Escuela Politécnica Superior de Linares

Grado en Ingeniería Telemática

UNIVERSIDAD DE JAÉN
Escuela Politécnica Superior de Linares

Trabajo Fin de Grado

REGULADOR ONLINE PROGRAMABLE BASADO EN ARDUINO PARA EL SISTEMA DE CLIMATIZACIÓN JOHNSON CONTROL EXISTENTE EN LA ESCUELA POLITÉCNICA SUPERIOR DE LINARES.

Alumno: Cristian Gheorghe Maier

Tutor: José Vicente Muñoz Díez
Depto.: Ingeniería Electrónica y Automática

Tutor: José Enrique Muñoz Expósito
Depto.: Ingeniería de Telecomunicación

Septiembre, 2018

UNIVERSIDAD DE JAÉN

Escuela Politécnica Superior de Linares

Trabajo Fin de Grado

REGULADOR ONLINE PROGRAMABLE BASADO EN ARDUINO PARA EL SISTEMA DE CLIMATIZACIÓN JOHNSON CONTROL EXISTENTE EN LA ESCUELA POLITÉCNICA SUPERIOR DE LINARES.

Alumno: **Cristian Gheorghe Maier**

Firma: Cristian Gheorghe Maier

MAIER GHEORGHE CRISTIAN - X7976058A

Firmado digitalmente por MAIER GHEORGHE CRISTIAN - X7976058A
Nombre de reconocimiento (DN): c=ES, serialNumber=IDCES-X7976058A, givenName=GHEORGHE CRISTIAN, sn=MAIER, cn=MAIER GHEORGHE CRISTIAN - X7976058A
Fecha: 2018.09.05 12:46:14 +02'00'

Firma: José Vicente Muñoz Díez

MUÑOZ DIEZ JOSE VICENTE - 77323693Q

Firmado digitalmente por MUÑOZ DIEZ JOSE VICENTE - 77323693Q
Fecha: 2018.09.05 10:42:08 +02'00'

Firma: José Enrique Muñoz Expósito

MUÑOZ EXPOSITO JOSE ENRIQUE - 27386513E

Firmado digitalmente por MUÑOZ EXPOSITO JOSE ENRIQUE - 27386513E
Nombre de reconocimiento (DN): c=ES, serialNumber=27386513E, sn=MUÑOZ EXPOSITO, givenName=JOSE ENRIQUE, cn=MUÑOZ EXPOSITO JOSE ENRIQUE - 27386513E
Fecha: 2018.09.05 13:05:12 +02'00'

ÍNDICE GENERAL

ÍNDICE FIGURAS

ÍNDICE TABLAS

MEMORIA

1. Capítulo 1: Introducción

1.1. Promotor

Los promotores de este trabajo fin de grado han sido el Departamento de Ingeniería Electrónica y Automática junto con el Departamento de Ingeniería de Telecomunicación, los cuales han tratado de estudiar la viabilidad técnica de instalar un sistema de bajo coste capaz de automatizar de forma individual la climatización de los despachos del edificio departamental de la Escuela Politécnica Superior de Linares

1.2. Entorno Tecnológico

1.2.1. Domótica. Sistemas embebidos. Internet de las cosas

La domótica puede definirse como el conjunto de sistemas que automatizan los diferentes dispositivos eléctricos o electrónicos de una vivienda o edificio. Con el uso de la domótica se consigue un mayor ahorro de energía, mayor seguridad y comodidad.

La proliferación de los sistemas electrónicos de reducido tamaño, con conectividad a Internet y bajo coste -como es el caso de los sistemas embebidos- ha provocado un creciente interés en la domótica. Sistema embebido es el nombre genérico que reciben los equipos electrónicos que incluyen un procesamiento de datos pero que a diferencia de un ordenador personal, están diseñados para satisfacer una función específica.

Los sistemas embebidos, -también llamados sistemas empotrados- no solo se pueden encontrar en procesos industriales, sino que se están presentes en cualquier dispositivo electrónico de nuestro día a día. Gran cantidad de equipos integran estos sistemas, como por ejemplo coches, ascensores, electrodomésticos, juguetes, dispositivos móviles, etc. La unidad de control de un sistema embebido es típicamente un microcontrolador o un microprocesador y su diseño está optimizado para reducir su tamaño, su costo y su consumo energético. Un microcontrolador incluye en su interior las tres principales unidades funcionales de un ordenador: unidad central de procesamiento (*CPU*), memoria (ROM, RAM) y periféricos de entrada/salida. (Figura 1).

