ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA

TRABAJO FIN DE GRADO PARA LA TITULACIÓN DE

INGENIERÍA INFORMÁTICA – INGENIERÍA DE COMPUTADORES

Diseño e implementación de un sistema domótico basado en la red Mysensor.org

Realizado por

Manuel Jesús Bellido Lama

Dirigido por

Manuel Jesús Bellido Díaz

Departamento

DEPARTAMENTO DE TECNOLOGÍA ELECTRÓNICA

Sevilla, Septiembre de 2017

Agradecimientos

Me gustaría agradecer en este apartado, a todos los profesores y amigos que me han ayudado durante la carrera, y me han apoyado y aportado ideas hasta llegar a tener la visión que tengo actualmente. En especial a mi padre, el profesor Manuel Jesús Bellido Díaz, que me ofreció la oportunidad de hacer con él este proyecto y la ayuda necesaria durante el desarrollo del mismo.

Índice

Capítulo 1: Introducción y Objetivos

1.1 Introducción

En los albores del siglo XXI, sería difícil dudar de la creciente importancia que están empezando a tomar los sistemas domóticos en los nuevos diseños de nuestras casas. Así, algo tan sencillo como el control de la iluminación, por ejemplo, puede suponer un considerable ahorro de energía, del mismo modo que tareas básicas y cotidianas, como el control de riego o la apertura y cierre de persianas de la vivienda, conllevan una notable mejora en la calidad de vida de sus habitantes.

Lo que pretendemos hacer con esta propuesta de Trabajo de Fin de Grado, es explorar el desarrollo de sistemas domóticos, con el propósito de llevar a la práctica los progresos que vayamos haciendo. La idea surge a partir de un seminario al que asistí en la asignatura Laboratorio de Desarrollo hardware (LDH) de 4º curso de Ingeniería de Computadores, que fue impartido por D. Jose Ignacio Villar, egresado de la ETSII como Ingeniero Informático, y actualmente, Director Técnico de la empresa Oteara S.L. El tema que allí se trató, fue el de la implementación de sistemas domóticos caseros. D. José Ignacio Villar, estableció una comparación entre la situación de hace unos años con respecto a este tema, cuando desarrollar un sistema domótico casero requería, prácticamente desarrollar casi todos los aspectos (implementación del sistema, diseño del protocolo de comunicación, diseño de la red, etc) y la situación actual, en la que, afortunadamente, han surgido multitud de opciones, sobre todo de proyectos abiertos que facilitan significativamente la implementación de estos sistemas. Concretamente, hizo la presentación de la red domótica mysensor.org [1], llegando a implementar durante el seminario, un ejemplo de control de iluminación de una manera bastante sencilla, intuitiva y rápida y con un coste muy reducido, sobre todo si lo comparamos con la implementación de sistemas domóticos comerciales.

Así pues, como he apuntado más arriba, me he apoyado en los resultados expuestos durante este seminario, para proponer llevar a cabo una experiencia de implementación real de un sistema domótico en una parte vivienda basado en la red mysensor.

En concreto, lo que se plantea es estudiar las características de la planta baja de la vivienda del autor del trabajo, para llevar a cabo la domotización de la misma en aquellos aspectos que puedan ser más útiles.

1.2 Objetivos del trabajo

El objetivo de este trabajo de fin de grado será diseñar e implementar un modelo de domotización de la planta baja de la vivienda del autor (incluyen un patio, salon principal y cuarto de baño) y los diferentes dispositivos, tanto actuadores como controladores, que encontramos en cada una de ellas, por ejemplo una depuradora en el patio para una piscina o los interruptores del salón principal.

Para llevar a cabo este, nuestro fin principal, tendremos que ir cumplimentando varios objetivos que desglosaremos a continuación:

1. Realizar un análisis de requisitos que nos lleve a decidir cuáles serán los dispositivos que incluiremos en nuestro sistema domótico y cuáles dejaremos fuera de éste.

2. Desarrollar cada una de la actuaciones que aparezcan en el análisis de requisitos, comenzando con un trabajo a nivel de prototipado del sistema y una posterior evolución para la implementación final.

3.Pruebas del correcto funcionamiento de nuestro sistema.

1.3 Organización de la documentación del Trabajo Fin de Grado

En base a los puntos expuestos anteriormente, el resto de la documentación del TFG se organiza de la siguiente manera:

En el capítulo 2 se va a realizar el análisis de requisitos del sistema y se establecerán las especificaciones del mismo. También se presentarán las diferentes herramientas que se han usado, tanto para el diseño de la placa como para la implementación del sistema domótico.

En el capítulo tres se presentará el proceso de diseño y fabricación que se ha llevado a cabo con las placas, así como el diseño e implementación del sistema domótico en cada una de las estancias.

En el capítulo cuatro se mostrarán los resultados finales del proyecto, incluyendo los problemas que han ido surgiendo durante la implementación del mismo y las soluciones que se han propuesto para resolver dichos problemas. También se valorarán los posibles aspectos negativos que se han detectado.

Por último, se mostrarán las conclusiones más relevantes, obtenidas tras la realización de este proyecto.

Capítulo 2: Análisis y especificaciones

En este capítulo vamos a hacer una pequeña revisión del estado del arte en el mundo de la domotización, así como las distintas posibilidades que podemos encontrar para el desarrollo de este tipo de sistemas.

A continuación, se presentará el análisis de los requisitos del sistema y, posteriormente, las diferentes tecnologías que se van a emplear en el desarrollo del trabajo.

2.1 Estado del arte

A medida que avanza la tecnología, aumenta el número de dispositivos que podemos llamar inteligentes y que se van instalando en nuestras viviendas. Así, crece por ejemplo, la cantidad de electrodomésticos con la capacidad de usar el internet de las cosas y que pretenden facilitarnos su uso y hacernos la vida más fácil.

Son ya bastante conocidos los sistemas en viviendas que permiten controlar la iluminación a través del encendido de bombillas, o el control de persianas y/o cortinas de manera automática. En estos sistemas hay un elemento principal o controlador, que es el que toma las decisiones sobre qué hacer, y unos elementos de sensado actuadores, que realizan la función de obtener datos (sensado) y realizar las tareas indicadas (actuadores).

Hay ya muchas empresas que se dedican a la venta de dispositivos domóticos, muchas veces ya programados para cumplir una función concreta. Pero, en general, estos productos tienden a tener un coste elevado, fundamentalmente por la innovación que supone, aún hoy en día, la implementación de este tipo de sistemas.

En el comienzo del desarrollo de sistemas domóticos comerciales, ocurrió que cada empresa desarrolló sus propios equipos hardware, basados en sistemas de comunicaciones con protocolos que también eran propiedad de cada una de las empresas, lo que implicaba que los diferentes equipos eran incompatibles entre sí. Para evitar esta proliferación de equipos y facilitar la compatibilidad, se desarrollaron especificaciones de sistemas domóticos con la característica de que pudieran ser empleados por las diferentes empresas fabricantes de los sistemas domóticos, facilitando la compatibilidad entre equipos. Como ejemplos principales están KNX [2], o lonwork [3] . A este tipo de especificaciones de sistemas domóticos, disponibles para todas las empresas, se las ha denominado “abiertas”, por el hecho de que cualquier empresa puede acceder a ellas.

Sin embargo, queremos aclarar que no son “abiertas” en el sentido de los sistemas Software o Hardware abiertos (Open Software u Open Hardware), porque realmente son unas especificaciones que no se pueden modificar y/o distribuir libremente, sino que se accede a ellas a través de una licencia de uso.

En cualquier caso, es cierto que este tipo de especificaciones de sistemas domóticos con especificaciones disponibles para todos, es el que se emplea principalmente en los denominados sistemas domóticos profesionales (domotización de edificios, infraestructuras e instalaciones de cierta envergadura, y sistemas de cierta complejidad y/o envergadura). Quizás, el problema principal de este tipo de especificaciones es que van evolucionando de una manera relativamente lenta, si la comparamos con la evolución tecnológica. Esto es debido a que, para mantener la robustez en los sistemas, los cambios de especificaciones se hacen lentos con respecto a las tecnologías, que realmente, han evolucionado de una manera mucho más rápida.

Por otra parte, en el mundo de la domótica de viviendas unifamiliares, si bien se pueden emplear este tipo de sistema que hemos comentado antes, se ha abierto otra alternativa basada tanto en Software como en Hardware abierto (aquí “abierto” sí es en el sentido ampliamente conocido de poder modificar y/o distribuir libremente los productos) . El desarrollo, en los últimos años, de plataformas hardware abiertas de muy bajo coste, como pueden ser Arduino [4], como unidad terminal y Raspberry Pi [5], como unidad de control, ha dado lugar al desarrollo de multitud de aplicaciones para este tipo de sistemas y, en particular, al desarrollo de diversos sistemas domóticos completamente abiertos. Pasemos a comentar alguno de ellos:

Calaos [6]*:* está diseñado para automatizar tareas del hogar de forma completa, con interfaz táctil, apps web, aplicaciones nativas para iOS y Android, así como para Linux.

Domoticz [7]*:* es un sistema de automatización con soporte para diferentes dispositivos, como controles remotos, estaciones de climatología, detectores de humo, etc. Diseñado en HTML5 y accesible desde los navegadores webs e incluso optimizado para ser ligero y poderse ejecutar incluso en sistemas como la Rapsberry Pi.

Home Assistant [8]: es otra plataforma abierta que puede ser implementada en casi cualquier plataforma en la que se pueda instalar Python 3, la base para su funcionamiento.

OpenHAB [9]*:* es la abreviatura de Open Home Automation Bus, y es muy conocida. Cuenta con una gran comunidad detrás y permite usar un gran número de dispositivos compatibles. Está escrito en Java y permite diseñar incluso tu propia interfaz de usuario para tu sistema casero.

OpenMotics [10]*:* sistema domótico que tiene, tanto hardware, como software abierto, creado para dotar de un sistema completo para adherir dispositivos de terceros mediante cableado, simplificando esta labor.

MySensors: es una librería software para implementar automatización en el hogar con una comunicación segura. Cuenta con una gran comunidad de software y hardware abierto respaldándola y aporta una base que facilita el trabajo de programación, usando esta librería, puesto que tenemos códigos sencillos preparados para implementar pequeños diseños de automatización, con los que podremos entender cómo funciona. Es importante mencionar que esta librería de componentes, desarrollado en mysensor, puede emplearse con la mayoría de los controladores de sistemas domóticos abiertos, como es el caso de Calaos., Domoticz, OpenHAB, etc.

MySensors ha sido el sistema de automatización elegido para el proyecto, debido, fundamentalmente a la la librería de componentes que ya están hechos y que, como hemos mencionado antes, pueden emplearse, realmente, con la mayoría de los sistemas domóticos abiertos. Precisamente, además de la librería mysensor hay que seleccionar el controlador domótico que se va a emplear para realizar la interconexión entre todos los dispositivos hardware, así como la interfaz para poder manejar/controlar el sistema.
En mysensor existe una amplia gama de controladores disponibles, entre los cuales están la mayoría de los sistemas domóticos abiertos. En nuestro caso, hemos seleccionado el controlador denominado “mycontroller” [11]. Este controlador es el originario desarrollado para la librería mysensor y, por ello, tiene la máxima compatibilidad con los componentes de la librería.

2.2 Análisis de requisitos

Una vez decidido que vamos a llevar a cabo una experiencia real de desarrollo e implantación de un sistema domótico, vamos a plantear el análisis de requisitos que pretendemos cumplir en el desarrollo del proyecto.

Nuestro sistema final está planteado para la domotización de las tres estancias de las que contamos en planta baja de la vivienda, que son: un patio, un salón y un cuarto de baño. En cada uno de éstos, nos encargaremos de controlar distintos aspectos, con diferentes unidades terminales que incluyen sensores y/o actuadores, todas ellas gestionadas (o controladas) desde un único controlador que debe ser accesible a través de interfaz web (esto permitirá, por ejemplo, controlar el sistema desde un móvil en la misma red). Hecha esta especificación principal del sistema, vamos a establecer el análisis de requisitos, esto es , indicar concretamente qué tareas pretendemos que realice nuestro sistema, dividiéndolas por cada una de la estancias de la vivienda:

Requisito 1: PATIO:

El patio dispone de iluminación y depuradora. Se plantea realizar la automatización de ambas, para lo cual se necesita controlar la señal eléctrica que llega al patio, actualmente conectada sólo a la bombilla de iluminación.

a) Con respecto a la depuradora se plantean dos cosas: por un lado, que opere con una temporización diaria de cuatro horas de actividad, para llevar a cabo el filtrado de la piscina; por otro lado, poder controlar el encendido/apagado de la misma en cualquier momento, para poder realizar otro de tipo de tareas (limpieza de filtro, limpieza de suelos, etc). También es necesario que, cuando se deje de utilizar la piscina, la depuradora quede inactiva (corte de la señal eléctrica).

b) Con respecto a la iluminación, es importante mencionar la escasa frecuencia con que se utiliza en el patio, siendo éste un lugar muy poco transitado, cuyo uso se restringe a unos pocos días al año. Por ello, se plantea un control básico encendido/apagado similar al que se realiza actualmente mediante interruptor.

Requisito 2: SALÓN PRINCIPAL

El salón principal de la planta baja dispone de una iluminación controlada por dos interruptores, localizados en los dos puntos de acceso a la misma: por la puerta del patio y por la escalera de acceso en configuración exor. También existe un estore en la ventana.

a) Con respecto a la iluminación, se pretende seguir manteniendo el encendido/apagado desde los interruptores, así como poder controlar el encendido/apagado de manera automática, desde un dispositivo conectado a la lan del hogar.

b) Con respecto al estore, se pretende automatizar la subida/bajada del mismo, mediante interruptor o control a través de internet.

Requisito 3: CUARTO DE ASEO

Esta habitación no tiene iluminación natural y, por tanto, se debe proceder a su encendido, siempre que entre alguien y su apagado posterior, cuando quede vacío. Es por ello que se plantea el uso de un detector de presencia, para automatizar el encendido/apagado. Además, se pretende controlar el encendido/apagado de manera automática desde un dispositivo conectado a la lan del hogar.

2.3 Tecnologías empleadas

En esta apartado describiremos, de forma breve, las herramientas software y hardware, que hemos usado, tanto para el desarrollo de nuestro sistema, como para el de las PCBs que hemos creado y fabricado.

2.3.1 Mysensors

MySensors es una comunidad de hardware y software libre, que se centra en facilitar al usuario la ayuda necesaria para que pueda desarrollar por sí mismo, tanto proyectos de automatización del hogar como el internet de las cosas.

Ofrecen instrucciones de montaje, fáciles de seguir, para la conexión de componentes como radio en arduino y raspberry, o la instalación de otras herramientas como mycontroller en una raspberry.

