UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Control y monitorización de una vivienda mediante Arduino y Telegram

Juan José Segura Garrido

Tutor: Ignacio Miró Orozco

Universidad Politécnica de Valencia – Campus de Alcoy

Alcoy, 28 de junio de 2016

Resumen

Se ha desarrollado un proyecto que engloba los conceptos de “domótica” y de “Internet de las cosas” mediante el uso del módulo NodeMCU. Se trata de un modelo de microprocesador que incorpora un módulo Wifi ESP8266 y que a pesar de su reducido tamaño incorpora las conexiones necesarias para controlar diferentes elementos de un vivienda. A pesar de la gran variedad de alternativas que existen a este planteamiento, se ha decidido implementar el control de esta placa en una red social; la aplicación de mensajería instantánea Telegram. Se trata de una aplicación instalada en millones de dispositivos y compatible con cualquier sistema operativo, lo que proporciona numerosas opciones para el control de la vivienda.

Contenido

1. Introducción

Con el auge de las redes sociales y de la cada vez más popular domótica, surge la idea de unir dos usos de la tecnología totalmente diferentes para poder realizar el control de una vivienda mediante una red social. Hoy en día, existen muchas alternativas para poder automatizar una vivienda, como la que nos ofrece Philips con sus bombillas inteligentes Hue o Sonoff, una de las marcas más populares de interruptores para domótica debido a su precio.

Todas las alternativas existentes requieren del uso de aplicaciones propias para ser utilizadas desde *smartphones* y muy pocas de ellas pueden ser utilizadas desde ordenadores. Es por ello por lo que la idea de crear una alternativa al control de una vivienda a distancia sin necesidad de instalar aplicaciones extra resulta especialmente atractiva. Con esta idea en mente, se decide utilizar una de las aplicaciones más populares a nivel mundial y que acumula más de 100 millones de usuarios en todo el mundo, una aplicación de mensajería instantánea: Telegram.

Telegram es un servicio de mensajería basado en la nube, por lo que podemos utilizar la misma cuenta en cualquier dispositivo, ya sea un *smartphone*, una tablet o un ordenador, además de contar con compatibilidad con cualquier sistema operativo moderno. A pesar de tratarse de una aplicación de mensajería instantánea como lo es WhatsApp, Telegram ofrece una mayor capacidad de personalización de los chats, pero no de manera estética, ya que hace uso de los denominados Bots, unos chats automáticos que reaccionan a ciertos comandos para realizar diferentes acciones.

Sobre estas bases se centra el proyecto que se ha realizado, donde gracias a un Bot de Telegram se puede controlar una vivienda nos encontremos donde nos encontremos y a pesar de no estar conectados a la misma red WiFi de la placa que se ha creado para este proyecto.

2. Objetivos

- Trabajar con la placa programable NodeMCU con el entorno oficial de Arduino.
- Conseguir la implementación de la aplicación de Telegram en el sketch de Arduino.
- Trabajar con un circuito de relés que puedan soportar la tensión y la corriente de red que se puede encontrar en cualquier hogar.
- Implementar un sensor de corriente para identificar si los elementos de la instalación están activos o no, aunque no se activen mediante el sketch.
- Trabajar con un sensor de temperatura y humedad para poder controlar las condiciones de los espacios de la casa donde se sitúa la placa en cuestión.
- Control inteligente de las persianas de una casa según la situación del sol.
- Creación de un Bot de Telegram para poder controlar el circuito creado a distancia y sin estar conectados a la misma red WiFi.
- Control de la vivienda mediante Google Assistant.

3. Búsqueda de información

Antes de empezar a buscar la información referente a cada uno de los componentes, conviene hacer un planteamiento de lo que se busca realizar para poder escoger lo que más se adapte a las necesidades del proyecto.

El objetivo principal del proyecto es el de poder controlar una vivienda a distancia con una placa que utilice el entorno de programación de Arduino. Para ello, deberemos utilizar conexión WiFi, por lo que la opción más económica es optar por el módulo WiFi ESP8266, aunque uno de sus inconvenientes es que precisa de una placa programable para poder utilizarlo. Se podría utilizar la placa Arduino UNO, pues es la más estandarizada de todas y al ser una placa de desarrollo abierto, hay una gran cantidad de alternativas económicas a la misma.

La combinación de Arduino UNO junto con el módulo ESP8266 es una solución viable, pero se descarta pues lo que se busca es poder realizar un montaje lo más compacto posible, así que las grandes dimensiones del Arduino UNO (comparado con otras placas) y el añadido del módulo WiFi hacen que no encajen en las bases de este proyecto. Por ello, se procede a la búsqueda de una placa programable que cuente con un módulo WiFi incorporado. La solución que más se acerca a lo que buscamos es la placa Arduino MKR1000, pues cuenta con un tamaño reducido y conectividad WiFi, pero su elevado precio en relación con lo que podemos encontrar en el mercado hacen que la creación del proyecto sea demasiado cara y no sea viable para el desarrollo de varias unidades posteriormente, pues su precio en la tienda oficial de Arduino es de 30,99€.

Esto obliga a seguir buscando una alternativa para el proyecto y hay una placa que junta varios de los requisitos que se solicitan: la placa NodeMCU. Se trata de una placa programable mediante el entorno de Arduino y cuenta con un módulo WiFi ESP8266 incorporado. Además, cuenta con un tamaño lo suficientemente reducido como para poder ser instalada sin necesidad de aumentar en exceso el tamaño del proyecto. También cuenta con un número de pines más que suficiente para poder realizar este proyecto y su precio es muy inferior al del Arduino MKR1000, pues se puede comprar por un precio cercano a los 4€ con envío desde España. Si se importa desde China el precio baja hasta los 2,5€ aproximadamente.

Puesto que la finalidad de utilizar una placa de estas características es para poder actuar de alguna forma sobre las diferentes cargas de una instalación doméstica, necesitaremos alguna clase de actuador que se pueda activar de manera digital con un voltaje lo suficientemente bajo como para que pueda ser provisto por la placa NodeMCU. El actuador escogido es el relé, que permitirá utilizarlo como interruptor o como conmutador según el uso que se le vaya a dar al montaje final.

Puesto que con la placa NodeMCU tenemos más posibilidades que las de controlar los relés u otro tipo de actuadores, se ha decidido implementar la capacidad de controlar las condiciones que se presentan en cada una de las habitaciones en las que se instalará el proyecto. Siendo objetivos, lo que más puede interesar conocer del estado de una habitación, es la temperatura de esta, por lo que con un sensor de temperatura sería más que suficiente, pero ya que los sensores de temperatura y humedad para Arduino son realmente económicos, se ha optado por escoger uno que nos permitirá conocer ambas condiciones.

Por otro lado, para las conexiones que se puedan realizar con el relé actuando como conmutador, también se podrán realizar de manera manual con un conmutador físico en dicha habitación, por lo que nos interesa buscar algún modo de conocer si dicho circuito está activo o no, para saber si tenemos que activarlo o desactivarlo a distancia, en caso de que no recordemos

si una luz se ha quedado encendida o algo por el estilo. Para ello, se ha buscado un sensor de corriente que permitiera detectar cuando está circulando corriente a través de un circuito y, por tanto, poder detectar si está encendido o apagado a pesar de que el estado del pin asignado a dicho relé esté en el estado opuesto.

3.1 NodeMCU

Cuando se habla de NodeMCU no se hace otra cosa más que hablar sobre a una placa programable de desarrollo *open source*, al igual que la placa Arduino UNO estándar. Esta placa programable, permite programar un microcontrolador, apodado como MCU (Microcontroller Unit), de ahí el nombre de la propia placa. Las ventajas de utilizar esta placa radican en su similitud con las placas Arduino, por lo que se podrá utilizar el mismo entorno para programarlas, es decir, el programa oficial de Arduino, aunque cualquier otra alternativa será igual de válida utilizando las librerías correspondientes.

¿Cuál es la ventaja de utilizar el NodeMCU a cualquiera de las alternativas de Arduino? Principalmente el precio, puesto que es muy económico y cercano a los 3$ por lo que realizar proyectos útiles con esta placa permitirá mantener un precio ajustado. En segundo lugar, la placa NodeMCU cuenta con un módulo WiFi incorporado, concretamente el ESP8266, que permite que se conecte a cualquier red WiFi de 2,4 GHz.

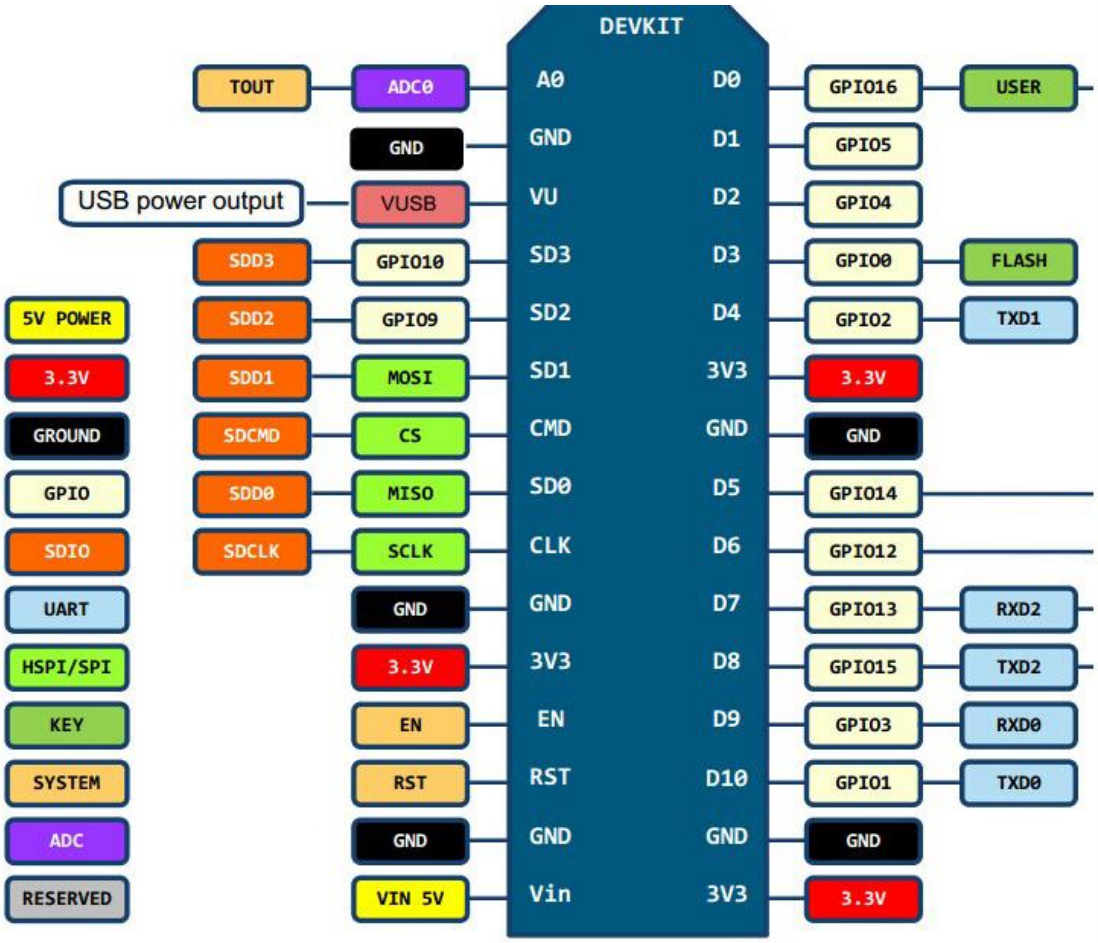


Imagen 3.1.1

Otro de los factores por los que se ha escogido la placa NodeMCU es su tamaño, pues es mucho más reducido que el de otras placas con funciones similares e incluso que la placa Arduino más popular, Arduino UNO junto al módulo ESP8266 para dotarla de conexión WiFi. Gracias al tamaño de la placa NodeMCU, el diseño de la placa final del circuito puede quedar muy compacto para una instalación de este más sencilla.

