UNIVERSIDAD AUTONOMA DE MADRID

ESCUELA POLITECNICA SUPERIOR

PROYECTO FIN DE CARRERA
Ingeniería de Telecomunicación

CONTROL REMOTO DE EQUIPOS ELÉCTRICOS MEDIANTE TERMINALES ANDROID

Antonio Rovira Moreno

Enero 2016

CONTROL REMOTO DE EQUIPOS ELÉCTRICOS MEDIANTE TERMINALES ANDROID

AUTOR: Antonio Rovira Moreno
TUTOR: Eduardo Boemo Scalvinoni

DSLab
Dpto. de Tecnología Electrónica y Comunicaciones
Escuela Politécnica Superior
Universidad Autónoma de Madrid
Enero de 2016

Resumen

Este proyecto consiste en definir, diseñar e implementar un sistema que permita el control remoto de equipos alimentados con energía eléctrica mediante el uso de un smartphone y la red de comunicaciones.

El sistema constará de varias partes (subsistemas) bien diferenciadas, teniendo **un lado remoto** donde se encuentran los equipos que se quieren controlar y **un lado usuario** a través del cual se darán las órdenes.

Ambos lados se comunicarán entre sí aprovechando las facilidades que proporciona la red de comunicaciones.

Palabras clave

Android, Arduino, terminal móvil, relé, aplicación, sketch, red de comunicaciones

Abstract

This Project consists of the design and development of a system that must be able to remotely control electrically systems using smartphone and communication network.

The project consists of several parts, an equipment to be controlled and an interface for the user.

Both sides will communicate using the Internet

Key words

Android, Arduino, mobile terminal, relay, application, sketch, communications network.

Agradecimientos

Con motivo de la finalización de mis estudios me gustaría recordar a todas aquellas personas que han contribuido a que esto sea posible. Gracias a ellas a día de hoy puedo afirmar que las experiencias vividas durante la época universitaria la han convertido sin ninguna duda en una de las mejores etapas de mi vida.

En primer lugar me gustaría agradecer a mi tutor Eduardo Boemo la posibilidad de realizar este proyecto que ha requerido mucho esfuerzo, dedicación y me ha permitido ampliar mis conocimientos sobre la parte de la carrera que más me apasiona. Ha sido toda una experiencia de la que he aprendido mucho y le estoy muy agradecido por todo ello.

Me gustaría agradecer también a la Universidad Autónoma de Madrid y especialmente a la Escuela Politécnica Superior y todo su equipo por haberme abierto las puertas y también por formarme para ser un buen profesional. Ha sido un trabajo duro que ha requerido mucho sacrificio puesto que se trata de una carrera difícil, pero también es cierto que aquello que cuesta más esfuerzo conseguir termina siendo lo que más se valora.

La gente que he conocido durante estos años también un lugar especial en mi memoria. De mis compañeros de carrera tengo el placer de presumir del cariño con el que me he sentido arropado en todo momento. Estoy seguro que la camaradería que se forja en los momentos difíciles es el mejor ejemplo de lo que significa la palabra amistad. Y yo estoy muy orgulloso de haberlo podido experimentar.

Tampoco me gustaría olvidarme de las experiencias que he vivido en la universidad con gente que pertenecía a otras carreras. Ellos me han contagiado el placer de mirar el mundo con otros ojos para comprender que muchas veces aquellos que tienen una forma diferente de pensar y de sentir son los que más tienen que enseñarte acerca de ti mismo.

También quiero agradecer a mis padres su apoyo constante durante toda la carrera. Han sido el principal pilar gracias al que he podido soportar y superar los momentos más difíciles, porque precisamente su esfuerzo y su compromiso es lo que me ha permitido forjar el carácter con el que he conseguido llegar hasta aquí

Es necesario incluir un recuerdo especial y muy emotivo a mis abuelos, que sin duda habrían disfrutado este momento como el mejor de sus vidas. De forma directa o indirecta gran parte de lo que soy se lo debo a ellos, por lo que este éxito también es suyo.

Por último me gustaría agradecer a mis familiares y amigos, especialmente a todos aquellos que me han acompañado desde la infancia y con los que he compartido tantos buenos momentos que me sería imposible enumerarlos.

Muchas gracias a todos.

INDICE DE CONTENIDOS

INDICE DE FIGURAS

INDICE DE TABLAS

1 Introducción

1.1 Motivación

En los últimos años las redes www tienen cada vez más cobertura, son más fiables y ofrecen una mayor velocidad en el intercambio de datos gracias por un lado a la implantación cada vez mayor de la fibra óptica como soporte físico en las redes fijas por un lado, y a la generalización de las tecnología 4G con una tendencia a evolucionar a la tecnología 5G con una previsible implantación para el 2020 en las redes móviles.

El uso cada vez más generalizado de terminales móviles inteligentes y una cobertura casi universal nos lleva a la idea de controlar equipos alimentados con energía eléctrica desde cualquier lugar.

1.2 Objetivos

El objetivo principal de este Proyecto es definir, diseñar e implementar un sistema que permita el control remoto de equipos alimentados con energía eléctrica mediante el uso de un smartphone y la red de comunicaciones.

El sistema constará de varias partes (subsistemas) bien diferenciadas, teniendo en primer lugar **un lado remoto** donde se encuentran los equipos que se quieren controlar y **un lado usuario** a través del cual se darán las órdenes.

Ambos lados se comunicarán entre sí aprovechando las facilidades que proporciona la red de comunicaciones.

En el lado usuario, donde se darán las órdenes tendremos:

- Un terminal Android con conexión a la red.
- Corriendo sobre el terminal Android tenemos las aplicaciones que gestionan las órdenes y las comunicaciones.

En el lado remoto, donde se encuentra físicamente los elementos a controlar tendremos:

- Un gestor de órdenes encargado de recibir las instrucciones de encendido y apagado de los distintos enchufes de la regleta y ejecutarlas, encendiendo y apagando físicamente los mismos.
- Un gestor de comunicaciones que se encargará de recibir las ordenes vía WIFI y transmitirlas al gestor de órdenes.
- Un router WIFI con conexión al mundo exterior.

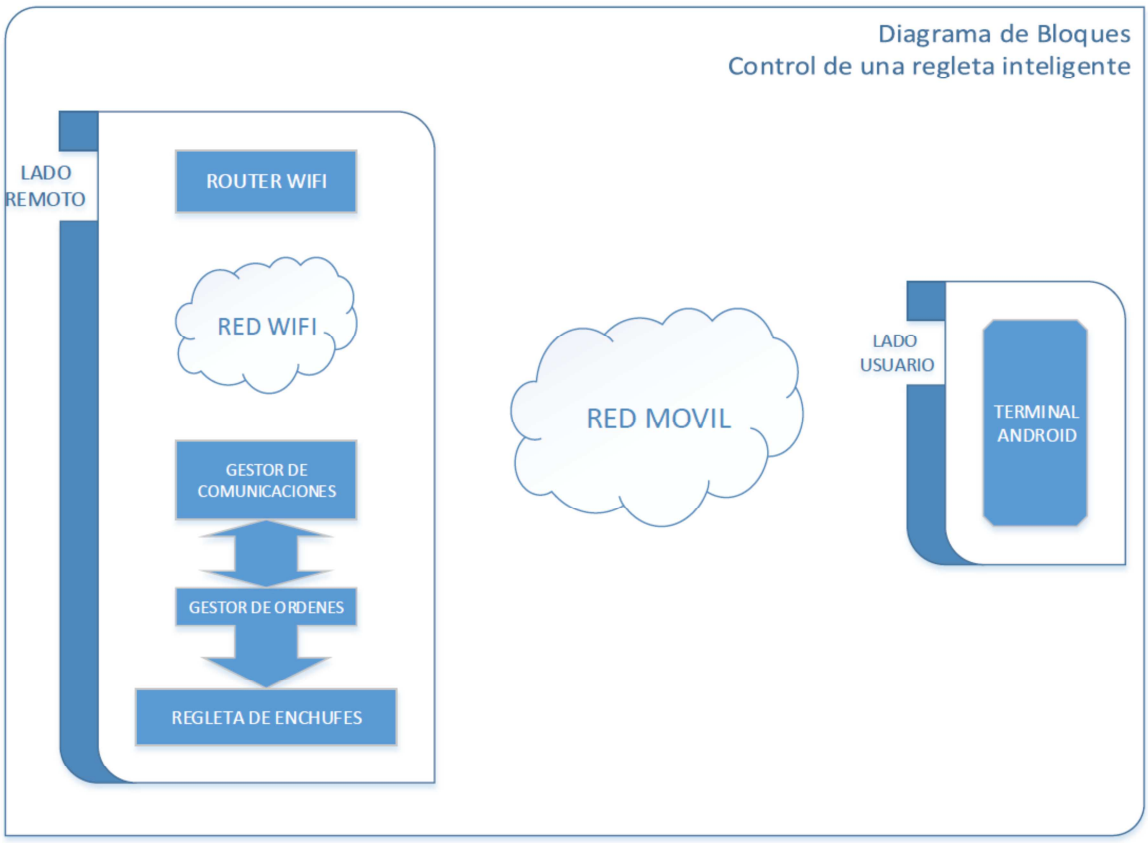


Figura 1: Diagrama de Bloques

Se pretende que el sistema sea de utilidad para el gran público por lo que se deberá tener una puesta en marcha sencilla y una interface de usuario amigable.

1.3 Organización de la memoria

La memoria consta de los siguientes capítulos:

- Capítulo 1: Introducción
- Capítulo 2: Estado del arte
- Capítulo 3: Diseño del proyecto
- Capítulo 4: Desarrollo del proyecto
- Capítulo 5: Integración, pruebas y resultados
- Capítulo 6: Conclusiones y trabajo futuro

2 Estado del arte

En este apartado se analiza la situación actual de los principales elementos necesarios para el desarrollo del proyecto, así se analiza la implantación de las redes de comunicaciones tanto las fijas ADSL y de fibra óptica, como las móviles en tecnología 4G que aunque constituyen un elemento externo son fundamentales para el buen desarrollo del proyecto.

Analizaremos también la situación de los sistemas operativos de terminales móviles con el objeto de seleccionar para el proyecto el sistema más implantado en España.

Por último se analiza la disponibilidad en el mercado de lo que hemos definido como terminal de gestión y control, un equipo capaz de comunicarse vía wifi con el mundo exterior y capaz de interpretar y ejecutar las ordenes de encendido y apagado que se le envíe desde el terminal móvil.

Así el apartado analiza:

- Redes fijas Fibra óptica
- Redes móviles Tecnología 4G
- Implantación de Sistemas Operativos de Terminales móviles
- Terminales de gestión y Control

2.1 Fibra óptica

A fecha de hoy conviven en redes fijas las tecnologías de ADSL y de fibra óptica no obstante todo tiende a que la fibra se imponga al ADSL por lo que el análisis se centra en la implantación de la fibra óptica con los datos aportados por las compañías operadoras.

En la actualidad **Movistar y Fibra Ono/Vodafone s**on los dos grandes operadores nacionales que lideran los despliegues de fibra óptica. Aunque operadores como **Jazztel** y **Orange** también tienen planes de despliegue de fibra.

Según datos de 2015 del Ministerio de Industria el 60% de los hogares están cubiertos con una conexión de tipo fibra óptica. En números absolutos, la cobertura de fibra tipo FTTH llega a 17 millones de hogares y las de HFC (conexiones de cable-fibra, como las de Fibra Ono) llegan a 10 millones. En total suman 27 millones de accesos de nueva generación en toda España.

Hasta ahora la mayoría de estas conexiones por fibra se han concentrado en las grandes ciudades, como Madrid, Barcelona, Valencia, Sevilla…. aunque la previsión es que en un plazo relativamente corto estas se extiendan a otras poblaciones más pequeñas. A modo de ejemplo Movistar espera cubrir todas las poblaciones de más de 1.000 habitantes para 2020 y llegar también a algunas de tan sólo 500 habitantes.

2.1.1 Movistar

Movistar es la compañía que lidera el despliegue de fibra óptica en España. De hecho, tanto es así que instala por defecto una conexión de fibra siempre que haya cobertura y está cerrando centrales telefónicas de ADSL donde sea posible sustituirlas por fibra.

El plan de despliegue de Movistar es muy ambicioso. A finales de este 2015 su cobertura debería haber alcanzado los 14 millones de hogares (con lo que supera a la red de Fibra Ono - Vodafone). En el 2017, esperan llegar a 20 millones de hogares. Y para 2020, esperan que la fibra de Telefónica llegue al 97% de los hogares. Esto se traduce en que todas las poblaciones de más de 1.000 habitantes contarán con fibra de Movistar.

Las conexiones que ofrece Movistar son de 30 Mb (30/3) y 300 Mb (300/30).

2.1.2 Fibra Ono - Vodafone

Desde que a mediados de 2014 Vodafone y Fibra Ono se fusionaron, ambas compañías comparten sus redes de fibra.

La red preexistente de Fibra Ono es de tipo HFC (cubre 7 millones de hogares) y la propia de Vodafone es de FTTH (con algo más de 1,2 millones de hogares cubiertos). Son técnicamente distintas pero en la práctica son equivalentes y tanto una como la otra ofrecen conexiones de 50 Mb, 120 Mb y 300 Mb. La diferencia más notable es que Vodafone ha informado de que sus conexiones sobre FTTH han sido convertidas a simétricas, pero no así las de HFC por el momento.

Vodafone comenzó el 2015 de una cobertura total de fibra que alcanzaba los 8,4 millones de hogares y debería haber llegado a 11 millones a finales del mismo año. De momento no se conocen planes de despliegue para los próximos años.

2.1.3 Orange y Jazztel

Orange compró Jazztel a mediados de 2015 y se quedó también su red de fibra óptica, de manera similar a lo ocurrido entre Vodafone y Ono. La integración entre Orange y Jazztel aún no es total aunque la previsión es que se realice en breve.

La red conjunta de Orange y Jazztel alcanza en estos momentos (finales de 2015) los 5,2 millones de hogares. Sin embargo, para finales de 2016 espera duplicar y llegar a los 10 millones de clientes.

En el 2020, llegaría a 14 millones de hogares, lo que equivale a una cobertura superior al 80% de la población.

Las ofertas de Orange y Jazztel son todavía distintas. Jazztel cuenta con conexiones de 20 Mb, 50 Mb y 200 Mb (simétricos) mientras que en Orange, las conexiones son de 30 Mb y 300 Mb (simétricas ambas opciones).

2.2 Redes Móviles Tecnología 4G

Según informe del observatorio de las telecomunicaciones y de la SI (ONTSI) en abril de 2015, el número de líneas de telefonía móvil automática superó los 53,6 millones, lo que supone un 1,4% más de líneas que en abril de 2014. Del total de líneas, 50,37 millones corresponden a líneas móviles, un 0,2% más que en la misma fecha del año anterior, situándose la tasa de penetración de la telefonía móvil en España en el 108,5% mientras que el año pasado era 108,1%.

Los 50,37 millones de líneas móviles se corresponden con la suma de líneas prepago, postpago y de datacards. De esta manera, los 53,6 millones de líneas totales, se corresponden a la suma de las líneas móviles (citado anteriormente) más las líneas asociadas a máquinas (M2M), que en este mes de abril es de 3,3 millones, lo que supone un aumento del 25,3% respecto del mismo mes del año anterior.

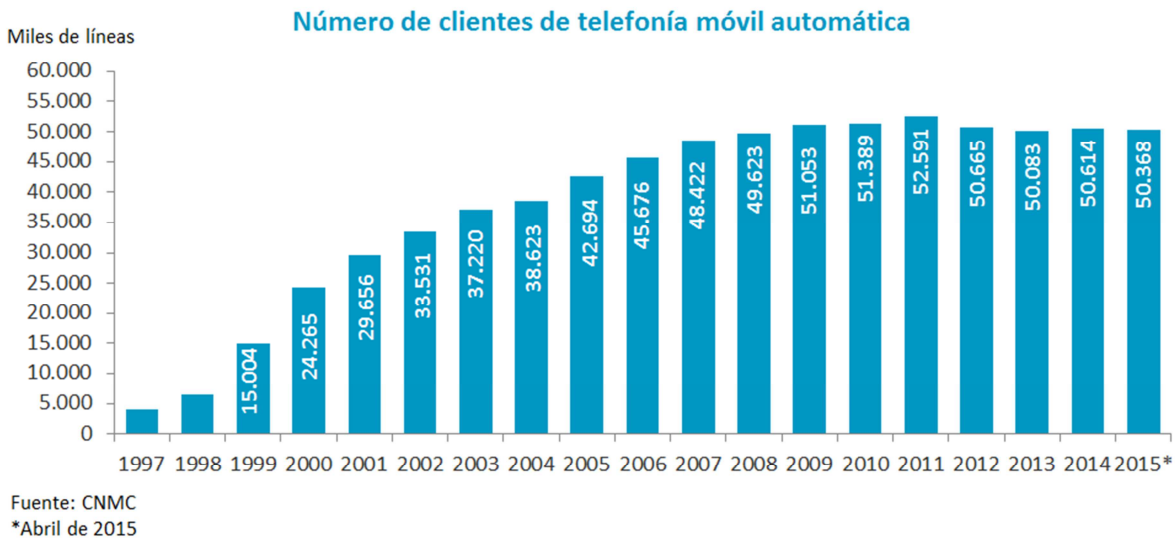


Figura 2: Evolución Telefonía Móvil Automática

2.2.1 Cobertura 4G

La **cobertura 4G** continúa extendiéndose y las operadoras siguen introduciendo mejoras para lograr más velocidad y estabilidad.

Desde la implantación del 4G en España hace dos años, el periodo ha estado marcado por el aumento de la cobertura y por las mejoras incorporadas a la red que permiten velocidades de hasta 300 Mbps en terminales compatibles. Ahora con la liberación de **la banda de frecuencias 800 MHz**, se consigue una mayor penetración en interiores y una mejor cobertura, con lo que la previsión es que el 4G deje atrás el resto de tecnologías y la convierte en la principal.

A cierre del año 2014, la cobertura 4G de Movistar alcanza al 70% de la población, la de Vodafone al 74% de la población, la de Orange al 70% y la de Yoigo al 65%. Para finales de 2015 la mayoría de operadores quiere cubrir al menos al 85% de la población, un porcentaje bastante importante.

Por otro lado tenemos la cobertura de lo que se ha dado a conocer como 4G+ o 4,5G. Esta tecnología es en realidad el nombre comercial que se ha utilizado para denominar al LTE-A, que combina la capacidad de dos o más bandas de frecuencia para aumentar la velocidad. Actualmente, este tipo de conexiones que permiten hasta 300 Mbps están disponibles en las ciudades:

- Movistar: Madrid, Barcelona.
- Vodafone: Madrid, Barcelona, Valencia, Sevilla, Bilbao, Málaga, Zaragoza y La Coruña.
- Orange: Madrid, Barcelona. Valencia.

2.2.2 Velocidad 4G

El 4G permite velocidades muy superiores a las del 3G si bien depende mucho del ancho de banda que destinen los operadores. Así pues, tenemos a Movistar y Yoigo ofreciendo 75 Mbps de bajada y 25 Mbps de subida con medias entre 20 y 40 Mbps de descarga. Por su parte, Vodafone y Orange ofrecen 150 Mbps de bajada y 50 Mbps de subida, con medias algo superiores. También tenemos el caso del LTE-A o 4G+ de Vodafone, Movistar y Orange con velocidades de hasta 300 Mbps.

2.3 Terminales Móviles

En la actualidad en España, el mercado de los terminales móviles inteligentes se reparte casi en su totalidad entre dos principales sistemas operativos, IOS y Android.

Según Device Atlas empresa que se ha constituido en el standard de facto para el análisis de trafico web, **Android** domina en 67 países, frente a los 37 de **iOS**.

IOS se imponen en los países occidentales, en los más ricos. Todos sus móviles y tablets son de gama alta. Por el contrario **Android** vence en las economías emergentes, gracias a la gran variedad de modelos y precios.