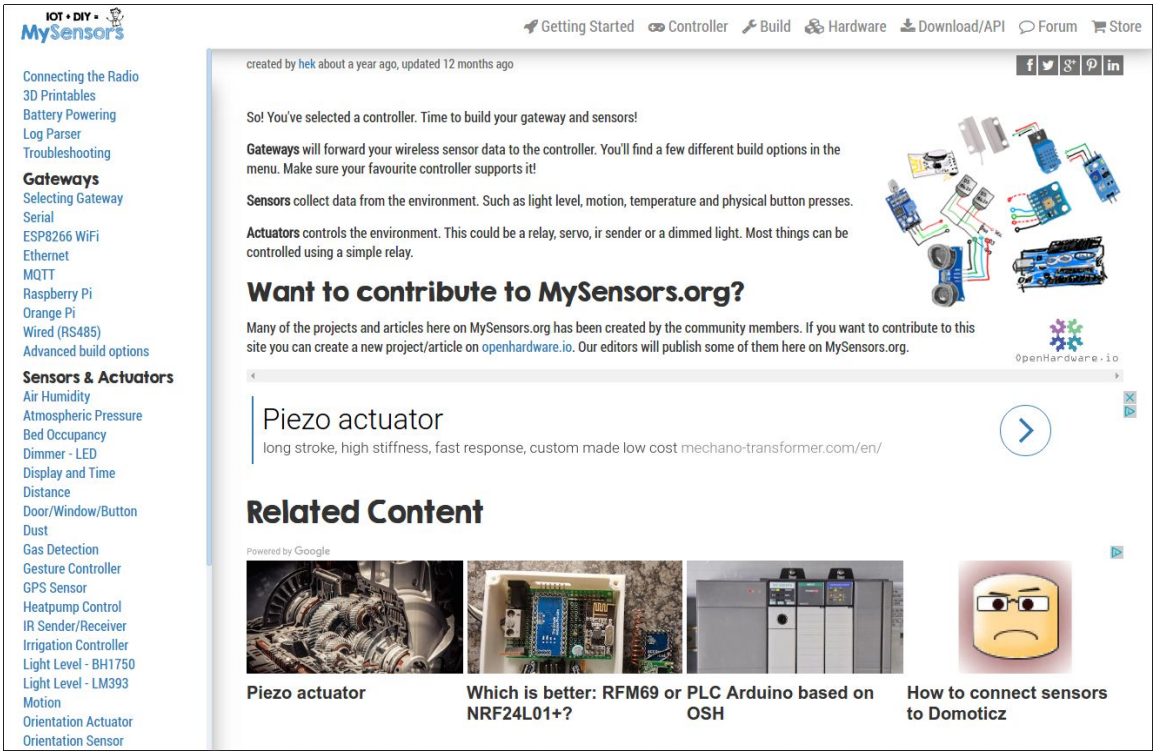


Figura 2.1: Web MySensors

Por otra parte, cuenta con varios programas completos, que sirven como base para la programación de arduino de gran cantidad de sensores y actuadores, todo esto usando la librería de MySensors, para tener una comunicación segura entre los dispositivos que ha logrado medirse con varios controladores de automatización líderes actualmente en el mercado.

Como puede verse a la derecha, en la imagen 2.1 tenemos una lista con todos los manuales de montaje, instrucciones de instalación y código de programas para los dispositivos. El código usado en el trabajo será una modificación de estos programas, que podemos encontrar en la web de MySensors, con el fin de ajustarse a nuestras necesidades concretas.

2.3.2 MyController. (4)

MyController, es un controlador para los sensores y actuadores, que fue creado en principio para dar soporte a MySensors, aunque actualmente ha cambiado su arquitectura para poder dar soporte también a otros proyectos.

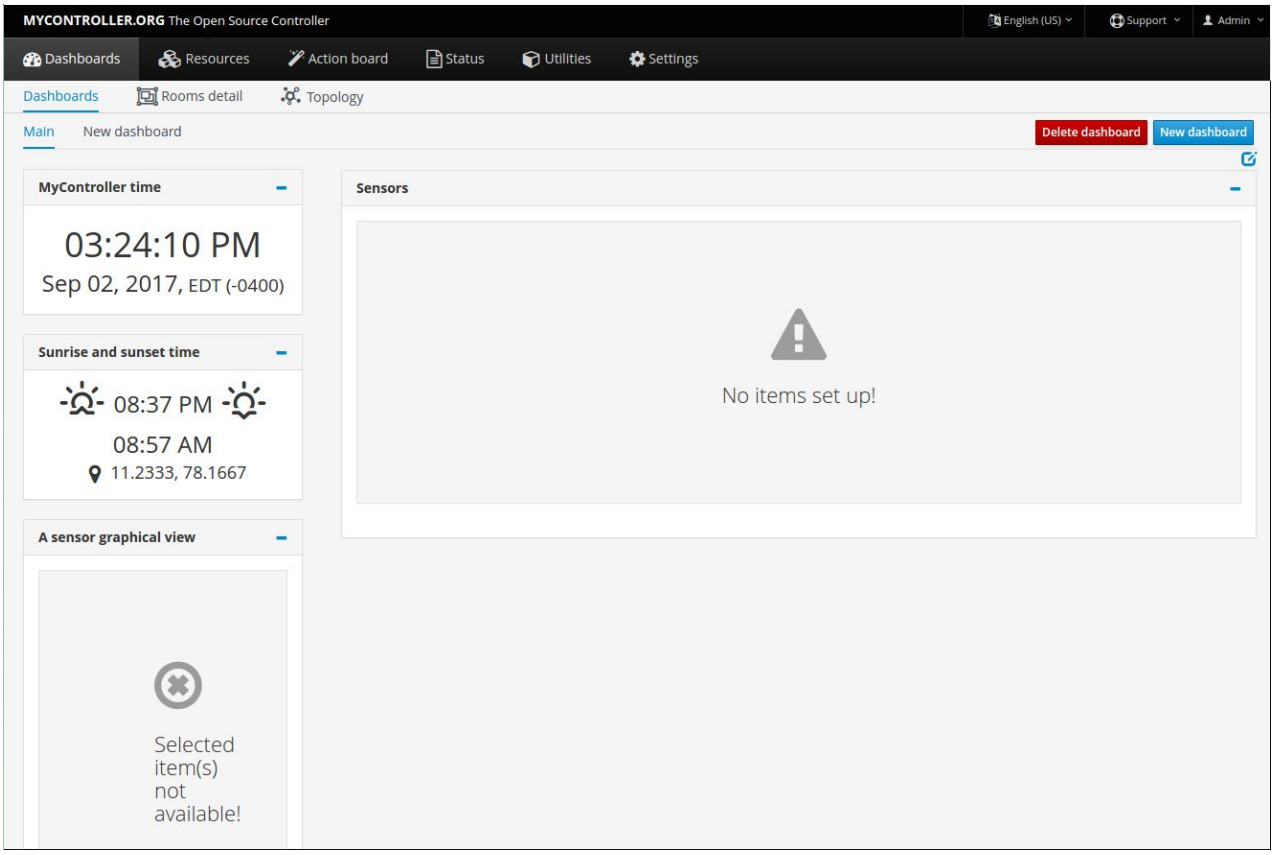


Figura 2.2: Web MyController

Este controlador nos da la posibilidad de controlar nuestra casa una vez domotizada, desde cualquier lugar en el que estemos accediendo al servidor que hayamos creado.

Asimismo, nos proporciona también la opción de crear páginas principales en las que añadir los dispositivos o información que creamos más importantes, además de poder crear scripts en python o javascript para manejar nuestros dispositivos y, por supuesto, reglas y timers, los cuales nos permitirán, en caso de llegar una hora concreta o detectarse, mediante un sensor, algún suceso, realizar una operación, como, por ejemplo, subir una persiana.

Por último, también podemos distribuir nuestros sensores y nodos en “rooms” (habitaciones), lo cual nos permitirá acceder a todos los sensores que haya en un mismo lugar de un modo más sencillo y rápido y nos dará una visión más clara del sistema que estamos montando.

Será una herramienta que tendremos instalada, en nuestro caso, en la placa raspberry pi, y a la que le corresponderá encargarse de controlar todo el sistema.

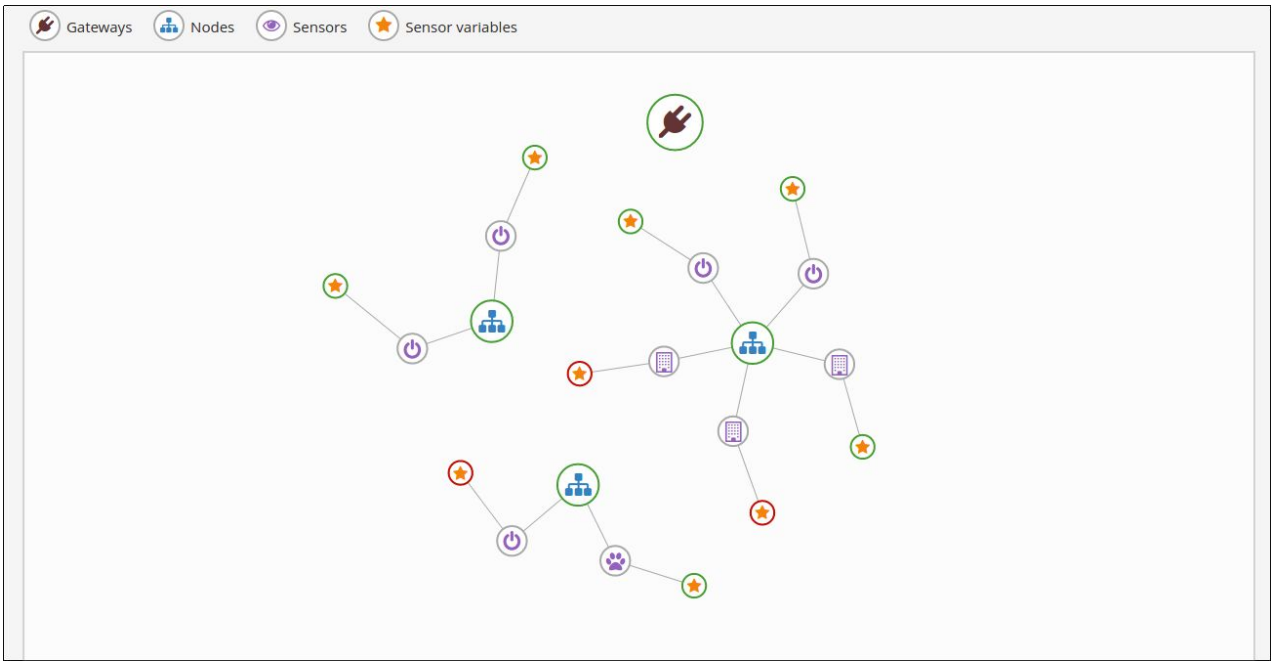


Figura 2.3: Organización MyController

Este controlador organiza nuestros componentes, como se puede ver en el ejemplo de la figura, por puertas de enlace, que sería el mismo controlador que se comunica con los nodos, los cuales serán los arduinos conectados a las radios. Estos últimos se comunican, a su vez, con los distintos sensores que tengan conectados, es decir, los relés, detectores de presencia, etc, que tenga nuestro arduino.

2.3.3 Arduino

Arduino es una plataforma de hardware libre, basada en una placa con un microcontrolador y un entorno de desarrollo, diseñada para facilitar el uso de la electrónica en proyectos multidisciplinarios.

Figura 2.3: Arduino Uno

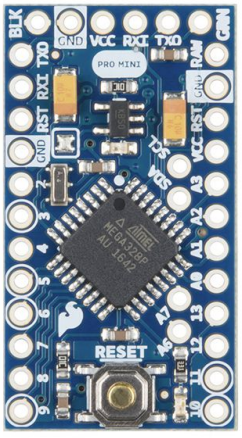


Figura 2.4:Arduino Pro Mini

En este proyecto se han usado dos tipos de placa Arduino.

La primera que observamos en la figura 2.3, es la arduino uno, la cual se ha usado, en general, en las pruebas, debido a que era más sencillo trabajar con ella a la hora de programarla, así como montar los sistemas que se probaban y analizar los resultados del programa que se subiera a la placa.

Por otro lado, tenemos la Arduino pro mini, que es la se ha montando en las PCBs que se han fabricado para el proyecto, debido a tener un tamaño muy reducido y la potencia necesaria para las aplicaciones que tendrán los nodos de nuestro sistema.

Las placas que hemos incluido en nuestros sistemas usan 3,3V y cuentan con trece salidas digitales y cuatro analógicas; además, 6 de las salidas digitales pueden ser usadas para implementar pwm, lo que nos hará falta para algunos de nuestros objetivos.

En nuestro sistema, las placas arduino serán las encargadas de recibir la informacion de los sensores, pasando luego esta informacion a nuestro controlador, y de controlar nuestros actuadores, según las órdenes que reciba del controlador.

2.3.4 Arduino IDE

Arduino IDE, es el entorno de desarrollo para Arduino en el que escribiremos el código que luego usaremos en las placas de arduino, y lo usaremos para subir estos programas a nuestras placas, especificando previamente el modelo de la placa y el puerto al que se encuentran conectadas.

También nos será de gran utilidad, al poder usar el comando Serial.println(X), para ver algunos datos concretos sobre el programa que estemos desarrollando. Esto nos permitirá saber, paso a paso, el desarrollo del programa, así como encontrar los fallos que puedan aparecer con relativa facilidad.

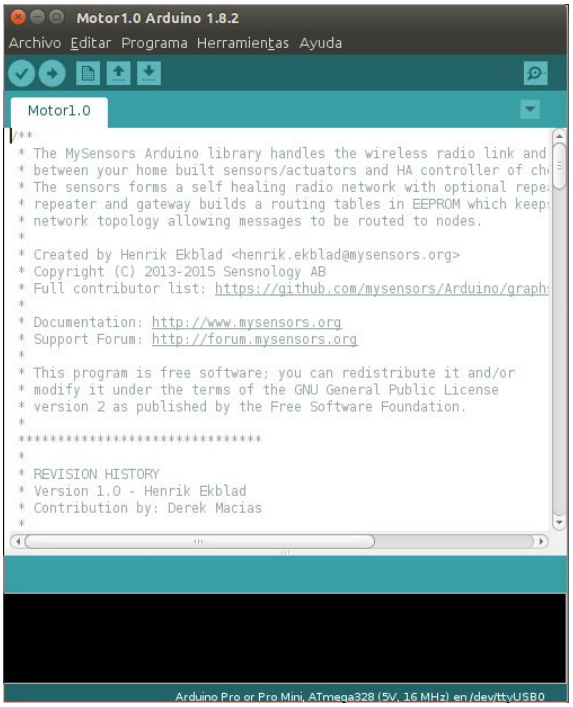


Figura 2.5:Programa de Arduino

2.3.5 Raspberry Pi

Raspberry pi es un computador de placa reducida, o computador de placa simple de bajo coste, desarrollado con el objetivo de estimular la enseñanza de ciencias de computación en las escuelas.

