1 x Módulo 4 relés 5V
- 8 x *Jumper* de conexión para placa *protoboard*, macho a hembra
- 1 x Alimentador electrónico universal 9V, 2ª
- 1 x Caja estanca con conos, 220x170x85mm
- 2 x Base enchufe doble, 16ª, 250V
- 1 x Regleta de conexión
- 1 x Clavija enchufe, patas de 4,8mm, 16ª, 250V
- 1 x Interruptor conmutador luminoso rojo de dos circuitos
- 4 x terminal plano hembra 6,4mm, aislado, azul 2,5mm
- 1 x Diodo LED 5mm, 5V, azul, con cable de 18cm
- 1 x Termorretráctil corto
- 24 x Arandela plana, 3mm
- 24 x Tornillo DIN965 cabeza plana
- 24 x Tuerca 934

- 1 x Cable manguera 3x1,5mm blanco, 4,5m
- 1 x Tablero contrachapado 600x300x5mm

Veremos a continuación una pequeña descripción sobre cuáles son las características más destacables de los componentes clave del desarrollo del trabajo.

10.1.1 Arduino UNO Rev3

La placa Arduino UNO es un microcontrolador basado en el ATmega328. Tiene 14 pines de entrada/salida –de los cuales 6 de ellos pueden ser utilizados como salidas PWM (Pulse With Modulation)–, 6 entradas analógicas, un conector USB, un conector tipo *jack*, un conector ICSP (*In Circuit Serial Programming*) y un pulsador de reinicio. Dispone de todo lo necesario para dar soporte al microcontrolador ATmega328; requiere únicamente conectar la placa a un ordenador mediante el puerto USB o una fuente de alimentación al conector tipo *jack*.

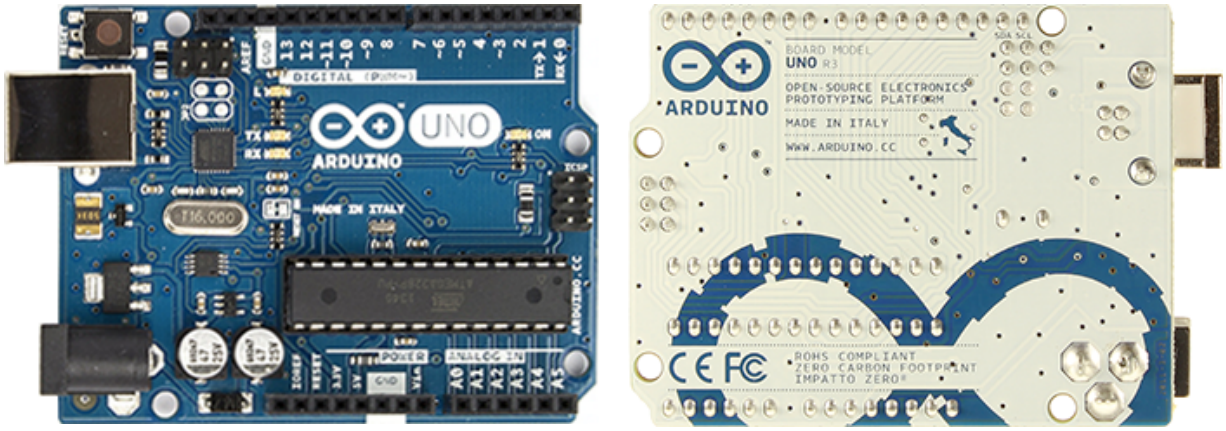


Ilustración 12: Frontal y reverso de la placa Arduino UNO

Características técnicas

Microcontrolador	ATmega328
Voltaje operativo	5V
Voltaje de entrada recomendado	7-12V
Voltaje de entrada soportado	6-20V
Pines digitales de entrada/salida	14 (6 de ellos pueden ser salidas de PWM)
Pines de entrada analógica	6
Intensidad en DC por pin entrada/salida	40mA

Intensidad en DC por pin de 3,3V	50mA
Memoria *flash*	32KB (ATmega328), 0,5KB utilizados por el *bootloader* (gestor de arranque)
SRAM	2KB (ATmega328)
EEPROM	1KB (ATmega328)
Velocidad de reloj	16MHz

Tabla 1: Características técnicas Arduino UNO

Conexiones

Hemos visto que la Arduino UNO dispone de 6 entradas analógicas. Los pines de estas entradas se encuentran serigrafiados en la placa, en un rango que va desde la entrada A0 a la A1. Estos pines ofrecen una resolución de 10 bits, o lo que es lo mismo, 1024 valores posibles. Estos pines están pensados para ser utilizados como entrada de datos, pudiendo acceder a la lectura de sus valores a través de la función `analogRead()`.

Por otro lado, disponemos de 14 pines digitales, serigrafiados en el rango que va de 0 a 13. Pero además, algunos de estos pines –concretamente el 3, 5, 6, 9, 10 y 11– pueden ser utilizados en modo de salida PWM a través de la función `analogWrite()`. Pero, ¿si son pines digitales, por qué se llama a la función `analogWrite()` y no a la función `digitalWrite()`? Esto tiene sentido si entendemos qué es una salida PWM.

La modulación por ancho de pulsos –conocida también como PWN– es un ingenio que a través de la variación del ciclo de trabajo de una señal periódica logra emular en algunos casos el comportamiento de una señal analógica. De modo que se hace posible, por ejemplo, efectuar un efecto de atenuación o *fading* de un diodo LED.

Vemos en el gráfico, además, pines destinados a la alimentación. Tenemos pines a tierra (GND), pines a 3,3 y 5 V, e incluso un pin de alimentación de voltaje variante (Vin), que cambia su salida en función de la fuente de alimentación que se conecte a la placa a través del conector de alimentación tipo *jack*.

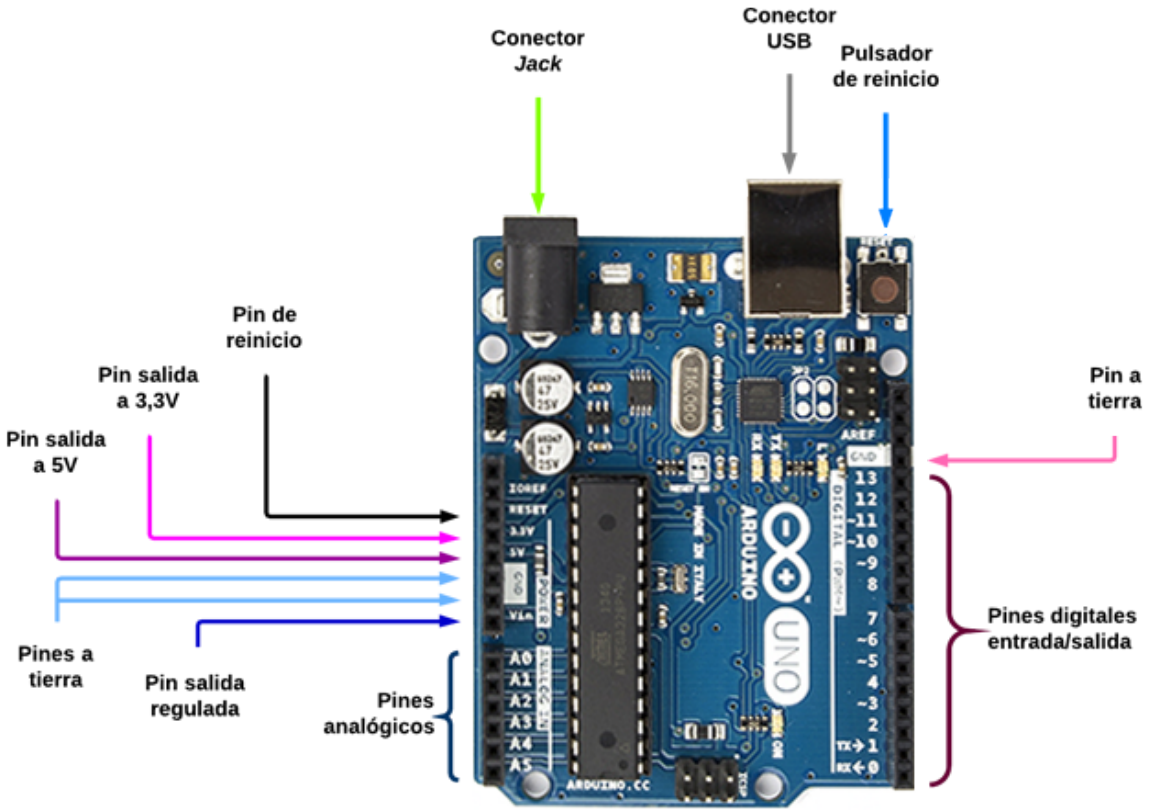


Ilustración 13: Conexiones Arduino UNO

Existen otros pines –como por ejemplo el pin de reinicio– cuya descripción detallada no es objeto de este trabajo. (Para mayor información, puede ser consultado el sitio web http://arduino.cc/en/Main/ArduinoBoardUno)

10.1.2 Arduino WiFi Shield SD

El Arduino WiFi Shield SD es otro de los componentes fundamentales de nuestro prototipo. Esta extensión –acoplada de forma directa a la placa Arduino UNO– nos va a permitir conectar nuestra placa a internet de forma inalámbrica.

La última versión de este accesorio de Arduino está basado en el circuito encapsulado –de consuma ultrabajo– HDG204, que ofrece la conexión Wireless LAN 802.11b/g. Además de un microcontrolador Atmel 32UC3 que posibilita la comunicación con la placa a través de los protocolos de red UDP y TCP.

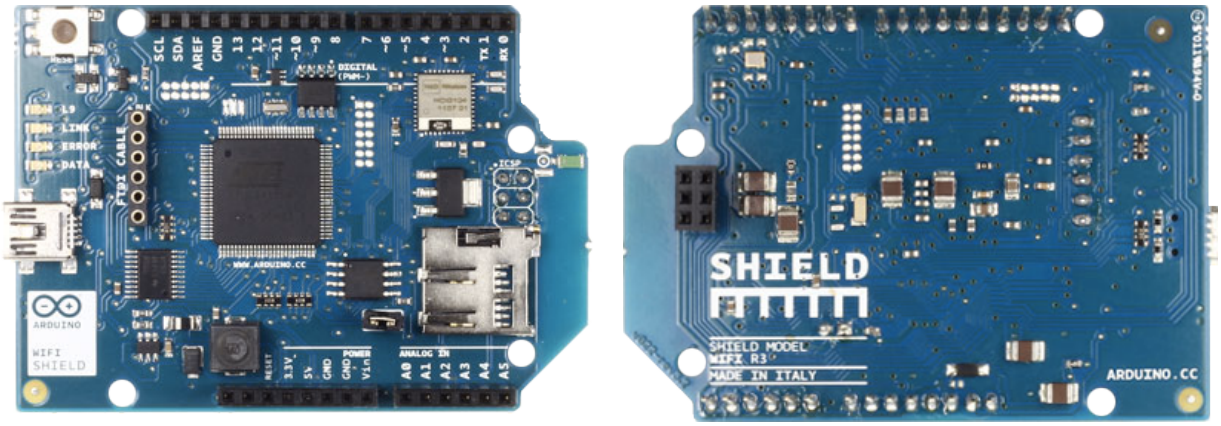


Ilustración 14: Frontal y reverso de la placa Arduino WiFi Shield

Características técnicas

Requisito	Tarjeta Arduino
Voltaje operativo	5V
Vía de conexión	Redes 802.11b/g
Conexión con Arduino	Puerto SPI (a través de las cabeceras ICSP)
Actualización *firmware*	A través del puerto Mini-USB
Diagnóstico	A través de conector FTDI
Otros	Ranura para tarjetas micro SD

Tabla 2: Características técnicas Arduino WiFi Shield

Conexiones

Pines reservados por la placa

Es preciso reservar algunos pines digitales de la placa Arduino UNO para el uso de este *shield*. Los pines 11, 12 y 13 son utilizados por Arduino para la comunicación con el microcontrolador WiFi y el lector de tarjetas micro SD. El pin número 10 es utilizado por el microcontrolador HDG204 y el pin 4 por el lector SD. Por último el pin 7 es utilizado como pin de negociación entre el WiFi *shield* y la tarjeta Arduino.

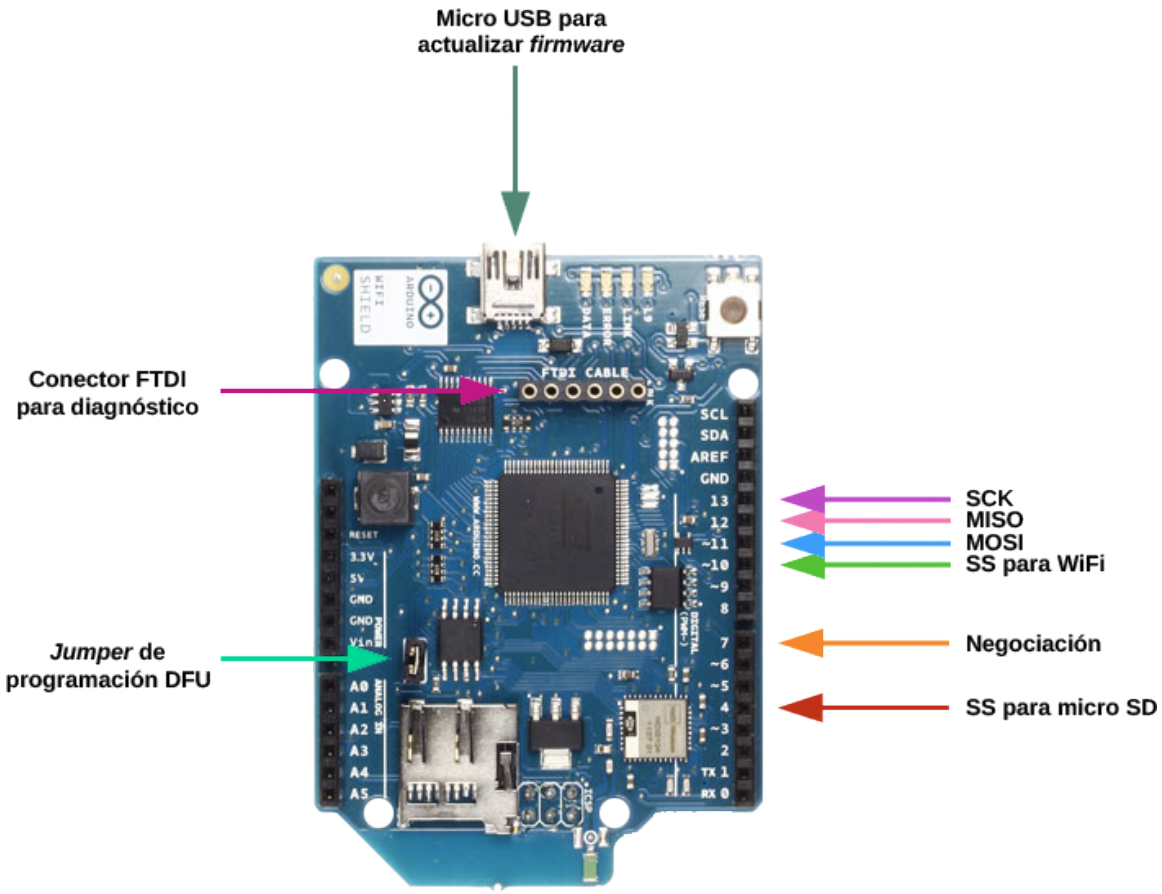


Ilustración 15: Conexiones de la placa Arduino WiFi Shield

10.1.3 Módulo de 4 relés de 5V

Para el encendido y apagado de los enchufes vamos a hacer uso de un sistema de 4 relés, que operan a 5V –el mismo voltaje al que operan las salidas digitales de la tarjeta Arduino–.

> "El relé o relevador es un dispositivo electromecánico. Funciona como un interruptor controlado por un circuito eléctrico en el que, por medio de una bobina y un electroimán, se acciona un juego de uno o varios contactos que permiten abrir o cerrar otros circuitos eléctricos independientes."[3]

[3] http://es.wikipedia.org/wiki/Rel%C3%A9, 2014

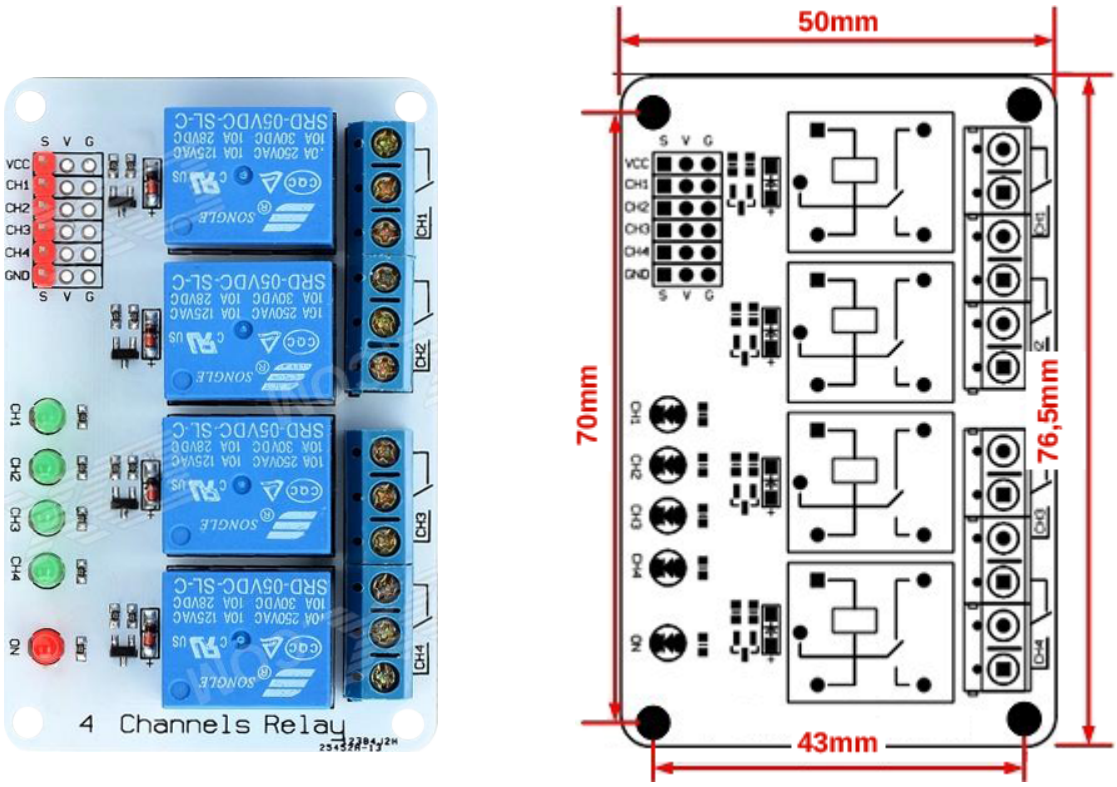


Ilustración 16: Módulo de 4 relés para Arduino

Características técnicas

Número de relés	5
Intensidad en DC por canal	10[a]
Voltaje operativo	5V
Dimensiones	7,7x5,0x1,8cm
Peso	59gr

Tabla 3: Características técnicas módulo de 4 relés para Arduino

Cada uno de los relés tiene 3 entradas: común, normalmente abierto y normalmente cerrado. Además la placa dispone de 6 pines: 1 por cada relé, 1 para la toma a tierra y 1 pin para la alimentación a 5V. Los pines asociados a cada relé se denominan canales, al pasar corriente por un canal su relé asociado pasa de conectar el conector común y normalmente cerrado a conectar el común y el normalmente abierto. Estado que vuelve a recuperarse al dejar de pasar corriente por el canal.

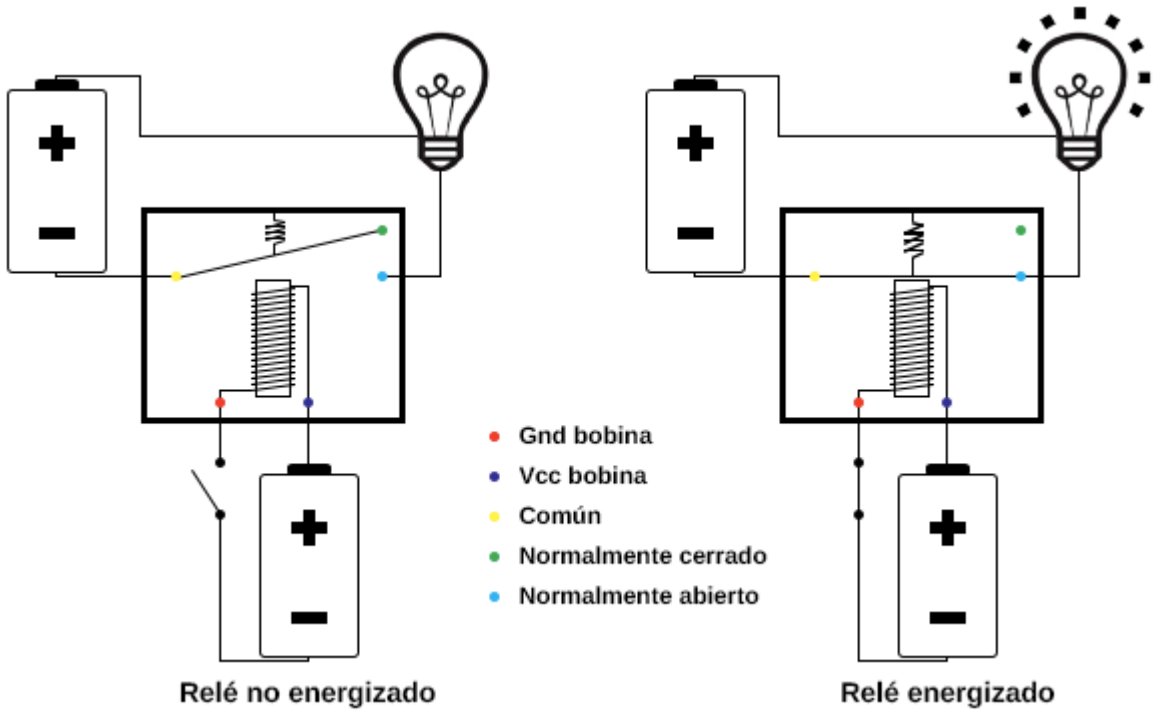


Ilustración 17: Funcionamiento de un relé

10.2 *Software*

10.2.1 NetBeans 8.0

El IDE de nuestra elección para el desarrollo de código de nuestras distintas aplicaciones ha sido NetBeans.

NetBeans IDE permite de forma rápida y sencilla desarrollar aplicaciones Java de escritorio, para móviles, aplicaciones web, así como aplicaciones HTML5 con HTML, JavaScript y CSS. El IDE además ofrece un gran conjunto de herramientas para desarrolladores de PHP y C/C++. Es gratuita y de código abierto y tiene una gran comunidad de usuarios y desarrolladores alrededor del mundo.

Un IDE es mucho más que un editor de texto. El Editor NetBeans indenta líneas, identifica palabras, paréntesis, llaves y corchetes, así como resalta código fuente de forma sintáctica y semántica. Además ofrece plantillas de código, consejos en la codificación, y herramientas para la refactorización.

El editor soporta diferentes lenguajes de programación, desde Java, C/C++, XML y HTML, a PHP, Groovy, Javadoc, JavaScript y JSP. Y además, como el editor ofrece extensiones, es posible añadir soporte para otros muchos lenguajes.

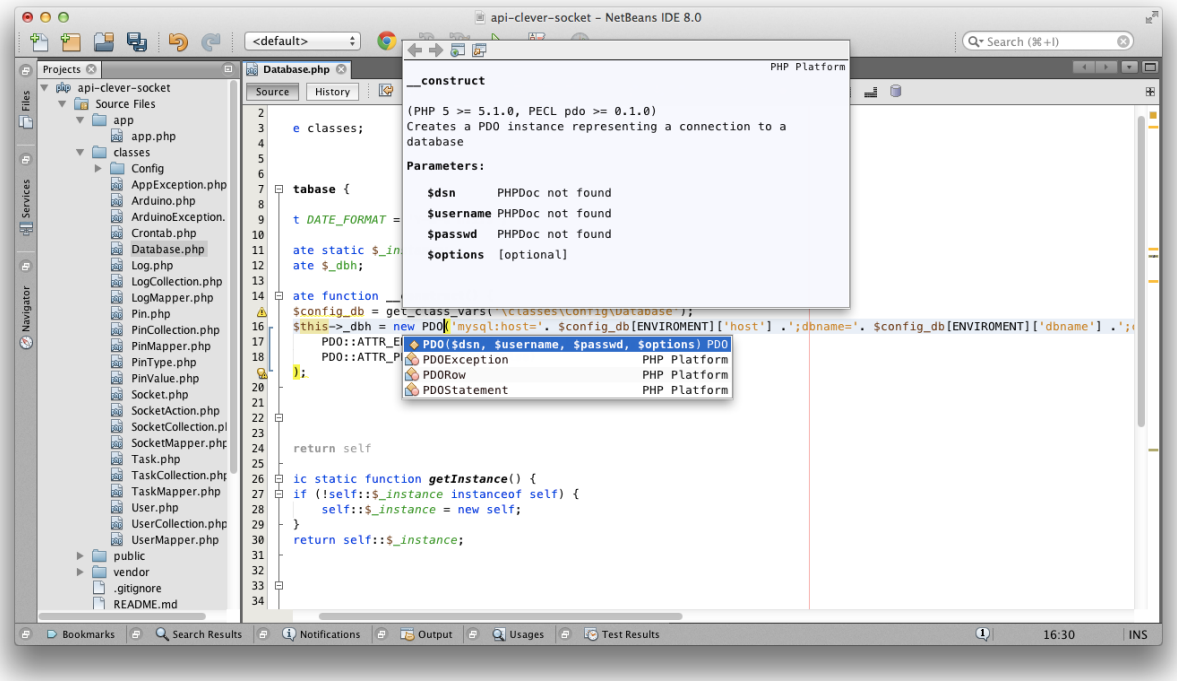


Ilustración 18: NetBeans IDE 8.0

Para poder desarrollar código sobre Arduino, así como para poder cargar los programas en la memoria interna de la placa por vía del puerto serie, hemos tenido que llevar a cabo algunos pasos, que se detallan en los anexos en el apartado Anexo II: Configuración de NetBeans para desarrollo de código para Arduino.

10.2.2 Arduino 1.0.5

Arduino suministra una plataforma de desarrollo orientada a la programación de código, así como a la carga de este en sus diferentes dispositivos.

Además de ofrecer un IDE de desarrollo escrito en JAVA, ofrece también diferentes librerías que hacen más sencillo el desarrollo de aplicaciones sobre la plataforma. Así como diferentes ejemplos predefinidos para entender mejor el uso de las librerías. El lenguaje de programación de Arduino se basa en C/C++.

En nuestro caso, no hemos hecho uso del IDE de Arduino para la codificación, ya que es muy limitado y no ofrece las múltiples funcionalidades para el desarrollo que ofrece NetBeans. Sin embargo, sí hemos hecho uso de las librerías, así como de los diferentes ejemplos que vienen predefinidos sobre el uso de éstas.

Ilustración 19: Arduino IDE 1.0.5

10.2.3 MySQL Workbench 6.1

Esta es la herramienta *software* que hemos utilizado para realizar el diseño de nuestras bases de datos.

MySQL Workbench es una herramienta visual unificada para arquitectos, desarrolladores y administradores de bases de datos.

Esta herramienta permite a administradores de bases de datos, desarrolladores o arquitectos de la información diseñar de forma visual, modelar, generar y gestionar bases de datos. Incluye todo lo que un modelador de información necesita para crear modelos de entidad-relación complejos, ingeniería directa y reversa, así como la posibilidad de llevar a cabo tareas de gestión y documentación que normalmente requerirían mucho tiempo y esfuerzo.

Gracias a esta herramienta, hemos podido realizar de forma sencilla el diseño de nuestras bases de datos y llevar a cabo la implementación de su modelo. Además, gracias a su sistema de ingeniería directa y reversa, hemos podido sincronizar los cambios en el diseño y el modelo. Facilitando la tarea de hacer modificaciones en la estructura de la base de datos.

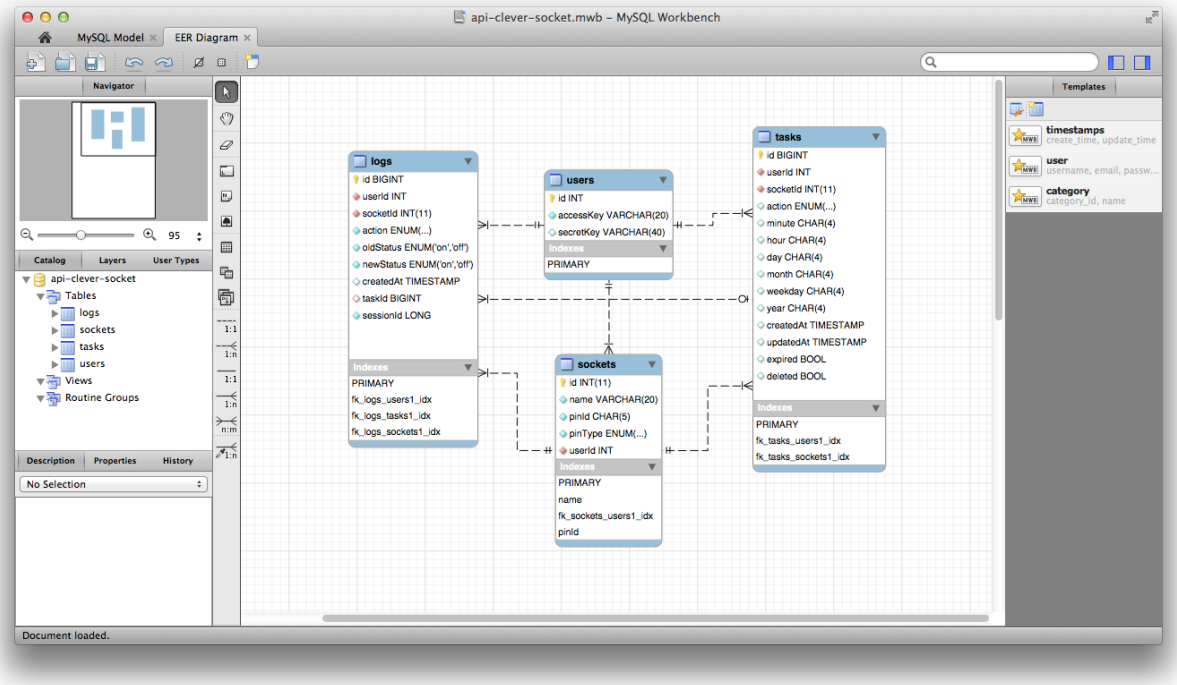


Ilustración 20: MySQL Workbench 6.1

10.2.4 Sequel Pro 1.0.2

Hemos elegido Sequel Pro como cliente para la realización de consultas a nuestras bases de datos.

Sequel Pro es una fantástica herramienta para la gestión de bases de datos. Permite realizar modificaciones en la estructura, cambios en campos, inserción de nuevos registros, etc. Sin embargo, el uso principal que nosotros le hemos dado a esta aplicación, es la realización de consultas, ya que permite de un modo muy sencillo y directo llevar a cabo esta tarea.

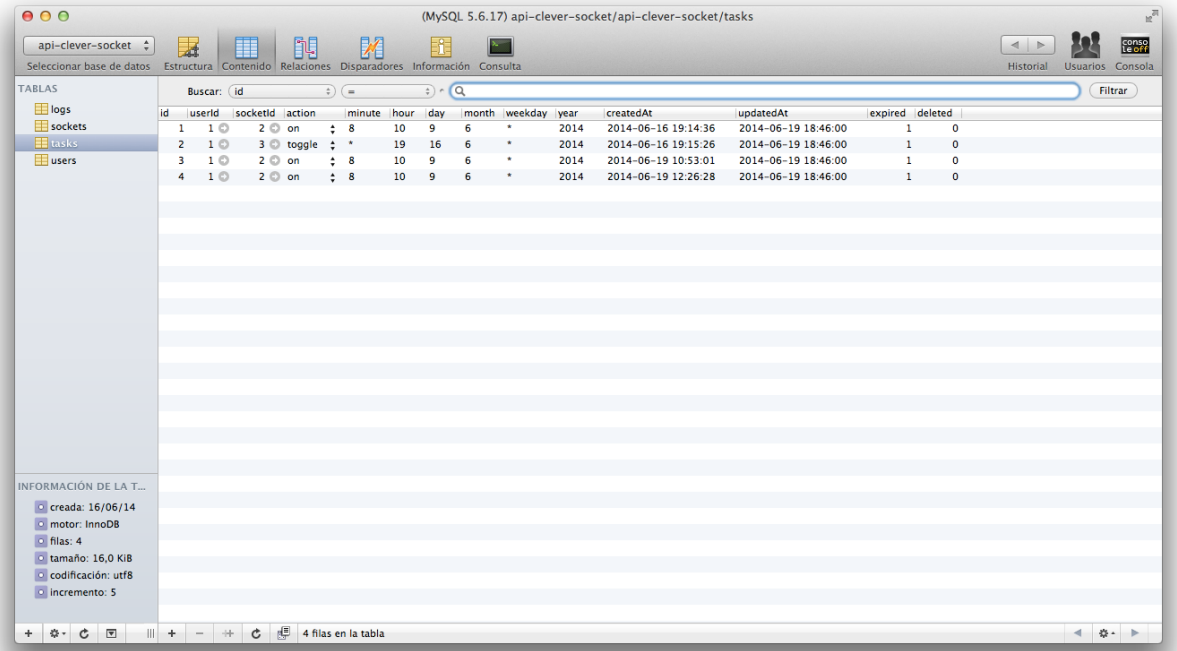


Ilustración 21: Sequel Pro 1.0.2

10.2.5 Cocoa Rest Client 1.3.7

Esta es la aplicación que hemos elegido para realizar las consultas y pruebas a nuestros servicios web de tipo REST.

Cocoa Rest Client es una aplicación cliente de código abierto, desarrollada para Mac OS y destinada al testeo de comunicación entre extremos HTTP/REST.

Entre las funcionalidades principales de esta aplicación, destacamos las siguientes:

- La posibilidad de realizar llamadas de tipo GET, PUT, POST, DELETE y HEAD.
- Establecer cabeceras así como mostrar su contenido en las respuestas de las comunicaciones.
- Detección y formateado automático de contenido XML y JSON.
- Guardado rápido de las URLs, cuerpo y cabeceras de consultas realizadas.

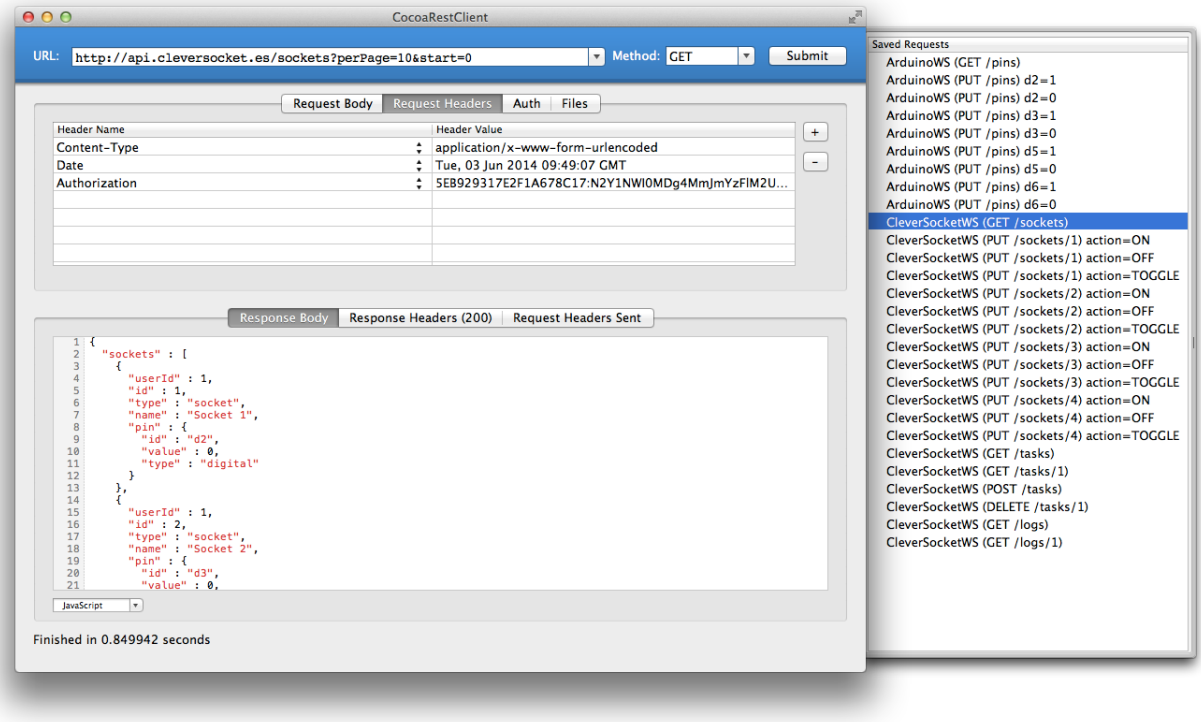


Ilustración 22: Cocoa Rest Client 1.3.7

10.2.6 Fritzing Beta 0.8.7

Esta aplicación se ha popularizado mucho en los últimos tiempos. La aplicación ayuda en la automatización de diseños electrónicos, permitiendo la realización de esquemas eléctricos de forma muy sencilla y visual.

> "Fritzing fue creado bajo los principios de Processing y Arduino, y permite a los diseñadores, artistas, investigadores y aficionados documentar sus prototipos basados en Arduino y crear esquemas de circuitos impresos para su posterior fabricación. Además cuenta con un sitio web complementario que ayuda a compartir y discutir bosquejos y experiencias y a reducir los costos de fabricación."[4]

Nosotros hemos hecho uso de la aplicación para documentar el montaje eléctrico del prototipo realizado en la fase de validación de la propuesta.

[4] http://es.wikipedia.org/wiki/Fritzing, 2014

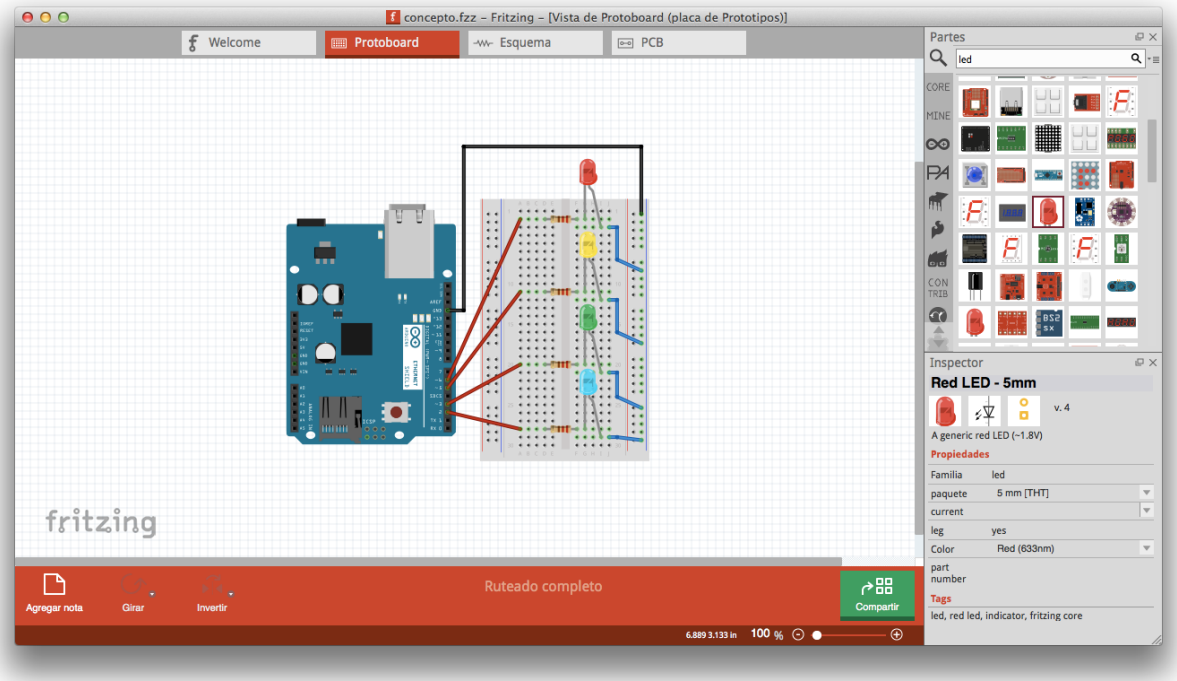


Ilustración 23: Fritzing Beta 0.8.7

10.2.7 Git 1.9.1

Para la gestión de versiones en este trabajo hemos hecho uso del conocido *software* Git.

Git es un *software* destinado al control de versiones diseñado por el desarrollador de *software* Linus Torvalds. En la actualidad su uso se encuentra ampliamente extendido entre la comunidad de programadores.

Entre sus características principales, se encuentran:

- Apoyado en el desarrollo no lineal.
- Rapidez en la gestión de ramas y sincronización entre diferentes versiones.
- Gestión distribuida, cada desarrollador dispone de una copia local completa de todo el repositorio.
- Gestión eficiente de grandes proyectos *software*.