El NodeMCU cuenta con 30 pines de conexión en total distribuidos en dos filas de 15. Para este proyecto, solo serán útiles 19 de estos pines, donde 3 de ellos están destinados a la alimentación de 3,3 V que es a la que trabaja el NodeMCU y el módulo ESP8266, otros 5 pines están destinados a la conexión a tierra, 9 de los restantes son las entradas y salidas digitales y los dos restantes corresponden a un pin de alimentación a 5 V y a una entrada analógica. La distribución de los pines se puede observar en la *Imagen 3.1.1*.

La conexión entre el NodeMCU y el ordenador desde el que la programaremos se hará mediante un puerto MicroUSB y una velocidad de baudio de 9600 bps, tal y como nos indica la propia placa en su parte trasera.

3.2 Librería de Telegram para Arduino

Si lo que se busca es trabajar en una comunicación entre Telegram y el entorno Arduino, se deberán instalar las librerías necesarias para que esto funcione de manera correcta. Supuestamente la librería de Telegram solo funciona con la placa Arduino MKR1000, pero la verdad es que funciona con cualquier otra placa que se programe desde el entorno Arduino y que disponga de conexión WiFi, por lo que la placa NodeMCU podrá trabajar con dicha librería.

Concretamente, el nombre de la librería es *Telegram BOT Library*, es decir, que no se trabajará directamente con la aplicación de Telegram, puesto que para ello se debería tener una cuenta propia para la placa en cuestión, por lo que la manera más sencilla y viable es mediante un Bot de esta red social.

Lo primero es dejar claro que es un Bot en Telegram. Se tratan de cuentas que trabajan mediante software y de manera automática, a diferencia de las cuentas de los usuarios, que son los propios usuarios los que trabajan con ellas. Con estos Bots, se pueden crear infinidad de características para la propia aplicación, como búsqueda automática de imágenes en internet, crear encuestas o lo que crearemos gracias a al sketch de este proyecto, el control de una vivienda. Para realizar esto, una de las primeras cosas que se aprecia es que la definición de Bot la desvirtuamos levemente, pues no solo será controlado por software, ya que el propio hardware de la placa creada ayudará a su control.

Dicha librería se puede descargar directamente desde GitHub y gracias a ella se pueden leer y contestar los mensajes dentro de una conversación con el Bot, sea desde el sistema operativo que sea, pues Telegram es un servicio de mensajería instantánea multiplataforma y basado en la nube, por lo que se mantiene actualizado en todos los dispositivos en los que se tiene instalado. Para ello, se deberá introducir el *Bot Token* en el sketch del proyecto para darle permiso a entrar en la conversación. El *Bot Token* no es más que una firma alfanumérica única que tiene cada Bot y que solo puede ver el creador de este.

3.3 Sensor de temperatura y humedad DHT22

Para poder analizar las condiciones de temperatura y humedad de cada habitación en la que se instalase el circuito final, se ha utilizado el sensor de temperatura y humedad DHT22 que hace que este proceso sea muy sencillo, pues en el propio sensor está incluido un pequeño procesador que realiza la medición y que es trasmitida al NodeMCU mediante un pin digital. En el caso de este proyecto, se ha escogido el DHT22, un sensor algo más caro que el DHT11 pero que proporciona una mayor precisión tanto en temperatura y humedad, además de una mayor frecuencia de muestreo.

	DHT11	DHT22
Temperatura	0 – 50ºC, Precisión de ±2ºC	-40 – 80ºC, Precisión de ±0,5ºC
Humedad	20 – 80%, Precisión de ±5%	0 – 100%, Precisión ±2%
Frecuencia de muestreo	1 Hz (1 vez por segundo)	2 Hz (2 veces por segundo)
Color	Azul	Blanco

A pesar de no tratarse de un sensor de alta precisión, el DHT22 es más que suficiente para el control de la temperatura y la humedad de una vivienda y las diferencias respecto al DHT11 son lo suficientemente grandes como para que valga la pena gastarse un poco más de dinero.

Este sensor debe estar conectado a tierra y a alimentación (3,3 V – 6 V), además de a un pin digital. En la unidad utilizada, el propio sensor estaba conectado a una pequeña placa donde se aprecia una resistencia, conectada entre la alimentación y la salida del sensor. La conexión con el NodeMCU se puede apreciar en la *Imagen 3.3.1*.

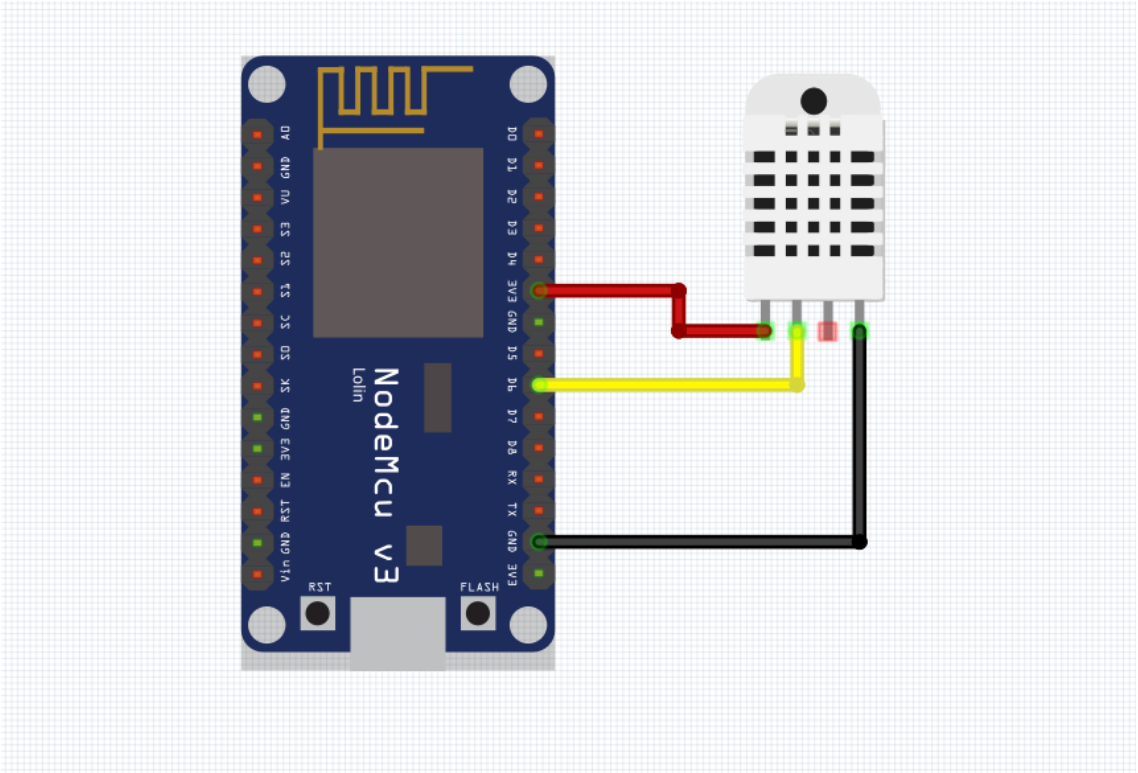


Imagen 3.3.1

3.4 Sensor de corriente ACS712

Puesto que a pesar de que el objetivo final de este proyecto es el de poder controlar una vivienda a distancia mediante Internet, la posibilidad de interactuar con los diferentes elementos de la instalación de forma manual no se suprime, pues en muchas ocasiones es más cómodo activar un interruptor manualmente que hacerlo desde el teléfono, por ejemplo. Por ello, se ha decidido implementar un sensor de corriente para poder saber cuándo hay algo activado en la instalación. En concreto, se ha utilizado el sensor de corriente ACS712.

El ACS712 es un sensor de efecto Hall, el cual detecta el campo magnético que se produce por inducción de la corriente que circula por la línea que atraviesa las borneras del propio sensor. A diferencia del sensor DHT22 de temperatura y humedad, el pin de salida del ACS712 va conectado al pin analógico del NodeMCU. Los otros pines del ACS712 son el de tierra y el de alimentación. En el caso de este proyecto, como el NodeMCU trabaja a 3,3 V deberemos conectarlo a esta tensión para no provocar errores en las lecturas. En la *Imagen 3.4.1* podemos apreciar su conexión al NodeMCU.

Dentro del sensor ACS712 se pueden encontrar diferentes variantes, según la corriente máxima que podamos medir, habiendo un sensor para tensiones de hasta 5 A, otro de hasta 20 A y un último para corrientes de hasta 30 A. Estos tres sensores se pueden diferenciar por su nombre completo, ACS712-05A, ACS712-20A y ACS712-30A. En este proyecto, como se va a trabajar con elementos domésticos, nos servirá el sensor de 5 A, pues sobre todo lo utilizaremos para saber si las luces están o no activas y la corriente de la iluminación de una vivienda es muy inferior a esos 5 A.

Modelo	Rango	Sensibilidad
ACS712-05A	-5 – 5 A	185 mV/A
ACS712-20A	-20 – 20 A	100 mV/A
ACS712-30A	-30 – 30 A	66 mV/A

A pesar de que el propio fabricante del sensor indique que la sensibilidad de la unidad es de 185 mV/A, es recomendable calcularla manualmente. Se sabe que el sensor entrega un valor de 2.5 V cuando la corriente que atraviesa las bornas es de 0A. La limitación de utilizar el sensor de corriente ACS712 junto con el NodeMCU es que el sensor requiere de una conexión a un pin analógico y precisamente el NodeMCU solo cuenta con un pin de estas características, por lo que no se puede utilizar más de un sensor de corriente por cada placa NodeMCU que se utilice, por lo que no se puede controlar la corriente que pasa por todos los circuitos de una misma habitación con una sola placa.

