PROYECTO FINAL DE GRADO

Ingeniería electrónica industrial y automática

Easy Control Smart Home

ECSH

Control de un hogar inteligente

Autor: Francesc Agustin Mora (franagustinmora@hotmail.com)

Tutor: Manuel Lamich Arocas

Fecha: Mayo 2017

Departament d'Enginyeria Electrònica

Escola d'Enginyeria de Terrassa

UNIVERSITAT POLITÈCNICA DE CATALUNYA

AGRADECIMIENTOS

Al tutor del proyecto Manel Lamich por ofrecerme la posibilidad de llevar a cabo este proyecto y por su apoyo, guía y colaboración durante la duración de este.

A todos aquellos compañeros y profesores que han hecho que el camino haya sido menos complicado.

A mis amigos por todos esos momentos en los que me han ayudado a desconectar y poder coger fuerzas para continuar, trabajando y estudiando.

Por último, a mí familia por la ayuda, dedicación y como no la educación, ya que siempre han sido un ejemplo para mí. Sin olvidar la motivación que siempre me han inculcado para llevar a cabo todo lo que me he propuesto.

Y un especial agradecimiento a mi esposa por su compañía, motivación, consejos y apoyo incondicional para conseguir todo aquello que nos propongamos.

RESUMEN

El propósito de este proyecto es desarrollar una aplicación con la que podremos controlar desde una pantalla táctil, Smartphone o pulsadores autónomos sistemas como la iluminación, el control de persianas, el control de temperatura, cámaras de seguridad, entre otros sistemas existentes en nuestro hogar mediante la comunicación WIFI del Router.

Me gustaría remarcar que el sistema será diseñado con un bajo coste, código abierto y modular, para que cualquier usuario sea capaz de diseñarse su sistema de control y modificarlo con unos conocimientos básicos.

El proyecto consta de tres sistemas importantes. El primero se trata de la central de procesamiento, se ha utilizado una Raspberry Pi 3, esta es la encargada de interpretar y procesar todas las señales del sistema de control. El lenguaje de programación es Python debido a que favorece a que el código sea legible y su programación orientada a objetos. Este dispositivo nos permite dotarlo de una pantalla táctil o un control remoto desde un Smartphone.

A simple vista en el segundo sistema no apreciamos ninguna diferencia respecto a un interruptor convencional, pero internamente está formado por un generador inductivo, que nos ofrece un amplio abanico de posibilidades. Este interruptor del fabricante EnOcean no precisa mantenimiento, baterías y su punto fuerte es la facilidad de modificar su localización sin realizar ningún tipo de obra.

Por último nos encontramos con el receptor WIFI, que recibirá las indicaciones de la central de procesamiento. El cerebro del receptor se basa en un revolucionario chip ESP8266, y su versatilidad para poder controlar mediante WIFI cualquier aplicación que se nos ocurra y actualmente su precio ha disminuido considerablemente, lo que lo hace todavía más accesible. Su programación es compatible con el IDE Arduino.

Si te ha parecido interesante este resumen, te animo a que dediques unos minutos a leer el proyecto y te beneficies del esfuerzo que he dedicado para facilitarte el diseño de tu propio sistema de control.

RESUM

El propòsit d'aquest projecte es desenvolupar una aplicació amb la que serem capaços de controlar des de una pantalla tàctil, Smartphone o polsadors autònoms sistemes com la il·luminació, el control de persianes, control de la temperatura, càmeres de seguretat, i altres sistemes existents a la nostra llar mitjançant la comunicació WIFI del Router.

M'agradaria remarcar que el sistema està dissenyat amb un cost baix, codi obert i modular, per tal de que qualsevol usuari sigui capaç de dissenyar-se el seu sistema de control i modificar-lo amb uns coneixements bàsics.

El projecte consta de tres sistemes importants. El primer es tracta de la central de processament, s'ha utilitzat una Raspberry Pi 3, serà l'encarregada d'interpretar i processar totes les senyals del sistema de control. El llenguatge de programació es Python degut a que afavoreix a que el codi sigui llegible y la seva programació esta orientada a objectes. Aquest dispositiu ens permet dotar-lo d'una pantalla tàctil o un control remot des de un Smartphone.

El segon sistema no sembla que hi existeixi cap diferència amb un interruptor convencional, però internament esta format per un generador inductiu, que ens ofereix un ampli ventall de possibilitats. Aquest interruptor del fabricant EnOcean no precisa manteniment, bateries i el seu punt fort es la facilitat de modificar la seva localització sense necessitat de realitzar cap tipus d'obra.

Per últim ens trobem amb el receptor WIFI, que rebrà les indicacions de la central de processament. El cervell del receptor es basa en un revolucionari chip ESP8266, i la seva versatilitat per poder controlar a traves de WIFI qualsevol aplicació que se'ns acudeixi i actualment el seu preu ha disminuït considerablement, el que el fa encara més accessible. La seva programació es compatible amb l'IDE Arduino.

Si t'ha semblat interesant aquest resum, t'animo a que li dediquis uns minuts a la lectura del projecte, i et beneficiïs del esforç que he dedicat per facilitar-te el disseny del teu propi sistema de control.

ABSTRACT

This project develops an application that utilizes a WIFI Communication Router to control lighting, blinds, temperature, security cameras, among other systems in their homes with a touch of a screen, simply by using their smartphone or autonomous push buttons.

I would like to point out that this system will be particularly designed to be low cost, open source and modular, in order for users to design and assemble their control system and modify it with basic knowledge.

This project consists of three important systems. The first one component is the processing center, which uses a Raspberry Pi 3. This component is responsible of interpreting and processing all the signals of the control system. The programming language used is Python, as it facilitates the code to be readable and allows the programming to be oriented to objects. This device allows us to equip it with a touch screen or a remote control directly from a Smartphone.

At a glance, the second system does not seem to be any different from a conventional switch. However, internally it is formed by an inductive generator that offers us a wide range of possibilities to be utilized. This particular switch, from the manufacturer EnOcean does not require maintenance, batteries, and its resistance facilitates the modification of its location without doing any type of work.

Finally, we have WIFI receiver, which receives indications from the processing center. The brain of the receiver is formed by a revolutionary ESP8266 chip and its versatility will allow us to control any application by WIFI. Currently, its price has decreased considerably, which makes it even more accessible. Its programming is also compatible with the Arduino IDE.

If you have found this information to be useful, I welcome you to take a few minutes to read the project and benefit from the effort that I have dedicated in order to facilitate the design of your own control system.

INDICE DE LA MEMORIA

INDICE DE IMAGENES

INDICE DE DIAGRAMAS

INDICE DE TABLAS

1. INTRODUCCIÓN

Cada año, aproximadamente unos 25 millones de toneladas de cobre se subministran en todo el planeta, es esencial para la vida, ya que, se utiliza para instalaciones eléctricas, electrónicas y subministro de agua potable

Actualmente, el cobre cada vez es más escaso y su demanda aumenta, aumentando de esta forma su coste. Por este motivo se están buscando alternativas, como puede ser la utilización de cableado de otros materiales para instalaciones eléctricas.

La sociedad utiliza en exceso los materiales que nos proporciona el planeta, pero estos no son ilimitados. Por lo tanto hemos de plantear alternativas para evitar acabar con estas fuentes.

En primer lugar, el enfoque del proyecto en cuanto impacto medioambiental tiene como objetivo ahorrar materiales y abaratar el coste de la instalación eléctrica, ya sea antigua o nueva dicha instalación.

Por otro lado cabe remarcar que en las últimas dos décadas las nuevas tecnologías han crecido exponencialmente. Si miramos a nuestro alrededor podremos observar personas con tecnología en sus manos. Se hace extraño encontrar una persona en nuestra sociedad que no disponga de un "Smartphone". La principal finalidad de dicho dispositivo es la de mantenernos comunicados con nuestro entorno social, incluso si esa persona se encuentra a miles de kilómetros de distancia.

En segundo lugar, estos dispositivos nos permiten controlar mediante aplicaciones nuestro correo, hacer búsquedas inteligentes en internet, estar al corriente de nuestro perfil en las redes sociales, realizar compras rápidamente, monitorear nuestras rutas, incluso hace innecesario la compra de un GPS o un reproductor de audio, entre infinidad de aplicaciones que podamos imaginar.

Y fue en este punto donde me surgió la pregunta,

¿Por qué no controlar nuestro hogar mediante un "Smartphone"?

2. ANTECEDENTES

Domótica es el término que se utiliza para denominar la parte de la tecnología (electrónica e informática), que integra el control y supervisión inteligente de los elementos existentes en un edificio de oficinas o en un hogar. Permite una gestión eficiente del uso de la energía, que aporta seguridad y confort, además de comunicación entre el usuario y el sistema. Se creó una asociación española que regula este sector, CEDOM (Asociación Española de Domótica e Inmótica).

El uso de las tecnologías de la información y las comunicaciones en la vivienda genera nuevas aplicaciones y tendencias basadas en la capacidad de proceso de información y en la integración y comunicación entre los equipos e instalaciones. Así concebida, una vivienda inteligente puede ofrecer una amplia gama de aplicaciones en áreas tales como:

- Seguridad.
- Gestión de la energía.
- Automatización de tareas domésticas.
- Formación, cultura y entretenimiento.
- Monitorización de salud.
- Comunicación con servidores externos.
- Ocio y mantenimiento.
- Operación y mantenimiento de las instalaciones, etc.

Imagen 1. Domótica

La implantación de un sistema domótico supone una gestión inteligente de la energía consumida en la vivienda. El control de la iluminación, climatización, el caudal de agua y los electrodomésticos entre otros, permite un aprovechamiento mayor de los recursos utilizados y por tanto una reducción de tipo energética y por lo tanto de tipo económica.

También supone un plus en el manejo de los elementos del hogar, facilitando así la accesibilidad al usuario, incluso a personas con ciertas discapacidades, ajustándose a sus necesidades.

Asimismo, la utilización de un sistema domótico en el hogar implica un mayor confort para el usuario, desde el control remoto de los dispositivos a la automatización de encendidos o apagados según la hora del día.

Por último, un sistema domótico aporta seguridad a una vivienda. Mediante subsistemas de vigilancia tales como controles de intrusión, cierres automáticos, simulación de presencia y alarmas, los cuales pueden estar conectados a diversas asistencias.

Los sistemas domóticos, también conocidos como sistemas de automatización, gestión técnica de energía y seguridad para viviendas y edificios, se denominan internacionalmente como HBES (Home and Buildings Electronic Systems). Actualmente la norma que define los requisitos técnicos generales de estos sistemas es la UN-EN 50090-2-2.

Hoy en día la domótica es algo todavía poco extendido debido fundamentalmente:

1. El más importante desde mi punto de vista, es el coste.
2. No existe un estándar que regule y unifique.
3. Falta de información al usuario para un concepto nuevo, lo que genera desconfianza.
4. Complejidad de instalación y lo difícil que es la ampliación de esta a nivel usuario.

A pesar de esto las ventajas que ofrece la domótica son numerosas y pueden ser muy beneficiosas sobre todo para ciertos colectivos de personas como los ancianos o los discapacitados.

Nos encontramos con aplicaciones en las cuales se ha implantado indirectamente este concepto, como en el control de persianas y toldos que se ha extendido considerablemente, o dos aplicaciones relativamente antiguas como es la apertura de la puerta de párquines a distancia o cambiar el canal de la televisión.

Como se ha comentado en la introducción, al igual que ha sucedido con otras tecnologías, se piensa que la implantación generalizada de la domótica es sólo cuestión de tiempo.

3. OBJETIVO DEL PROYECTO

La finalidad del proyecto tiene varios objetivos, los cuales se podrían definir en dos puntos importantes:

1. Simplificar y abaratar la instalación eléctrica, mediante un control de todos los periféricos[1] existentes en una vivienda a través de comunicación Wireless de nuestro Router.
2. El diseño del sistema de control será de bajo coste, código abierto y modular, lo que implica una fácil ampliación del sistema para usuarios con escasos conocimientos.

En el proyecto nos encontraremos con unos objetivos específicos:

- Estudiar el funcionamiento de una Raspberry Pi 3.
- Programar en lenguaje Python (IDE PyCharm).
- Estudiar el funcionamiento de la comunicación wireless.
- Estudiar el funcionamiento del ESP8266 y su microcontrolador basado en Arduino.
- Programar C con el IDE Arduino.
- Estudiar el funcionamiento de una conexión remota con la Raspberry Pi 3.
- Estudiar el funcionamiento de un sensor de movimiento con tecnología infrarroja.
- Implementar un sistema de adquisición de imágenes (cámara de seguridad).
- Estudiar el funcionamiento de un sensor de temperatura y humedad (DHT22).
- Estudiar el funcionamiento de un sensor de monóxido de carbono y combustión (MQ-7).
- Diseñar un entorno grafico para una pantalla táctil (programa de entorno gráfico).
- Diseñar un receptor con un programa de diseño de PCBs y fabricar los gerbers.
- Fabricar prototipos del receptor WIFI con la LPKF.

[1] Aparatos y/o dispositivos auxiliares e independientes conectados a la unidad central de procesamiento. Como pueden ser interruptores, puntos de luz, termostato, cámaras, etc.

4. DOMÓTICA

4.1. Elementos que conforman un sistema domótico

Un sistema domótico es capaz de recoger información proveniente de unos sensores o entradas, procesarla mediante un controlador y emitir órdenes a unos actuadores o salidas. El sistema puede tener la capacidad de acceder a redes exteriores de comunicación o información o servicios, por ejemplo, la red telefónica conmutada o internet. Estos sistemas se componen de diferentes tipos de elementos:

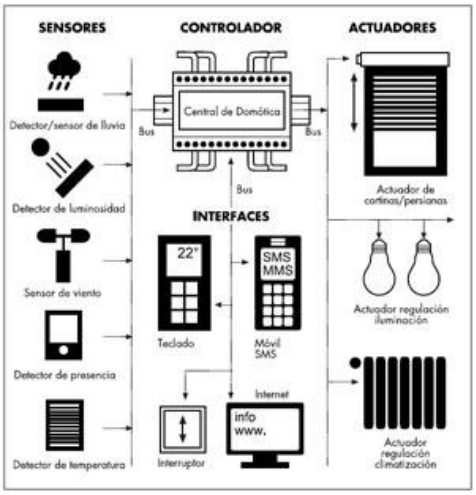


Imagen 2. Sistema domótico

- **Controlador:** Los controladores son los dispositivos que gestionan la información que reciben del sistema y deciden qué hacer en función de la programación previa. Puede haber un solo controlador, o varios distribuidos en función de la arquitectura del sistema.
- **Sensor:** El sensor es el dispositivo que monitoriza el entorno captando información que transmite al sistema (sensores de luz, temperatura, movimiento, humedad, lluvia, agua, gas, humo, viento, etc.).
- **Actuador:** El actuador es un dispositivo capaz de recibir una orden y ejecutarla, así cambiando las características del entorno domótico (encendido/apagado, subida/bajada, apertura/cierre, etc.).
- **Bus:** Es el medio de comunicación que transporta la información entre los distintos dispositivos, ya sea por una red propia, o por las redes de otros sistemas (red eléctrica, red telefónica, red de datos, etc.). Un sistema puede disponer de diferentes buses.
- **Interfaz:** La interfaz son los dispositivos (pantallas, móvil, Internet, interruptores) en que se muestra la información del sistema para los usuarios y donde los mismos pueden interactuar con el sistema.

4.2. Arquitecturas de los sistemas domóticos

4.2.1. Arquitectura descentralizada

La arquitectura descentralizada coexisten diversos controladores, interconectados por un bus. La inteligencia del sistema se reparte, y cada controlador contiene su propia configuración. El bus permite el envío de información entre los controladores y también a los distintos actuadores e interfaces conectados a los controladores.

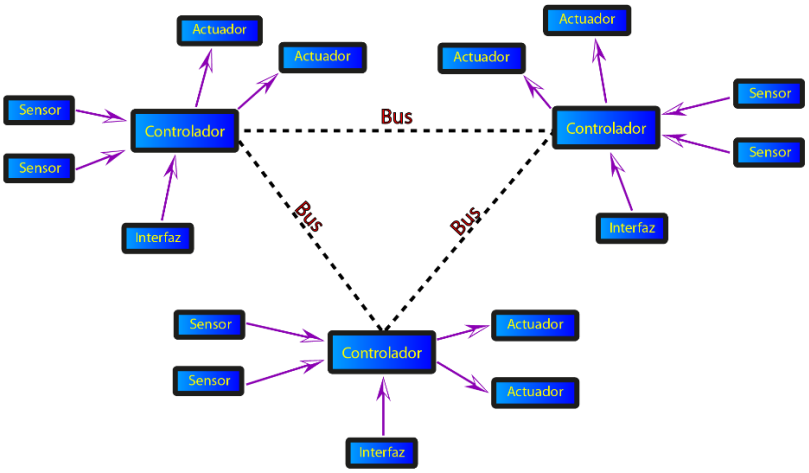


Imagen 3. Arquitectura descentralizada

4.2.2. Arquitectura centralizada

La arquitectura centralizada, es la que utilizaremos en el proyecto, y se basa en, que el controlador se dispone en el centro del sistema, ya que ejerce de intermediario entre la información que recibe de los sensores y la orden que envía a los actuadores, mediante las posibles interfaces, según el programa y la configuración establecida por el usuario. El cableado es en estrella cuyo centro es la unidad central de control y no existe comunicación entre sensores y actuadores.

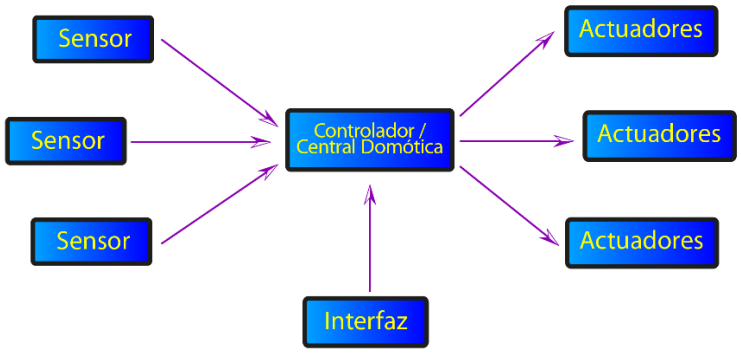


Imagen 4. Arquitectura centralizada

4.2.3. Arquitectura distribuida

El modelo de arquitectura distribuida, donde cada sensor y actuador ejerce también de controlador capaz de actuar y enviar información al sistema, e interactúa con los distintos elementos del sistema. Todos los elementos disponen de un acoplador al bus con una interfaz de acceso compartido y técnicas de direccionamiento para que la recepción y el envío de información quede definida y el dialogo entre elementos esté asegurado.

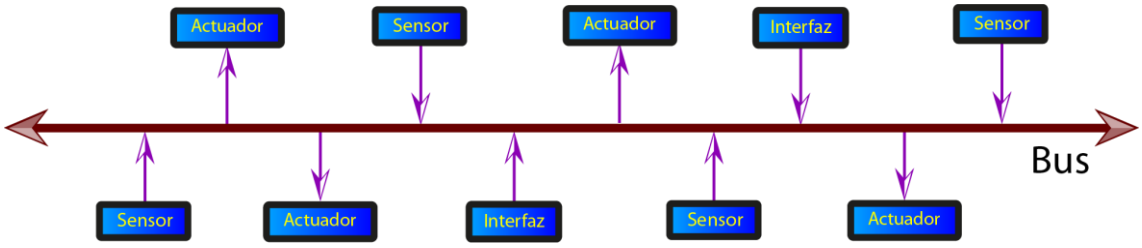


Imagen 5. Arquitectura distribuida

4.2.4. Arquitectura mixta

Y por último, en el modelo de arquitectura mixta se combinan las diversas arquitecturas de los sistemas centralizadas, descentralizadas y distribuidas. A la vez que puede disponer de un controlador central o varios controladores descentralizados, los dispositivos de interfaces, sensores y actuadores pueden también ejercer de controladores, como en un sistema distribuido.

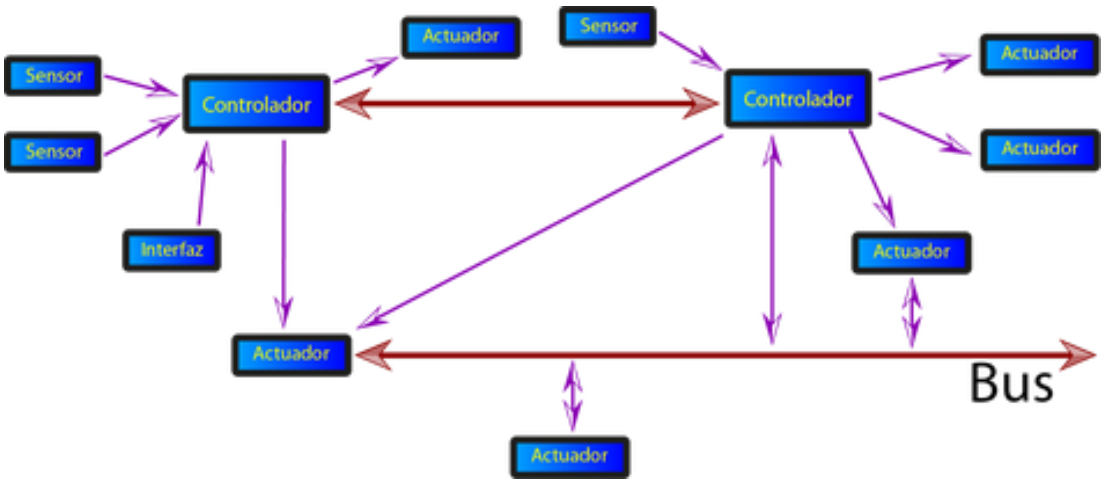


Imagen 6. Arquitectura mixta

5. HERRAMIENTAS DE TRABAJO (HARDWARE)

Antes de empezar a comentar cada herramienta de trabajo quiero explicar mi punto de vista tecnológico sobre todo lo fabricado en China. La gran mayoría de personas hemos tenido experiencias negativas a la hora de compra artículos donde leemos "Made in China", automáticamente, lo catalogábamos como un producto económico y de baja calidad. Pero creo que en la actualidad y tecnológicamente hablando, ha cambiado esta relación calidad/precio. Todos los fabricantes han facilitado a este país que crezca sin límite en el sector, estos se aprovechan del bajo coste de la mano de obra, pero olvidando o no queriendo ver que su producto queda expuesto a la copia. En mi vida laboral, trabaje en un proyecto en el cual tuve contacto con el representante de una empresa alemana, fabricante líder en tecnología de iluminación. Me comentó el por qué su empresa prefería fabricar exclusivamente en Alemania, todo y sabiendo que el coste es como mínimo cinco veces superior. Esta empresa ha invertido en investigación para mejorar su producto, un gasto muy elevado, pero necesario para poder tener en el mercado un producto competitivo. Su frase fue: "en el momento que accedes a fabricar allí, al día siguiente tienes una réplica exacta de tu producto". Compartió conmigo la experiencia de otra empresa en el sector, que les ha enseñado a que se enfrentan si deciden fabricar en China su producto, al principio todo es genial, el precio del producto disminuye considerablemente y te permite sacar al mercado un producto a un precio inmejorable. Pero como era de esperar al cabo de un tiempo fabricando allí, aparece un producto de una marca china con las mismas características y dimensiones, pero obviamente el color era distinto, seguramente debido a los materiales utilizados, y el precio mucho más económico que el que habías conseguido. Es obvio que esta empresa no ha tenido gastos de investigación, se han limitado a seguir el mismo proceso de fabricación con materiales de calidad inferior. Vamos un negocio redondo.

Donde quiero llegar con esto, fabricar en China ha brindado a este país, disponer de diversidad de tecnologías al alcance para poder crear a bajo coste sus propios productos. Por este motivo, como consumidores de tecnología nos planteamos, un móvil coreano cuesta 800€ y uno de una empresa china con experiencia en el sector 400€, con experiencia me refiero a un móvil 100% chino y que ha pasado los controles de calidad europeos. La cuestión es, si una empresa de móviles coreana fabrica en China, seguramente este móvil de la empresa china tendrá las mismas características pero con materiales de inferior calidad, pero la tecnología y el proceso de fabricación será el mismo.

Mi conclusión sobre el tema de comprar componentes procedentes de China es la siguiente. Si realmente precisas de un producto fiable, con garantía y la tecnología de este es relativamente nueva y exclusiva, evita comprar productos en dicho país. De lo contrario, si el componente se basa en una tecnología en la que se tienen años de experiencia, y la diferencia de precio es considerable, personalmente en la mayoría de casos me decanto por comprar en China y cada vez quedo más asombrado de la relación calidad/precio de los componentes.

Por ejemplo para realizar el proyecto he adquirido una pantalla táctil fabricada en china, porque creo que es una tecnología con muchos años de experiencia y su precio en Europa ronda los 100€ y en China tenemos una de las mismas características por 30€. En el caso de la Raspberry Pi 3, todo y que existen modelos chinos, he preferido adquirirla en una web española, que me aseguran que ha pasado un control de calidad y tienen una garantía. En cambio la placa D1 mini Pro, la encontramos en China por menos de 5€ y aquí encontramos una versión anterior por 25€, no queda otra opción que decantarte por la más económica.

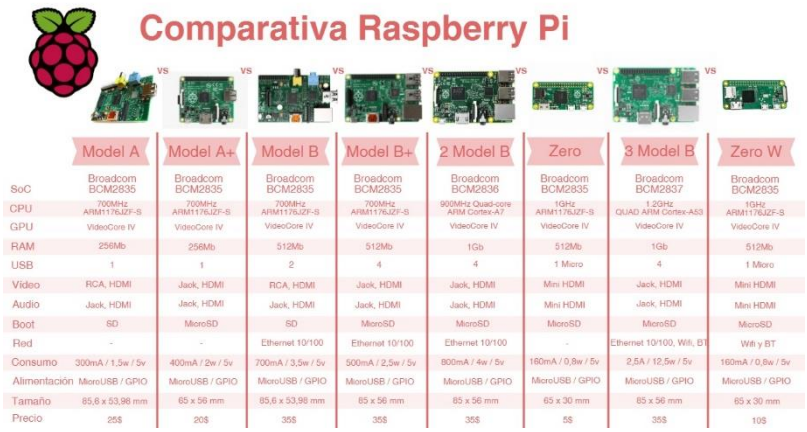
Para cumplir con la expectativa de bajo coste que he mencionado en los objetivos del proyecto, observaremos en los siguientes componentes que me he decantado por comprar productos procedentes de China.

5.1. Raspberry Pi 3

El primer prototipo se fabricó en el año 2012, pero no se llevó a cabo el proyecto Raspberry Pi hasta el 2006, cuando Eben Upton y algunos compañeros de trabajo de la Universidad de Cambridge decidieron solucionar un problema muy básico, la mayoría de los estudiantes tenían poco o ningún conocimiento de programación. Una parte del problema, pensó Upton, podría ser que ninguno de ellos poseía un PC realmente apropiado para programar y probar cosas nuevas, de forma que empezó con el diseño del micro-ordenador.

Para 2008 los procesadores se convirtieron en componentes suficientemente baratos y rápidos como para incluirlos en el aparato que tenía en mente. El Model A fue diseñado con unas características básicas pero a la vez muy completas, con salida de video y audio, un procesador capaz de realizar tareas interesantes como la reproducción de video de alta definición, un tamaño pequeño y un buen precio. Se estableció que el Model A se vendería por unos 25 dólares, pero la gente empezó a pedir conexión a internet, y por lo tanto la fundación Raspberry Pi decidió fabricar el Model B, con un puerto Ethernet y un precio de unos 35 dólares. Estas fueron los primeros modelos comercializables.

Actualmente nos encontramos con tres modelos importantes que son la RPI1, la RPI2 y la RPI3. Desde la primera versión hasta la tercera, encontramos cambio significativos en la CPU, la memoria RAM, número de puertos USB, comunicación WIFI, comunicación bluetooth, consumo, eficiencia, entre otras características que se pueden observar en la tabla 1. El precio siempre ha rondado los 20-35$. Se ha fabricado un modelo de menores dimensiones y precio, obviamente sus características son menos potentes que la RPI3.



	Model A	Model A+	Model B	Model B+	2 Model B	Zero	3 Model B	Zero W
SoC	Broadcom BCM2835	Broadcom BCM2835	Broadcom BCM2835	Broadcom BCM2835	Broadcom BCM2836	Broadcom BCM2835	Broadcom BCM2837	Broadcom BCM2835
CPU	700MHz ARM1176JZF-S	700MHz ARM1176JZF-S	700MHz ARM1176JZF-S	700MHz ARM1176JZF-S	900MHz Quad-core ARM Cortex-A7	1GHz ARM1176JZF-S	1.2GHz QUAD ARM Cortex-A53	1GHz ARM1176JZF-S
GPU	VideoCore IV	VideoCore IV	VideoCore IV	VideoCore IV	VideoCore IV	VideoCore IV	VideoCore IV	VideoCore IV
RAM	256Mb	256Mb	512Mb	512Mb	1Gb	512Mb	1Gb	512Mb
USB	1	1	2	4	4	1 Micro	4	1 Micro
Video	RCA, HDMI	Jack, HDMI	RCA, HDMI	Jack, HDMI	Jack, HDMI	Mini HDMI	Jack, HDMI	Mini HDMI
Audio	Jack, HDMI	Jack, HDMI	Jack, HDMI	Jack, HDMI	Jack, HDMI	Mini HDMI	Jack, HDMI	Mini HDMI
Boot	SD	MicroSD	SD	MicroSD	MicroSD	MicroSD	MicroSD	MicroSD
Red	-	-	Ethernet 10/100	Ethernet 10/100	Ethernet 10/100	-	Ethernet 10/100, Wifi, BT	Wifi y BT
Consumo	300mA / 1,5w / 5v	400mA / 2w / 5v	700mA / 3,5w / 5v	500mA / 2,5w / 5v	800mA / 4w / 5v	160mA / 0,8w / 5v	2,5A / 12,5w / 5v	160mA / 0,8w / 5v
Alimentación	MicroUSB / GPIO	MicroUSB / GPIO	MicroUSB / GPIO	MicroUSB / GPIO	MicroUSB / GPIO	MicroUSB / GPIO	MicroUSB / GPIO	MicroUSB / GPIO
Tamaño	85,6 x 53,98 mm	65 x 56 mm	85,6 x 53,98 mm	85 x 56 mm	85 x 56 mm	65 x 30 mm	85 x 56 mm	65 x 30 mm
Precio	25$	20$	35$	35$	35$	5$	35$	10$

Tabla 1. Comparativa de diferentes modelos de la Raspberry Pi

La aparición de este miniordenador ha sido un gran avance tecnológico para el mundo informático, no es necesario instalar un ordenador de sobremesa o un portátil para aplicaciones

que no precisan de tanta potencia. También ha estimulado a personas sin conocimientos de informática a iniciarse en el mundo de la programación con una gran variedad de proyectos que podemos encontrar en internet y crear nuestros proyectos propios.

En cuanto al software que utiliza la RPI es Open Source, siendo su sistema operativo una versión oficial, esta versión es adaptada de Debian (Linux), denominada RaspBian. Aunque permite otros sistemas operativos, incluyendo la versión de Windows 10 para desarrolladores, entre muchos otros sistemas que se instalan directamente en la microSD.

Para este proyecto se ha optado por utilizar el modelo B de la Raspberry Pi 3, con sistema operativo Raspbian. Se ha adquirido en Farnell una de las mayores tiendas web de componentes, todo y que existen otras como Mouser, RS Amidata y alguna más, siempre he confiado en esta, ya que, siempre ha cumplido con mis expectativas laborales.

Imagen 7. Raspberry Pi 3 Top

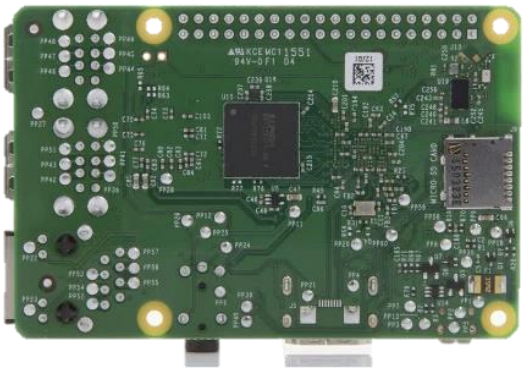


Imagen 8. Raspberry Pi 3 Bottom

Raspberry Pi 3 GPIO Header

Pin#	NAME		NAME	Pin#
01	3.3v DC Power		DC Power 5v	02
03	GPIO02 (SDA1 , I²C)		DC Power 5v	04
05	GPIO03 (SCL1 , I²C)		Ground	06
07	GPIO04 (GPIO_GCLK)		(TXD0) GPIO14	08
09	Ground		(RXD0) GPIO15	10
11	GPIO17 (GPIO_GEN0)		(GPIO_GEN1) GPIO18	12
13	GPIO27 (GPIO_GEN2)		Ground	14
15	GPIO22 (GPIO_GEN3)		(GPIO_GEN4) GPIO23	16
17	3.3v DC Power		(GPIO_GEN5) GPIO24	18
19	GPIO10 (SPI_MOSI)		Ground	20
21	GPIO09 (SPI_MISO)		(GPIO_GEN6) GPIO25	22
23	GPIO11 (SPI_CLK)		(SPI_CE0_N) GPIO08	24
25	Ground		(SPI_CE1_N) GPIO07	26
27	ID_SD (I²C ID EEPROM)		(I²C ID EEPROM) ID_SC	28
29	GPIO05		Ground	30
31	GPIO06		GPIO12	32
33	GPIO13		Ground	34
35	GPIO19		GPIO16	36
37	GPIO26		GPIO20	38
39	Ground		GPIO21	40

Imagen 9. Raspberry Pi 3 Pinout

Características de la Raspberry Pi 3:

- CPU ARMv8 de 4 núcleos a 1,2GHz y 64-bits.
- 1GB de RAM.
- Conectividad Bluetooth 4.1 y WIFI 802.11n.
- Salida HDMI.
- 4 puertos USB.
- Interfaz de entradas y salidas de propósito general GPIO de 40 PINS.
- Puerto Ethernet.
- Jack audio de 3,5 mm.
- Slot para tarjeta micro SD.
- VideoCore IV 3D graphics y una interfaz para cámara y pantalla externas.

5.2. D1 mini Pro WeMos

No cabe duda de que los chips ESP8266 han sido una auténtica revelación en el mercado de la comunicación Wireless WIFI.

Hace algo más de un año, conseguir que tu Arduino se conectara a una red WIFI, suponía gastarte algo más de 70€, además de tu Arduino.

Hasta que apareció el ESP8266 que empezaron a fabricarse multitud de modelos sencillos basados en él, de modo que conseguir una comunicación WIFI era sencillo y totalmente accesible para cualquiera comprando en China por poco menos de 5€, lo que ha sido toda una revolución sin duda.

La D1 mini Pro es una placa gobernada por un ESP8266 que se encarga tanto del procesamiento, como del control de la comunicación WIFI.

En cuanto a la programación, tenemos que agradecer el trabajo que realizo y compartió una persona anónima, estos drivers y librerías nos permiten programar en C con el IDE de Arduino, lo que facilita mucho su programación. Además esta placa soporta la programación OTA (Over The Air), es decir, sin hilos directamente a través del WIFI.

En el proyecto utilizaremos dos versiones la D1 mini y la versión pro, no existen grandes diferencias entre ambas, hablaremos de la versión pro debido a que es la más actual y la primera versión han dejado de fabricarla. En el mercado podemos encontrar varias placas con dicho chip, pero lo que me hizo decantarme a elegir esta, fue debido a que su distribución y composición se asemejaba a la famosa placa Arduino UNO, y su tamaño se reduce aproximadamente a la mitad.