10.2.8 Bitbucket

Bitbucket es un conocido servicio de almacenamiento y gestión de repositorios Git en línea. A diferencia del también popular sistema de hospedaje GitHub, Bitbucket permite la creación de forma gratuita de repositorios privados.

Gracias a este servicio podemos además de almacenar y gestionar nuestros repositorio en internet, compartir nuestro código de forma sencilla, permitiendo que otros programadores puedan ver nuestro código y colaborar.

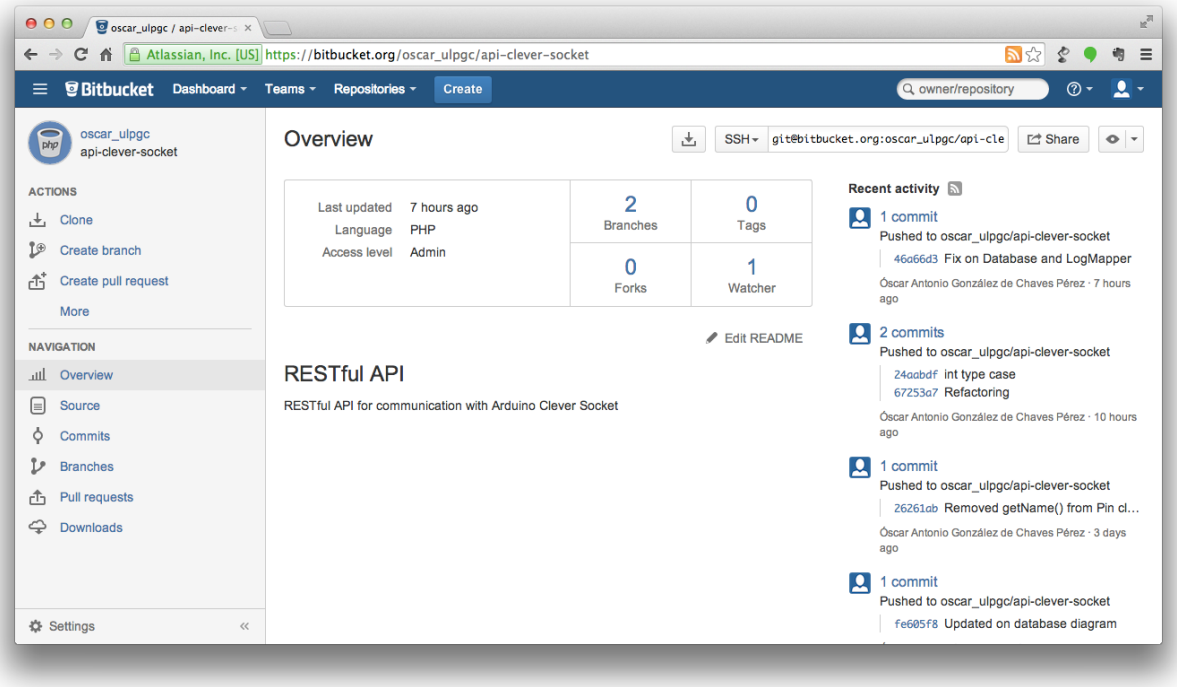


Ilustración 24: Bitbucket

10.2.9 Google Drive y Google Docs

A la hora de llevar a cabo la gestión de nuestros archivos y documentos para la realización de esta memoria, hemos optado por la opción en la nube de Google Drive.

Google Drive es un conjunto de herramientas ofimáticas disponible a través de internet y accesible a través del navegador web. Entre sus principales ventajas, destacamos el hecho de poder disponer en cualquier momento y lugar de todas nuestras anotaciones, imágenes, documentos, etc.

La fórmula gratuita de Google Drive ofrece 15 gigabytes de almacenamiento, que pueden ser ampliables mediante pago.

Dentro del paquete ofimático encontramos todas las aplicaciones típicas de este tipo de *software*: desde un editor de texto, una hoja de cálculo, un editor de presentaciones, etc. Además, el número de aplicaciones que trae por defecto, puede ser ampliado a través de aportaciones de terceros, con lo cual su utilidad puede aumentar de manera exponencial.

De entre las aplicaciones ofrecidas por Google no obstante, hemos hecho uso principalmente del editor de texto, Google Docs. En el siguiente apartado veremos otras aplicaciones de terceros también utilizadas.

Además, otra de las ventajas de esta oferta de aplicaciones está en la posibilidad de trabajar de forma colaborativa. De modo que podemos no solo compartir documentos con otros contactos, sino también ofrecer la posibilidad de que esos mismos contactos colaboren en la edición de los documentos. Pudiendo hablar con ellos en línea a través de una funcionalidad de *chat* disponible desde cualquier parte de la aplicación.

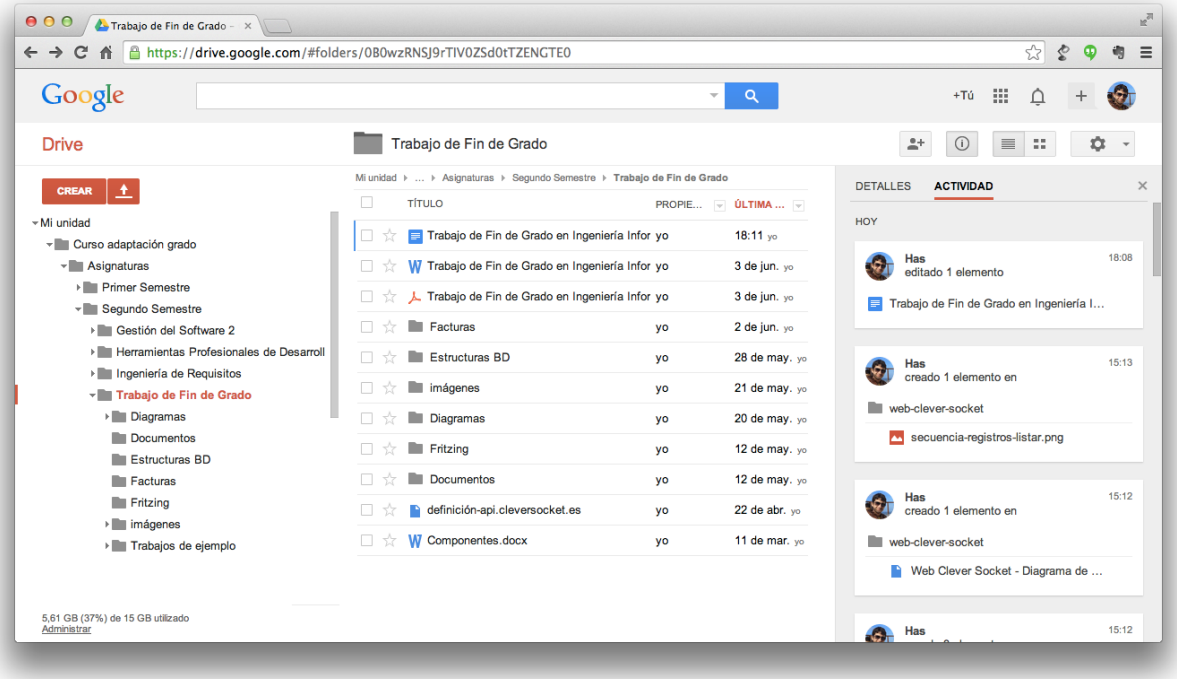


Ilustración 25: Google Drive

10.2.10 Lucidchart

Esta herramienta nos ayudado a realizar muchos de los diagramas utilizados en este trabajo. Es además, una de las soluciones de terceros que se integra perfectamente con Google Drive y que comentábamos en el apartado anterior.

Gracias a la integración con Google Drive, podemos hacer uso de la aplicación y de sus archivos del mismo modo que lo hacemos con el resto de utilidades y documentos de la *suite*.

Lucidchart tiene una gran variedad de tipos de diagramas, entre los que se encuentran los típicos diagramas con nodos y enlaces, diagramas conceptuales, y por supuesto ofrece también la opción de realizar diagramas de tipo UML. La oferta de tipo de diagramas, como

también ocurre con el espacio ofrecido por Google Drive, puede ser ampliada a través de fórmulas de pago.

Nosotros hemos hecho uso de esta aplicación sobre todo para los diagramas UML, pero también para algunos diagramas no tan estandarizados.

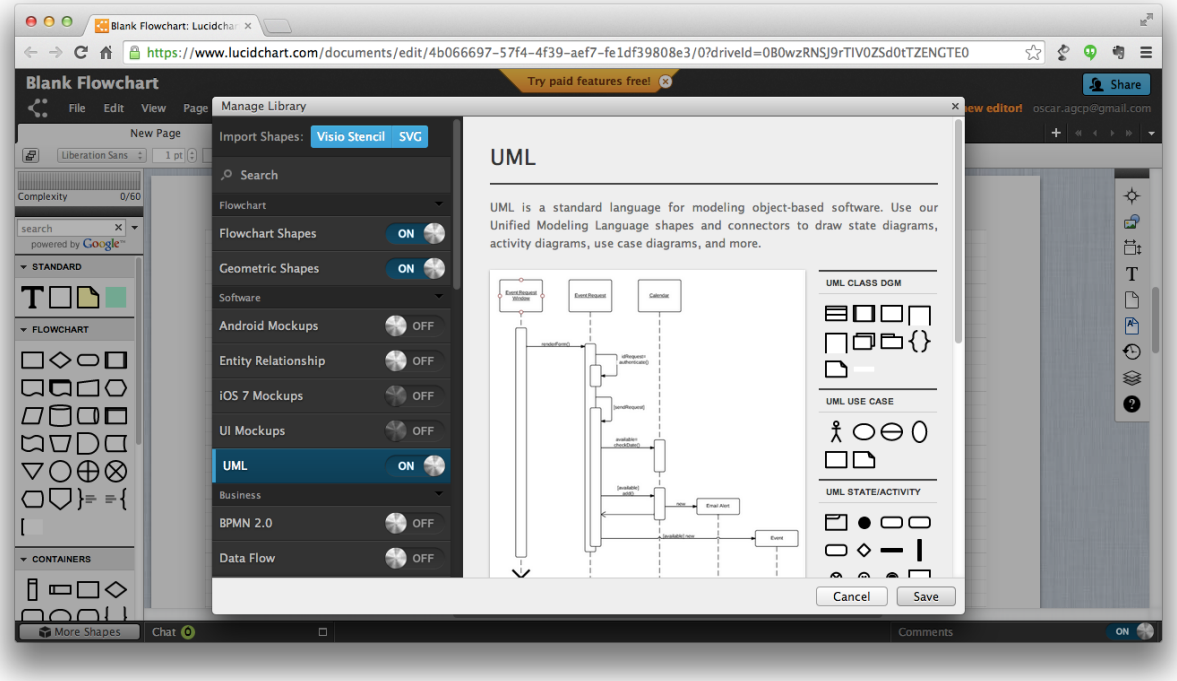


Ilustración 26: Lucidchart

10.2.11 Moqups

Moqups es otra de las aplicaciones que se integran dentro de Google Drive. En esta ocasión la funcionalidad ofrecida por la aplicación es la de realizar prototipos de interfaces(o como también se le llama a veces por el término inglés, *mockups*).

A la hora de realizar los prototipos de nuestras interfaces de usuario, nos ha sido de gran ayuda poder disponer de esta herramienta. Con tan solo unos poco clics, hemos podido realizar presentaciones vistosas e ilustrativas que consiguen plasmar perfectamente nuestros conceptos.

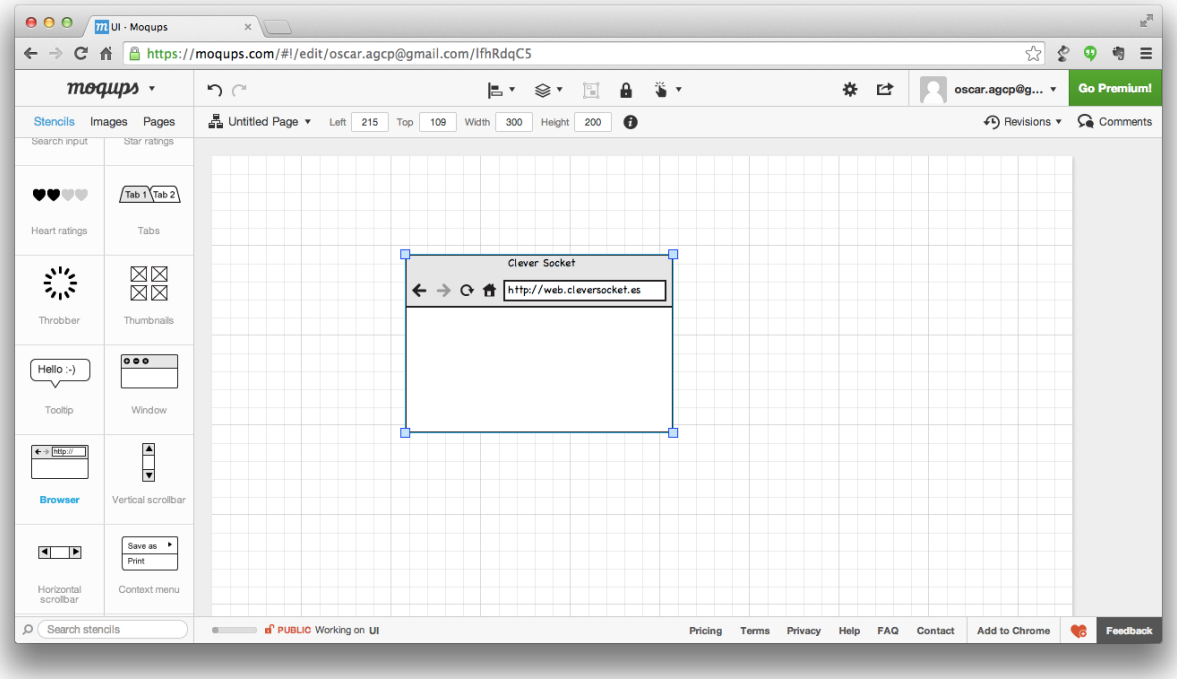


Ilustración 27: Moqups

11 Análisis

11.1 Introducción

La fase del análisis es el punto de origen fundamental de cualquier proyecto. Es a partir de ella desde donde comienza a germinar el trabajo. Por este motivo, es importante prestar atención a este cometido, ya que a partir de una buena base, se podrán desarrollar de forma sistemática y organizada el resto de acontecimientos.

A grandes rasgos, deberemos prestar atención a dos aspectos principales: el desarrollo de los requisitos y la gestión de sus cambios.

Para comenzar con el desarrollo de los requisitos, es fundamental dar respuesta a algunas preguntas clave sobre nuestro sistema o solución: ¿Qué problema queremos resolver?, ¿qué características funcionales debe tener?, ¿qué servicios debe ofrecer? ¿cómo lo vamos a hacer?, etc.

11.2 ¿Qué problema queremos resolver y por qué?

Con el desarrollo de este TFG nos hemos propuesto resolver el problema de poder controlar el gobierno de un sistema o conjunto de enchufes a través de una red de comunicación, como puede ser una red de tipo LAN o WAN.

Para ello, hemos realizado en primera instancia un estudio acerca de la situación del estado actual, sobre cómo la tecnología está dando respuesta al problema en este momento.

El estudio anterior, nos ha impulsado a pensar en nuevas soluciones, propuestas más innovadoras y viables.

Después de descartar otras opciones más complejas, nos decantamos por utilizar un sistema *hardware* y *software* apoyado completamente en la plataforma de desarrollo electrónico Arduino.

Los motivos que nos llevaron a tomar la decisión de elegir esta plataforma, se fundamentan principalmente en los siguientes aspectos:

- Es una plataforma de desarrollo de código abierto.
- Está basada en una placa con un sencillo microcontrolador.
- Posee un entorno de desarrollo para crear software (programas) para la placa.
- Lenguaje de programación ampliamente extendido, C.
- Amplia comunidad internacional respaldando el proyecto.
- Gran cantidad de módulos, paquetes y librerías existentes.
- Sencillez de prototipado electrónico.
- Modularidad y facilidad a la hora de ampliar las funcionalidades básicas de la placa, a través de distintas ampliaciones denominadas *shields*.

La placa de desarrollo Arduino, ofrece una serie de pines de conexión, a través de los cuales se hace posible interactuar con ella. Estos pines pueden ser configurados como entradas o como salidas. Además, es posible elegir entre pines analógicos y pines digitales.

En los pines digitales se pueden establecer los valores *HIGH* y *LOW,* que significan 5V y 0V respectivamente.

En cambio, en los pines analógicos se puede diferenciar cualquier valor intermedio entre los 5V y los 0V. La resolución entre los valores máximos y mínimos difiere de un microprocesador a otro. Arduino únicamente puede distinguir 1024 niveles en ese rango de voltaje.

Se pueden ver a continuación dos ejemplos de código, que muestran lo sencillo que resulta el trabajo con los pines analógicos y digitales:

Ejemplo 1: Configuración del pin digital número 8 como salida, y valor a HIGH*.*

```
pinMode(8, OUTPUT);      // establece el pin 8 como salida
digitalWrite(8, HIGH);   // pone el valor del pin 8 a HIGH
```

Ejemplo 2: Configuración del pin analógico 4 como entrada, y lectura de su valor.

```
pinMode(4, INPUT);       // establece el pin 4 como entrada
analogRead(4);           // lee valor de tensión del pin 4
```

Tabla 4: Características placa Arduino

Todo lo anterior, nos ha llevado a pensar en Arduino como una gran opción para llevar adelante nuestra idea de proyecto.

11.3 ¿Qué solución planteamos?

11.3.1 Del lado del *hardware*

Como hemos comentado, Arduino ofrece multitud de módulos y opciones de ampliación. En la actualidad el modelo básico de Arduino (Arduino UNO) no ofrece comunicación a través de redes tipo LAN o WAN de forma directa. Por ese motivo, se hace necesario incorporar algún tipo de módulo de expansión.

Existe una gran oferta en lo relativo a módulos de conectividad para Arduino, diferentes fabricantes y diferentes opciones. Nosotros estudiamos principalmente dos posibilidades:

- Arduino Ethernet Shield
- Arduino WiFi Shield

El motivo por el que descartamos otras alternativas y nos centramos en estas opciones, se basa fundamentalmente en la amplia documentación existente para trabajar con ellos, así como la gran cantidad de código libre disponible para su aprovechamiento. Además, cabe destacar que ambas placas son ensambladas y certificadas por Arduino, lo que nos ofrece mayores garantías.

Ambas opciones son interesantes, y ambas ofrecen ventajas e inconvenientes. Así que para la elección final entre una o otra placa, tuvimos que valorar las siguientes ventajas y desventajas:

Arduino Ethernet Shield

Ventajas:

- Precio reducido.
- Buena documentación.
- Librería "Ethernet Library" proporcionada por Arduino.
- Gran cantidad de código libre existente para su utilización.

Desventajas:

- No posee conexión inalámbrica.

Arduino WiFi Shield

Ventajas:

- Buena documentación.
- Librería "WiFi Library" proporcionada por Arduino.
- Buena cantidad de código libre existente para su utilización.
- Conectividad inalámbrica.

Desventajas:

- Precio superior.

Comparativamente, ambos *shields* poseen características similares, y ambos son buenas opciones. Sin embargo, nosotros nos decantamos finalmente por el el *shield* Arduino WiFi Shield. El motivo para dicha elección fue fundamentalmente el factor de la conectividad inalámbrica proporcionado por el *shield*.

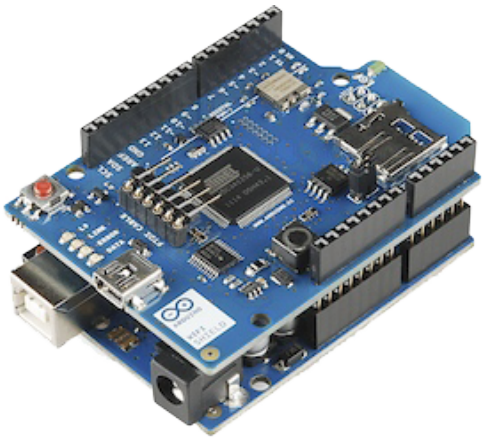


Ilustración 28: Arduino UNO y Arduino WiFi Shield

Tras tomar una decisión acerca de qué sistema utilizar para permitir la comunicación entre la placa Arduino y la red de comunicación, necesitamos buscar una solución para poder gobernar el encendido y apagado de las bases de los enchufes. Para ello, pensamos en la opción de utilizar relés, una solución efectiva y económica.

La utilización de relés en el prototipado con Arduino es algo común y de uso ampliamente extendido, de modo que existe gran oferta de sistema *ad hoc* para trabajar con nuestra placa de desarrollo. En la sección de recursos materiales se puede ver la opción elegida.

11.3.2 Del lado del *software*

Por el lado del *software* nos propusimos algunos objetivos generales:

- Aportar una interfaz sencilla de comunicación con el sistema de enchufes.
- Permitir la conectividad a través de Internet.
- Ofrecer un servicio web que ofrezca la posibilidad de comunicarse con el sistema desde diferentes aplicaciones (web, móvil, etc.).
- Modularizar el diseño, de manera que exista una correcta separación, en función a la responsabilidad de cada capa.

De esta forma, y para dar respuesta a estas necesidades, pensamos conveniente establecer los siguientes niveles de abstracción o capas:

- Servicio web a bajo nivel del lado del sistema Arduino.
- Servicio web a alto nivel para la comunicación entre las aplicaciones y el servicio web de Arduino.
- Interfaz de usuario a partir de una aplicación web *ad hoc*.

Diagrama 1: Diagrama de despliegue

Servicio Web Arduino

El *shield* Arduino WiFi Shield tiene la habilidad de poder desplegar un servidor HTTP. Gracias a esta característica nos planteamos la posibilidad de desarrollar un servicio web de comunicación directa con la placa, para la lectura y escritura de sus pines.

En este sentido, podría obviarse la necesidad de hacer uso de la capa intermedia –servicio web de Clever Socket– que hemos propuesto. Sin embargo, la escasez de memoria y potencia del microcontrolador ATmega328 de la placa, desaconsejan el desarrollo de un servicio web complejo de comunicación. Además, tener un servicio web para cada propósito hace que respetemos uno de los principios perseguidos por el agilismo, el principio de responsabilidad única.

Por este motivo, hemos pensado desarrollar un servicio web muy simple y de bajo nivel que se encuentre disponible en el servidor HTTP desplegado por la Arduino WiFi Shield. Con la única responsabilidad de aceptar peticiones de lectura y escritura de los pines analógicos y digitales de la placa.

Para llevar a cabo dicha tarea, hemos explorado la posibilidad de desarrollar nuestro propio servicio web desde cero, haciendo uso de la librería "WiFi Library". Sin embargo, nos hemos dado cuenta de la dificultad de crear un servicio web eficiente con las limitaciones ofrecidas por el sistema hardware.

Analizando diferentes posibles soluciones que pudieran adaptarse a nuestras necesidades, encontramos un proyecto desarrollado bajo licencia del MIT, llamado Webduino, con algunas de las siguientes características:

- Analizador de parámetros URL

- Soporte para los métodos HTTP: GET, HEAD, POST, PUT, DELETE, PATCH
- Interfaz JSON/RESTful
- Autenticación básica http

Este proyecto *software* tiene amplio recorrido, habiendo sido lanzada su primera versión a comienzos del año 2009. Su código está bien testado y documentado, y cubre todas nuestras exigencias para el nivel en el que nos encontramos.

Por este motivo decidimos desarrollar nuestro servicio web Arduino bajo esta solución *software*.

Servicio Web Clever Socket

A este nivel nos planteamos la opción de prestar un servicio web de tipo REST para la gestión del sistema de enchufes. Este servicio debe tener un nivel de abstracción superior al servicio web del sistema Arduino.

En contrapartida al anterior servicio, no vamos a operar directamente sobre los pines de la placa Arduino, en su lugar, se tendrá una visión superior, en esta capa nos vamos a abstraer hasta el nivel de los enchufes (o como también serán mencionados a lo largo de este documento, *sockets*). De modo que al utilizar este servicio, no estaremos pensando en la comunicación directa con la placa Arduino, sino que estaremos pensando en una comunicación con el sistema global de enchufes, donde el *shield* WiFi y la placa Arduino son un elemento más.

A la hora de decantarnos por una tecnología para el diseño y codificación de este servicio, estudiamos diferentes opciones. Teníamos en mente la idea de encontrar una solución simple, ya que el servicio que se va a prestar es muy sencillo, y no necesita de artefactos muy complejos. Nos interesaba una opción que permitiera levantar un servicio RESTful de forma rápida y sencilla, pero que a la vez, fuese potente, suficientemente testada.

Barajamos distintas opciones de *frameworks* como :

- BulletPHP
- Fat-Free Framework
- Limonade
- Phalcon
- Recess PHP
- Silex
- Slim
- Tonic

- Wave Framework
- Zaphpa

Y finalmente, después de analizar ventajas e inconvenientes de unos y otros, nos terminamos decantando por el Slim Framework.

Ilustración 29: Slim Framework

Slim Framework es definido por sus desarrolladores como un micro *framework* para PHP, que ayuda en el proceso de desarrollo de aplicaciones web y APIs de forma no sólo rápida sino también potente.

Algunas de sus características que nos hicieron pensar en él como candidato, son:

- Uno de los micro *frameworks* más ligeros y rápidos disponibles
- Buena documentación
- Provee de todas las características que este tipo de *framework* debe tener, ni más, ni menos
- Al contrario de lo que ocurre con otros *frameworks*, existe un gran seguimiento del proyecto, muchas fuentes de información disponibles en la red, desde la propia página oficial hasta páginas de terceros, incluyendo vídeo tutoriales en YouTube.

La manera que tiene Slim Framework de establecer los métodos y rutas de comunicación es muy sencilla.

Véase un ejemplo de su utilización:

```
<?php

$app = new \Slim\Slim();

$app->get('/hello/:name', function ($name) {
```

```
        echo “Hello, $name”;
    });
    $app->run();
```

De esta forma al hacer una solicitud de tipo GET a la dirección /hello/World, el sistema responderá con un: Hello, World.

Del mismo modo que hemos utilizado el método GET en el ejemplo anterior, podemos utilizar el resto de métodos estándar de HTTP, como POST, PUT o DELETE.

Tabla 5: Características de Slim Framework

Este *framework* cumple satisfactoriamente con nuestras exigencias. Por ello, para el desarrollo de nuestro API, haremos uso de él.

Interfaz de usuario

La interfaz de usuario es un elemento importante dentro de nuestro sistema *software*. Es la ventana de comunicación entre el usuario y nuestro sistema de enchufes, una abstracción de los elementos físicos del montaje *hardware* al entorno virtual, donde el usuario podrá interactuar.

Desde un primer momento, pensamos que la interfaz de usuario debía cumplir con las siguientes características:

- Sencillez de uso, manejo intuitivo.
- Baja complejidad en lo relativo a funcionalidades y opciones.
- Correcto visionado en un amplio rango de dispositivos; incluyendo ordenadores de escritorio, tabletas digitales o *smartphones*.

Nuevamente en esta ocasión, para encontrar la mejor solución tecnológica, realizamos una labor de investigación para encontrar la opción que más se pudiese adaptar a nuestras preferencias.

Teniendo en cuenta que disponemos de un API para la comunicación con el sistema de enchufes, el abanico de posibilidades para el desarrollo de la interfaz de usuario es bastante

amplia. Serían válidos tanto soluciones que pasan por aplicaciones de escritorio, aplicaciones nativas para terminales móviles hasta aplicaciones web.

Pensando en el propósito y dimensión de este trabajo, del tiempo con el que contamos para su elaboración y de nuestra intención de poder llegar con él a un espectro amplio de dispositivos, decidimos finalmente que la opción tecnológica más conveniente es la de una aplicación web, que tenga además un diseño *responsive*.

Eligiendo este tipo de diseño, tenemos la certeza de que los usuarios no tendrán ningún tipo de problema a la hora de visualizar la interfaz, independientemente del dispositivo o la resolución utilizados, ya que la visualización se adaptará automáticamente a las particularidades del aparato.

En un principio, barajamos también la idea de hacer uso de una opción de tipo *adaptative*. Esta alternativa, aunque tiene el mismo propósito que la anterior, difiere de la primera en que el sistema detecta desde qué dispositivo se realiza la petición, y en función de ello envía un contenido u otro adaptado a las características propias del dispositivo.

De modo que, con la idea en mente de llevar a cabo un diseño *responsive*, encontramos una solución que se adapta muy bien a nuestras necesidades, Bootstrap.

Ilustración 30: Bootstrap *front-end framework*

Bootstrap es un *framework* del lado del *front-end*, pensado enteramente para llevar a cabo diseños que cumplan con el principio *responsive*.

Detrás de este proyecto se encuentran los desarrolladores de Twitter, y aporta muchos puntos a favor que nos han hecho decantarnos por él como solución tecnológica:

- Contiene plantillas de diseño con tipografía, formularios, botones, cuadros, menús de navegación y otros elementos de diseño basado en HTML5 y CSS3.
- Extensiones en JavaScript para diverso tipo de acciones y manejo de eventos.
- Es uno de los proyectos más populares en GitHub.
- Se encuentra en contínuo desarrollo y existe gran cantidad de documentación.
- Soporte para dos de los preprocesadores de CSS más populares del mercado, Less y Sass.

Resuelta la solución del lado del *front-end* pasamos a estudiar la mejor opción para del lado del *back-end*.

En este terreno, existen multitud de posibilidades, muchas de ellas completamente válidas y que se podrían adaptar muy bien a nuestras necesidades. De forma que en este punto, sopesamos sobre todo la idea de encontrar un producto que tuviera detrás una buena documentación, que hubiese sido utilizada con éxito en un gran número de proyectos, y que tuviese una buena comunidad de desarrolladores detrás.

Además, nos planteamos nuevamente la idea de encontrar un producto con el cuál poder desarrollar nuestra aplicación de una forma rápida, con poca complejidad, con ligereza.

Conocíamos de antemano algunos *frameworks* muy populares como Zend, CakePHP, CodeIgniter o Symfony. Pero queríamos tomar la decisión más adecuada para nuestro propósito y comparando las características de unos y otros, llegamos a conocer un nuevo *framework*, derivado de Symfony y que en los últimos tiempos se ha ido haciendo tremendamente popular, Laravel.

Ilustración 31: Laravel PHP *framework*

Algunas de sus características son:

- Curva de aprendizaje relativamente pequeña.
- Potente ORM basado en el patrón ActiveRecord para el trabajo con la capa de persistencia.
- Su sistema de enrutamiento es muy sencillo.
- Tiene una gran comunidad de desarrolladores detrás.
- Trae un fantástico sistema de plantillas.
- Al estar desarrollado como una capa por encima de componentes básicos de Symfony, su código está testeado y es muy confiable.

11.4 Especificación de los requisitos

Después de analizar y haber respondido a preguntas como qué esperamos del sistema o cómo pensamos dar respuesta a esas demandas, intentaremos explicitar de una forma más detallada cuáles son los principales requisitos funcionales y no funcionales del sistema.

Recordemos las siguientes definiciones sobre qué entendemos por requisito funcional y no funcional:

> "Un **requisito funcional** define una función del sistema de software o sus componentes. Una función es descrita como un conjunto de entradas, comportamientos y salidas. Los requerimientos funcionales pueden ser: cálculos, detalles técnicos, manipulación de datos y otras funcionalidades específicas que se supone, un sistema debe cumplir. Los requerimientos de comportamiento para cada requerimiento funcional se muestran en los casos de uso. Son complementados por los requisitos no funcionales, que se enfocan en cambio en el diseño o la implementación."[5]

> "Un **requisito no funcional** o atributo de calidad es, en la ingeniería de sistemas y la ingeniería de software, un requisito que especifica criterios que pueden usarse para juzgar la operación de un sistema en lugar de sus comportamientos específicos, ya que éstos corresponden a los requisitos funcionales. Por tanto, se refieren a todos los requisitos que no describen información a guardar, ni funciones a realizar."[6]

11.4.1 Requisitos funcionales

En este apartado intentaremos dar respuesta a qué comportamientos se esperan del sistema. A partir de qué entradas se obtienen qué salidas.

Para ello, nos ayudaremos de diagramas UML y de plantillas para la especificación de los casos de uso.

[5] http://es.wikipedia.org/wiki/Requisito_funcional, 2013
[6] http://es.wikipedia.org/wiki/Requisito_no_funcional, 2014

Actores

Un actor, dentro de nuestro sistema, es una entidad que especifica un rol llevado a cabo. Este rol puede ser desempeñado tanto por un usuario, como por cualquier otro subsistema que interactúe con nuestro sistema.

Vamos a distinguir entre dos tipos de actores, los usuarios del sistema y los servicios.

Usuarios del sistema

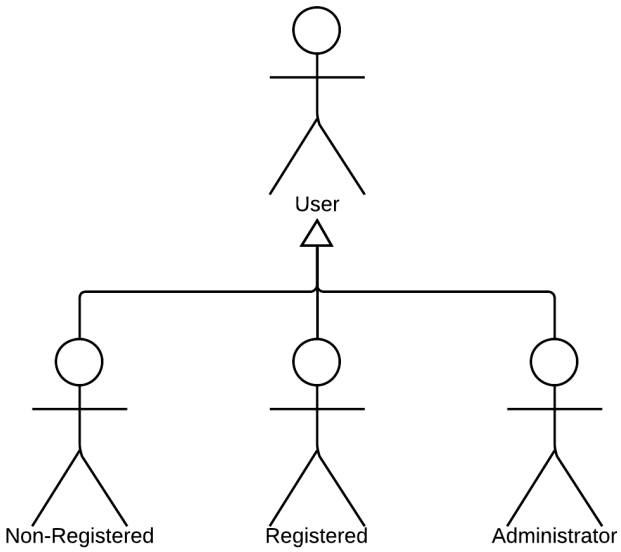


Diagrama 2: Usuarios del sistema

Usuario no registrado

El usuario no registrado (*non-registered user*), juega el papel de aquel usuario del sistema que todavía no se ha registrado o identificado frente al sistema. La funcionalidad que se le otorga es muy limitada, pudiendo únicamente acceder al apartado público de la aplicación.

Usuario registrado

El usuario registrado (*registered user*) es el **actor principal**, integra además de las funcionalidades del usuario no registrado (a excepción del registro), la capacidad de ejecutar las principales funcionalidades ofrecidas por el sistema. En definitiva, es el usuario final a quien se dirige la aplicación.

Administrador

El usuario administrador (*administrator user*), integra todas las funcionalidades de los usuarios anteriores (a excepción del registro), y además se le asignan funcionalidades de administración y gestión.

Subsistemas como servicios

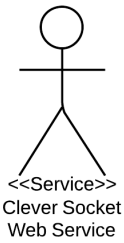


Diagrama 3: Servicios

Servicio Web Clever Socket

El Servicio Web Clever Socket (*Clever Socket Web Service*), representa al actor que provee el servicio principal de la aplicación, permite la comunicación con el dispositivo Clever Socket a través de un servicio web tipo REST.

Modelado de casos de uso

En esta fase del proyecto vamos a prestar atención únicamente a las funcionalidades que atañen al usuario registrado. Dejaremos para futuras revisiones, aquellas funcionalidades reservadas a los usuarios no registrados y administrador, como son las funciones de registro y de administración y gestión.

Usuario registrado

Como dijimos anteriormente, el usuario registrado es quien disfrutará de las principales funcionalidades ofrecidas por el sistema. A continuación mostramos las principales acciones que puede llevar a cabo:

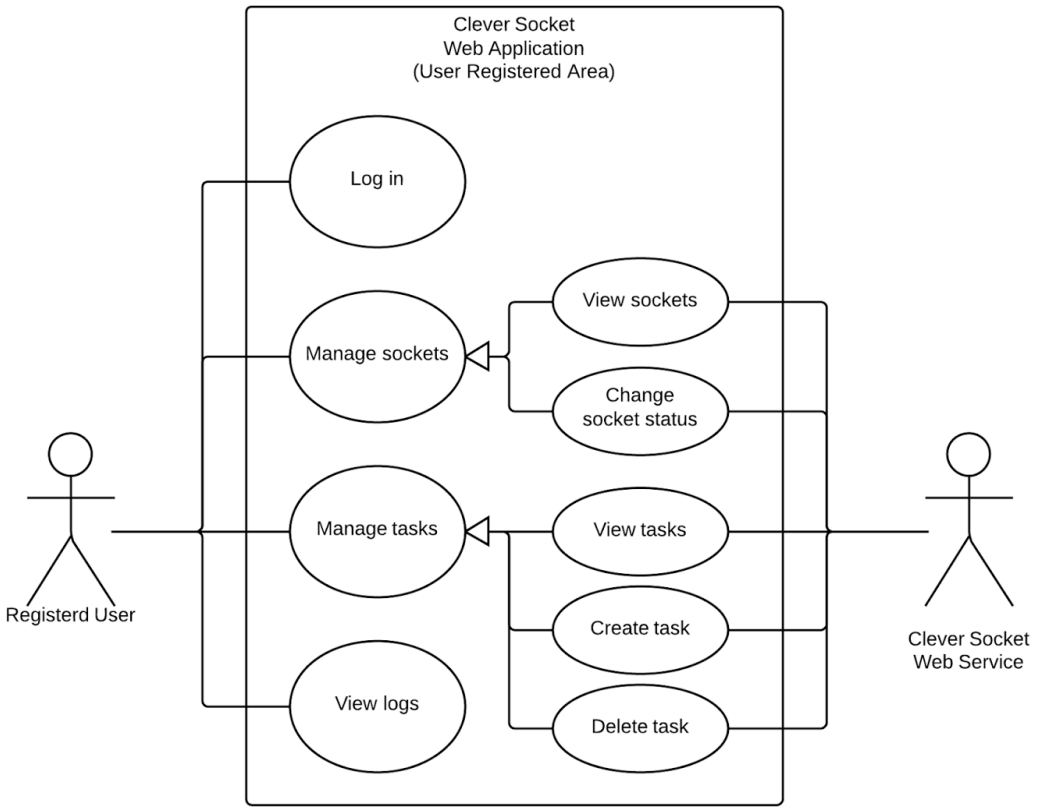


Diagrama 4: Casos de uso

Veremos en el siguiente apartado, que el modelado del caso de uso "Cambiar estado del *socket*" (*Change socket status*) es una generalización de tres tipos diferentes de casos de uso: encender, apagar y conmutar entre el encendido y el apagado de un *socket*.

Casos de uso

Como ya hemos mencionado, los actores interactúan con el sistema. En este sentido, un caso de uso no es más que un conjunto de interacciones que son llevadas a cabo entre el sistema y sus actores en respuesta a una acción o evento ejecutado por uno de sus actores principales.

Del mismo modo que los diagramas de casos de uso nos sirven para especificar la comunicación y el comportamiento de un sistema y sus actores, es posible llevar a cabo un mayor nivel de detalle a través de la especificación por plantillas de los casos de uso.

Para la especificación de los casos de uso, vamos a servirnos de la plantilla sugerida por Ignacio Solinis Camalich, miembro del Instituto Universitario de Ciencias y Tecnologías Cibernéticas. Esta plantilla surge de una modificación de la realizada por Karl E. Weigers, consultor de la compañía Process Impact en Portland, Oregon.

Listado de casos de uso

Actor principal	**Caso de uso**
Usuario registrado	1. Identificarse (*log in)*
Usuario registrado	2. Ver *sockets (view sockets)*
Usuario registrado	3. Encender *socket (turn socket on)*
Usuario registrado	4. Apagar *socket (turn socket off)*
Usuario registrado	5. Conmutar *socket (toggle socket)*
Usuario registrado	6. Ver tareas (*view tasks*)
Usuario registrado	7. Crear tarea (*create task*)
Usuario registrado	8. Eliminar tarea (*delete task*)
Usuario registrado	9. Ver registros (*view logs*)

Tabla 6: Listado de casos de uso

Las plantillas con el detalle de los casos de uso pueden ser consultadas en el Anexo V: Plantillas de casos de uso.

11.4.2 Requisitos no funcionales

Para poder implementar y ejecutar los subsistemas que componen Clever Socket, solo podemos garantizar el funcionamiento óptimo del sistema, cumpliendo al menos con los siguiente requisitos no funcionales:

1. Procesador a 2 GHz Intel Core 2 Duo
2. Memoria RAM 4 GB 1067 MHz DDR3
3. Sistema operativo Mac OS 10.9.3
4. Servidor HTTP Apache versión Apache/2.2.26 (Unix)
5. Módulo PHP 5.5.12
6. MySQL Community Server 5.6.17
7. Compilador Apple LLVM 5.1
8. Acceso a red WiFi WPA2
9. Dispositivo *hardware* Clever Socket

12 Prueba del concepto

Antes de comenzar con la etapa del diseño, hemos querido trabajar sobre la validación de la idea. Tenemos más o menos clara cuál es la idea que queremos diseñar y posteriormente implementar, pero deseamos comprobar primero si nuestros planteamientos teóricos podrán ser llevados realmente a la práctica.

Para poder lograr lo anterior, vamos a realizar una prueba de concepto. Un prototipo muy limitado en funcionalidad y de bajo coste, tanto en términos económicos como temporales.

Nuestro planteamiento teórico, como ya hemos visto anteriormente en la fase del análisis, es poder controlar un conjunto relés. Los relés, van a permitir o limitar el paso de energía eléctrica a las bases de los enchufes, y con ello, manejar así también los dispositivos que se conecten a ellas.