Apple controla menos territorios, pero tiene bajo su "imperio" a los más poderosos: Estados Unidos, Reino Unido, Francia y Japón, entre otros.

A modo de curiosidad, no vencen en todos los países. BlackBerry domina en Sudáfrica, con el 39% del mercado, y Windows Phone lo hace en Bangladesh, con el el 26% del tráfico web.

En España, Android genera el 57% de todo el tráfico web móvil mientras que iOS ronda el 41%.

Es este reparto de implantación en España el que justifica el uso de Android en este proyecto aunque una opción interesante de desarrollo a futuro puede ser la adaptación de las aplicaciones incluidas en el proyecto al sistema operativo IOS para el control de equipos mediante terminales Iphone

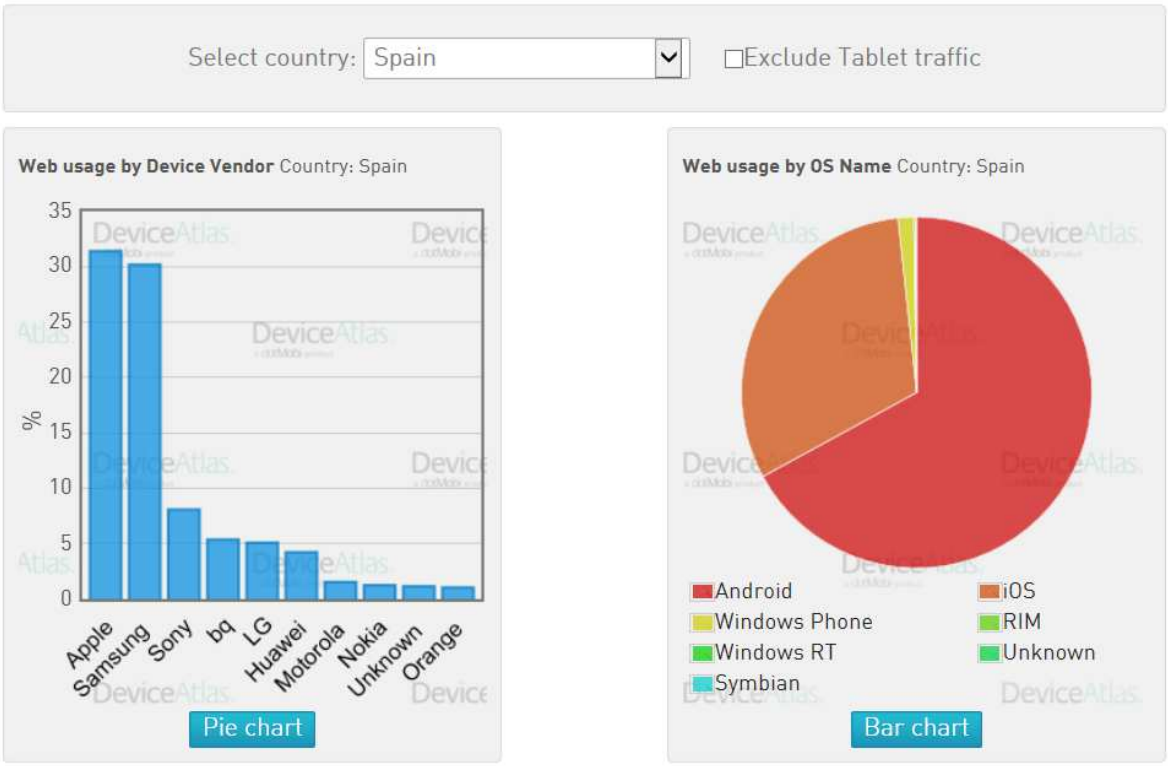


Figura 3: Implantación Android en España

2.4 Equipo de Gestión y Control

Se analizan las posibilidades que ofrece el mercado para lo que hemos definido como terminal de gestión y control, un equipo capaz de comunicarse vía wifi con el mundo exterior y sea capaz de interpretar y ejecutar las ordenes de encendido y apagado que se le envíe desde el terminal móvil.

De los equipos disponibles y que más se adecuan a las necesidades de este proyecto analizamos Raspberry y Arduino seleccionando al final el que consideramos idóneo para el proyecto.

2.4.1 Raspberry Pi

Se trata de una computadora completamente funcional que puede ejecutar un sistema operativo real en Linux. Sus puntos fuertes radican en la posibilidad de realizar varias tareas, soportar dos puertos USB y la posibilidad de conectarse de forma inalámbrica a Internet.

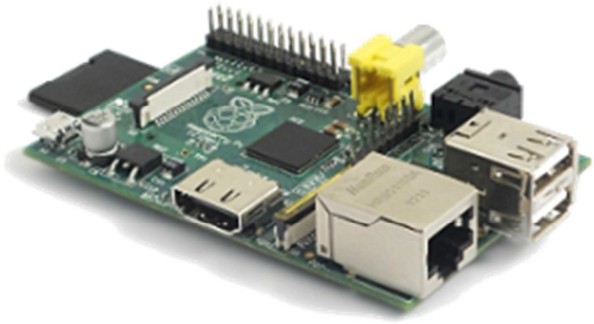


Figura 4: Placa Rasperry Pi

2.4.2 Arduino

Se trata de una plataforma electrónica "open-source" (código abierto). Es un microcontrolador que también puede ser utilizado como interfaz entre un ordenador u otro dispositivo y el mundo físico, mediante la utilización de sensores y actuadores

Figura 5: Placas Familia Arduino

2.4.3 Comparativa y selección

Dada la naturaleza del proyecto, se ha valorado favorablemente la simplicidad que ofrece Arduino a la hora de realizar proyectos hardware frente a la mayor capacidad de computación que ofrece Raspberry Pi, y que para el desarrollo del proyecto no será necesaria.

La placa Arduino funciona como interfaz entre la aplicación y la regleta, recibiendo las órdenes vía Wi-Fi y realizando su ejecución para controlar los enchufes de la regleta. De todas las placas que se encuentras actualmente en el mercado la más idónea es Arduino Yun, valorándose de ella de forma muy positiva los siguientes aspectos.

- **Capacidades de conexión vía Wi-Fi:** Este tipo de sistemas Arduino hace uso de la distribución Linux Linio, específica para Arduino, lo que facilita la interacción de Arduino con servicios web complejos.

- **Microcontrolador Arduino:** Cuenta con un sistema Leonardo clásico (basado en el procesador Atmega32U4) y cuenta con diferentes salidas a las que conectar sensores y actuadores de forma clara y sencilla.

- **Comunicación:** Arduino Yun contiene librerías para comunicar los sketchs de arduino y sus entradas y salidas con todos los recursos del sistema operativo Linux.

- **Uso de shields:** El diseño de Arduino Yun mantiene los pines de E/S exactamente igual a los de Arduino Leonardo, por tanto es compatible con la mayoría de shields diseñados para Arduino hasta la fecha, incluyendo el shield (Tarjeta de Relés) que contiene los relés que utilizaremos para controlar los enchufes.

Como consecuencia de lo anterior seleccionamos la placa ARDUINO YUN para este proyecto.

Figura 6: Arduino YUN

3 Diseño

3.1 Propósito del Sistema

El objetivo de este proyecto es desarrollar un sistema básico que aproveche las posibilidades que ofrecen los terminales móviles actuales para el control remoto de equipos encendiendo y apagando a voluntad del usuario cualquier sistema conectado a la red eléctrica.

3.2 Requisitos del Sistema

El sistema consta de varias partes (subsistemas) bien diferenciadas que interactúan entre sí de forma coordinada para llevar a cabo las acciones que el usuario solicite. Cada uno de estos subsistemas puede clasificarse dentro de dos grandes grupos.

- **Lado remoto:** Se encuentra físicamente lo que llamamos regleta inteligente. Se encarga de recibir las órdenes del usuario vía WiFI, interpretarlas y ejecutarlas encendiendo y apagando físicamente cada uno de los diferentes enchufes de la regleta.
- **Lado usuario:** Se trata de una aplicación que funcionará sobre un terminal Android con conexión a la red que gestionará tanto las órdenes dadas por parte del usuario como las comunicaciones entre el lado remoto y el lado usuario.

La aplicación ofrece una interfaz sencilla e intuitiva sobre la que el usuario puede interactuar con la regleta para controlar cada uno de sus enchufes a voluntad y de forma independiente. Todas las órdenes dadas por el usuario se ejecutarán en tiempo real informando además por pantalla de los resultados de la acción.

3.3 Diseño de las aplicaciones Android

En este apartado tratamos tanto de la aplicación a desarrollar sobre Android.

3.3.1 Requisitos de la aplicación Android

En esta sección se explican las características y las funcionalidades que se ha desarrollado para en la aplicación Android. Se ha buscado ante todo la comodidad del usuario mediante un esquema de funcionamiento simple e intuitivo.

3.3.1 Idioma

A la hora de desarrollas la aplicación se ha buscado un correcto funcionamiento libre de errores y una correcta gestión de las órdenes del usuario por encima de criterios estéticos y comerciales.

El idioma en el que ha sido escrito el contenido de la aplicación ha sido el español. Sin embargo el funcionamiento intuitivo de la misma hace que no sea necesario dominar el

idioma para manejar la interfaz. Por supuesto no se descarta añadir el contenido en nuevos idiomas (especialmente el inglés) en futuras actualizaciones.

3.3.2 Versiones de Android

En primer lugar, antes de diseñar la aplicación Android se ha realizado un estudio para decidir cuál será el número de versiones que la aplicación va a soportar.

Es un tema de vital importancia, debido a que cuanta más antigua es la versión sobre la que se desarrolla la aplicación mayor será el número de dispositivos que podrán utilizarla, pero esto será a costa de no disponer de las funciones y características más actuales.

Dada la naturaleza del proyecto muchas de esas características no eran realmente necesarias por lo que se adoptado como criterio que la aplicación pudiera ser accesible a la mayor cantidad posible de los dispositivos Android actuales. La decisión final se ha tomado después de analizar los datos publicados por Android.

Según Android la distribución de versiones con datos tomados en un periodo de siete días que terminó el 7 de Diciembre de 2015 es la reflejada en la tabla y diagrama siguientes, en los que no se ha tenido en cuenta las distribuciones menores de 0.1%.

VERSION	CODENAME	API	DISTRIBUTION
2.2	Froyo	8	0.2%
2.3.3-2.3.7	Gingerbread	10	3.4%
4.0.3 -4.0.4	Ice Cream Sandwich	15	2.9%
4.1.x	Jelly Bean	16	10.0%
4.2.x		17	13.0%
4.3		18	3.9%
4.4	KitKat	19	36.6%
5.0	Lollipop	21	16.3%
5.1		22	13.2%
6.0	Marshmallow	23	0.5%

Tabla 1: Distribución de Versiones Android a 7 de Diembre de 2015

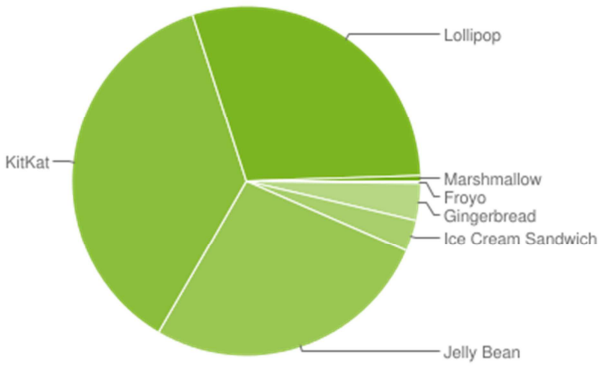


Figura 7: Distribución versiones ANDROID

Teniendo en cuenta los datos recopilados y el objetivo mencionado de que la aplicación sea accesible a la mayor parte de dispositivos Android que operan actualmente se ha optado por API 15: Android 4.0.3 (IceCreamSandwich) que nos otorgan un margen muy satisfactorio al conseguir que la aplicación funcione en aproximadamente el 87.9% de los dispositivos que están activos en estos momentos en la Google Play.

El resto de usuarios lo compone un 12.1% un porcentaje muy pequeño que irá disminuyendo poco a poco con las sucesivas actualizaciones de Android y la venta de nuevos dispositivo. Por tanto el 87.9% antes mencionado es un porcentaje lo suficientemente alto para cumplir con los objetivos marcados para la aplicación al conseguir ser utilizable por la amplia mayoría de usuarios de teléfonos Android.

3.3.3 Módulos de la aplicación en Arduino

A continuación muestra el diseño de todos los módulos de la aplicación, se explica además la función de cada uno de ellos y la experiencia de usuario, remarcando los principales retos encontrados y las soluciones aportadas.

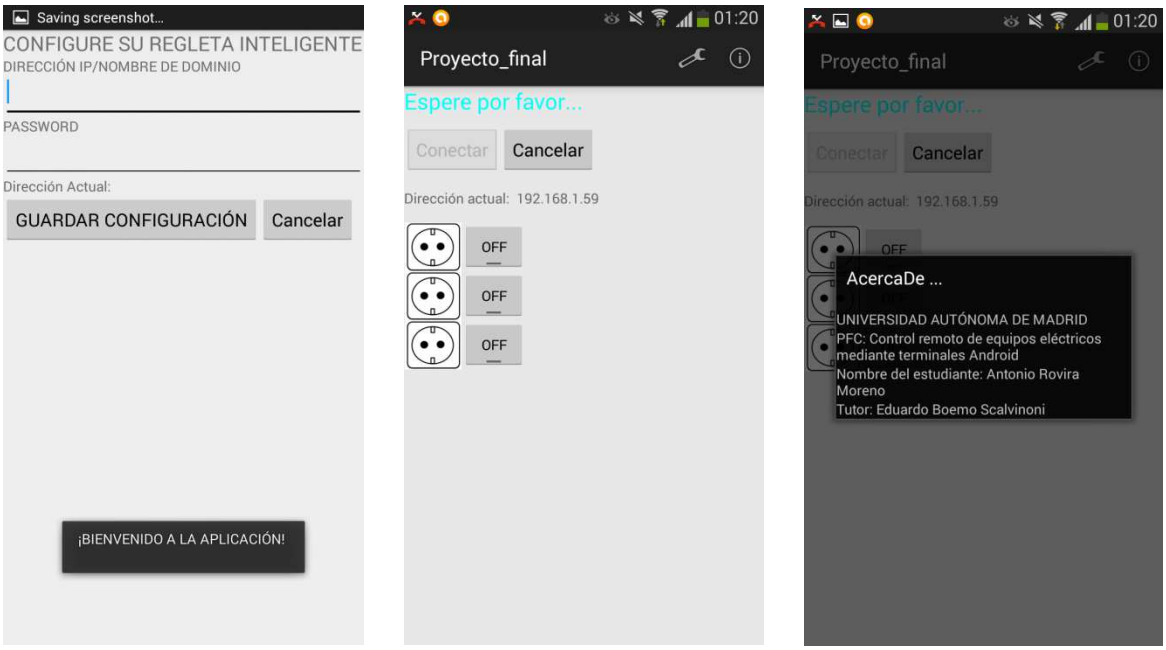


Figura 8: Pantallas de la Aplicaición

Para ello es necesario entender que la interacción del usuario con la aplicación se lleva a cabo a través de tres pantallas que sirven de interfaz para gestionar las órdenes dadas por el propio usuario.

- **Pantalla de Control:** Se trata de la pantalla principal, en ella se gestiona la conexión a internet de la regleta y se dan la ordenes deseadas a cada uno de los tres enchufes de forma independiente

- **Pantalla de Configuración:** En esta ventana se inscribe el formulario necesario para el funcionamiento del sistema en el que el usuario introducirá los siguientes datos:

    - La dirección IP asignada a la regleta inteligente.

    - La contraseña mediante la cual en futuras ampliaciones podrá servir para cifrar las comunicaciones.

- **Pantalla de Información:** Se trata de un pequeño cuadro de información aportando datos acerca del desarrollo de la aplicación.

La ventana de información es muy sencilla y únicamente contiene texto, sin embargo las pantallas de **control** y **configuración** tienen un diseño más complejo que será abordado en profundidad junto a la **experiencia de usuario** que ofrecen. Esto se realizará a lo largo de los siguientes apartados.

3.3.3.1 Experiencia de usuario

Se define la experiencia de usuario como el conjunto de factores y elementos relativos a la interacción del usuario con la aplicación. Para realizar el análisis se detallan todas las etapas que involucran la funcionalidad de la aplicación.

- **Bienvenida:** La primera vez que el usuario ejecuta la aplicación, esta le recibe con un mensaje de tipo toast dándole la bienvenida y mostrando la pantalla de configuración. La aplicación necesita saber cuál es la dirección IP asignada a la regleta para poder conectarse a ella por lo que el usuario permanecerá en la ventana de configuración hasta que rellene el formulario.

- **Gestión:** Una vez rellenado el formulario esta será la ventana que recibirá al usuario cada vez que vuelva a ejecutar la aplicación. Desde esta ventana el usuario puede realizar la conexión con la regleta, dar órdenes a cada uno de los enchufes, o incluso acceder al menú superior para cambiar la configuración mediante el formulario.

Una vez han quedado definidas cada una de las etapas que encontrará el usuario en el manejo de la aplicación, se continuará analizando en los siguientes apartados el diseño de cada una de las ventanas.

3.3.3.2 Pantalla de Configuración.

Es la ventana que recibe al usuario la primera vez que ejecuta la aplicación.

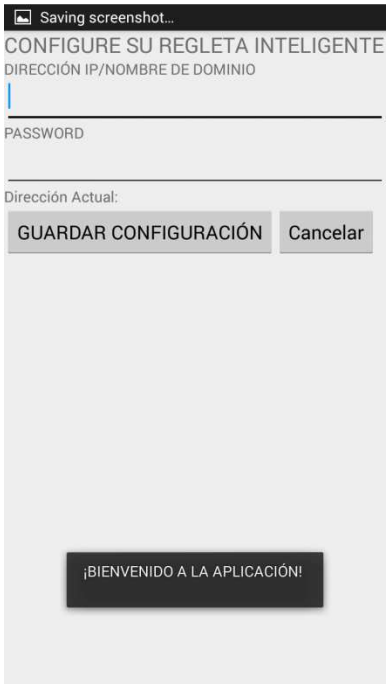


Figura 9: Pantalla de Configuración

El formulario consta de dos campos “IP” y “PASSWORD”, un campo con la dirección que actualmente se tiene almacenada en memoria y dos botones, “Guardar Configuración” y “Cancelar”. A continuación se pasa a explicar en detalle cada uno de estos elementos.

- **DIRECCIÓN IP/ NOMBRE DE DOMINIO:** Almacena en la memoria interna del teléfono la dirección IP asignada a la regleta para poder conectarse a ella y enviar las órdenes. La conexión puede realizarse desde la misma red WiFi utilizando la IP privada, o desde el exterior, mediante una IP pública. En caso de haber asociado la IP a un nombre (por ejemplo mediante los protocolos DNS) el formulario también permite almacenar el nombre asociado.

- **PASSWORD:** Almacena en la memoria interna del teléfono la contraseña mediante la cual se cifrarán las comunicaciones entre la aplicación y la regleta. Este objetivo forma parte de futuras actualizaciones del sistema, a pesar de ello ya se incluye la opción de almacenar dicha contraseña.

- **Campo “Dirección Actual”:** Tiene una función meramente informativa, simplemente lee de memoria interna cuál es el valor IP almacenado y se lo muestra al usuario para que en todo momento sepa a qué dirección está tratando de conectar.

- **Botón "Guardar Configuración":** Una vez pulsado se encarga de comprobar si los datos introducidos por el usuario son válidos, y en caso afirmativo los almacena en la memoria interna del teléfono. Es importante resaltar que la comprobación es muy exhaustiva debido a la gran importancia de los datos introducidos a la hora de realizar la conexión. Sobre todo esto se profundiza en los siguientes apartados.

- **Botón "Cancelar":** Como su propio nombre indica permite al usuario abandonar la ventana de configuración si guardar en memoria los datos introducidos en el formulario.

Hasta ahora se ha explicado en detalle el diseño de la ventana de **configuración** y el funcionamiento de cada uno de sus apartados. Al realizar el diseño se ha buscado una buena gestión de errores y experiencia de usuario. Todo esto se explica en el siguiente apartado.