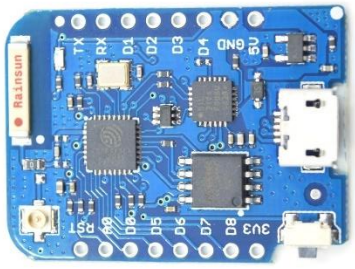


Imagen 10. D1 mini Pro WeMos Top

Imagen 11. D1 mini Pro WeMos Bottom

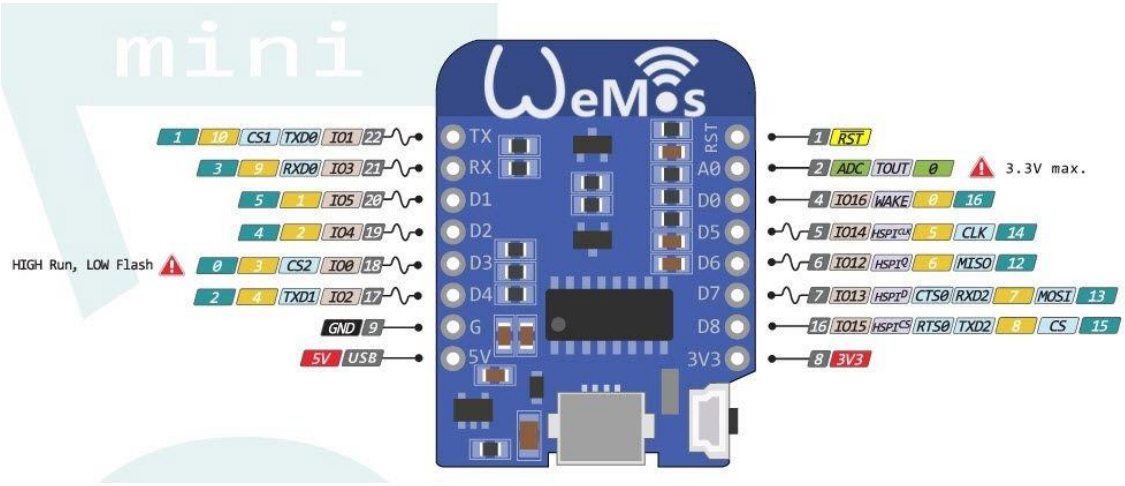


Imagen 12. Pin Out D1 mini Pro WeMos

Características de la D1 mini Pro WeMos:

- SOC (System On Chip o procesador central) de 32 bits.
- 11 entradas/salidas digitales.
- Interrupción/PWM/I2C/one-wire.
- 1 entrada analógica (3,2V entrada máxima).
- Memoria Flash de 16 Mbytes (128Mbits).
- Antena cerámica y conector para antena externa.
- Conector microUSB (Alimentación 5V)

5.3. Fuente de alimentación (TSP-05)

Fuente de alimentación integrada TSP-05, es capaz de convertir los 220V de corriente alterna de nuestra red en los 5V de corriente continua necesarios para la alimentación de nuestra electrónica.

Por menos de 2€ tenemos resuelto el problema que muchas veces nos encontramos, alimentar nuestros sistemas electrónicos con la energía de la red eléctrica, y con un tamaño reducido.

Imagen 13. Fuente alimentación integrada 220VAC a 5VDC

Características de la fuente de alimentación TSP-05:

- Regulador de voltaje.
- Entrada 110 – 220 VAC.
- Salida 5 VDC, 3Watt.
- Bajo rizado y ruido.
- Peso 20 gramos.

5.4. Relé electromecánico para D1 mini pro WeMos

Los relés electromecánicos se han utilizado desde hace décadas, pero como veremos en el siguiente punto los relés de estado sólido están dejando obsoleta esta tecnología.

Para cargas superiores a 2A utilizaremos esta "Shield"[2] que nos permite una carga máxima de 10A, como gran desventaja es que estos generan ruido cuando se activan.

El precio unitario es de menos de 2€, encaja perfectamente como bajo coste porque el sistema de D1 mini pro, con la "Shield" de relé y la fuente de alimentación sale por menos de 10€.

Imagen 14. Relé electromecánico "Shield"

Imagen 15. Conexión D1 mini pro y Relé "Shield"

Características del Relé electromecánico para D1 mini pro WeMos:

- Capacidad de conmutación para 10A en un diseño de tamaño pequeño para técnicas de montaje en PCB's de alta densidad.
- Selección de material plástico para alta temperatura y una mejor disipación térmica.
- Indicador led con activación.
- Voltaje nominal: 5V y corriente nominal: 70 a 90mA.
- Capacidad para carga: 28VDC 7A, 125VAC 10A, 240VAC 10A.

[2] Son PCB's que pueden ser conectadas encima de una placa entrenadora extendiendo sus capacidades.

5.5. Relés electromecánicos de cuatro canales para D1 mini pro WeMos

Para aprovechar al máximo nuestra D1 mini pro, utilizaremos esta placa de cuatro canales de relés electromecánicos. Tendremos cuatro salidas para controlar cuatro dispositivos independientes, como puede ser el control de persianas, el termostato y un sistema de iluminación.

Las características de estos relés electromecánicos son idénticas a las del relé comentado en el apartado anterior. Su precio es de aproximadamente 4€.

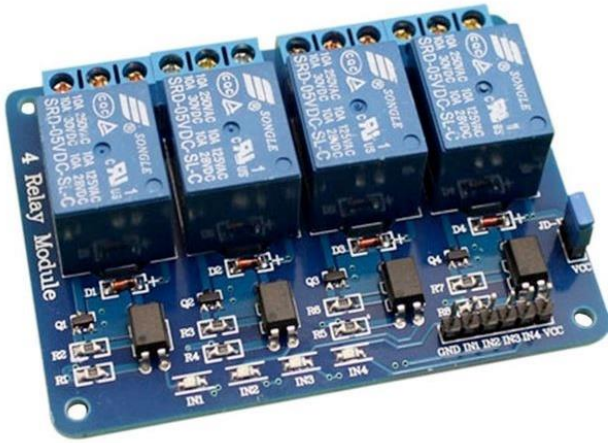


Imagen 16. Relés electromecánicos de cuatro canales

Características de los Relés electromecánicos:

- Capacidad de conmutación para 10A en un diseño de tamaño pequeño para técnicas de montaje en PCB's de alta densidad.
- Selección de material plástico para alta temperatura y una mejor disipación térmica.
- Indicador led con activación.
- Voltaje nominal: 5V y corriente nominal: 70 a 90mA.
- Capacidad para carga: 28VDC 7A, 125VAC 10A, 240VAC 10A.

5.6. Relé de estado sólido (G3MB-202P)

Poco a poco los relés de estado sólido están desterrando a los relés electromecánicos, debido a las ventajas que nos ofrece ante las otras tecnologías.

Internamente se componen por un optoacoplador conectado a un TRIAC, que cuando aplicamos 5V en los terminales de entrada, enciende un led y este activa el TRIAC a la salida.

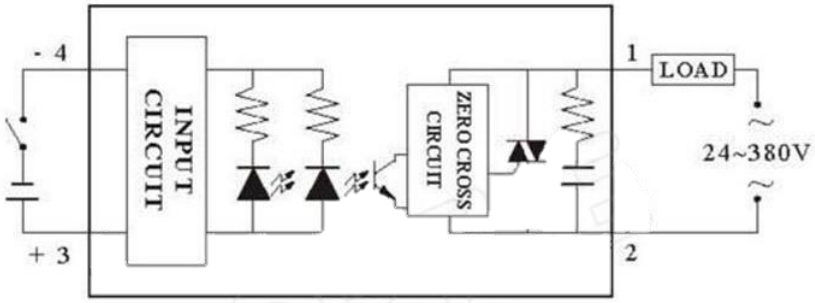


Imagen 17. Esquema interno de un relé de estado sólido

Ventajas del Relé de estado sólido sobre el electromecánico:

- Menor tamaño, permitiendo elementos más compactos y automatizables.
- Menor tensión de trabajo, se activan desde 1,5V o menos.
- Funcionamiento totalmente silencioso.
- Los relés de estado sólido son más rápidos que los relés electromecánicos, su tiempo de conmutación depende del tiempo requerido para encender el LED de control, del orden de microsegundos a milisegundos.
- Vida útil más larga, incluso si se activa muchas veces, ya que no hay partes mecánicas que se desgasten o contactos que se deterioren a altos amperajes.
- La resistencia de salida se mantiene constante independientemente del uso.
- Limpieza de conexión, no hay rebote en la conmutación de los contactos.
- Sin chispas, no se producen arcos eléctricos, lo que permite ser usados en ambientes explosivos donde es crítico que no se produzcan chispas en la conexión.
- Mucho menos sensible al almacenaje y ambiente operativo, como los golpes, vibraciones, humedad, y campos magnéticos externos.
- No produce ondas electromagnéticas que puedan producir interferencias en otros equipos.

Desventajas del Relé de estado sólido sobre el electromecánico:

- Con circuito cerrado, mayor resistencia (pérdidas en forma de calor).
- En abierto, menor resistencia, con una pequeña corriente inversa de pérdida.
- Las propiedades voltaje/corriente no son lineales (no puramente resistivas), distorsionando las alternas conmutadas hasta cierto punto. Un relé electromecánico tiene baja resistencia óhmica (lineal) del interruptor mecánico asociado cuando se activa, y una enorme resistencia de la separación de aire y las partes aislantes cuando está abierto.
- La polaridad de la salida afecta a algunos tipos de relés de estado sólido, a los mecánicos no les afecta.
- Posibilidad de falsas conmutaciones debido a cargas transitorias, debido a la capacidad de conmutación mucho más rápida.
- Se requiere una alimentación aislada para el circuito de la puerta de activación.
- Mayor tiempo de recuperación de la corriente inversa transitoria debido a la presencia del cuerpo del diodo.
- Tienen tendencia a quedar en circuito cerrado cuando fallan, mientras que los mecánicos tienden a quedar en abierto, que suele ser preferible.

Para el proyecto he elegido un relé de estado sólido con una carga máxima de 2A a la salida, potencia razonable para un sistema de iluminación. El precio de este es de menos de 1€ comprando 10.

Imagen 18. Relé de estado sólido G3MB-202P

Características del relé de estado sólido G3MB-202P:

- Tensión de control 5V.
- Menor ruido y mayor efectividad.
- Corriente de carga 2A.
- Tensión de carga 240V.
- Dimensiones 50 x 50 x 5mm y 15 gramos de peso.

5.7. Kit de desarrollo EnOcean (EDK 350)

Se trata de un kit bastante completo con una placa entrenadora para entender, comprobar y programar los periféricos de la marca EnOcean basados en energy harvesting[3].

EnOcean ha diseñado periféricos que se comunican mediantes señales de radiofrecuencia de 868.3MHz (Estándar Europeo). Estos no precisan de baterías, ni alimentación externa.

El coste del kit es de un poco menos de 200€, es relativamente elevado, pero hay que remarcar que las posibilidades de este dispositivo son muy interesantes y el precio del pulsador completo es de aproximadamente 30€. El pulsador es autónomo, tiene una gran cobertura de emisión y puede instalarse donde deseemos sin necesidad de cablear y realizar una obra en nuestro hogar.

En el kit consta de:

- EOP 350 Programador/placa entrenadora.
- USB 300 módulo receptor USB.
- TCM 320 módulo de transmisión.
- STM 300 módulo sensor con adaptador.
- PTM 21x pulsador transmisor.
- ECO 200 generador de energía mecánico.
- PTM 330 modulo transmisor.
- Software DolphinStudio y DolphinView.

Imagen 19. Kit de desarrollo EnOcean

[3] Proceso a través del cual se obtiene energía que proviene de fuentes externas naturales. Energía limpia y gratuita que puede substituir las baterías.

Daremos énfasis a los dos periféricos que hemos utilizado en el proyecto, se trata del USB300 módulo receptor USB y el PTM210 pulsador transmisor.

Módulo receptor USB (USB300)

El módulo receptor se conecta a ordenadores por el puerto USB, se encarga de recibir y traducir los datos que llegan por radiofrecuencia de los periféricos, para que puedan ser procesados por el ordenador. El módulo receptor de radiofrecuencia está formado por el TCM300, y es bidireccional.

EnOcean ha creado su propio protocolo de comunicación, llamado EnOcean Serial Protocol 3.

Imagen 20. Módulo transmisor (TCM300)

Imagen 21. Módulo receptor USB300

Pulsador transmisor (PTM210)

El pulsador está diseñado para identificar cuatro pulsaciones, y puede diferenciar cuando oprimimos y soltamos el pulsador, es decir, tendremos diferenciadas ocho señales.

Está formado por dos sistemas conectados entre sí, uno se trata del generador inductivo ECO200, encargado de proporcionar la energía necesaria en cada pulsación para enviar el dato al receptor. El segundo sistema es el PTM330, se encarga de enviar el id del pulsador, tipo de dato y que pulsador ha sido presionado.

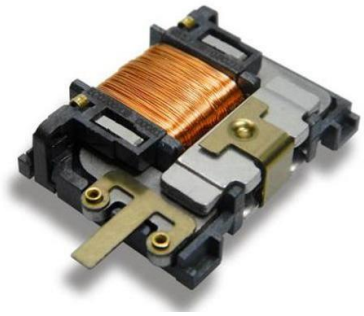


Imagen 22. Generador inductivo (ECO200)

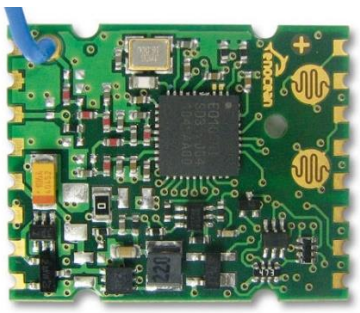


Imagen 23. Sistema transmisión (PTM330)

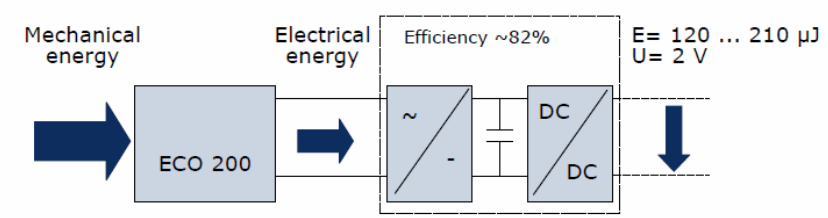


Imagen 24. Esquema eléctrico de la generación y gestión de la energía inductiva

Estos dos sistemas conforman el pulsador TCM210 y sus características son:

- Fuente de alimentación autónoma mediante generador inductivo.
- Antena cerámica integrada.
- Frecuencia de funcionamiento 868.300 MHz (Estándar Europeo).
- Velocidad de transmisión de datos 125Kbps, dos canales con cuatro estados de acción cada uno.
- Rango de transmisión 300m en campo libre y 30m en interior.
- Tamaño 40 x 40 x 11.2mm.

En la imagen 14 podemos observar que aparentemente no se aprecia ninguna diferencia respecto a un pulsador convencional cableado, lo que facilita la substitución o una nueva instalación en cajetín universal europeo.

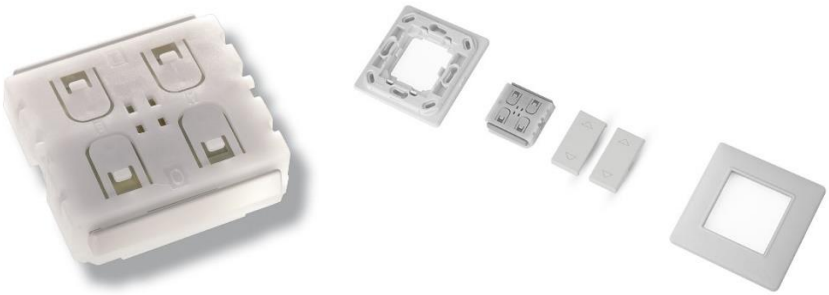


Imagen 25. Pulsador EnOcean TCM210 y sistema completo

Imagen 26. Doble pulsador y pulsador simple para TCM210

5.8. Pantalla táctil

Para controlar nuestra central de procesamiento es indispensable utilizar una interfaz de usuario (HMI[4]). La pantalla táctil nos permitirá comprobar el estado y realizar cambios en nuestro sistema, y configurar nuestros periféricos.

Actualmente podemos encontrar gran variedad de pantallas para la RPI3 en el mercado, la gran mayoría se conectan mediante los pines de propósito general de entradas y salidas (GPIO), pero existiendo un puerto HDMI, porque no utilizarlo con una pantalla de mayores dimensiones. He buscado diferentes fabricantes para encontrar una pantalla que sea de calidad y algo importante en el transcurso de todo el proyecto, el precio.

Tras una exhaustiva búsqueda me he decantado por una pantalla táctil de 7" (AT070TN90) del fabricante chino Innolux, este producto tiene buenas referencias y el precio es de 30€, al que hay que incluir la carcasa de protección que cuesta unos 10€, por 40€ tenemos una pantalla táctil de 7". Se trata de una pantalla táctil con salida VGA y HDMI, con la alimentación independiente de su sistema de control de imagen, de esta forma no sobrecargamos la alimentación de la RPI3. Las dimensiones son 155 x 86mm, tamaño más que aceptable para nuestro sistema.

Imagen 27. Componentes que conforman la pantalla táctil AT070TN90

[4] Interfaz de usuario por sus siglas en inglés, (Human Machine Interface) que se utiliza para referirse a la interacción entre humanos y máquinas. Aplicable a sistemas de Automatización de procesos.

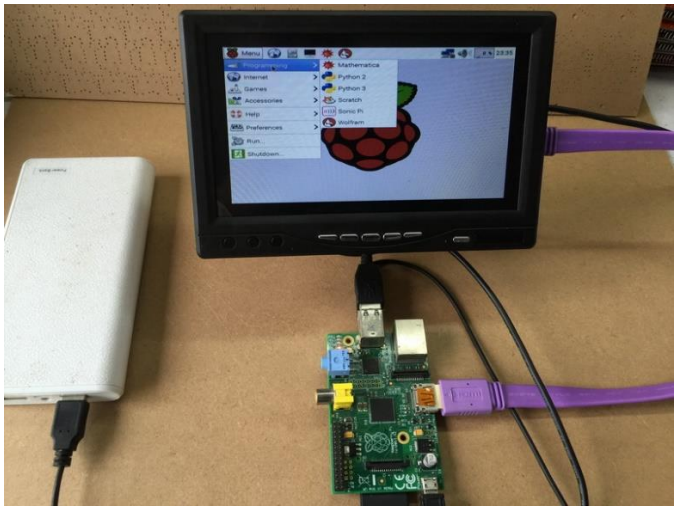


Imagen 28. Pantalla completa en funcionamiento con RPI

Características de la pantalla táctil:

- Tamaño LCD 7 pulgadas diagonal (154.08 x 85.92mm).
- Driver a-Si TFT active matrix.
- Resolución de 800 x 3(RGB) x 480.
- Tratamiento de la superficie anti reflejante.
- Interfaz digital, paralela 8bit RGB.
- Alimentación en 6.5 y 13.5V.
- El rango de funcionamiento de temperatura entre -20 y 70°C.
- Consumo de 0.226W.
- 45 gramos de peso.

5.9. Sensor de temperatura y humedad DHT22

El sensor de temperatura y humedad DHT22 es el predecesor del sensor de temperatura y humedad DHT11. El DHT22 supone una mejora considerable en las características técnicas con mayores rangos de temperatura y humedad, y más precisión.

Este sensor 2 en 1 lo podemos encontrar por aproximadamente 5€, años atrás era imposible encontrar un sensor de temperatura o de humedad por este precio y con esas características, nuevamente el mercado chino ha conseguido unirlos y venderlo a un precio muy competitivo.

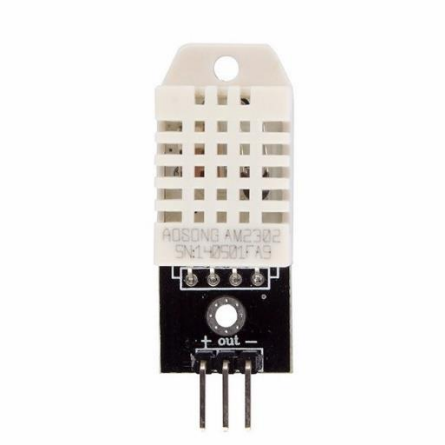


Imagen 29. Sensor de temperatura y humedad DHT22

Características del sensor de temperatura y humedad DHT22:

- Alimentación: 3.3Vdc ≤ Vcc ≤ 6Vdc.
- Rango de medición de temperatura: -40°C a 80 °C.
- Precisión de medición de temperatura: <±0.5 °C.
- Resolución Temperatura: 0.1°C.
- Rango de medición de humedad: De 0 a 100% RH.
- Precisión de medición de humedad: 2% RH.
- Resolución Humedad: 0.1%RH.
- Tiempo de sensado: 2s.

5.10. Sensor de monóxido de carbono y combustión MQ-7

El sensor MQ-7 es un sensor para la detección de monóxido de carbono (CO) para medir la concentración de este gas en el aire. Es capaz de medir concentraciones de gas monóxido entre 20 a 2000 partes por millón (ppm)

El sensor MQ7 posee una alta sensibilidad y rápido tiempo de respuesta, es muy fácil además realizar la interfaz del sensor con un microcontrolador, ya que podemos usar un pin de entrada analógico para medir la concentración del gas. Las conexiones que requiere el sensor son muy básicas solo requiere alimentación de 5V para el elemento calefactor

He adquirido un módulo que contiene toda la electrónica básica necesaria para el funcionamiento del sensor, entregando una señal analógica y digital. La sensibilidad de la señal digital puede ajustarse mediante un potenciómetro colocado en el módulo. Su coste es de menos de 2€.

Imagen 30. Sensor de monóxido de carbono Top

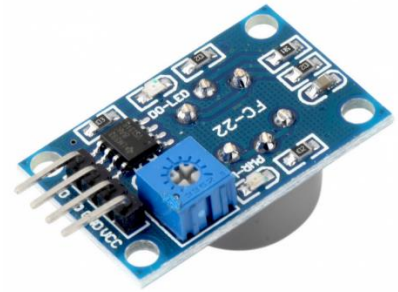


Imagen 31.Sensor de monóxido de carbono Bottom

Características del MQ-7 Sensor de Monóxido de Carbono

- Módulo que incluye el sistema mínimo necesario para el funcionamiento del sensor MQ-7.
- Basado en el sensor LM393.
- Salida analógica y digital, esta última funciona mediante un comprador analógico.
- Voltaje de alimentación 5 voltios.
- Detección de concentración de gas monóxido, ideal para alarmas de gas y aplicaciones industriales.
- Ajuste de umbral para salida digital mediante potenciómetro.

6. HERRAMIENTAS DE TRABAJO (SOFTWARE)

6.1. VNC

Las siglas VNC provienen del inglés de *Virtual Network Computing* (Computación Virtual en Red).

El programa VNC es software libre basado en una estructura cliente-servidor que permite tomar el control del ordenador servidor remotamente a través de un ordenador cliente. También se denomina software de escritorio remoto. VNC no impone restricciones en el sistema operativo del ordenador servidor con respecto al del cliente: es posible compartir la pantalla de una máquina con cualquier sistema operativo que admita VNC conectándose desde otro ordenador o dispositivo que disponga de un cliente VNC portado.

La versión original del VNC se desarrolló en el Reino Unido, concretamente en los laboratorios AT&T Olivetti Research Laboratory, en Cambridge. El programa era de código abierto, por lo que cualquiera podía modificarlo, y existen hoy en día varios programas para el mismo uso.

El sistema operativo Raspbian lleva instalado por defecto el VNC para que podamos acceder desde un dispositivo remotamente.

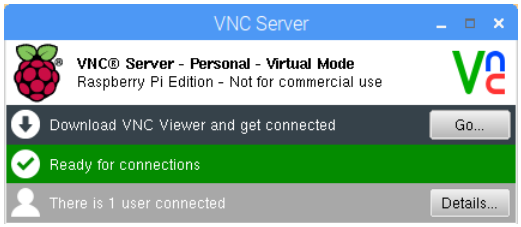


Imagen 32. VNC Server para RPI

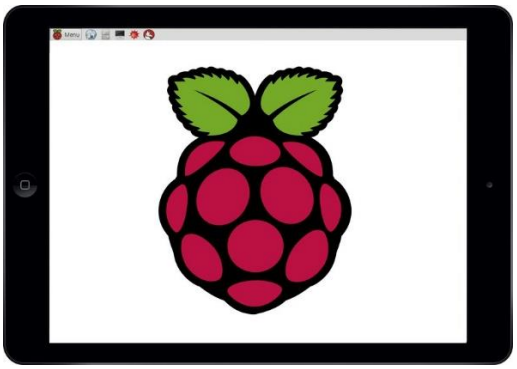


Imagen 33. Conexión remota con la RPI con Real VNC

Características de Real VNC:

- Controlar equipos remotamente.
- Proteger conexión con contraseña.
- Ofrecer soporte remoto.
- Transferir archivos.
- Imprimir archivos remotamente.

6.2. IDE PyCharm

PyCharm es un entorno de desarrollo integrado (IDE) utilizado en la programación de computadoras, específicamente para el lenguaje Python. Es desarrollado por la compañía checa JetBrains. Proporciona análisis de código, un depurador gráfico, un probador de unidad integrado, integración con sistemas de control de versiones.

La RPI3 nos permite programar en varios lenguajes de programación, incluso en C, que es un lenguaje que he utilizado en varios proyectos, pero me atreví a utilizar el lenguaje Python del que había escuchado que era sencillo y para la aplicación que quería dar a la RPI se adaptaba mejor.

Actualmente existen varias versiones de Python, actualmente la más nueva es la versión 3.x que empieza a utilizar-se, pero como todo entorno al que te habitúas, es costoso dar el salto. Y es por esto que actualmente muchos programadores prefieren continuar con la versión 2.7 y continuar con los programas que se han desarrollado, sin tener que actualizar librerías y encontrarte como es mi caso con incompatibilidades entre versiones. Por este motivo he decidido, todo y probar la versión 3.5, utilizar la versión 2.7 en mi desarrollo.

Cuando empecé a programar los primeros ejemplos en Python, utilizaba el terminal de Rasbian pero una vez me introducía en el lenguaje, y con la experiencia en programación en otros lenguajes, sabía que debía buscar un entorno de desarrollo. Investigue sobre cuál era el entorno que me facilitara en muchos casos que errores estaba cometiendo, y tras probar algunos me decante por el PyCharm, debido a que muchos tutoriales utilizaban este entorno y existe una comunidad que facilita las inquietudes que te puedan surgir. En cuanto al entorno es completo, agradable y hace que puedas ejecutar o depurar el programa de forma sencilla.

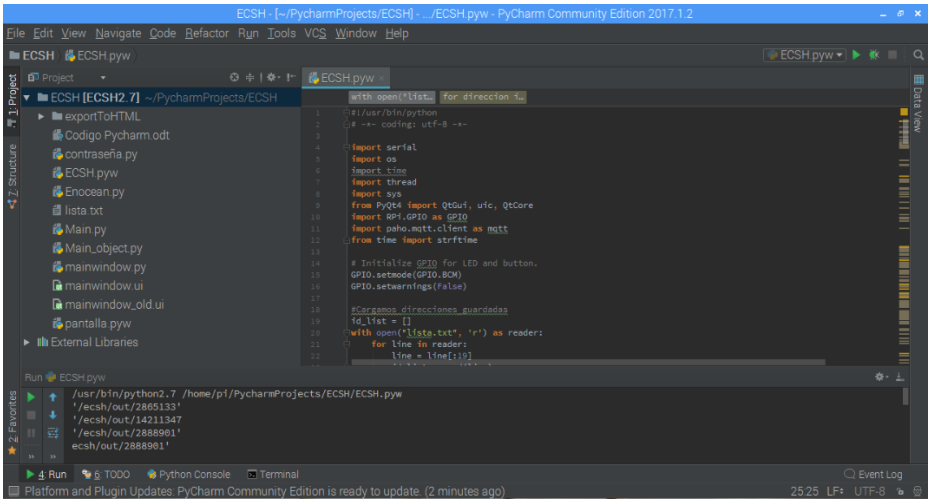


Imagen 34. Entorno de programación PyCharm

6.3. Qt Creator (Interfaz gráfica)

Qt Creator es una herramienta de desarrollo que nos permite crear interfaces graficas de usuario. Nos proporciona un conjunto de componentes estándar y un mecanismo de conexión llamado signal-slot, con el que conectaremos eventos de la interfaz con la lógica de programa que han de soportar.

Las posibilidades que disponemos con esta herramienta son amplias, pero existen dos posibilidades, programar todo sobre este programa o utilizarlo para posicionar todos los elementos gráficos y después programar todo el código con otro IDE. Debido a que debía programar toda la parte de comunicación WIFI y radiofrecuencia, he optado como utilizarlo solamente para posicionar cada objeto de la interfaz gráfica, y no utilizar la herramienta signal-slot.

Con esta herramienta generaremos una ventana con todos nuestros objetos posicionados, y seguidamente en el programa Python que crearemos con el entorno PyCharm programaremos las acciones y modificaremos sus estados. Qt Creator, nos creara un archivo tipo mainwindow.ui, el cual ejecutaremos en nuestro programa Python.

Imagen 35. Entorno programación interfaz gráfica Qt Creator

6.4. IDE Arduino

En primer lugar, explicaremos en que se basa Arduino, es una plataforma de electrónica abierta para la creación de prototipos basada en software y hardware libre, flexibles y fáciles de usar. Se creó para artistas, diseñadores, aficionados y cualquier interesado en crear entornos u objetos interactivos.

El hardware Arduino más sencillo consiste en una placa con un microcontrolador y una serie de puertos de entrada y salida**.** Los microcontroladores AVR más usados son el Atmega168, Atmega328, Atmega1280, y Atmega8 por su sencillez y bajo coste que permiten el desarrollo de múltiples diseños, aunque también nos encontramos microcontroladores CortexM3 de ARM de 32 bits, que coexistirán con las más limitadas, pero también económicas AVR de 8 bits. ARM y AVR son plataformas diferentes, pero gracias al IDE de Arduino los programas se compilan y luego se ejecutan sin cambios en cualquiera de las plataformas.

Imagen 36. Placa Arduinio Mega

La diferencia entre los distintos Arduino la encontraremos por un lado en la tensión utilizada en las placas. Generalmente las microcontroladoras con CortexM3 tienen un voltaje de 3,3 voltios, mientras que la mayor parte de las placas con AVR utilizan una tensión de 5 voltios. Esto luego es fundamental para utilizar lógica TTL (frente a lógica CMOS) por ejemplo, lo que abre la posibilidad de utilizar chips baratos y complementar el Arduino con alguna funcionalidad externa. También hay placas que pueden conmutar el voltaje, así que tampoco es un factor determinante para seleccionar una placa u otra. Y, por otra parte, el número de conexiones, procesador utilizado, memoria y, sobre todo, el número de entradas y salidas y la posibilidad de alimentar distintos elementos desde la propia placa Arduino.

Arduino puede tomar información del entorno a través de sus pines de entrada, para esto toda una gama de sensores puede ser usada y puede afectar aquello que le rodea controlando luces, motores y otros actuadores. El microcontrolador en la placa Arduino se programa mediante el lenguaje de programación Arduino (basado en Wiring) y el entorno de desarrollo Arduino (basado en Processing). Los proyectos hechos con Arduino pueden ejecutarse sin necesidad de conectarlo a un ordenador, si bien tienen la posibilidad de hacerlo y comunicar con diferentes tipos de software (p.ej. Flash, Processing, MaxMSP).

El IDE de Arduino es un entorno de desarrollo basado en java, multiplataforma.

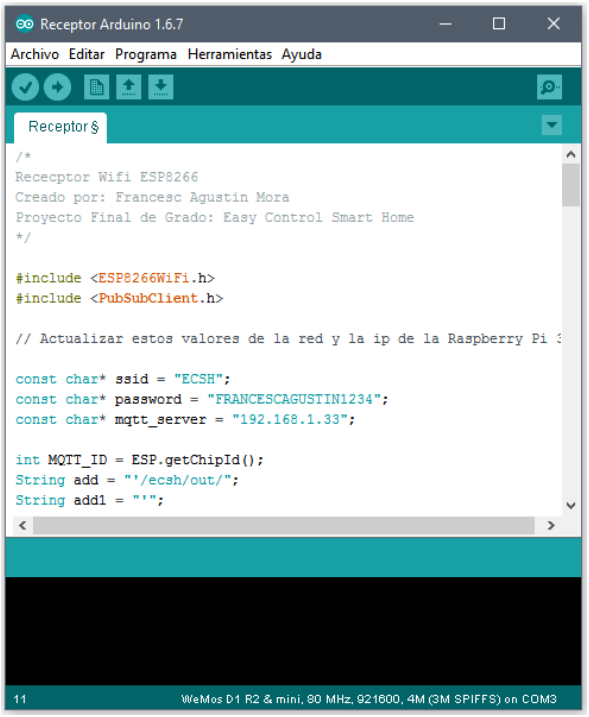


Imagen 37. IDE Arduino

Es una IDE bastante sencilla sin demasiadas opciones pero cumple con su cometido. El lenguaje de programación es sencillo, sobre todo si ya tienes experiencia en otros lenguajes de programación como C o Java ya que Wiring / Processing para su programación se basa en ellos. El microcontrolador en la placa Arduino se programa mediante el lenguaje de programación Arduino (basado en Wiring) y el entorno de desarrollo Arduino (basado en Processing).

6.5. Eagle

EAGLE, (siglas de **E**asily **A**pplicable **G**raphical **L**ayout **E**ditor) es un programa de diseño de diagramas y PCBs con autoenrutador. Famoso alrededor del mundo de los proyectos electrónicos, debido a que muchas versiones de este programa tienen una licencia Freeware y gran cantidad de bibliotecas de componentes alrededor de la red.

Existen infinidad de programas de diseño, entre ellos Altium Designer, por mi experiencia el más potente en muchos sentidos, pero con el inconveniente de no ser gratuito y el precio de su licencia es elevado.

Siempre he utilizado Altium para PCBs complejas, donde puedes aprovechar al máximo todas las posibilidades que este posee, pero para crear PCBs sencillas me acostumbre a utilizar Eagle y la licencia es gratuita.

Como la gran mayoría de programas de diseño, disponemos de tres entornos.

En primer lugar debemos crear o encontrar nuestros componentes, cada fabricante de componentes electrónicos suele facilitar las librerías necesarias para crear nuestro diagrama, pero en muchos casos se utiliza el editor de librerías, para crear nuestros componentes para su utilización.

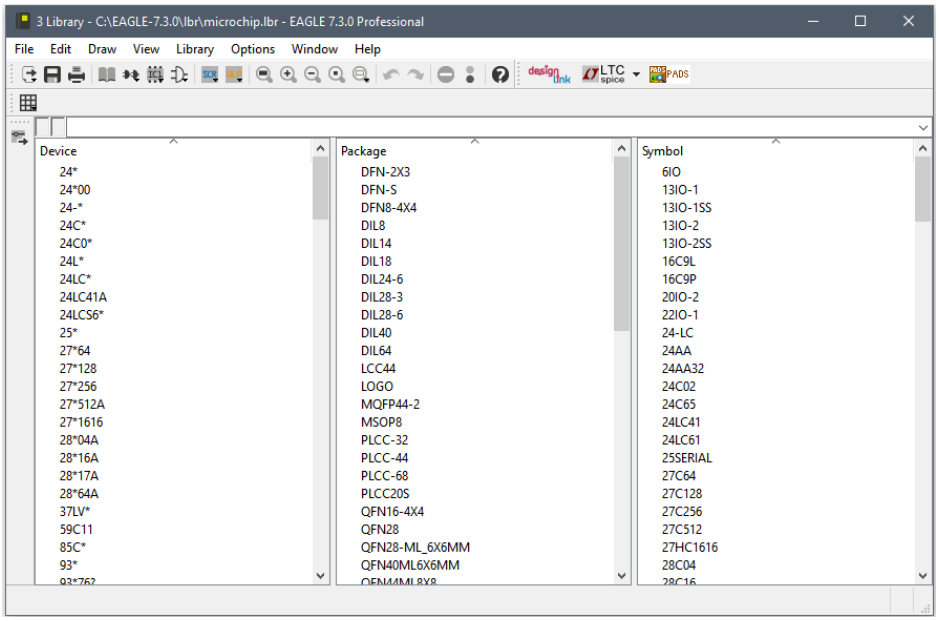


Imagen 38. Librería Eagle

Una vez tenemos creados y/o localizados los componentes que vamos a utilizar, con el editor de diagramas electrónicos (Layout o Esquemático), colocaremos los componentes en el diagrama con un solo click y fácilmente enrutables con otros componentes a base de "cables" o etiquetas. De esta forma crearemos un circuito electrónico donde podemos observar fácilmente su funcionamiento.