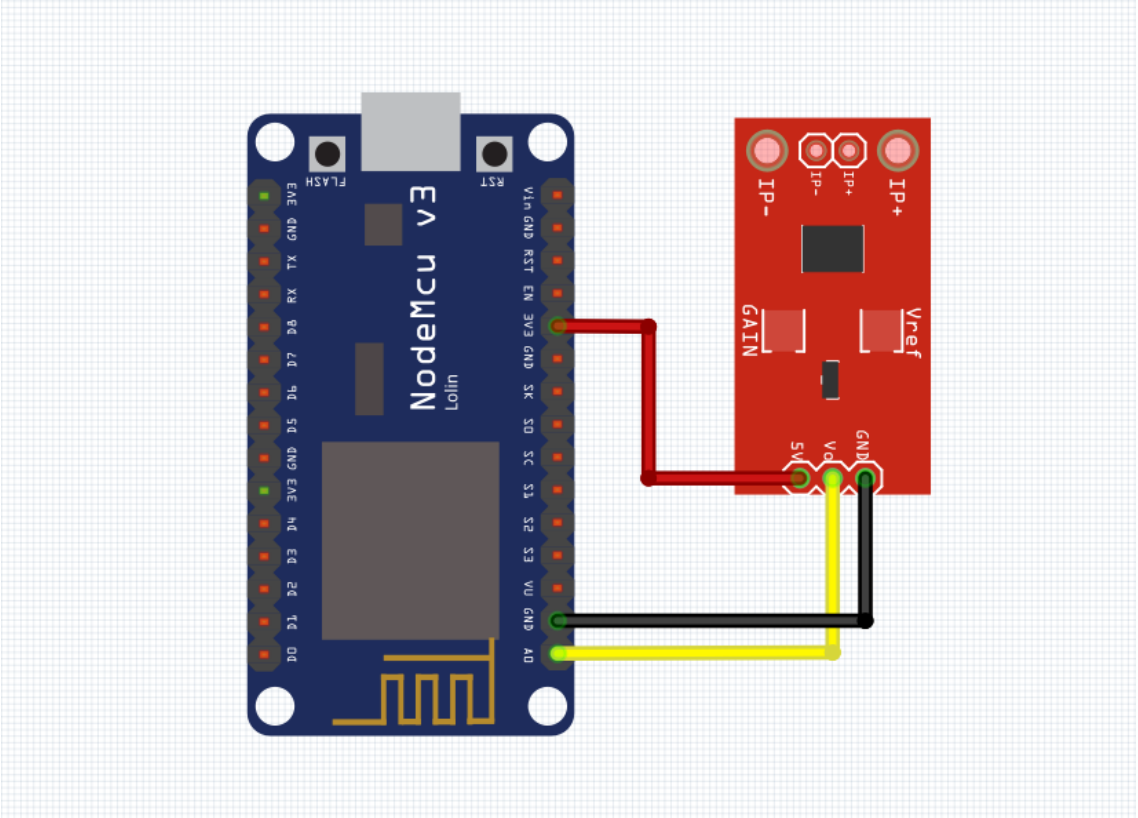


Imagen 3.4.1

3.5 Relés

Puesto que lo que se busca es la activación y desactivación de ciertos elementos conectados a la red eléctrica de una casa (por ejemplo, la iluminación de la cocina), se deberá utilizar un elemento que soporte esas tensiones, que pueda actuar como interruptor o como conmutador y que a su vez pueda ser activado mediante el NodeMCU, por lo que su tensión de activación no debe superar los 5 V. Todos estos requisitos los cumple un único componente, el relé.

Puesto que el NodeMCU tiene una tensión máxima de alimentación de 5 V y su tensión de trabajo es de 3,3 V, se deberá trabajar con relés de 5 V o de 3 V, siendo más sencillos de encontrar y más económicos los primeros. Por supuesto, este no es el único requisito que deben cumplir, pues deben soportar como mínimo los 230 V que es la tensión de red en España y 5 A que es la corriente máxima que soporta el sensor de corriente.

En este proyecto, se han utilizado relés de 5 V que aguantan tensiones de hasta 250 V en corriente alterna y 10 A, mientras que en corriente continua se quedan en 30 V y también 10 A. Como el uso que se va a hacer de ellos es en corriente alterna, se podrán utilizar sin problema. Puesto que los relés se suelen vender montados en una placa con conectores para Arduino, su tamaño se ve bastante incrementado y en este proyecto, donde una de las intenciones es que el tamaño sea reducido, no encajaba del todo con las bases de este. Por ello, se ha decidido replicar el circuito que encontramos en dichas placas para montarlo por separado de manera compacta y así poder reducir el tamaño a la mínima expresión. El circuito se puede apreciar en la *Imagen 3.5.1*.

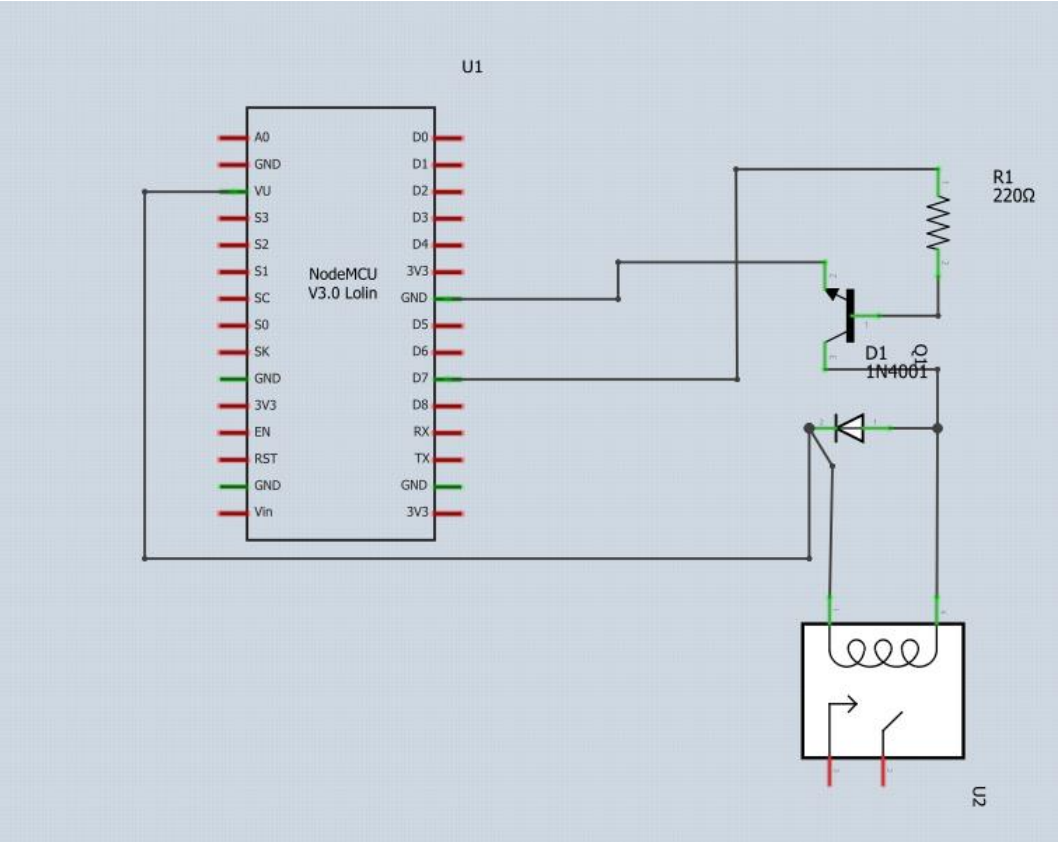


Imagen 3.5.1

4. Desarrollo del proyecto

En este apartado de la memoria, se detalla paso a paso el trabajo realizado, desde la instalación de todas las librerías necesarias en el entorno de programación de Arduino, hasta la implementación de los diferentes componentes hasta llegar al desarrollo final de una placa de prototipo donde se puede apreciar de manera definitiva el trabajo realizado, además de poder probarse en un entorno real.

4.1 Desarrollo del programa

4.1.1 Implementación de todas las librerías necesarias para el proyecto

Para poder hacer uso de la placa NodeMCU, es necesario instalar las librerías adecuadas, sobre todo por el uso del módulo WiFi ESP8266 que necesita una librería en concreto para poder trabajar de manera adecuada. Si hay algo bueno de programar con el entorno de Arduino, es la gran comunidad que hay detrás de esta placa, lo que facilita la solución de problemas que surgen. La primera librería con la que trabajaremos será la ESP8266WiFi que se puede encontrar en el repositorio de código GitHub, como la mayoría de las librerías utilizadas para este proyecto.

Puesto que este proyecto se basa en la implementación de la red social de mensajería instantánea Telegram en el entorno Arduino, se precisa de una librería que permita acceder al Bot de dicha aplicación para poder interactuar con el usuario una vez se desarrolle el programa. De nuevo, se recurre a GitHub donde el usuario Gianbacchio ha adaptado la librería original para poder utilizarse con nuestro módulo WiFi, el ESP8266. El funcionamiento de esta librería es sencillo, aunque está muy limitado a las variables que utilicemos en nuestro programa.

Para que el sensor de temperatura y humedad pueda implementarse en el programa, se precisan dos librerías diferentes. La primera, DHT-sensor-library, de la compañía Adafruit Indsutries, que se encuentra en GitHub, pero que da errores constantes al compilar el código, puesto que precisa la librería propia de la empresa. Dicha librería es Adafruit_Sensor y también la proporciona la misma compañía. Ambas librerías son las mismas que se deberían de usar en caso de utilizar el sensor DHT11 en vez del DHT22 que es el utilizado en nuestro proyecto.

A diferencia del resto de componentes, el sensor de corriente ACS712 no precisa de ninguna librería, pues la señal que le envía al NodeMCU ya ha sido procesada por el propio sensor y son solo valores numéricos. Para los relés tampoco es necesaria la inclusión de ninguna librería, son cargas que se activan cuando están conectadas a un pin digital activo del NodeMCU.

4.1.2 Creación del Bot de Telegram

La creación de un Bot de Telegram para un uso básico que es el que se utilizará con este proyecto, no requiere de conocimientos de programación, pues los propios desarrolladores de Telegram han creado una herramienta para que cualquier usuario pueda crear su propio Bot, algo necesario en el fin de este proyecto, pues su finalidad es que cualquier usuario con conocimientos básicos de electrónica y programación pueda automatizar una vivienda. La

herramienta que se utilizará se llama BotFather y se trata de un Bot de creación de nuevos Bots y como hemos comentado anteriormente, ha sido creado por la propia Telegram.

Para poder hacer uso de este Bot, se deberá descargar la aplicación de Telegram o bien para smartphones o bien para ordenador, donde buscando su nombre completo: @BotFather, se podrá acceder al chat de creación de Bots. Una vez lo localizado e iniciado, el propio Bot dará las indicaciones para empezar a trabajar, además de proporcionar un manual sobre cómo trabajan los Bots en la aplicación.

Para la creación del Bot, se deberá escribir el comando /newbot para a continuación darle un nombre al mismo y crear una pequeña descripción que aparecerá en la pantalla principal del Bot la primera vez que vayamos a iniciarlo. Para el uso que se le va a dar al Bot, no se necesita hacer nada más de momento con @BotFather, aunque cuando se vaya a implementar con el sketch del proyecto, se volverá a trabajar tanto con el Bot creado como con el BotFather.

4.1.3 Conexión a la red WiFi con NodeMCU

Para conectar el NodeMCU a una red WiFi, la librería del ESP8266 lo pone realmente fácil. Primero se tendrán que guardar dos variables de texto, una para el nombre de la red WiFi y otra para la contraseña de esa misma red.

```
const char* ssid     = "Nombre de la red";
const char* password = "Contraseña de la red";
```

Posteriormente, en el *void setup* se establecerá conexión mediante el siguiente comando:

```
WiFi.begin(ssid, password);
```

Con estos dos pasos ya se tendrá el NodeMCU conectado a cualquier red WiFi. Si la red está muy saturada o demasiado lejos de la placa, en el puerto serial se puede ver como aparece una larga lista de errores. Acercando más el NodeMCU a la fuente de la señal o pausando los procesos que más ancho de banda consuman (descargas o reproducción de vídeos), se soluciona. Estos errores solo aparecen mientras se conecta, una vez está conectado no influye la saturación de la red.

4.1.4 Implementación del Bot con NodeMCU

Gracias a la librería que se ha instalado anteriormente, poder implementar el Bot en el sketch se realiza mediante un solo comando. Este comando debe introducirse antes del *void setup* del sketch y básicamente se trata de asignar el Bot Token, una firma digital que tiene cada uno de los Bots creados y que es diferente a la de cualquier otro Bot y almacenarla en como *const char**, la cual la almacena como texto para que quede asignada una vez iniciemos el Bot.

