Universidad Zaragoza
1542

Trabajo Fin de Grado en Ingeniería Electrónica y Automática

DISEÑO DE UN SISTEMA DOMÓTICO CON COMUNICACIÓN WIFI

Design of a domotic system with wifi comunication

Autor

Ángel Pina Beatove

Director

Roberto Casas Nebra

ESCUELA DE INGENIERÍA Y ARQUITECTURA

2017

TRABAJOS DE FIN DE GRADO / FIN DE MÁSTER

Escuela de Ingeniería y Arquitectura Universidad Zaragoza

DECLARACIÓN DE AUTORÍA Y ORIGINALIDAD

(Este documento debe acompañar al Trabajo Fin de Grado (TFG)/Trabajo Fin de Máster (TFM) cuando sea depositado para su evaluación).

D./Dª. ÁNGEL PINA BEATOVE,

con nº de DNI 73029751J en aplicación de lo dispuesto en el art. 14 (Derechos de autor) del Acuerdo de 11 de septiembre de 2014, del Consejo de Gobierno, por el que se aprueba el Reglamento de los TFG y TFM de la Universidad de Zaragoza,

Declaro que el presente Trabajo de Fin de (Grado/Máster) Grado, (Título del Trabajo) DISEÑO DE UN SISTEMA DOMÓTICO CON COMUNICACIÓN WIFI

es de mi autoría y es original, no habiéndose utilizado fuente sin ser citada debidamente.

Zaragoza, 6 de Noviembre de 2017

Fdo: ÁNGEL PINA BEATOVE

RESUMEN

El presente TFG se basa en el diseño de un sistema domótico con comunicación WiFi. El objetivo principal del mismo es aplicar los conocimientos obtenidos durante el grado junto con la investigación a través de tutoriales y búsqueda en distintas fuentes de información con el fin de elaborar un producto domótico eficiente, de bajo coste y con interoperabilidad y flexibilidad a partir de un diseño general de hardware.

Para conseguir dicho objetivo, se han de estudiar distintas soluciones bloque a bloque de los que compondrán el proyecto: desde la fuente de alimentación, pasando por los componentes a utilizar, y finalizando con un software sencillo y bien estructurado.

La domótica es un campo muy utilizado actualmente en las viviendas y tiene como objetivo integrar la tecnología en sistemas de seguridad, gestión energética, bienestar y comunicaciones. Uno de los principales problemas de este campo es el alto coste de los productos que se ofrecen en el mercado. Por ello uno de los objetivos más importantes de este proyecto es conseguir un producto de bajo coste con una buena eficiencia energética.

Así pues, el presente documento trata a continuación por partes los bloques del sistema, analizando las distintas soluciones que se podrían adoptar y eligiendo finalmente las que más se adapten a los objetivos fijados.

Primero se analiza el módulo WiFi escogido, así como el firmware que se utilizará para programar el microcontrolador integrado en él. En segundo lugar, se estudian los componentes (sensores y actuadores) en base a mínimo consumo y bajo coste. Posteriormente se profundiza en las opciones de plataformas IoT existentes actualmente en la nube, eligiendo la más adecuada para nuestra aplicación. Igualmente, también se trabaja en el diseño de la fuente de alimentación para nuestro producto una vez elegidos los componentes y sabiendo el consumo de los mismos. Por último, se desarrolla una explicación detallada del programa integrado en el producto atendiendo a las especificaciones. Éstas serán detalladas a través de diagramas de bloques, y anotaciones necesarias para una fácil comprensión del código que gobernará la aplicación.

Se realiza el diseño del esquemático con su correspondiente diseño de la PCB del producto, el cual deberá responder a una buena compatibilidad electromagnética de los componentes, evitando interferencias y funcionamientos indeseados de nuestro sistema. Para ello serán aplicadas las distancias de seguridad entre los componentes que exijan los fabricantes de los mismos.

ÍNDICE

1 INTRODUCCIÓN

1.1. MARCO DEL PROYECTO

Desde hace años, la tecnología avanza a pasos agigantados. Este avance llega hasta el cliente final, el cual está experimentando grandes progresos en cuanto a comodidad y control sobre sus posesiones. Debido a ello, el proyecto que se presenta a continuación se centra precisamente en esta comodidad y en el control de una vivienda.

En concreto, se concentra en maximizar las posibilidades para el usuario a un precio muy reducido, pudiendo controlar desde el móvil, a través de una comunicación Wifi, elementos de la vivienda como son las persianas, o la calefacción, por ejemplo. Estos aspectos eran impensables hace no muchos años, pero poco a poco está generando un mayor grado de interés en la población de hoy en día gracias a la comodidad y a la facilidad que todo ello conlleva.

Por todo ello y por la gran relación que este campo comparte con el grado de Electrónica y Automática se ha decidido profundizar en esta memoria en el campo de la domótica y la comunicación inalámbrica.

1.2. OBJETIVOS

El principal objetivo de este proyecto es el diseño completo de un sistema domótico con comunicación WiFi, incluyendo el desarrollo electrónico del producto (hardware), y la programación del mismo (firmware).

A este objetivo primordial, también se le suman los siguientes:

- Diseñar el hardware completo, teniendo en mente un coste final reducido de los componentes, así como una eficiencia energética máxima.

- Conseguir la interoperabilidad y flexibilidad del producto, ya que a partir de un diseño general hardware el cliente tendrá la posibilidad de utilizarlo con distintos fines (por ejemplo: luminosidad, control temperatura, control presencia, etc.).

- Implementar una comunicación con la nube vía WiFi, haciendo uso de una plataforma de IoT (El Internet de las cosas) para la comunicación entre los distintos dispositivos que puedan formar la red (por ejemplo: móvil, producto 1 para luminosidad y producto 2 para temperatura).

Mediante los objetivos mencionados, se pretende conseguir un producto de bajo coste y eficiente que además proporcione al usuario una amplia variedad de usos según sus necesidades.

A continuación, se resumen los posibles casos de uso que igualmente serán implementados en el sistema:

1. Encendido de luces con presencia y manual:

El usuario tendrá la capacidad de encender y apagar las luces de la habitación (manualmente) a través de una aplicación móvil. Esto será posible mediante la comunicación de la APP (Aplicación Móvil) con el servidor, y éste a su vez con el sistema que actuará sobre la electrónica conectada a la luz.
También estará implementado un control automático de la luminosidad de la sala, de forma que al detectar presencia en la habitación se enciendan las luces si ésta no es suficiente.

2. Control de temperatura automático y manual:

El cliente fijará un intervalo de tiempo o un valor límite de temperatura con en el que el sistema accionará automáticamente la calefacción.

Este modo también llevará implementado la capacidad de ser activado el termostato manualmente a través de la aplicación móvil.

3. Simulación de presencia a través del movimiento de las persianas:

Mediante una programación previa de los intervalos de tiempo de subida y bajada de las persianas (o modo aleatorio) el usuario podrá simular la presencia en la vivienda, por ejemplo, en un período de vacaciones en el que se encuentre ausente.

4. Simulación de presencia a través de la luz:

Este modo es equivalente al anterior. Sin embargo, en este caso los intervalos de tiempo que sean fijados por el usuario serán para el encendido y apagado de las luces de la vivienda.

5. Modo de control de una persiana:

Esta programación buscará subir y bajar las persianas según la opción elegida por el usuario: de forma manual o en distintos intervalos de tiempo.
El usuario contará además con la posibilidad de mover las persianas automáticamente a través de la APP.

6. Monitorización de variables y envío a la nube:

En este modo se irán obteniendo los datos de los sensores elegidos previamente por el cliente (temperatura, presencia, humedad, etc.) y almacenados en la nube, pudiendo así realizar gráficas que muestren las estadísticas de un intervalo de tiempo determinado.

7. Modo de detección de presencia en la vivienda:

La función del sistema en este es detectar la presencia de un sujeto en el hogar a través del sensor magnético, para posteriormente mandar a la aplicación móvil o a la nube una alarma o notificación. Esta opción hará trabajar al producto en modo seguridad en el hogar.

8. Modo telefonillo:

A través de esta variante el usuario tendrá la posibilidad de abrir la puerta de su portal a través de la aplicación móvil. Al pulsar el botón “abrir” en la APP, ésta se comunicará con la nube que a su vez dará la orden al módulo WiFi de activar la patilla correspondiente para accionar el telefonillo, consiguiendo así dicho objetivo.

1.3. METODOLOGÍA

En este proyecto se hace uso del diseño de PCB (Placas de Circuito Impreso) utilizando como herramienta base programas para el desarrollo del hardware, así como un entorno de desarrollo firmware para todo lo relacionado con el software.

La metodología a seguir, a partir de los conocimientos obtenidos en el grado y el estudio mediante tutoriales y documentación de las partes del proyecto más novedosas, se basa por un lado en generar el hardware necesario (con el mínimo coste) y, por otro lado, en crear un firmware óptimo para cumplir las especificaciones fijadas.

A continuación, se muestra una planificación temporal del trabajo a realizar a partir de un diagrama de Gantt.

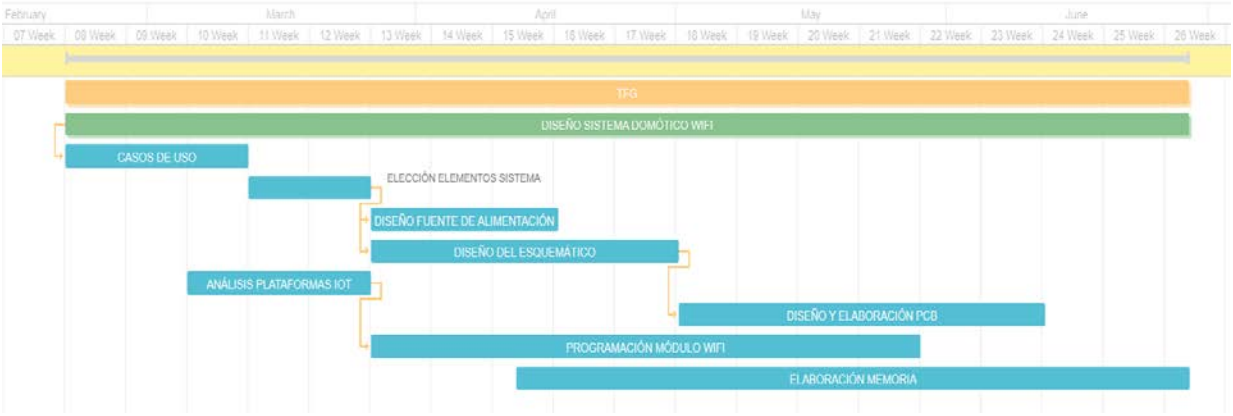


Figura 1: Diagrama de Gantt del proyecto

Tal y como se puede observar en la figura 1 (Diagrama de Gantt del proyecto), las fases en las que se divide el trabajo de este proyecto son las siguientes:

- Establecimiento de la funcionalidad del sistema: detallar los casos de uso que puede afrontar el producto y analizar el tipo de alimentación que se utilizará.

- Estudio y elección bloque a bloque de los elementos que constituyen el sistema: el módulo WiFi, los sensores, los actuadores, etc. Buscando en todo caso el bajo coste y la eficiencia de los mismos.

- Diseño del esquemático incluyendo los componentes elegidos.

- Diseño de la fuente de alimentación, buscando fiabilidad y eficiencia para los posibles usos que quiera dar el cliente al producto.

- Análisis de las distintas plataformas IoT disponibles para acoplar al proyecto. Se hace uso de una herramienta sencilla e intuitiva de utilizar para favorecer la interacción del cliente con el producto.

- Modelado de la PCB y elaboración de la misma.

- Programación del módulo WiFi con los distintos casos de uso que se han decidido implementar. Esta programación debe ser bien estructurada y preparada para una fácil configuración por parte del usuario. El código tendrá la capacidad de interactuar con el usuario para elegir el modo en el que funcionará el sistema.

2 DISEÑO HARDWARE

Uno de los bloques principales de este proyecto, y cuyo diseño y elaboración es indispensable, es el diseño físico del producto. En él se ha invertido una gran parte del tiempo y es el que permite llevar a la realidad lo programado en el microcontrolador del módulo ESP12.

Durante el proceso, se ha partido de una idea inicial de versatilidad y eficiencia en comunión con un bajo coste. Esto ha sido determinante en el diseño del hardware que incluye la elección de los componentes y el diseño de la PCB.

Un punto importante de esta versatilidad lo proporciona la múltiple alimentación posible del sistema, así como un mismo diseño hardware para siete aplicaciones distintas.

Cabe destacar el análisis de precios que se ha hecho de los componentes a la hora de una hipotética fabricación masiva de productos. Se han analizado distintas posibilidades según el uso que se le fuera a dar, buscando máxima eficiencia exclusivamente o centrándose en un coste mínimo (ver Anexo 1). Como hecho más significativo se puede decir que, en el caso de producir mil unidades, suponiendo sólo los costes de los materiales y eligiendo la opción de alimentación más barata (sin fuente), el precio por unidad quedaría en 7,60€. Sin embargo, si lo que se busca es una máxima eficiencia con componentes de proveedores oficiales el precio asciende a 33,44€ por unidad en un lote de 1000 unidades.

Para el uso del prototipo creado, se han utilizado componentes de proveedores oficiales, de los cuales adquiere material de manera regular la Escuela de Ingeniería y Arquitectura, como son Mouser y Farnell. Por este motivo el precio real del prototipo se ha incrementado respecto al análisis inicial de coste del material para dicho proyecto.

2.1. DIAGRAMA DE BLOQUES

A continuación, se presenta un diagrama de bloques (Figura 2) con los elementos de los que se compone el hardware del proyecto. En base a este esquema, se ha realizado el siguiente diseño completo del hardware.

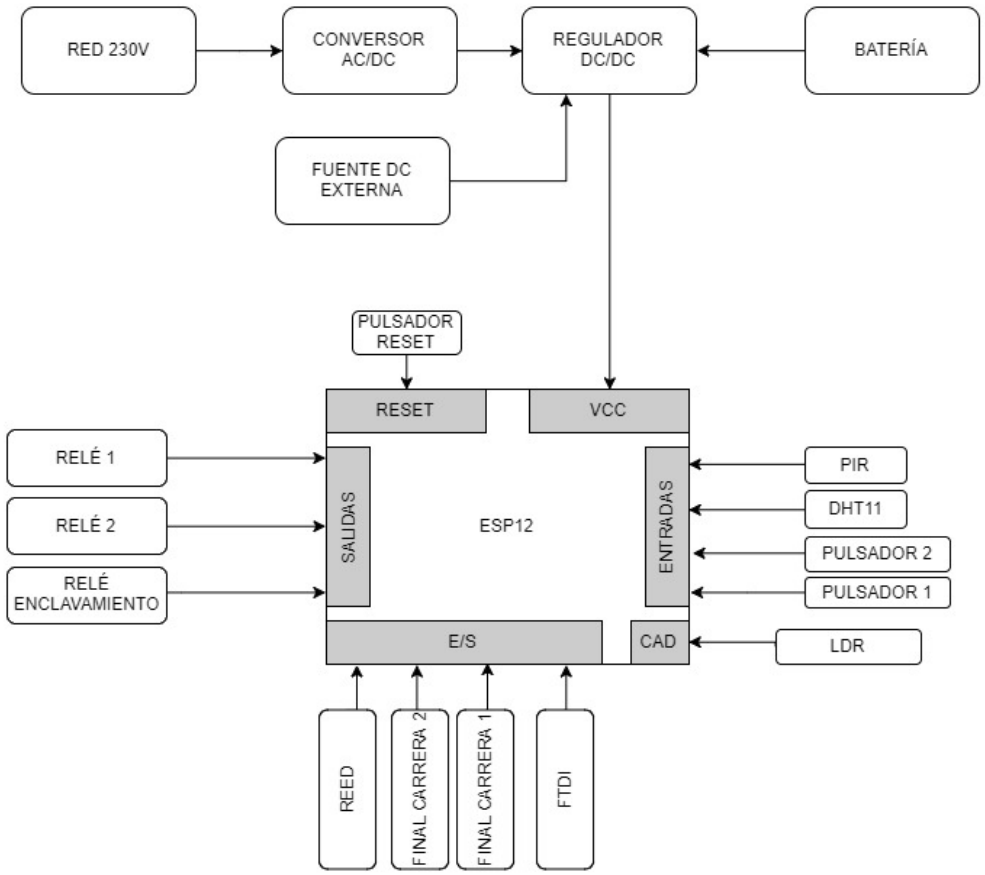


Figura 2: Diagrama de bloques del proyecto

2.1.1. MÓDULO DE CONTROL Y COMUNICACIONES – ESP8266

El módulo Wifi elegido en este trabajo es el módulo Wifi ESP8266 en la versión ESP12. De todos los productos existentes hoy en día en el mercado pensados para IOT, éste es el de mayor utilización debido principalmente a su bajo coste y al abanico de posibilidades que ofrece.

Otra razón de su extendido uso es la facilidad de programación gracias a las librerías gratuitas que te permiten implementar aplicaciones de relativa dificultad de una manera sencilla. Existe también una gran comunidad de usuarios que comparten sus conocimientos sobre el producto en foros abiertos al público.

Además, este módulo lleva un microcontrolador incorporado, y en la versión ESP12 están habilitados 16 pines para poder controlar varias entradas y salidas.

Por todo ello se hace uso de este módulo y se aprovechan las siguientes posibilidades:

- Una entrada analógica para la LDR en este diseño en concreto.
- Las GPIO cuentan con la capacidad de utilizar interrupciones. En nuestra aplicación estas interrupciones se utilizan para la detección de presencia y el aviso por la apertura de la puerta.

En los últimos años la utilización del módulo ESP8266 ha aumentado considerablemente debido a su gran capacidad de realizar tareas en relación a su bajo coste.

2.2. COMPONENTES ELEGIDOS

El proceso de elección de los componentes se ha realizado acorde a las especificaciones fijadas en el inicio del proyecto, debiendo elegir aquellos que permiten un funcionamiento correcto de la aplicación con las premisas de mínimo coste y eficiencia. A partir de estas condiciones y de los casos de uso pensados, los componentes son los siguientes: temperatura y humedad, presencia, luminosidad y magnético.

2.2.1. REGULADORES

El diseño en relación a los reguladores ha tomado una doble vía: Una en la que la premisa era un bajo coste y otra en la que el principal objetivo era una eficiencia máxima.

El escogido para bajo coste es un regulador lineal 78M33 (Figura 3). Es un elemento de Texas Instruments que cumple con la misión de disminuir la tensión de 5V que le llegan desde la fuente hasta los 3,3V necesarios para el funcionamiento de varios componentes de la placa.

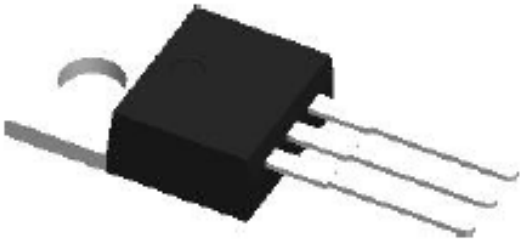


Figura 3: Regulador lineal 78M33

En cuanto a la eficiencia máxima se ha optado por un regulador de conmutación no aislado como es el P7805-S (Figura 4). Ha sido elegido este regulador debido a su eficiencia del 96%, además de contar con el mismo orden de los pines. Gracias a esto, para el mismo diseño de la PCB es posible colocar uno u otro.

Figura 4: Regulador P7805-S máxima eficiencia

2.2.2. SENSORES

A continuación, se detallan los modelos concretos de sensores elegidos con sus características principales.

2.2.2.1. DHT11

El sensor de temperatura: permite llevar a cabo el caso de uso tanto de la estación meteorológica como el del control de la calefacción. El modelo elegido es el DHT11 por la necesidad de utilizar un solo pin del ESP12 y de tener señal digital. Este sensor permite una fácil programación gracias a las librerías ya implementadas en el Arduino IDE.

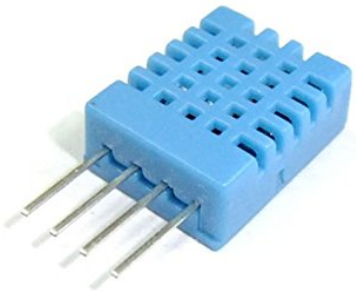


Figura 5: Sensor de temperatura y humedad DHT11

El DHT11 (Figura 5) es un sensor de temperatura y humedad de bajo coste. Estas medidas las obtiene a partir de un termistor y un sensor capacitivo que mide las características del aire circundante, mostrando finalmente los datos de forma digital por un solo pin de datos. Es un componente simple a la hora de utilizarlo y programarlo, ya que Arduino tiene implementada una librería para un uso general.

El inconveniente principal del DHT11 es el intervalo mínimo de 2 segundos entre medidas consecutivas. Sin embargo, para la aplicación que se utilizará en este trabajo es más que suficiente, ya que simplemente deberá medir la temperatura y humedad ambiente de una vivienda sin ser sometido a cambios bruscos de las variables sensadas.

Sus principales características son:

- Alimentación: 3Vdc ≤ Vcc ≤ 5Vdc
- Rango de medición de temperatura: 0 a 50 °C
- Precisión de medición de temperatura: ±2.0 °C .
- Resolución Temperatura: 0.1°C
- Rango de medición de humedad: 20% a 90% RH.
- Precisión de medición de humedad: 4% RH.
- Resolución Humedad: 1% RH
- Tiempo de sensado: 1 seg.

2.2.3.2. PIR HC-SR501

El sensor de presencia: da la posibilidad de una gestión eficiente de la luminosidad de la vivienda junto con el sensor LDR (Light Dependent Resistor) de luminosidad. Su uso también es indispensable en el caso de la detección de presencia en la vivienda como elemento de seguridad. Nuevamente se trata de un componente que requiere un solo pin del micro y una comunicación todo o nada al detectar presencia. Permite una fácil programación utilizando esta salida del sensor como interrupción en la aplicación.

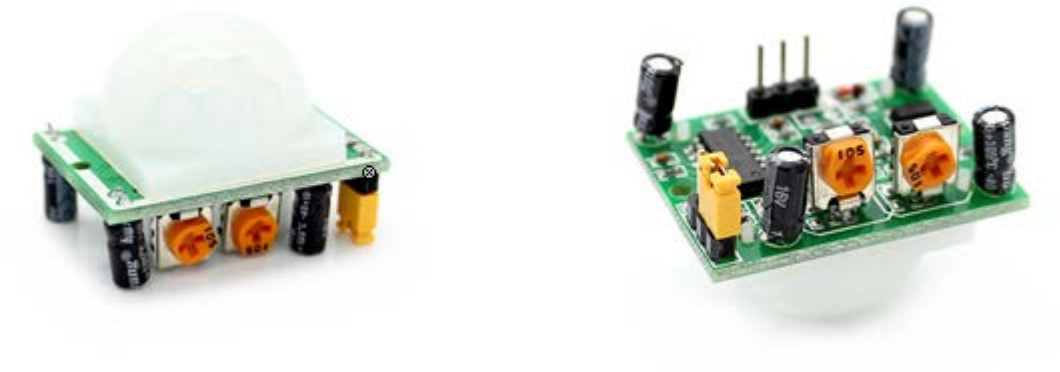


Figura 6: Detector de presencia PIR HC-SR501

El detector de presencia PIR HC-SR501 (Figura 6) es un sensor piroeléctrico de muy bajo consumo (<50uA) y bajo coste. Se basa en los infrarrojos pasivos para detectar el movimiento de personas, animales, etc.; y tiene un amplio cono de acción.

El módulo incluye el sensor, lente, controlador PIR BISS0001 y todos los componentes de apoyo para una fácil utilización.

Igualmente, el módulo HC-SR501 tiene 3 pines de conexión (Vcc, OUT, GND) y dos resistencias variables de calibración: una para el rango de detección y otra para ajustar el tiempo de activación de la salida al detectar la presencia.

Para esta aplicación se utilizará el mayor rango posible de detección (7 metros) y el tiempo será bajo, debido a que la función del detector es activar una interrupción.

Sus principales características son:

- Rango de detección: 3 m a 7 m, ajustable mediante trimmer (Sx)
- Lente fresnel de 19 zonas, ángulo < 100º
- Salida activa alta a 3.3 V
- Tiempo en estado activo de la salida configurable mediante trimmer (Tx)
- Redisparo configurable mediante jumper de soldadura
- Consumo de corriente en reposo: < 50 µA
- Voltaje de alimentación: 4.5 VDC a 20 VDC

2.2.2.3. RELÉ REED

Se ha considerado interesante incorporar un sensor magnético reed, el cual permite en el caso de seguridad de la vivienda saber si se ha abierto una ventana o puerta de la misma. Se ha incorporado en el hardware compartiendo pin con los finales de carrera, puesto que no coinciden en ningún caso de uso. La forma de utilización de este sensor en la aplicación es a través de interrupción de forma análoga que el sensor PIR (Sensor Infrarrojo Pasivo).

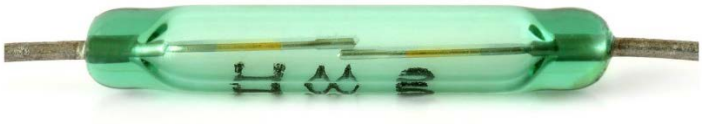


Figura 7: Relé Reed

El Relé Reed (Figura 7) hace la función de interruptor. Está formado por dos láminas ferromagnéticas de hierro y níquel herméticamente cerradas en una cápsula de cristal.

Este sensor se basa en el paso off-on al detectar un campo magnético creado por un imán, poniéndose las dos láminas en contacto.

Sus características principales son:

- Puede llegar a conmutar valores de hasta 5 amperios.
- Puede trabajar con valores de voltaje de nanovoltios.
- Puede llegar a trabajar con valores de 6 GigaHertz con una mínima pérdida de señal.
- Gran aislamiento entre contactos (hasta 1015ohms).
- Resistencia de contacto muy baja (50mOhms).
- No requiere alimentación ni circuitería para funcionar

La aplicación en este proyecto será la de detectar la apertura de una puerta a través de un imán en la misma.

2.2.2.4. LDR

La puesta en marcha del control automático de las luces de la vivienda tiene una eficiencia máxima gracias al sensor de luminosidad LDR, ya que sólo permite encender las luces en caso de que sea de noche o no haya suficiente luz en la habitación. Es un sensor de bajo coste que, en este caso, ocupa el pin analógico del módulo ESP12. Tiene una fácil configuración de hardware con un simple divisor resistivo que evite superar la máxima tensión admisible por el pin analógico (1V).

Figura 8: Sensor de luminosidad LDR

El sensor de luminosidad (Figura 8) es un componente electrónico muy utilizado dentro de la optoelectrónica. Posee una resistencia interna variable que aumenta o disminuye según la luz que este incidiendo en él. De esta manera, la resistencia de un fotoresistor disminuye a medida que aumenta la intensidad de luz.

Este elemento se utilizará en el diseño como resistencia variable en un divisor resistivo y será una condición necesaria para controlar las luces de la habitación. La variación de esta resistencia será detectada en forma de variación de tensión en el pin analógico del ESP8266.

2.3. ESQUEMÁTICO PCB

En este apartado se procede a explicar por bloques el diseño electrónico completo del producto, dando importancia a las dificultades que surgen según las especificaciones de la propuesta inicial.

Como se ha remarcado durante todo el documento, se desea un comportamiento óptimo, pero atendiendo a las premisas de bajo coste y eficiencia, buscando el máximo equilibrio posible entre ellas. Este es precisamente una de las principales dificultades durante el diseño, puesto que unas mayores eficiencias de los componentes implican evidentemente un incremento en el coste de los materiales.

Otro punto a tener especial cuidado es la búsqueda de componentes de pequeñas dimensiones y un diseño compacto. Esto se debe a que, por ejemplo, en el caso de detectar la apertura de una puerta, el producto se sitúa en el marco de la puerta, por lo que con sus dimensiones no puede superar la extensión del mismo.

2.3.1. BLOQUE RELÉS

Para separar la parte de control y la de potencia de la aplicación se hace uso de relés (Figura 9) de uso general. En los casos en los que se sube o baja la persiana y en el del telefonillo se utilizan relés sin enclavamiento y un contacto conmutado.

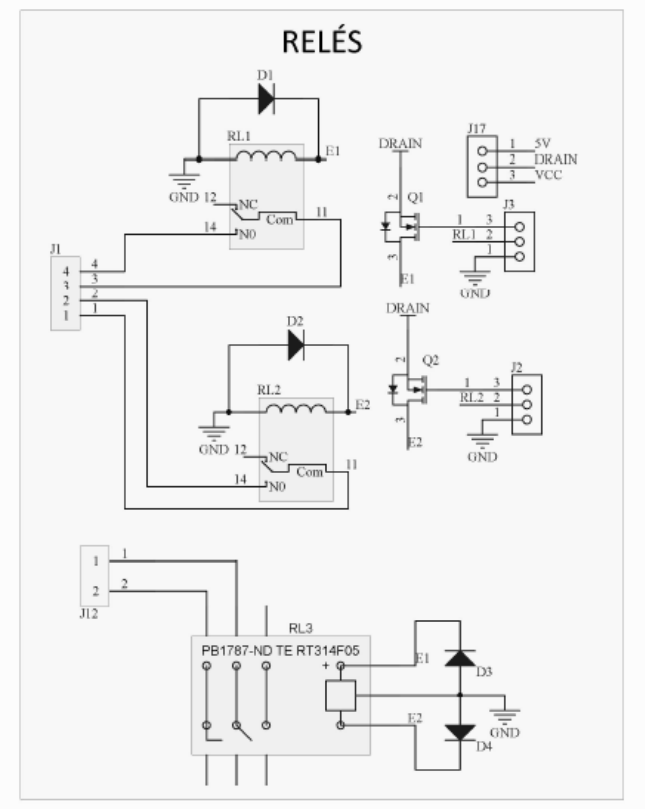


Figura 9: Esquemático del bloque de relés

En un primer momento, sólo se iba a hacer uso de este estilo en el diseño de la PCB. Sin embargo, debido al problema de un consumo excesivo en la situación mantener encendidas las luces y la calefacción, fue necesario añadir un relé de enclavamiento para el on-off, necesitando corriente únicamente en los periodos de transición, y no durante el periodo de on, que puede llegar a tener una duración de horas.

El contacto con los circuitos de potencia externos a la PCB (luces o motor de la persiana, por ejemplo) se hace a través de fichas dispuestas en la periferia de la placa.

Otra observación interesante es la necesidad de incluir diodos antirretorno de la corriente. En caso contrario, las conmutaciones serían más agresivas pudiendo dañar los relés antes de lo esperado. Análogamente, se hace uso de transistores mosfet NPN con el objetivo de alimentar el circuito de control de los relés, ya que la corriente proporcionada por el ESP8266 sería insuficiente por sí sola.

Por último, cabe destacar las tiras de pines que ha sido necesario añadir para poder utilizar las mismas GPIO del micro tanto a la hora de la configuración del micro y la introducción del firmware, como al momento de funcionamiento habitual de la aplicación con los relés.

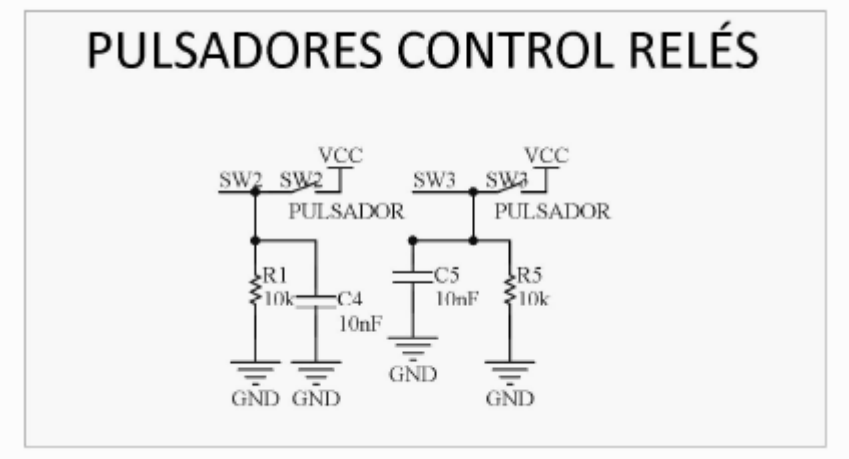


Figura 10: Esquemático del bloque de pulsadores

El diseño implementado en este proyecto ofrece la posibilidad de un control doble de los relés, y, por tanto, de lo que esté conectado a ellos ya sea un motor para las persianas, las luces de la habitación, el termostato de temperatura, etc. Este control por partida doble (Figura 10) implica la oportunidad de manejarlo mediante hardware a través de los pulsadores de la figura, o de software a través de la nube y nuestro smartphone o PC.

2.3.2. BLOQUE SENSORES

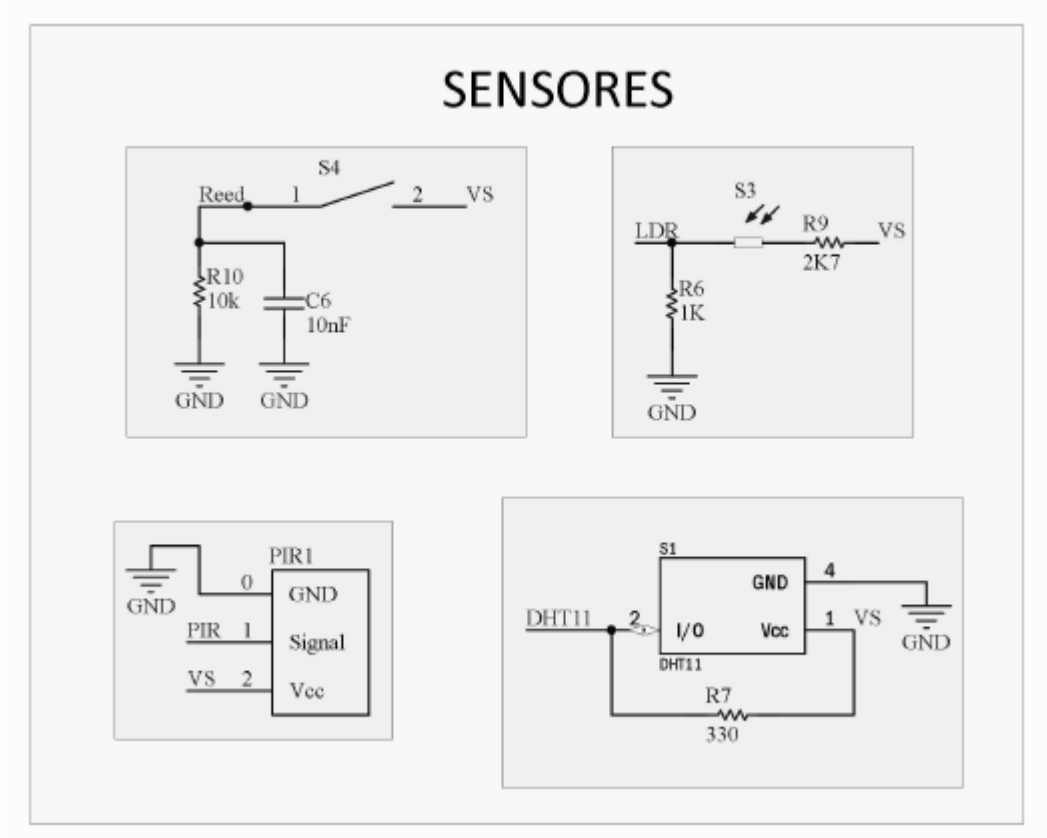


Figura 11: Esquemático del bloque de sensores

Este bloque (Figura 11) tiene como objetivo informar al microcontrolador, cuando la aplicación lo requiera, de las variables a medir, ya sea temperatura, humedad, luminosidad, presencia, etc.

