UNIVERSIDAD AUTONOMA DE MADRID

ESCUELA POLITECNICA SUPERIOR

PROYECTO FIN DE CARRERA
Ingeniería de Telecomunicación

CONTROL DE ACCESO, OBJETOS Y EQUIPOS ELÉCTRICOS MEDIANTE TELÉFONO INTELIGENTE

Luis Pablo González Marabini

Julio 2016

CONTROL DE ACCESO, OBJETOS Y EQUIPOS ELÉCTRICOS MEDIANTE TELÉFONO INTELIGENTE

AUTOR: Luis Pablo González Marabini
TUTOR: Carlos Minchola

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

Resumen

Este proyecto consiste en definir, diseñar, desarrollar e implementar en una maqueta un sistema que permita el control remoto de actuadores de corriente continua y alterna a discapacitados visuales mediante el uso de un Smartphone y la tecnología Buetooth.

El sistema constará de varias partes (subsistemas) diferenciadas, teniendo el lado **circuito remoto** donde se encuentran los actuadores que se quieren controlar, y el lado del usuario con las **aplicaciones móviles** a través del cual se enviarán las órdenes.

Ambos lados se comunicarán entre sí a través de Bluetooth y un microcontrolador que hará de mediador entre el circuito remoto y las aplicaciones móviles.

Palabras clave

Actuador, Android, aplicación, Arduino, Bluetooth, circuito interruptor, domótica, DOMUS, CAECUS, gestos, maqueta demostradora, MOSFET, sistema

Abstract

The aim of this project is to define, design, develop and implement a mock-up of a system that allows remote control over AC and CC actuators to blind people through Smartphones and Buetooth technology.

The system will consist of two several parts (subsystems). On one hand will be the **remote circuit** whith the actuators to be controlled, and on the other hand will be the **mobile applications** where the orders are sent by the user..

Both sides will communicate with each other via Bluetooth and a microcontroller that will mediate between the remote circuit and the mobile applications.

Keywords

Actuator, Android, application, Arduino, Bluetooth, CAECUS, DOMUS, home automation, gestures, mock-up, MOSFET, system, switch circuit.

Agradecimientos

Ha llegado el momento de cerrar una etapa de mi vida que me ha aportado tanto a nivel formativo como personal. Si bien me alegra haber podido disfrutar de esta viaje que ya mismo termina, me ilusiona pensar en el camino que comienza a partir de ahora.

Querría empezar agradeciendo a los que merecen todo el crédito de éste trabajo, ya que sin ellos, no hay lugar a duda que no estaría escribiendo estas líneas. Mamá, papá, gracias de corazón. Sois los únicos que pueden presumir de haberme acompañado de principio a fin, y de haberme estado apoyando en cada éxito, y sobretodo, en cada fracaso. No puedo olvidarme de mi hermano Jorge, quién abrió el camino en esta escuela, y del resto de mi familia que en ningún momento dudó de que llegase aquí. A mi abuela y a mi abuelo, a mis tías y a mis tíos, a mis primas y a mis primos, y a aquellos que ya no estáis aquí, también sois parte de esto.

No puedo dejar de mencionar a mi otra familia, esa que se elige. Tanto a los de fuera como a los de dentro. Mis compañeros de carrera, de fatigas, de sufrir exámenes y no sufrir tanto las noches. A los de siempre, por no haberme presionado nunca para que llegase aquí, se que deseáis esto más que yo. Va por vosotros!

Y a vosotros, los romanos, a los que no ya no distingo si sois compañeros, amigos o familia, hemos compartido el mejor año de nuestra vida, y eso no se olvida. A los cavalieri desde Principe Eugenio hasta Garbatella, gracias por tantos momentos, tantas risas y sobretodo, tanto fútbol. Y si, también a ti Jaime, el fratello que llegó un año tarde.

A los que habéis aportado vuestro granito de arena a este proyecto, no puedo olvidarme de vosotros. A la familia Lopez Morales, por acogerme como a uno más, en especial a Edu y Elena por enseñarme y marcarme el camino, gracias de corazón. A Juan Marín, por su arte, espero que tu obra quede reflejada. A Eduardo Boemo, por brindarme la oportunidad de realizar este proyecto, y ayudarme durante su desarrollo.

Por último quiero agradecer a mi novia, quien ha puesto voz al proyecto y me ha puesto cabeza a mi. Me has acompañado de cerca durante todos estos meses, aguantado y apoyando incondicionalmente. Has estado en los problemas y en has estado en las soluciones. Has conseguido que esta etapa, presuntamente dura, no sólo haya sido mucho más ligera, sino que la has hecho divertida. Gracias por ser y gracias por estar.

Índice de contenido

Índice de Ilustraciones

Índice de tablas

1. Introducción

1.1 Motivación

Con el aumento constante del uso de Smartphones, siendo estos casi imprescindibles en nuestro actual modo de vida y estando en cada bolsillo de cualquier persona, nos llega la idea de controlar dispositivos electrónicos cotidianos a través de ellos.

1.2 Objetivos

Se busca diseñar y realizar un sistema de varios actuadores electrónicos (**circuito remoto**), que a través de un dispositivo móvil Android (**Smartphone**) se accionen bajo demanda del usuario de forma independiente.

El puente entre los actuadores y el Smartphone será un microcontrolador electrónico (tipo Arduino) a elegir tras un estudio dedicado a ello.
Dicho microcontrolador se comunicará con el Smartphone mediante tecnología inalámbrica de corta distancia (WiFi, Bluetooth o Zigbee) a elegir también tras su respectivo estudio.

Se pretende la realización un sistema con una interfaz intuible y de uso sencillo para enfocarlo a un público de lo más amplio posible, intentando llegar incluso al sector invidente. Por eso se busca diseñar tanto una aplicación móvil para el sector de población vidente (**DOMUS**) como otra para el invidente (**CAECUS**).

También se quiere crear una maqueta demostrativa para darle un aspecto visual al proyecto y poder interactuar con ella a modo de prueba física.

1.3 Organización de la memoria

La memoria consta de los siguientes capítulos:

1. Introducción

2. Estado del arte

3. Diseño del proyecto

4.Desarrollo del proyecto

5. Integración, pruebas y resultados

6. Conclusiones y trabajo futuro

2. Estado del arte

En este punto serán comparadas y contrastadas características y especificaciones técnicas tanto los microcontroladores disponibles en el mercado, como de los diferentes protocolos de comunicaciones inalámbricas actuales, con el fin de encontrar y elegir lo más optimo y ajustado a las necesidades del proyecto.

Se analizará también el estado tecnológico actual de los distintos elementos del proyecto, así como sistemas similares al que se pretende realizar, observando sus características y diferencias con este proyecto.

2.1 Protocolos de comunicación Wireless de corta distancia

Existen multitud de tecnologías inalámbricas, sería difícil elegir la mejor, pero si nos centramos en las características que buscamos el abanico de elecciones se hace más pequeño.

Para los intereses de este proyecto se está buscando una tecnología de cortas distancias ya que será usada en hogares, y de forma presencial, sin necesidad de un intercambio masivo de datos ni de velocidades muy altas.

Centrados en estos objetivos, algunas de las disponibles son ANT+, Bluetooth, Infrared, ISA100a, Wi-Fi o Zigbee. Algunas como los infrarrojos ofrecen ventajas como alta velocidad pero son para distancias muy pequeñas y requieren de visión punto a punto, otras como ANT+ o ISA100a no están muy extendidas.

Serán puestas a estudio las tecnologías Wi-Fi, Zigbee y Bluetooth por su extensión, aplicaciones, características, disponibilidad y acceso a ellas. Se tendrán en cuenta aspectos técnicos como: velocidad, alcance y consumo.

2.1.1 Wi-Fi

Wi-Fi es el nombre comercial de la tecnología inalámbrica definido por los estándares IEEE 802.11. Es con diferencia la tecnología inalámbrica más extendida.

Wi-Fi opera en la banda de 2.4 GHz también conocida como ISM (Industrial Scientific and Medical) y en actualmente también en la de 5 GHz , en la que, si es cierto que tiene menor alcance por ser una frecuencia más alta, no hay posibilidad de interferencias con otras tecnologías microondas, ya que no operan en esta banda.

En cuanto a las características técnicas del wifi hay que tener en cuenta que tiene un ancho de banda de 1Gbps, una seguridad moderada, alcance de 100 metros (hasta 300 en condiciones óptimas) y un consumo de energía elevado. Además tarda de media 4 segundos en unirse a una red.

2.1.2 ZigBee

ZigBee es un protocolo de red de malla diseñado para transportar pequeños paquetes de datos en distancias cortas, manteniendo un consumo bajo de energía así como una velocidad de transmisión baja.

Opera también en la banda de 2.4 GHz de modo global, además de 900 MHz en Estados Unidos y 868 MHz en Europa. El método de modulación empleado es distinto según la banda que empleemos para la primera emplea modulación O-QPSK, mientras que para las dos últimas emplea modulación BPSK.

Si comparamos sus características, ZigBee cuenta con un ancho de banda de 250 kbps, alcance entre 10 y 70 metros y un consumo especialmente bajo de energía, así como posibilidad de funcionar en redes con topología estrella, malla, ad-hoc y punto a punto. Tiene además una tiempo medio de 30ms de conexión a una red. Convirtiendo así al ZigBee en una tecnología a considerar seriamente para la implementación de nuestro proyecto.

2.1.3 Bluetooth

Bluetooth es otra tecnología de tipo PAN (Personal Area Network), inicialmente creada por Ericsson para uso en auriculares inalámbricos para móviles, ha ido ganando popularidad con los años y actualmente se encuentra en prácticamente todos los dispositivos celulares del mercado.

Opera a 2.4 GHz de modo global, donde comparte banda con las dos tecnologías anteriores y otras muchas más, además la modulación usada por esta tecnología es la GFSK.

Pensado para conexiones PC-móvil y accesorios para dichos dispositivos, cuenta con un ancho de banda de 1Mbps, alcance de entre 10 y 30 metros y un consumo de energía medio (bajo, en sus versiones low energy). El tiempo de conexión del Bluetooth ronda los 3 segundos.

Siendo las características del Bluetooth bastante óptimas para el sistema, y sobre todo por la presencia de esta tecnología en prácticamente todos los dispositivos móviles actuales, **será la elegida para implementarse en el proyecto** y ser la encargada de la comunicación inalámbrica entre móvil y microcontrolador.

2.2 Microcontroladores y módulo Bluetooth

La elección de la tarjeta microcontroladora también será puesta bajo estudio, se estudiarán y se elegirá la que más se ajuste a las necesidades del proyecto, optimizando recursos y precio, para no elevar el coste del producto así como el tipo de alimentación requerido, ya que ésta irá estará integrada en la maqueta.

Puesto que no existen tarjetas microcontroladoras comerciales que incluyan tecnología Bluetooth de base, y con el fin de reducir gastos y espacio, también se buscará un módulo Bluetooth independiente, compatible con la tarjeta que se escoja.

El trabajo del microcontrolador será recibir e interpretar las órdenes enviadas por el Smartphone a través de Bluetooth. Así como encargarse de accionar los actuadores del circuito remoto en función de dichas órdenes.

Las tarjetas microcontroladoras más extendidas en el mercado y más versátiles son Arduino, Raspberry Pi y BeagleBone.

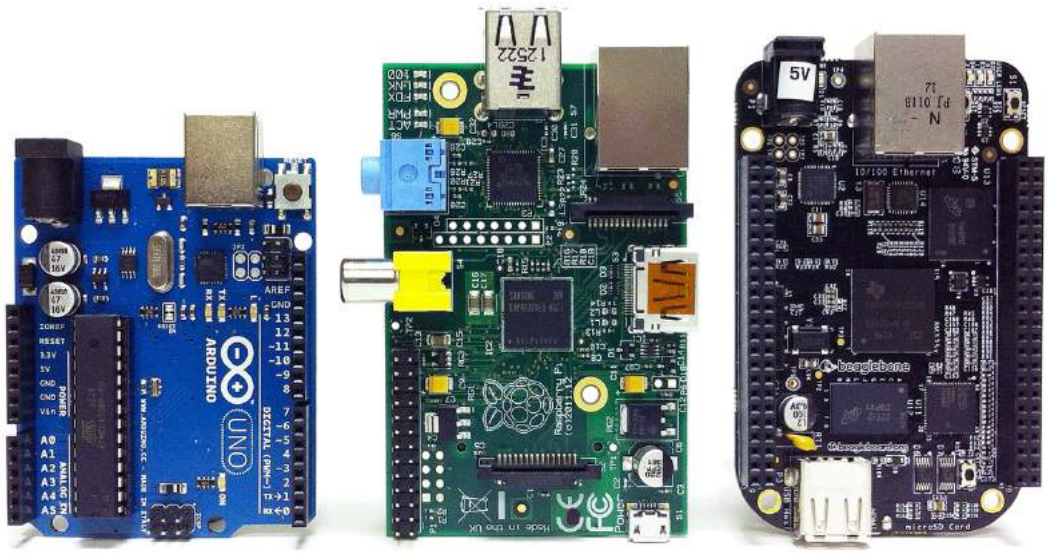


Ilustración 1: Arduino, Raspberry pi y BeagleBone

Seguidamente serán puestas bajo estudio, teniendo en cuenta factores como el rendimiento, el tamaño de la tarjeta microcontroladora, su consumo, especificaciones técnicas y precio.

2.2.1 Raspberry Pi

Se trata de un microcontrolador que corre el sistema operativo Linux desde una tarjeta SD introducida en la tarjeta.

Raspberry Pi es, en resumen, una pequeña computadora con el tamaño de una tarjeta de crédito. Al ser un miniordenador, puede correr en sistemas operativos como Linux o Android y podemos utilizarlo para desarrollar aplicaciones bastante más complejas que con otros microcontroladores, ya que puede utilizar lenguajes de programación de alto nivel como Python, C++ y Java.

Cuenta con las siguientes características técnicas:

- Velocidad: 700 MHz.
- Memoria RAM: 512 MB.
- Memoria Flash: según tarjeta SD insertada.
- Número de pines entrada/salida : 8 digitales.
- Consumo medio: 3W.
- Alimentación: 5V.
- Extras: salida de video HDMI, salida de audio 3.5mm y puertos USB y Ethernet.
- Precio medio: 35 € aprox.

2.2.2 BeagleBone

BeagleBone es un microcontrolador de altas prestaciones. Tiene el sistema Linux Angstrom instalado de serie, por lo que al igual que Raspberry Pi, se puede utilizar como un ordenador independiente si se desea.

Similar a Raspberry Pi pero más potente, cuenta con un procesador mayor, más memoria y un numero de entradas/salidas mucho mayor. Pensado para usar como ordenador en proyectos grandes que requieran de una capacidad de procesamiento y tareas simultáneas mucho mayores que las necesidades de nuestro sistema domótico.

Sus características técnicas son:

- Velocidad: 1 GHz.
- Memoria RAM: 512 MB.
- Memoria Flash: según tarjeta SD insertada.
- Número de pines entrada/salida : 69 digitales y 7 analógicos.
- Consumo medio: 0.85 W.
- Alimentación: 5V.
- Extras: salida de video HDMI, salida de audio 3.5mm y puertos USB y Ethernet.
- Precio medio: 90 € aprox.

2.2.3 Arduino

El Arduino es una plataforma de código abierto y versátil para el desarrollo de productos electrónicos, además consume muy poca potencia. Su principal diferencia con las dos tarjetas anteriores es que Arduino no puede ejecutar un sistema operativo y cuenta con un entorno de desarrollo propio llamado Arduino tool.

Se trata del más popular y existe una gran comunidad de usuarios, por lo que encontrar información sobre el también resultará más sencillo. Además cuenta con una gran cantidad de gadgets y Shields (complementos conectables que aumentan o aportan nuevas funcionalidades) que pueden proporcionarle tecnología como Ethernet o GSM.

En cuanto a características técnicas:

- Velocidad: 16 MHz.
- Memoria RAM: 2 KB
- Memoria Flash: 32 KB.
- Número de pines entrada/salida : 14 digitales y 6 analógicos.
- Consumo medio: 0.3 W.
- Alimentación: 7-12 V.
- Extras: memoria EEPROM 1 KB,.
- Precio medio: 20 € aprox.

Se sitúa con estas cualidades como la tarjeta idónea para automatización del hogar y por ende, para el proyecto. **Éste será el microcontrolador escogido**. Concretamente el modelo **Arduino UNO R3 smd edition**, un modelo básico, pequeño y compacto que satisfará las necesidades del proyecto.

2.2.4 Módulo Bluetooth

Dado que la tarjeta Arduino no dispone de Bluetooth, para que pueda contar con ésta tecnología tenemos varias opciones.

- Adquirir un shield Bluetooth para Arduino. Los shields son tarjetas compatibles con Arduino que se montan encima de el y le proporcionan funciones que antes no tenía.
- Conseguir un módulo Bluetooth independiente, compatible con Arduino. Los módulos Bluetooth son tarjetas diminutas que conectadas a un controlador dotan a éste de su tecnología.

Bajo el interés de reducir el tamaño y coste, se toma la elección de adquirir un módulo Bluetooth programable.

Se escogen los modelos HC-05 y HC-06, un módulo Bluetooth de tipo máster y el otro tipo esclavo, ambos de dimensiones muy reducidas (28x15x2.35 mm) y 1 g de peso,

que pueden ser alimentados y conectados directamente a través de los pines del Arduino.
Son capaces de transmitir datos a velocidades de 2.5 Mbps y cuentan además con un una luz LED informativa del estado actual del módulo mediante distintos patrones de intermitencia.

Se realizarán pruebas y entrenamiento con ellos (ver anexo A) para llegar a la decisión de que **el módulo que irá embebido en el proyecto, junto al Arduino, será el HC-06.**

Ilustración 2: módulo Bluetooth HC-06

2.3 Actuadores

Para demostrar la eficacia del sistema y demostrar su funcionamiento se elegirán una serie de actuadores que serán activados a voluntad del usuario y estarán incluidos en la maqueta a modo de ejemplo. Se valora el tamaño, eficacia y alimentación requerida, debido a que el sistema será alimentado desde una misma fuente de alimentación (necesitamos que se accionen al mismo voltaje).
Además se requiere que el estado de reposo de los actuadores sea tipo cerrado, para evitar malfuncionamiento del sistema en caso de corte de luz o fallo de alimentación.

Dado que el carácter del proyecto es demostrativo los actuadores elegidos serán de tamaño acorde a la maqueta, y sin reparar en su validez en un sistema implantado en una vivienda real.

2.3.1 Cerradura

El primer actuador a usar es un cerrojo, y simulará la cerradura de una puerta.
Se busca en tiendas electrónicas y se encuentra que el más útil y aplicable al proyecto es una cerradura de tipo solenoide con muelle. Este tipo de cerradura es de estado normal cerrado, pero cuando se le aplica corriente eléctrica, el émbolo es empujado hacia el interior del solenoide y la puerta o elemento que esté sujetando queda liberado.

2.3.2 Electroválvula

La electroválvula escogida para el proyecto debe ser de funcionamiento similar al de la cerradura, en estado normal se ha de encontrar cerrada y no dejar pasar fluidos de uno de sus extremos al otro, pero cuando se le suministra corriente eléctrica el émbolo que hace de tapón es empujado hacia dentro del solenoide, dejando así paso de fluidos de un lado al otro.

Simulará la misma función que una electroválvula en el hogar, ya sea para activar un riego automático o distribuir el circuito de calefacción.

2.3.3 Lámpara de LEDs

Para simular la luz en un cuarto, se combinarán una serie de LEDs, dispuestos en circuito paralelo creando una lámpara. Dicha lámpara podrá que ser encendida y apagada, contará con un regulador de intensidad luminosa a voluntad del usuario.
Se deberán de usarán LEDs de alta luminiscencia y colores claros para su implementación.

2.3.4 Base de enchufe (y módulo Relé)

Dado que el resto de actuadores son de corriente directa y para demostrar que el sistema también es capaz de accionar actuadores de corriente alterna, se implantará un enchufe de pared con toma de tierra, para que el usuario pueda accionar cualquier dispositivo de corriente alterna conectado a este. Para activar y desactivarlo usaremos un módulo relé de 5V, para que pueda ser accionado con el microcontrolador Arduino.

2.4 Sistemas de domótica por Smartphone actuales

Se presentarán los sistemas similares al que se va a desarrollar que se encuentren actualmente en el mercado. Además se realizará una pequeña conclusión detallando las ventajas y diferencias del sistema propio con el resto.

2.4.1 GOJI, KEVO y AUGUST

- **GOJI**

Goji es una cerradura digital inteligente para la puerta del hogar que funciona por conexión inalámbrica WI-Fi y Bluetooth.

Este dispositivo es capaz de mandar fotos de los visitantes en tu puerta a tiempo real, permitiendo que el propietario les de acceso temporal a tu casa a cualquier momento en el que no esté en casa. También lleva un registro de todas las actividades de la cerradura, sabiendo quién y cuando ha entrado o salido de casa.

Ilustracion 3: Cerradura electrónico GOJI

Para abrirse permite tanto el uso manual como unas llaves electrónicas tipo tarjeta. En el caso en que el usuario se quede encerrado dentro de casa o fuera de ella, existe un servicio 24h de atención al cliente que permite abrir la cerradura bajo demanda de un código secreto.

El **precio del sistema** *Goji*, incluyendo cerrojo, 3 SmartKeys y dos llaves es de **280 €**.

- **KÉVO**

El sistema *KÉVO*, perteneciente a la marca *Kwikset*, es un módulo de cerradura que funciona de una forma muy particular. Usando la tecnología Bluetooth, detecta cuando el Smartphone del usuario (o un dispositivo Bluetooth autónomo) está cerca, permitiendo que el cerrojo se abra con tan solo tocar la cerradura con la mano, gracias a tecnología capacitiva instalada en el pomo de la cerradura.

Para poder funcionar con el Smartphone necesita de la instalación previa de la aplicación móvil *KÉVO*, y ha de estar abierta en el momento de tocar la cerradura, para que la comunicación Bluetooth pueda llevarse a cabo. En caso de no disponer de un Smartphone, también se venden unos dispositivos Bluetooth autónomos, del tamaño de una llave de garaje, llamados *FOB* que harán la labor del Smartphone y sólo necesitas llevarlos encima.

Ilustracion 4: Cerradura electrónica Kévo

También cuenta con el servicio de *eKeys,* el cual trata de llaves virtuales que puedes enviar a quien quieras para otorgarle acceso a tu casa desde su móvil. Estas *eKeys* son limitadas, aunque pueden ser reasignadas y es un servicio que se paga individualmente.

Además, también cuenta con el modo de apertura clásico de llave física, por si hay un fallo de corriente, no cuentas con batería en el Smartphone o *FOB*.

El **precio del conjunto** cerradura *KÉVO, FOB* y dos *eKeys* es de **200 €**.

- **AUGUST**

AUGUST es un sistema de cerradura bastante completo. Cuenta con cámara externa para ver quién llama a tu puerta, conexión Wi-Fi y tecnología Smart Bluetooth 4.0.

La cerradura *AUGUST* puede funcionar tanto electrónicamente mediante la aplicación de Smartphone como mediante el método tradicional de llave física..

Cuenta con una pantalla LED retro-iluminada donde nos detallará información básica y nos recibirá al llegar a casa.