Para poder controlar un relé, lo único que necesitamos por parte de la placa Arduino, es enviar un 1 o un 0 lógico que alterne entre los estados de normalmente abierto y normalmente cerrado del relé.

Con la idea anterior en mente, y como pensamos implementar un sistema de cuatro bases de enchufes, vamos a realizar un esquema muy sencillo en una placa *protoboard* que nos permita validar el planteamiento.

El esquema que hemos planteado utiliza diodos LED en lugar de relés. De este modo podemos validar de forma rápida rápido y sencilla nuestra propuesta. Además, esto no solo nos va permitir comprobar que la idea es viable, sino que también –y este aspecto es muy importante– nos va a permitir realizar pruebas de verificación durante la etapa de desarrollo e implementación del *software*.

En esta prueba de concepto, aprovecharemos que ya disponemos de una tarjeta Arduino UNO y un módulo Arduino Ethernet Shield, así como LEDs y resistencias. Por lo que no hemos tenido que gastar nada para realizar el montaje del prototipo.

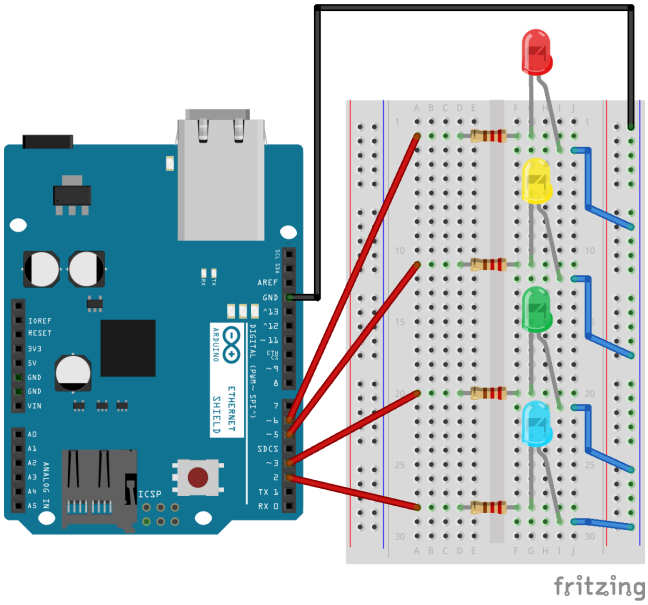


Ilustración 32: Esquema eléctrico de la prueba de concepto

Con este prototipo hemos podido realizar pruebas de código sencillas, que nos han permitido confirmar y verificar nuestras hipótesis.

13 Diseño

13.1 Arquitectura del hardware

En este apartado vamos a cubrir el diseño realizado, previo a la implementación del prototipo. Desde el conexionado del circuito eléctrico, a la preparación de la caja que albergará el montaje.

13.1.1 Esquema eléctrico del circuito

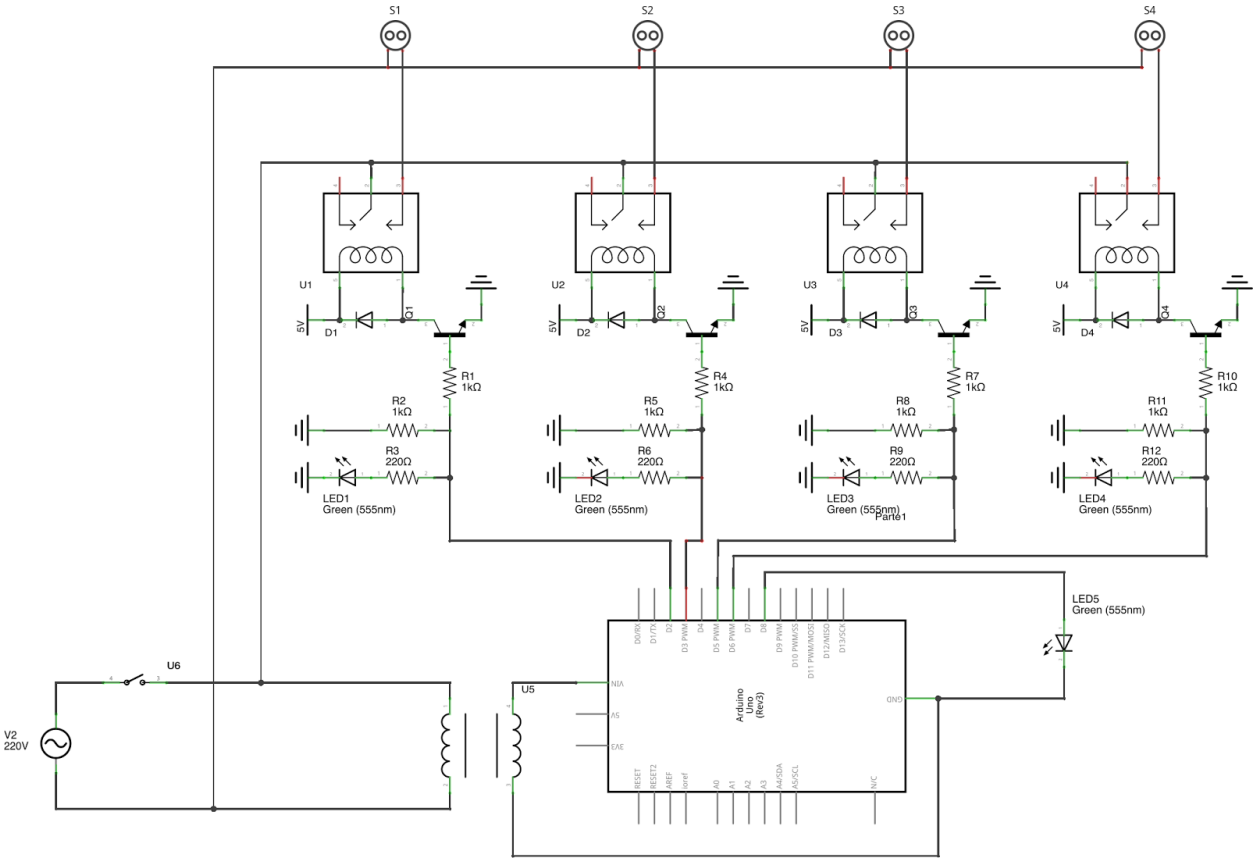


Ilustración 33: Esquema eléctrico del prototipo

13.1.2 Exterior de la caja

La caja en la cuál se introducirá el montaje, requiere de un acondicionamiento. En la tapa superior, colocaremos primero el juego de enchufes, luego el interruptor de encendido/apagado y por último el diodo LED de estado de conexión a la red WiFi. Para ello, hemos realizado el diseño que se muestra a continuación:

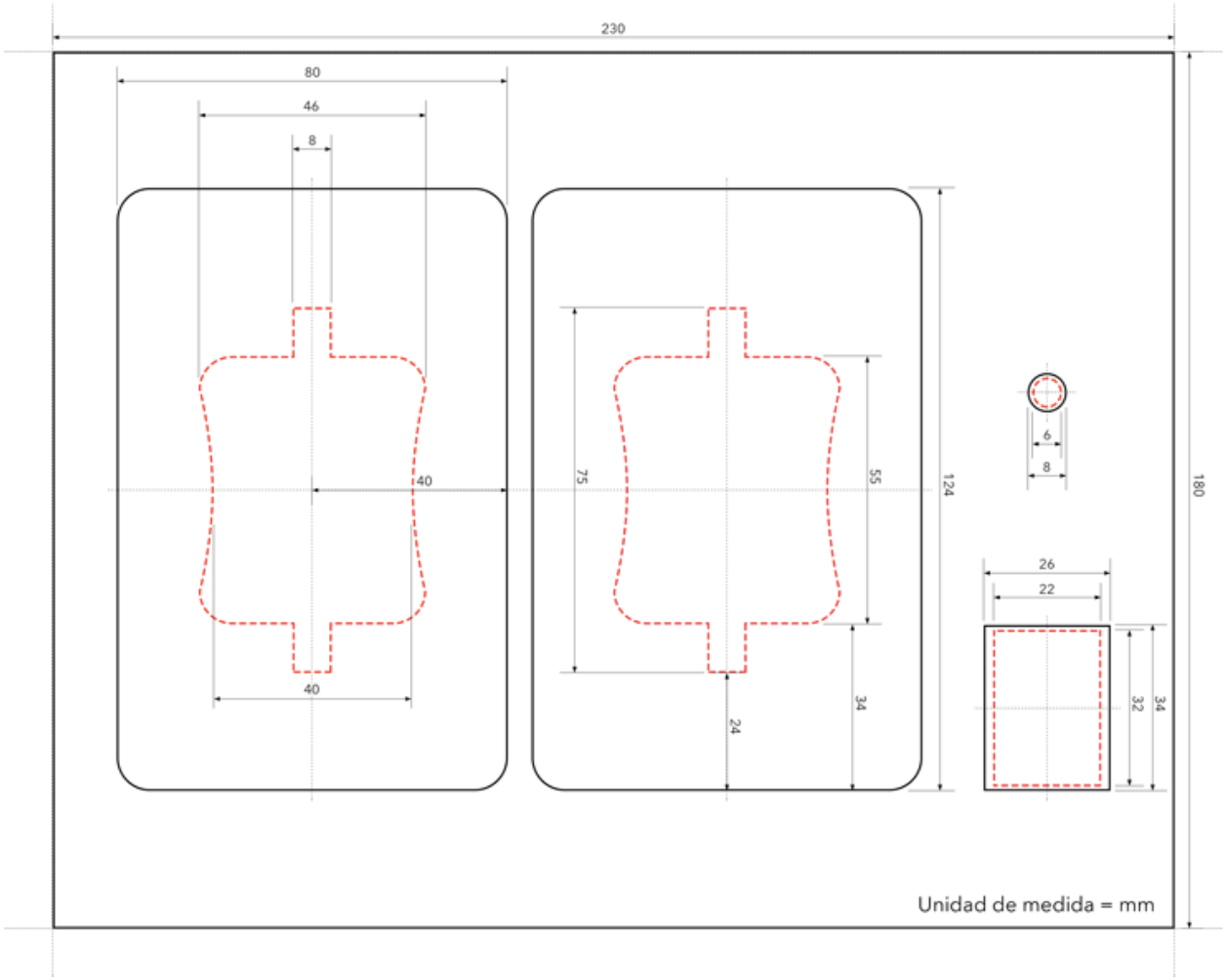


Ilustración 34: Plantilla de recorte de la tapa de la caja

13.1.3 Interior de la caja

Con el diseño del exterior de la caja, nos resta diseñar la forma en la cual irán configurados los componentes electrónicos en su interior.

El interior de la caja alberga los siguientes elementos:

- Placa Arduino UNO + Arduino WiFi Shield
- Módulo de 4 relés
- Transformador
- Regleta de conexionado

Entre los componentes y el conexionado, no disponemos de demasiado espacio en el interior de la caja, por ese motivo la disposición en que son colocados es muy importante.

Además, es también relevante conseguir un cierto grado de robustez en el conjunto, la caja debe poder soportar ligeros golpes y movimientos sin alterar la configuración de la estructura.

Para dar respuesta a estas necesidades, hemos optado por colocar un tablero de contrachapado de 5mm de espesor en el interior, fijado a la base de la caja, sobre el que reposan atornillados los componentes del montaje.

Después de diferentes pruebas de distribución, mostramos a continuación la plantilla con la versión final:

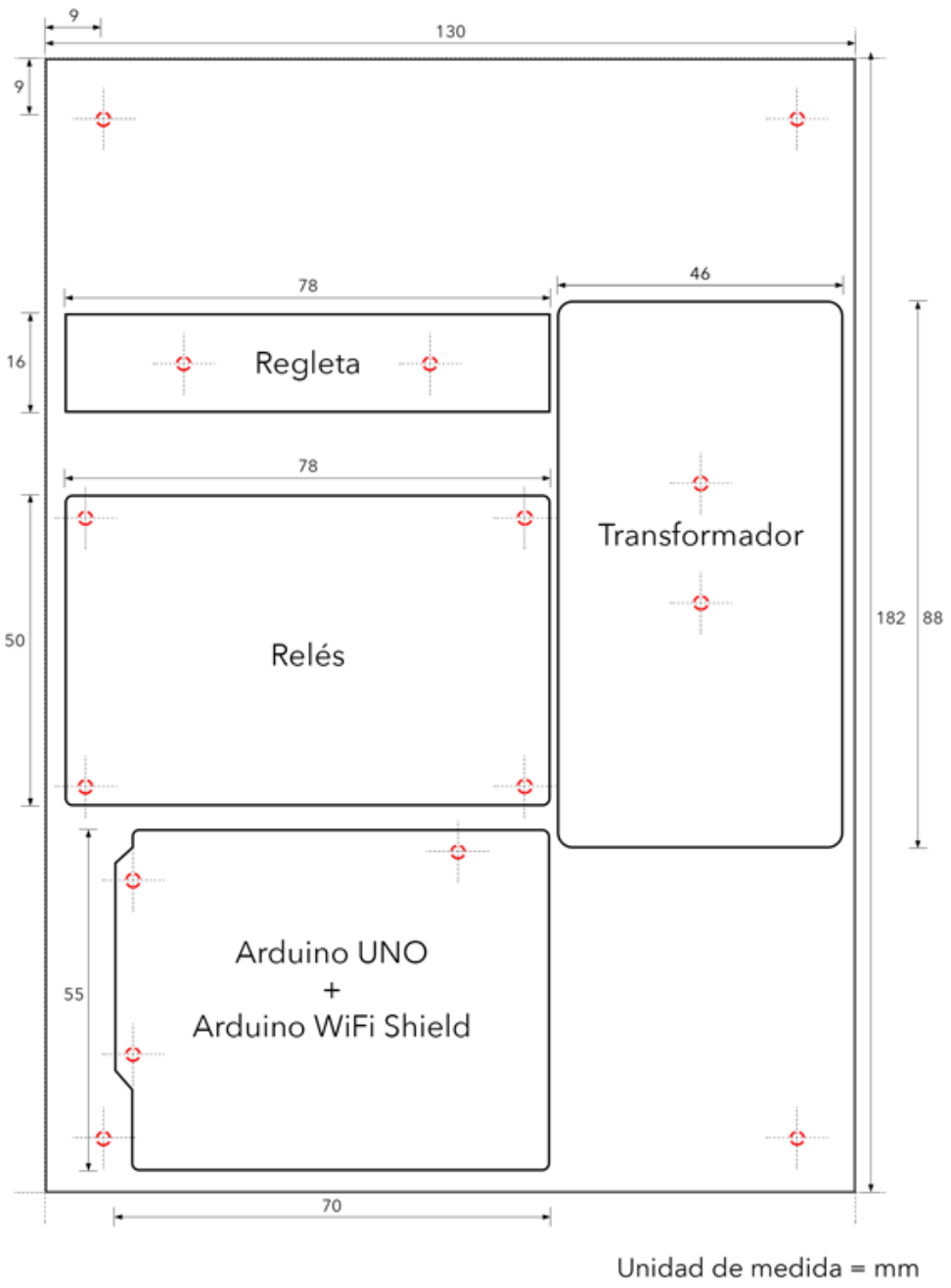


Ilustración 35: Plantilla con la disposición de los componentes electrónicos en el interior de la caja

13.2 Arquitectura del software

Para el desarrollo del apartado del *software*, se ha decidido establecer una subdivisión de tres niveles o capas que se detallarán a continuación (ordenados de menor a mayor nivel de abstracción):

- Servicio Web Arduino: Comunicación con la placa Arduino a bajo nivel. Atiende peticiones sobre las entradas y salidas de la placa a través de un API tipo REST y el

formato de comunicación estándar JSON. Disponible (en el entorno local) en la dirección **arduino.cleversocket.es**.

- *Servicio Web Clever Socket*: Comunicación con el sistema de enchufes. Permite la comunicación con el Servicio Web Arduino a través de un API de tipo REST, utilizando el formato de comunicación estándar JSON. Disponible (en el entorno local) en la dirección **api.cleversocket.es**.

- Interfaz de usuario: Aplicación web para el manejo del sistema de enchufes. Hace uso del Servicio Web Clever Socket. Disponible (en el entorno local) en la dirección **web.cleversocket.es**.

Ilustración 36: Componentes *software* de Clever Socket

13.2.1 Servicio Web Arduino

Por su extensión, los diagramas UML de casos de uso, actividad y secuencia, así como las operaciones del servicio web disponibles han sido desplazados al Anexo VI: Diagramas y operaciones del Servicio Web Arduino.

Definición

Peticiones HTTP permitidas

- GET - Utilizada para la obtención de recursos
- PUT - Utilizada para la actualización de recursos

Autenticación de las peticiones

Para autenticar las peticiones con el servicio web se utiliza la autenticación de acceso básica *HTTP*.

La autenticación de acceso básica es un método diseñado para permitir a un programa cliente, proveer credenciales en la forma de usuario y contraseña.

Cabe comentar que este sistema de autenticación es muy básico en términos de seguridad. Webduino implementa únicamente este sistema para la autenticación debido fundamentalmente a las limitaciones ofrecidas por el *hardware*.

Las credenciales son enviadas al servidor codificadas en base64. Esta codificación es comparada con la que se almacena en el servidor, dando acceso al servicio en caso de coincidencia.

Respuestas del servidor

- 200 OK - La consulta fue satisfactoria
- 401 *Unauthorized* - La autenticación falló o el usuario no tiene permiso para la operación solicitada
- 500 *Internal Server Error* - Error interno en el servidor

Operaciones

`GET` `/pins` : Devuelve una lista de los valores de los pines analógicos y digitales.

`PUT` `/pins` : Modifica el valor de uno o varios pines.

13.2.2 Servicio Web Clever Socket

Por su extensión, los diagramas UML de casos de uso, actividad y secuencia, así como la estructura de la base de datos y las operaciones del Servicio Web Clever Socket, han sido desplazados al apartado Anexo VII: Diagramas y operaciones del Servicio Web Clever Socket.

Definición

Peticiones HTTP permitidas

- GET - Utilizada para la obtención de recursos
- PUT - Utilizada para la actualización de recursos

Autenticación de las peticiones

Todos los usuarios disponen de credenciales para establecer comunicación con el servicio. Estos credenciales están compuestos de dos claves:

- Una clave pública o *accessKey*
- Una clave privada o *secretKey*

El servicio utiliza un esquema personalizado *HTTP* basado en un *hash* del mensaje del código de autenticación. La autenticación requiere de la concatenación de algunos elementos seleccionados de la petición (ruta, tipo de contenido y fecha) para formar un *string*. Después, haciendo uso de la clave privada o *secretKey*, se calcula el *HMAC* de ese *string*. A esta última operación, la denominamos de manera informal, firma del *string*, ya que simula las características de seguridad reales de una firma.

En cada petición, será necesario enviar la cabecera *Authorization*. Esta cabecera tiene la siguiente sintaxis:

{*accessKey*}:{*signature*}

La firma, o *signature*, debe estar formado como se muestra a continuación:

```
$path = $verb . '/' . '{option}';

$stringToSign = urlencode($path . '{headers/Content-Type}' .
'{headers/Date}');

$signature = base64_encode((hash_hmac('sha1', $stringToSign,
'{secretKey}')));
```

Tabla 7: Pseudocódigo de la obtención de la firma

Respuestas del servidor

- 200 *OK* - La consulta fue satisfactoria
- 401 *Unauthorized* - La autenticación falló o el usuario no tiene permiso para la operación solicitada
- 404 *Not found* - El elementos solicitado en la petición no fue encontrado
- 500 *Internal Server Error* - Error interno en el servidor
- 503 *Service unavailable* - No fue posible establecer comunicación con el Servicio Web Arduino

Operaciones

GET `/sockets` : Devuelve una lista de los sockets disponibles e información sobre sus estados.

GET `/sockets/{id}` : Devuelve el socket con el {id} solicitado e información sobre su estado.

PUT `/sockets/{id}` : Modifica el estado del socket con el {id} indicado.

GET `/tasks` : Muestra las tareas programadas.

GET `/tasks/{id}` : Devuelve la tarea con el {id} solicitado.

POST `/tasks` : Añade una nueva tarea.

DELETE `/tasks/{id}` : Elimina la tarea con el {id} solicitado.

GET `/logs` : Muestra los registros de actividad.

GET `/logs/{id}` : Devuelve el registro de actividad con el {id} solicitado.

13.2.3 Aplicación Web Clever Socket

Por su extensión, los diagramas de casos de uso, actividad y secuencia, así como la estructura de la base de datos, han sido desplazados al apartado Anexo VIII: Diagramas de la Aplicación Web Clever Socket.

Los prototipos realizados para la interfaz de usuario, pueden encontrarse en el Anexo IX: Prototipo de interfaz de usuario.

Diagramas de despliegue del sistema

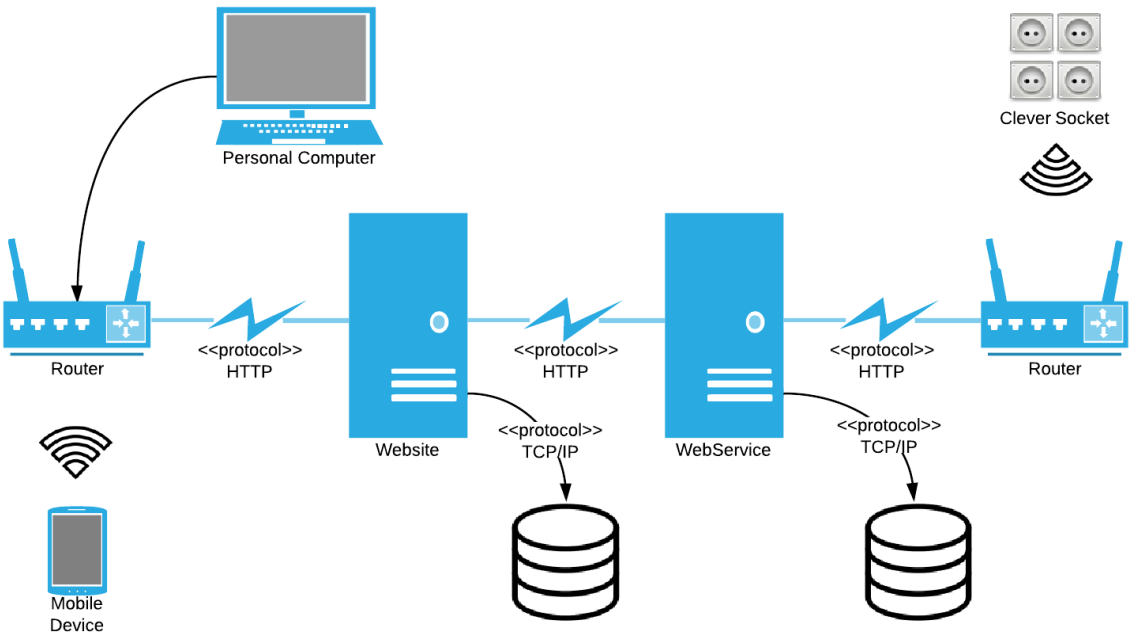


Diagrama 5: Diagrama de despliegue del sistema

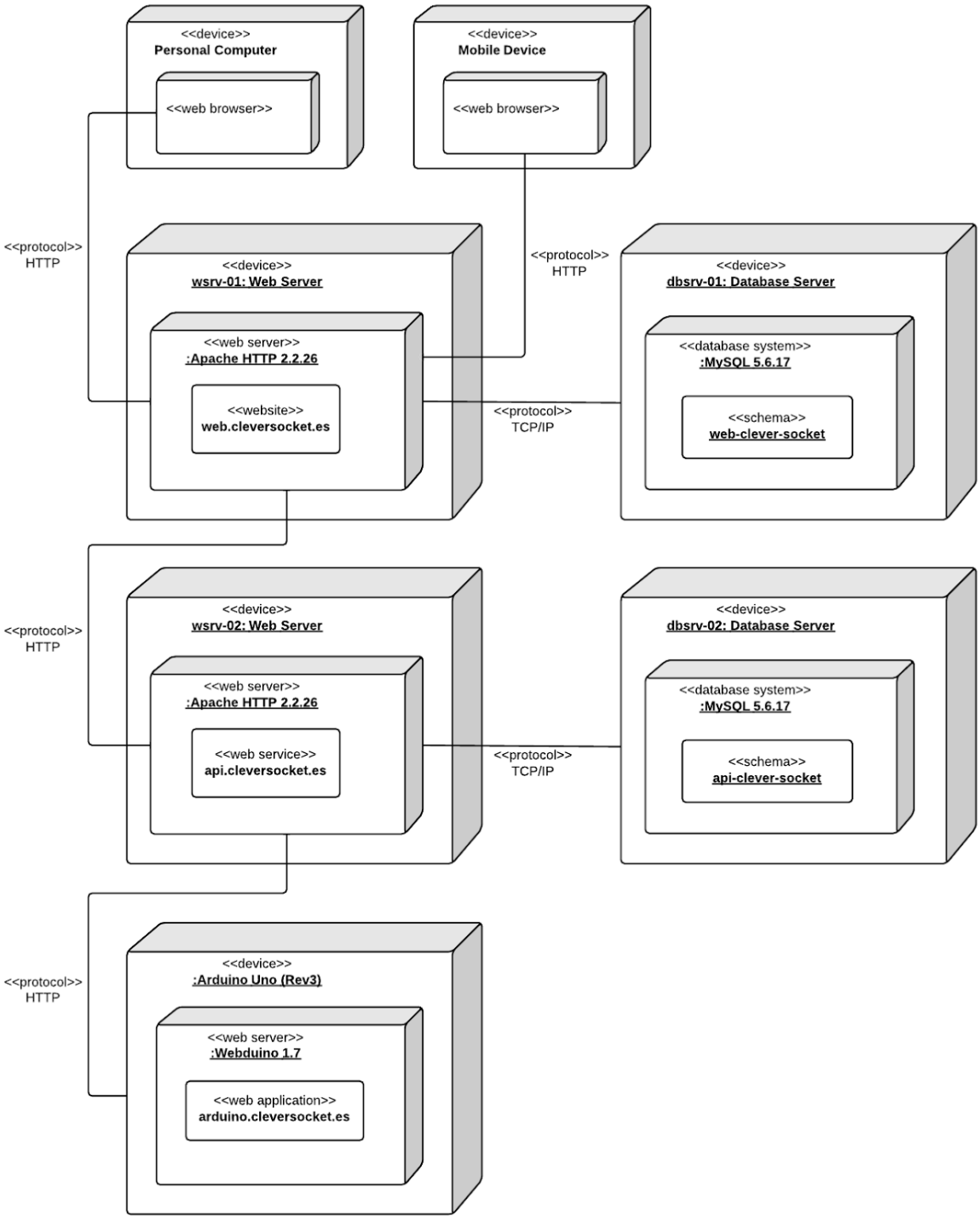


Diagrama 6: Diagrama de despliegue del sistema UML

14 Implementación

Este apartado del trabajo cubre la parte de la configuración y despliegue del sistema. Por una parte vamos a detallar los pasos necesarios para la correcta puesta en marcha del sistema

desde el punto de vista del *software*, y por otro lado detallaremos los paso llevados a cabo para la configuración del prototipo *hardware*.

14.1 Del lado del *software*

El código fuente de las tres aplicaciones desarrolladas, se puede localizar en el CD adjunto a esta memoria. A continuación indicamos las rutas de los directorios donde se encuentra localizado el código de cada aplicación:

- Servicio Web de Arduino: `/source/arduino-clever-socket`
- Servicio Web de Clever Socket: `/source/api-clever-socket`
- Aplicación Web de Clever Socket: `/source/web-clever-socket`

14.1.1 Servidor Webduino

Especificación del nombre de la red y clave WPA

Webduino hace uso de la librería *WiFi* –de *Arduino*–, y una de sus opciones de configuración requeridas es la especificación del nombre de la red (en inglés *Service Set Identifier*, o SSID) a la que se va a conectar, así como la clave o contraseña WPA.

En nuestro programa principal, especificamos estos valores de la siguiente manera:

```
char ssid[] = "****"; // Service Set IDentifier
char pass[] = "****"; // WPA Password


void setup() {
    // ...
    status = WiFi.begin(ssid, pass);
    // ...
}
```

Tabla 8: Especificación del nombre de la red y la clave WPA, Servidor Webduino

Especificación de la dirección IP

Del mismo modo que los fabricantes utilizan la dirección *MAC* para identificar de forma unívoca cada dispositivo de red, la dirección *IP* identifica al dispositivo dentro de nuestra red.

Por defecto, al conectar a nuestra red, la clase WiFi intenta obtener una dirección *IP* de manera automática por DHCP por parte del *router*. Sin embargo, en nuestro caso, nos interesa tener una dirección *IP* fija (192.168.1.210), de forma que podamos saber dónde se encontrará el servicio web.

De modo que para conseguir esto, tendremos que hacer lo siguiente:

```
IPAddress ip(192, 168, 1, 210);

void setup() {
    // ...
    WiFi.config(ip); // before WiFi.begin(ssid, pass);
    // ...
}
```

Tabla 9: Especificación de la dirección IP, Servidor Webduino

14.1.2 Archivo *hosts*

El archivo *hosts* es utilizado por el sistema operativo para traducir nombres de dominio y direcciones *IP*. Para el propósito de este trabajo, hemos definido los siguientes nombres de dominio:

- *Arduino Web Service*: http://arduino.cleversocket.es
- *Clever Socket Web Service*: http://api.cleversocket.es
- *Clever Socket Web Application*: http://web.cleversocket.es

En nuestro sistema operativo, el archivo *hosts* se encuentra localizado en la ruta /etc/hosts, y su definición es la siguiente:

```
##
# Host Database
#
# localhost is used to configure the loopback interface
# when the system is booting.  Do not change this entry.
##

127.0.0.1       localhost
255.255.255.255 broadcasthost
```

```
::1             localhost
fe80::1%lo0     localhost

# Arduino Web Service
192.168.1.210   arduino.cleversocket.es  www.arduino.cleversocket.es

# Clever Socket Web Service
127.0.0.1       api.cleversocket.es      www.api.cleversocket.es

# Clever Socket Web Application
127.0.0.1       web.cleversocket.es      www.web.cleversocket.es
```

Tabla 10: Contenido archivo hosts

14.1.3 Servidor Apache

Dos de nuestras aplicaciones funcionan bajo el Servidor HTTP Apache:

- Servicio Web de Clever Socket
- Aplicación Web de Clever Socket

Virtual Hosting

Para que las aplicaciones anteriores funcionen bajo la misma máquina e instancia de servidor Apache, necesitamos establecer una configuración de *Virtual Hosting*.

> "El término *Virtual Hosting* se refiere a hacer funcionar más de un sitio web (tales como www.company1.com y www.company2.com) en una sola máquina. Los sitios web virtuales pueden estar "basados en direcciones IP", lo que significa que cada sitio web tiene una dirección IP diferente, o "basados en nombres diferentes", lo que significa que con una sola dirección IP están funcionando sitios web con diferentes nombres (de dominio). El hecho de que estén funcionando en la misma máquina física pasa completamente desapercibido para el usuario que visita esos sitios web."[7]

[7] http://httpd.apache.org/docs/2.0/es/vhosts/, 2013

En nuestra instalación local, esta configuración la hemos realizado a través del fichero /private/etc/apache2/extra/httpd-vhosts.conf, como se muestra a continuación:

```
#
# Virtual Hosts
#

NameVirtualHost *:80

<VirtualHost *:80>
    ServerName localhost
    DocumentRoot /Library/WebServer/Documents/
</VirtualHost>

<VirtualHost *:80>
    ServerAdmin webmaster@api.cleversocket.es
    DocumentRoot "/Library/WebServer/Documents/api-clever-socket"
    ServerName api.cleversocket.es
    ServerAlias www.api.cleversocket.es
    ErrorLog "/private/var/log/apache2/api.cleversocket.es-
error_log"
    CustomLog "/private/var/log/apache2/api.cleversocket.es-
access_log" common
    <Directory "/Library/WebServer/Documents/api-clever-socket">
        Options Indexes FollowSymLinks
        AllowOverride All
        Order allow,deny
        Allow from all
    </Directory>
</VirtualHost>

<VirtualHost *:80>
    ServerAdmin webmaster@web.cleversocket.es
    DocumentRoot "/Library/WebServer/Documents/web-clever-
socket/public"
    ServerName web.cleversocket.es
    ServerAlias www.web.cleversocket.es
    ErrorLog "/private/var/log/apache2/web.cleversocket.es-
error_log"
    CustomLog "/private/var/log/apache2/web.cleversocket.es-
access_log" common
    <Directory "/Library/WebServer/Documents/web-clever-
socket/public">
        Options Indexes FollowSymLinks
        AllowOverride All
        Order allow,deny
        Allow from all
```

```
    </Directory>
</VirtualHost>
```

Tabla 11: Contenido archivo httpd-vhosts.conf

14.2 Del lado del *hardware*

En este apartado veremos cómo se realizó el montaje del prototipo, desde el acondicionamiento de la caja estanca, la disposición de los diferentes elementos en su interior y el conexionado de todas las partes.

14.2.1 Preparación de la caja

El montaje eléctrico del prototipo se encuentra alojado en el interior de una caja estanca (recordemos que sus dimensiones eran de 220x170x85mm).

Como ya vimos en la parte del diseño, tenemos una plantilla con las perforaciones que tenemos que realizar en la tapa de la caja.

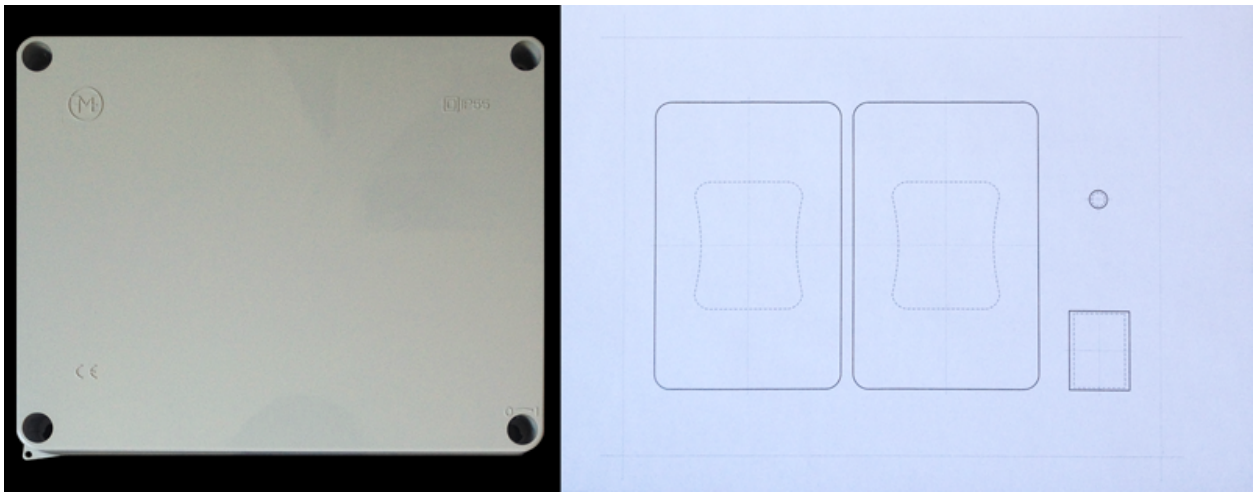


Ilustración 37: Detalle de la parte superior de la caja y la plantilla de recorte

De modo que solo nos resta marcar los cortes necesarios sobre la superficie de la caja, para a continuación proceder a realizar las perforaciones donde irán embutidos los distintos elementos.

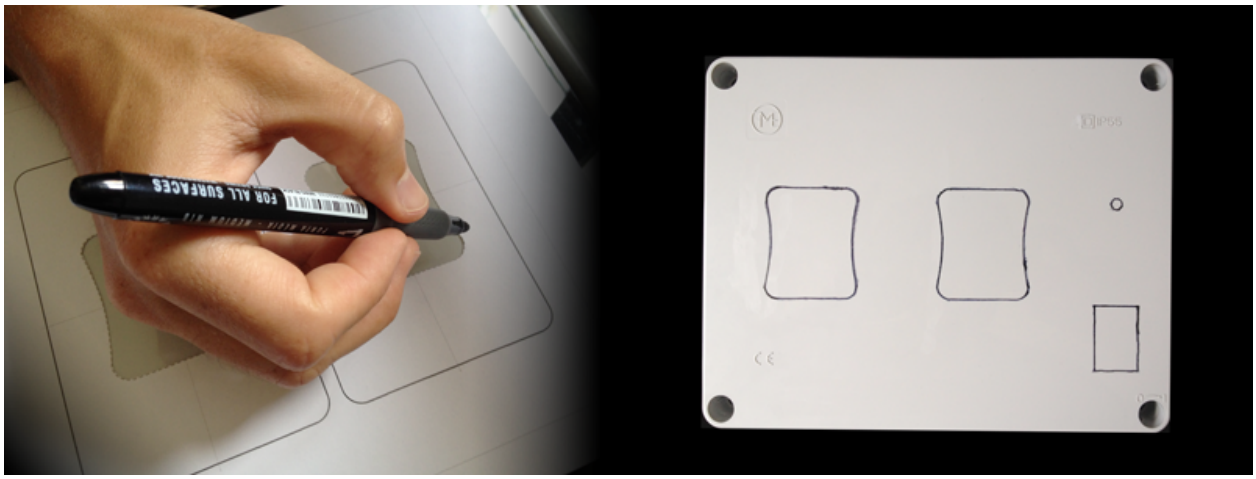


Ilustración 38: Detalle del dibujo de la plantilla sobre la caja

Con el trazado realizado sobre la tapa, empezamos a realizar las perforaciones. Para ello nos ayudamos de un sierra de mano que nos facilitó la tarea.

Ilustración 39: Detalle de las perforaciones en la tapa de la caja

Una vez abierto el espacio para introducir las bases de los enchufes, nos dimos cuenta de que sería más conveniente poder introducir las bases por el interior de la caja en lugar de desde arriba, de tal forma que al atornillar el embellecedor por la parte superior, no sobresaliese demasiado, dejando mucha holgura. Por ello, tomamos la decisión de realizar dos aberturas en los extremos superior e inferior para poder dejar paso a los conectores de toma a tierra.

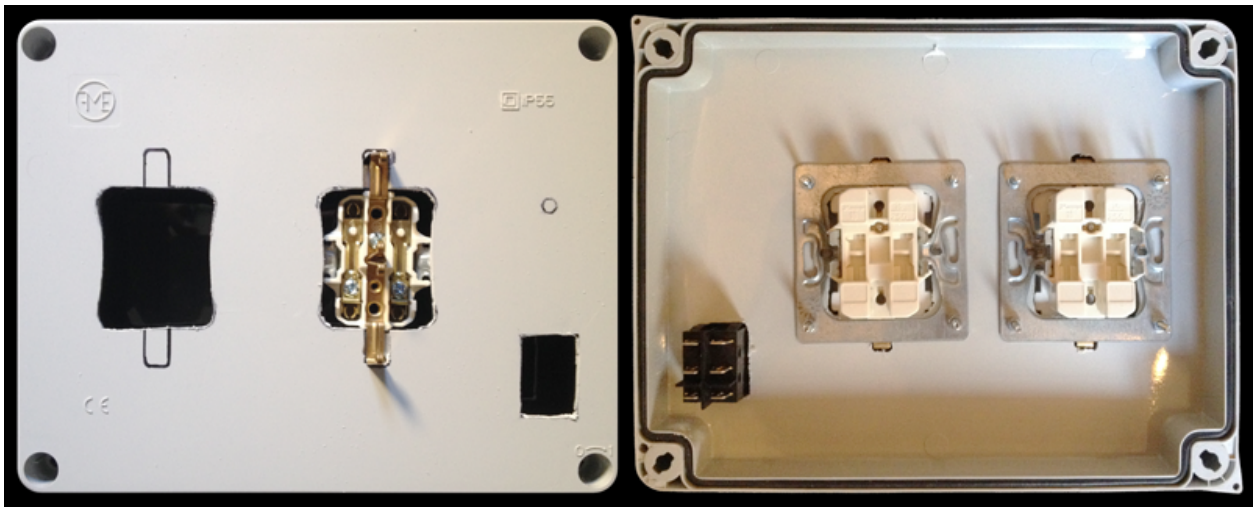


Ilustración 40: Frontal y reverso de la tapa de la caja

Finalmente, tras colocar y atornillar las dos bases a la superficie de caja, y tras embutir el interruptor, colocamos lo embellecedores.

Ilustración 41: Frontal de la caja con las bases de enchufe y el interruptor instalados

14.2.2 Disposición de los elementos y conexionado

Los componentes electrónicos, como ya vimos en el apartado del diseño, van a ir situados sobre un tablero de contrachapado de 5mm. Este tablero además, estará sujeto a la base de la caja por 4 tornillos.

Tenemos una plantilla con la disposición de los elementos en el interior de la caja. Para mayor comodidad a la hora de trasladar el diseño al tablero, hemos impreso la plantilla.