Se ha buscado utilizar sensores de bajo coste y bajo consumo, como se ha detallado anteriormente. La elección de estos sensores y no más ha sido como consecuencia del limitado número de entradas/salidas con las que cuenta el microcontrolador.
De la misma manera, en la elección de los componentes ha sido prioridad el hecho de que necesiten el mismo número de pines para transmitir los datos. Tanto el DHT11 como el PIR son sensores con salida única y digital, perfectos para la aplicación deseada.

Tanto el PIR como el Relé Reed, al activar la patilla del micro a la que está conectada, ponen en marcha una interrupción que desencadena una rutina en el programa, variando según el modo en el que se encuentre (Control de luces, control de presencia, etc.).

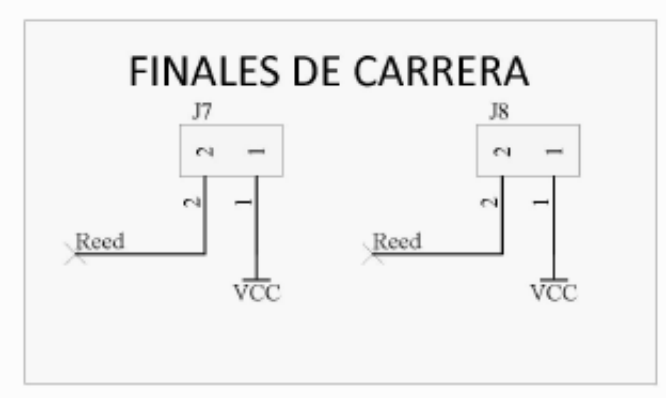


Figura 12: Esquemático del bloque de finales de carrera

Por último, se contempla la posibilidad de utilizar finales de carrera (Figura 12) como detonante de una interrupción en el ESP12 que provoque la desactivación del relé que activa el motor para subir o bajar las persianas. En un primer momento se planteó la posibilidad de hacerlo mediante una configuración software en la que el usuario pudiera elegir el tiempo de bajada o subida.

2.3.3. BLOQUE MÓDULO WIFI ESP12

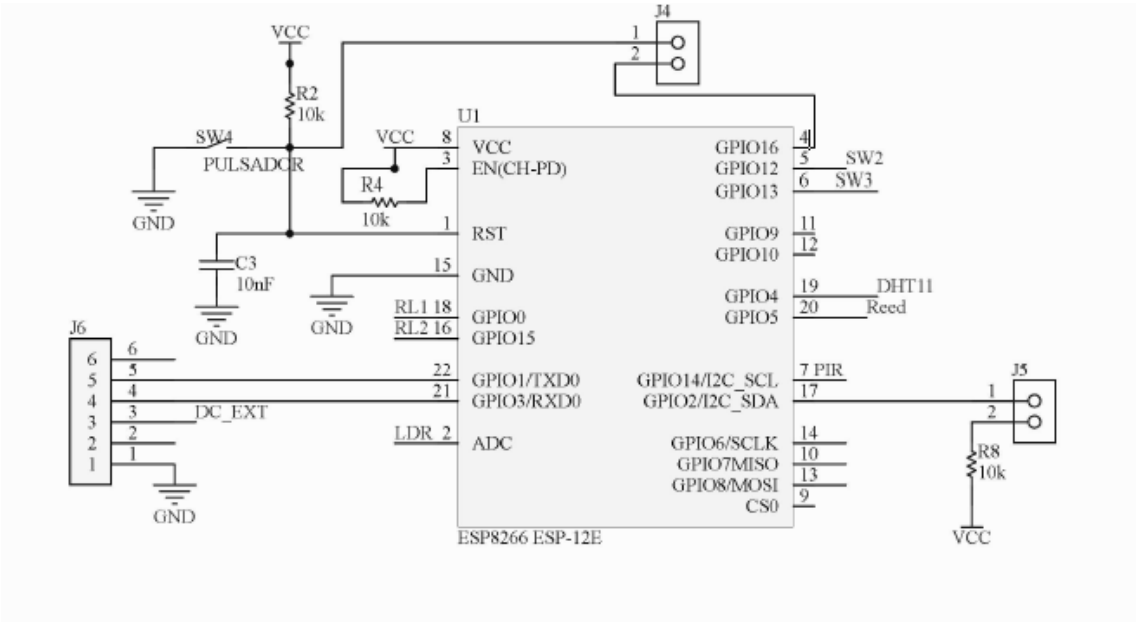


Figura 13: Esquemático del bloque del módulo Wifi ESP12

Este bloque (Figura 13) es el más importante del proyecto, puesto que alberga en él al cerebro de todas las operaciones, el microcontrolador. Los componentes que rodean al módulo Wifi son básicamente los necesarios para un correcto funcionamiento del mismo (condensadores de filtrados, resistencias pull up, etc.

La tira de pines J6 es la puerta de comunicación entre el micro y el programador, ya que van a parar dos de los pines a la patilla de recepción y envío de datos serie.

Como se ha explicado anteriormente, las salidas GPIO0 y GPIO15 van a parar a otras tiras de pines que permiten diferenciar un estado de configuración del micro (momento en el que se le inserta el programa) de otro de funcionamiento normal (funciona la aplicación habitual para la que ha sido creado).

Así pues, este es uno de los puntos clave de este proyecto, donde existe la capacidad de una comunicación con la nube, pudiendo almacenar datos y actuar sobre las salidas del microcontrolador.

Cabe destacar en este apartado que hay que tener especial cuidado con las posiciones de los jumpers en todo momento, ya que una mala posición de los mismos implica un incorrecto funcionamiento de toda la aplicación (Ver anexo 4).

2.4. ALIMENTACIÓN

Ésta es una de las partes más importantes del proyecto, ya que proporciona una gran versatilidad al producto. Uno de los objetivos principales reflejados en la propuesta es conseguir dicha característica. Esto es necesario porque, debido a los casos de uso anteriormente comentados, no siempre se dispone de la misma fuente de energía para hacer funcionar el módulo Wifi y todo el circuito.

El producto creado debe funcionar tanto en entornos donde haya conexión a la red eléctrica directamente, como en los que no haya toma posible. Por ello se hace el diseño de la fuente para tres casos distintos.

2.4.1. ANÁLISIS DE CONSUMO

Para un correcto diseño de la alimentación hay que realizar en primera instancia un análisis de los consumos de los bloques de los que se compone la aplicación. Por ello se analiza el peor caso posible de consumo, y se elige una fuente de energía y una circuitería acorde a estas necesidades energéticas.

En algunos casos de uso de la aplicación, para conseguir un menor consumo en los periodos en los que no es necesario el micro en pleno funcionamiento, se utiliza una de las características interesantes de este módulo wifi: la posibilidad de llevar al ESP8266 a un estado de reposo (Sleep). Esto es especialmente interesante en los casos en los que se necesitan baterías para alimentar el micro, ya que consume en su funcionamiento habitual 150 mA.

A continuación, se analizan los tres modos de ahorro de energía de los que dispone el ESP8266:

- Modem sleep: Permite desactivar la conexión Wifi de tipo Station, la cual ha sido establecida con un punto de acceso, cuando no es necesaria y volver a activarla cuando se necesite. El consumo en este modo es de 15 mA.
- Light sleep: Este modo de ahorro permite mantener la conexión Wifi de tipo Sation, reduciendo el consumo de intensidad en los momentos en los que no se envía la información. En este caso el consumo es de 0.5 mA.
- Deep sleep: Es el modo de mayor ahorro, ya que deja la placa en suspenso, inhabilitando tanto las interrupciones como las comunicaciones. La única parte de la placa que funciona durante este modo es el reloj en tiempo real para poder reiniciarla al finalizar el intervalo de reposo. El consumo típico es de 20 uA.

Se adjunta una tabla (Tabla 1) con los ítems que permanecen activados según el modo de ahorro:

Item		Modem-sleep	Light-sleep	Deep-sleep
Wi-Fi		OFF	OFF	OFF
System clock		ON	OFF	OFF
RTC		ON	ON	ON
CPU		ON	Pending	OFF
Substrate current		15 mA	0.4 mA	~ 20 μA
Average current	DTIM = 1	16.2 mA	1.8 mA	-
	DTIM = 3	15.4 mA	0.9 mA	
	DTIM = 10	15.2 mA	0.55 mA	

Tabla 1: Consumo según el modo de ahorro

En este proyecto se trabaja con el modo Deep sleep porque, además de ser el de menor consumo, es el único implementado en las librerías de Arduino.

La peor situación que se puede dar es sin duda el momento en el que deba estar el módulo Wifi a pleno rendimiento, así como los relés correspondientes.

Componente	Consumo máx. (mA)
RELÉ STANDARD	72
RELÉ ENCLAV	Rated coil power 595 mW. Rated current (sqrt (0,595/42))->119mA
PIR	0,05
HALL	6
DHT11	2,5
ESP12	150
LDR	1
TOTAL	350,55

Tabla 2: Consumo de los componentes principales del proyecto

Observando la tabla 2 se aprecia como el consumo de los sensores es despreciable en comparación con el microcontrolador y los relés (solamente un 2.72% del total).

Debido al limitado número de pines del ESP12, desde dos GPIO (General Purpose Input/Output) se controlan tanto los relés standard como el relé de enclavamiento. Por lo que el peor caso tiene lugar en las aplicaciones siguientes:

- Control de la persiana: durante la subida y bajada de la persiana se consume los 341 mA correspondientes a los relés y el módulo Wifi.
- Control de las luces: únicamente en el momento de encendido y apagado de la iluminación, la PCB consume los 341 mA. El resto del tiempo el consumo es de los 150mA en el modo automático en el que el microcontrolador está en todo momento alerta.
- Modo telefonillo: En este caso, igual que en el caso del control de la persiana, durante el tiempo en el que se acciona el telefonillo, tanto los relés como el módulo Wifi solicitan corriente de la fuente, por lo que se necesitan los 341mA.

Repasando estos tres peores casos se ha elegido, para alimentación desde la red, una fuente de alimentación de reducidas dimensiones y que puede suministrar al circuito con hasta 600mA durante largos periodos de tiempo.

2.4.2. ALIMENTACIÓN DESDE LA RED

Esta opción es la que se utiliza en los casos que tengamos acceso directamente a la red eléctrica (Figura 14). Esto sucede al controlar las luces, por ejemplo.

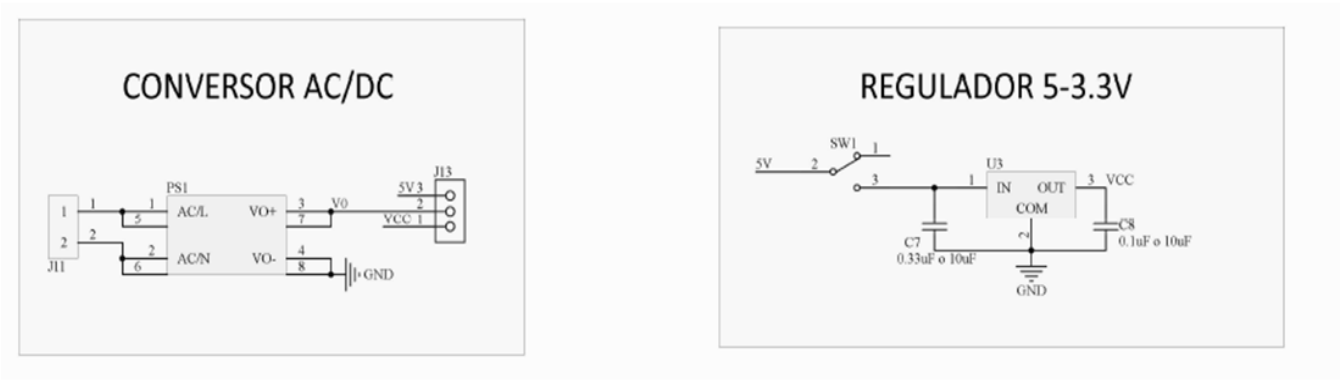


Figura 14: Esquemático del bloque de alimentación desde la red

Este bloque se consigue a partir de un convertidor AC/DC que toma de la red los 230VAC y saca por la salida 5VDC. A la hora de realizar los footprint se tienen en cuenta dos posibles componentes: una versión low cost y una de proveedor oficial.

La salida del convertidor está conectada a un regulador 5-3.3V. En este caso, con el mismo footprint, se han elegido dos posibilidades de regulador según si nuestro objetivo es abaratar costes o maximizar la eficiencia: uno lineal y otro de alta eficiencia. En el Anexo 1, se aprecia la diferencia de precio de elegir una opción a otra. También se contempla la posibilidad de no hacer uso del regulador y poner un convertidor AC/DC con salida 3.3VDC.

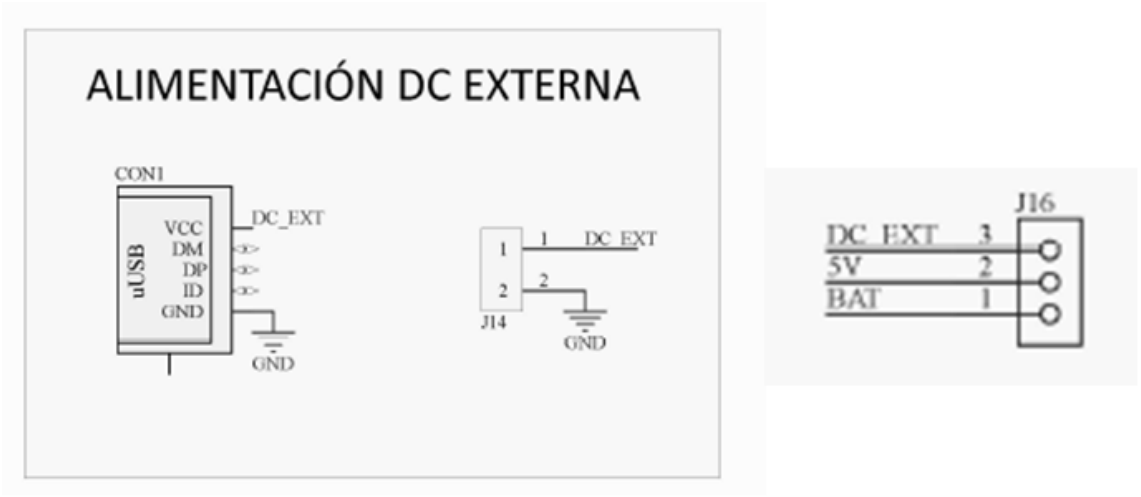


Figura 15: Esquemático del bloque de alimentación dc externa

2.4.3. ALIMENTACIÓN DC DIRECTA

Este caso tiene lugar en los momentos en los que se cuenta con una fuente de continua directamente, como por ejemplo en el caso del telefonillo.

Para esta situación de tensión continua se tienen como posibles entradas a la PCB un conector micro usb, que sirve únicamente como alimentación; y una ficha a la que llevar directamente los dos cables. Una vez aquí, con el conector J16 puenteado con el pin DC_EXT, la corriente llega al regulador para seguir el mismo camino que en el caso anterior de alimentación directa de la red.

2.4.4. ALIMENTACIÓN POR BATERÍAS

Para las situaciones en las que el producto no tenga acceso a ninguna fuente de energía externa cuenta con la posibilidad de alimentarse a través de baterías.

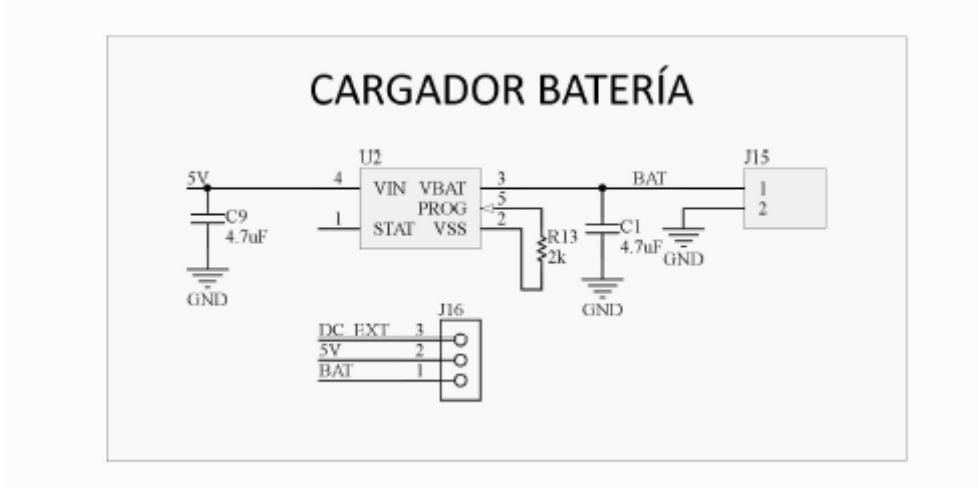


Figura 16: Esquemático del bloque de carga de la batería

Este es uno de los casos más delicados, ya que los casos de uso que requieren de esta alimentación deben estar programados de forma que tenga el mínimo consumo. Si no es así, la batería tendrá una duración insuficiente y deberá ser cargado con demasiada frecuencia. Por ello se utiliza el modo deepSleep comentando anteriormente en la memoria.

Como componente que gestiona la energía de la batería se utiliza el MCP73831. En el momento en el que se le da tensión por la patilla de VIN, el integrado procede a la carga de la batería, controlando la temperatura y diversos parámetros ajustables.
Esta opción es utilizada en los casos de uso de control de temperatura, envío de datos a la nube de temperatura y humedad, y detección de apertura de puertas o ventanas.

2.4.5. ALIMENTACIÓN SENSORES

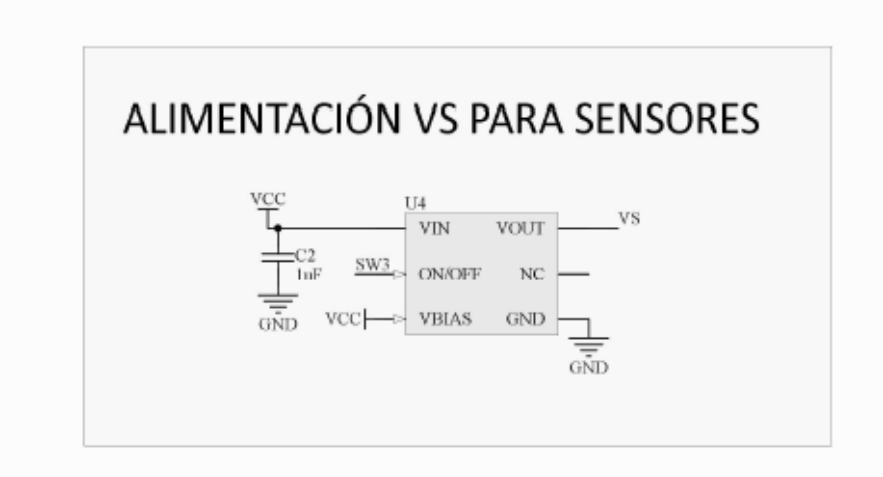


Figura 17: Esquemático del bloque de alimentación de los sensores

Para un mayor ahorro, y dado que la aplicación lo permite, se utiliza una alimentación de los sensores (Figura 17) a través del módulo Wifi ESP8266. Sin embargo, este módulo no tiene la capacidad de suministrar la corriente suficiente para el correcto funcionamiento de los sensores. Por tanto, se incorpora el distribuidor de alimentación TSP22860 para solucionar este problema. De esta manera los sensores funcionarán únicamente a la orden del microcontrolador principal en el momento que sean necesarios.

Se hace uso de esta opción para los sensores de presencia, el relé reed y LDR. Todo ello hay que tenerlo en cuenta a la hora de programar el micro, debiendo activar y desactivar la GPIO correspondiente según convenga.

2.5. PCB

En este apartado se explica cómo están distribuidos los componentes en la PCB (Ver Anexo 3). Lo primero a resaltar son los planos de masa utilizados. Como se puede apreciar en el Anexo 3, se ha dejado sin cubrir con el plano de masa la zona de la antena del módulo Wifi. Con ello se evita un funcionamiento incorrecto en la comunicación Wifi. Por otra parte, estos planos son importantes especialmente en aplicaciones de alta frecuencia (como es el caso de este proyecto en cuanto al módulo Wifi), ya que facilita el retorno de la corriente y aísla del ruido.

En cuanto a la disposición de los componentes el criterio seguido es el siguiente:

- Las conexiones entre el exterior y la PCB se ubican en la periferia de la placa, separando por una parte la zona de potencia (alimentación y salidas de los relés) y la de señal (fichas para los finales de carrera y pines para conectar el cable serie).
- Los condensadores de filtrado se colocan cercanas al elemento a filtrar y evitar interferencias (GPIO del módulo Wifi).
- La parte izquierda de la placa corresponde a la parte de potencia, incluyendo los tres relés y la fuente de alimentación, dejando la parte derecha para los sensores y el módulo Wifi.
- La disposición de los componentes tiene como objetivo general utilizar pistas lo más rectas y cortas posibles, para una circulación óptima de la corriente. Por ello se intenta disponer los elementos dejando únicamente el espacio necesario entre ellos, reduciendo así la extensión total de la placa.
- El relé reed se encuentra en una esquina con el fin de colocar el producto verticalmente en el marco de la puerta. El imán colocado en la puerta completará la aplicación de cerrar o abrir el interruptor con la apertura o cierre de la puerta.

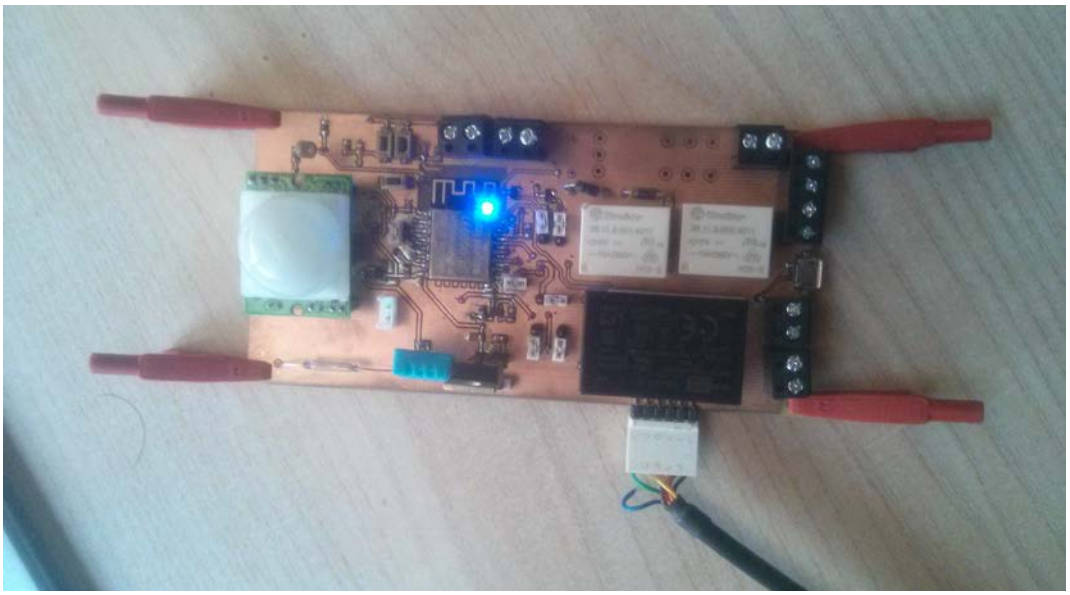


Figura 18: Vista aérea de la PCB

3 DISEÑO FIRMWARE

En este apartado se explica el software del proyecto con la ayuda de diversos diagramas de flujo, los cuales dan una idea más intuitiva de cómo está estructurado el código, sin entrar en todos los detalles de las líneas programadas.

Primeramente, aclarar que, tal y como está reflejado en el anexo 2 con todo el código de la aplicación, son alrededor de unas 4000 líneas de código que han tenido que ser bien estructuradas y diferencias por bloques para una mejor lectura posterior. Gran parte de este código alberga la parte de configuración, desde la gestión de la memoria (cargar y guardar) hasta las máquinas de estados implementadas para una puesta a punto del caso de uso que desee el usuario a través de la propia consola del Arduino IDE. Así, se establece un diálogo guiado con el cliente para fijar las opciones iniciales con las que funcionar la aplicación de manera óptima.

Otra parte importante de este software, además de los casos de uso que se detallan a continuación, es la parte del código necesaria para la comunicación con la nube, programada con las librerías ya creadas para la interacción entre la plataforma Thinger.io y el módulo ESP8266. Con este programa se habilita la posibilidad de control desde la aplicación móvil de las GPIO del ESP12. A partir de este control podemos gobernar los relés, mandar un email de alarma al cliente cuando detecta presencia en la vivienda, o recoger los datos del sensor de temperatura en la nube.

3.1. MODOS DE USO DEL MÓDULO ESP8266

A continuación, se analizan las distintas posibilidades que tiene el módulo Wifi para ser programado, incluyendo tanto ventajas como inconvenientes de cada uno. Como conclusión se expondrá el firmware escogido para este proyecto y los motivos de esta elección.

3.1.1. COMANDOS AT

Este es el firmware que viene por defecto en los módulos al comprarlos. Mediante éste, a través de los pines RX y TX, se le puede enviar al módulo comandos AT para por ejemplo: escanear redes Wifi disponibles, utilizar el módulo como cliente para conectarlo a un router, crear un servidor, etc.

Para el aprendizaje de este modo se ha hecho uso de tutoriales, pudiendo así conocer la forma de utilizar el módulo a través de estos comandos.

La ventaja que se encuentra es su sencillez de utilización para aplicaciones de poca dificultad (encender y apagar un led a través de la barra del navegador por ejemplo).

Por otro lado, la parte negativa reside en que con este firmware no se aprovecha el potencial de este módulo ya que se necesita de otro microcontrolador (por ejemplo el de Arduino) para control de sensores y actuadores, siendo que el propio micro que lleva integrado el módulo ya tiene la capacidad (a través de sus GPIO) de realizar esta misma función sin intermediarios.

3.1.2. NODEMCU

Una vez flasheado el módulo con este firmware, en vez de escribir comandos AT a través de la UART, se puede escribir en lenguaje Lua y éste será interpretado. Es decir, es una consola interactiva de Lua. Además, dispone de IDEs como Node Explorer o una parte del Arduino IDE que facilita la tarea de programar el dispositivo.

Su principal ventaja es que permite crear aplicaciones relativamente complejas con pocas líneas de código, como por ejemplo conectar a una red Wifi cercana. Otra ventaja está en que las pruebas son interactivas a través de la consola de Lua.

Del mismo modo, los inconvenientes encontrados son, entre otros: muy poca memoria de programa disponible, puesto que al ser un lenguaje 'interpretado' necesita aún más memoria; si se acumulan unas pocas líneas de código da fallo de memoria. Otra desventaja es la no existencia de demasiadas librerías para poder trabajar con facilidad.

3.1.3. ARDUINO IDE CON LENGUAJE ARDUINO (C/C++)

Este Firmware permite programar el micro del módulo Esp8266 como si fuera el de Arduino, es decir, se pueden controlar las GPIO y el resto de patillas de la misma forma que se tratan a los pines de la ya muy extendida placa de Arduino.

Las ventajas que se pueden encontrar son:

- Tiene un gran número de librerías incluidas en Arduino IDE listas para ser utilizadas. Con ellas se pueden controlar todo tipo de sensores, motores, pantallas, etc.

- Cuenta con una gran comunidad de usuarios por todo el mundo donde compartir y buscar todo tipo de información.

Y, por otro lado, los inconvenientes son:

- Es necesario flashear el dispositivo para probar el código cada vez que es modificado, en vez de simplemente escribirlo en la UART como se hace con NodeMCU. Esto hace que el proceso de prueba del programa sea algo lento (le transferir el programa al micro en algunos casos mucho tiempo). Es una desventaja relativa porque en el caso de Arduino también es necesario flashearlo cada vez que se modifica el programa.

Después de haber analizado las distintas posibilidades de firmware que se utilizan hoy en día en el módulo Esp8266, se ha decidido elegir una programación del micro a través del Arduino IDE mediante el lenguaje de Arduino (C/C++). Esto se debe a que es el entorno con el que más se ha trabajado en la titulación (lenguaje C y Arduino) y proporciona numerosas facilidades con sus librerías integradas.

Una vez elegido el modo de programación del micro puede pasarse al planteamiento del programa que hay que crear para el correcto funcionamiento del producto domótico.

3.2. LIBRERÍAS DISPONIBLES UTILIZADAS

Las librerías de las que se ha hecho uso durante este proyecto han sido todas de carácter gratuito. Las más importantes son:

- *ESP8266WiFi.h:* A partir de esta librería se inicializa el micro, se conecta mediante usuario y contraseña (previa introducción por parte del usuario) a la red Wifi más cercana, y se programa en arduino el ESP12 como si fuera prácticamente un Arduino UNO en cuanto a control de entradas y salidas. Esta librería lleva también implementadas funciones que te permiten utilizar el modo deepSleep o reiniciar el microcontrolador.

- *ThingerESP8266.h:* Gracias a ésta, es posible establecer comunicación entre la plataforma ThingerIO y el módulo Wifi ESP12. A través de esta librería es posible controlar las GPIO desde la página web de thinger o incluso por medio de la APP del smartphone.

- *DHT.h:* Esta librería permite una lectura sencilla de las variables temperatura y humedad a través de dos simples funciones.

- *Time.h:* La posibilidad del microcontrolador de saber en todo momento la hora en la que se encuentra la brinda esta librería. Con esta librería se ha programado de manera fácil e intuitiva los casos de uso en los que es necesario saber la hora, como por ejemplo el control de las luces o de las persianas.

- *SoftwareSerial.h:* La comunicación entre el usuario y el ESP12 a la hora de la configuración del modo elegido, usuario y contraseña etc. ha sido posible gracias a esta librería, con la que se inicializa el terminal de Arduino IDE y se gestiona el envío y recepción de datos a través del puerto serie.

- *EEPROM.h:* Toda la gestión de la memoria de la aplicación ha sido programada por medio de esta libería. Lleva implementadas funciones de fácil uso como lo son el guardar y cargar información en la memoria del ESP12, imprescindible para el almacenamiento de las credenciales, variables de la aplicación, etc.

3.3. CASOS DE USO

3.3.0 SET UP

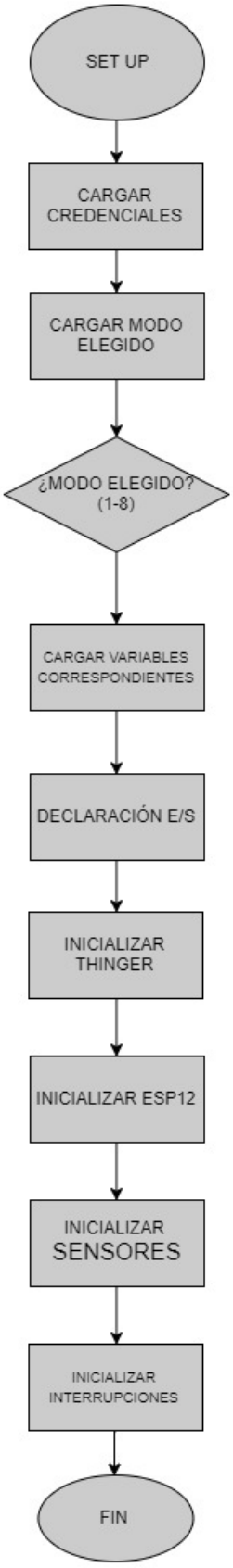


Figura 19: Diagrama de flujo del set up de la aplicación

Como se observa en la figura 19, el set up de la aplicación se ejecuta siempre una única vez al arrancar el microcontrolador (ver anexo 2, línea 1442). Es una estructura secuencial en la que se cargan los datos de la memoria (línea 1448 y 1449) tras un periodo de sleep del micro, por ejemplo, comenzando por las credenciales para conectarse a la red Wifi, y pasando por la recuperación del modo elegido por el usuario.

Una vez cargados estos datos, se recuperan de la memoria las variables necesarias según el caso de uso, resumidos todos en la introducción (ver anexo 2, líneas 1451-1512). Estos datos son imprescindibles para el correcto funcionamiento de la aplicación. Sin hacer uso de esta memoria, al resetear el micro una vez terminado el tiempo en off (sleep) se perderían todos los datos de las entradas y salidas de la aplicación, así como el usuario y contraseña de la red Wifi a la que conectarse.

Tras esto se declaran todas las GPIO como E/S según el diseño pensado, se inicializa la comunicación con Thinger.io, se inicializan los sensores que lo necesiten como el DHT11 y, finalmente, se declaran todas las interrupciones que pueden saltar durante el funcionamiento del bucle del programa (ver anexo 2, líneas 1515-1559).

3.3.1. CONTROL LUCES

Este es el primero de los modos (Figura 20), con el cual se tiene la posibilidad de gobernar la iluminación de la habitación en la que se instala el producto. Este control de la luz se puede realizar de dos formas distintas que puede elegir el usuario en la configuración inicial de la aplicación: una primera manera manual y una automática.