La gran ventaja de ésta cerradura es que el usuario puede acceder a ella remotamente a través de la aplicación, y disponer de todo tipo de información sobre lo que está ocurriendo en la entrada de su casa, desde recibir notificaciones instantáneas en tus dispositivos móviles de quien ha entrado y cuando lo hizo, hasta poder ver imagen y video en tiempo real de quien se encuentra llamando a su puerta.
Si bien son cualidades de una gran utilidad, todas están limitada a la necesidad de que la cerradura debe estar conectada vía Wi-Fi a Internet para poder disfrutar ellas.

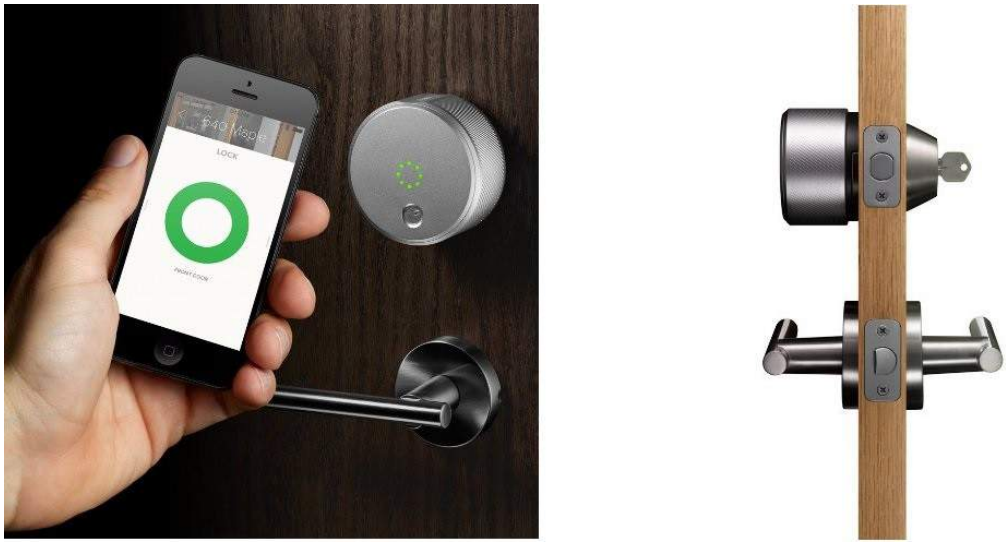


Ilustración 5: Cerradura electrónica AUGUST

Para la utilidad de abrir la puerta localmente sin embargo, nos puede valer tan sólo el Bluetooth, aunque debemos de tener la aplicación móvil instalada en nuestro Smartphone y éste ha de ser compatible con Smart Bluetooth 4.0.

El precio de este producto no varía mucho de sus competidores, pudiendo adquirirse un **kit básico** de una cerradura *AUGUST* y dos llaves físicas por un **valor de 220 €**.

2.4.2 Belkin WeMo

La empresa americana Belkin, ampliamente conocida por sus productos de conexión inalámbrica como routers y PWCs, también tiene un sistema domótico bastante popularizado llamado *WeMo.*

El sistema se basa en un módulo central que recoge la información del usuario a través de su Smartphone mediante tecnología Wi-Fi y la comparte al resto de módulos (todos con ésta tecnología incluida) por medio del mismo método.

Ilustración 6: Sistema de domótica WeMo

La gama de productos *WeMo* incluye diversos módulos, desde un dispositivo conectado que controla un enchufe de pared, hasta un sensor de movimiento que también cuenta con Wi-Fi, detectando e informando al sistema de cualquier cambio en el cuarto en que esté colocado. Todo ello es controlable desde una aplicación de iOS gratuita.

Además cuenta con la integración de hardware y software junto con **IFTTT**, que corresponden a las siglas de **IF-This-Then-That**, un servicio gratuito a través de internet que permite la definición de reglas que activen comportamientos o acciones en otras aplicaciones y la conexión de determinados dispositivos en el hogar. Lo que convierte a este sistema, a grandes rasgos, en programable.

Al ser un sistema controlable por WiFi su gran ventaja es que podemos conectarnos a la red sin necesidad de estar en el hogar y controlar el sistema distancia.

Los precios varían según los módulos que queramos adquirir:

- **Módulo central** (necesario) **90 €**
- Módulo enchufe 60 €
- Módulo interruptor 50 €
- Modulo luz y bombillas 80 €

2.4.3 SOMFY TaHoma

La firma *Somfy* cuenta con un sistema de domótica llamado *TaHoma* que reúne a la vez control sobre sostenibilidad energética, seguridad y aspectos de comodidad del hogar.

Con este sistema el usuario puede gestionar todos los equipos del hogar a través de una conexión a Internet, desde cualquier punto desde el que se encuentre. Dichos equipos pueden ser cámaras, alarmas, luces o incluso calefacción.

El sistema permite domotizar la casa del usuario paso a paso, según sus necesidades mediante el añadido de distintos dispositivos, ofreciendo la opción de controlar el hogar desde el Smartphone o tableta. Esto se lleva a cabo por medio de un receptor en casa y mediante una aplicación móvil con el mismo nombre que el sistema.

Además cuenta con la ventaja de ser compatible con una gama completa de motores, sensores, mandos y detectores en el mercado, de diferentes marcas

Ilustración 7: Sistema de domótica TaHoma

Por último, el sistema es programable, permite al usuario crear perfiles para gestionar diferentes equipos de forma simultánea, como el ejemplo de que un sensor detecte movimiento y encienda ciertas luces, o que en un día de fiesta la calefacción tenga una rutina distinta.

El **precio básico** para tener la unidad central que controla todo el proceso es de **400 €**, cantidad a la que debemos sumar el coste de cada uno de los accesorios.

2.4.4 Philips HUE

La gran compañía neerlandesa de electrónica *Philips* cuenta con su propio sistema de control de luces inteligente, llamado *HUE.*

Es un sistema de iluminación inalámbrico personal, diseñado para facilitar la vida en casa y adecuarse personalmente a cada usuario. Combina luz LED y eficiencia energética.

Ilustración 8: Sistema de iluminación Philips HUE

Éstas luces LED, controladas inalámbricamente por el usuario tienen capacidad de atenuarse, parpadear e incluso ciertas bombillas son capaces de emitir luz de cualquier color. Disponibles en distintas formas y tamaños hacen de ellas que puedan ser adaptables para cualquier hogar.

Philips HUE, sólo es compatible con bombillas de su misma marca, y de momento sólo están centrados en controlar la luz.

El **precio del Kit básico**, que incluye *Base HUE* y 3 lámparas *Philips HUE* es de **180 €.**

2.4.5 Ventajas del proyecto frente a los sistemas existentes

Observamos que los sistemas de domótica actuales son muy focalizados a determinados productos, y si bien existen sistemas más completos, estos obligan a comprar un módulo extra por cada necesidad nueva que el usuario quiera cubrir. Además saltan a la vista los altos precios que tienen todo este tipo de productos.

La mayoría de estos sistema basan sus servicios en la existencia previa de una red Wi-Fi con conexión a internet para poder funcionar. Significando que en el caso de perder ésta conexión, o de no tenerla instalada previamente se encuentran mermados en la mayoría de sus servicios, e incluso en algunos casos, totalmente anulados.

Por último, no se ha encontrado un sistema que ofrezca soluciones de domótica para el colectivo invidente o con baja visión, representando éste un porcentaje significativo de la población (y con previsiones de aumentar) cómo podemos observar en la siguiente tabla:

Estimaciones globales de discapacidad visual, por regiones de la OMS (miles)

	África	América	Europa del Este	Europa	Sudeste asiático	Oeste Pacífico
Población	672.238	852.551	502.823	877.886	1.590.832	1.717.536
Nº y Porcentaje (%) de personas ciegas	6.782 (1,01%)	2.419 (0,28%)	4.026 (0,80%)	2.732 (0,31%)	11.587 (0,72%)	9.312 (0,54%)
Nº y Porcentaje (%) de personas con baja visión	19.996 (2,97%)	13.116 (1,54%)	12.444 (2,47%)	12.789 (1,46%)	33.496 (2,11%)	32.481 (1,89%)
Nº Total y Porcentaje (%) Total de personas con discapacidad visual	26.778 (3,98%)	15.535 (1,82%)	16.469 (3,27%)	15.521 (1,77%)	45.083 (2,83%)	41.793 (2,43%)

Tabla 1: Distribución global de invidentes y discapacitados visuales

El sistema a realizar no sólo presenta un bajo coste de producción, sino que también presenta soluciones para el colectivo invidente.

Además, al funcionar mediante tecnología Bluetooth, no necesita de la existencia de una red Wi-Fi ni de internet, pudiendo ser instalado en cualquier lugar que cuente con electricidad, o incluso sin ella, pudiendo ser alimentado en casos puntuales con baterías externas.

3. Diseño del sistema

Los objetivos de este capítulo son definir los aspectos funcionales del proyecto.

Se indicará cómo trabajarán la aplicaciones, su aspecto y su actividad. También será detallado el resultado final que se espera de la maqueta demostradora y cómo ejecutará las órdenes que sean emitidas a través del usuario por medio de las aplicaciones DOMUS y CAECUS.

3.1 Requisitos del sistema

El sistema debe ser capaz de actuar los elementos electrónicos a través de un Smartphone, usando la tecnología Bluetooth y un equipo microcontrolador que se gestione las órdenes Bluetooth y las ejecute, y envíe feedback de nuevo al Smartphone.

Se pueden diferenciar fácilmente dos partes o subsistemas:

- **Circuito remoto**: éste incluye el microcontrolador Arduino, los circuitos que conectan Arduino con los actuadores y los actuadores en sí. Su tarea es recibir las órdenes Bluetooth provenientes del Smartphone, interpretarlas, llevar a cabo las tareas de encendido/apagado de actuadores y posteriormente enviar información de vuelta al Smartphone.

- **Smartphone**: se trata de las dos aplicaciones que funcionan sobre un terminal móvil conectado mediante Bluetooth al circuito remoto. Éstas reciben ordenes del usuario mediante botones (en la versión DOMUS) o gestos en pantalla (en la versión para invidentes CAECUS) y las envían al circuito remoto, el cual les responderá por la misma vía un mensaje de información (**feedback**) confirmando el proceso de la orden, o en caso contrario, un error.

Ambos subsistemas funcionan en tiempo real, mostrando los resultados de las órdenes mediante el prendido de LEDs en el circuito remoto o mediante imágenes, mensajes en pantalla y mensajes de audio, en el Smartphone.

3.2 Diseño del circuito remoto

En este apartado se definirá el diseño del circuito, los componentes electrónicos usados así como las conexiones con los actuadores, la fuente de alimentación y la caja encerradora del Arduino. También se incluirá una gestión de errores y previsión de fallos.

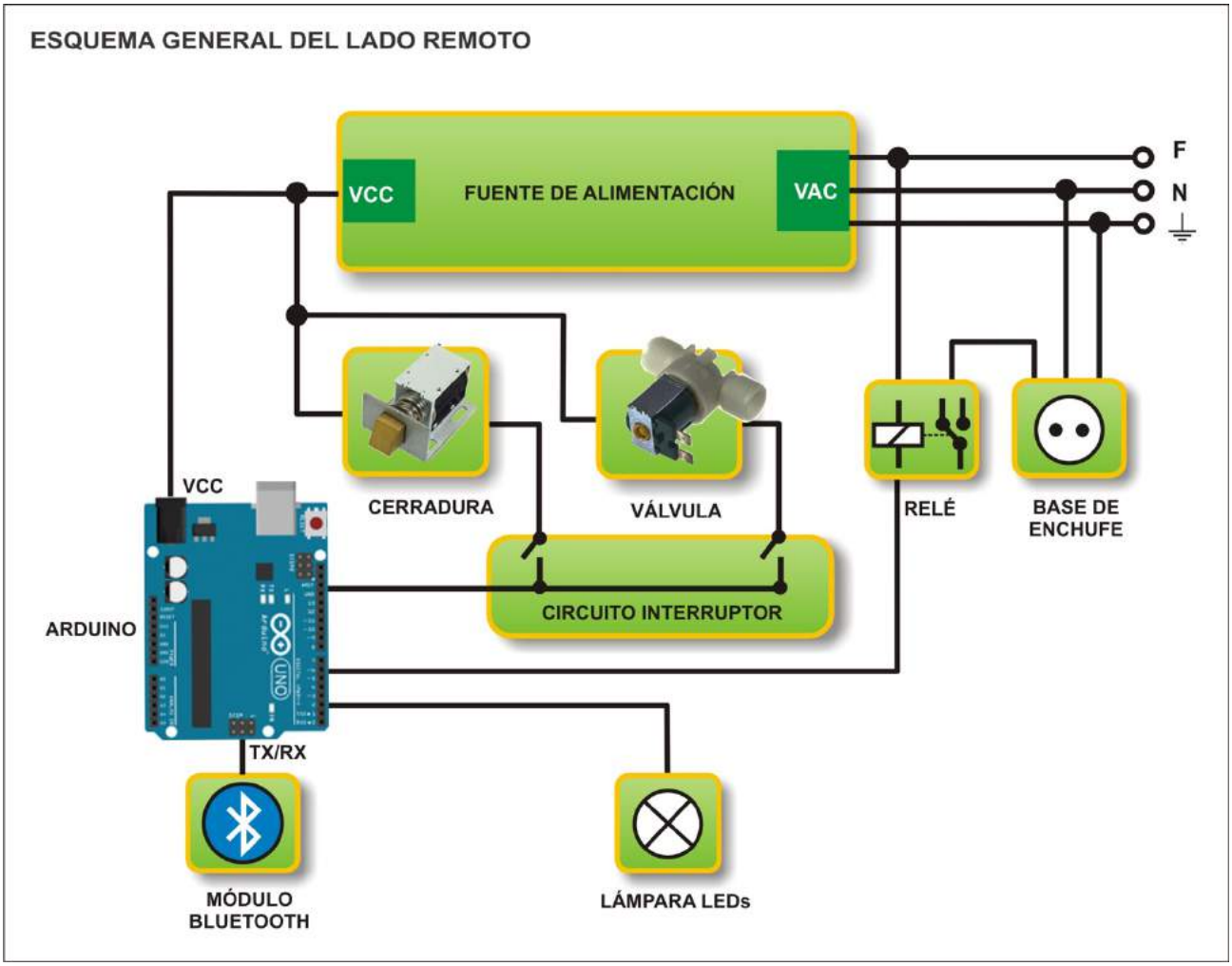


Ilustración 9: Esquema general del circuito remoto

Los componentes del lado remoto son:

- Arduino y módulo Bluetooth
- Circuito interruptor
- Caja de instrumentos con raíl DIN
- Relé
- Base de enchufe
- Actuadores
- Fuente de Alimentación
- Toma de corriente general

3.2.1 Componentes del circuito remoto

A continuación analizaremos con más detalle cada uno de los elementos que componen el circuito remoto y que estarán implementados en la maqueta. Se tiene en cuenta que los actuadores elegidos han sido pensados en escala para ser incluidos en una maqueta demostradora, por lo que aunque funcionales, en una situación de hogar real se incluirían unos de mejores características.

3.2.1.1 Arduino y módulo Bluetooth

El Arduino junto al módulo Bluetooth y el circuito interruptor estarán encerrados dentro una caja modular de tipo rail DIN, con el fin de proteger y aislar la parte delicada del proyecto.

El microcontrolador Arduino se alimentará con una fuente de alimentación de 10 V, lo que entra dentro del rango que acepta el Arduino, ya que cuenta con un regulador de tensión interno.

El Arduino conectará de forma directa con el relé que activa el enchufe, así como con la lámpara de LEDs y los LEDs informativos, ya que sus pines de salida son capaces de suministrar la suficiente corriente como para accionar estos elementos independientemente. Sin embargo, para el caso de los actuadores de tipo solenoide (válvula y cerradura) se queda muy lejos de poder activarlos y lo hará a través del circuito interruptor (al que irá obviamente, conectado) y la fuente de alimentación.

El módulo Bluetooth irá conectado directamente a través de sus pines al Arduino, proporcionándole así al segundo la capacidad de recibir datos de manera inalámbrica.

3.2.1.2 Diseño del circuito interruptor

Llamamos circuito interruptor al conjunto de elementos usados para controlar los actuadores que se accionan a través de la fuente de alimentación bajo las órdenes del microcontrolador.

El circuito, que se encuentra duplicado puesto que cada actuador usa uno independiente, está formado por:

- Un **MOSFET** (transistor de efecto de campo metal-óxido-semiconductor) y su correspondiente disipador de calor para evitar subidas de temperatura perjudiciales. Conectado a la fuente, actuador y gobernado por el Arduino, para que cuando éste lo active, el MOSFET deje pasar la corriente desde la fuente hasta el actuador, activándolo.

- Un disipador de calor para el MOSFET, para evitar subidas de temperaturas que puedan dañar los componentes electrónicos.

- Una resistencia de bajo valor (20 Ohm) conectada a la puerta del MOSFET para limitar la corriente que el Arduino le entrega.

- Un diodo de protección, polarizado en inversa y en paralelo a los actuadores, ya que al ser estos solenoides, al conmutar el circuito liberarán la corriente en sentido inverso y podría dañar nuestro circuito.

- Un LED informativo conectado en paralelo al actuador, para que confirme la activación/desactivación de estos en aquellos casos donde el actuador no sea visible.

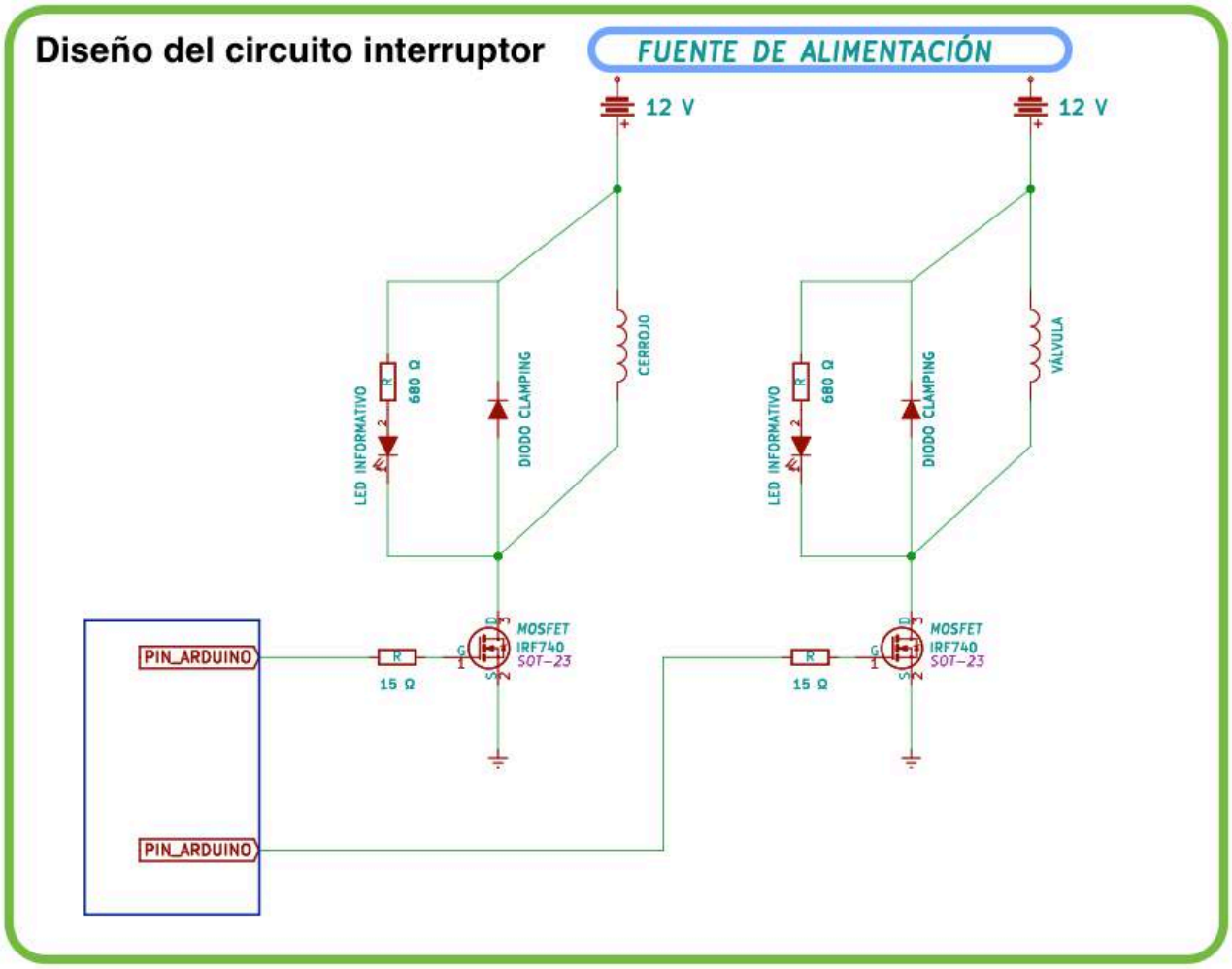


Ilustración 10: Esquemático del circuito interruptor

3.2.1.3 Caja de instrumentos con raíl DIN

Dada la naturaleza doméstica del proyecto, microcontrolador y circuitos deben ir encerrados y compactos en una caja modular que tan solo permita acceso a los puertos de alimentación y datos del microcontrolador, y salida a los cables.

Se escoge como una solución útil, práctica y elegante una caja de policarbonato preparada para sujeción con raíl DIN en su parte posterior y pestañas perforadas para la salida de cables.

Se escoge como una solución útil, práctica y elegante una caja de perfil bajo hecha de policarbonato, que cuenta con una cubierta de panel opaca y una selección de protectores de terminal resistentes y perforados que aprovecharemos para pasar los cables que tienen que salir del Arduino y del circuito interruptor hacia el resto del circuito sistema.

Para la integración en superficies la caja cuenta con carril DIN en la parte posterior.

Será necesaria la perforación de dos orificios laterales habilitando las entradas de alimentación y USB del Arduino, además de un tercero en la superficie de la caja para tener acceso visual al LED informativo del módulo Bluetooth.

3.2.1.4 Relé

A diferencia del resto del sistema, el relé trabajará con dos tipos de corriente, ya que conmutará corriente alterna y estará gobernado por corriente continua. Por seguridad se elige colocarlo fuera de la caja DIN, para evitar que un mal contacto de la corriente alterna pueda dañar cualquiera de los elementos electrónicos situados dentro.

El relé será el conmutador, bajo órdenes del Arduino, de la fase de la toma de corriente general que se conecta a la base de enchufe.

Cuando el Arduino envié la señal que activa el relé, este conmutará la fase de la toma de corriente (que se encontrará en estado normal abierta) cerrando el circuito y proporcionándole operatividad a la base del enchufe.

3.2.1.5 Base de enchufe

La base de enchufe seleccionada admite una intensidad de trabajo de hasta 16 A, tiene toma de tierra y además cuenta con raíl DIN en su parte posterior, siendo fácilmente integrable en el proyecto.

La base estará alimentada por la toma de corriente general, al igual que la fuente de alimentación, a diferencia que el estado normal de la base será inactivo, ya que su cable de fase está conmutado por el relé que gobierna el Arduino, y solo estará activa cuando el usuario envíe la orden.

3.2.1.6 Actuadores

Los actuadores serán activados independientemente y bajo demanda del usuario, y más bajo nivel, del Arduino. Podemos distinguir dos tipos, los directos y los indirectos.

- Actuadores directos: se conectan directamente al microcontrolador y éste es capaz de activarlos sin necesidad de la fuente de alimentación. Se trata de la lámpara de LEDs y el relé.