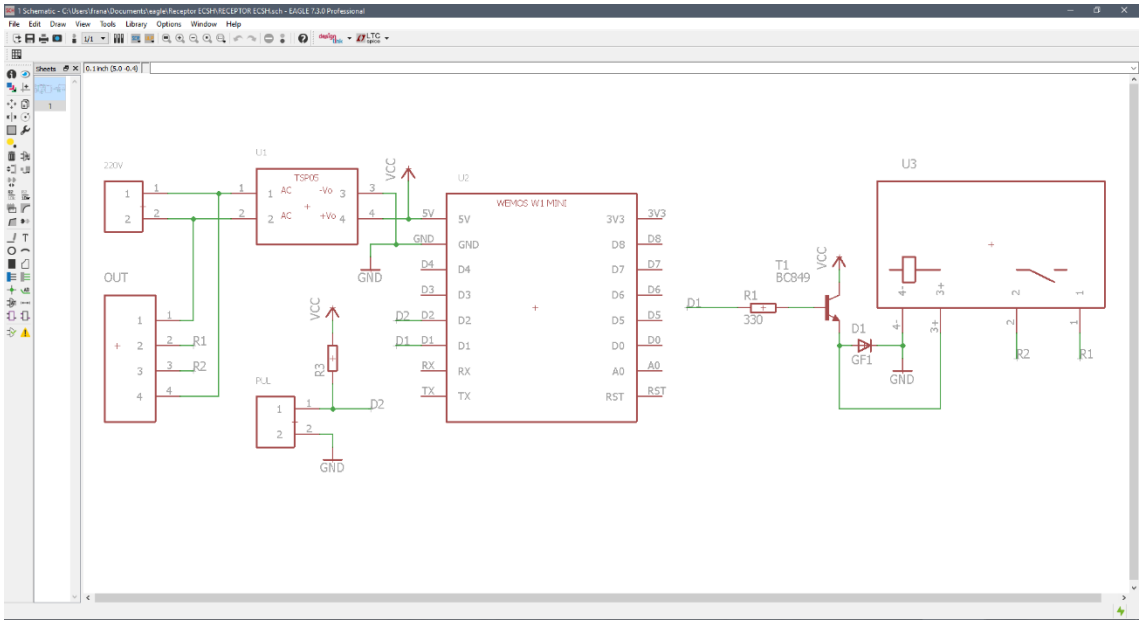


Imagen 39. Editor de diagramas electrónicos de Eagle

Cuando tengamos diseñado nuestro diagrama electrónico, crearemos la PCB final, donde las conexiones serán las mismas que tenemos en nuestro diagrama anterior pero con las conexiones sobre los componentes reales (componente físico). La gran mayoría de programas de diseño disponen de la utilidad autoenrutador (autoroute) bastante eficiente en algunos casos, pero la gran mayoría de compañeros y comparto esta opinión, no es un gran aliado en muchos casos, ya que, normalmente genera más problemas, que el tiempo que te puede ahorrar. Se debe a que cada circuito es diferente, las directrices, normas o requerimientos que hay que indicar para el autoenrutador son difíciles de controlar. Por ese motivo, mucha gente rechaza esta herramienta y realiza las conexiones manualmente.

Para este proyecto hemos utilizado doble capa, como observamos en la Imagen 40. Editor del diseño de la PCB de Eagle, Top (Capa superior en color rojo) y la Bottom (Capa inferior en color azul). En este editor fijaremos las dimensiones de nuestra PCB y crearemos taladros para su fijación.

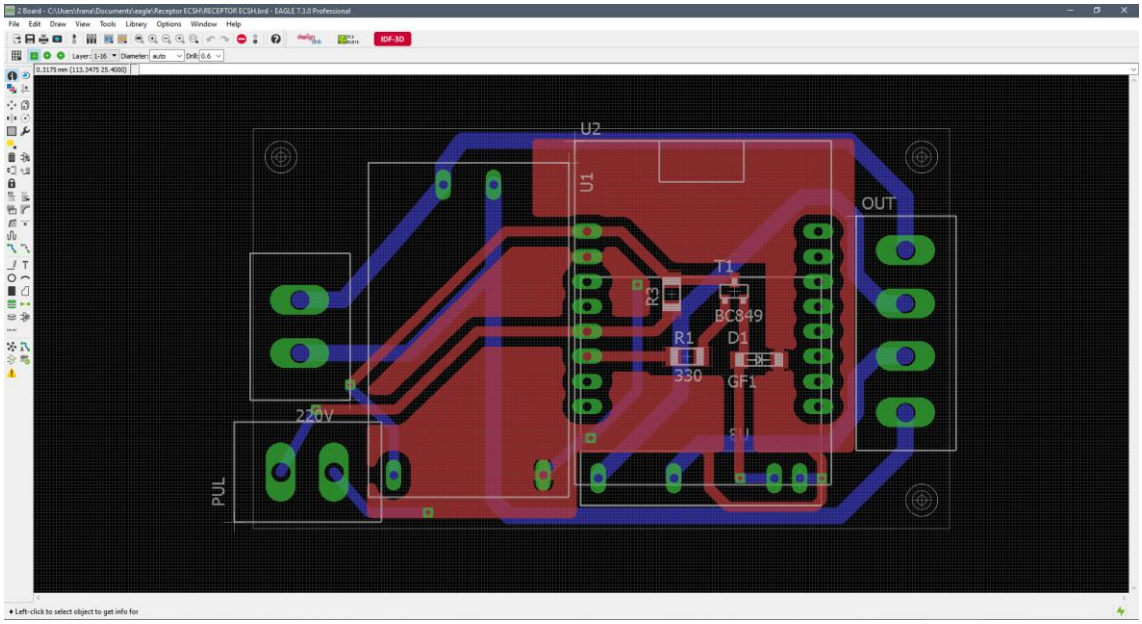


Imagen 40. Editor del diseño de la PCB de Eagle

El último paso, es crear los archivos para que el programa que se encarga de la producción de la PCB entienda que debe realizar, se trata de los gerbers.

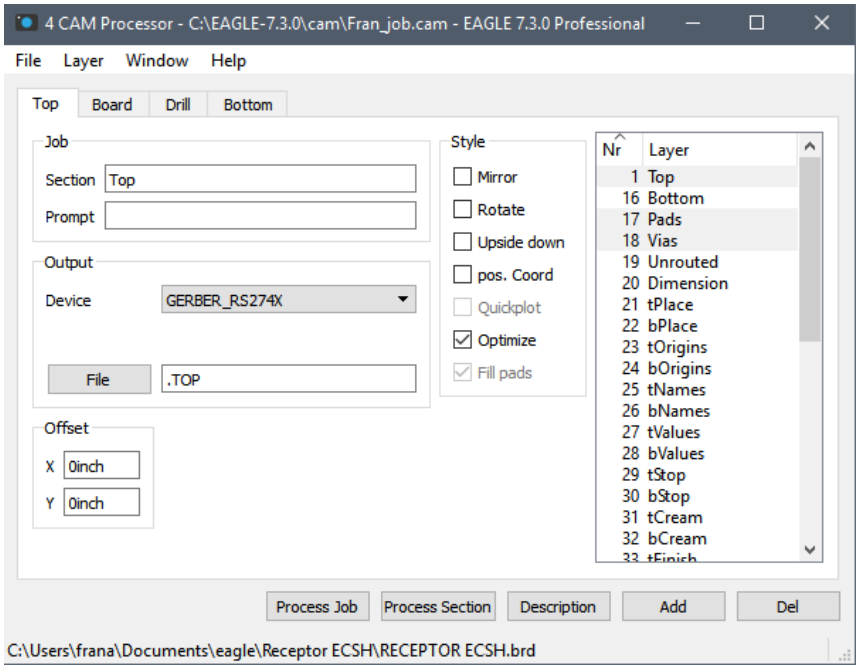


Imagen 41. Configuración de los Gerbers de Eagle

7. DESCRIPCIÓN GENERAL

El sistema de control consta de los siguientes componentes:

7.1. Sistema de comunicación, Router WIFI.

7.2. Central de procesamiento se utilizará una Raspberry Pi 3 (RPI3) y un receptor RF

7.3. Periféricos de entrada:

- Interruptores EnOcean RF[5] (energy harvesting).
- Cámara de seguridad.

7.4. Periféricos entrada y salida:

- Pantalla táctil de 7".
- Smartphone o Tablet.
- Receptor WIFI con salida a 220V.
- Receptor WIFI con sensor de temperatura, humedad y de monóxido de carbono (CO).
- Receptor WIFI con sensor de movimiento.

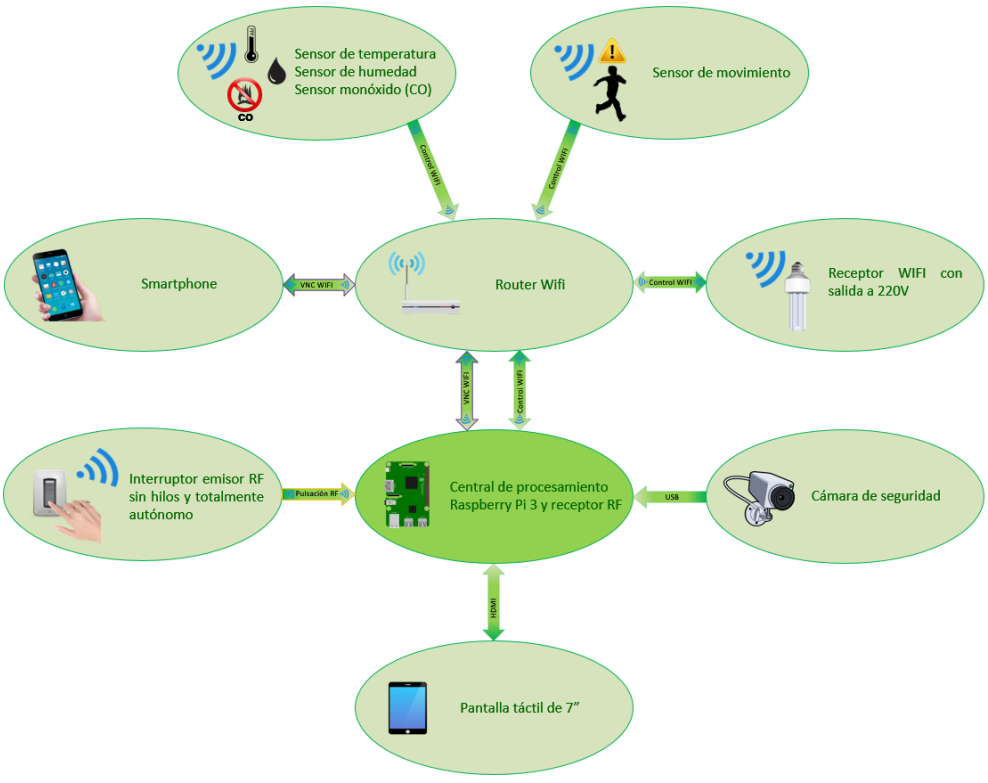


Diagrama 1. Diagrama de conexión del sistema de control

[5] Radiofrecuencia, sistema de comunicación mediante ondas electromagnéticas.

DESCRIPCIÓN DE FUNCIONAMIENTO

Existen dos comunicaciones importantes basadas en ondas electromagnéticas, comunicación por RF y por WIFI. En el caso de la RF, se utiliza para detectar si se ha sido presionado un pulsador EnOcean, que como el resto de componente más adelante se detallara su composición y funcionamiento. Por otro lado la comunicación WIFI que la obtendremos gracias al Router, por donde enviaremos y recibiremos señales a los receptores WIFI.

También existe una conexión tipo VNC, basada en una conexión WIFI, se utiliza para conectar un Smartphone o Tablet a la Raspberry Pi 3 y controlarla remotamente.

Y por último en cuanto a comunicaciones, existen las conexiones físicas (cableadas) que son HDMI, para conectar la pantalla a la RPI3 y las conexiones USB, para el control de la pantalla táctil, el receptor RF y la cámara de seguridad.

En el Diagrama 1. Diagrama de conexión del sistema de control observamos el conexionado del sistema completo, diferenciando el tipo de comunicaciones y si es unidireccional (entrada) o bidireccional (entrada y salida).

El cerebro del sistema se encuentra en la central de procesamiento (RPI3), que es la encargada de gestionar, almacenar y distribuir, las señales de control recibidas. Los periféricos serán de entrada o salida dependiendo si envían datos a la central o reciben datos de la central.

Existen dos periféricos de entrada, en el caso del pulsador EnOcean autosuficiente estará controlado por el usuario, en el caso de que uno de estos sea pulsado, se enviara una señal a la central para que accione uno o varios receptores previamente configurados. El sistema de seguridad consta de dos periféricos, la cámara que será unidireccional de entrada, únicamente enviará las imágenes a la central, en el caso de que haya detectado actividad el sensor de movimiento, que enviará la señal a la central para que empiece a recibir imágenes. El sensor de movimiento será únicamente un periférico de entrada, pero a continuación explicaré él porque es entrada y salida.

Y en paralelo con los pulsadores EnOcean encontramos periféricos de entrada y salida, como la interfaz de usuario (HMI), que la tendremos junto a la central, y otra inalámbrica, que se tratara de una conexión remota a la RPI3 desde nuestro Smartphone o Tablet. Tendremos los dos sentidos de comunicación, porque como entrada podremos interactuar sobre el estado de los receptores y en el modo configuración modificaremos el sistema a nuestro gusto. Pero también como salida, ya que observaremos el estado de los receptores, sensores y las imágenes.

Finalmente tenemos los receptores WIFI, que serán periféricos de entrada y salida, porque es capaz de recibir y enviar datos. En todas las variantes del receptor se comportará como entrada de datos a la central, porque cada vez que lo conectemos automáticamente enviará a la central su identificador, para que sea almacenada en el caso de que no se haya guardado anteriormente.

En el caso del sensor de temperatura y humedad, el sensor de monóxido de carbono y el sensor de movimiento enviarán la señal medida cada cierto tiempo a la central, para que esta la procese. Estos sensores serán puramente periféricos de entrada pero el receptor WIFI, tiene la posibilidad de comunicarse en ambas direcciones.

Y por último la variante de receptor WIFI con salida a 220V, que será puramente de salida a la central, ya que esta modificará el estado del receptor.

8. DESCRIPCIÓN DEL PROYECTO

A continuación, se explicará detalladamente como se ha llevado a cabo el proyecto, se comentará la función y el funcionamiento de cada componente del sistema.

Rigiéndome a los objetivos del proyecto, me limitaré a explicar únicamente lo necesario e indispensable para que un usuario con conocimientos básicos pueda crear su sistema de control.

Para facilitar la explicación separemos el conjunto en tres sistemas diferenciados:

- Sistemas de comunicación.
- Sistemas receptores WIFI (programado IDE Arduino).
- Sistemas entorno a la central de procesamiento (programado en Python).

8.1. Sistemas de comunicación

En el proyecto encontramos tres comunicaciones:

- Comunicación mediante WIFI.
- Comunicación mediante radiofrecuencia.
- Comunicación cableada.

8.1.1. Comunicación WIFI

WIFI es una tecnología de comunicación inalámbrica inicialmente utilizada en redes de área local y luego convertida en un medio para acceder a Internet de banda ancha.

La norma IEEE 802.11 (ISO/CEI 8802-11) es un estándar internacional que describe las características de una red de área local inalámbrica (WLAN). WIFI (contracción de Wireless Fidelity, en español Fidelidad inalámbrica) es el nombre dado inicialmente a esta certificación por la WECA (Wireless Ethernet Compatibility Alliance), el organismo encargado de certificar que los equipos cumplan la norma 802.11. Debido a un abuso de lenguaje y por razones de *marketing* el nombre de la norma se confunde hoy en día con el nombre de la certificación.

De este modo una red WIFI es en realidad una red conforme a la norma 802.11.

Imagen 42. Logo WIFI

Gracias la WIFI es posible crear redes de área local inalámbricas de banda ancha. De este modo WIFI permite que se comuniquen PC portátiles, equipos de escritorio, asistentes personales (PDA) e incluso periféricos con una conexión de banda ancha (11 Mbit/s) dentro de un radio de varias decenas de metros al interior (generalmente entre 20 y 50 metros). Al aire libre el alcance puede ser de varias centenas de metros y en condiciones óptimas varias decenas de kilómetros.

Por esto los proveedores de acceso a Internet comienzan a ofrecer acceso inalámbrico a Internet en zonas con bastante concentración de usuarios (estaciones de tren, aeropuertos, hoteles, trenes, etc.). Estas zonas de acceso a Internet son llamadas *hot spots*.

Las características principales de una red WIFI son:

Transmiten frecuencias de 2,4 GHz o 5 GHz. Esta frecuencia es considerablemente más alta que las frecuencias utilizadas por los teléfonos móviles, los walkie-talkies y las televisiones. Las frecuencias altas permiten que la señal transmita mayor información.

Utilizan los estándares de la red 802.11, que tiene diferentes variantes como la 802.11a a 54MB/s, la 802.11b a 11MB/s, 802.11g a 54Mb/s, 802.11n a 450KB/s y la 802.11ac a 1300MB/s.

Otro estándar 802.11 se centra en las aplicaciones específicas de la red wireless, como un área amplia de conexiones (WANs) dentro de los vehículos o de la tecnología, de forma que te permita moverte de una red inalámbrica a otra sin problemas.

Las radios WIFI pueden transmitir en cualquiera de sus tres bandas de frecuencia. O, pueden realizar "*saltos de frecuencia*" rápidos entre las diferentes bandas. A menudo el realizar estos saltos reduce la interferencia y permite que múltiples dispositivos utilicen la misma conexión, simultáneamente.

Protocolo MQTT

MQTT (Message Queue Telemetry Transport), un protocolo usado para la comunicación machine-to-machine (M2M) en el "Internet of Things"[6]. Este protocolo está orientado a la comunicación de sensores, debido a que consume muy poco ancho de banda y puede ser utilizado en la mayoría de los dispositivos con pocos recursos, como en nuestro caso la D1 mini pro. La arquitectura de MQTT sigue una topología de estrella, con un nodo central que hace de servidor o "broker" con una capacidad de hasta 10000 clientes. El “broker” es el encargado de gestionar la red y de transmitir los mensajes, para mantener activo el canal, los clientes mandan periódicamente un paquete (PINGREQ) y esperan la respuesta del “broker” (PINGRESP).

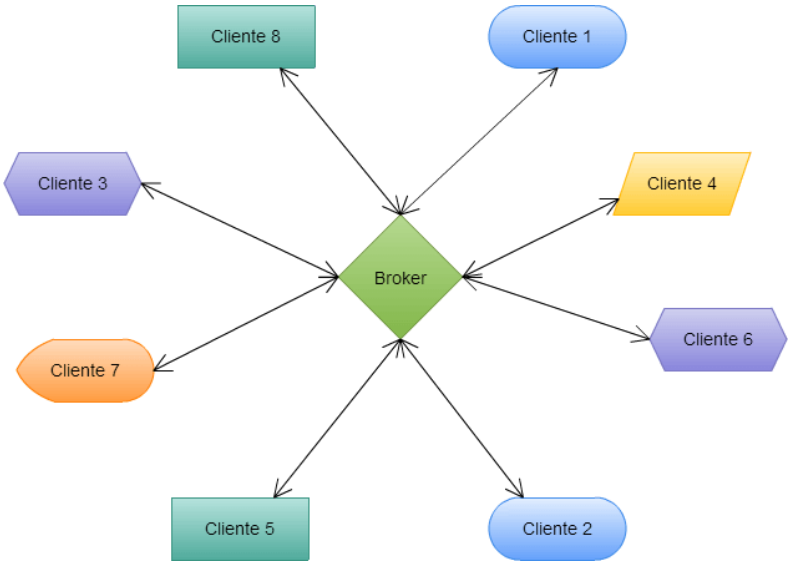


Diagrama 2. Diagrama de comunicación MQTT

Resumiendo, el protocolo MQTT se basa en que alguien publica un mensaje con el identificador de un “topic” específico (no te preocupes porque veremos los “topics” con más detalle). El broker distribuye el mensaje a todos los clientes (aplicaciones o dispositivos) que le constan como subscritos a ese “topic”. Esos clientes reciben y consumen los datos de esos mensajes.

La comunicación puede ser cifrada entre otras muchas opciones. La comunicación se basa en unos "topics" (temas), que el cliente que publica el mensaje crea y los nodos que deseen recibirlo

[6] Concepto que se refiere a la interconexión digital de objetos cotidianos con internet. Alternativamente, Internet de las cosas es la conexión de Internet con “cosas u objetos”.

deben subscribirse a él. Subscribirse significaría crear un medio de comunicación ("topic"), por el cual otro dispositivo podrá enviar datos a este. El dispositivo subscrito comprobará continuamente si ha llegado un dato. La comunicación puede ser de uno a uno, o de uno a muchos. Un "topic" se representa mediante una cadena y tiene una estructura jerárquica. Cada jerarquía se separa con '/'.

De esta forma se pueden crear jerarquías de clientes que publican y reciben datos, como podemos ver en diagrama siguiente.

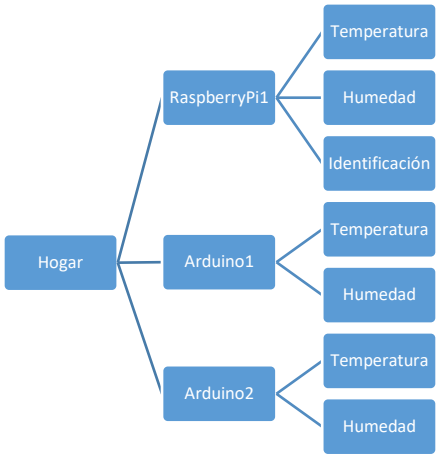


Diagrama 3. Diagrama jerárquico de una estructura "topics" de MQTT

Por ejemplo, si el dispositivo Arduino1 se subscribe a "/Hogar/Arduino1/Temperatura". Cualquier dispositivo, por ejemplo la RaspberryPi1 podrá publicar (enviar) un dato al Arduino1 mediante el topic "/Hogar/Arduino1/Temperatura".

Un nodo puede subscribirse a un "topic" para recibir datos de Temperatura "/Hogar/Arduino1/Temperatura" o recibir todos los datos si se subscribe "/Hogar/Arduino", donde todos los Arduinos recibirán la información enviada por este "topic".

Para que se entienda con un ejemplo, si el Arduino1 se subscribe a "/Hogar/Arduino1/Temperatura" y el Arduino2 se subscribe a través del "topic" "/Hogar/Arduino2/Temperatura", y la RaspberryPi1 publica (envía) un dato en el "topic" "/Hogar/Arduino" o "/Hogar", ambos Arduinos recibirán el dato.

A primera vista parece algo complejo pero cuando entremos en la programación y el tratamiento de los "topics", veremos que realmente es un sistema muy sencillo y con muchas posibilidades.

Lógicamente, el modelo publisher/subscriber sobre el que se basa el protocolo MQTT no es perfecto. El desacoplo que se produce entre el productor y consumidor trae algunos contratiempos. Cuando el filtro de mensajes se hace por topics, el más evidente de los

problemas es que ambos actores tienen que saber de antemano el conjunto de "topics" que se utilizan.

Otro aspecto importante es que el productor del mensaje no se entera de si los subscriptores están leyendo sus mensajes o no.

Para solventar todo esto, el protocolo MQTT implementa hasta tres niveles de QoS (Quality of Service) para garantizar que los mensajes son entregados de forma correcta a los nodos. Cuanto más alto el nivel, más fiable es la transmisión, pero también supone un mayor consumo de ancho de banda o una mayor latencia.

Además, el servidor es capaz de mantener el mensaje incluso después de ser enviado a los nodos subscritos, de manera que si se producen nuevas subscripciones a los "topics" de los mensajes retenidos, estos se envían a los nuevos clientes.

Configuración del Router

El sistema está diseñado para funcionar con cualquier Router convencional que nos proporcionan las compañías telefónicas.

Imagen 43 Router convencional

En el caso de no disponer de una conexión telefónica aprovecharemos la red WIFI (WLAN) de un Router configurándola para utilizarla como medio de comunicación.

Utilizaremos el SSID y la contraseña de nuestra WLAN para que nuestros periféricos y la central de procesamiento puedan comunicarse.

Para acceder al modo de configuración de nuestra WLAN deberemos conectarnos a la red con el SSID y la contraseña predeterminadas que se indican en la parte posterior del Router. Una vez conectados teclearemos en nuestro navegador "192.168.1.1". Como se muestra en la imagen 2, cambiaremos el nombre y contraseña de nuestra red inalámbrica.

(Tendremos un entorno de configuración diferente para cada modelo de Router y compañía).

Imagen 44. Configuración sistema de comunicación WLAN

8.1.2. Radiofrecuencia

EnOcean Serial Protocol 3

EnOcean ha creado su protocolo de comunicación radiofrecuencia llamado EnOcean Serial Prolocol 3. Los dispositivos de EnOcean pueden enviar 1 byte de dato (1BS) o 4 bytes de datos (4BS), por ejemplo el pulsador EnOcean envía un 1 byte y los sensores de temperatura envían 4 bytes.

En nuestro caso nos centraremos en el protocolo para enviar 1 byte (RPS). El paquete enviada con cada pulsación está formado por 14 bytes como podemos observar en la siguiente imagen.

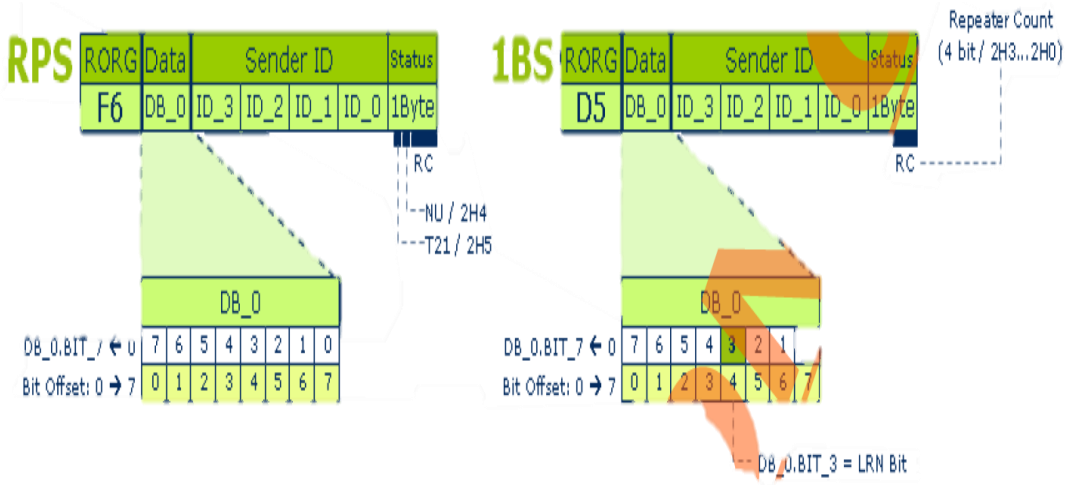


Imagen 45. Protocolo EnOcean Serial 3

El paquete que recibiremos cuando se accione el pulsador EnOcean será el siguiente:

Byte 1 y 2: RORG. "F6" comunicación RPS (1 byte de datos)

Byte 3 y 4: DATA. Recibiremos el número del pulsador de los cuatro que forman el pulsador EnOcean. En nuestro caso tendremos cuatro datos cuando se pulsen y cuatro para cuando se despulsen.

Byte 5 a 13: SENDER ID. Cada Pulsador EnOcean tendrá un Identificador formado de 9bytes.

Byte 14: STATUS.

Este protocolo de comunicación se tratará en Python, para detectar cada pulsación y que la central de procesamiento ejecute la acción que programemos.

8.1.3. Cableada

La pantalla táctil se conectará a la central de procesamiento mediante HDMI, protocolo utilizado en la mayoría de conexiones externas de sistemas de imagen.

El control táctil de la pantalla y el receptor de comunicación de radiofrecuencia se conectará a la central de procesamiento mediante USB.

8.2. Sistemas receptores WIFI

En este apartado de explicará paso a paso como configurar el IDE de Arduino e instalar librerías. Y entraremos en la programación de cada tipología de receptor WIFI.

8.2.1. Configuración del IDE de Arduino

Instalación de las tarjetas ESP8266

Debemos instalar el Arduino IDE versión 1.6.4 o superior, que podemos encontrar en: https://www.arduino.cc/en/main/software

Una vez instalado, debemos configurar nuestro IDE de Arduino para que reconozca a nuestra D1 mini pro como una tarjeta.

Vamos a archivo>Preferencias y en la casilla "Gestor de URLs Adicionales de Tarjetas" agregamos:
http://arduino.esp8266.com/package_esp8266com_index.json

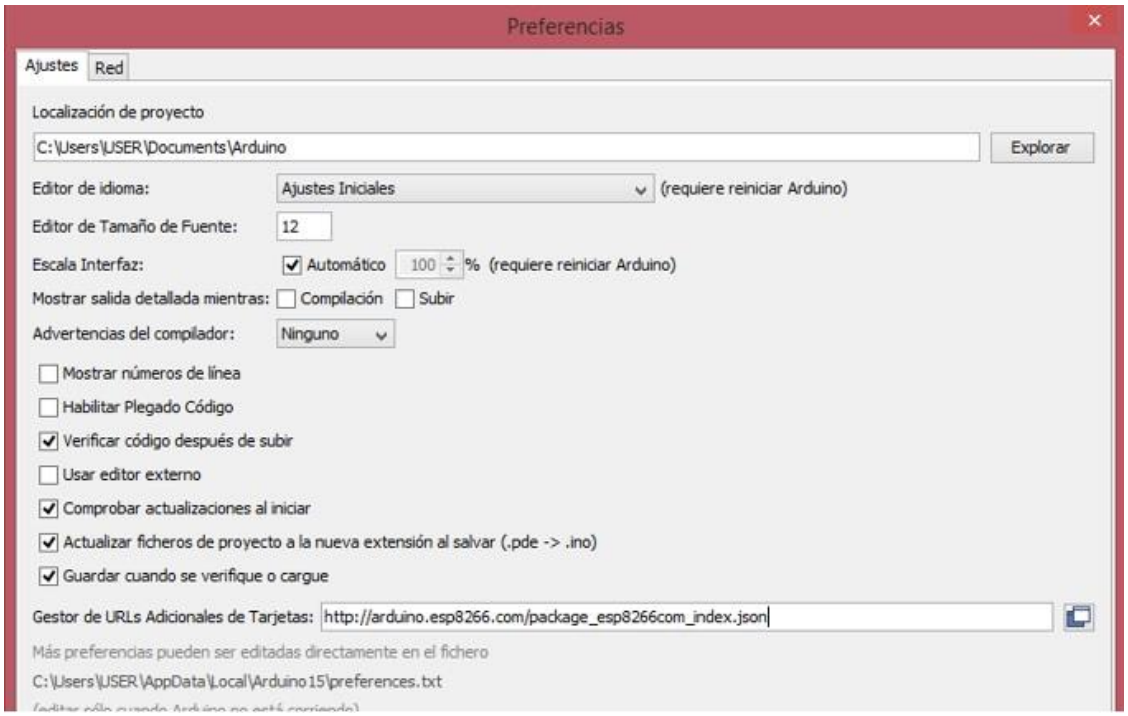


Imagen 46. Agregamos la URL para que IDE Arduino añada las tarjetas ESP8266

Seguidamente vamos a Herramientas>Placa: … >Gestor de Tarjetas.

Y buscamos en la lista "esp8266 by ESP8266 Community", lo seleccionamos e instalamos.

Imagen 47. Instalar tarjetas ESP8266

Esperamos a que finalice la instalación, y comprobamos que el ítem del ESP8266 este instalado.

Ahora en herramientas>placas, deben de estar las nuevas placas instaladas.

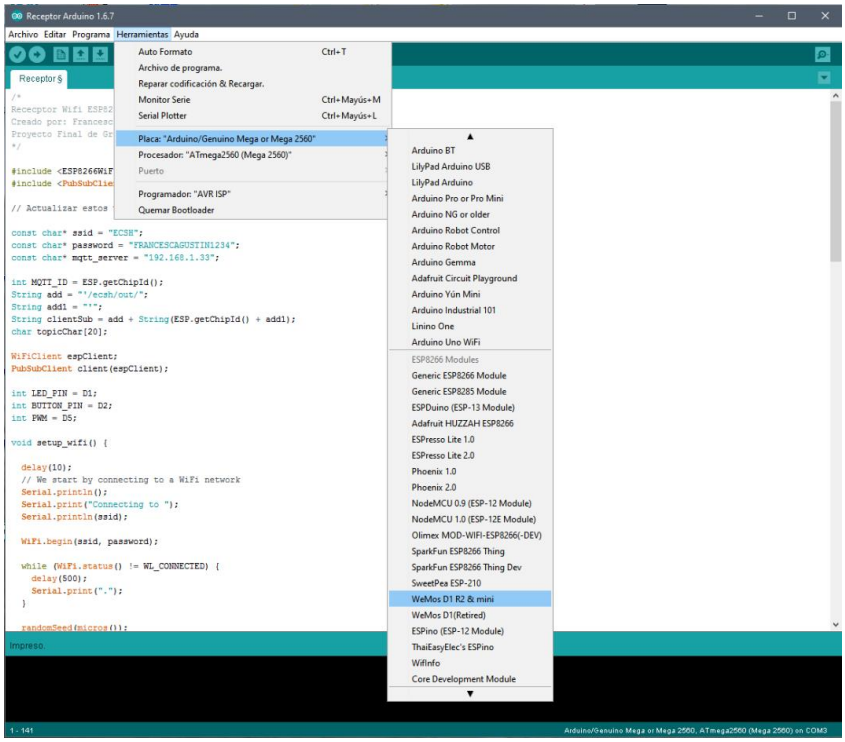


Imagen 48. Comprobación de la instalación de las tarjetas ESP8266

Instalación de librerías en IDE Arduino

Lo primero que tenemos que hacer es descargar la librería que queremos instalar.

Librería ESP8266WIFI, para utilizar las funciones de la placa D1 mini pro:

https://github.com/ekstrand/ESP8266WiFi/archive/master.zip

Librería PubSubClient, para utilizar el protocolo WIFI MQTT:

https://github.com/knolleary/pubsubclient/archive/v2.6.zip

Librería DHT, para capturar las lecturas del sensor de temperatura y humedad:

https://github.com/markruys/arduino-DHT/archive/master.zip

El procedimiento para instalar una librería es muy sencillo y basta seguir unos pocos pasos para hacerlo con éxito.

Elegimos en el menú del IDE: Programa>Incluir Librería>Añadir Librería .ZIP…

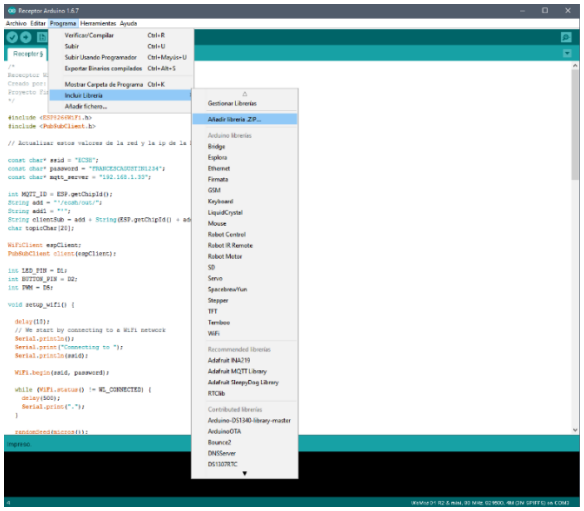


Imagen 49. Añadir librería IDE Arduino

Ahora nos pedirá la dirección del fichero descargado en formato .zip.