3.3.3.2.1 Gestión de errores en la ventana de configuración.

Los datos introducidos en el formulario son una pieza clave para realizar la conexión con la regleta. Estos datos se almacenan en la memoria interna del teléfono y se utilizan para realizar peticiones mediante HTTP al servidor web capaz de ofrecer servicios REST configurado en el lado Linio de la placa Arduino Yun que sirve para gestionarla regleta.

La aplicación Android se encarga de realzar peticiones GET mediante HTTP para conectar con la regleta. Para que pueda realizar con éxito es preciso que los datos almacenados en la dirección IP tengan el formato adecuado, puesto que ciertos caracteres (como la 'ñ' o ciertos símbolos) al realizar las pruebas se comprobó que generaban muchos problemas.

En la práctica esto se ha corregido permitiendo que únicamente puedan almacenarse aquellos caracteres que sean verdaderamente necesarios para nuestra aplicación. A continuación se ofrece una lista de todos los caracteres permitidos y las razones que han llevado a tomar esa decisión.

- **Números del cero al nueve (0-9) y el punto '.'** Se tomado esta decisión debido a que las direcciones IPv4 se expresan mediante un número de 32 bits divididos en cuatro octetos. En la expresión de las direcciones IPv4 en decimal se separa cada octeto (comprendido entre 0 y 255) por un carácter único '.'

- **Letras minúscula de la 'a' a la 'z' excluyendo la 'ñ'** En caso de asociar la IP con un nombre de dominio se ha de tener en cuenta que los nombres de dominio normalmente constante de dos o más partes (llamadas etiquetas) separadas por puntos '.' Cada una de las etiquetas estará escrito en minúscula excluyendo la 'ñ'

Respetar la nomenclatura es esencial para poder realizar las peticiones HTTP de la forma correcta. Los módulos programados implementan la solución de forma óptima, avisando al usuario cuando está tratando de guardar un carácter no válido e impidiendo el almacenamiento de los datos en la memoria del teléfono hasta que estos sean los adecuados.

3.3.3.3 Pantalla de Control

Se trata de la ventana desde la que el usuario podrá realizar la conexión con la regleta y dar órdenes de forma independiente a cada uno de los enchufes.

Figura 10: Pantalla de Control

Como puede verse en la imagen, consta de una ventana de información indicando el estado de conexión y varios botones para dar las órdenes. Para una mejor gestión de errores cada vez que se solicita una conexión con la regleta se procede a comprobar el estado de cada uno de los enchufes de forma independiente. A continuación se explica cada uno de los elementos en detalle:

- **Ventana de información**: Indica el estado general de la conexión entre la aplicación y la regleta. Se comprueba cada vez que se envía alguna orden o se realiza la conexión completa. Estos son los estados posibles.
    - **Conexión en curso**: Es el estado de conexión que se activa cada vez que se solicita una orden o se trata de iniciar la conexión. En ese momento la acción solicitada aún está ejecutándose y hasta que acabe no podremos conocer el resultado. Esto es indicado por la aplicación mostrando el mensaje “Espere por favor…”
    - **Conexión realizada con éxito**: Se ha solicitado la conexión con la regleta y se ha realizado de forma exitosa. Se informa al usuario del resultado de la operación y se le permite comenzar a enviar órdenes.

- **Conexión parcial**: Se ha solicitado la conexión con la regleta pero únicamente se ha podido comprobar el estado de algunos de los enchufes (Podría ocurrir de forma excepcional por problemas de conexión). En tal caso se informa al usuario por pantalla y le permite actuar sobre aquellos enchufes en los que la conexión ha sido óptima.

- **Conexión abortada**: Se ha solicitado la conexión pero esta no ha llegado a producirse. Esto puede ocurrir por falta de conexión a internet, porque la dirección IP guardada en el formulario de la aplicación no coincide con la dirección de la regleta o porque el propio usuario ha decidido cancelar la conexión antes de que pudiera llevarse a cabo. Como en todos los demás casos también se informa al usuario por pantalla.

- **Botones de la barra Superior**: Se trata de los botones "Conectar" y "Cancelar" desde los cuales el usuario puede iniciar o parar el proceso de conexión en el momento que vea oportuno.

    - **Conectar:** Inicia el proceso de conexión de la regleta comprobando el estado de cada enchufe en orden y de forma individual. Para favorecer la experiencia de usuario sólo se puede enviar una orden de este tipo cada vez aunque puede ser cancelada en cualquier momento.

    - **Cancelar:** Este botón le da al usuario la capacidad de cancelar el proceso de conexión en cualquier momento. Únicamente es visible cuando el proceso se está llevando a cabo y su ejecución es inmediata.

- **Campo "Dirección actual":** Este elemento ya había aparecido previamente en la ventana de configuración. Se encarga de mostrar al usuario la dirección almacenada en memoria a la que se está tratando de conectar. Puede ser modificada en cualquier momento desde la ventana de configuración.

- **Botones de control de los enchufes:** Son los botones mediante los cuales el usuario puede dar órdenes de encendido o apagada a cada uno de los enchufes de forma independiente.

Además, es preciso indicar que una vez que el formulario ha sido rellenado, cada vez que el usuario ejecute la aplicación, esta le recibirá con la ventana de control (y no con la de configuración como ocurría la primera vez) y para mayor comodidad del usuario se ejecutará de forma inmediata la conexión con la regleta.

Una vez que se ha explicado en detalle el diseño de la ventana de **control** y el funcionamiento de cada uno de sus elementos se procederá a mostrar cómo interactúan todos esos elementos entre sí y cómo es el diseño adoptado con el que se ha conseguido una buena gestión de errores y experiencia de usuario. Todo esto se explica en el siguiente apartado.

3.3.3.3.1 Gestión de errores y experiencia de usuario en la ventana de control.

Todo el funcionamiento de la ventana de control gira alrededor de las peticiones HTTP que se realizan para establecer la conexión con la regleta y gestionar cada uno de los enchufes de forma independiente, por tanto gestionar esas peticiones ha sido fundamental para ofrecer un servicio que sea a la vez eficiente y cómodo al usuario.

Para gestionar de forma adecuada las peticiones HTTP hay que tener claro que bajo ciertas circunstancias, para la aplicación puede suponer una gran cantidad de tiempo y recursos. Para solucionar estos problemas se han tomado acciones diseñadas a evitar tanto la sobrecarga del sistema como también a dar en todo momento un control absoluto al usuario sobre la aplicación. A continuación se muestran los principales problemas encontrados y las soluciones que se han aplicado.

- **Evitar que las peticiones HTTP obliguen a la aplicación a permanecer en estado de espera:** Cada aplicación en Android se ejecuta sobre el hilo principal, también llamado "hilo de interfaz de usuario". Para que el usuario no pierda el control de la aplicación es fundamental que este hilo no quede bloqueado por una gran cantidad de trabajo, es por esto que el código ha sido diseñado para que las peticiones HTTP se realicen en hilos secundarios.

- **Evitar la sobrecarga de peticiones HTTP:** En una aplicación de estas características en casos de sobrecarga el usuario puede perder temporalmente el control de la aplicación. Esto ocurre cuando por error se envía la misma orden de forma repetida sin dar tiempo a que pueda ejecutarse (como podría ocurrir al pulsar repetidamente el botón de conexión o el botón que controla cada uno de los enchufes sin dar tiempo a finalizar ninguna de las acciones). Para solucionar esto se ha establecido un "bloqueo de seguridad" que impide ejecutar nuevas órdenes hasta que estas hayan sido finalizadas. De esta manera y junto a la opción de cancelar que será explicada a continuación el usuario nunca pierde el control de la aplicación.

- **Evitar que el usuario pierda el control:** Como se ha comentado anteriormente bajo ciertas circunstancias una petición HTTP puede llegar a durar mucho tiempo, más incluso del que el usuario desearía esperar. Para esos casos se ha incluido el botón cancelar, que aparece cada vez que se está gestionando una petición HTTP, y que permite al usuario terminar su ejecución en cualquier momento y además de forma inmediata.

- **Evitar que problemas leves en la conexión impidan el uso de la regleta:** El sistema que se ha diseñado permite al usuario controlar cada uno de los enchufes de forma independiente. En caso de que por problemas de conexión esta se lleve a cabo de forma parcial con algunos enchufes el usuario podrá seguir controlándolos de forma independiente.

La gestión de las peticiones HTTP ha sido clave para ofrecer una buena experiencia de usuario, ofreciendo una interfaz sencilla e intuitiva con la que gestionar las órdenes de

la aplicación, evitando siempre la sobrecarga del sistema y ofreciendo al usuario en todo momento el control de las tareas.

3.3.3.4 Pantalla de Información.

Comparada con el resto de ventanas de la aplicación esta es la que posee una funcionalidad y apariencia mucho más simple, no en vano su función es meramente informativa, aparece como un cuadro de diálogo emergente por lo que su estética es diferente de las otras ventanas, y además en ella no puede llevarse a cabo ninguna acción nueva sobre el sistema, su interés únicamente radica en que aporta al usuario algunos datos acerca de la versión de la aplicación y del proyecto que se ha llevado a cabo.

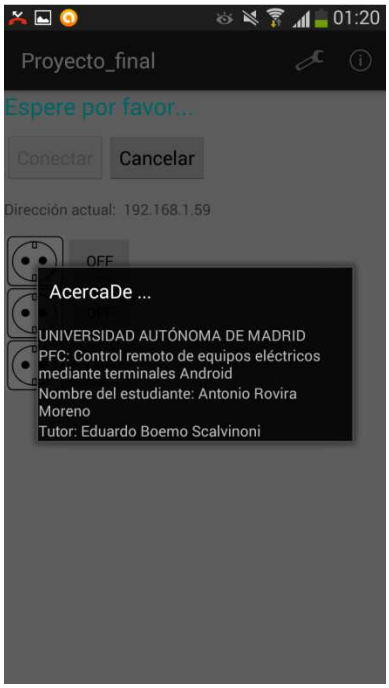


Figura 11: Pantalla de Información

3.4 Diseño sobre Arduino

Como ya se ha explicado, el diseño del proyecto requiere de un lado remoto en el que se encontrará físicamente la regleta inteligente, que a su vez estará formado por un gestor de comunicaciones que se encargará de recibir vía Wifi las órdenes dadas por el usuario y transmitirla a un gestor de órdenes que se encargará de ejecutarlas a través de unos relés que controlan la corriente eléctrica que utiliza cada enchufe.

Tanto el diseño del gestor de órdenes como el diseño del gestor de comunicaciones se han llevado a cabo sobre la misma plataforma hardware, una placa de Arduino Yun que cuenta con tarjeta de conexiones Wifi y Ethernet, contando además en la misma placa con un procesador Linux llamado Linio y el microcontrolador Arduino, de modo que de forma muy sencilla puede combinarse las capacidades avanzas de internet de Linux con la

programación Arduino. Ambos procesadores pueden comunicarse a través de la librería Bridge. Esto ha sido esencial para la realización tanto del gestor de órdenes como del gestor de comunicaciones.

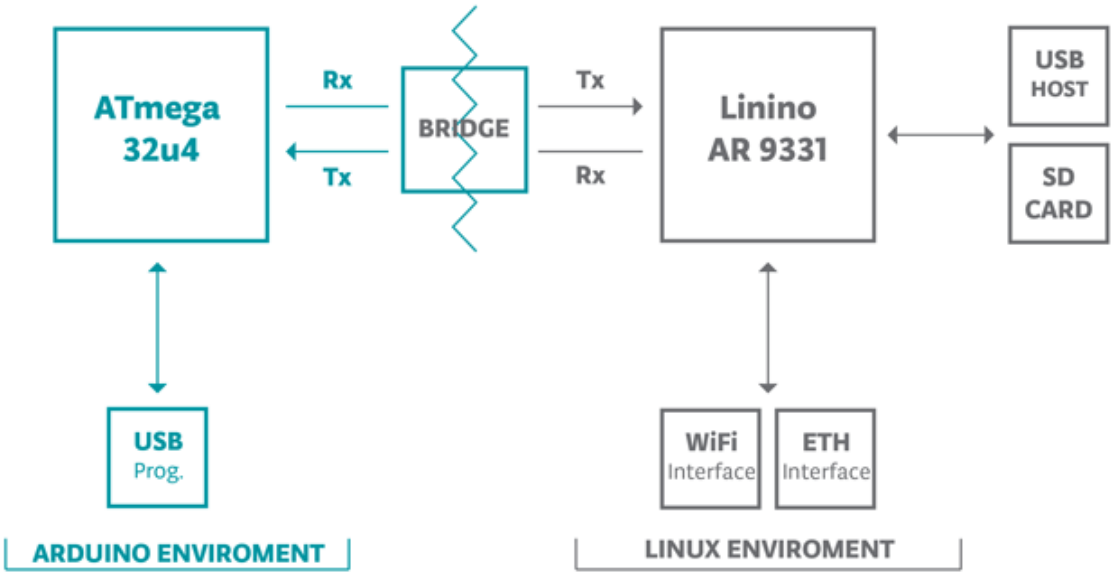


Figura 12: Diagrama de bloques de Arduino YUN

- **Procesador Linux**: Se trata de un procesador que ejecuta una distribución de Linux llamada Linio. Esto ha permitido convertir la placa de Arduino Yun en un servidor web capaz de atender peticiones HTTP realizadas por la aplicación Android en el lado usuario. El gestor de comunicaciones se desarrolla mediante ese servidor web por el que se recibirán las órdenes dadas por el usuario.

- **Microcontrolador Arduino**: Se trata de un microcontrolador Arduino que ejecutará las órdenes dadas por el usuario para el control de la regleta una vez que estas hayan sido recibidas por el servidor web en el lado Linio. El gestor de control se implementa sobre el microcontrolador de modo que cada uno de los enchufes podrá ser gestionado a voluntad por parte del usuario.

- **Librería Bridge**: Se trata de la librería que gestiona la comunicación entre el procesador Linux y el microcontrolador Arduino, facilitando por tanto la interacción de Arduino con servicios web complejos.

Una vez que cada uno de los elementos ha quedado descrito todavía queda definir cómo se gestionará el comportamiento completo de la regleta y la coordinación entre todos los elementos. Esto se realiza mediante una sección de código fuente llamada sketch que hará uso de la librería Bridge para comunicar ambos procesadores y gestionará de manera concreta cómo deben ser interpretadas las órdenes enviadas por el usuario a través de la aplicación.

3.5 Diseño Hardware

En este apartado se describe únicamente el diseño del equipo situado en el lado remoto incluido en el proyecto "CONTROL REMOTO DE EQUIPOS ELECTRICOS MEDIANTE TERMINALES ANDROID" ya que en el lado usuario el equipo estará formado por un terminal móvil Android. Se enumeran los elementos que conformarán este equipo y se definen las características de cada uno de ellos.

El equipo permite el control de tres elementos independientes cada uno de ellos conectado a una base de enchufe en serie con el contacto normalmente abierto NA de un relé controlado mediante un procesador Arduino.

El diagrama siguiente muestra el esquema básico del conjunto del equipo situado en el lado remoto donde se encuentran los elementos a controlar..

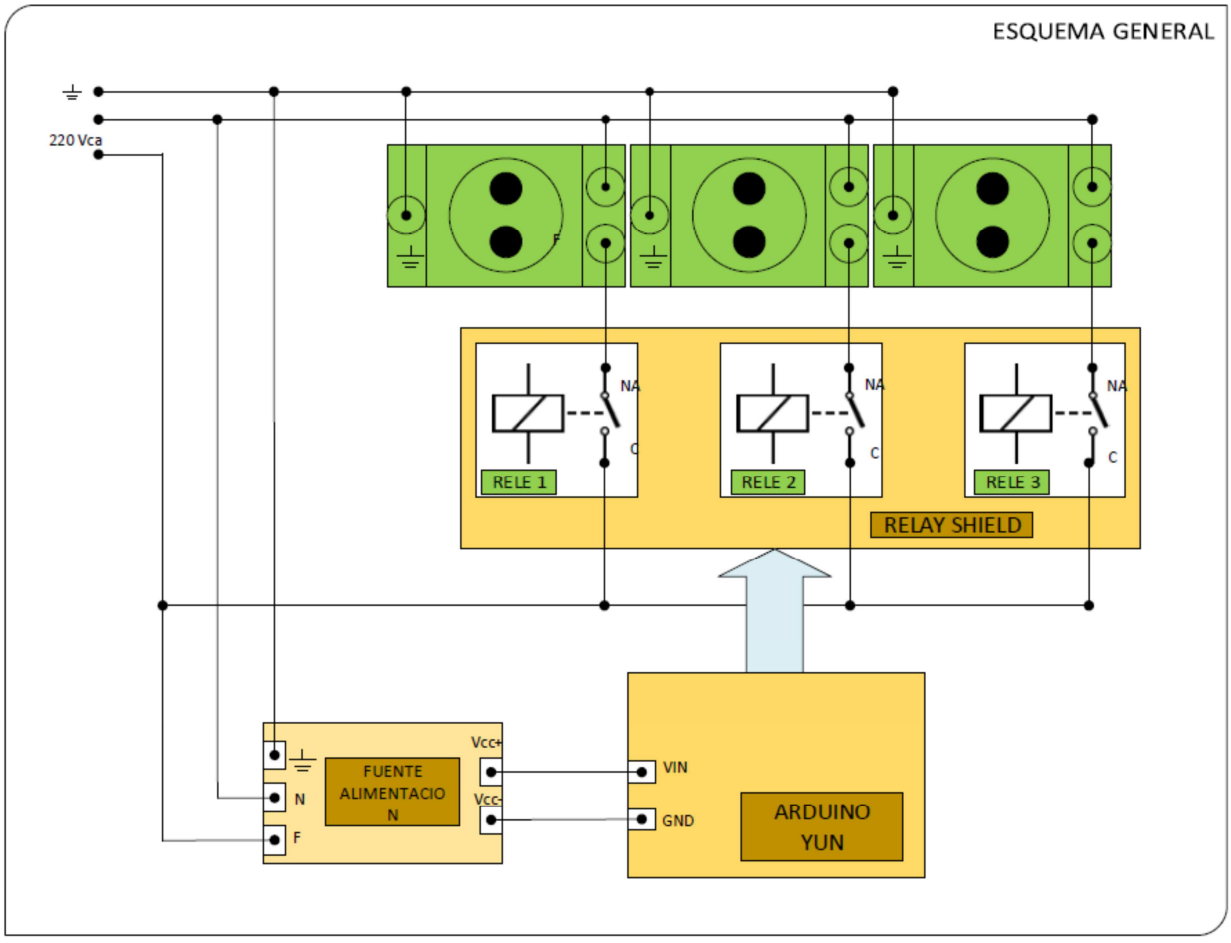


Figura 13: Esquema General Lado Remoto

Como se indica en el esquema, cada uno de los enchufes a controlar tiene intercalado en su alimentación un contacto de relé normalmente abierto NA. El relé que maneja el contacto pertenece a una placa de relés que recibe las órdenes de actuación desde el controlador Arduino.

El controlador Arduino se conecta vía wifi – Internet con el terminal Android que da las ordenes de encendido / apagado.

Este equipo de lado remoto está compuesto desde el punto de vista hardware por los elementos siguientes:

- Caja compacta
- Bases de enchufe
- Arduino YUN
- Placa de relés
- Fuente de alimentación
- Regulador de tensión
- Toma USB
- Toma de alimentación general.

3.5.1 Caja Compacta.

El equipo está inicialmente pensado para controlar equipos doméstico por lo que se presenta en una caja compacta que alberga todos los elementos de control dejando accesible únicamente las tres tomas de enchufe, el cable de alimentación general y una toma USB para la programación y configuración.

Se ha optado por una caja de superficie para montaje de elementos en carril DIN de 8 módulos.

La elección de esta caja con carril DIN facilita el montaje de los elementos limitando el mecanizado de la caja a unos taladros para el paso del cable de alimentación general y un cajeado para el acceso desde el exterior al conector USB.