- Actuadores indirectos: se conectan a través del circuito interruptor (MOSFET) con la fuente de alimentación y con el Arduino, se trata de la cerradura y la electroválvula.

3.2.1.7 Fuente de alimentación

La fuente se encargará de alimentar tanto al microcontrolador Arduino como a los actuadores indirectos y LEDs informativos. En definitiva, es la encargada de repartir corriente continua a todo el sistema.

El principal criterio a la hora de elegir fuente era el voltaje nominal que tuviese de salida, para que fuese capaz de activar los distintos actuadores. Se ha elegido una fuente con salida de 12V y 2A, suficiente para activar todos elementos del sistema. Además la fuente cuenta con carril DIN, lo que simplifica su integración en la maqueta.

Dicha fuente también debe estar alimentada, y esa es la función de la toma de corriente general.

3.2.1.8 Toma de corriente general

Para activar la fuente necesitamos conectarla a una toma de corriente alterna de 220V, para ello se ha escogido un cable con enchufe y toma tierra, para aumentar la seguridad del circuito y poder añadirle la toma de tierra a la base de enchufe.

Como añadido, se ha instalado un interruptor en dicho cable, y así poder apagar la fuente de alimentación sin necesidad de desenchufar la toma de corriente general cada vez que se desee apagar totalmente la maqueta.

3.2.3 Gestión de errores y prevención de fallos en el circuito remoto

Errores en fallo de alimentación

Dado que el sistema es completamente eléctrico, en caso de pérdida de la alimentación, ha de reaccionar de manera segura, por eso todos los actuadores han de ser de estado natural (sin alimentación) cerrado.

Siendo el caso más notable el de la puerta, por el gran problema que supondría dejarla abierta si hay un fallo de luz. Aunque la cerradura implementada en la maqueta demostradora no tiene una método manual de apertura, en un caso real un fallo de luz no bloquearía la puerta permanentemente, ya que también podría abrirse por el medio habitual, la llave.

Errores de temperatura del circuito interruptor

Debido a la temperatura que alcanzan los MOSFET del circuito interruptor, se decide que se implementarán disipadores de calor para rebajar su temperatura y evitar posibles daños y malfuncionamiento del sistema.

3.3 Diseño de las aplicaciones Android

A continuación se definirá el diseño de las dos aplicaciones móviles, el sistema operativo objetivo, el entorno de desarrollo escogido y las herramientas usadas.

También se establecerán los objetivos a cubrir y la gestión de posibles errores que habrá que evitar.

3.3.1 Bases de las aplicaciones

A la hora de desarrollar una aplicación en Android es importante señalar la versión mínima de sistema operativo a la que va dirigida (minimum required SDK), permitiéndote así el entorno de desarrollo usar más o menos recursos durante su programación, y definir la versión mínima que ha de poseer un usuario para poder correr la app en su terminal.

La importancia de ésta elección radica en el balance de nuevos recursos a ser utilizados en la aplicación contra el número de usuarios que serán capaces de disfrutar de ella, dado que no todos ellos cuentan con la última versión de Android.

Para lograr abarcar un mayor número de usuarios y dispositivos, se ha escogido como versión mínima requerida la “Ice Cream Sandwich 4.0.4”, logrando así un rango de utilidad en dispositivos actuales de casi el 98% y sin realmente haber perdido operatividad potencial, ya que los recursos disponibles en esa versión son más que suficientes para el correcto desarrollo y funcionamiento de las dos aplicaciones.

Para desarrollar ambas aplicaciones se ha elegido el entorno de desarrollo propio de google llamado Android Studio, ampliamente recomendado por todos los portales de desarrollo Android.

Dado que se van a desarrollar dos aplicaciones les otorgaremos nombres distintos para diferenciarlas claramente, así como iconos representativos de cada una.
Para la aplicación de domótica simple se optó por el nombre DOMUS por su significado ("casa" en latín) y su sencillez. Por otro lado, a la aplicación destinada a domótica para personas invidentes el nombre escogido fue CAECUS, a imagen de su aplicación "hermana" predominando la sencillez y significado ("ciego" en latín).

Basaremos el diseño de las aplicaciones en el **Human Centered Design**, (HCD) una metodología de diseño que desarrolla soluciones mediante la participación de la perspectiva humana durante las etapas del proceso resolutivo de problemas, obteniendo así una aplicación tan robusta frente a errores como intuitiva para el usuario.

Para definir el diseño de ambas aplicaciones, trataremos de explicar la interfaz, los requisitos de funcionalidad, los errores a evitar y las soluciones aplicadas desde un punto de vista general, explicándose en detalles técnicos en el capítulo siguiente (Desarrollo del sistema).

3.3.2 Aplicación DOMUS

Esta aplicación de domótica está dirigida a usuarios con capacidad visual, proporcionándole las facilidades de activar una serie de actuadores de su casa desde una misma pantalla en su Smartphone, a través de una interfaz intuitiva con botones e imágenes representativas.

Para definir el funcionamiento de DOMUS dividiremos la aplicación en sus únicas dos pantallas:

Ilustración 21: Logo de DOMUS

- **Pantalla de elección de dispositivo**: lista de dispositivos Bluetooth enlazados al terminal del usuario, en el que permite elegir a cual conectarse.

- **Pantalla de control de actuadores**: interfaz gráfica con botones e imágenes donde el usuario activa/desactiva los actuadores a su elección.

3.3.2.1 Pantalla de elección de dispositivo

Se trata de la pantalla inicial al abrir la aplicación, nos mostrará todos los dispositivos Bluetooth con los que previamente se ha enlazado nuestro Smartphone. Recordamos que la tecnología Bluetooth exige que los dispositivos estén enlazados antes de realizar una conexión y comenzar una transferencia de datos.

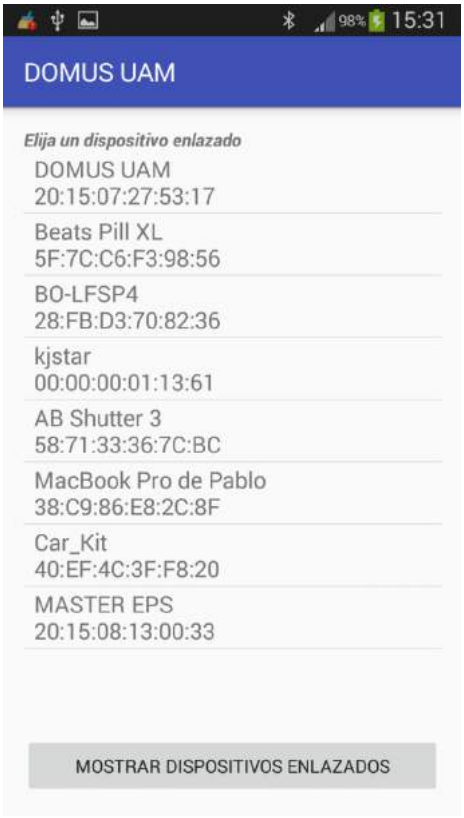


Ilustración 12: Pantalla de elección de dispositivo (DOMUS)

3.3.2.1.1 Requisitos de la pantalla de elección de dispositivo

La pantalla debe mostrar al usuario todos los dispositivos con los que su Smartphone se encuentre enlazado, por lo que si el número de estos fuera mayor del espacio en pantalla, es necesario implementar un scroll o deslizante vertical de pantalla para que pueda ser capaz de visionar y elegir todos y cada uno de ellos.

Si el usuario se equivoca al elegir uno debe poder ser capaz de regresar a esta pantalla y cambiar su elección, por lo que se implementará un botón de desconexión que regresará a esta pantalla de elección de dispositivo.
El usuario ha de tener algún tipo de información para saber si ha presionado un botón y si se está realizando la conexión, para que no dude sobre si ha presionado correctamente o ha dejado de funcionar la aplicación. Aplicaremos sombras en los dispositivos que dejen evidencia cuando se ha presionado uno, además se implementará un cartel informativo que saltará en pantalla, informando de que el proceso de conexión al dispositivo Bluetooth escogido ya ha comenzado.

3.3.2.1.2 Funcionamiento de la pantalla de elección de dispositivo

Al iniciar la pantalla esta se encuentra vacía y sólo encontramos un botón en la base de la pantalla con título "Mostrar dispositivos enlazados", al tocarlo aparecerá el listado de todos los dispositivos enlazados, ordenados verticalmente y representados por su nombre seguido de la dirección MAC de cada uno.
Es entonces cuando la pantalla permite al usuario conectarse al dispositivo que desee presionando sobre su nombre. En el caso de la maqueta el usuario debe conectarse al módulo Bluetooth que se ha instalado junto al Arduino, al que se le fijó previamente el nombre como "**DOMUS UAM**".

Al elegir un dispositivo de la lista, comenzará el proceso de conexión al Bluetooth escogido y accediendo automáticamente a la segunda pantalla de la aplicación, estando esta bloqueada mediante un cartel informativo que, bajo el texto "Conectando..", no permitirá accionar ningún botón hasta que la conexión se haya completado.

Ilustración 13: Pantalla bloqueada por mensaje de conexión (DOMUS)

Cuando la conexión se haya completado, el mensaje informativo desaparecerá y se podrá comenzar a utilizar la aplicación.

3.3.2.2 Pantalla de control de actuadores

Esta pantalla permitirá al usuario activar o desactivar los actuadores disponibles mediante el presionado una serie de botones situados intuitivamente junto a una imagen del actuador. Contará además con feedback informativo para que el usuario sea consciente de que la orden enviada ha sido recibida, así como del estado actual de cada actuador.

Ilustración 14: Control de actuadores en DOMUS, actuadores apagados (Izq.) y encendidos (dcha.)

3.3.2.2.1 Requisitos de la pantalla de control de actuadores

Al igual que la pantalla inicial, debe ser adaptable a todos los dispositivos independientemente de su tamaño, y todos los actuadores deben poder aparecer en pantalla, por lo que también se añadirá una barra de deslizamiento vertical.

Debe ser sencilla e intuitiva, por lo que se optará por un diseño modular, igual para todos los actuadores basado en dos botones para encendido y apagado, seguido de una imagen de estado, que representa al actuador y su estado actual.

El diseño modular permitirá la futura incorporación de más actuadores sin que resulte engorroso o desencaje con el diseño anterior de la pantalla.

La interfaz de usuario debe además informar al usuario de que la aplicación está funcionando correctamente, así como dar confirmación de que las órdenes enviadas por el usuario han llegado al circuito remoto y se han podido ejecutar de manera adecuada. Se implementarán para ello sombras en los botones, que confirmen cuando han sido presionados.

Además, se va a establecer un **sistema de feedback** mediante el cual nuestra aplicación, tras enviar una orden, espera la respuesta del otro extremo de la conexión (Arduino) confirmando su correcta recepción y comportamiento en consecuencia. Este feedback llega al usuario de modo visual mediante un mensaje informativo en pantalla, junto a una imagen que manifiesta el estado actual del actuador.

Concretamente en el caso del actuador "lámpara de LEDs", se debe dar la opción de regular la intensidad o brillo de la luz. También debe evitarse que si se usa la barra con el actuador desactivado, no ocurra nada y la luz se mantenga apagada. Se implementará bajo los botones de encendido y apagado una barra deslizante horizontal para que el usuario sea capaz controlar esta intensidad mediante el simple arrastre del dedo por ello. Además se añadirá un texto que indicará de manera numérica la intensidad actual a la que se haya establecida la luz.

El usuario debe ser capaz de desconectarse del dispositivo Bluetooth para dejarlo libre, en vista de que otro usuario quiera usarlo o que simplemente desee acabar con la conexión para, por ejemplo, ahorrar batería de su Smartphone. Se implementará para ello un botón siempre visible en la base de la pantalla que al ser pulsado comience el protocolo de desconexión y libere el dispositivo.

Por último, en caso de conexión al sistema, habiendo sido éste previamente encendido y utilizado, la aplicación ha de ser capaz de reconocer el estado actual de todos los dispositivos y mostrarlo por pantalla. Es decir, si por ejemplo la luz quedó encendida la última vez que un usuario usó el sistema, la aplicación debe reconocerlo al conectarse y actualizar las imágenes de estado del sistema. Para implementar esta función se creará un método que consistirá en una única consulta de la aplicación al Arduino en el mismo instante en que comiencen la conexión entre ambos, será llamado **informe inicial** y basado en sus resultados se realizará la primera actualización de estados en la pantalla.

3.3.2.2.2 Funcionamiento de la pantalla de control de actuadores

Se accederá a esta pantalla automáticamente tras haber presionado sobre un dispositivo Bluetooth en la pantalla anterior, y tras acabarse el protocolo de conexión, el usuario ya tendrá permisos para controlar los actuadores a su voluntad.

Presionando sobre los botones de cada actuador, se podrá activarlos y desactivarlos, sin ningún retardo advertible desde el pulsado del botón y la actuación del elemento elegido. Acto seguido, la imagen de estado del actuador accionado, cambiará a su estado actual, debido al mensaje de confirmación (feedback) enviado por el Arduino a través del módulo Bluetooth remoto.

En el módulo de botones del actuador de la luz, existirá también una barra deslizante que nos permitirá controlar su brillo (únicamente cuando la luz se encuentre encendida).

Por último, dispondremos de un botón en la base de la pantalla que nos permitirá en cualquier momento desconectarnos del dispositivo Bluetooth, liberándolo así para que pueda usarlo otro usuario. Esto no hará sin embargo que los actuadores que se encuentren accionados dejen de estarlo, dotando de estabilidad al sistema.

3.3.3 Aplicación CAECUS

Ilustración 15: Logo de CAECUS

Esta aplicación de domótica está especialmente dirigida a personas invidentes o con capacidad visual requerida, y por ello se ha desarrollado de manera íntegra para que la vista no sea en necesaria en ningún momento para su uso. El método por el cual el usuario podrá accionar los actuadores será mediante el dibujado de gestos en pantalla.
La información y feedback para el usuario vienen dados a través del altavoz del Smartphone, con mensajes de voz.

Se dispondrá de la pantalla entera como superficie para dibujar gestos, ya que no se necesita espacio para botones y evitando así la posibilidad de fallo por dibujar fuera de rango.

La interfaz de CAECUS cuenta con una única pantalla en la que transcurre todo el ciclo de uso de la aplicación. Será llamada pantalla de control por gestos.

3.3.3.1 Pantalla de control por gestos

Es la única pantalla de la aplicación, permite al usuario el accionado de los actuadores mediante gestos específicos dibujados en la superficie de la pantalla. Se trata de una pantalla cuya total superficie está definida como zona detectora de gestos.

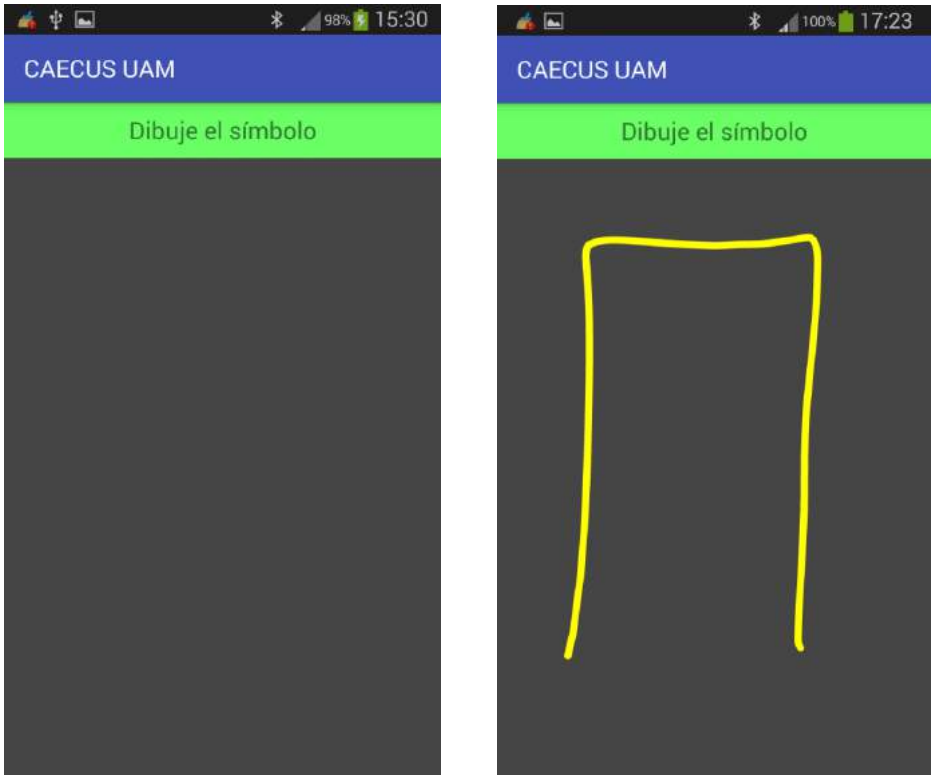


Ilustración 36: Control de actuadores en CAECUS, pantalla en reposo (izq.) y gesto dibujado (dcha.)

3.3.3.1.1 Requisitos de la pantalla de control por gestos

Los gestos han de ser simples e intuitivos, para que un usuario invidente pueda dibujarlos sin dificultad en pantalla y se sienta familiarizado con ellos. Se crea una pila de gestos sencillos realizables en un solo trazo, con formas que guardan relación con el actuador al que están enlazados, y fáciles de recordar. (Ver anexo C)

La aplicación ha de ser capaz de detectar los gestos, independientemente del tamaño con que sean dibujados y de pequeñas variaciones en su forma. Se creará un banco de gestos con múltiples versiones de cada uno, enseñando a la aplicación a reconocerlos aunque sufran alteraciones en figura o dimensión.

La aplicación debe estar íntegramente dedicada a personas invidentes, por lo que ha de carecer de botones o elemento en pantalla que requiera del uso de la vista.

Debe poder ser abierta sin necesidad de pulsar su icono en el escritorio de Android, se le ha dado un nombre fácil de detectar y pronunciar, para que pueda iniciarse desde la búsqueda por voz instalada de serie en todos los dispositivos Android.

La conexión al Bluetooth ha de ser inmediata y no puede depender de botones. Para ello se implementará la aplicación de modo que lo primero que realice al abrirse, sea tratar de conectarse al módulo Bluetooth remoto, en caso desfavorable debe informar del error cerrarse automáticamente.

La información de feedback o de cualquier otro tipo para el usuario, no puede llegar de modo visual, por lo que se implementan mensajes de audio que informan en todo momento del estado actual de la aplicación y actuadores, en tiempo real.

Dichos sonidos han de poder ser escuchados siempre, independientemente de si el dispositivo Android está en modo silencio. Se programará la aplicación de modo que el sonido dentro de ella sea independiente al establecido en ese momento en el Smartphone, consiguiendo así que siempre esté activado, aunque el dispositivo Android se encuentre silenciado.

Debido a la ausencia de botones, el modo de desconectar la aplicación del Bluetooth para que este quede liberado, será a través del botón físico home. Una vez el usuario lo pulse, automáticamente saldrá de la aplicación (como sucede en cualquier otra aplicación) y además el Bluetooth será desconectado en ese instante.

Para evitar que la puerta se quede abierta, por motivos de seguridad, si una vez abierto el cerrojo, el usuario no ejecuta la orden de desactivar el cerrojo, éste se cerrará automáticamente pasados diez segundos.

Debe existir una función rápida extra para el accionado del actuador cerrojo, que no dependa de un gesto en pantalla, para situaciones en las que la pantalla no reacciona (pantalla mojada, guantes en las manos..). Se establecerá un protocolo que detecte una agitación veloz del móvil para activar dicho actuador. Para evitar una apertura del actuador cerrojo al caerse el Smartphone al suelo, u otras agitaciones involuntarios, se establecerán unos parámetros de detección de movimiento que sólo activaran el actuador cerrojo en caso de movimientos veloces y con dirección vertical ascendente.

3.3.3.1.2 Funcionamiento de la pantalla de control por gestos

Al abrir la aplicación, se realizará automáticamente el intento de conexión al módulo Bluetooth remoto, informando mediante el altavoz del Smartphone cuando el proceso se haya concluido con éxito, o en caso contrario cerrando la aplicación, informando también de ello.

Al haberse terminado con éxito el proceso de conexión al módulo Bluetooth (2-3 segundos), se reproduce por audio un resumen del estado actual de todos los actuadores (**informe inicial**), al igual que ocurría en el caso de DOMUS con imágenes en pantalla. Una vez acabado el mensaje de audio, el usuario puede ya dibujar gestos (que ha debido aprender previamente) en cualquier punto de la pantalla, los cuales accionarán los distintos actuadores de manera independientemente y en tiempo real.

Cada vez que un gesto sea reconocido, la aplicación emitirá un mensaje de audio informativo, indicando la acción realizada y el estado en el que queda el actuador con esa orden.

Al realizar un movimiento energético con el Smartphone en dirección vertical ascendente, la aplicación lo detecta mediante el sensor de movimiento integrado en el dispositivo, y emite la orden de accionar el actuador cerrojo.

Los actuadores accionados mediante los gestos, mantendrán su estado siempre y cuando no se realice el gesto que los altere, ni aunque se pierda conexión o cerremos la aplicación; a excepción del actuador cerrojo, que en cualquier caso será siempre desactivado automáticamente después de 10 segundos desde su apertura.

En el momento que el usuario presione el botón físico “home” de su dispositivo, la aplicación se cerrará y se desconectará automáticamente del dispositivo Bluetooth.

3.3.4 Gestión de errores y prevención de fallos en Android

Error en cambio de orientación de pantalla involuntario

Tanto en el caso de DOMUS como sobretodo en el de CAECUS, el cambio de pantalla vertical a horizontal supone un inconveniente. Si bien en el caso de DOMUS el único inconveniente es que se ven menos actuadores en pantalla y hace obligatorio el uso del scroll para poder accionar todos, en el caso de CAECUS si conlleva un problema, ya que la persona invidente no tiene modo de conocer la orientación de la pantalla y podría dibujar los gestos con un giro de 90º, lo que dificultaría el reconocimiento de ellos por la aplicación. Para resolverlo se optará por bloquear el cambio automático de orientación, y se fijarán ambas aplicaciones con orientación vertical, sea cual sea la inclinación en la que se disponga el Smartphone.

Errores al accionar ciertos botones físicos (Home, Back y Lock Screen)

Una pieza clave de ambas aplicaciones es la conexión al módulo Bluetooth, siendo al primera acción que realiza CAECUS automáticamente, y en el caso de DOMUS similar, con la excepción de que es el usuario el que ejecuta la orden de conexión.
Dado que el módulo Bluetooth al que debe conectarse es de tipo esclavo (ver anexo A) sólo permite una conexión simultánea, lo que haría imposible a la aplicación conectarse si está ocupado en ese momento.
El problema surge, una vez ya conectados al módulo Bluetooth, al salir de la aplicación y tratar de volver a entrar, vemos los casos de DOMUS y CAECUS por separado:

- **DOMUS**: al volver a abrir la aplicación después de haberla cerrado, aparecemos en la pantalla de selección de dispositivo, y si tratamos de conectarnos al módulo Bluetooth remoto descubriremos que da error, ya que éste se encuentra ocupado por nosotros mismos, ya sea desde CAECUS o DOMUS.
- **CAECUS**: al volver a abrir la aplicación después de cerrarla, automáticamente trata de conectarse al módulo Bluetooth remoto, resultando al igual que en el caso anterior como error de conexión, debido a que el módulo Bluetooth ya está ocupado y no es capaz de mantener múltiples conexiones.