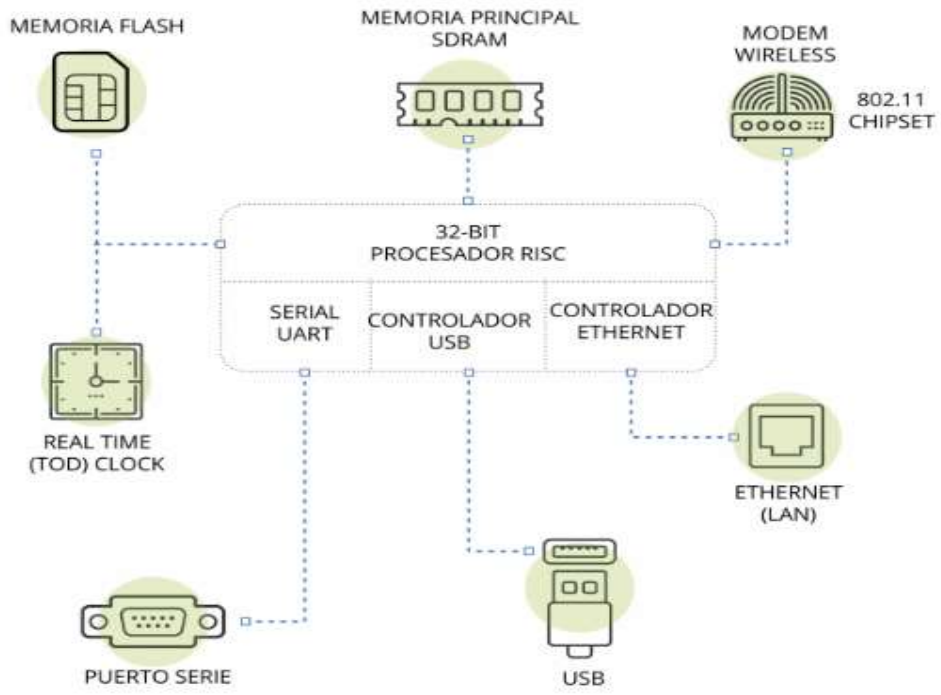


Figura 1.1 - Esquema General de un Sistema Embebido [19]

El Internet de las cosas (IoT, del término en inglés Internet of Things) es una arquitectura de red basada en la interconexión y control de dispositivos electrónicos de uso cotidiano valiéndose de Internet. Los referidos dispositivos electrónicos se valen de sistemas embebidos que les permite no solo la conectividad a Internet, sino que pueden ejecutar tareas específicos en función de eventos recibidos remotamente. Para controlar los dispositivos desde la red de Internet es necesario que los sistemas embebidos estén dotados de una comunicación inalámbrica o una conexión cableada mediante un puerto RJ45. Dentro de los sistemas embebidos dotados de comunicación inalámbrica se pueden encontrar sistemas embebidos carentes de sistema operativo (como el NodeMCU) o dispositivos que tengan un sistema operativo instalado (Raspberry Pi). La utilización de estos sistemas embebidos con o sin sistema operativo en IoT es posible gracias a protocolos de comunicación que son sensibles a las características de los sistemas empotrados utilizados. En el IoT uno de los protocolos de comunicación más utilizados es el protocolo MQTT (del término en inglés Message Queuing Telemetry Transport [2]) el cual está diseñado para comunicaciones M2M (Máquina a Máquina, del término en inglés Machine to Machine). Las comunicaciones M2M se realizan a través de redes donde se envían cantidades pequeñas de información entre diferentes nodos y por tanto no se necesita un gran ancho de banda. MQTT dispone de una arquitectura en la que se sigue una topología en estrella donde existe un nodo central que realiza la función de servidor MQTT y se encarga de gestionar la red enviando información a los diferentes clientes. El

servidor más utilizado en la arquitectura MQTT es probablemente el servidor Mosquitto que permite la comunicación inalámbrica entre los diferentes clientes MQTT. Los clientes MQTT pueden ser sistemas embebidos carentes de sistema operativo que utilizan el protocolo MQTT incorporando un software encargado de recoger información de diferentes sensores y enviar dicha información al servidor. De esta manera dichos sistemas se pueden controlar mediante un smartphone o un ordenador con conexión a Internet, lo que permite su control de forma remota. Este hecho unido al abaratamiento y aumento de las prestaciones de los sistemas embebidos, ha despertado un creciente interés de la industria y de los usuarios de Internet en el uso de estos sistemas empotrados en domótica.