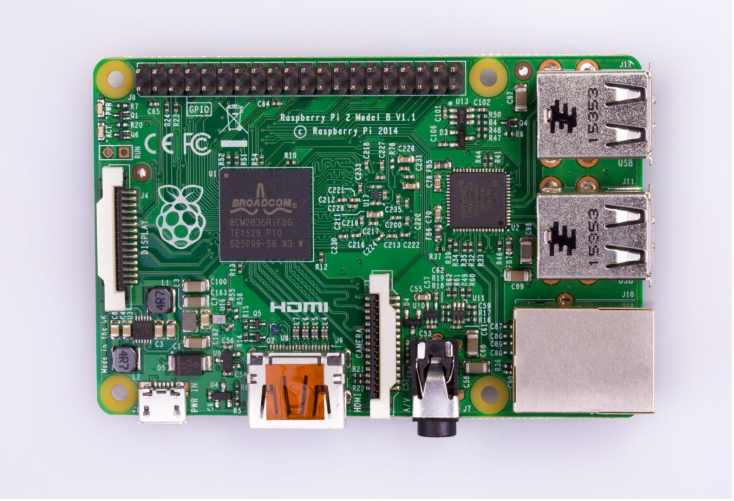


Figura 2.6: Raspberry Pi 2

Para nuestro controlador hemos usado una placa raspberry pi 2, como la que se puede apreciar en la figura 2.6. En ésta se instaló un sistema ubuntu mate como sistema operativo y sobre éste, se instalaron herramientas como mycontroller, para generar el servidor web con el que controlamos todo el sistema, conectándonos, después, desde cualquier dispositivo a la IP de la raspberry.

Para que en el arranque del sistema se lance directamente el servidor de mycontroller, hemos incluido un script que se ejecuta al encenderse, y que hace todo lo necesario para poner el servidor en marcha, de manera que, en casos de cortes de luz o problemas del parecidos, no hubiera que reiniciar manualmente la aplicación.

2.3.6 KiCad (5)

Para la creación del diseño de nuestras Pcbs se ha utilizado esta herramienta, la cual se nos enseñó a usar también durante este curso, en la asignatura de Laboratorio de hardware, lo que hacía que resultara más familiar y se tuvieran más conocimientos acerca de su uso.

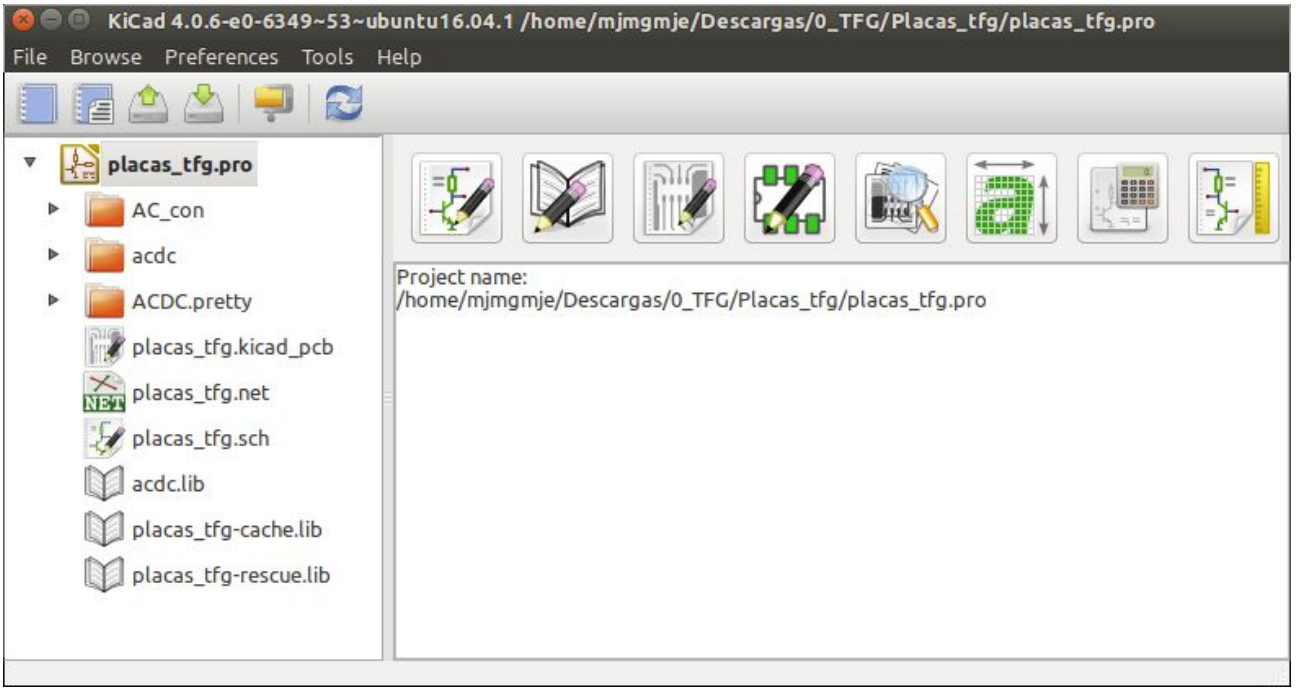


Figura 2.7: KiCad

En la figura 2.7 podemos apreciar en el entorno de KiCad, el cual nos permite diseñar, tanto la Pcb que queramos desarrollar, como componentes que vayamos a añadir en ésta y con los que no contemos previamente en librerías y/o huellas.

Para usarla, en primer lugar, se crea el esquemático de la placa, añadiendo los componentes que vayamos a usar y tengamos en las librerías del programa, y realizando las conexiones entre éstos. En segundo lugar, se pasa al desarrollo de la huella de la PCB, eligiendo las huellas para cada uno de los componentes que vayamos a usar en nuestro sistema, y conectando luego éstas, tal como se ha especificado en el esquemático del proyecto, previamente.

Mediante esta herramienta se han creado las tres Pcbs diferentes que se usan en el sistema domótico, para reunir los componentes principales.

2.3.7 Herramientas de soldadura

Para soldar los componentes a las Pcbs se han usado varias herramientas que resultan fundamentales en el proyecto, tales como un soldador, estaño, la malla desoldadora y el flux, para asegurar un buen soldado en la placa.

Figura 2.8: Herramientas de soldadura

2.3.8 Sensores

Como último punto de las herramientas, llegamos a los sensores y actuadores usados para construir los sistemas. Se han comprado, en general, modelos bastante bajos de precio, con el objetivo de mantener nuestro sistema low-cost, sin que esto reduzca lo más mínimo su plena funcionalidad.

Relé:

El relé es capaz de controlar un circuito de salida de mayor potencia que el de entrada, por lo cual, es el actuador usado para el control del encendido y apagado de los dispositivos conectados a corrientes alternas.

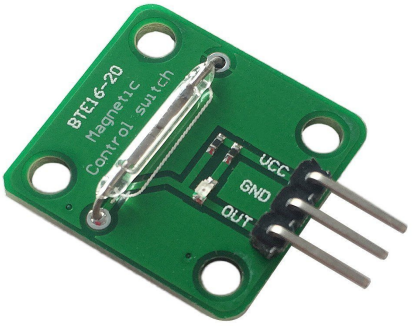
Detector de presencia:

El detector de presencia es capaz de detectar los movimientos, en un rango intermedio desde su localización. Se usa para detectar las presencia de personas en lugares concretos, no demasiado amplios.

Interruptor:

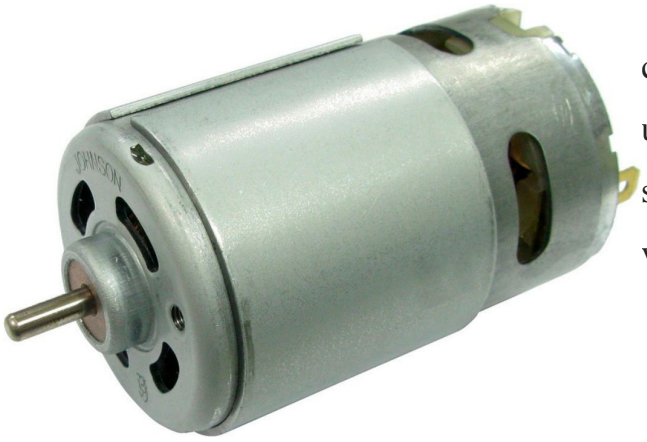
El interruptor, simplemente, abre o cierra un circuito, con lo que manejaremos el encendido o apagado de algunos de nuestros actuadores o incluso de nodos completos, en caso de que no consideremos necesario su uso durante un gran periodo.

Detector campo magnético:



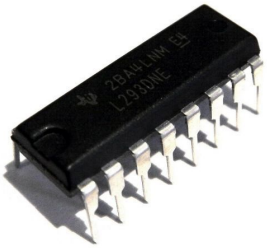
Estos sensores son capaces de detectar los campos magnéticos cercanos desactivándose, de modo que los usaremos junto con un imán, para detectar si un objeto del que queremos saber la posición, se encuentra suficientemente cerca de éstos, en cuyo caso, se desactivará la señal de este sensor.

Motor de corriente continua:



Se usará también un motor de corriente continua con bastante potencia, para poder usarse en objetos relativamente pesados. Éste se controlará mediante pwm para regular su velocidad y el sentido de su giro

Ld293dne:

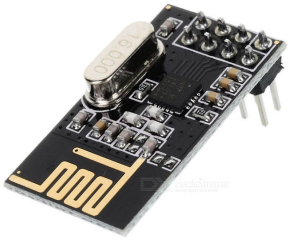


Este pequeño chip nos servirá para, en el programa en que usamos el motor, controlar, tanto la velocidad de éste, como el sentido del giro, mediante un par de señales de pwm que usaremos desde el arduino, lo cual es necesario para la implementación que vamos a usar.

Transformador:

El transformador es usado para abastecer de corriente a nuestras placas arduino, las cuales, para poder ir conectadas al cableado del hogar, tendrán que usar los 220V mediante este transformador, que pasa esos 220V a 5V para que todo funcione correctamente. Además será útil para algunos sensores concretos que necesitan de corrientes externas al arduino o corrientes de 5V en lugar de 3,3V

NRF24L01:



El NRF24L01 es el módulo de radio que utilizamos en este proyecto. Es un módulo bastante pequeño y con un alcance suficiente para el espacio que tenemos que domotizar.

Capítulo 3: Diseño del sistema

3.1 Introducción

En el tercer capítulo nos centraremos en describir el desarrollo del sistema del proyecto de fin de grado, explicando cómo se han ido cumplimentando los objetivos que nos habíamos propuesto a lo largo del proyecto. Empezaremos por ver cómo se plantean los diseños en cada una de las estancias, para seguir con el desarrollo del código usado en cada uno de los nodos y, por último, mostrar la implementación con la placa PCB del sistema completo finalizado.

3.2 Diseño del sistema en cada una de las estancias

En el espacio que tenemos para crear nuestro sistema domótico se diferencian, como ya dijimos anteriormente, tres estancias: un patio, un salón y un cuarto de baño. Incorporaremos en ellas, uno o varios nodos, según las necesidades de cada una, para alcanzar los objetivos que nos hemos propuesto en cuanto a domotizar.

3.2.1 Patio

En el patio contábamos con dos elementos: una piscina que tiene un depurador, el cual debe funcionar periódicamente cada dia, estando activado cuatro horas, y una bombilla, que debe ser controlable tan sólo desde un botón digital que añadiremos al principio de la pagina Web de MyController.

Figura 3.1: sistema patio

Para alcanzar estos objetivos hemos decidido usar tan sólo un nodo, es decir una sola Pcb, con el transformador de 200V a 5V, un arduino Pro Mini y la radio con la que se comunicará al controlador; todo esto incluido en la PCB.

Además, para controlar el encendido y apagado de estos dispositivos, se usarán un par de relés, cada uno controlando uno de ellos, la depuradora y la bombilla. El arduino deberá encargarse de usar una salida en modo output para cada uno de los relés, aunque por problemas de reseteos y fallos advertidos en la pagina de mySensors, en lugar de alimentar el relé con el VCC y GND de la placa de arduino, esta tarea se hará con los 5V que salen del transformador.

Los 220V para alimentar el transformador, se cogerán del cable que iba hacia la bombilla y se llevarán a ésta, pasando por el relé, igual que a la depuradora y, por último, directamente hacia el transformador. Así, el interruptor que antes manejaba la bombilla, se

convertirá en un controlador del sistema completo; si se apaga, se apagará tanto el nodo, como los actuadores conectados a éste.

3.2.2 Salón

En nuestra segunda estancia tenemos que controlar de nuevo varios dispositivos: un par de bombillas, controladas en principio por dos conmutadores, y una persiana que se mueve de forma manual.

Para cada uno de estos objetivos se usará un nodo distinto, debido a la distancia entre ambos. El primer nodo, el de las bombillas que iluminan el salón, debe hacer que éstas se enciendan o se apaguen, controladas por los conmutadores que las manejaban antes y por un botón que añadiremos a la web de MyController.

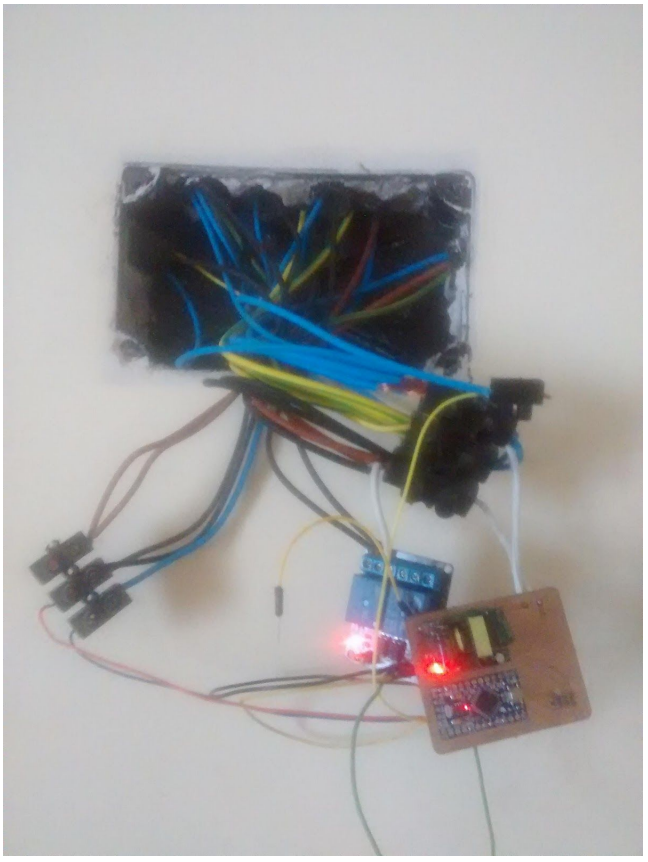


Figura 3.2: Sistema iluminación

Para esto hemos usado un relé que maneja ambas bombillas al mismo tiempo, cambiando las conexiones para hacer que dejen de controlarse por los conmutadores y lo hagan directamente por el relé, únicamente.

Los conmutadores se han conectado a la placa arduino también como simples interruptores que, al cambiar de estado, hacen que el estado del relé también cambie, manteniendo así su antigua función de controlar la bombilla, para poder seguir usándolos como de costumbre.

El sistema estará situado en el interior de la caja de cables, donde se encuentran tanto los de las bombillas como los de los conmutadores. Una vez analizados estos cables, para saber cuál corresponde a cada dispositivo, se conectará de nuevo el transformador a 220V, añadiendo un par de cables más al sistema y, en esta ocasión, al usarse un único relé, se podrá conectar directamente a Vcc y GND de la placa arduino. En cuanto a los conmutadores, como hacen la función de interruptores, no se usarán los 3 cables que tienen, sino tan sólo un par de ellos, para percibir los cambios en los botones.