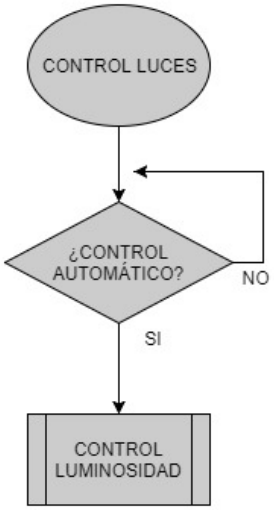


Figura 20: Diagrama de flujo del control de luces

La forma manual consiste en el control sobre las bombillas a partir de la aplicación del smartphone Thinger.io. En ella, al elegir el modo Control de luces aparece una pestaña con un interruptor virtual que gobierna, a través de la nube, las GPIO del módulo Wifi correspondientes al relé de enclavamiento. Con estas GPIO se activa y desactiva el relé y, por consiguiente, las luces de la habitación (ver anexo 2, líneas 1608-1614).

La segunda manera de controlar las luces es de forma automática si así lo elige en la configuración inicial el usuario. Esta manera también lleva implementada la opción manual de encendido y apagado de las luces.

El funcionamiento de dicho modo automático (Figura 21) es el siguiente: lo primero en medir es la luminosidad de la sala a través del sensor LDR. Si no hay luz suficiente en la habitación se procede a comprobar si hay alguien en ella a través del sensor de presencia. En el caso de que el detector entienda que hay movimiento pone en alto el GPIO correspondiente al enclavamiento del relé, encendiendo así las luces. En caso de haber suficiente luz en la habitación, pasa a comprobar si las luces están encendidas. Si efectivamente hay luminosidad artificial, no detecta presencia y además han pasado dos minutos desde la última activación, el micro desactiva el relé de enclavamiento a partir de la GPIO correspondiente (ver anexo 2, líneas 374-386).

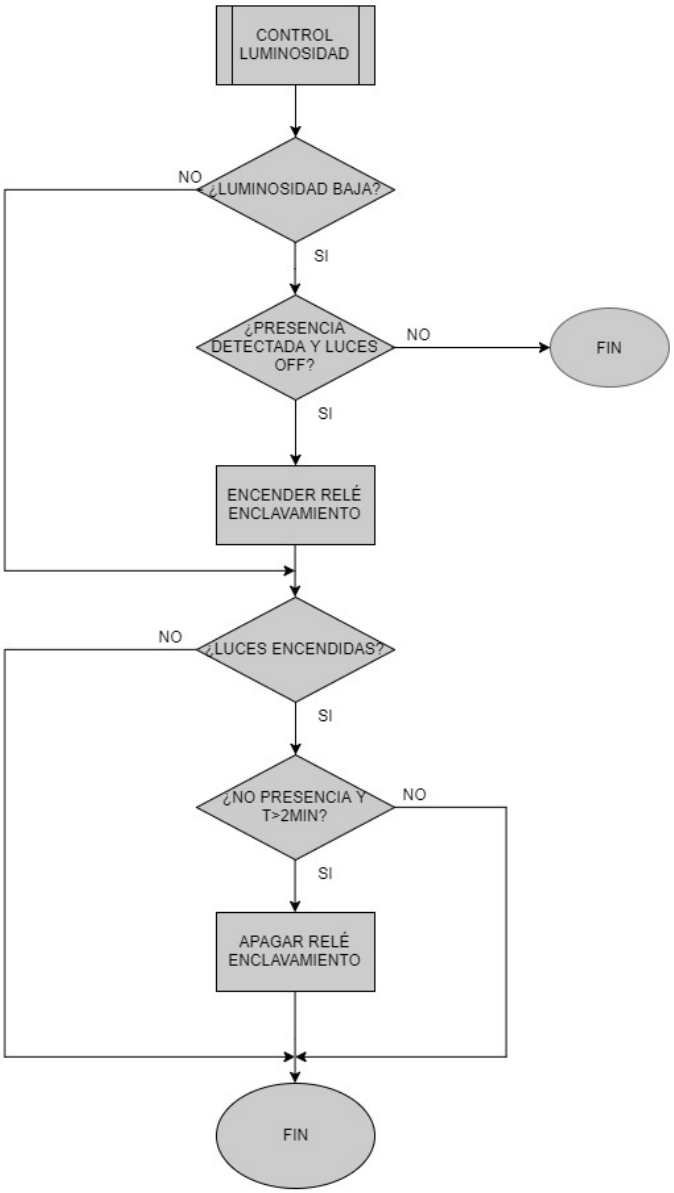


Figura 21: Diagrama de flujo del control automático de las luces

3.3.2. CONTROL TEMPERATURA

El segundo caso de uso de la aplicación tiene por objetivo controlar la calefacción de la estancia a través de la activación del relé de enclavamiento (Figura 22). En este caso, igual que en el anterior, a la parte de potencia del relé se debe conectar el circuito eléctrico que haga funcionar la calefacción. De nuevo están implementadas dos posibilidades a la hora de gobernar el termostato, pero en este caso será un control por temperatura (Figura 20) o uno por tiempo.

En este modo, a diferencia del anterior, tiene incorporado el modo ahorro mediante la utilización de la opción Deep Sleep del ESP12. Esta configuración es posible porque tanto la temperatura como el tiempo son variables que admiten un cierto margen de actuación, no siendo necesario (como es el caso de la detección de presencia) una alerta permanente ante los cambios.

Al elegir en un primer momento este caso de uso en la configuración inicial del ESP8266, aparecerá en la APP una pestaña con un interruptor que permita actuar sobre el termostato.

El primer modo de control, a través de la temperatura, esta implementado de la siguiente manera (ver anexo 2, líneas 392-404): al inicio se ha de alimentar el DHT11 para posteriormente leer la temperatura. Una vez leída, se compara con el valor límite establecido por el usuario en la configuración inicial. Cabe decir que este valor está almacenado en la memoria EEPROM del micro, ya que, si no fuera así, al entrar en modo suspensión se borraría de la memoria flash no pudiendo ser recuperado. Si hay menor temperatura en la habitación que la impuesta por el usuario se activa el pin correspondiente para enclavar el relé, enchufando la calefacción. En el caso de que la temperatura no sea inferior al límite, se comprueba si la temperatura es superior al valor extremo más un margen establecido de tres grados. En tal caso se desactiva el relé de enclavamiento para apagar la calefacción.

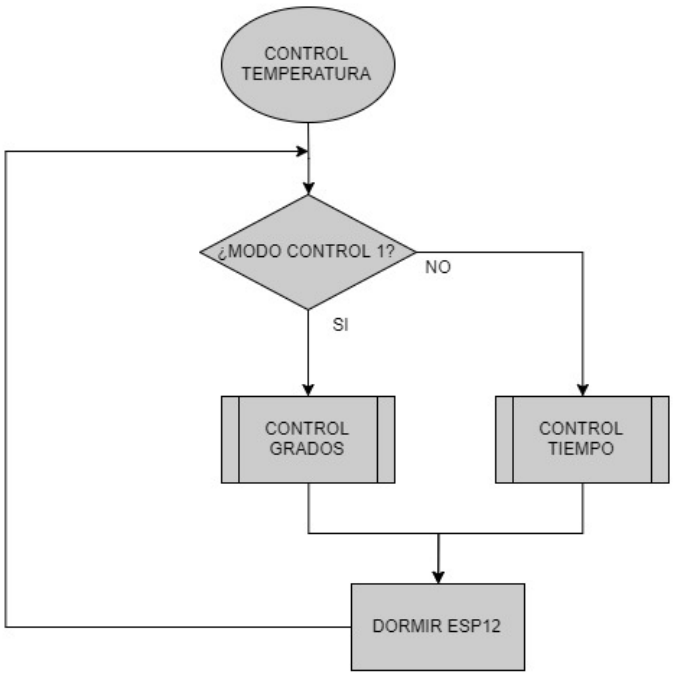


Figura 22: Diagrama de flujo del control de la calefacción

La segunda posibilidad de control del termostato es por tiempo (ver anexo 2, líneas 408-418), pudiendo elegir los intervalos del día en el que el cliente desea tener encendida la calefacción. Esta opción es equivalente al control por tiempo de las luces en vacaciones que será explicado más adelante.

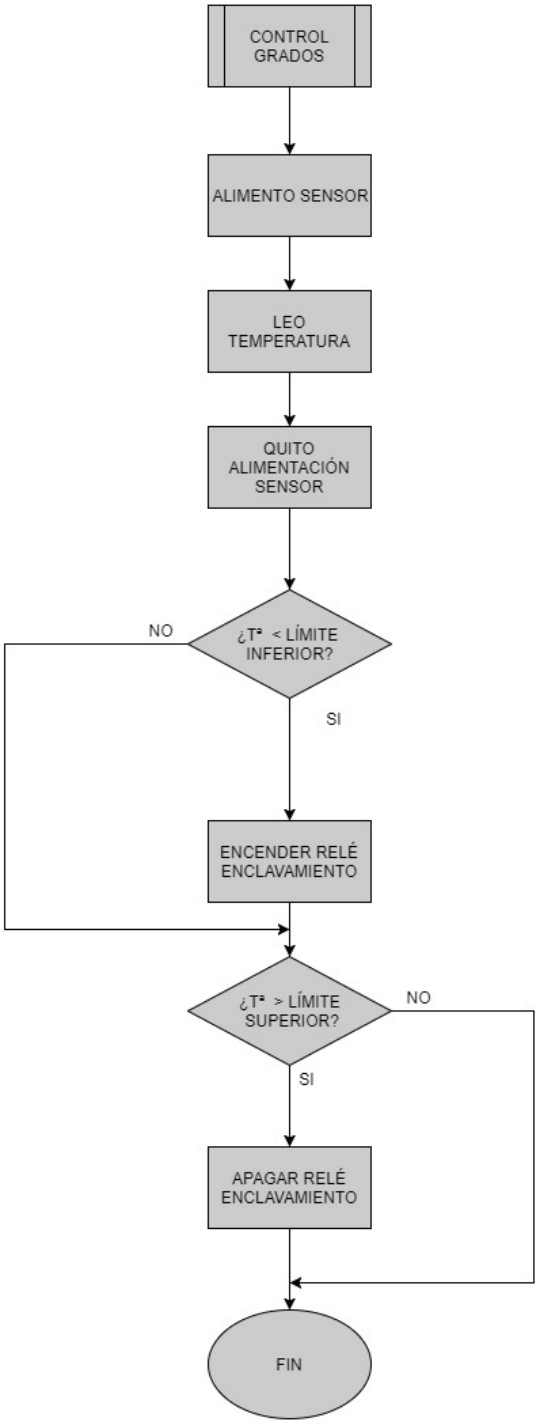


Figura 23: Diagrama de flujo del control según temperatura

3.3.3. CONTROL PERSIANA VACACIONES

Este modo (Figura 24) proporciona la posibilidad de simular presencia en la vivienda cuando el usuario se encuentra de vacaciones y va estar fuera un largo período de tiempo. Esta simulación a través del movimiento de la persiana es posible con dos modos distintos según la decisión del cliente. Una primera opción incluye la elección de los tiempos de subida y de bajada por parte del dueño, y, la segunda, un control aleatorio. En este modo existe la implementación del modo ahorro del micro, puesto que lo que se controla es el tiempo, admitiendo un margen error ocasionado por este ahorro.

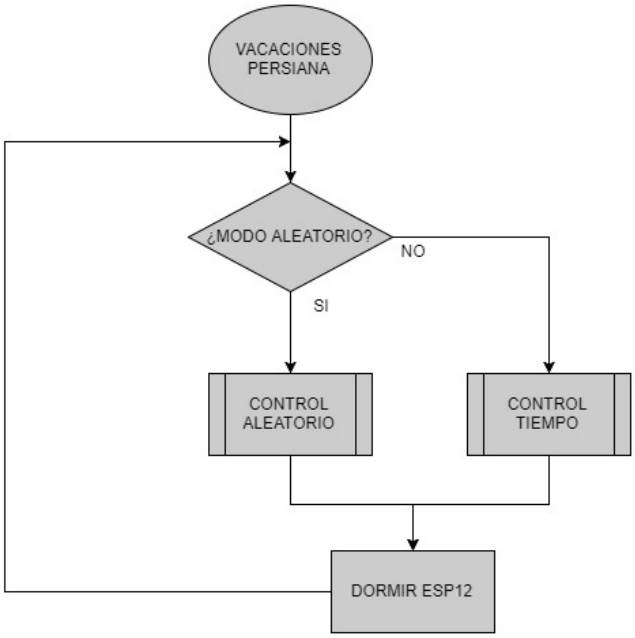


Figura 24: Diagrama de flujo del control de la persiana en vacaciones

Al comenzar la aplicación, en cada bucle, el modo vacaciones persiana (Figura 24) distingue entre si está almacenado en memoria el modo de control aleatorio o el de tiempo. Si es este último caso pasa a ejecutar el diagrama de flujo inferior (ver anexo 2, líneas 473-499). El funcionamiento del mismo consiste en una comparación de la hora actual con la hora fijada por el usuario. En el caso de que no esté activado el final de carrera de persiana arriba y la hora actual esté dentro del intervalo de hora de subida se activa el relé 1 que mueve el motor de la persiana en un sentido tal que suba la persiana por completo. Este motor se parará al saltar la interrupción producida por el final de carrera de persiana arriba.

Posteriormente se compara la hora actual con la hora de bajada. Si está dentro del intervalo procede a activar el relé 2 que hace al motor de la persiana bajar la misma.

En el momento en el que entra en juego la comparación de tiempos, es necesario poseer una fuente de información que nos permita saber la hora actual. El ESP12 te brinda más de una opción (reloj interno, por ejemplo), pero la decidida en este proyecto es la utilización de una librería específica (Time.h) que brinda la posibilidad de, a través de internet, extraer la hora en cada instante, y poder así utilizarla para comparaciones y cualquier función con total precisión. De esta manera se elimina el problema del reinicio del reloj cada vez que el micro entra en modo Sleep, lo cual fue un problema en los inicios del proyecto.

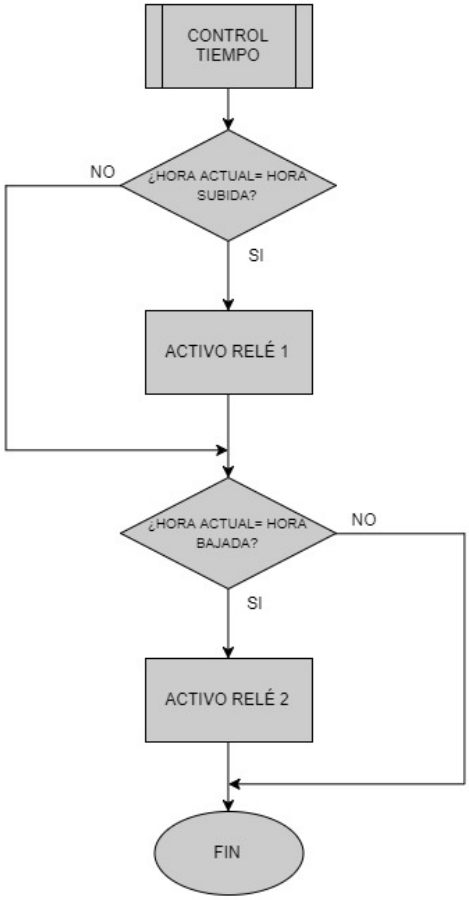


Figura 25: Diagrama de flujo del control por tiempo de la persiana

El control aleatorio (Figura 26) consta de un funcionamiento básico (ver anexo 2, líneas 436-470). Está programado para que, como se muestra en el diagrama de flujo, la primera vez de cada ciclo se elige aleatoriamente una hora de subida y bajada de la persiana. La de subida está acotada entre las 8 y las 11, mientras que la de bajada es 14 horas después que la de subida.

Una vez fijada está hora por el programa implementado, el funcionamiento de la aplicación es equivalente al del modo control de tiempo establecido por el usuario.

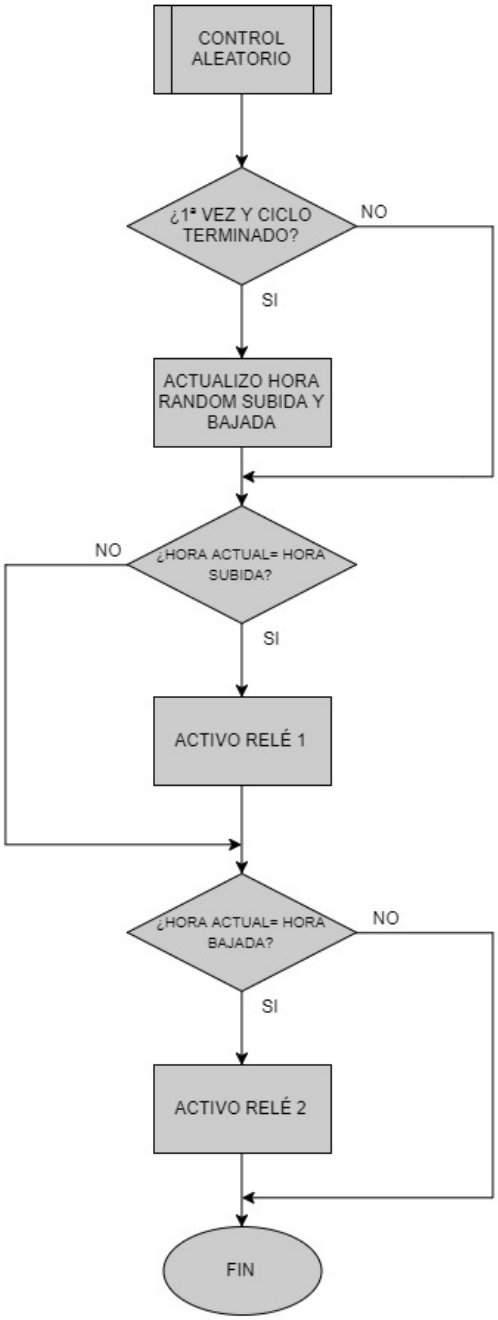


Figura 26: Diagrama de flujo del control la persiana en modo aleatorio

3.3.4. CONTROL LUCES VACACIONES

Este apartado tiene el mismo objetivo que el anterior modo. Se ha implementado por el hecho de que, no en todas las viviendas hay persianas gobernadas por motores. Sin embargo, el control de las luces (Figura 27) será posible en todo tipo de lugares.

El concepto teórico y a nivel de código de la aplicación es el mismo. Sin embargo, en este caso el hardware que se gobierna no son los relés estándar, sino el relé de enclavamiento, que permite un mayor ahorro consumiendo corriente sólo en los momentos de conmutación.

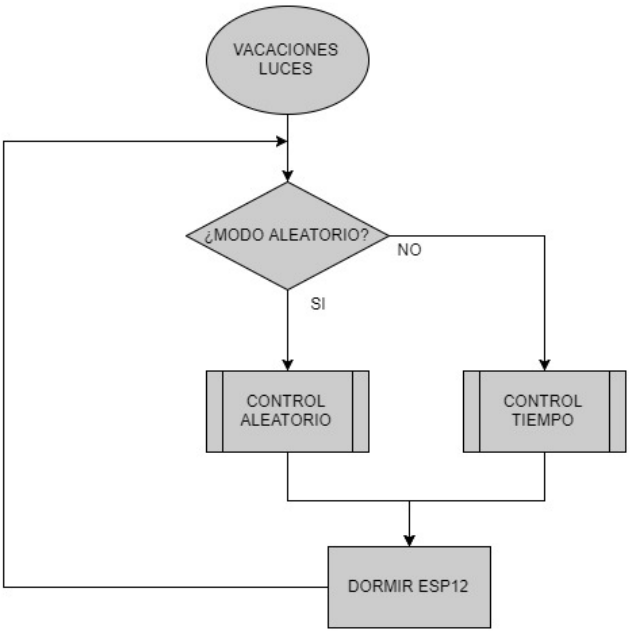


Figura 27: Diagrama de flujo del control de las luces en vacaciones

Como se aprecia en el diagrama de flujo inferior (ver anexo 2, líneas 540-555), en una primera comparación se mira si la hora de encendido fijada por el usuario coincide con la hora actual. En tal caso, si las luces están apagadas, el ESP12 activa el pin correspondiente durante dos segundos para enclavar el relé que cierre el circuito de potencia, donde se encuentra el sistema de alumbrado.

Esta activación sólo se realiza la primera vez que se entra en el intervalo de activación de las luces. Esto es gracias a una variable auxiliar que nos indica el estado de la salida y que se guarda en memoria cada vez que es modificada. De este modo, aunque el micro entre en modo sleep, al reiniciarse la aplicación el sistema sabe el estado en el que se encontraba previamente.

De igual manera sucede con el intervalo en el que las luces deben estar apagadas. Esta vez se mira que las luces se hallen encendidas y la comparación temporal. Si se cumplen estas dos premisas el ESP12 mediante una de las GPIO desactiva el relé de enclavamiento. Este consumo para la desactivación se producirá sólo la primera vez en la que se encuentre dentro del intervalo de apagado, consiguiendo un funcionamiento lo más eficiente posible.

Tras el bucle del control de tiempo de las luces (Figura 28) el micro entra en modo suspensión durante cinco minutos.

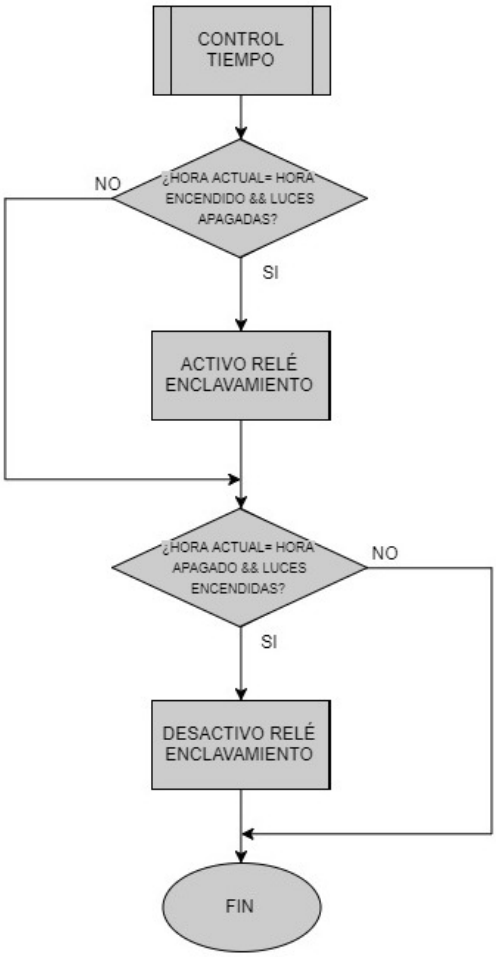


Figura 28: Diagrama de flujo del control de las luces por tiempo

3.3.5. CONTROL PERSIANA

El funcionamiento de este modo es similar al de Vacaciones Persiana, pero en este caso la aplicación se centra en una modalidad de funcionamiento por tiempo fijada por el usuario o un modo manual, donde el cliente controla la persiana desde su dispositivo móvil o a través de los pulsadores habilitados en el producto.

3.3.6. MONITORIZACIÓN VARIABLES

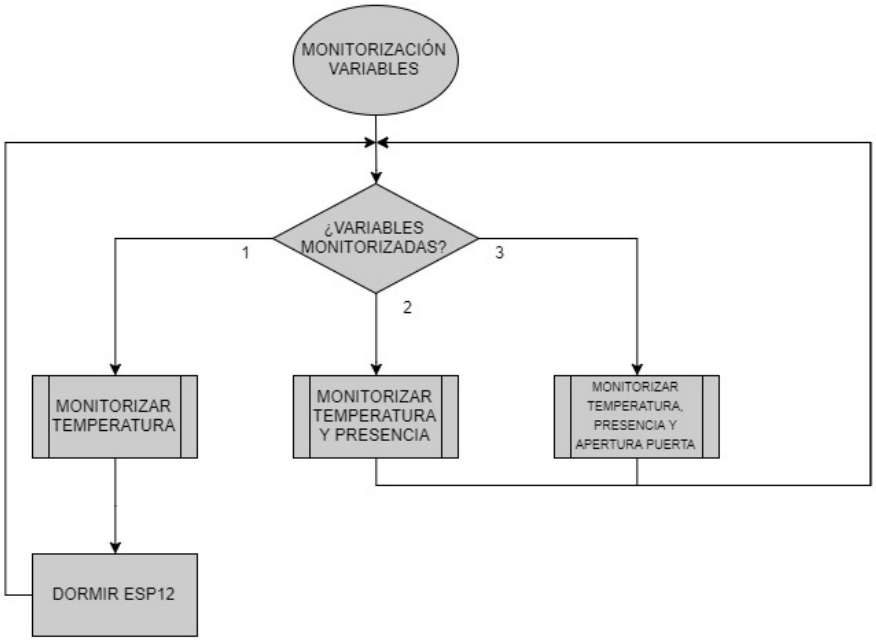


Figura 29: Diagrama de flujo del modo monitorización de variables

Este modo (Figura 29) permite utilizar el producto como un sensor que puede actuar de dos maneras. Por un lado, hacer la función de una estación meteorológica que almacene los datos en la nube de temperatura y humedad (ver anexo 2, líneas 1560-1569). Y, por otro lado, consistir en un dispositivo de seguridad que le permite al usuario conocer en todo momento si hay presencia en su casa y/o se abre una ventana o puerta (según la ubicación del producto). En el caso de detectar presencia, a través de la nube se le enviará al cliente un correo de alarma que le ponga en conocimiento de esta información al instante (ver anexo 2, líneas 585-590).

Debido a las distintas exigencias de cada modo, sólo en el primer caso se podrá utilizar el modo suspensión del microcontrolador, mientras que en los otros dos será necesario tener la aplicación funcionando en todo momento.

La función de monitorización de la temperatura se ejecuta a través de la plataforma Thinger.io. A través del código y valiéndose de la librería habilitada en Arduino IDE para esta plataforma se crea una función que ejecuta la siguiente secuencia cada cinco minutos: Alimentación del sensor a través de la GPIO habilitada para ello, lectura de las variables temperatura y humedad, almacenamiento en el data bucket de la nube y, finalmente se retira la alimentación del sensor. La orden de ejecución de esta función la da la plataforma según el intervalo de tiempo que se haya fijado previamente. Estos datos de temperatura y humedad se pueden ver tanto en directo como una gráfica con el historial de las medidas.

3.3.7. CONTROL PRESENCIA

Este control permite al usuario en todo momento que esté fuera de su casa conocer si alguien irrumpe en ella. Está diseñado para que salte una interrupción en el micro al detectarse presencia, ya sea manifestada por el sensor PIR o por la apertura de una puerta o ventana (según dónde se sitúe el producto).

Esta interrupción hace saltar un aviso en la nube que automáticamente mandar un correo a la dirección de email del cliente avisándole que ha sido detectado un intruso (ver anexo 2, líneas 585-590).

Como inconveniente de esta aplicación es la necesidad de estar el micro siempre alerta, no siendo posible conseguir un gran ahorro en el consumo, como es el caso de otros modos explicados anteriormente, donde hay un margen de tiempo entre actuación y actuación que permite dormir el módulo obteniendo una mayor eficiencia.

3.3.8. TELEFONILLO

Este es uno de los modos más sencillos del producto (ver anexo 2, líneas 1620-1630). Su función es básica y clara, conseguir acceder a la urbanización o portal a través de la aplicación móvil. Para ello, el ESP12 recibe la orden de activar el relé estándar por medio de la nube. Se trata de una comunicación con la siguiente secuencia: Dispositivo móvil – thinger.io – ESP12 – relé – telefonillo. Al igual que en el caso de monitorización de temperatura, cuando se recibe el estímulo creado por la aplicación (al pulsar el botón de abrir puerta en la pantalla) se ejecuta la función asociada en el código, que consiste simplemente en la activación durante tres segundos del relé al que está conectado el circuito del telefonillo.

3.3.9. CONFIGURACIÓN

Gracias a esta parte de la aplicación se puede dar la comunicación entre la máquina y el usuario, ya que es la que lleva implementada todo el código que permite la lectura de los caracteres introducidos por el cliente en la consola, así como la ejecución de la máquina de estados que da la opción a la persona de elegir el modo, la configuración del wifi y todas las variables necesarias para el funcionamiento específico de cada uno de los clientes.

Esta configuración (Figura 30) se ejecuta únicamente en el momento en el que el cliente acciona durante 5 segundos el pulsador SWITCH2, momento en el cual se inicia el proceso detallado por el diagrama de flujo.

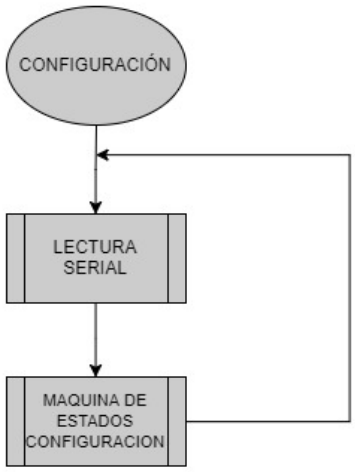


Figura 30: Diagrama de flujo general de la configuración

La función Lectura serial (Figura 31) está implementada de la siguiente manera (ver anexo 2, líneas 637-652): se comprueba que el serial de la aplicación Arduino se encuentre disponible, para posteriormente comprobar si hay almacenada información en él. Si la hay se procede a la lectura carácter a carácter y se almacenan en una variable de tipo String. Una vez que no queda nada almacenado, termina la función para ser utilizada esta variable String guardada.

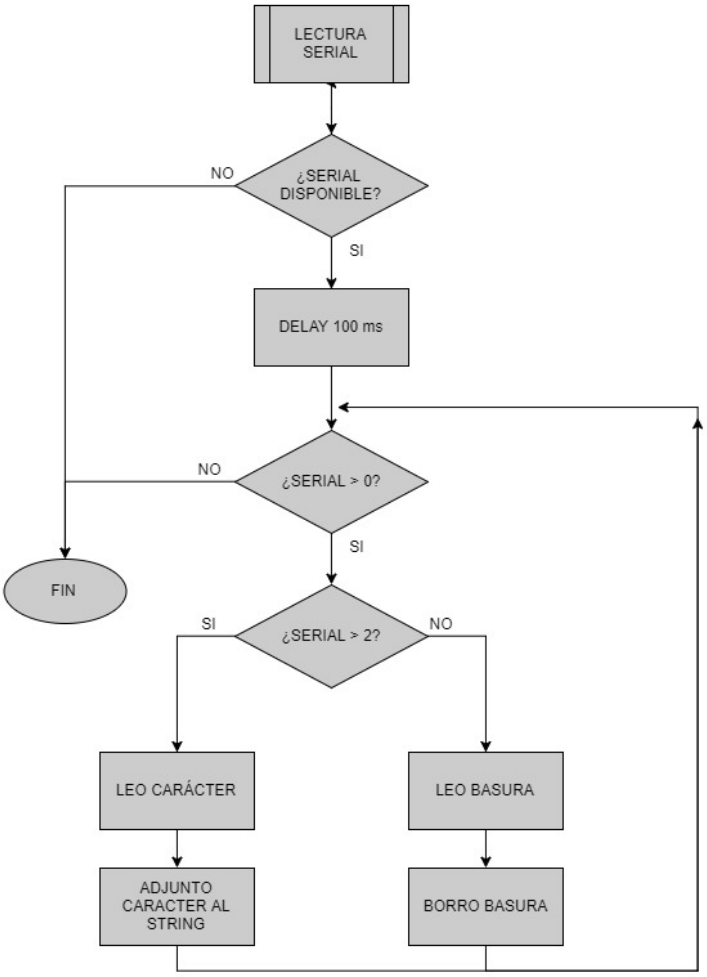


Figura 31: Diagrama de flujo de la función de lectura del puerto serie

El diagrama de flujo correspondiente a la máquina de estados de la configuración (Figura 32) representa una parte importante de la aplicación y de las más extensas de la misma (ver anexo 2, líneas 892-1437). Alberga todo el proceso inicial previo al funcionamiento habitual del producto según el modo escogido.

La primera vez que se ejecuta siempre pasa al estado credenciales (ver anexo 2, líneas 894-984). En este estado se lee a través de la consola el usuario y contraseña del wifi del usuario. Una vez leídos se guardan en la memoria para ser cargados siempre que se reinicie el microcontrolador. Como último paso de este estado se le pide al cliente que elija el modo de funcionamiento que desea para el producto. Dependiendo de esta elección salta a un estado u otro. En cada estado hay un proceso distinto con unas variables finalmente almacenadas en una posición de la memoria concreta, de forma que, a la hora de cargar la información, sabiendo de qué estado se procede se carguen las variables que corresponden.