1.2.2. Sistema climático Escuela Politécnica Superior de Linares

El nuevo edificio departamental de la Escuela Politécnica Superior de Linares dispone de un sistema de climatización central HVAC (de las siglas en inglés Heating, Ventilating and Air Conditioning) de la compañía Johnson Controls. El sistema dispone de un mando de control individual de la misma compañía, instalado en los diferentes despachos del edifico departamental, concretamente el modelo NS-ATP7002 (Figura 1.2). El mando permite al usuario el apagado/encendido del sistema HVAC así como un ajuste parcial de la temperatura (varios grados por encima o por debajo de un valor predefinido). Transparente para el usuario, un conjunto de conexiones cableadas físicamente accesibles mediante un conector empotrado en la pared permite establecer una comunicación bidireccional. Así, estos mandos individuales están gobernados por un actuador central instalado en la red del sistema climático encargado de recibir las comunicaciones provenientes de cada mando.

La información técnica referente al mando NS-ATP7002 puede ser descargada en Internet en la página web del fabricante [46]. Dos archivos, el primero incorporando la instrucciones de instalación del mando [26] y un segundo incorporando información técnica sobre el protocolo de comunicaciones BACnet Master-Slave/Token-Passing (MS/TP) [27], pueden ser descargados libremente.

Figura 1.2 - Mando Control Sistema Climático

La comunicación entre el mando de control y el actuador central se realiza sobre el protocolo de comunicaciones BACnet MS/TS a través de un bus de comunicaciones denominado SA Bus (Sensor Actuator bus). El SA Bus conforma la interfaz física del protocolo de comunicaciones BACnet MS/TP. El bus de comunicaciones integra la alimentación para el mando de control, siendo el actuador central el encargado de suministrar energía al mando NS-ATP7002 para su correcto funcionamiento.

Este sistema de climatización tal y como está configurado a día de hoy entraña un problema: cada dos horas el sistema se desconecta y es necesario volver a pulsar el botón de encendido/apagado para que éste siga funcionando. Esto supone un inconveniente ya que si no se está en el despacho en el momento en el que el sistema se apaga automáticamente, éste permanecerá apagado. Como consecuencia en invierno cuando se llega al despacho el sistema está apagado y es necesario al menos 1 hora para alcanzar la temperatura prefijada. Los mismos inconvenientes ocurren en los últimos meses de docencia cuando las temperaturas son elevadas y el sistema de climatización a la hora de llegar al despacho igualmente se encuentra apagado.

Teniendo en cuenta los avances en domótica gracias a la utilización de sistemas empotrados y protocolos de comunicación específicos de IoT se ha creído conveniente utilizar un sistema embebido con comunicación inalámbrica y conexión a Internet que permita la automatización del encendido y apagado del mando de control NS-ATP7002. El sistema diseñado incorpora dos sistemas embebidos. El primero de ellos no constaría de sistema operativo y se encargaría del control directo del mando NS-ATP7002. El segundo sistema embebido incorporara sistema operativo así como un servidor web con interfaz de usuario para hacer más sencilla la automatización del apagado y encendido.

1.3. Objetivo. Descripción general. Antecedentes.

El objetivo principal de este trabajo fin de grado es el diseño y construcción de un prototipo que compuesto por dos sistemas embebidos comerciales, unos periféricos y una arquitectura software propia de IoT, sea capaz de controlar individualmente en cada despacho y remotamente el encendido y apagado del sistema de climatización existente en la Escuela Politécnica Superior de Linares.

Al prototipo que se pretende construir para la automatización del sistema de climatización se le exigirá que cumpla una serie de requisitos que a continuación se enumeran.

Desde el punto de vista hardware los requisitos son:

- Los módulos utilizados han de ser comerciales y de reducido tamaño.
- El sistema ha de funcionar sin hacer uso de baterías.
- El dispositivo ha de contar con un conversor analógico digital para realizar lecturas de tensión.
- El dispositivo ha de contar con entradas/salidas digitales.
- Ha de disponer de una interfaz de comunicación inalámbrica.
- Ha de permitir el accionamiento del pulsador del mando de control.
- Ha de contar de un puerto con conexión microUSB para su programación.
- Ha de incorporar un interruptor para apagado/encendido manual del prototipo.