El segundo sistema, el de la persiana, es algo más complejo. Constará de un motor de corriente continua (cc), que debe tener la fuerza suficiente para hacer girar a la persiana, un par de detectores de campo magnético, y un botón interruptor para controlar el movimiento.

El motor irá conectado a una polea, cuyo cometido consistirá en mover la persiana mediante una cuerda a su alrededor, que llega hasta el eje de la misma. Se podrá controlar el sentido de giro del motor para hacer que la cortina baje o suba, deteniendo el movimiento al toparse con uno de los detectores de campo magnético.

Para frenar con los detectores, hará falta también que la persiana lleve incorporado un pequeño imán, que active estos sensores al pasar cerca de ellos. Éstos se colocarán en las posiciones de la persiana abierta y cerrada, para hacer que el giro del motor pare, una vez alcanzada cualquiera de estas posiciones.

Por último, el nodo también usará el interruptor antes mencionado, para controlar el giro del motor, haciendo que, en uno de sus estados, gire en una dirección, y en el estado contrario, active el giro en dirección contraria, con lo que, tan sólo pulsando el interruptor, haremos que la persiana se mueva en dirección opuesta a la que se encontrará inicialmente.

3.2.3 Cuarto de baño

Por último, en el cuarto de baño queremos instalar un sistema que automáticamente encienda y apague la luz, en función de si hay o no alguien dentro en ese momento, para lo cual, se ha usado de nuevo un relé para la bombilla y un sensor de movimiento dentro del cuarto.

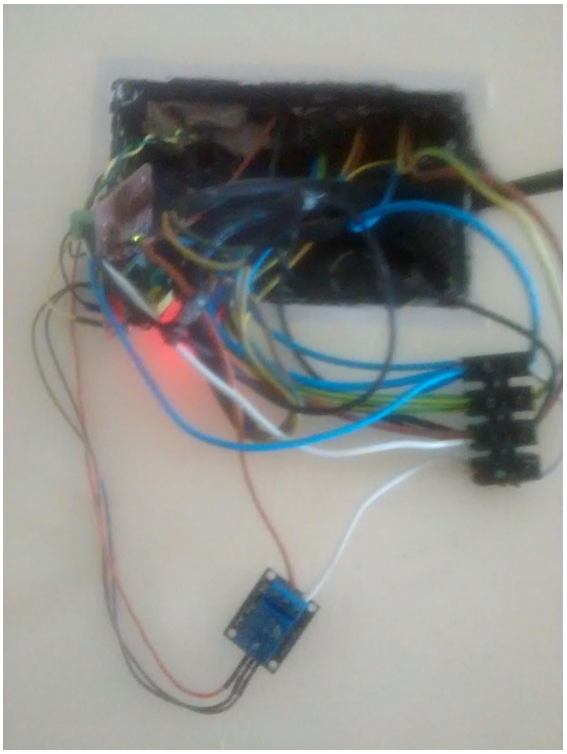


Figura 3.3: Sistema baño

El relé controla la bombilla que hay dentro del cuarto, mientras que el interruptor, que era el encargado de encender y apagar esta bombilla anteriormente, se encarga ahora de activar el encendido y apagado de todo el sistema, como vimos previamente en el caso

del patio. Esto hace que, si sabemos que durante largos periodos de tiempo no se usará este cuarto, podamos desactivar el sistema, con tan sólo abrir el circuito del interruptor.

El sensor de movimiento está conectado a la placa arduino a través de un agujero en la pared, y se ha colocado dentro del cuarto de baño, concretamente, en el techo, de manera que sólo hace falta abrir la puerta para que se encienda la luz, al detectar este movimiento el sensor. Tras un período, reajustable en el sensor, de varios de segundos, la luz vuelve a apagarse si no se ha detectado ningún nuevo movimiento.

De nuevo se usa la caja de cables de la bombilla y el interruptor para situar nuestro sistema, tanto la Pcb como el relé, y aparte, el sensor de movimiento, como se acaba de explicar. De nuevo el relé, al ser uno solo, puede estar alimentado directamente por el propio arduino.

3.3 Diseño del código de los nodos y operaciones en el controlador

En este apartado se verá el código que se usa en cada uno de los nodos, así como las operaciones, reglas y scripts que usaremos desde el controlador, para hacer efectivo el control de los sensores sobre nuestros relés.

Veremos por partes, tanto el código del nodo en cuestión, como las operaciones que se hayan creado en el controlador para crear a éste, empezando por el nodo del patio y concluyendo con el del cuarto de baño, al igual que en el apartado anterior.

Código nodos

Nodo patio:

```
#define MY_DEBUG
```

```
// Enable and select radio type attached
#define MY_RADIO_NRF24


// Enable repeater functionality for this node
#define MY_REPEATER_FEATURE

#include <MySensors.h>

#define RELAY_1  3  // Arduino Digital I/O pin number for first
relay (second on pin+1 etc)
#define NUMBER_OF_RELAYS 2 // Total number of attached relays
#define RELAY_ON 1  // GPIO value to write to turn on attached
relay
#define RELAY_OFF 0 // GPIO value to write to turn off attached
relay


void before()
{
    for (int sensor=1, pin=RELAY_1; sensor<=NUMBER_OF_RELAYS;
sensor++, pin++) {
        // Then set relay pins in output mode
        pinMode(pin, OUTPUT);
        // Set relay to last known state (using eeprom storage)
        digitalWrite(pin, loadState(sensor)?RELAY_ON:RELAY_OFF);
    }
}

void setup()
{

}

void presentation()
{
    // Send the sketch version information to the gateway and
Controller
    sendSketchInfo("Relay", "1.0");

    for (int sensor=1, pin=RELAY_1; sensor<=NUMBER_OF_RELAYS;
sensor++, pin++) {
        // Register all sensors to gw (they will be created as
child devices)
        present(sensor, S_BINARY);
    }
}


void loop()
```

```
{

}

void receive(const MyMessage &message)
{
    // We only expect one type of message from controller. But we
better check anyway.
    if (message.type==V_STATUS) {
        // Change relay state
        digitalWrite(message.sensor-1+RELAY_1,
message.getBool()?RELAY_ON:RELAY_OFF);
        // Store state in eeprom
        saveState(message.sensor, message.getBool());
        // Write some debug info
        Serial.print("Incoming change for sensor:");
        Serial.print(message.sensor);
        Serial.print(", New status: ");
        Serial.println(message.getBool());
    }
 }
```

El código usado en el nodo del patio es el que puede verse arriba. Apenas tiene modificaciones del código que podemos encontrar en MySensors, ya que nos sirve tal cual, cambiando simplemente el número de relés a dos, dado que usaremos uno para la depuradora y otro distinto para la bombilla del patio.

Podemos ver cómo primero se presentan los sensores que vayamos a utilizar en la función `presentation` y luego se crea la función que se ejecutará en caso de que cambiemos, desde el controlador, el estado que debe tener el relé, la función `receive`, la que recibe el mensaje enviado por el controlador al nodo, y luego en función del estado indicado en el mensaje, da al pin del relé el valor 0 o 1.

En la web del controlador se crearon un par de timers, que ejecutan una operación, para el funcionamiento de la depuradora: el primer timer se activa diariamente a la 1 del mediodía y ejecuta una operación que activa el relé que controla la depuradora, el segundo se ejecuta a las 5 de la tarde y efectúa una operación que desactiva este mismo relé.

Además, se han añadido en el dashboard principal, 2 botones dentro de una sección llamada patio, para controlar tanto el relé de la bombilla como el de la depuradora y poder controlarlos directamente desde la web, con facilidad.

Figura 3.4: Botones del Patio

Nodo salón, luces:

```
// Enable debug prints to serial monitor
#define MY_DEBUG

// Enable and select radio type attached
#define MY_RADIO_NRF24
//#define MY_RADIO_RFM69

#define MY_REPEATER_FEATURE

#include <SPI.h>
#include <MySensors.h>
#include <Bounce2.h>

#define CHILD_IDB1 3
```

```
#define BUTTON_PIN 3
#define CHILD_IDB2 4
#define BUTTON_PIN2 4

#define RELAY_1  5  // Arduino Digital I/O pin number for first
relay (second on pin+1 etc)
#define NUMBER_OF_RELAYS 1 // Total number of attached relays
#define RELAY_ON 1  // GPIO value to write to turn on attached
relay
#define RELAY_OFF 0 // GPIO value to write to turn off attached
relay

Bounce debouncer = Bounce();
Bounce debouncer2=Bounce();
int oldValue=-1;
int oldValue2=-1;
int relvalue=-1;

MyMessage msgb1(CHILD_IDB1,V_TRIPPED);
MyMessage msgb2(CHILD_IDB2,V_TRIPPED);

void before()
{
    for (int sensor=1, pin=RELAY_1; sensor<=NUMBER_OF_RELAYS;
sensor++, pin++) {
        // Then set relay pins in output mode
        pinMode(pin, OUTPUT);
        // Set relay to last known state (using eeprom storage)
        digitalWrite(pin, loadState(sensor)?RELAY_ON:RELAY_OFF);
        relvalue=loadState(sensor)?RELAY_ON:RELAY_OFF;
    }
}



unsigned long timeOfLastChange = 0;
bool attachedServo = false;

int pin2=6;   //Entrada 2 del L293D
int pin7=5;  //Entrada 7 del L293D


void setup()
{
  // Request last servo state at startup
  pinMode(pin2,OUTPUT);
  pinMode(pin7, OUTPUT);
  Serial.begin(9600);
  pinMode(BUTTON_PIN,INPUT);
  pinMode(BUTTON_PIN2,INPUT);
  digitalWrite(BUTTON_PIN, HIGH);
```

```
  digitalWrite(BUTTON_PIN2, HIGH);
  debouncer.attach(BUTTON_PIN);
  debouncer2.attach(BUTTON_PIN2);
  debouncer.interval(5);
  debouncer2.interval(5);
}

void presentation()  {
  // Send the sketch version information to the gateway and
Controller
  sendSketchInfo("bombillas_int", "1.0");

  // Register all sensors to gw (they will be created as child
devices)
  present(CHILD_IDB1, S_DOOR);
  present(CHILD_IDB2, S_DOOR);
    sendSketchInfo("Relay", "1.0");

    for (int sensor=1, pin=RELAY_1; sensor<=NUMBER_OF_RELAYS;
sensor++, pin++) {
        // Register all sensors to gw (they will be created as
child devices)
        present(sensor, S_BINARY);
    }

}

void loop()
{
  Serial.println("looped");
  //if (attachedServo && millis() - timeOfLastChange >
DETACH_DELAY) {
  //   myservo.detach();
    // attachedServo = false;
  //}
  debouncer.update();
  debouncer2.update();
  int value =debouncer.read();
  int value2=debouncer2.read();

  Serial.println(value);
  Serial.println(value2);

  if(value!=oldValue){
    send(msgb1.set(value==HIGH ? 1:0));
    oldValue=value;
  }

    if(value2!=oldValue2){
    send(msgb2.set(value2==HIGH ? 1:0));
    oldValue2=value2;
```

```
  }


void receive(const MyMessage &message)
{
    // We only expect one type of message from controller. But we
better check anyway.
    if (message.type==V_STATUS) {
        // Change relay state
        relvalue=relvalue==RELAY_ON?RELAY_OFF:RELAY_ON;
        digitalWrite(message.sensor-1+RELAY_1, relvalue);

        // Store state in eeprom
        saveState(message.sensor, message.getBool());
        // Write some debug info
        Serial.print("Incoming change for sensor:");
        Serial.print(message.sensor);
        Serial.print(", New status: ");
        Serial.println(message.getBool());
    }
}
```

Este es el código del nodo del sensor que usamos en arduino. Este código une varios ejemplos de interruptores junto con un relé; la parte del relé sigue siendo muy parecida a la anterior, con la diferencia de que ahora se trata de un único relé, pero podemos ver cómo aquí se añaden a la presentación un par de dispositivos más: los interruptores que manejan las luces.También podemos ver más arriba la declaración de unos mensajes `MyMessage msgb1(CHILD_IDB1,V_TRIPPED);` estos mensajes que se han creado, serán los que usen los botones para comunicarse en la función `loop` desde el nodo hasta el controlador, informándole de si hay algún cambio en los interruptores, para que puedan producirse las operaciones configuradas en el sistema.

El mensaje que se envía es del tipo `V_TRIPPED`, que informará de si nuestro interruptor está abierto o cerrado. Además,tenemos que usar también la librería bounce2.h, para crear objetos tipo debouncer, que eviten la lectura de los posibles rebotes al cambiar de estado los interruptores.

En MyController se creó para este nodo un script que, sencillamente, cambiaba el estado del relé, haciendo que se encendiera si estaba apagado y viceversa. Pues bien, con

esto y un par de reglas, para que cada vez que se vea un cambio en el estado de los interruptores, se ejecute este script, conseguimos implementar el sistema de iluminación.

```
var myImports = new JavaImporter(java.io, java.lang, java.util,
java.text);

with(myImports) {
     var hallwayLight = mcApi.sensor().getVariable(58);
     //var hallwayLight= mcApi.SensorVariableJson(hallwaylight);

     if(hallwayLight.value == '0'){
          hallwayLight.setValue('1');
          mcApi.sensor().sendpayload(hallwayLight);
     }

     else if(hallwayLight.value == '1'){
          hallwayLight.value='0';
          mcApi.sensor().sendpayload(hallwayLight);
     }

}
```

En el script escrito usando javascript, se crea un objeto SensorVariableJson, hallwayLight, el cual nos permitirá actualizar el estado de una variable de un sensor, una vez lo hayamos cambiado en el propio script con el método sendpayload(SensorVariableJson), y hacemos que este objeto hallwayLight se corresponda con el relé que queremos cambiar, identificándolo con su ID (58).