Nos encontramos con una situación similar cuando el usuario pulsa el botón físico “Back”, regresa a la pantalla anterior sin desconectar el Bluetooth, esto puede ser un grave error ya que en muchos dispositivos el botón es táctil y en muchas ocasiones se toca involuntariamente, y puede ser un problema para el caso de usuario invidente.

La solución que se implementará para ambas aplicaciones será ejecutar el protocolo de desconexión cada vez que se cierre la aplicación (se pulse el botón Home), así como deshabilitar la función del botón Back dentro de ellas.

En el caso de que la pantalla sea apagada por inactividad, o por el pulsado del botón físico “bloquear pantalla”, no se procederá a desconectar y la aplicación resumirá en el estado en el que se encontraba una vez se desbloquee la pantalla. Se seguirá el mismo comportamiento en casos de llamada entrante, alarma en pantalla y, si el Smartphone del usuario dispone de función multitarea, cambio de aplicación sin llegar a cerrar la actual (sin pulsar el botón Home).

Errores en fallo de Bluetooth y detección de pérdida de conexión

Las aplicaciones han de detectar perdida conexión Bluetooth, ya que ante la desconexión la aplicación no tiene ningún uso y puede desconcertar al usuario. Diferenciamos tres tipos de pérdida de conexión:

- **Pérdida por lejanía o muy baja señal**: se produce cuando la comunicación es tan débil que acaba por cortarse. Es debida a una distancia muy grande (alrededor de 20 metros) entre Smartphone y

módulo Bluetooth remoto, o a obstáculos en la comunicación (muros gruesos, estructuras de metal encerrando el Smartphone..)

- **Pérdida por desactivación de Bluetooth local**: se produce cuando el Smartphone pierde su señal Bluetooth propia. Esto es debido a que el usuario desactive voluntariamente el Bluetooth local durante la comunicación, apague su Smartphone o se quede éste sin batería.
- **Pérdida apagado de Bluetooth remoto**: se produce cuando el módulo Bluetooth remoto deja de emitir señal. Esto es debido a que no esté alimentado correctamente, o que la fuente que alimenta todo el sistema (maqueta demostradora en nuestro caso) haya sido apagada.

La respuesta de ambas aplicaciones ante cualquiera de estos tres casos será informar al usuario de la pérdida de conexión y cerrar la aplicación, indicando que vuelva a abrirla para reintentar la conexión.

Error al abrir la aplicación teniendo el Bluetooth local desactivado

Si el Bluetooth del Smartphone ha sido desactivado, se debe solicitar al usuario que lo active cuando se abra la aplicación, antes de que pueda intentar realizar una conexión y falle el programa. Se implementará un mensaje informativo tipo alerta, con la opción de activar el Bluetooth directamente desde el mensaje pulsando sobre él.

Errores en conexiones fallidas

Si se trata de iniciar una conexión a un dispositivo Bluetooth que se encuentra apagado o fuera de radio, la aplicación debe informar del error y actuar en consecuencia.
La aplicación DOMUS mostrará un mensaje informativo del fallo de conexión y regresará a la pantalla de elección de dispositivo, donde el usuario podrá elegir de nuevo un dispositivo sobre el que comenzar la conexión.
La aplicación CAEUS emitirá un audio informativo y regresará a la pantalla inicio del Smartphone, donde el usuario deberá volver a abrir la aplicación para reintentar la conexión automática, ya que esta se realiza cada vez que la aplicación es abierta.

Errores de audio en solapamientos y silencios

En el caso de CAECUS, los mensajes informativos han de ser sonoros, por lo que no pueden darse varios a la vez ya que confundiría al usuario, y serían difíciles de distinguir. Para ellos se reducirá el número de canales sonoros a uno, y se establecerá un tiempo medio de espera para cada mensaje.

Al ser el audio el modo de enviar Feedback al usuario invidente, ha de estar siempre encendido, aunque el Smartphone se encuentre en modo silencio, al similar que actúa la función alarma de un móvil. Se implementará para ello en el código de la aplicación CAECUS un método de audio independiente al del Smartphone.

3.4 Diseño del código Arduino

A continuación se definirá el comportamiento deseado para el microcontrolador Arduino, así como la gestión de posibles errores que debemos evitar.

3.4.1 Comportamiento del Arduino

El Arduino actuará como intermediario entre el módulo Bluetooth remoto y el resto del circuito. No sólo se encargará de interpretar los mensajes que llegan desde el Bluetooth y de accionar los actuadores, sino que además se encargará de recolectar la información del circuito y entregársela al módulo Bluetooth para que lo envíe como feedback al Smartphone.

Para llevar a cabo estas funciones, hay que implementar el código que estará reproduciéndose en bucle, y logrando el resultado objetivo. Usaremos el IDE (Entorno de Desarrollo Integrado) de Arduino para ello, llamado Arduino Tool y basado en Visual Studio. El lenguaje de programación es muy sencillo y basado en C.

Se crearán dos códigos distintos, para funcionar con DOMUS y con CAECUS óptimamente, y por último un tercer código que sea capaz de funcionar correctamente con ambas aplicaciones. La creación de éste tercer código será para implementarlo en la maqueta demostradora, a modo de que sea posible enseñar el comportamiento de ambas aplicaciones sin tener que cambiar el programa que está corriendo en el Arduino en ese momento.

El programa ha de detectar cuando se ha realizado una conexión al módulo Bluetooth remoto, y realizar un testeo del estado actual de todos los actuadores para entregárselo al módulo Bluetooth, que lo enviará al Smartphone, el ya conocido informe inicial.

A partir de éste momento quedará a la escucha permanente de los datos que lleguen al módulo Bluetooth, y cuando llegue una orden la llevará a cabo, mediante el encendido o apagado de sus pines conectados al actuador en concreto. Acto seguido entregará el feedback al módulo Bluetooth para que éste lo envíe al Smartphone.

3.4.2 Gestión de errores y prevención de fallos en Arduino

El Arduino es quien se encarga de encender y apagar cada uno de los actuadores, por lo que en caso de fallo del sistema, desconexión de la aplicación móvil o pérdida de alimentación ha de seguir un protocolo que sea seguro.

Errores en pérdida de conexión

> Cuando el módulo Bluetooth pierde la conexión, los actuadores han de mantener el estado que tenían previo a la pérdida de conexión, ya que ésta puede deberse a que simplemente el usuario ha cerrado la aplicación, o se ha alejado lo suficiente como para que la conversación entre Bluetooth no pueda darse.

Esto no debe repercutir en los estados de los actuadores, por lo que se escribirá el programa de forma que así sea.

Prevención de fallo de seguridad en la puerta

Relacionado directamente con el punto anterior, ya que si se pierde conexión voluntariamente o involuntariamente, los actuadores van a mantener su estado, en el caso de la puerta puede ser un fallo de seguridad que la puerta quede abierta, por ello se implementa un cronometro en el programa que calcule diez segundos desde que la puerta es abierta, y la cierre si en ese intervalo el usuario no ha enviado la orden de cerrado.

4. Desarrollo del sistema

A continuación se detallará como se ha llevado a cabo la elaboración del sistema completo, desde la creación de las aplicaciones y el programa que corre en Arduino, hasta la construcción del circuito remoto y la maqueta demostradora.

4.1 Desarrollo del circuito remoto

Serán definidos los elementos usados, así como la implementación y conexionado del sistema completo integrado en la maqueta demostradora.

Cómo ya fue previamente definido en el apartado del diseño, el sistema de domótica está controlado por el microcontrolador Arduino, que irá encerrado en una caja de instrumentos junto al circuito interruptor, con los únicos accesos a la alimentación del Arduino y a su puerto USB.

Para contar el desarrollo completo del circuito remoto primero es necesario definir los elementos usados en él, ya nombrados en el apartado de diseño.

4.1.1 Arduino y módulo Bluetooth

El modelo escogido de microcontrolador es el Arduino UNO R3, debido a su pequeño tamaño, nos permitirá integrarlo fácilmente en una caja de instrumentos de reducidas dimensiones.

Para su conexión con el módulo Bluetooth HC-06, habría que conectar los pines de Tx (transmisión de datos) y Rx (recepción de datos) del módulo a los pines correspondientes al puerto serie del Arduino, y viceversa.
El ocupar el puerto serie del Arduino con el módulo Bluetooth tiene el inconveniente de que si queremos meter un nuevo programa al Arduino por medio del USB, dará error por estar ya ocupado con el módulo Bluetooth. Para evitarlo haremos uso de la función Software Serial de Arduino, que nos permite usar otros pines del microcontrolador como un puerto serie virtual, dejando los originales libres.

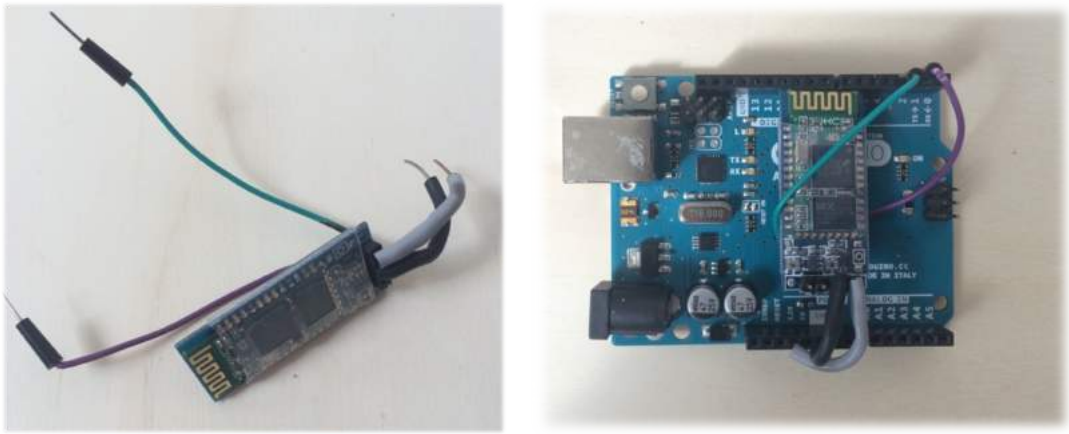


Ilustración 17: Módulo HC-06 con cables soldados (izq.) y su integración en Arduino (dcha.)

Además, el módulo Bluetooth debe recibir alimentación desde el Arduino.
Para una optimización de espacio, se optará por colocar el módulo sobre el Arduino, soldando sus pines a cable, para poder conectar sus 4 pines correctamente a los pines correspondientes del Arduino.

4.1.2 Circuito interruptor

Su función será, como su nombre indica, ejercer de conmutador mediante un MOSFET entre la fuente de alimentación y los actuadores de tipo solenoide (cerrojo y válvula).

Este circuito interruptor consta de dos MOSFET **IRF740** (en encapsulado *SOT-23*), apoyados en un disipador de calor, sendas resistencias para limitar la corriente a la puerta de cada MOSFET, así como dos diodos que servirán para desviar la corriente liberada por los solenoides cuando sean desactivados y evitar que se dañe el circuito. Usamos dos pequeñas placas recortadas de circuito prototipado para realizar las conexiones cada conjunto de MOSFET-Resistencia-Diodo.

El circuito interruptor estará colocado junto al Arduino dentro de la caja de instrumentos.

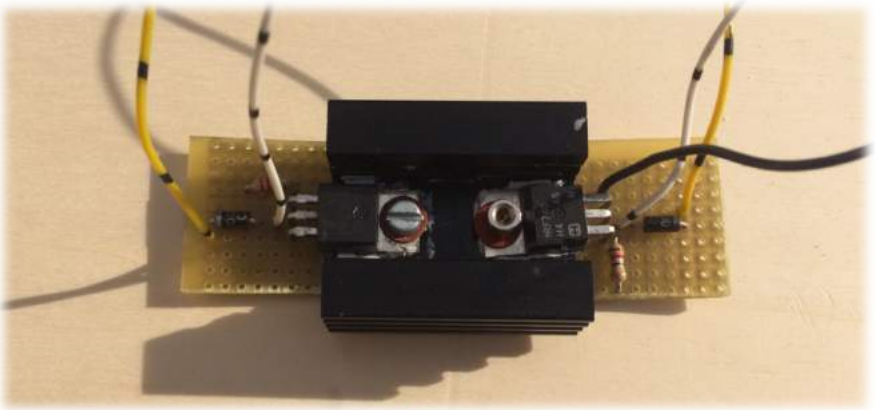


Ilustración 18: Circuito interruptor

Para añadir los MOSFET al disipador de calor se usarán dos pequeñas láminas de mica, así como pasta térmica para un total contacto con el disipador y que éste pueda ejercer su función de modo correcto.

4.1.3 Caja de instrumentos con raíl DIN

La caja de instrumentos elegida es lo mas reducida posible para que el módulo microcontrolador ocupe lo mínimo, y se encuentre encerrado de forma compacta, para que no pueda moverse y sufrir daños una vez se cierre la caja.

Dentro encerraremos el conjunto de Arduino con módulo Bluetooth, junto al circuito interruptor, y usaremos las pestañas de terminales para pasar los cables necesarios.

Ilustración 19: Caja de instrumentos con carril DIN

Sus características:

- Las medidas de la caja son 106.2 x 90 x 31.9 mm.
- Está fabricada en policarbonato UL94-V0.
- Carcasa de parte superior sólida de bajo perfil con 4 terminales y 2 protectores.
- Posibilidad de montaje en raíl DIN.
- Presenta certificado de conformidad RoHS.

4.1.4 Relé

El módulo relé va a conmutar la fase de la toma de corriente general hacia la base de enchufa, proporcionándole actividad a esta o apagándola. Además ha de poder ser accionado por el Arduino, que lo máximo que puede entregar uno de sus pines pines activo es 5V.

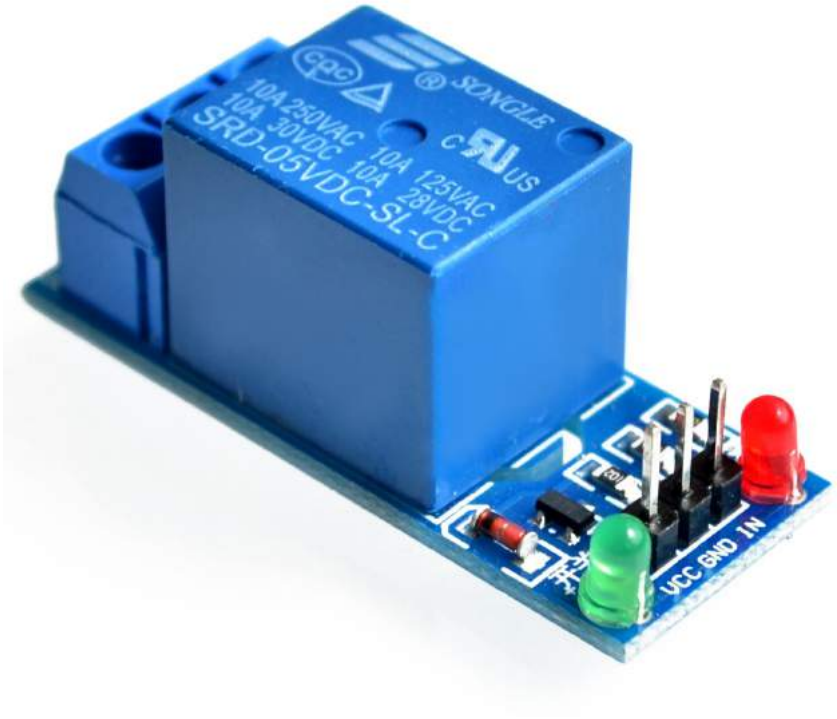


Ilustración 20: Módulo relé para Arduino

Por ello se ha escogido un módulo relé de la compañía SONGLE con las siguientes características:

- Control de carga 250 VAC a 10 A.
- Señal de control de 5V hasta 12V.
- Alimentación de 5V.
- Diodo de protección y circuito transistor de activación.
- LED rojo de estado de alimentación.
- LED verde de estado activo.

4.1.5 Base de enchufe

Se escoge una base de enchufe modular para montaje en carril DIN, con toma de tierra. Fabricada de material anti golpes y con bornes de conexión integrados en su estructura, para facilitar el pasado de cables, que deben llegarle desde la toma de corriente general de la maqueta demostradora.

Ilustración 21: Base de enchufe

Sus características técnicas son:

- Tipo de conexión: 2 Polos + Tierra.
- Corriente nominal: 16 A.
- Tensión de empleo: 250 VAC.
- Máxima tensión de empleo: 500 VAC.
- Grado de protección: IP-20.
- Tipo de montaje: Sobre carril DIN.
- Capacidad de conexión de cable: 4 $mm^{2.}$

4.1.6 Actuadores de corriente continua

El sistema será capaz de activar, aparte de la base de enchufe, otros tres elementos que funcionan con corriente continua, suministrada a través de la fuente de alimentación.

4.1.6.1 Actuador cerrojo

El cerrojo a usar es de tipo solenoide, estos son básicamente electroimanes. Hecho de una gran bobina de alambre de cobre con un lingote de metal en el medio. Cuando se energiza la bobina el lingote se mueve en la bobina, haciendo que el solenoide sea capaz de tirar de un extremo y mover el cerrojo, liberando lo que esté encerrando.

Ilustración 22: Cerrojo de tipo solenoide

Sus características técnicas son:

- Tensión nominal: 12 V.
- Corriente: 1.3 A.
- Potencia 16 W.
- Consumo de 650 mA a 12 V una vez activado.
- Diseñado para 1-10 segundos de tiempo de activación.
- Posibilidad de girar el émbolo del cerrojo 180º.
- Dimensiones: 23.57 x 67.44 x 27.59 mm.
- Peso: 200 g.

4.1.6.2 Actuador electroválvula

La función de la válvula es permitir el paso de fluidos a través de ella a voluntad del usuario.

Se ha escogido una válvula de dos vías de tipo solenoide de la marca Hydroelectrics, su funcionamiento es similar al del cerrojo, ante la energización de la bobina que contiene un émbolo metálico, éste es retraído dejando pasar los fluidos que están ejerciendo presión desde uno de los extremos de la válvula al otro.

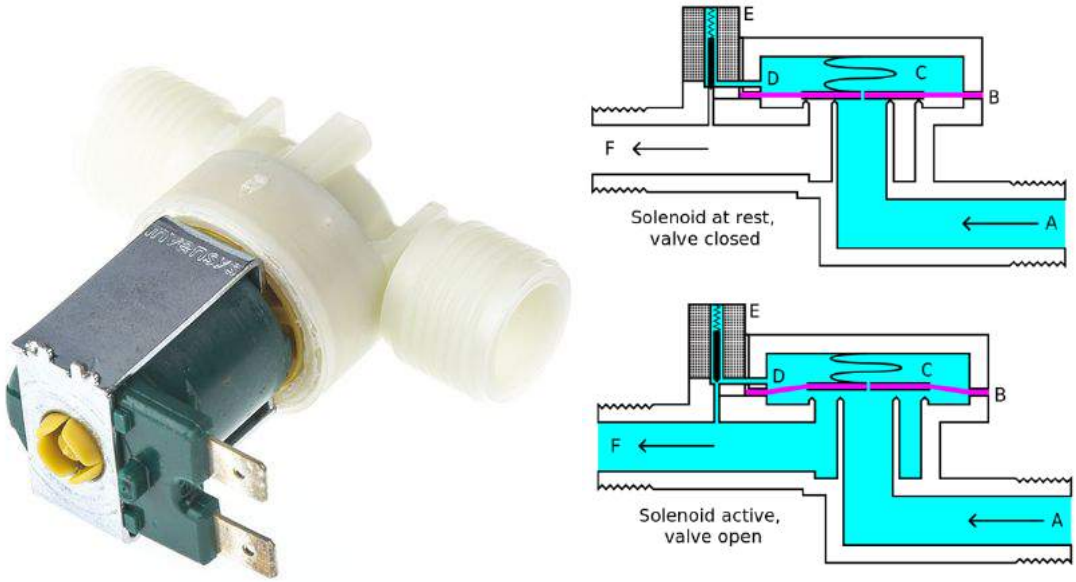


Ilustración 23: Electroválvula de tipo solenoide (ixq.) y funcionamiento de elctroválvula (dcha.)

La válvula, de estado normalmente cerrado, tiene un funcionamiento mediante diafragma que controla los fluidos a una presión mínima de 0,2 bares. Presenta un valor nominal continuo cuando la temperatura del líquido está por debajo de 25 ºC, a temperatura ambiente por debajo de 60 ºC y un caudal mayor a 5 l/min.

Especificaciones técnicas:

- Tensión de alimentación: 12 V.
- Consumo de potencia: 4W.
- Número de puertos: 2.
- Posición de la válvula por defecto: normalmente cerrada.
- Material del cuerpo: poliamida.
- Rango de presión: 0,2 a 10 bares.
- Caudal máximo de 17 l/min.
- Tipo de conexión: rosca estándar BSP.

4.1.6.3 Lámpara de LEDs

La función de lámpara no es otra que la de iluminar, y demostrar la capacidad de ajustar el brillo a través de la aplicación de Smartphone, por lo que se necesita una alta visibilidad incluso a la luz del día.

Para la fabricación de la lámpara de LEDs, se han dispuesto una serie de LEDs de alta luminiscencia de la marca Fologar, de colores blanco y azul, en un corcho fino que los mantiene en pie. Debido a que el Arduino no es capaz de entregar voltaje suficiente para encender varios LEDs en serie, deben de ser todos cableados en paralelo.

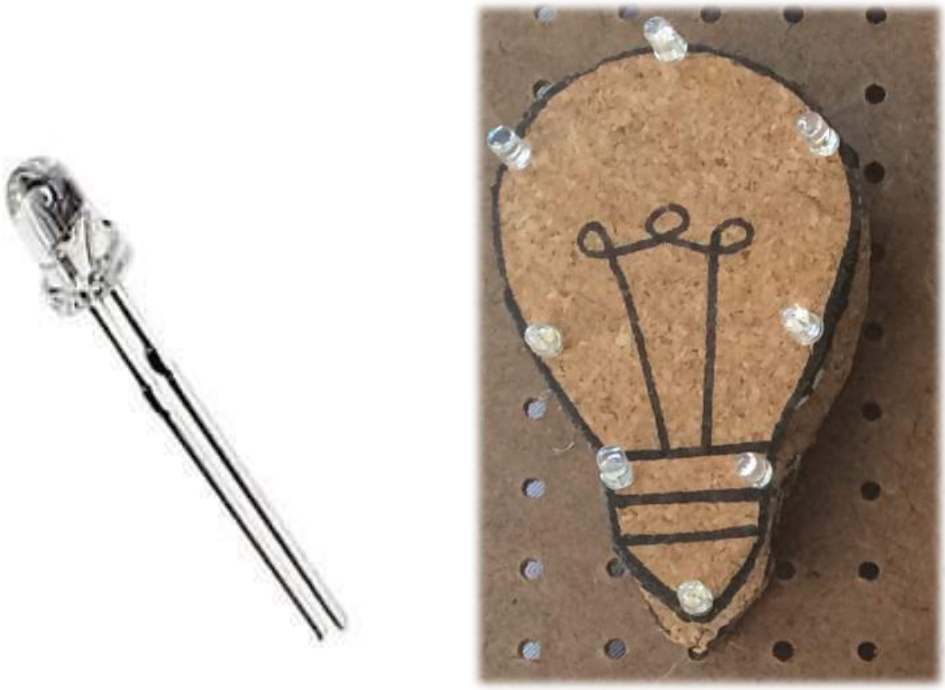


Ilustración 24: Diodo LED (izq.) y lámpara de LEDs (dcha.)