Desde el punto de vista software los requisitos son:

- Ha de disponer de una interfaz de usuario de manera que sea accesible desde dispositivos móviles y dispositivos de escritorio.
- Ha de ser accesible desde Internet.
- Posibilidad de configurar un horario de encendido y apagado del sistema de climatización.
- Posibilidad de trabajar con el protocolo MQTT.
- Posibilidad de configurar el sistema como un cliente MQTT.
- Posibilidad de creación y gestión de bases de datos.
- El software ha de ser capaz de actualizar su reloj interno a través de Internet.

Hasta donde conocemos no existe ninguna alternativa de bajo coste y con acceso remoto a Internet que ofrezca una solución al problema antes comentado relativo a la desconexión automática del sistema climático al cabo de 2 horas de inactividad. La opción

que más se acerca es una solución comercializada por Johnson Controls que utiliza pantallas táctiles y ofrece la posibilidad de automatizar el encendido y apagado [44], [45]. Sin embargo este equipo resulta de elevado coste a la vez que sería necesario cambiar la instalación.

Así, para alcanzar el objetivo que en este trabajo fin de grado se plantea se ha utilizado en primer lugar un sistema embebido con comunicación inalámbrica WiFi de tal manera que éste sea accesible de forma remota y permita el encendido/apagado de la climatización de forma automática. Dentro de los sistemas embebidos se ha creído conveniente utilizar el sistema de desarrollo NodeMCU. Las razones por las que se ha utilizado este sistema han sido principalmente su conectividad inalámbrica, su pequeño tamaño y su bajo coste. NodeMCU también dispone de un conversor analógico digital que ha sido utilizado para la monitorización del led incorporado en el mando de control de la climatización. De esta manera es posible si el mando está apagado o encendido.

Al sistema de desarrollo NodeMCU se le ha incorporado un periférico para la apertura y cierre del interruptor de encendido, así como un sistema de alimentación. El sistema de apertura y cierre consiste en un circuito integrado comercial incorpora un relé. Exactamente el módulo utilizado ha sido el Latching Relay Featherwing del fabricante Adafruit [43]. Por último para alimentar el sistema de desarrollo NodeMCU se ha utilizado la alimentación incorporada en el SA bus del mando de control como se explicará más adelante en detalle. Cabe mencionar que la alimentación que proporciona dicho bus es de 16 V y para realizar una adaptación se ha utilizado un circuito integrado DC/DC, el circuito es un conversor DC/DC LM2596. Es necesario adaptar la tensión del bus debido a que el sistema de desarrollo NodeMCU trabaja con una tensión comprendida entre los 3.3 V y los 5 V. A su vez, el sistema de desarrollo NodeMCU proporciona alimentación al sistema electrónico de apertura y cierre.

Para proporcionarle al prototipo una conectividad a Internet y debido a la imposibilidad de acceder a la red Eduroam mediante el sistema embebido NodeMCU, se ha utilizado un sistema embebido Raspberry Pi [35]. El sistema Raspberry Pi funciona como un punto a acceso WiFi al cual se conecta el sistema de desarrollo NodeMCU. Raspberry Pi se conecta a Internet a través de la red cableada disponible en la Escuela Politécnica Superior de Linares.

Como interfaz de usuario (figura 1.3) y pensada para hacer más sencilla la automatización de la apertura y cierre se ha diseñado una aplicación web alojada en la memoria del sistema Raspberry Pi. Para acceder a la aplicación desde fuera de la red local de la universidad se ha instalado también una aplicación encargada de abrir un túnel seguro que da acceso a la aplicación web. Ngrok es el nombre de la aplicación encargada de generar el túnel seguro y sus características serán explicadas en detalle en la sección

3.9 de ésta memoria. Además, en el sistema Raspberry Pi se ha instalado un servidor MQTT que es el encargado de gestionar los mensajes provenientes de la aplicación web a través de la red WiFi y enviarlos al NodeMCU.

Con el prototipo construido el usuario es capaz de programar el encendido y apagado del sistema de climatización con la ayuda de un smartphone u ordenador sin la necesidad de estar presente en las dependencias de la Escuela Politécnica. Este prototipo se ha instalado en uno de los despachos del edificio departamental y se ha probado con éxito.

En los siguientes capítulos se explicará con más detalle el hardware y el software utilizado en la implementación del citado prototipo.