Nodo salón, persiana

```
     // Enable debug prints to serial monitor
#define MY_DEBUG

// Enable and select radio type attached
#define MY_RADIO_NRF24
//#define MY_RADIO_RFM69

#include <SPI.h>
#include <MySensors.h>
#include <Bounce2.h>

#define CHILD_IDB1 3
#define BUTTON_PIN 3
```

```
#define CHILD_IDB3 7
#define BUTTON_PIN3 7
#define CHILD_IDB2 4
#define BUTTON_PIN2 4

#define MOTOR_MIN 0
#define MOTOR_MAX 1023
#define DETACH_DELAY 900 // Tune this to let your movement finish
before detaching the servo
#define CHILD_ID 10   // Id of the sensor child

Bounce debouncer = Bounce();
Bounce debouncer2=Bounce();
Bounce debouncer3=Bounce();

int oldValue=-1;
int oldValue2=-1;
int oldValue3=-1;

MyMessage msg(CHILD_ID, V_DIMMER);
MyMessage msgb1(CHILD_IDB1,V_TRIPPED);
MyMessage msgb2(CHILD_IDB2,V_TRIPPED);
MyMessage msgb3(CHILD_IDB3,V_TRIPPED);


unsigned long timeOfLastChange = 0;
bool attachedServo = false;

int pin2=6;   //Entrada 2 del L293D
int pin7=5;  //Entrada 7 del L293D


void setup()
{
  // Request last servo state at startup
  request(CHILD_ID, V_DIMMER);
  pinMode(pin2,OUTPUT);
  pinMode(pin7, OUTPUT);
  Serial.begin(9600);
  pinMode(BUTTON_PIN,INPUT);
  pinMode(BUTTON_PIN2,INPUT);
  pinMode(BUTTON_PIN3,INPUT);
  digitalWrite(BUTTON_PIN3, HIGH);
  digitalWrite(BUTTON_PIN, HIGH);
  digitalWrite(BUTTON_PIN2, HIGH);
  debouncer.attach(BUTTON_PIN);
  debouncer2.attach(BUTTON_PIN2);
  debouncer3.attach(BUTTON_PIN3);
  debouncer.interval(5);
  debouncer2.interval(5);
  debouncer3.interval(5);
```

```
}

void presentation()  {
  // Send the sketch version information to the gateway and
Controller
  sendSketchInfo("motor", "1.0");

  // Register all sensors to gw (they will be created as child
devices)
  present(CHILD_ID, S_COVER);
  present(CHILD_IDB1, S_DOOR);
  present(CHILD_IDB2, S_DOOR);
  present(CHILD_IDB3, S_DOOR);

}

void loop()
{
  Serial.println("looped");
  debouncer.update();
  debouncer2.update();
  debouncer3.update();
  int value =debouncer.read();
  int value2=debouncer2.read();
  int value3=debouncer3.read();

  Serial.println(value);
  Serial.println(value2);

  if(value!=oldValue){
    send(msgb1.set(value==HIGH ? 1:0));
    oldValue=value;
  }

    if(value2!=oldValue2){
    send(msgb2.set(value2==HIGH ? 1:0));
    oldValue2=value2;
  }
  if(value3!=oldValue3){
    send(msgb3.set(value3==HIGH ? 1:0));
    oldValue3=value3;
  }


}
void updateVal(int val){
  Serial.println("updating");
  int pwm1 = map(val, 0, 1023, 0, 255);
  int pwm2 = map(val, 0, 1023, 255, 0);
  Serial.print("pwm1");
  Serial.println(pwm1);
```

```
  Serial.print("pwm2");
  Serial.println(pwm2);
  analogWrite(pin2,pwm1);
  analogWrite(pin7,pwm2);
  }

void receive(const MyMessage &message) {
  Serial.println("mesage recived");
  if (message.type==V_DIMMER) { // This could be M_ACK_VARIABLE or
M_SET_VARIABLE
      int val = message.getInt();
      int potenc= MOTOR_MAX + (MOTOR_MIN-MOTOR_MAX)/100 * val;
      updateVal(potenc);
      Serial.print("Motor changed. new state: ");
      Serial.println(val);
    } else if (message.type==V_UP) {
      Serial.println("motor UP command");
      int potenc=MOTOR_MIN;
      updateVal(potenc);
      send(msg.set(100));
    } else if (message.type==V_DOWN) {
      Serial.println("motor DOWN command");
      int potenc=MOTOR_MAX;
      updateVal(potenc);
      send(msg.set(0));
    } else if (message.type==V_STOP) {
      Serial.println("motor STOP command");
        int potenc=512;
      updateVal(potenc);
      send(msg.set(50));

    }
}
```

El código superior hace que el nodo controle un motor de corriente continua mediante dos señales de pwm, situadas en las salidas cinco y seis. Además de esto, tenemos ahora tres interruptores, que se corresponden con los dos detectores de campo magnético, y el interruptor encargado de activar el movimiento de la persiana.

Esta vez tenemos un nuevo tipo de mensaje V_DIMMER, que enviará al controlador la informacion de un valor entre cien y cero para señalar a los pwm las señales que deben enviar, traduciéndose cero como la máxima velocidad en un sentido de giro y cien como la máxima velocidad en el sentido de giro contrario; una velocidad de cincuenta haría, por tanto, que el motor se parara.

En la presentación vemos ahora a un nuevo tipo de dispositivo, o sea, el motor, que usa el tipo S_COVER en lugar de S_BINARY. Aparte de eso, el loop sigue igual que el que hemos visto anteriormente, pero con un interruptor más, y luego encontramos una función nueva `updateVal`, la cual nos permite mapear los valores que nos dan las señales pwm, a

nuevos valores que se ajustan a lo que queremos, haciendo que en lugar de limitarse de cero a doscientos cincuenta y cinco, vayan hasta mil veintitrés.

Finalmente, entramos en la función `receive`, en la cual se espera un mensaje, y, según el tipo de mensaje que se reciba, se enviará un porcentaje a la función updateVal para que ésta actualice el valor de las salidas de pwm. Una vez hecho esto, se manda un mensaje V_DIMMER con el valor del porcentaje, para que se actualice en la pagina, de forma visible, en qué estado se encuentra el motor.

Podemos ver en el controlador una serie de reglas y operaciones simples, creadas para controlar este nodo. Primero se ha creado un botón virtual con la misma función que el interruptor físico, activar el desplazamiento de la persiana en ambos sentidos, para lo cual, al activarse el botón, da un porcentaje del treinta al motor, y al desactivarse manda un porcentaje del setenta. A continuación, cuando los sensores de campo magnético detectan un imán cercano, hacen que el motor pare de girar automáticamente, quedando así en la posición deseada.

Nodo Cuarto de baño:

```
// Enable debug prints to serial monitor
#define MY_DEBUG

// Enable and select radio type attached
#define MY_RADIO_NRF24

// Enable repeater functionality for this node
#define MY_REPEATER_FEATURE

#include <MySensors.h>

#define RELAY_1  6  // Arduino Digital I/O pin number for first
relay (second on pin+1 etc)
#define NUMBER_OF_RELAYS 1 // Total number of attached relays
#define RELAY_ON 1  // GPIO value to write to turn on attached
relay
#define RELAY_OFF 0 // GPIO value to write to turn off attached
relay

unsigned long SLEEP_TIME = 1200; // Sleep time between reports (in
milliseconds)
#define DIGITAL_INPUT_SENSOR 3   // The digital input you attached
your motion sensor.  (Only 2 and 3 generates interrupt!)
#define CHILD_ID 5   // Id of the sensor child

MyMessage msg(CHILD_ID, V_TRIPPED);

void before()
```

```
{
    for (int sensor=1, pin=RELAY_1; sensor<=NUMBER_OF_RELAYS;
sensor++, pin++) {
        // Then set relay pins in output mode
        pinMode(pin, OUTPUT);
        // Set relay to last known state (using eeprom storage)
        digitalWrite(pin, loadState(sensor)?RELAY_ON:RELAY_OFF);
    }
}

void setup()
{
    Serial.begin(19200);
    pinMode(DIGITAL_INPUT_SENSOR, INPUT);      // sets the motion
sensor digital pin as input
}

void presentation()
{
    Serial.println("presentation");
    // Send the sketch version information to the gateway and
Controller
    sendSketchInfo("Relay", "1.0");

    for (int sensor=1, pin=RELAY_1; sensor<=NUMBER_OF_RELAYS;
sensor++, pin++) {
        // Register all sensors to gw (they will be created as
child devices)
        present(sensor, S_BINARY);
    }
        // Send the sketch version information to the gateway and
Controller
    sendSketchInfo("Motion Sensor", "1.0");

    // Register all sensors to gw (they will be created as child
devices)
    present(CHILD_ID, S_MOTION);
}

void receive(const MyMessage &message)
{
    // We only expect one type of message from controller. But we
better check anyway.
    if (message.type==V_STATUS) {
        // Change relay state
        digitalWrite(message.sensor-1+RELAY_1,
message.getBool()?RELAY_ON:RELAY_OFF);
        // Store state in eeprom
        saveState(message.sensor, message.getBool());
        // Write some debug info
        Serial.print("Incoming change for sensor:");
```

```
        Serial.print(message.sensor);
        Serial.print(", New status: ");
        Serial.println(message.getBool());
    }
}



void loop()
{
    Serial.println("entering loop");

    // Read digital motion value
    bool tripped = digitalRead(DIGITAL_INPUT_SENSOR) == HIGH;

    Serial.println(tripped);
    send(msg.set(tripped?"1":"0"));  // Send tripped value to gw

    // Sleep until interrupt comes in on motion sensor. Send
update every two minute.
    sleep(digitalPinToInterrupt(DIGITAL_INPUT_SENSOR), CHANGE,
SLEEP_TIME);
}
```

Aquí podemos ver que en la parte de programación, el sensor de movimiento es muy parecido a cualquier interruptor; al igual que éstos, es un objeto de tipo binario, que envía un mensaje tripped, y realiza esta acción dentro de la función `loop`, en la que comprueba su estado y lo guarda en una variable booleana, para, según el valor de ésta, enviar un uno o un cero al controlador.

El relé mantiene su código, tal como lo hemos visto en otras ocasiones, con la función `receive` sin ningún cambio.

Al controlador le encargamos la tarea de activar el relé, sencillamente, una vez que se haya activado el sensor de movimiento, y cuando se desactiva, desactivamos también el relé.

En el propio sensor podemos configurar el tiempo que se mantiene el sensor activado cuando detecta movimiento, y la sensibilidad para detectar estos movimientos, en nuestro caso, la hemos configurado para que tarde unos siete segundos en apagarse, con lo que, una vez salimos del cuarto y cerramos la puerta, la luz se apaga automáticamente a los siete segundos.

3.4 Diseño de las PCBs

En este apartado veremos cómo se ha hecho el diseño de las tres Pcbs que se usarán en el proyecto:

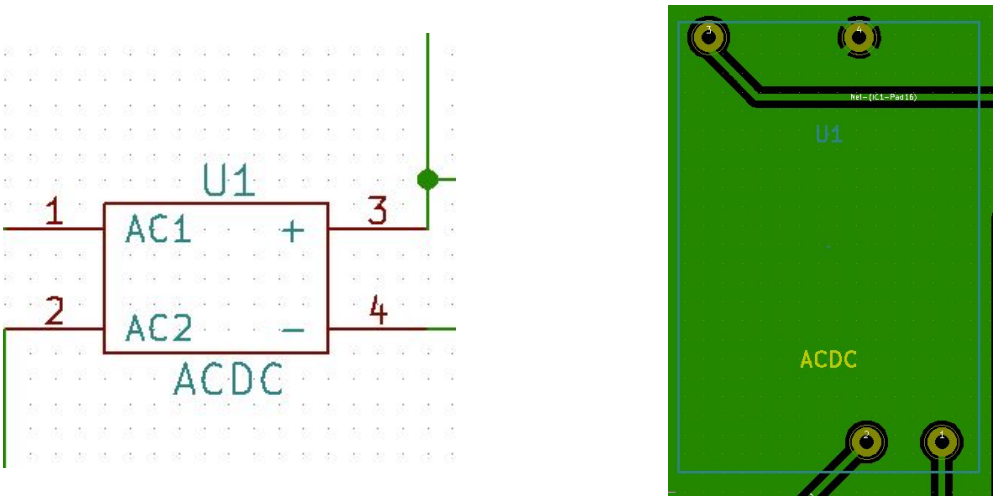
Una primera compuesta por el transformador, arduino y el módulo de radio. De estas se usarán varias en el proyecto, concretamente tres, en los nodos del baño, las luces y el patio.

Una segunda placa, aunando los componentes anteriores con un módulo L293DN, que necesitamos para el control del motor. Dicho módulo resultaba bastante complicado de usar contando sólo con cables, por ello ha sido adherido a la PCB.

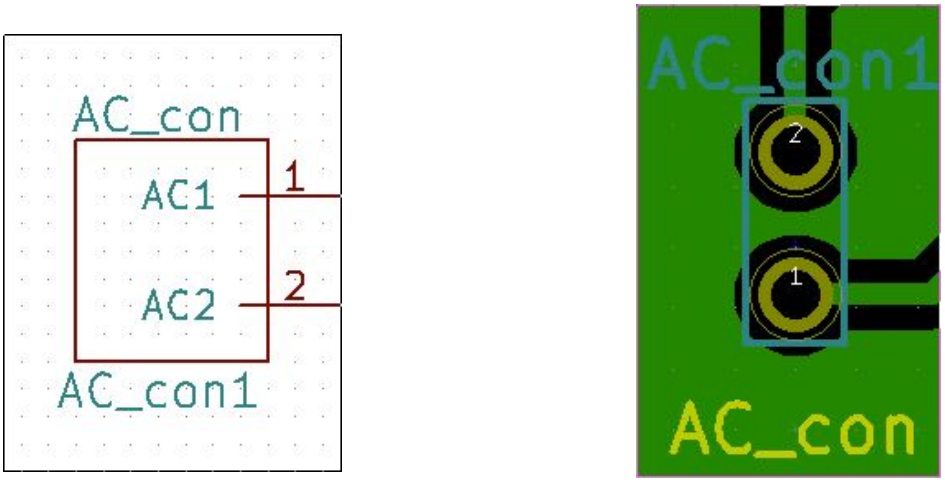
Y por último, una PCB sencilla en la que unir los componentes de la puerta de enlace, o sea, la raspberry, que hace la función de controlador, y la radio conectada a ésta.

3.4.1 Primera placa

Para la creación de la primera placa se han creado también, de un par de componentes, dos librerías y huellas que hacían falta. El componente del transformador, cuenta con cuatro pines, dos para las conexiones a corriente alterna y los otros dos con la corriente de de salida de 5V.



Figuras 3.5 y 3.6: Esquema y huella del transformador



Figuras 3.7 y 3.8: Esquema y huella del conector a corriente alterna

Para crear las librerías se utiliza la herramienta library Editor de Kicad, ésta abrirá una nueva ventana en blanco en la que podremos dibujar nuestro componente, darle nombre y añadir los pines de este nombrando también a éstos. Para el esquemático no hace falta más información que esta: los pines que tendrá el componente.

Una vez tenemos la librería, se hace la huella que requiere de más información. La huella tiene que ser muy exacta, con lo que hay tres factores que tenemos que conocer perfectamente:

a) Las dimensiones del componente.

b) La distancia entre sus distintos pines y los bordes.

c) El grosor de estos pines. Por ejemplo, esto último ha ocasionado un pequeño problema con la huella del conector a corriente alterna. En esta huella, el tamaño del agujero para los pines se quedó, al final, algo estrecho y hay que usar algo de fuerza para conseguir montar correctamente la pieza.

También deberemos dejar los mismos números y nombres que hayamos dado a los pines de la librería en la huella, para que los pines actúen como deben cuando hagamos las conexiones en el editor de PCB.

Con las todas las librerías y huellas disponibles en nuestra plataforma, comenzamos a realizar el diseño de nuestra PCB, empezando por el diseño esquemático. Para ello usamos el editor de esquemáticos de KiCad y añadimos los componentes de las librerías que vayamos a utilizar en nuestra placa; una vez añadidos, se hacen las conexiones entre estos componentes, uniéndolos con los cables del esquemático.