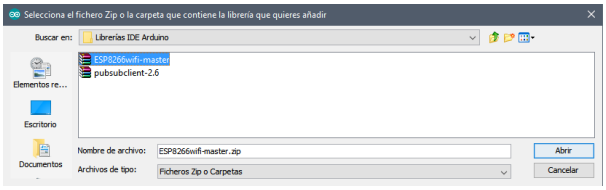


Imagen 50. Selección de la librería .zip

Busca y selecciona la librería. Cuando aceptemos se instalará. Si no hay ningún problema podemos comprobar que aparece en la lista de librerías instaladas y que podemos seleccionarla.

8.2.2. Receptores WIFI con una salida

Distribución de los receptores

Colocaremos un receptor en cada punto que queramos controlar independientemente de otro, por ejemplo, si queremos controlar la iluminación de dos habitaciones y el comedor, colocaremos un receptor para cada habitación y otro para el comedor.

Diagrama de flujo del receptor WIFI con una salida

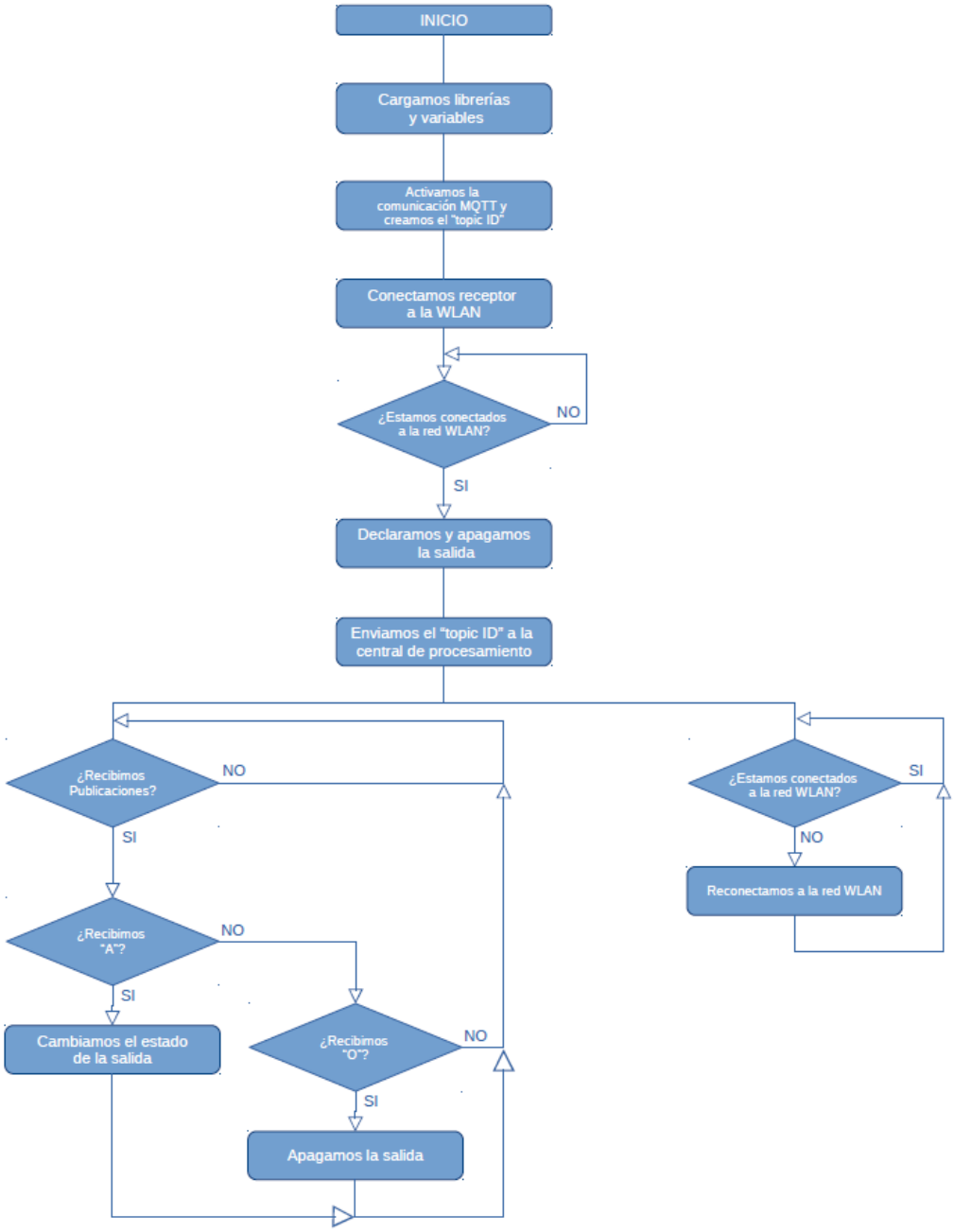


Diagrama 4. Diagrama de flujo del receptor WIFI con una salida

Funcionamiento

Este dispositivo está pensado para que cuando se alimente a 110-220VAC automáticamente se conecte a la red WIFI que previamente habremos configurado. Seguidamente enviará su identificador a la central de procesamiento para que esta lo guarde en el caso de que sea la primera vez que se haya conectado.

Existen dos modalidades de receptor WIFI, dependiendo de la potencia que deseemos a la salida, uno nos permite una salida de tensión máxima de 250VAC y una corriente máxima de 2A. Y el otro modelo una salida de tensión máxima de 250VAC o 30VDC y una corriente máxima de 10A.

Ambos modelos funcionarán de igual forma, pero a su salida conectaremos un relé de estado sólido o un relé mecánico.

En cuanto a los programas para los siguientes dispositivos, tendremos dos, uno que consistirá en el receptor WIFI con una salida y los sensores. Donde se activará automáticamente la lectura de las señales de los sensores en el caso de que recibamos información del sensor de temperatura. Y otro programa únicamente para el receptor WIFI con cuatro salidas.

Receptor WIFI con relé de estado sólido 220VAC

El modelo de receptor WIFI con salida de 220V y corriente máxima de 2A, se utiliza un relé de estado sólido con unas características superiores al otro modelo. Un punto fuerte que quería remarcar se trata de que cuando activemos la salida no escuchamos el ruido de la conmutación mecánica del relé, ya que como bien hemos explicado, es una conmutación electrónica y será más rápida. En su contra este modelo únicamente permite una conexión de corriente alterna. Con este modelo podemos conectar sistemas hasta aproximadamente 400W.

De este modelo se ha diseñado una PCB con entrada a 220V, y la salida a la misma tensión, explicaremos el diseño.

He protegido la cara Bottom de la PCB porque tenemos todas las conexiones de 220V por esa cara, para poder manipularla sin riesgo de electrocución.

Esta versión está formada por los siguientes componentes:

- D1 mini pro, con el chip ESP8266.
- Fuente alimentación integrada 220VAC a 5VDC.
- Relé de estado sólido G3MB-202P.

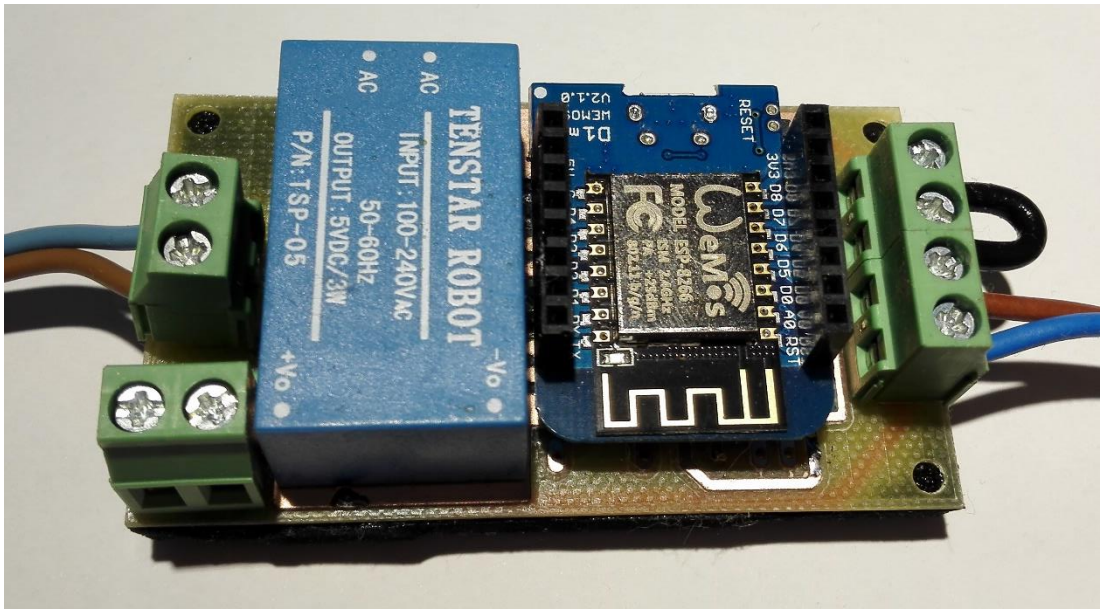


Imagen 51. Vista superior del receptor WIFI con relé de estado sólido 220VAC

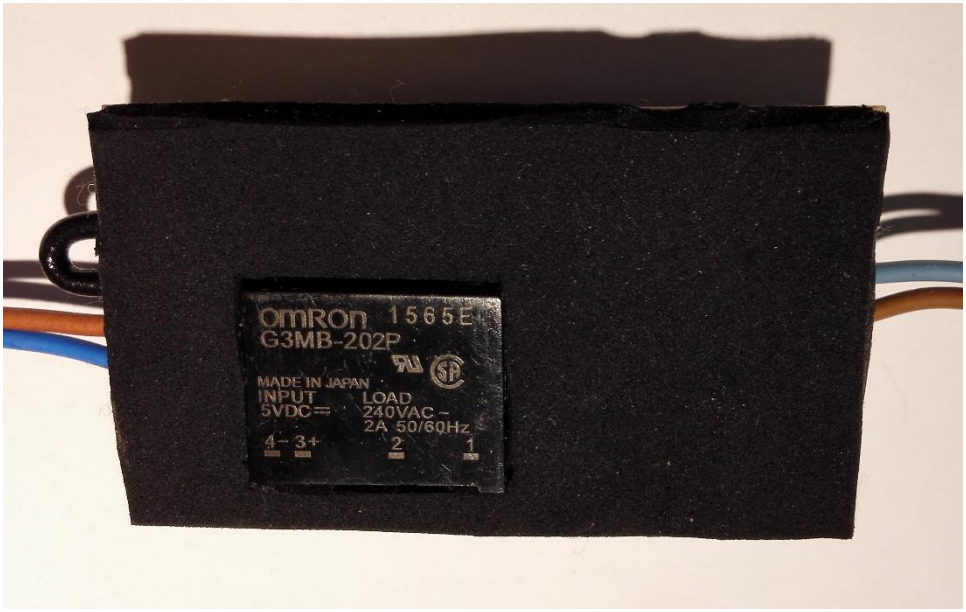


Imagen 52. Vista inferior del receptor WIFI con relé de estado sólido 220VAC

Receptor WIFI con relé electromecánico

El modelo de receptor WIFI con relé electromecánico nos permite una salida de un contacto conmutado (normalmente abierto y cerrado), al cual podremos conectar un dispositivo de corriente alterna o continua con una corriente máxima de 10A. Con este modelo podemos conectar sistemas hasta aproximadamente 2000W en corriente alterna y aproximadamente 300W para corriente continua.

Imagen 53. Conexión D1 mini pro y Relé electromecánico "Shield" (sin fuente de alimentación)

Esta versión está formada por los siguientes componentes:

- D1 mini pro, con el chip ESP8266.
- Fuente alimentación integrada 220VAC a 5VDC.
- Relé electromecánico para D1 mini pro WeMos.

Explicación del funcionamiento del código

El orden utilizado para explicar el código no es el que nos encontramos en el programa real, pero lo he modificado para que sea más entendible. El programa utilizado para el receptor WIFI con una salida será el mismo que el utilizado para el receptor WIFI con sensores, pero en el caso de no detectar un valor de temperatura únicamente funcionará como receptor. En este apartado comentaremos el código exclusivamente del receptor.

1) En la cabecera del código encontramos información del proyecto, nombre del autor e incluimos las librerías necesarias para este apartado, que son la "ESP8266WiFi.h" y la "PubSubClient.h".

```
/*
Receptor Wifi ESP8266
Creado por: Francesc Agustin Mora
Proyecto Final de Grado: Easy Control Smart Home
*/

//Incluimos librerías

#include <ESP8266WiFi.h>
#include <PubSubClient.h>
#include "DHT.h"

DHT dht;
```

2) Introduciremos el nombre y la contraseña de la red WLAN de nuestro sistema. Y por último indicaremos la dirección IP de nuestra central de procesamiento (RPI3).

```
// Introducimos los datos de nuetra red WIFI e introducimos la IP de la RPI

const char* ssid = "ECSH";
const char* password = "FRANCESCAGUSTIN1234";
const char* mqtt_server = "192.168.1.33";
```

3) Adquirimos el identificador de nuestra D1 mini pro y añadimos el "topic" para enviar este dato a la RPI3 y esta lo guarde como nuevo receptor, en el caso de que no se hubiera conectado anteriormente. El "topic" será el siguiente "/ecsh/out/ID", donde el ID será el número del identificador de la placa. De este modo tendremos un "topic" por el que recibiremos los datos, único para cada identificador.

```
// Guardamos el ID de nuestra Placa D1 mini pro y lo preparamos para enviar a la RPI

int MQTT_ID = ESP.getChipId();
String add = "'/ecsh/out/";
String add1 = "'";
String clientSub = add + String(ESP.getChipId() + add1);
char topicChar[20];
```

4) Creamos el nuevo cliente para la comunicación WIFI y habilitamos la comunicación MQTT del cliente.

```
// Creamos el cliente y habilitamos MQTT

WiFiClient espClient;
PubSubClient client(espClient);
```

5) Declaramos variables, para el receptor únicamente crearemos la declaración de un "Integer" con nombre de Salida1 y estará conectada al pin D1 de la D1 mini pro.

```
// Declaramos entradas y salidas

int Salida1 = D1;
String temp_str;
String hum_str;
String MQ_str;
char temp[50];
char hum[50];
char MQ[50];
long lastMsg = 0;
```

6) "Setup()" únicamente se ejecutará en la primera ejecución del programa, donde llamaremos a "setup_WiFi()", que lo veremos en el siguiente punto. Y llamaremos a los métodos "setServer" y "setCallback", para que puedan utilizarse. Por último declaramos la variable Salida como salida y la apagamos.

```
// Únicamente se ejecutará setup en la primera ejucución del programa
// Ejecutamos setup_wifi y llamamos a los métodos setServer y setCallback

void setup() {
  setup_wifi();
  setup_sensor();
  Serial.begin(9600);
  client.setServer(mqtt_server, 1883);
  client.setCallback(callback);

  // Declaramos las Salidas como salidas y la apagamos
  pinMode(Salida1, OUTPUT);
  digitalWrite(Salida1, LOW);
}
```

7) "Setup_WiFi()" se llama desde "setup()" pero se han separado los "setups" del programa normal y el que solo trata el WIFI, en este conectaremos la placa D1 mini pro a la red WLAN.

```
// Conectamos nuestra Placa D1 mini pro a la red WLAN

void setup_wifi() {

  delay(10);
  WiFi.begin(ssid, password);

  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
  }
  randomSeed(micros());
}
```

8) Cuando recibamos un dato el código saltará a este método, donde comprobaremos que datos hemos recibido y en caso de recibir "A", cambiaremos el estado de la salida. Cuando encendamos la central de procesamiento enviaremos "O", para apagar todos los receptores.

```
// Cuando detectemos que ha llegado un dato, en este caso "I", cambiaremos el estado de la salida

void callback(char* topic, byte* payload, unsigned int length) {

  if ((char)payload[0] == 'A') {
    digitalWrite(Salida1, !digitalRead(Salida1));
  }
  else if ((char)payload[0] == 'O') {
    digitalWrite(Salida1, LOW);
  }
}
```

9) En el caso de que el sistema no esté conectado o se haya desconectado se ejecutará desde el "loop" principal. En el caso de no poder conectarse, esperaremos 5 segundos y volveremos a intentarlo.

   Por otro lado subscribiremos la placa en "/ecsh/out/ID", este será el "topic" por el que recibiremos los datos. De esta forma podemos enviar el mismo dato de activación desde la central de procesamiento, pero únicamente lo recibirá el dispositivo con el identificador del "topic".

```
// En el caso de que el sistema este desconectado o se haya desconectado se ejecutará reconnect
// Cuando nos conectemos automáticamente enviaremos el id a la RPI

void reconnect() {

  // Solo en el caso de que estemos desconectados entraremos en el loop
  while (!client.connected()) {
    // Intentamos conexión
    String clientId = "ESP8266Client-";
    clientId += String(random(0xffff), HEX);
    if (client.connect(clientId.c_str())) {
      // Pasamos el topic + id a char
      clientSub.toCharArray(topicChar, 20) ;
      Serial.println(topicChar);
      for (int i = 0; i < 9; i++) {
        Serial.print(" 0x");
        Serial.print(topicChar[i], HEX);
      }
      Serial.println();
      // Enviamos id del ESP8266 a la RPI
      client.publish("/ecsh/out/id", topicChar);
      // Subcribimos los topics para recibir datos
      client.subscribe("inTopic");
      client.subscribe(topicChar);

    } else {
      // En el caso de no poder conectarnos esperamos 5 segundos e intentamos de nuevo
      delay(5000);
    }
  }
}
```

10) Una vez se haya ejecutado el "setup()" y el resto de programa, el programa se quedará comprobando que seguimos conectados a la red WLAN, y quedará a la espera de que llegue un dato de la central de procesamiento. En el caso de que caiga la conexión llamaremos a "reconnect()", e intentaremos conectarnos de nuevo.

```
// Quedaremos en el loop comprobando que seguimos conectados y leyendo los sensores, hasta que llegue un dato.

void loop() {

  sensores();
  // Comprobamos que seguimos conectados
  if (!client.connected()) {
    reconnect();
  }
  client.loop();
}
```

8.2.3. Receptor WIFI con cuatro salidas

Este receptor es una adaptación del anterior con la finalidad de poder controlar varios sistemas con un mismo receptor. Por ejemplo, para controlar dos sistemas de iluminación y el control de persianas.

En este caso el receptor dispone de cuatro relés a la salida y habría que modificar el código comentado en el punto anterior para que tuviera cuatro salidas y que la central de procesamiento sea capaz de enviar cuatro variables de control para un mismo identificador.

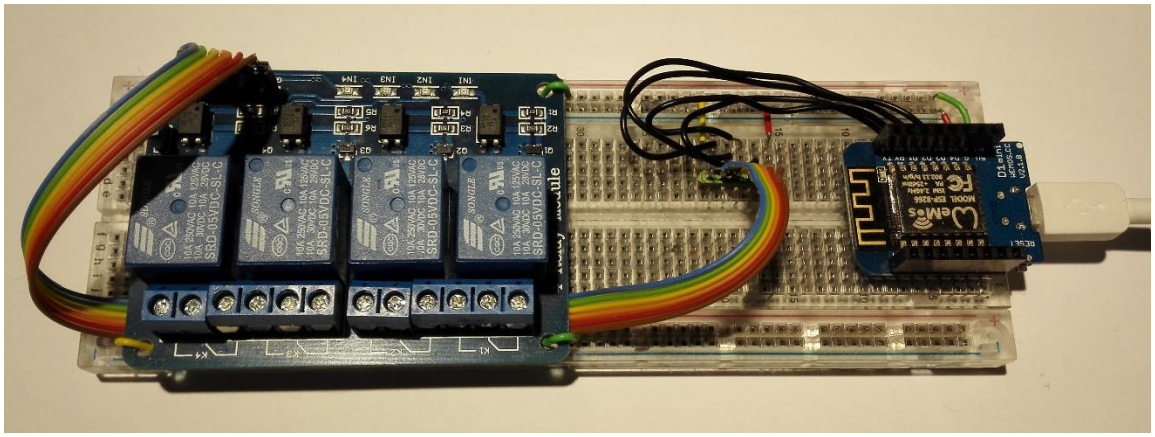


Imagen 54. Receptor WIFI con cuatro salidas a relé electromecánico

Diagrama de flujo del receptor WIFI con cuatro salidas

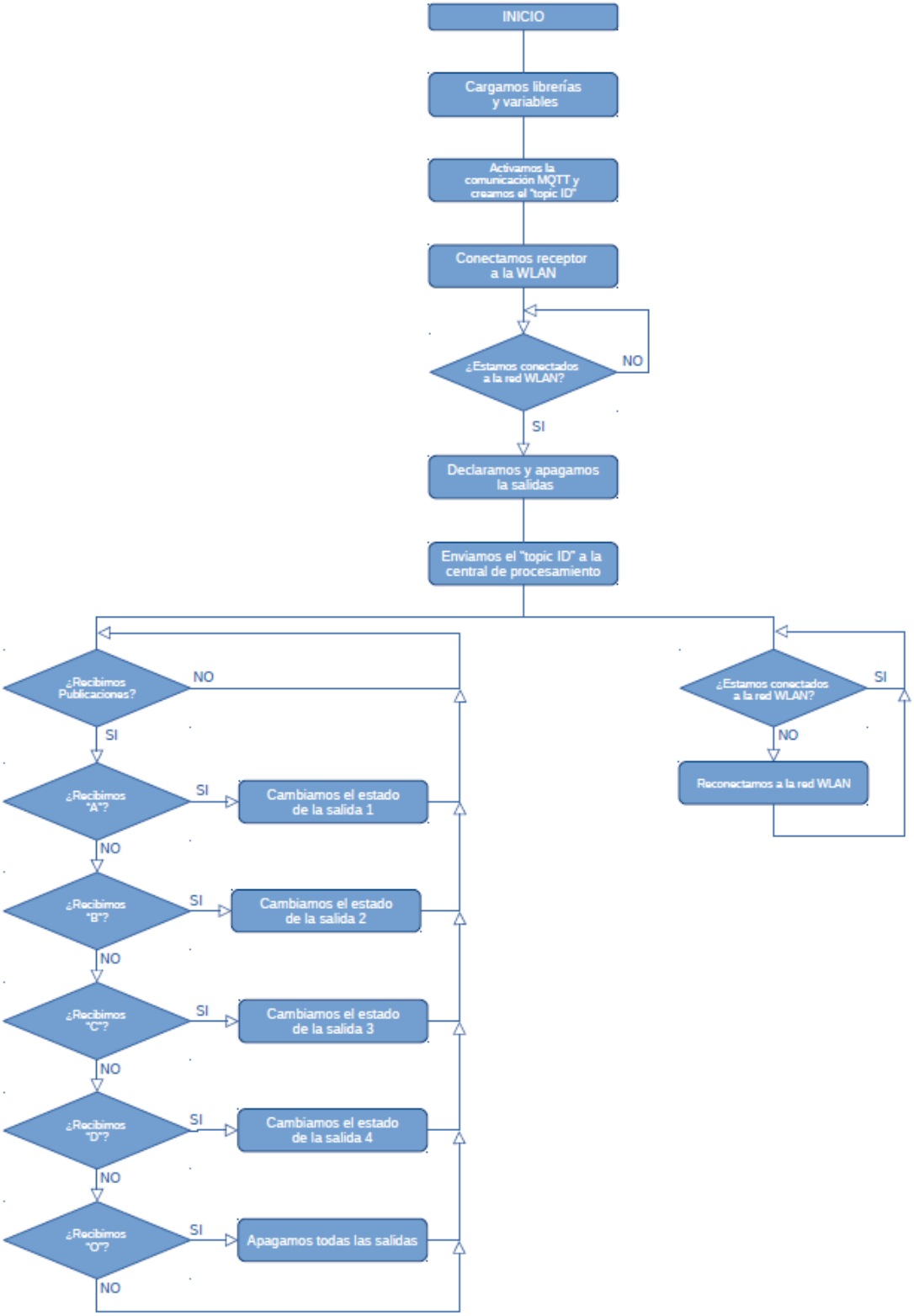


Diagrama 5. Diagrama de flujo del receptor WIFI con cuatro salidas

Explicación del funcionamiento del código

Las modificaciones del código del receptor con cuatro salidas son las siguientes:

1) Declaramos las cuatro salidas.

```
// Declaramos entradas y salidas

int Salida1 = D1;
int Salida2 = D2;
int Salida3 = D3;
int Salida4 = D4;
```

2) "Setup()" únicamente se ejecutará en la primera ejecución del programa. El cambio que podemos observar es que declaramos las cuatro variables de salida como salida y forzamos su apagado.

```
// Únicamente se ejecutará setup en la primera ejucución del programa
// Ejecutamos setup_wifi y llamamos a los métodos setServer y setCallback

void setup() {
  setup_wifi();
  client.setServer(mqtt_server, 1883);
  client.setCallback(callback);

  // Declaramos las Salidas como salidas y la apagamos
  pinMode(Salida1, OUTPUT);
  pinMode(Salida2, OUTPUT);
  pinMode(Salida3, OUTPUT);
  pinMode(Salida4, OUTPUT);
  digitalWrite(Salida1, LOW);
  digitalWrite(Salida2, LOW);
  digitalWrite(Salida3, LOW);
  digitalWrite(Salida4, LOW);
}
```

3) Al igual que en el caso de una sola salida si recibimos una "A", cambiaremos la salida 1. En el caso de que recibamos una "B", "C" o "D", cambiaremos el estado de la salida 2, 3 o 4, respectivamente. Cada vez que encendamos la central de procesamiento forzaremos el apagado de todas las salidas, podemos observar que al contrario del receptor de una salida, en este caso los relés funcionan a la inversa, con un estado alto se desactivan.

```
// Cuando detectemos que ha llegado un dato, en este caso "I", cambiaremos el estado de la salida

void callback(char* topic, byte* payload, unsigned int length) {

  if ((char)payload[0] == 'A') {
    digitalWrite(Salida1, !digitalRead(Salida1));
  }
  if ((char)payload[0] == 'B') {
    digitalWrite(Salida2, !digitalRead(Salida2));
  }
  if ((char)payload[0] == 'C') {
    digitalWrite(Salida3, !digitalRead(Salida3));
  }
  if ((char)payload[0] == 'D') {
    digitalWrite(Salida4, !digitalRead(Salida4));
  }
  else if ((char)payload[0] == '0') {
    digitalWrite(Salida1, HIGH);
    digitalWrite(Salida2, HIGH);
    digitalWrite(Salida3, HIGH);
    digitalWrite(Salida4, HIGH);
  }
}
```

8.2.4. Receptor WIFI con sensores de temperatura, humedad y de CO

Este tipo de receptor está pensado para capturar las medidas de temperatura, humedad y concentración de monóxido de carbón y que sean enviadas a la central de procesamiento para que se traten dichas señales.

Por ejemplo, podríamos detectar la temperatura y para el sistema de refrigeración de nuestro hogar con una histéresis o incluso con un control PID.

O en el caso de la concentración de monóxido de carbono, enviar una señal de alarma, o incluso, encender un sistema de recirculación de aire para evacuar dicho gas. Con este sensor podemos detectar varios gases, y conjuntamente con el sensor de temperatura podríamos detectar un incendio, como veremos a continuación cuando tratemos dicho sensor de CO.

Independientemente del sistema de medición, se ha añadido un relé a este sistema con la finalidad de poder actuar dependiendo de las señales, o simplemente aprovechar e instalar otro sistema de iluminación a este receptor.

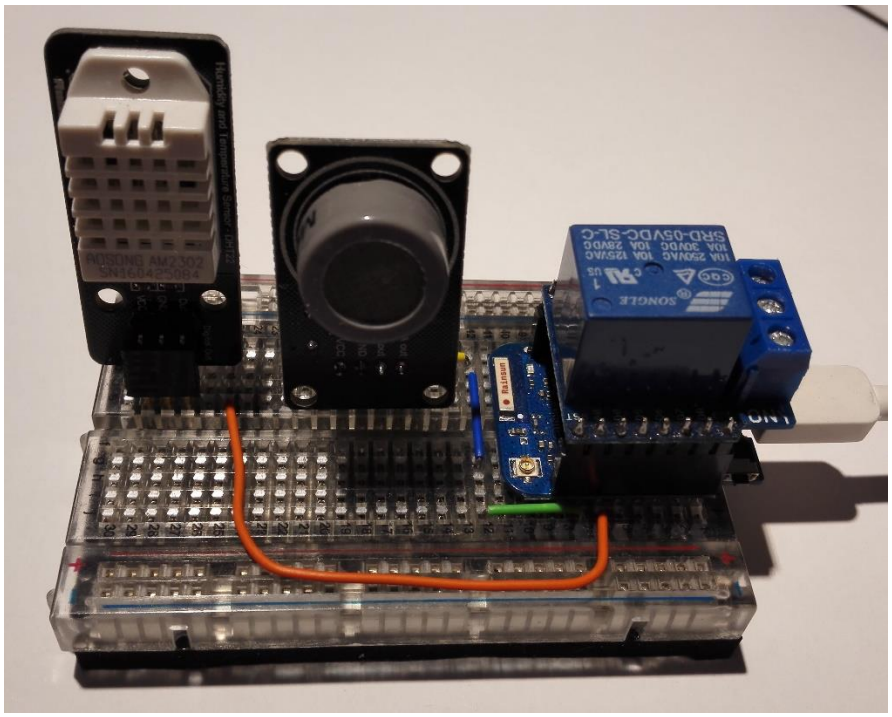


Imagen 55. Receptor WIFI con sensores de temperatura, humedad y concentración de monóxido de carbono

Funcionamiento

Este dispositivo tendrá el mismo funcionamiento que el receptor WIFI de una salida, pero se ha añadido el sistema de medición. Trataremos el funcionamiento de cada sensor por separado.

Diagrama de flujo del receptor WIFI con sensores

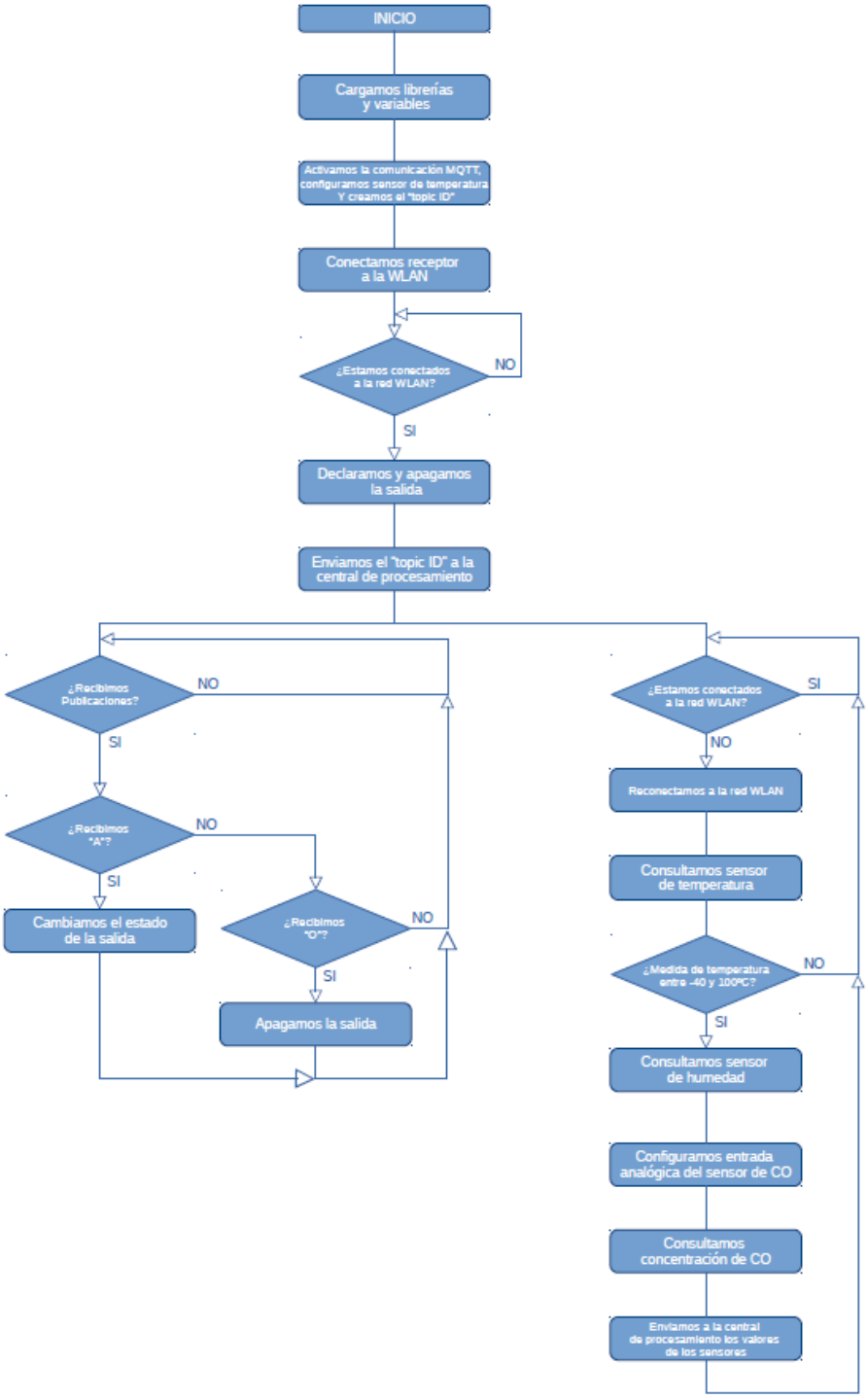


Diagrama 6. Diagrama de flujo del receptor WIFI con sensores

Sensor de temperatura y humedad DHT22

El sensor que utilizado es el DHT22, también conocido por AM2302. Se trata de un sensor de bajo coste, pero con unas capacidades más que aceptables para proyectos de este calibre, gracias a su sencillez de uso y montaje.

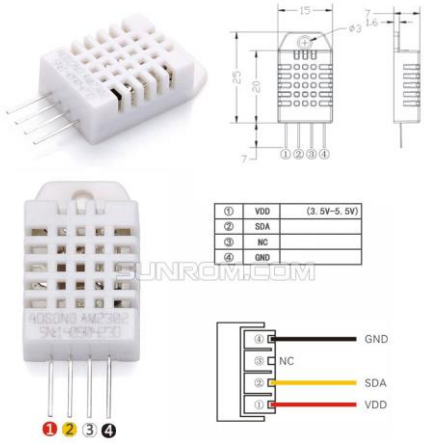


Imagen 56. Patillaje del DHT22

Según el esquema anterior, el patillaje del DHT22 se observa desde la parte frontal, de modo que la patilla 1 (izquierda) es la alimentación, de 3,3V a 6V DC, la 2 es la patilla de datos, y la 4 es GND. La 3 no se utiliza, por lo que puede quedar al aire sin problemas. En algunas webs he podido leer que es posible conectar el 3 a GND también.

Internamente monta un sensor polímero capacitivo para la humedad y el transistor DS18B20 para la temperatura, capaz de ofrecer lecturas digitales en el pin 2. La lectura es bastante precisa y no necesita calibración. Pero aún mejor, está preparado para ser conectado directamente a la alimentación y para obtener la señal directamente en un solo hilo, puesto que incluye una resistencia pull-up conectada desde el DATA hasta VDD, con lo que el conexionado es directo. Esto favorece el hecho de poder utilizarlo en proyectos de manera directa y muy sencilla. A continuación se muestran al detalle sus características:

- Alimentación: 3.3Vdc ≤ Vcc ≤ 6Vdc.
- Rango de medición de temperatura: -40°C a 80 °C.
- Precisión de medición de temperatura: <±0.5 °C.
- Resolución Temperatura: 0.1°C.
- Rango de medición de humedad: De 0 a 100% RH.
- Precisión de medición de humedad: 2% RH.
- Resolución Humedad: 0.1%RH.
- Sample rate: 0.5Hz (1 lectura cada 2s mínimo).

Sensor de monóxido de carbono MQ-7

Estos sensores son electroquímicos y varían su resistencia cuando se exponen a determinados gases, internamente posee un calentador encargado de aumentar la temperatura interna y con esto el sensor pueda reaccionar con los gases provocando un cambio en el valor de la resistencia. El calentador dependiendo del modelo puede necesitar un voltaje entre 5 y 2 voltios, el sensor se comporta como una resistencia y necesita una resistencia de carga (RL) para cerrar el circuito y con este hacer un divisor de tensión y poder leerlo desde un microcontrolador.