Las especificaciones técnicas de los LEDs son:

- Emisión de color: blanco y azul.
- Tensión de funcionamiento: 1,8 a 3,8 V.
- Máxima corriente continua directa 35 mA.
- Ángulo de visión: 20º a 25º.
- Luminosidad: 5000 MCD a 20000 MCD.
- Tamaño: 5x9 mm.
- Temperatura de funcionamiento: -25 ºC/ 80 ºC.
- Esperanza de vida: 100.000 Horas.

4.1.7 Fuente de Alimentación

La función de la fuente de alimentación es suministrar corriente continua a todos los elementos del circuito a excepción de la base de enchufe. Dado que la tensión máxima a necesitar es de 12 V, debido tanto a la válvula como al cerrojo, la fuente a usar dispondrá de esta tensión nominal de salida. Para elementos que requieran menos voltaje se usarán reguladores de tensión.

Para alimentarla requiere de una toma de corriente universal de 110/220 VAC, se usará la misma que alimenta a la base del enchufe.

La fuente empleada para su integración en la maqueta demostradora es de la marca RS, con soporte para carril DIN y salida única. Cuenta con protección contra cortocircuitos, sobrecargas y sobretensión. Además dispone de refrigeración mediante convección de aire libre y señal activa de OK de DC integrada.

Ilustración 25: Fuente de Alimentación

Sus especificaciones técnicas son:

- Tensión de salida: 12V, regulable hasta 9.5 V.
- Corriente de salida: 1.67 A.
- Potencia nominal: 20 W.
- Dimensiones: 100 x 22.5 x 90 mm.
- Temperatura min / Max: -20 ºC / +70ºC.
- Consumo de energía sin carga < 0,75 W.

4.1.8 Toma de corriente general

La función de la toma de corriente general es alimentar a la fuente de alimentación con corriente alterna, así como a la base de enchufe.

Para su integración se ha usado un cable de enchufe con toma de tierra y se le ha instalado un interruptor que conmuta tanto fase como neutro, evitando así que si el enchufe se conecte del revés, le esté llegando señal a la fuente. La tierra se ha mantenido intacta, pasándola por un lado del interruptor.

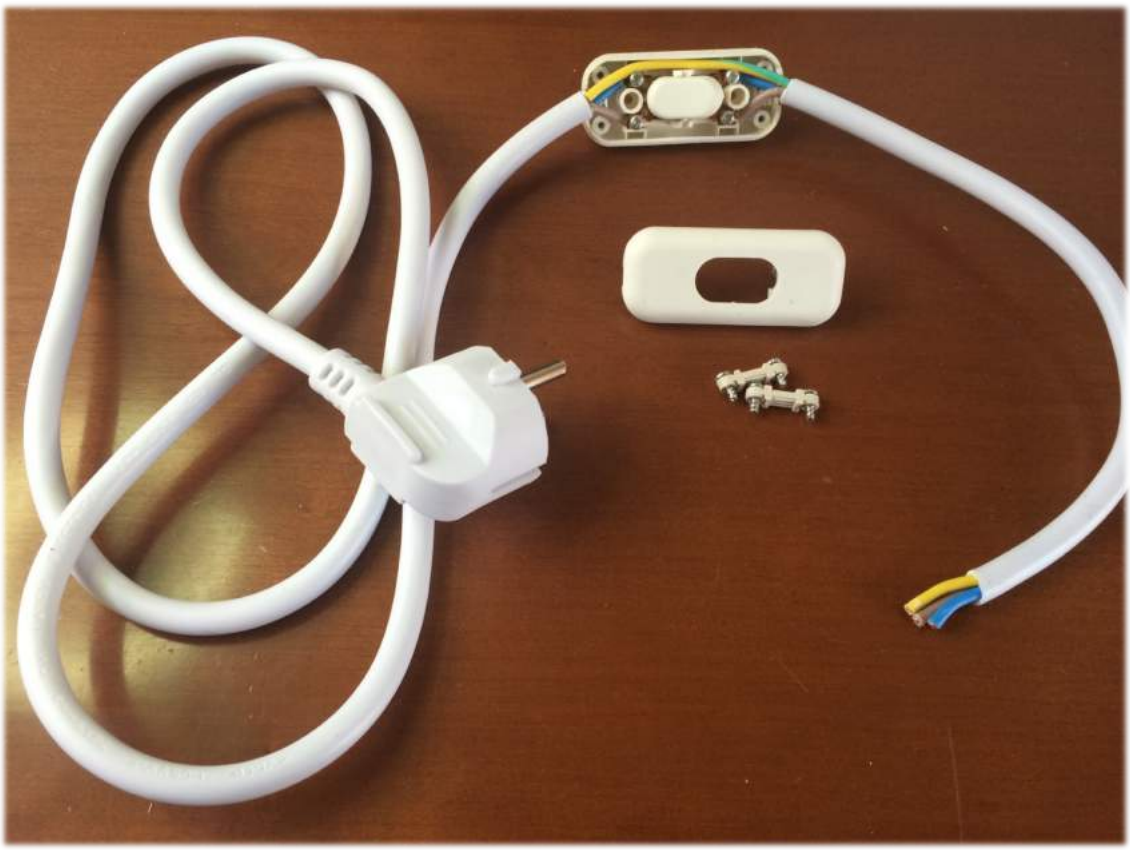


Ilustración 26: Instalación de interruptor en la toma de corriente general

Se usará una clema para poder alimentar con el mismo cable a la fuente de alimentación y a la base de enchufe (ésta última a través del relé).

Especificaciones técnicas:

- Conector: conector macho europeo.
- Longitud: 2.5 m.
- Corriente nominal: 16 A.
- Tensión nominal: 250 VAC.

4.1.9 Montaje de la maqueta demostradora

Se resumirá con texto acompañado de imágenes la construcción y desarrollo del montaje de la maqueta.
Se dedujeron las medidas de la maqueta de la casa guardando especial cuidado en dejar espacio en cada cuarto para el actuador, LED informativo y espacio para actuar (suficiente para un globo hinchado en el caso de la válvula). Una vez tomadas se procedió a cortar la madera.

Los materiales usados fueron:

- Contrachapado fino para las paredes intermedias.
- Tablón grueso para la base de la maqueta.
- Tablón medio para paredes y suelos intermedios
- Contrachapado agujereado para el fondo, facilitando el paso de los cables.
- Lámina de metacrilato para el frontal del tejado, donde colocar los logos.
- Lámina de corcho, para el tejado y para dar forma a la lámpara de LEDs.
- Ángulos, diversos tornillos y silicona para sujeción.
- Raíl DIN de 15 cm para la fuente de alimentación, base de enchufe y caja.

4.1.9.1 Construcción de la casa

Con las medidas ya tomadas se procede a cortar la madera y atornillar las distintas piezas, dejando el fondo de la casa sin encajar, ya que será en él donde instalaremos todos los actuadores y elementos.

Ilustración 27: Maqueta demostradora en construcción

Una vez montada, se comprueba que el espacio entre pisos es el adecuado, y se instala el carril DIN al fondo de la casa.

4.1.9.2 Integración del Arduino y circuito interruptor

Ya que será de aquí de donde saldrán los cables al resto del circuito será el primero en colocarse en la maqueta. Una vez soldado el circuito interruptor, se colocará en la base de la caja de instrumentos, junto al Arduino.

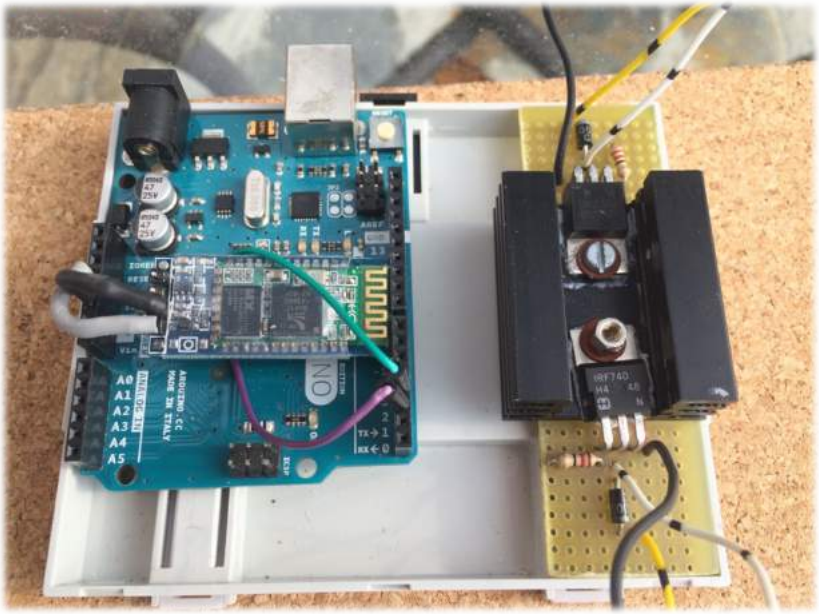


Ilustración 28: Arduino con Bluetooth junto a circuito interruptor sin conexión

Acto seguido se procede al conectado de cables entre el Arduino y circuito interruptor; Se usará una bornera para facilitar el interconectado de tierra entre Arduino y ambas partes del circuito interruptor.

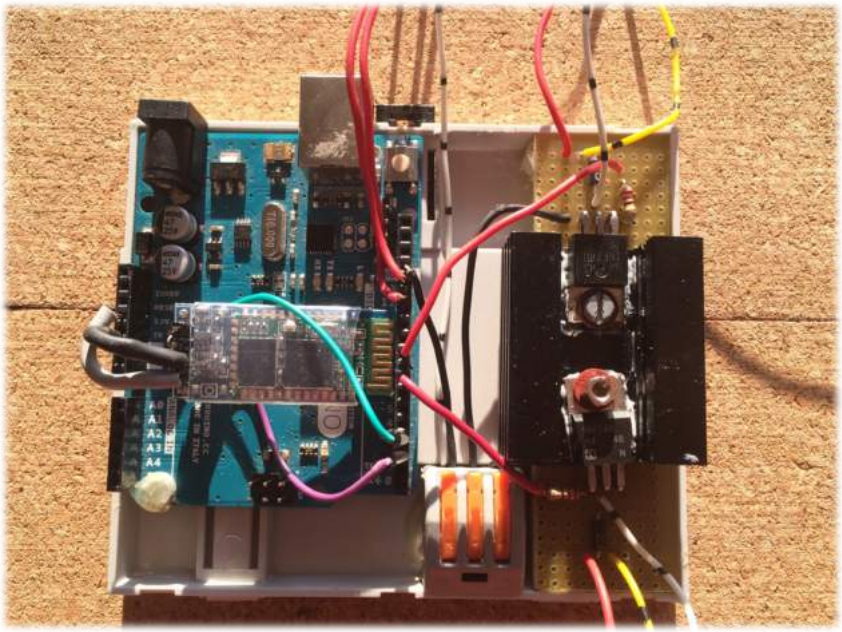


Ilustración 29: Arduino con módulo Bluetooth junto a circuito interruptor y conexiones completas y fijaciones a la caja de instrumentos

Ahora que ya tenemos conectado todo el circuito al microcontrolador se puede cerrar la caja de instrumentos.

Para ello sacamos realizamos dos aberturas en la caja para poder acceder a la entrada de alimentación del Arduino y a su puerto serie con la caja cerrada, así como un pequeño orificio en la tapa superior a la altura del módulo Bluetooth, para poder tener a la vista el LED informativo de éste.
Para finalizar utilizamos las pestañas laterales para sacar todos los cables necesarios por las pestañas, y los pasaremos también a través de los agujeros del contrachapado que hace de fondo, evitando que queden a la vista.

Una vez acabado éste proceso, se insertará la caja en el carril DIN junto a la fuente de alimentación.

4.1.9.2 Integración de la fuente de alimentación

Se colocará la fuente en el carril DIN, junto a la caja de instrumentos, de modo que con un cable conector a su salida, conecte con la entrada de alimentación del Arduino, dotándolo de energía para su funcionamiento.

Además también conectaremos a su salida positiva dos largos cables que alimentarán los actuadores cerrojo y electroválvula, y a su salida negativa los cables que salen del circuito interruptor, para cerrar el circuito cuando nuestro Arduino active los MOSFET.

Para que la fuente de alimentación pueda funcionar se necesita conectar su entrada a 220V, para ello conectaremos el cable de corriente general a los bornes de alimentación de la fuente.

4.1.9.2 Integración de los actuadores

Los distintos actuadores se irán disponiendo en el fondo agujereado de la casa según se vayan cableando.

Base de enchufe y relé

Junto a la caja de instrumentos se instalará la base de enchufe, ya que también cuenta con rail DIN. Dicha base estará alimentada al igual que la fuente, con la toma de corriente general, a diferencia de que en este caso el cable de fase pasará por el módulo relé, que la conmutará bajo orden del Arduino.

Se colocará el módulo relé encima de la base de enchufe, con los LEDs informativos visibles, para que el usuario pueda siempre saber si la base de enchufe esta activada o desactivada.
Para alimentar el módulo relé y conectarle también la señal de entrada que lo lidera, usaremos los cables correspondientes que salen del Arduino y que previamente hemos sacado de la caja de instrumentos a través de sus pestañas.

Lámpara de LEDs

Para realizar la lámpara de LEDs utilizamos una lámina de corcho, para darle un aspecto representativo se le dará forma de bombilla. Acto seguido se dispondrán sobre ella una serie de LEDs blancos y azules alternando colores, y posteriormente se procederá a soldarlos de modo que todos queden conectados en paralelo.

Una vez cableados, se conectarán a los cables correspondientes que salen de la caja de instrumentos. Para darle profundidad a la lámpara se formara una estructura de dos pisos, dejando el pequeño circuito de conexión en medio, consiguiendo así protegerlo y ocultarlo.

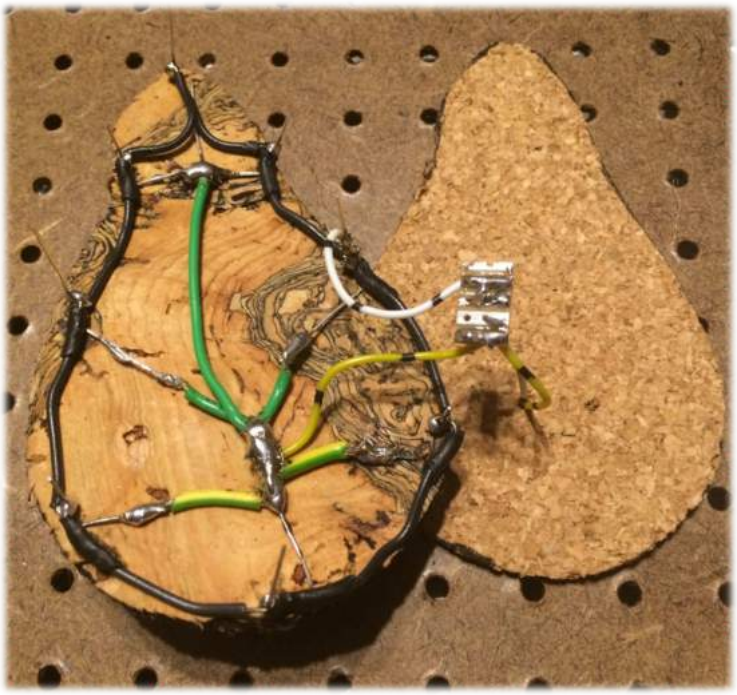


Ilustración 30: Cara interior de la lámpara de LEDs

Se recuerda que el pin del Arduino que controlará la lámpara debe ser un pin con PWM (Pulse Widht Modulation), ya que para controlar el brillo de la lámpara haremos uso de ésta función del microcontrolador. Dicha utilidad permite entregar voltajes que se encuentran entre 5 y 0 voltios mediante el cambio de la porción de tiempo que la señal emitida por este pin se mantiene activa.

Cerradura y electroválvula

Ambos actuadores son de tipo solenoide, y su activación se lleva a cabo de la misma forma. Un conector del actuador irá cableado al polo positivo de la salida de la fuente de alimentación, mientras que el otro se conectará a uno de los dos cables que salen de la caja de instrumentos y conectan con el circuito interruptor (cada uno al suyo correspondiente).

Junto a la señal que activa los actuadores se situará también un LED con función informativa que se encenderá junto al actuador, e informará al usuario de ello.
Así como la luz que iluminará el frontal del tejado, dividido en DOMUS y CAECUS, informando de la aplicación que está siendo usada en el momento.

El resultado final de todos los actuadores integrados en el fondo de la maqueta y conectados es el siguiente:

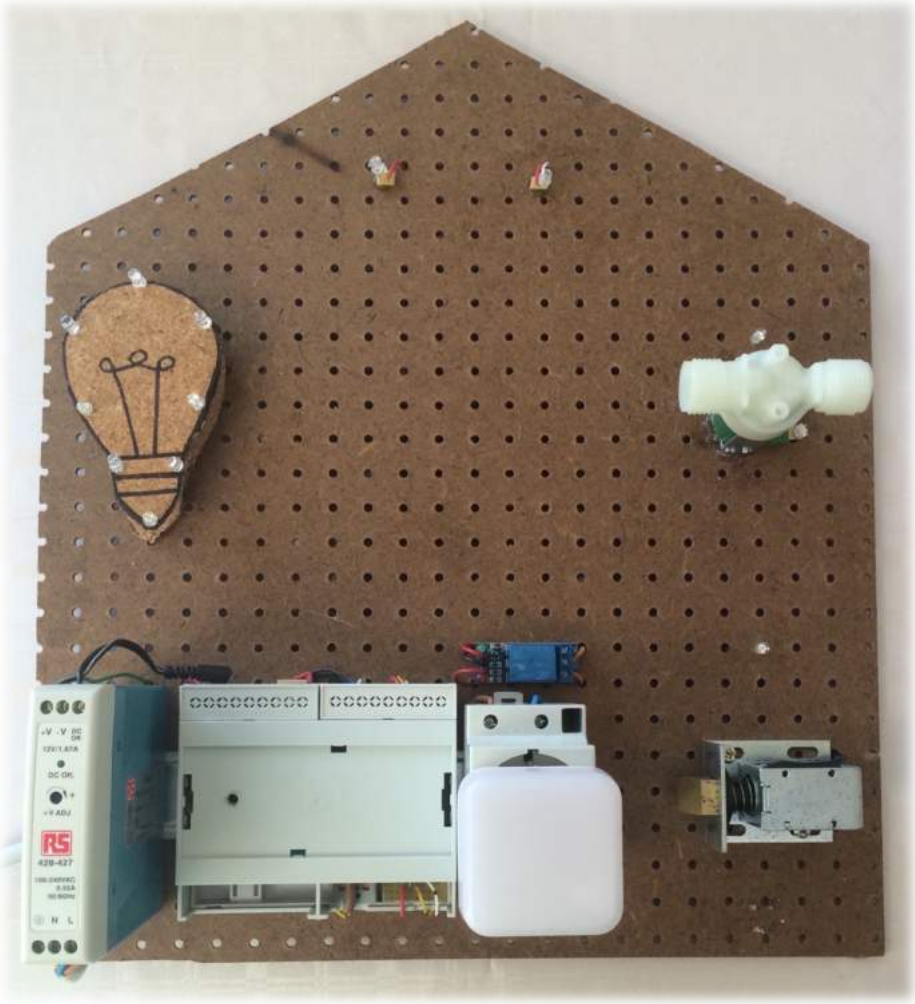


Ilustración 31: Fondo de la maqueta demostradora con todos los actuadores y LEDs informativos

4.1.9.3 Acabado, tejado y frontal

Una vez integrados todos los actuadores en el fondo de la maqueta se procederá a instalar el fondo en la estructura de la casa.

Por último se añadirá el tejado con dos láminas de corcho, y el frontal de metacrilato previamente cortado con la forma triangular del tejado. Le añadiremos los logos de DOMUS Y CAECUS impresos en papel de acetato.
Los logos adheridos al frontal de acetato estarán retro iluminados por un diodo LED colocado en el fondo de la maqueta, que conectadas al Arduino se encenderán cuando se detecte la aplicación en uso, a modo informativo.

En consecuencia nunca se encontrarán ambos encendidos al mismo tiempo, ya que es imposible conectarse a la maqueta con las dos aplicaciones al mismo tiempo.

Para finalizar se añadirá un globo de aire a la válvula cerrada, para demostrar la pérdida de aire sufrida al accionar y abrir la válvula.

Ilustración 32: Maqueta demostradora finalizada

4.2 Desarrollo de las Aplicaciones Android

Se detallará en este punto cómo se han hecho las aplicaciones móviles DOMUS y CAECUS, empezando con el entrenamiento y prototipos desarrollados con la herramienta ***MIT App Inventor*** , y concluyendo con el desarrollo final, realizado empleando ***Android Studio***.

4.2.1 Prototipo en MIT APP INVENTOR

Para realizar un primer acercamiento al mundo Android, y comprobar la viabilidad y resultados del proyecto, se opta por crear un prototipo de aplicación simple para gobernar el Arduino desde un Smartphone a través de Bluetooth.

Para ello se usa el entorno de desarrollo creado por el MIT para aprendizaje de Android llamado *MIT App Inventor*. Se trata de un programa que permite crear aplicaciones de forma sencilla e intuitiva mediante la unión de diversos módulos funcionales prediseñados, como si se tratase de un "*lego*" de programación.

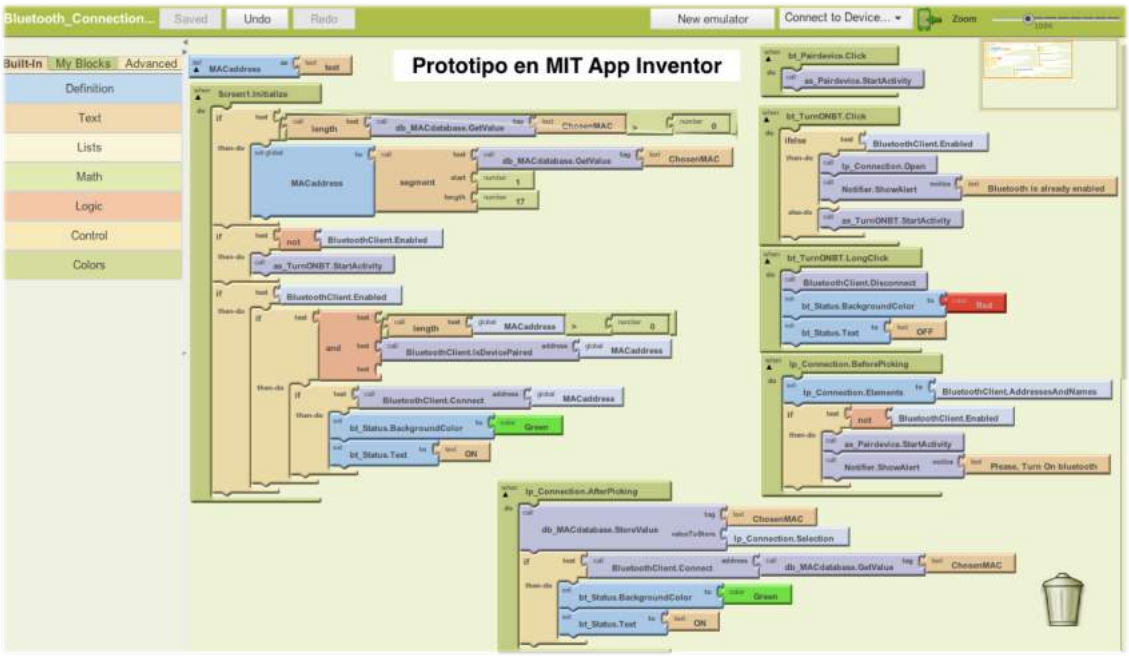


Ilustración 33: Prototipo de aplicación realizada con MIT App Inventor

Usando ésta herramienta se crea una aplicación prototipo con funciones básicas, que una vez instalada en un dispositivo Android, será capaz de conectarse vía Bluetooth al Arduino (junto al módulo Bluetooth) y a través de un botón en pantalla enviará la orden de encender un LED. Arduino interpretará el mensaje y activará el pin correspondiente.