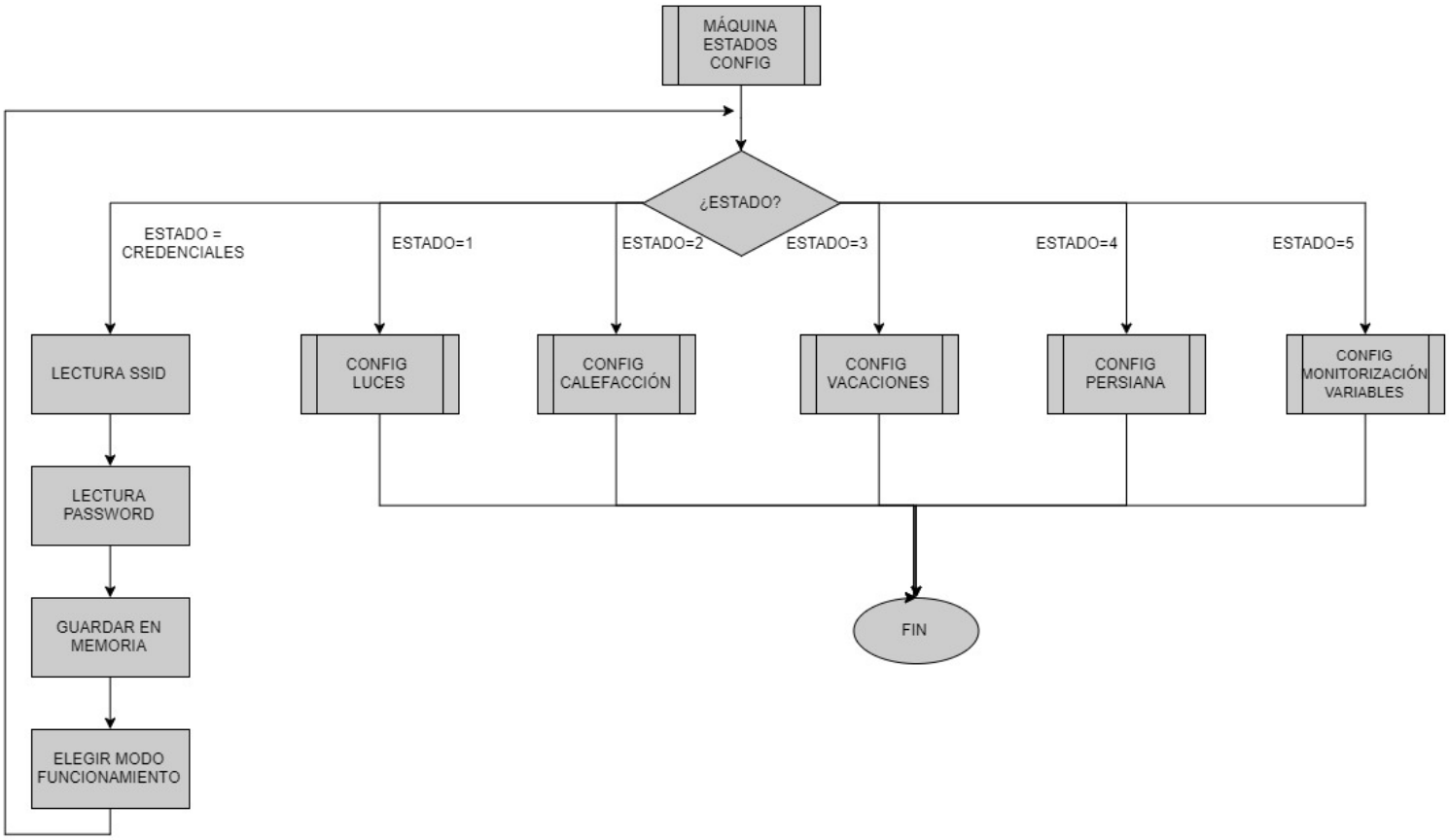


Figura 32: Diagrama de flujo de la máquina de estados del modo configuración

Si el usuario desea controlar las luces, la máquina de estados tras la lectura del modo pasa a ejecutar la sección de código representada por este diagrama de flujo (Figura 33) (ver anexo 2, líneas 986-1021).

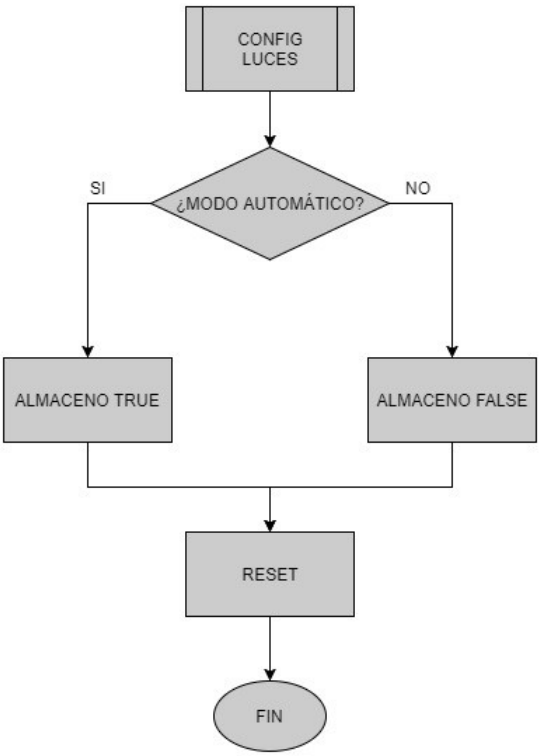


Figura 33: Diagrama de flujo de la configuración del modo de control de las luces

Se le pregunta a través de la consola al cliente sobre si prefiere el modo automático o un control manual de las luces de la habitación. Según la respuesta leída por la función Lectura Serial se almacena en memoria true o false. Este dato será recuperado por el micro durante el funcionamiento normal de la aplicación cada vez que se reinicie el ESP12.

Si el usuario ha elegido el modo de funcionamiento de vacaciones, ya sea de luces o de la persiana se ejecuta el código representado por el siguiente diagrama de flujo (Figura 34) (ver anexo 2, líneas 1156-1270).

Se pide en un primer momento si se quiere un control aleatorio de la hora. Si la respuesta es sí se almacena en memoria true, mientras que si es no se procede a solicitar al usuario las horas en las que quiere actuar sobre las luces o la persiana. Cabe destacar que se han implementado, para una mayor robustez de la aplicación, los pasos a seguir si la respuesta por parte del cliente no es la adecuada. En este caso, si los números introducidos no entran en el rango propio de la hora se le pide repetir e introducir un número coherente.

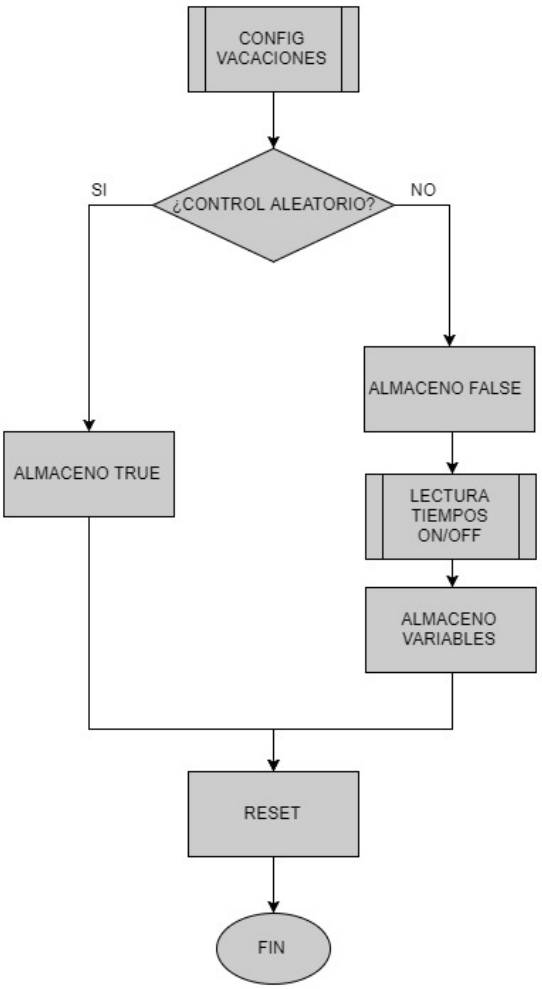


Figura 34: Diagrama de flujo de la configuración del modo simulación de presencia en vacaciones

En cuanto a la configuración de la calefacción (Figura 35) se le pide al usuario en primera instancia que introduzca el modo de control (ver anexo 2, líneas 1023-1154). Si el número leído es 1 pasa a configurar la temperatura límite, mientras que, si el número leído es 2, el cliente deberá fijar los horarios de encendido y apagado de la calefacción. Tras estas lecturas se procede a almacenar en memoria la configuración elegida. Por último, se reinicia el ESP12 para comenzar con el funcionamiento normal del modo escogido.

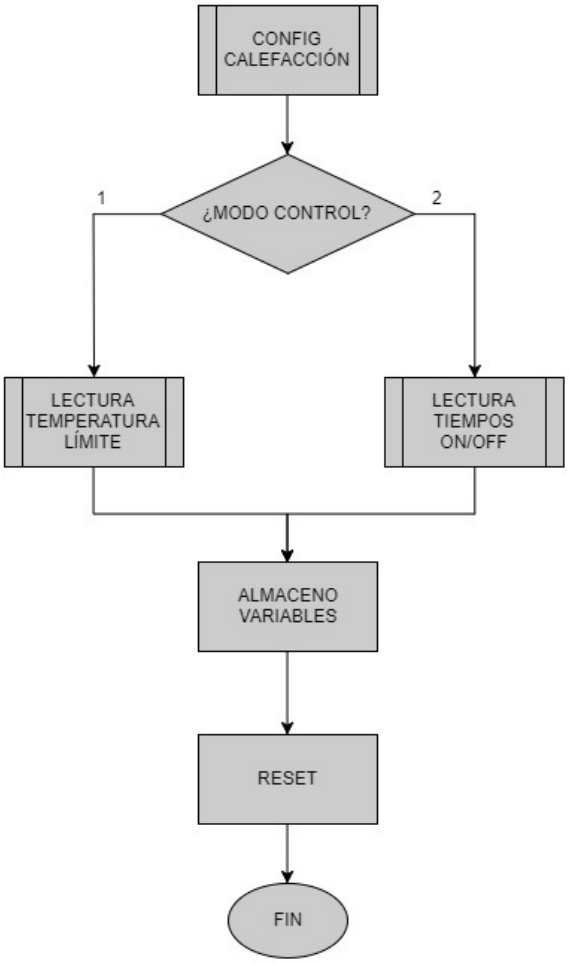


Figura 35: Diagrama de flujo de la configuración de la calefacción

En esta última rama de la configuración representada en el diagrama inferior (Figura 36) se sigue una estructura totalmente secuencial (ver anexo 2, líneas 1383-1435). Inicialmente se le pide al usuario que introduzca el submodo a ejecutar, almacenando el dato leído en memoria. Por último, se pide una frecuencia de lectura de la temperatura y humedad entre 1 minuto y 1 hora. Si el dato leído es correcto se almacena y se pasa a la ejecución de la aplicación habitual.

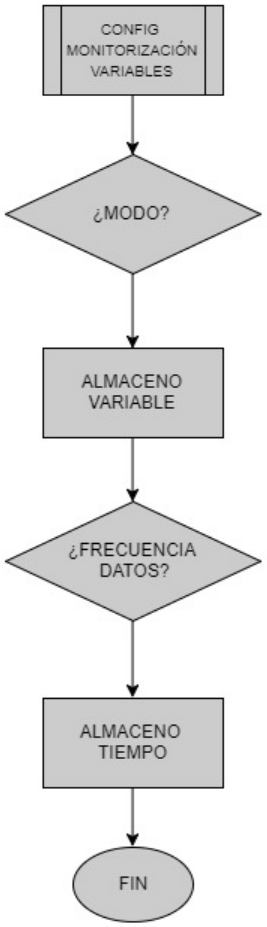


Figura 36: Diagrama de flujo de la configuración del modo monitorización de variables

4 PLATAFORMA WEB

4.1. ANÁLISIS PREVIO DE PLATAFORMAS

Desde un primer momento, uno de los objetivos de este proyecto ha sido conseguir una comunicación inalámbrica entre el dispositivo a diseñar y el teléfono móvil u ordenador. Por ello, se ha planteado como medio para conseguir esta comunicación utilizar la nube.

Una vez planteada la necesidad de utilizar una plataforma se han analizado algunas de carácter gratuito que encajan con la aplicación. De forma especial han sido examinadas tres plataformas: Easyiot, Thingspeak y Thinger.io, eligiendo finalmente ésta última.

Easyiot ha sido la menos analizada porque, a pesar de cumplir la condición de ser gratuita, desde un primer momento y de tener varios ejemplos, no había demasiados comentarios positivos en Internet acerca de una fácil utilización.

En segundo lugar, se ha analizado Thingerspeak, la cual ha sido una plataforma con posibilidades de ser elegida hasta el último momento. Esto es debido a que ofrece la posibilidad de trabajar y exportar datos a Matlab, lo que supone una amplia capacidad de análisis de los posibles datos obtenidos de los sensores de la aplicación. Como principal inconveniente está el hecho de la dificultad de manejo de la misma, puesto que iba a suponer una mayor inversión de tiempo que no era asumible debido a la naturaleza de este proyecto que incluye tanto diseño de hardware como de software. La falta de tutoriales y menor desarrollo en el entorno de Arduino IDE han sido determinante en la decisión.

Finalmente, la plataforma elegida es Thinger.io. Se trata de una plataforma española de código abierto que ofrece gratuitamente gran parte de sus servicios, suficientes para la elaboración de este proyecto. Consta de una comunidad grande de usuarios y de librería en Arduino IDE que permite una programación sencilla y práctica, con la que establecer comunicación y gobernar el ESP12 desde la nube sin problemas. A continuación, se detalla de manera más profunda esta plataforma.

4.2. COMUNIACIÓN ESP8266 CON LA PLATAFORMA THINGER.IO

La programación es sencilla, ya que dispone de una librería en Arduino IDE compatible con el ESP8266, aspecto de gran interés en esta aplicación.

La librería utilizada es *ThingerESP8266.h*. Para establecer la comunicación entre ESP8266 y la plataforma son necesarios los siguientes pasos:

- Definir nombre de usuario, ID y credenciales de la cuenta que sea utilizada (ver anexo 2, línea 87).

- Declarar en el setup del código los distintos objetos con funciones asociadas que son las que gobiernan el micro a través de la plataforma (ver anexo 2, líneas 1560-1635).

- Configurar en la plataforma de thinger.io las gráficas y datos que se quieren visualizar y almacenar.

Con esta plataforma es posible subir datos a la misma y consultarlos desde el dispositivo móvil, pero también te da la posibilidad de actuar sobre un relé (Figura 37 y 38).

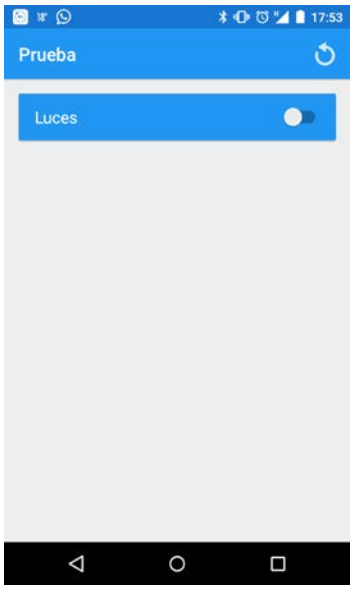


Figura 37: APP en el modo de control de luces

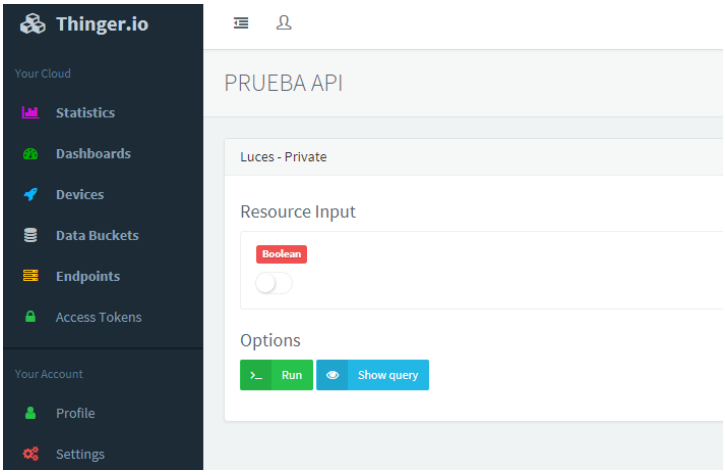


Figura 38: Control del relé desde la nube

A continuación (Figura 39), se muestra el diagrama de funcionamiento de Thinger.io:

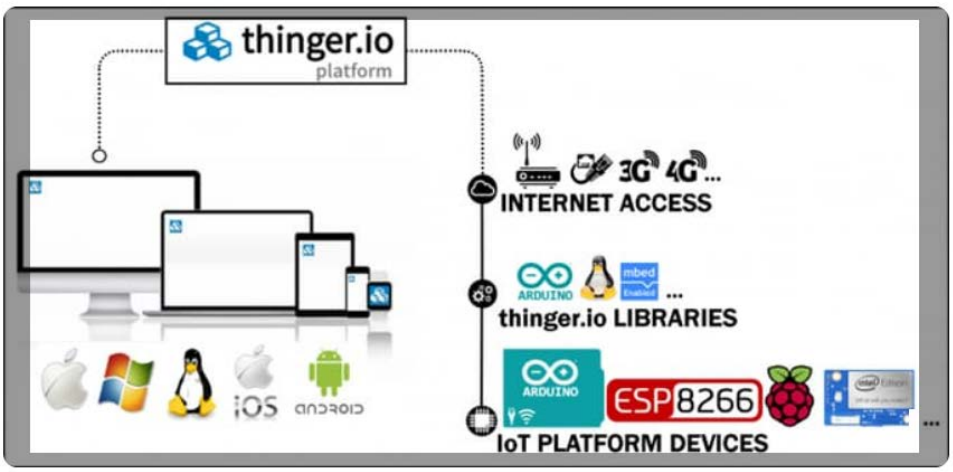


Figura 39: Diagrama de funcionamiento de la plataforma Thinger.io

A partir de este esquema se aprecia lo siguiente:
Para accionar por ejemplo un relé gobernado por un módulo ESP8266, el proceso que sigue la información parte de una solicitud por parte del usuario a través de un ordenador, Tablet o smartphone. La plataforma antes de llegar a dar la orden al ESP8266 pasa dos filtros como son la comprobación de usuario y contraseña y la traducción a un lenguaje que entienda el dispositivo final por medio de librerías. En este proyecto la librería utilizada es la implementada en Arduino IDE, teniendo como dispositivo final el ESP8266.

Thinger.io es gratuita para cualquier usuario, pero cuenta con las siguientes limitaciones (Figura 40):

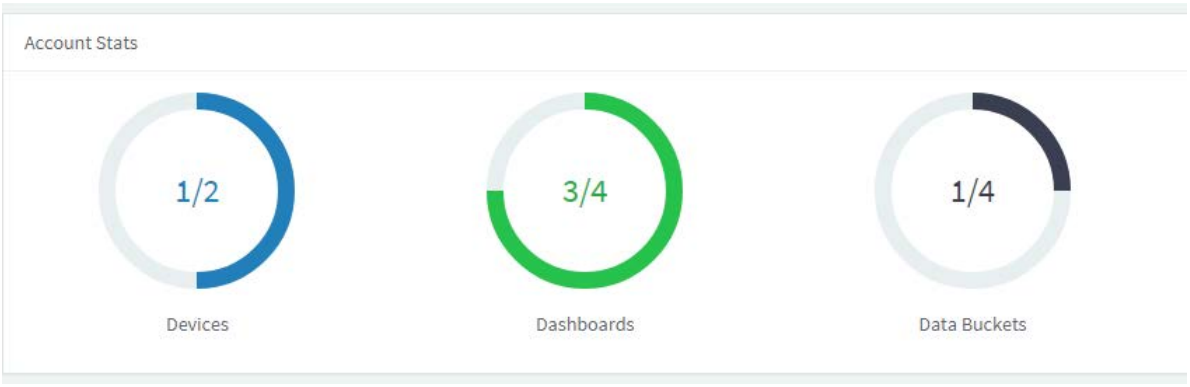


Figura 40: Limitaciones de Thinger.io en cuanto a número de dispositivos

- Permite conectar un máximo de 2 dispositivos.
- No hay limitación en los recursos de cada dispositivo. Esto es que es posible medir varios parámetros con los distintos sensores que haya conectados.
- Los valores de los parámetros recibidos se pueden almacenar en la nube a través de Data Buckets (Máximo 4), siendo posible guardar un dato cada minuto como mínimo.
- Los datos de los sensores pueden ser visualizados hasta un máximo de 4 pantallas gráficas. Estos datos pueden proceder del Data Bucket o de la lectura en tiempo real del sensor.

Date	Humedad	Temperatura
2017-06-29T13:26:53.165+0200	32	26
2017-06-26T18:48:05.118+0200	55	30
2017-06-26T18:47:05.117+0200	57	30
2017-06-26T18:46:05.122+0200	56	40
2017-06-26T18:45:04.842+0200	59	30
2017-06-26T18:43:41.660+0200	60	30
2017-06-26T18:42:05.410+0200	58	30
2017-06-26T18:39:46.546+0200	53	30
2017-06-26T18:38:46.549+0200	53	30
2017-06-26T18:37:46.538+0200	55	31

Tabla 3: Datos de un databucket según información proporcionada por el sensor DHT11

5 IMPLEMENTACIÓN Y PRUEBAS

En este apartado se explica el proceso que se ha seguido a la hora de llevar el diseño del hardware y del firmware a un producto físico real.

5.1. SOLDADURA Y ENSAMBLAJE DEL PRODUCTO

La placa de circuito impreso (Figura 41) ha sido hecha físicamente por las máquinas especializadas del taller de la Universidad de Zaragoza a partir de los ficheros de Circuit Maker (programa de diseño de PCB). Una vez trazadas las pistas el primer paso es soldar las vías una a una, e ir comprobando que efectivamente hay continuidad entre la cara top y bottom en esos puntos.

Ya con la base lista (pistas y vías) se van soldando los componentes siguiendo un criterio intermedio entre facilidad espacial a la hora de soldar y bloques del sistema (alimentación, sensores, etc.). Así pues, en una primera fase se sueldan los componentes smd incluido el módulo ESP12. Una vez verificado que no hay cortocircuitos se procede a la soldadura de componentes thd siguiendo una lógica por bloques, empezando por los relacionados con la alimentación (reguladores, fichas de entrada de tensión continua, conector micro USB para tensión continua).

En este momento, en el que ya se dispone de una alimentación básica y de los circuitos necesarios para el correcto funcionamiento del microcontrolador, se inician las pruebas de comunicación serie con el ordenador para comprobar que el módulo Wifi no tiene ningún error de fábrica ni cuenta con defectos en la soldadura. Comprobado esto se pasa al siguiente bloque del producto: los sensores.

El procedimiento seguido ha sido soldar y comprobar sensor a sensor para evitar errores en cadena. Una vez verificado el correcto funcionamiento de los mismos se suelda en última instancia los relés.

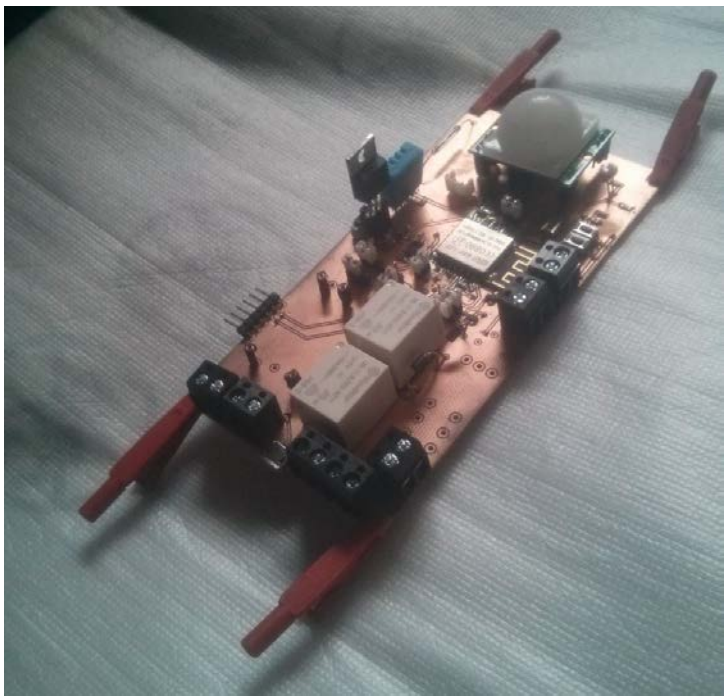


Figura 41: Vista en perspectiva de la PCB

5.2. PUESTA A PUNTO DEL CÓDIGO EN EL PROTOTIPO

La última parte del proyecto consiste en llevar a cabo una comprobación final de que el diseño, tanto de hardware como de programación, tiene como resultado el comportamiento deseado.

Las pruebas que mejores resultados han tenido son:

- La aplicación del telefonillo (Figura 42 y 43) funciona a la perfección con un retraso mínimo desde que se acciona en la aplicación móvil hasta que el relé se activa abriéndose la puerta del portal. En el caso de la figura 42 se ha simulado lo que sería el telefonillo mediante un led y un pequeño altavoz piezoeléctrico alimentados a través de un Arduino UNO.

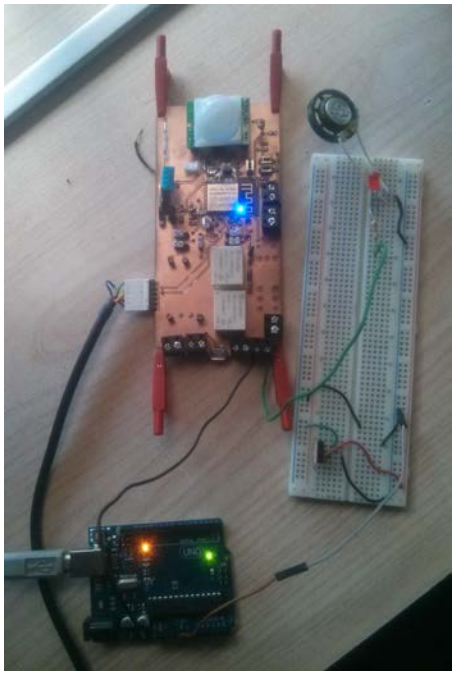


Figura 42: ESP8266 controlando una carga

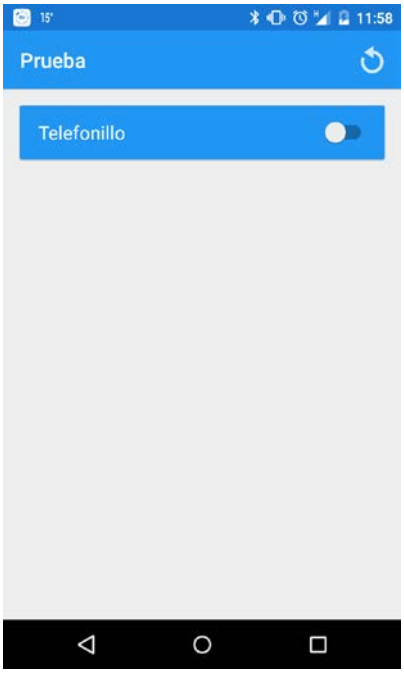


Figura 43: APP en modo telefonillo

- El modo de estación meteorológica ha tenido también un buenísimo resultado (Figuras 44 y 45), almacenando los datos en la nube de temperatura y humedad cada minuto con una precisión aceptable teniendo en cuenta el bajo coste del sensor DHT11. En la APP (Figura 45) únicamente se puede ver la temperatura y humedad en el instante en el que se actualiza. Sin embargo, si queremos hacer un seguimiento temporal hay que recurrir a la nube en el PC.

Figura 44: Gráfica con datos de temperatura y humedad tomados por el sensor DHT11

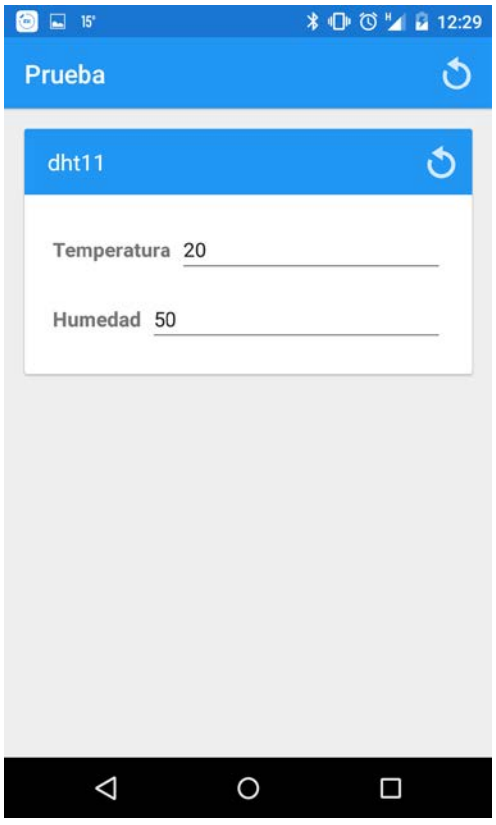


Figura 45: APP en modo monitorización de variables

-Los controles manuales de las GPIO a través de la aplicación móvil. Este buen resultado se aprecia en los modos de control de luces, de persiana y calefacción.

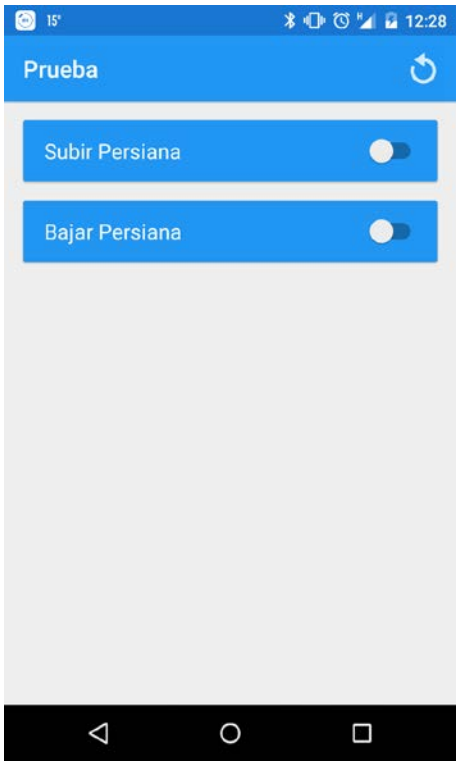


Figura 46: APP en modo control de la persiana

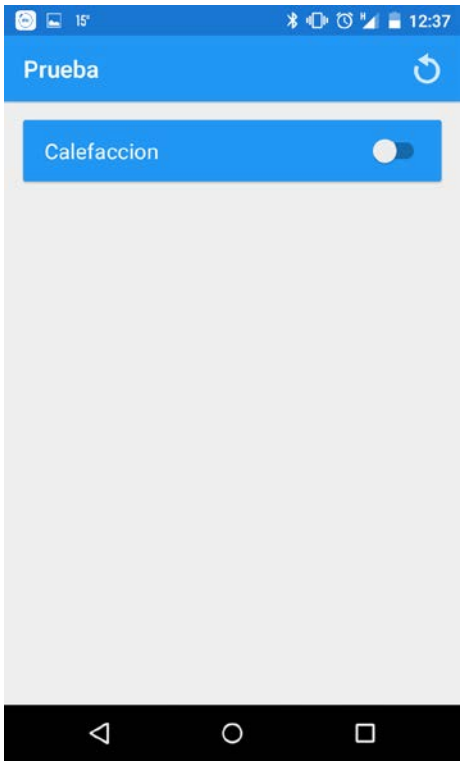


Figura 47: APP en modo control de la calefacción

-La parte de la configuración e interacción del usuario con la consola del Arduino IDE ha funcionado correctamente para todos los modos. En la figura 48 se expone un ejemplo con el modo de monitorización de variables. La consola va pidiendo en orden una serie de parámetros a fijar por el usuario, leyéndolos y guardándolos en memoria.

```
Cambiando a modo configuracion...
Introduzca la SSID:
ssid leido
Introduzca el password:
password leido
Credenciales guardadas
Introduzca el numero de modo de funcionamiento:
modo 1: Control de luces
modo 2: Control de temperatura
modo 3: Persiana en vacaciones
modo 4: Luces en vacaciones
modo 5: Control de la persiana
modo 6: Monitorizacion de sensores
modo 7: Control de presencia
modo 8: Telefonillo
Modo elegido:
6
Modo de funcionamiento guardado
Introduzca el numero de modo de variables monitorizadas:
1: Temperatura y humedad
2: Temperatura,humedad y presencia movimiento
3: Temperatura,humedad y apertura de puerta
Numero de variables a monitorizar:
1
Variables monitorizadas guardadas
Introduzca un numero de segundos entre envio de datos a la nube (min 60, maximo 3600):
Tiempo entre envio de datos a la nube:
60 segundos
Tiempo entre envio de datos guardado

 ets Jan  8 2013,rst cause:2, boot mode:(3,6)
```

Figura 48: Consola del Arduino IDE en modo configuración

En cuanto a las pruebas cuyo resultado ha sido menos satisfactorio destacan:

- El sensor de presencia no ha dado el resultado esperado porque en determinados momentos ha detectado presencia cuando en realidad no había ninguna persona cerca, por lo que no sería demasiado fiable para el caso de la alarma antirrobo cuando el cliente este en vacaciones. Como mejora se debería adquirir un sensor de mejores características que proporcione mayor seguridad en los resultados. En la figura 49 se muestra la notificación que llega al correo en el momento en el que el sensor PIR detecta una presencia en la vivienda.

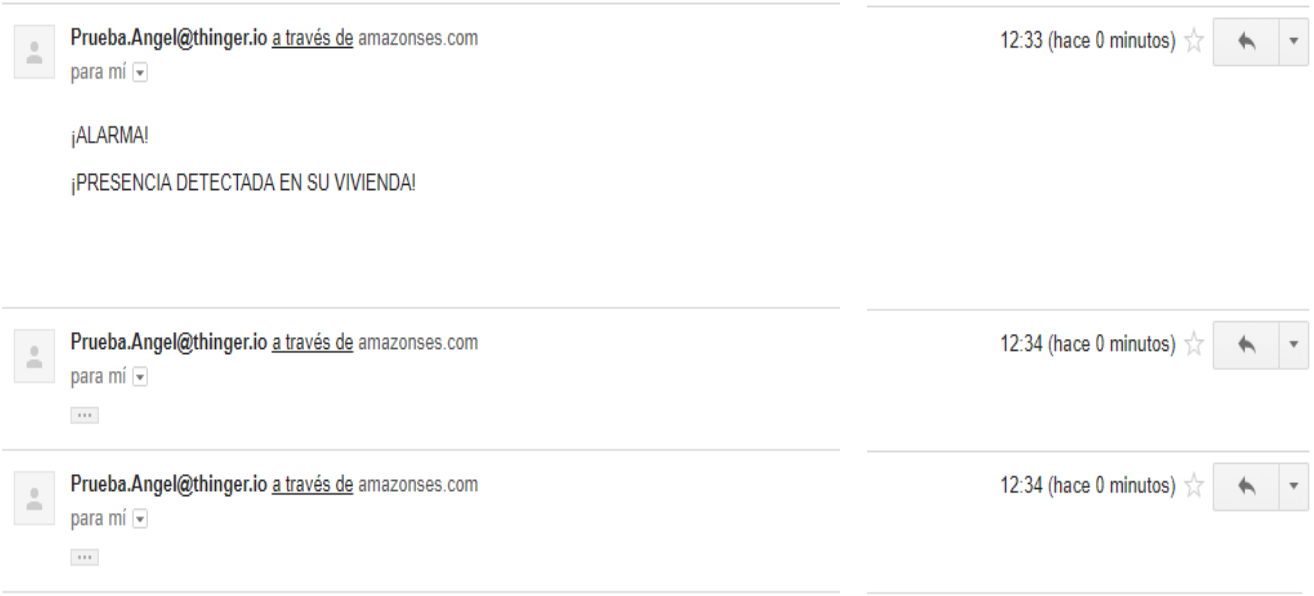


Figura 49: Notificación en el correo de la presencia detectada en la vivienda

- El Relé Reed funciona correctamente, pero es necesario colocar un imán potente en la puerta para que la aplicación siga el comportamiento deseado. Esto incrementará el coste del producto final.