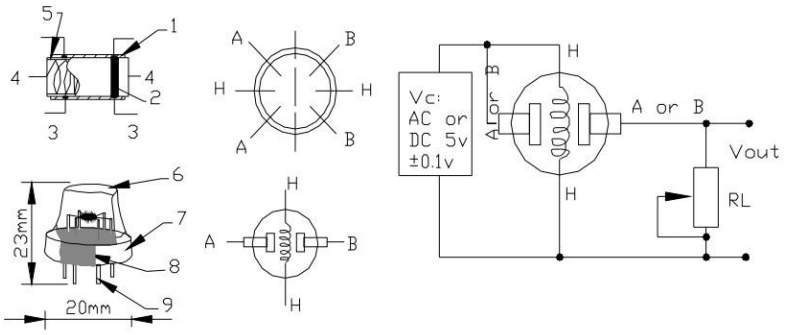


Imagen 57. Funcionamiento interno del sensor MQ-7

Debido al calentador es necesario esperar un tiempo de calentamiento para que la salida sea estable y tenga las características que el fabricante muestra en sus datasheet, dicho tiempo dependiendo del modelo puede ser entre 12 y 48 horas.

En nuestro caso el sensor MQ-7 viene integrado en un módulo, lo que nos simplifica la parte de conexiones y nos facilitan su uso, solo basta con alimentar el módulo y empezar a leer el sensor, estos módulos también tienen una salida digital la cual internamente trabaja con un comparador y con la ayuda de un potenciómetro podemos calibrar el umbral y así poder interpretar la salida digital como presencia o ausencia del gas.

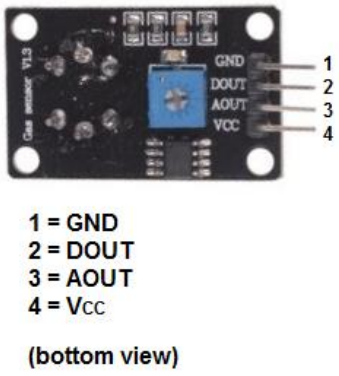


Imagen 58. Patillaje del sensor MQ-7

Para nuestro proyecto me ha parecido más razonable utilizar la entrada analógica de la placa D1 mini pro, y obtener una lectura más precisa. Sus características son:

- Alimentación 5V.
- Salida analógica o digital con comparador regulada mediante potenciómetro.
- Alta sensibilidad al monóxido de carbono.
- Rápida respuesta.

Explicación del funcionamiento del código de los sensores

El código de este dispositivo es exactamente el mismo que el receptor WIFI con una salida, en este caso detectaremos que recibimos una señal de temperatura, lo que activará la parte de medición de los sensores independientemente de la parte del receptor. Únicamente se explicará lo que esté relacionado con la lectura de los sensores, el resto se ha explicado en el receptor WIFI de una salida

1) Añadimos la librería "DHT.h" y la creamos para utilizarla posteriormente.

```
/*
Receptor Wifi ESP8266
Creado por: Francesc Agustin Mora
Proyecto Final de Grado: Easy Control Smart Home
*/

//Incluimos librerias

#include <ESP8266WiFi.h>
#include <PubSubClient.h>
#include "DHT.h"

DHT dht;
```

2) Declaramos variables de las tres mediciones a realizar y una para gestionar el tiempo de envío.

```
// Declaramos entradas y salidas

int Salida1 = D1;
String temp_str;
String hum_str;
String MQ_str;
char temp[50];
char hum[50];
char MQ[50];
long lastMsg = 0;
```

3) Creamos una llamada a "setup_sensor()".

```
// Únicamente se ejecutará setup en la primera ejucución del programa
// Ejecutamos setup_wifi y llamamos a los métodos setServer y setCallback

void setup() {
  setup_wifi();
  setup_sensor();
  Serial.begin(9600);
  client.setServer(mqtt_server, 1883);
  client.setCallback(callback);

  // Declaramos las Salidas como salidas y la apagamos
  pinMode(Salida1, OUTPUT);
  digitalWrite(Salida1, LOW);
}
```

4) Indicamos donde tenemos conectado el sensor DHT22.

```
// Inicializamos la lectura en el pin D0 de la Placa

void setup_sensor() {
  dht.setup(D0);
}
```

5) En cada ciclo de "loop" llamaremos a sensores para leer y enviar a la central de procesamiento la lectura.

```
// Quedaremos en el loop comprobando que seguimos conectados y leyendo los sensores, hasta que llegue un dato.

void loop() {

  sensores();
  // Comprobamos que seguimos conectados
  if (!client.connected()) {
    reconnect();
  }
  client.loop();
}
```

6) En esta subrutina "sensores()" realizaremos todas las mediciones de los sensores.

Para comprobar si se trata de un receptor WIFI con una salida o un receptor WIFI con sensores, realizaremos una medida de temperatura y en el caso de recibir un dato entre -40°C y 100°C continuaremos leyendo el resto de mediciones. En caso contrario saldremos de la subrutina.

La librería "DHT.h" nos permite poder obtener el dato en formato "float" del sensor con la instrucción "dht.getTemperatura()", lo que nos facilita su lectura. Seguidamente preparamos el dato en formato "float" y lo pasamos a "String", y finalmente lo empaquetamos en un "CharArray" para poder publicarlo.

Por último, leemos la entrada del convertidor analógico/digital (ADC), y la preparamos para poder enviarla.

Y finalmente enviamos por cada "topic" correspondiente el valor de las tres mediciones a la central de procesamiento.

```
void sensores() {
  // Lectura del sensor DHT22

  // Obtenemos los datos desde los sensores en el caso de que la temperatura nos entregue una medida
  // Temperatura en °C
  float temperature = dht.getTemperature();

  // Enviamos medidas de los sensores a la RPI siempre que tengamos medida de temperatura
  if (temperature <= -40 && temperature >= 100) {
    long now = millis();
    if (now - lastMsg > 200) {
      lastMsg = now;

    // Tanto por ciento de humedad
    float humidity = dht.getHumidity();

    // Temperatura en °F
    float ftemperature = dht.toFahrenheit(temperature);

    //Midiendo temperatura y humedad
    temp_str = String(temperature); //convertimos la temperatura de float a string
    temp_str.toCharArray(temp, temp_str.length() + 1); //Empaquetamos la temperatura para publicarla
    hum_str = String(humidity); //convertimos la humedad de float a string
    hum_str.toCharArray(hum, hum_str.length() + 1); //Empaquetamos la humedad para publicarla

    // Medimos la concentración de CO a traves de la entrada analógica
    int adc_MQ = analogRead(A0); //Lemos la salida analógica del MQ
    float voltaje = adc_MQ * (5.0 / 1023.0); //Convertimos la lectura en un valor de voltaje

    MQ_str = String(adc_MQ); //convertimos la temperatura de float a string
    MQ_str.toCharArray(MQ, MQ_str.length() + 1); //Empaquetamos la concentración de CO para publicarla

    client.publish("/ecsh/out/temperature", temp);
    client.publish("/ecsh/out/humidity", hum);
    client.publish("/ecsh/out/concentracion", MQ);
    }
  }
}
```

8.3. Sistemas entorno a la central de procesamiento

8.3.1. Listado de direcciones de los periféricos y configuración

En primer lugar explicaremos la forma de direccionar cada sistema de entrada y salida. En el caso de los pulsadores de EnOcean, cuando lo pulsemos por primera vez el sistema detectará el ID y en el caso de no tener el ID almacenado, creará a continuación de los que tengamos un nuevo ID de periférico de entrada en el documento. En cada pulsador EnOcean disponemos de cuatro pulsadores, es decir para cada ID tendremos cuatro entradas.

En la interfaz gráfica tendremos botones de estado, que también nos permitirán modificar el estado de la salida que conexionemos en paralelo con los pulsadores EnOcean. Un punto importante que debemos tener en cuenta es que el nombre que demos en el Qt Creator debe ser el mismo que indicamos en el documento de direcciones para el pulsador EnOcean y para el receptor.

Para los receptores, en cuanto sean alimentados enviarán su identificador junto al "topic" para que sea guardado en los periféricos de salida del documento de direcciones. De esta forma cuando conectemos un periférico nuevo podremos verlo en el documento de direcciones donde lo encontraremos sin nombre, lo nombraremos y conectaremos una entrada con una salida como se muestra en la Imagen 59. Documento de direccionamiento y conexionado de entradas y salidas. En las tres primeras columnas vemos la configuración de los pulsadores EnOcean, en la cuarta columna los botones de la interfaz gráfica y de la sexta a la octava columna encontramos la configuración de los receptores. Para configurarlo únicamente deberemos detectar el ID o el "topic ID" y nombrarlo de igual forma para toda la configuración.

	A	B	C	D	E	F	G	H	I
1	Nombre del pulsador EnOcean	ID de periférico de entrada	Número del pulsador	Nombre del botón de la interfaz gráfica	Conexionado entradas y salidas	Nombre del periférico de salida	Topic con ID de periférico de salida	Número relé	
2	Comedor 1	294818	10	Comedor 1		Comedor 1	'/ecsh/out/14211347	A	
3	Comedor 2	294818	30	Comedor 2		Comedor 2	'/ecsh/out/14211347	B	Cuatro relés
4	Persiana Subir	294818	50	Persiana Subir		Persiana Subir	'/ecsh/out/14211347	C	
5	Persiana Bajar	294818	70	Persiana Bajar		Persiana Bajar	'/ecsh/out/14211347	D	
6				Habitacion		Habitacion	'/ecsh/out/2888901'	A	Sensores
7				Exterior		Exterior	'/ecsh/out/2865133'	A	Estado sólido
8									
9									

Imagen 59. Documento de direccionamiento y conexionado de entradas y salidas

En ambos casos si se detecta el ID de la entrada en la lista automáticamente se ejecutará la acción que hayamos conexionado en la fila. Por ejemplo, en el caso de que pulsemos el pulsador EnOcean con el ID "294818" y número de pulsador "10" con el nombre Comedor 1, enviaremos al "topic" del receptor nombrado Comedor 1, un cambio de estado del primer relé, enviando una "A" a través del '/ecsh/out/2865133' y modificaremos el estado del botón de la interfaz

gráfica. Si pulsamos en la interfaz gráfica el botón Comedor 1, tendremos la misma acción que en el caso del pulsador EnOcean, es decir, funcionan en paralelo.

	A	B	C	D	E	F	G	H
1	Nombre del pulsador EnOcean	ID de periférico de entrada	Número del pulsador	Nombre del botón de la interfaz gráfica	Conexionado entradas y salidas	Nombre periférico de salida	Topic con ID de periférico de salida	Número relé
2	Comedor 1	294818	10	Comedor 1		Comedor 1	'/ecsh/out/14211347	A

Imagen 60. Direccionamiento de entradas y salidas del ejemplo

De esta forma tendremos en cada fila el ID de EnOcean (Azul), numero de pulsador (Verde), botón interfaz gráfica (Rojo) y por último el receptor (Naranja) con su número de relé (Lila), el relé del receptor lo accionarán los dispositivos de entrada de la misma fila. Tendremos la facilidad de direccionar y conexionar nuestro sistema de entradas y salidas sin tener que modificar el código.

Un punto importante antes de explicar cómo nos aparecerá el nuevo dispositivo, es que cada vez que se cree el ID de un dispositivo, deberemos configurarlo antes de iniciar de nuevo el sistema. En el caso de no tener bien configurado el documento de direcciones el direccionamiento se ejecutará incorrectamente.

En momento en el que alimentemos un receptor o pulsemos un pulsador por primera vez, el sistema detectará si tenemos guardado el ID del dispositivo. En el caso de no tenerlo almacenado se añadirá al archivo Excel como se muestra en la Imagen 61. Nuevo pulsador almacenado para el pulsador EnOcean y como se muestra en la Imagen 61. Nuevo pulsador almacenado Imagen 62. Nuevo receptor almacenado para el caso del receptor.

	A	B	C
1	Nombre del pulsador EnOcean	ID de periférico de entrada	Número del pulsador
2	Comedor 1	00294818	10
3	Comedor 2	00294818	30
4	Persiana Subir	00294818	50
5	Persiana Bajar	00294818	70
6			
7			
8	Nuevo pulsador	01804327	

Imagen 61. Nuevo pulsador almacenado

	F	G	H
1	Nombre periférico de salida	Topic con ID de periférico de salida	Número relé
2	Comedor 1	'/ecsh/out/14211347	A
3	Comedor 2	'/ecsh/out/14211347	B
4	Persiana Subir	'/ecsh/out/14211347	C
5	Persiana Bajar	'/ecsh/out/14211347	D
6	Habitacion	'/ecsh/out/2888901'	A
7	Exterior	'/ecsh/out/28651	A
8	Nuevo receptor	'/ecsh/out/2865133'	

Imagen 62. Nuevo receptor almacenado

Disponemos de dos tipos de receptores, de una salida y de cuatro, para el de una salida en el número de relés deberemos indicar "A". Si nuestro receptor es de cuatro salidas, como se aprecia en la Imagen 63. Configuración receptor cuatro salidas, deberemos copiar cuatro veces el "Topic" del ID del receptor e indicar el número de relé, para el primero una "A", el segundo una "B", el tercero una "C" y para el último una "D".

	B	C	D	E	F	G	H	I
1	ID de periférico de entrada	Número del pulsador	Nombre del botón de la interfaz gráfica	Conexionado entradas y salidas	Nombre del periférico de salida	Topic con ID de periférico de salida	Número relé	
2	00294818	10	Comedor 1		Comedor 1	'/ecsh/out/14211347	A	Cuatro relés
3	00294818	30	Comedor 2		Comedor 2	'/ecsh/out/14211347	B	
4	00294818	50	Persiana Subir		Persiana Subir	'/ecsh/out/14211347	C	
5	00294818	70	Persiana Bajar		Persiana Bajar	'/ecsh/out/14211347	D	
6	01804327	10	Habitacion		Habitacion	'/ecsh/out/2888901'	A	Sensores
7			Exterior		Exterior	'/ecsh/out/28651	A	Estado sólido
8					Nuevo receptor	'/ecsh/out/2865133'		

Imagen 63. Configuración receptor cuatro salidas

Los pulsadores EnOcean siempre tienen la misma numeración de sus pulsadores, se muestra en la Imagen 64. Configuración Pulsador EnOcean, deberemos copiar cuatro veces el ID del pulsador EnOcean y copiar los números de pulsador, en el orden siguiente: "10", "30", "50" y "70".

	B	C	D	E	F	G	H	I
1	ID de periférico de entrada	Número del pulsador	Nombre del botón de la interfaz gráfica	Conexionado entradas y salidas	Nombre del periférico de salida	Topic con ID de periférico de salida	Número relé	
2	00294818	10	Comedor 1		Comedor 1	'/ecsh/out/14211347	A	Cuatro relés
3	00294818	30	Comedor 2		Comedor 2	'/ecsh/out/14211347	B	
4	00294818	50	Persiana Subir		Persiana Subir	'/ecsh/out/14211347	C	
5	00294818	70	Persiana Bajar		Persiana Bajar	'/ecsh/out/14211347	D	
6	01804327	10	Habitacion		Habitacion	'/ecsh/out/2888901'	A	Sensores
7			Exterior		Exterior	'/ecsh/out/28651	A	Estado sólido
8					Nuevo receptor	'/ecsh/out/2865133'		

Imagen 64. Configuración Pulsador EnOcean

8.3.2. Central de procesamiento (Raspberry Pi 3)

Los receptores WIFI, se han diseñado con un funcionamiento con un objetivo concreto, que es el de conmutar su salida y recibir información en el caso de los sensores. Pero la central de procesamiento es el cerebro del sistema, toda la información procedente de los periféricos de entrada y de salida llegan a esta para que se procese. Es decir, en nuestra RPI3 tendremos control absoluto de todas las señales, y es por esto, que la programación queda abierta a modificaciones.

En este proyecto se han configurado los pulsadores de EnOcean para que conmuten la salida de un receptor, pero las acciones que queramos dar a estos después de cada pulsación pueden ser modificada mediante código con la finalidad de realizar lo que el usuario decida. Por ejemplo, en el caso de que no nos encontremos en nuestro hogar, y la central detecte que la temperatura de la vivienda aumenta, podemos programar bajar las persianas cuando sobrepasemos cierta temperatura. De esta forma evitaremos llegar a la vivienda y tener que encender el aire acondicionado. O quizás otra aplicación sería en el caso de detectar una concentración elevada de monóxido de carbono, activar una alarma sonora y encender un sistema de extracción de aire.

Lo que realmente hace interesante a un sistema domótico, es la flexibilidad. Cada usuario puede crear aplicaciones para cada sistema que se le ocurra, marcando sus directrices, con lo cual podríamos decir que, las aplicaciones del sistema son casi infinitas. Obviamente en este apartado no explicaré todas las ideas que como usuario pueda realizar, me limitaré a explicar las funciones básicas para una instalación domótica y de qué forma deberemos modificar el código para que podamos adaptarlo a nuestras necesidades.

He creado un directorio en Github donde encontraremos una imagen del sistema Rasbian completo para introducir a nuestra Raspberry Pi 3, simplemente copiando el contenido en nuestra tarjeta microSD. También encontraremos los programas de Arduino, y un video del sistema en funcionamiento.

https://github.com/azafran85/Easy-Control-Smart-Home.git

Arrancaremos nuestra RPI3 y la conectaremos a la red WLAN, para que la central se conecte automáticamente al sistema. Abrimos Inicio/programación/PyCharm y buscaremos el proyecto ECSH para ejecutarlo.

Explicación del funcionamiento de la central de procesamiento

El código de este dispositivo es quizás la parte más compleja y extensa de este proyecto, pero lo separaré en diferentes partes, con la finalidad de que dentro de su complejidad, sea lo más entendible posible. Me basaré en el diagrama de flujo, una vez lo entendamos, pasaremos al código.

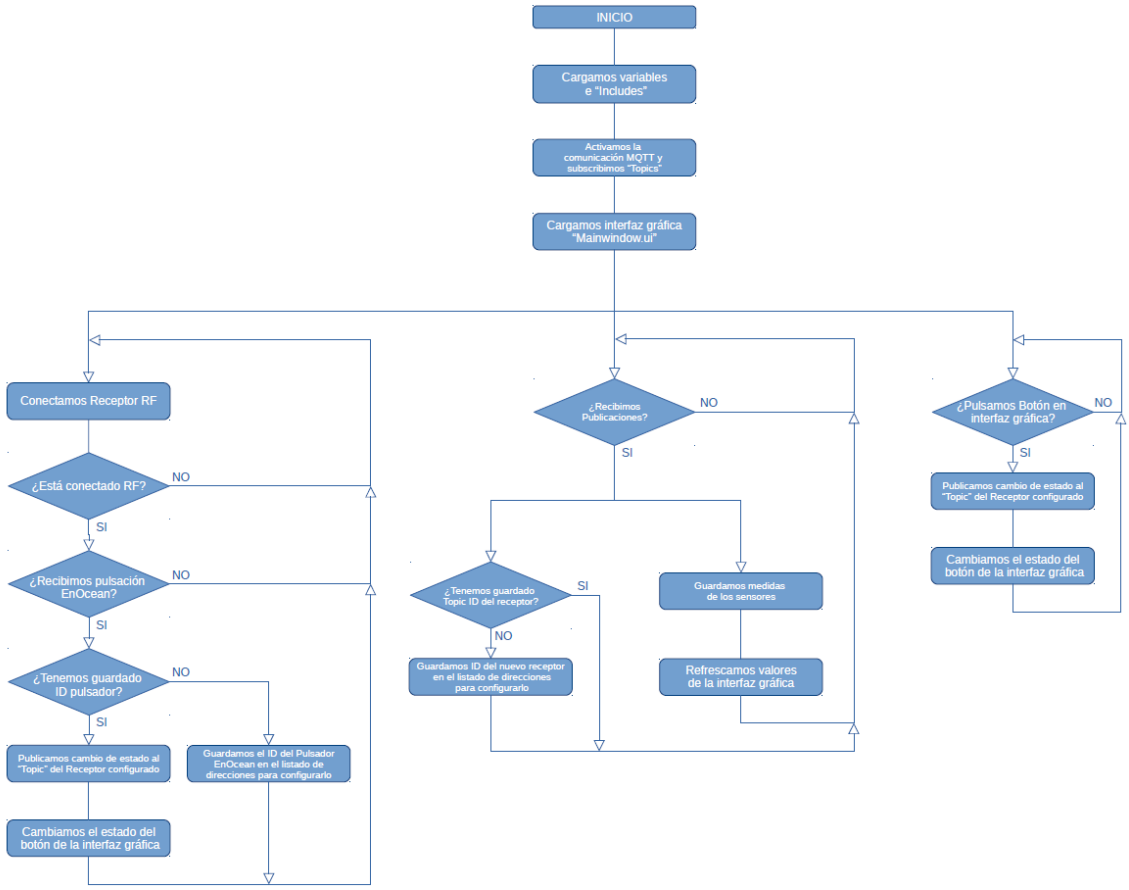


Diagrama 7. Diagrama de flujo de la central de procesamiento

En primer lugar, cargamos las librerías y las variables. Activamos la comunicación MQTT y subscribimos los "Topics" por donde recibiremos toda la información del exterior.

Seguidamente, abrimos la interfaz gráfica, creada anteriormente con el Qt Creator, donde podemos observar el estado de las salidas, las mediciones de los sensores y podemos modificar mediante botones el estado de las salidas.

En este instante se crean tres hilos en paralelo, en primer lugar el hilo de la recepción RF de pulsadores EnOcean, después el hilo de la subscripciones a MQTT por donde recibiremos la información de los sensores y los identificadores de los receptores, y por último con las entradas

de la interfaz gráfica (eventos tipo click de los botones). A continuación entraremos en más detalle para cada uno de los tres hilos.

La primera acción que realizaremos en el hilo de la transmisión RF de los pulsadores de EnOcean, será conectar el dispositivo receptor USB300 a uno de los puertos USB, en el caso de que la conexión no sea posible, intentaremos de nuevo la conexión, hasta que consigamos la conexión. Una vez el sistema de recepción RF esté preparado, quedaremos a la espera a que un pulsador sea activado y comprobaremos en todo momento que la conexión del receptor RF no haya caído. En el momento que recibamos una pulsación, buscaremos el ID que ha generado esta señal en el documento de direcciones. En el caso de que sea la primera vez que lo pulsamos, guardaremos su ID para su configuración. Si ya existe una configuración, publicaremos un cambio de estado al "topic" del receptor que hayamos configurado y cambiaremos el estado en la interfaz gráfica.

Por otro lado tenemos el hilo por donde recibiremos las publicaciones del sistema de comunicación MQTT, recibiremos dos tipos de datos.

El primer tipo de dato que podemos recibir se trata del "topic ID" de los receptores, que está programado para que se envíen una vez estos son alimentados, donde comprobaremos que exista dicho dispositivo en el listado de direcciones. En el caso de que sea la primera vez que conectamos este receptor guardaremos su "topic ID", para que podamos configurarlo, si ya existe no lo guardaremos, y en ambos casos continuaremos comprobando si nos llega otro dato.

El segundo tipo de dato que podemos recibir en la central de procesamiento por MQTT, es la medida de los sensores de temperatura, humedad y concentración de monóxido de carbono. Cada medida dispone de un "topic", por donde recibiremos cada segundo la señal medida y refrescaremos la interfaz gráfica con el nuevo valor.

Po último, tendremos un hilo por el que crearemos un evento tipo click para cada botón, y que en el momento en que pulsemos un botón de la interfaz gráfica, enviaremos un cambio de estado al receptor que hayamos programado anteriormente en el documento de direcciones.

Explicación del funcionamiento del código de la RPI

El código de la central de procesamiento es bastante extenso, pero se ha separado y comentado para facilitar que se entienda correctamente. Como ya he comentado con anterioridad no se pretende explicar todo el código línea por línea, pero si entender que se hace en cada apartado y enfatizar los puntos donde se debe modificar el código para ampliar o cambiar el sistema domótico.

1) Añadimos las librerías y declaramos variables.

```
#!/usr/bin/python
# -*- coding: utf-8 -*-

import serial
import os
import time
import thread
import sys
from PyQt4 import QtGui, uic, QtCore
import RPi.GPIO as GPIO
import paho.mqtt.client as mqtt
from time import strftime
import xlrd
from xlrd import open_workbook
from xlutils.copy import copy

# Initialize GPIO for LED and button.
GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)
```

2) Declaramos cadenas para guardar información de los dispositivos conectados. Crearemos cadenas para el identificador de pulsador EnOcean, el número de pulsador EnOcean, el nombre de los botones de la interfaz gráfica, el identificador de los receptores ("topics") y el número de relé del receptor.

   A continuación, abriremos el documento Excel de DIRECCION.xls y cargaremos en cada cadena los datos del Excel, para poder gestionar cada dispositivo como ya se ha comentado en el documento de direccionamiento en el apartado anterior. Finalmente imprimimos los resultados de las cadenas para detectar fallos.

```
# Direccionamiento de los sistemas configurados en el Excel
id_receptor = []
id_boton = []
id_pulsador = []
n_pulsador = []
n_rele = []

book = xlrd.open_workbook("DIRECCION.xls")
sh = book.sheet_by_index(0)

for fila in range(1, sh.nrows):
    dirId = sh.cell_value(fila, 1)
    id_pulsador.append(dirId)

for fila in range(1, sh.nrows):
    dirId = sh.cell_value(fila, 2)
    n_pulsador.append(dirId)
fila = 0

for fila in range(1, sh.nrows):
    dirId = sh.cell_value(fila, 3)
    id_boton.append(dirId)
fila = 0

for fila in range(1, sh.nrows):
    dirId = sh.cell_value(fila, 6)
    id_receptor.append(dirId)

fila = 0

for fila in range(1, sh.nrows):
    dirId = sh.cell_value(fila, 7)
    n_rele.append(dirId)

for filap in range(fila):
    print id_pulsador[filap], n_pulsador[filap], id_boton[filap], id_receptor[filap],
          n_rele[filap]
fila = 0
filap = 0
```

3) En este apartado del código tenemos la configuración de la comunicación WIFI mediante la librería MQTT, que se basa en el envío la información mediante "topics". Nos encontramos con tres partes de código.

   En primer lugar, la definición de "on_connect", donde nos subscribimos a los "topics" por los cuales recibiremos información, como el ID del receptor, temperatura, humedad, concentración de monóxido de carbono. Y también subscribimos todos los receptores aunque realmente no es necesario porque los sensores y el ID tienen su propio "topic".

   Por otro lado tenemos la definición de "on_message", en esta definición el programa quedará a la espera de recibir información por un "topic" subscrito en el punto anterior. Cuando detecta que llega un dato, comprobamos por el "topic" que lo recibimos y procesamos la información. En el caso de los sensores guardamos el valor medido para mostrarlo posteriormente en la interfaz gráfica. Si el dato recibido es un identificador de un receptor, comprobamos si ya está almacenado en el documento de direcciones, y si no es así lo guardamos como nuevo receptor para configurarlo posteriormente.

   Por último, creamos un cliente para la conexión MQTT, donde activamos las dos definiciones anteriores, creando un hilo de ejecución paralelo, en el diagrama de funcionamiento observamos que tenemos tres hilos en paralelo. Debemos indicar la IP por donde crearemos la comunicación, introduciendo "localhost" automáticamente administrará la IP de nuestra RPI3.

```
############### Comunicación con los receptores y RPI
#####################################
#Creamos canal de comunicación por el cual nos enviaran los id
def on_connect(client, userdata, flags, rc):
    # Subscribing in on_connect() means that if we lose the connection and
    # reconnect then subscriptions will be renewed.
    client.subscribe("/ecsh/out/id")
    client.subscribe("/ecsh/out/temperature")
    client.subscribe("/ecsh/out/humidity")
    client.subscribe("/ecsh/out/concentracion")
    client.subscribe("outTopic")

    #Subscribimos todos los id_receptores (no es necesario porque solo enviaremos el reset
    de salidas)
    for subs in id_receptor:
        client.subscribe(subs)
        client.publish(subs, 'O')

# Creamos canales de comunicación por el cual enviaremos a cada receptor las acciones
def on_message(client, userdata, msg):

    # Datos sensores
    if msg.topic == "/ecsh/out/temperature":
        global temperatura
        temperatura = str(msg.payload)
        #print('temperatura: ' + temperatura)

    if msg.topic == "/ecsh/out/humidity":
        global humedad
        humedad = str(msg.payload)
        #print('Humedad: ' + humedad)

    if msg.topic == "/ecsh/out/concentracion":
        global concentracion
        concentracion = str(msg.payload)
        #print('Concentación CO: ' + concentracion)
```

```
    # ID del cliente
    if msg.topic == '/ecsh/out/id':
        payload = str(msg.payload)
        payload = payload[0:]
        payload = payload[:19]

        # Comprobamos que el cliente no existe y si es así creamos uno nuevo.
        if payload in id_receptor:
            print('Ya existe el receptor: ' + payload)
        else:
            book = xlrd.open_workbook("DIRECCION.xls")
            sh = book.sheet_by_index(0)
            nFilasSave = sh.nrows
            print('Receptor guardado: ' + payload)
            rb = open_workbook('DIRECCION.xls', formatting_info=True)
            wb = copy(rb)
            ws = wb.get_sheet(0)
            ws.write(nFilasSave, 5, 'Nuevo receptor')
            ws.write(nFilasSave, 6, payload)
            wb.save('DIRECCION.xls')


# Creamos nuestro cliente y lo conectamos al localhost
client = mqtt.Client()
client.on_connect = on_connect
client.on_message = on_message  # Decimos que hacer cuando llegue un mensaje
client.connect('localhost', 1883, 60)

# Conectamos MQTT y el proceso de mensages queda en otro hilo (thread)
client.loop_start()

# Main loop to listen for button presses.
print('Esperando acción...')
```

4) Ya tenemos configurada la comunicación MQTT y las direcciones guardadas en las cadenas, procederemos a crear la interfaz gráfica, creando una clase "ventana", de esta forma creamos la ventana en un nuevo hilo y abrimos el archivo "mainwindow.ui" creado con el Qt Creator, le damos un título, un color de fondo y maximizamos la ventana.

   A continuación, nombramos el texto que observaremos en los botones, utilizando el nombre que hayamos indicado en el documento de direcciones para cada botón, y por último creamos los eventos de cada botón y del tiempo para mostrar la fecha, hora y refrescar las medidas de los sensores.

```
########## Entorno grafico#############################################################################
# Creamos el entorno y llamamos al ui creado en QtDesigner
class Ventana(QtGui.QMainWindow):
    def __init__(self):
        super(Ventana, self).__init__()
        uic.loadUi('mainwindow.ui', self)

        self.setWindowTitle('ECSH')
        self.setStyleSheet("background-color: rgb(56, 56, 56); color: rgb(150, 150, 150);")
        self.showMaximized()    # Mostrar la ventana maximizada
        # Creamos eventos de los componentes del entorno grafico
        self.comedor1.setText('Encender ' + id_boton[0])
        self.comedor2.setText('Encender ' + id_boton[1])
        self.subirPersiana.setText('Encender ' + id_boton[2])
        self.bajarPersiana.setText('Encender ' + id_boton[3])
        self.habitacion.setText('Encender ' + id_boton[4])
        self.exterior.setText('Encender ' + id_boton[5])
        self.comedor1.clicked.connect(self.comedorClick1)
        self.comedor2.clicked.connect(self.comedorClick2)
        self.subirPersiana.clicked.connect(self.subirPersianaClick)
        self.bajarPersiana.clicked.connect(self.bajarPersianaClick)
        self.habitacion.clicked.connect(self.habitacionClick)
        self.exterior.clicked.connect(self.exteriorClick)
        self.conf.clicked.connect(self.modo_conf)
        self.show()

        # Reloj
        self.timer = QtCore.QTimer(self)
        self.timer.timeout.connect(self.Time)
        self.timer.timeout.connect(self.Sensor)
        self.timer.start(1000)
```

5) Una vez hemos añadido los dos tipo de evento, explicaremos en que consiste cada uno.

El primero es un evento de tipo tiempo donde refrescaremos cada segundo los datos como la hora, la fecha y el valor medido por los sensores.

El segundo evento, es de tipo "click", es decir, cada vez que pulsemos un botón. En el código que se muestra a continuación sólo tenemos un evento, pero en el código del apéndice podemos observar que tendremos un evento por cada botón de la interfaz gráfica.

En este evento debemos configurar el número de la fila del documento de direcciones, en nuestro caso el comedor 1 corresponde a "i = 0", el comedor 2 será "i = 1" y así consecutivamente con el direccionamiento configurado.

Se ha programado cada evento tipo "click" para que el único dato que debamos cambiar del código sea la fila del documento que queremos relacionar con este botón, modificando la variable "i". En el caso del comedor 1, cuando este sea pulsado enviaremos un cambio de estado al id del receptor y numero relé de la fila seleccionada.

Finalmente, con la finalidad mejorar la interacción del usuario con la interfaz, se mostrará en el botón la acción que realizaremos y el color de la acción que ejecutaremos con esa pulsación. Cuando el receptor se encuentre apagado, leeremos Encender Comedor 1 y el botón se encontrará de color verde. Si el receptor se encuentra encendido, en el botón leeremos Apagar Comedor 1 y el botón se encontrará de color rojo.

```
# -------- Slots -----------------------------------------

    def Time(self):
        sender = self.sender()
        self.hora.setText(strftime("%H" + ":" + "%M" + ":" + "%S  " + "%d" + "/" + "%m" +
        "/" + "%y"))

    def Sensor(self):
            self.temperaturaLabel.setText("Temperatura: " + temperatura + " Celsius")
            self.humedadLabel.setText("Humedad: " + humedad + "% HR")
            self.COLabel.setText("Concentracion CO: " + concentracion + "%")

    def comedorClick1(self):
        i = 0
        sender = self.sender()
        # Send a toggle message to the ESP8266 LED topic.
        client.publish(id_receptor[i], n_rele[i])

        if sender.text() == ('Encender ' + id_boton[i]):
            self.comedor1.setText('Apagar ' + id_boton[i])
            self.comedor1.setStyleSheet("background-color: rgb(158, 2, 4); color: rgb(255,
            175, 175);")
        else:
            self.comedor1.setText('Encender ' + id_boton[i])
            self.comedor1.setStyleSheet("background-color: rgb(0, 89, 1); color: rgb(82,
            255, 38);")
```

6) El tercer hilo de funcionamiento en paralelo, se ha creado para detectar por el receptor RF la actividad de los pulsadores EnOcean. En el cual llamamos a la función "getSerialData", que es la encargada de guardar el id del pulsador y el número de pulsador en "serialData". No explicaré la forma en que se almacena el dato, ya que el código es muy extenso debido a que el Protocolo Serial 3 de EnOcean es bastante extenso y no debemos modificar ninguna parte de este código. En el apéndice se puede observar el programa completo y vemos como se establece la conexión USB al módulo receptor RF y de qué forma de extraer el dato.

   Una vez tenemos el dato "serialData", comprobamos que el ID del pulsador lo tenemos almacenado, en el caso de no tenerlo almacenado lo guardaremos en el documento de direcciones para configurarlo posteriormente. En el caso de que ya exista en el listado, comprobaremos el ID y el número del pulsador, y publicaremos el cambio de estado del relé al receptor que se encuentre direccionado en la misma fila.