Se pretende un montaje lo más compacto y reducido posible por lo que se ha ido a una caja de solo ocho módulos espacio suficiente para albergar las tres bases de enchufes. Cada una de las tres bases ocupa 2,5 módulos lo que da una ocupación de 7,5 módulos tapando el medio módulo que sobra con la correspondiente tapa.

Por otro lado es previsible que el equipo atienda actividades en el exterior por lo que debemos dotar a la caja de un cierto nivel protección al menos un IP 30 y una tapa que proteja los enchufes cuando no están en uso.

3.5.2 Bases de Enchufe

Para montar los enchufes en la caja seleccionada se prevé el uso de bases de enchufe para carril DIN con toma de tierra y que admitan una intensidad de trabajo de 16 A.

3.5.3 Arduino YUN

Como se ha indicado en el apartado de **Comparativa y Selección** de equipo de control, para este proyecto se ha seleccionado el Arduino YUN .

A diferencia de otros modelos de Arduino este no dispone de regulador de 5VCC interno por lo que el fabricante recomienda alimentarlo vía el conector micro-USB o mediante el pin Vin utilizando en este caso un regulador externo de 5Vcc para evitar que una sobretensión accidental dañe la placa.

En nuestro caso necesitamos tener libre la entrada micro-USB para posibles reprogramaciones o configuraciones del equipo por lo que alimentamos a través de la entrada Vin instalando un regulador de 5Vcc a su entrada.

3.5.4 Placa de Relés

Para controlar el paso de corriente a cada uno de las bases de enchufe utilizamos una placa de relés. Una ventaja del sistema Arduino es que dispone de shield´s, placas de extensión que le dotan de características externas específicas de las que no dispone inicialmente.

Estos shield´s se adaptan a la placa Arduino tanto geométrica como eléctricamente utilizando para ello los pines de entrada /salida del Arduino sin necesidad de cableado adicional. La placa shield se "enchufa" en los pines del Arduino haciéndolos accesibles desde la propia placa shield. El shield se alimenta desde los pines del Arduino y a través de ellos se comunica con él.

En el proyecto utilizamos un shield de 4 relés de los que inicialmente solo utilizaremos tres.

Las características principales del shield de relés son:

- Compatible con Ardudino YUN al que se conecta a través de los pines de las placas.
- Interface vía digital a través de los pines I/O 4,5,6, y 7
- Conexiones a los relés mediante tornillos.
- LED indicadores de estado para cada rele.
- Relés de gran calidad
- Por cada relé pines de COM, NO (Normalmente Abierto), and NC (Normalmente Cerrado)

Figura 14: Placa Shield de Relés

Ésta placa permite controlar mediante su conexión a un Arduino cuatro relés para conmutar cargas externas tales como bombillas, motores etc . Los relés pueden conmutar hasta 10 A a 30Vcc o 10A a 250 Vca.

Dispone de un film transparente aislante por debajo que protege la placa Arduino de cualquier contacto no deseado cuando se conecta a ella

3.5.5 Alimentación

Dado el consumo de los relés, el fabricante recomienda disponer de una alimentación externa a 5 Vcc para alimentar la placa dado que el puerto USB no proporciona suficiente corriente.

Por otra parte para asegurar que no se suministran más de 5Vcc, lo que dañaría los circuitos del Arduino ponemos a su salida un circuito regulador que limita la salida a 5Vcc.

3.5.5.1 Regulador de tensión a 5Vcc

Para asegurar la tensión de salida a 5Vcc utilizamos el circuito regulador de tensión LM7805

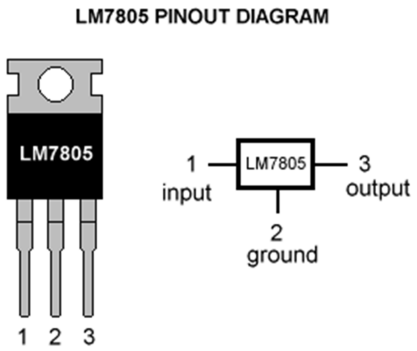


Figura 15: Regulador de Tensión a 5Vcc

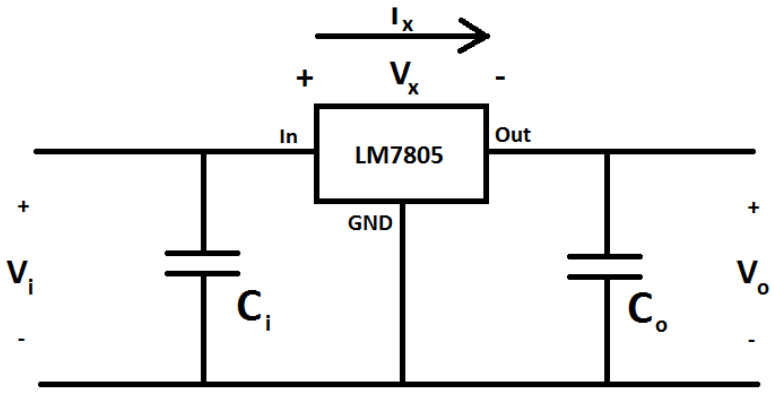


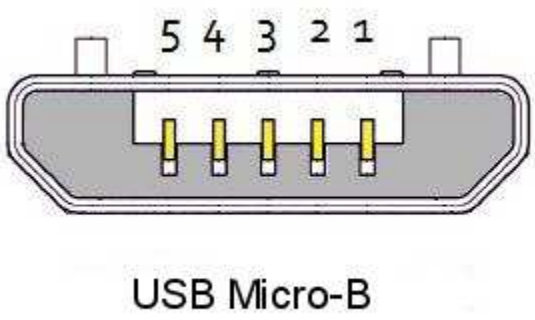
Figura 16: Montaje Regulador de Tensión

Los condensadores electrolíticos C1 y C2 son necesarios para un buen filtrado de la tensión. Siguiendo la recomendación del fabricante utilizamos condensadores de 100 microfaradios

3.5.6 Conector USB

Para la puesta en marcha del equipo y posibles reprogramaciones del mismo se necesitará acceder al conector USB del Arduino YUN. Como el equipo se presenta en modo compacto se proyecta un conector USB accesible desde el exterior para lo que se mecanizará la caja con un cajeado y se dispondrá de un conector USB encajado en el cajeado y conectado internamente al micro USB del Arduino YUN.

Cabe la posibilidad de que el equipo se encuentre alimentado a través del cable general de alimentación cuando se conecte el cable USB. Para evitar posibles problemas en la alimentación se eliminan del conector USB las conexiones de alimentación.



Pin	Name	Cable color	Description
1	VCC	Red	+5 VDC
2	D-	White	Data -
3	D+	Green	Data +
4	ID	n/a	USB OTG ID
5	GND	Black	Ground

Figura 17: PinOut MicroUSB

A la vista del diagrama anterior solo se llevará al conector USB exterior los pines 2 y 3.

4 Desarrollo

En este apartado se van a detallar como están desarrolladas cada una de las partes del proyecto. Explicando cada una de las etapas en profundidad, con referencias al código y analizando los principales retos que han ido surgiendo durante el desarrollo del mismo.

El proyecto desarrolla un sistema básico que permite el control remoto de equipos, encendiendo y apagando a voluntad del usuario cualquier sistema conectado a la red de forma remota.

El sistema consta de varias partes (subsistemas) bien diferenciadas, por lo que todas las etapas de desarrollo del proyecto perteneces a uno de estos dos grandes grupos.

- **Lado remoto:** Su desarrollo lo definen todas las etapas para construir lo que llamamos "regleta inteligente".
    - Selección de todos los componentes necesarios, destacando de entre ellos plataforma hardware utilizada.
    - Montaje físico de la regleta inteligente.
- **Lado Usuario:** Su desarrollo lo definen el diseño y la implementación de la aplicación Android cuya función será la siguiente
    - Gestionar las órdenes dadas por el usuario
    - Gestionar la comunicación entre la aplicación y la regleta inteligente.

4.1 Desarrollo de la Aplicación Android

En este apartado se va a comentar el desarrollo de cada una de las partes de la aplicación, entrando en detalles y con referencias al código. Se ha puesto especial atención en explicar la comunicación a través de red entre la aplicación y la regleta por ser este el pilar fundamental en el que se basa la aplicación

4.1.1 Herramientas utilizadas:

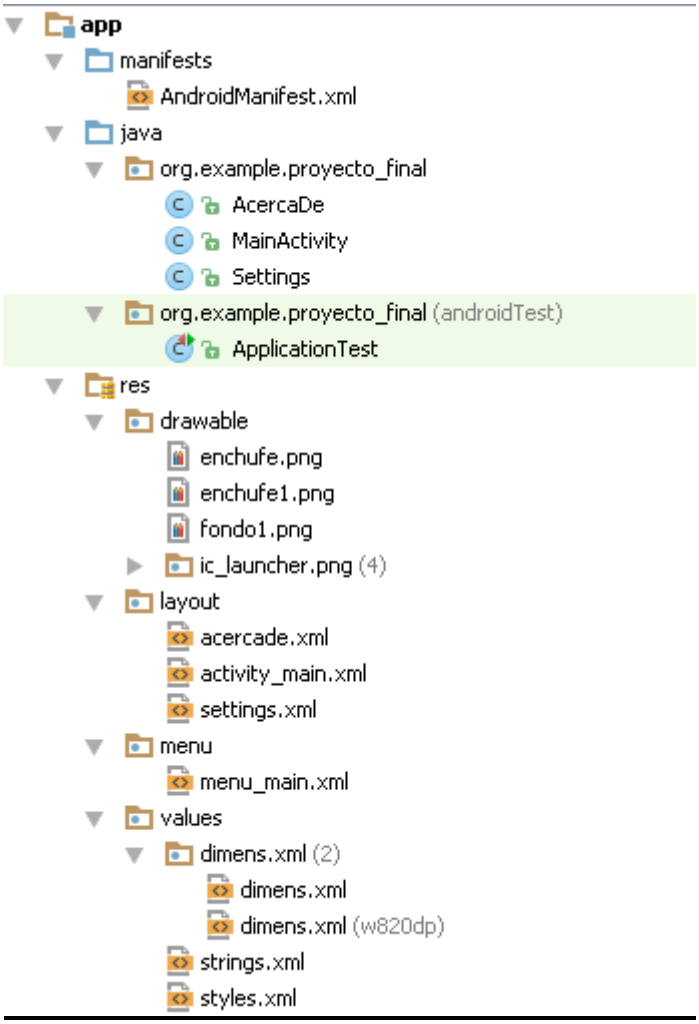
Aunque el IDE por excelencia para el desarrollo de aplicaciones en Android había sido siempre Eclipse, este comenzó en Mayo de 2013 Android Studio fuera presentado por Google como la alternativa oficial.

Android Studio es más potente, versátil y ha conseguido desplazar a Eclipse como el IDE Oficial de Android. Es por este motivo que Android Studio ha sido el IDE elegido para la realización de la aplicación.

En concreto, para la realización de la aplicación han sido necesarios lo siguientes elementos:

- Ordenador personal
- Android Studio
- Java SDK
- Dispositivo móvil virtual creado con el SDK
- Dispositivo móvil con Android

4.1.2 Estructura de carpetas de la aplicación



Todo proyecto Android está organizado en diferentes carpetas, almacenando un descriptor de la aplicación (AndroidManifest.xml), el código fuente en Java y una serie de ficheros con recursos. Tenemos tres elementos principales:

- **AndroidManifest.xml:** Es el descriptor de la aplicación, contiene las características generales de este y sus componentes. En él se indican las actividades, las intenciones, los servicios, los permisos y los proveedores de contenidos de la aplicación. Ha sido modificado para conseguir
    - Permiso para la utilización de la red.
    - Configuración para permitir la rotación de pantalla.
- **Java:** Carpeta que contiene el código fuente de la aplicación. Se organiza en paquetes de igual manera que las aplicaciones Java. Para dirigir el funcionamiento de la aplicación se han creado tres clases diferentes
    - **MainActivity:** Contiene la mayor parte del código y lleva el peso de la aplicación; comprueba la conexión a internet, realiza la conexión con la regleta y ejecuta las órdenes que el usuario solicita a través de los diferentes botones de la interfaz.
    - **Settings:** Contiene el código que gestiona la configuración de la regleta. Se encarga de almacenar el password y la dirección IP de la regleta o el nombre de dominio (en caso de utilizar un servidor DNS) en un fichero desde el que puede recuperarlas cada vez que sea necesario.
    - **AcercaDe:** Muestra información sobre el proyecto y la universidad.
- **Res:** Carpeta que contiene los recursos utilizados para la aplicación. Contiene una serie de archivos en formato XML con datos referentes a los recursos utilizados en la aplicación. Se han utilizado los siguientes
    - **Drawable:** Almacena los ficheros de imágenes (JPG o PNG) utilizados tanto para el fondo como para la representación de la regleta.
    - **Layout:** Almacena las vistas de la aplicación en forma de ficheros XML, cada uno relacionado con su correspondiente clase Java. Los ficheros de layout utilizados son **activity_main.xml**, **settings.xml** y **acercade.xml**.
    - **Strings.xml:** Almacena todas las cadenas de texto en formato XML de modo que sea sencillo acceder a cada recurso utilizando ficho formato.

4.1.3 Gestión de los datos con Shared Preferences

La aplicación necesita almacenar ciertos datos dados por el usuario puesto que para poder ejecutar sus órdenes de forma correcta la aplicación necesita realizar la conexión con la regleta de una forma eficaz. Son dos tipos de datos distintos:

- **Dirección IP/Nombre de dominio:** Es necesario que la aplicación tenga almacenada la dirección a la que se encuentra conectada la regleta, para que el usuario pueda realizar con éxito las acciones deseadas.

- **Password**: Es la clave mediante la cual el usuario podrá cifrar su conexión con la regleta.

Para almacenar estos tipos de datos se ha optado por las preferencias (clase SharedPreferences) por ser un mecanismo que permite a los usuarios modificar parámetros de configuración de la aplicación.

Las preferencias son almacenadas en un fichero XML en la memoria interna del teléfono, de modo que todas las variables quedan guardadas con su nombre y su valor.

Para obtener una referencia a una colección determina se utilizará el método getSharedPreferences() al que se le pasará un identificador de la colección y un modo de acceso.

El almacenamiento de datos se ha realizado de la siguiente manera:

```
SharedPreferences preferencias= getSharedPreferences("FicheroPreferencias", MODE_PRIVATE);

SharedPreferences.Editor editor = preferencias.edit();
editor.putString("NOM_ARDUINO", nom_arduino.getText().toString());
editor.putString("PASS_ARDUINO", pass_arduino.getText().toString());
```

La carga de datos se ha realizado de la siguiente manera:

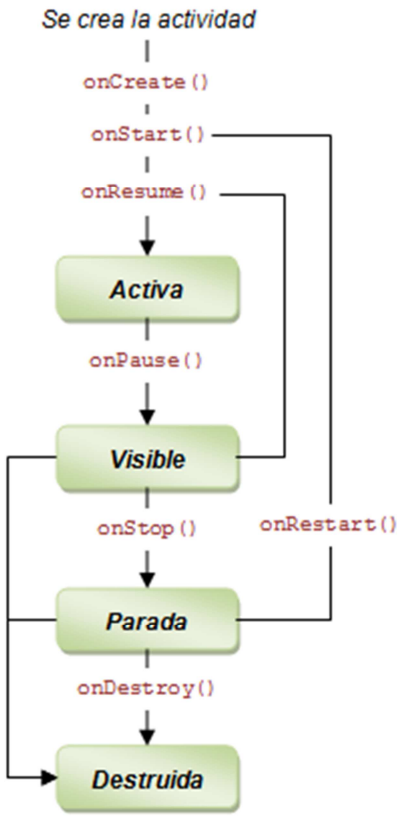
```
SharedPreferences preferencias= getSharedPreferences("FicheroPreferencias", MODE_PRIVATE);

String nombre = preferencias.getString("NOM_ARDUINO","");
```

Para una mayor seguridad de los datos de la aplicación se ha optado por seleccionar el modo de acceso como MODE_PRIVATE de modo que nuestra aplicación será la única de todo el dispositivo que tenga acceso a este fichero.

Aunque todos los datos de la aplicación se almacenan en forma de string, las preferencias también permiten almacenar variables de tipo booleano, real y entero.

El método de las preferencias está diseñado específicamente para administrar los datos de "configuración", y precisamente por ello se ha optado por este método. Además sirven para comunicar diferentes actividades, almacenando los se guardan en un fichero XML para mayor comodidad.

4.1.4 Gestión del ciclo de vida de las actividades



4.1.5 Unificación de los textos

Almacenado en "res/values" se encuentra el fichero "strings.xml" dónde se han almacenado todos los textos que necesita la aplicación (esto incluye las etiquetas de los objetos Button, los textos fijos TextView y todos los controles que muestran un texto fijo en el dispositivo).

Las variables se almacenan con formato etiqueta-valor de la siguiente forma:

```
<string name="ETIQUETA">TEXTO</string>
```

Esta distribución presenta varias ventajas, para empezar permite gestionar todo cambio realizado en los textos de la aplicación desde ese mismo fichero.

Además la agrupación de texto posibilita facilitar la implementación de múltiples idiomas para nuestra aplicación.

Para referencias los textos desde cualquier parte del código se utiliza la clase R mediante el método getString (pertenece a la clase Context) utilizando el identificador que cada uno de los texto posee.

```
getString(R.string.identificador)
```

4.1.6 Gestión de las órdenes dadas por el usuario.

La aplicación instalada en el terminal Android sirve como gestor de órdenes con el que el usuario pueda controlar de forma remota equipos eléctricos a través de internet. La comunicación con la regleta se lleva a cabo mediante peticiones HTTP, que han sido el eje alrededor del que se ha desarrollado la aplicación, poniendo especial énfasis en los siguientes puntos.

- **Permisos**: Son los permisos que son necesarios declarar en AndroidManifest.xml para que la aplicación pueda conectarse a Internet y enviar órdenes a la regleta.

- **Tareas asíncronas**: En Android todas las aplicaciones se ejecutan mediante el hilo principal (también llamado hilo de la interfaz de usuario) una solución que destaca por su sencillez pero que se vuelve ineficiente a la hora de realizar cálculos complejos o conexiones a internet. Para solucionarlo la aplicación ejecutará las órdenes dadas por el usuario en varios hilos secundarios mediante la utilización de la clase AsyncTask.

Para ejecutar las órdenes dadas por el usuario se ha desarrollado un total de seis funciones que se dividirán en dos grupos en función de si se utilizan para establecer la conexión con la regleta o para ejecutar las órdenes del usuario sobre cada uno de los enchufes de forma independiente.

- **Establecer la conexión:** Se trata de tres funciones que se ejecutan de manera consecutiva a partir del momento en el que se pulsa el botón “Conectar”. Cada una de las funciones inicializa el estado de su enchufe correspondiente y recogen información para comunicar al usuario el estado de la conexión. Las funciones utilizadas son las siguientes:

  - **MiTarea_con1:**

    - Inicializa el estado del primer enchufe.

    - Actualiza la barra de órdenes para permitir al usuario cancelar el proceso de conexión en cualquier momento.

  - **MiTarea_con2:**

    - Inicializa el estado del segundo enchufe

  - **MiTarea_con3:**

    - Inicializa el estado del tercer enchufe.

    - Llama a la función mostrar_resultado que se encarga de informar al usuario del estado de la conexión tras el intento realizado.

    - Actualiza la barra de órdenes para permitir al usuario reiniciar la conexión desde el principio si así lo cree necesario.

- **Gestionar los enchufes:** Una vez se ha establecido la conexión con éxito y el estado de los enchufes ha sido inicializado con éxito, el usuario podrá actuar sobre cada uno de ellos de forma independiente para permitir o no permitir el paso de corriente. Las funciones utilizadas son las siguientes
    - **MiTarea1**:
        - Gestiona la orden dada sobre el primer enchufe
        - Comprueba si la orden se ha ejecutado de manera satisfactoria
        - Controla y evita que repetidas pulsaciones por parte del usuario puedan sobrecargar el sistema.
    - **MiTarea2**:
        - Gestiona la orden dada sobre el segundo enchufe
        - Comprueba si la orden se ha ejecutado de manera satisfactoria
        - Controla y evita que repetidas pulsaciones por parte del usuario puedan sobrecargar el sistema.
    - **MiTarea3:**
        - Gestiona la orden dada sobre el tercer enchufe
        - Comprueba si la orden se ha ejecutado de manera satisfactoria
        - Controla y evita que repetidas pulsaciones por parte del usuario puedan sobrecargar el sistema.