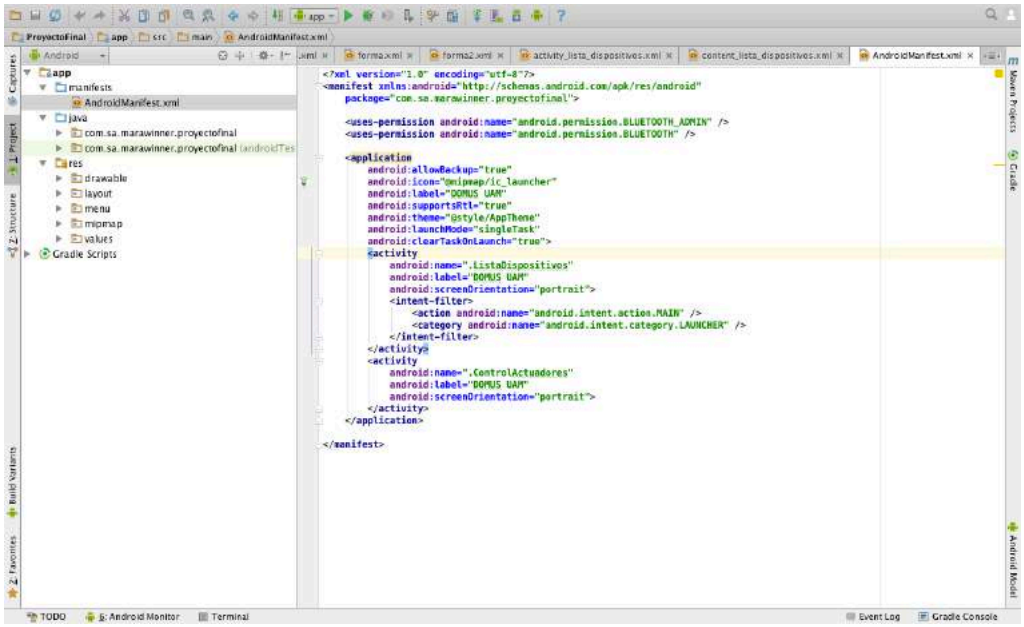
Se comprueba así la viabilidad del proyecto y de crear la aplicación, y se procede a su elaboración usando una herramienta más profesional y, aunque mucho más compleja, con una versatilidad completamente superior: *Android Studio*.

4.2.2 Desarrollo en Android Studio

Antes de explicar el desarrollo de la aplicación usando esta herramienta se va a proceder a explicar los principales apartados y carpetas que contiene un proyecto de Android Studio y el contenido que debe ir en cada uno. Se acompañará con capturas de pantalla de nuestras aplicaciones y fragmentos de código significativos.

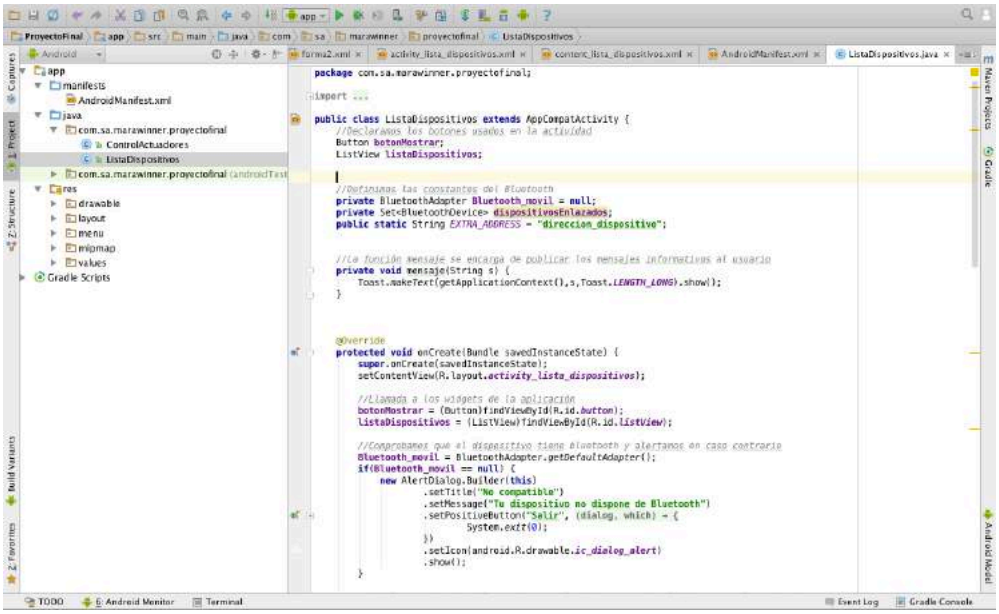
- **Manifest**

El *manifest* de una aplicación Android puede compararse a una declaración de intenciones en cuanto a usos recursos software y hardware de un Smartphone.
En esta carpeta debe ir la declaración de los recursos que nuestra aplicación va a usar del Smartphone, en el caso del proyecto, las aplicaciones deben tener permitido el acceso al Bluetooth del móvil, por lo que debe declararse aquí.



- **Java**

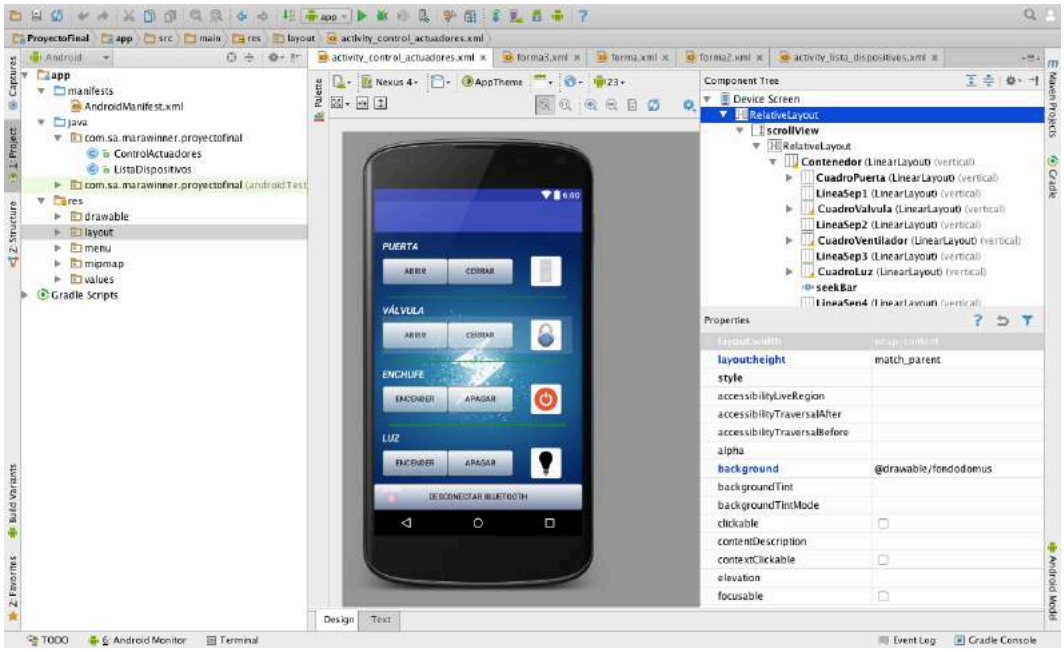
La carpeta Java contendrá el código fuente de nuestra aplicación. Aquí estarán almacenados todos los archivos con terminación .java. Estos archivos contienen el código que definirá el comportamiento de la aplicación. Por cada pantalla distinta que contiene la aplicación se ha optado por crear una clase distinta.



- **Resources**

Esta carpeta contiene todo lo relativo al aspecto visual de la aplicación, lo que el usuario va a ver, las pantallas en sí.
Éstas pantallas son llamadas **layout**, y deben ser definidas en esta carpeta, gracias a las herramientas de Android Studio tras un poco de aprendizaje es relativamente sencillo agregar botones a la pantalla, barras deslizantes, imágenes y diseñar a gusto del usuario las pantallas que harán de interfaz en la aplicación.

Además también contendrá los recursos de imágenes (subcarpeta **drawable**) y audios (subcarpeta **raw**) que usará la aplicación. Si se quiere añadir una imagen o un sonido nuevo, debe añadirse a esta carpeta. En las aplicaciones del proyecto contendrán las imágenes del logo (subcarpeta **mipmap**), las imágenes informativas de los botones en el caso de DOMUS y los audios informativos en el caso de CAECUS.



Además Android Studio cuenta con la posibilidad de emular la aplicación según se va desarrollando en dispositivos virtuales que podemos crear especificando versión de Android e incluso modelo y tamaño de pantalla. Será usado este método en las primeras pruebas y en diseño, sin embargo, al no permitir el uso del Bluetooth, una vez más avanzado el código se usarán dispositivos Android físicos intentando probar varias versiones y tamaños.

4.2.2.1 Desarrollo de DOMUS

Detallaremos cómo funciona la app desde el momento en que la abrimos y la ponemos a funcionar junto a la maqueta demostradora.

- **Clase: ListaDispositivos**

El código comienza creando la clase *ListaDispositivos*, en ella acontecerá todo lo que ocurre en la primera pantalla, de elección de dispositivos Bluetooth.

Se declaran los botones y textos que aparecerán en pantalla. En este caso el único botón es el que al presionarlo mostrará todos los dispositivos que están enlazados por Bluetooth con el móvil del usuario.

Se comprueba que el dispositivo dispone de Bluetooth, y si es así, que esté encendido. En caso contrario creamos una alerta que avise al usuario de ello, y solicita con un mensaje en pantalla que lo active, pudiendo hacerlo desde ese mismo mensaje a través de un botón incluido en él.

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_lista_dispositivos);

    //Llamada a los widgets de la aplicación
    botonMostrar = (Button)findViewById(R.id.button);
    listaDispositivos = (ListView)findViewById(R.id.listView);

    //Comprobamos que el dispositivo tiene bluetooth y alertamos en caso contrario
    Bluetooth_movil = BluetoothAdapter.getDefaultAdapter();
    if(Bluetooth_movil == null) {
        new AlertDialog.Builder(this)
                .setTitle("No compatible")
                .setMessage("Tu dispositivo no dispone de Bluetooth")
                .setPositiveButton("Salir", new DialogInterface.OnClickListener() {
                    public void onClick(DialogInterface dialog, int which) {
                        System.exit(0);
                    }
                })
                .setIcon(android.R.drawable.ic_dialog_alert)
                .show();
    }

    //Si el usuario tiene bluetooth, pero lo tiene apagado solicitamos que lo encienda
    else if(!Bluetooth_movil.isEnabled()){
        Intent enciendeBluetooth = new Intent(BluetoothAdapter.ACTION_REQUEST_ENABLE);
        startActivityForResult(enciendeBluetooth,1);
        mensaje("Por favor, active el Bluetooth para poder usar la aplicación");

    }

    //Si el usuario pulsa el botón de mostrar dispositivos, se llama a la rutina que los publica en pantalla
    botonMostrar.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            listaDispositivosEnlazados();

        }
    });

}
```

Una vez comprobado el Bluetooth la aplicación procede a acceder a los registros de Bluetooth del móvil, obteniendo todos los dispositivos con los que está enlazado, y publicándolos en una lista junto a su nombre y dirección MAC. Para ello, la aplicación esperará a que el usuario pulse el botón "mostrar dispositivos enlazados".

```
private void listaDispositivosEnlazados(){
    dispositivosEnlazados = Bluetooth_movil.getBondedDevices();
    ArrayList lista = new ArrayList();

    //Obtenemos los nombres y direcciones de los dispositivos BT enlazados y los mostramos en pantalla, en caso de no tener se indica
    if (dispositivosEnlazados.size()>0){
        for(BluetoothDevice bt : dispositivosEnlazados ){
            lista.add(bt.getName() + "\n" + bt.getAddress());

        }
    }

    else{
        mensaje("No se encontraron dispositivos bluetooth enlazados.");
    }

    //Comprueba que el usuario no ha desconectado el bluetooth, y en caso contrario solicita que lo encienda
    if(!Bluetooth_movil.isEnabled()){
        Intent enciendeBluetooth = new Intent(BluetoothAdapter.ACTION_REQUEST_ENABLE);
        startActivityForResult(enciendeBluetooth,1);
        mensaje("Por favor, active el Bluetooth para poder usar la aplicación");

    }

    //Cuando el usuario presione sobre un dispositivo se llamará al metodo siguiente
    final ArrayAdapter adapter = new ArrayAdapter(this,android.R.layout.simple_list_item_1, lista);
    listaDispositivos.setAdapter(adapter);
    listaDispositivos.setOnItemClickListener(listaClickListener);

}
```

En el momento que el usuario pulse sobre uno de los dispositivos que aparecen por pantalla se generará el evento que nos lleva a la siguiente pantalla. Para ello, el método que se encuentra a la escucha de qué dispositivo es presionado realiza la llamada a la clase "*ControlActuadores*" y le pasa la información pertinente del dispositivo Bluetooth al que se ha elegido conectar. Esta información consiste en los últimos 17 caracteres del dispositivo seleccionado, que pertenecen a su dirección MAC Bluetooth y sus respectivos separadores.

```
//Al elegir un dispositivo se acaba esta actividad y comienza la conexión en otra actividad
private AdapterView.OnItemClickListener listaClickListener = (av, v, arg2, arg3) -> {

        //Comprueba que el usuario no ha desconectado el bluetooth, y en caso contrario solicita que lo encienda
        if(!Bluetooth_movil.isEnabled()){
            Intent enciendeBluetooth = new Intent(BluetoothAdapter.ACTION_REQUEST_ENABLE);
            startActivityForResult(enciendeBluetooth,1);
            mensaje("Por favor, active el Bluetooth para poder usar la aplicación");

        }

        //Obtiene la direccion MAC del dispositivo
        String info = ((TextView) v).getText().toString();
        String MAC = info.substring(info.length() - 17);

        //Comienza la siguiente actividad "Control de Actuadores" pasándole la MAC del dispositivo seleccionado
        Intent i = new Intent(ListaDispositivos.this, ControlActuadores.class);
        i.putExtra(EXTRA_ADDRESS, MAC);
        startActivity(i);
};
```

- **Clase: ControlActuadores**

Esta segunda actividad lleva todo el proceso importante de la aplicación, lo primero que hace es usar los parámetros recibidos a partir de la anterior actividad para iniciar la conexión con el método "*conectarBluetooth()*". Éste se encarga de realizar la conexión con el dispositivo Bluetooth escogido por el usuario, en caso de no lograr éxito en la conexión se informará al usuario y se retornará a la primera actividad (*listaDispositivos*).

```
//Aquí se ejectuta la conexión de bluetooth entre dispositivo y modulo remoto
private class conectarBluetooth extends AsyncTask<Void, Void, Void> {
    private boolean exitoConexion = true;

    @Override
    protected void onPreExecute() {
        progress = ProgressDialog.show(ControlActuadores.this, "Conectando..", "Por favor espere.");
    }

    @Override
    protected Void doInBackground(Void... devices) {
        try {
            if (Socket == null || !BtConectado) {

                Bluetooth_movil = BluetoothAdapter.getDefaultAdapter();
                BluetoothDevice dispositivo = Bluetooth_movil.getRemoteDevice(MAC);
                Socket = dispositivo.createRfcommSocketToServiceRecord(identificadorBtSPP);
                BluetoothAdapter.getDefaultAdapter().cancelDiscovery();
                Socket.connect();
                inStream = Socket.getInputStream();
            }
        } catch (IOException e) {
            exitoConexion = false;
        }
        return null;
    }

    @Override
    protected void onPostExecute(Void result) {
        super.onPostExecute(result);

        if (!exitoConexion) {
            mensaje("Conexión fallida. Puede que el bluetooth que intenta conectar no se encuentre en rango o que no se trate del dispositivo DOMUS UAM)
            finish();
        }
        else {
            mensajecorto("Conectado con éxito.");
            BtConectado = true;
            beginListenForData();
            recibirInforme();
        }
        progress.dismiss();
    }
```

Para poder llevar a cabo esta tarea sin interrumpir o dejar bloqueada el hilo principal de la (Conocido con *User Interface Thread*) es necesario crear un hilo secundario, esto lo conseguimos a través de la clase "**Async Task**". Ésta clase permite realizar tareas en el "*Background*" de la aplicación para después publicar sus resultados en el antes mencionado **UI Thread.**

Una vez realizada la conexión la aplicación, todos los botones que controlan los actuadores estarán a la espera de ser presionados, en caso de que esto ocurra, cada uno mandará una trama representativa por medio de Bluetooth.

```
//Definimos las funciones que enviarán comandos por bluetooth al arduino al pulsar los botones

private void abrirPuerta() {
    if (Socket != null) {
        try {
            Socket.getOutputStream().write("POH".toString().getBytes());
        }
        catch (IOException e) {
            mensajecorto("Error al intentar abrir la puerta");
        }
    }
}

private void cerrarPuerta() {
    if (Socket != null) {
        try {
            Socket.getOutputStream().write("PFH".toString().getBytes());
        }
        catch (IOException e) {
            mensajecorto("Error al intentar cerrar la puerta");
        }
    }
}
```

Cada trama de cada botón es única e independiente y son interpretadas por el Arduino para llevar a cabo el accionado de los actuadores. Para ser enviadas escribiremos estas tramas en el **Socket** Bluetooth que hemos abierto al realizar la conexión.

```
//Cada botón llamará a una función al ser presionado

doorON.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        compruebaBluetooth();
        abrirPuerta();
    }
});
doorOFF.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v){
        compruebaBluetooth();
        cerrarPuerta();
    }
});
```

Por último la aplicación debe mantenerse a la escucha de la información que llegue vía Bluetooth desde el Arduino, ésta información (*Feedback*) nos da la confirmación de que la orden previamente enviada al Arduino ha sido recibida correctamente y se ha ejecutado.
Para poder mantenerse a la escucha de información sin interrumpir en las tareas del *UI Thread* se un hilo secundario y se manejará con un **Handler.**

Debido a que a los objetos pertenecientes al hilo principal (*UI Thread*) solo se puede acceder desde el mismo hilo principal, para mover datos desde hilos secundarios (*Background*) al principal se usa una herramienta que nos permite administrar los distintos hilos, esta es un *Handler*.

```
//Aquí se maneja el otro hilo secundario creado para RECIBIR datos por bluetooth, el feedback de Arduino
void beginListenForData() {
    final Handler handler = new Handler();
    final byte delimiter = 10; //Código ASCII de línea nueva

    stopWorker = false;
    readBufferPosition = 0;
    readBuffer = new byte[64];
    workerThread = new Thread(new Runnable() {
        public void run() {
            while (!Thread.currentThread().isInterrupted() && !stopWorker) {
                try {
                    int bytesAvailable = inStream.available();
                    if (bytesAvailable > 0) {
                        byte[] packetBytes = new byte[bytesAvailable];
                        inStream.read(packetBytes);
                        for (int i = 0; i < bytesAvailable; i++) {
                            byte b = packetBytes[i];
                            if (b == delimiter) {
                                final byte[] encodedBytes = new byte[readBufferPosition];
                                System.arraycopy(readBuffer, 0, encodedBytes, 0, encodedBytes.length);
                                final String feedback = new String(encodedBytes, "US-ASCII");
                                readBufferPosition = 0;
                                handler.post(() -> {
                                        infoArduino(feedback);//Llama al método que interpretará el feedback enviado a nuestro móvil por bluetooth
                                    });
                            } else {
                                readBuffer[readBufferPosition++] = b;
                            }
                        }
                    }
                } catch (IOException ex) {
                    stopWorker = true;
                }
            }
        }
    });

    workerThread.start();
}
```

Gracias a este *Handler,* la aplicación se mantiene en constante escucha de los datos que puedan llegar del Arduino a través de Bluetooth. En su mayoría de casos estos datos serán el citado *Feedback,* con el cual y a través del método *infoArduino()* serán interpretados y servirán para actualizar las imágenes de estado de la pantalla *ControlActuadores.*

```
//Según el feedback recibido desde arduino gestiona las imagenes y mensajes informativos de cambio de estado
void infoArduino(String feedback) {

    if (feedback.equals("puertaon\r")) {
        ImageView estado = (ImageView) findViewById(R.id.imageView);
        estado.setImageResource(R.drawable.opendoor);
        mensajecorto("Puerta abierta");
    }
    else if (feedback.equals("puertaoff\r")) {
        ImageView estado = (ImageView) findViewById(R.id.imageView);
        estado.setImageResource(R.drawable.closedoor);
        mensajecorto("Puerta cerrada");
    }
```

La primera vez que veamos este método trabajar será al realizar una nueva conexión, gracias al método *informeInicial()* que realiza una consulta al Arduino nada más establecer conexión para conocer el estado de los actuadores y actualizar las imágenes informativas de todos los actuadores.

Una particularidad de DOMUS es la posibilidad de ajustar el la intensidad luminosa de la lámpara de LEDs, para ello se dispone de un **widget** llamado *SeekBar* que consiste en una barra horizontal con un botón deslizante y que otorga valores ascendentes según más deslicemos el botón hacia la derecha. En el caso de la barra implementada se han configurado valores mínimos de 10 y máximos de 255. Dicho valor será enviado al Arduino (vía Bluetooth) y éste ajustará el brillo relacionado en modo directamente proporcional al valor de dicha barra.

```
brillo.setOnSeekBarChangeListener(new SeekBar.OnSeekBarChangeListener() {
    @Override
    public void onProgressChanged(SeekBar seekbar, int progress, boolean fromUser) {
        if (fromUser == true) {

            indicadorBrillo.setText(String.valueOf(progress));
            try {
                Socket.getOutputStream().write((String.valueOf(progress+10)+("J")).getBytes());
            } catch (IOException e) {

            }
        }
    }
```

Para finalizar, un paso muy importante a la hora de trabajar con el módulo Bluetooth reside en la necesidad de cerrar el *Socket* una vez se vaya a cerrar la aplicación o se pierda la conexión. Esto significa que debemos desocupar el módulo Bluetooth una vez no vayamos a usarlo más, para que pueda ser usado por otro usuario, o por el mismo usuario que quiera volver a conectarse más tarde.

Además, un error bastante frecuente en Android es el dejar un hilo huérfano, es decir, cerrar un hilo principal sin haber cerrado un secundario que depende de él. Por ello es muy importante que también se cierren todos los hilos secundarios abiertos antes cerrar el principal, o de lo contrario recibiremos errores.

Se crea el método *desconectarBt()* que será llamado cuando se quiera desee cerrar la conexión, ya sea por medio del botón de desconexión o mediante el cierre de la aplicación por el botón HOME.

```
//Deja el socket libre y acaba con el hilo secundario cuando se pulse el boton de desconexión
private void desconectarBt() {
    stopWorker = true;
    if (Socket != null) {
        try {
            Socket.close();
            mensajecorto("Dispositivo desconectado con éxito");
        }
        catch (IOException e) {
            mensaje("Error al desconectar");
        }
    }
    //Vuelve a la pantalla de inicio
    finish();
}
```

Debido entonces a la necesidad de cortar correctamente la conexión Bluetooth para liberar el módulo Bluetooth remoto, se debe ejecutar el protocolo de desconexión cuando:

- El usuario pulsa el botón de desconexión situado en la base de la aplicación:
- El usuario pulsa el botón físico "HOME" de su Smartphone.