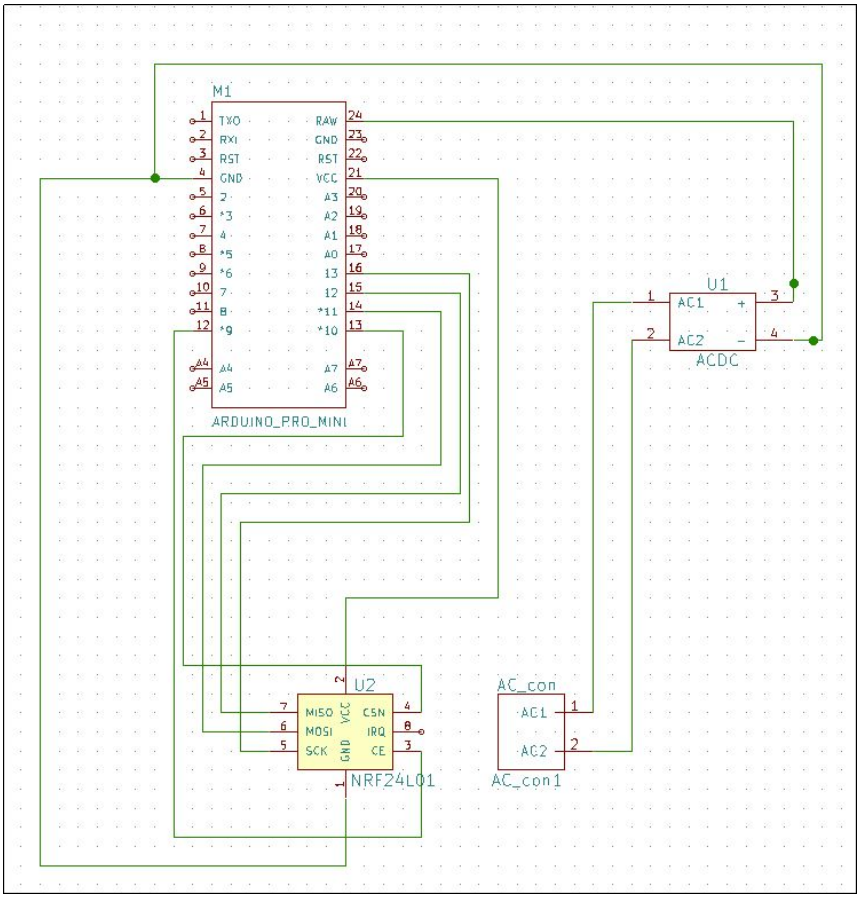


Figura 3.9: Esquemático primera PCB

En este esquemático podemos ver la conexión entre el componente transformador que hemos creado y la placa de arduino.Éste, conecta su parte positiva al valor RAW de la placa de arduino y la negativa, sencillamente, a GND.

Para la conexión con la radio, sólo necesitamos siete cables, ya que el pin IRQ de la misma, es tan sólo opcional y no nos hace falta en ningún momento para nuestros nodos. La conexión sería la siguiente:

Pro Mini / Nano	Mega	NRF24L01+
GND	GND	GND
5VReg -> 3.3V	3.3V	VCC
9	49	CE
10	53	CSN/CS
13	52	SCK
11	51	MOSI
12	50	MISO
2	2	IRQ

Asimismo, podemos ver en la figura 3.9 la conexión entre el conector AC y el transformador; sencillamente un par de cables conectados a cada uno de los pines del conector AC, que van directamente a los pines de corriente alterna de nuestro transformador.

Cuando tenemos el diseño esquemático completado, pasamos al diseño de la PCB. Para esto, damos una huella a cada uno de los componentes que hemos usado en el esquemático asociándolos, y usamos para ello la función CvPCB. Después generamos un netlist, que contendrá los componentes y las conexiones entre éstos, y lo usamos en la ventana de edición de PCB.

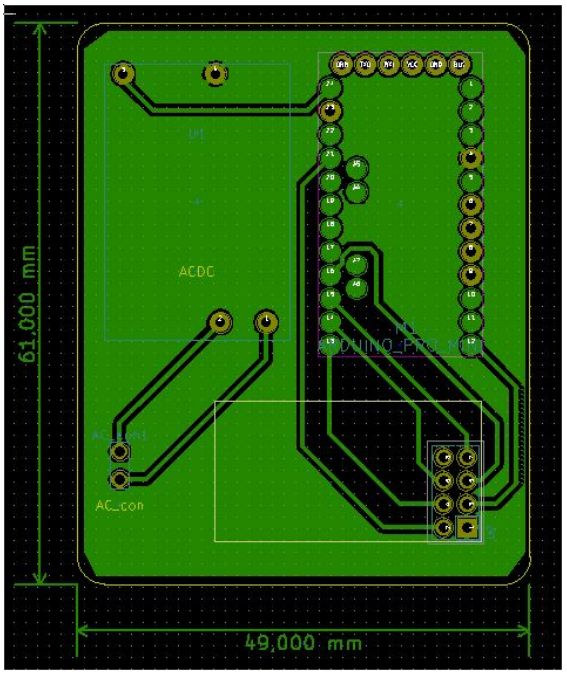


Figura 3.10: Diseño primera PCB

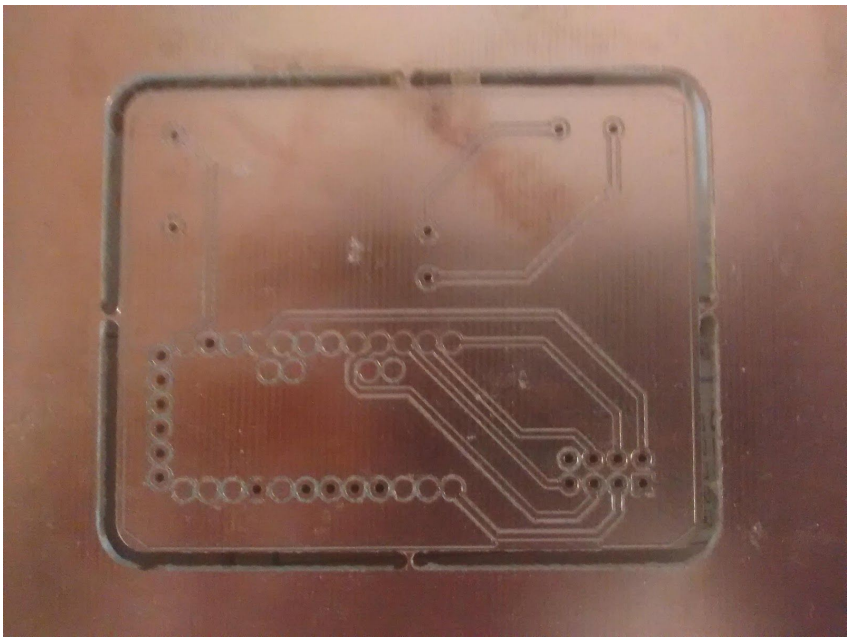
En el diseño superior podemos ver que la PCB está diseñada a una sola cara, la cara inferior; que el tamaño usado para las vías es de 0,474 milímetros, y que éstas han sido conectadas del mismo modo que habíamos designado en nuestro esquemático. Además tenemos un plano de tierra, es decir, todas las conexiones de GND van conectadas al gran plano verde que rodea el resto de las vías, como se puede apreciar en la imagen.

Los componentes y vías están dispuestos para que no se pisen unos con otros, y en la placa de arduino que va soldada a modo SMD, podemos ver que hay varios pines Through-hole. Pues bien, esto se debe a que, para realizar las conexiones con el resto de componentes que usamos, como el relé o el sensor de movimiento, necesitaremos tener pines a los que adherir los cables, por lo que, en lugar de soldar la placa directamente sobre la PCB, algunos tendrán un pin atravesando ambas placas, para las conexiones.

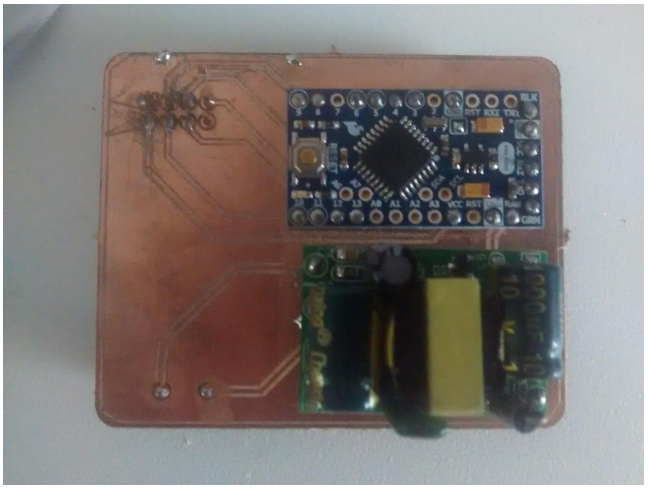
El resto de componentes no podían soldarse como smd, dado que, en el caso del módulo de radio y el conector a corriente alterna, ya tenían pines desde un comienzo y se precisaba que fueran, por tanto, through-hole; mientras que en el caso del transformador,

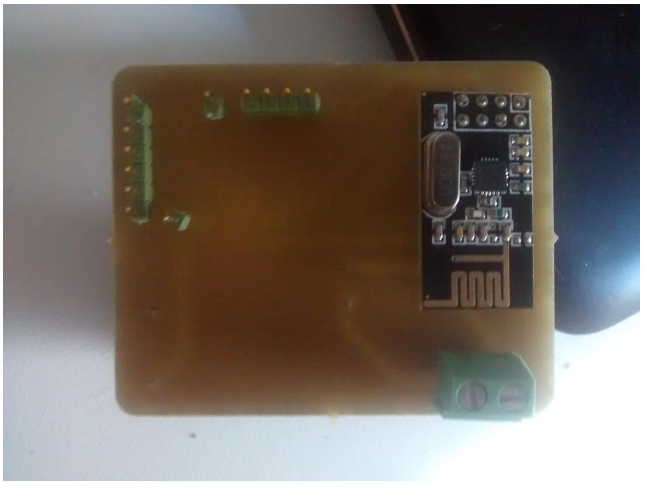
que no tenía pines, sí contaba, sin embargo, en la parte inferior, con varios componentes que impedían que pudiera pegarse a la PCB para realizar el soldado SMD, por lo que hubo que añadirle unos pines especialmente largos, para poder soldar a la placa con cierta comodidad.

El resultado final es una PCB de apenas seis centímetros de largo por cinco de ancho, suficientemente pequeña para poder entrar en las cajas de cables que nos encontramos en el sótano.



Una vez soldamos los componentes a la placa, para tenerla por fin terminada, el resultado es el siguiente:





3.4.2 Segunda Placa

La segunda placa aúna los componentes anteriores junto con un módulo L293DNE. Se ha usado la placa anterior como base para generar ésta, ya que el resto de componentes siguen siendo los mismos e incluso con las mismas conexiones en ambas placas.

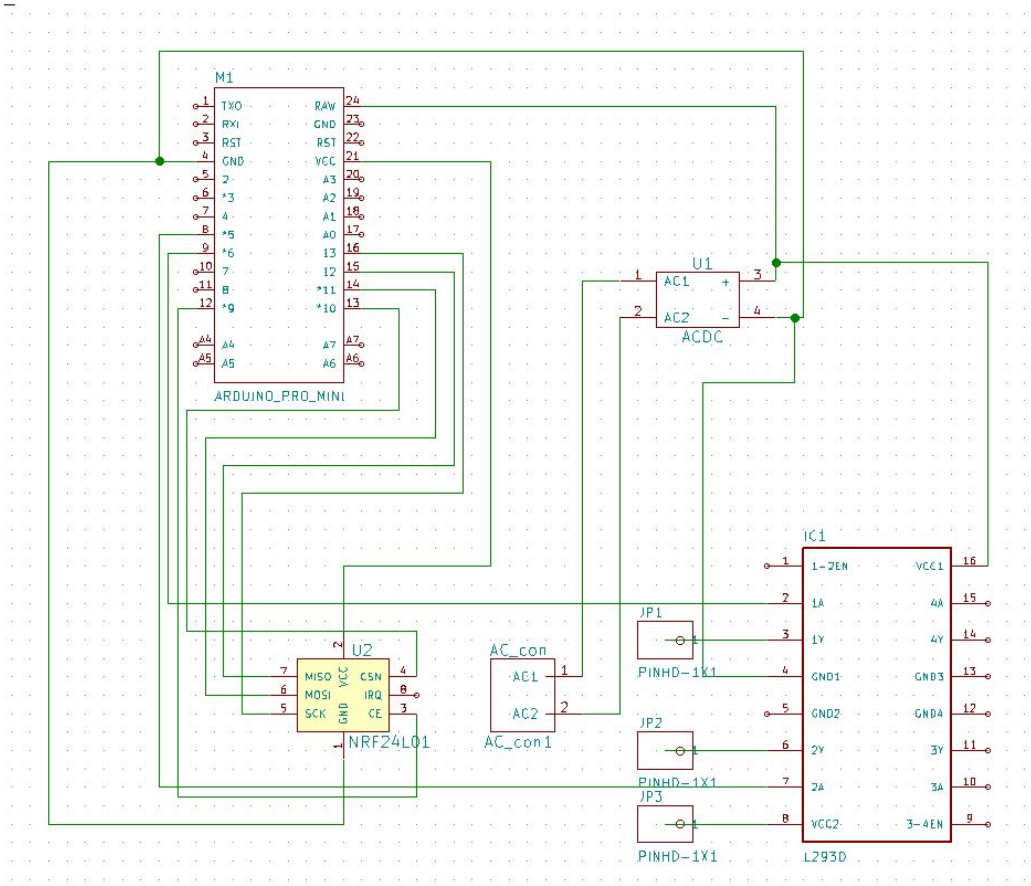


Figura 3.11:Esquemático segunda PCB

En el esquemático de la figura 3.1, podemos ver las conexiones del módulo LD293DNE. Éste conecta señales 1A y 2A a las señales pwm del arduino que están configuradas, como se podía ver en el código, en los pines cinco y seis de arduino. Además de esto, se necesitan 5V en el pin VCC 1, que es la señal de VCC lógica, mientras que en

VCC2, la señal VCC que se usará para controlar el motor, va conectada directamente a la parte positiva de una pila de 9V. Por último, conectamos GND1 a GND, aunque también se podría haber hecho esto mismo con GND2.

Del mismo modo, podemos ver que salen tres vías del módulo, entre ellas la de VCC2. Esto es porque necesitamos conectar cables a estos pines del módulo, y lo haremos añadiendo nuevos pines, para facilitar la conexión; los otros dos pines en los que podemos ver esto son los que llevarán conectado al motor, 1V y 2V , sin importar dónde conectemos el positivo y el negativo, ya que sólo cambiará el sentido de giro según los datos del pwm, pero no resulta relevante a la hora de la implementación.

Volvemos de nuevo al editor de PCB y ahora tenemos que hacer varios cambios más en la placa que en el esquemático.