Cada uno de estos apartados contiene detalles esenciales para el buen funcionamiento de la aplicación, por lo que serán tratados en profundidad a lo largo de las siguientes secciones de la memoria.

4.1.7 Comunicación con el servidor web de la regleta

La regleta es capaz de recibir y ejecutar las acciones enviadas desde la aplicación Android gracias a un gestor de órdenes diseñado a partir de una plataforma hardware, Arduino Yun.

La comunicación entre la aplicación y Arduino Yun se realiza mediante peticiones REST, es decir, la aplicación Android interactuará con el gestor de órdenes mediante HTTP utilizando las operaciones de este protocolo (GET, POST, PUT, DELETE).

Para llevar a cabo este objetivo necesitamos que nuestra aplicación asuma las características de un cliente HTTP, y para ello se utilizarán las siguientes librerías de APACHE.

```
import org.apache.http.HttpResponse;
import org.apache.http.client.HttpClient;
import org.apache.http.client.methods.HttpGet;
import org.apache.http.impl.client.DefaultHttpClient;
```

Los pasos que se llevarán a cabo serán siempre los mismos

- En primer lugar se creará el cliente HTTP
- En segundo lugar se llevará a cabo la petición GET a la dirección web deseada (almacenada en una variable de tipo String llamada dir_web)
- Por último se utilizará la instrucción out.toString() para almacenar la información obtenida por parte del servidor en una variable de tipo string.

```
HttpClient httpClient = new DefaultHttpClient();
httpGet1 = new HttpGet(dir_web);

        try {

            HttpResponse response = httpClient.execute(httpGet1);
            ByteArrayOutputStream out = new ByteArrayOutputStream();
            response.getEntity().writeTo(out);
            out.close();

            return out.toString();

} catch (Exception e) {
             cancel(true);
        }
```

4.1.8 Cómo leer y cambiar el estado de un enchufe

Mediante la aplicación del Sketch, Arduino Yun es un servidor web capaz de ofrecer servicios REST.

Cuando se recibe una petición se divide la cadena y en función de su contenido se llevaran a cabo diferentes acciones.

Los pines que actúan sobre el relé que controla cada enchufe son

- **Relé 1**: Pin 7
- **Relé 2**: Pin 6
- **Relé 3**: Pin 5

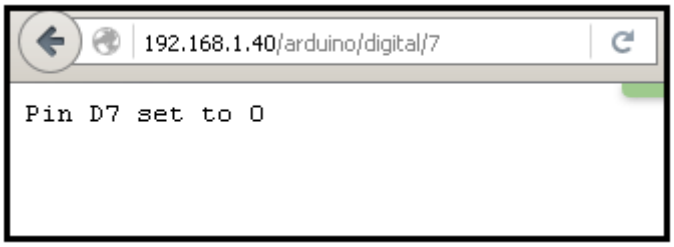
Las peticiones http utilizadas para leer el estado de un enchufe en nuestra aplicación son:

- **Enchufe 1:** http://"+ip+"/arduino/digital/7
- **Enchufe 2:** http://"+ip+"/arduino/digital/6
- **Enchufe 3:** http://"+ip+"/arduino/digital/5

Se trata de una petición GET en la que se recibe el estado del cada pin. Como ejemplo se muestra el código utilizado en la función MiTarea_con1 para leer el estado del pin 7

```
@Override
protected String doInBackground(Integer... n) {
    publishProgress();

     HttpClient httpClient = new DefaultHttpClient();

    SharedPreferences preferencias= getSharedPreferences("FicheroPreferencias", MODE_PRIVATE);
    String ip = preferencias.getString("NOM_ARDUINO","");
    String dir_web = ("http://"+ip+"/arduino/digital/7");
    httpGet = new HttpGet(dir_web);

    HttpResponse response;
```

Como resultado de la petición GET se obtendrá una cadena que podrá ser tratada como un string que nos informará sobre el estado del pin que controla el correspondiente relé (y por tanto controla también el enchufe). Esta información también es accesible realizando una petición GET desde el navegador, tal y como se muestra en el siguiente ejemplo.

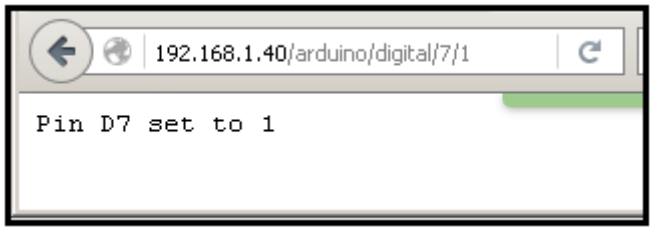


Las peticiones http utilizadas para cambiar el estado de un enchufe en nuestra aplicación son:

- **Enchufe 1:**
  - **Encender:** http://"+ip+"/arduino/digital/7/1
  - **Apagar:** http://"+ip+"/arduino/digital/7/0
- **Enchufe 2:**
  - **Encender:** http://"+ip+"/arduino/digital/6/1
  - **Apagar:** http://"+ip+"/arduino/digital/6/0
- **Enchufe 3:**
  - **Encender:** http://"+ip+"/arduino/digital/5/1

- **Apagar:** http://"+ip+"/arduino/digital/5/0

Al igual que en el caso anterior se trata de una petición GET, con la única diferencia de que al añadir "/0" o añadir "/1" el sketch de Arduino lo gestiona cambiando el estado del pin a voluntad del usuario. Al igual que en el caso anterior se puede realizar la petición desde un navegador. Si se compara el ejemplo del caso anterior puede comprobarse que el resultado es muy similar.



4.1.9 Cancelar órdenes

Las órdenes dadas por el usuario se llevan a cabo mediante peticiones HTTP, por tanto si en algún momento el usuario quiere cancelar una orden dada mientras esta aún no ha terminado de ejecutarse únicamente se ha de terminar con el intento de conexión. Para ello se ha habilitado el botón "Cancelar".

Todas las peticiones HTTP se realizan mediante tareas asíncronas, por lo tanto para cancelar una orden únicamente se ha de cancelar dicha tarea y abortar la petición Get correspondiente.

El usuario puede cualquiera de las órdenes que esté en proceso, esto le permite cancelar tanto la orden de conexión con la regleta como la orden de encendido o apagado sobre cada uno de los enchufes. Todas las instrucciones necesarias para a ello forman parte del código de la función "cancelar_regleta".

- **Cancelar la orden de conexión con la regleta:** La orden de conexión lanza tres tareas asíncronas que pueden ser canceladas directamente mediante la instrucción cancel(true) .y también realiza tres peticiones GET que pueden ser terminadas mediante la instrucción abort()

```
tarea_con1.cancel(true);
httpGet.abort();
tarea_con2.cancel(true);
httpGet_con2.abort();
tarea_con3.cancel(true);
httpGet_con3.abort();
```

- **Cancelar la orden de encendido o apagado sobre cada enchufe:** En este caso se utiliza una instrucción if-else para identificar cuál es el enchufe al que se está enviando la orden. Una vez identificado se procede a su cancelación de la misma manera que en el caso anterior.

```
if (!btn2.isEnabled() && !btn3.isEnabled()) {
    tarea1.cancel(true);
    httpGet1.abort();

} else if (!btn.isEnabled() && !btn3.isEnabled()) {
    tarea2.cancel(true);
    httpGet2.abort();
} else if (!btn.isEnabled() && !btn2.isEnabled()) {
    tarea3.cancel(true);
    httpGet3.abort();

}
```

4.1.10 Permisos necesarios en el dispositivo

Para poder conectar con los equipos eléctricos es necesario hacer uso de internet por lo que ha sido necesario añadir a la aplicación el siguiente permiso

-android.permission.INTERNET

4.1.11 Funcionamiento de (AsincTask)

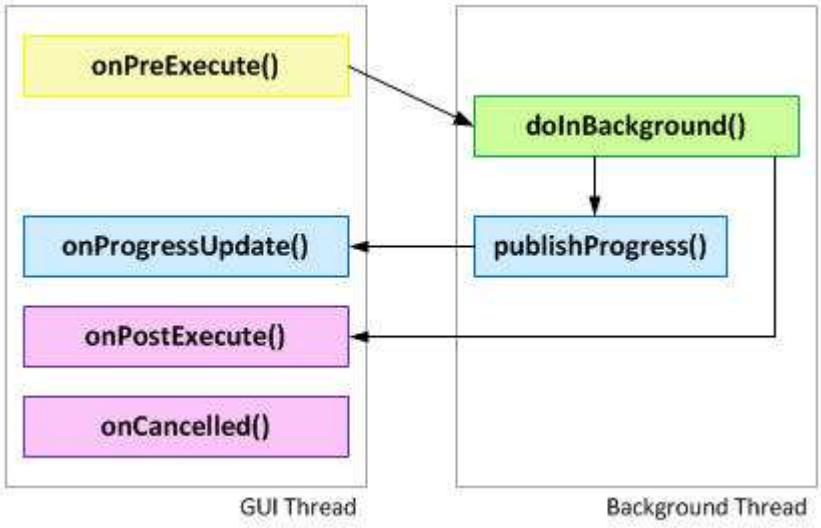
En Android cada vez que se lanza una aplicación el sistema crea un nuevo hilo de ejecución (también llamado hilo principal o hilo de la interfaz de usuario) encargado de ejecutar todas las actividades y servicios de la aplicación.

Esta configuración crea ciertos inconvenientes desde el punto de vista del usuario cuando la aplicación tiene que llevar cabo un trabajo intensivo que pueda dejar bloqueado el hilo principal durante un periodo de tiempo indeterminado dando la impresión de que el sistema se ha colgado.

Dada la naturaleza de nuestro proyecto, de todas las tareas que pueden necesitar ese trabajo intensivo será “el acceso a la red” la más importante de todas ellas, ya que cada orden dada por el usuario mediante la aplicación necesitará ser gestionada a través de la red para poder realizarse el control de los equipos eléctricos de forma efectiva.

El acceso a la red se lleva el mayor peso en el trabajo de la aplicación y por ello como solución se ha optado por la creación de un hilo secundario de modo que el hilo principal pueda seguir atendiendo a los eventos de usuario. Para la creación de ese hilo secundario se ha utilizado la clase AsyncTask.

La clase AsyncTask se utiliza para llevar a cabo una tarea asíncrona, es decir una tarea que se ejecuta en el hilo secundario cuyo resultado se publicará en el hilo de interfaz de usuario. A continuación se muestra el ciclo de vida de la clase AsyncTask.



A la izquierda tenemos el hilo principal de la aplicación y a la derecha el hilo secundario creado por esta clase, que como puede comprobarse como trabajan de forma muy estrecha. A continuación se resume la función de cada método en función del hilo que lo ejecute

- **Hilo principal:**
    - **onPreExecute:** Se ejecuta antes de iniciar el proceso. Sirve para iniciar una barra de progreso que muestre el estado de la tarea o informar de que la tarea está a punto de comenzar.
    - **onProgressUpdate:** Mediante la función publishProgress() desde doInBackground recibe la información del progreso de la tarea. Se suel utilizar para informar al usuario del progreso en tiempo real.
    - **onPostExecute:** Se ejecuta una vez que finaliza el método doInBackground(), recibiendo el resultado para tratarlo y actualizar la interfaz de usuario en consecuencia.
    - **onCancelled:** Se ejecuta una vez que la ejecución ha sido cancelada.
- **Hilo secundario**
    - **doInBackground**: Se ejecuta el cuerpo de la tarea en segundo plano, como puede ser la resolución de cálculos complejos o el acceso a la red.
    - **publishProgress:** Sirve para interactuar con el hilo principal a través del método onProgressUpdate. Puede ser invocado desde doInBackground.

Trasladando este diseño a nuestra aplicación se comprueba como todas las funciones que se realizan en segundo plano tienen una estructura común que se explica a continuación:

- **Hilo principal:**
    - **onPreExecute:** Para ofrecer una mejor experiencia de usuario y evitar sobrecargas únicamente puede estar ejecutándose una tarea asíncrona en cada momento, por lo que los botones de la regleta se bloquean hasta el final de la tarea.
    - **onProgressUpdate:** Informa al usuario de que la conexión a red se está llevando a cabo mostrando un mensaje por pantalla "Espere por favor…"
    - **onPostExecute:** Se liberan los botones para permitir al usuario interactuar con los enchufes de nuevo. Además actualiza el estado del pin correspondiente y muestra el estado de la conexión por pantalla.
    - **onCancelled:** Este método se invoca cuando la conexión se ha cancelado voluntariamente o no ha llegado a producirse Informa al usuario por pantalla. Será necesario reiniciar la conexión para poder enviar órdenes a la aplicación
- **Hilo secundario**
    - **doInBackground**: Se encarga de leer la variable IP de la memoria interna del teléfono y de realizar petición HTTP en segundo plano para interactuar con Arduino.
    - **publishProgress:** Se utiliza para informar al usuario de que la petición HTTP aún no ha finalizado.

4.2 Desarrollo de las aplicaciones Arduino

Una de las ventajas a la hora de programar en Arduino es la facilidad de manipular los pines de E/S con tan sólo unas pocas líneas de código. El IDE (Entorno de Desarrollo Integrado) utilizado ha sido Arduino IDE 1.5.8 Beta, directamente descargado da la web oficial de Arduino.

La configuración de las comunicaciones entre la placa y el PC se realiza a través del menú del IDE de Arduino de una forma rápida e intuitiva, el paso siguiente implica utilizar las herramientas proporcionadas en el IDE para el desarrollo del programa.

Denominamos Sketch a la parte de código fuente sobre la que se escribirá todo el comportamiento que tendrá la regleta, incluyendo la conexión Wifi y la respuesta ante las órdenes dadas por el usuario a través de la aplicación móvil. El sketch de la regleta debe cumplir los siguientes objetivos:

- **Recibir las órdenes dadas por el usuario:** Esto se consigue convirtiendo el lado Linux de la placa en un servidor web capa de trabajar de forma conjunta con el microcontrolador Arduino.

- **Ejecutar las órdenes dadas por el usuario:** Esto se consigue estableciendo el control sobre los pines 7,6,5 que se encargan directamente de modificar el estado de los relés del shield utilizado que a su vez controlan cada enchufe de forma independiente.

Estos dos objetivos se consigues mediante una modificación realizada a partir del sketch bridge. Este sketch está incluido en la sección de ejemplos el IDE, por lo que se explicará de forma breve cuál es su funcionamiento y cómo está programado para logra así un mejor entendimiento de las modificaciones realizadas.

4.2.1 Estructura del sketch bridge

El sketch bridge tiene como principal objetivo convertir a nuestro Arduino Yun en un servidor Web. Para comprender su funcionamiento explicaremos por separado y de forma independiente cada una de los tres apartados que conforman la estructura de un sketch.

- **Librerías:** Necesarias para permitir a la placa de Arduino Yun ofrecer servicios REST.

- **Setup:** Contiene el código que será ejecutado únicamente una vez cuando se encienda el Arduino. Se utiliza principalmente para inicializar las librerías y el resto de recursos que nuestro proyecto necesite (como configurar lo pines de E/S que vayan a ser utilizados).

- **Loop:** Procedimiento que ejecuta todas las líneas de código que contiene dentro de un bucle infinito. En este caso en concreto se utiliza para comprobar cada cierto tiempo si ha llegado alguna petición del lado linio, y si la respuesta es afirmativa se encarga de procesarla.

A continuación y para una mejor compresión se profundiza en cada uno de los procedimientos antes expuestos mostrando además las líneas de código. Se comienza por las **librerías.**

```
#include <Bridge.h>
#include <YunServer.h>
#include <YunClient.h>

// Listen on default port 5555, the webserver on the Yún
// will forward there all the HTTP requests for us.
YunServer server;
```

- **Bridge.h:** Permite establecer la conexión entre los dos procesadores de la placa arduino yun. Recordemos que los pines de E/S se encuentran en el lado Arduino, mientras que la WiFi se encuentra en el lado linio.

- **YunServer.h y YunClient.h:** Permiten reenvíar todas las peticiones que lleguen al puerto 80 del lado linio al puerto 5555 de modo que sean recibidas por el Sketch.

En lo referente al procedimiento **Setup** tenemos un desarrollo que es esencial para el buen funcionamiento del sketch en el que podemos diferencias tres tipos de acciones.

```
void setup() {
  // Bridge startup
  pinMode(13, OUTPUT);
  digitalWrite(13, LOW);

  pinMode(5,OUTPUT);
  digitalWrite(5,LOW);
  pinMode(6,OUTPUT);
  digitalWrite(6,LOW);
  pinMode(7,OUTPUT);
  digitalWrite(7,LOW);

  Bridge.begin();
  digitalWrite(13, HIGH);

  // Listen for incoming connection only from localhost
  // (no one from the external network could connect)
  server.listenOnLocalhost();
  server.begin();
}
```

- **Configuración de los pines de E/S**: Los pines 7,6 y 5 actúan directamente para abrir o cerrar los relés, mientras que el pin 13 indica que la librería Bridge se está inicializando. Para llevar a cabo su función se llama a dos procedimientos muy diferentes.

    - **pinMode**: Se utiliza para que configurar los pines 13,7,6 y 5 como pines de salida (OUTPUT).

- **digitalWrite:** Se utiliza para encender (HIGH) o apagar (LOW) los pines. Al encender por defecto los pines 7,6 y 5 estarán apagados. El pin 13 se trata de un caso

- **Se inicializa la librería Bridge**: Se lleva a cabo con la instrucción "Bridge.begin()" como puede comprobarse el pin 13 inicialmente se encuentra apagado, pero una vez que la librería ha sido inicializada este se enciende (se utiliza sobre todo en pruebas, para indicar que la inicialización ha terminado con éxito).

- **Se inicializa el servidor Yun:** Se lleva a cabo mediante las instrucciones "server.listenOnLocalhost()" y "server.begin()".Una vez que se ha realizado con éxito el servidor web estará operativo y listo para desarrollarse en el procedimiento Loop.

Como ya se ha indicado, el procedimiento **Loop** ejecuta un bucle infinito que cada cierto tiempo gestiona y procesa toda petición que llegue al lado linio. Esta sección del código es más extensa que las anteriores por lo que aquí sólo se explicaran las líneas más importantes.

```
void loop() {
  // Get clients coming from server
  YunClient client = server.accept();

  // There is a new client?
  if (client) {
    // Process request
    process(client);

    // Close connection and free resources.
    client.stop();
  }

  delay(50); // Poll every 50ms
}

void process(YunClient client) {
  // read the command
  String command = client.readStringUntil('/');

  // is "digital" command?
  if (command == "digital") {
    digitalCommand(client);
  }
```

- **Comprobar cada cierto tiempo si ha llegado un nuevo cliente:** La comprobación se realiza cada 50 ms mediante la utilización de la función delay. Para saber si ha llegado un nuevo cliente se ejecutan las líneas de códigos incluidas dentro del bucle "if (client)"

- **Procesar las peticiones que se han llevado a cabo:** Esta parte incluye el resto del código del procedimiento, que es un mayor parte no se está mostrando en la imagen y cuya función reside en trocear la cadena para realizar las acciones que el usuario ha ordenado a través de la aplicación.

-

4.2.2 Gestión y nomenclatura de las peticiones HTTP:

Como ya se ha indicado el objetivo del sketch es convertir a la placa Arduino Yun en un servidor web capaz de ofrecer servicios REST que se encargará de recibir las peticiones http que la aplicación Android se encargará de enviar.