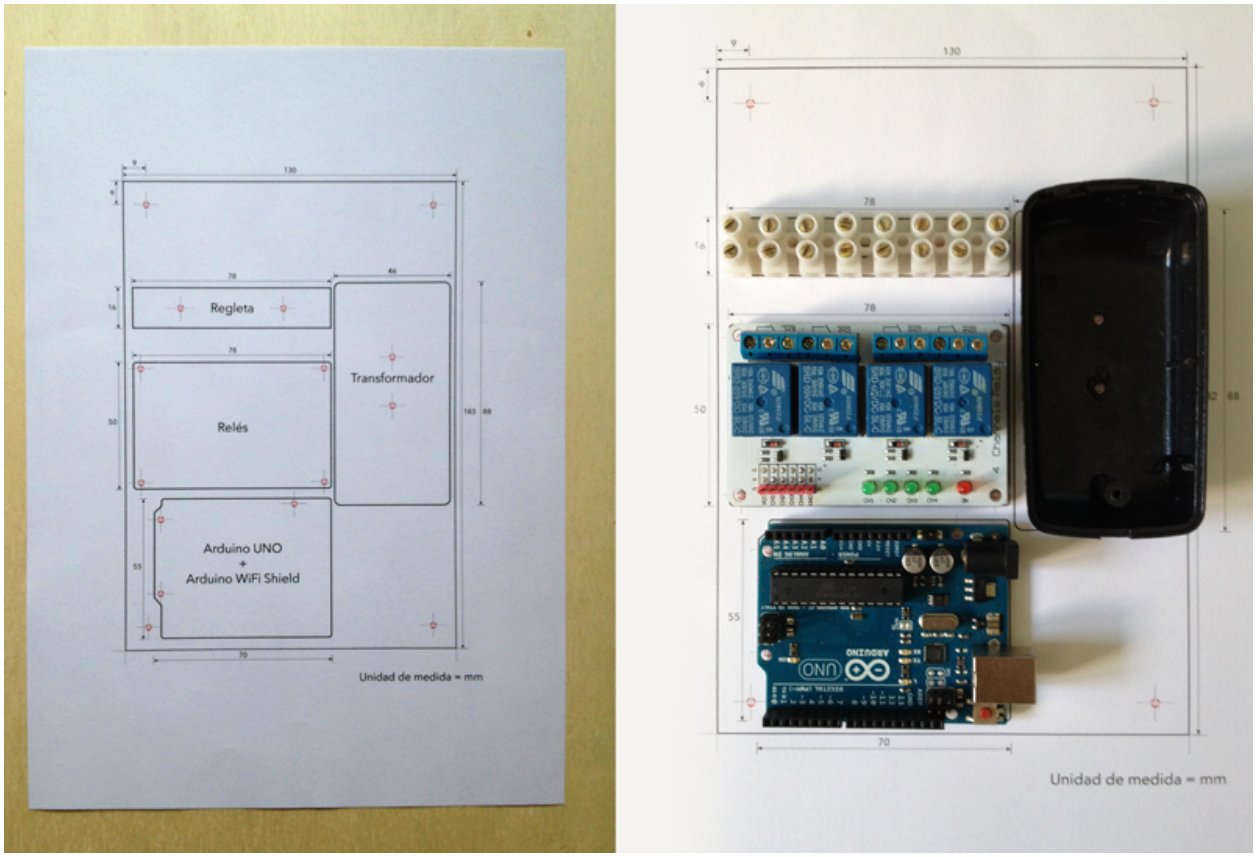


Ilustración 42: Plantilla con la disposición de los elementos en el interior de la caja

Con la plantilla impresa, nos queda trasladar sus puntos corte y perforación al tablero. Para ello, hemos empezado recortando el contorno y dibujando sobre el tablero las líneas de corte.

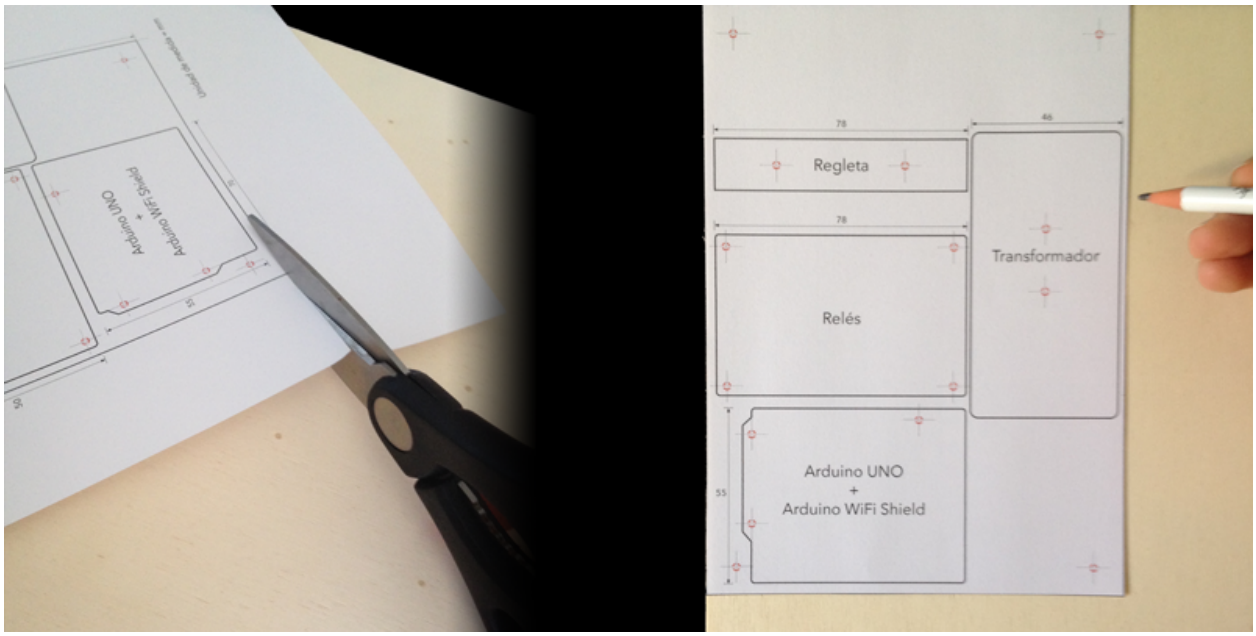


Ilustración 43: Detalle del recorte de la plantilla del interior de la caja

Ya con las medidas dispuestas sobre el tablero, pasamos a realizar el corte haciendo uso de una pequeña sierra de mano.

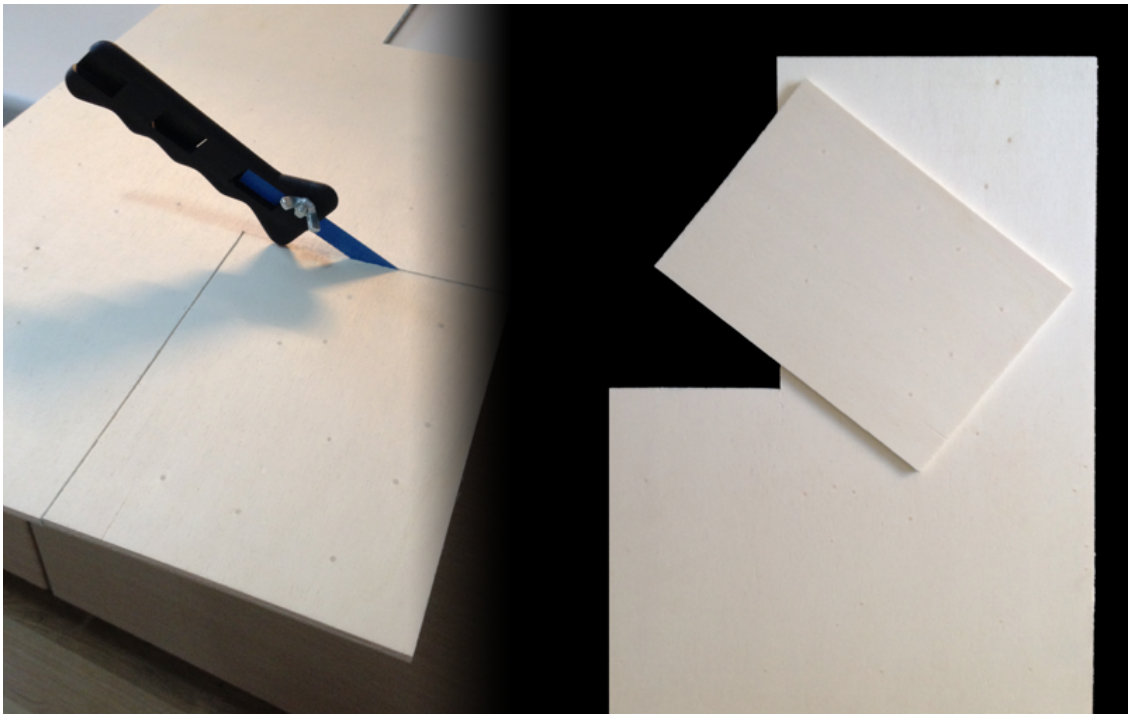


Ilustración 44: Recorte del tablero sobre el que se instalarán los componentes

Con la base recortada, hemos superpuesto la plantilla sobre el tablero para marcar los puntos a perforar. Huecos en los cuales, irán atornillados los componentes.

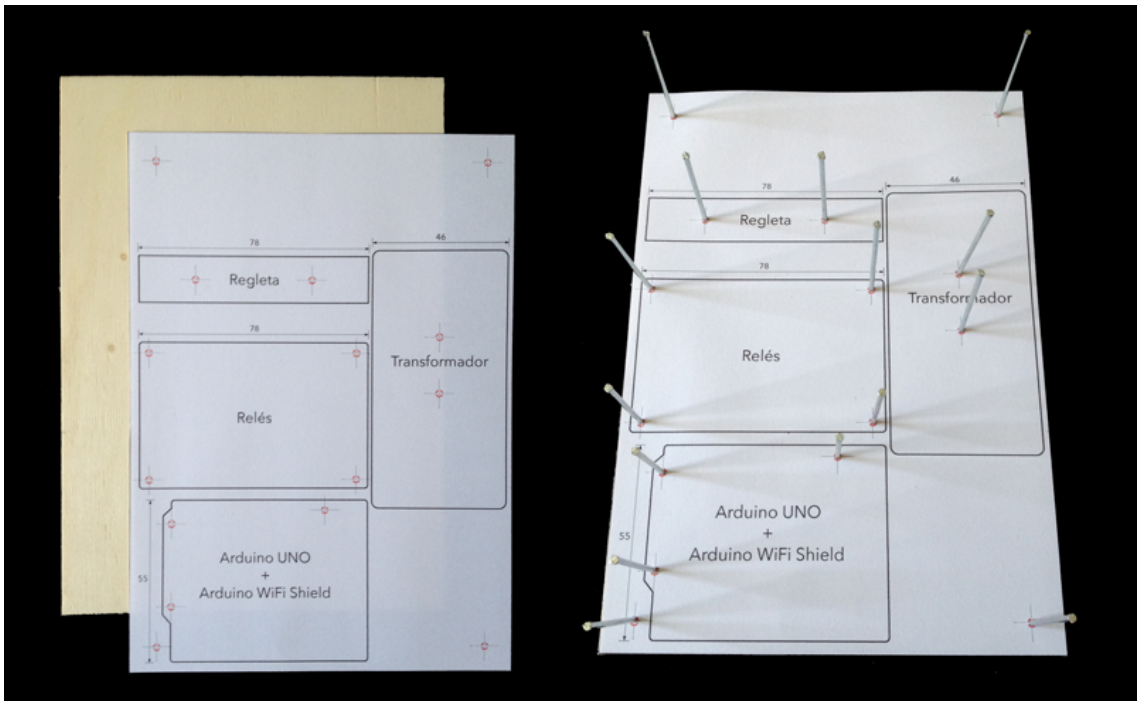


Ilustración 45: Plantilla sobre el tablero de componentes

Con los puntos marcados, hemos procedido a realizar los huecos y situado los componentes en su posición.

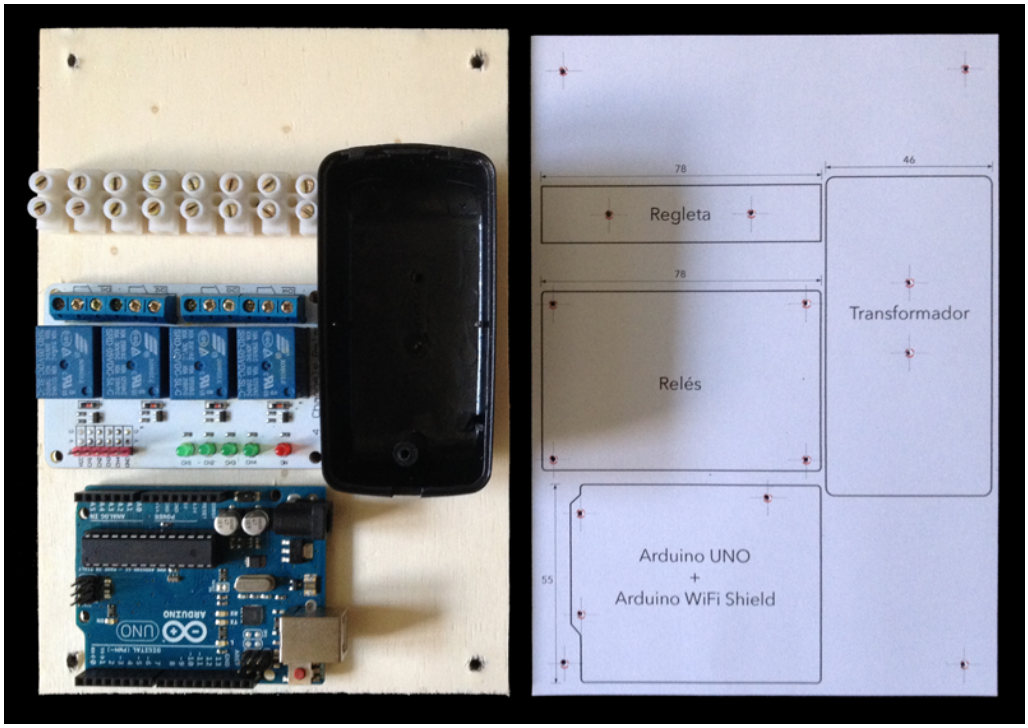


Ilustración 46: Componentes colocados sobre el tablero

Antes de fijar de forma definitiva todas las partes, hemos cortado los cables necesarios que van a unir las salidas del módulo de relés con la regleta para que sea luego más sencillo.

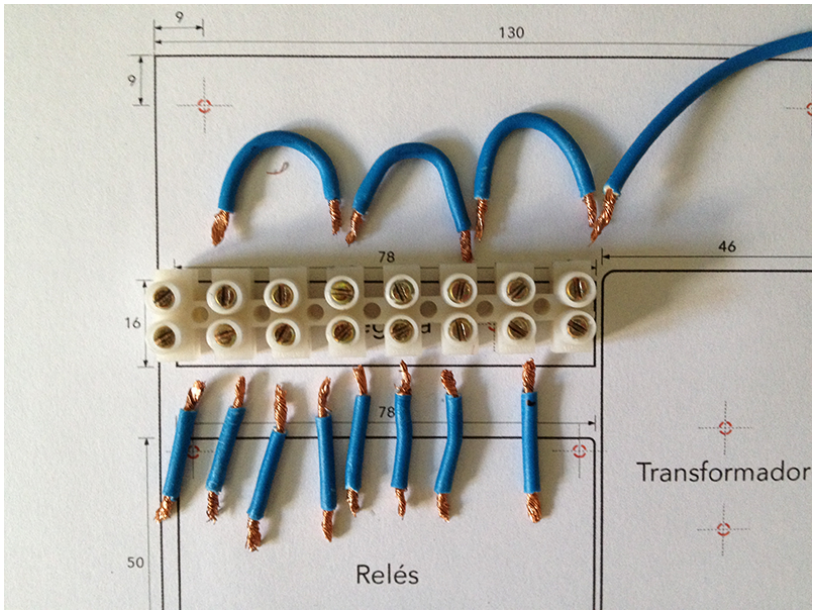


Ilustración 47: Detalle de la regleta de conexiones

Ahora ya, podemos terminar de atornillar la placa Arduino, el módulo con los 4 relés, la regleta de conexionado y el transformador. Quedando conectado el módulo de relés con la regleta.

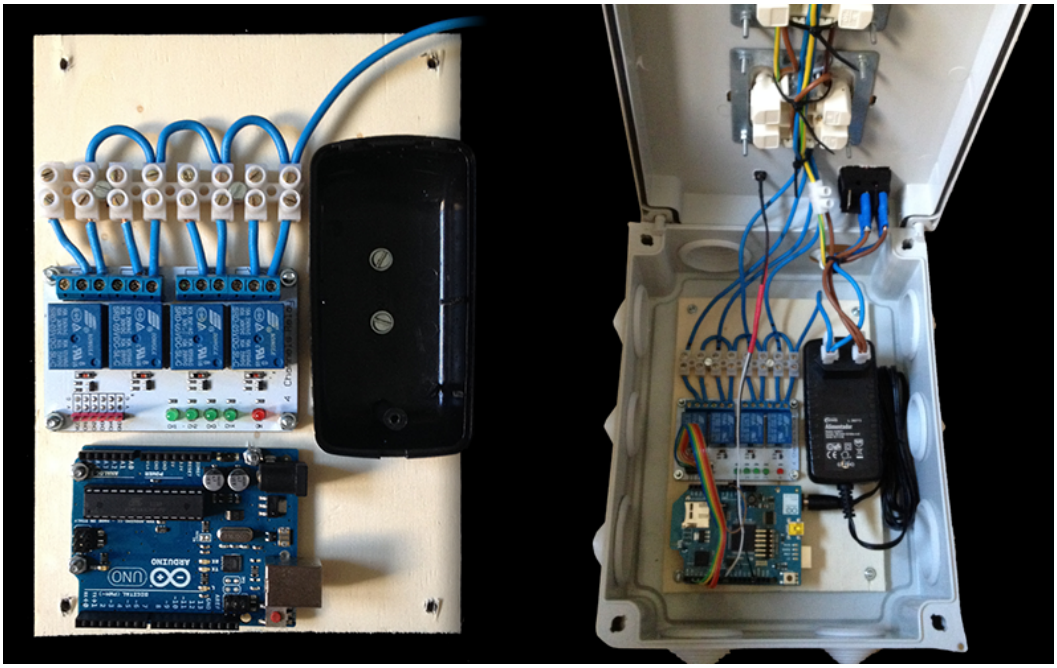


Ilustración 48: Componentes en el interior de la caja

En la imagen de la derecha vemos ya el transformador colocado en la carcasa y conectado al cable de alimentación. Además, se puede ver el *shield* WiFi incorporado encima de la placa Arduino UNO, y conectado al módulo de relés a través de los conectores macho a hembra.

Quedaba pendiente todavía colocar el LED de estado de conexión. Lo hemos embutido en la tapa de la caja. Por otro lado, para conectar sus terminales a los pines de la placa Arduino, hemos aprovechado nuestros conectores macho a hembra. Para ello, nos deshicimos de los extremos hembra, y soldamos los hilos de cable a los del diodo LED. Por último, para asegurar la unión, utilizamos tubo termorretráctil.

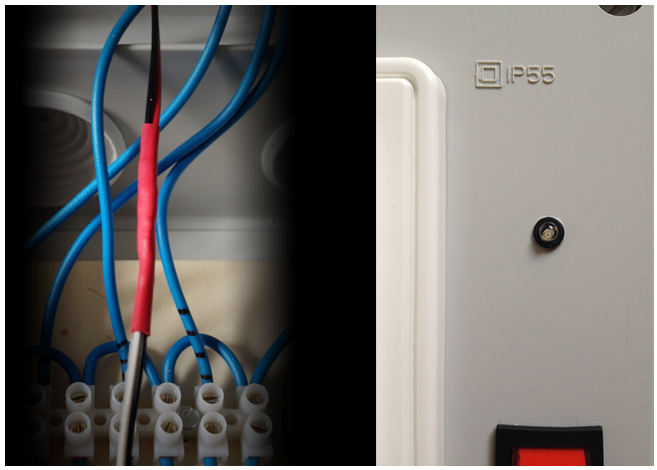


Ilustración 49: Detalle del cable y del piloto LED

Finalmente, después de montar la clavija de enchufe del cable de alimentación y de cerrar la caja, este fue el resultado final:

Ilustración 50: Resultado final del montaje del dispositivo Clever Socket

15 Normativa y legislación

15.1 Reglamento que afecta al TFG

En base a los Estatutos de la Universidad de Las Palmas de Gran Canaria, según se publica en el DECRETO 30/2003, de 10 de marzo, Artículo 133:

1. La realización de un proyecto fin de carrera o tesina llevará consigo, necesariamente, la elaboración de un trabajo en el ámbito disciplinario elegido, en régimen de tutoría. El Consejo de Gobierno establecerá un reglamento que defina e incentive la tutoría de proyectos y tesinas. El plan de estudios correspondiente establecerá la forma de evaluación de ese proyecto fin de carrera o tesina.
2. En virtud del artículo 7 del Real Decreto Legislativo 1/1996, de 12 de abril, por el que se aprueba el Texto Refundido de la Ley de Propiedad Intelectual, el proyecto fin de carrera se considera una obra en colaboración entre el estudiante y el tutor o tutores, en su caso.
3. La explotación industrial de un proyecto fin de carrera debe ser objeto de convenio entre la Universidad y el organismo o empresa que la realizará. Para fomentar la realización de proyectos fin de carrera con empresas, la Universidad podrá, en el convenio citado, ceder la propiedad industrial de los trabajos contenidos en el proyecto fin de carrera.

15.2 Licencias del *software*

En este trabajo se hace uso principalmente de cuatro elementos *software* de terceros:

- Webduino
- Slim Framework
- Laravel PHP Framework
- Twitter Bootstrap

Todos ellos comparten la misma licencia de *software*, The MIT License (MIT).

> "Una **licencia de software** es un contrato entre el licenciante (autor/titular de los derechos de explotación/distribuidor) y el licenciatario del programa informático (usuario consumidor /usuario profesional o empresa), para utilizar el software cumpliendo una serie de términos y condiciones establecidas dentro de sus cláusulas.
>
> Las licencias de software pueden establecer entre otras cosas: la cesión de determinados derechos del propietario al usuario final sobre una o varias copias del

programa informático, los límites en la responsabilidad por fallos, el plazo de cesión de los derechos, el ámbito geográfico de validez del contrato e incluso pueden establecer determinados compromisos del usuario final hacia el propietario, tales como la no cesión del programa a terceros o la no reinstalación del programa en equipos distintos al que se instaló originalmente."[8]

15.2.1 Licencia del *MIT*

Condiciones

La condición de uso de la licencia del *MIT* es que la nota de *copyright* y la parte de los derechos sean incluídas en todas las copias o partes sustanciales del *software*. De no cumplirse esta condición, la licencia quedaría invalidada.

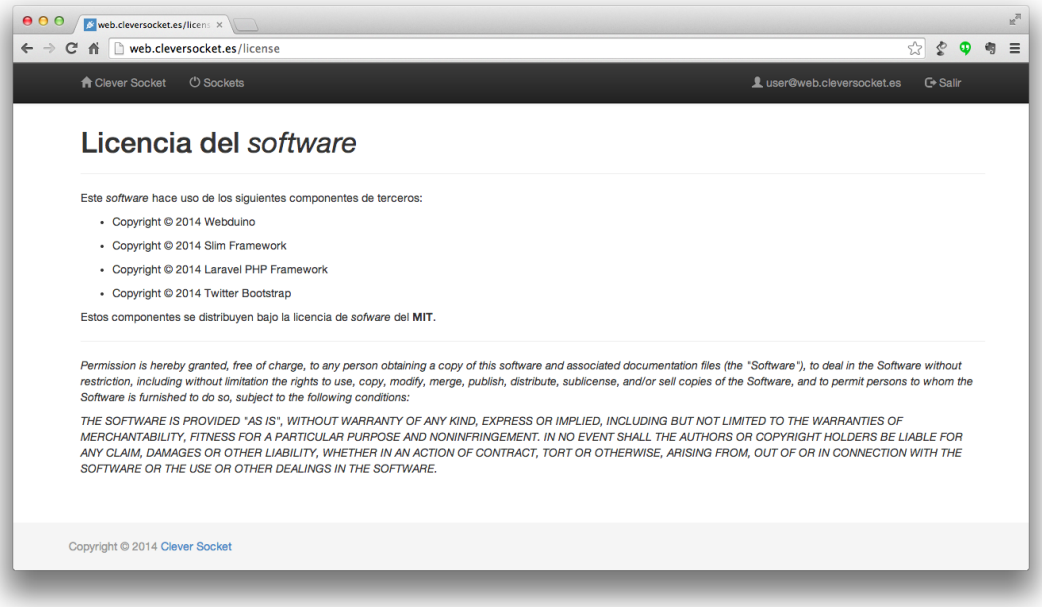


Ilustración 51: Página de la licencia del *software*

Derechos

Se concede permiso de forma gratuita a cualquier persona que obtenga una copia de este *software* y de los archivos de documentación asociados (el "*Software*"), para utilizar el *Software* sin restricción, incluyendo sin limitación los derechos de usar, copiar, modificar,

[8] http://es.wikipedia.org/wiki/Licencia_de_software, 2014

fusionar, publicar, distribuir, sublicenciar, y/o vender copias de este *Software*, y para permitir a las personas a las que se les proporcione el *Software* a hacer lo mismo.

Limitación de responsabilidad

El *software* se proporciona "tal cual", sin garantía de ningún tipo, expresa o implícita, incluyendo pero no limitado a garantías de comercialización, idoneidad para un propósito particular y no infracción. En ningún caso los autores o titulares del *copyright* serán responsables de ninguna reclamación, daños u otras responsabilidades, ya sea en un litigio, agravio o de otro modo, que surja de o en conexión con el *software* o el uso u otro tipo de acciones en el *software*.

Puede consultar la plantilla de la licencia en el apartado Anexo X: Plantilla de Licencia del MIT.

15.3 Utilización de *cookies*

El apartado segundo del artículo 22 de la Ley 34/2002, de 11 de julio, de servicios de la sociedad de la información y de comercio electrónico (en adelante LSSI), tras su modificación por el Real Decreto-ley 13/2012, de 30 de marzo, establece que:

> 2. Los prestadores de servicios podrán utilizar dispositivos de almacenamiento y recuperación de datos en equipos terminales de los destinatarios, a condición de que los mismos hayan dado su consentimiento después de que se les haya facilitado información clara y completa sobre su utilización, en particular, sobre los fines del tratamiento de los datos, con arreglo a lo dispuesto en la Ley Orgánica 15/1999, de 13 de diciembre, de Protección de Datos de Carácter Personal.
>
> Cuando sea técnicamente posible y eficaz, el consentimiento del destinatario para aceptar el tratamiento de los datos podrá facilitarse mediante el uso de los parámetros adecuados del navegador o de otras aplicaciones, siempre que aquél deba proceder a su configuración durante su instalación o actualización mediante una acción expresa a tal efecto.
>
> Lo anterior no impedirá el posible almacenamiento o acceso de índole técnica al solo fin de efectuar la transmisión de una comunicación por una red de comunicaciones electrónicas o, en la medida que resulte estrictamente necesario, para la prestación de un servicio de la sociedad de la información expresamente solicitado por el destinatario.

La *LSSI* precisa además, que el precepto transcrito es aplicable a cualesquiera "dispositivo de almacenamiento y recuperación de datos" en cualquiera "equipos terminales de los destinatarios" y que el Anexo de la citada LSSI define como "Destinatario del servicio o destinatario" a la "persona física o jurídica que utiliza, sea o no por motivos profesionales, un servicio de la sociedad de la información".[9]

De esta forma, el Artículo 22 de la LSSI hace referencia a la utilización de *cookies* y tecnologías similares, utilizadas para almacenar y recuperar datos de un equipo terminal de una persona física o jurídica, que utiliza un servicio de la sociedad de la información.

Por último, cabe mencionar que quedan exceptuadas del cumplimiento de las obligaciones establecidas en el Artículo 22.2 de la LSSI las *cookies* utilizadas para alguna de las siguientes finalidades:

- Permitir únicamente la comunicación entre el equipo del usuario y la red.
- Estrictamente prestar un servicio expresamente solicitado por el usuario.

En este sentido, la "Guía sobre el uso de las cookies"[10] redactada por la Agencia Española de Protección de Datos (AGDP), interpreta que entre las cookies exceptuadas estarían aquellas que tienen por finalidad:

- Cookies de entrada del usuario.
- Cookies de autenticación o identificación de usuario (únicamente de sesión).
- Cookies de seguridad del usuario.

Nuestra aplicación web hace uso de este tipo de *cookies* y por ese motivo, estaríamos exentos de tener que notificar al usuario sobre su utilización. No obstante, y en previsión de futuras modificaciones sobre el tipo de *cookies* utilizadas, hemos creído conveniente seguir las indicaciones de la *LSSI* y cumplir con el deber de avisar e informar al usuario sobre el uso de este tipo de opciones de almacenamiento de información.

Además de notificar sobre la utilización de las *cookies*, es necesario solicitar el consentimiento del usuario. En este sentido, para que el usuario pueda decidir sobre si dar o no su consentimiento, se hace necesario mostrar información suficiente y completa.

---

[9] BOE núm. 166, 2002

[10] Agencia Española de Protección de Datos, 2013

En nuestro caso, la *Ley* nos indica que debemos proceder de la siguiente manera en relación con el usuario:

1. Alertar sobre la utilización de *cookies*.
2. Solicitar su consentimiento.
3. Informar de forma suficiente y completa sobre los siguientes aspectos:
    a. Qué son las *cookies*.
    b. Qué tipo de *cookies* utiliza el sistema.
    c. Cómo administrar *cookies* en el sistema o revocar el consentimiento.

De entre las diferentes opciones existentes para notificar al usuario, en nuestra aplicación, hemos optado por mostrar un cuadro de alerta en la parte superior. Esta alerta dispone de tres elementos:

1. Aviso breve notificando de la utilización de *cookies* por parte del sistema.
2. Un enlace hacia una sección con información extendida sobre *cookies*.
3. Un botón que solicita la conformidad del usuario.

Véase a continuación una captura de este mensaje de alerta:

Ilustración 52: Detalle del mensaje de aviso sobre el uso de *cookies*

Al mostrarse este mensaje, existen tres posibles acciones a tomar por parte del usuario:

1. No tomar ningún tipo de acción, con lo que el mensaje continuará apareciendo durante la navegación del usuario.
2. Acceder al enlace de "Más información".
3. Hacer clic sobre el botón de "Entendido", dando así consentimiento al sistema sobre la utilización de *cookies* y eliminando el mensaje de alerta.

Por último, en el caso de que el usuario acceda a la sección de informativa sobre *cookies*. Se mostraremos la siguiente información:

15.3.1 ¿Qué es una *cookie*?

Una *cookie* es un pequeño fragmento de texto que los sitios web que visitas envían al navegador y que permite que el sitio web recuerde información sobre tu visita, como tu idioma preferido y otras opciones, lo que puede facilitar tu próxima visita y hacer que el sitio te resulte más útil. Las *cookies* desempeñan un papel muy importante, ya que sin ellas el uso de la Web sería una experiencia mucho más frustrante.

15.3.2 Tipos de *cookies* que utilizamos

Utilizamos diferentes tipos de *cookies* para el correcto funcionamiento de Clever Socket. Una parte o la totalidad de las *cookies* identificadas a continuación se pueden almacenar en tu navegador.

Tipo	**Descripción**
Preferencias	Estas *cookies* permiten que nuestro sitio web recuerde información que cambia el aspecto o el comportamiento del sitio. La pérdida de la información almacenada en una *cookie* de preferencias puede hacer que la experiencia del sitio web sea menos funcional, pero no debe afectar a su funcionamiento.
Seguridad	Utilizamos *cookies* de seguridad para autenticar a usuarios, evitar el uso fraudulento de credenciales de inicio de sesión y proteger los datos de usuarios frente a terceros no autorizados.

Tabla 12: Tipo de *cookies* que utilizamos

15.3.3 ¿Cómo administrar cookies en el navegador?

Puede usted permitir, bloquear o eliminar las *cookies* instaladas en su equipo mediante la configuración de las opciones del navegador instalado en su ordenador.

- Para más información sobre el navegador Firefox pulse aquí
- Para más información sobre el navegador Chrome pulse aquí
- Para más información sobre el navegador Explorer pulse aquí
- Para más información sobre el navegador Safari pulse aquí
- Para más información sobre el navegador Opera pulse aquí

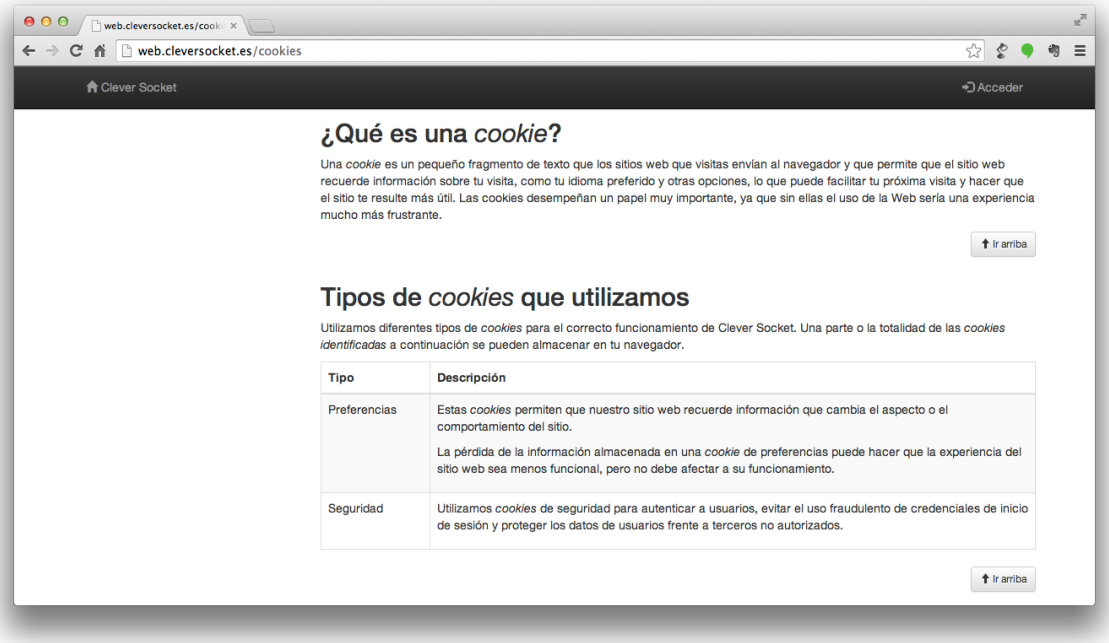


Ilustración 53: Tipos de *cookies* que utilizamos

15.4 Ley Orgánica de Protección de Datos (LOPD)

En el artículo 9 de la Ley Orgánica 15/1999, de 13 de diciembre, de Protección de Datos de Carácter Personal, se dice que "El responsable del fichero, y, en su caso, el encargado del tratamiento deberán adoptar las medidas de índole técnica y organizativas necesarias que garanticen la seguridad de los datos de carácter personal y eviten su alteración, pérdida, tratamiento o acceso no autorizado, habida cuenta del estado de la tecnología, la naturaleza de los datos almacenados y los riesgos a que están expuestos, ya provengan de la acción humana o del medio físico o natural"[11].

En este sentido, nosotros como encargados y responsables del tratamiento de la información de carácter personal de los usuarios, debemos ser conscientes de lo que rige sobre este respecto la Ley, y en consecuencia, cumplir con lo exigido.

Pero antes de profundizar en el asunto, veamos algunas definiciones relevantes que establece la Ley:

[11] BOE núm. 298 de 14 de Diciembre de 1999

- **Datos de carácter personal**: cualquier información concerniente a personas físicas identificadas o identificables.
- **Fichero**: todo conjunto organizado de datos de carácter personal, cualquiera que fuere la forma o modalidad de su creación, almacenamiento, organización y acceso.
- **Tratamiento de datos**: operaciones y procedimientos técnicos de carácter automatizado o no, que permitan la recogida, grabación, conservación, elaboración, modificación, bloqueo y cancelación, así como las cesiones de datos que resulten de comunicaciones, consultas, interconexiones y transferencias.
- **Responsable del fichero o tratamiento**: persona física o jurídica, de naturaleza pública o privada, y órgano administrativo, que decida sobre la finalidad, contenido y uso del tratamiento.
- **Encargado del tratamiento**: la persona física o jurídica, autoridad pública, servicio o cualquier otro organismo que, sólo o conjuntamente con otros, trate datos personales por cuenta del responsable del tratamiento.

15.4.1 Deberes del responsable

Entre los deberes principales del responsable del tratamiento de la información, la LOPD establece los siguientes:

15.4.2 Inscripción de ficheros

Tenemos la obligación de notificar ante el Registro General de Protección de Datos (en adelante, RGPD) de la Agencia Española de Protección de Datos (en adelante, AEPD) aquellos ficheros que contengan información de usuarios de carácter personal.

Los ficheros podrán ser de tipo **automatizado**, cuando sean ficheros que almacenan su información sobre soporte informático; de tipo **no automatizados**, cuando su almacenamiento no sea automatizado, esto es, cuando no se encuentren almacenados sobre soporte informático; y **mixto**, cuando la la información de carácter personal se disponga tanto en soporte informático como no.

En nuestro caso, los datos de carácter personal que sobre los usuarios almacenamos y tratamos, se encuentran dispuestos de manera exclusiva en ficheros de tipo automatizado.

Consideraciones a tener en cuenta a la hora de la inscripción

A la hora de realizar la inscripción de nuestro fichero en el RGPD, tendremos que rellenar previamente el formulario de notificación de ficheros, también conocido como NOTA

(acrónimo establecido para denominar al formulario de Notificaciones Telemáticas a la AEPD).

Este formulario se puede conseguir a través de su descarga desde la página web de la AEPD. Su cumplimiento debe ser realizado de forma telemática y su presentación podrá ser hecha a través de Internet, con y sin firma electrónica –para lo cual deberá remitir a la Agencia la Hoja de solicitud correspondiente al envío realizado debidamente firmada–, o de manera presencial en la Agencia.

Ilustración 54: Formulario de solicitud de inscripción de fichero

Calidad de los datos

En este sentido el artículo 4 de la presente Ley establece, entre otros preceptos, que "Los datos de carácter personal sólo se podrán recoger para su tratamiento, así como someterlos a dicho tratamiento, cuando sean adecuados, pertinentes y no excesivos en relación con el ámbito y las finalidades determinadas, explícitas y legítimas para las que se hayan obtenido", así como también dice que "Los datos de carácter personal serán exactos y puestos al día de forma que respondan con veracidad a la situación actual del afectado".

Derecho de información en la recogida de los datos

Los usuarios de nuestro servicio, a quienes vayamos a solicitar sus datos personales, deberán ser debidamente informados de manera expresa, precisa e inequívoca.

Por lo que habrá que informar al usuario de: "...la existencia de un fichero o tratamiento de datos de carácter personal, de la finalidad de la recogida de éstos y de los destinatarios de la información.", "...el carácter obligatorio o facultativo de sus respuesta a las preguntas que les sean planteadas.", "...las consecuencias de la obtención de los datos o de la negativa a suministrarlos.", entre otros preceptos.

Otros principios sobre la protección de datos

La Ley estipula además otros principios, en los cuales no profundizaremos, pero que nosotros como responsables de la información de carácter personal de los usuario deberemos atender, como son:

- El consentimiento del afectado.
- Los datos especialmente protegidos.
- La seguridad de los datos.
- El deber de secreto.
- La comunicación de datos.
- El acceso a los datos por cuenta de terceros.

16 Proyección futura del trabajo

Durante el transcurso de este trabajo, tan solo hemos ido depositando las primeras piedras de un camino que, esperamos, puede tener un largo recorrido.

Con la finalización de nuestro prototipo por un lado, hemos demostrado su viabilidad. Por otro lado, hemos podido entrever acciones futuras de mejora, y también, funcionalidades e incorporaciones de elementos que podrían incrementar las capacidades del dispositivo.

En el futuro, nos gustaría poder mejorar algunos aspectos de nuestro trabajo, tanto en el lado del *hardware* como del *software*:

En lo que atañe a la parte física del trabajo, al dispositivo, hay mucho por hacer. El resultado de nuestro trabajo es un prototipo que, aún siendo funcional, no garantiza la robustez que de un producto final se pueda esperar. Además, el precio de las partes que forman el conjunto es superior al que debería tener un dispositivo de estas características para una posible comercialización.

Por otro lado, sería necesario realizar un rediseño de la disposición de los componentes en el interior de la caja, así como de su aspecto exterior. El tamaño y el peso del dispositivo son superiores a lo deseable. Actualmente, aunque se ha intentado cuidar la colocación de cada elemento, su configuración es susceptible de mejora, haciendo más sencilla la apertura y cierre de la caja para su mantenimiento.

En el apartado del aspecto exterior, nos gustaría poder trabajar en un diseño más elaborado y atractivo. Reducir el tamaño del equipo se haría necesario, así como asemejar el aspecto final del producto al de otros dispositivos de este tipo existentes en el mercado. Creemos que sería conveniente trabajar en detalles estéticos que lo hicieran destacar y sobresalir sobre el resto.

No descartamos en un futuro realizar una selección diferente de los componentes internos. Creemos conveniente continuar estudiando alternativas que se adapten de mejor manera a nuestros propósitos, pudiendo además reducir los costes globales.