6 CONCLUSIONES

Como conclusión final, es correcto apuntar que este TFG es una prueba en la que ha sido necesario aplicar muchos de los conocimientos obtenidos durante el grado, diseño electrónico, programación estructurada incorporando interrupciones, utilización de memoria del micro, máquinas de estados, etc.

También señalar que este proyecto ha supuesto un proceso completo en el desarrollo de un producto, empezando por una idea, eligiendo lo necesario para llevarla a cabo (Microcontrolador a utilizar, plataforma IOT, casos de uso y sensores), efectuando el diseño electrónico y elección de los componentes y sensores, para después hacer un presupuesto. Posteriormente, realizar la programación del micro y proceder a la fabricación de la placa soldando los componentes. Y, por último, poner a punto el producto para que funcione según el objetivo inicial de la aplicación.

6.1. RESULTADOS OBTENIDOS

Llegados a este punto de la memoria, es momento de ser crítico y objetivo con el trabajo realizado. Este proyecto ha sido de larga duración, con muchos baches y dificultades por el camino y sobre todo caracterizado por un trabajo constante que ha tenido como recompensa el aprendizaje y asimilación de varios puntos importantes tratados en el grado, así como otros puntos que no se han tocado y poseen gran interés.

Estos temas interesantes han sido por ejemplo el mundo de las telecomunicaciones, con la comunicación Wifi en especial. Pero también todo el campo de la domótica, sensado y el diseño electrónico partiendo desde una idea y llegando hasta su última fase de fabricación y puesta en marcha.

Objetivamente, la sensación final es de un trabajo satisfactorio ya que, finalmente, el producto ha ejecutado prácticamente todas las funciones para las que ha sido diseñado, desde el caso del uso del telefonillo, pasando por las interrupciones de las bajadas y subidas de persiana; e incluso del modo ahorro de energía despertando el micro cada cinco minutos para mandar datos a la nube y crear una estación meteorológica casera.

Siendo críticos y conscientes de que ha sido el primer diseño real de un producto electrónico, ha habido fallos de cálculo posteriormente corregidos, errores a la hora de soldar los componentes, debido a la inexperiencia en este campo, que han sido solventados con un número mayor de horas de trabajo. En un primer momento, la progresión fue lenta hasta conocer el módulo ESP12 y aprender a programarlo correctamente.

Ha quedado patente que el ESP8266 te ofrece la posibilidad de un concepto interesante de la domótica con el que puedes controlar y estar informado a través de tu smartphone y tu ordenador mediante la plataforma ThingerIO.

6.2. DIFICULTADES GENERALES

Las dificultades encontradas durante la realización del proyecto no han sido pocas. Se van a enumerar a continuación las más relevantes:

- La utilización de un nuevo microcontrolador no utilizado en la carrera ha supuesto un gran número de horas para familiarizarse con él y la comunicación Wifi, ver las posibilidades que te ofrece, etc. En adición a esto apareció como dificultad añadida que, durante la fase de conocimiento del ESP8266, se estuvo trabajando con uno defectuoso, provocando una progresión muy lenta del proyecto.

- Durante el diseño electrónico del producto, una gran parte del mismo ha sido la elección de unos componentes adecuados, eligiendo en primera instancia de tipo THD por facilidad de soldadura y terminando con componentes SMD por la necesidad de ocupar menos espacio, aspecto muy importante del producto creado. A esta dificultad en la elección de materiales hay que añadir el hecho de tener como objetivo un producto final low cost, lo cual implicaba una búsqueda exhaustiva de componentes que cumplieran las características de eficiencia al menor costo posible. Esto ha supuesto un porcentaje importante del trabajo realizado, teniendo que navegar por páginas web de distribuidores tales como Farnell, Rs, Mouser, Digikey, etc.

6.3. MEJORAS PARA UN FUTURO

Una vez terminado el proyecto han surgido varios campos de mejora que harían el producto más competitivo en el mercado.

Las modificaciones posibles serían:

1. Al haber una comunicación entre la nube y el ESP12, una mejora sería el crear en vez de un diseño general de hardware a partir del cual cubrir todos los casos de uso, separar el producto en módulos, teniendo un control general sobre todos ellos desde la nube. Cada producto tendría su módulo Wifi, pero la ventaja reside en el hecho de que solamente llevaría el hardware necesario para su aplicación, reduciendo el espacio necesario, así como el coste de los sensores y parte de alimentación que no se vayan a utilizar.

   Por ejemplo, en el caso del control de las luces no es necesario ni los relés sin enclavamiento, ni el relé reed, ni la alimentación por baterías. Sin embargo, en el caso del telefonillo sería necesaria únicamente la presencia de un relé sin enclavamiento, siendo inútiles todos los sensores. En conclusión, si el cliente quiere controlar las luces, medir la temperatura y controlar su persiana debería comprar los 3 artículos correspondientes a un precio muy bajo, siendo que con el producto actual tendría que comprar tres productos iguales para hacer el control simultáneo de las tres aplicaciones, incrementando así costo y el espacio del producto.

2. La creación de una aplicación propia, sin necesidad de utilizar la creada ya por la plataforma thingerIO. Esto nos permitiría una personalización más acorde a nuestra aplicación y la capacidad de mejorar en los temas de mostrar gráficas o recibir notificaciones, algo que no está implementado en la aplicación actual de thinger.

   La razón por la que no ha sido creada es por la falta de tiempo y conocimiento en la creación de aplicaciones móviles, ya que una buena aplicación funcional y que cumpla su papel requiere de un tiempo que un estudiante con todo cuarto curso completo no posee.

7 BIBLIOGRAFÍA

7.1. LINKOGRAFÍA

SENSORES
- Código, (2017, Septiembre 20). Recuperado de http://raulpruebass.blogspot.com.es/

- Molina. J. Tecnología REED. *Monolitic Componentes Electrónicos.* https://es.scribd.com/doc/67762700/reed-switch-espanol

- Fotorresistencia: definición, características y tipos, (2017, enero 11). Recuperado de https://ingenieriaelectronica.org/fotorresistencia-definicion-caracteristicas-y-tipos/

ESP8266
- Díaz, M. A. (28 de enero de 2017). Modo deep-sleep en ESP8266. *Arduino a muete.* http://arduinoamuete.blogspot.com.es/2017/01/modo-deep-sleep-en-esp8266.html#more

THINGER.IO
- Díaz, M. A. (11 de diciembre de 2016). Introducción a Tiguer.io. *Arduino a muete.* http://arduinoamuete.blogspot.com.es/2016/12/thingerio.html

7.2 LIBROS

Casas, R., López J. M. (2017), Fundamentos para el diseño de sistemas electrónicos-

Casas, R., López J. M. (2017), Diseño de circuitos electrónicos

8 ANEXOS

8.1. ANEXO 1: ANÁLISIS ECONÓMICO DEL PRODUCTO

COMPONENTES Para 100 productos (Más barato por unidad, más caro el pedido total)		Distribuidor no oficial					Distribuidor oficial		
Descripción del material	**Unidades**	**Precio unitario(€)**	**Precio lotes**	**Precio envío**	**Precio total (lote+envío)**	**Link Fabricante**	**Precio**	**Referencia**	**Cod distribuidor**
ESP12	100	1,93 €	190 €	2,62 €	193,02 €	https://es.aliexpress.com/store/pr	642,00 €	83-16993	317060015
Fuente de alimentación (230ac-5VDC)	100	1,80 €	180 €	24,45 €	204,45 €	https://es.aliexpress.com/store/pr	469,00 €	IRM-01-5	709-IRM01-5
Fuente de alimentación (230ac-3.3VDC)	100	2,14 €	207 €	3,75 €	210,35 €	https://es.aliexpress.com/store/pr	470,00 €	IRM-01-3,3	709-IRM01-3,3
Mosfet	200	0,01 €	1,88 €	0,94 €	2,32 €	https://es.aliexpress.com/store/pr	12,40 €	2N7002 H6327	2N7002H6327
Pulsador	300	0,02 €	5,70 €	0,45 €	6,09 €	https://es.aliexpress.com/item/50	48,00 €	EAR99	8536509065
Sensor temperatura y humedad (DHT11)	100	0,60 €	59,4 €	1,17 €	60,58 €	https://es.aliexpress.com/store/pr	368,05 €	386	386545
Sensor luminosidad(LDR)	100	0,03	2,45 €	0 €	2,45 €	https://es.aliexpress.com/store/pr	356,00 €	VT90N1	2293503
Sensor presencia piroeléctrico	100	0,69 €	1,35 €	5,26 €	74,26 €	https://es.aliexpress.com/store/pr	333,33 €	9SIV0KY4462758	CEG007100
Rele reed	100	0,35 €	35,00 €	0,00 €	35,00 €	http://www.tme.eu/es/details/ksk	53,80 €	KSK-1A66/3-1015	876-KSK-1A66/3-1015
Relé enclavamiento(5V)	100	1,71 €	171,99 €	21,36 €	193,35 €	https://es.aliexpress.com/store/pr	416,00 €	RT314A05	680-3760
Relé enclavamiento(3V)	100	2,68 €	268,85 €	0,00 €	268.85€	https://www.avnet.com/shop/apa	268.85€	RT314A03	
Relé standard (5V)	200	0,32 €	63,60 €	1,37 €	64,97 €	https://es.aliexpress.com/store/pr	100,00 €	F36119-005A	36.11.9.005.4011
Relé standard (3V)	200	0,35 €	67,80 €	3,45 €	71,25 €	https://es.aliexpress.com/item/20	100,00 €	F36119-003A	36.11.9.003.4011
Diodos protección	400	0,01 €	2,36 €	1,02 €	3,38 €	https://es.aliexpress.com/store/pr	7,20 €	1N4148	512-1N4148
Pin headers (necesito 25 pines macho en total para un producto)(lotes 400)	7	0,79 €	5,53 €	0,73 €	6,26 €	https://es.aliexpress.com/store/pr	38,53 €	68000-406HLF	BRG68000-406HLF
Jumpers	800	0,01 €	7,12 €	2,26 €	9,38 €	https://es.aliexpress.com/store/pr	54,88 €	68786-302LF	1740371
Pin headers (necesito 3 pines hembra en total para un producto)(lotes 80)	4	0,66 €	2,64 €	0,45 €	3,09 €	https://es.aliexpress.com/store/pr	17,74 €	310-87-103-41-001101	1212-1090-ND
Resistencia 10K	600	0,01 €	6,51 €	1 €	7,51 €	https://es.aliexpress.com/store/pr	6,42 €	CRCW080510K0JNEA	541-10KACT-ND
Resistencia 2K7	100	0,01 €	0,83 €	0,50 €	1,33 €	https://es.aliexpress.com/store/pr	1,96 €	CRCW08052K70JNEA	1514827
Resistencia 1 K	100	0,01 €	1,71 €	1,48 €	3,19 €	https://es.aliexpress.com/store/pr	1,64 €	CRCW08051K00FKEA	1469847
Resistencia 330	100	0,01 €	0,88 €	1,77 €	2,65 €	https://es.aliexpress.com/item/10	1,60 €	CRCW0805330RFKEA	1469918RL
Condensador cerámico 10nF	400	0,01 €	2,05 €	1,48 €	3,53 €	https://es.aliexpress.com/store/pr	4,64 €	CC0805KRX7R9BB103	3019937
Interruptor	100	0,03 €	2,69 €	1,39 €	4,08 €	https://es.aliexpress.com/item/10	35,70 €	EG1218	E-Switch
Ficha 4 pines (para relés)	100	0,10 €	9,32 €	0,91 €	10,23 €	https://es.aliexpress.com/store/pr	18,30 €	CTB1202/4BK	1716995
Ficha 2 pines (finales carrera, rele encl, fuente AC, fuente DC)	500	0,06 €	26,16 €	0,99 €	27,15 €	https://es.aliexpress.com/store/pr	41,20 €	CTB1202/2BK	1716993
Cargador batería MCP73831	100	0,01 €	6,10 €	1,48 €	7,58 €	https://es.aliexpress.com/store/pr	39,60 €	MCP73831T-2ACI/OT	579-MCP73831T-2ACIOT
Condensador 4,7uF (ceramico)(cargador)	200	0,01 €	1,86 €	0,68 €	2,54 €	https://es.aliexpress.com/store/pr	1,60 €	0603F475Z100CT	78Y5262
Resistencia 2k (cargador)	100	0,01 €	0,54 €	0,50 €	1,04 €	https://es.aliexpress.com/store/pr	1,22 €	CRCW06032K00FKEA	1469764
JST ficha 2 pines para batería	100	0,05 €	3,81 €	0,00 €	3,81 €	https://es.aliexpress.com/store/pr	7,78 €	B2B-EH-A(LF)(SN)	2399159
Regulador 5 a 3.3V (eficiencia máxima)	100	4,12 €	412 €	0,00 €	412 €	http://www.mouser.es/ProductDe	412 €	P7805-Q24-S3-S	490-P7805-Q24-S3-S
Condensador cerámico 10uF	200	0,01 €	2,18 €	0,00 €	2,18 €	http://www.mouser.es/ProductDe	2,18 €	VJ0805G106KXQTW1BC	77-VJ0805G106KXQTBC
Regulador 5 a 3.3V (lineal lowcost)	100	0,46 €	45,8	0,00 €	45,80 €	http://www.mouser.es/ProductDe	45,80 €	UA78M33CKCSE3	595-UA78M33CKCSE3
Condensador cerámico 0,33uF	100	0,01 €	0,40 €	0,00 €	0,40 €	http://www.newark.com/multicon	0,40 €	MC0603X334K100CT	06R5245
Condensador cerámico 0,1uF	100	0,01 €	0,30 €	0,00 €	0,30 €	http://www.newark.com/multicon	0,30 €	MC0603B104M160CT	06R4929
Conector microUSB	100	0,47 €	4,70 €	1,40 €	6,10 €	https://es.aliexpress.com/store/pr	36,40 €	10104110-0001LF	2293753
Distribuidor de alimentación TSP22860	100	0,35 €	35,30 €	0,00 €	35,30 €	http://www.mouser.es/ProductDe	35,30 €	TPS22860DBVR	595-TPS22860DBVR
Condensador cerámico 1uF	100	0,02 €	1,74 €	0 €	1,74 €	http://es.farnell.com/walsin/0603f	1,74 €	0603F105Z100CT	2524858

	low cost				distrib oficiales			
	low-cost (con fuente 5V, cargador de bateria y regulador lineal)	**max eficiencia (con cargador de batería, fuente 5V, regulador eficiente)**	**low-cost (sin cargador, con fuente 3.3 V, sin regulador)**	**low-cost (con cargador, sin fuente, con regulador)**	**low-cost (con fuente 5V, cargador de bateria y regulador lineal)**	**max eficiencia (con cargador de batería, fuente 5V, regulador eficiente)**	**low-cost (sin cargador, con fuente 3.3 V, sin regulador)**	**low-cost (con cargador, sin fuente, con regulador)**
Precio del pedido:	**1.019,07 €**	**1.390,56 €**	**780,24 €**	**818,43 €**	**3.190,75 €**	**3.566,21 €**	**2.686,83 €**	**2.729,53 €**
Precio de cada producto:	**10,1907**	**13,9056**	**7,8024**	**8,1843**	**31,9075**	**35,6621**	**26,8683**	**27,2953**

COMPONENTES Para 1000 productos (Más barato por unidad, más caro el pedido total)		Distribuidor no oficial					Distribuidor oficial		
Descripción del material	**Unidades**	**Precio unitario(€)**	**Precio lotes**	**Precio envío**	**Precio total(lote+envío)**	**Link Fabricante**	**Precio**	**Referencia**	**Cod distribuidor**
ESP12	1000	2 €	1.904 €	2,62 €	1.906,62 €	https://es.aliexpress.com/store/pr	6.420,00 €	83-16993	317060015
Fuente de alimentación (230ac-5VDC)	1000	1,80 €	1.800 €	137,76 €	1.937,76 €	https://es.aliexpress.com/store/pr	4.350,00 €	IRM-01-5	709-IRM01-5
Fuente de alimentación (230ac-3.3VDC)	1000	2,14 €	2.066 €	166,74 €	2.232,74 €	https://es.aliexpress.com/store/pr	4.200,00 €	IRM-01-3.3	709-IRM01-3.3
Mosfet	2000	0,01 €	9,40 €	0,94 €	10,34 €	https://es.aliexpress.com/store/pr	84,00 €	2N7002 H6327	2N7002H6327
Pulsador	3000	0,02 €	56,40 €	1,59 €	57,99 €	https://es.aliexpress.com/item/50	396,00 €	EAR99	8536509065
Sensor temperatura y humedad (DHT11)	1000	0,60 €	594 €	54,88 €	648,88 €	https://es.aliexpress.com/store/pr	3.680,00 €	386	
Sensor luminosidad(LDR)	1000	0,03	19,39 €	0 €	19,39 €	https://es.aliexpress.com/store/pr	3.360,00 €	VT90N1	2293503
Sensor presencia piroeléctrico	1000	0,69 €	690,00 €	75,09 €	765,09 €	https://es.aliexpress.com/store/pr	3.333,30 €	9SIV0KY4462758	CEG007100
Rele reed	1000	0,35 €	350,00 €	0,00 €	350,00 €	http://www.tme.eu/es/details/ksk	379,00 €	KSK-1A66/3-1015	876-KSK-1A66/3-1015
Relé enclavamiento (5V)	1000	1,71 €	1.719 €	106,59 €	1.826,49 €	https://es.aliexpress.com/store/pr	4.160 €	RT314A05	680-3760
Relé enclavamiento(3V)	1000	2,68 €	2.688,50 €	0,00 €	2.688,50 €	https://www.avnet.com/shop/apa	2.688 €	RT314A03	
Relé standard (5V)	2000	0,33 €	638,60 €	95,21 €	733,81 €	https://es.aliexpress.com/store/pr	940,00 €	F36119-005A	36.11.9.005.4011
Relé standard (3V)	2000	0,35 €	678,00 €	34,48 €	712,48 €	https://es.aliexpress.com/item/20	940,00 €	F36119-003A	36.11.9.003.4011
Diodos protección	4000	0,01 €	14,96 €	6,25 €	21,21 €	https://es.aliexpress.com/store/pr	36,00 €	1N4148	512-1N4148
Pin headers (necesito 25 pines macho en total para un producto)(lotes 400)	63	0,79 €	49,77 €	2,48 €	52,25 €	https://es.aliexpress.com/store/pr	385,08 €	68000-406HLF	BRG68000-406HLF
Jumpers	8000	0,01 €	71,20 €	11,26 €	82,46 €	https://es.aliexpress.com/store/pr	526,40 €	68786-302LF	1740371
Pin headers (necesito 3 pines hembra en total para un producto)(lotes)	38	0,66 €	25,09 €	1,89 €	26,98 €	https://es.aliexpress.com/store/pr	126,71 €	310-87-103-41-001101	1212-1090-ND
Resistencia 10K	6000	0,004 €	24,96 €	0 €	24,96 €	http://www.digikey.es/product-de	24,96 €	CRCW080510K0JNEA	541-10KACT-ND
Resistencia 2K7	1000	0,01 €	8,39 €	0,59 €	8,89 €	https://es.aliexpress.com/store/pr	14,00 €	CRCW08052K70JNEA	1514827
Resistencia 1 K	1000	0,01 €	6,54 €	0 €	6,54 €	https://es.aliexpress.com/store/pr	8,30 €	CRCW08051K00FKEA	1469847
Resistencia 330	1000	0,01 €	1,18 €	1,82 €	3,00 €	https://es.aliexpress.com/store/pr	8,30 €	CRCW0805330RFKEA	1469918RL
Condensador cerámico 10nF	4000	0,01 €	13,22 €	0 €	13,22 €	https://es.aliexpress.com/store/pr	28,80 €	CC0805KRX7R9BB103	3019937
Interruptor	1000	0,03 €	26,90 €	5,36 €	32,26 €	https://es.aliexpress.com/item/10	301 €.	EG1218	E-Switch
Ficha 4 pines (para relés)	1000	0,10 €	93,20 €	10,97 €	104,17 €	https://es.aliexpress.com/store/pr	183,00 €	CTB1202/4BK	1716995
Ficha 2 pines (finales carrera, rele encl, fuente AC, fuente DC)	5000	0,06 €	235,50 €	86,37 €	321,87 €	https://es.aliexpress.com/store/pr	412,00 €	CTB1202/2BK	1716993
Cargador batería MCP73831	1000	0,10 €	91,60 €	0 €	91,60 €	https://es.aliexpress.com/store/pr	396 €	MCP73831T-2ACI/OT	579-MCP73831T-2ACIOT
Condensador 4,7uF (ceramico)(cargador)	2000	0,01 €	12,20 €	1,48 €	13,68 €	https://es.aliexpress.com/store/pr	10,00 €	0603F475Z100CT	78Y5262
Resistencia 2k (cargador)	1000	0,01 €	1,32 €	1,76 €	3,08 €	https://es.aliexpress.com/store/pr	6,80 €	CRCW06032K00FKEA	1469764
JST ficha 2 pines para batería	1000	0,05 €	38,09 €	13,25 €	51,34 €	https://es.aliexpress.com/store/pr	56,40 €	B2B-EH-A(LF)(SN)	2399159
Regulador 5 a 3.3V (eficiencia máxima)	1000	3,46 €	3.460 €	0 €	3.460 €	http://www.mouser.es/ProductDe	3.460 €	P7805-Q24-S3-S	490-P7805-Q24-S3-S
Condensador cerámico 10uF	2000	0,01 €	12,26 €	0 €	12,26 €	http://www.mouser.es/ProductDe	12,26 €	VJ0805G106KXQTW1BC	77-VJ0805G106KXQTBC
Regulador 5 a 3.3V (lineal lowcost)	1000	0,32 €	320 €	0 €	320,00 €	https://buy.rocelec.com/parts/TIS/	367,00 €	UA78M33CKCSE3	595-UA78M33CKCSE3
Condensador cerámico 0,33uF	1000	0,01 €	4,00 €	0 €	4,00 €	http://www.newark.com/multicon	4,00 €	MC0603X334K100CT	06R5245
Condensador cerámico 0,1uF	1000	0,01 €	3,00 €	0 €	3,00 €	http://www.newark.com/multicon	3,00 €	MC0603B104M160CT	06R4929
Conector microUSB	1000	0,47 €	47,00 €	1,40 €	48,40 €	https://es.aliexpress.com/store/pr	351,00 €	10104110-0001LF	2293753
Distribuidor de alimentación TSP22860	1000	0,35 €	35,30 €	0,00 €	35,30 €	http://www.mouser.es/ProductDe	282,00 €	TPS22860DBVR	595-TPS22860DBVR
Condensador cerámico 1uF	1000	0,01 €	9,00 €	0,00 €	9,00 €	http://es.farnell.com/walsin/0603f	9,00 €	0603F105Z100CT	2524858

	low cost				distrib oficiales			
	low-cost (con fuente 5V, cargador de bateria y regulador lineal)	**max eficiencia (con cargador de batería, fuente 5V, regulador eficiente)**	**low-cost (sin cargador, con fuente 3.3 V, sin regulador)**	**low-cost (con cargador, sin fuente, con regulador)**	**low-cost (con fuente 5V, cargador de bateria y regulador lineal)**	**max eficiencia (con cargador de batería, fuente 5V, regulador eficiente)**	**low-cost (sin cargador, con fuente 3.3 V, sin regulador)**	**low-cost (con cargador, sin fuente, con regulador)**
Precio del pedido:	**9.482,24 €**	**12.678,84 €**	**10.182,54 €**	**7.595,82 €**	**30.284,65 €**	**33.439,31 €**	**27.876,33 €**	**25.991,05 €**
Precio de cada producto:	**9,48 €**	**12,68 €**	**10,18 €**	**7,60 €**	**30,28 €**	**33,44 €**	**27,88 €**	**25,99 €**