Cada vez que una petición se recibe la cadena se trocea para actuar sobre los pines digitales en función de las instrucciones dadas. A continuación se muestran la estructura utilizada para leer el estado de un pin, ponerlo a cero (abriendo el relé correspondiente y evitando que atraviese corriente por el enchufe) y poniéndolo a uno (cerrando el relé y permitiendo que atraviese corriente por el enchufe). Los ejemplos dados son los utilizados para actuar sobre el primer enchufe (pin 7), se ha tener en cuenta de que para actuar sobre el segundo y el tercer enchufe únicamente sería necesario actuar sobre los pines 6 y 7 respectivamente.

- Solicitar el estado de un pin: http://ip/arduino/digital/7
- Poner un pin a cero: http://ip/arduino/digital/7/0
- Poner un pin a uno: http://ip/arduino/digital/7/1

4.3 Desarrollo Hardware

En este apartado se describe la implementación física del equipo de control situado en el lado remoto, es decir el equipo que actúa directamente sobre los equipos a controlar, para ello se seleccionan los equipos que la componen de acuerdo con las características definidas en el apartado de diseño y se detalla tanto su montaje y distribución en la caja como su conexionado.

Como se indicó en el apartado de diseño el equipo que se implementa está dirigido a control domótico por lo que se presenta en una caja compacta que alberga todos los elementos de control dejando accesible únicamente las tres tomas de enchufe, el cable de alimentación general y una toma USB para la programación y configuración del Arduino.

El equipo está compuesto por los elementos siguientes:

- Caja compacta
- Bases de enchufe
- Arduino YUN
- Placa de relés

- Fuente de alimentación
- Regulador de tensión
- Toma USB
- Toma de alimentación general.

El esquema siguiente indica la distribución de cableado entre los distintos elementos que forman el equipo.

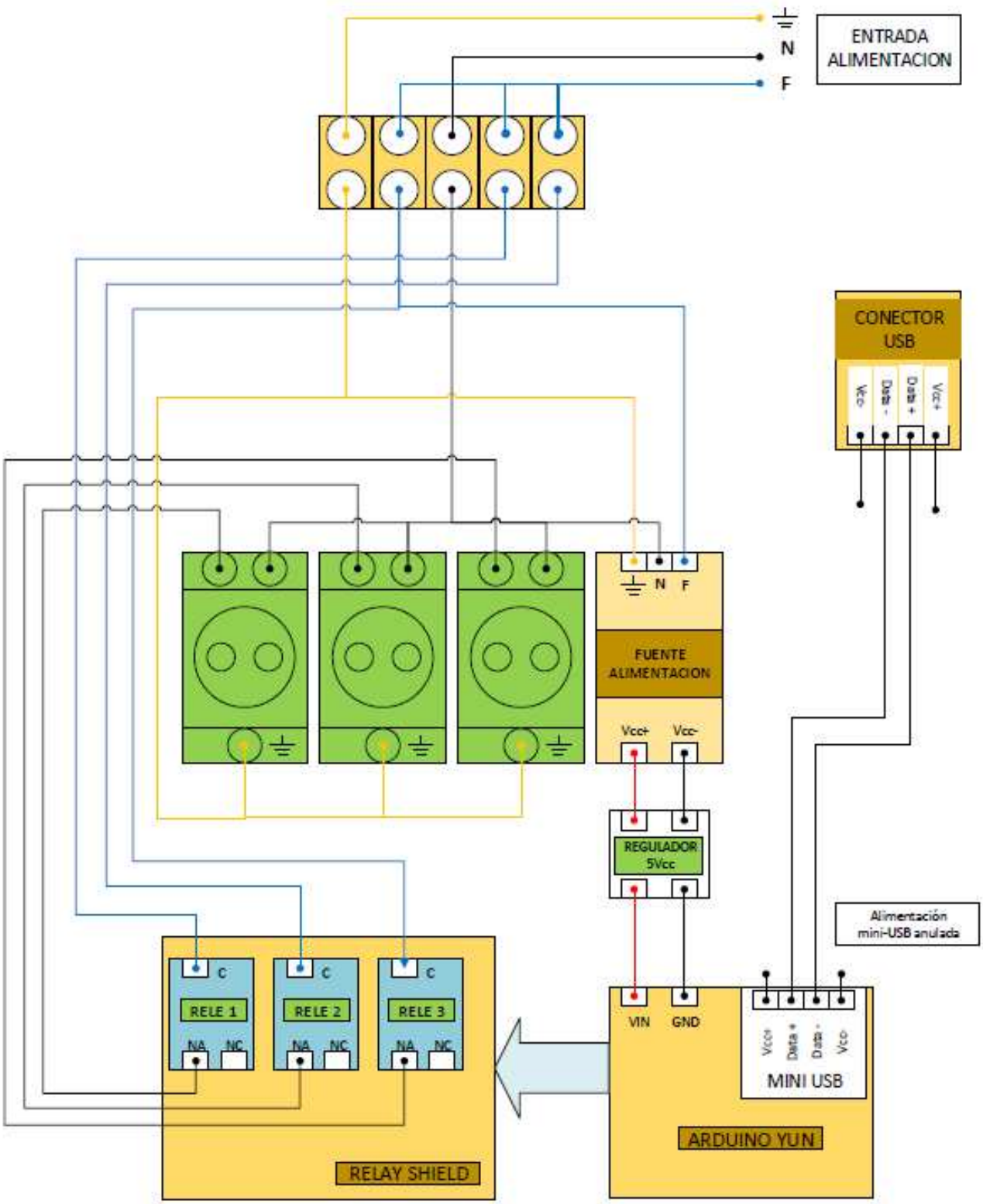


Figura 18: Esquema de Conexionado del Equipo de Lado Remoto

4.3.1 Caja Compacta.

Se ha optado por una caja de superficie de marca Legrand para montaje de elementos en carril DIN de 8 módulos.

Figura 19: Caja de Montaje Legrand

La caja elegida dispone además de una tapa que protege las bases de enchufe cuando el equipo no está en uso.

Las características de esta caja son:

- IP 40 con puerta
- IP 30 sin puerta
- Clase II – Blanco RAL 9010
- Conforme a la Norma IEC 60439-3
- Autoextinguible 650ºC
- Equipada con puerta de plástico transparente

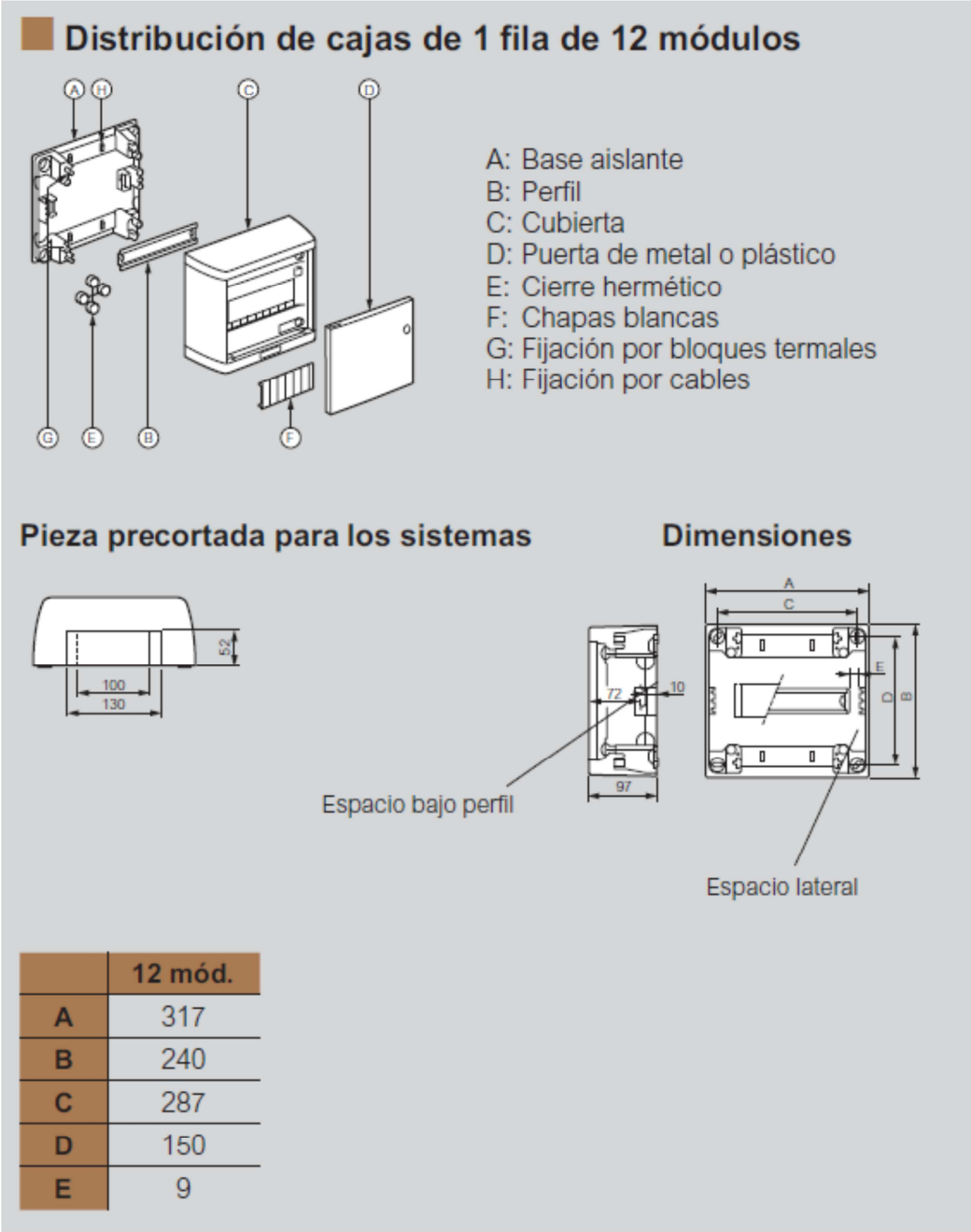


Figura 20: Características Caja de Montaje

4.3.2 Bases de Enchufe

Para las bases de enchufe se ha seleccionado una toma específica para carril DIN con toma de tierra con las características siguientes:

- Montaje en carril DIN
- Con toma de tierra
- Permite cable de hasta 2,5 mm2

- Color gris
- Intensidad máxima 16 A

Figura 21: Enchufe con TT

4.3.3 Arduino YUN

Como se ha indicado en el apartado 2..4.3. Comparativa y selección del equipo de gestión y control para este proyecto se ha seleccionado el Arduino YUN como elemento de control.

En este apartado se describen sus características físicas y eléctricas, así como su montaje en la caja.

El Arduino YUN se presenta en forma de tarjeta de 58,3 mm x 68,6 mm con cuatro taladros para su sujeción. Utilizamos estos taladros para sujetarlo a la caja mediante tornillos y separadores de plástico.

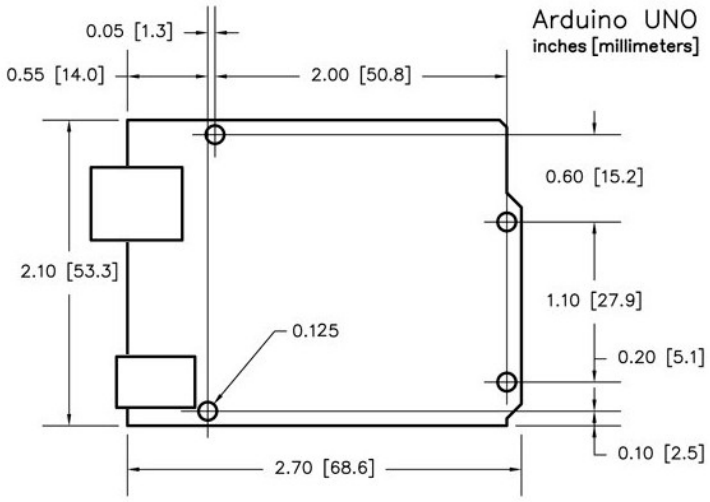


Figura 22: Dimensiones Placa Arduino YUN

Como opción y en el caso de utilizar una caja de mayores dimensiones se puede acoplar a un soporte para carril DIN y montarlo en el mismo carril que las bases de enchufe.

Las características eléctricas de este Arduino son

- Tensión de operación …… 5V
- Voltaje de entrada …… 5V
- Pines I/O digitales …… 20
- Canales PWM …… 7
- Pines entradas Analógicas ……. 12
- Corriente Continua por PIN I/O ……. 40 mA
- Corriente continua por PIN 3,3 V …… 50 mA

A diferencia de otros modelos de Arduino este no dispone de regulador de 5VCC interno por lo que el fabricante recomienda alimentarlo vía el conector micro-USB o mediante el pin Vin utilizando en este caso un regulador externo de 5Vcc para evitar que una sobretensión accidental dañe la placa.

En nuestro caso necesitamos tener libre la entrada micro-USB para posibles reprogramaciones o configuraciones del equipo por lo que alimentamos a través de la entrada Vin instalando un regulador de 5Vcc a su entrada.

4.3.4 Placa de Relés

Para controlar el paso de corriente a cada uno de las bases de enchufe utilizamos una placa de relés. Una ventaja del sistema Arduino es que dispone de shield´s, placas de extensión que le dotan de características externas específicas de las que no dispone inicialmente.

Estos shield´s se adaptan a la placa Arduino tanto geométrica como eléctricamente utilizando para ello los pines de entrada /salida del Arduino sin necesidad de cableado adicional. La placa shield se "enchufa" en los pines del Arduino haciéndolos accesibles desde la propia placa shield. El shield se alimenta desde los pines del Arduino y a través de ellos se comunica con él.

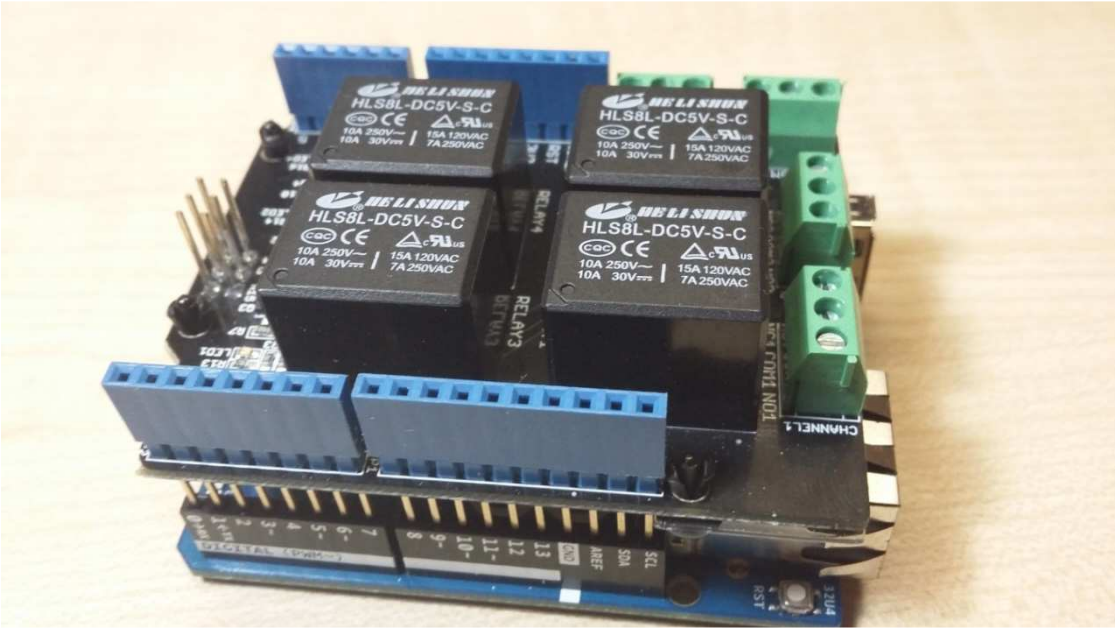


Figura 23: Shield de Relés insertada en Arduino YUN

En el proyecto utilizamos un shield de 4 relés de la casa Seeedstudio de los que inicialmente solo utilizaremos tres.

Las características principales de este shield de relés son:

- Compatible con Arudino YUN al que se conecta a través de los pines de las placas.
- Interface vía digital a través de los pines I/O 4,5,6, y 7
- Conexiones a los relés mediante tornillos.
- LED indicadores de estado para cada rele.
- Relés de gran calidad

- Por cada relé pines de COM, NO (Normalmente Abierto), and NC (Normalmente Cerrado)

Esta placa permite controlar mediante una placa Arduino cuatro relés para conmutar cargas externas tales como bombillas, motores etc. Los relés pueden conmutar hasta 10 A a 30Vcc o 10A a 250 Vca

Dispone de un film transparente aislante por debajo que protege la placa Arduino de cualquier contacto no deseado cuando se conecta a ella

Su sujeción al conjunto es muy simple ya que como hemos indicado anteriormente se "Enchufa" Pin a Pin sobre la placa Arduino YUN.

4.3.5 Alimentación

Dado el consumo de los relés, el fabricante recomienda disponer de una alimentación externa para alimentar la placa dado que el puerto USB no proporciona suficiente corriente.

Por otra parte para asegurar que no se suministran más de 5Vcc, lo que dañaría los circuitos del Arduino ponemos a su salida un circuito regulador que limita la salida a 5Vcc.

Fuente alimentación 5V

Seleccionamos una fuente de alimentación con marcado CE de 5v con salida de 2A, de pequeño tamaño, ideal para instalaciones domoticas con Arduino, su pequeño tamaño le permite ser instalada en pequeños espacios.

Figura 24: Fuente de Alimentación de 5Vcc

Caracteristicas:

- Corriente de salida: 2A
- Voltaje de entrada: 110/220 V
- Tensión de salida: 5 V
- Factor de regulación de voltaje: 1%
- 7.1 cm x 4 cm x 3 cm
- Regulación de la carga: 2%
- Potencia de salida: 10W
- Ruido de la ondulación de la salida: 1%
- Marcado CE y FE

Esta Fuente la sujetamos a una de los enchufes que a su vez está acoplado al carril DIN quedando así un conjunto compacto.

Regulador de tensión a 5Vcc

Para asegurar la tensión de salida a 5Vcc utilizamos el circuito regulador de tensión LM7805.

El circuito de regulación incorpora un condensadores electrolíticos de 100 microfaradios tanto a la entrada como a la salida al objeto de obtener un buen filtrado de la tensión.

Tanto el regulador como los condensadores se sueldan a una placa de circuito impreso, intercalándose esta entre la salida de la Fuente de Alimentación y la entrada Vin del Arduino YUN.

4.3.6 Conector USB

Para tener el conector acceisble desde el exterior se mecaniza la caja haciendo una ventana y utilizamos un conector USB para PCB soldado a una placa de circuito impreso a la que tambien se han soldados los hilos de Data+ y Data- del cable microUSB conectado al Arduino YUN.

4.3.7 Montaje Final

Se incluyen a continuación unas fotografías del interior y exterior del equipo lado remoto terminado.