Como propuesta de futuro, nos gustaría poder trabajar sobre la idea de utilizar una tarjeta Raspberry Pi, que dé más estabilidad y seguridad en la capa de comunicación.

Creemos que sería conveniente también, valorar alternativas en los dispositivos específicos de comunicación, como la posibilidad de utilizar *routers* de muy reducido tamaño, conocidos también como *mini routers*, que llegan a lograr tamaños tan ajustados como los de una tarjeta de crédito, o lo que es prácticamente lo mismo, el tamaño de una placa Arduino.

Ilustración 55: Mini *router* TP-Link WL-WR702N

Utilizar un *router* de estas características, permitiría prescindir de la Arduino WiFi Shield, utilizando en su lugar la Arduino Ethernet Shield. Consiguiendo así, reducir el precio del conjunto y añadir mayor robustez del lado del *software* (hay que tener en cuenta que la tarjeta Arduino Ethernet Shield se encuentra actualmente más distribuida entre la

comunidad Arduino, y su librería “Ethernet.h” ha recibido mayores mejoras de estabilidad y funcionalidad que su homóloga “WiFi.h”).

En el caso de incorporar un *mini router* y hacer uso de la tarjeta Raspberry Pi, se podría prescindir de complementos accesorios, ya que la tarjeta Raspberry Pi incorpora de serie un tarjeta de red Ethernet y junto con un *router* tendríamos todo lo necesario para un proyecto como el nuestro.

Estas son solo algunas posibles modificaciones y líneas de investigación sobre las que nos gustaría poder trabajar en el futuro para mejorar en el aspecto *hardware*.

En el camino del *software* también hay mucho por recorrer. Nos gustaría trabajar más en el aspecto de la seguridad. Actualmente el servidor web que ejecuta Webduino no ofrece las garantías de seguridad necesarias. Aunque hemos trabajado en el aspecto de la seguridad en la capa del Servicio Web Clever Socket –incluyendo el uso de autenticación HMAC– el Servicio Web Arduino por contra, ofrece únicamente un sistema de autenticación de acceso básico (HTTP *Basic Access Authentication*).

El Servicio Web Clever Socket también es susceptible de mejora. Nos gustaría mejorar en el aspecto del manejo de errores y excepciones. Incrementar el número de funcionalidades, así como ofrecer mayores opciones en cada una de ellas.

En el apartado de la aplicación web, queda por incorporar un sistema de gestión de usuarios. Mejorar aspectos en el apartado de la programación de tareas y la gestión de los registros.

Quedan por explorar nuevas aplicaciones y usos en conjunción con el servicio RESTful de Clever Socket. La posibilidad de abrir el dispositivo a aplicaciones de terceros, hace el desarrollo de aplicaciones futuras ilimitado.

Nos resulta de especial interés el nexo entre nuestro dispositivo y los dispositivos móviles de última generación. Pudiendo aprovechar, por ejemplo, el uso de sus sensores para disparar determinados eventos.

Creemos que sería posible explotar el factor de la movilidad, gracias al hecho de que la mayoría de nosotros llevamos a todas partes uno de estos dispositivos en el bolsillo. Este factor, junto con el acceso a internet, posibilitan multitud de posibles usos que ahora mismo se nos escapan.

Para concluir y como última idea, mencionar la posibilidad de desarrollar una aplicación para la carga de baterías de nuestros dispositivos electrónicos, como móviles, tabletas, ordenadores portátiles, etc. Una aplicación nativa que a través de la medida de su propio estado de carga,

pudiera comunicarse con el Servicio Web Clever Socket para una vez alcanzado el nivel máximo de carga, desconectarse del suministro eléctrico.

17 Conclusiones

La tecnología está haciendo más sencilla nuestras vidas día a día. Acciones que en el pasado eran inevitablemente manuales, hoy pueden ser ejecutadas de forma automática por dispositivos provenientes del ingenio humano. Nosotros, con este trabajo, hemos intentado poner nuestro granito de arena para continuar abonando un terreno fértil, que en el futuro traerá consigo nuevos y sorprendentes avances en el campo de la automática.

Nuestra andadura comenzó pensando en posibles formas y maneras de aportar valor. Para nosotros, era esencial encontrar una parcela sobre la que fuera justificado realizar una propuesta de trabajo.

Después de analizar diferentes opciones, llegamos a la idea de poder aunar nuestros conocimientos en el desarrollo de *software* con nuestra pasión por los dispositivos electrónicos de última generación. De ahí nació la propuesta de este trabajo que, con esfuerzo e ilusión, se ha convertido en algo real y de lo que podemos estar orgullosos.

Gracias a la introducción en los últimos tiempos de dispositivos electrónicos de reducido tamaño y precio, proyectos como el presente son hoy posibles. El gran respaldo que ofrece la comunidad de usuarios de Arduino y el auge del intercambio de información en internet, nos han servido de base de lanzamiento para nuestro proyecto.

No solo el desarrollo actual de la electrónica ha significado un gran apoyo. La existencia de *frameworks* y aplicaciones que facilitan de forma significativa la labor del desarrollador de *software* nos han ayudado a seguir adelante.

Cuando creímos en la utilidad de poder manejar el encendido y el apagado de nuestros aparatos domésticos a través del ordenador o del teléfono móvil, no éramos conscientes de todo el trabajo y el esfuerzo que esta tarea, aparentemente baladí, traería consigo.

Han sido muchas horas dedicadas a este trabajo, muchas vigilias antes del sueño pensando en cómo resolver determinadas situaciones. Pero sin duda, nunca antes hasta ahora, después de haber recorrido el camino, después de haber disfrutado con cada éxito y sufrido con cada adversidad, hemos sido conscientes del placer por el trabajo hecho.

Desde el planteamiento de la idea, su verificación, su diseño, su desarrollo y su implementación, han pasado muchas cosas. Todas sin duda valorables.

Creemos que con esta labor hemos madurado en lo técnico, en lo teórico y en lo personal. No podemos por más que agradecer la oportunidad, que con pretexto de este trabajo, se nos ha brindado.

Esperamos que el futuro nos siga aportando nuevas excusas que nos hagan afrontar tareas similares, que nos llenen de ilusión y nos ayuden a contribuir con nuestro esfuerzo a una sociedad mejor.

18 Fuentes de información

18.1 Consultas terminológicas

- Diccionario de la Real Academia de la Lengua Española - http://lema.rae.es/drae/
- WordReference - http://www.wordreference.com/

18.2 Apartado sobre metodología

- Metodología - http://es.wikipedia.org/wiki/Metodolog%C3%ADa
- Método - http://es.wikipedia.org/wiki/M%C3%A9todo
- Ingeniería del Software, apartado sobre la metodología - http://es.wikipedia.org/wiki/Ingenier%C3%ADa_de_software#Metodolog.C3.ADa
- Principios del Manifiesto Ágil - http://agilemanifesto.org/iso/es/principles.html
- *Object Oriented Design* - http://butunclebob.com/ArticleS.UncleBob.PrinciplesOfOod

18.3 Apartado sobre análisis

- *Best available PHP RESTful micro frameworks* - http://www.gajotres.net/best-available-php-restful-micro-frameworks/
- Actor ~ UML - http://es.wikipedia.org/wiki/Actor_(UML)

18.4 Recursos

18.4.1 Materiales

- Relé o relevador - http://es.wikipedia.org/wiki/Rel%C3%A9

18.4.2 Software

- Características de NetBeans 8.0 - https://netbeans.org/features/index.html

- Configuración NetBeans para programar para Arduino - https://code.google.com/p/arduino-netbeans/
- Entorno de desarrollo Arduino - http://www.arduino.cc

18.4.3 Implementación

- Dirección MAC - http://es.wikipedia.org/wiki/Direcci%C3%B3n_MAC
- *Virtual Hosting* - http://httpd.apache.org/docs/2.0/es/vhosts/

18.5 Normativa y legislación

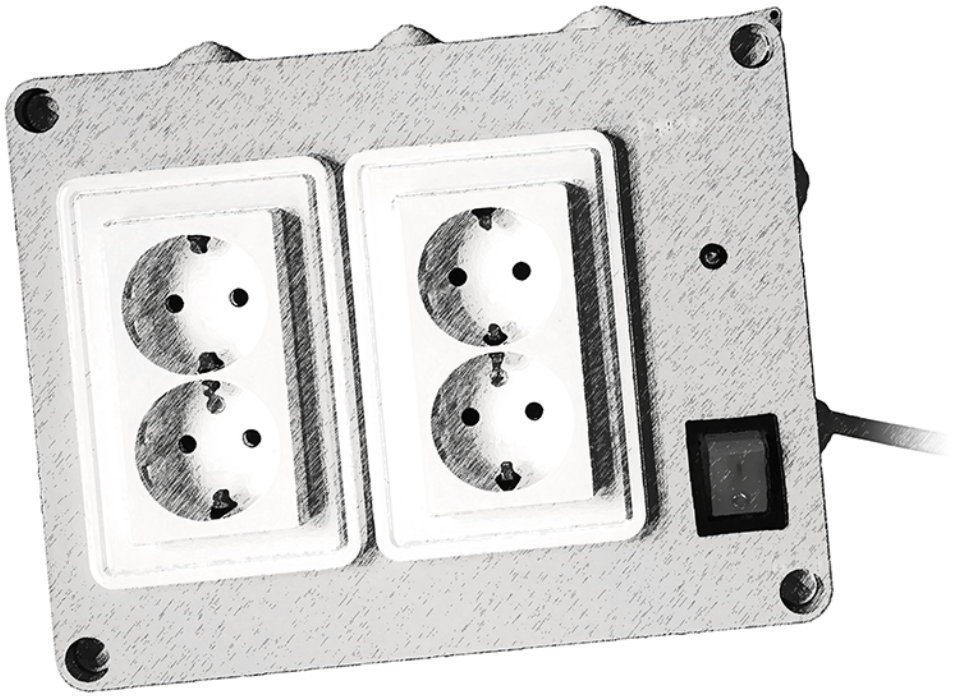
- Guía sobre el uso de las *cookies* - http://www.agpd.es/portalwebAGPD/canaldocumentacion/publicaciones/common/Guias/Guia_Cookies.pdf
- Ley 34/2002, de 11 de julio, de servicios de la sociedad de la información y de comercio electrónico - https://www.boe.es/buscar/pdf/2002/BOE-A-2002-13758-consolidado.pdf
- Licencia de *software* - http://es.wikipedia.org/wiki/Licencia_de_software
- Licencia del MIT - http://es.wikipedia.org/wiki/Licencia_MIT
- MIT *license* - http://opensource.org/licenses/mit-license.php

18.6 Proyección futura del trabajo

- *Mini router* TP-Link WL-WR702N - http://www.tplink.com/us/products/details/?model=TL-WR702N

19 Anexos

19.1 Anexo I: Manual de usuario Clever Socket



19.1.1 Introducción

El siguiente manual pretende servir de guía para el usuario novel, acercándose de forma clara y sencilla a las principales propiedades y funcionalidades del sistema.

El manual comienza con una breve introducción hacia las principales características del dispositivo. Ofreciendo indicaciones sobre su correcta puesta en funcionamiento e instrucciones de utilización. Posteriormente, continuará con el apartado referido a la utilización de la aplicación web, que como interfaz, permitirá al usuario asumir el control del sistema *hardware*.

19.1.2 El dispositivo

El dispositivo Clever Socket constituye la parte física del sistema. A él se conectan los distintos aparatos cuyo control de encendido y apagado quiera llevarse a cabo.

En su panel frontal, se pueden distinguir cuatro bases de enchufe, un interruptor de encendido y apagado y un piloto LED de estado. En la siguiente imagen podemos ver en detalle la disposición de sus partes:

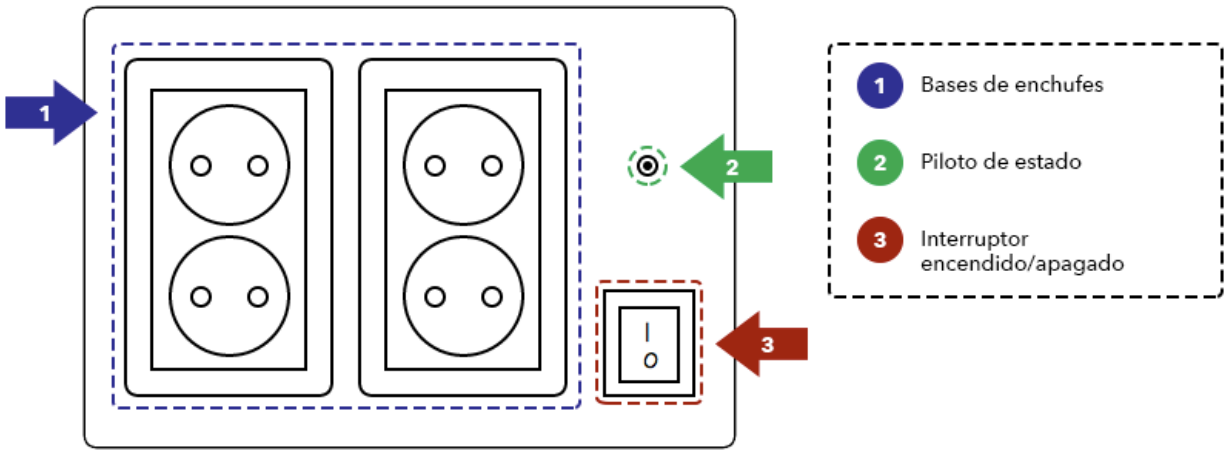


Ilustración 56: Partes del dispositivo

19.1.3 Bases de enchufes

Se cuenta con cuatro bases de enchufes. Cada base está preparada para soportar una intensidad de corriente de hasta 16 A. Según el marco europeo de normalización, estas bases entran dentro del tipo F, y son también conocidas como *shuckos* (origen, Alemania). Están preparadas para clavijas gruesas de hasta 4,8 mm de diámetro y poseen toma de tierra lateral por contacto y superior por recepción.

Se encuentran numeradas de arriba a abajo y de izquierda a derecha. Siendo la base situada en la esquina superior izquierda la número 1 y la base situada en la esquina inferior derecha la número 4.

El dispositivo soporta la utilización de todas las bases de manera simultánea sin por ello verse afectado su funcionamiento.

Piloto de estado

Existe una piloto luminoso de estado en la parte derecha del panel frontal, sobre el interruptor de encendido y apagado. Este diodo LED indica, al momento de iluminarse, que el dispositivo Clever Socket ha conseguido conectarse de forma satisfactoria a la red WiFi, y que por tanto, está listo para ser utilizado y gestionado a través de la aplicación web.

Interruptor de encendido/apagado

Este interruptor de dos estados –encendido y apagado–, es un interruptor de tipo bipolar, que permite la conexión y desconexión completa de todo el circuito interno del dispositivo Clever Socket. De tal modo que, al estar apagado, no permite la circulación de corriente a ninguno de los dispositivos conectados.

Una vez encendido, puede observarse cómo el el interruptor se ilumina con una luz de color rojo. Esta luz, de igual forma, desaparece al pasar a la fase de apagado.

No puede darse por hecho, que una vez encendido el interruptor, el dispositivo se encuentra completamente operativo; es necesario esperar unos segundos hasta que el aparato consiga conectarse a la red inalámbrica y el piloto de estado esté encendido.

19.1.4 La aplicación web

A través de la aplicación web, el usuario podrá gestionar las distintas fases de encendido y apagado de las bases de enchufe del dispositivo Clever Socket.

Vamos a continuación a dar un repaso por los diferentes elementos que componen la interfaz de usuario.

Página principal

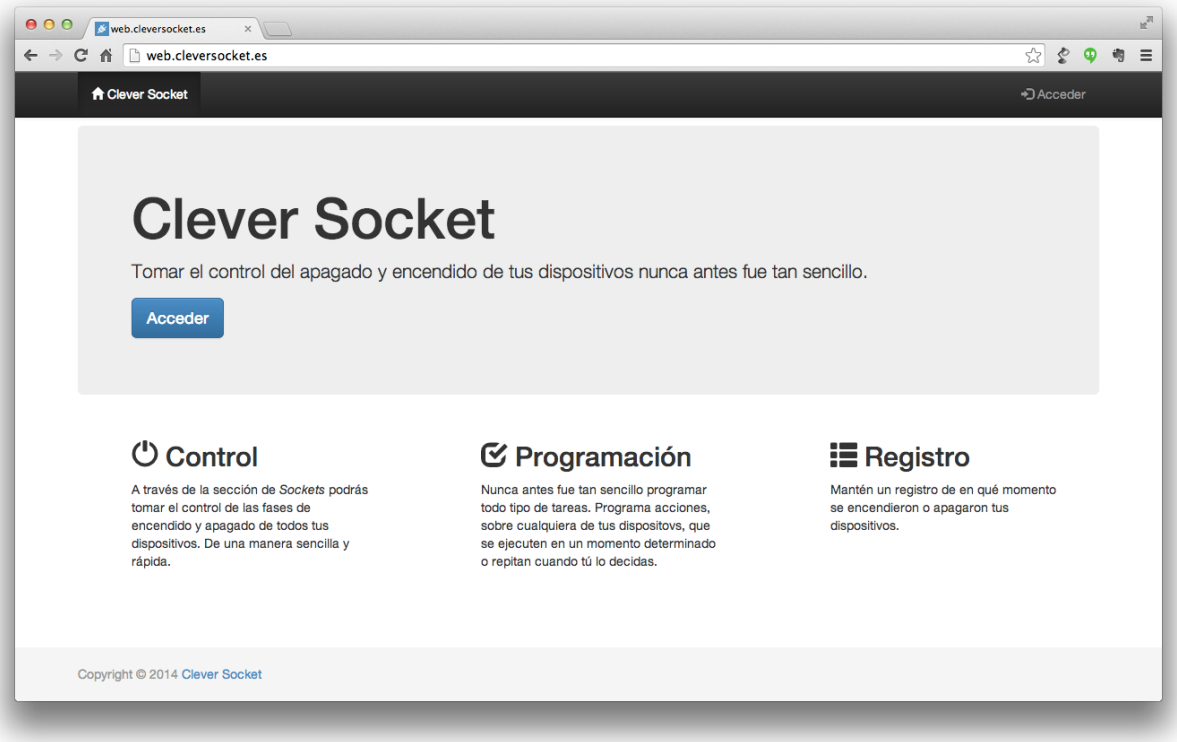


Ilustración 57: Página principal de la Aplicación Clever Socket

La página principal, es una página de bienvenida. Donde el usuario llegará nada más acceder a la aplicación. Desde aquí se resumen algunas de las características y funcionalidades principales del servicio.

Además, el usuario tiene la opción de acceder al panel de control de Clever Socket a través del botón "Acceder" o haciendo clic en el enlace de la barra de navegación del mismo nombre.

Página de los *Sockets*

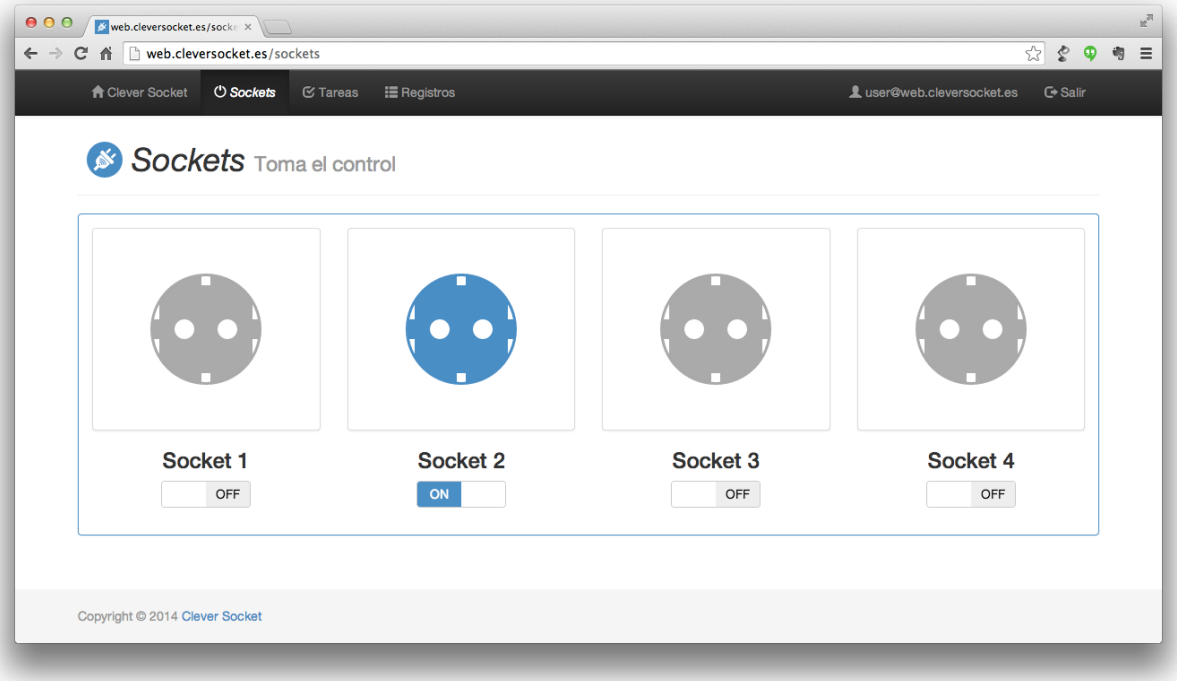


Ilustración 58: Página de *Sockets* de la Aplicación Clever Socket

En esta página, el usuario podrá ver todos los *sockets* disponibles en el sistema. Podrá operar sobre el encendido y el apagado de cada uno de ellos. Además de conocer en todo momento, cuál es el estado actual de cada dispositivo conectado.

Para encender o apagar cualquiera de los dispositivos, el usuario tiene dos opciones:

1. Hacer clic sobre el icono representativo de base de enchufe.
2. Hacer clic en botón deslizante "*ON*" u "*OFF*" que se encuentra debajo de cada *socket*.

Si el usuario opta por la primera opción, la acción que ejecutará el sistema, será la de conmutar el estado de encendido o apagado. Es decir, si el interruptor se encuentra apagado, se encenderá y si se encuentra encendido se apagará.

Las opciones "*ON*" y "*OFF*" representan los estados de encendido y apagado respectivamente.

Página de tareas

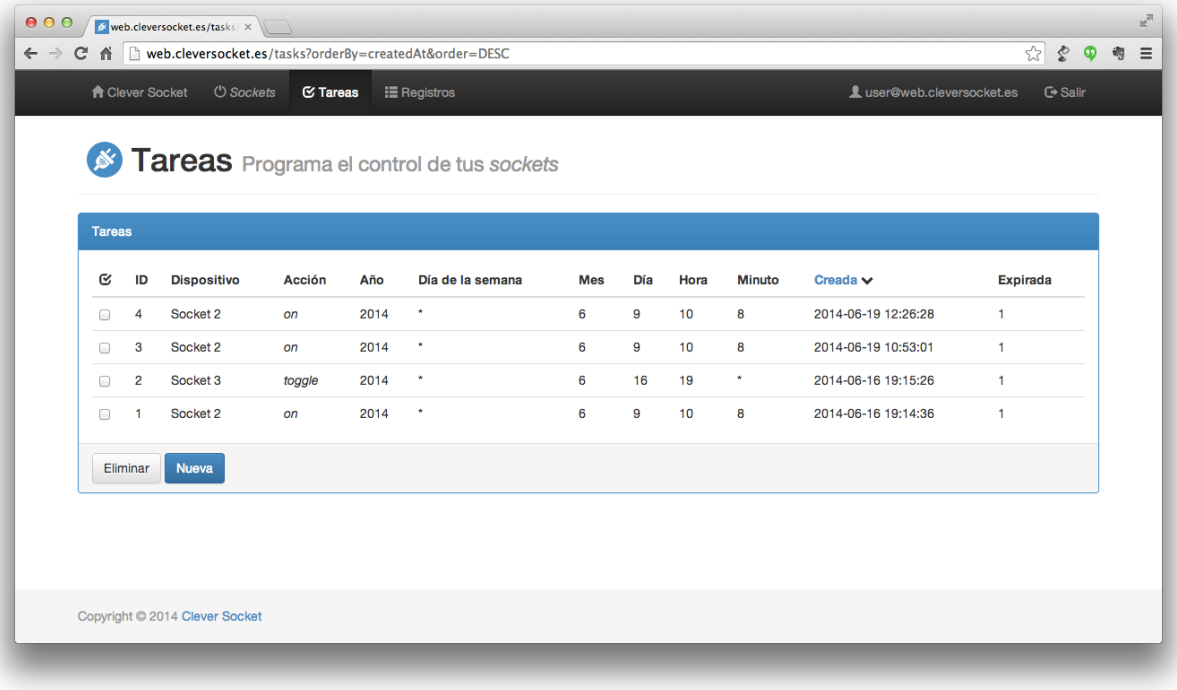


Ilustración 59: Página de Tareas de las Aplicación Clever Socket

En esta sección de la aplicación, el usuario podrá por un lado, ver las tareas que se han programado y por otro programar nuevas.

Se muestra una tabla con todas las tareas. Aparecen ordenadas por fecha de creación. No obstante, este orden puede cambiarse haciendo clic sobre el nombre de la columna “Creada”. De esta forma, el orden pasa a ser el contrario, la primera tarea creada pasa a la primera posición de la tabla.

Podemos fijarnos también en la columna “Expirada”, que indicará precisamente eso, si la tarea en cuestión se encuentra pendiente de ejecución o ya ha expirado.

Luego cabe mencionar otras columnas, como “ID” o “Dispositivo” que hacen referencia al dispositivo sobre el que se ha programado la tarea; “Acción” que muestra la acción que se programó ejecutar; y otras columnas como “Año”, “Día de la semana”, “Mes”, “Día”, “Hora” o “Minuto” que especifican el momento de ejecución.

Para las columnas que especifican el momento de ejecución, no solamente es posible observar una fecha u hora determinadas, también es posible ver el símbolo “*”. Este símbolo indica repetición; es decir que, si encontramos el símbolo en la columna “Año”, indicará que la tarea se repetirá cada año. Lo mismo ocurre para el resto de elementos de fecha u hora.

Además de la lista con las tareas, existe la posibilidad eliminar una o varias tareas haciendo clic sobre el selector de fila de cada tarea.

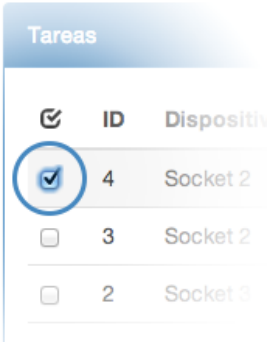


Ilustración 60: Detalle de selección de tarea

A continuación, bastará con pulsar la opción "Eliminar" y la tarea desaparecerá de la lista de tareas programadas.

Por último, existe la opción de crear una nueva tarea. Para ello pulsaremos sobre el botón "Nueva".

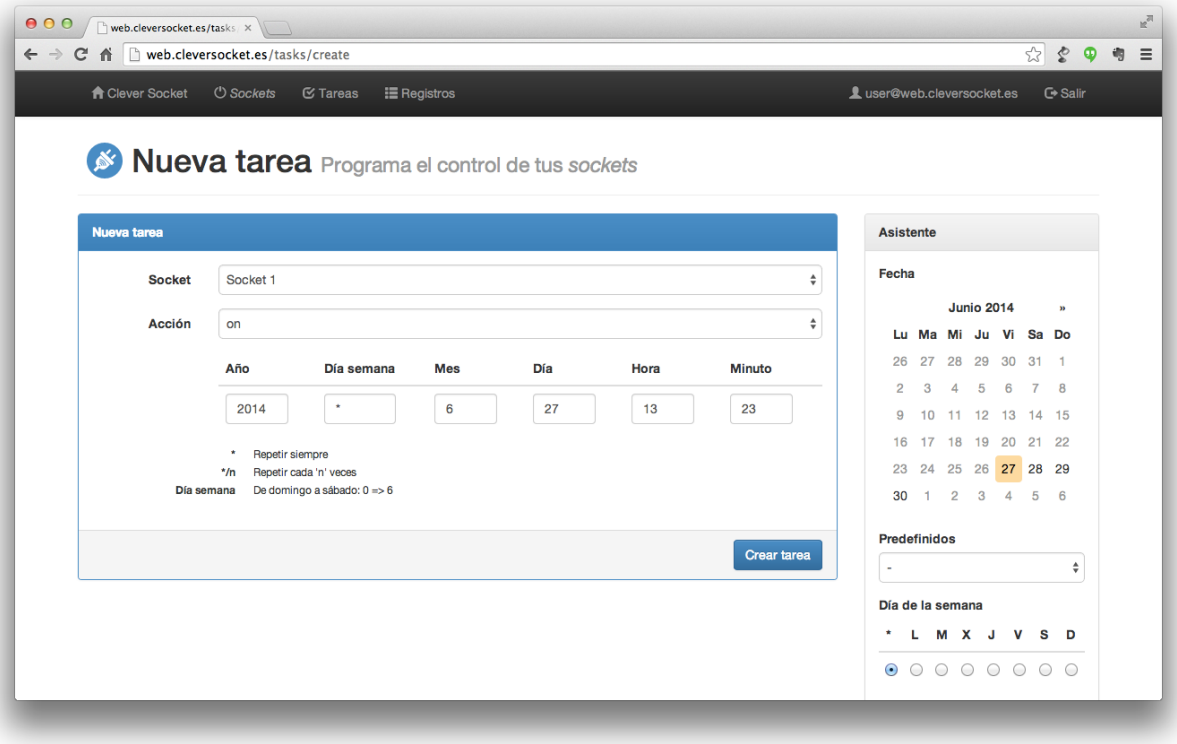


Ilustración 61: Detalle sobre la creación de una nueva tarea

En esta página nos vamos a encontrar con dos paneles. El panel principal muestra un formulario que deberemos rellenar para crear una nueva tarea. En el panel secundario, con nombre “Asistente”, encontraremos una ayuda para rellenar los campos del panel principal.

Podemos rellenar el formulario sin la ayuda del asistente, tan solo necesitamos concretar cada uno de los campos solicitados. Pero si queremos hacerlo, podemos valernos de los atajos propuestos por el panel de asistencia.

Los campos solicitados en el formulario son los siguientes:

- *Socket*: Aquí debemos indicar el dispositivo sobre el que deseamos programar una nueva tarea.
- Acción: Debemos elegir una de entre las tres opciones disponibles:
    - On: Encender
    - Off: Apagar
    - Toggle: Conmutar
- Año: Indicaremos el año en que deseamos programar la acción.
- Día de la semana: Indicaremos el día de la semana en que deseamos programar la acción. Los días deben ser indicados introduciendo un número entre el 0 y el 6, donde 0 representa al domingo y el 6 al sábado.
- Mes: Mes en que deseamos programar la acción. Se deberá introducir un número entre el 1 y el 12, donde el 1 representa a enero y el 12 a diciembre.
- Día: Día del mes en que deseamos programar la acción.
- Hora: Hora en que deseamos programar la acción.
- Minuto: Minuto en que deseamos programar la acción.

Además, es posible introducir el símbolo “*” en los campos referentes a la fecha y la hora. Indicando repetición. O también la fórmula “*/n”, indicando repetición cada “n” veces. (Por ejemplo: Minuto: */15, se repetirá cada 15 minutos.)

Página de registros

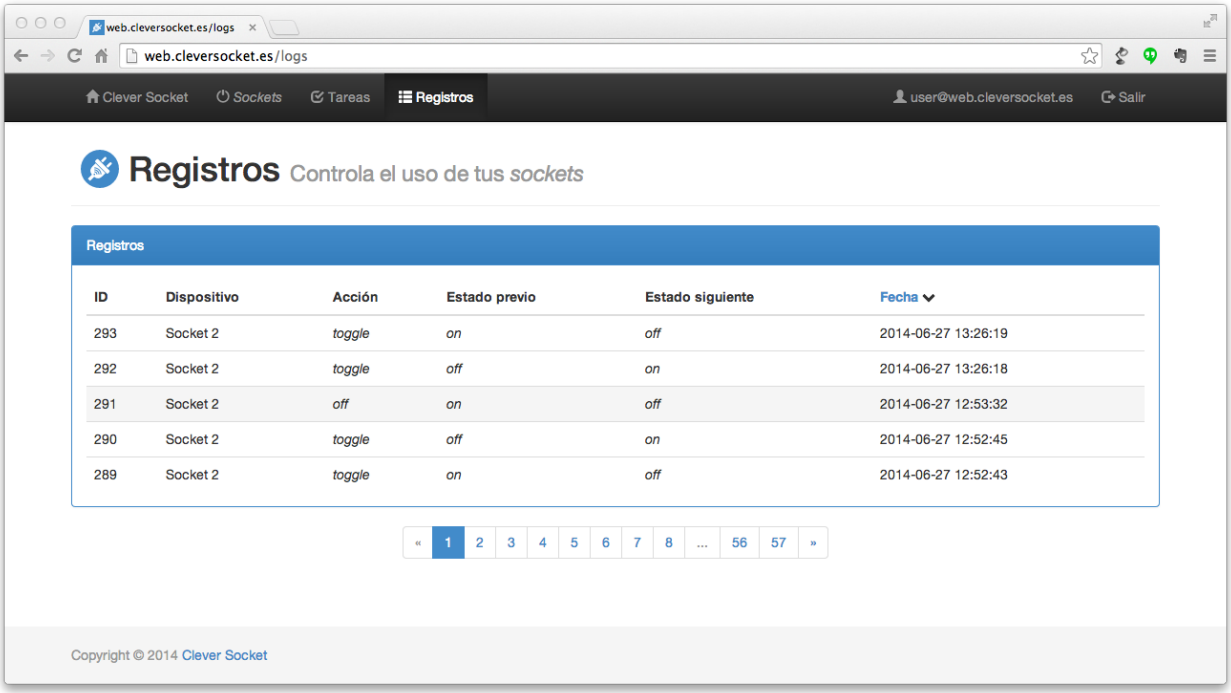


Ilustración 62: Página de Registros de la Aplicación Clever Socket

La página de registros informa sobre los eventos de encendido, apagado o conmutado sucedidos en el pasado. La lista de registros se muestra tabulada, y ordenada de más reciente a menos reciente. Orden, este último, que puede ser cambiado haciendo clic sobre la columna con nombre "Fecha". Mostrándose entonces en primer lugar, el primero de los eventos sucedido.

Las columnas que se muestran, son las siguientes: "ID" y "Dispositivo" que identifican al dispositivo en relación; "Acción" que indica la acción ejecutada; "Estado previo" y "Estado siguiente" que indican el cambio de estado sucedido; y "Fecha", indicando la fecha y la hora en que se produjo el evento.

19.2 Anexo II: Configuración de NetBeans para desarrollo de código para Arduino

19.2.1 Requisitos

Conjunto de herramientas de compilado

Conjunto de herramientas de Arduino para compilar el código (Cross Compiler Toolchain).

En Mac OS, estas herramientas se encuentran disponibles en el directorio:

```
/Applications/Arduino.app/Contents/Resources/Java/hardware/tools/avr/bin
```

Librería base de Arduino y otras

Arduino ofrece una buena librería base, así como librerías adicionales y ejemplos de código.

Estas librerías en Mac OS se encuentran disponibles en las siguientes rutas:

```
/Applications/Arduino.app/Contents/Resources/Java/hardware/arduino
```

```
/Applications/Arduino.app/Contents/Resources/Java/libraries
```

```
/usr/local/arduino1.0/variants/standard/pins_arduino.h
```

19.2.2 Configuración de NetBeans

Configurando un nuevo conjunto de herramientas

NetBeans necesita saber qué conjunto de herramientas de compilación debe usar para cada proyecto y dónde encontrarlas:

Netbeans, Preferences, C/C++: Add Tool Collection

Seleccionaremos 'GNU' como *Tool Collection Family* e introduciremos las siguientes rutas:

Base Directory:	/Applications/Arduino.app/Contents/Resources/Java/hardware/tools/avr/bin
C Compiler:	/Applications/Arduino.app/Contents/Resources/Java/hardware/tools/avr/bin/avr-gcc
C++ Compiler:	/Applications/Arduino.app/Contents/Resources/Java/hardware/tools/avr/bin/avr-g++
Fortran Compiler:	/Applications/Arduino.app/Contents/Resources/Java/hardware/tools/avr/bin/make
Assembler:	/Applications/Arduino.app/Contents/Resources/Java/hardware/tools/avr/bin/avr-as
Make Command:	/Applications/Arduino.app/Contents/Resources/Java/hardware/tools/avr/bin/make
Debugger Command:	/Applications/Arduino.app/Contents/Resources/Java/hardware/tools/avr/bin/avr-gdb

Tabla 13: Rutas de configuración *Tool Collection Family*

Hecho esto, tendremos que ir a "*Code Assistance*", seleccionaremos la *Tool Collection* creada anteriormente, y tanto para C como para C++, introduciremos las siguientes rutas:

Ilustración 63: Rutas del *Code Assistance*

Arduino Plugin

Descargar el *plugin* de la siguiente dirección:

http://plugins.netbeans.org/plugin/46054/arduino

En NetBeans, ir a *Tools* > *Plugins* > *Downloaded* > Clic sobre "*Add Plugins...*" y seleccionar el archivo .nbm descargado.

Finalmente seleccionar el *plugin* y hacer clic en *Install*.

Makefile

Cada vez que creemos un nuevo proyecto Arduino, tendremos que abrir el fichero *Makefile* y editar algunos campos como *COM_PORT*, *ARDUINO_BASE_DIR* o *INCLUDE_LIBS*, de forma que se ajusten a nuestra configuración.

```
// ...
COM_PORT = /dev/tty.usbmodem411
// ...
ARDUINO_BASE_DIR = /Applications/Arduino.app/Contents/Resources/Java
// ...
INCLUDE_LIBS=EEPROM;SPI;WiFi;WiFi/utility;Streaming;Sha;
// ...
```

Tabla 14: Configuración del Makefile

Ahora, ya podremos compilar y subir nuestros programas a la memoria interna de nuestra placa Arduino.

Uso de la consola para leer el puerto serie

Si queremos además poder leer por consola el puerto serie, tendremos que incluir la siguiente librería:

```
#include <HardwareSerial.h>
```

Así como inicializar el puerto serie:

```
Serial.begin(9600);
```

De modo que podremos hacer:

```
Serial.println(“Hello World!);
```

Para leer el puerto serie desde la consola, podemos utilizar el siguiente comando (suponiendo que nuestro dispositivo está conectado al puerto /dev/tty.usbmodem411:

```
~ screen /dev/tty.usbmodem411
```

Para salir de la aplicación Screen, pulsaremos la combinación de teclas “Cmd+A”, y posteriormente pulsaremos la tecla “k”, nos preguntarán si estamos seguros de que queremos salir, pulsamos “y” y habremos salido de la aplicación.

19.3 Anexo III: Actualización de *firmware* Arduino WiFi Shield

Una de las primeras cosas que tuvimos que hacer nada más adquirir el componente de expansión Arduino WiFi Shield, fue actualizar su *firmware*. Por algún motivo, la instalación del *software* que trae de fábrica viene con errores conocidos y reportados por la comunidad.

Desde la propia compañía Arduino, recomiendan de manera imperiosa realizar esta actualización.

A continuación explicamos los pasos que tuvimos que dar para realizar la actualización del *firmware* del *shield* (la actualización se ha realizado desde un Mac OS 10.9.3):

19.3.1 Instalación de Ports

Lo primero que tenemos que hacer es comprobar si tenemos instalado en nuestro equipo el gestor de paquetes "MacPorts". Para ello podemos ejecutar el siguiente comando en el terminal:

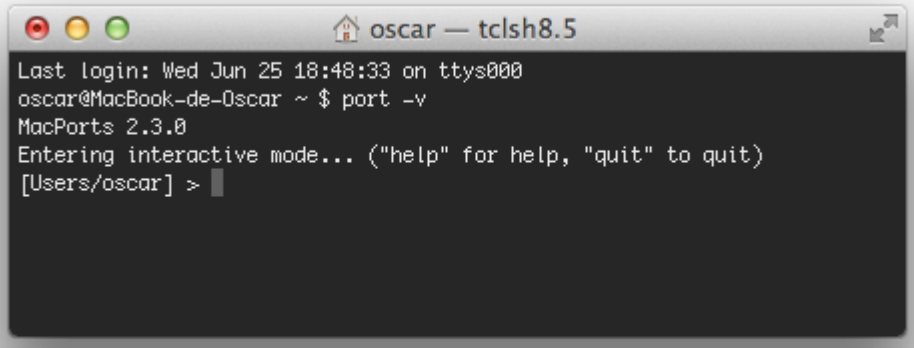


Ilustración 64: Versión de Ports

Si tras ejecutar el comando "$ `port -v`" obtienes un mensaje como el anterior, puedes pasar al siguiente paso, ya que tendrás instalado el gestor. En caso contrario, explicamos a continuación cómo instalarlo:

Acudimos a la página http://www.macports.org/install.php. Aquí se nos dice que debemos tener instalado Xcode y el conjunto de herramientas desde línea de comandos Xcode Command Line Tools.

La instalación de Xcode no reviste complicación ya que puede ser realizada directamente desde la App Store. Sin embargo, para instalar Xcode Command Line, tendremos que hacer lo siguiente:

1. Introducir el siguiente comando en el terminal: $ `xcode-select–install` y continuar con la instalación.

2. O acceder a la dirección (necesitaremos tener una cuenta de desarrollador) https://developer.apple.com/downloads/index.action y descargar el paquete de instalación.