8.2. ANEXO 2: CÓDIGO DE LA APLICACIÓN

```
1
2   ////////////////////////////////
3   //INCLUDES
4
5   #include <ESP8266WiFi.h>
6   #include <ThingerESP8266.h>
7   #include <ThingerSmartConfig.h>
8
9   #include <Adafruit_Sensor.h>//van en pack
10  #include <DHT.h>//van en pack
11
12  #include <TimeLib.h>
13  #include <Time.h>
14  #include <TimeAlarms.h>
15  #include <WiFiUdp.h>
16
17  //Includes config y memoria
18  #include <SoftwareSerial.h>
19  #include <command.h>
20  #include <EEPROM.h>
21  ////////////////////////////////////////////////////////////////////////////////
    ////////////////////////////////
22  ////////////////////////////////////////////////////////////////////////////////
    ////////////////////////////////
23  //DEFINE
24
25  //DEFINE MODOS
26  #define ControlLuces 1
27  #define ControlTemperatura 2
28  #define VacacionesPersiana 3
29  #define VacacionesLuces 4
30  #define ControlPersiana 5
31  #define MonitorizacionVariables 6
32  #define ControlPresencia 7
33  #define Telefonillo 8
34  #define Configuracion 9
35
36  //DEFINE MAQUINA ESTADOS CONFIG Y MEMORIA
37  #define Credenciales 0
38  #define EleccionParametros1 1//Control Luces
39  #define EleccionParametros2 2//Control Temperatura
40  #define EleccionParametros3 3//Vacaciones Persiana y Vacaciones Luces
41  #define EleccionParametros4 4//Control Persiana
42  #define EleccionParametros5 5//Monitorizacion Variables
43
44  //DEFINE PINES
45  #define RELE1 0
46  #define RELE2 15
47  #define SWITCH2 14
48  #define SWITCH3 12
49  #define DHTPIN 4
50  #define REED 5
51  #define FINALCARRERA1 5
52  #define FINALCARRERA2 5
53  #define PIR 13
54  #define LDR A0
55  #define LED 2
56
57
58  //DEFINE THINGERIO
59  #define USERNAME "Angel"
60  #define DEVICE_ID "Prueba"
61  #define DEVICE_CREDENTIAL "Prueba"
62
63  //DEFINE ESP12
64  #define SLEEP_TIME 10
65
66  //DEFINE CLOCK
67  #define TIME_MSG_LEN  11   // time sync to PC is HEADER followed by unix time_t as
    ten ascii digits
```

```
#define TIME_HEADER  'T'   // Header tag for serial time sync message
#define TIME_REQUEST  7    // ASCII bell character requests a time sync message

//DEFINE DHT
#define DHTTYPE DHT11
////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////
//DECLARACION OBJETOS
    // NTP Servers:
IPAddress timeServer(132, 163, 4, 101); // time-a.timefreq.bldrdoc.gov
// IPAddress timeServer(132, 163, 4, 102); // time-b.timefreq.bldrdoc.gov
// IPAddress timeServer(132, 163, 4, 103); // time-c.timefreq.bldrdoc.gov
WiFiUDP Udp;

//SERIAL
SoftwareSerial swSer(3, 1, false, 256);


ThingerESP8266 thing(USERNAME, DEVICE_ID, DEVICE_CREDENTIAL);

DHT dht(DHTPIN, DHTTYPE);
////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////
//DECLARACION VARIABLES

int modo;

    //CONFIG Y MAQUINA DE ESTADOS
int modo_elegido=0;
int estado=0;

char caracter=0;
char basura=0;
String string_formado;

    //Credenciales y eleccion de modo
boolean password_leido=false;
boolean ssid_leido=false;
boolean modo_elegido_leido=false;

char ssid[20];
char password[20];


    //EleccionParametros1
boolean modo_automatico;
boolean modo_automatico_elegido=false;//uso esta en la logica de la comunicacion con el usuario

int respuesta_int;//Con este almaceno en memoria


    //EleccionParametros2
boolean modo_control_calefaccion_leido=false;
boolean leer_temperatura_limite=false;
boolean leer_tiempo=false;
boolean hora_on_leida=false;
boolean hora_off_leida=false;
boolean minutos_on_leidos=false;
boolean minutos_off_leidos=false;

int modo_control;
int hora_on;
int hora_off;
int minutos_on;
```

```
int minutos_off;
int temperatura_limite;

    //EleccionParametros4
boolean modo_control_persiana_leido;

    //EleccionParametros3 y 5
boolean modo_aleatorio=false;
boolean modo_aleatorio_elegido=false;
char modo_aleatorio_char;
String modo_aleatorio_string;

    //EleccionParametros6
boolean variables_monitorizadas_leidas=false;
boolean tiempo_entre_envio_datos_leido=false;

int variables_monitorizadas;
int tiempo_entre_envio_datos;

    //Interrupciones SW2 para config
volatile unsigned long t_on_pulsado_SW2_actual;
volatile unsigned long t_on_pulsado_SW2_anterior=0;
boolean SW2_on=false;

    //VARIABLES MODO CONTROLLUCES
unsigned long t_actual;
boolean SW3_es_salida;

    //VARIABLES MODO CONTROLTEMPERATURA
boolean calefaccion_encendida=false;

    //VARIABLES MODO VACACIONESPERSIANA
int hora_subida;
int minutos_subida;
int hora_bajada;
int minutos_bajada;

boolean ciclo_terminado=false;
boolean ya_subida=false;
boolean ya_bajada=false;
boolean no_dormir;
int random_hora_subida;
int random_minutos_subida;

    //VARIABLES MODO VACACIONESLUCES
boolean luces_encendidas=false;
int random_hora_on;
int random_hora_off;
int random_minutos_on;

    //VARIABLES MODO MONITORIZACION VARIABLES
boolean temperatura=true;
boolean presencia_magnetico=true;
boolean presencia_pir=true;
//int variables_monitorizadas=1;//1=temp y hum.... 2=temp,hum y pir... 3=temp,hum y reed...   Introducir este dato por el usuario
//int tiempo_entre_envio_datos=60;//esto lo debe fijar el usuario
bool presencia_detectada = false;


    //VARIABLES MODO TELEFONILLO
boolean estado_RELE1 = false;
boolean primera_vez = true;
float tiempo;
time_t hora_actual;


//VARIABLES RELOJ
const int timeZone = 2;     // Central European Time... poner a 1 si quiero retrasar una hora
//const int timeZone = -5;  // Eastern Standard Time (USA)
```

```
//const int timeZone = -4;  // Eastern Daylight Time (USA)
//const int timeZone = -8;  // Pacific Standard Time (USA)
//const int timeZone = -7;  // Pacific Daylight Time (USA)
unsigned int localPort = 8888;  // local port to listen for UDP packets
time_t prevDisplay = 0; // when the digital clock was displayed


//VARIABLES MILLIS
unsigned long previousMillis = 0;
unsigned long currentMillis = 0;
long interval = 70;//EN SEGUNDOS (HABRÁ QUE MULTIPLICAR POR 1000 PORQUE MILLIS ME LO DA EN MS)


////////////////////////////////////////////////////////////////////////////////////////////////////
//FUNCIONES UTILIZADAS

/*-------- NTP code ----------*/

const int NTP_PACKET_SIZE = 48; // NTP time is in the first 48 bytes of message
byte packetBuffer[NTP_PACKET_SIZE]; //buffer to hold incoming & outgoing packets

void sendNTPpacket(IPAddress &address)
{
    // set all bytes in the buffer to 0
    memset(packetBuffer, 0, NTP_PACKET_SIZE);
    // Initialize values needed to form NTP request
    // (see URL above for details on the packets)
    packetBuffer[0] = 0b11100011;   // LI, Version, Mode
    packetBuffer[1] = 0;     // Stratum, or type of clock
    packetBuffer[2] = 6;     // Polling Interval
    packetBuffer[3] = 0xEC;  // Peer Clock Precision
    // 8 bytes of zero for Root Delay & Root Dispersion
    packetBuffer[12]  = 49;
    packetBuffer[13]  = 0x4E;
    packetBuffer[14]  = 49;
    packetBuffer[15]  = 52;
    // all NTP fields have been given values, now
    // you can send a packet requesting a timestamp:
    Udp.beginPacket(address, 123); //NTP requests are to port 123
    Udp.write(packetBuffer, NTP_PACKET_SIZE);
    Udp.endPacket();
}

time_t getNtpTime()
{
    while (Udp.parsePacket() > 0) ; // discard any previously received packets
    swSer.println("Transmit NTP Request");
    sendNTPpacket(timeServer);
    uint32_t beginWait = millis();
    while (millis() - beginWait < 1500) {
        int size = Udp.parsePacket();
        if (size >= NTP_PACKET_SIZE) {
            swSer.println("Receive NTP Response");
            Udp.read(packetBuffer, NTP_PACKET_SIZE);  // read packet into the buffer
            unsigned long secsSince1900;
            // convert four bytes starting at location 40 to a long integer
            secsSince1900 =  (unsigned long)packetBuffer[40] << 24;
            secsSince1900 |= (unsigned long)packetBuffer[41] << 16;
            secsSince1900 |= (unsigned long)packetBuffer[42] << 8;
            secsSince1900 |= (unsigned long)packetBuffer[43];
            return secsSince1900 - 2208988800UL + timeZone * SECS_PER_HOUR;
        }
    }
    swSer.println("No NTP Response :-(");
    return 0; // return 0 if unable to get the time
}
```

```
/*------------------------------*/

void encender_rele_enclavamiento(){
  swSer.println("Encendiendo rele de enclavamiento");
  digitalWrite(RELE1, HIGH);
  delay(2000);
  digitalWrite(RELE1, LOW);
  swSer.println("Rele de enclavamiento encendido");
  luces_encendidas=true;
  calefaccion_encendida=true;
}

void apagar_rele_enclavamiento(){
  swSer.println("Apagando rele de enclavamiento");
  digitalWrite(RELE2, HIGH);
  delay(2000);
  digitalWrite(RELE2, LOW);
  swSer.println("Rele de enclavamiento apagado");
  luces_encendidas=false;
  calefaccion_encendida=false;
}
void alimento_sensor(){//declaro el SW3 como salida para alimentar los sensores
  pinMode(SWITCH3, OUTPUT);
  digitalWrite(SWITCH3, HIGH);
}

void quito_alimentacion_sensor(){
  digitalWrite(SWITCH3, HIGH);
  pinMode(SWITCH3, INPUT);
}

void interr_SWITCH2(){
  SW2_on=!SW2_on;
  no_dormir=true;
  swSer.println(SW2_on);
  if(SW2_on){
      t_on_pulsado_SW2_anterior=millis();
  }
  else{
      t_on_pulsado_SW2_actual=millis();
      swSer.println("SW2_on es false y el tiempo actual es: ");
      swSer.println(t_on_pulsado_SW2_actual);
      unsigned long resta=t_on_pulsado_SW2_actual-t_on_pulsado_SW2_anterior;
      swSer.println(resta);
      if((t_on_pulsado_SW2_actual-t_on_pulsado_SW2_anterior)>(5000)){
          modo=9;
          modo_elegido=9;
          primera_vez=true;
          swSer.println("Cambiando a modo configuracion...");
      }
      else{
          swSer.println("PULSADO DEL SW2 NORMAL PARA ACTIVAR RELES ");
          if(modo== VacacionesPersiana || modo==ControlPersiana){
              digitalWrite(RELE2, LOW);//Para en el caso de que le den a los dos SW
              a la vez
              digitalWrite(RELE1, HIGH);//se pondrá en LOW cuando termine de subir
              la persiana con la interr del final de carrera
              no_dormir=true;
              ya_subida=true;
              ya_bajada=false;
          }
          else if(modo== ControlLuces || modo==ControlTemperatura ||
          modo==VacacionesLuces){
              digitalWrite(RELE2, LOW);//Para en el caso de que le den a los dos SW
              a la vez
              encender_rele_enclavamiento();
              no_dormir=false;
          }
      }
  }
}
```

```
338
339
340  void interr_SWITCH3(){
341    if(SW3_es_salida==false){
342        if(modo_elegido== VacacionesPersiana || modo_elegido== ControlPersiana){
343            digitalWrite(RELE1, LOW);//Para en el caso de que le den a los dos SW a la
               vez
344            digitalWrite(RELE2, HIGH);//se pondrá en LOW cuando termine de bajar la
               persiana con la interr del final de carrera
345            no_dormir=true;
346            ya_bajada=true;
347            ya_subida=false;
348        }
349        else if(modo_elegido== ControlLuces || modo_elegido==ControlTemperatura ||
           modo_elegido==VacacionesLuces){
350            digitalWrite(RELE1, LOW);//Para en el caso de que le den a los dos SW a la
               vez
351            apagar_rele_enclavamiento();
352            no_dormir=false;
353        }
354    }
355  }
356
357  boolean luminosidad_baja(){
358    boolean poca_luz;
359    SW3_es_salida=true;
360    alimento_sensor();//alimento sensor LDR activando la patilla correspondiente (VS)
361    delay(100); //ajustado para darle tiempo a que se encienda y se ponga operativo
362    int cantidad_luz= analogRead(LDR);//coger valor pin ADC
363    if (cantidad_luz<500){//ajustar el valor segun el sensor (numero entre 0 y 1024)
364      poca_luz=true;
365      swSer.println("Baja luminosidad");
366    }
367    else
368      poca_luz=false;
369    quito_alimentacion_sensor();
370    SW3_es_salida=false;
371    return poca_luz;
372  }
373
374  void control_luminosidad(){
375    if(luminosidad_baja()){
376      if(presencia_detectada && !luces_encendidas){
377        swSer.println("presencia detectada con luminosidad baja");
378        encender_rele_enclavamiento();
379        presencia_detectada=false;
380        t_actual=millis();//cojo la hora a la que enciendo la luz para mantenerla
           encendida durante un mínimo de tiempo
381      }
382    }
383    if(luces_encendidas){
384      if(!presencia_detectada && ((millis()-t_actual)>(15*1000))){//le doy un tiempo
         de 2 minutos encendida (120 segs)o 1 (60 segs)
385        apagar_rele_enclavamiento();
386        t_actual=millis();
387      }
388    }
389
390  }
391
392  void control_grados(){
393    SW3_es_salida=true;
394    alimento_sensor();
395    delay(100);
396    float temperatura_actual = dht.readTemperature();
397    swSer.println(temperatura_actual);
398    delay(2000);//Pongo este delay porque al sensor le cuesta unos 2 segundos mandar
       la información
399    quito_alimentacion_sensor();
400    SW3_es_salida=false;
401    if(temperatura_actual<(float)temperatura_limite && !calefaccion_encendida)
```

```
402       encender_rele_enclavamiento();
403     if(temperatura_actual>(float)temperatura_limite+3.0 && calefaccion_encendida)//Se
        da un margen de 3 grados antes de desactivar la calefacción
404       apagar_rele_enclavamiento();
405
406   }
407
408   void control_tiempo_calefaccion(){
409       int margen=5;//ajustar esto según el sleep time para que al menos una vez entre
          sí o sí en los intervalos y se encienda/apague
410       time_t t=now();
411       if(hour(t)==hora_on && !calefaccion_encendida){
412           if(minute(t)>=(minutos_on-margen) && minute(t)<=(minutos_on+margen)){
413               encender_rele_enclavamiento();
414           }
415       }
416       if(hour(t)==hora_off && calefaccion_encendida){
417           if(minute(t)>=(minutos_off - margen) && minute(t)<=(minutos_off + margen)){
418               apagar_rele_enclavamiento();
419           }
420
421       }
422
423   }
424
425
426   void interr_limite_alcanzado_persiana(){
427       no_dormir=false;
428       ciclo_terminado=true;
429       digitalWrite(RELE1, LOW);
430       digitalWrite(RELE2, LOW);
431       swSer.println("persiana tocando el final de carrera");
432
433   }
434
435
436   void control_aleatorio_persiana(){
437       randomSeed(millis()); //necesario poner esto para el correcto funcionamiento del
          random
438       int margen=5;//utilizo margen si duermo el micro
439       if(primera_vez && ciclo_terminado){
440           int random_hora_subida=random(8,11); //escoge aleatoriamente un numero entre
              8 y 11
441           int random_minutos_subida=random(0,59); //escoge aleatoriamente un numero
              entre 0 y 59
442           int hora_bajada=random_hora_subida+14;
443           int minutos_bajada=random_minutos_subida;
444           primera_vez=false;
445
446       }
447       time_t t=now();
448       if(hour(t)==random_hora_subida && !ya_subida){
449           if(minute(t)>=random_minutos_subida-margen &&
              minute(t)<=random_minutos_subida+margen){
450               digitalWrite(RELE1, HIGH);
451               swSer.println("subiendo persiana...");
452               ya_bajada=false;
453               ya_subida=true;
454           }
455
456       }
457
458       if(hour(t)==hora_bajada && !ya_bajada){
459           if(minute(t)>=minutos_bajada - margen && minute(t)<=minutos_bajada + margen){
460               digitalWrite(RELE2, HIGH);
461               swSer.println("bajando persiana...");
462               ya_bajada=true;
463               ya_subida=false;
464               ciclo_terminado=false;
465               primera_vez=true;
466           }
```

```
    }


}


void control_tiempo_persiana(){
    int margen=5;//utilizo margen si duermo el micro

    time_t t=now();
    if(hour(t)==hora_subida && !ya_subida){
        if(minute(t)>=minutos_subida-margen && minute(t)<=minutos_subida+margen){
            digitalWrite(RELE1, HIGH);
            swSer.println("subiendo persiana...");
            ya_bajada=false;//actualizo el valor de estas variables para que no entren continuamente en los if
            ya_subida=true;
            no_dormir=true;
        }
    }
    else if(hour(t)==hora_bajada && !ya_bajada){
        if(minute(t)>=minutos_bajada - margen && minute(t)<=minutos_bajada + margen){
            digitalWrite(RELE2, HIGH);
            swSer.println("bajando persiana...");
            ya_bajada=true;
            ya_subida=false;
            no_dormir=true;
        }
    }
    else
        no_dormir=true;//esto esta puesto para el caso en que quiera dormir el micro


}

void control_aleatorio_luces(){
    randomSeed(millis());
    int margen=5;
    if(primera_vez){
         random_hora_on=random(20,24);
         random_minutos_on=random(0,59);
         swSer.print("Hora elegida aleatoriamente encendido: ");
         swSer.print(random_hora_on);
         swSer.print(":");
         swSer.println(random_minutos_on);
         if(random_hora_on>22){
            hora_off=0+(random_hora_on-22);
         }
         else{
            hora_off=random_hora_on+2;
         }
         minutos_off=random_minutos_on;
         swSer.print("Hora apagado: ");
         swSer.print(hora_off);
         swSer.print(":");
         swSer.println(minutos_off);
         primera_vez=false;
    }
    time_t t=now();
    if(hour(t)==random_hora_on && !luces_encendidas){
        if(minute(t)>=random_minutos_on-margen && minute(t)<=minutos_on+margen){
          encender_rele_enclavamiento();
        }
    }
    if(hour(t)==hora_off && luces_encendidas){
        if(minute(t)>=minutos_off + margen && minute(t)<=minutos_off + margen){
            apagar_rele_enclavamiento();
            primera_vez=true;
        }
    }
```

```
536
537   }
538
539
540   void control_tiempo_luces(){
541       int margen=5;//ajustar esto según el sleep time para que al menos una vez entre
          sí o sí en los intervalos y se encienda/apague
542       time_t t=now();
543       if(hour(t)==hora_on && !luces_encendidas){
544           if(minute(t)>=minutos_on-margen && minute(t)<=minutos_on+margen){
545               encender_rele_enclavamiento();
546           }
547       }
548       if(hour(t)==hora_off && luces_encendidas){
549           if(minute(t)>=minutos_off - margen && minute(t)<=minutos_off + margen){
550               apagar_rele_enclavamiento();
551           }
552
553       }
554
555   }
556
557   void monitorizar_temperatura(){
558       if(primera_vez){
559           primera_vez=false;
560           previousMillis = millis();
561       }
562
563   }
564
565   //Funciones que se ejecutan al saltar las interrupciones
566
567   void interr_presencia_detectada_REED(){
568       presencia_detectada=true;
569   }
570
571   void interr_presencia_detectada_PIR() {
572       presencia_detectada = true;
573   }
574
575   void monitorizar_apertura_puerta(){
576     if(presencia_detectada){
577       thing.call_endpoint("PIR_DETECTION");
578       presencia_detectada = false;
579       swSer.println("Puerta abierta");
580       }
581
582   }
583
584
585   void monitorizar_presencia_movimiento(){
586     if(presencia_detectada){
587       thing.call_endpoint("PIR_DETECTION");
588       presencia_detectada = false;
589       swSer.println("Presencia detectada");
590       }
591
592   }
593
594
595   void control_telefonillo(){
596     if(estado_RELE1){
597       if(primera_vez){
598         digitalWrite(RELE1, HIGH);
599         tiempo=millis();
600         swSer.print("Tiempo: ");
601         swSer.println(tiempo);
602         hora_actual=now();
603         swSer.println(hora_actual);
604         primera_vez=0;
605       }
```

```
606       if(now()-hora_actual >= 5){
607         swSer.println(now());
608         digitalWrite(RELE1, LOW);
609       }
610     }
611   }
612
613   void printDigits(int digits){
614     // utility for digital clock display: prints preceding colon and leading 0
615     swSer.print(":");
616     if(digits < 10)
617       swSer.print('0');
618     swSer.print(digits);
619   }
620
621   void digitalClockDisplay(){
622     // digital clock display of the time
623     swSer.print(hour());
624     printDigits(minute());
625     printDigits(second());
626     swSer.print(" ");
627     swSer.print(day());
628     swSer.print("/");
629     swSer.print(month());
630     swSer.print("/");
631     swSer.print(year());
632     swSer.println();
633   }
634
635   //Función para la lectura del serial
636
637   void lecturaSerial(){
638       if(swSer.available() > 0){
639             delay(100);//si no pongo este delay el primer caracter no me lo coge
640             while (swSer.available() > 0) {
641                 if(swSer.available()>2){
642                     caracter=swSer.read();
643                     delay(10);
644                     string_formado+=caracter;
645                    // swSer.write(prueba);//write es para escribir números (en forma
                       de caracter por pantalla)
646
647                 }
648                 else{
649                     basura=swSer.read();
650                     delay(50);
651                     basura=0;
652
653                 }
654             }
655       }
656   }
657
658   //Funciones de lectura y escritura sobre la memoria EEPROM
659
660   /** Store WLAN credentials to EEPROM */
661   void saveCredentials() {
662       EEPROM.begin(512);
663       EEPROM.put(0, ssid);
664       EEPROM.put(0+sizeof(ssid), password);
665       EEPROM.commit();
666       EEPROM.end();
667       swSer.println("Credenciales guardadas");
668   }
669
670   /** Load WLAN credentials from EEPROM */
671   void loadCredentials() {
672       EEPROM.begin(512);
673       EEPROM.get(0, ssid);
674       EEPROM.get(0+sizeof(ssid), password);
675       EEPROM.end();
```

```
676         swSer.println("Recovered credentials:");
677         swSer.println(ssid);
678         swSer.println(password);
679         swSer.println(strlen(password)>0?"********":"<no password>");
680     }
681
682     void saveModoElegido(){
683         EEPROM.begin(512);
684         delay(50);
685         EEPROM.put(0+sizeof(ssid)+sizeof(password), modo_elegido);
686         EEPROM.commit();
687         EEPROM.end();
688         swSer.println("Modo de funcionamiento guardado");
689     }
690
691     void loadModoElegido(){
692         EEPROM.begin(512);
693         EEPROM.get(0+sizeof(ssid)+sizeof(password), modo_elegido);
694         EEPROM.end();
695         swSer.println("Recovered modo_elegido:");
696         swSer.println(modo_elegido);
697         swSer.println((modo_elegido)>0?"********":"<no temperatura_limite>");
698     }
699
700     void saveModoAutomatico(){
701         EEPROM.begin(512);
702         delay(50);
703         EEPROM.put(0+sizeof(ssid)+sizeof(password)+sizeof(modo_elegido),modo_automatico);
704         EEPROM.commit();
705         EEPROM.end();
706         swSer.println("Modo automatico (o no) guardado");
707     }
708
709     void loadModoAutomatico(){
710         EEPROM.begin(512);
711         EEPROM.get(0+sizeof(ssid)+sizeof(password)+sizeof(modo_elegido),modo_automatico);
712         EEPROM.end();
713         swSer.println("Recovered modo_automatico:");
714         swSer.println(modo_automatico);
715         swSer.println((modo_automatico)>0?"********":"<no modo_automatico encontrado>");
716     }
717
718     void saveModoControlCalefaccion(){
719         EEPROM.begin(512);
720         delay(50);
721         EEPROM.put(0+sizeof(ssid)+sizeof(password)+sizeof(modo_elegido), modo_control);
722         EEPROM.commit();
723         EEPROM.end();
724         swSer.println("Modo de control de calefaccion guardado");
725     }
726
727     void loadModoControlCalefaccion(){
728         EEPROM.begin(512);
729         EEPROM.get(0+sizeof(ssid)+sizeof(password)+sizeof(modo_elegido), modo_control);
730         EEPROM.end();
731         swSer.println("Recovered modo_control:");
732         swSer.println(modo_control);
733         swSer.println((modo_control)>2?"********":"<no modo_control_calefaccion
            encontrado>");
734     }
735
736     void saveLimiteTemperatura(){
737       EEPROM.begin(512);
738       delay(50);
739
          EEPROM.put(0+sizeof(ssid)+sizeof(password)+sizeof(modo_elegido)+sizeof(modo_control)
          , temperatura_limite);
740       EEPROM.commit();
741       EEPROM.end();
742       swSer.println("Limite de temperatura guardado");
743     }
```

```
744
745   void loadLimiteTemperatura(){
746     EEPROM.begin(512);
747
      EEPROM.get(0+sizeof(ssid)+sizeof(password)+sizeof(modo_elegido)+sizeof(modo_control)
      , temperatura_limite);
748     EEPROM.end();
749     swSer.println("Recovered temperatura_limite:");
750     swSer.println(temperatura_limite);
751     swSer.println((temperatura_limite)>0?"********":"<no temperatura_limite>");
752   }
753
754   void saveModoAleatorio(){
755       EEPROM.begin(512);
756       delay(50);
757       EEPROM.put(0+sizeof(ssid)+sizeof(password)+sizeof(modo_elegido), modo_aleatorio);
758       EEPROM.commit();
759       EEPROM.end();
760       swSer.println("Modo aleatorio (o no) guardado");
761   }
762
763   void loadModoAleatorio(){
764       EEPROM.begin(512);
765       EEPROM.get(0+sizeof(ssid)+sizeof(password)+sizeof(modo_elegido), modo_aleatorio);
766       EEPROM.end();
767       swSer.println("Recovered modo_aleatorio:");
768       swSer.println(modo_aleatorio);
769       swSer.println((modo_aleatorio)>2?"********":"<no modo_aleatorio encontrado>");
770
771   }
772
773   void saveHora(){
774       EEPROM.begin(512);
775       EEPROM.put(0+sizeof(ssid)+sizeof(password)+sizeof(modo_elegido)+sizeof(int),
      hora_on);
776
      EEPROM.put(0+sizeof(ssid)+sizeof(password)+sizeof(modo_elegido)+sizeof(int)+sizeof
      (hora_on), minutos_on);
777
      EEPROM.put(0+sizeof(ssid)+sizeof(password)+sizeof(modo_elegido)+sizeof(int)+sizeof
      (hora_on)+sizeof(minutos_on), hora_off);
778
      EEPROM.put(0+sizeof(ssid)+sizeof(password)+sizeof(modo_elegido)+sizeof(int)+sizeof
      (hora_on)+sizeof(minutos_on)+sizeof(hora_off), minutos_off);
779       EEPROM.commit();
780       EEPROM.end();
781       swSer.println("Hora guardada");
782   }
783
784   void loadHora(){
785       EEPROM.begin(512);
786       EEPROM.get(0+sizeof(ssid)+sizeof(password)+sizeof(modo_elegido)+sizeof(int),
      hora_on);
787
      EEPROM.get(0+sizeof(ssid)+sizeof(password)+sizeof(modo_elegido)+sizeof(int)+sizeof
      (hora_on), minutos_on);
788
      EEPROM.get(0+sizeof(ssid)+sizeof(password)+sizeof(modo_elegido)+sizeof(int)+sizeof
      (hora_on)+sizeof(minutos_on), hora_off);
789
      EEPROM.get(0+sizeof(ssid)+sizeof(password)+sizeof(modo_elegido)+sizeof(int)+sizeof
      (hora_on)+sizeof(minutos_on)+sizeof(hora_off), minutos_off);
790       EEPROM.end();
791       swSer.println("Recovered hora:");
792       swSer.print("Hora activacion/subida: ");
793       swSer.print(hora_on);
794       swSer.print(":");
795       swSer.println(minutos_on);
796       swSer.print("Hora desactivacion/bajada: ");
797       swSer.print(hora_off);
798       swSer.print(":");
```

```
799         swSer.println(minutos_off);
800         hora_subida=hora_on;
801         hora_bajada=hora_off;
802         minutos_subida=minutos_on;
803         minutos_bajada=minutos_off;
804     }
805
806     void saveVariablesBooleanas(boolean var_bool_1, boolean var_bool_2){
807         EEPROM.begin(512);
808
            EEPROM.put(0+sizeof(ssid)+sizeof(password)+sizeof(modo_elegido)+sizeof(int)+sizeof
            (hora_on)+sizeof(minutos_on)+sizeof(hora_off)+
            sizeof(minutos_off),var_bool_1);//la variable booleana varía según el modo en el
            que esté
809
            EEPROM.put(0+sizeof(ssid)+sizeof(password)+sizeof(modo_elegido)+sizeof(int)+sizeof
            (hora_on)+sizeof(minutos_on)+sizeof(hora_off)+ sizeof(minutos_off)+
            sizeof(var_bool_1),var_bool_2);
810         EEPROM.commit();
811         EEPROM.end();
812         swSer.println("Variable booleana guardada");
813     }
814
815     boolean loadVariableBooleana1(){
816         EEPROM.begin(512);
817         boolean variable_booleana_1;
818
            EEPROM.get(0+sizeof(ssid)+sizeof(password)+sizeof(modo_elegido)+sizeof(int)+sizeof
            (hora_on)+sizeof(minutos_on)+sizeof(hora_off)+ sizeof(minutos_off),
            variable_booleana_1);
819         EEPROM.end();
820         swSer.print("Variable booleana 1 cargada: ");
821         swSer.println(variable_booleana_1);
822         return variable_booleana_1;
823     }
824
825     boolean loadVariableBooleana2(){
826         EEPROM.begin(512);
827         boolean variable_booleana_2;
828
            EEPROM.get(0+sizeof(ssid)+sizeof(password)+sizeof(modo_elegido)+sizeof(int)+sizeof
            (hora_on)+sizeof(minutos_on)+sizeof(hora_off)+ sizeof(minutos_off)+
            sizeof(variable_booleana_2), variable_booleana_2);
829         EEPROM.end();
830         swSer.print("Variable booleana 2 cargada: ");
831         swSer.println(variable_booleana_2);
832         return variable_booleana_2;
833     }
834
835     void saveModoControlPersiana(){
836         EEPROM.begin(512);
837         delay(50);
838         EEPROM.put(0+sizeof(ssid)+sizeof(password)+sizeof(modo_elegido), modo_control);
839         EEPROM.commit();
840         EEPROM.end();
841         swSer.println("Modo de control de persiana guardado");
842     }
843
844     void loadModoControlPersiana(){
845         EEPROM.begin(512);
846         EEPROM.get(0+sizeof(ssid)+sizeof(password)+sizeof(modo_elegido), modo_control);
847         EEPROM.end();
848         swSer.println("Recovered modo_control:");
849         swSer.println(modo_control);
850         swSer.println((modo_control)>2?"********":"<no modo_control_persiana
            encontrado>");
851     }
852
853
854     void saveVariablesMonitorizadas(){
855         EEPROM.begin(512);
```

```
856         delay(50);
857         EEPROM.put(0+sizeof(ssid)+sizeof(password)+sizeof(modo_elegido),
            variables_monitorizadas);
858         EEPROM.commit();
859         EEPROM.end();
860         swSer.println("Variables monitorizadas guardadas");
861     }
862
863     void loadVariablesMonitorizadas(){
864         EEPROM.begin(512);
865         EEPROM.get(0+sizeof(ssid)+sizeof(password)+sizeof(modo_elegido),
            variables_monitorizadas);
866         EEPROM.end();
867         swSer.println("Recovered variables_monitorizadas:");
868         swSer.println(variables_monitorizadas);
869         swSer.println((variables_monitorizadas)>0?"********":"<no
            variables_monitorizadas encontrado>");
870     }
871
872     void saveTiempoEntreEnvioDatos(){
873         EEPROM.begin(512);
874         delay(50);
875
            EEPROM.put(0+sizeof(ssid)+sizeof(password)+sizeof(modo_elegido)+sizeof(variables_m
            onitorizadas), tiempo_entre_envio_datos);
876         EEPROM.commit();
877         EEPROM.end();
878         swSer.println("Tiempo entre envio de datos guardado");
879     }
880
881     void loadTiempoEntreEnvioDatos(){
882         EEPROM.begin(512);
883
            EEPROM.get(0+sizeof(ssid)+sizeof(password)+sizeof(modo_elegido)+sizeof(variables_m
            onitorizadas), tiempo_entre_envio_datos);
884         EEPROM.end();
885         swSer.println("Recovered tiempo_entre_envio_datos:");
886         swSer.println(tiempo_entre_envio_datos);
887         swSer.println((tiempo_entre_envio_datos)>0?"********":"<no
            tiempo_entre_envio_datos encontrado>");
888     }
889
890     //Máquina de estados de la configuración
891
892     void maquinaEstadosConfig(){
893         switch (estado){
894                 case Credenciales:
895                     if(!ssid_leido && primera_vez){
896                         swSer.println("Introduzca la SSID:");
897                         primera_vez=false;
898                     }
899                     if(!password_leido && ssid_leido && primera_vez){
900                         swSer.println("Introduzca el password:");
901                         primera_vez=false;
902                     }
903                     if(!modo_elegido_leido && ssid_leido && password_leido &&
                        primera_vez){
904                         swSer.println("Introduzca el numero de modo de funcionamiento:");
905                         swSer.println("modo 1: Control de luces");
906                         swSer.println("modo 2: Control de temperatura");
907                         swSer.println("modo 3: Persiana en vacaciones");
908                         swSer.println("modo 4: Luces en vacaciones");
909                         swSer.println("modo 5: Control de la persiana");
910                         swSer.println("modo 6: Monitorizacion de sensores");
911                         swSer.println("modo 7: Control de presencia");
912                         swSer.println("modo 8: Telefonillo");
913
914                         primera_vez=false;
915                     }
916
917
```

```
                if(string_formado!= "" && !ssid_leido) {
                    string_formado.toCharArray(ssid, 20);
                    string_formado="";
                    ssid_leido=true;
                    primera_vez=true;
                    swSer.println("ssid leido");

                }

                if(string_formado!= "" && !password_leido) {
                    string_formado.toCharArray(password, 20);
                    string_formado="";
                    password_leido=true;
                    primera_vez=true;
                    swSer.println("password leido");
                    saveCredentials();
                }

                if(string_formado!= "" && !modo_elegido_leido) {
                    modo_elegido=string_formado.toInt();
                    string_formado="";
                    primera_vez=true;
                    modo_elegido_leido=true;
                    switch(modo_elegido){
                        case 1:
                            estado=EleccionParametros1;
                        break;

                        case 2:
                            estado=EleccionParametros2;
                        break;

                        case 3:
                        case 4:
                            estado=EleccionParametros3;
                        break;

                        case 5:
                            estado=EleccionParametros4;
                        break;

                        case 6:
                            estado=EleccionParametros5;
                        break;

                        //los pongo sin break para que así cualquiera que sea se ejecutará lo mismo
                        case 7:
                        case 8:
                            modo=modo_elegido;
                        break;

                        default:
                            swSer.println("Valor incorrecto. Introduzca un numero entre 1 y 8");
                            modo_elegido=0;
                            modo_elegido_leido=false;//Con esto volveré a pedir al usuario que introduzca un modo correcto

                    }

                    swSer.println("Modo elegido: ");
                    swSer.println(modo_elegido);
                    if(modo_elegido_leido){//Compruebo si se ha metido correctamente el modo
                        saveModoElegido();
                    }
                    if((modo_elegido== 7) || (modo_elegido== 8))
                        ESP.restart();
                }
            break;
```

```
985
986             case EleccionParametros1://Control luces
987                 if(!modo_automatico_elegido && primera_vez){
988                     swSer.println("¿Desea un control automatico de la luz? (SI/NO)");
989                     primera_vez=false;
990                 }
991
992                 if(string_formado!= "" && !modo_automatico_elegido) {
993                     //string_formado.toCharArray(respuesta, 3);
994                     modo_automatico_elegido=true;
995                     primera_vez=true;
996                     if(string_formado=="SI"){
997                       modo_automatico=true;
998                       //respuesta_int="true";
999                     }
1000                    else if(string_formado=="NO"){
1001                      modo_automatico=false;
1002                      //respuesta_int="false";
1003                    }
1004                    else{
1005                      modo_automatico_elegido=false;
1006                      swSer.println("Respuesta no valida. Introduzca SI o NO");
1007                    }
1008
1009
1010                    if(modo_automatico_elegido){
1011                      saveModoAutomatico();
1012                      modo=modo_elegido;//Aqui le indico ya que comience con la
                          aplicacion correspondiente
1013                      swSer.println("Respuesta recibida:");
1014                      swSer.println(string_formado);
1015                      ESP.restart();//esta funcion me resetea completamente el micro
1016                    }
1017                    string_formado="";
1018                }
1019
1020
1021            break;
1022
1023            case EleccionParametros2://Control Temperatura(calefaccion)
1024                if(!modo_control_calefaccion_leido && primera_vez){
1025                    swSer.println("Introduzca el numero de modo de control:");
1026                    swSer.println("0: Control manual del usuario a traves de la
                        aplicacion movil");
1027                    swSer.println("1: Control por temperatura");
1028                    swSer.println("2: Control por tiempo");
1029                    primera_vez=false;
1030                }
1031                if(leer_temperatura_limite && modo_control_calefaccion_leido &&
                    primera_vez){
1032                    swSer.println("Introduzca la temperatura limite:");
1033                    primera_vez=false;
1034                }
1035                if(leer_tiempo && modo_control_calefaccion_leido && !hora_on_leida
                    && primera_vez){
1036                    swSer.println("Introduzca la hora de encendido de la calefaccion
                        (Entre 00 y 23):");
1037                    primera_vez=false;
1038                }
1039                else if(leer_tiempo && modo_control_calefaccion_leido &&
                    !minutos_on_leidos && primera_vez){
1040                    swSer.println("Introduzca los minutos de encendido (Entre 00 y
                        59):");
1041                    primera_vez=false;
1042                }
1043                else if(leer_tiempo && modo_control_calefaccion_leido &&
                    !hora_off_leida && primera_vez){
1044                    swSer.println("Introduzca la hora de apagado de la calefaccion
                        (Entre 00 y 23):");
1045                    primera_vez=false;
1046                }
```

```
1047                else if(leer_tiempo && modo_control_calefaccion_leido &&
                    !minutos_off_leidos && primera_vez){
1048                    swSer.println("Introduzca los minutos de apagado (Entre 00 y
                        59):");
1049                    primera_vez=false;
1050                }
1051
1052
1053                if(string_formado!= "" && !modo_control_calefaccion_leido) {
1054                    modo_control=string_formado.toInt();
1055                    string_formado="";
1056                    modo_control_calefaccion_leido=true;
1057                    primera_vez=true;
1058                    if(modo_control==0){
1059                      saveModoControlCalefaccion();
1060                      ESP.restart();
1061                    }
1062                    else if(modo_control==1){
1063                        leer_temperatura_limite=true;
1064                        leer_tiempo=false;
1065                    }
1066                    else if(modo_control==2){
1067                        leer_temperatura_limite=false;
1068                        leer_tiempo=true;
1069                    }
1070                    else{
1071                        swSer.println("Valor incorrecto. Introduzca 0,1 o 2");
1072                        modo_control=3;
1073                        modo_control_calefaccion_leido=false;//Con esto volveré a
                            pedir al usuario que introduzca un modo correcto
1074                    }
1075
1076                  swSer.println("Modo de control calefaccion leido:");
1077                  swSer.println(modo_control);
1078
                    saveModoControlCalefaccion();

1079                }
1080
1081                if(string_formado!= "" && leer_temperatura_limite) {
1082                    temperatura_limite=string_formado.toInt();
1083                    string_formado="";
1084                    primera_vez=true;
1085                    leer_temperatura_limite=false;
1086                    swSer.println("Temperatura limite leida");
1087                    swSer.println(temperatura_limite);
1088                    saveLimiteTemperatura();
1089                    modo=modo_elegido;
1090                    ESP.restart();//esta funcion me resetea completamente el micro
1091                }
1092                if(string_formado!= "" && leer_tiempo && !hora_on_leida) {
1093                    hora_on=string_formado.toInt();
1094                    string_formado="";
1095                    primera_vez=true;
1096                    hora_on_leida=true;
1097                    if(hora_on<0 || hora_on>23){
1098                        swSer.println("Hora de encendido de la calefaccion
                            Incorrecta:");
1099                        hora_on_leida=false;
1100                    }
1101                    if(hora_on_leida){
1102                        swSer.println("Hora de encendido de la calefaccion leida:");
1103                        swSer.println(hora_on);
1104                    }
1105
1106                }
1107                if(string_formado!= "" && leer_tiempo && !minutos_on_leidos) {
1108                    minutos_on=string_formado.toInt();
1109                    string_formado="";
1110                    primera_vez=true;
1111                    minutos_on_leidos=true;
```

```
1112                    if(minutos_on<0 || minutos_on>59){
1113                        swSer.println("Minutos de encendido de la calefaccion
                            Incorrecta:");
1114                        minutos_on_leidos=false;
1115                    }
1116                    if(hora_on_leida){
1117                        swSer.println("Minutos de encendido de la calefaccion
                            leidos:");
1118                        swSer.println(minutos_on);
1119                    }
1120                }
1121                 if(string_formado!= "" && leer_tiempo && !hora_off_leida) {
1122                    hora_off=string_formado.toInt();
1123                    string_formado="";
1124                    primera_vez=true;
1125                    hora_off_leida=true;
1126                    if(hora_off<0 || hora_off>23){
1127                        swSer.println("Hora de apagado de la calefaccion
                            Incorrecta:");
1128                        hora_off_leida=false;
1129                    }
1130                    if(hora_off_leida){
1131                        swSer.println("Hora de apagado de la calefaccion leida:");
1132                        swSer.println(hora_off);
1133                    }
1134
1135                }
1136                if(string_formado!= "" && leer_tiempo && !minutos_off_leidos) {
1137                    minutos_off=string_formado.toInt();
1138                    string_formado="";
1139                    primera_vez=true;
1140                    minutos_off_leidos=true;
1141                    if(minutos_off<0 || minutos_off>59){
1142                        swSer.println("Minutos de apagado de la calefaccion
                            Incorrecta:");
1143                        minutos_off_leidos=false;
1144                    }
1145                    if(minutos_off_leidos && hora_off_leida && minutos_on_leidos &&
                    hora_on_leida){
1146                        swSer.println("Minutos de apagado de la calefaccion leidos:");
1147                        swSer.println(minutos_off);
1148                        saveHora();
1149                        modo=modo_elegido;
1150                        ESP.restart();//esta funcion me resetea completamente el micro
1151                    }
1152                }
1153
1154            break;
1155
1156            case EleccionParametros3://Vacaciones Persiana y Vacaciones Luces
1157                if(!modo_aleatorio_elegido && primera_vez){
1158                    swSer.println("¿Desea un control aleatorio del horario de
                        activacion/desactivacion (o subida/bajada)? (SI/NO)");
1159                    primera_vez=false;
1160                }
1161                if(leer_tiempo && modo_aleatorio_elegido && !hora_on_leida &&
                primera_vez){
1162                    swSer.println("Introduzca la hora de activacion/subida (Entre 00
                        y 23):");
1163                    primera_vez=false;
1164                }
1165                else if(leer_tiempo && modo_aleatorio_elegido && !minutos_on_leidos
                && primera_vez){
1166                    swSer.println("Introduzca los minutos de activacion/subida
                        (Entre 00 y 59):");
1167                    primera_vez=false;
1168                }
1169                else if(leer_tiempo && modo_aleatorio_elegido && !hora_off_leida &&
                primera_vez){
1170                    swSer.println("Introduzca la hora de desactivacion/bajada(Entre
                        00 y 23):");
```

```
1171                    primera_vez=false;
1172                }
1173                else if(leer_tiempo && modo_aleatorio_elegido && !minutos_off_leidos
                    && primera_vez){
1174                    swSer.println("Introduzca los minutos desactivacion/bajada
                        (Entre 00 y 59):");
1175                    primera_vez=false;
1176                }
1177
1178                if(string_formado!= "" && !modo_aleatorio_elegido) {
1179                    modo_aleatorio_elegido=true;
1180                    primera_vez=true;
1181                    if(string_formado=="SI"){
1182                      modo_aleatorio=true;
1183                      modo=modo_elegido;//Aqui le indico ya que comience con la
                        aplicacion correspondiente
1184                      leer_tiempo=false;
1185
1186                    }
1187                    else if(string_formado=="NO"){
1188                      modo_aleatorio=false;
1189                      leer_tiempo=true;
1190                      respuesta_int=0;
1191                    }
1192                    else{
1193                      modo_aleatorio_elegido=false;
1194                      swSer.println("Respuesta no valida. Introduzca SI o NO");
1195                    }
1196
1197
1198                    if(modo_aleatorio_elegido){
1199                      saveModoAleatorio();
1200                      swSer.println("Respuesta recibida:");
1201                      swSer.println(string_formado);
1202                      if(modo_aleatorio){
1203                        ESP.restart();//esta funcion me resetea completamente el micro
1204                      }
1205                    }
1206                    string_formado="";
1207                }
1208                if(string_formado!= "" && leer_tiempo && !hora_on_leida) {
1209                    hora_on=string_formado.toInt();
1210                    string_formado="";
1211                    primera_vez=true;
1212                    hora_on_leida=true;
1213                    if(hora_on<0 || hora_on>23){
1214                        swSer.println("Hora de activacion/subida Incorrecta:");
1215                        hora_on_leida=false;
1216                    }
1217                    if(hora_on_leida){
1218                        swSer.println("Hora de activacion/subida leida:");
1219                        swSer.println(hora_on);
1220                    }
1221
1222                }
1223                if(string_formado!= "" && leer_tiempo && !minutos_on_leidos) {
1224                    minutos_on=string_formado.toInt();
1225                    string_formado="";
1226                    primera_vez=true;
1227                    minutos_on_leidos=true;
1228                    if(minutos_on<0 || minutos_on>59){
1229                        swSer.println("Minutos de activacion/subida Incorrecta:");
1230                        minutos_on_leidos=false;
1231                    }
1232                    if(hora_on_leida){
1233                        swSer.println("Minutos de activacion/subida leidos:");
1234                        swSer.println(minutos_on);
1235                    }
1236                }
1237                 if(string_formado!= "" && leer_tiempo && !hora_off_leida) {
1238                    hora_off=string_formado.toInt();
```

```
1239                    string_formado="";
1240                    primera_vez=true;
1241                    hora_off_leida=true;
1242                    if(hora_off<0 || hora_off>23){
1243                        swSer.println("Hora de desactivacion/bajada Incorrecta:");
1244                        hora_off_leida=false;
1245                    }
1246                    if(hora_off_leida){
1247                        swSer.println("Hora de desactivacion/bajada leida:");
1248                        swSer.println(hora_off);
1249                    }
1250
1251                }
1252                if(string_formado!= "" && leer_tiempo && !minutos_off_leidos) {
1253                    minutos_off=string_formado.toInt();
1254                    string_formado="";
1255                    primera_vez=true;
1256                    minutos_off_leidos=true;
1257                    if(minutos_off<0 || minutos_off>59){
1258                        swSer.println("Minutos de desactivacion/bajada Incorrecta:");
1259                        minutos_off_leidos=false;
1260                    }
1261                    if(minutos_off_leidos && hora_off_leida && minutos_on_leidos &&
                    hora_on_leida){
1262                        swSer.println("Minutos de desactivacion/bajada leidos:");
1263                        swSer.println(minutos_off);
1264                        saveHora();
1265                        modo=modo_elegido;
1266                        ESP.restart();//esta funcion me resetea completamente el micro
1267                    }
1268                }
1269
1270            break;
1271
1272            case EleccionParametros4://ControlPersiana
1273                if(!modo_control_persiana_leido && primera_vez){
1274                    swSer.println("Introduzca el numero de modo de control:");
1275                    swSer.println("0: Control manual del usuario a traves de la
                    aplicacion movil");
1276                    swSer.println("1: Control por tiempo");
1277                    primera_vez=false;
1278                }
1279                if(leer_tiempo && modo_control_persiana_leido && !hora_on_leida &&
                primera_vez){
1280                    swSer.println("Introduzca la hora de subida de la persiana
                    (Entre 00 y 23):");
1281                    primera_vez=false;
1282                }
1283                else if(leer_tiempo && modo_control_persiana_leido &&
                !minutos_on_leidos && primera_vez){
1284                    swSer.println("Introduzca los minutos de subida de la persiana
                    (Entre 00 y 59):");
1285                    primera_vez=false;
1286                }
1287                else if(leer_tiempo && modo_control_persiana_leido &&
                !hora_off_leida && primera_vez){
1288                    swSer.println("Introduzca la hora de bajada de la persiana
                    (Entre 00 y 23):");
1289                    primera_vez=false;
1290                }
1291                else if(leer_tiempo && modo_control_persiana_leido &&
                !minutos_off_leidos && primera_vez){
1292                    swSer.println("Introduzca los minutos de bajada de la persiana
                    (Entre 00 y 59):");
1293                    primera_vez=false;
1294                }
1295
1296
1297                if(string_formado!= "" && !modo_control_persiana_leido) {
1298                    modo_control=string_formado.toInt();
1299                    string_formado="";
```

```
1300                    modo_control_persiana_leido=true;
1301                    primera_vez=true;
1302                    if(modo_control==0){
1303                      saveModoControlPersiana();
1304                      ESP.restart();
1305                    }
1306                    else if(modo_control==1){
1307                        leer_tiempo=true;
1308                    }
1309                    else{
1310                        swSer.println("Valor incorrecto. Introduzca 0 o 1");
1311                        modo_control=5;
1312                        modo_control_persiana_leido=false;//Con esto volveré a pedir
                            al usuario que introduzca un modo correcto
1313                    }
1314
1315                    swSer.println("Modo de control de persiana leido:");
1316                    swSer.println(modo_control);
1317                    saveModoControlPersiana();
1318                }
1319
1320                if(string_formado!= "" && leer_tiempo && !hora_on_leida) {
1321                    hora_on=string_formado.toInt();
1322                    string_formado="";
1323                    primera_vez=true;
1324                    hora_on_leida=true;
1325                    if(hora_on<0 || hora_on>23){
1326                        swSer.println("Hora de subida de la persiana Incorrecta:");
1327                        hora_on_leida=false;
1328                    }
1329                    if(hora_on_leida){
1330                        swSer.println("Hora de subida de la persiana leida:");
1331                        swSer.println(hora_on);
1332                    }
1333
1334                }
1335                if(string_formado!= "" && leer_tiempo && !minutos_on_leidos) {
1336                    minutos_on=string_formado.toInt();
1337                    string_formado="";
1338                    primera_vez=true;
1339                    minutos_on_leidos=true;
1340                    if(minutos_on<0 || minutos_on>59){
1341                        swSer.println("Minutos de subida de la persiana Incorrecta:");
1342                        minutos_on_leidos=false;
1343                    }
1344                    if(hora_on_leida){
1345                        swSer.println("Minutos de subida de la persiana leidos:");
1346                        swSer.println(minutos_on);
1347                    }
1348                }
1349                 if(string_formado!= "" && leer_tiempo && !hora_off_leida) {
1350                    hora_off=string_formado.toInt();
1351                    string_formado="";
1352                    primera_vez=true;
1353                    hora_off_leida=true;
1354                    if(hora_off<0 || hora_off>23){
1355                        swSer.println("Hora de bajada de la persiana Incorrecta:");
1356                        hora_off_leida=false;
1357                    }
1358                    if(hora_off_leida){
1359                        swSer.println("Hora de bajada de la persiana leida:");
1360                        swSer.println(hora_off);
1361                    }
1362
1363                }
1364                if(string_formado!= "" && leer_tiempo && !minutos_off_leidos) {
1365                    minutos_off=string_formado.toInt();
1366                    string_formado="";
1367                    primera_vez=true;
1368                    minutos_off_leidos=true;
1369                    if(minutos_off<0 || minutos_off>59){
```

```
1370                        swSer.println("Minutos de bajada de la persiana Incorrecta:");
1371                        minutos_off_leidos=false;
1372                    }
1373                    if(minutos_off_leidos && hora_off_leida && minutos_on_leidos &&
                        hora_on_leida){
1374                        swSer.println("Minutos de bajada de la persiana leidos:");
1375                        swSer.println(minutos_off);
1376                        saveHora();
1377                        modo=modo_elegido;
1378                        ESP.restart();//esta funcion me resetea completamente el micro
1379                    }
1380                }
1381            break;
1382
1383            case EleccionParametros5://MonitorizacionVariables
1384                if(!variables_monitorizadas_leidas && primera_vez){
1385                    swSer.println("Introduzca el numero de modo de variables
                        monitorizadas:");
1386                    swSer.println("1: Temperatura y humedad");
1387                    swSer.println("2: Temperatura,humedad y presencia movimiento");
1388                    swSer.println("3: Temperatura,humedad y apertura de puerta");
1389                    primera_vez=false;
1390                }
1391                if(!tiempo_entre_envio_datos_leido && variables_monitorizadas_leidas
                    && primera_vez){
1392                    swSer.println("Introduzca un numero de segundos entre envio de
                        datos a la nube (min 60, maximo 3600):");
1393                    primera_vez=false;
1394                }
1395
1396
1397                if(string_formado!= "" && !variables_monitorizadas_leidas) {
1398                    variables_monitorizadas=string_formado.toInt();
1399                    string_formado="";
1400                    variables_monitorizadas_leidas=true;
1401                    primera_vez=true;
1402                    if(variables_monitorizadas<1 || variables_monitorizadas>3){
1403                        swSer.println("Valor incorrecto. Introduzca 1,2 o 3");
1404                        variables_monitorizadas=0;
1405                        variables_monitorizadas_leidas=false;//Con esto volveré a
                            pedir al usuario que introduzca un modo correcto
1406                    }
1407                    if(variables_monitorizadas_leidas){
1408                        swSer.println("Numero de variables a monitorizar:");
1409                        swSer.println(variables_monitorizadas);
1410                        saveVariablesMonitorizadas();
1411                    }
1412
1413                }
1414                if(string_formado!= "" && !tiempo_entre_envio_datos_leido) {
1415                    tiempo_entre_envio_datos=string_formado.toInt();
1416                    string_formado="";
1417                    primera_vez=true;
1418                    tiempo_entre_envio_datos_leido=true;
1419                    if(tiempo_entre_envio_datos<60 || tiempo_entre_envio_datos>3600){
1420                        swSer.println("Valor incorrecto. Introduzca un numero de
                            segundos entre 60 y 3600");
1421                        tiempo_entre_envio_datos=0;
1422                        tiempo_entre_envio_datos_leido=false;//Con esto volveré a
                            pedir al usuario que introduzca un modo correcto
1423                    }
1424                    if(tiempo_entre_envio_datos_leido){
1425                        swSer.println("Tiempo entre envio de datos a la nube:");
1426                        swSer.print(tiempo_entre_envio_datos);
1427                        swSer.println(" segundos");
1428                        saveTiempoEntreEnvioDatos();
1429                        modo=modo_elegido;
1430                        ESP.restart();//esta funcion me resetea completamente el micro
1431                    }
1432
1433                }
```

```
            break;
        }
    }

    ////////////////////////////////////////////////////////////////////////////////////////////////
    ////////////////////////////////
    ////////////////////////////////////////////////////////////////////////////////////////////////
    ////////////////////////////////
    //SETUP
    void setup() {
        swSer.begin(115200);
        delay(500);
       attachInterrupt(SWITCH2,interr_SWITCH2, CHANGE);
      //Cargar configuracion guardada en memoria
        loadCredentials();
        loadModoElegido();

        if(modo_elegido<1 || modo_elegido>8){
            modo=9;//Configuracion
        }
        switch(modo_elegido){
            case ControlLuces:
                loadModoAutomatico();
            break;

            case ControlTemperatura:
                loadModoControlCalefaccion();
                calefaccion_encendida=loadVariableBooleana1();
                if(calefaccion_encendida!=0 && calefaccion_encendida!=1)
                    calefaccion_encendida=false;
                if(modo_control==1){
                    loadLimiteTemperatura();
                }
                else if(modo_control==2){
                    loadHora();
                }
            break;

            case VacacionesPersiana:
                loadModoAleatorio();
                ya_subida=loadVariableBooleana1();
                if(ya_subida!=0 && ya_subida!=1)
                    ya_subida=false;
                ya_bajada=loadVariableBooleana2();
                if(ya_bajada!=0 && ya_bajada!=1)
                    ya_bajada=false;
                if(modo_aleatorio==0)
                  loadHora();
            break;

            case VacacionesLuces:
                loadModoAleatorio();
                luces_encendidas=loadVariableBooleana1();//me devuelve un booleano que
                lee de la memoria
                if(luces_encendidas!=0 && luces_encendidas!=1)
                    luces_encendidas=false;
                if(modo_aleatorio==0){
                  loadHora();
                }
            break;

            case ControlPersiana:
                loadModoControlPersiana();
                if(modo_control==1){
                    ya_subida=loadVariableBooleana1();
                    if(ya_subida!=0 && ya_subida!=1)
                        ya_subida=false;
                    ya_bajada=loadVariableBooleana2();
                    if(ya_bajada!=0 && ya_bajada!=1)
```

```
                    ya_bajada=false;
                loadHora();
            }
        break;

        case MonitorizacionVariables:
            loadVariablesMonitorizadas();
            loadTiempoEntreEnvioDatos();
        break;
    }
    modo=modo_elegido;

    //Declaracion entradas/salidas ESP12
    pinMode(DHTPIN, INPUT);
    pinMode(REED, INPUT);
    pinMode(LDR, INPUT);
    pinMode(SWITCH2, INPUT);
    pinMode(SWITCH3, INPUT);
    pinMode(PIR, INPUT);

    pinMode(RELE1, OUTPUT);
    pinMode(RELE2, OUTPUT);
    digitalWrite(RELE1, LOW);
    digitalWrite(RELE2, LOW);
    pinMode(LED, OUTPUT);

    //Conexión wifi a thinger
    thing.add_wifi(ssid, password);
    swSer.println("Wifi conectado a thinger");
    //Sincronizacion hora
    WiFi.begin(ssid, password);
    while (WiFi.status() != WL_CONNECTED) {
      delay(500);
      swSer.print(".");
      if(modo==9){
        break;
      }

    }
    Udp.begin(localPort);
    setSyncProvider(getNtpTime);

    //Inicialización DHT11 (o 22)
        dht.begin();

    //Interrupción PIR
     attachInterrupt(PIR, interr_presencia_detectada_PIR, RISING); //este tema esta
     sin entender muy bien

    //Interrupción REED
     attachInterrupt(REED, interr_presencia_detectada_REED, RISING);

    //Interrupción Finales carrera
     attachInterrupt(FINALCARRERA1,interr_limite_alcanzado_persiana,
     RISING);//también vale para el final de carrera 2

    //Interrupción Interruptores
//     attachInterrupt(SWITCH2,interr_SWITCH2, CHANGE);
     attachInterrupt(SWITCH3,interr_SWITCH3, RISING);

    if(modo== MonitorizacionVariables){
        thing["dht11"] >> [](pson& out){
          alimento_sensor();
          delay(100);
          out["Temperatura"] = dht.readTemperature();
          out["Humedad"] = dht.readHumidity();
          swSer.println("Temperatura y humedad leidas");
          quito_alimentacion_sensor();
        };
    }