```
//Cuando el boton home es presionado, liberamos el socket de bluetooth para no dar fallos
@Override
protected void onUserLeaveHint()
{
    super.onUserLeaveHint();
    desconectarBt();

    finish();
}
```

Además el módulo Bluetooth remoto quedará automáticamente liberado, y la actividad se cerrará automáticamente (informando al usuario de ello) y regresando a la pantalla de lista de dispositivos cuando:

- El usuario desconecta el Bluetooth de su Smartphone durante una conexión Bluetooth activa.
- El usuario se aleja lo suficiente del módulo Bluetooth como para perder la conexión.

Para que la aplicación sea capaz de detectar una pérdida de conexión o un apagado del Bluetooth local se definen filtros que perciban dicho evento o **intent** de pérdida de conexión.

Después se crea el método *BroadcastReceiver()*, el cual detectará cuando el Bluetooth ha sido desconectado, y se encargará de informar al usuario de ello así como de cerrar la actividad.

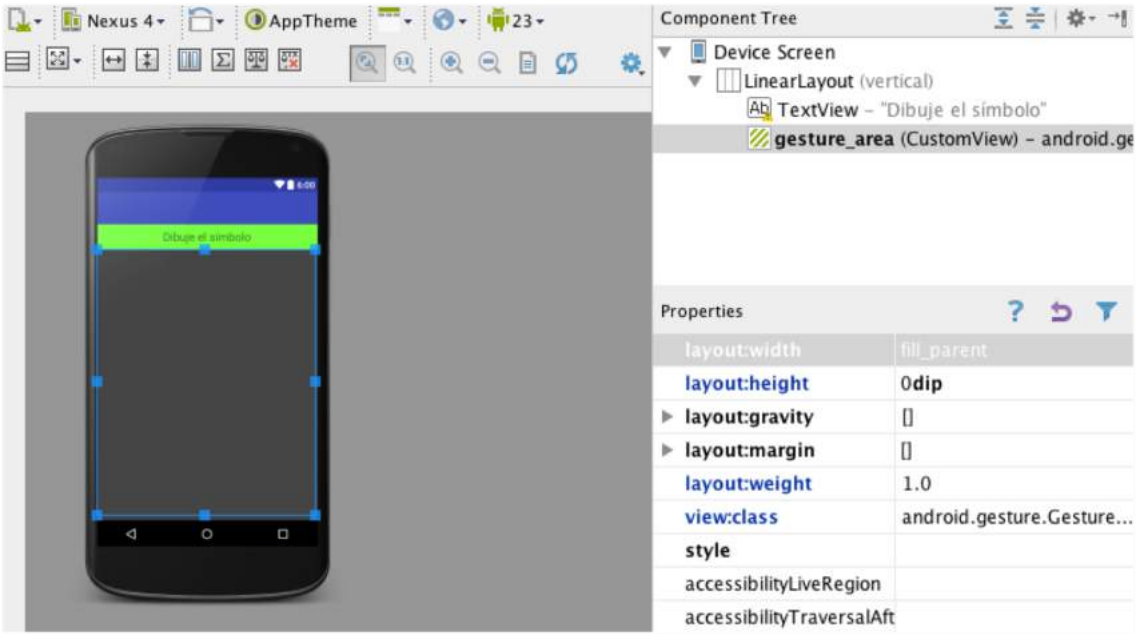
```
//Método que informará sobre pérdida de conexión con el módulo remoto
private final BroadcastReceiver mReceiver = new BroadcastReceiver() {
    @Override
    public void onReceive(Context context, Intent intent) {
        String action = intent.getAction();
        BluetoothDevice device = intent.getParcelableExtra(BluetoothDevice.EXTRA_DEVICE);

        //Cuando detecta pérdida de conexión procede a informar y a cerrar la actividad
        if (device.ACTION_ACL_DISCONNECTED.equals(action)) {
            mensaje("Conexión bluetooth perdida. Compruebe que el módulo remoto sigue encendido" +
                    " o acérquese y vuelva a conectarse.");
            finish();
        }
    }
};
```

4.1.2.2 Desarrollo de CAECUS

Gran parte del desarrollo de la aplicación CAECUS coincide con el de DOMUS, nos centraremos en detallar las diferencias y la parte del código que no ha sido previamente explicada en la anterior explicación.

La primera de las diferencias de CAECUS es su conexión Bluetooth, aunque el método de conexión es mismo, en CAECUS no se presenta opción al usuario de que elija dispositivo, sino que se conectará automáticamente al módulo Bluetooth instalado en la maqueta demostradora. Se ha elegido esta opción por la dificultado que representa a una persona invidente elegir un dispositivo entre una larga lista en pantalla.
A diferencia de DOMUS, si el resultado de la conexión Bluetooth resulta desfavorable, la aplicación se cerrará automáticamente y se rogará al usuario que vuelva a abrirla para intentar de nuevo establecer la conexión.

Otra particularidad de esta aplicación es que la interfaz será totalmente distinta, aparte de que cuenta con una única pantalla, se trata de un *layout* en negro, en el que toda la pantalla se presenta como superficie capaz de detectar gestos, dando la opción a la persona invidente o con baja visón de dibujar en cualquier parte de ella.
Esto se logra creando un *layout* que ocupe toda la pantalla y en el que toda su superficie esté ocupada por una vista customizada "*Gesture Area*" en la que se permite el dibujado de gestos.



Para el reconocimiento de los gestos se crea una librería (library) con los diferentes gestos, dibujados de diferentes formas para ayudar en el reconocimiento **de ellos.**

Se usará la herramienta **Gesture Builder** incluida en SDK Android para ello.
Esta herramienta permite dibujar los gestos y después guardarlos añadiéndoles una identidad, pudiendo incluso guardar varios gestos con la identidad repetida si así se

desea. Para ser más precisos se usará un dispositivo Android emulado y se dibujarán los gestos con el ratón del ordenador.
Una vez creada la librería de gestos se guarda en la carpeta *Resources*, y desde allí los utilizará la aplicación.

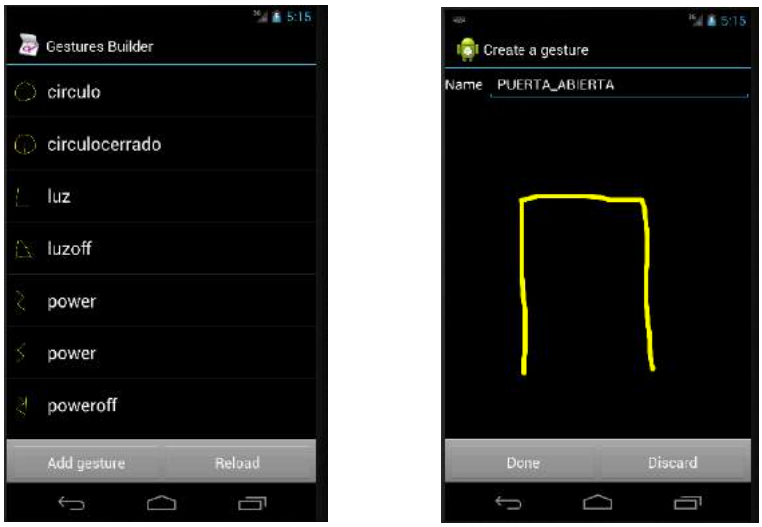


Ilustración 34: Gesture Builder

Una vez creada la librería se desarrolla la parte de código que estará a la escucha de los dibujos que se hagan en pantalla y decidirá si coincide con un gesto de la librería o no, y actuará en consecuencia mandando la trama por Bluetooth o informando al usuario del gesto no reconocido.

Para lograrlo se usa el método *onGesturePerformed()* que realizará una predicción de semejanza entre el gesto dibujado y los existentes en la librería de gestos realizada con el *Gesture Builder*.

```
@Override
//Definimos el metodo que se encarga de reconocer los gestos dibujados en pantalla y compararlos con la biblioteca creada
public void onGesturePerformed(GestureOverlayView overlay, Gesture gesture) {
    ArrayList<Prediction> predictions = gesLib.recognize(gesture);

    if (predictions.size() > 0) {
        Prediction prediction = predictions.get(0);
        compruebaBluetooth();

        if (prediction.score > 1.0) {
            String nombregesto = prediction.name;
            if(!Bluetooth_movil.isEnabled()) {
                compruebaBluetooth();
            }

            else {
                deteccion(nombregesto);
            }
        }
        else {
            mensajecorto("Gesto no reconocido");
            soundPool.play(soundID13, 1, 1, 1, 0, 1f);
        }
    }
}
```

En caso de resultado de predicción positiva (hay coincidencia con un gesto) se enviará dicho gesto al método *detección()*, que se encargará de envian la trama correspondiente

mediante Bluetooth al Arduino, según el gesto que se haya reconocido; Si por el contrario el resultado de la predicción es negativo, se informará al usuario mediante mensaje de audio que el gesto no ha sido reconocido.

```
//Según el gesto reconocido hace las llamadas a las funciones de los acturadores pertinentes
private void deteccion(String nombregesto) {

    if (nombregesto.equals("puerta")) {
        abrirPuerta();
    }

    else if (nombregesto.equals("puertacerrada")) {
        cerrarPuerta();
    }
```

Aunque el sistema de *Feedback en CAECUS* en términos de código y funcionamiento es el mismo, existe una pequeña diferencia que radica en la forma que llegan al usuario estos mensajes informativos.

Cómo se ha visto en DOMUS los mensajes llegan en cuadros de texto, y el estado de los actuadores se observa en imágenes dinámicas que cambian según el estado de éstos. En CAECUS no se pueden usar recursos que requieran de visión de pantalla, y por ello cuando la aplicación recibe el feedback desde Arduino, informa al usuario desde mensajes de audio pregrabados.

Para llevar a cabo esta tarea se ha usado una herramienta de Android llamada **Soundpool** que permite el reproducido de sonidos en formato .ogg e incluso customizarlos bajo una serie de parámetros. Se graban los mensajes informativos pertinentes en dicho formato, y se insertan en la carpeta *Resources* para que la aplicación pueda hacer uso de ellos cuando corresponda.

```
//Fijamos los botones físicos del movil para que usuario pueda ajustar el volumen
this.setVolumeControlStream(AudioManager.STREAM_MUSIC);

//Cargamos los distintos sonidos y los hacemos compatibles para distintas versiones de android
if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.LOLLIPOP) {

    AudioAttributes audioAttrib = new AudioAttributes.Builder()
            .setUsage(AudioAttributes.USAGE_GAME)
            .setContentType(AudioAttributes.CONTENT_TYPE_SONIFICATION)
            .build();
    soundPool = new SoundPool.Builder().setAudioAttributes(audioAttrib).setMaxStreams(1).build();
}
else {

    soundPool = new SoundPool(1, AudioManager.STREAM_MUSIC, 0);
}


soundPool.setOnLoadCompleteListener(new OnLoadCompleteListener() {
    @Override
    public void onLoadComplete(SoundPool soundPool, int sampleId, int status) {
        loaded = true;
    }
});

//Damos valor las variables que corresponderán a cada mensaje informativo
soundID1 = soundPool.load(this, R.raw.puertaabierta, 1);
soundID2 = soundPool.load(this, R.raw.puertacerrada, 1);
soundID3 = soundPool.load(this, R.raw.valvulaon, 1);
soundID4 = soundPool.load(this, R.raw.valvulaoff, 1);
soundID5 = soundPool.load(this, R.raw.poweron, 1);
soundID6 = soundPool.load(this, R.raw.poweroff, 1);
soundID7 = soundPool.load(this, R.raw.luzon, 1);
soundID8 = soundPool.load(this, R.raw.luzoff, 1);
```

Además se configura el audio de la aplicación, para que éste funcione independientemente del que esté configurado en el Smartphone del usuario, logrando así que los mensajes informativos suenen siempre, incluso con el móvil en silencio

Por último, otra función única en CAECUS es la de apertura rápida de puerta, mediante el agitado del Smartphone, haciendo uso del acelerómetro del Smartphone gracias al método *sensorEventListener()* y *onSensorChanged()* que detectarán cambios en la posición y velocidad sufrida por el Smartphone.

Tras varias pruebas se fijan unos parámetros para que el evento no "salte" con movimientos leves que una persona puede hacerle al móvil, así como para que sólo se active con movimientos ascendentes y evitar que se active con caídas.

```
//Metodo que detecta que el dispositivo ha sido agitado por encima de un margen, en caso positivo llama a abrirpuerta()
private final SensorEventListener mSensorListener = new SensorEventListener() {

    //Se establecen los parámetros de modo que sean detectados sólo aceleramientos negativos(hacia arriba) rápidos
    public void onSensorChanged(SensorEvent se) {
        float x = se.values[0];
        float y = se.values[1];
        float z = se.values[2];
        mAccelLast = mAccelCurrent;
        mAccelCurrent = (float) Math.sqrt((double) (x*x + y*y + z*z));
        float delta = mAccelCurrent - mAccelLast;
        mAccel = mAccel * 0.9f + delta;
        if (mAccel > 8) {
            abrirPuerta();
        }
    }