Una vez instalados estos dos componentes, debemos aceptar la licencia de Xcode, para ello, tan solo debemos introducir el siguiente comando y seguir las instrucciones:

```
$ sudo xcodebuild -license
```

Ahora ya estamos preparados para descargarnos y ejecutar el programa de instalación de MacPorts Pro. Para ello descargamos la versión de nuestro sistema operativo; en nuestro caso:

https://distfiles.macports.org/MacPorts/MacPorts-2.3.0-10.9-Mavericks.pkg

19.3.2 Instalación `dfu-programmer`

Ahora que tenemos MacPorts instalado, podemos instalar el programa que nos ayudará a actualizar el *firmware* de nuestro *shield* ejecutando el siguiente comando:

```
$ sudo port install dfu-programmer
```

19.3.3 Preparar archivos para la actualización del *firmware*

Ahora, simplemente, debemos cerciorarnos de que tenemos instalada la última versión del *software* de Arduino. Para ello, podemos abrir el IDE de Arduino, hacemos clic sobre la opción de la barra de herramientas “Arduino” y a continuación sobre “Acerca de Arduino”, veremos algo similar a esto:

Ilustración 65: Versión Arduino IDE

Si como nosotros, tienes la versión 1.0.5 o posterior, enhorabuena, tienes en tu equipo los archivos necesarios para actualización del *firmware*, si no, puedes descargarte la última versión del *software* de Arduino desde http://arduino.cc/en/Main/Software.

Ahora, para preparar los archivos necesarios para la actualización, accede a la ruta /Applications/Arduino.app/Contents/Resources/Java/hardware/arduino/firmwares desde el terminal y ejecuta los siguientes:

```
$ mkdir wifi-shield
```

```
$ cp wifishield/binary/* wifi-shield/
```

19.3.4 Conexión con el ordenador y actualización

Ya tenemos todo preparado en nuestro equipo para realizar la instalación. Ahora solo resta, conectar el Arduino WiFi Shield a nuestro equipo a través del puerto Mini USB. Hay que tener en cuenta dos cosas:

1. El Arduino WiFi Shield no debe estar montado sobre la placa Arduino.
2. El DFU *programming jumper* J3 debe estar conectado como se muestra en la imagen siguiente:

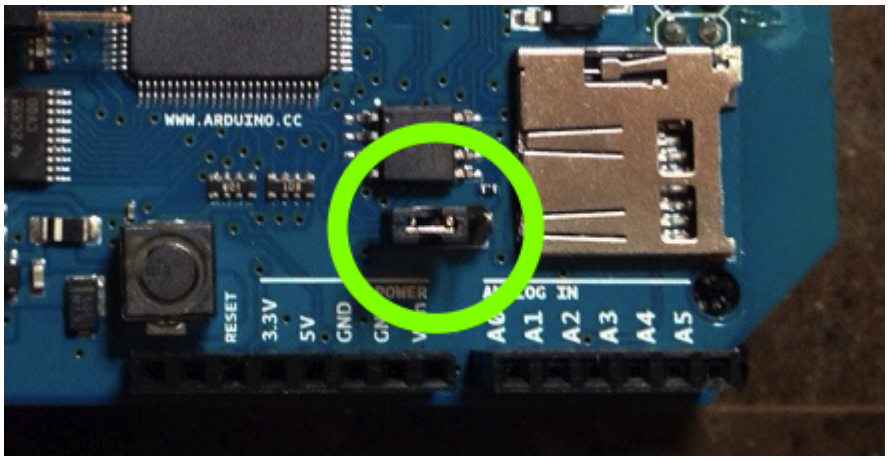


Ilustración 66: Detalle del *jumper* J3 para la programación DFU

Accedemos al directorio /Applications/Arduino.app/Contents/Resources/Java//hardware/arduino/firmwares/wifishield/scripts y ejecutamos el siguiente comando desde terminal:

```
$ sudo sh ArduinoWifiShield_upgrade_mac.sh -a
/Applications/Arduino.app/Contents/Resources/Java -f shield
```

Si todo fue correctamente, deberíamos ver un mensaje como el siguiente:

```
Done. Remove the J3 jumper and press the RESET button on the shield.

Thank you!
```

19.4 Anexo IV: Presupuesto

A continuación se detalla el presupuesto de los distintos componentes utilizados para la elaboración del prototipo Clever Socket:

Cantidad	**Componente**	**Precio**
1 x	Arduino UNO Rev3	21,19 €
1 x	Arduino WiFi Shield SD	73,08 €
1 x	Módulo 4 relés 5V	12,71 €
8 x	*Jumper* de conexión para placa *protoboard*, macho a hembra	1,62 €
1 x	Alimentador electrónico universal 9V, 2A	8,64 €
1 x	Caja estanca con conos, 220x170x85mm	9,50 €
2 x	Base enchufe doble, 16A, 250V	12,80 €
1 x	Regleta de conexión	2,10€
1 x	Clavija enchufe, patas de 4,8mm, 16A, 250V	1,30 €
1 x	Interruptor conmutador luminoso rojo de dos circuitos	2,85 €
4 x	Terminal plano hembra 6,4mm, aislado, azul 2,5mm	0,80 €
1 x	Diodo LED 5mm, 5V, azul, con cable de 18cm	0,54 €
1 x	Termorretráctil corto	0,64 €
24 x	Arandela plana, 3mm	0,22 €
24 x	Tornillo DIN965 cabeza plana	0,48 €
24 x	Tuerca 934	0,48 €
1 x	Cable manguera 3x1,5mm blanco, 4,5m	4,50 €
1 x	Tablero contrachapado 600x300x5mm	2,65 €
Total (bruto):		**156,10 €**

Tabla 15: Presupuesto de los componentes del prototipo

19.5 Anexo V: Plantillas de casos de uso

Nombre	Identificarse	**ID**	UR.1
Creado por	Óscar A.	**Fecha**	26/04/2014
Modificada por		**Fecha modificación**	

Actor principal: Usuario registrado
Personal involucrado o intereses: 1. Usuario registrado: El usuario quiere poder identificarse para poder acceder a los servicios ofrecidos para su dispositivo Clever Socket.

Descripción: El usuarios se identifica a través de la página de acceso a la aplicación web.
***Trigger*:** El usuario hace clic en el enlace "Acceder" disponible desde el menú principal.
Precondición: 1. Estar registrado en el sistema.
Postcondición: 1. El usuario se encuentra identificado y es dirigido hacia el panel de control de Clever Socket.
Flujo normal: 1. El usuario hace clic en el enlace "Acceder". 2. El usuario introduce sus credenciales. 3. El sistema da la bienvenida al usuario y lo dirige hacia la página principal del panel de control.
Flujo alternativo: 1. … 2. El usuario introduce sus credenciales. a. El correo electrónico y/o la contraseña no son correctos. i. El sistema muestra un mensaje advirtiendo del error.
Excepción: El dispositivo Clever Socket no se encuentra conectado o disponible. El sistema muestra un mensaje alertando de que el servicio no se encuentra disponible en ese momento.
***Includes*:**
Requisitos especiales:
Notas:

Tabla 16: Caso de uso "Identificarse"

Nombre	Ver *sockets*	**ID**	UR.2
Creado por	Óscar A.	**Fecha**	26/04/2014

Modificada por		**Fecha modificación**	

Actor principal: Usuario registrado
Personal involucrado o intereses: 1. Usuario registrado: El usuario quiere poder ver los *sockets* disponibles, así como obtener información sobre ellos y sus estados.
Descripción: El sistema muestra los *sockets* disponibles del usuario, así como sus nombres y si estos se encuentran encendidos o apagados.
***Trigger*:** El usuario accede a la sección de "*Sockets*" en el panel de control.
Precondición: 1. El usuario debe estar identificado.
Postcondición: 1. Se muestran los *sockets* disponibles e información relativa a ellos.
Flujo normal: 1. El usuario accede a la sección *Sockets* dentro del panel de control. 2. El sistema muestra los *sockets* disponibles e información sobre cada dispositivo y su estado.
Flujo alternativo: 1. El usuario accede a la sección *Sockets* dentro del panel de control. b. No existen *sockets* registrado del usuario. i. El sistema no muestra dispositivos y sus estados.
Excepción: Se pierde conexión con el dispositivo Clever Socket a través del servicio web. El sistema muestra un mensaje alertando de que el servicio no se encuentra disponible en ese momento y el usuario es sacado del panel de control, perdiendo la identificación.
***Includes*:** 1. Caso de uso "Identificarse".
Requisitos especiales:
Notas:

Tabla 17: Caso de uso "Ver *sockets*"

Nombre	Encender *socket*	**ID**	UR.3
Creado por	Óscar A.	**Fecha**	26/04/2014
Modificada por		**Fecha modificación**	

Actor principal: Usuario registrado
Personal involucrado o intereses: 1. Usuario registrado: El usuario quiere poder encender un *socket* que se encuentra en estado "*OFF*" (apagado).
Descripción: El usuario solicita al sistema encender un *socket* que se encuentra en el estado de "*OFF*" (apagado). El *socket* objeto de la acción es encendido.
***Trigger*:** El usuario hace clic sobre la opción "*ON"* (encendido) en la sección de "*Sockets*".
Precondición: 1. El usuario debe estar identificado. 2. El estado del *socket* debe ser "*OFF*" (apagado).
Postcondición: 1. El estado del socket pasa a ser "*ON*" (encendido).
Flujo normal: 1. El usuario hace clic sobre la opción "*ON*" (encendido) del *socket* objeto de la acción. 2. El sistema cambia el estado del *socket* de "*OFF*" a "*ON*".
Flujo alternativo:
Excepción: Se pierde conexión con el dispositivo Clever Socket a través del servicio web. El sistema muestra un mensaje alertando de que el servicio no se encuentra disponible en ese momento y el usuario es sacado del panel de control, perdiendo la identificación.
***Includes*:** 1. Caso de uso "Identificarse".

Requisitos especiales:
Notas:

Tabla 18: Caso de uso "Encender *socket*"

Nombre	Apagar *socket*	**ID**	UR.4
Creado por	Óscar A.	**Fecha**	26/04/2014
Modificada por		**Fecha modificación**	

Actor principal: Usuario registrado
Personal involucrado o intereses: 1. Usuario registrado: El usuario quiere poder apagar un *socket* que se encuentra en estado "*ON*" (encendido).
Descripción: El usuario solicita al sistema apagar un *socket* que se encuentra en el estado de "*ON" (encendido)*. El *socket* objeto de la acción es apagado.
***Trigger*:** El usuario hace clic sobre la opción "*OFF"* (apagado) en la sección de "*Sockets*".
Precondición: 1. El usuario debe estar identificado. 2. El estado del *socket* debe ser "*ON*" (encendido).
Postcondición: 1. El estado del socket pasa a ser "*OFF*" (apagado).
Flujo normal: 1. El usuario hace clic sobre la opción "*OFF*" (apagado) del *socket* objeto de la acción. 2. El sistema cambia el estado del *socket* de "*ON*" a "*OFF*".
Flujo alternativo:
Excepción: Se pierde conexión con el dispositivo Clever Socket a través del servicio web. El sistema muestra un mensaje alertando de que el servicio no se encuentra disponible en

ese momento y el usuario es sacado del panel de control, perdiendo la identificación.
***Includes*:** 1. Caso de uso “Identificarse”.
Requisitos especiales:
Notas:

Tabla 19: Caso de uso "Apagar *socket*"

Nombre	Conmutar *socket*	**ID**	UR.5
Creado por	Óscar A.	**Fecha**	26/04/2014
Modificada por		**Fecha modificación**	

Actor principal: Usuario registrado
Personal involucrado o intereses: 1. Usuario registrado: El usuario quiere poder conmutar el estado actual de un *socket*.
Descripción: El usuario solicita al sistema conmutar el estado de un *socket* que se puede encontrar en estado “*ON*” (encendido) u “*OFF*” (apagado). El *socket* objeto de la acción conmuta su estado actual.
***Trigger*:** El usuario hace clic sobre la opción de conmutado en la sección de “*Sockets*”.
Precondición: 1. El usuario debe estar identificado.
Postcondición: 1. El estado del socket conmuta; si antes era “ON” ahora es “*OFF*” y si antes era “*OFF*” ahora es “*ON*”.
Flujo normal: 1. El usuario hace clic sobre la opción de conmutado del *socket* objeto de la acción. 2. El sistema cambia el estado del *socket* de “*ON*” a “*OFF*” o *de* “*OFF*” a “*ON*” en función a estado inicial.

Flujo alternativo:
Excepción: Se pierde conexión con el dispositivo Clever Socket a través del servicio web. El sistema muestra un mensaje alertando de que el servicio no se encuentra disponible en ese momento y el usuario es sacado del panel de control, perdiendo la identificación.
***Includes*:** 1. Caso de uso “Identificarse”.
Requisitos especiales:
Notas:

Tabla 20: Caso de uso "Conmutar *socket*"

Nombre	Ver tareas	**ID**	UR.6
Creado por	Óscar A.	**Fecha**	26/04/2014
Modificada por		**Fecha modificación**	

Actor principal: Usuario registrado
Personal involucrado o intereses: 1. Usuario registrado: El usuario quiere poder ver las tareas que existen programadas en el sistema.
Descripción: El usuario accede a la sección “Tareas” y el sistema muestra las tareas programadas.
***Trigger*:** El usuario accede a la sección “Tareas”.
Precondición: 1. El usuario debe estar identificado.
Postcondición: 1. El sistema muestra una lista con las tareas programadas.

Flujo normal: 1. El usuario accede a la sección “Tareas”. 2. El sistema sistema muestra una lista paginada con las tareas programadas.
Flujo alternativo: 1. … 2. El sistema sistema muestra una lista paginada con las tareas programadas. a. No se han encontrado tareas programadas. i. El sistema muestra una notificación en la que dice que no existen tareas programadas.
Excepción:
***Includes*:** 1. Caso de uso “Identificarse”.
Requisitos especiales:
Notas:

Tabla 21: Caso de uso "Ver tarea"

Nombre	Crear tarea	**ID**	UR.7
Creado por	Óscar A.	**Fecha**	26/04/2014
Modificada por		**Fecha modificación**	

Actor principal: Usuario registrado
Personal involucrado o intereses: 1. Usuario registrado: El usuario quiere poder crear una tarea.
Descripción: El usuario elige crear una nueva tarea, programando una acción de entre las disponibles para ejecutar sobre un *socket*, y concretando el momento de disparo.
***Trigger*:** El usuario hace clic sobre la opción “Nueva tarea”.

Precondición: 1. El usuario debe estar identificado. 2. Deben existir *sockets* disponibles para el usuario.
Postcondición: 1. Una nueva tarea es introducida en el sistema. 2. Se muestra la nueva tarea en la primera posición de la lista de tareas.
Flujo normal: 1. El usuario hace clic sobre la opción “Nueva tarea”. 2. El usuario selecciona el *socket* sobre el que desea crear la tarea. 3. El usuario selecciona una acción de entre las disponibles para ejecutar sobre el *socket*. 4. El usuario programa el momento de disparo de la acción. 5. El usuario confirma la creación de la tarea. 6. El sistema muestra la tarea creada en la primera posición de la lista de tareas.
Flujo alternativo:
Excepción:
***Includes*:** 1. Caso de uso “Identificarse”.
Requisitos especiales:
Notas:

Tabla 22: Caso de uso "Crear tarea"

Nombre	Eliminar tarea	**ID**	UR.8
Creado por	Óscar A.	**Fecha**	26/04/2014
Modificada por		**Fecha modificación**	

Actor principal: Usuario registrado
Personal involucrado o intereses:

1. Usuario registrado: El usuario quiere poder eliminar una o varias tareas de entre las existentes.
Descripción: El usuario elige una o varias tareas de entre la lista de tareas existentes para proceder a su eliminación.
***Trigger*:** Clic sobre la opción “Eliminar”.
Precondición: 1. El usuario debe estar identificado. 2. Debe estar seleccionada al menos una tarea de las existentes en la lista de tareas.
Postcondición: 1. Las tareas seleccionadas pasan a estar eliminadas. 2. Las tareas eliminadas no aparecen entre la lista de tareas existentes.
Flujo normal: 1. El usuario selecciona aquella o aquellas tareas que desea eliminar. 2. El usuario hace clic sobre la opción “Eliminar”. 3. El sistema muestra la lista de tareas existentes, no incluyendo las tareas eliminadas.
Flujo alternativo:
Excepción:
***Includes*:** 1. Caso de uso “Identificarse”.
Requisitos especiales:
Notas:

Tabla 23: Caso de uso "Eliminar tarea"

Nombre	Ver registros	**ID**	UR.9
Creado por	Óscar A.	**Fecha**	26/04/2014

Modificada por		**Fecha modificación**	

Actor principal: Usuario registrado

Personal involucrado o intereses:

2. Usuario registrado: El usuario quiere poder ver el registro de acciones ejecutadas en el pasado sobre sus *sockets*.

Descripción:

El usuario solicita ver las acciones ejecutadas sobre sus *sockets* y el sistema muestra la acción ejecutada, el estado previo y siguiente, así como el momento en que la acción fue ejecutada.

***Trigger*:**

Acceso a la sección "Registros".

Precondición:

1. El usuario debe estar identificado.

Postcondición:

1. El sistema muestra los registros existentes.

Flujo normal:

1. El usuario accede a la sección "Registros".
2. El sistema muestra un lista paginada de las acciones registradas sobre los *sockets* del usuario.

Flujo alternativo:

1. …
2. El sistema muestra un lista paginada de las acciones registradas sobre los *sockets* del usuario.
    a. No existen registros de los *sockets* del usuario.
        i. El sistema muestra un mensaje avisando de que no existen registros sobre los *sockets* hasta la fecha.

Excepción:

***Includes*:**

1. Caso de uso "Identificarse".

Requisitos especiales:

Notas:

Tabla 24: Caso de uso "Ver registros"

19.6 Anexo VI: Diagramas y operaciones del Servicio Web Arduino

19.6.1 Diagrama de casos de uso

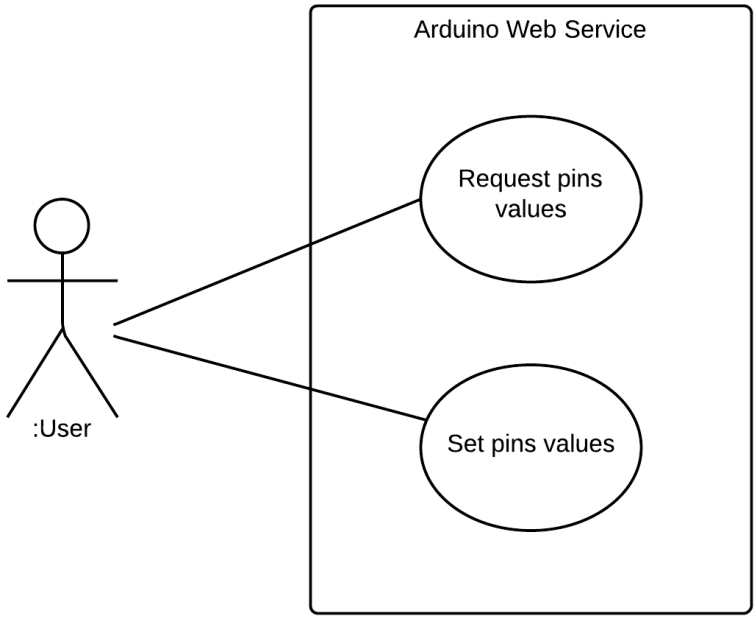


Diagrama 7: Casos de uso del Servicio Web Arduino

19.6.2 Diagramas de actividad

Lectura de los pines

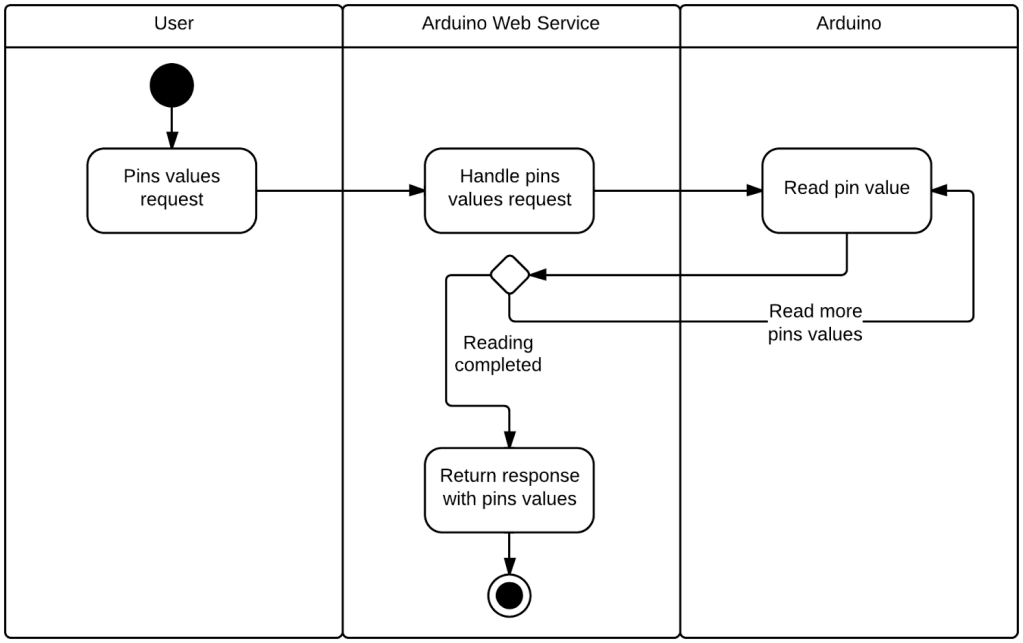


Diagrama 8: Diagrama de actividad de la lectura de pines, Servicio Web Arduino

Escritura de los pines

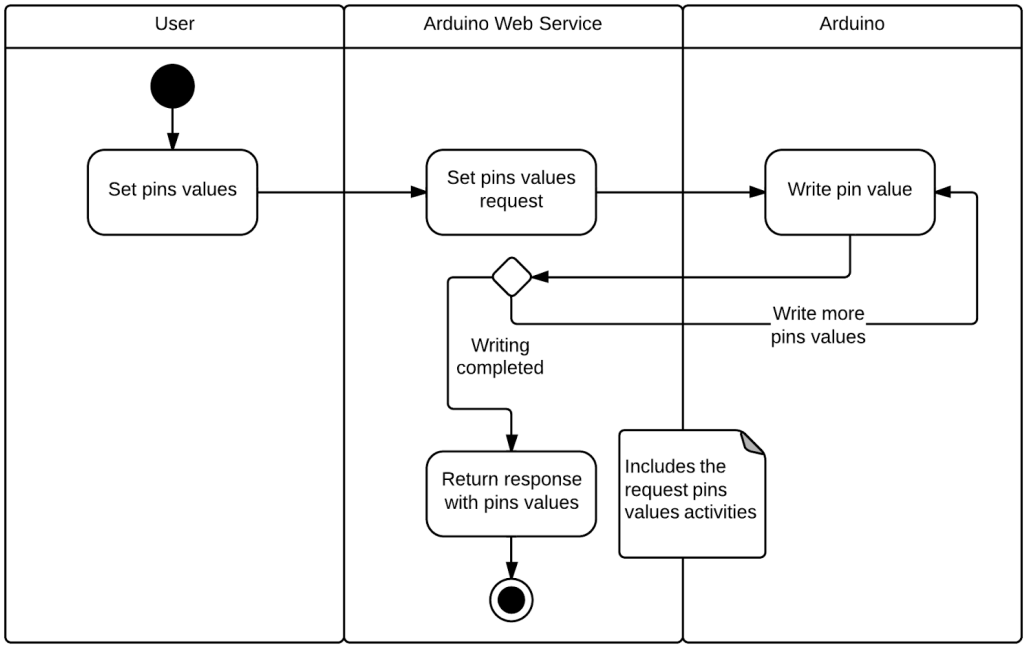


Diagrama 9: Diagrama de actividad de la escritura de pines, Servicio Web Arduino

19.6.3 Diagramas de secuencia

Lectura de los valores de los pines

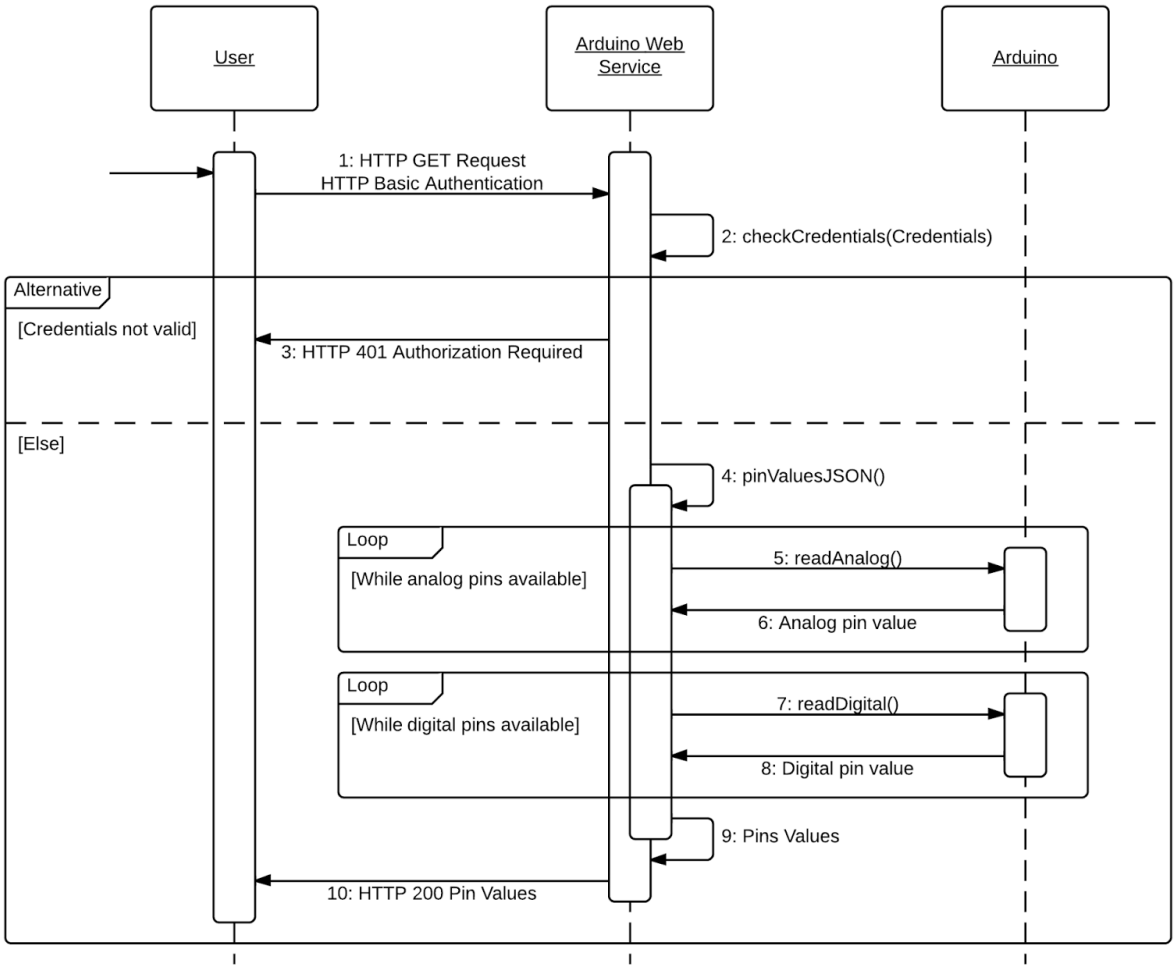


Diagrama 10: Diagrama de secuencia de la lectura de los valores de los pines, Servicio Web Arduino

Escritura de los valores de los pines

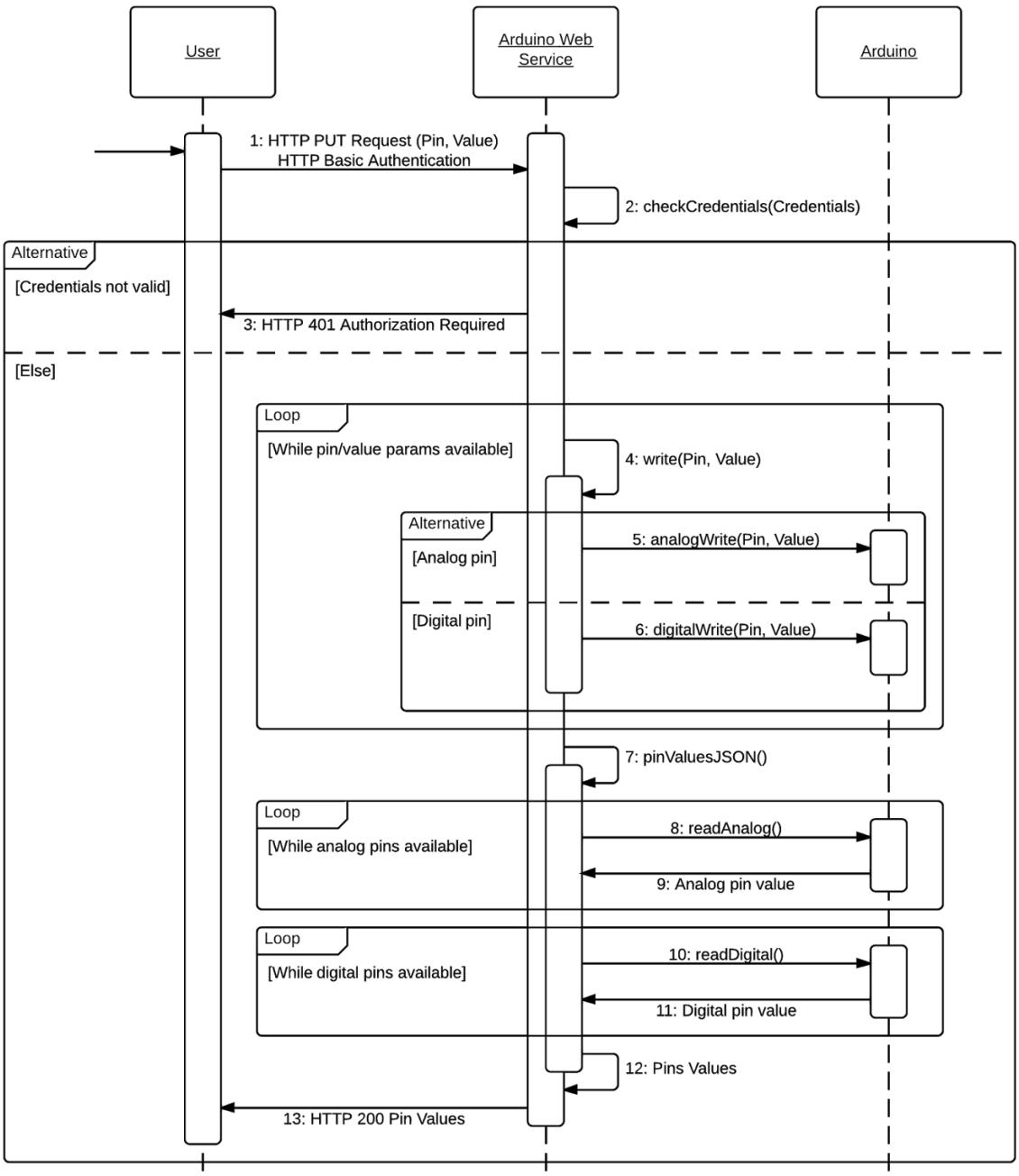


Diagrama 11: Diagrama de secuencia de la escritura de los valores de los pines, Servicio Web Arduino

19.6.4 Operaciones

/pins

Método	**GET**

Verbo	`/pins`
Descripción	Devuelve una lista de los valores de los pines analógicos y digitales.

Ejemplo	`GET http://arduino.cleversocket.es/pins`
	
	`// Request`
	
	`// Response`
	
	`200`
	
	`Content-Type: application/json`
	
	`{`
	`  a0: 260,`
	`  a1: 263,`
	`  a2: 316,`
	`  a3: 415,`
	`  a4: 465,`
	`  a5: 432,`
	`  d0: 0,`
	`  d1: 0,`
	`  d2: 0,`
	`  d3: 0,`
	`  d4: 0,`
	`  d5: 0,`
	`  d6: 0,`
	`  d7: 0,`
	`  d8: 0,`
	`  d9: 0`
	`}`

Tabla 25: Operación GET /pins

Método	**PUT**
Verbo	`/pins`
Descripción	Modifica el valor de uno o varios pines.

Opciones	• Establecer valor de pin analógico:
	  ○ `a{id}={0|1}`
	• Establecer valor de pin digital:
	  ○ `d{id}={0|1}`

Ejemplo	`PUT http://arduino.cleversocket.es/pins`

```
// Request

Content-Type: application/x-www-form-urlencoded

d2=1&d3=1&d4=0&d5=1&d6=1

// Response

200

Content-Type: application/json

{
    a0: 260,
    a1: 263,
    a2: 316,
    a3: 415,
    a4: 465,
    a5: 432,
    d0: 0,
    d1: 0,
    d2: 1,
    d3: 1,
    d4: 0,
    d5: 1,
    d6: 1,
    d7: 0,
    d8: 0,
    d9: 0
}
```

Tabla 26: Operación PUT /pins

19.7 Anexo VII: Diagramas y operaciones del Servicio Web Clever Socket

19.7.1 Diagrama de casos de uso

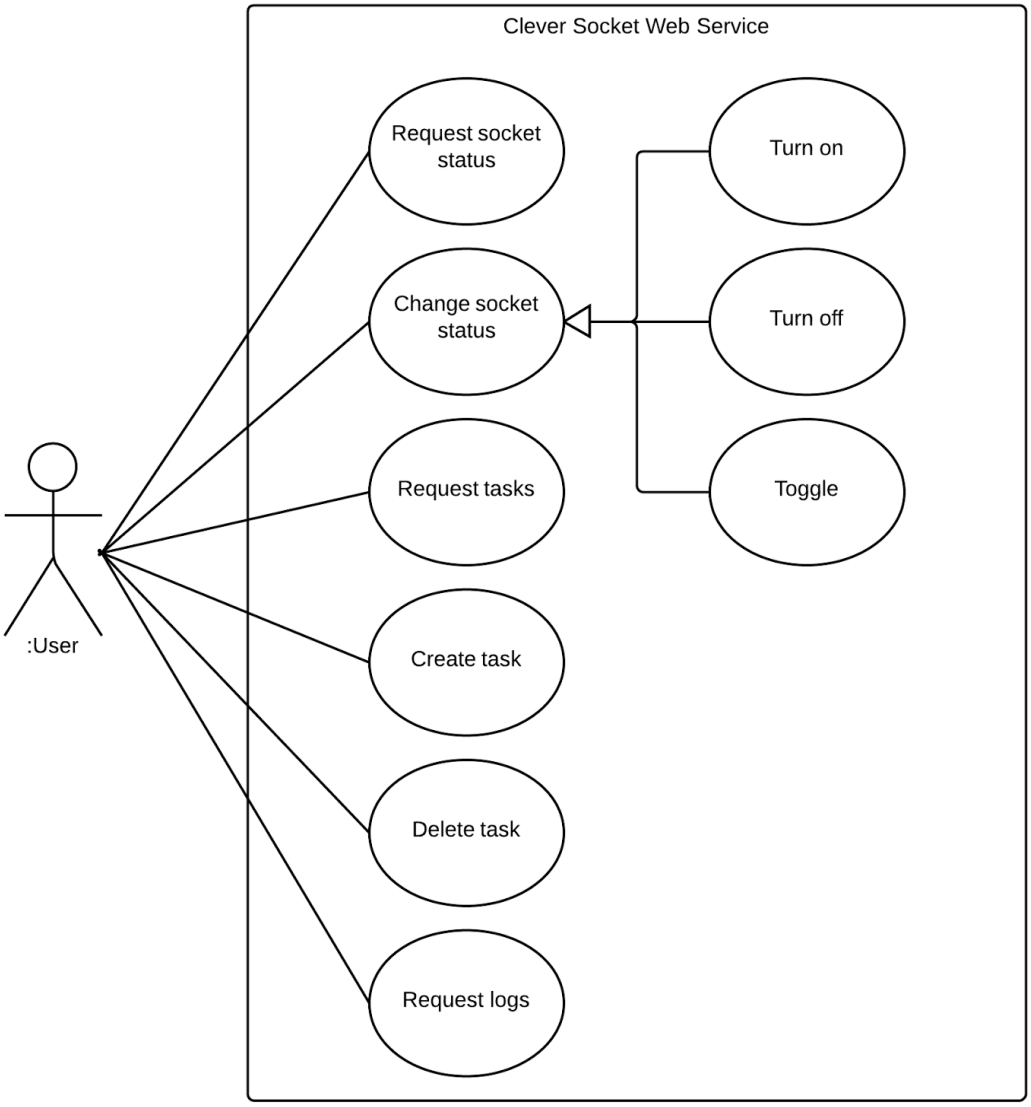


Diagrama 12: Diagrama de casos de uso del Servicio Web de Clever Socket

19.7.2 Diagramas de actividad

Lectura del estado de los *sockets*

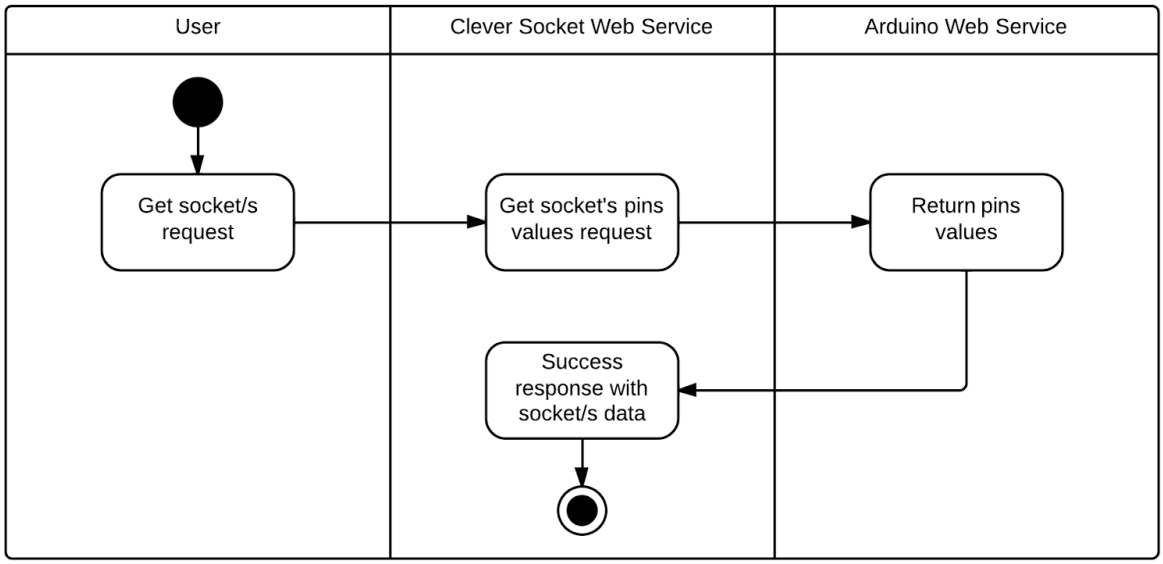


Diagrama 13: Diagrama de actividad de la lectura del estado de los *sockets*, Servicio Web Clever Socket

Encendido/apagado de un *socket*

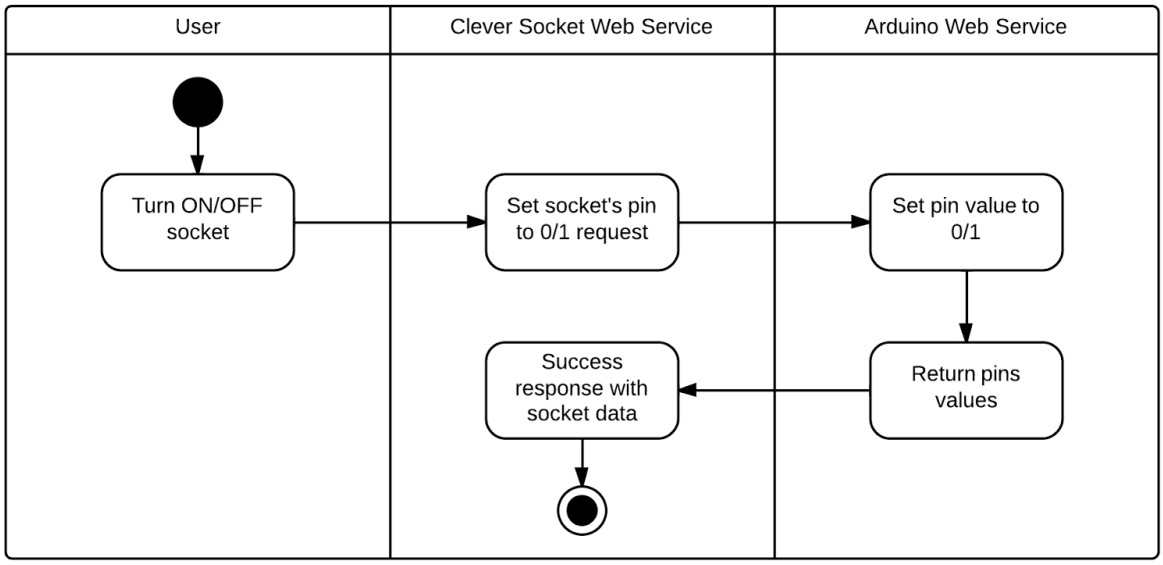


Diagrama 14: Diagrama de actividad del encendido/apagado de un *socket*, Servicio Web Clever Socket