Por tanto, para iniciar la conexión con el Bot de Telegram se escribirá el siguiente código:

```
const char BotToken[] = "Bot Token";
WiFiClientSecure net_ssl;
TelegramBot bot (BotToken, net_ssl);
```

Para obtener el Bot Token, se debe volver a hacer uso del chat iniciado con BotFather, donde una vez creado nuestro Bot, se puede escoger la opción API Token, la cual proporcionará el Bot Token que se debe de introducir en ese fragmento de código. De esta forma, ya se puede interactuar con el Bot creado, tanto leyendo los mensajes que le enviemos desde cualquier dispositivo como enviando mensajes de contestación. Serán los mensajes que se le envíen los que desencadenarán las diferentes acciones que permitirán controlar elementos de la instalación eléctrica.

Los comandos que se van a estar utilizando referentes a Telegram, van a ser solo tres:

```
message m = bot.getUpdates();
```

Este comando permite leer los nuevos mensajes que vaya recibiendo el Bot.

```
m.text.equals("Mensaje que se le envíe")
```

Con este comando, se puede comparar el mensaje recibido por el Bot con el que queramos para usar como desencadenante de alguna acción, por lo que se utilizará como condición dentro de un *if*.

```
bot.sendMessage(m.chat_id, "Mensaje de contestación");
```

Este comando permite programar la respuesta del Bot ante ciertas situaciones. La contestación está limitada al texto escrito entre comillas o a una variable de texto. En nuestro sketch, utilizamos variables *const char** y mensajes concretos. Solo con estos tres comandos se puede interactuar con la placa NodeMCU para que realice diferentes tareas según el mensaje que se le envíe. En un principio, los mensajes habría que escribirlos manualmente, por lo que es

recomendable que sean muy sencillos, peor una vez finalizado el proyecto, se puede mejorar el Bot para que no sea necesario escribir y se pueda realizar mediante botones.

La base del proyecto se basa en la lectura de diferentes mensajes enviados por el usuario final, los cuales, con una cadena de condiciones, permitirán conocer el estado de las conexiones, la temperatura y la humedad de cada habitación y por supuesto, la activación o la desactivación de los diferentes elementos conectados a la red del hogar. En el caso de crear varias placas para diferentes ubicaciones de la vivienda, cada una respondería a ciertos mensajes específicos, pudiendo diferenciar cada una de las habitaciones por su nombre (cocina, comedor, dormitorio…).

4.1.5 Sensor DHT22

El sensor DHT22 envía la salida del propio sensor mediante un pin digital, pero se necesita la librería que se ha instalado anteriormente para poder leer los datos. Para ello, lo primero que se tiene que hacer es indicarle el tipo de sensor con el que se va a trabajar, puesto que la librería sirve tanto para el DHT11 como para el DHT22.

```
#define tipo_de_sensor DHT22
```

Una vez se ha definido una variable con el nombre del sensor en cuestión, se hace uso de su librería, indicando en que pin está conectado y el tipo de sensor que estamos utilizando.

```
DHT dht(sensor_humedad, tipo_de_sensor);
```

Para poder leer tanto la humedad como la temperatura que mide el sensor, se debe utilizar el código que nos proporciona la librería que se ha instalado previamente, asignando los valores a una variable *float*, que permite definir un número con decimales.

```
float h = dht.readHumidity();
float t = dht.readTemperature();
```

Posteriormente, esos números con coma flotante se transforman a caracteres únicamente, para poder ser enviados a través de Telegram, pues una de las limitaciones de la librería de Telegram utilizada es que no se pueden enviar variables numéricas.

4.1.6 Sensor ACS712

El sensor de corriente ACS712 es capaz de medir tanto en corriente continua como en corriente alterna, pero en este proyecto, como se va a estar midiendo la corriente que atraviesa ciertos circuitos en la instalación de una vivienda, vamos a estar midiendo corriente alterna, lo que provoca que el código para que esto sea así sea un poco más complejo. Para ello, primero se ha trabajado con el sensor para corriente continua, para familiarizarse con el código y así poder implementarlo de manera más sencilla en el código final.

En primer lugar, lo que se debe de hacer es definir la sensibilidad del sensor de corriente. Se ha asignado a una variable *float* porque se trata de un número con coma flotante. Las características que proporciona el fabricante del sensor aseguran que su sensibilidad es de 185 mV/A, que en el programa se ha convertido a V/A:

```
float Sensibilidad= 0.185;
```

Tras esto, dentro del *void loop ()* se ha de implementar dos ecuaciones, tanto para la obtención del voltaje que mide el sensor como la de la obtención de la corriente del conductor. Puesto que se trata de un sensor para Arduino cuya lectura se hace a través del puerto analógico de la placa, los valores medidos por el sensor tienen un rango entre 0 y 1023. Para el caso del sensor ACS712 no es diferente, por lo que se tiene que convertir esa lectura a un valor con el que podamos trabajar. Sabiendo que el sensor con el que estamos trabajando es de 5 A, este valor se corresponde a la máxima medida que obtendremos por el sensor, representada por el número 1023. Por ello, si se divide el valor máximo medido por el sensor entre el valor máximo que este ofrece y se multiplica por la medición, se obtiene el voltaje que está midiendo:

```
float voltajeSensor= analogRead(A0)*(5.0 / 1023.0);
```

Con esta pequeña línea de código, se asigna el valor medido del voltaje a una variable de número con coma flotante para poder almacenar los números decimales. El sensor ACS712 ofrece una tensión de 2,5 V para una corriente de 0 A, por lo que se deberá convertir la tensión medida a corriente habiendo corregido previamente dicha característica. Para ello, se hace uso de la variable de la sensibilidad que hemos creado anteriormente:

```
float I=(voltajeSensor-2.50)/Sensibilidad;
```

Se vuelve a asignar el resultado de la operación a una variable de coma flotante, en este caso será la intensidad que se utilizaría en un proyecto en el que se quisiera medir corriente continua.

Uno de los mayores defectos de este sencillo código es que el sensor no posee ninguna clase de filtro, por lo que, si se representan los valores de la corriente en el monitor serial de Arduino, se puede observar que varían con cada medición, por lo que lo ideal es aplicar un filtro a la medición para que sea más exacta y no varíe tanto.

Para crear un filtro, se hará desde una variable fuera del *void loop ()*. A pesar de que complique el código de la lectura del sensor ACS712, aplicar el filtro se basa en asignar el valor de la corriente a una variable de coma flotante y dividir el valor acumulado de dicha corriente entre el número de muestras que se realizan. Para ello, lo primero que se debe realizar es crear la variable en la que se realizará todo este proceso y esto se hará una vez finalizado el *void loop ()*.

```
float get_corriente(int num_muestras)
{
  float voltajeSensor;
  float corriente=0;
```

Posteriormente, como lo que se quiere es que se haga una ponderación entre todas las muestras, se deberá crear un bucle en el que con cada ciclo se realice una muestra más. En este bucle es donde se situarán las dos ecuaciones expuestas anteriormente:

```
for(int i=0;i<num_muestras;i++)
  {
      voltajeSensor = analogRead(A0) * (5.0 / 1023.0);
      corriente=corriente+(voltajeSensor-2.58)/Sensibilidad;
  }
```

Una vez creado ese bucle, solo queda hacer la ponderación de la corriente para obtener un valor medio mucho más constante de lo que se obtenía antes de aplicar este filtro:

```
corriente=corriente/num_muestras;
  return(corriente);
}
```

Como lo que se busca en el proyecto es la medición de la corriente en una instalación eléctrica de un hogar, no se puede utilizar el mismo programa para el sensor ACS712 que se utilizaría para corriente continua. El siguiente código se ha obtenido de Naylamp Mechatronics y está lo suficientemente optimizado como para que se pueda aplicar al proyecto realizado. En este código para corriente alterna, además de poder obtener la corriente de pico, también se puede obtener el valor eficaz de la misma y la potencia.

En este código, además de tener que definir una variable para la sensibilidad del sensor, también se deberá definir una variable para el *offset* y así poder reducir el ruido en el muestreo. En las primeras pruebas al *offset* se le ha asignado un valor igual a 0 para poder conocer aproximadamente la amplitud del error y corregirlo en la mayor medida posible:

```
float Sensibilidad=0.185;
float offset=0.050;
```

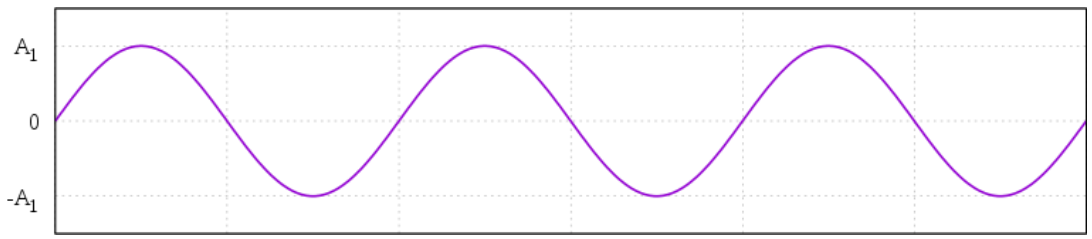
De nuevo, al igual que para la aplicación del sensor para corriente continua con filtro, crearemos una variable para obtener la corriente después del *void loop ()*.

```
float get_corriente()
{
  float voltajeSensor;
  float corriente=0;
  long  tiempo=millis();
  float Imax=0;
  float Imin=0;
```

En este caso se utiliza una variable de tiempo *millis()* para poder asignar un tiempo de muestreo del sensor, que será cada 500 milisegundos, por lo que crearemos un bucle que solo se ejecutará cuando la variable *millis()* tenga un valor menos a los 500 milisegundos. La fórmula de obtención del voltaje del sensor sigue siendo la misma que en los anteriores ejemplos, pero en este caso la fórmula de la corriente sí que se ve alterada. Para reducir el ruido, se ha aplicado un filtro paso bajo, similar a un promedio de 10 muestras:

```
while(millis()-tiempo<500)
  {
    voltajeSensor = analogRead(A0) * (5.0 / 1023.0);
    Serial.println (voltajeSensor);
    corriente=0.9*corriente+0.1*((voltajeSensor-2.5)/Sensibilidad);
    if(corriente>Imax)Imax=corriente;
    if(corriente<Imin)Imin=corriente;
  }
```

Véase que se han creado dos condiciones para asignador el valor de la corriente a dos variables diferentes. Esto es debido a que la corriente alterna es sinusoidal y esto provoca que tenga tanto valores positivos como valores negativos, por lo que se ha asignado el valor mínimo medido en cada muestra a una variable diferente a la del valor máximo, para así poder ponderar y obtener la corriente de pico.



Para ello, se ha creado una pequeña fórmula que devolverá el valor obtenido al *void loop ()*. Donde se resta el valor de la intensidad mínima al de la máxima, pero al ser el valor mínimo negativo, se está realizando una suma de sus valores absolutos, por lo que habrá que dividir entre dos para obtener el valor medio de dicha corriente en valor absoluto. Una vez hecho eso, se le restará el valor del *offset* para reducir el error, aunque se deben realizar varias pruebas para obtener un valor de *offset* que sea real.

```
return(((Imax-Imin)/2)-offset);
}
```

Por tanto, en el *void loop()* podremos leer este valor y asignárselo a la variable con coma flotante de la intensidad de pico. En caso de querer obtener el valor eficaz de la corriente, este se corresponde a:

$$I_{rms} = \frac{I_p}{\sqrt{2}} = I_p \cdot 0{,}707$$

Al igual que la potencia que se puede obtener con una sencilla ecuación:

$$P = I_{rms} \cdot V_{red} = I_{rms} \cdot 230$$

Por lo tanto, en el programa quedará de la siguiente forma:

```
float Ip=get_corriente();
float Irms=Ip*0.707;
float P=Irms*230.0;
```

De este modo, ya se podrá aplicar el sensor de corriente al código del proyecto para detectar cuando una de las líneas está activa o no, a pesar de no haberlo hecho mediante Telegram, sino que se haya hecho de forma manual mediante un conmutador.