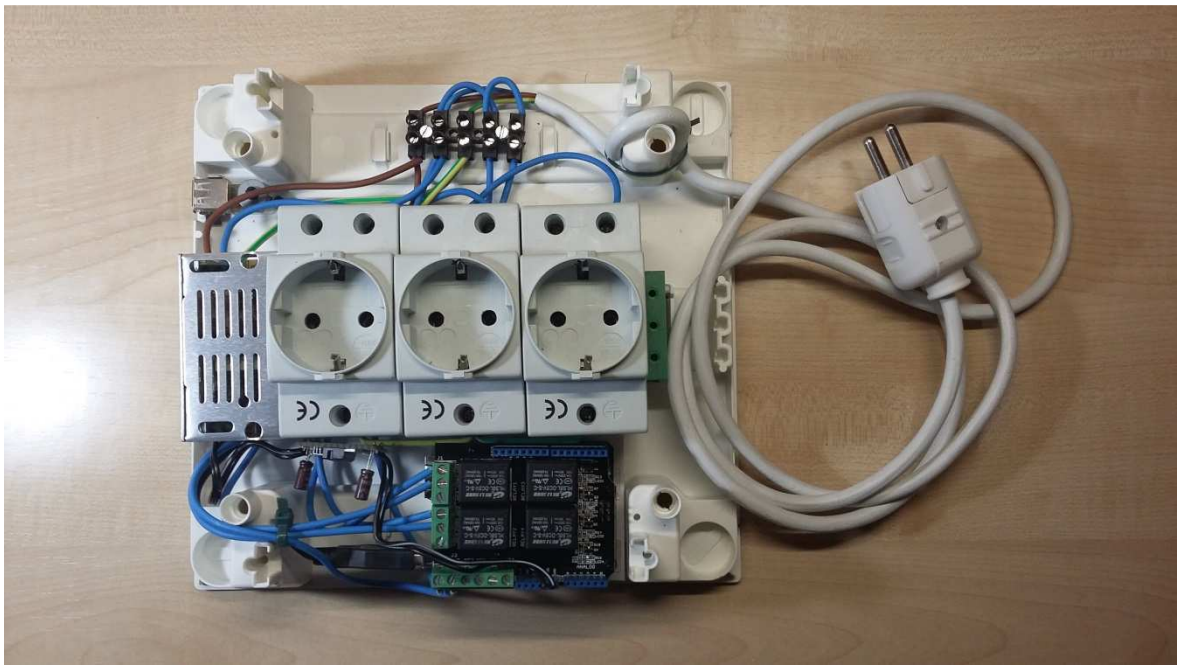


Figura 25: Viste Interior del equipo Lado Remoto Terminado

Figura 26: Vista exterior del Equipo lado Remoto Terminado

5 Integración, pruebas y resultados

Durante la definición y desarrollo del proyecto se han ido realizado diversas pruebas parciales que han permitido corregir errores y determinar finalmente el buen funcionamiento del sistema.

5.1 Carga de Aplicaciones en Terminal Android

La aplicación de carga desde el entorno de desarrollo de Arduino

5.2 Carga de Aplicaciones en Arduino del Equipo Lado Remoto

Para cargar la aplicación Arduino se conecta la placa al ordenador mediante su puerto micro-USB y desde la plataforma arduino se carga la aplicación.

5.3 Pruebas en Prototipo

Con las primeras versiones de las aplicaciones cargadas se han realizado pruebas parciales sobre un prototipo montado sobre una placa Protoboard encendiendo y apagando diodos led en local

5.4 Pruebas eléctricas equipo lado remoto

Desde el punto de vista del equipo de lado remoto una vez montado y cableado todos los elementos se han realizado las siguientes pruebas:

- Revisión visual de todos los cableados comprobando que no hay errores en el mismo.
- Timbrado con polímetro de todos los cables confirmando su continuidad eléctrica
- Con el equipo conectado a os 220 Vca de entrada y sin conctar la alimentación al Arduino YUN se mide con un voltímetro la tensión en los siguientes puntos:
    - Entrada a la Fuente de alimentación: 220 Vca
    - Salida de la Fuente de alimentación: Se regula a la máxima tenson de salida 6,5 Vcc
    - Se comprueba con el voltímetro que esta es la tensión que tenemos a la entrada del regulador.
    - Se mide la salida del regulador confirmando que tenemos una salida estable de 5Vcc
- Se desconecta la alimentación de entrada y se conecta la salida del regulador a los pines Vin y Ground del Arduino

- Se conecta de nuevo la alimentación de entrada y se comprueban las tensiones en bornas de los enchufes comprobándose que no están alimentados.

- Forzamos uno a uno la conmutación de cada uno de los relés conectados a los enchufes forzando el cierre del contacto Normalmente abierto comprobando que en bornas de cada enchufe hay 220Vca

5.5 Integración del conjunto y pruebas finales

Una vez cargadas las aplicaciones correspondientes en el terminal Android y una vez montado el equipo remoto y cargadas sus aplicaciones se ha pasado a realizar las pruebas finales.

Para ello se ha conectado una bombilla en cada uno de los enchufes procediéndose a encender y apagar una a una cada una de ellas mediante órdenes de encendido y apagado en el terminal Android verificándose su correcto funcionamiento.

También se han comprobado los mecanismos de seguridad implementados en el equipo:

Sobre la pantalla de configuración en el campo dirección IP / Nombre de dominio se han introducido caracteres que nunca podrían formar parte de una dirección IP o un nombre de dominio, como letras mayúsculas, la letra ñ y signos de puntuación como exclamaciones o interrogaciones. En todos los casos la aplicación ha respondido satisfactoriamente presentando el mensaje “por favor seleccione una configuración válida”

Con una dirección IP que no se corresponde con la de la regleta la conexión queda abortada mostrándose el mensaje “conexión abortada” en la pantalla de control. En esta situación se ha intentado dar órdenes de encendido y apagado a cada uno de los enchufes no llegando la orden a ejecutarse y apareciendo en pantalla el mensaje “primero debe realizarse la conexión con la regleta”

Con una interrupción momentánea de la red wifi mientras se establece la conexión solo se permite leer el estado de algunos enchufes. Se comprueba que en este caso solo se tiene control sobre los enchufes que han realizado la conexión y la pantalla de control presenta el mensaje “Conexión Parcial”

Con la conexión realizada con el equipo remoto si se corta la conexión por una interrupción en la señal Wifi y enviamos la orden al enchufe, la aplicación detecta que que no se ha podido llevar a cabo y nos informa por pantalla con el mensaje “no se ha podido establecer la conexión”

6 Conclusiones y trabajo futuro

6.1 Conclusiones

Analizando los resultados del proyecto se puede concluir que se han cumplido los objetivos previstos para este proyecto

- Se ha implementado un sistema capaz de controlar remotamente tres equipos independientes encendiendo y apagando su alimentación.
- Las aplicaciones implementadas son estables
- Se han puesto las bases para de una manera sencilla ampliar el número de equipos a controlar.

Desde el punto de vista didáctico

- Se ha entrenado el uso en las herramientas Android de programación.
- Se ha entrenado el uso en las herramientas de programación de Arduino

6.2 Trabajo futuro

La utilización de terminales móviles para controlar equipos eléctricos es un campo que todavía no forma parte de nuestros hogares. Este sistema es una primera aproximación que ofrece muy buenos resultados pero que todavía tiene cosas que mejorar. A continuación se propone una lista de caminos que deberían tomarse a la hora de ampliar el proyecto, todos estos caminos se dividen en tres grupos:

- **Mejoras de funcionalidad:** Son mejoras destinadas desarrollar y aumentar los servicios y prestaciones ofrecidos por el sistema. Ejemplo de vías que formarían parte de este grupo serían mejoras en la seguridad y la inclusión de un tutorial
- **Mejoras comerciales:** Son mejoras destinadas tanto a comercializar el sistema como a hacer que este sea accesible a un público más amplio. De este grupo formarían parte iniciativas como crear una aplicación equivalente que funcione en dispositivos iOS y que ambas aplicaciones estén disponibles en varios idiomas.
- **Desarrollo de nuevos equipos:** Se trata del desarrollo de nuevos sistemas con características especiales que serían mucho más útiles en entornos diferentes. Más adelante en esta misma sección se propone un sistema alternativo diseñado para ser utilizado en entornos industriales.

Una vez se han mostrado los tres grandes grupos y para una mejor organización se procederá a clasificar todas las mejoras sugeridas en función de estos tres grandes grupos. Cada una cuenta con una descripción explicando el motivo por el que debe llevarse a cabo.

- **Mejoras de funcionalidad**

    - Tutorial: Aunque se trata de una aplicación cuyo manejo es fácil e intuitivo cualquier diseño novedoso que pretenda ser utilizado por un gran número de personas debe tener como objetivo simplificar su manejo todo lo posible. En el momento en el que el proyecto continúe creciendo se hará muy necesario incluir un tutorial integrado en la propia aplicación que explique el manejo y la instalación del sistema completo.

    - Seguridad: Si hay un concepto que con el desarrollo de las comunicaciones a distancia se está volviendo cada vez más importante, ese es el de seguridad. Cifrar las comunicaciones entre la aplicación y la regleta sería por tanto una posible mejora a tener muy en cuenta.

    - Programador automático: En estos momentos el sistema permite controlar una regleta a distancia en tiempo real dando órdenes a cada uno de los enchufes de forma independiente. Por lo tanto el siguiente paso sería dar la posibilidad al usuario de dar las órdenes programando cuando y durante cuánto tiempo quiere que los enchufes estén encendidos o apagados.

- **Mejoras comerciales**

    - Idiomas: Hasta el momento la aplicación se encuentra en español, que es el segundo idioma más hablado de todo el mundo, sin embargo a día de hoy el inglés sigue siendo una lengua fundamental a nivel global, tanto en número de hablantes como en el mundo de los negocios. Si se quiere que la aplicación llegue al mayor número de personas es fundamental que también se encuentre disponible en inglés.

    - Disponible en dispositivos iOS: El principal motivo elegido para desarrollar la aplicación en Android ha sido porque es el sistema operativo de móviles más utilizado en España y además tiene mayor implantación territorial en el resto del mundo. Sin embargo iOS está muy extendido en países muy importantes para el mercado como Estados Unidos, Reino Unido, Francia y Japón entre otros. A día de hoy, si se pretende que una aplicación esté disponible para el gran público ésta debe poder utilizarse en ambos sistemas operativos.

    - Inclusión de la aplicación tanto en la Play Store de Android como en la App Store de iOS para que los usuarios puedan acceder a ella.

- **Desarrollo de nuevos equipos**

    - Se propone el desarrollo de un sistema para entornos industriales como el que se ha apuntado en el apartado de desarrollo hardware.

        - En este caso se plantearía un montaje menos compacto con todos los elementos montados sobre carril DIN dentro de un armario de mando y control, y con los cableados distribuidos por canaleta.

- El Arduino YUN y la placa de relés se incluiría en una caja adaptada al carril DIN y para la FA se utilizaría un adaptador para carril DIN.
- Dependiendo de la aplicación concreta se utilizarían bien tomas de enchufe para carril DIN similares a las utilizadas en el proyecto o se conectarían los equipos a controlar directamente a bornas a tornillo sobre carril DIN.

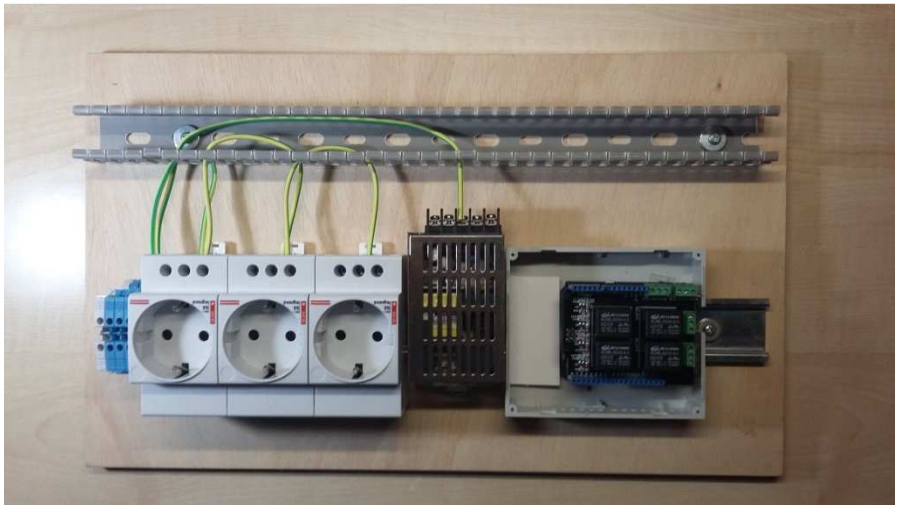


Figura 27: Prototipo de desarrollo para entornos industriales

Referencias

[1] "El gran libro de Android", Jesús Tomás Girones, 2011.
[2] "Desarrollo de aplicaciones para dispositivos móviles", EPS-UAM.
[3] "Pagina WEB de Android": http://developer.android.com/
[4] "Página web de Arduino" : https://www.arduino.cc/
[5] "Pagina WEB SeeedStudio" : http://www.seeedstudio.com/wiki/Relay_Shield_V2.0
[6] "Observatorio Nacional de las Telecominicaciones y de la SI": http://www.ontsi.red.es/ontsi/es/indicador/evoluci%C3%B3n-del-n%C3%BAmero-de-clientes-de-telefon%C3%ADa-m%C3%B3vil-en-espa%C3%B1a
[7] "Página WEB DEVICE ATLAS": https://deviceatlas.com/device-data/explorer/webusage/traffic/mobile/country/es?search=Apple%20iPhone%205
[8] "Texas Instruments " (data sheet regulador 7800)

Anexos

A Manual de instalación

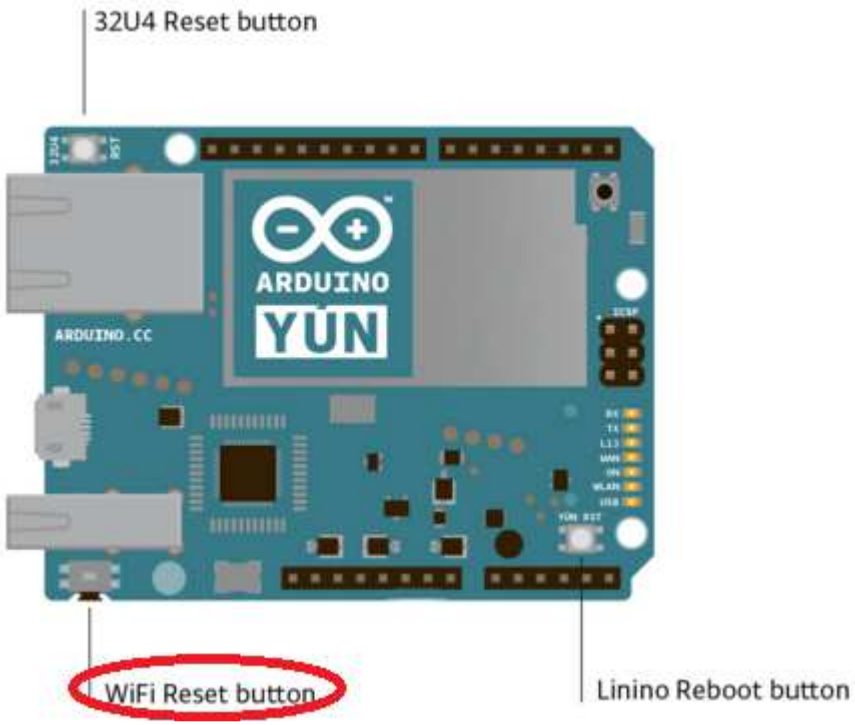
La instalación se lleva a cabo mediante tres sencillos pasos que permitirán al usuario utilizar su terminal Android para controlar la regleta inteligente y por tanto cualquier equipo eléctrico que se halle enchufado a ella.

- Paso 1: Conectar la regleta a la red Wifi
- Paso 2: Instalar el sketch que permite funcionar a la regleta
- Paso 3: Configurar la aplicación en el terminal Android

Es fundamental tener en cuenta que Arduino Yun es la plataforma hardware que controla la regleta, lo que es una ventaja, ya que instalar y configurar la regleta será al menos tan sencillo como configurar una de estas placas.

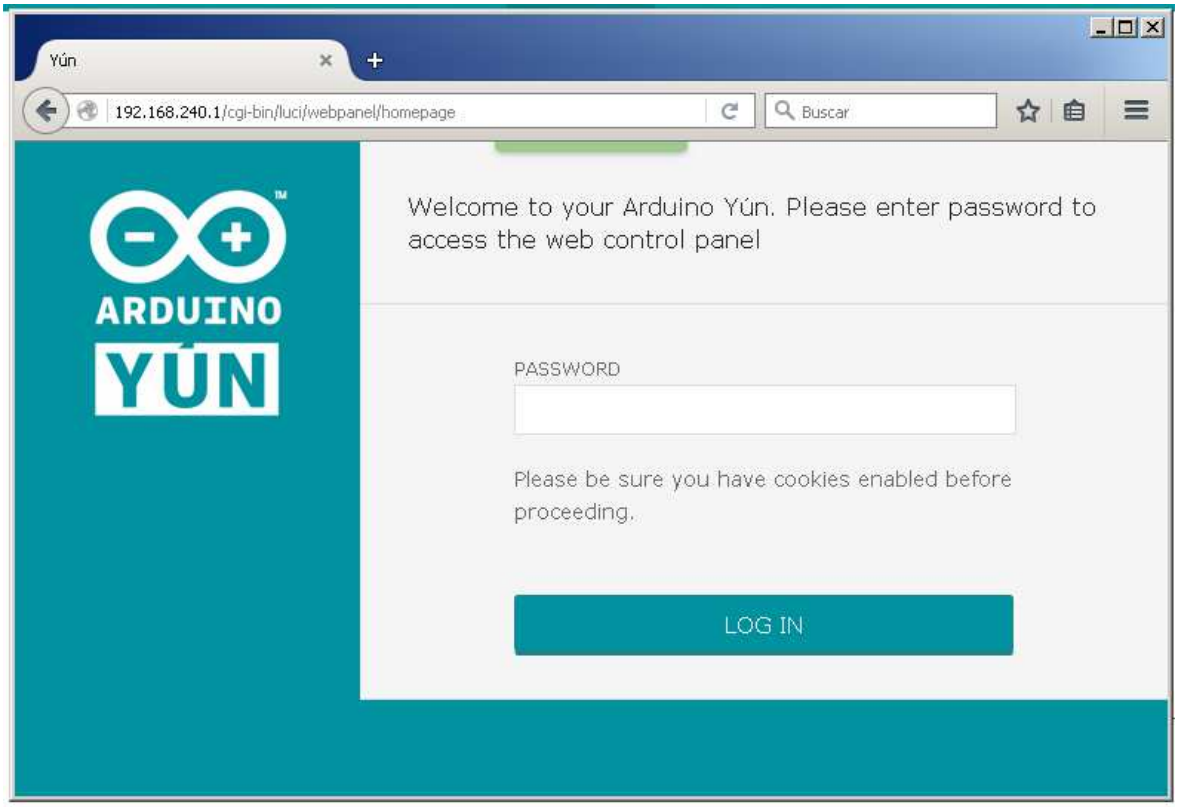
A.1 Conectar la regleta a la red Wifi

La primera etapa de configuración implica devolver a la placa Arduino a su estado inicial, para ello se pulsa el botón de Reset WiFI (WLAN RST) durante al menos 30 segundos. Esto borra todos los ficheros instalados y las características de red.