   Y quedaremos esperando en el bucle hasta que llegue un nuevo dato de un pulsador EnOcean.

```
def acciones(serialData):
    if serialData[4:12] in id_pulsador:
        print('Ya existe el receptor: ' + serialData[4:12])

    else:
        book = xlrd.open_workbook("DIRECCION.xls")
        sh = book.sheet_by_index(0)
        nFilasSave = sh.nrows
        print('Pulsador guardado: ' + serialData[4:12])
        rb = open_workbook('DIRECCION.xls', formatting_info=True)
        wb = copy(rb)
        ws = wb.get_sheet(0)
        ws.write(nFilasSave, 0, 'Nuevo pulsador')
        ws.write(nFilasSave, 1, serialData[4:12])
        wb.save('DIRECCION.xls')

    if serialData[4:12] == id_pulsador[i]:
        print serialData[2:4]
        if serialData[2:4] == '10':
            client.publish(id_receptor[0], n_rele[0])
        elif serialData[2:4] == '30':
            client.publish(id_receptor[1], n_rele[1])
        elif serialData[2:4] == '50':
            client.publish(id_receptor[2], n_rele[2])
            print (n_pulsador[2], id_receptor[2], n_rele[2])
        elif serialData[2:4] == '70':
            client.publish(id_receptor[3], n_rele[3])
        else:
            ()
    return


########## Thread pulsador enocean ############################################################
def enocean(mensaje):
    while True:
        getSerialData()
        if checkPacketType('01'):
            acciones(serialData)
            printPacketType1(serialData)
```

7) En último lugar tenemos el hilo de la interfaz gráfica donde llamamos a “ventana()”, y desde donde creamos realmente el hilo del receptor RF de EnOcean.

```
if __name__ == '__main__':
    mensaje = ""
    main()
    thread.start_new_thread(enocean, (mensaje,))
    app = QtGui.QApplication(sys.argv)
    window = Ventana()
    sys.exit(app.exec_())
```

Explicación del funcionamiento y configuración de la interfaz gráfica en Qt Creator

El entorno de programación de la interfaz gráfica es amigable y sencillo de utilizar.

Existen muchas opciones para modificar pero básicamente lo único que nos interesa es la forma de añadir o modificar los botones que anteriormente habremos creado en nuestro código de la RPI3 y en el documento de direcciones.

Para modificar el nombre de un botón y que coincida con nuestras direcciones, simplemente abrimos el documento "mainwindow.ui", seleccionamos el botón y cambiamos el nombre a la derecha donde observamos "objectName".

Para crear un nuevo botón, lo más rápido es copiare uno de los que ya están creados, y modificaremos el nombre. De esta forma evitaremos volver a diseñar el botón.

Podremos modificar las características de los botones e incluso la distribución de estos, el tema del diseño y distribución de la interfaz queda abierta dependiendo de cada usuario.

Imagen 65. Modificación de la Interfaz gráfica

9. DISEÑO DE PCB

En el mundo de la electrónica, como en muchos otros, la simulación y el prototipo de un producto queda lejos del producto industrializable final. Se deben efectuar rigurosos controles de calidad y seguridad, hasta poder comercializar un producto.

En este punto quería comentar un poco mi experiencia laboral. He trabajado durante ocho años en el departamento de electrónica de un centro tecnológico, y después de estos años puedo contar con los dedos de las manos, los proyectos que han sido industrializados directamente por el centro. El punto más complejo de la industrialización es, proteger tu producto ante la debilidad de ser copiado. El coste que supone a una empresa es muy elevado, ya solo la creación de la patente internacional significa un desembolso económico elevado. Desde mi punto de vista no tiene mucho sentido una patente a nivel nacional o europeo, teniendo en cuenta que una multinacional tiene sedes en varios países del planeta. Para terminar, mi experiencia en esta empresa me ha permitido a aprender a fabricar mis propios circuitos. Diseñándolo paso a paso, croquis de funcionamiento, búsqueda de componentes, simulación en protoboard, creación del esquemático, diseño del circuito, fabricación del prototipo y finalmente obtenemos un producto listo para chequear su funcionamiento.

La fabricación del prototipo ha sido más sencilla, gracias a la herramienta de fabricación de placas electrónicas (PCBs), del fabricante LPKF, ya que en el departamento de electrónica disponen de un modelo similar a la que utilizaba. Así que me decidí a fabricar mi prototipo y acercar un poco más este proyecto a la industrialización.

A continuación se muestra con imágenes, los pasos seguidos para fabricar este prototipo.

a) Croquis del sistema, normalmente realizado a mano.

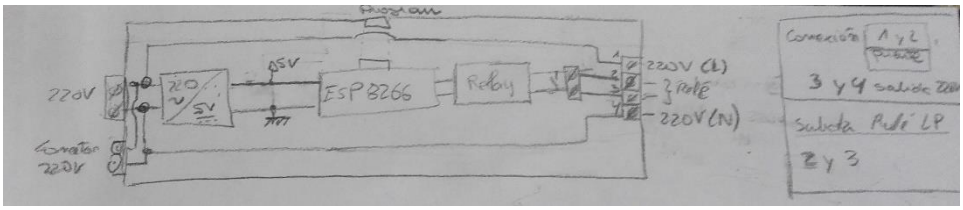


Imagen 66. Croquis del sistema

b) Búsqueda y compra de componentes.

c) Simulación en "protoboard".

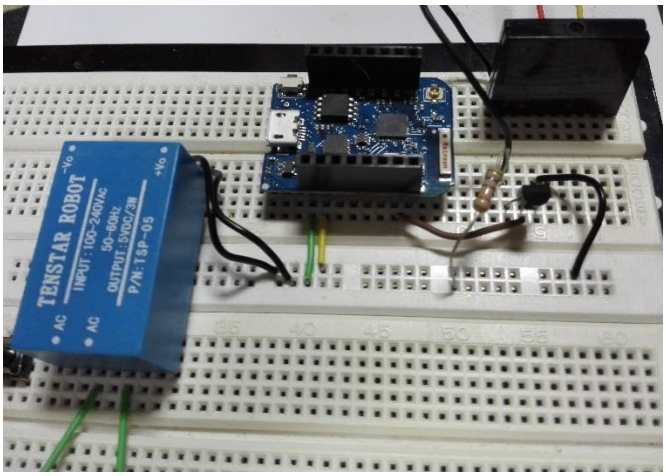


Imagen 67. Sistema simulado en "protoboard"

d) Creación del esquemático (Layout).

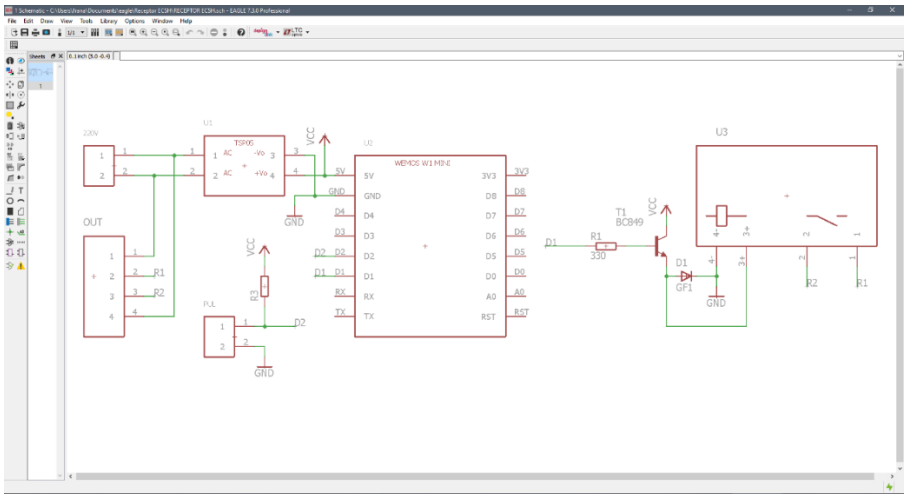


Imagen 68. Diseño del esquemático en Eagle

e) Diseño del circuito (PCB).

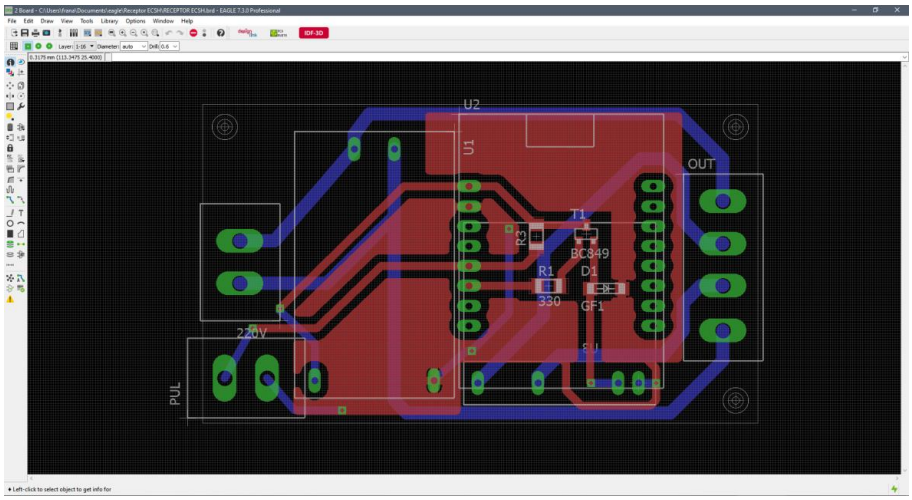


Imagen 69. Diseño de la PCB en Eagle

f) Creamos los "Gebers", conjunto de capas del diseño de la PCB, que se utilizan para que el programa de fabricación entienda los pasos que debe seguir.

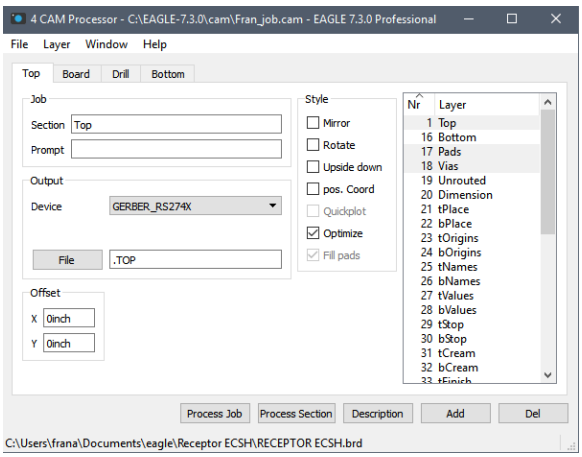


Imagen 70. Creación de Gerbers con Eagle

g) Fabricación del prototipo (LPKF).

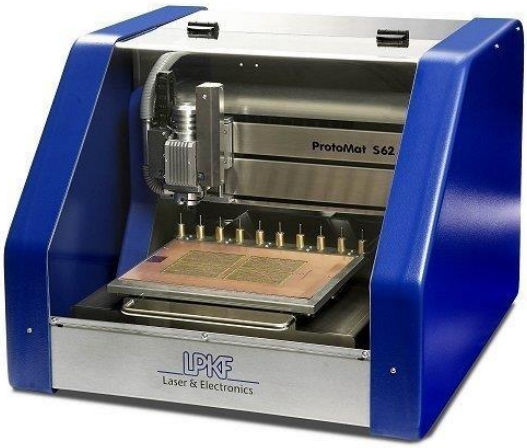


Imagen 71. Máquina de fabricación de PCBs. LPKF.

h) Se colocan y sueldan los componentes.

i) Se comprueba que su funcionamiento sea el correcto.

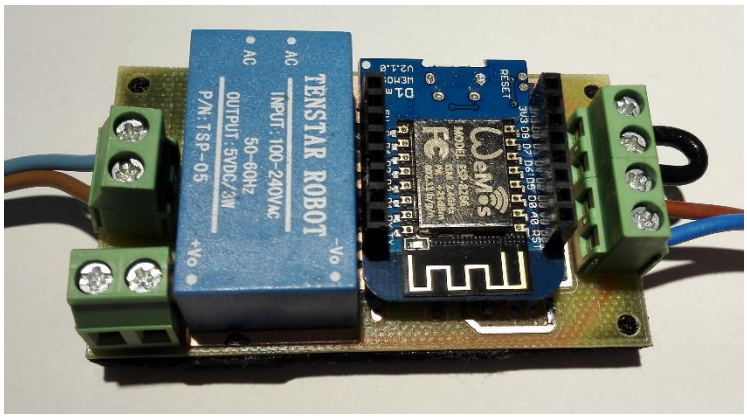


Imagen 72. Prototipo final del receptor WIFI con una salida

10. SISTEMA COMPLETO

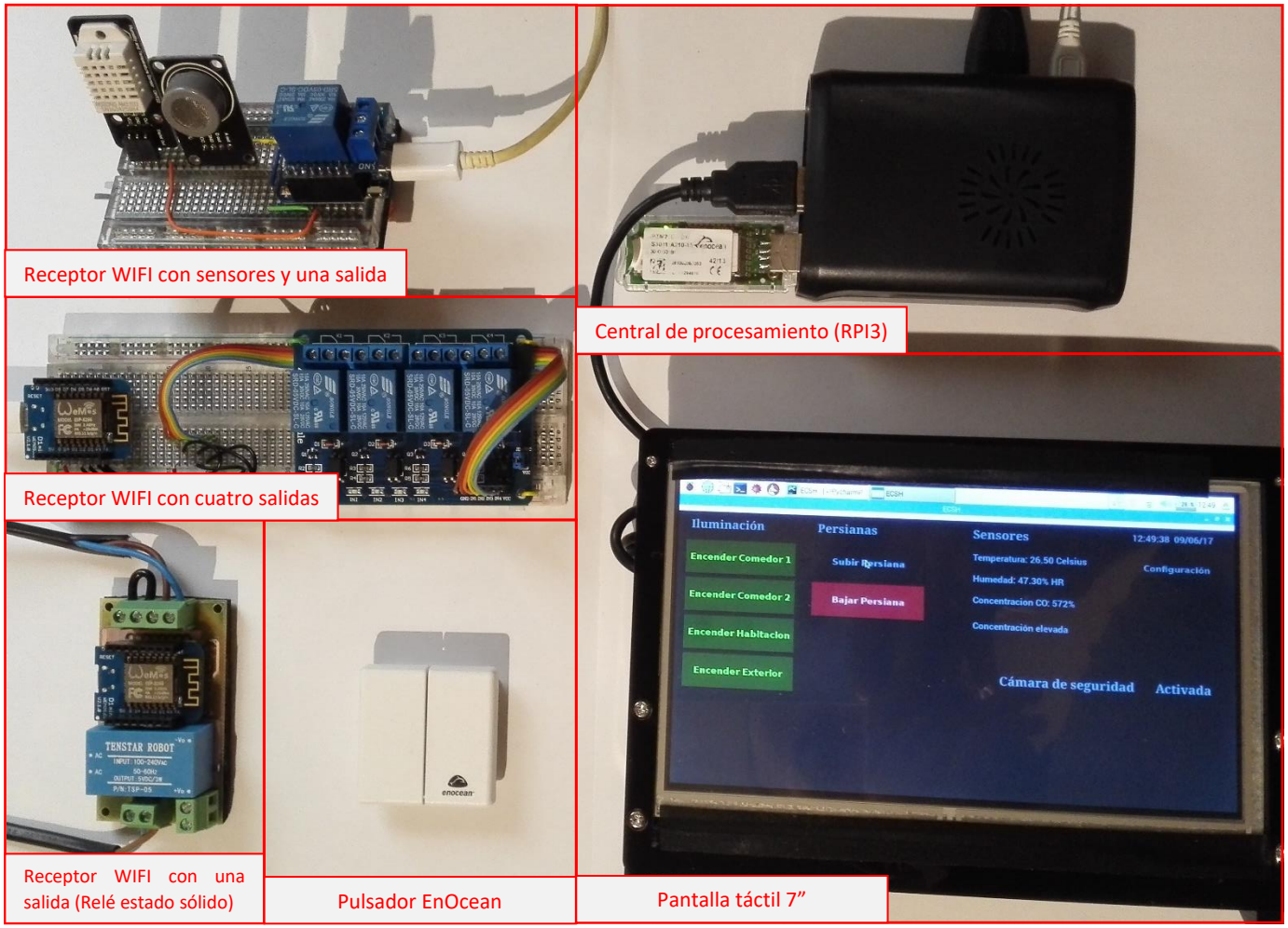


Imagen 73. Hardware que conforman el sistema domótico

Imagen 74.Interfaz gráfica del sistema

11. PRESUPUESTO

En éste apartado, entendemos por sistema domótico el conjunto de controladores, la interfaz gráfica, los receptores WIFI completos, Pulsadores y receptor EnOcean y todos los elementos que permiten la comunicación del sistema.

En la Tabla 2. Precio unitario del hardware tenemos todos los precios unitarios de todos los componentes utilizados para este proyecto, si clicamos en la referencia nos direccionará al lugar web donde se ha comprado cada producto.

Descripción	Referencia	Precio unitario
Raspberry Pi 3 completa	RPI3	42,93 €
Tarjeta microSD	microSD	12,88 €
Pantalla táctil	INNOLUX 7"	27,74 €
Soporte pantalla	AT070TN90	13,36 €
Alimentador Pantalla	AC/DC	8,25 €
Cable HDMI	HDMI	2,10 €
D1 mini pro	D1 mini pro	6,59 €
Fuente de alimentación	TSP-05	2,29 €
Relé electromecánico "Shield"	Shield	3,51 €
Relé electromecánico cuatro canales	4 Relés	4,03 €
Relé de estado sólido	G3MB-202P	0,77 €
Pulsador EnOcean	PTM210	29,39 €
Receptor EnOcean	USB300	32,56 €
Sensor temperatura y humedad	DHT22	5,85 €
Sensor de monóxido de carbono	MQ-7	3,25 €

Tabla 2. Precio unitario del hardware

11.1. Coste de sistemas completos

Sistema de control de procesamiento	Precio
Raspberry Pi 3 completa	42,93 €
Tarjeta microSD	12,88 €
Pantalla táctil	27,74 €
Soporte pantalla	13,36 €
Alimentador Pantalla	8,25 €
Cable HDMI	2,10 €
Receptor EnOcean	32,56 €
Total	**139,82 €**

Tabla 3. Coste de la central de procesamiento

Receptor WIFI una salida relé estado sólido	Precio
D1 mini pro	6,59 €
Fuente de alimentación	2,29 €
Relé de estado sólido	0,77 €
Conectores y componentes	4,00 €
PCB fabricación y soldadura	7,00 €
Total	**20,65 €**

Tabla 4. Coste de un receptor WIFI con salida relé de estado sólido

Receptor WIFI una salida relé electromecánico	Precio
D1 mini pro	6,59 €
Fuente de alimentación	2,29 €
Relé electromecánico "Shield"	3,51 €
Conectores y componentes	4,00 €
PCB fabricación y soldadura	7,00 €
Total	**23,39 €**

Tabla 5. Coste de un receptor con salida relé electromecánico

Receptor WIFI cuatro salidas relés electromecánicos	Precio
D1 mini pro	6,59 €
Fuente de alimentación	2,29 €
Relé electromecánico cuatro canales	4,03 €
Conectores y componentes	4,00 €
PCB fabricación y soldadura	7,00 €
Total	**23,91 €**

Tabla 6. Coste de un receptor con cuatro salidas relés electromecánicos

Receptor WIFI con sensores y una salida relé electromecánico	Precio
D1 mini pro	6,59 €
Fuente de alimentación	2,29 €
Relé electromecánico "Shield"	3,51 €
Sensor temperatura y humedad	5,83 €
Sensor de monóxido de carbono	3,25 €
Conectores y componentes	4,00 €
PCB fabricación y soldadura	7,00 €
Total	**32,47 €**

Tabla 7. Coste de un receptor WIFI con sensores y con salida relé electromecánico

11.2. Presupuesto del sistema domótico de una vivienda

Para poder simular el presupuesto de una vivienda estándar, se ha utilizado la vivienda de la Imagen 75. Vivienda para simulación, supondremos que tenemos una vivienda con pulsadores EnOcean y otra sin estos.

Imagen 75. Vivienda para simulación

El sistema está formado por:

- 1 Central de procesamiento completa.
- 9 puntos de luz.
- 8 pulsadores EnOcean.
- Control persianas.

Presupuesto del sistema domótico de una vivienda con pulsadores Enocean

Sistema domótico completo de una vivienda con pulsadores EnOcean	Precio unitario	Cantidad	Precio
Sistema de la central de procesamiento	139,82 €	1	139,82 €
Receptor WIFI salida relé estado sólido	20,65 €	7	144,55 €
Receptor WIFI cuatro salidas relés electromecánicos	23,91 €	1	23,91 €
Receptor WIFI con sensores y una salida relé electromecánico	32,47 €	1	32,47 €
Pulsador EnOcean	29,39 €	4	117,56 €
Total			**458,31 €**

Tabla 8. Coste de una vivienda domótica completa con pulsadores EnOcean

Presupuesto del sistema domótico de una vivienda sin pulsadores Enocean

Sistema domótico completo de una vivienda con pulsadores EnOcean	Precio unitario	Cantidad	Precio
Sistema de la central de procesamiento	107,26 €	1	107,26 €
Receptor WIFI salida relé estado sólido	20,65 €	7	144,55 €
Receptor WIFI cuatro salidas relés electromecánicos	23,91 €	1	23,91 €
Receptor WIFI con sensores y una salida relé electromecánico	32,47 €	1	32,47 €
Total			**308,19 €**

Tabla 9. Coste de una vivienda domótica completa sin pulsadores EnOcean

Podemos observar en ambos presupuestos que tenemos una vivienda domótica completa por un precio de aproximadamente 500€ con pulsadores EnOcean y por 300€ sin pulsadores.

12. EFICIENCIA ENERGÉTICA

El punto más importante del sistema domótico es su impacto en la gestión de la energía de la vivienda, ya que se propuso como objetivo al principio del proyecto.

Desde un punto de vista social, el ahorro y la eficiencia energética no sólo aseguran el abastecimiento energético y mejoran el medio ambiente, sino que también ayudan a incrementar la competitividad del sector industrial, beneficiando el aumento del Producto Interior Bruto del país.

Un consumo eficiente y responsable, según los datos de la guía práctica publicada en el 2007 por el IDAE (Instituto para la Diversificación y el Ahorro de la Energía), los españoles año tras año aumentamos el consumo de energía. A nivel mundial, al ritmo que actualmente consumimos energía, sólo se tardarán 35 años en duplicar el consumo de energía y menos de 55 años en triplicarlo.

El consumo de energía de las familias españolas supone ya un 30% del consumo total de energía del país, el 18% corresponde al consumo doméstico. Cada hogar es responsable de producir hasta 5 toneladas de CO2 anuales.

La implantación de un sistema domótico supone una gestión inteligente de la energía consumida en la vivienda. El control de la iluminación, climatización, el caudal de agua y los electrodomésticos entre otros, permite un aprovechamiento mayor de los recursos utilizados y por tanto una reducción de tipo energética y por lo tanto de tipo económica.

Los precios de la electricidad, el agua, y los combustibles como el gas natural evolucionan con una tendencia alcista como consecuencia del carácter perecedero de las energías no renovables y el imparable incremento de la intensidad energética (indicador que relaciona el consumo de energía y el Producto Interior Bruto). En los últimos 5 años el precio del gas y la electricidad han aumentado en torno a un 15%.

Un hogar medio en España consume unos 4.000kWh al año. Pero teniendo en cuenta las ventajas de un sistema domótico, provoca ahorros energéticos en torno a un 80% en el consumo energético de iluminación, del 15% al 25% de ahorro en sistemas de climatización, y entre un 5% y un 10% en otros sectores como el agua caliente.

Consumo de los sistemas

Consumo de los sistemas	Consumo (Wh)
Sistema de la central de procesamiento	2,2
Pantalla táctil	2,4
Receptor WIFI salida relé estado sólido	0,4
Receptor WIFI salida relé electromecánico "Shield"	0,35
Receptor WIFI cuatro salidas relés electromecánicos	0,35
Receptor WIFI con sensores y una salida relé electromecánico	1
Pulsador EnOcean	0

Tabla 10. Consumo energético de los sistemas

Consumo del sistema domótico de una vivienda

Consumo del sistema domótico de una vivienda	Consumo unitario (Wh)	Cantidad	Consumo (Wh)
Sistema de la central de procesamiento	2,2	1	2,2
Pantalla táctil	2,4	1	2,4
Receptor WIFI salida relé estado sólido	0,4	7	2,8
Receptor WIFI cuatro salidas relés electromecánicos	0,35	1	0,35
Receptor WIFI con sensores y una salida relé electromecánico	1	1	1
Total			**8,75Wh**

Tabla 11. Consumo del sistema domótico de una vivienda

13. DIAGRAMA DE TRABAJO

TAREAS	INICIO	FINAL
Croquis del proyecto	26-ene	27-ene
Búsqueda exhaustiva de información sobre el control periféricos WIFI	26-ene	31-ene
Búsqueda de información programación Raspberry Pi	26-ene	31-ene
Búsqueda de información ESP8266	26-ene	31-ene
Búsqueda de pantallas táctiles HDMI	31-ene	01-feb
Compra de D1 mini pro y shield relé	01-feb	01-feb
Compra de pantalla táctil HDMI y programar	15-feb	23-feb
Programación Raspberry Pi	24-feb	10-jun
Compra sensor temperatura y humedad, sensor CO	28-feb	28-feb
Programación D1 mini pro para un receptor	08-mar	05-may
Redactar primeros apartados de la memoria	29-abr	30-abr
Programación sensores	05-abr	30-abr
Compra sensor de movimiento, Relés de estado sólido, TSP-05	30-abr	30-abr
Diseño PCB receptor	30-abr	05-may
Soldadura y verificación del funcionamiento de la PCB	09-may	15-may
Programación D1 mini pro para cuatro receptores	22-may	24-may
Acabar de redactar memoria	24-may	10-jun

Diagrama 8. Diagrama de Gantt 1/2

TAREAS	INICIO	FINAL
Croquis del proyecto	26-ene	27-ene
Búsqueda exhaustiva de información sobre el control periféricos WIFI	26-ene	31-ene
Búsqueda de información programación Raspberry Pi	26-ene	31-ene
Búsqueda de información ESP8266	26-ene	31-ene
Búsqueda de pantallas táctiles HDMI	31-ene	01-feb
Compra de D1 mini pro y shield relé	01-feb	01-feb
Compra de pantalla táctil HDMI y programar	15-feb	23-feb
Programación Raspberry Pi	24-feb	10-jun
Compra sensor temperatura y humedad, sensor CO	28-feb	28-feb
Programación D1 mini pro para un receptor	08-mar	05-may
Redactar primeros apartados de la memoria	29-abr	30-abr
Programación sensores	05-abr	30-abr
Compra sensor de movimiento, Relés de estado sólido, TSP-05	30-abr	30-abr
Diseño PCB receptor	30-abr	05-may
Soldadura y verificación del funcionamiento de la PCB	09-may	15-may
Programación D1 mini pro para cuatro receptores	22-may	24-may
Acabar de redactar memoria	24-may	10-jun

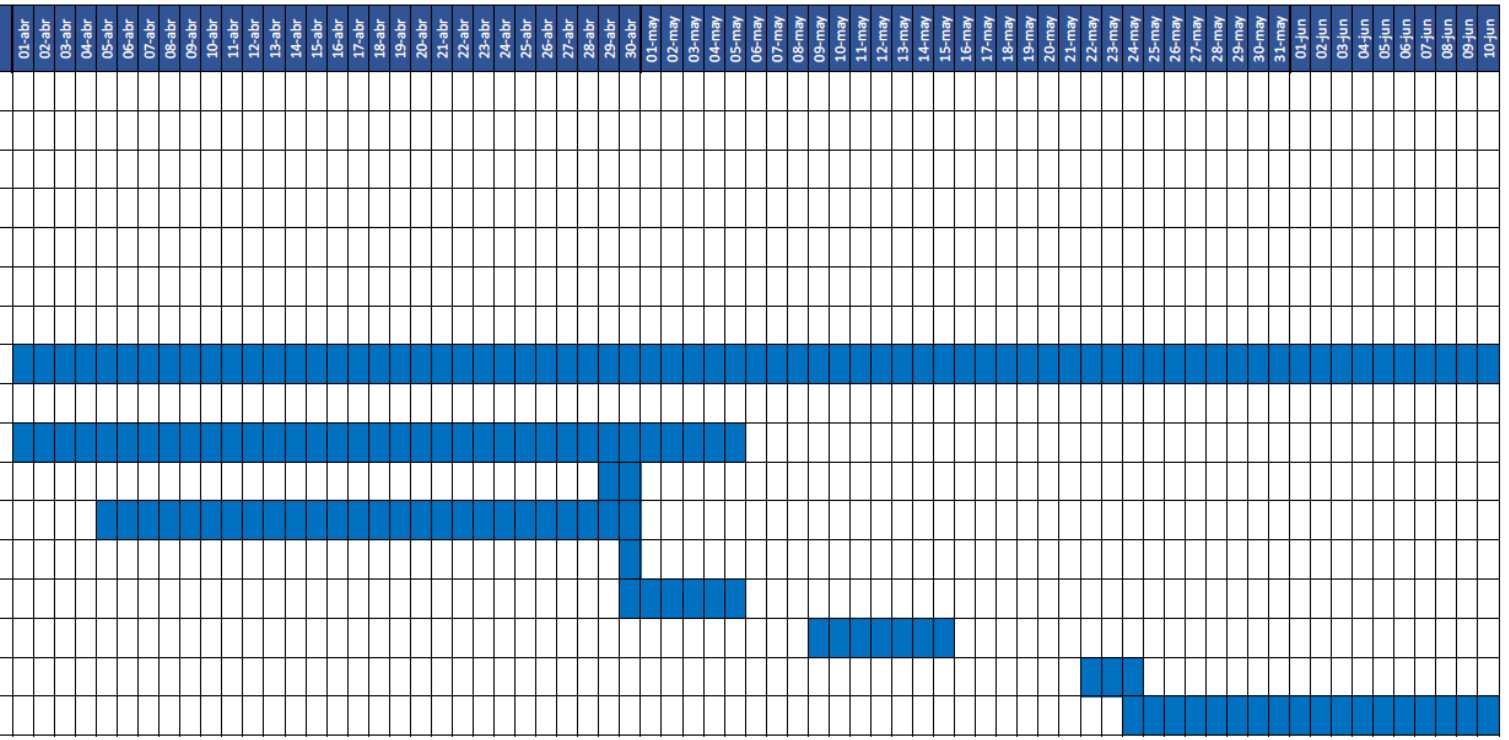


Diagrama 9. Diagrama de Gantt 2/2

14. CONCLUSIONES

Se ha conseguido realizar un sistema domótico, mediante comunicación WIFI, totalmente funcional. Los prototipos que se han fabricado, podrían estar funcionando en un hogar sin ningún inconveniente. Se han adquirido nuevos conocimientos, debido a que la gran mayoría de los sistemas utilizados eran nuevos para mí, desconocía su funcionamiento y su lenguaje de programación.

Las conclusiones que se van a mencionar a continuación, han ido surgiendo durante el transcurso del proyecto, gracias al estudio previo, al desarrollo del sistema y a su funcionamiento final.

En primer lugar, analizaremos los objetivos del proyecto que se propuso que alcanzaría.

- El objetivo de simplificar y abaratar la instalación eléctrica, se ha cumplido, puesto que no necesitaremos cablear más que dos cables de alimentación (220V) por los puntos de luz. No será necesario cablear interruptores o cajas de empalmes.
- Se ha conseguido una comunicación Wireless de todos los sistemas, a través de nuestro Router.
- Se ha alcanzado un coste del sistema domótico completo de entre 300€ y 500€ para un hogar estándar. El precio de un receptor comercial de la marca EnOcean tiene un coste aproximado de 80€ (podemos observar los productos en el apartado EnOcean de la webgrafia), mientras que el diseñado en el proyecto tiene un coste aproximado de 20€.
- El código queda abierto, para que el usuario tenga la posibilidad de programar sus propias aplicaciones.
- Es un proyecto flexible y totalmente modular, debido a que siempre podemos ampliar el número de sistemas de entrado o salida, el pulsador EnOcean es autónomo y en cuanto al receptor, únicamente alimentándolo y conectándole la salida lo tendremos disponible para configurarlo.
- El sistema es rentable, ya que gracias a él no beneficiaremos de un ahorro económico y energético. La inversión económica se verá recuperada en un corto periodo tiempo, con el ahorro energético, que hará que descienda el consumo mensual.
- La interfaz gráfica o el control mediante Smartphone, facilitan situaciones cotidianas a las personas con discapacidades psíquicas o físicas.
- Garantiza una robustez y una fiabilidad, puesto que, la comunicación entre periféricos se basa en una comunicación poco sensible a interferencias, esto no ocurre en otros sistemas de comunicación Wireless.

- El proyecto se puede implementar en viviendas de obra nueva o en instalaciones ya cableadas.

Desde mi punto de vista y como conclusión final, se han alcanzado los objetivos propuestos de manera satisfactoria, y personalmente estoy muy satisfecho del producto final conseguido. He aprendido nuevos conceptos y sistemas de programación.

Para concluir, cuando me encontraba desarrollando una parte del proyecto, me han surgido nuevas ideas y en muchas ocasiones me han hecho cambiar ligeramente el rumbo del proyecto, o incluso ampliarlo o perfeccionarlo. Lo que me hace pensar que se trata de un proyecto que siempre va a tener puntos que mejorar o modificar, ya que cada usuario dicta sus directrices y le permite crear una aplicación a medida sin modificar la instalación.

15. FUTURAS MEJORAS

Después de concluir el proyecto, se exponen algunos de los aspectos a mejorar del sistema domótico. Como ya he comentado anteriormente este proyecto conforme lo vas ejecutando, van apareciendo ideas nuevas, lo que confirma que se trata de un proyecto con amplias posibilidades. Por este motivo han aparecido muchas mejoras, debido a que el tiempo es limitado y realmente el proyecto puede ampliarse todo lo que nuestra imaginación nos permita, pero la base del proyecto facilita dichas ampliaciones o modificaciones.

A continuación voy a mencionar futuras mejoras que podrían dar un valor añadido al sistema:

- El código de Python podría estructurarse y depurarse. En todos los lenguajes de programación, y más si se trata del primer programa que realizamos en este lenguaje, seguramente no será un programa bien estructurado, que obviamente una persona con experiencia mejoraría, pero mi finalidad era que fuese un programa funcional.
- Referente a la programación Python, sería interesante crear un modo configuración dónde podamos realizar la misma función que tiene el documento de direcciones, pero directamente en la interfaz gráfica.
- La creación de una aplicación Android y Apple para el sistema, que funcionen en paralelo. La solución VNC que he utilizado en el proyecto es funcional, pero la creación de una APP, daría un valor añadido al proyecto.
- Diseñar un receptor WIFI de sensores de monitorización de consumos energéticos, para tener un control de qué sistema está consumiendo más y de esta forma tener consciencia y razonar aquello que debemos cambiar para disminuir nuestro consumo.
- Incluso monitorizando señales de consumo, podríamos actuar directamente sobre el aparato que está consumiendo y gestionarlo con el sistema domótico. Además crear un sistema inteligente capaz de interactuar con el entorno, con la finalidad de disminuir el consumo y beneficiarnos de un mayor confort
- Incorporar un reconocimiento por voz, para personas con discapacidades psíquicas o físicas, de esta forma facilitaríamos lo que en muchos casos es un gran problema para este sector.
- Añadir un sistema de vigilancia que se active con un sensor de movimiento, y almacene en el sistema su actividad cuando detecte actividad. También sería interesante gestionar las zonas de transito de la casa con detectores de movimiento, y temporizando su activación. En estas zonas malgastamos innecesariamente energía, temporizando estos sistemas de iluminación, veríamos un ahorro energético considerable.

- Incorporar sensores de luminosidad para controlar la intensidad de luz de los sistemas de iluminación, en el caso de que tengamos luz natural del exterior. Por ejemplo, en el caso de que la temperatura de nuestro hogar aumentará y con este sensor detectemos que tenemos una excesiva exposición a la luz solar, actuar, bajando las persianas para bajar la temperatura interior.
- Crear un receptor WIFI con salida de intensidad regulable, donde podamos modificar la intensidad de nuestros sistemas de iluminación.
- Diseñar un sistema de alarmas, para crear nuestros entornos programados para un momento del día. Por ejemplo, encender la luz progresivamente a la hora que programemos levantarnos y finalmente abra las persianas.
- Añadir energías renovables, y mediante sensores controlar el uso de estas para alimentar nuestro sistema domótico, incluso algún sistema de iluminación.
- Esta mejora la he dejado para la última, ya que me parece muy interesante, pero tiene un riesgo desde mi punto de vista elevado. Se trata de conectar nuestro hogar a internet y poder controlarlo desde cualquier lugar donde tengamos conexión con nuestro Smartphone. El riesgo que veo en esta mejora es que exponemos nuestro hogar a que pueda ser controlado por cualquier hacker informático, y si disponemos de sistema de seguridad puede jugar en nuestra contra, puesto que pueden controlar cuando estamos en el hogar e incluso invadir nuestra privacidad.