4.2 Desarrollo del circuito

Con todos los componentes que se van a utilizar claro, se procede al diseño del circuito que será controlado por la placa NodeMCU y que será programado con las diferentes referencias de código de los apartados anteriores. Puesto que se va a tratar de crear un circuito lo más compacto posible, se ha limitado el número de actuadores a 3, es decir, el diseño a seguir se ha realizado con solo 3 relés, pues en una vivienda, cada habitación no tiene muchas más líneas, sobre todo si solo contamos con las de iluminación.

Cada uno de estos relés necesita un pin digital libre para poder actuar sobre él, al igual que el sensor de temperatura y humedad DHT22, que solamente precisa de un pin para transmitir los datos. Puesto que los relés funcionan a una tensión de 5 V también necesitan estar conectados a tierra, se ha creado una línea paralelo del pin de alimentación de la placa NodeMCU, que, a pesar de trabajar a 3,3 V, si está alimentada con una fuente de 5 V, permite sacar por ese pin esa tensión. Lo mismo ocurriría con una fuente de 9 V, por ejemplo, dicho pin sacaría una tensión de 9 V.

Por otro lado, el sensor de corriente ACS712 requiere estar conectado a un pin analógico y puesto que la placa NodeMCU solo cuenta con un pin analógico, tan solo se podrá colocar un sensor de corriente por cada circuito creado. Tanto este sensor como el sensor DHT22 estarán alimentados a 3,3 V, puesto que es la tensión de trabajo del NodeMCU y trabajar a 5 V puede causar errores en la medición. El esquema final del circuito se puede apreciar en la *Imagen 4.2.1*.

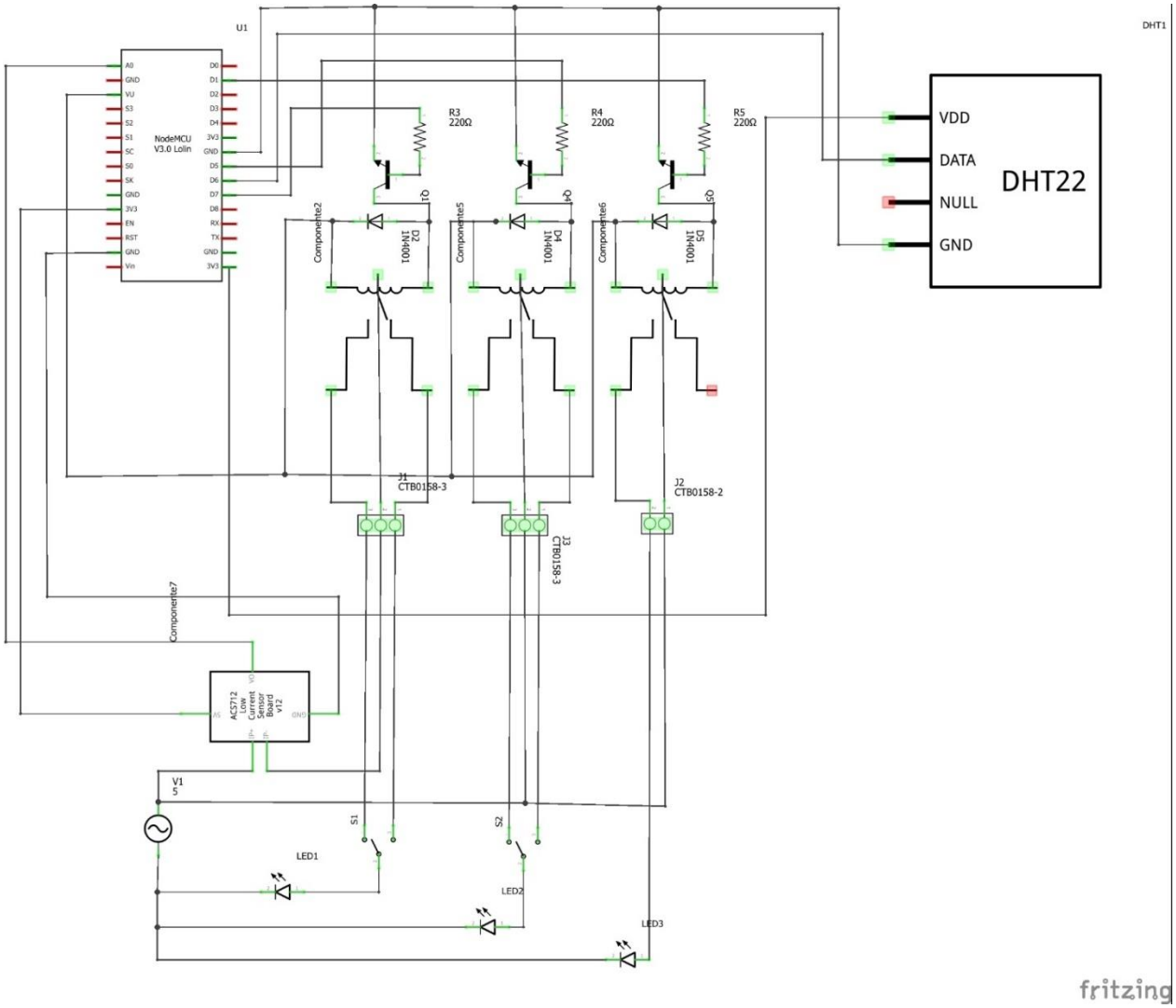

U1
NodeMCU
V3.0 Lolin
DHT1
DHT22
VDD
DATA
NULL
GND
R3
220Ω
R4
220Ω
R5
220Ω
Q1
Q4
Q5
D2
1N4001
D4
1N4001
D5
1N4001
Componette2
Componette5
Componette6
Componette7
J1
CTB0158-3
J2
CTB0158-2
J3
CTB0158-3
ACS712
Low
Current
Sensor
Board
V12
V1
5
S1
S2
LED1
LED2
LED3
fritzing

Imagen 4.2.1

En el esquema, se puede apreciar la conexión de dos de los relés a una bornera de tres terminales. Estos dos relés serán los que actúen como conmutadores, mientras que el tercer relé, conectado a bornera de dos terminales actuará como interruptor. Estos tres relés están conectados a los pines D1, D5 y D7, que se corresponden con las entradas GPIO5, GPIO14 y GPIO13 respectivamente. Es el número de la entrada el cual se deberá escribir en el sketch final en el entorno Arduino.

Por otro lado, el sensor de temperatura y humedad DHT22 está conectado al pin D6, que corresponde a la entrada GPIO12. Todos estos elementos están conectados tanto a alimentación como a tierra.

En la parte inferior del esquema, se ha representado de manera escueta una instalación en corriente alterna para poder representar el sensor de corriente ACS712 de manera correcta. La representación de los diodos LED se ha hecho porque en el programa fritzing que se ha utilizado para el diseño del circuito, no tiene implementado ningún símbolo de una bombilla o cualquier carga, por lo que el diodo LED es lo más parecido a una bombilla tradicional. Este sensor está conectado al único pin analógico de la placa NodeMCU, al igual que a tierra y con alimentación a 3,3 V. Si lo alimentamos a 5 V, como la placa NodeMCU trabaja a 3,3 V la medición no será correcta.

En la bornera de este sensor está conectado uno de los conductores que después entran en una de las borneras de los relés que actuarán como conmutadores. Esta conexión debe ser en serie, para que la medición pueda realizarse. La función de este sensor será indicar que hay un flujo de corriente por el conductor que lo atraviesa, para así poder conocer el estado de alguna de las conexiones, habiendo sido activada o no de manera remota con Telegram.

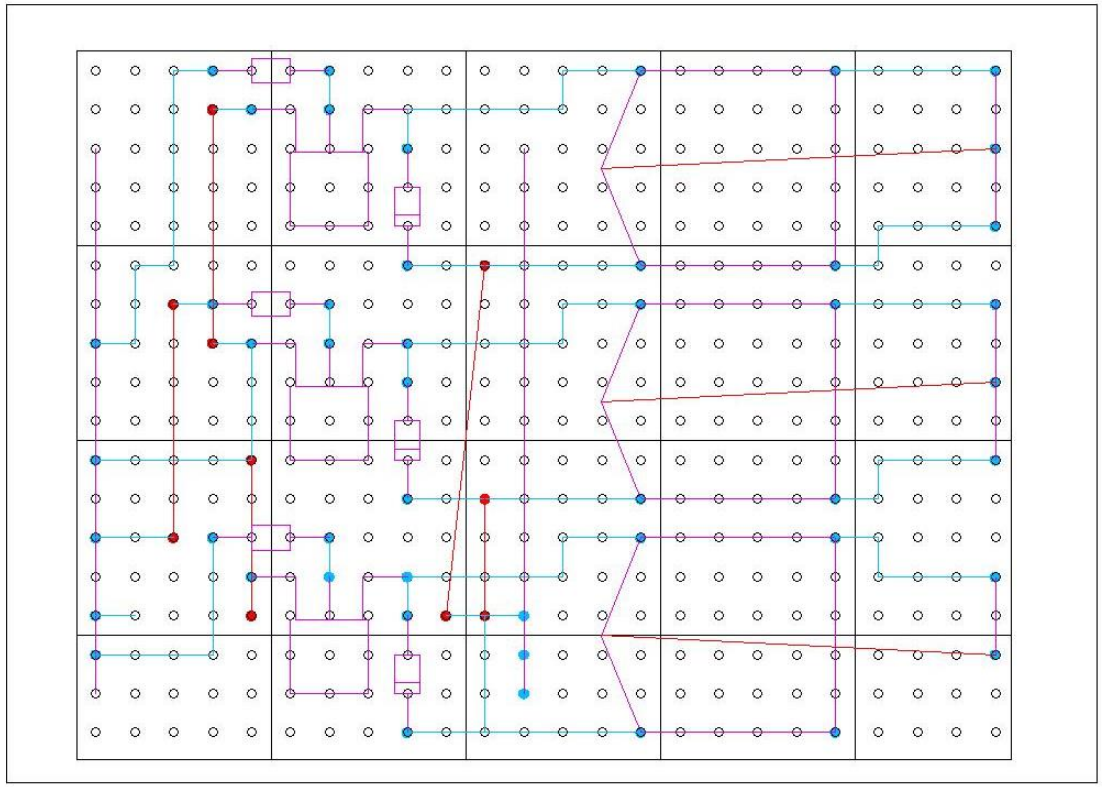
4.3 Desarrollo de la placa prototipo

Como ya se ha comentado en varias ocasiones, uno de los objetivos de este proyecto es que el montaje sea lo suficientemente pequeño como para poder realizar una instalación del mismo sin que haya grandes cambios en la instalación eléctrica y en la estructura de la habitación, por ejemplo, que no haya necesidad de picar un agujero más grande para que quepa la placa final y los interruptores o conmutadores. Por ello, se ha decidido realizar el montaje en una placa de prototipo, soldando los componentes para que ocupen el menor espacio posible.