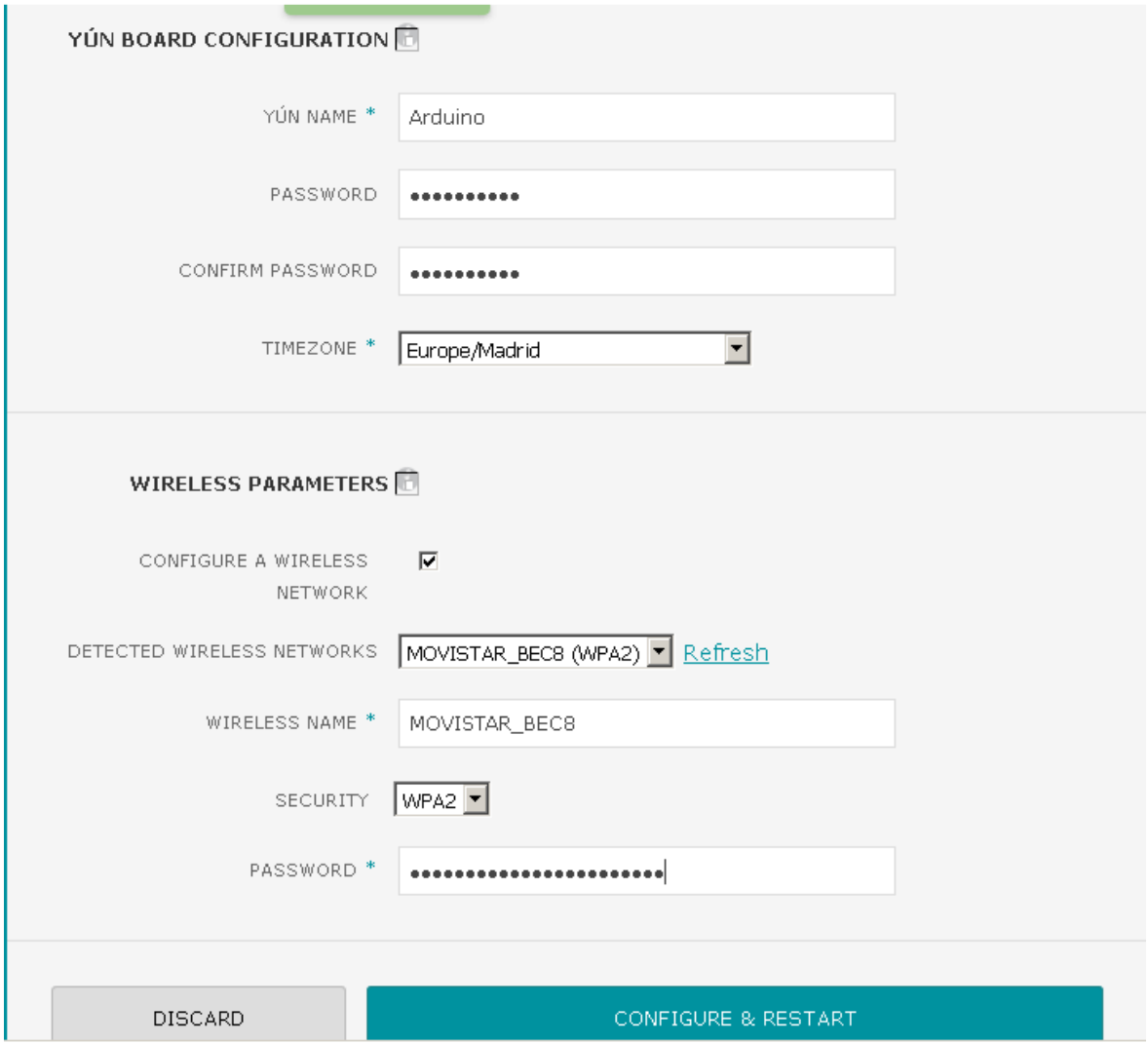


A continuación se conectará con un PC utilizando el cable microUSB y se esperará pacientemente mientras se instala el software controlador del dispositivo. Una vez se haya realizado Arduino Yun creará de forma automática una red WiFi llamada ArduinoYun-XXXXX. Se procederá a conectar el ordenador a esa red.

Desde el navegador local se puede encontrar a la pantalla de configuración mediante la dirección 192.168.240.1. Para acceder es necesario introducir la contraseña “arduino”

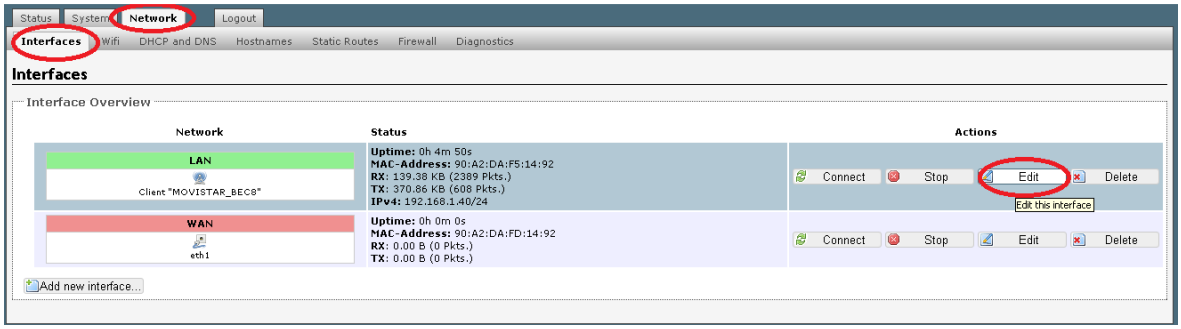


Una vez hecho eso se accederá a la pantalla de configuración y se rellenará el formulario donde se le dará un nombre a nuestra regleta, se incluirá una contraseña de al menos 8 caracteres, se indicará nuestra zona horaria y el país en el que nos encontramos, y por último se seleccionará la red WiFi a la que se desee conectar la regleta incluyendo la contraseña para que la conexión pueda llevarse a cabo. Es muy importante marcar los servicios REST como “open” en la opción justo debajo del formulario.



Una vez realizado todos estos pasos se guardará la configuración y Arduino se reiniciará de forma automática. Llegados a este punto es conveniente volver a conectar la placa al PC y tras esperar un minuto para dar tiempo a la placa a iniciarse, buscar la ip que ha sido asignada a la regleta en la red WiFi. Una vez hecho esto se podrá acceder con la nueva contraseña que se había elegido. Nuestro objetivo será asignar una IP fija a nuestro Arduino.

El proceso es más sencillo todavía que en el caso anterior, únicamente hay que acceder al panel de configuraciones avanzadas (luci), y de allí a Newtwork, Interface y Edit. Una vez hecho esto se asignará una IP estática



A.2 Instalación del sketch

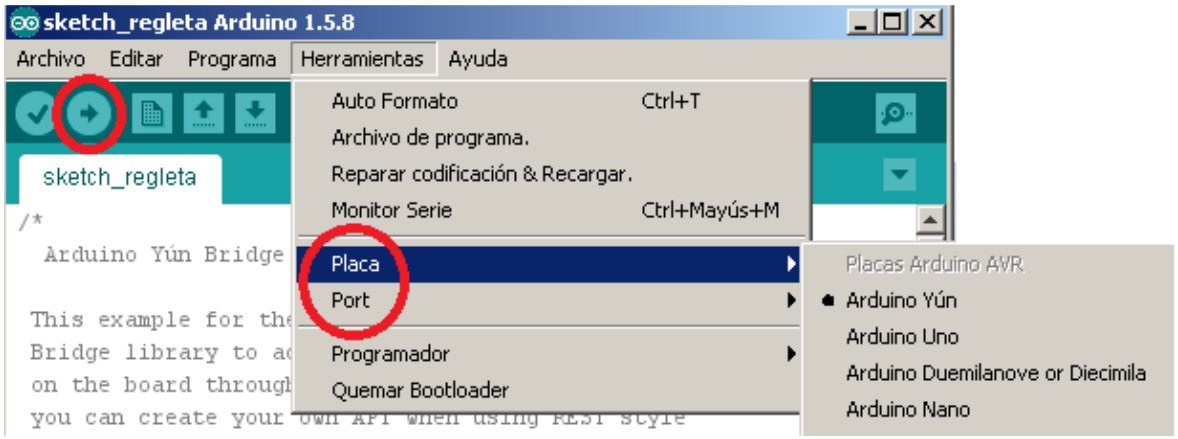
Como ya se ha indicado en los apartados de diseño y desarrollo de la memoria se pretende que tanto el lado Linio como el microprocesador Arduino trabajen de forma conjunta para ejecutar las órdenes del usuario. Para ello es necesario que el lado Linux funcione con un servidor web, y que además tenga comunicación directa con el lado Arduino sirviéndose de la librería Bridge.

Todo eso se consigue instalando el sketch diseñado para el proyecto que a su vez es una modificación del sketch Bridge tal y como se ha explicado en apartados anteriores de la memoria.

Para la instalación del sketch es necesario en primer lugar descargar la última versión de Arduino Software (IDE) de la página oficial de arduino.

https://www.arduino.cc/en/Main/Software

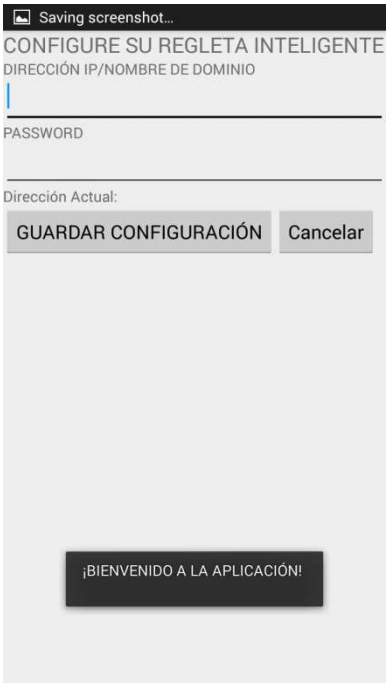
Una vez descargado tan sólo hay que ejecutarlo, seleccionar la placa y el puerto y almacenar el programa dentro de nuestra plataforma hardware. De modo que la regleta quedará lista para utilizarse



A.3 Configurar la aplicación en el terminal Android

Este se trata posiblemente del paso más intuitivo de todos, únicamente hay que abrir la aplicación y acceder a la pantalla de configuración (la propia aplicación nos enviará a esa página si detecta que es la primera vez que la estamos utilizando).

Una vez allí simplemente hay que rellenar el formulario almacenando la dirección IP que corresponde con el servidor de nuestra regleta y el equipo estará listo para utilizarse.



PRESUPUESTO

1) Ejecución Material

- Terminal Movil Android..... ..... 300,00 €
- Material para el Prototipo
  - Caja superficie..... 20,00 €
  - 3 Base enchufe carril din..... 31,50 €
  - 1 Arduino Yun ..... 57,90 €
  - 1 placa Shiel 4 reles ..... 23,15 €
  - 1 Fuente alimentación + regulador 5V..... 12,00 €
  - Cables y Pequeño material 25,00 €1
- Material de oficina ..... 150,00 €
- **Total de ejecución material ..... 619,55 €**

2) Gastos generales

- 16 % sobre Ejecución Material ..... 99,13 €

3) Beneficio Industrial

- 6 % sobre Ejecución Material ..... 37,17 €

4) Honorarios Proyecto

- 640 horas a 15 € / hora ..... 9600,00 €

5) Material fungible

- Gastos de impresión ..... 60,00 €
- Encuadernación..... 200,00 €

6) Subtotal del presupuesto

- Subtotal Presupuesto ..... 10.615,85 €

7) I.V.A. aplicable

- 21 % Subtotal Presupuesto..... 2.229,33 €

8) Total presupuesto

- Total Presupuesto..... 12.845,18 €

Madrid, Enero de 2016

El Ingeniero Jefe de Proyecto

Fdo.: Antonio Rovira Moreno
Ingeniero de Telecomunicación

PLIEGO DE CONDICIONES

Este documento contiene las condiciones legales que guiarán la realización, en este proyecto, de un SISTEMA DE CONTROL REMOTO DE EQUIPOS ELECTRICOS MEDIANTE TERMINALES ANDROID. En lo que sigue, se supondrá que el proyecto ha sido encargado por una empresa cliente a una empresa consultora con la finalidad de realizar dicho sistema. Dicha empresa ha debido desarrollar una línea de investigación con objeto de elaborar el proyecto. Esta línea de investigación, junto con el posterior desarrollo de los programas está amparada por las condiciones particulares del siguiente pliego.

Supuesto que la utilización industrial de los métodos recogidos en el presente proyecto ha sido decidida por parte de la empresa cliente o de otras, la obra a realizar se regulará por las siguientes:

Condiciones generales

1. La modalidad de contratación será el concurso. La adjudicación se hará, por tanto, a la proposición más favorable sin atender exclusivamente al valor económico, dependiendo de las mayores garantías ofrecidas. La empresa que somete el proyecto a concurso se reserva el derecho a declararlo desierto.

2. El montaje y mecanización completa de los equipos que intervengan será realizado totalmente por la empresa licitadora.

3. En la oferta, se hará constar el precio total por el que se compromete a realizar la obra y el tanto por ciento de baja que supone este precio en relación con un importe límite si este se hubiera fijado.

4. La obra se realizará bajo la dirección técnica de un Ingeniero Superior de Telecomunicación, auxiliado por el número de Ingenieros Técnicos y Programadores que se estime preciso para el desarrollo de la misma.

5. Aparte del Ingeniero Director, el contratista tendrá derecho a contratar al resto del personal, pudiendo ceder esta prerrogativa a favor del Ingeniero Director, quien no estará obligado a aceptarla.

6. El contratista tiene derecho a sacar copias a su costa de los planos, pliego de condiciones y presupuestos. El Ingeniero autor del proyecto autorizará con su firma las copias solicitadas por el contratista después de confrontarlas.

7. Se abonará al contratista la obra que realmente ejecute con sujeción al proyecto que sirvió de base para la contratación, a las modificaciones autorizadas por la superioridad o a las órdenes que con arreglo a sus facultades le hayan comunicado por escrito al Ingeniero Director de obras siempre que dicha obra se haya ajustado a los preceptos de los pliegos de condiciones, con arreglo a los cuales, se harán las modificaciones y la valoración de las diversas unidades sin que el importe total pueda exceder de los presupuestos aprobados. Por consiguiente, el número de unidades que se consignan en el proyecto o en el presupuesto, no podrá servirle de fundamento para entablar reclamaciones de ninguna clase, salvo en los casos de rescisión.

8. Tanto en las certificaciones de obras como en la liquidación final, se abonarán los trabajos realizados por el contratista a los precios de ejecución material que figuran en el presupuesto para cada unidad de la obra.

9. Si excepcionalmente se hubiera ejecutado algún trabajo que no se ajustase a las condiciones de la contrata pero que sin embargo es admisible a juicio del Ingeniero Director de obras, se dará conocimiento a la Dirección, proponiendo a la vez la rebaja de precios que el Ingeniero estime justa y si la Dirección resolviera aceptar la obra, quedará el contratista obligado a conformarse con la rebaja acordada.

10. Cuando se juzgue necesario emplear materiales o ejecutar obras que no figuren en el presupuesto de la contrata, se evaluará su importe a los precios asignados a otras obras o materiales análogos si los hubiere y cuando no, se discutirán entre el Ingeniero Director y el contratista, sometiéndolos a la aprobación de la Dirección. Los nuevos precios convenidos por uno u otro procedimiento, se sujetarán siempre al establecido en el punto anterior.

11. Cuando el contratista, con autorización del Ingeniero Director de obras, emplee materiales de calidad más elevada o de mayores dimensiones de lo estipulado en el proyecto, o sustituya una clase de fabricación por otra que tenga asignado mayor precio o ejecute con mayores dimensiones cualquier otra parte de las obras, o en general, introduzca en ellas cualquier modificación que sea beneficiosa a juicio del Ingeniero Director de obras, no tendrá derecho sin embargo, sino a lo que le correspondería si hubiera realizado la obra con estricta sujeción a lo proyectado y contratado.

12. Las cantidades calculadas para obras accesorias, aunque figuren por partida alzada en el presupuesto final (general), no serán abonadas sino a los precios de la contrata, según las condiciones de la misma y los proyectos particulares que para ellas se formen, o en su defecto, por lo que resulte de su medición final.

13. El contratista queda obligado a abonar al Ingeniero autor del proyecto y director de obras así como a los Ingenieros Técnicos, el importe de sus respectivos honorarios facultativos por formación del proyecto, dirección técnica y administración en su caso, con arreglo a las tarifas y honorarios vigentes.

14. Concluida la ejecución de la obra, será reconocida por el Ingeniero Director que a tal efecto designe la empresa.

15. La garantía definitiva será del 4% del presupuesto y la provisional del 2%.

16. La forma de pago será por certificaciones mensuales de la obra ejecutada, de acuerdo con los precios del presupuesto, deducida la baja si la hubiera.

17. La fecha de comienzo de las obras será a partir de los 15 días naturales del replanteo oficial de las mismas y la definitiva, al año de haber ejecutado la provisional, procediéndose si no existe reclamación alguna, a la reclamación de la fianza.

18. Si el contratista al efectuar el replanteo, observase algún error en el proyecto, deberá comunicarlo en el plazo de quince días al Ingeniero Director de obras, pues transcurrido ese plazo será responsable de la exactitud del proyecto.

19. El contratista está obligado a designar una persona responsable que se entenderá con el Ingeniero Director de obras, o con el delegado que éste designe, para todo relacionado con ella. Al ser el Ingeniero Director de obras el que interpreta el proyecto, el contratista deberá consultarle cualquier duda que surja en su realización.

20. Durante la realización de la obra, se girarán visitas de inspección por personal facultativo de la empresa cliente, para hacer las comprobaciones que se crean oportunas. Es obligación del contratista, la conservación de la obra ya ejecutada hasta la recepción de la misma, por lo que el deterioro parcial o total de ella, aunque sea por agentes atmosféricos u otras causas, deberá ser reparado o reconstruido por su cuenta.

21. El contratista, deberá realizar la obra en el plazo mencionado a partir de la fecha del contrato, incurriendo en multa, por retraso de la ejecución siempre que éste no sea debido a causas de fuerza mayor. A la terminación de la obra, se hará una recepción provisional previo reconocimiento y examen por la dirección técnica, el depositario de efectos, el interventor y el jefe de servicio o un representante, estampando su conformidad el contratista.

22. Hecha la recepción provisional, se certificará al contratista el resto de la obra, reservándose la administración el importe de los gastos de conservación de la misma hasta su recepción definitiva y la fianza durante el tiempo señalado como plazo de garantía. La recepción definitiva se hará en las mismas condiciones que la provisional, extendiéndose el acta correspondiente. El Director Técnico propondrá a la Junta Económica la devolución de la fianza al contratista de acuerdo con las condiciones económicas legales establecidas.

23. Las tarifas para la determinación de honorarios, reguladas por orden de la Presidencia del Gobierno el 19 de Octubre de 1961, se aplicarán sobre el denominado en la actualidad “Presupuesto de Ejecución de Contrata” y anteriormente llamado ”Presupuesto de Ejecución Material” que hoy designa otro concepto.

Condiciones particulares

La empresa consultora, que ha desarrollado el presente proyecto, lo entregará a la empresa cliente bajo las condiciones generales ya formuladas, debiendo añadirse las siguientes condiciones particulares:

1. La propiedad intelectual de los procesos descritos y analizados en el presente trabajo, pertenece por entero a la empresa consultora representada por el Ingeniero Director del Proyecto.

2. La empresa consultora se reserva el derecho a la utilización total o parcial de los resultados de la investigación realizada para desarrollar el siguiente proyecto, bien para su publicación o bien para su uso en trabajos o proyectos posteriores, para la misma empresa cliente o para otra.

3. Cualquier tipo de reproducción aparte de las reseñadas en las condiciones generales, bien sea para uso particular de la empresa cliente, o para cualquier otra aplicación, contará con autorización expresa y por escrito del Ingeniero Director del Proyecto, que actuará en representación de la empresa consultora.

4. En la autorización se ha de hacer constar la aplicación a que se destinan sus reproducciones así como su cantidad.

5. En todas las reproducciones se indicará su procedencia, explicitando el nombre del proyecto, nombre del Ingeniero Director y de la empresa consultora.

6. Si el proyecto pasa la etapa de desarrollo, cualquier modificación que se realice sobre él, deberá ser notificada al Ingeniero Director del Proyecto y a criterio de éste, la empresa consultora decidirá aceptar o no la modificación propuesta.

7. Si la modificación se acepta, la empresa consultora se hará responsable al mismo nivel que el proyecto inicial del que resulta el añadirla.

8. Si la modificación no es aceptada, por el contrario, la empresa consultora declinará toda responsabilidad que se derive de la aplicación o influencia de la misma.

9. Si la empresa cliente decide desarrollar industrialmente uno o varios productos en los que resulte parcial o totalmente aplicable el estudio de este proyecto, deberá comunicarlo a la empresa consultora.

10. La empresa consultora no se responsabiliza de los efectos laterales que se puedan producir en el momento en que se utilice la herramienta objeto del presente proyecto para la realización de otras aplicaciones.

11. La empresa consultora tendrá prioridad respecto a otras en la elaboración de los proyectos auxiliares que fuese necesario desarrollar para dicha aplicación industrial, siempre que no haga explícita renuncia a este hecho. En este caso, deberá autorizar expresamente los proyectos presentados por otros.

12. El Ingeniero Director del presente proyecto, será el responsable de la dirección de la aplicación industrial siempre que la empresa consultora lo estime oportuno. En caso contrario, la persona designada deberá contar con la autorización del mismo, quien delegará en él las responsabilidades que ostente.