Como ya he comentado anteriormente, las capacidades y posibilidades del sistema son muy amplias, y es por esto que este apartado queda abierto a posibles ideas que como usuarios se os puedan ocurrir y aplicar a vuestro sistema.

16. BIBLIOGRAFÍA

- Subspace Identification Methods, De Cock, K.; De Moor, B. (1995). Leuven, Belgium: K. U. Leuven. Department of Electrical Engineering (ESAT- SCD)
- Modeling of Dinamic Sistems. Ljung, L.; Glad, T. (1994). Englewood Ciffs, New Jersey: PTR Prentice Hall. 0-13-597097-D
- Domótica. Edificios Inteligentes. Huidobro, J.M; Millán, R.J. Creaciones Copyright. ISBN 84-933336-9-7.
- Domótica e Inmótica. Viviendas y edificios inteligentes. Romero,C ; Vázquez, F; De Castro, C. Editorial Ra-Ma. ISBN 84-7897-653-1.
- Domótica y Hogar Digital. Junestrand, S; Passaret, X; Vázquez, D. Thomson Paraninfo ISBN 84-283-2891-9.
- Instalaciones automatizadas en Viviendas y edificios. Molina, L; Ruiz, J. Mc-GrawHill. ISBN 84-481-9946-4.
- El Hogar Digital. Fernández, V; Ruz E. Creaciones Copyright. ISBN 84- 96300- 07-2.
- Implementación de aspectos de eficiencia energética en el hogar digital: un caso práctico". Gil Pascual, Miriam. Valencia, 2008.
- Sistemas De Control Para Viviendas Y Edificios Domóticos, José María Quinteiro González, Javier Lamas Graziani, Juan D. Sandoval González.
- Application of Subspace Identification Method in Modeling Multi-zone Buildings. Cai, J.; Kim, D.; Braun, J. (2013). Ray W. Herrick Laboratories, Purdue University.
- Guía técnica de aplicación de instalaciones de sistemas de automatización gestión técnica de energía y seguridad para viviendas y edificios, Ministerio de Industria, Turismo y Comercio. Guía BT-5. Edición Feb. 07. Revisión 1.
- Build Your Own: Smart Home, Robert C. Elsenpeter, Toby J. Velte, ed. McGraw-Hill.
- Subspace System Identification: Theory and applications. Di Ruscio, D. (1995). Porsgrunn, Norway: Telemark Institute of Technology.

17. WEBGRAFÍA

Raspberry Pi 3

Comunicación MQTT

https://learn.adafruit.com/diy-esp8266-home-security-with-lua-and-mqtt/configuring-mqtt-on-the-raspberry-pi (Consultado el 10/01/2017)

https://www.baldengineer.com/mqtt-tutorial.html (Consultado el 14/02/2017)

WeMos

https://www.wemos.cc/product/d1-mini-pro.html (Consultado el 15/03/2017)

http://www.prometec.net/wemos-d1-esp8266-WiFi/ (Consultado el 17/01/2017)

Instalar pluggin de la placa D1 mini pro con ESP8266:

http://www.prometec.net/esp8266-pluggin-arduino-ide/ (Consultado el 05/04/2017)

EnOcean

Web oficial EnOcean:

https://www.EnOcean.com/en/ (Consultado el 22/02/2017)

Web EnOcean Alliance:

https://www.EnOcean-alliance.org/products/ (Consultado el 01/06/2017)

http://www.farnell.com/datasheets/1725395.pdf?_ga=2.243218688.307655482.1495821177-1306401285.1487768282 (Consultado el 25/04/2017)

Receptor comercial de EnOcean

https://www.greenelectric.eu/EnOcean-relay-ABR-152-for-blinds-shutters-with-repeater-function (Consultado el 23/04/2017)

Programación Python

Creación de una aplicación Phyton con interfaz gráfica

http://jquery-manual.blogspot.com.es/2015/08/tutorial-de-python-3-y-pyqt5-interfaz.html (Consultado el 15/01/2017)

Creación lectura y modificación de una hoja de Excel con Python

https://ubuntulife.wordpress.com/2011/09/25/crear-leer-y-modificar-una-hoja-excel-con-python/ (Consultado el 02/06/2017)

Programación Python para Pulsador EnOcean:

http://bryanedelman.com/?p=76 (Consultado el 12/02/2017)

Video tutoriales Python

https://www.youtube.com/playlist?list=PLE549A038CF82905F (Consultado el 05/03/2017)

https://www.youtube.com/channel/UCWIJXDfj7QNvnxb0g6k8-xQ (Consultado el 05/03/2017)

https://www.youtube.com/watch?v=xC9zV3QVqVs&t=7s (Consultado el 05/03/2017)

Sensores

Sensor de temperatura y humedad DHT22

http://rduinostar.com/documentacion/datasheets/dht22-caracteristicas-am2302/

(Consultado el 26/05/2017)

Sensor de concentración de monóxido de carbono

http://www.naylampmechatronics.com/blog/42_Tutorial-sensores-de-gas-MQ2-MQ3-MQ7-y-MQ13.html (Consultado el 26/05/2017)

18. APÉNDICE

18.1. Links apéndices

En la actualidad los "datasheets" están sufriendo una modificación en su formato, han pasado a ser un directorio en la red, donde encontramos una amplia información sobre el dispositivo, como explicaciones, ejemplos, aplicaciones, librerías y más información. Aprovecharemos este sistema de información para algunos elementos.

Podéis encontrar toda la información sobre el proyecto en el github que he creado para el proyecto:

https://github.com/azafran85/Easy-Control-Smart-Home.git

Información sobre la RPI3:

https://github.com/raspberrypi (Consultado el 05/06/2017)

Información sobre la D1 mini pro:

https://github.com/wemos/D1_mini_Examples (Consultado el 08/05/2017)

Información sobre EnOcean:

https://github.com/kipe/enocean (Consultado el 26/01/2017)

Información sobre DHT22:

https://gist.github.com/balloob/1176b6d87c2816bd07919ce6e29a19e9

18.2. Código Arduino receptor WIFI sensores y una salida

```
/*
Rececptor Wifi ESP8266
Creado por: Francesc Agustin Mora
Proyecto Final de Grado: Easy Control Smart Home
*/

//Incluimos librerías

#include <ESP8266WiFi.h>
#include <PubSubClient.h>
#include "DHT.h"

DHT dht;

// Introducimos los datos de nuetra red WIFI e introducimos la IP de la RPI

const char* ssid = "ECSH";
const char* password = "FRANCESCAGUSTIN1234";
const char* mqtt_server = "192.168.1.33";

// Guardamos el ID de nuestra Placa D1 mini pro y lo preparamos para enviar a la RPI

int MQTT_ID = ESP.getChipId();
String add = "'/ecsh/out/";
String add1 = "'";
String clientSub = (add + String(ESP.getChipId()) + add1);
char topicChar[20];

// Creamos el cliente y habilitamos MQTT

WiFiClient espClient;
PubSubClient client(espClient);


// Declaramos entradas y salidas

int Salida1 = D1;
String temp_str;
String hum_str;
String MQ_str;
char temp[50];
char hum[50];
char MQ[50];
long lastMsg = 0;

// Conectamos nuestra Placa D1 mini pro a la red WLAN

void setup_wifi() {

  delay(10);
  WiFi.begin(ssid, password);

  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
  }
  randomSeed(micros());
}

// Inicializamos la lectura en el pin D0 de la Placa

void setup_sensor() {
  dht.setup(D0);
}


// Cuando detectemos que ha llegado un dato, en este caso "T", cambiaremos el estado de la salida

void callback(char* topic, byte* payload, unsigned int length) {

  if ((char)payload[0] == 'A') {
    digitalWrite(Salida1, !digitalRead(Salida1));
  }
  else if ((char)payload[0] == 'O') {
    digitalWrite(Salida1, LOW);
  }
}
```

```
// Únicamente se ejecutará setup en la primera ejucución del programa
// Ejecutamos setup_wifi y llamamos a los métodos setServer y setCallback

void setup() {
  setup_wifi();
  setup_sensor();
  Serial.begin(9600);
  client.setServer(mqtt_server, 1883);
  client.setCallback(callback);

  // Declaramos las Salidas como salidas y la apagamos
  pinMode(Salida1, OUTPUT);
  digitalWrite(Salida1, LOW);
}


// En el caso de que el sistema este desconectado o se haya desconectado se ejecutará reconnect
// Cuando nos conectemos automáticamente enviaremos el id a la RPI

void reconnect() {

  // Solo en el caso de que estemos desconectados entraremos en el loop
  while (!client.connected()) {
    // Intentamos conexión
    String clientId = "ESP8266Client-";
    clientId += String(random(0xffff), HEX);
    if (client.connect(clientId.c_str())) {
      // Pasamos el topic + id a char
      clientSub.toCharArray(topicChar, 20) ;
      Serial.println(topicChar);
      for (int i = 0; i < 9; i++) {
        Serial.print(" 0x");
        Serial.print(topicChar[i], HEX);
      }
      Serial.println();
      // Enviamos id del ESP8266 a la RPI
      client.publish("/ecsh/out/id", topicChar);
      // Subcribimos los topics para recibir datos
      client.subscribe("inTopic");
      client.subscribe(topicChar);

    } else {
      // En el caso de no poder conectarnos esperamos 5 segundos e intentamos de nuevo
      delay(5000);
    }
  }
}
```

```
void sensores() {
  // Lectura del sensor DHT22

  // Obtenemos los datos desde los sensores en el caso de que la temperatura nos entregue una medida
  // Temperatura en °C
  float temperature = dht.getTemperature();

  // Enviamos medidas de los sensores a la RPI siempre que tengamos medida de temperatura
  if (temperature >= -40 && temperature <= 100) {
    long now = millis();
    if (now - lastMsg > 200) {
      lastMsg = now;
      //Serial.println(temperature);
      // Tanto por ciento de humedad
      float humidity = dht.getHumidity();

      // Temperatura en °F
      float ftemperature = dht.toFahrenheit(temperature);

      //Midiendo temperatura y humedad
      temp_str = String(temperature); //convertimos la temperatura de float a string
      temp_str.toCharArray(temp, temp_str.length() + 1); //Empaquetamos la temperatura para publicarla
      hum_str = String(humidity); //convertimos la humedad de float a string
      hum_str.toCharArray(hum, hum_str.length() + 1); //Empaquetamos la humedad para publicarla

      // Medimos la concentración de CO a traves de la entrada analógica
      int adc_MQ = analogRead(A0); //Lemos la salida analógica del MQ
      float voltaje = adc_MQ * (5.0 / 1023.0); //Convertimos la lectura en un valor de voltaje

      MQ_str = String(adc_MQ); //convertimos la temperatura de float a string
      MQ_str.toCharArray(MQ, MQ_str.length() + 1); //Empaquetamos la concentración de CO para publicarla

      client.publish("/ecsh/out/temperature", temp);
      client.publish("/ecsh/out/humidity", hum);
      client.publish("/ecsh/out/concentracion", MQ);
    }
  }
}


// Quedaremos en el loop comprobando que seguimos conectados y leyendo los sensores, hasta que llegue un dato.

void loop() {

  sensores();
  // Comprobamos que seguimos conectados
  if (!client.connected()) {
    reconnect();
  }
  client.loop();
}
```

18.3. Código Arduino receptor WIFI cuatro salidas

```
/*
Rececptor Wifi ESP8266
Creado por: Francesc Agustin Mora
Proyecto Final de Grado: Easy Control Smart Home
*/

//Incluimos librerías

#include <ESP8266WiFi.h>
#include <PubSubClient.h>
#include "DHT.h"

DHT dht;

// Introducimos los datos de nuetra red WIFI e introducimos la IP de la RPI

const char* ssid = "ECSH";
const char* password = "FRANCESCAGUSTIN1234";
const char* mqtt_server = "192.168.1.33";

// Guardamos el ID de nuestra Placa D1 mini pro y lo preparamos para enviar a la RPI

int MQTT_ID = ESP.getChipId();
String add = "'/ecsh/out/";
String add1 = "'";
String clientSub = (add + String(ESP.getChipId()) + add1);
char topicChar[20];

// Creamos el cliente y habilitamos MQTT

WiFiClient espClient;
PubSubClient client(espClient);

// Declaramos entradas y salidas

int Salida1 = D1;
int Salida2 = D2;
int Salida3 = D3;
int Salida4 = D4;

// Conectamos nuestra Placa D1 mini pro a la red WLAN

void setup_wifi() {

  delay(10);
  WiFi.begin(ssid, password);

  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
  }
  randomSeed(micros());
}

// Cuando detectemos que ha llegado un dato, en este caso "T", cambiaremos el estado de la salida

void callback(char* topic, byte* payload, unsigned int length) {

  if ((char)payload[0] == 'A') {
    digitalWrite(Salida1, !digitalRead(Salida1));
  }
  if ((char)payload[0] == 'B') {
    digitalWrite(Salida2, !digitalRead(Salida2));
  }
  if ((char)payload[0] == 'C') {
    digitalWrite(Salida3, !digitalRead(Salida3));
  }
  if ((char)payload[0] == 'D') {
    digitalWrite(Salida4, !digitalRead(Salida4));
  }
  else if ((char)payload[0] == '0') {
    digitalWrite(Salida1, HIGH);
    digitalWrite(Salida2, HIGH);
    digitalWrite(Salida3, HIGH);
    digitalWrite(Salida4, HIGH);
  }
}
```

```
// Únicamente se ejecutará setup en la primera ejucución del programa
// Ejecutamos setup_wifi y llamamos a los métodos setServer y setCallback

void setup() {
  setup_wifi();
  client.setServer(mqtt_server, 1883);
  client.setCallback(callback);

  // Declaramos las Salidas como salidas y la apagamos
  pinMode(Salida1, OUTPUT);
  pinMode(Salida2, OUTPUT);
  pinMode(Salida3, OUTPUT);
  pinMode(Salida4, OUTPUT);
  digitalWrite(Salida1, HIGH);
  digitalWrite(Salida2, HIGH);
  digitalWrite(Salida3, HIGH);
  digitalWrite(Salida4, HIGH);
}


// En el caso de que el sistema este desconectado o se haya desconectado se ejecutará reconnect
// Cuando nos conectemos automáticamente enviaremos el id a la RPI

void reconnect() {

  // Solo en el caso de que estemos desconectados entraremos en el loop
  while (!client.connected()) {
    // Intentamos conexión
    String clientId = "ESP8266Client-";
    clientId += String(random(0xffff), HEX);
    if (client.connect(clientId.c_str())) {
      // Pasamos el topic + id a char
      clientSub.toCharArray(topicChar, 20) ;
      Serial.println(topicChar);
      for (int i = 0; i < 9; i++) {
        Serial.print(" 0x");
        Serial.print(topicChar[i], HEX);
      }
      Serial.println();
      // Enviamos id del ESP8266 a la RPI
      client.publish("/ecsh/out/id", topicChar);
      // Subcribimos los topics para recibir datos
      client.subscribe("inTopic");
      client.subscribe(topicChar);

    } else {
      // En el caso de no poder conectarnos esperamos 5 segundos e intentamos de nuevo
      delay(5000);
    }
  }
}


// Quedaremos en el loop comprobando que seguimos conectados, hasta que llegue un dato.

void loop() {

  if (!client.connected()) {
    reconnect();
  }
  client.loop();
}
```

18.4. Código de la central de procesamiento comentado

```
#!/usr/bin/python
# -*- coding: utf-8 -*-

import serial
import os
import time
import thread
import sys
from PyQt4 import QtGui, uic, QtCore
import RPi.GPIO as GPIO
import paho.mqtt.client as mqtt
from time import strftime
import xlrd
from xlrd import open_workbook
from xlutils.copy import copy

# Initialize GPIO for LED and button.
GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)

# Direccionamiento de los sistemas configurados en el Excel
id_receptor = []
id_boton = []
id_pulsador = []
n_pulsador = []
n_rele = []

book = xlrd.open_workbook("DIRECCION.xls")
sh = book.sheet_by_index(0)

for fila in range(1, sh.nrows):
    dirId = sh.cell_value(fila, 1)
    id_pulsador.append(dirId)

for fila in range(1, sh.nrows):
    dirId = sh.cell_value(fila, 2)
    n_pulsador.append(dirId)
fila = 0

for fila in range(1, sh.nrows):
    dirId = sh.cell_value(fila, 3)
    id_boton.append(dirId)
fila = 0

for fila in range(1, sh.nrows):
    dirId = sh.cell_value(fila, 6)
    id_receptor.append(dirId)

fila = 0

for fila in range(1, sh.nrows):
    dirId = sh.cell_value(fila, 7)
    n_rele.append(dirId)

for filap in range(fila):
    print id_pulsador[filap], n_pulsador[filap], id_boton[filap], id_receptor[filap],
          n_rele[filap]
fila = 0
filap = 0
```

```
################ Comunicación con los receptores y RPI ###################################
#Creamos canal de comunicación por el cual nos enviaran los id
def on_connect(client, userdata, flags, rc):
    # Subscribing in on_connect() means that if we lose the connection and
    # reconnect then subscriptions will be renewed.
    client.subscribe("/ecsh/out/id")
    client.subscribe("/ecsh/out/temperature")
    client.subscribe("/ecsh/out/humidity")
    client.subscribe("/ecsh/out/concentracion")
    client.subscribe("outTopic")

    #Subscribimos todos los id_receptores (no es necesario porque solo    enviaremos el reset
    de salidas)
    for subs in id_receptor:
        client.subscribe(subs)
        client.publish(subs, 'O')

# Creamos canales de comunicación por el cual enviaremos a cada receptor las acciones
def on_message(client, userdata, msg):

    # Datos sensores
    if msg.topic == "/ecsh/out/temperature":
        global temperatura
        temperatura = str(msg.payload)
        #print('temperatura: ' + temperatura)

    if msg.topic == "/ecsh/out/humidity":
        global humedad
        humedad = str(msg.payload)
        #print('Humedad: ' + humedad)

    if msg.topic == "/ecsh/out/concentracion":
        global concentracion
        concentracion = str(msg.payload)
        #print('Concentación CO: ' + concentracion)

    # ID del cliente
    if msg.topic == '/ecsh/out/id':
        payload = str(msg.payload)
        payload = payload[0:]
        payload = payload[:19]

        # Comprobamos que el cliente no existe y si es así creamos uno nuevo.
        if payload in id_receptor:
            print('Ya existe el receptor: ' + payload)
        else:
            book = xlrd.open_workbook("DIRECCION.xls")
            sh = book.sheet_by_index(0)
            nFilasSave = sh.nrows
            print('Receptor guardado: ' + payload)
            rb = open_workbook('DIRECCION.xls', formatting_info=True)
            wb = copy(rb)
            ws = wb.get_sheet(0)
            ws.write(nFilasSave, 5, 'Nuevo receptor')
            ws.write(nFilasSave, 6, payload)
            wb.save('DIRECCION.xls')


# Creamos nuestro cliente y lo conectamos al localhost
client = mqtt.Client()
client.on_connect = on_connect
client.on_message = on_message  # Decimos que hacer cuando llegue un mensaje
client.connect('localhost', 1883, 60)

# Conectamos MQTT y el proceso de mensages queda en otro hilo (thread)
client.loop_start()

# Main loop to listen for button presses.
print('Esperando acción...')
```

```
########## Entorno grafico#############################################################################
# Creamos el entorno y llamamos al ui creado en QtDesigner
class Ventana(QtGui.QMainWindow):
    def __init__(self):
        super(Ventana, self).__init__()
        uic.loadUi('mainwindow.ui', self)

        self.setWindowTitle('ECSH')
        self.setStyleSheet("background-color: rgb(56, 56, 56); color: rgb(150, 150, 150);")
        self.showMaximized()    # Mostrar la ventana maximizada
        # Creamos eventos de los componentes del entorno grafico
        self.comedor1.setText('Encender ' + id_boton[0])
        self.comedor2.setText('Encender ' + id_boton[1])
        self.subirPersiana.setText('Encender ' + id_boton[2])
        self.bajarPersiana.setText('Encender ' + id_boton[3])
        self.habitacion.setText('Encender ' + id_boton[4])
        self.exterior.setText('Encender ' + id_boton[5])
        self.comedor1.clicked.connect(self.comedorClick1)
        self.comedor2.clicked.connect(self.comedorClick2)
        self.subirPersiana.clicked.connect(self.subirPersianaClick)
        self.bajarPersiana.clicked.connect(self.bajarPersianaClick)
        self.habitacion.clicked.connect(self.habitacionClick)
        self.exterior.clicked.connect(self.exteriorClick)
        self.conf.clicked.connect(self.modo_conf)
        self.show()

        # Reloj

        self.timer = QtCore.QTimer(self)
        self.timer.timeout.connect(self.Time)
        self.timer.timeout.connect(self.Sensor)
        self.timer.start(1000)

# -------- Slots ------------------------------------------

    def Time(self):
        sender = self.sender()
        self.hora.setText(strftime("%H" + ":" + "%M" + ":" + "%S  " + "%d" + "/" + "%m" + "/"
        + "%y"))
    def Sensor(self):
            self.temperaturaLabel.setText("Temperatura: " + temperatura + " Celsius")
            self.humedadLabel.setText("Humedad: " + humedad + "% HR")
            self.COLabel.setText("Concentracion CO: " + concentracion + "%")

    # Acciones de los eventos del entorno grafico
    def modo_conf(self):
        print("MODO CONFIGURACIóN")

    def comedorClick1(self):
        i = 0
        sender = self.sender()
        # Send a toggle message to the ESP8266 LED topic.
        client.publish(id_receptor[i], n_rele[i])

        if sender.text() == ('Encender ' + id_boton[i]):
            self.comedor1.setText('Apagar ' + id_boton[i])
            self.comedor1.setStyleSheet("background-color: rgb(158, 2, 4); color: rgb(255,
            175, 175);")
        else:
            self.comedor1.setText('Encender ' + id_boton[i])
            self.comedor1.setStyleSheet("background-color: rgb(0, 89, 1); color: rgb(82, 255,
            38);")

    def comedorClick2(self):
        i = 1
        sender = self.sender()
        # Send a toggle message to the ESP8266 LED topic.
        client.publish(id_receptor[i], n_rele[i])

        if sender.text() == ('Encender ' + id_boton[i]):
            self.comedor2.setText('Apagar ' + id_boton[i])
            self.comedor2.setStyleSheet("background-color: rgb(158, 2, 4); color: rgb(255,
            175, 175);")
        else:
            self.comedor2.setText('Encender ' + id_boton[i])
            self.comedor2.setStyleSheet("background-color: rgb(0, 89, 1); color: rgb(82, 255,
            38);")

    def subirPersianaClick(self):
        i = 2
        sender = self.sender()
        # Send a toggle message to the ESP8266 LED topic.
```

```
        client.publish(id_receptor[i], n_rele[i])

        if sender.text() == ('Encender ' + id_boton[i]):
            self.subirPersiana.setText('Apagar ' + id_boton[i])
            self.subirPersiana.setStyleSheet("background-color: rgb(158, 2, 4); color:
            rgb(255, 175, 175);")
        else:
            self.subirPersiana.setText('Encender ' + id_boton[i])
            self.subirPersiana.setStyleSheet("background-color: rgb(0, 89, 1); color: rgb(82,
            255, 38);")

    def bajarPersianaClick(self):
        i = 3
        sender = self.sender()
        # Send a toggle message to the ESP8266 LED topic.
        client.publish(id_receptor[i], n_rele[i])

        if sender.text() == ('Encender ' + id_boton[i]):
            self.bajarPersiana.setText('Apagar ' + id_boton[i])
            self.bajarPersiana.setStyleSheet("background-color: rgb(158, 2, 4); color:
            rgb(255, 175, 175);")
        else:
            self.bajarPersiana.setText('Encender ' + id_boton[i])
            self.bajarPersiana.setStyleSheet("background-color: rgb(0, 89, 1); color: rgb(82,
            255, 38);")

    def habitacionClick(self):
        i = 4
        sender = self.sender()
        # Send a toggle message to the ESP8266 LED topic.
        client.publish(id_receptor[i], n_rele[i])

        if sender.text() == ('Encender ' + id_boton[i]):
            self.habitacion.setText('Apagar ' + id_boton[i])
            self.habitacion.setStyleSheet("background-color: rgb(158, 2, 4); color: rgb(255,
            175, 175);")
        else:
            self.habitacion.setText('Encender ' + id_boton[i])
            self.habitacion.setStyleSheet("background-color: rgb(0, 89, 1); color: rgb(82,
            255, 38);")

    def exteriorClick(self):
        i = 5
        sender = self.sender()
        # Send a toggle message to the ESP8266 LED topic.
        client.publish(id_receptor[i], n_rele[i])

        if sender.text() == ('Encender ' + id_boton[i]):
            self.exterior.setText('Apagar ' + id_boton[i])
            self.exterior.setStyleSheet("background-color: rgb(158, 2, 4); color: rgb(255,
            175, 175);")
        else:
            self.exterior.setText('Encender ' + id_boton[i])
            self.exterior.setStyleSheet("background-color: rgb(0, 89, 1); color: rgb(82, 255,
            38);")
```

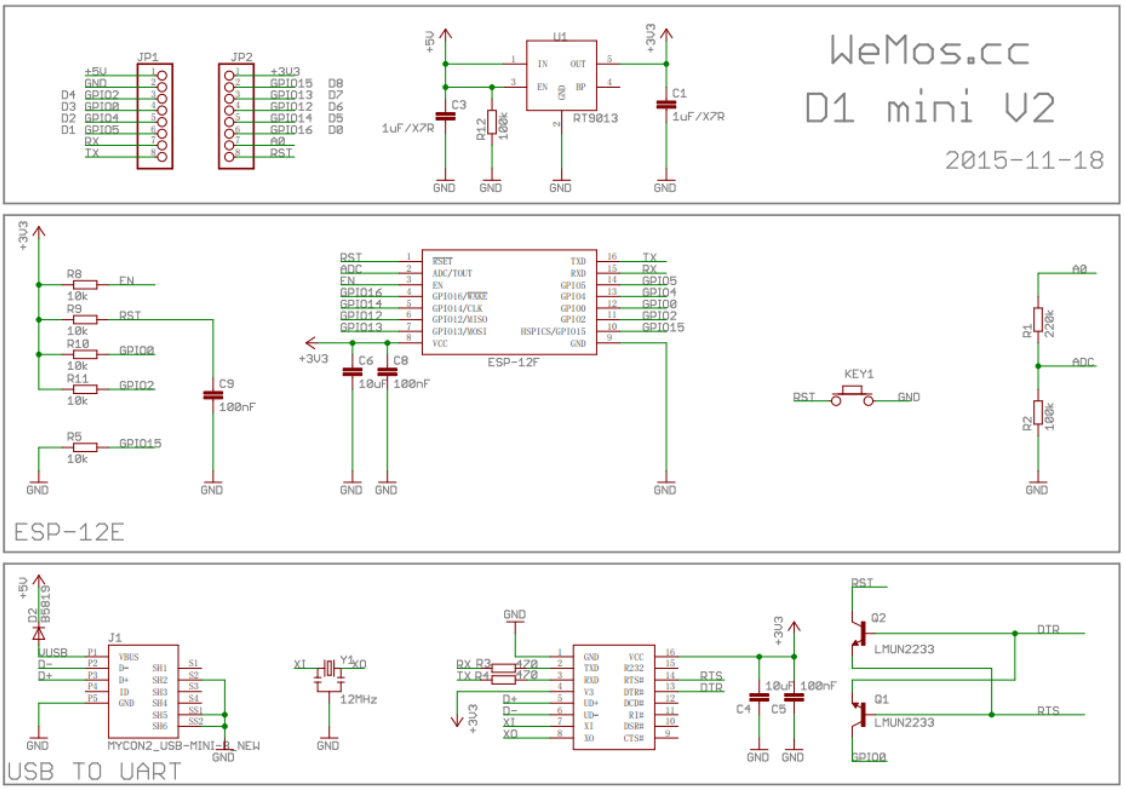
```
###### Enocean ###############################################################################

u8CRC8Table = [
  0x00, 0x07, 0x0e, 0x09, 0x1c, 0x1b, 0x12, 0x15,
  0x38, 0x3f, 0x36, 0x31, 0x24, 0x23, 0x2a, 0x2d,
  0x70, 0x77, 0x7e, 0x79, 0x6c, 0x6b, 0x62, 0x65,
  0x48, 0x4f, 0x46, 0x41, 0x54, 0x53, 0x5a, 0x5d,
  0xe0, 0xe7, 0xee, 0xe9, 0xfc, 0xfb, 0xf2, 0xf5,
  0xd8, 0xdf, 0xd6, 0xd1, 0xc4, 0xc3, 0xca, 0xcd,
  0x90, 0x97, 0x9e, 0x99, 0x8c, 0x8b, 0x82, 0x85,
  0xa8, 0xaf, 0xa6, 0xa1, 0xb4, 0xb3, 0xba, 0xbd,
  0xc7, 0xc0, 0xc9, 0xce, 0xdb, 0xdc, 0xd5, 0xd2,
  0xff, 0xf8, 0xf1, 0xf6, 0xe3, 0xe4, 0xed, 0xea,
  0xb7, 0xb0, 0xb9, 0xbe, 0xab, 0xac, 0xa5, 0xa2,
  0x8f, 0x88, 0x81, 0x86, 0x93, 0x94, 0x9d, 0x9a,
  0x27, 0x20, 0x29, 0x2e, 0x3b, 0x3c, 0x35, 0x32,
  0x1f, 0x18, 0x11, 0x16, 0x03, 0x04, 0x0d, 0x0a,
  0x57, 0x50, 0x59, 0x5e, 0x4b, 0x4c, 0x45, 0x42,
  0x6f, 0x68, 0x61, 0x66, 0x73, 0x74, 0x7d, 0x7a,
  0x89, 0x8e, 0x87, 0x80, 0x95, 0x92, 0x9b, 0x9c,
  0xb1, 0xb6, 0xbf, 0xb8, 0xad, 0xaa, 0xa3, 0xa4,
  0xf9, 0xfe, 0xf7, 0xf0, 0xe5, 0xe2, 0xeb, 0xec,
  0xc1, 0xc6, 0xcf, 0xc8, 0xdd, 0xda, 0xd3, 0xd4,
  0x69, 0x6e, 0x67, 0x60, 0x75, 0x72, 0x7b, 0x7c,
  0x51, 0x56, 0x5f, 0x58, 0x4d, 0x4a, 0x43, 0x44,
  0x19, 0x1e, 0x17, 0x10, 0x05, 0x02, 0x0b, 0x0c,
  0x21, 0x26, 0x2f, 0x28, 0x3d, 0x3a, 0x33, 0x34,
  0x4e, 0x49, 0x40, 0x47, 0x52, 0x55, 0x5c, 0x5b,
  0x76, 0x71, 0x78, 0x7f, 0x6A, 0x6d, 0x64, 0x63,
  0x3e, 0x39, 0x30, 0x37, 0x22, 0x25, 0x2c, 0x2b,
  0x06, 0x01, 0x08, 0x0f, 0x1a, 0x1d, 0x14, 0x13,
  0xae, 0xa9, 0xa0, 0xa7, 0xb2, 0xb5, 0xbc, 0xbb,
  0x96, 0x91, 0x98, 0x9f, 0x8a, 0x8D, 0x84, 0x83,
  0xde, 0xd9, 0xd0, 0xd7, 0xc2, 0xc5, 0xcc, 0xcb,
  0xe6, 0xe1, 0xe8, 0xef, 0xfa, 0xfd, 0xf4, 0xf3
]

def proccrc8(CRC, u8Data):
    return u8CRC8Table[(CRC ^ u8Data) & 0xff];

def printHeader(dataLength,opDataLength,packetType,headerCRC):
    print("Header:",int(dataLength,16),int(opDataLength,16),int(packetType,16),headerCRC)
    return
def printSerialData(serialData):
    print("Serial Data: ", serialData[0:int(dataLength,16)*2])
    return

def printPacketType1(serialData):
    strRorg = "RORG: "
    strRorg += serialData[0:2]
    print strRorg

    strData = "Data: "
    if strRorg[len(strRorg)-2:len(strRorg)] == 'f6':
        strData += serialData[2:4]
        strSrcId = "Source ID: "
        strSrcId += serialData[4:12]
        print strSrcId

    elif strRorg[len(strRorg)-2:len(strRorg)] == 'a5':
        strData += serialData[2:10]
        strSrcId = "Source ID: "
        strSrcId += serialData[10:18]
        print strSrcId
    else:
        ()
    print strData + "\n"
    return

def printOpData():
    return

def checkHeaderCRC():
    u8CRCHeader = 0

    u8CRCHeader = ( proccrc8(u8CRCHeader, int(dataLength,16)>>8))
    u8CRCHeader = ( proccrc8(u8CRCHeader, int(dataLength,16)&0xff))
    u8CRCHeader = ( proccrc8(u8CRCHeader, int(opDataLength,16)))
    u8CRCHeader = ( proccrc8(u8CRCHeader, int(packetType,16)))

    if u8CRCHeader == int(headerCRC,16):
        return True
```

```
    else:
        return False

def checkDataCRC(serialData):
    u8CRCData = 0
    i = 0

    while (i<len(serialData)-2):
        u8CRCData = proccrc8(u8CRCData, (int(serialData[i]+serialData[i+1],16)&0xff))
        i=i+2

    if (u8CRCData == int(serialData[len(serialData)-2]+serialData[len(serialData)-1],16)):
        return True
    else:
        return False

def checkPacketType( x):
        if packetType == x: #x is 'x'
            return True
        else:
            return False

def getSerialData():
    global dataLength, opDataLength, packetType, headerCRC, totalDataLength, serialData
    s = 0
    i = 0
    while s != '55':
        if ser.inWaiting() != 0:
            s = ser.read(1).encode("hex")

    while ser.inWaiting() < 5:
        ()
    dataLength = ser.read(2).encode("hex") #read length field
    opDataLength = ser.read(1).encode("hex") #read op length field
    packetType = ser.read(1).encode("hex") #read packet type field
    headerCRC = ser.read(1).encode("hex") #read header crc field

    if (checkHeaderCRC()):
        totalDataLength = (int(dataLength,16) + int(opDataLength,16))

        while ser.inWaiting() < totalDataLength:
            ()

        serialData = ser.read(totalDataLength+1).encode("hex")
        if checkDataCRC(serialData):
            return serialData
        return "Data CRC Failed"
    return "Header CRC Failed"

def calcESP3HeaderCRC(telegramHeader):
    u8CRC = 0;
    u8CRC = proccrc8(u8CRC,telegramHeader[1])
    u8CRC = proccrc8(u8CRC,telegramHeader[2])
    u8CRC = proccrc8(u8CRC,telegramHeader[3])
    u8CRC = proccrc8(u8CRC,telegramHeader[4])
    return u8CRC

def calcESP3DataCRC(telegramData):
    u8CRC = 0;
    for index in range(len(telegramData)):
        u8CRC = proccrc8(u8CRC,telegramData[index])

    return u8CRC


def calcESP3Header(packetType,packetData, *arg): #assumes 0 optional data
    pHeader = [0x55] #sync
        #for now we support max of 255 byte packets
    pHeader.append(0x00) #MSB Data Length
    pHeader.append(len(packetData))#LSB Data Length
    if len(arg) == 0:
        pHeader.append(0x00) #optional data length
    else:
        pHeader.append(arg[0])
    pHeader.append(packetType) #packet type
    pHeader.append(calcESP3HeaderCRC(pHeader))# Header CRC
    return pHeader

def sendESP3Packet(packetType, packetData):
    pESP3Packet = calcESP3Header(packetType,packetData)
    pESP3Packet += packetData
    pESP3Packet.append(calcESP3DataCRC(packetData))
```