En este caso se ha escogido una placa de 7 cm x 5 cm, unas dimensiones casi idénticas a las que encontramos en un Arduino UNO, por lo que se puede observar a simple vista, que la elección del NodeMCU es más que correcta para este proyecto. El diseño de las conexiones para esta placa prototipo de ha realizado mediante el programa AutoCAD, puesto que para este tipo de placas no se ha encontrado ningún programa adecuado, aunque si después se quisiese crear una placa PCB, es decir, una placa de circuito impreso, sí que deberíamos de usar un programa especializado.

El diseño de esta placa se ha basado en la máxima compactación del circuito, por lo que algunos de los componentes se encuentran debajo de la placa NodeMCU. En concreto los componentes que se encuentran soldados debajo de esta placa son las resistencias, diodos y transistores del circuito de los relés. Puesto que los pines de conexión de la placa NodeMCU ya ocupan prácticamente la totalidad de los agujeros disponibles en la posición en la que se encuentra, se ha optado por cortar algunos de los conectores de la placa NodeMCU que no estaban utilizando en el proyecto para que así se pudieran crear las pistas de estaño sin mayores dificultades.

El compactar tanto el diseño repercute de manera negativa al propio diseño de las pistas, pues en un espacio tan reducido se complica la posibilidad de crear las pistas de estaño suficientes, por lo que se ha recurrido a la utilización de conductores unifilares de cobre para conectar de manera externa algunos puntos del circuito. Con ayuda de la *Imagen 4.3.1* se puede apreciar mejor.



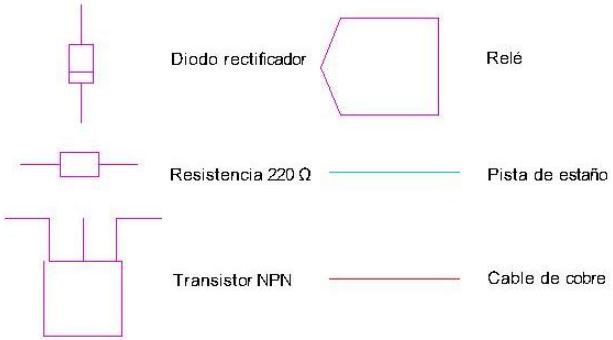

Diodo rectificador
Relé
Resistencia 220 Ω
Pista de estaño
Transistor NPN
Cable de cobre

Imagen 4.3.1

En el esquema de la placa programable, los puntos que están rellenos indican que en ese lugar habrá una soldadura de algún componente. Si el relleno es en azul, se trata de algún componente como tal, por ejemplo, las resistencias de 220 Ω. En caso de que el relleno sea de color rojo, indica que ese punto de soldadura se corresponde con un conductor de cobre. Los agujeros marcados como soldadura de algún componente y que no están conectados en ese diseño a ningún lugar, son los pines destinados al sensor de temperatura DHT22 y al sensor de corriente ACS712, que se encuentran fuera de la placa. Esta decisión, además de haberse realizado para compactar todo el montaje lo máximo posible es para poder situar el sensor DHT22 en un lugar exterior a la instalación eléctrica, al igual que el sensor ACS712 que debe estar en línea con el conductor que entre a uno de los relés y para no forzar los conductores se situará fuera de la placa.

4.4 Montaje de la placa prototipo

Se ha plasmado en imágenes el montaje progresivo de la placa prototipo siguiendo el diseño realizado en AutoCAD. Una de las resistencias, ha tenido que ser cambiada de posición debido a que uno de los agujeros de conexión fue quemado en el proceso de soldadura y ya el estaño ya no se adhería.

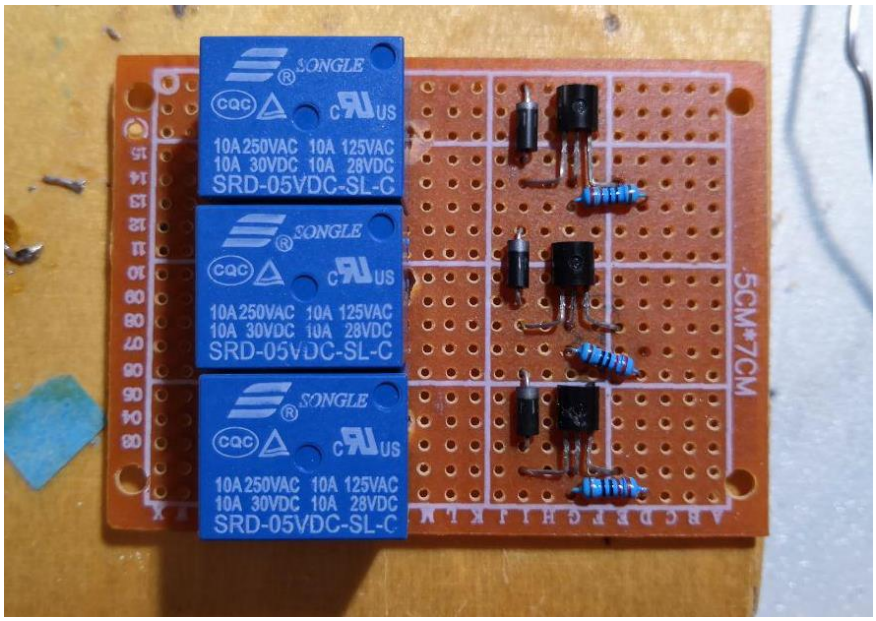


Imagen 4.4.1

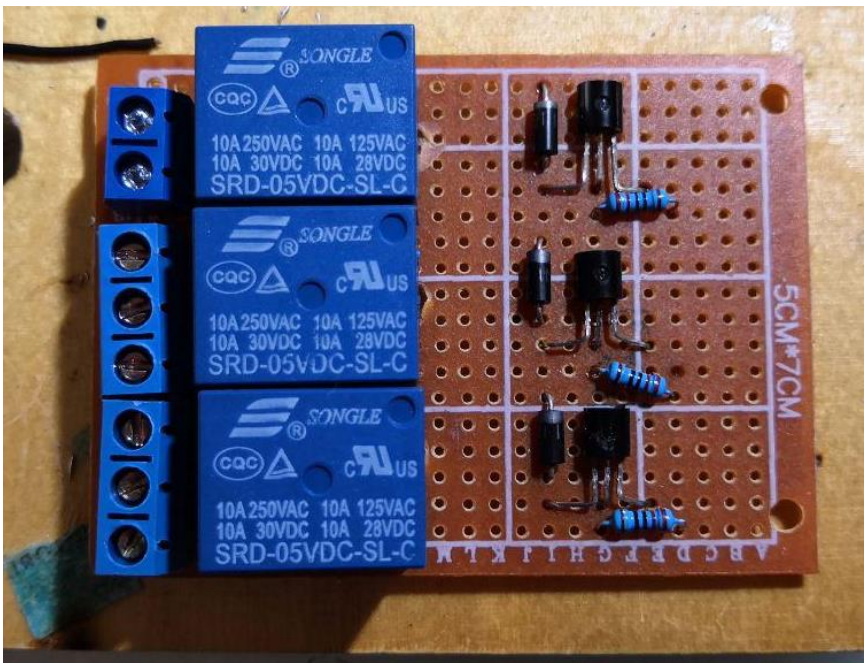


Imagen 4.4.2

En este punto del montaje, se ha tenido que hacer unos agujeros más grandes para la conexión del pin central de los relés, pues además de tener un mayor tamaño que los agujeros, no estaba centrado en ellos, por lo que se ha tenido que crear el agujero en cuestión.

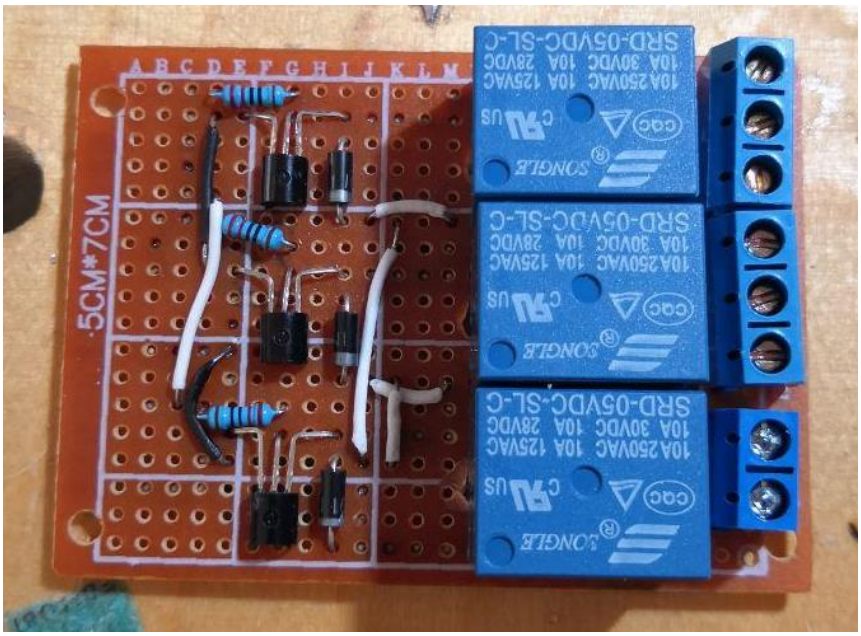


Imagen 4.4.3

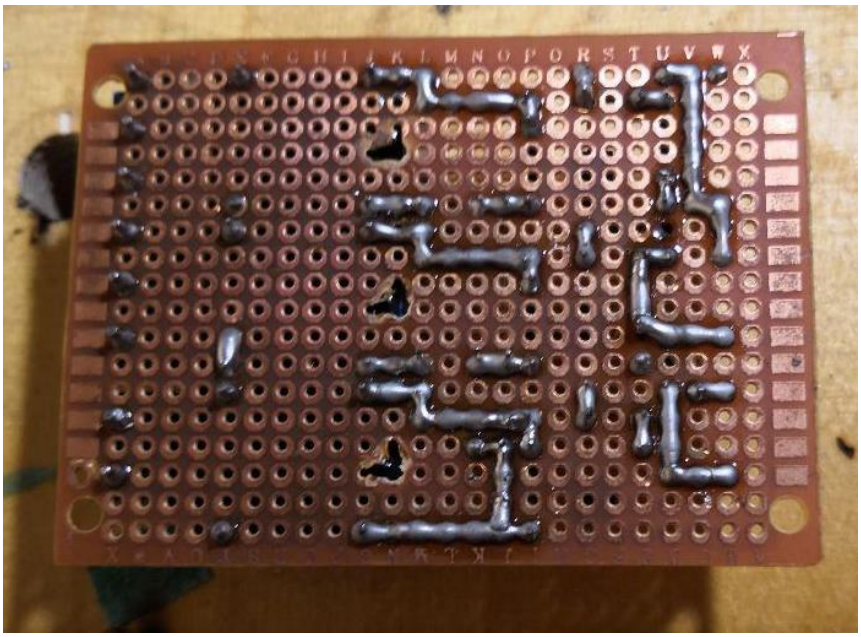


Imagen 4.4.4

En la *Imagen 4.4.4* se puede apreciar la conexión de todos los componentes mediante pistas de estaño, mientras que en la *Imagen 4.4.3* se observan los conductores unifilares de cobre que permiten la conexión de algunos puntos comunes, como la conexión a tierra de una de las ramas de los transistores o la conexión en una línea de alimentación de los tres relés de 5 V. Todas estas conexiones se han hecho respetando los pines de conexión del NodeMCU, a excepción de dos de ellos que se han tenido que cortar para su implementación a la placa prototipo.