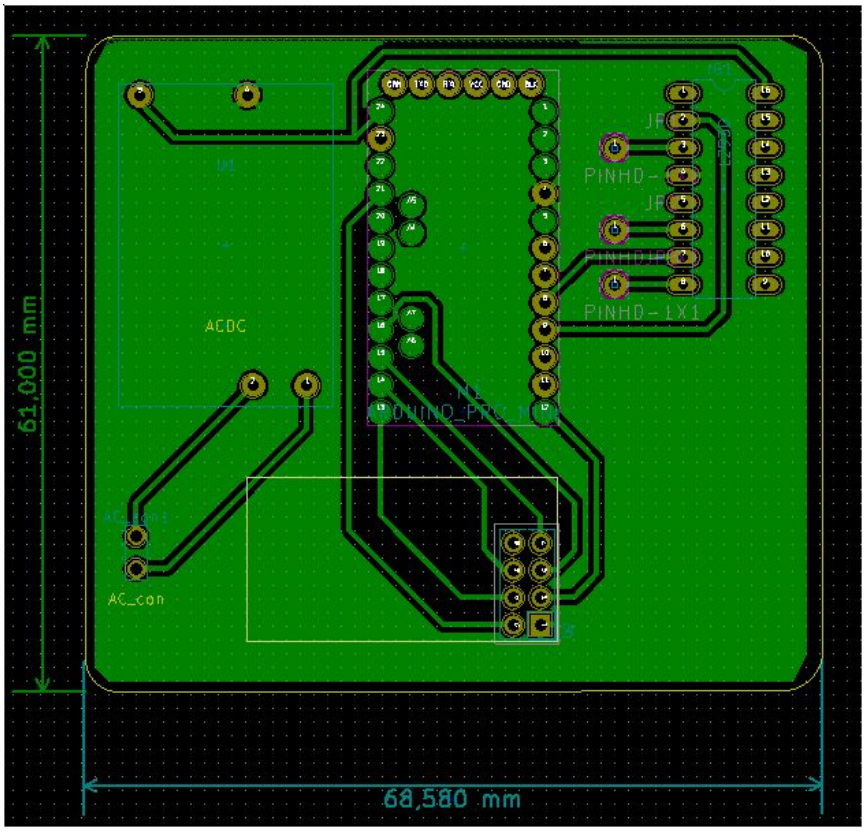


Figura 3.12: diseño PCB segunda placa

En este diseño, la parte izquierda, en general, se mantiene prácticamente igual, pero en la derecha añadimos el módulo LD293DNE, con los tres agujeros para los pines que dispusimos en el diseño del esquemático.

Además de esto, ahora tenemos que incorporar las vías que tiene el módulo con la placa de arduino (las dos señales de pwm) y con el transformador, para alimentarse con sus 5V, en lugar de los 3,3V de arduino, que no bastarían para que funcionara correctamente. GND, como va directamente al plano de tierra, no necesita de ninguna vía adicional en el diseño.

Asimismo, para los tres interruptores necesitaremos algunos pines extra, razón por la cual los pines siete y ocho de arduino han pasado a ser trough-hole en lugar de SMD, aunque realmente sólo haría falta uno, pero se ha decidido usar dos, por si hubiera fallos con alguno de ellos.

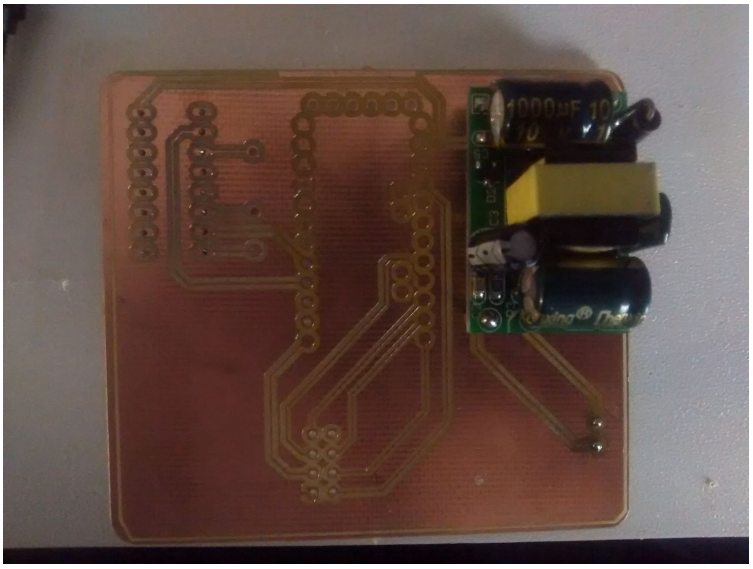
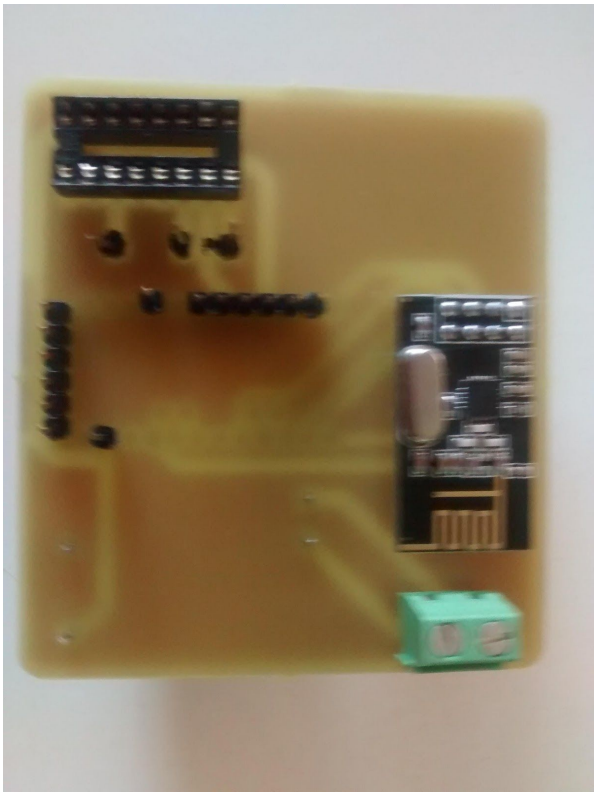


Figura 3.13: Placa de la segunda PCB

Como puede apreciarse comparando la figura 3.13 con la 3.7, el tamaño de esta PCB es notablemente superior, debido a que, como podemos ver en la parte izquierda de la imagen, se ha colocado el componente L293DNE y los pines necesarios para que éste controle el motor adecuadamente.



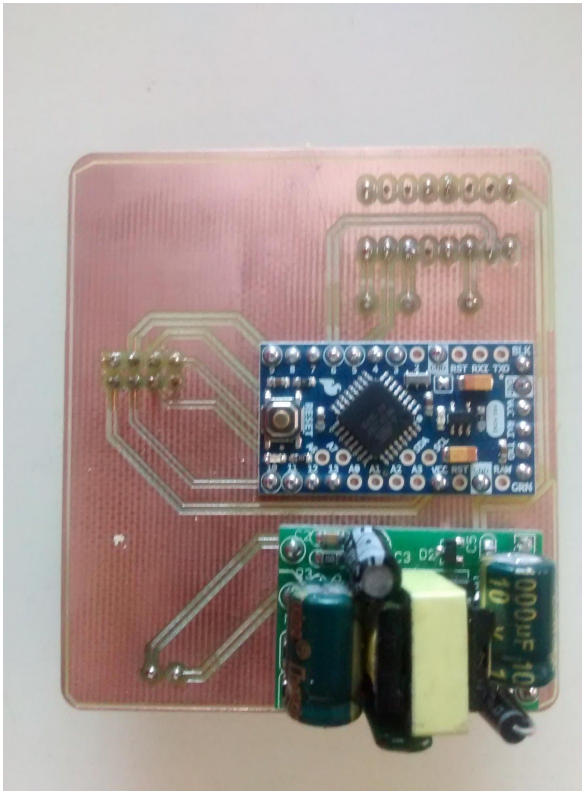


Figura 3.14 y 3.15: Placa de la segunda PCB soldado superior e inferior

3.5 Diseño de las Pruebas

En este apartado veremos cuáles han sido las pruebas realizadas para asegurar el correcto funcionamiento de los nodos, y cómo se han ido desarrollando, con sus errores y cambios, hasta llegar al modelo final.

3.5.1 Descripción de las pruebas realizadas

Se ha seguido siempre con los nodos un patrón de tres pruebas para ir asegurando el correcto funcionamiento de todas partes. En primer lugar se hacían pruebas con placas de arduino y la radio, más los componentes que fueran necesarios para el nodo. De este modo nos aseguramos de que el programa que estamos subiendo en arduino funciona correctamente.

En esta fase se ha usado normalmente una placa de arduino uno, en lugar de las arduino pro mini, por varias razones: facilitaba la programación de la placa reiteradas veces, nos permitía conectar los componentes con mayor facilidad, nos proporcionaba una salida de corriente de 5V, necesaria en algunos casos, como con el motor o el detector de movimiento y nos permite, con menos problemas, ver los resultados de nuestros `Serial.print()` en el IDE de arduino.

También podríamos asegurar con esto que los sensores y actuadores no produjeran ningún error inesperado, y en caso de encontrar algún componente defectuoso, se podría haber desechado sin causar más problemas, ya que, por ejemplo, si se soldara una radio defectuosa a una de las placas, el coste de tiempo y esfuerzo para corregir ese fallo hubiera sido mucho mayor que detectándolo antes de soldar.

Este primer paso, en general, no supone muchos problemas, más allá de alguna pequeña confusión a la hora de escribir algún programa.

Una vez seguros de que el código funcionaba correctamente, pasamos a la segunda fase, que consistió en probarlo en una placa ya soldada. Con esta fase verificábamos

que todo funcionaba correctamente con la placa, ya que a veces, a pesar de asegurarnos con un polímetro, siempre que las conexiones parecían estar correctas se pasaban detalles por alto.

También resulta, por supuesto, mucho más sencillo programar el arduino antes de colocarlo en la posición que deba ocupar, ya que suelen ser sitios de acceso algo complejos. En esta fase tenemos que tener la placa ya soldada con los componentes que usaremos y comprobar que no aparece ningún nuevo problema.

A pesar de todo, surgieron problemas con bastante frecuencia, por distintas causas: a veces la radio parecía no funcionar por algún motivo, otras veces alguno de los pines no estaba bien soldado o, a la hora de apretar el agarre de los cables en el conector de corriente alterna, éste se movía y dejaba de estar soldado a la placa. Es una fase en la que, a pesar de no haberse encontrado tantos fallos como en la primera, éstos han requerido bastante esfuerzo para corregirse y, por tanto, son los que más tiempo han exigido a lo largo de todo el desarrollo del proyecto.

Finalmente, la tercera fase de las pruebas consistía en probar el sistema una vez instalado y con las conexiones que iba a tener realmente. En esta fase comprobamos que no sólo funcionaba todo lo anterior, sino también las reglas y operaciones creadas en el controlador.

Aquí, la mayor parte de los fallos se produjo por confusiones con el cableado o por cables mal conectados en la placa, debido a la dificultad a la hora de hacer estas conexiones; al trabajar con tantos cables, algunas veces resultaba complicado saber cuáles podrían ser los que necesitábamos para nuestro sistema, así como a la hora de conectar los cables con los empalmes, la incomodidad, debido a la altura a la que éstos se encuentran, hacía muy difícil conectar correctamente el sistema, incluso usando fichas de empalme para hacerlo más sencillo.

También eran frecuentes los fallos a la hora de programar un script para el controlador, ya que las funciones y métodos que se usaban en este entorno no eran fáciles de encontrar y, a pesar de tener varios ejemplos,o bien requerían muchas modificaciones poco intuitivas o bien, por usar una versión diferente a la de los ejemplos de MyController acababa dando algún tipo de error.

3.5.2 Errores y cambios

Durante la primera fase de pruebas no hay mucho que comentar, ya que, a pesar de ser la fase donde más errores se han producido, todos se trataban de algún problema en el código, fácil de cambiar, o de un cable conectado en mal lugar.

El código que más problemas dio fue el del motor, puesto que no había ningún programa de motor en mysensors, y hubo que crearlo partiendo de un servo y cambiando, prácticamente, todo el código para hacer que, en lugar de indicar una posición, manejara el giro del motor.

En ese programa fue muy importante poder ver los resultados de los print en pantalla con facilidad y reprogramar la placa fácilmente, ya que hubo que hacer ambas cosas en muchas ocasiones, para ir puliendo el programa hasta conseguir que funcionara tal como como se deseaba. En principio, las señales que se cogieron de pwm no eran adecuadas, porque los mapeados no funcionaban como debían, y el motor, en consecuencia, no era capaz de operar como estaba previsto.

Aparte de este programa, el resto resultó bastante simple gracias a los códigos que podemos encontrar en la web de mysensors. Tampoco se encontró ningún componente defectuoso durante esta fase.

La segunda fase de las pruebas resultó sin duda las más dura, dado que los fallos estaban, en general, relacionados con problemas durante el soldaje de la placa, y esto requería el volver a por el equipo de soldaje, encontrar el punto o puntos que fallaban y soldarlos de nuevo, o eliminar la parte soldada que estaba en contacto con el plano de tierra, lo que resultaba una tarea muy tediosa.

Las correcciones realizadas en esta fase fueron, efectivamente, las que conllevaron una mayor cantidad de esfuerzo y tiempo. Las placas tardaron, cada una de ellas, alrededor de una hora en soldarse, y, luego, las correcciones que hicieron falta en todas, conllevaron un par de horas más, como mínimo. En algunos casos también aparecieron errores aparentemente absurdos, como el fallo de una de las radios que, después de quitarse y cambiarse por otra en una PCB, se probó a parte, como en la primera fase, y se comprobó que funcionaba correctamente. Esto ha sucedido más veces después, y con tan sólo desconectar y conectar el sistema, en general, se soluciona.

Otro de los problemas de esta fase, consistió en el extremo cuidado que había que poner cuando se manipulaba la PBC, al estar enchufada directamente a una corriente de 220V. Uno de los transformadores acabó destrozado al realizar un movimiento algo más brusco y unirse los cables que transmitían la corriente hasta la PCB, y en más de una ocasión, al presionar el botón de reset de arduino he recibido una descarga de la placa por apoyar la mano donde no debía.

Por esto era siempre mejor realizar las conexiones de cables antes de conectar la placa a corriente, y una vez conectada, si hacía falta tocarla en algún momento, usar un guante de plástico para evitar las posibles descargas.

Finalmente, en la última fase, los errores más destacables fueron: los fallos a la hora de escoger los cables que debían conectarse, o el modo en que debían conectarse al sistema, y el fallo de funcionamiento de la radio que se comentó anteriormente, que se dio en alguna ocasión con el nodo del cuarto de baño.

El mayor problema, no obstante, fue que, lo que en la primera fase hubiera sido un simple cambio de cables, en la última fase, al tener que hacerlo mediante fichas de empalme, subido a una escalera y cortando la luz previamente, resultó mucho más complicado.

Antes de conectar los cables se intentaba hacer planos de cuales serían cada uno de los cables que se veían, o al menos los que eran necesarios para el sistema, pero a pesar de todo, las confusiones fueron inevitables en la mayoría de ocasiones. Además, incluso con las fichas de empalme, los cables no resultaban fáciles de conectar, debido a su cantidad; en algunas de estas fichas había tantos, que no permitían apenas la entrada de un cable más, que fuera necesario, por ejemplo, para alimentar la placa.

En definitiva, gracias a la división de pruebas, se consiguió ir asegurándose del funcionamiento del sistema de la forma más sencilla posible, quitándose de en medio la mayor cantidad de errores posibles de la forma más simple, ya que si, por ejemplo, los errores de código hubieran sido cubiertos en la tercera fase de las pruebas, el tiempo requerido y esfuerzo requerido para solventarlos hubiera sido mucho mayor que el que nos ha llevado, haciéndolo con placas de arduino uno en una mesa.