```

```
1571
1572        if(modo==ControlTemperatura){
1573            thing["Calefaccion"] << [](pson& in){
1574                if(in){
1575                    encender_rele_enclavamiento();
1576                }
1577                else{
1578                    apagar_rele_enclavamiento();
1579                }
1580            };
1581        }
1582
1583        if(modo==ControlPersiana){
1584            thing["Subir Persiana"] << [](pson& in){
1585                if(in){
1586                    digitalWrite(RELE1,HIGH);
1587                    digitalWrite(RELE2,LOW);
1588                }
1589                else{
1590                    digitalWrite(RELE1,LOW);
1591                }
1592            };
1593
1594            thing["Bajar Persiana"] << [](pson& in){
1595                if(in){
1596                    digitalWrite(RELE2,HIGH);
1597                    digitalWrite(RELE1,LOW);
1598                }
1599                else{
1600                    digitalWrite(RELE2,LOW);
1601                }
1602            };
1603
1604
1605
1606        }
1607
1608        if(modo== VacacionesLuces || modo==ControlLuces){
1609            thing["Luces"] << [](pson& in){
1610                if(in){
1611                    encender_rele_enclavamiento();
1612                }
1613                else{
1614                    apagar_rele_enclavamiento();
1615                }
1616            };
1617        }
1618
1619        //Telefonillo
1620        if(modo == Telefonillo){
1621            thing["Telefonillo"] << [](pson& in){
1622                if(in){
1623
1624                    digitalWrite(RELE1,HIGH);
1625                    delay(3000);
1626                    digitalWrite(RELE1,LOW);
1627                    in=0;
1628
1629                }
1630                else{
1631                }
1632            };
1633
1634        }
1635    }
1636    ////////////////////////////////////////////////////////////////////////////////
    ////////////////////////////////
1637
1638    ////////////////////////////////////////////////////////////////////////////////
    ////////////////////////////////
1639    //BUCLE PRINCIPAL
```

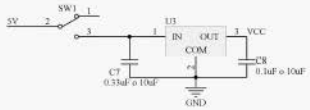
```
1640  void loop() {
1641        if(modo!=9){
1642            thing.handle();
1643            if (timeStatus() != timeNotSet) {
1644              if (now() != prevDisplay) { //update the display only if time has changed
1645                prevDisplay = now();
1646                digitalClockDisplay();
1647              }
1648            }
1649        }
1650       switch (modo){
1651         case ControlLuces:
1652           if(modo_automatico){
1653             control_luminosidad();
1654           }
1655         break;
1656
1657         case ControlTemperatura:
1658           if(modo_control==0){}
1659           else if(modo_control==1){
1660                 control_grados();
1661                 if(currentMillis - previousMillis > (interval*1000)) {
1662                     previousMillis = currentMillis;
1663                     primera_vez=true;
1664                     if(!no_dormir){
1665                       saveVariablesBooleanas(calefaccion_encendida,0);
1666                       ESP.deepSleep(SLEEP_TIME * 1000000, WAKE_RF_DEFAULT);//cuando
                          han pasado SLEEP_TIME segundos por la GPIO16 mando un pulso al
                          RST que despierta el micro y vuelve a funcionar
1667                     }
1668                 }
1669           }
1670           else if(modo_control==2) {
1671                 control_tiempo_calefaccion();
1672                 if(!no_dormir){
1673                       saveVariablesBooleanas(calefaccion_encendida,0);
1674                       ESP.deepSleep(SLEEP_TIME * 1000000, WAKE_RF_DEFAULT);//cuando
                          han pasado SLEEP_TIME segundos por la GPIO16 mando un pulso al
                          RST que despierta el micro y vuelve a funcionar
1675                 }
1676           }
1677         break;
1678
1679         case VacacionesPersiana:
1680           if(modo_aleatorio){
1681             control_aleatorio_persiana();
1682           }
1683           else{
1684             control_tiempo_persiana();
1685           }
1686           if(!no_dormir){
1687             saveVariablesBooleanas(ya_subida,ya_bajada);
1688             ESP.deepSleep(SLEEP_TIME * 1000000, WAKE_RF_DEFAULT);//cuando han pasado
                SLEEP_TIME segundos por la GPIO16 mando un pulso al RST que despierta el
                micro y vuelve a funcionar
1689           }
1690         break;
1691
1692         case VacacionesLuces:
1693           if(modo_aleatorio){
1694             control_aleatorio_luces();
1695           }
1696           else{
1697             control_tiempo_luces();
1698           }
1699           if(!no_dormir){
1700             saveVariablesBooleanas(luces_encendidas,0);
1701             ESP.deepSleep(SLEEP_TIME * 1000000, WAKE_RF_DEFAULT);//cuando han pasado
                SLEEP_TIME segundos por la GPIO16 mando un pulso al RST que despierta el
                micro y vuelve a funcionar
1702           }
```

```
1703        break;
1704
1705        case ControlPersiana:
1706            if(modo_control==0){}
1707            else if(modo_control==1) {
1708                  control_tiempo_persiana();
1709                  if(!no_dormir){
1710                        saveVariablesBooleanas(ya_subida,ya_bajada);
1711                        ESP.deepSleep(SLEEP_TIME * 1000000, WAKE_RF_DEFAULT);//cuando
                          han pasado SLEEP_TIME segundos por la GPIO16 mando un pulso al
                          RST que despierta el micro y vuelve a funcionar
1712                  }
1713              }
1714        break;
1715
1716        case MonitorizacionVariables:
1717          alimento_sensor();
1718          delay(500);
1719          if(variables_monitorizadas==1){
1720            //monitorizar_temperatura();
1721            currentMillis = millis();
1722
1723            if(currentMillis - previousMillis > (interval*1000)) {
1724              previousMillis = currentMillis;
1725              ESP.deepSleep(tiempo_entre_envio_datos * 1000000,
                  WAKE_RF_DEFAULT);//cuando han pasado tiempo segundos por la GPIO16 mando
                  un pulso al RST que despierta el micro y vuelve a funcionar
1726            }
1727          }
1728          else if(variables_monitorizadas==2){
1729            monitorizar_temperatura();
1730            monitorizar_presencia_movimiento();
1731          }
1732          else {
1733            monitorizar_temperatura();
1734            monitorizar_apertura_puerta();
1735          }
1736          quito_alimentacion_sensor();
1737        break;
1738
1739        case ControlPresencia:
1740          if(presencia_detectada){
1741            swSer.println("presencia_detectada");
1742            thing.call_endpoint("PIR_DETECTION");
1743            presencia_detectada=false;
1744          }
1745        break;
1746
1747        case Telefonillo:
1748          control_telefonillo();
1749        break;
1750
1751        case Configuracion:
1752            lecturaSerial();
1753            maquinaEstadosConfig();
1754        break;
1755      }
1756  }
1757  ////////////////////////////////////////////////////////////////////////////////
      ////////////////////////////////
1758
1759
1760
```

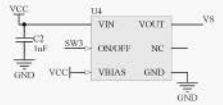
8.3. ANEXO 3: PLANOS PCB

CONVERSOR AC/DC

REGULADOR 5-3.3V

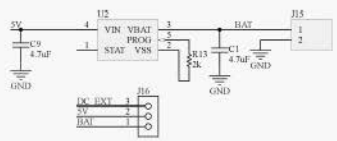


ALIMENTACIÓN VS PARA SENSORES

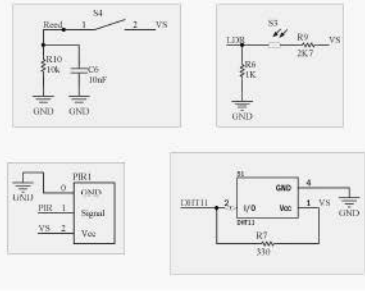


ALIMENTACIÓN DC EXTERNA

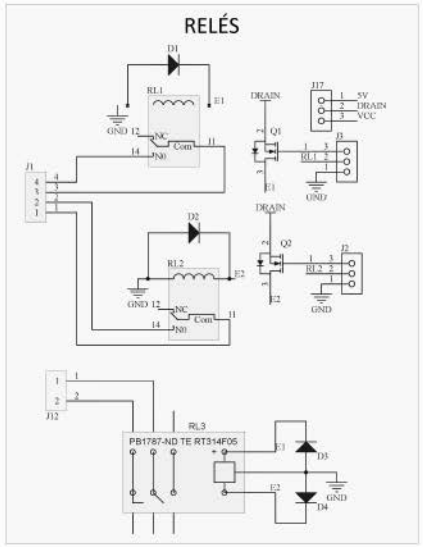
CARGADOR BATERÍA

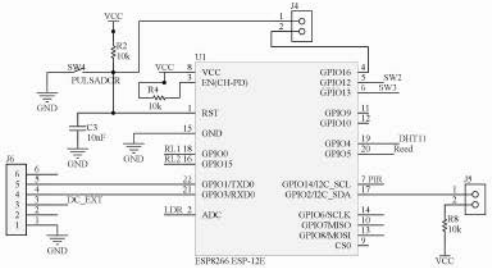


SENSORES



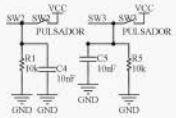
RELÉS





FINALES DE CARRERA

PULSADORES CONTROL RELÉS



	Nombre	Fecha	
Dibujado	Ángel Pina	20/5/2017	

Escuela de Ingeniería y Arquitectura
UniversidadZaragoza

Escala: S/E

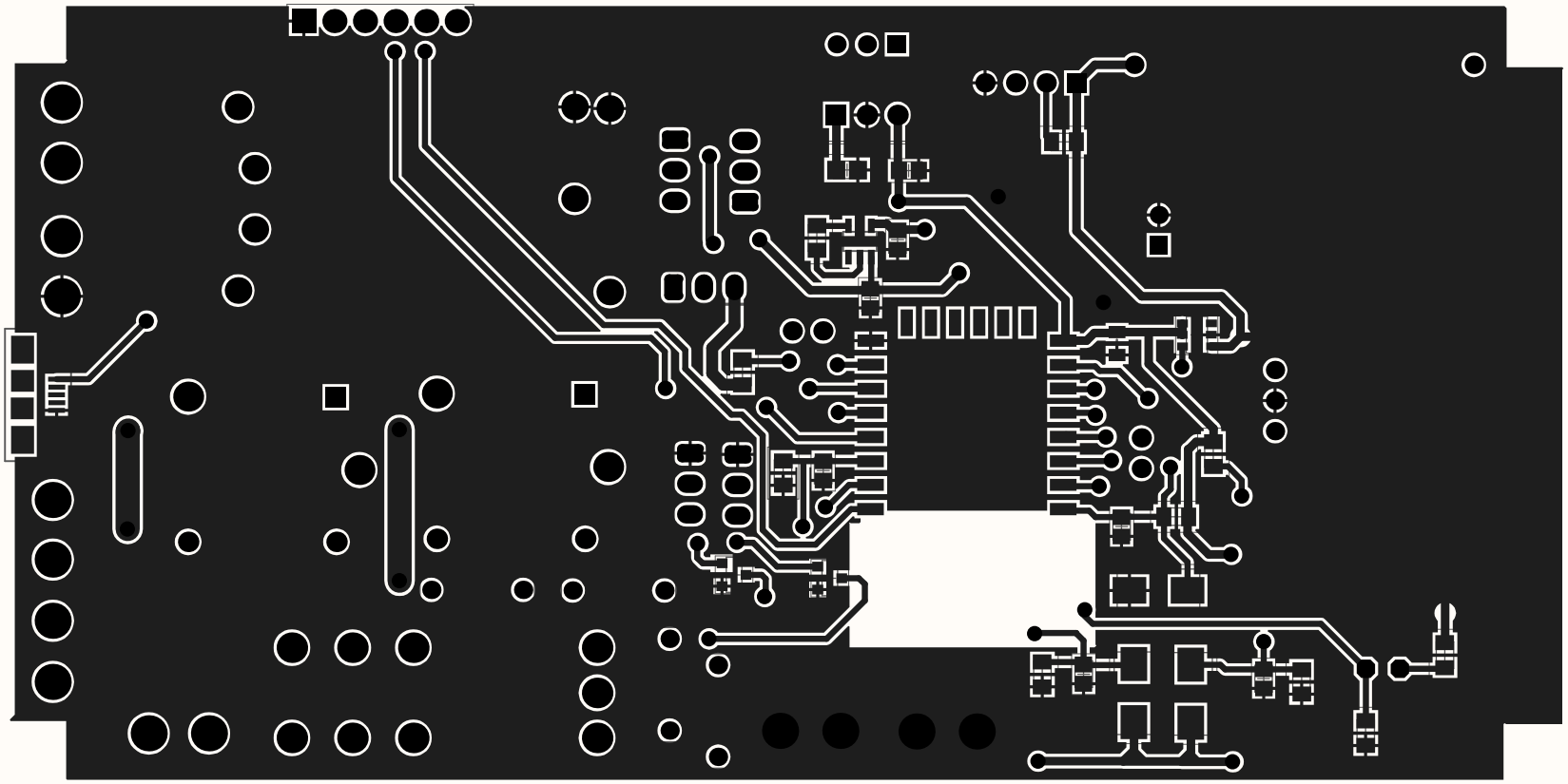
Título: DISEÑO DE UN SISTEMA DOMÓTICO CON COMUNICACIÓN WIFI

ESQUEMÁTICO

Nº de Plano: 1

NIA: 684074

Curso: 4º curso de GIEyA



	Nombre	Fecha	
Dibujado	Ángel Pina	20/5/2017	

Escuela de Ingeniería y Arquitectura
UniversidadZaragoza

Escala: S/E

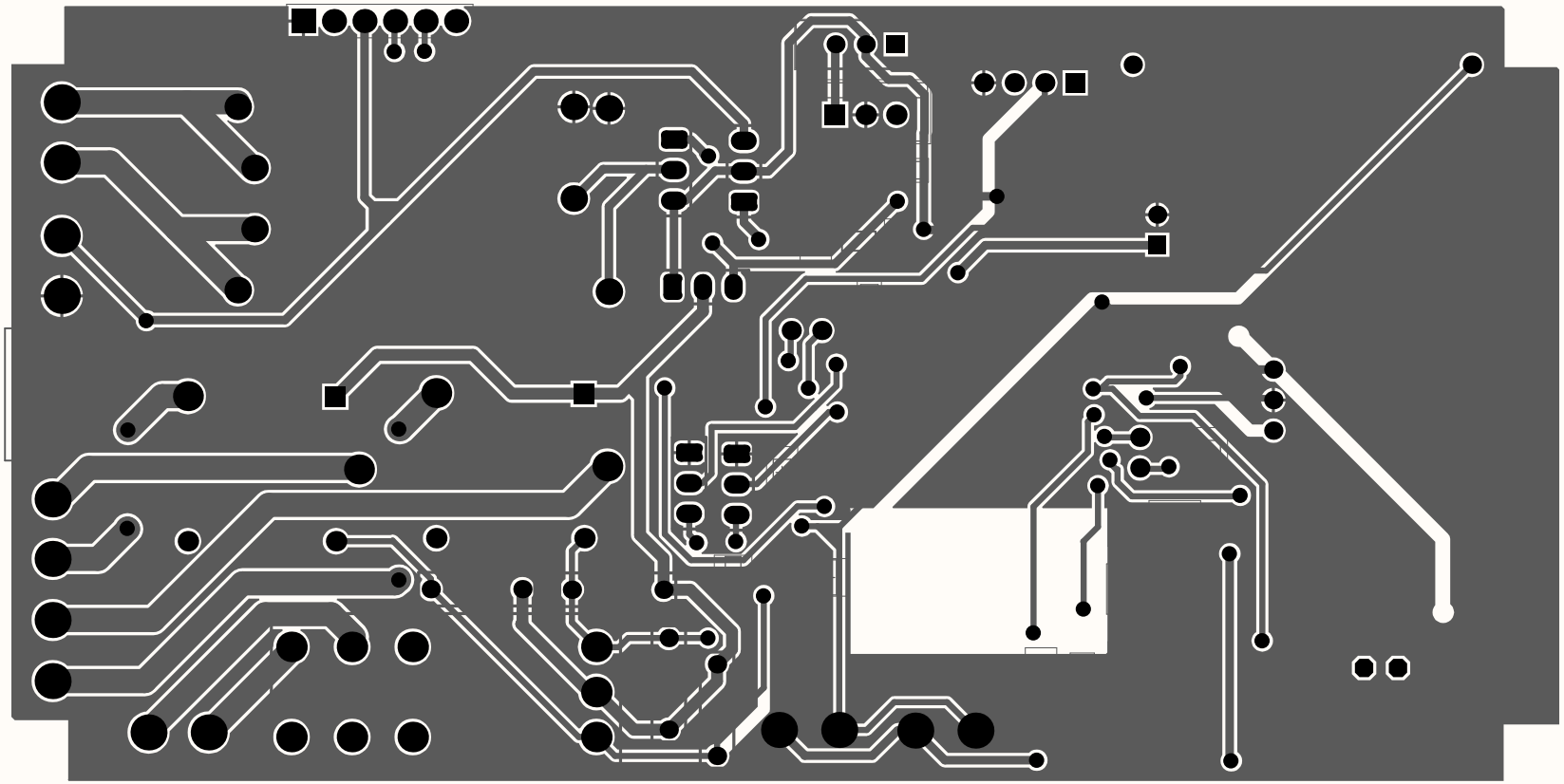
Título:
DISEÑO DE UN SISTEMA DOMÓTICO CON COMUNICACIÓN WIFI