Conmutado de un *socket*

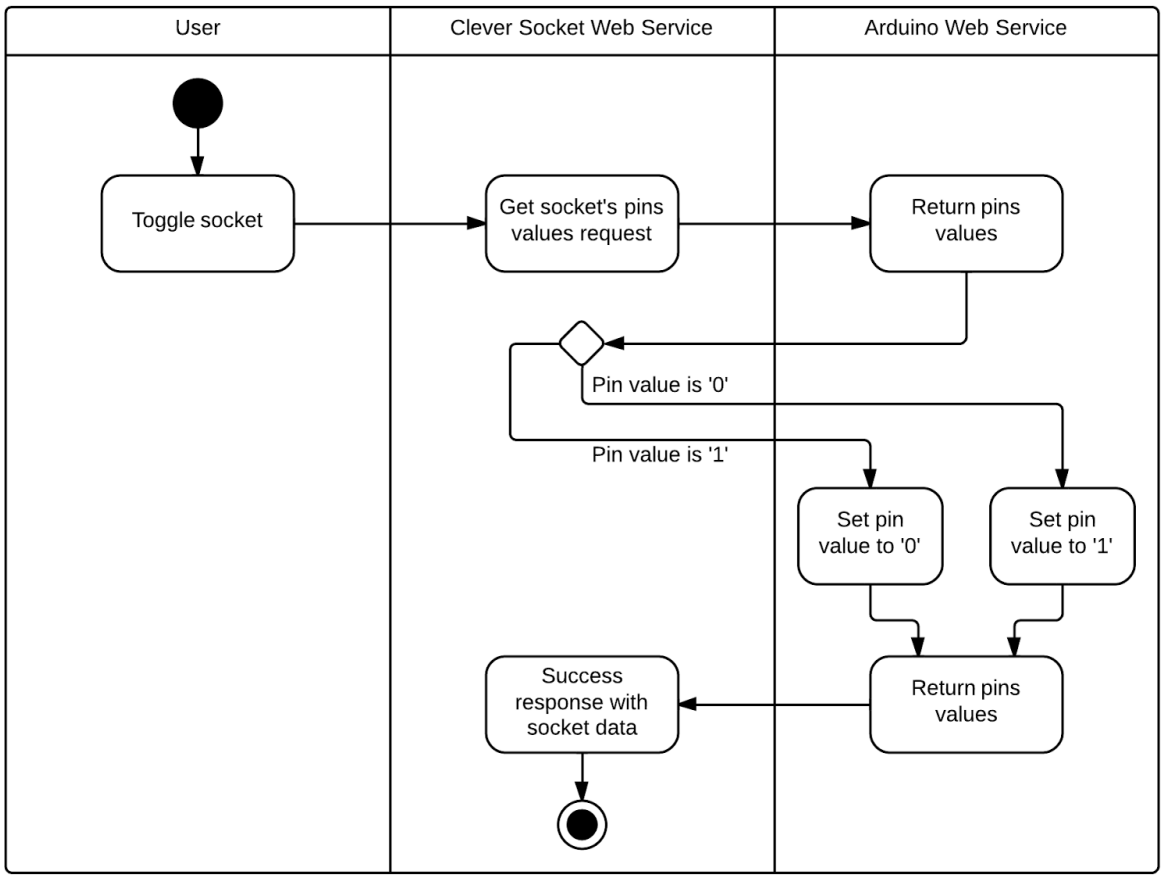


Diagrama 15: Diagrama de actividad del conmutado de un *socket*, Servicio Web Clever Socket

Lectura de tareas

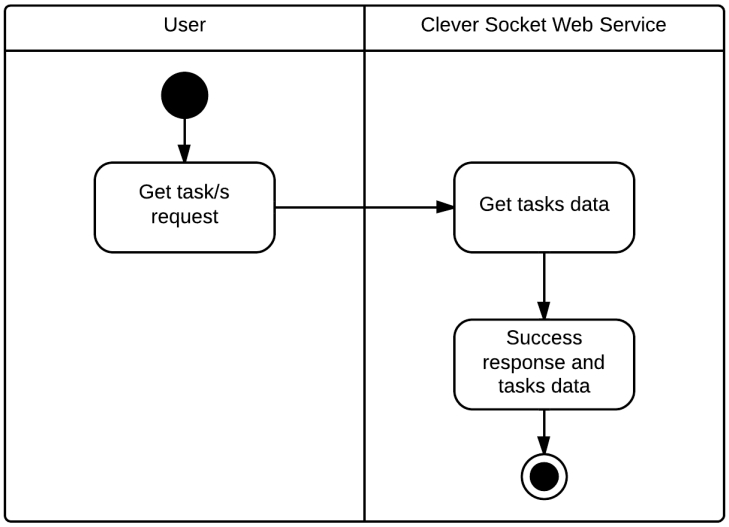


Diagrama 16: Diagrama de actividad de lectura de tareas, Servicio Web Clever Socket

Crear tarea

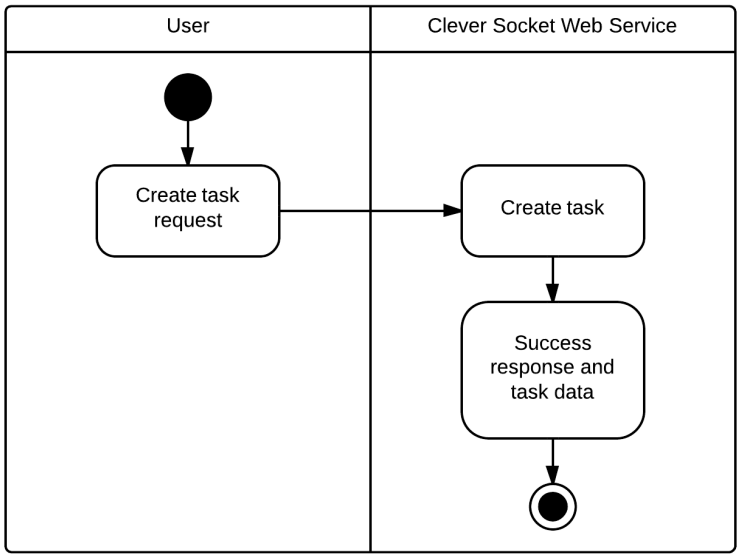


Diagrama 17: Diagrama de actividad de crear tarea, Servicio Web Clever Socket

Eliminar tarea

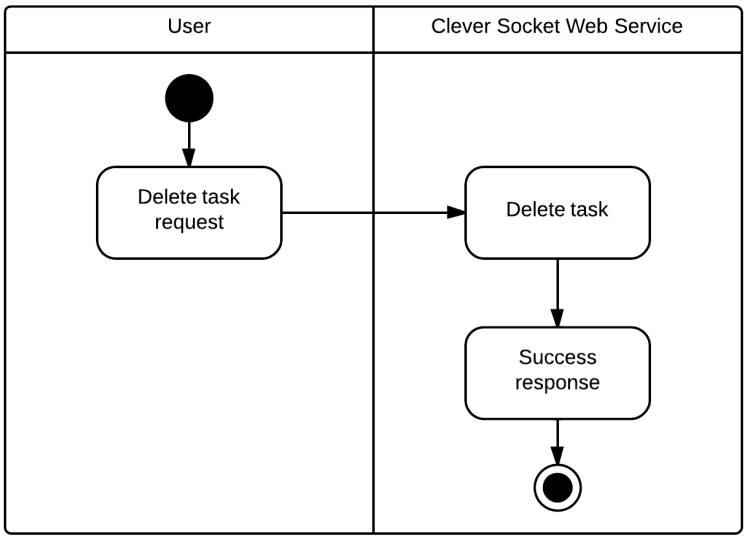


Diagrama 18: Diagrama de actividad de eliminar tarea, Servicio Web Clever Socket

Lectura registros

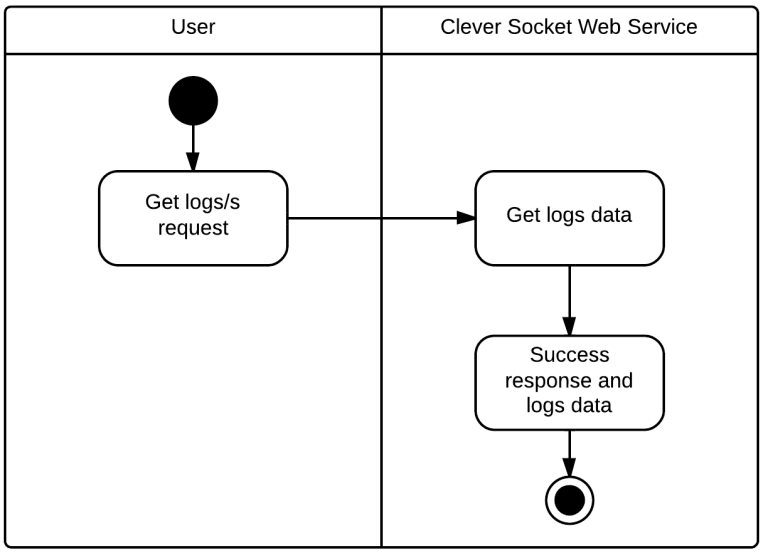


Diagrama 19: Diagrama de actividad de lectura de registros, Servicio Web Clever Socket

19.7.3 Diagramas de secuencia

Proceso de autenticación mediante HMAC (Hash-based message authentication code)

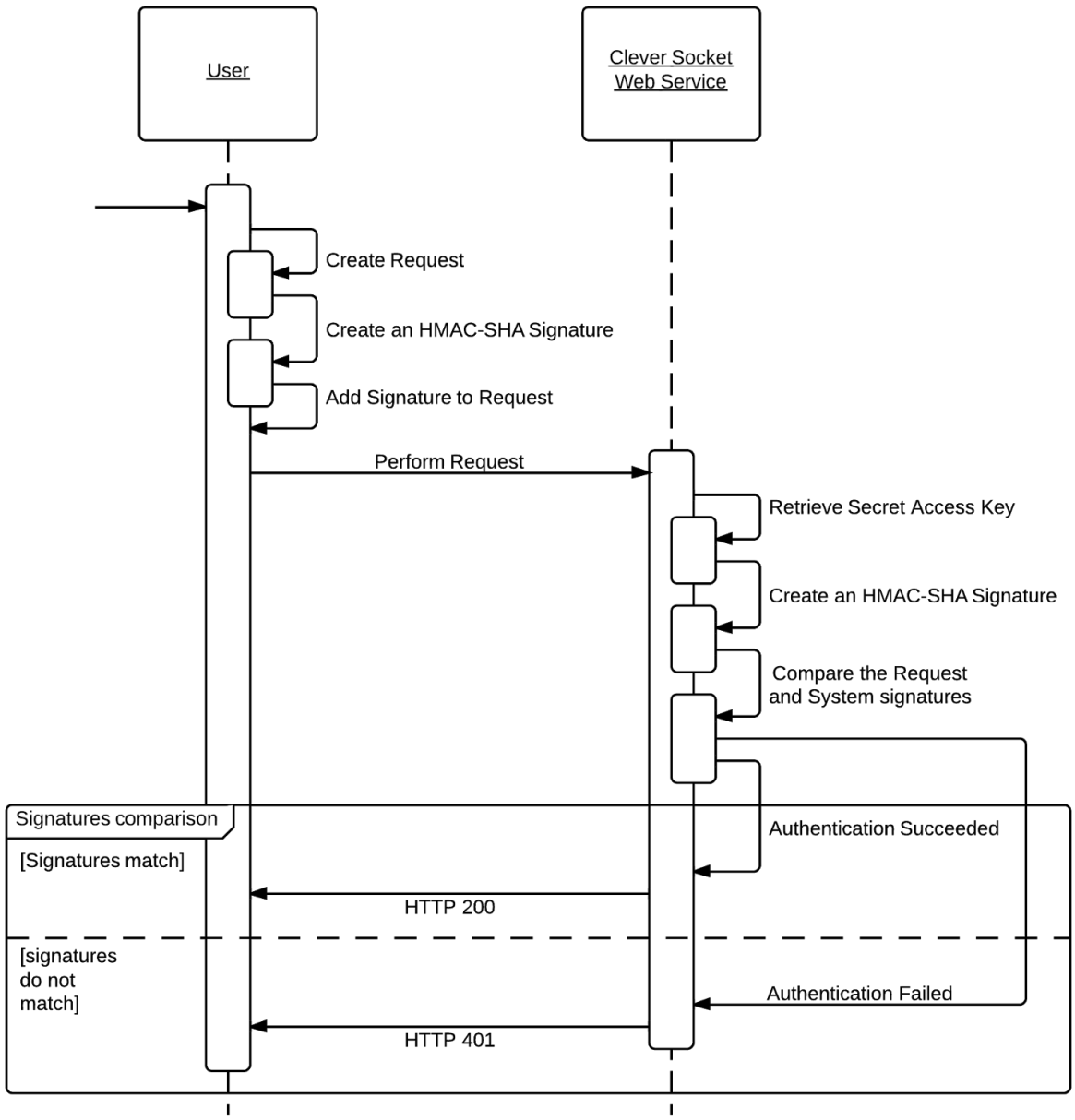


Diagrama 20: Diagrama de secuencia del proceso de autenticación mediante HMAC

Lectura de estados de los *sockets*

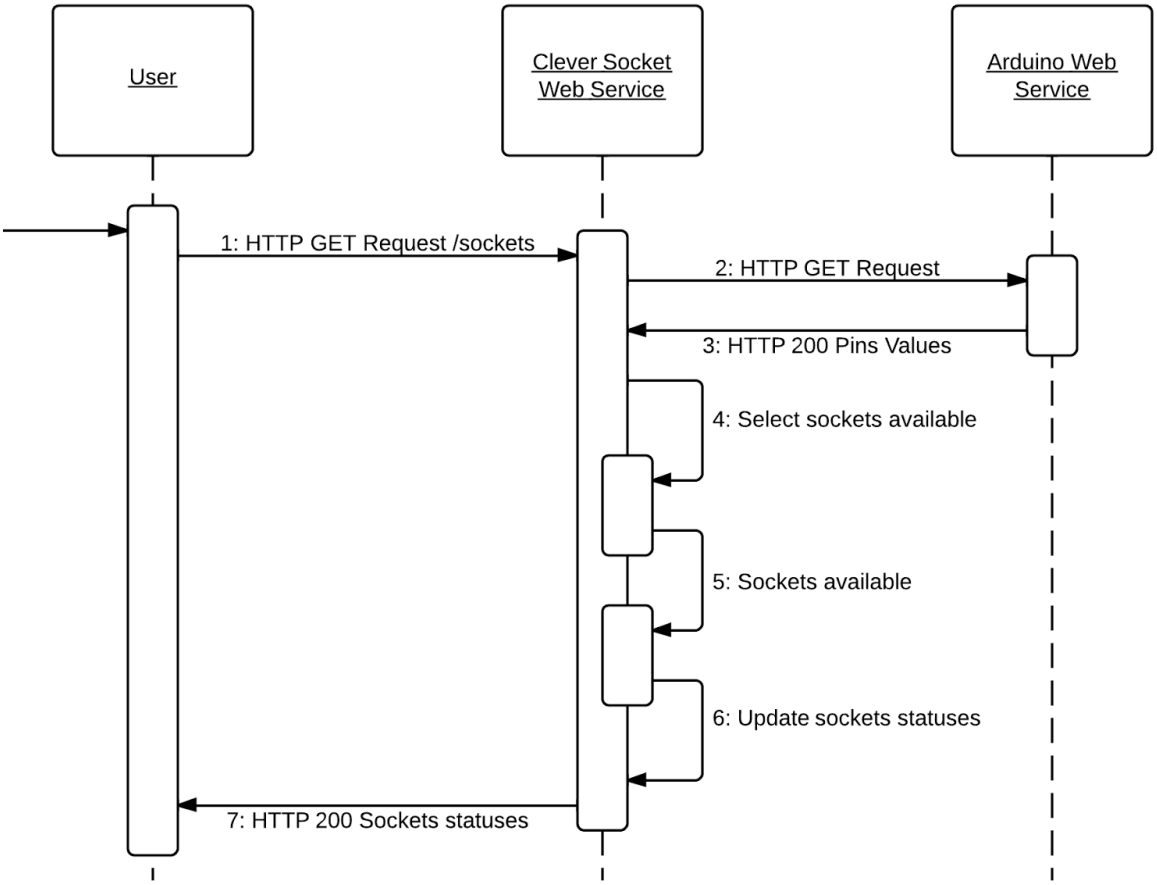


Diagrama 21: Diagrama de secuencia de la lectura de estados de los *sockets*, Servicio Web Clever Socket

Encendido/apagado de un *socket*

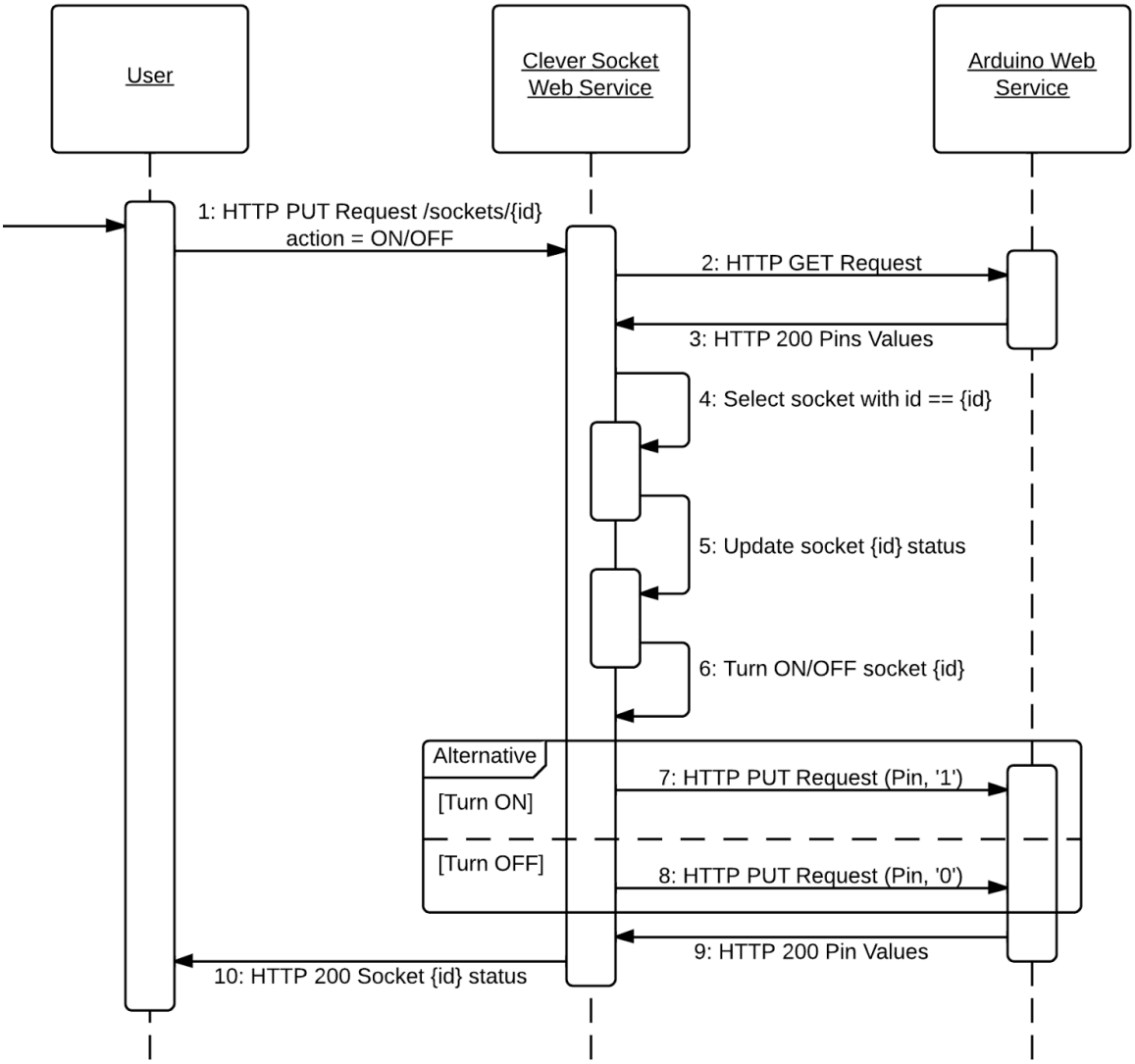


Diagrama 22: Diagrama de secuencia del encendido/apagado de un *socket*, Servicio Web Clever Socket

Conmutado de un *socket*

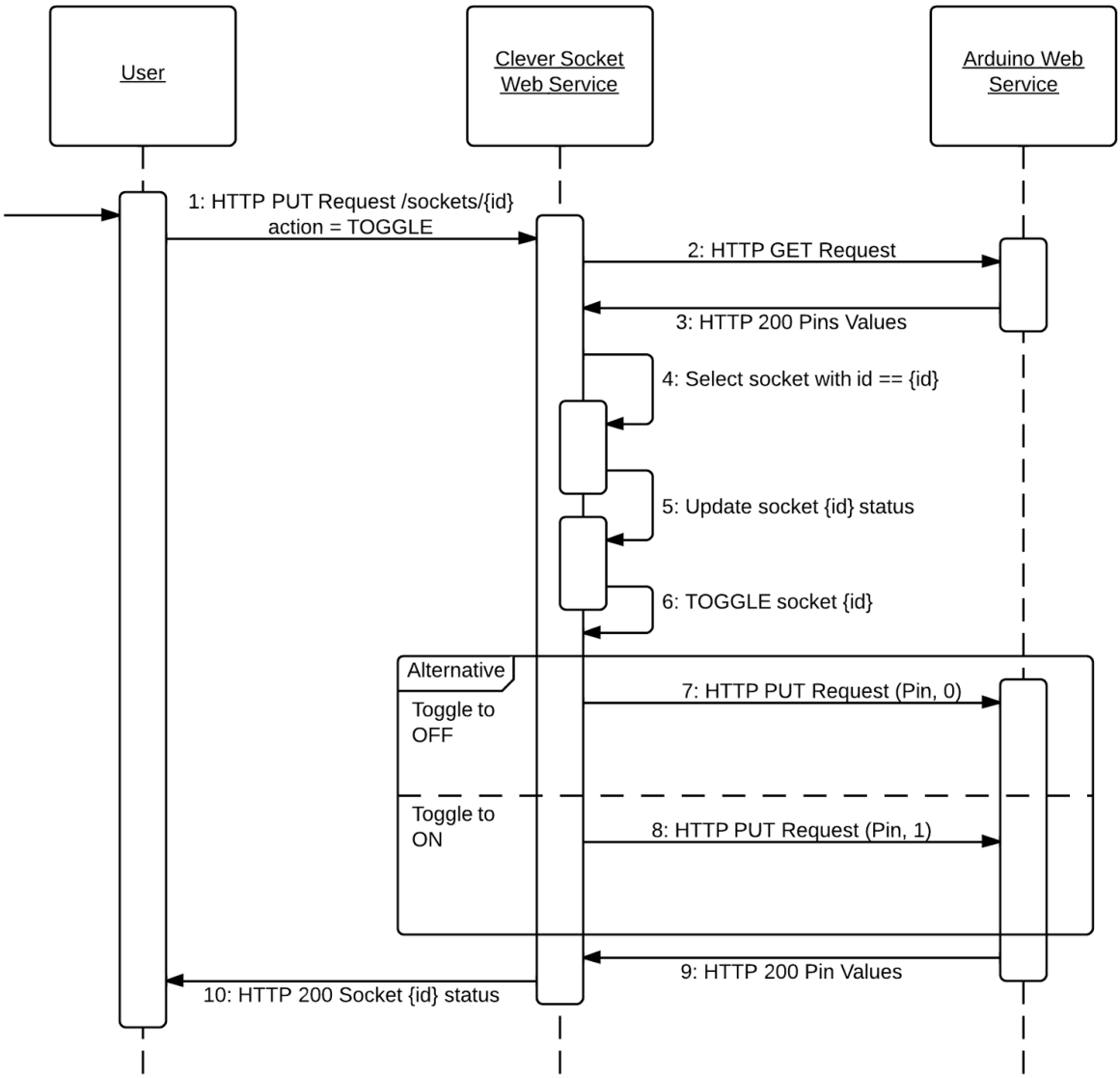


Diagrama 23: Diagrama de secuencia del conmutado de un *socket*, Servicio Web Clever Socket

Lectura de tareas

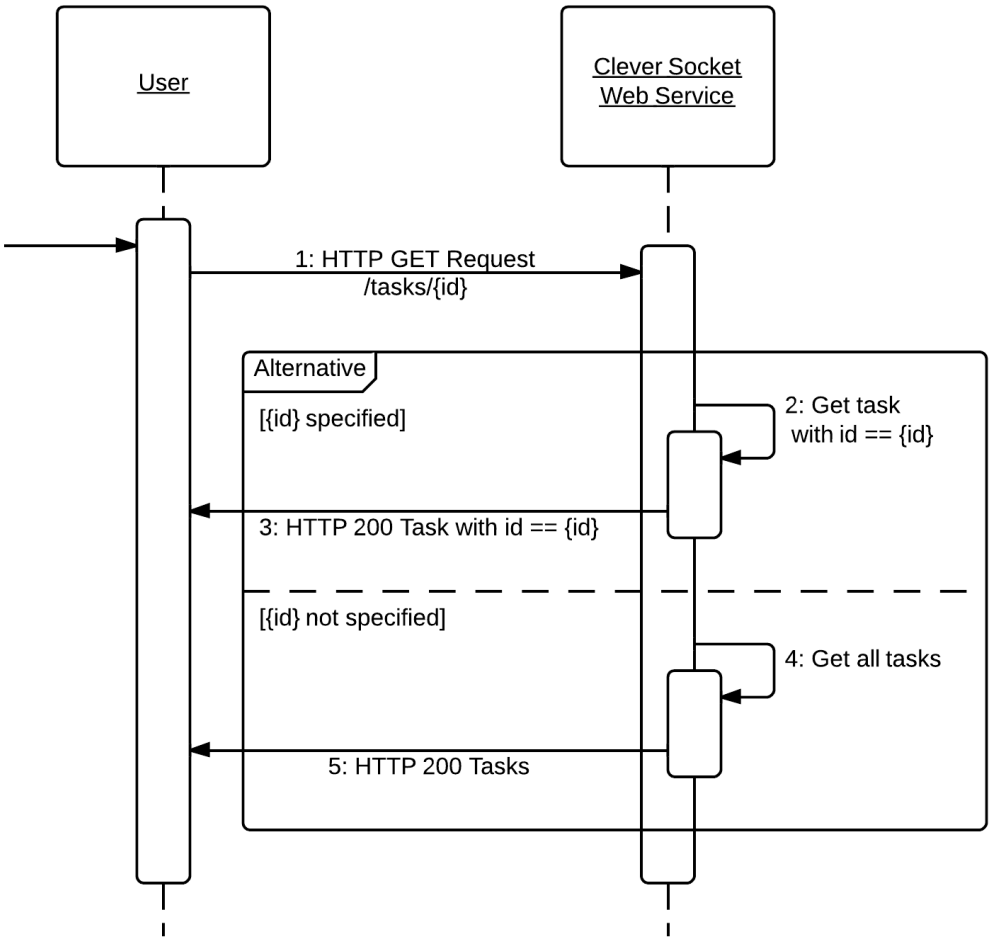


Diagrama 24: Diagrama de secuencia de la lectura de tareas, Servicio Web Clever Socket

Crear tarea

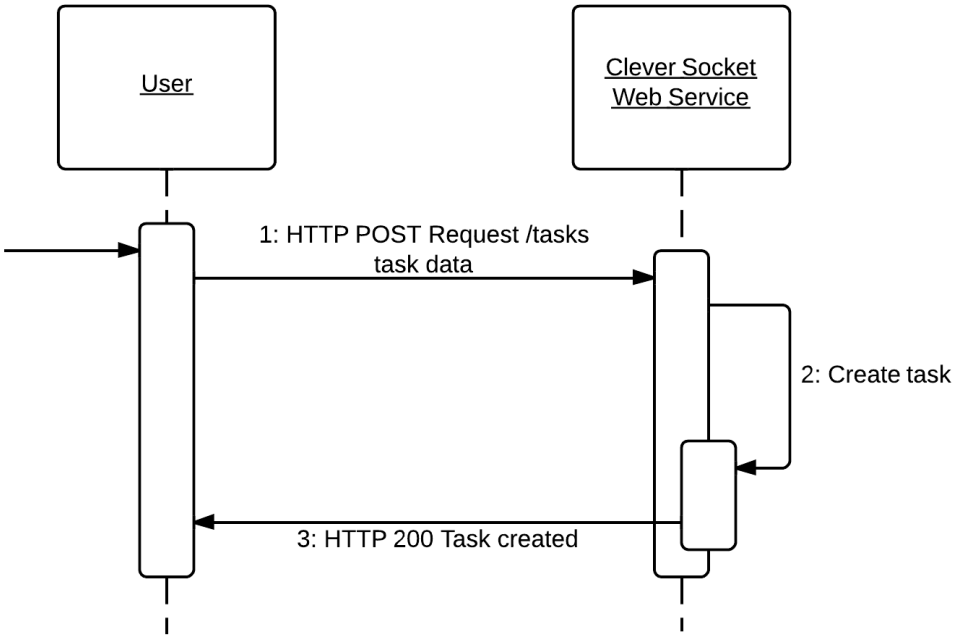


Diagrama 25: Diagrama de secuencia de crear tarea, Servicio Web Clever Socket

Eliminar tarea

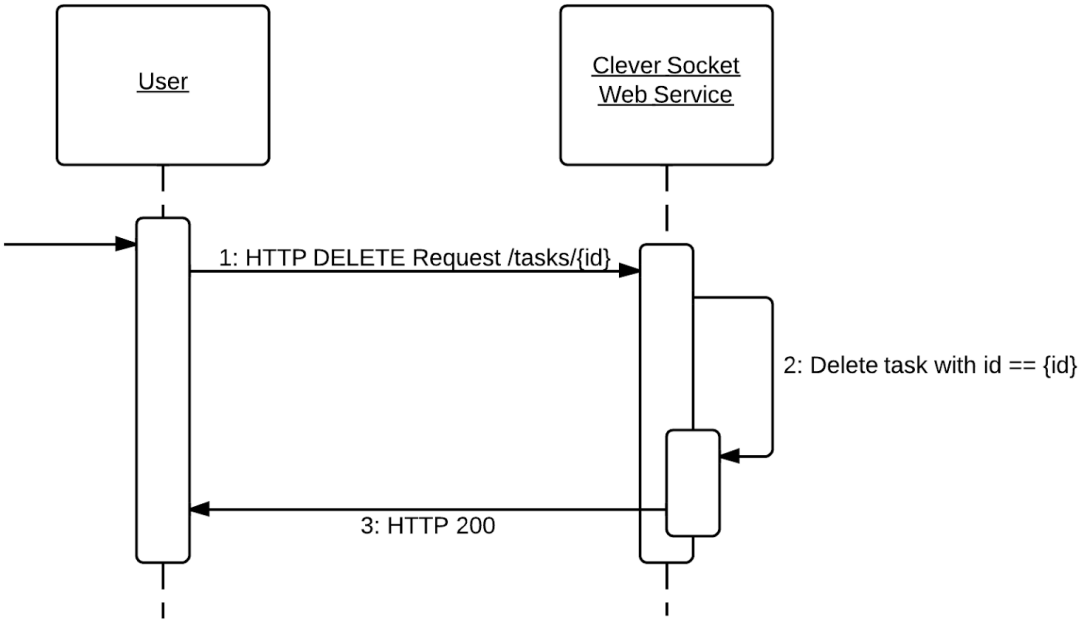


Diagrama 26: Diagrama de secuencia de eliminar tarea, Servicio Web Clever Socket

Lectura de registros

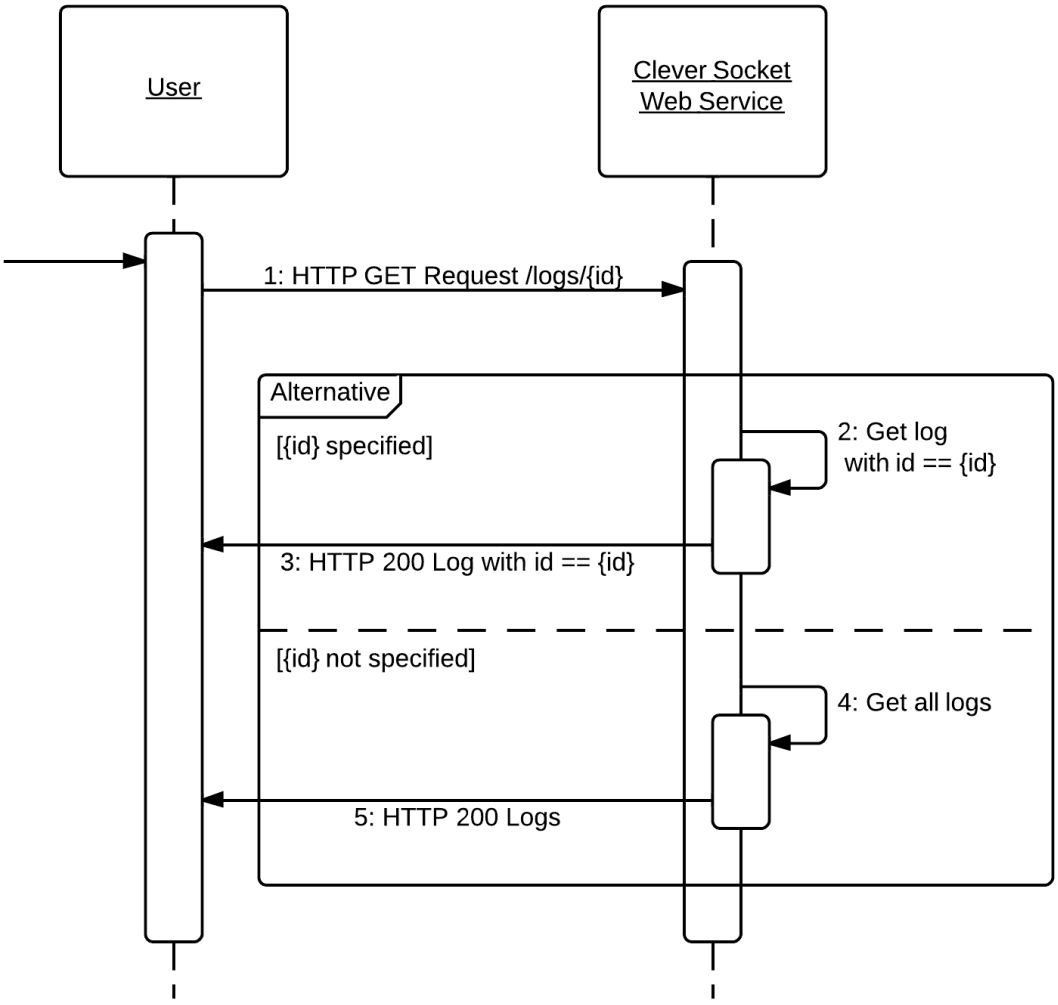


Diagrama 27: Diagrama de secuencia de lectura de registros, Servicio Web Clever Socket

19.7.4 Diagrama y estructura de la base de datos

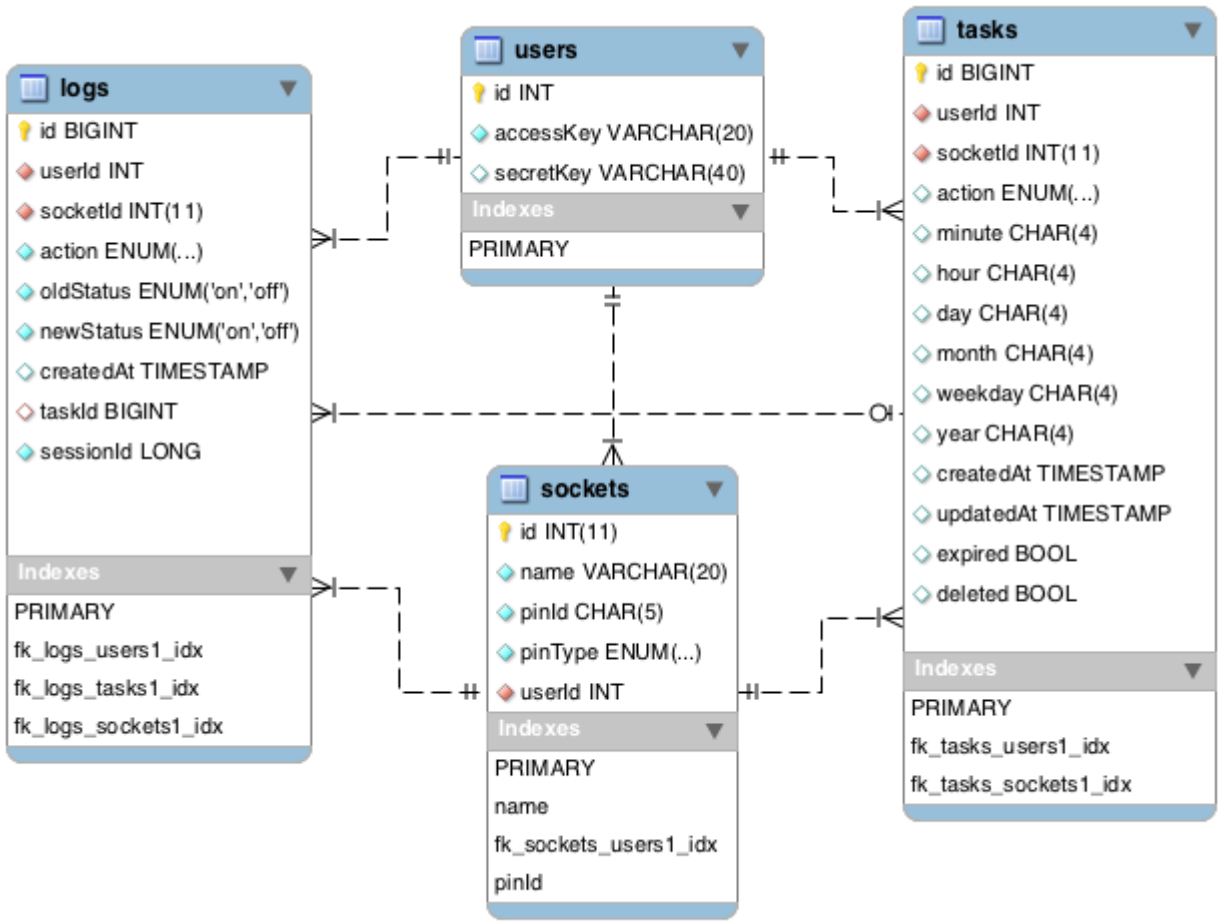


Diagrama 28: Base de datos api-clever-socket

api-clever-socket

```
SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0;

SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS,
FOREIGN_KEY_CHECKS=0;

SET @OLD_SQL_MODE=@@SQL_MODE,
SQL_MODE='TRADITIONAL,ALLOW_INVALID_DATES';

-- -----------------------------------------------------

-- Schema api-clever-socket
```

```
-- -----------------------------------------------------

CREATE SCHEMA IF NOT EXISTS `api-clever-socket` DEFAULT CHARACTER
SET utf8 ;

USE `api-clever-socket` ;

-- -----------------------------------------------------

-- Table `api-clever-socket`.`users`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `api-clever-socket`.`users` (

  `id` INT NOT NULL AUTO_INCREMENT,

  `accessKey` VARCHAR(20) NOT NULL,

  `secretKey` VARCHAR(40) NULL,

  PRIMARY KEY (`id`))

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `api-clever-socket`.`sockets`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `api-clever-socket`.`sockets` (

  `id` INT(11) NOT NULL AUTO_INCREMENT,

  `name` VARCHAR(20) NOT NULL,

  `pinId` CHAR(5) NOT NULL,

  `pinType` ENUM(`analog`,`digital`) NOT NULL,

  `userId` INT NOT NULL,

  PRIMARY KEY (`id`),

  INDEX `name` (`name` ASC),

  INDEX `fk_sockets_users1_idx` (`userId` ASC),
```

```
  INDEX `pinId` (`pinId` ASC),
  CONSTRAINT `fk_sockets_users1`
    FOREIGN KEY (`userId`)
    REFERENCES `api-clever-socket`.`users` (`id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB
AUTO_INCREMENT = 5
DEFAULT CHARACTER SET = utf8;
-- -----------------------------------------------------
-- Table `api-clever-socket`.`tasks`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `api-clever-socket`.`tasks` (
  `id` BIGINT NOT NULL AUTO_INCREMENT,
  `userId` INT NOT NULL,
  `socketId` INT(11) NOT NULL,
  `action` ENUM('on','off','toggle') NULL,
  `minute` CHAR(4) NULL DEFAULT '*',
  `hour` CHAR(4) NULL DEFAULT '*',
  `day` CHAR(4) NULL DEFAULT '*',
  `month` CHAR(4) NULL DEFAULT '*',
  `weekday` CHAR(4) NULL DEFAULT '*',
  `year` CHAR(4) NULL DEFAULT '*',
  `createdAt` TIMESTAMP NULL DEFAULT NOW(),
  `updatedAt` TIMESTAMP NULL DEFAULT NOW(),
```

```
  `expired` TINYINT(1) NULL DEFAULT 0,
  `deleted` TINYINT(1) NULL DEFAULT 0,
  PRIMARY KEY (`id`),
  INDEX `fk_tasks_users1_idx` (`userId` ASC),
  INDEX `fk_tasks_sockets1_idx` (`socketId` ASC),
  CONSTRAINT `fk_tasks_users1`
    FOREIGN KEY (`userId`)
    REFERENCES `api-clever-socket`.`users` (`id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_tasks_sockets1`
    FOREIGN KEY (`socketId`)
    REFERENCES `api-clever-socket`.`sockets` (`id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;

-- -----------------------------------------------------
-- Table `api-clever-socket`.`logs`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `api-clever-socket`.`logs` (
  `id` BIGINT NOT NULL AUTO_INCREMENT,
  `userId` INT NOT NULL,
  `socketId` INT(11) NOT NULL,
  `action` ENUM('on','off','toggle') NOT NULL,
  `oldStatus` ENUM('on','off') NOT NULL,
```

```
  `newStatus` ENUM('on','off') NOT NULL,
  `createdAt` TIMESTAMP NULL DEFAULT NOW(),
  `taskId` BIGINT NULL,
  `sessionId` MEDIUMTEXT NOT NULL,
  PRIMARY KEY (`id`),
  INDEX `fk_logs_users1_idx` (`userId` ASC),
  INDEX `fk_logs_tasks1_idx` (`taskId` ASC),
  INDEX `fk_logs_sockets1_idx` (`socketId` ASC),
  CONSTRAINT `fk_logs_users1`
    FOREIGN KEY (`userId`)
    REFERENCES `api-clever-socket`.`users` (`id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_logs_tasks1`
    FOREIGN KEY (`taskId`)
    REFERENCES `api-clever-socket`.`tasks` (`id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_logs_sockets1`
    FOREIGN KEY (`socketId`)
    REFERENCES `api-clever-socket`.`sockets` (`id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
SET SQL_MODE=@OLD_SQL_MODE;
```

```
SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS;

SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS;
```

Tabla 27: Definición de la base de datos api-clever-socket

19.7.5 Operaciones

`/sockets`

Método	**GET**
Verbo	`/sockets`
Descripción	Devuelve una lista de los *sockets* disponibles e información sobre sus estados.