    public void onAccuracyChanged(Sensor sensor, int accuracy) {
    }
};
```

Cuando el acelerómetro detecte un movimiento acelerado en sentido ascendente, enviará por Bluetooth la trama que corresponde a abrir la cerradura, y una vez la reciba el Arduino, éste accionará el actuador cerrojo.

Los métodos de desconexión y detección de pérdida de conexión serán idénticos a los utilizados en DOMUS.

4.3 Desarrollo del código Arduino

El código desarrollado en Arduino se detallará a continuación, acompañado de referencias al código y detallando los pines usados. Es importante recalcar que para evitar un bloqueo del puerto serie y mantenerlo libre para futuras operaciones en él, se usará la herramienta **Software Serial** de Arduino, convirtiendo dos pines en desuso como Tx/Rx y dejando los originales libres.

Se desarrollan códigos diferentes para el funcionamiento óptimo de DOMUS y CAECUS, pero finalmente se opta por un código único que encaje bien con ambas, y así poder mostrar el funcionamiento de ambas sin tener que acceder al Arduino a cambiar el programa cada vez que se quiera usar una aplicación diferente.

El listado de pines usados y su control correspondiente es la siguiente:

- Pin 2 (Tx): conectado al pin Rx del módulo Bluetooth.
- Pin 3 (Rx): conectado al pin Tx del módulo Bluetooth.
- Pin 6 (PWM): controla a la lámpara de Leds.
- Pin 7: controla a la cerradura (a través del circuito interruptor).
- Pin 8: controla a la válvula (a través del circuito interruptor).
- Pin 9: controla a la base de enchufe (a través del Relé).
- Pin 12: controla la retro iluminación de DOMUS.
- Pin 13: controla la retro iluminación de CAECUS.

El código se puede dividir en dos partes bastante diferenciadas:

4.3.1 Recepción de datos y ejecución de órdenes

En el código Arduino están definidas todas las tramas que pueden ser enviadas a través de las aplicaciones.

Cuando una trama de datos es recibida (las tramas acaban cuando se reciben el carácter de fin de trama) ésta es comparada con las definidas, y si coincide con alguna de ellas se lleva a cabo la ejecución de ella, pudiendo ser el activado o desactivado de un pin.

Si esta trama es de tipo numérico corresponde al brillo de la lámpara de LEDs, y será ignorada siempre que la lámpara no esté encendida. En caso contrario, se procede a realizar una conversión de la secuencia a un numero entero y se establece el pin correspondiente la lámpara, dotado de PWM, a la intensidad recibida.

Ilustración 35: Pantalla de código de DOMUS en Arduino Tool

Además, cuenta con un contador que se encargará de desactivar el Pin que gobierna el actuador cerrojo a los 10 segundos de que este ha sido activado, con el fin de evitar calentamientos y añadir seguridad al sistema.

4.3.2 Análisis de estado y envío de datos

Cuando al Arduino le llega una trama que corresponde con la función *informe(),* éste realiza un análisis completo del estado de sus pines conectados a actuadores y manda sus resultados mediante el módulo Bluetooth al Smartphone del usuario, haciéndole conocedor del estado actual de todos los actuadores del sistema.

También enviará feedback al usuario cada vez que este envíe una trama de activación/ desactivación de un actuador, a modo de confirmación de que la orden ha llegado correctamente y se ha ejecutado de forma adecuada.

5. Integración, pruebas y resultados

Se realizaron pruebas tanto a la parte Hardware del proyecto (maqueta demostradora) como a la parte Software (aplicaciones móviles), de robustez, efectividad y prueba de errores durante la elaboración del sistema. Así como pruebas finales una vez terminado el proyecto.

5.1 Pruebas Hardware en prototipo

Se realizan unas pruebas preliminares antes de realizar las conexiones finales y encerrar el Arduino y el circuito interruptor en la caja de herramientas.

Para dichas pruebas usaremos una placa de entrenamiento o *protoboard* donde realizaremos el circuito que más tarde implementaremos en dos pequeñas placas de circuito impreso perforadas

.

Probaremos los MOSFET y la fuente, comprobamos si ésta es capaz de accionar y mantener activados todos los actuadores a la vez, y sin perder efectividad.

5.2 Pruebas Hardware en maqueta demostradora

Primero se realizan pruebas de conexión, se asegura el correcto funcionamiento del cable de corriente general, comprobando con un multímetro que al activar el interruptor disponemos de los 220 VAC correspondientes.

También se llevarán a cabo pruebas de conexión entre todos los cables conectados, primero con el multímetro timbrando para comprobar la continuidad eléctrica y una vez asegurado, cerramos el circuito conectando a la fuente y realizamos pruebas de correcto funcionamiento.

5.2.1 Pruebas en la fuente de alimentación

Se comprobará que los cables de alimentación están conectados correctamente, una vez confirmado se procede a conectar a la pared el cable de corriente general y se enciende su interruptor.

Con la fuente en funcionamiento se proceden a realizar medidas a su salida con un multímetro. Con el regulador de tensión que lleva incorporado, se ajusta a la tensión a 10 V, que es la deseada para el correcto funcionamiento de todo el sistema y evitar que el Arduino se caliente demasiado (se recuerda que el Arduino posee un regulador interno de tensión, y que tiene una alimentación recomendada entre 7 y 12 V).

Una vez medida la tensión a la salida de la fuente y comprobados los 10V, se procede a conectar en los bornes de la salida el adaptador que alimentará el Arduino, y se conectará al mismo. Se comprueba el correcto funcionamiento del microcontrolador y que no se calienta en exceso.

5.2.2 Pruebas en actuadores

Una vez encendido el sistema se comprueba que ningún actuador se activa involuntariamente, así como que no se calientan y mantienen la temperatura.

Acto seguido se procede a probar su activación, accionándolos desde las aplicaciones y observando su correcto comportamiento. Para ello probamos a activar los distintos actuadores desde sus botones en DOMUS, como dibujando los gestos respectivos en CAECUS.

También se realizan pruebas específicas a cada actuador:

Pruebas en actuador cerrojo

- Se comprueba que pasados 10 segundos desde la activación del cerrojo éste se cierra automáticamente, como se desea.
- Por último se comprueba que el LED verde de estado del cerrojo se enciende cuando éste se activa, y se apaga cuando el cerrojo se desactiva.

Ilustración 36: Cerrojo activado

Pruebas en actuador electroválvula

- Se confirma que en estado cerrado no puede fluir el aire de un extremo al otro.
- Se acciona la válvula desde ambas aplicaciones, se realiza la prueba de que una vez abierta permite el paso de aire.
- Al igual que en el cerrojo, se confirma que el LED verde de estado de la válvula acompaña al estado activo de esta, encendiéndose sólo mientras la válvula esté activa.

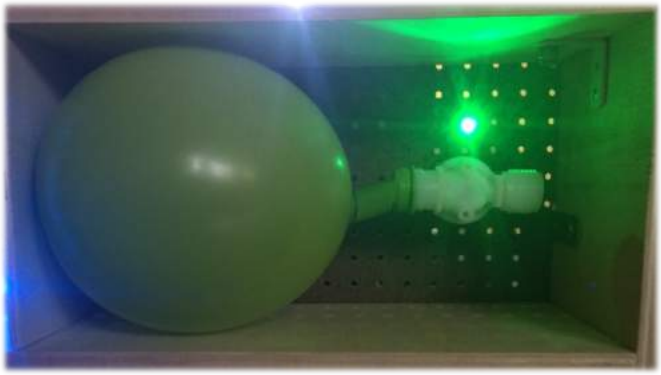


Ilustración 37: Válvula activada

Pruebas en la base de enchufe

- Se mide la tensión en los bornes de la base de enchufe, comprobando que con el sistema encendido y el relé activado llegan los 220 VAC correspondientes.

- Se comprueba conectando diferentes elementos de corriente alterna que cuando se activa el relé que gobierna la base, ésta es capaz de entregar corriente suficiente para activarlos.

- Con un busca polos se verifica que fase y neutro están correctamente colocados. También se comprueba con el timbrado del multímetro que la tierra se ha conectado correctamente.

Ilustración 38: Base de enchufe activada

Pruebas en la lámpara de LEDs

- Se realizan pruebas de cambio del brillo desde DOMUS, comprobando que el brillo de la lámpara concuerda con el movimiento de la barra deslizante de la aplicación.

- Se comprueba que todos los LEDs están correctamente conectados y se encienden todos.

Ilustración 39: Lámpara de LEDs encendida

5.2.3 Pruebas en Arduino y módulo Bluetooth

Para las primeras pruebas del Arduino se implementaron las funciones de prueba que lleva el IDE de Arduino incluidos, se procedió a descargar estos programas dentro de la tarjeta microcontroladora y se comprobó su funcionamiento, midiendo con el polímetro la tensión y la corriente entregada por un pin activo.

También, una vez creada la primera aplicación prototipo, se realizaron pruebas del funcionamiento del conjunto Arduino + Bluetooth. Además, con el osciloscopio se comprobó el funcionamiento de los pin PWM, muy importantes de cara al ahorro energético y sobre todo, para la función de control de brillo en la lámpara de LEDs.

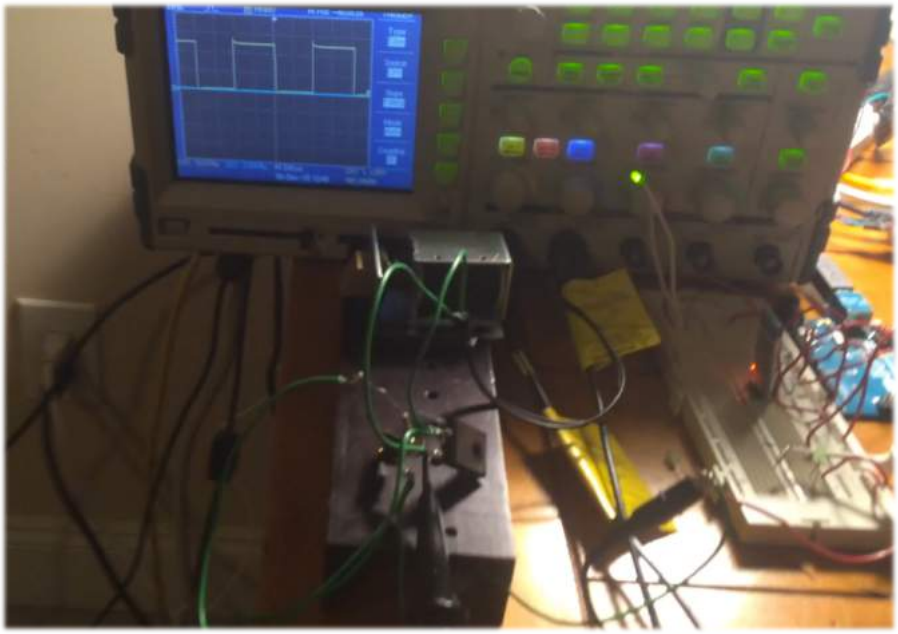


Ilustración 40: Pruebas de PWM realizadas con osciloscopio al Arduino

Con el módulo Bluetooth se realizarán pruebas de conexión a distintos dispositivos Bluetooth, como Smartphones y Tablets e incluso otros módulos Bluetooth. Usaremos programas como ***CoolTerm*** y aplicaciones móviles como ***BlueTerm***, que son interfaces para PC y móvil respectivamente, capaces de enviar y recibir datos por medio de Bluetooth, mostrando en pantalla la información enviada y recibida.

Se realizan pruebas de envío de caracteres, medida de potencia de emisión/recepción, alcance de distancia sin errores y máximo de **Baud Rate** sin errores.
Obtenemos resultados muy favorables para el proyecto, arrojando las pruebas una distancia máxima de envío sin error de 40 metros y un *Baud Rate* de 1382400.

5.3 Pruebas Software

Se cargarán ambas aplicaciones en distintos tipos de dispositivos Android (tabletas y teléfonos) y de distintas versiones de Android, siempre y cuando no sean inferiores a la ya muy anticuada *4.0.4 Ice Cream Sandwich*.

Serán realizadas pruebas de robustez y funcionamiento en ambas aplicaciones móviles creadas. También se llevarán a cabo diversas pruebas de reconocimiento de desconexión: por distancia, desactivación de Bluetooth local y Pérdida de alimentación en módulo Bluetooth remoto.

5.3.1 Pruebas en DOMUS

- Será testeado el funcionamiento de los botones en un uso extremo, para ellos se intensifica el presionado de ellos y se observa que el comportamiento sea correcto, actuando siempre en consecuencia y sin quedarse bloqueada la aplicación.

- Se pone a prueba la detección de pérdida de Bluetooth por los tres métodos, así como que la aplicación responde correctamente ante ella.

- Se verifica que los mensajes informativos llegan correctamente, observando que tanto el momento y duración como el contenido sean el adecuado.

5.3.2 Pruebas en CAECUS

- Se realizan dibujos con pequeñas variaciones y se comprueba que siguen siendo reconocidos correctamente, independientemente de tamaño con el q se dibujen o pequeñas desviaciones.

- Se realizan gestos que no existen y se comprueba que la aplicación informa de que es un gesto no válido.

- Se comprueba que aun con el Smartphone en modo silencio, los mensajes de audio informativos siguen sonando.

- Se verifica que todos los mensajes de audio informativos son correctos y no se solapan.

6. Conclusiones y trabajo futuro

6.1 Conclusiones

Se analizan los resultados del proyecto llevado a cabo para comprobar si se han cumplido objetivos previstos y concluye en que se han alcanzado todas las metas satisfactoriamente.

- Se ha desarrollado e implementado un sistema de domótica de bajo coste con cuatro actuadores controlados independientemente a través de un Smartphone.

- Se han desarrollado dos aplicaciones robustas y estables para el mismo fin usando recursos totalmente distintos y funcionalidad idéntica.

- Se han desarrollado un sistema de domótica adaptado a personas con discapacidad visual.

- Se ha desarrollado una maqueta demostradora para que futuros alumnos puedan interactuar con ella y ver un ejemplo de los proyectos realizados en el DSLab.

- Se ha realizado un entrenamiento y aprendizaje en el uso de herramientas Android así como en Arduino.

- Se ha realizado un entrenamiento y aprendizaje en el uso de tecnología y herramientas Bluetooth.

- Se han sentado bases para la integración de sistemas inteligentes y nuevas tecnologías relacionados con domótica en hogares invidentes.

6.2 Trabajo futuro

Ya se ha conseguido un sistema totalmente funcional, pero aún puede mejorarse en diversos aspectos:

- **Miniaturización**: Aunque el sistema microcontrolador no destaque por su gran tamaño, con la salida al mercado de microcontroladores de tamaños mucho más reducidos podría reducir las dimensiones del conjunto microcontrolador y módulo Bluetooth a mas de la mitad, optimizando el espacio más aún.

- **Nuevas tecnologías Bluetooth**: La implementación en el sistema de un módulo con la tecnología *Bluetooth 4.0 low energy* no sólo convertiría el sistema en puntero tecnológicamente hablando, sino que además reduciría el consumo energético,: punto que trataremos a continuación.

- **Reducir consumo eléctrico:** exprimiendo más el uso de los pines PWM (Pulse Width Modulation) podríamos reducir el consumo eléctrico reduciendo la tensión que entregamos a los actuadores. Sabemos que una vez superado el pico que necesita para activarse, la tensión necesaria para mantenerlos activados es menor, pudiendo regularla gracias a la tecnología PWM aplicada al MOSFET.

- **Zigbee:** usando para comunicar cada uno de los módulos actuadores, evitando así el cableado que los conectan al Arduino. Se sabe que esta tecnología tiene muy bajo consumo y funciona muy bien en sistemas de domótica, ya que no es necesario el uso de una comunicación con gran tasa binaria.

- **Actuadores automáticos:** dado el aprendizaje de la tecnología Bluetooth se ha pensado en la capacidad de ella para realizar acciones automáticamente mediante el mero hecho de detectar su presencia, sin necesidad de aplicaciones ni movimientos. Pudiendo prescindir del Smartphone para realizar acciones como abrir una puerta.

- **Ampliación:** debido a la estructura modular de las aplicaciones, es muy sencillo añadir nuevas funciones que correspondan al accionado de nuevos actuadores, tales como toldos, persianas, alarmas o incluso termostatos.

Referencias

[1] “Desarrollo de aplicaciones para Android (manual imprescindible)”, Joan Ribas Lequerica, 2015.
[2] “The Fundamentals of Short-Range Wireless Technology”, Lou Frenzel, 2012.
[3] “Bluetooth: Criterios de Selección y Comparativa con Otras Tecnologías Inalámbricas”, Carlos Marín Pascual, 2012.
[4] “Página web de Android developers”: http//developer.android.com/
[5] “Página web de Arduino” : https://www.arduino.cc/
[6] “Página web de Elekfreks” (documento de comandos AT de los módulos Bluetooth): http://www.elekfreaks.com/
[7] “VISHAY” (datasheet MOSFET): http://www.vishay.com/docs/91054/91054.pdf

Anexos

A. MÓDULO BLUETOOTH

Se adquirieron varios módulos Bluetooth (HC-05 y HC-06) y se realizó un entrenamiento y todo tipo de pruebas entre ellos mismos y con otros dispositivos. Se presenta una breve guía de lo aprendido y que puede ser útil para entender y usar el sistema desarrollado.

Los módulos Bluetooth se dividen en dos grandes clases, los módulos **Master** y los módulos **Slave**.

Los primeros contienen un set de instrucciones muy amplio y son totalmente configurables, y son capaces de realizar conexiones por ellos mismos, sin necesitar que nadie les pida conectarse a ellos, y pueden conectarse hasta otros 5 módulos. Además, son capaces de ser configurados en modo Slave y funcionar como tal.

Por el contrario los módulos de tipo Slave tienen un set de instrucciones mucho más escueto, y sólo realizan conexiones Bluetooth bajo demanda de un módulo Master, es decir, no pueden tener iniciativa propia a la hora de realizar una conexión.

Ambos tipos de módulo Bluetooth fueron insertados y testeados en el proyecto, dando mejor resultado los Slave, si bien por su docilidad y pasividad como por su sencillez. Sumando la ventaja de que cuentan con un precio más reducido.

Para la comunicación directa con éstos módulos y poder configurarlos a gusto del usuario contienen un set de instrucciones denominado **Comandos AT**. Por medio de estos comandos es posible incluso contactar con un módulo Bluetooth que esté conectado, de modo que si le mandamos un comando de este tipo, en lugar de enviarlo a destinatario como haría con cualquier otra trama, la interpreta como un mensaje a si mismo y nos responde en consecuencia.

Se realizan múltiples pruebas aprovechando la versátil lista de comandos del módulo Master, para comprobar la potencia de recepción y transmisión de dichos módulos variando distancias y obstáculos, obteniendo resultados prometedores.

A.1 CONFIGURACIÓN DEL MÓDULO HC-06

Presentamos a continuación algunos de los comandos del set de *Comandos AT* del módulo HC-06 (Slave) más útiles para este proyecto. Recordemos que en el caso del Master la lista de comandos contiene muchas más funciones, incluyendo auto conexión y descubrimiento de módulos Bluetooth cercanos, midiendo su potencia.

Comando AT	Respuesta	Función
AT	OK	Comprobación de funcionamiento
AT+BAUDx	OKxxxx	Cambia el bit rate de la comunicación, siendo x un número entre 1 y 9 que corresponde a una velocidad proporcional.
AT+NAME*xxxx*	OK*xxxx*	Cambia la denominación del dispositivo al que elijamos. (x = nombre)
At+PINxxxx	OKsetpin	Cambia la contraseña del módulo Bluetooth a la que elija el usuario. (xxxx = contraseña)

Los parámetros iniciales que se deben configurar en la interfaz que usemos para conectarnos al módulo HC-06 son:

- Baud Rate: 9600
- Parity: None
- Data bits: 8
- Stop bits:1
- Contraseña: 1234

Además el fin de línea solo debe llevar retorno de carro, **sin ajuste de línea** (*line feed*)

Para comunicarnos con el módulo Bluetooth enviarle estos comandos se ha realizado tanto mediante una aplicación móvil de tipo terminal llamada **BlueTerm**, cómo conectando al ordenador el módulo Bluetooth mediante un conversor USB a *señal TTL*, y una interfaz de puerto serie llamada **CoolTerm.**

A.2 ANÁLISIS DE ESTADOS DEL MÓDULO BLUETOOH

Con el objetivo de hacer entender mejor como se configura un módulo y su funcionamiento, se ha realizado un diagrama de flujo detallando los distintos estados en los que puede encontrarse el módulo Bluetooth y cómo es su transición, detallando los *Comandos AT* necesarios para pasar de un estado a otro.

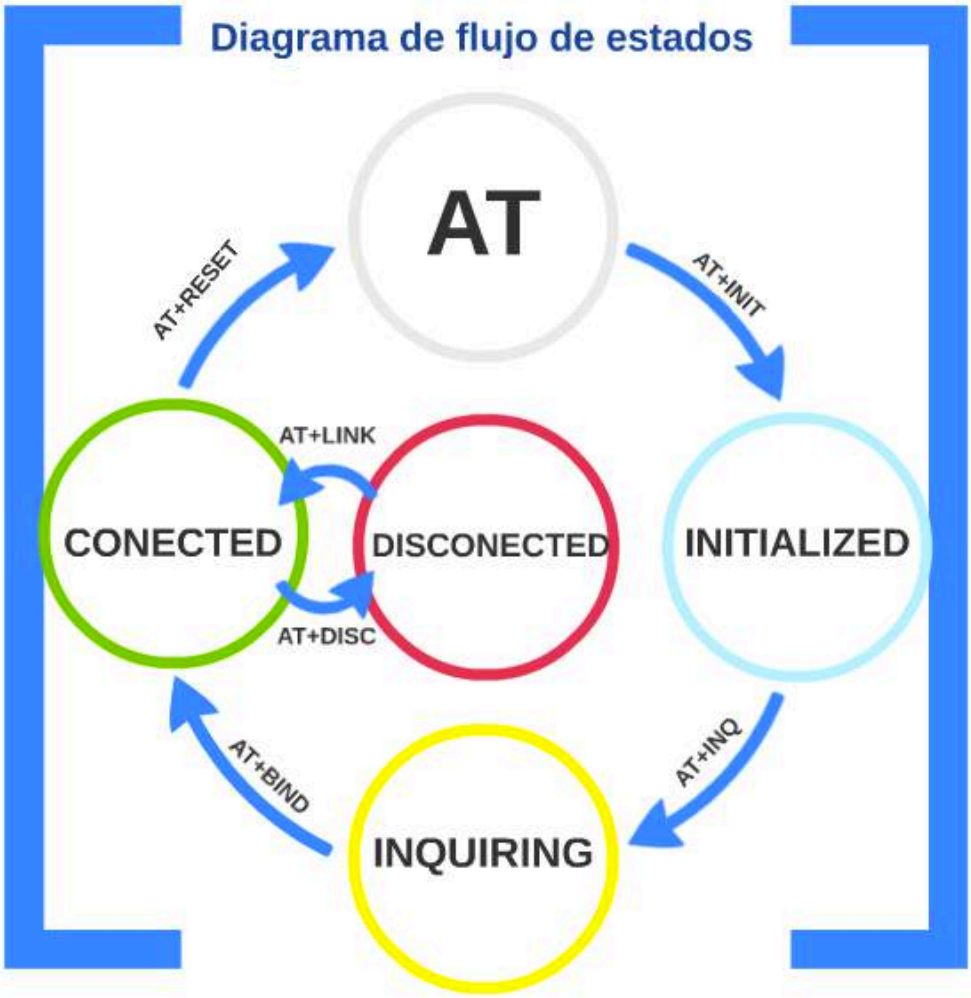


Ilustración 41: Flujo de estados del módulo Bluetooth

B. MANUAL DE INSTALACIÓN

Para el uso de la maqueta sólo hacen falta unos sencillos pasos:

1. Conectar la toma de corriente general de la maqueta y pulsar el interruptor. Observar que el LED informativo de la fuente de alimentación se enciende, junto al LED rojo del módulo relé y también se aprecia un doble parpadeo en la retro iluminación del tejado CAECUS.

2. Descargar el código de Arduino en la tarjeta microcontroladora a través del puerto serie USB y la herramienta Arduino Tool.

3. Descargar la APK de la aplicación DOMUS o CAECUS en cualquier dispositivo Android.

4. Abrir el Bluetooth del dispositivo y enlazarte al Bluetooth llamado "DOMUS UAM", la contraseña es 1414.

5. Abrir cualquiera de las aplicaciones descargadas y conectarte al Bluetooth "DOMUS UAM", en el caso de CAECUS la conexión será automática.

C. MANUAL DE USO DE CAECUS

Para el uso de la aplicación CAECUS, especialmente diseñada para personas con discapacidad visual, ton solo hace falta abrirla estando la maqueta encendida y en un radio de 25 metros de nosotros. La conexión se realizará automáticamente, en caso contrario la aplicación se cerrará automáticamente y nos pedirá que repitamos el proceso acercándonos más.

Una vez abierta ya podemos dibujar los gestos que controlan los actuadores, se presentan a continuación la lista de gestos y el modo en que deben ser dibujados, es importante que se hagan en el sentido en que viene indicado por la flecha.

Gestos predeterminados de CAECUS:

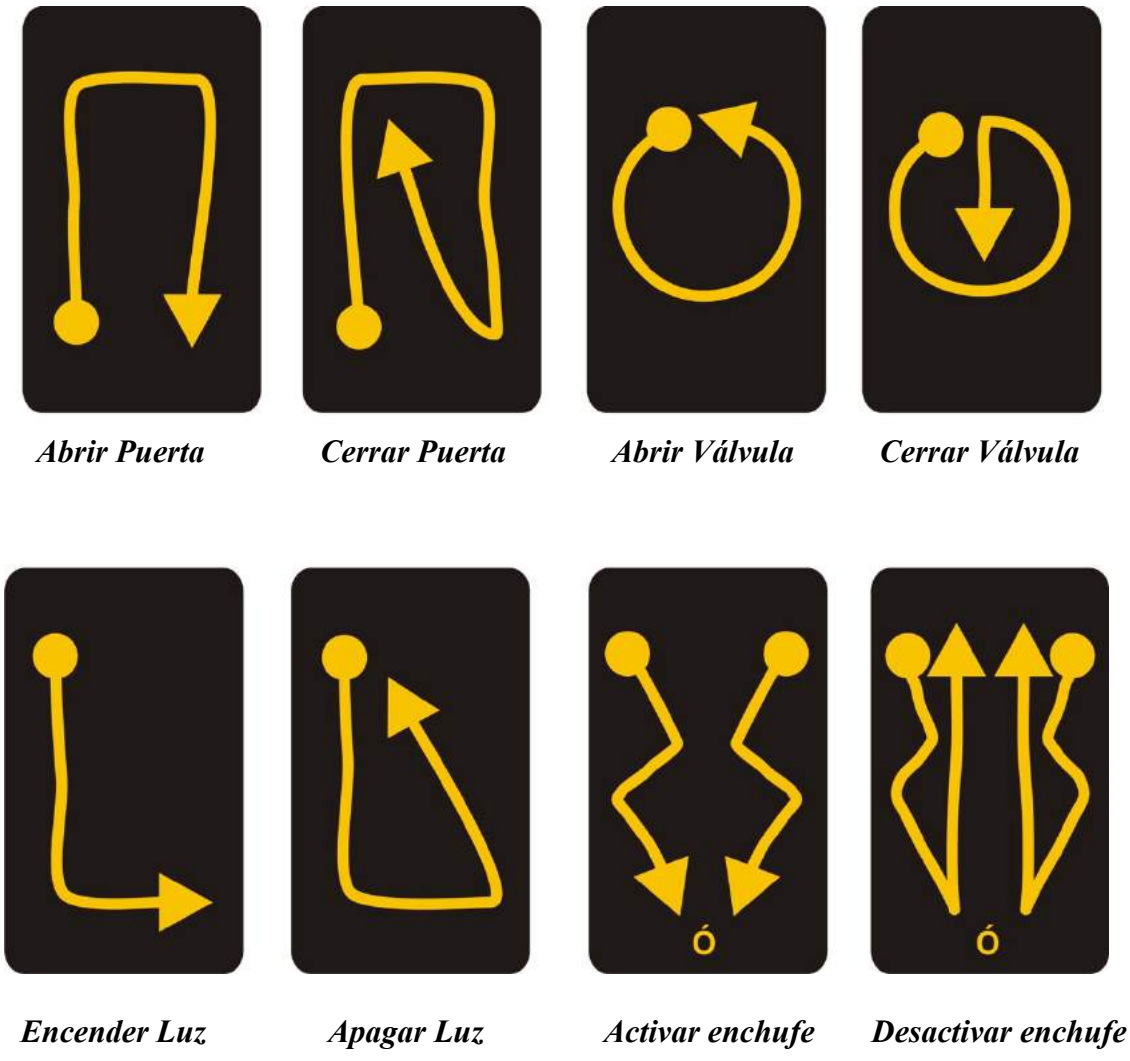


Ilustración 42: Manual de gestos CAECUS

D. Presupuesto

1) Ejecución Material

- Terminal Móvil Android: ..... 150,00 €
- Material para la maqueta demostradora:
  - Fuente de alimentación ..... 35,85 €
  - Caja de instrumentos..... 12,00 €
  - Cerrojo de solenoide ..... 10,68 €
  - Válvula de solenoide..... 16,00 €
  - Módulo Relé de Arduino ..... 5,50 €
  - Base de enchufe con carril DIN ..... 11,35 €
  - Arduino UNO ..... 22,00 €
  - Cables, MOSFET, resistencias, LEDs, diodos, etc..... .5,00 €
  - Maderas, metacrilato, tornillos y herramientas..... 43,00 €
- Material de oficina ..... 150,00 €
- **Total de ejecución material ..... 451,38 €**

2) Gastos generales

- 16 % sobre Ejecución Material ..... 72,22 €

3) 3) Beneficio Industrial

- 6 % sobre Ejecución Material ..... 27,08 €

4) 4) Honorarios Proyecto

- 1440 horas a 15 €/ hora ..... 21.600,00 €

5) 5) Material fungible

- Gastos de impresión..... 100,00 €
- Gastos de encuadernación..... 200,00 €

6) 6) Subtotal del Presupuesto

- Subtotal del Presupuesto ..... 22.450,68 €

7) 7) I.V.A. aplicable

- 21 % sobre el Subtotal del Presupuesto ..... 4.714,64 €

8) Total presupuesto…….27.165,32 €

Madrid, Julio de 2016

El Ingeniero Jefe de Proyecto

Fdo.: Luis Pablo González Marabini

Ingeniero de Telecomunicación

E. Pliego de condiciones

Este documento contiene las condiciones legales que guiarán la realización, en este proyecto, *CONTROL DE ACCESO, OBJETOS Y EQUIPOS ELÉCTRICOS MEDIANTE TELÉFONO INTELIGENTE* . En lo que sigue, se supondrá que el proyecto ha sido encargado por una empresa cliente a una empresa adjudicataria 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 adjudicadora se reserva el derecho a declararlo desierto.

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

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

4. El proyecto 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 adjudicatario 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 adjudicatario 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 adjudicatario después de confrontarlas.

7. Se abonará al adjudicatario la parte del proyecto 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 proyecto siempre que dicha ejecución 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 parciales como en la liquidación final, se abonarán los trabajos realizados por el adjudicatario a los precios de ejecución material que figuran en el presupuesto para cada unidad del proyecto.

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 Proyecto, 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 ejecución, quedará el contratista obligado a conformarse con la rebaja acordada.

10. Cuando se juzgue necesario emplear materiales o ejecutar trabajos que no figuren en el presupuesto de la contrata, se evaluará su importe a los precios asignados a otras trabajos o materiales análogos si los hubiere y cuando no, se discutirán entre el Ingeniero Director y el adjudicatario, 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 adjudicatario, con autorización del Ingeniero Director de Proyecto, 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 del Proyecto, o en general, introduzca en él cualquier modificación que sea beneficiosa a juicio del Ingeniero Director de Proyecto, no tendrá derecho sin embargo, sino a lo que le correspondería si hubiera realizado el Proyecto con estricta sujeción a lo proyectado y contratado.

12. Las cantidades calculadas para partes 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 adjudicatario 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 del Proyecto, 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 del trabajo ejecutado, de acuerdo con los precios del presupuesto, deducida la baja si la hubiera.

17. La fecha de comienzo del Proyecto será a partir de la fecha de firma del contrato. La recepción provisional se realizará a la finalización y entrega del Proyecto, y la

definitiva, al año de haber ejecutado la provisional, procediendo la empresa adjudicataria, si no existe reclamación alguna, a la reclamación de la fianza.

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

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

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

21. El adjudicatario, deberá realizar el Proyecto 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 del Proyecto, 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 adjudicatario.

22. Hecha la recepción provisional, se certificará al adjudicatario el resto del Proyecto, reservándose el cliente el importe de la fianza hasta su recepción definitiva que será 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 adjudicatario de acuerdo con las condiciones económicas legales establecidas.

23. Las tarifas para la determinación de honorarios se calcularán según la legislación vigente en la materia.

Condiciones particulares

La empresa adjudicataria, 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 adjudicataria representada por el Ingeniero Director del Proyecto.

2. La empresa adjudicataria 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 adjudicataria.

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 adjudicataria.

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 adjudicataria decidirá aceptar o no la modificación propuesta.

7. Si la modificación se acepta, la empresa adjudicataria 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 adjudicataria 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 adjudicataria.

10. La empresa adjudicataria 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 adjudicataria 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 adjudicataria lo estime oportuno. En caso contrario, la persona designada deberá contar con la autorización del mismo, quien delegará en él las responsabilidades que ostente.