PISTAS CARA TOP

Nº de Plano: 2

NIA: 684074

Curso: 4º curso de GIEyA



	Nombre	Fecha	
Dibujado	Ángel Pina	20/5/2017	

Escuela de Ingeniería y Arquitectura
UniversidadZaragoza

Escala: S/E

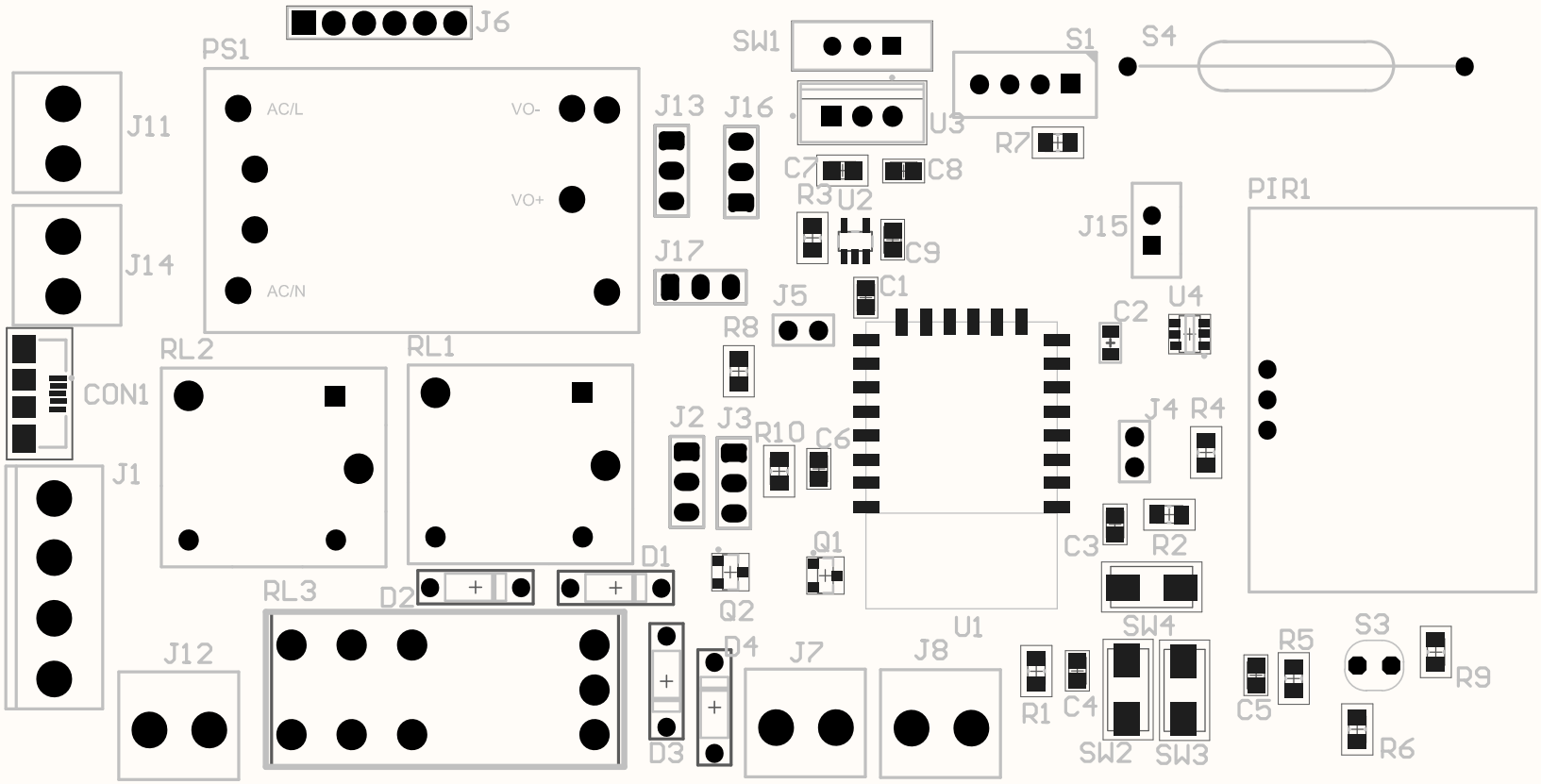
Título:
DISEÑO DE UN SISTEMA DOMÓTICO CON COMUNICACIÓN WIFI

PISTAS CARA BOTTOM

Nº de Plano: 3

NIA: 684074

Curso: 4º curso de GIEyA



	Nombre	Fecha	
Dibujado	Ángel Pina	20/5/2017	

Escuela de Ingeniería y Arquitectura
UniversidadZaragoza

Escala: S/E

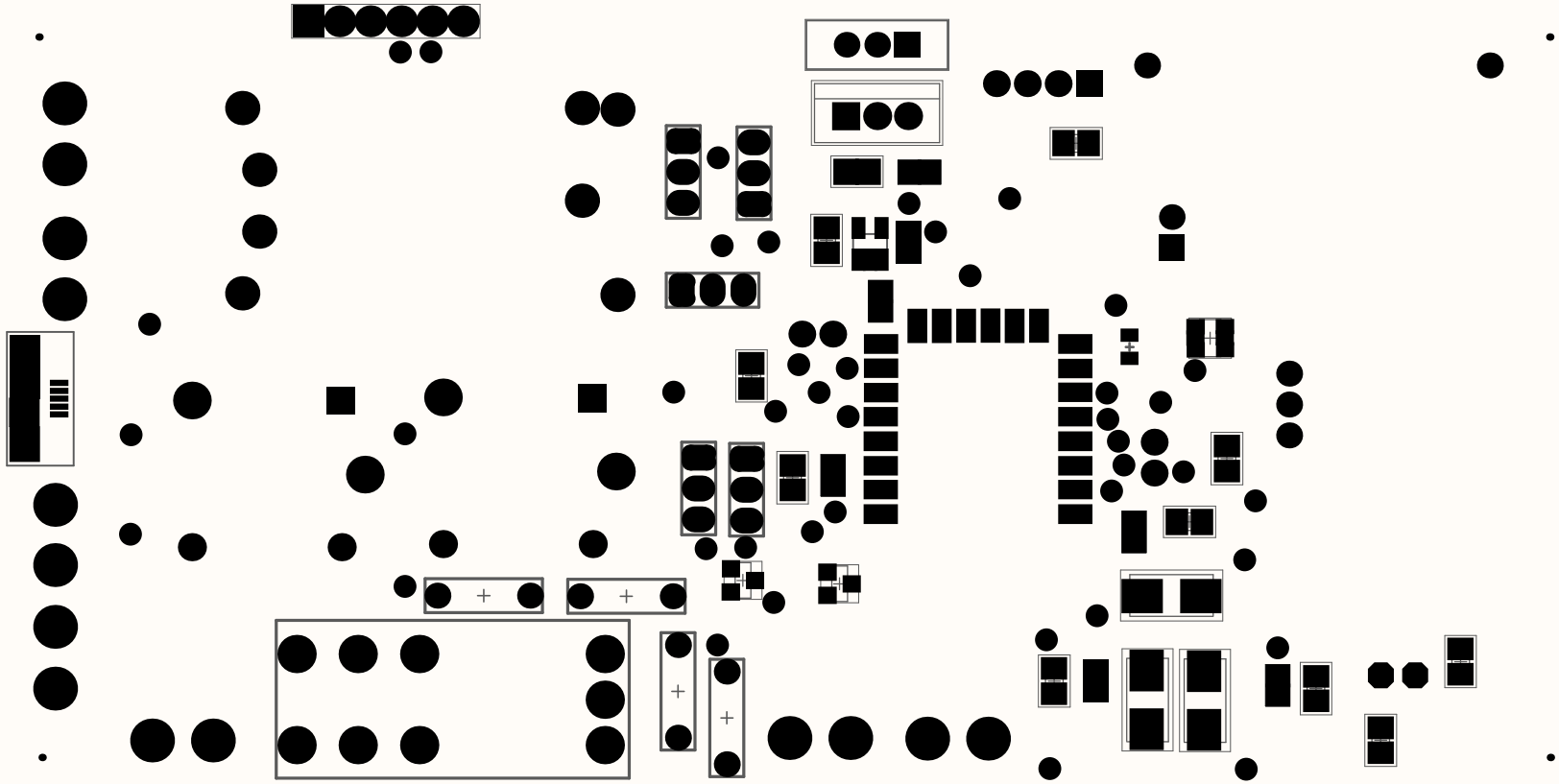
Título:
DISEÑO DE UN SISTEMA DOMÓTICO CON COMUNICACIÓN WIFI

IDENTIFICACIÓN DE COMPONENTES

Nº de Plano: 4

NIA: 684074

Curso: 4º curso de GIEyA



	Nombre	Fecha	
Dibujado	Ángel Pina	20/5/2017	

Escuela de Ingeniería y Arquitectura
UniversidadZaragoza

Escala: S/E

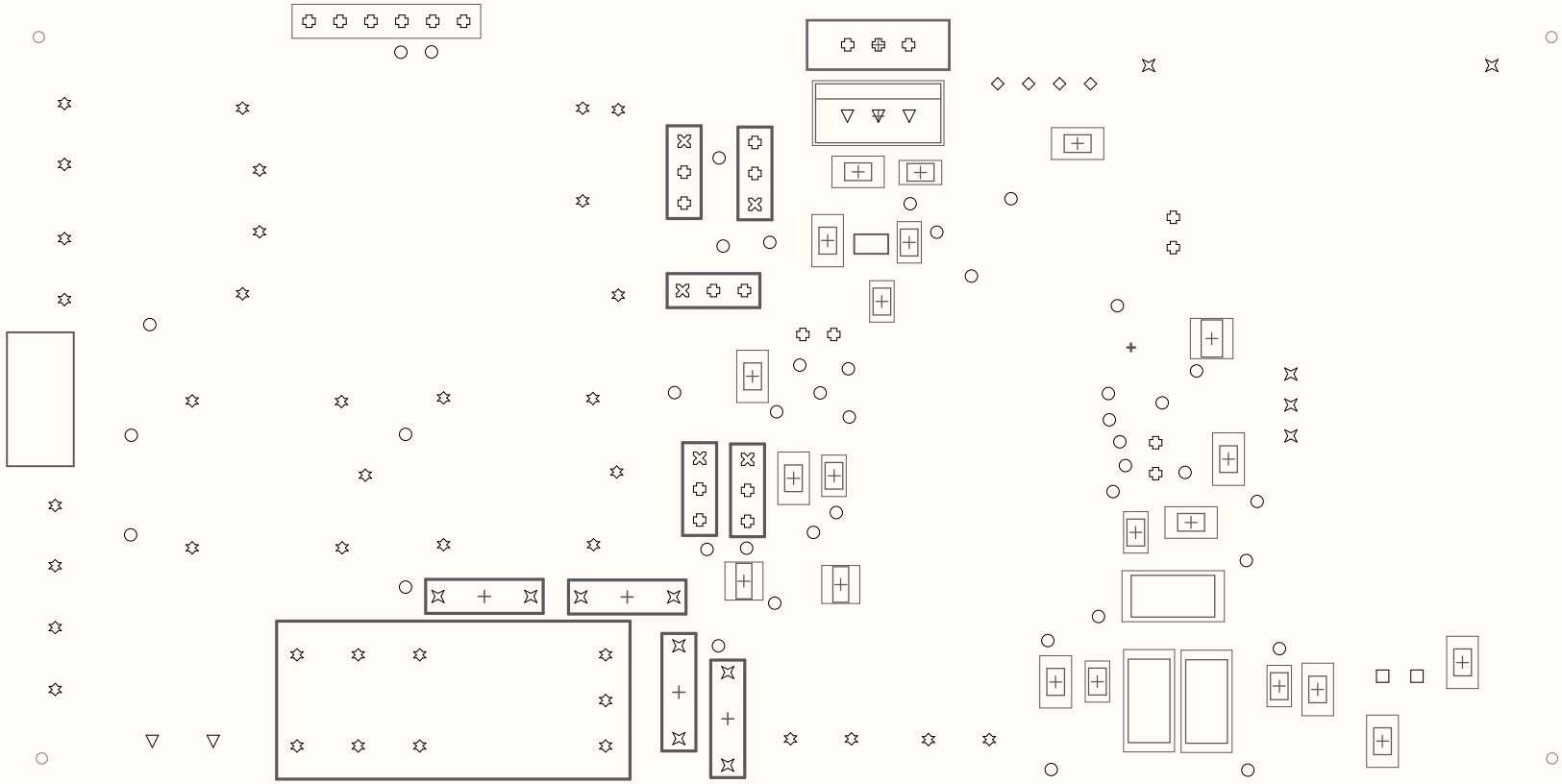
Título:
DISEÑO DE UN SISTEMA DOMÓTICO CON COMUNICACIÓN WIFI

SERIGRAFÍA DE LOS COMPONENTES

Nº de Plano: 5

NIA: 684074

Curso: 4º curso de GIEyA



TOOL	HOLE SIZE	HOLE TYPE
T1	0,7112mm	Round
T2	0,762mm	Round
T3	0,8mm	Round
T4	0,9mm	Round
T5	1mm	Round
T6	1,016mm	Round
T7	1,1mm	Round
T8	1,3mm	Round
T9	3mm	Round

	Nombre	Fecha	
Dibujado	Ángel Pina	20/5/2017	

Escala

S/E

Título

DISEÑO DE UN SISTEMA DOMÓTICO CON COMUNICACIÓN WIFI

PLANO DE TALADRADO

Escuela de Ingeniería y Arquitectura

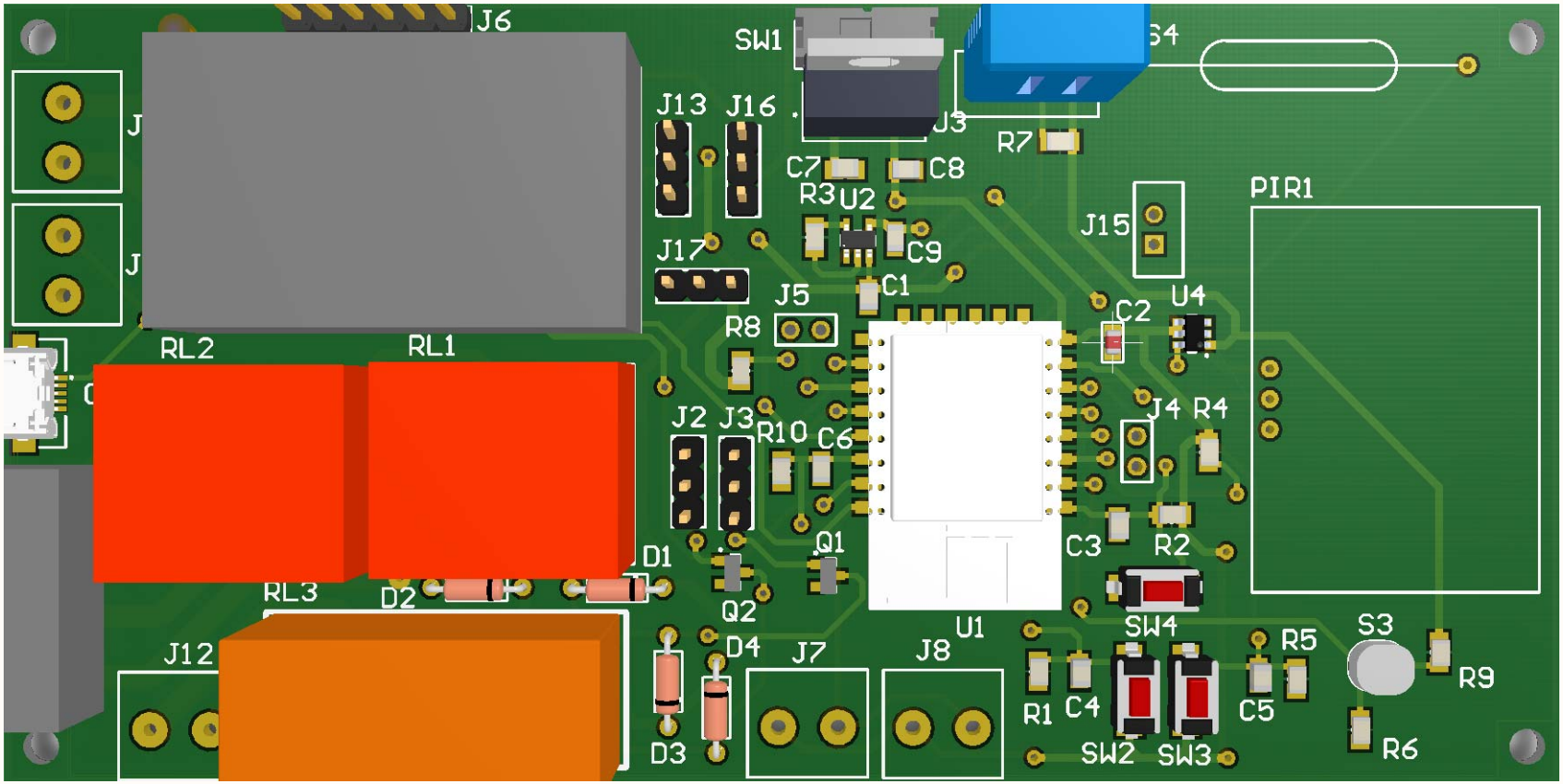
UniversidadZaragoza

1542

Nº de Plano: 6

NIA: 684074

Curso: 4º curso de GIEyA



	Nombre	Fecha		Escuela de Ingeniería y Arquitectura UniversidadZaragoza
Dibujado	Ángel Pina	20/5/2017		
Escala: S/E	Título: DISEÑO DE UN SISTEMA DOMÓTICO CON COMUNICACIÓN WIFI PLANO 3D			Nº de Plano: 7
				NIA: 684074
				Curso: 4º curso de GIEyA

8.4. ANEXO 4: DISPOSICIÓN JUMPERS

8.4.1. MODO ACTUALIZACIÓN CÓDIGO PC-ESP8266

Esta disposición de los jumpers es la empleada para las situaciones en las que queremos transferir el nuevo código desde el Arduino IDE al módulo ESP8266 a través del puerto serie. Los jumpers afectados en este caso son los J2, J3 y J5, conectados con GPIO15, GPIO0 y GPIO2. Para el correcto funcionamiento del micro en el intercambio de información, estos pines deben estar conectados a GND, GND y VCC respectivamente. Tanto al pasar de modo intercambio de datos a modo de funcionamiento habitual de la aplicación, como para introducir de nuevo información por el puerto serie, es necesario utilizar el pulsador de RESET (SW4).

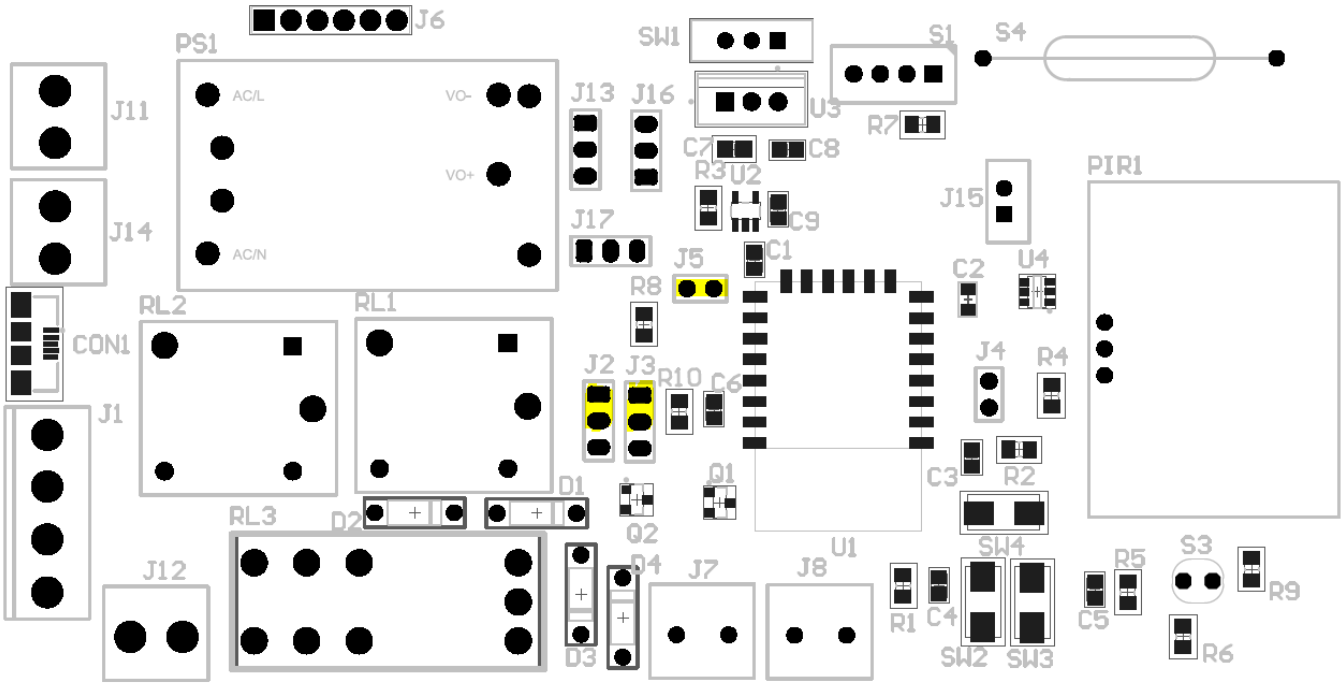


Figura 50: Posición para el intercambio de datos por el puerto serie

8.4.2. MODO HABITUAL DE FUNCIONAMIENTO

En este caso, los jumpers J2 y J3 están dispuestos de forma que están conectados los pines GPIO15 y GPIO0 con los relés.

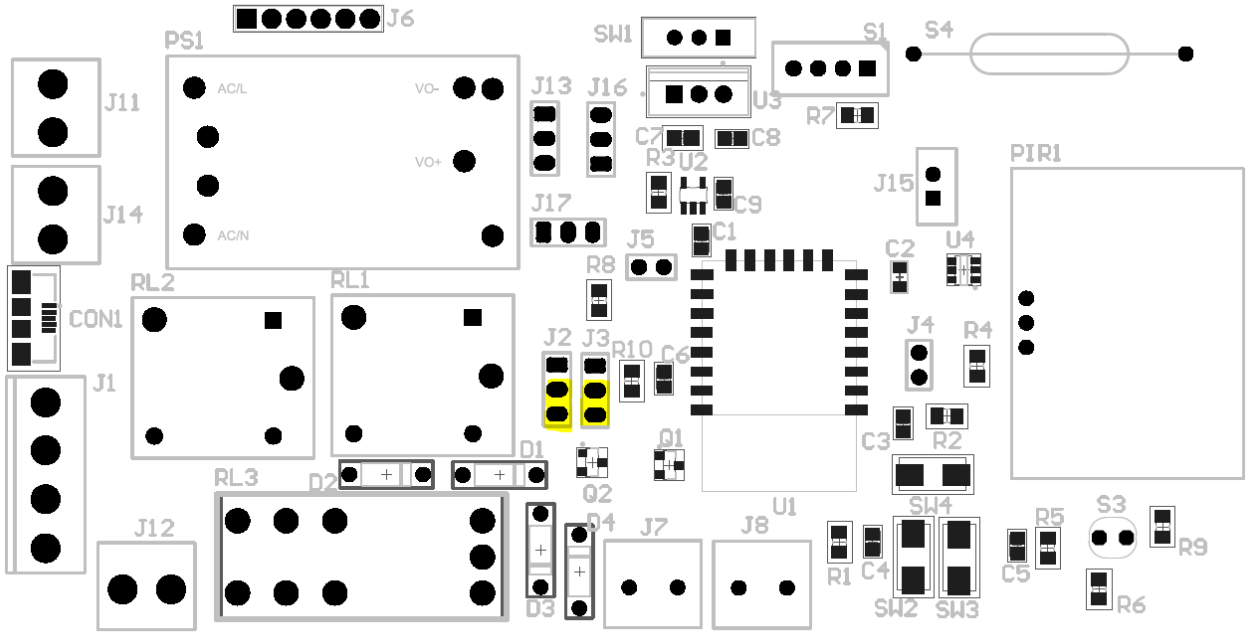


Figura 51: Modo de funcionamiento habitual del producto

8.4.3. DISPOSICIÓN SEGÚN ALIMENTACIÓN UTILIZADA

Si lo que se va a utilizar como fuente alimentación es la fuente DC externa el jumper J16 debe quedar de la siguiente manera:

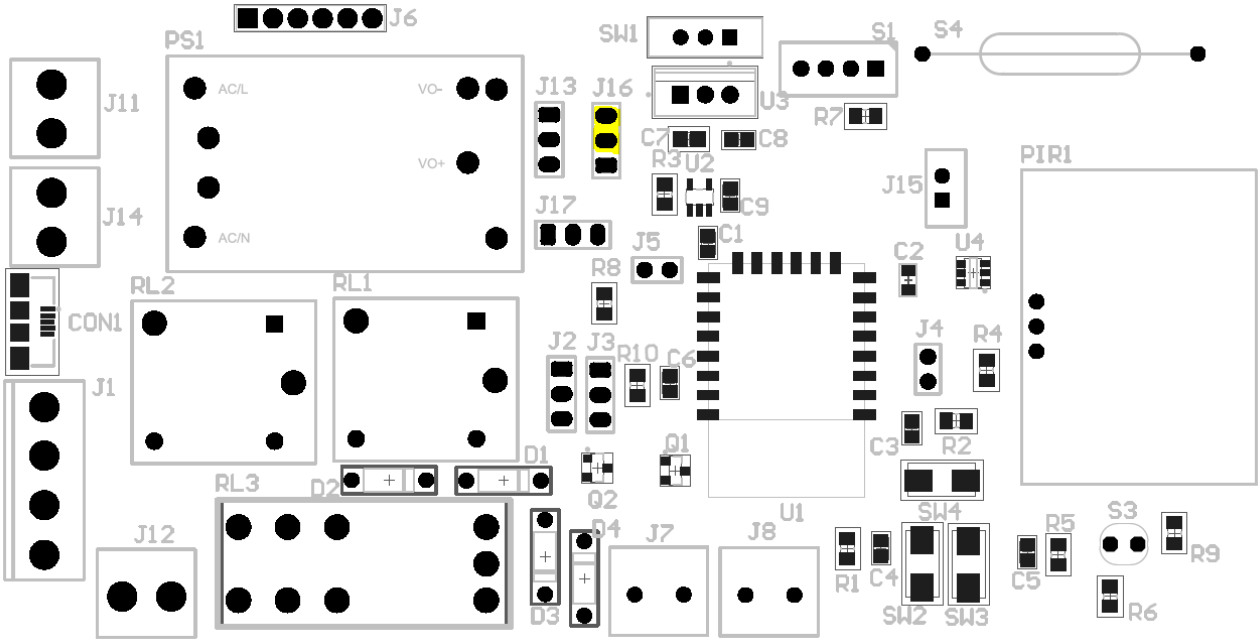


Figura 52: Alimentación mediante fuente dc externa

Mientras que, si la fuente de energía es una batería, el J16 debe ir como se muestra a continuación:

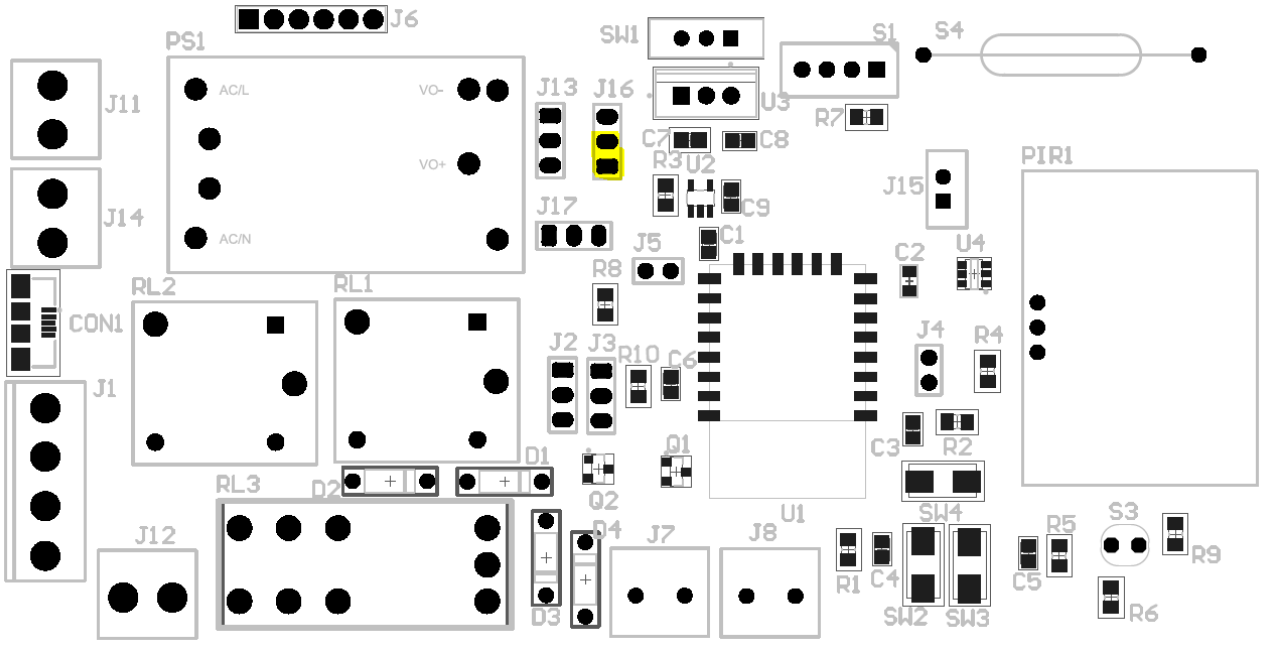


Figura 53: Alimentación por batería

Finalmente, según si hay convertidor AC/DC y relés a 3,3V o 5V ambos, la posición del J13 se muestra en las figuras 43 y 44 respectivamente:

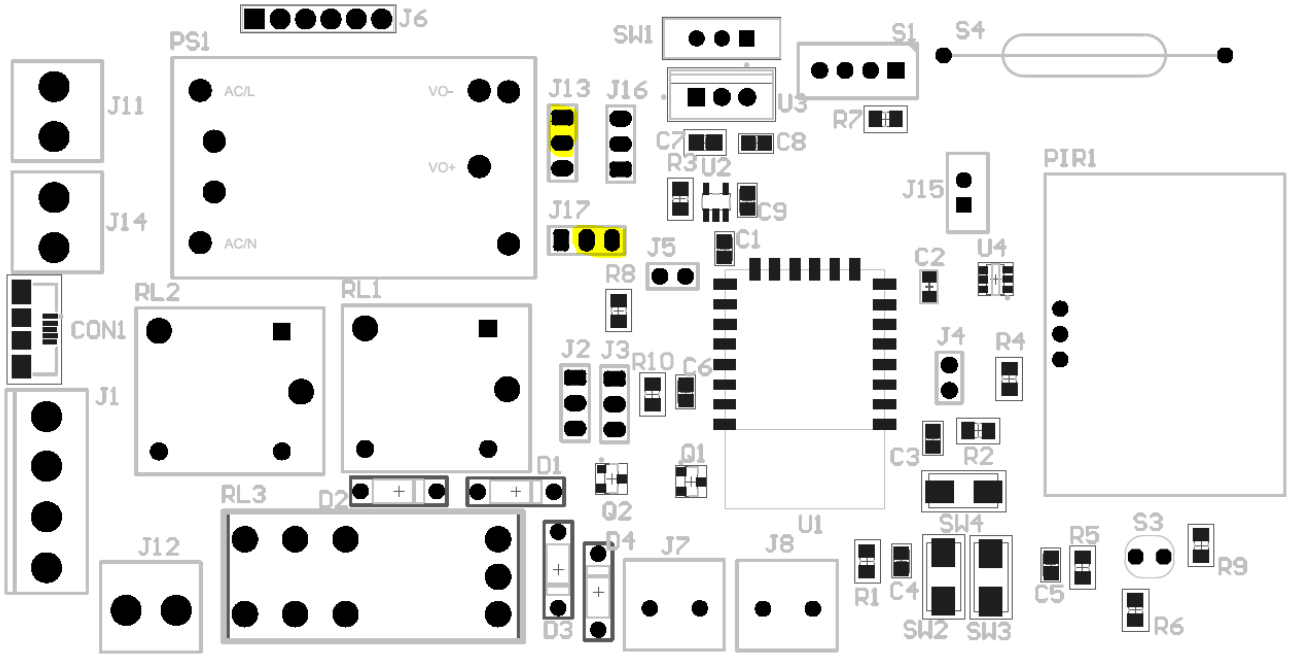


Figura 54: Relés y convertidor ac/dc de 3,3V

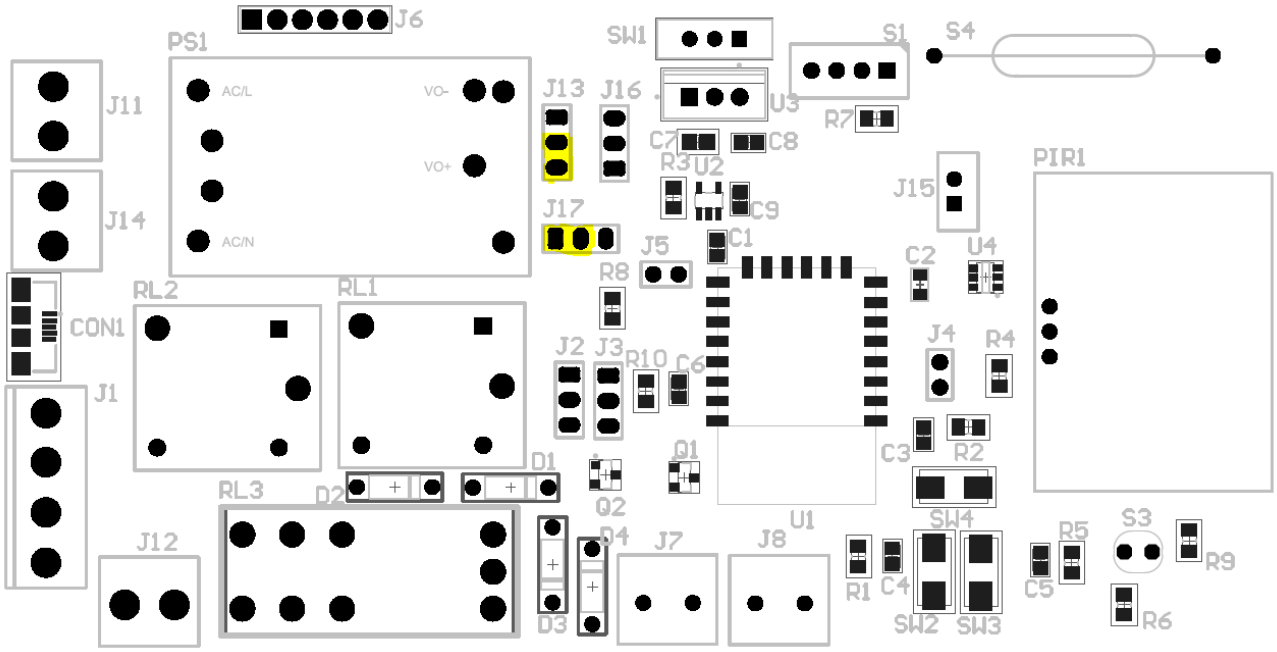


Figura 55: Relés y convertidor ac/dc de 5V