Cabeceras	• Content-Type:
	  ○ application/x-www-form-urlencoded
	• Date:
	  ○ RFC 850 (ejemplo: Tue, 03 Jun 2014 09:49:07 GMT)
	• Authorization:
	  ○ {accessKey}:{signature}

Ejemplo	GET http://api.cleversocket.es/sockets
	
	// Request headers
	
	Content-Type: application/x-www-form-urlencoded
	Authorization:
	5EB929317E2F1A678C17:ZGJiMWQ3ZmMxN2FjZWY5MWUxNDQwNTEwNDY2ZTgyMGQ0NDc5ZGEwZQ==
	Date: Tue, 03 Jun 2014 09:49:07 GMT
	
	// Response
	
	200
	
	Content-Type: application/json
	
	{
	  “sockets” : [
	  {
	  “id”: 1,
	  “type”: “socket”,
	  “name”: “Socket 1”,

	"pin": {
	"id": 2,
	"value": 0,
	"type": "digital"
	}
	},
	// ...
	]
	}

Tabla 28: Operación GET /sockets

Método	**GET**
Verbo	`/sockets/{id}`
Descripción	Devuelve el *socket* con el `{id}` solicitado e información sobre su estado.

Cabeceras	• Content-Type:
	  ○ application/x-www-form-urlencoded
	• Date:
	  ○ RFC 850 (ejemplo: Tue, 03 Jun 2014 09:49:07 GMT)
	• Authorization:
	  ○ {accessKey}:{signature}

Ejemplo	GET http://api.cleversocket.es/sockets/3
	
	// Request headers
	
	Content-Type: application/x-www-form-urlencoded
	Authorization:
	5EB929317E2F1A678C17:ZGJiMWQ3ZmMxN2FjZWY5MWUxNDQwNTEwNDY2ZTgyMGQ0NDc5ZGEwZQ==
	Date: Tue, 03 Jun 2014 09:49:07 GMT
	
	// Response
	
	200
	
	Content-Type: application/json
	
	{
	"socket": {
	"id": 3,
	"type": "socket",
	"name": "Socket 3",
	"pin": {
	"id": 5,

	"value": 0, "type": "digital" } } }

Tabla 29: Operación GET /sockets/{id}

Método	**PUT**
Verbo	/sockets/{id}
Descripción	Modifica el estado del *socket* con el `{id}` indicado.
Parámetros	• Encender, apagar o conmutar un *socket*: ○ action={ON|OFF|TOGGLE}

Cabeceras	• Content-Type:
	  ○ application/x-www-form-urlencoded
	• Date:
	  ○ RFC 850 (ejemplo: Tue, 03 Jun 2014 09:49:07 GMT)
	• Authorization:
	  ○ {accessKey}:{signature}

Ejemplo	PUT http://api.cleversocket.es/sockets/3
	
	// Request headers
	
	Content-Type: application/x-www-form-urlencoded
	Authorization:
	5EB929317E2F1A678C17:ZGJiMWQ3ZmMxN2FjZWY5MWUxNDQwNTEwNDY2ZTgyMGQ0NDc5ZGEwZQ==
	Date: Tue, 03 Jun 2014 09:49:07 GMT
	
	// Request body
	
	action=TOGGLE
	
	// Response
	
	200
	
	Content-Type: application/json
	
	{
	"socket": {
	"id": 3,
	"type": "socket",

	“name”: “Socket 3”,
	“pin”: {
	“id”: 5,
	“value”: 1,
	“type”: “digital”
	}
	}
	}

Tabla 30: Operación PUT /sockets/{id}

/tasks

Método	**GET**
Verbo	/tasks
Descripción	Muestra las tareas programadas.
Cabeceras	• Content-Type: ○ application/x-www-form-urlencoded • Date: ○ RFC 850 (ejemplo: Tue, 03 Jun 2014 09:49:07 GMT) • Authorization: ○ {accessKey}:{signature}

Ejemplo	GET http://api.cleversocket.es/tasks
	
	// Request headers
	
	Content-Type: application/x-www-form-urlencoded
	Authorization:
	5EB929317E2F1A678C17:ZGJiMWQ3ZmMxN2FjZWY5MWUxNDQwNTEwNDY2ZTgyMGQ0NDc5ZGEwZQ==
	Date: Tue, 03 Jun 2014 09:49:07 GMT
	
	// Response
	
	200
	
	Content-Type: application/json
	
	{
	“tasks” : [
	{
	“id” : 1,
	“updatedAt” : “2014-06-16 19:18:26”,
	“expired” : 0,

	"socket" : {
	// ...
	}
	},
	"day" : "9",
	"userId" : 1,
	"action" : "on",
	"year" : "2014",
	"month" : "6",
	"minute" : "8",
	"weekday" : "*",
	"hour" : "10",
	"createdAt" : "2014-06-16 19:14:36"
	},
	// ...
	],
	"foundRows" : 2,
	"_status" : 200
	}

Tabla 31: Operación GET /tasks

Método	**GET**
Verbo	`/tasks/{id}`
Descripción	Devuelve la tarea con el `{id}` solicitado.

Cabeceras	• Content-Type:
	  ○ application/x-www-form-urlencoded
	• Date:
	  ○ RFC 850 (ejemplo: Tue, 03 Jun 2014 09:49:07 GMT)
	• Authorization:
	  ○ {accessKey}:{signature}

Ejemplo	GET http://api.cleversocket.es/tasks/1
	
	// Request headers
	
	Content-Type: application/x-www-form-urlencoded
	Authorization:
	5EB929317E2F1A678C17:ZGJiMWQ3ZmMxN2FjZWY5MWUxNDQwNTEwNDY2ZTgyMGQ0NDc5ZGEwZQ==
	Date: Tue, 03 Jun 2014 09:49:07 GMT
	
	// Response

	200
	
	Content-Type: application/json
	
	{
	  “task” : {
	  “id” : 1,
	  “updatedAt” : “2014-06-19 09:32:01”,
	  “expired” : 1,
	  “socket” : {
	  // ...
	  },
	  “day” : “9”,
	  “userId” : 1,
	  “action” : “on”,
	  “year” : “2014”,
	  “month” : “6”,
	  “minute” : “8”,
	  “weekday” : “*”,
	  “hour” : “10”,
	  “createdAt” : “2014-06-16 19:14:36”
	  },
	  “_status” : 200
	}

Tabla 32: Operación GET /tasks/{id}

Método	**POST**
Verbo	/tasks
Descripción	Añade una nueva tarea.

Parámetros	• Identificador de usuario:
	  ◦ userId={userId}
	• Identificador de *socket*:
	  ◦ socketId={socketId}
	• Acción sobre *socket*:
	  ◦ action={ON|OFF|TOGGLE}
	• Minuto:
	  ◦ minute={[0-59]|*}
	• Hora:
	  ◦ hour={[0-59]|*}
	• Día:
	  ◦ day={[1-31]|*}
	• Mes:
	  ◦ month={[1-12]|*}

	• Día de la semana:
	  ◦ weekday={[0-6]|*}
	• Año:
	  ◦ year={[1900-2099]|*}

Cabeceras	• Content-Type:
	  ◦ application/x-www-form-urlencoded
	• Date:
	  ◦ RFC 850 (ejemplo: Tue, 03 Jun 2014 09:49:07 GMT)
	• Authorization:
	  ◦ {accessKey}:{signature}

Ejemplo	POST http://api.cleversocket.es/tasks
	
	// Request headers
	
	Content-Type: application/x-www-form-urlencoded
	Authorization:
	5EB929317E2F1A678C17:ZGJiMWQ3ZmMxN2FjZWY5MWUxNDQwNTEwNDY2ZTgyMGQ0NDc5ZGEwZQ==
	Date: Tue, 03 Jun 2014 09:49:07 GMT
	
	// Request body
	
	userId=1
	socketId=2
	action=ON
	minute=8
	hour=10
	day=9
	month=6
	weekday=*
	year=2014
	
	// Response
	
	200
	
	Content-Type: application/json
	
	{
	  “task” : {
	  “id” : 3,
	  “updatedAt” : “2014-06-19 10:53:01”,
	  “expired” : 0,
	  “socket” : {
	  // ...
	  }
	  },

	"day" : "9", "userId" : 1, "action" : "ON", "year" : "2014", "month" : "6", "minute" : "8", "weekday" : "*", "hour" : "10", "createdAt" : "2014-06-19 10:53:01" }, "_status" : 200 }

Tabla 33: Operación POST /tasks

Método	**DELETE**
Verbo	/tasks/{id}
Descripción	Elimina la tarea con el `{id}` solicitado.

Cabeceras	• Content-Type:
	  ○ application/x-www-form-urlencoded
	• Date:
	  ○ RFC 850 (ejemplo: Tue, 03 Jun 2014 09:49:07 GMT)
	• Authorization:
	  ○ {accessKey}:{signature}

Ejemplo	GET http://api.cleversocket.es/tasks/1
	
	// Request headers
	
	Content-Type: application/x-www-form-urlencoded
	Authorization:
	5EB929317E2F1A678C17:ZGJiMWQ3ZmMxN2FjZWY5MWUxNDQwNTEwNDY2ZTgyMGQ0NDc5ZGEwZQ==
	Date: Tue, 03 Jun 2014 09:49:07 GMT
	
	// Response
	
	200

Tabla 34: Operación DELETE /tasks/{id}

/logs

Método	**GET**

Verbo	/logs
Descripción	Muestra los registros de actividad

Cabeceras	• Content-Type:
	  ○ application/x-www-form-urlencoded
	• Date:
	  ○ RFC 850 (ejemplo: Tue, 03 Jun 2014 09:49:07 GMT)
	• Authorization:
	  ○ {accessKey}:{signature}

Ejemplo	GET http://api.cleversocket.es/logs
	
	// Request headers
	
	Content-Type: application/x-www-form-urlencoded
	Authorization:
	5EB929317E2F1A678C17:ZGJiMWQ3ZmMxN2FjZWY5MWUxNDQwNTEwNDY2ZTgyMGQ0NDc5ZGEwZQ==
	Date: Tue, 03 Jun 2014 09:49:07 GMT
	
	// Response
	
	200
	
	Content-Type: application/json
	
	{
	  “logs” : [
	  {
	  “userId” : 1,
	  “socket” : {
	  // ...
	  },
	  “action” : “toggle”,
	  “id” : 1,
	  “oldStatus” : “off”,
	  “sessionId” : “8353079”,
	  “newStatus” : “on”,
	  “taskId” : 2,
	  “createdAt” : “2014-06-16 19:16:01”
	  },
	  // ...
	  ],
	  “foundRows” : 136,
	  “_status” : 200
	}

Tabla 35: Operación GET /logs

Método	**GET**
Verbo	`/logs/{id}`
Descripción	Devuelve el registro de actividad con el `{id}` solicitado.

Cabeceras	• Content-Type:
	  ○ application/x-www-form-urlencoded
	• Date:
	  ○ RFC 850 (ejemplo: Tue, 03 Jun 2014 09:49:07 GMT)
	• Authorization:
	  ○ {accessKey}:{signature}

Ejemplo	GET http://api.cleversocket.es/logs/1
	
	// Request headers
	
	Content-Type: application/x-www-form-urlencoded
	Authorization:
	5EB929317E2F1A678C17:ZGJiMWQ3ZmMxN2FjZWY5MWUxNDQwNTEwNDY2ZTgyMGQ0NDc5ZGEwZQ==
	Date: Tue, 03 Jun 2014 09:49:07 GMT
	
	// Response
	
	200
	
	Content-Type: application/json
	
	{
	  “log” : {
	  “userId” : 1,
	  “socket” : {
	  // ...
	  }
	  },
	  “action” : “toggle”,
	  “id” : 1,
	  “oldStatus” : “off”,
	  “sessionId” : “8353079”,
	  “newStatus” : “on”,
	  “taskId” : 2,
	  “createdAt” : “2014-06-16 19:16:01”
	  },
	  “_status” : 200
	}

Tabla 36: Operación GET /logs/{id}

19.8 Anexo VIII: Diagramas de la Aplicación Web Clever Socket

19.8.1 Diagrama de casos de uso

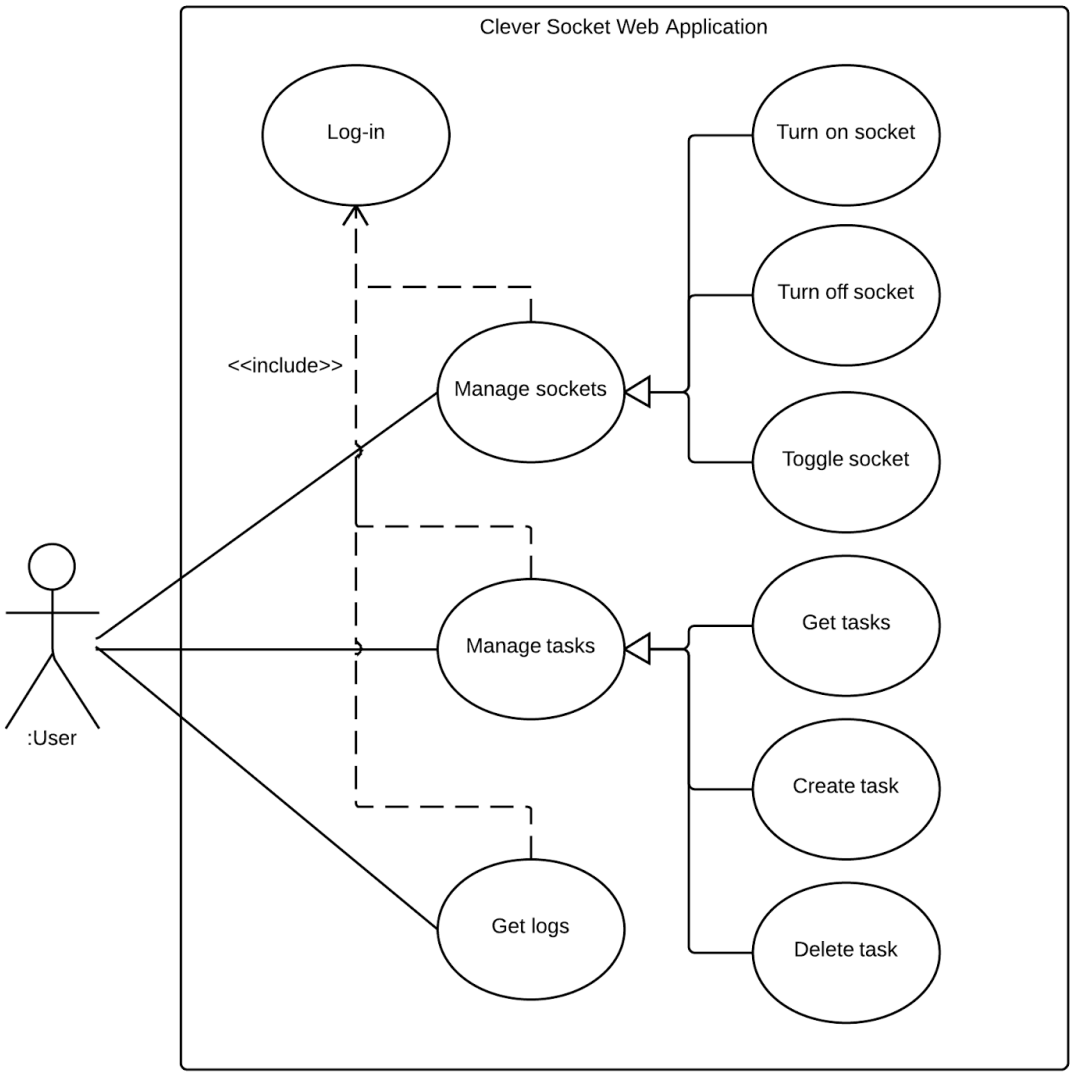


Diagrama 29: Diagrama de casos de uso, Aplicación Web Clever Socket

19.8.2 Diagrama de actividad

Gestión de los sockets

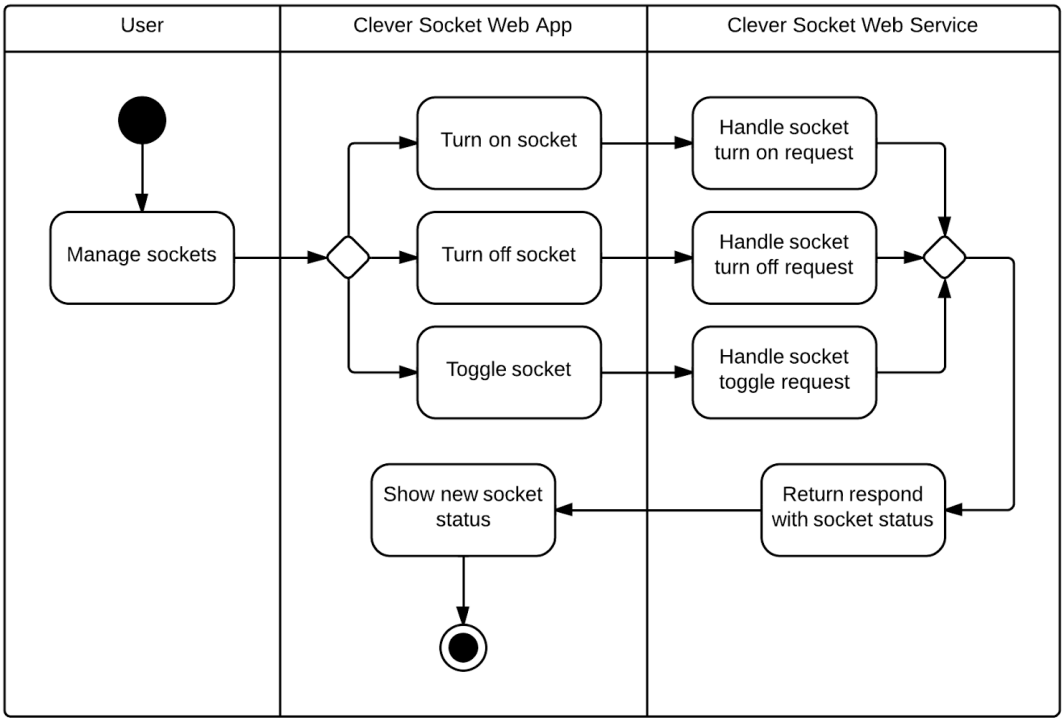


Diagrama 30: Diagrama de actividad de la gestión de los *sockets*, Aplicación Web Clever Socket

Gestión de las tareas

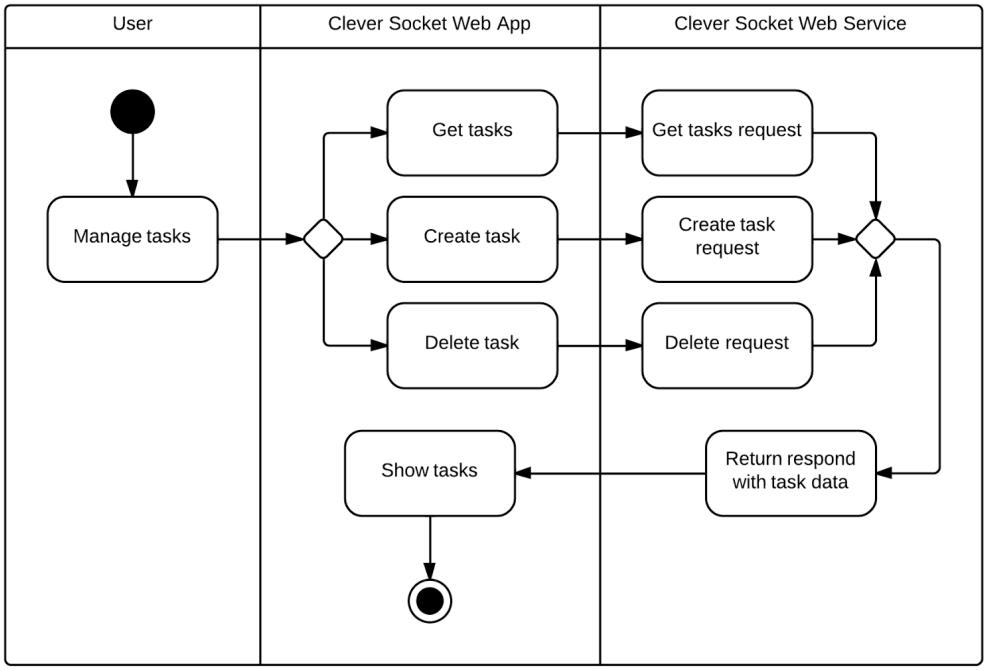


Diagrama 31: Diagrama de actividad de la gestión de las tareas, Aplicación Web Clever Socket

Lectura de los registros

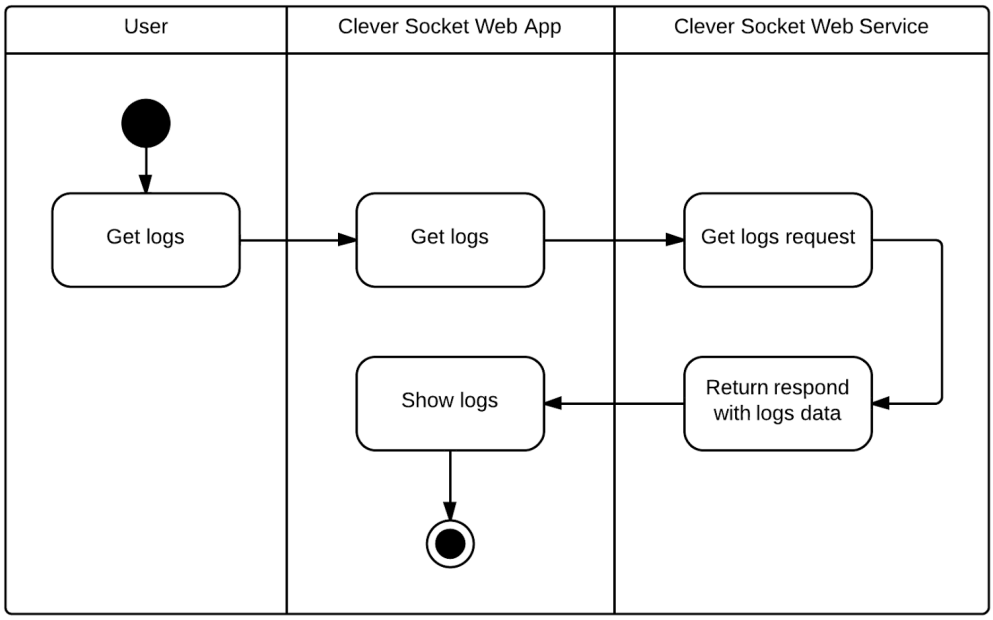


Diagrama 32: Diagrama de actividad de la lectura de los registros, Aplicación Web Clever Socket

19.8.3 Diagrama de secuencia

Gestión de los sockets

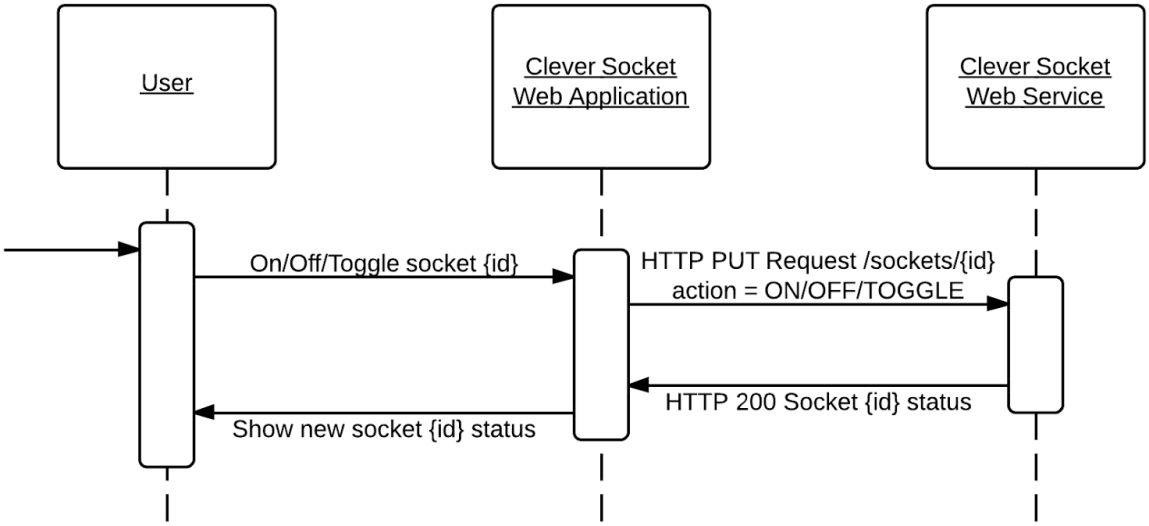


Diagrama 33: Diagrama de secuencia de la gestión de los sockets, Aplicación Web Clever Socket

Lectura de tareas

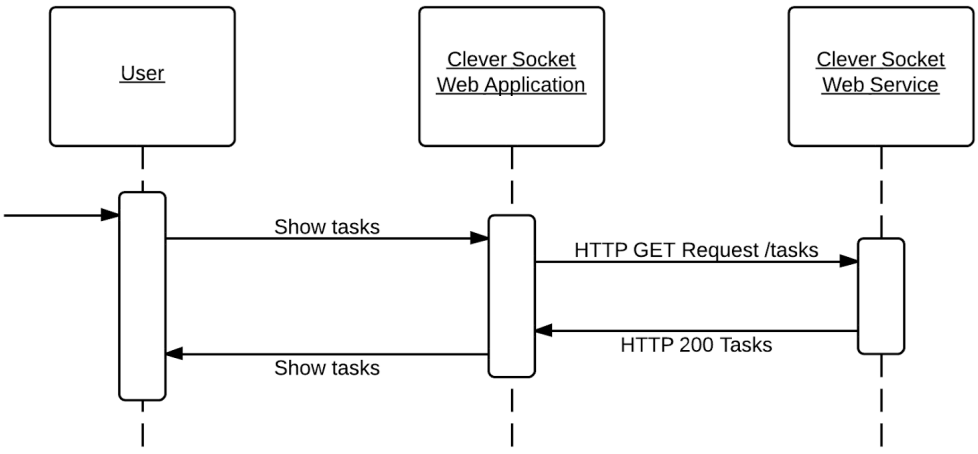


Diagrama 34: Diagrama de secuencia de la lectura de tareas, Aplicación Web Clever Socket

Crear tarea

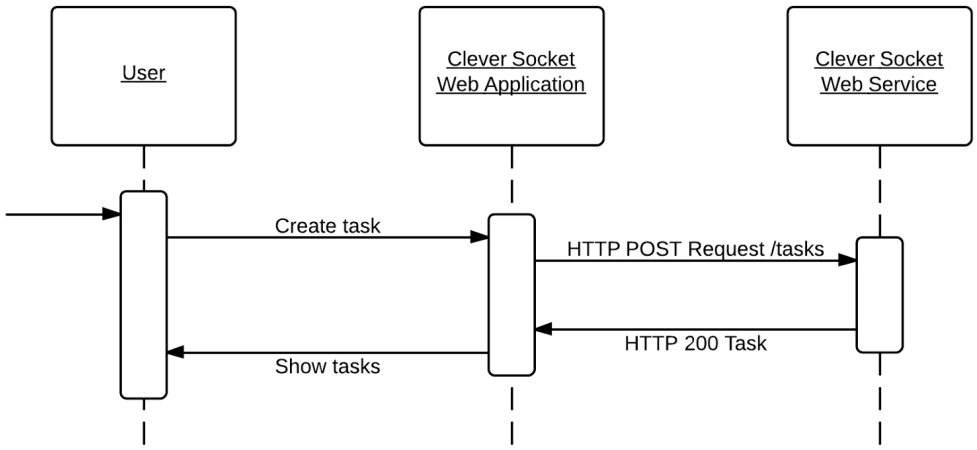


Diagrama 35: Diagrama de secuencia de crear tarea, Aplicación Web Clever Socket

Eliminar tarea

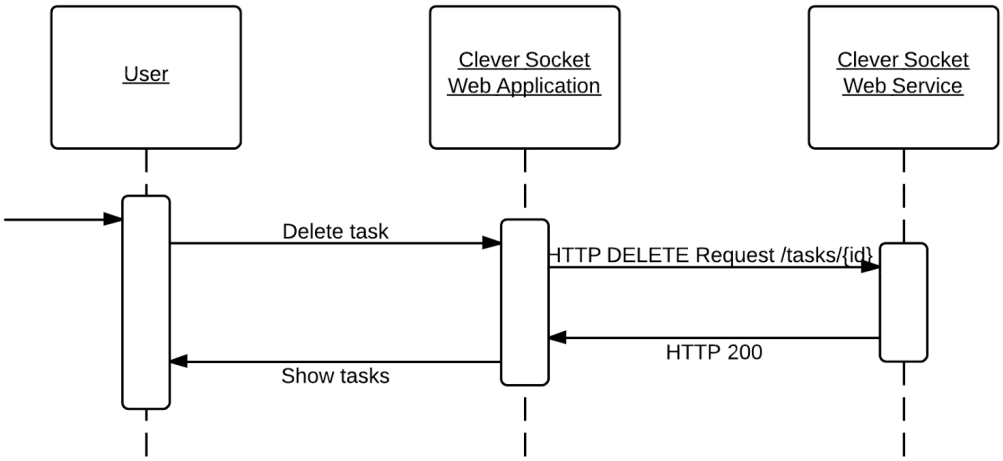


Diagrama 36: Diagrama de secuencia de eliminar tarea, Aplicación Web Clever Socket

Lectura de registros

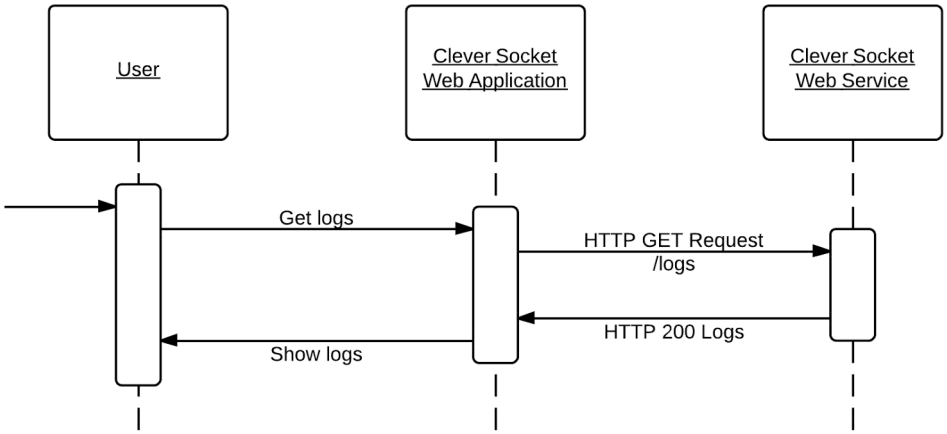


Diagrama 37: Diagrama de secuencia de lectura de registros, Aplicación Web Clever Socket

19.8.4 Diagrama y estructura de la base de datos

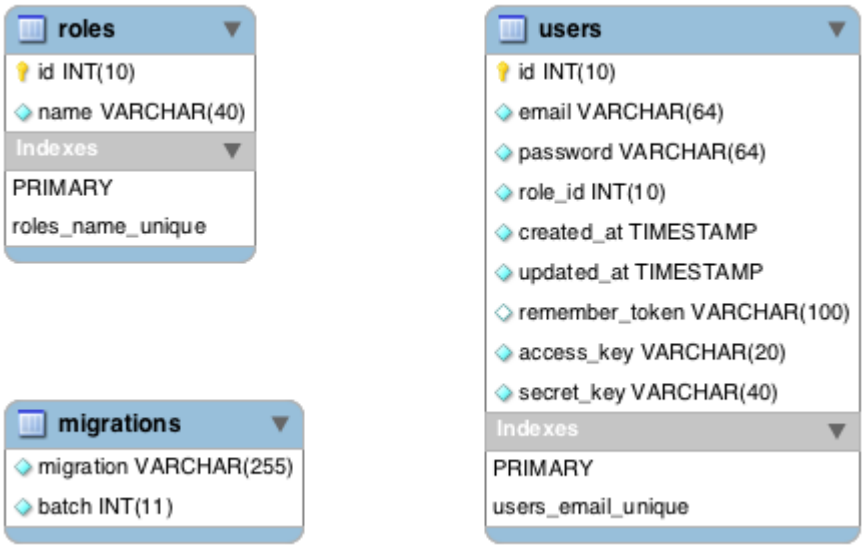


Diagrama 38: Base de datos web-clever-socket

web-clever-socket

```
-- ----------------------------------------------------
```

```
-- Schema web-clever-socket

-- -----------------------------------------------------

CREATE SCHEMA IF NOT EXISTS `web-clever-socket` DEFAULT CHARACTER
SET utf8 ;

USE `web-clever-socket` ;

-- -----------------------------------------------------

-- Table `web-clever-socket`.`migrations`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `web-clever-socket`.`migrations` (

  `migration`    VARCHAR(255)    CHARACTER    SET    'utf8'    COLLATE
'utf8_unicode_ci' NOT NULL,

  `batch` INT(11) NOT NULL)

ENGINE = InnoDB

DEFAULT CHARACTER SET = utf8

COLLATE = utf8_unicode_ci;

-- -----------------------------------------------------

-- Table `web-clever-socket`.`roles`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `web-clever-socket`.`roles` (

  `id` INT(10) UNSIGNED NOT NULL AUTO_INCREMENT,

  `name` VARCHAR(40) CHARACTER SET 'utf8' COLLATE 'utf8_unicode_ci'
NOT NULL,

  PRIMARY KEY (`id`),

  UNIQUE INDEX `roles_name_unique` (`name` ASC))

ENGINE = InnoDB

AUTO_INCREMENT = 3
```

```
DEFAULT CHARACTER SET = utf8

COLLATE = utf8_unicode_ci;

-- -----------------------------------------------------

-- Table `web-clever-socket`.`users`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `web-clever-socket`.`users` (

  `id` INT(10) UNSIGNED NOT NULL AUTO_INCREMENT,

  `email` VARCHAR(64) CHARACTER SET `utf8` COLLATE `utf8_unicode_ci`
NOT NULL,

  `password`    VARCHAR(64)    CHARACTER    SET    `utf8`    COLLATE
`utf8_unicode_ci` NOT NULL,

  `role_id` INT(10) UNSIGNED NOT NULL,

  `created_at` TIMESTAMP NOT NULL DEFAULT `0000-00-00 00:00:00`,

  `updated_at` TIMESTAMP NOT NULL DEFAULT `0000-00-00 00:00:00`,

  `remember_token`   VARCHAR(100)   CHARACTER   SET   `utf8`   COLLATE
`utf8_unicode_ci` NULL DEFAULT NULL,

  `access_key`    VARCHAR(20)    CHARACTER    SET    `utf8`    COLLATE
`utf8_unicode_ci` NOT NULL,

  `secret_key`    VARCHAR(40)    CHARACTER    SET    `utf8`    COLLATE
`utf8_unicode_ci` NOT NULL,

  PRIMARY KEY (`id`),

  UNIQUE INDEX `users_email_unique` (`email` ASC))

ENGINE = InnoDB

AUTO_INCREMENT = 3

DEFAULT CHARACTER SET = utf8

COLLATE = utf8_unicode_ci;
```

Tabla 37: Definición de la bases de datos web-clever-socket

19.9 Anexo IX: Prototipo de interfaz de usuario

Página principal

Ilustración 67: Prototipo de la página principal

Página de acceso

Ilustración 68: Prototipo de la página de acceso

Página de gestión de los *sockets*

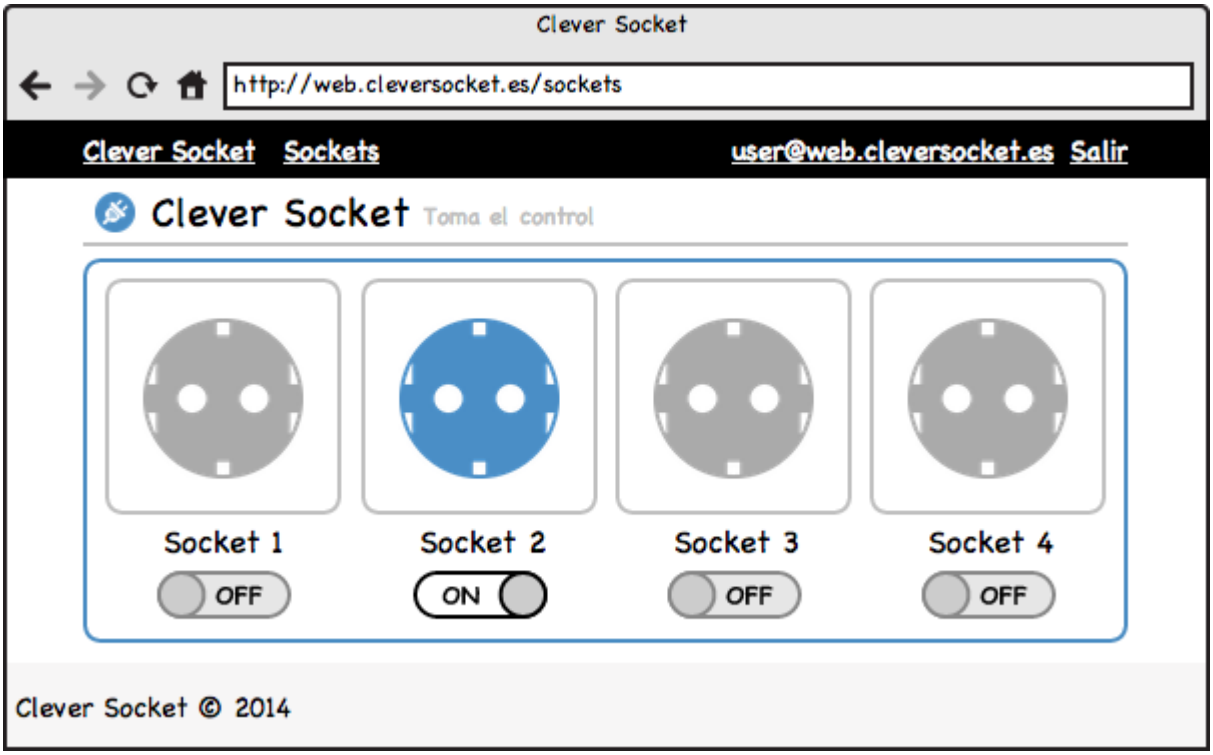


Ilustración 69: Prototipo de la gestión de los *sockets*

19.10 Anexo X: Plantilla de Licencia del MIT

The MIT License (MIT)

Copyright © <year> <copyright holders>

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.