Figura 1.3 - Interfaz principal aplicación web diseñada

2. Capítulo 2: Diseño del hardware

Para una mejor descripción del proyecto se ha creído conveniente explicar en primer lugar el hardware utilizado en la construcción del prototipo que permite la automatización encendido y apagado del sistema de climatización descrito con anterioridad. Tras este capítulo, en el capítulo 3 de esta memoria se describirá el software utilizado.

2.1. Descripción del sistema propuesto

El sistema diseñado en este trabajo fin de grado está compuesto por la unión de una serie de circuitos integrados comerciales existentes en el mercado. El objetivo principal del prototipo montado es el control del sistema de climatización individual instalado en uno de los despachos del edificio departamental de la Escuela Politécnica Superior de Linares.

En la figura 2.1 se muestra un diagrama del prototipo implementado. El sistema a nivel hardware consta básicamente de 2 sistemas embebidos: así, el sistema de desarrollo NodeMCU es el encargado junto con unos periféricos que se describirán a continuación de realizar un control directo sobre el mando NS-ATP7002. El NodeMCU se comunica vía WiFi con el segundo sistema embebido, el Raspberry Pi. Este dispositivo proporciona conectividad a Internet a la vez que incorpora un servidor web. Entre los periféricos que acompaña al NodeMCU se encuentra un sistema de alimentación con el que proporcionar corriente continua (DC) al dispositivo. Mediante el uso de un conversor DC/DC éste sistema de alimentación se encargará de adaptar la tensión proveniente del bus de comunicaciones al cual se encuentra conectado el mando de control individual del sistema de climatización. La tensión de alimentación del mando de control NS-ATP7002 es de 16 V con una corriente pico de 240 mA. Esta tensión de 16 V será reducida a 4.3 V para alimentar el módulo NodeMCU. Puesto que el módulo NodeMCU necesita una tensión mínima de 3.3 V para su correcto funcionamiento se ha creído conveniente proporcionar una tensión de alimentación mayor a la mínima permitida. Para reducir la tensión de 16 V a la tensión de 4.3 V se ha actuado sobre el potenciómetro que el conversor DC/DC dispone para ajustar la tensión de salida con precisión.

El sistema de desarrollo NodeMCU se encargará de realizar la comunicación a través de una red WiFi con un servidor de mensajería MQTT previamente instalado en el sistema embebido Raspberry Pi. La comunicación entre NodeMCU y el servidor MQTT se realiza a través de un programa previamente almacenado en la memoria del NodeMCU. Este programa se basa en la recepción de mensajes y su posterior decodificación utilizando el protocolo MQTT. Así mismo, el módulo NodeMCU se encargará de alimentar y controlar

un circuito integrado compuesto por un interruptor electrónico basado en un relé de enclavamiento de pequeñas dimensiones. Este circuito integrado es un sistema electrónico de apertura y cierre el cual actuará sobre el mando de control de la climatización conmutando los terminales del relé. De esta forma se realizará un *bypass* entre los terminales del pulsador incorporado en el mando.

Los tres dispositivos mencionados anteriormente se instalarán dentro de una caja diseñada específicamente para este prototipo, la cual ha sido construida con una impresora 3D.

Por último, el segundo de los sistemas embebidos utilizados el sistema Raspberry Pi, ha de ser conectado a la red cableada disponible en alguno de los despachos del edifico departamental o en alguna estancia que disponga de un puerto RJ45 con conexión a Internet. Este sistema almacena en su memoria un software encargado de gestionar la red inalámbrica WiFi. También incluye un servidor web utilizado para el alojamiento de la aplicación web empleada como interfaz de usuario que se explicará en detalle más adelante. Finalmente, el sistema Raspberry Pi incorporará un software encargado de realizar una pasarela entre la aplicación web e Internet, para así, poder controlar el sistema de climatización desde cualquier parte de la red externa.