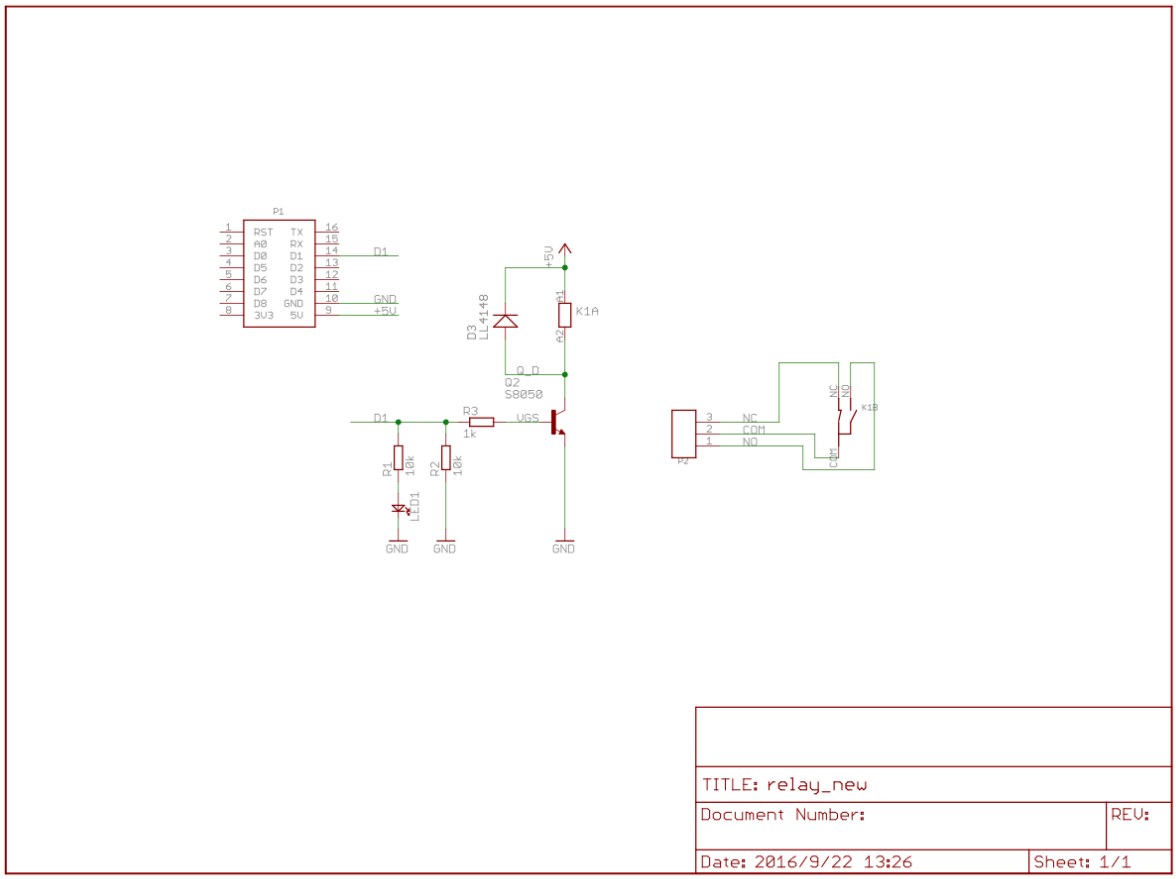
```
    for index in range(len(pESP3Packet)):
        pESP3Packet[index] = chr(pESP3Packet[index])
        #byte by byte tx
        ser.write(pESP3Packet[index])
    return getSerialData()

def sendTest():
    ser.write("\x55\x00\x07\x00\x01\x11\xD5\x55\x00\x00\x00\x00\x80\x5A")

#common commands

def commandReadVersion():
    returnData = sendESP3Packet(0x05, [0x03])

    if int(returnData[0:1],16) == 0x00:
        tmpStr = "Application Version: " + str(int(returnData[2:4],16)) \
        + '.' + str(int(returnData[4:6],16)) + '.' + str(int(returnData[6:8],16)) \
        + '.' + str(int(returnData[8:10],16))
        print tmpStr

        tmpStr = "API Version: " + str(int(returnData[10:12],16)) \
        + '.' + str(int(returnData[12:14],16)) + '.'+ str(int(returnData[14:16],16))+ '.'+
        str(int(returnData[16:18],16))
        print tmpStr

        print "Chip ID:",returnData[18:26]

        i = 34
        tmpStr = "Application: "
        while chr(int(returnData[i]+returnData[i+1],16)) != "\0":
            tmpStr+=(chr(int(returnData[i]+returnData[i+1],16)))
            i = i + 2
        print tmpStr

        return True
    else:
        print "Response Error:" + returnData[0]
        return False


def commandReadBaseId():
    returnData = sendESP3Packet(0x05, [0x08])

    if int(returnData[0:1],16) == 0x00:
        tmpStr = "BaseId: "
        i = 2
        while i < 10:
            tmpStr += returnData[i]+returnData[i+1]
            i = i + 2
        print tmpStr

        tmpStr = "Base Id Writes Remaining: "
        tmpStr += returnData[10]+returnData[11]
        print tmpStr
    else:
        print "Response Error:" + returnData[0]


def main():
    global ser
    print "Welcome to ESP3 with python\n"

    os.system("python -m serial.tools.list_ports \n")
    ser = serial.Serial(port="/dev/ttyUSB0", baudrate=57600, timeout=0)  # open serial port

    if commandReadVersion() != True:
        print "Communication not successful - exiting"
        sys.exit(0)
    commandReadBaseId()

def acciones(serialData):
    if serialData[4:12] in id_pulsador:
        print('Ya existe el receptor: ' + serialData[4:12])

    else:
        book = xlrd.open_workbook("DIRECCION.xls")
        sh = book.sheet_by_index(0)
        nFilasSave = sh.nrows
        print('Pulsador guardado: ' + serialData[4:12])
        rb = open_workbook('DIRECCION.xls', formatting_info=True)
        wb = copy(rb)
```

```
        ws = wb.get_sheet(0)
        ws.write(nFilasSave, 0, 'Nuevo pulsador')
        ws.write(nFilasSave, 1, serialData[4:12])
        wb.save('DIRECCION.xls')

    i = 0
    if serialData[4:12] == id_pulsador[i]:
        print serialData[2:4]
        if serialData[2:4] == '10':
            client.publish(id_receptor[0], n_rele[0])
        elif serialData[2:4] == '30':
            client.publish(id_receptor[1], n_rele[1])
        elif serialData[2:4] == '50':
            client.publish(id_receptor[2], n_rele[2])
            print (n_pulsador[2], id_receptor[2], n_rele[2])
        elif serialData[2:4] == '70':
            client.publish(id_receptor[3], n_rele[3])
        else:
            ()
    return


########## Thread pulsador enocean ##########################################################
def enocean(mensaje):
    while True:
        getSerialData()
        if checkPacketType('01'):
            acciones(serialData)
            printPacketType1(serialData)

if __name__ == '__main__':
    mensaje = "1234"
    main()
    thread.start_new_thread(enocean, (mensaje,))
    app = QtGui.QApplication(sys.argv)
    window = Ventana()
    sys.exit(app.exec_())
```

18.5. Esquemático D1 mini pro



18.6. Esquemático Shield D1



18.7. Datasheet Relé estado sólido GM3B

Solid State Relay
G3MB

Low cost Subminiature PCB mounting
2 amp Single in-line package (SIP) SSR

- Bottom is approximately 3 times smaller than G3M.
- Low cost "SIP" package switches up to 2A loads.
- Built in Snubber circuit and input resistor as option.
- Two footprints available for design flexibility.
- The G3MB-202PEG-4-DC20MA crosses directly to the Motorola M0C2A-60 series power triac.

Ordering Information

NOT FOR NEW DESIGN. Discontinuation planned for April, 2010.

To Order: Specify input voltage at end of part number. Example: G3MB-202P-*DC24*

Isolation	Output terminal pitch	Zero cross	Input resistor	Built-in snubber circuit	Rated output load	Rated input voltage	Model
Phototriac	7.62 mm	Yes	Yes	Yes	2 A at 100 to 240 VAC	5 VDC	G3MB-202P
						12 VDC	
						24 VDC	
		No			2 A at 100 to 240 VAC	5 VDC	G3MB-202PL
						12 VDC	
						24 VDC	
	5.08 mm	Yes			2 A at 100 to 240 VAC	5 VDC	G3MB-202P-4
						12 VDC	
						24 VDC	
		No			2 A at 100 to 240 VAC	5 VDC	G3MB-202PL-4
						12 VDC	
						24 VDC	
		Yes	No	No	2 A at 100 to 240 VAC	N/A *(See Note)	G3MB-202PEG-4-DC20MA
		No			2 A at 100 to 240 VAC	N/A *(See Note)	G3MB-202PLEG-4-DC20MA

Note: 1. For versions without input voltage specified, a current limiting resistor must be placed in series with the input. See LED drive specifications and recommendations.

2. TUV versions available. When ordering models certified by VDE (TUV), add "-UTU" to the model number given in the above table.

Specifications

■ Input Rating

Models with Input Resistor

Rated voltage	Operating range	Input impedance (-UTU Models)	Voltage Levels	
			Must operate voltage	Must release voltage
5 VDC	4 to 6 VDC	440 Ω ±20% (300 Ω ±20%)	4 VDC max.	1 VDC min.
12 VDC	9.60 to 14.40 VDC	1k Ω ±20% (750 Ω ±20%)	9.6 VDC max.	
24 VDC	19.20 to 28.80 VDC	2.20k Ω ±20% (1.6 kΩ ±20%)	19.2 VDC max.	

Models without Input Resistor

Input specifications	Operating characteristics			
Rated current	Continuous current	Must operate current	Must release current	Operating current
20 mA DC	20 mA DC	7 mA DC max.	1 mA DC min.	7 to 20 mA

LED forward current	50 mA max.
Repetitive peak LED forward current	1 A max.
LED reverse voltage	5 V max.

■ Recommended LED Operating Conditions

Models without Input Resistor

	Min.	Standard	Max.
LED forward current	5 mA	10 mA	20 mA
Must drop voltage	0	—	1 V

■ Output Rating

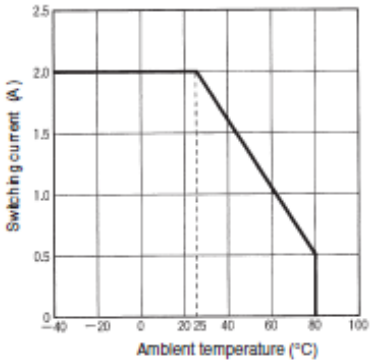
Model	Rated load voltage	Load voltage range	Load current	Surge current
G3MB-202	100 to 240 VAC, 50/60 Hz	75 to 264 VAC	0.10 to 2 A	30 A (60 Hz, 1 cycle)

■ Characteristics

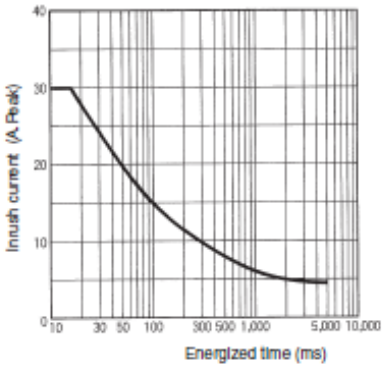
Type		G3MB-202P G3MB-202PEG	G3MB-202PL G3MB-202PLEG
Operate time		1/2 of load power source cycle + 1 ms max.	1 ms max.
Release time		1/2 of load power source cycle + 1 ms max.	
Output ON voltage drop		1.60 V (RMS) max.	
Leakage current		1.50 mA at 200 VAC	
Non-repetitive peak surge		30 A	
Output	PIV (V_{DRM})	600 V	
	di/dt	40 A/μs	
	dv/dt	100 V/μs	
	I^2t	4 A^2s	
Junction temperature (Tj)		125°C (257°F) max.	
Insulation resistance		1,000 MΩ min. at 500 VDC	
Dielectric strength		2500 VAC, 50/60 Hz for 1 minute	
Vibration	Malfunction	10 to 55 Hz, 0.75 mm (0.03 in) double amplitude	
Shock	Malfunction	Approx. 1,000 m/s² (approx. 100 G)	
Ambient temperature	Operating	-30° to 80°C (-22° to 176°F) with no icing or condensation	
	Storage	-30° to 100°C (-22° to 212°F) with no icing or condensation	
Humidity	Operating	45% to 85% RH	
Weight		Approx. 5 g (0.18 oz)	

■ Characteristic Data

Load current vs. ambient temperature characteristics



Inrush current resistivity
One cycle, non-repetitive (Keep the inrush current to half the rated value if it occurs repetitively.)

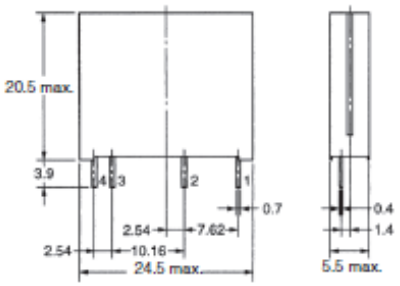


Dimensions

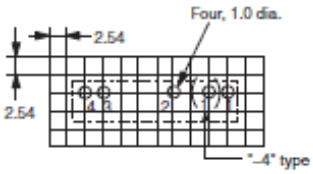
Unit: mm (inch)

■ Relays

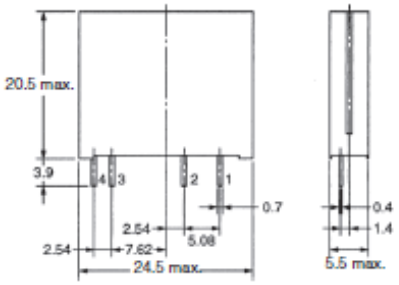
Models without "-4"



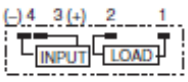
PCB Dimensions (Bottom View)



Models with "-4"



Terminal Arrangement/ Internal Connections (Bottom View)



18.8. Pantalla Táctil INNOLUX

INNOLUX

Date :2010/03/22
Page:1/18

1. General Specifications

No.	Item	Specification	Remark
1	LCD size	7.0 inch(Diagonal)	
2	Driver element	a-Si TFT active matrix	
3	Resolution	800 × 3(RGB) × 480	
4	Display mode	Normally White, Transmissive	
5	Dot pitch	0.0642(W) × 0.1790(H) mm	
6	Active area	154.08(W) × 85.92(H) mm	
7	Panel size	162.5(W) ×96.62(H) ×1.43(D) mm	Note 1
8	Surface treatment	Anti-Glare	
9	Color arrangement	RGB-stripe	
10	Display Color	16.7M	
11	Interface	Digital, Parallel 8-bit RGB	
12	Panel power consumption	0.226W (Typ.)	
13	Weight	45g(Typ.)	

Note 1: Refer to Mechanical Drawing.

3. Operation Specifications

3.1. Absolute Maximum Ratings

(Note 1)

Item	Symbol	Values		Unit	Remark
		Min.	Max.		
Power voltage	DV_{DD}	-0.3	5.0	V	
	AV_{DD}	6.5	13.5	V	
	V_{GH}	-0.3	40.0	V	
	V_{GL}	-20.0	0.3	V	
	V_{GH}-V_{GL}	-	40.0	V	
Operation Temperature	T_{OP}	-20	70	℃	
Storage Temperature	T_{ST}	-30	80	℃	

Note 1: The absolute maximum rating values of this product are not allowed to be exceeded at any times. Should a module be used with any of the absolute maximum ratings exceeded, the characteristics of the module may not be recovered, or in an extreme case, the module may be permanently destroyed.

Note 1: Definition of viewing angle range

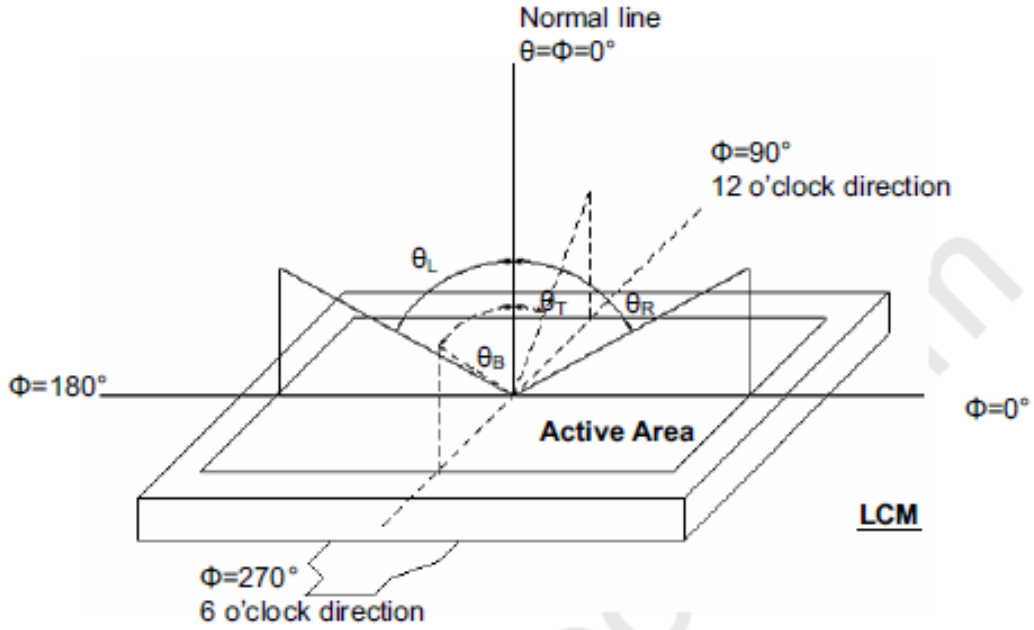


Fig. 4-1 Definition of viewing angle

Note 2: Definition of optical measurement system.
The optical characteristics should be measured in dark room. After 30 minutes operation, the optical properties are measured at the center point of the LCD screen. (Response time is measured by Photo detector TOPCON BM-7, other items are measured by BM-5A/Field of view: 1° /Height: 500mm.)

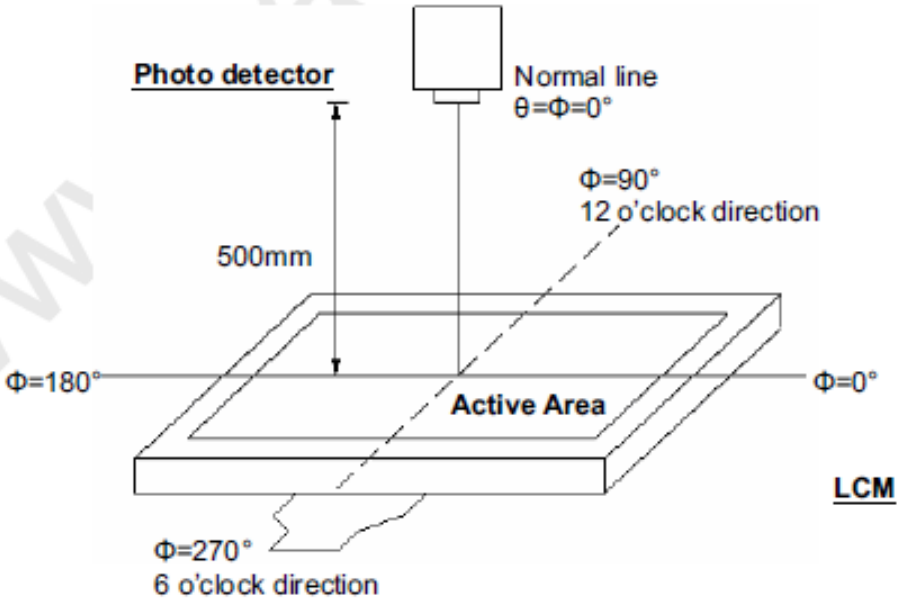


Fig. 4-2 Optical measurement system setup

18.9. Sensor monóxido de carbono MQ-7

Winsen 炜盛科技 Zhengzhou Winsen Electronics Technology Co., Ltd www.winsensor.com

MQ-7 Semiconductor Sensor for Carbon Monoxide

Profile

Sensitive material of MQ-7 gas sensor is SnO_2, which with lower conductivity in clean air. It make detection by method of cycle high and low temperature, and detect CO at low temperature(heated by 1.5V).The sensor's conductivity gets higher along with the CO gas concentration rising. At high temperature(heated by 5.0V),it cleans the other gases adsorbed at low temperature. Users can convert the change of conductivity to correspond output signal of gas concentration through a simple circuit.

Features

It has good sensitivity to carbon monoxide in wide range, and has advantages such as long lifespan, low cost and simple drive circuit &etc.

Main Applications

It is widely used in domestic CO gas leakage alarm, industrial CO gas alarm and portable CO gas detector.

Technical Parameters **Stable.1**

Model			MQ-7
Sensor Type			Semiconductor
Standard Encapsulation			Plastic cap
Target Gas			carbon monoxide
Detection range			10~500ppm CO
Standard Circuit Conditions	Loop Voltage	V_c	≤10V DC
	Heater Voltage	V_H	5.0V±0.1V AC or DC（High tem.） 1.5V±0.1V AC or DC（Low tem.）
	Heater Time	T_L	60 S±1S（High tem.）， 90 S±1S（Low tem.）
	Load Resistance	R_L	Adjustable
Sensor character under standard test conditions	Heater Resistance	R_H	29Ω±3Ω（room tem.）
	Heater consumption	P_H	≤900mW
	Sensitivity	S	Rs(in air)/Rs(in 150ppm CO)≥5
	Output Voltage	Vs	2.5V~4.3V (in 150ppm CO)
	Concentration Slope	α	≤0.6(R_{300ppm}/R_{50ppm} CO)
Standard test conditions	Tem. Humidity		20℃±2℃; 55%±5%RH
	Standard test circuit		Vc:5.0V±0.1V; V_H（High tem.）: 5.0V±0.1V; V_H（Low tem.）: 1.5V±0.1V
	Preheat time		Over 48 hours

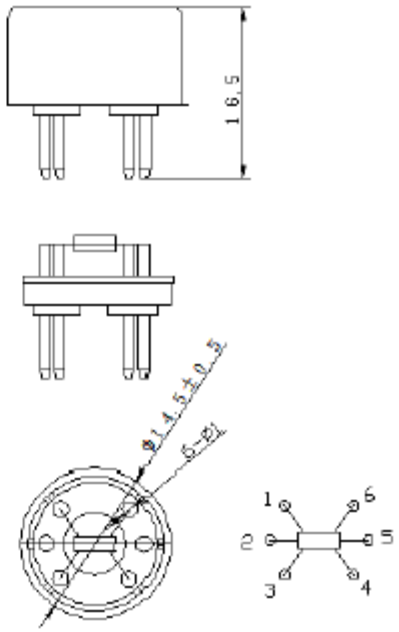


Fig1.Sensor Structure

Unit: mm

NOTE: Output voltage (Vs) is V_{RL} in test environment.

Tel: 86-371-67169097/67169670 Fax: 86-371-60932988 Email: sales@winsensor.com

Basic Circuit

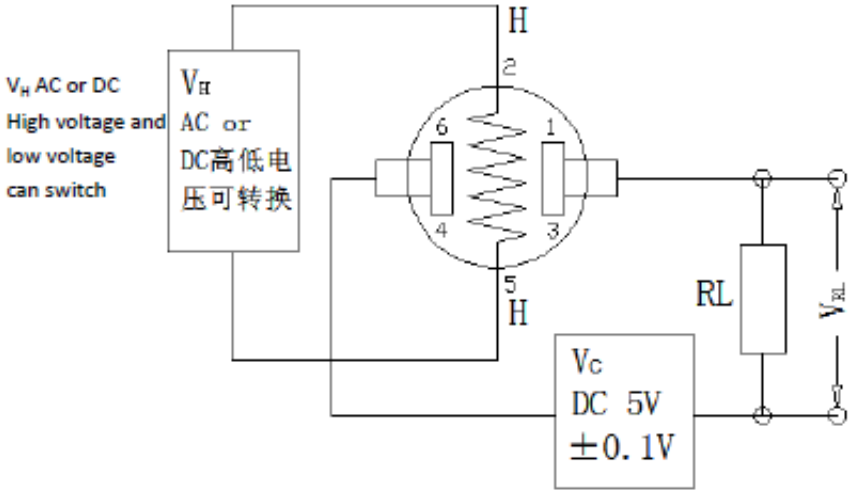


Fig2. MQ-7 Test Circuit

Instructions: The above fig is the basic test circuit of MQ-7.The sensor requires two voltage inputs: heater voltage (V_H) and circuit voltage(V_C). V_H is used to supply standard working temperature to the sensor and it can adopt DC or AC power. For this model sensor, V_H should be at 1.5V±0.1V low voltage when detect CO while should be at 5V±0.1V at non detection status(resuming period). V_{RL} is the voltage of load resistance R_L which is in series with sensor. Vc supplies the detect voltage to load resistance R_L and it should adopts DC power.

Description of Sensor Characters

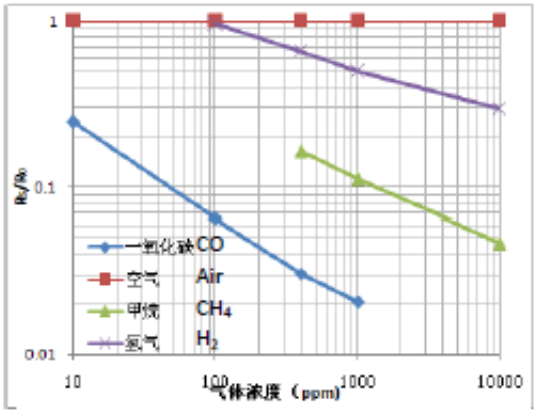


Fig3.Typical Sensitivity Curve

The ordinate is resistance ratio of the sensor (R_s/R_0), the abscissa is concentration of gases. Rs means resistance in target gas with different concentration, R_0 means resistance of sensor in clean air. All tests are finished under standard test conditions.

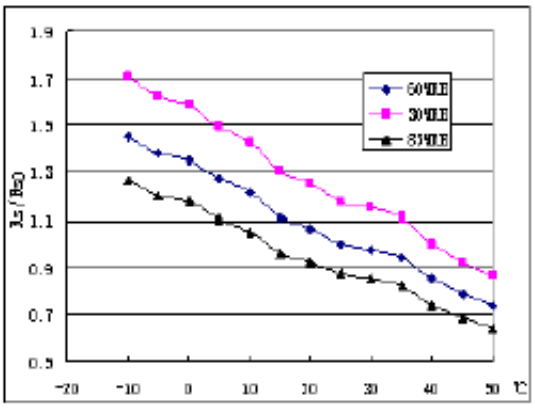


Fig4.Typical temperature/humidity characteristics

The ordinate is resistance ratio of the sensor (Rs/Rso).Rs means resistance of sensor in 150ppm CO gas under different tem. and humidity. Rso means resistance of the sensor in 150ppm CO gas under 20℃/55%RH.

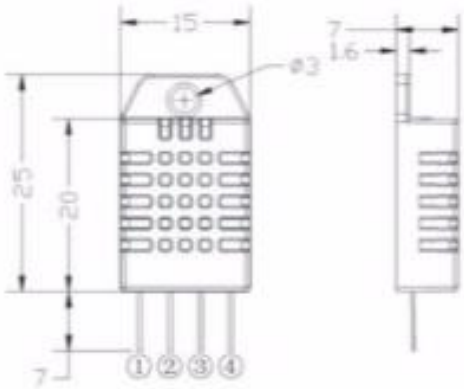
18.10. Sensor temperatura y humedad DHT22

AOSONG Temp、Humidity & Dew point measurement experts

1、Product Overview

AM2302 capacitive humidity sensing digital temperature and humidity module is one that contains the compound has been calibrated digital signal output of the temperature and humidity sensors. Application of a dedicated digital modules collection technology and the temperature and humidity sensing technology, to ensure that the product has high reliability and excellent long-term stability. The sensor includes a capacitive sensor wet components and a high-precision temperature measurement devices, and connected with a high-performance 8-bit microcontroller. The product has excellent quality, fast response, strong anti-jamming capability, and high cost. Each sensor is extremely accurate humidity calibration chamber calibration. The form of procedures, the calibration coefficients stored in the microcontroller, the sensor within the processing of the heartbeat to call these calibration coefficients. Standard single-bus interface, system integration quick and easy. Small size, low power consumption, signal transmission distance up to 20 meters, making it the best choice of all kinds of applications and even the most demanding applications. Products for the 3-lead (single-bus interface) connection convenience. Special packages according to user needs.

Physical map



Dimensions (unit: mm)

2、Applications

HVAC, dehumidifier, testing and inspection equipment, consumer goods, automotive, automatic control, data loggers, home appliances, humidity regulator, medical, weather stations, and other humidity measurement and control and so on.

3、Features

Ultra-low power, the transmission distance, fully automated calibration, the use of capacitive humidity sensor, completely interchangeable, standard digital single-bus output, excellent long-term stability, high accuracy temperature measurement devices.

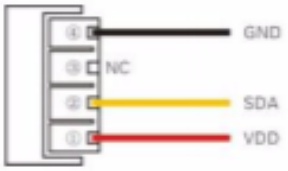
AOSONG Temp、Humidity & Dew point measurement experts

4、The definition of single-bus interface

4.1 AM2302 Pin assignments

Table 1: AM2302 Pin assignments

Pin	Name	Description
①	VDD	Power (3.3V-5.5V)
②	SDA	Serial data, bidirectional port
③	NC	Empty
④	GND	Ground



PIC1: AM2302 Pin Assignment

4.2 Power supply pins (VDD GND)

AM2302 supply voltage range 3.3V – 5.5V, recommended supply voltage is 5V.

4.3 Serial data (SDA)

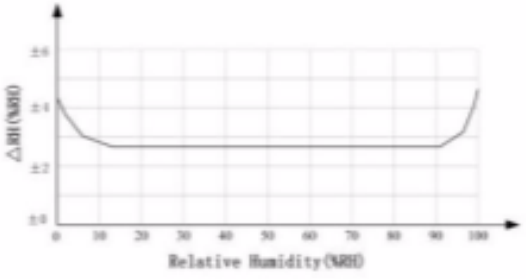
SDA pin is tri structure for reading, writing sensor data. Specific communication timing, see the detailed description of the communication protocol.

5、Sensor performance

5.1 Relative humidity

Table 2: AM2302 Relative humidity performance table

Parameter	Condition	min	typ	max	Unit
Resolution			0.1		%RH
Range		0		99.9	%RH
Accuracy [1]	25℃		±2		%RH
Repeatability			±0.3		%RH
Exchange		Completely interchangeable			
Response [2]	1/e(63%)		<5		S
Sluggish			<0.3		%RH
Drift [3]	Typical		<0.5		%RH/yr

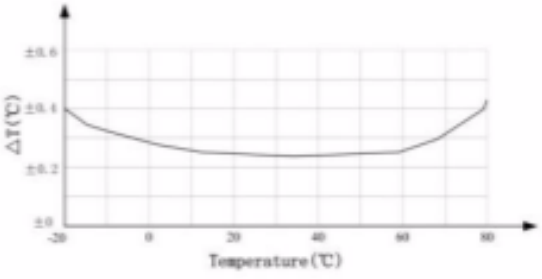


Pic2: At25℃ The error of relative humidity

5.2 Temperature

Table 3: AM2302 Relative temperature performance

Parameter	Condition	min	typ	max	Unit
Resolution			0.1		℃
			16		bit
Accuracy			±0.5	±1	℃
Range		-40		80	℃
Repeat			±0.2		℃
Exchange		Completely interchangeable			
Response	1/e(63%)		<10		S
Drift			±0.3		℃/yr



Pic3: The maximum temperature error

AOSONG Temp、Humidity & Dew point measurement experts

6、Electrical Characteristics

Electrical characteristics, such as energy consumption, high, low, input, output voltage, depending on the power supply. Table 4 details the electrical characteristics of the AM2302, if not identified, said supply voltage of 5V. To get the best results with the sensor, please design strictly in accordance with the conditions of design in Table 4.

Table 4: AM2302 DC Characteristics

Parameter	Condition	min	typ	max	Unit
Voltage		3.3	5	5.5	V
Power consumption [3]	Dormancy	10	15		μA
	Measuring		500		μA
	Average		300		μA
Low level output voltage	I_{OL} [4]	0		300	mV
High output voltage	Rp<25 kΩ	90%		100%	VDD
Low input voltage	Decline	0		30%	VDD
Input High Voltage	Rise	70%		100%	VDD
Rpu [6]	VDD = 5V VIN = VSS	30	45	60	kΩ
Output current	turn on		8		mA
	turn off	10	20		μA
Sampling period		2			S

[1] the accuracy of the factory inspection, the sensor 25℃ and 5V, the accuracy specification of test conditions, it does not include hysteresis and nonlinearity, and is only suitable for non-condensing environment.

[2] to achieve an order of 63% of the time required under the conditions of 25℃ and 1m / s airflow.

[3] in the volatile organic components, the values may be higher. See the manual application to store information.

[4] this value at VDD = 5.0V when the temperature is 25℃, 25 / time, under the conditions of the average.

[5] low output current.

[6] that the pull-up resistor.

7、Single-bus communication (ONE-WIRE)

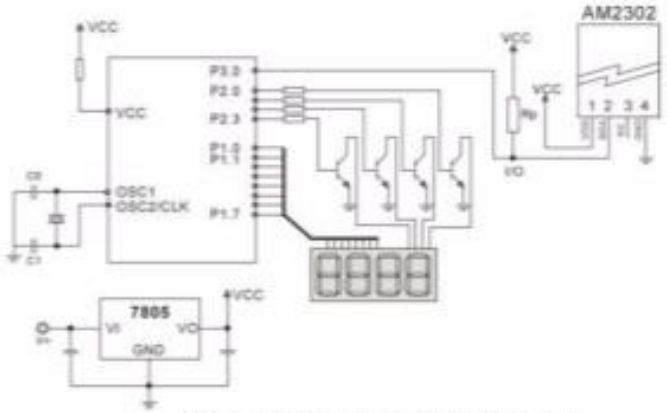
7.1 Typical circuits for single bus

Microprocessor and AM2302 connection typical application circuit is shown in Figure 4. Single bus communication mode, pull the SDA microprocessor I / O port is connected.

Special instructions of the single-bus communication:

1. Typical application circuit recommended in the short cable length of 30 meters on the 5.1K pull-up resistor pullup resistor according to the actual situation of lower than 30 m.
2. With 3.3V supply voltage, cable length shall not be greater than 100cm. Otherwise, the line voltage drop will lead to the sensor power supply, resulting in measurement error.
3. Read the sensor minimum time interval for the 2S; read interval is less than 2S, may cause the temperature and humidity are not allowed or communication is unsuccessful, etc..
4. Temperature and humidity values are each read out the results of the last measurement For real-time data that need continuous read twice, we recommend repeatedly to read sensors, and each read sensor interval is greater than 2 seconds to obtain accuratethe data.

AOSONG Temp、Humidity & Dew point measurement experts



Pic4: AM2302 Typical circuits for single bus

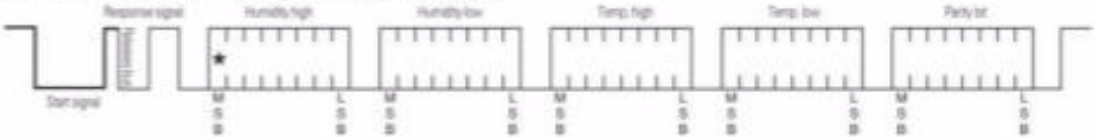
7.2、Single-bus communication protocol

○Single bus Description

AM2302 device uses a simplified single-bus communication. Single bus that only one data line, data exchange system, controlled by the data line to complete. Equipment (microprocessor) through an open-drain or tri-state port connected to the data line to allow the device does not send data to release the bus, while other devices use the bus; single bus usually require an external about 5.1kΩ pull-up resistor, so when the bus is idle, its status is high. Because they are the master-slave structure, only the host calls the sensor, the sensor will answer, so the hosts to access the sensor must strictly follow the sequence of single bus, if there is a sequence of confusion, the sensor will not respond to the host.

○Single bus to send data definition

SDA For communication and synchronization between the microprocessor and the AM2302, single-bus data format, a transmission of 40 data, the high first-out. Specific communication timing shown in Figure 5, the communication format is depicted in Table 5.



Pic5: AM2302 Single-bus communication protocol