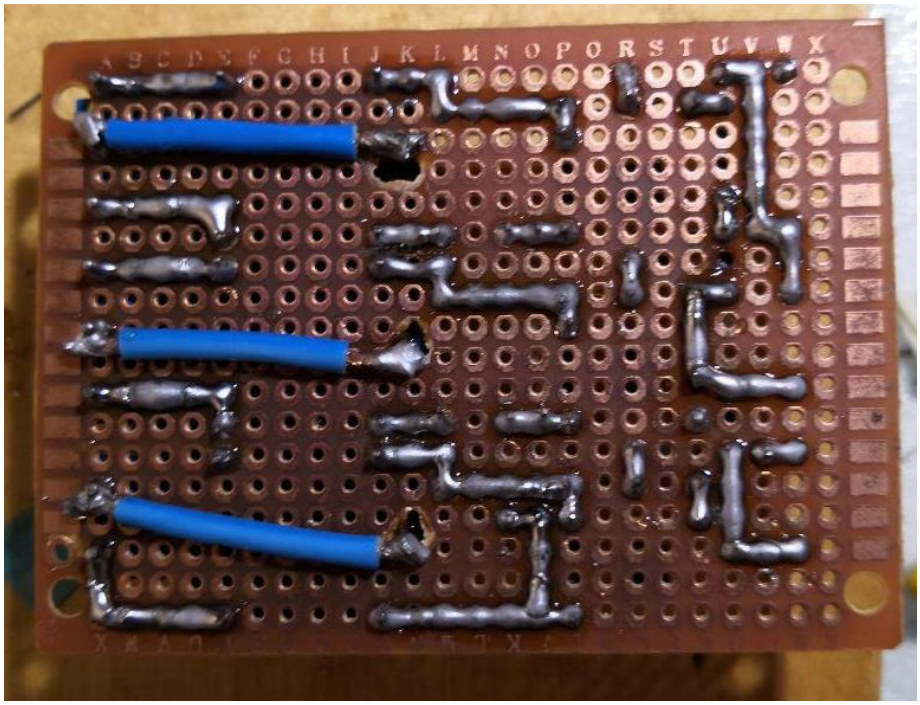


Imagen 4.4.5

En la *Imagen 4.4.5* se aprecia la conexión de los relés a las borneras. Esta conexión se ha realizado con un cable de 2 mm de diámetro para soportar la tensión de 230 V de una vivienda. Las otras conexiones, más cercanas entre bornera y relé, sí que se han realizado con estaño, pues su temperatura de fusión es mucho más alta de la que se tiene con el paso de la corriente, por lo que soportan tanto las tensiones como las corrientes de la vivienda.

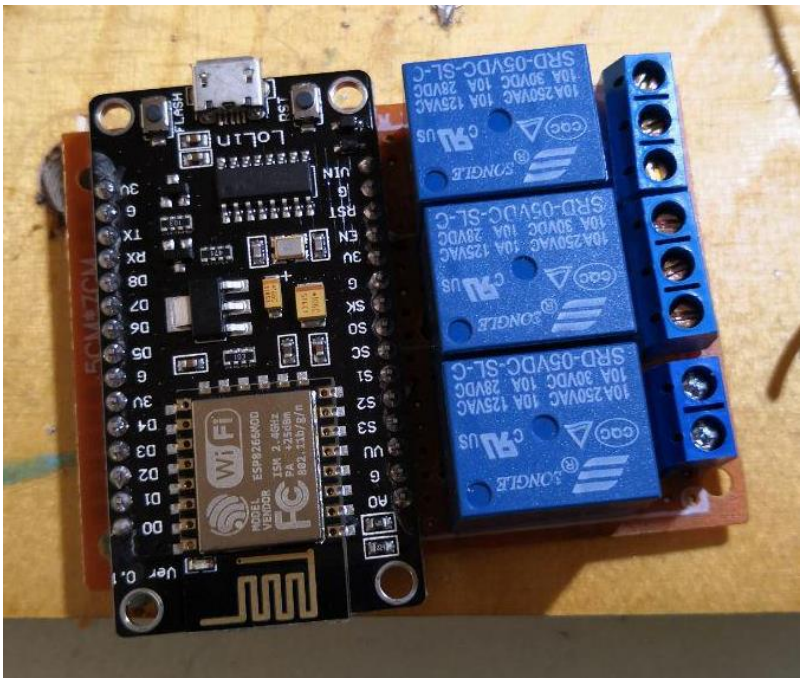


Imagen 4.4.6

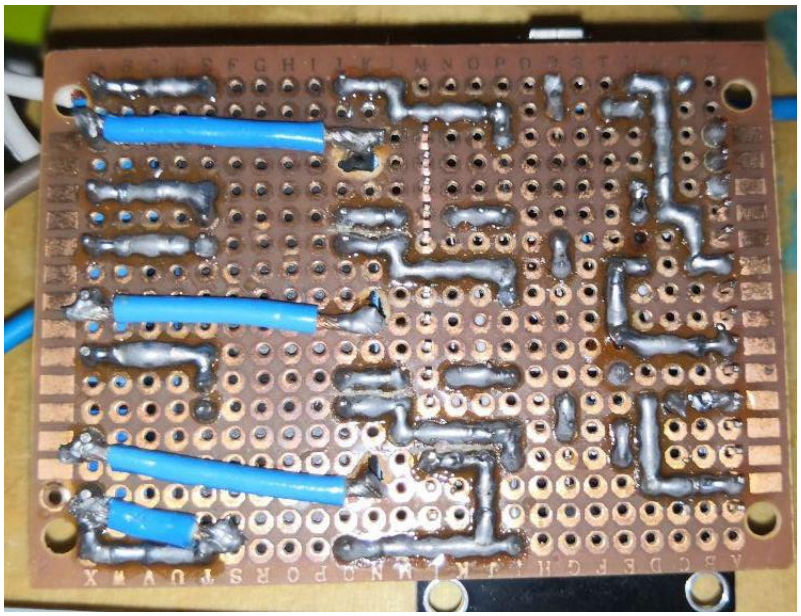


Imagen 4.4.7

4.5 Código final del programa

Con las diferentes implementaciones de código para cada componente que se han visto en el punto 4.1 Desarrollo del programa, se ha desarrollado un código final de programa que junta todos esos componentes en un solo sketch de Arduino:

```
float Ip=get_corriente();
//Incluimos las librerías necesarias para funcionar con la placa
NodeMCU, el sensor de humedad y temperatura y Telegram

#include  <ESP8266WiFi.h>
#include  <WiFiClientSecure.h>
#include  <TelegramBot.h>
#include <Adafruit_Sensor.h>
#include <DHT.h>

//La libreria del sensor soporta dos tipos diferentes, por lo que
especificamos cual usamos nosotros

#define tipo_de_sensor DHT22

//Definimos los puertos de la placa que usaremos

#define rele_primero 13
#define sensor_humedad 12
#define rele_segundo 14
#define rele_tercero 5

//Creamos una variable de estado para cada rele

int variable_primero;

const char* estado_primero;
const char* estado_segundo;
const char* estado_tercero;

const char* encendido = "ON";
const char* apagado = "OFF";

//Iniciamos la conexión WiFi introduciendo las credenciales de nuestra
red

const char* ssid     = "Nombre red WiFi";
const char* password = "Contraseña red WiFi";

//Iniciamos el bot de Telegram. Para ello necesitaremos introducir su
token

const char BotToken[] = "Bot Token"; //Cada bot tiene un token
diferente
WiFiClientSecure net_ssl;
TelegramBot bot (BotToken, net_ssl);

//Para el sensor de corriente, asignamos una sensibilidad según los
datos del fabricante

float Sensibilidad = 0.5;
```

```
float offset=0.100; //Amplitud aproximada del ruido generado por el
sensor

//Sensor de humedad y temperatura: definimos el pin en el que está y
que tipo de sensor es

DHT dht(sensor_humedad, tipo_de_sensor);

void setup()
  {

  Serial.begin(9600);
  delay(3000);
  //Nos conectamos a la red WiFi
  Serial.print("Connecting Wifi: ");
  Serial.println(ssid);
  WiFi.begin(ssid, password);
  Serial.println("");
  Serial.println("WiFi connected");
  bot.begin();
  dht.begin();

  //Definimos los diferentes pins como entradas o salidas

  pinMode(rele_primero, OUTPUT);
  pinMode(rele_segundo, OUTPUT);
  pinMode(rele_tercero, OUTPUT);

  //Definimos el valor inicial de la variable de estado de cada rele

  variable_primero = 0;
  variable_segundo = 0;

  //Definimos los relés que no cuentan con sensor de corriente como
apagados desde el principio

  estado_segundo = apagado;
  estado_tercero = apagado;

  //El valor de los relés de primeras estarán a 0

  digitalWrite (rele_primero, 0);
  digitalWrite (rele_segundo, 0);
  digitalWrite (rele_tercero, 0);

  }


void loop()
  {

  //Para medir la corriente que pasa por el conductor necesitamos muy
poco código siendo este el siguiente

  float Ip=get_corriente();//obtenemos la corriente pico

  //Actuamos sobre la variable de estado según si pasa o no corriente
por el conductor

  if (Ip>=0.006)
  {
```

```
    estado_primero = encendido;
    variable_primero = 1;
  }
  else
  {
    estado_primero = apagado;
    variable_primero = 0;
  }

  float h = dht.readHumidity();
  float t = dht.readTemperature();

  char humedad [6];
  char temperatura [4];
  dtostrf(h, 4, 1, humedad);
  dtostrf(t, 6, 1, temperatura);

//Esta parte si sabemos que el sensor funciona la podemos comentar
  if (isnan(h) || isnan(t)) {
      Serial.println("Failed to read from DHT sensor!");
      return;
   }
//

  message m = bot.getUpdates(); //Lee nuevos mensajes del bot

  if (m.text.equals("Estado cocina"))
  {
    Serial.println ("Mensaje recibido, enviando el estado de la
cocina");
    bot.sendMessage(m.chat_id, "Estado de las conexiones de la cocina:
");
    bot.sendMessage(m.chat_id, "La luz está: ");
    bot.sendMessage(m.chat_id, estado_primero);
    bot.sendMessage(m.chat_id, "La luz del fregadero está: ");
    bot.sendMessage(m.chat_id, estado_segundo);
    bot.sendMessage(m.chat_id, "La calefacción está: ");
    bot.sendMessage(m.chat_id, estado_tercero);
  }

  if (m.text.equals ("Temperatura cocina"))
  {
    Serial.println ("Mensaje recibido, enviando temperatura de la
cocina");
    bot.sendMessage(m.chat_id, "La temperatura en °C en la cocina es
de: ");
    bot.sendMessage(m.chat_id, temperatura);
    bot.sendMessage(m.chat_id, "La humedad en % en la cocina es de:
");
    bot.sendMessage(m.chat_id, humedad);
  }
  //El siguiente código variará según que elementos estemos
controlando de cada habitación

  if (m.text.equals("Luces cocina") && variable_primero == 1 &&
digitalRead(rele_primero) == 0)
  {
    Serial.println ("Mensaje recibido, se apagarán las luces de la
cocina");
    digitalWrite (rele_primero, 1);
```

```
    bot.sendMessage(m.chat_id, "Se han apagado las luces de la
cocina");
  }
  else if (m.text.equals("Luces cocina") && variable_primero == 1 &&
digitalRead(rele_primero) == 1)
  {
    Serial.println ("Mensaje recibido, se apagarán las luces de la
cocina");
    digitalWrite (rele_primero, 0);
    bot.sendMessage(m.chat_id, "Se han apagado las luces de la
cocina");
  }
  else if (m.text.equals("Luces cocina") && variable_primero == 0 &&
digitalRead(rele_primero) == 0)
  {
    Serial.println ("Mensaje recibido, se encenderán las luces de la
cocina");
    digitalWrite (rele_primero, 1);
    bot.sendMessage(m.chat_id, "Se han encendido las luces de la
cocina");
  }
  else if (m.text.equals("Luces cocina") && variable_primero == 0 &&
digitalRead(rele_primero) == 1)
  {
    Serial.println ("Mensaje recibido, se encenderán las luces de la
cocina");
    digitalWrite (rele_primero, 0);
    bot.sendMessage(m.chat_id, "Se han encendido las luces de la
cocina");
  }


  if (m.text.equals("Luces fregadero") && digitalRead(rele_segundo) ==
0)
  {
    Serial.println ("Mensaje recibido, se encenderán las luces del
fregadero");
    digitalWrite (rele_segundo, 1);
    estado_segundo = encendido;
    bot.sendMessage(m.chat_id, "Se han encendido las luces del
fregadero");
  }
  else if (m.text.equals("Luces fregadero") &&
digitalRead(rele_segundo) == 1)
  {
    Serial.println ("Mensaje recibido, se apagarán las luces del
fregadero");
    digitalWrite (rele_segundo, 0);
    estado_segundo = apagado;
    bot.sendMessage(m.chat_id, "Se han apagado las luces del
fregadero");
  }


   if (m.text.equals("Radio cocina") && digitalRead(rele_tercero) ==
0)
  {
    Serial.println ("Mensaje recibido, se encenderá la radio de la
cocina");
    digitalWrite (rele_tercero, 1);
    estado_tercero = encendido;
```

```
    bot.sendMessage(m.chat_id, "Se ha encendido la radio de la
cocina");
  }
  else if (m.text.equals("Radio cocina") && digitalRead(rele_tercero)
== 1)
  {
    Serial.println ("Mensaje recibido, se apagará la radio de la
cocina");
    digitalWrite (rele_tercero, 0);
    estado_primero = apagado;
    bot.sendMessage(m.chat_id, "Se ha apagado la radio de la cocina");
  }


  //Habría que repetir este mismo proceso para cada rele para
asegurarse de que se puede controlar de manera manual y por telegram
  }

  float get_corriente()
{
  float voltajeSensor;
  float corriente=0;
  long tiempo=millis();
  float Imax=0;
  float Imin=0;
  while(millis()-tiempo<500)//realizamos mediciones durante 0.5
segundos
  {
    voltajeSensor = analogRead(A0) * (5.0 / 1023.0);//lectura del
sensor
    corriente=0.9*corriente+0.1*((voltajeSensor-2.5)/Sensibilidad);
//Ecuación  para obtener la corriente
    if(corriente>Imax)Imax=corriente;
    if(corriente<Imin)Imin=corriente;
  }
  return(((Imax-Imin)/2)-offset);
}
```