Capítulo 4: Resultados del proyecto

4.1 Resultados Obtenidos

Durante este capítulo explicaremos los resultados obtenidos, así como las pruebas y ajustes que se han realizado en los sistemas, una vez montados definitivamente en la planta baja del hogar.

Una vez implementados los sistemas en la vivienda, hemos comprobado que son muy útiles para realizar determinadas tareas, como la activación de la depuradora durante un periodo largo de tiempo. Sin embargo, si la velocidad de respuesta de nuestro sistema resultara crucial, tendríamos un problema, ya que el encendido de luces, por ejemplo, se ha retrasado notablemente con respecto a la velocidad de respuesta, al pulsar los interruptores.

El mismo problema tenemos en el cuarto de baño, aunque a una menor escala. Así, en el salón, el retraso puede ser de hasta cinco segundos en algunas ocasiones, mientras que en el cuarto de baño, como máximo, apenas llega a tres. Esto seguramente se deba a que, entre las operaciones realizadas por el controlador de las luces, está la de ejecutar un script, que le resultará más costoso temporalmente que ejecutar una operación de encendido o apagado, sencilla.

En el cuarto de baño, una vez instalado el nodo, hubo que realizar algunos ajustes en el sensor, dado que los tiempos de encendido eran demasiado cortos y la sensibilidad de movimiento muy baja, con lo que se producían muchos cortes, incluso estando dentro del aseo.

Se aumentó el tiempo de encendido del sensor una vez que detecta movimiento, y la sensibilidad se subió un poco. Aún así, actualmente, si durante un periodo de tiempo intermedio nos mantenemos lo bastante quietos dentro del aseo, la luz se apagará,

obligándonos a realizar algún movimiento para que se encienda de nuevo. Pero los cortes han pasado de ser varios, a uno como máximo, y tan sólo en caso de moverse muy poco en el interior.

La bombilla exterior responde incluso más rápido que la del aseo, al estar implementado únicamente con un botón virtual. Ésta se ha usado relativamente poco en comparación con el resto del sistema, pero no aparenta dar ningún problema en su función, al igual que la depuradora, que desde que se instaló el sistema, se ha activado automáticamente durante un periodo de cuatro horas, tal y como nos lo marcaba nuestro objetivo.

Como se ve en las imágenes superiores, la planta inferior, una vez completada la instalación, sigue en apariencia prácticamente igual que antes, a excepción del sensor de movimiento, que podemos ver en el cuarto de baño asomando por el techo, y del nodo que se controla la persiana, el cual ha quedado expuesto al exterior.

En definitiva, creo que el sistema domótico que hemos instalado facilita algunas tareas, y nos permite tener una casa que podemos controlar a distancia, desde el móvil, y que, incluso, funciona automáticamente. A cambio tenemos la mencionada desventaja del aumento en el tiempo de respuesta, pero esto, a mi parecer, no es una desventaja muy relevante a la hora de controlar una casa, ya que el tiempo de respuesta no es esencial.

A continuación, en este capítulo se nos informará sobre el coste económico y la inversión de tiempo que se ha realizado, con su correspondiente distribución de horas para cada tarea.

4.2 Análisis de costes

En este apartado de la memoria vamos a desglosar los costes de la realización del proyecto, teniendo en cuenta para esto todas las placas utilizadas en cada uno de los nodos, así como los sensores que se han usado en la implementación del sistema domótico, las radios y los transformadores.

Materiales	Cantidad	Precio/Ud. (€)	Coste (€)
Arduino Pro Mini	4	1,32	5,28
RaspberryPi	1	22,50	22,50
Transformador	4	1,02	4,08
NRF24L01	5	0,65	3,25
3 unidades HC-SR501 PIR Sensor movimiento	1	4,99	4,99
Actuador relé	3	2,00	6,00
Motor CC	1	12,80	12,80
Paquete 5 sensores campo magnético BTE 16-20	1	9,99	9,99
Interruptor	1	1,60	1,60
L293DNE	1	3,97	3,97
WINGONEER 10 unidades sensores temperatura	1	14,99	14,99
Coste total de los materiales del prototipo			91,65

El presupuesto definitivo de nuestro sistema es de ciento veintiseis con noventa y cuatro euros. Este presupuesto es, en comparación con otros sistemas del mercado para realizar tareas iguales o similares al nuestro, muy reducido, ya que la adquisición de un sistema domótico como este, generalmente quintuplica este precio, llegando a más si lo quisiéramos con la instalación en el hogar incluida.

A esto debemos añadir los costes de la mano de obra de un personal cualificado, teniendo en cuenta el tiempo de instalación del sistema en un hogar, para calcular los costes totales.

Coste del material (€)	Costes del personal/hora (€)	Tiempo de instalación (hora)	Coste de instalación (€)
91,65	9	70	721,65

Los precios de instalación de sistemas domóticos, en general, alcanzan cifras de miles de euros, mientras que nuestro modelo consigue un precio mucho más asequible para una instalación como la que hemos hecho en nuestra planta inferior del hogar. Por supuesto, un mayor número de sistemas a domotizar en un hogar conlleva una subida en el precio total, sobre todo por el tiempo invertido para instalarlo. Pese a ello, se mantendría por debajo de lo que podemos encontrar a día de hoy.

4.3 Análisis temporal

Para el presente trabajo de fin de grado se han estimado una serie de hitos y de horas en relación a lo que se pensaba, en un principio, que se tardaría en completar la fase de desarrollo correspondiente. En primer lugar, presentaremos los hitos de forma concisa, explicando los objetivos que darán por finalizada dicha fase. En segundo lugar, una tabla

con las relaciones entre cada hito, el tiempo estimado para su finalización, y el tiempo real invertido para completarlo. Finalmente un diagrama de Gantt, para exponer visualmente, con los datos de la tabla anterior, el desarrollo temporal del proyecto de fin de grado.

4.3.1 Hitos

H01: Diseño de las estancias: concretar los objetivos que se plantean en cada parte del hogar, analizar los requisitos para completar estos objetivos, y comprobar qué habrá que modificar en la vivienda y cómo habrá que hacerlo, para alcanzar los objetivos propuestos.

H02: Generar el código que se usará en las placas de arduino mini, que irán en los nodos, y comprobar su correcto funcionamiento con los sensores.

H03: Crear las operación, reglas, scripts y timers necesarios en el entorno de MyController, y probar su correcto funcionamiento con los códigos desarrollados para los nodos.

H04: Diseñar las PCBs, que se usarán durante el proyecto, con la herramienta KiCad, y las librerías y huellas que hagan falta para crear estas PCBs. Comprobar que tanto el diseño, como las placas, son válidas y están conectadas correctamente.

H05: Soldar los componentes a las PCBs ya creadas y comprobar que se han soldado correctamente, sin pines que vayan al plano de tierra indebidamente. Comprobar también que, enchufado, el sistema se enciende, y funcionan correctamente todas las partes.

H06: Montaje final de los nodos en sus cajas de cables y comprobación de que el sistema automatizado, actúa conforme a lo descrito en los análisis de requisitos para cada nodo.

4.3.2 Distribución temporal

Hitos		Horas estimadas	Horas invertidas	Tiempo total estimado	Tiempo total invertido
H01	Diseño de las ideas a implementar	6	6	18	20
	Analizar requisitos	4	6		
	Comprobar instalación del hogar	8	8		
H02	Desarrollar código para los programas de arduino	50	55	80	75
	Testear código	30	30		
H03	Desarrollar operaciones para MyController	30	30	60	60
	Testear operaciones creadas	30	30		
H04	Diseño de las PCBs	35	35	50	50
	Prueba de las PCBs	15	15		
H05	Soldado de las PCBs	12	15	42	55
	Comprobación del soldado	30	40		
H06	Montaje final de los sistemas	40	45	105	120
	Pruebas del	15	15		

	sistema implementado				
	Redactar documentación	50	60		

Cuadro 1: Distribución temporal por hitos del proyecto

Observando el cuadro uno, vemos que la mayor carga de tiempo recae sobre el desarrollo del código de arduino y MyController, seguido de cerca por el montaje final y el diseño de las PCBs. Esto se debe a que fueron las tareas más complicadas y, en el caso del diseño del código, en la que más cantidad de fallos hubo que ir corrigiendo a lo largo de su desarrollo.

4.3.3 Diagrama de Gantt

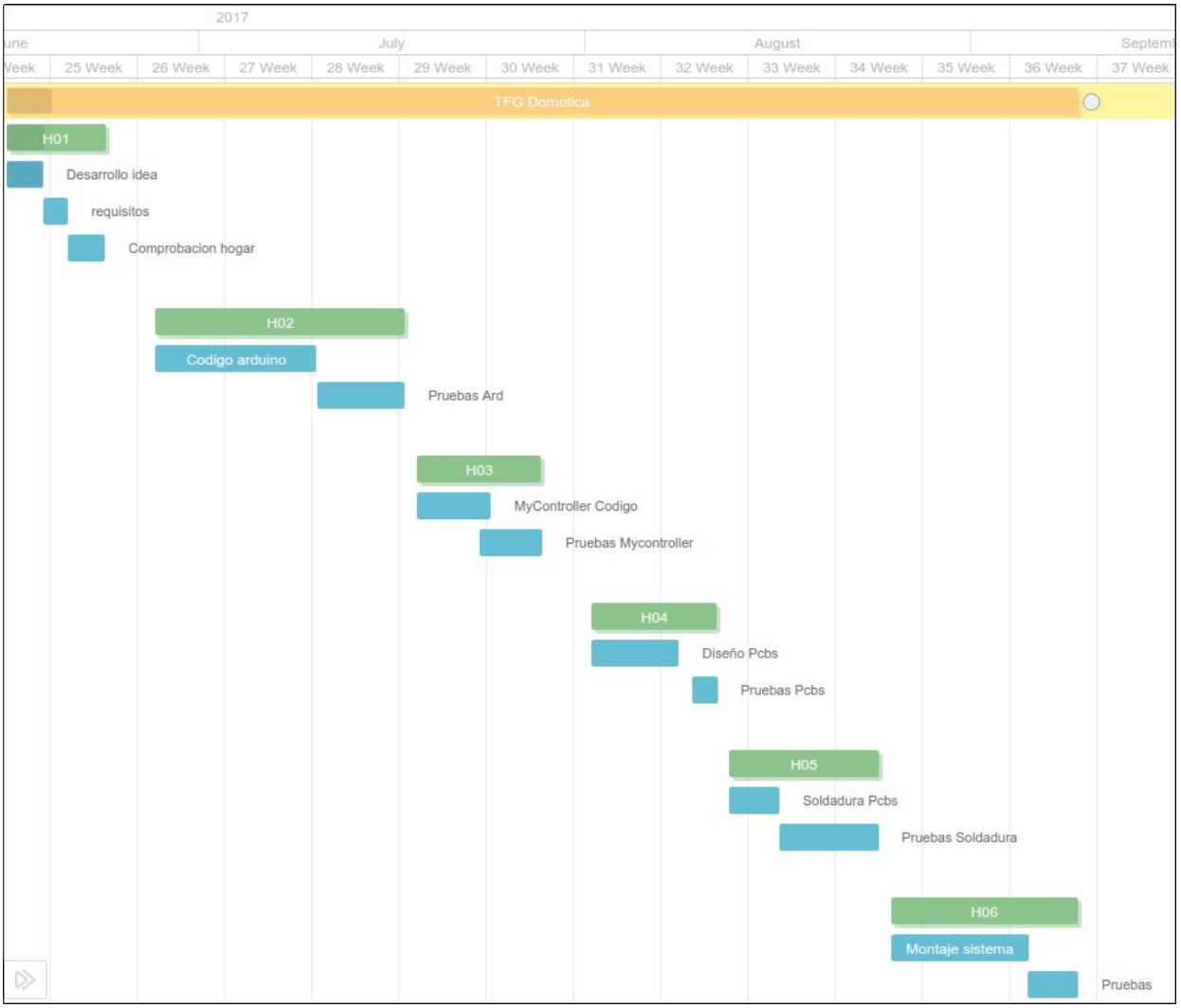


Figura 4.1: Diagrama de Gantt

En el diagrama de la figura 5.1 podemos ver la distribución temporal a lo largo de las semanas de cada mes. Comprobamos que, a pesar de que algunas de las tareas finales requirieron mucho tiempo, no se tardó demasiados días en completarse, debido a que, como la fecha límite se acercaba, la cantidad de horas dedicadas a completar el trabajo, se fue incrementando gradualmente, con el fin de alcanzar los objetivos en el plazo de tiempo adecuado.

Capítulo 5: Conclusiones

Los objetivos que nos planteamos al principio, se han ido viendo satisfactoriamente resueltos, siendo algunos de mayor complejidad que otros, pero alcanzando las metas previstas en cada uno de los nodos usados en el sistema y durante las distintas etapas de desarrollo, todo esto en los tiempos estimados en principio, y manteniendo los costes bastante bajos a la hora de implementar el sistema.

Hemos sido capaces de domotizar un sótano completo, usando herramientas y componentes bastante baratos, teniendo en cuenta que, si examinamos páginas dedicadas a la domotización del hogar, los precios de los componentes que resultan más sencillos de usar, tienen unos precios mucho más elevados, como norma general.

La domótica está cada vez más a la orden del día, y con este proyecto podemos ver que, con ganas y esfuerzo, no está solo al alcance de unos pocos, sino que todos podríamos permitirnos tener nuestra propia casa domotizada sin demasiadas complicaciones.

La implementación de este sistema, además, no resulta excesivamente compleja, gracias a toda la información que podemos encontrar en internet, especialmente, para este proyecto, en la página de MySensors, gracias a la cual se ha contado con enormes facilidades a la hora de realizar los programas de arduino, tarea que hubiera resultado un problema bastante mayor sin esta ayuda.

Hemos conseguido diseñar las placas que hemos necesitado (manteniendo también en este aspecto el interés de hacerlo de bajo coste, haciendo que las placas fueran a una sola cara en todos los casos), realizar el diseño de conexiones de nuestro sistema dentro del hogar, y crear todos los programas necesarios, tanto para los arduinos en los nodos del sistema, como para el mismo controlador.

Por último, una futura ampliación del proyecto, sería desarrollar una aplicación de móvil para android que funcionara como lo hace la web creada por MyController, con los botones que hemos añadido en la pantalla inicial de ésta para facilitar el acceso al control de la casa.

Referencias

1. Proyecto mysensor.org: https://www.mysensors.org/
2. Sistemas KNX: https://www.knx.org/es/
3. Sistema lonwork: https://en.wikipedia.org/wiki/LonWorks
4. Proyecto Arduino disponible en https://www.arduino.cc/
5. Proyecto Raspberry pi disponible en https://www.raspberrypi.org/
6. Sistema domótico abierto Calaos: https://calaos.fr/en/
7. Sistema domótico abierto Domoticz: https://domoticz.com/
8. Sistema domótico abierto Home Assistant: https://home-assistant.io/
9. Sistema domótico abierto OpenHAB: https://www.openhab.org/
10. Sistema domótico abierto OpenMotics: https://www.openmotics.com/
11. Proyecto Mycontroller: http://www.mycontroller.org/