En el programa final se han creado una serie de variables de estado para poder conocer el estado de las conexiones de la instalación y para poder conocer dicho estado mediante Telegram. Estas variables son:

```
int variable_primero;
int variable_segundo;
int variable_tercero;

const char* estado_primero;
const char* estado_segundo;
const char* estado_tercero;

const char* encendido = "ON";
const char* apagado = "OFF";
```

Donde *variable_primero* cambiará su valor entre 1 o 0 según si el sensor ACS712 detecta que hay flujo de corriente en la línea. Esta variable nos servirá en las condiciones del primer relé para saber si la línea está activa o no en caso de no haberla activado mediante el programa, pues la instalación permite una activación de manera manual mediante un conmutador. La condición para el cambio de estado en dicha línea se realiza con el siguiente comando:

```
if (m.text.equals("Luces cocina") && variable_primero == 1 &&
digitalRead(rele_primero) == 0){}
```

Donde según las condiciones que se quiera, hará una función u otra. En este ejemplo, la condición solo se cumplirá si el mensaje enviado por Telegram se corresponde con "Luces cocina" y *variable_primero* tiene una valor igual a 1 y el *rele_primero* no está activo. En caso de que cualquiera de esas tres condiciones no se cumpla, no habrá una consecuencia.

Otras variables que tienen una gran importancia son *estado_primero*, *estado_segundo* y *estado_tercero*, pues son las que nos permitirán saber si las conexiones de la instalación están activas o no. Para ello, se hará uso de las variables *encendido* y *apagado*, que están relacionadas a las palabras "ON" y "OFF" respectivamente. Para hacer uso de ellas, se relacionan las variables de *estado* con las de *encendido* y *apagado* del siguiente modo:

```
estado_tercero = encendido;
```

De este modo, la variable *estado_tercero* puede ser tanto "ON" como "OFF" para cuando se le pregunte por Telegram. En cambio, la variable *estado_primero* actúa de una forma un tanto diferente, pues su estado cambia según el valor detectado por el sensor de corriente:

```
if (Ip>=0.006)
  {
    estado_primero = encendido;
    variable_primero = 1;
  }
  else
  {
    estado_primero = apagado;
    variable_primero = 0;
  }
```

En este caso, si el sensor detecta una corriente cuyo valor sea superior a 0,006 A la variable *estado_primero* pasará a ser "ON" y la *variable_primero* valdrá 1. Lo contrario ocurrirá si la corriente detectada es menor a 0,006 A.

5. Resultados

El objetivo principal de este proyecto se ha cumplido: es posible controlar una vivienda mediante la aplicación de mensajería instantánea Telegram y una placa programable basada en el entorno Arduino. La placa responde a las órdenes enviadas mediante el chat con el Bot de Telegram activando o desactivando los diferentes componentes conectados sin importar si estamos en la misma red WiFi o si estamos en la calle utilizando los datos móviles, pues la placa NodeMCU puede leer todos los mensajes recibidos en el chat del Bot.

Si bien el prototipo es funcional, la velocidad de respuesta es lenta y no es del todo aplicable en un entorno real, donde resulta más rápido activar los diferentes elementos de forma manual que esperar la respuesta del proyecto. Sí que puede ser una opción viable para quien necesite realizar este control desde fuera de la vivienda, pues en este caso, esperar unos cuantos segundos de más no es de importancia, pero estando dentro de la vivienda, que la respuesta a nuestras órdenes tarde aproximadamente 30 segundos en realizarse hace que no sea factible en comparación a otras alternativas ya instauradas en el mercado.

Uno de los mayores defectos del proyecto es que no se tuvo en cuenta en un inicio que la placa NodeMCU solo tenía un pin analógico, por lo que solo se ha podido implementar un sensor de corriente ACS712, cuando lo ideal hubiese sido que por lo menos para los dos relés que actúan como conmutador se pudiese saber si hay flujo de corriente o no, por lo que es uno de los principales errores de este proyecto, pues la idea de utilizar el sensor de corriente surgió con la necesidad de saber el estado de las conexiones una vez el proyecto estaba en desarrollo.

Otro defecto del proyecto es el diseño de la placa de prototipo. Si bien ha funcionado y no se ha tenido ningún problema, el hecho de soldar directamente la placa NodeMCU a la placa de prototipo podría haber causado que las temperaturas provocadas por el soldador de estaño provocaran algún tipo de error en el NodeMCU, al igual que cortar algunos de los pines del NodeMCU no es la mejor solución para un error al crear las pistas de estaño.

Básicamente se han cumplido la mayoría de los objetivos del proyecto, lo que nos permite crear un hogar inteligente con el diseño de varias placas. Por supuesto el proyecto tiene un margen de mejora, pero se ha conseguido que sea funcional y que su instalación sea algo viable gracias al reducido tamaño. Uno de los principales objetivos que no se ha podido cumplir, es del control por voz mediante Google Assistant, pues para ello se tiene que poder incluir al Bot creado en un grupo de Telegram y en estos casos la librería de Telegram para Arduino no es capaz de leer los mensajes enviados al grupo en cuestión, a pesar de que el Bot sea el mismo.

En el siguiente enlace se puede ver un vídeo sobre el funcionamiento del proyecto: https://youtu.be/TIwLeMhDfgI

6. Perspectivas

Una vez finalizado el proyecto, la mejora todavía es posible una vez analizados todos los errores del proyecto. En primer lugar, uno de los principales elementos que podrían sustituirse es el NodeMCU, que, si bien es el corazón del proyecto, se podría incluir una placa programable mediante el entorno Arduino que contase con más de un pin analógico. Otra de las alternativas es utilizar una versión más compacta del NodeMCU, la cual permitiría reducir todavía más el tamaño del proyecto y las únicas pérdidas significativas son algunos pines digitales de los muchos que no se han utilizado. Además, a la hora de integrar la nueva placa programable en la placa de prototipo, se utilizarían conectores hembra de Arduino que quedarían soldados, mientras que la placa WeMos D1 mini, que es la versión compacta del NodeMCU, solo habría que pincharla en los conectores hembra, sin riesgo de romper la placa.

Si bien se ha intentado el control por voz con Google Assitant para actuar sobre el Bot de Telegram, también se podría integrar este control directamente con la placa, gracias a las librerías de Adafruit, aunque esto no era lo que se quería integrar en el proyecto, pues este se basa en el control mediante Telegram, pero para futuras versiones más versátiles resulta una buena alternativa.

En cuanto al control de persianas según la posición del sol, requeriría realizar una instalación para automatizar la persiana en cuestión a base de motores, lo que se escapa de las dimensiones de este proyecto.

Con las mejoras correspondientes, se podría diseñar el circuito para placas PCB de circuito impreso para un mejor y más profesional montaje del proyecto, lo que facilitaría la instalación final en algún tipo de caja para poder ser instalado en el interior de una pared y así poder controlar de manera real las diferentes conexiones de una habitación, aunque si se utiliza una placa programable con un solo pin analógico convendría que la mayoría de los relés actuasen únicamente sobre enchufes y no tanto sobre las luces, para poder controlar su estado si estamos fuera de casa.

7. Bibliografía

- Información oficial sobre los Bots de Telegram: https://telegram.org/blog/bot-revolution
- Introducción a los Bots de Telegram: https://core.telegram.org/bots
- Primeros pasos y explicación sobre la librería de Telegram para Arduino: https://create.arduino.cc/projecthub/Arduino_Genuino/telegram-bot-library-ced4d4
- Librería de Telegram para el módulo ESP8266: https://github.com/Gianbacchio/ESP8266-TelegramBot
- Librería del módulo WiFi ESP8266 para Arduino: https://github.com/esp8266/Arduino/tree/master/doc/esp8266wifi
- Librería del sensor de temperatura y humedad DHT22 para Arduino: https://github.com/adafruit/DHT-sensor-library
- Documentación sobre el sensor de temperatura y humedad DHT22: https://www.sparkfun.com/datasheets/Sensors/Temperature/DHT22.pdf
- Explicación y ejemplos del sensor de corriente ACS172: https://naylampmechatronics.com/blog/48_tutorial-sensor-de-corriente-acs712.html
- Tipos de variables en Arduino: http://diwo.bq.com/variables-en-arduino/
- Imagen de onda senoidal: https://commons.wikimedia.org/w/index.php?curid=52042139

Material utilizado:

- Sensor de corriente ACS712
- Transistor NPN
- Relé 5V 10A
- Bornera dos conectores
- Diodo rectificador
- Panel placa prototipo
- NodeMCU
- Bornera tres conectores
- Sensor DHT22