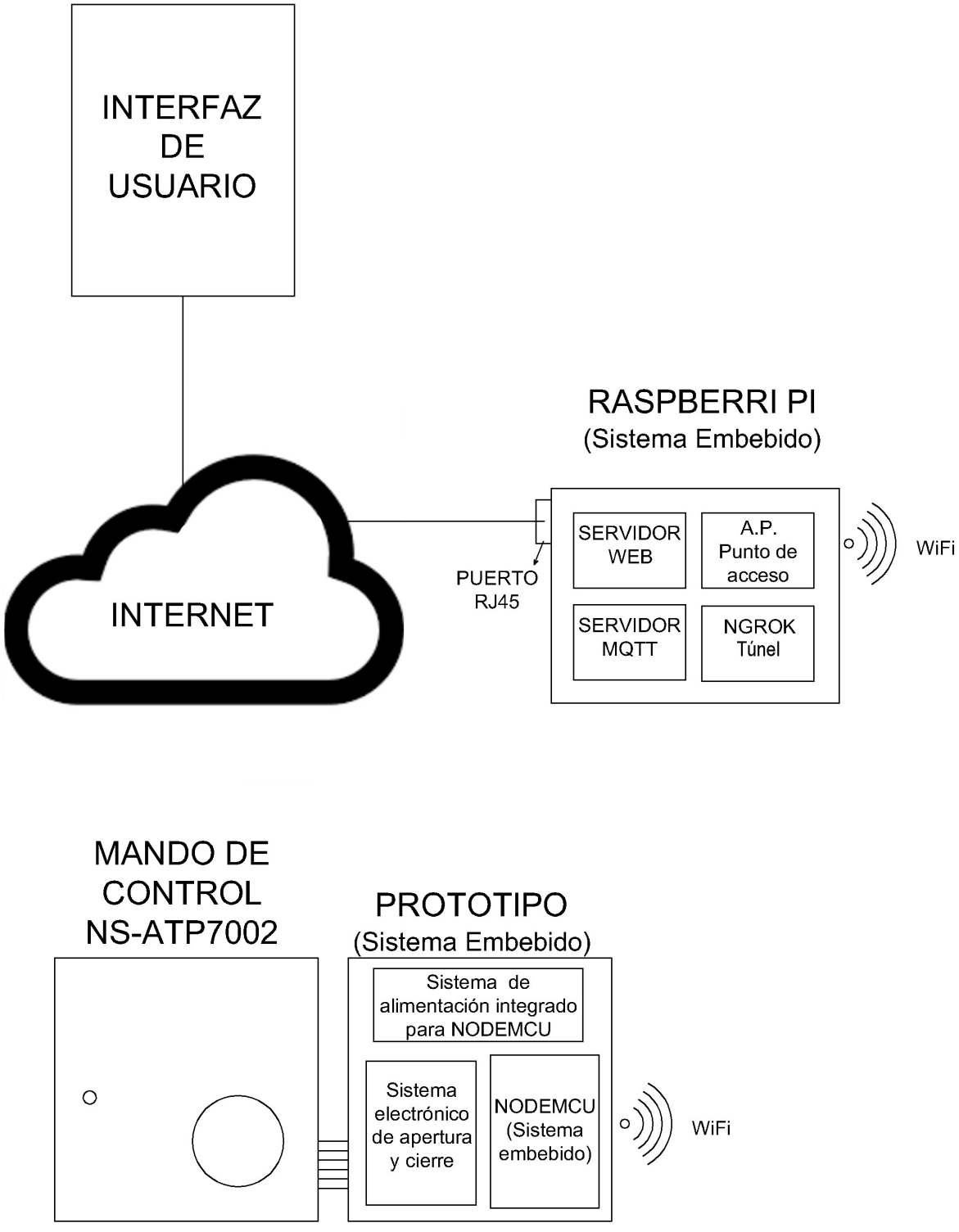


Figura 2.1 - Esquema general del prototipo implementado

2.1.1. Condicionantes previos – Sistema de Control Johnson Controls

El mando individual de la climatización montado en cada despacho de la Escuela Politécnica Superior de Linares corresponde al modelo NS-ATP7002 de Johnson Controls. En las hojas de especificaciones del fabricante este dispositivo es también llamado sensor de zona electrónico, dado que lleva incorporado un sensor de temperatura con el cual monitoriza la temperatura de cada despacho. Este mando está diseñado para funcionar con el protocolo *BACnet* Master-Slave / Token-Passing (MS/TP) utilizando un bus físico denominado SA Bus (del término en inglés *Sensor Actuator Bus*) característico de los sistemas HVAC de Johnson Controls.

Dada la poca información existente sobre los sensores NS-ATP y la negativa de los servicios de mantenimiento de la Escuela Politécnica Superior de Linares a proporcionar información, se ha tenido que llevar a cabo un proceso de ingeniería inversa para determinar el funcionamiento del mando de control.

Tras desmontar el mando de uno de los despachos del edificio departamental se tiene un conector de 4 pines empotrado en la pared tal y como aparece en la figura 2.2.

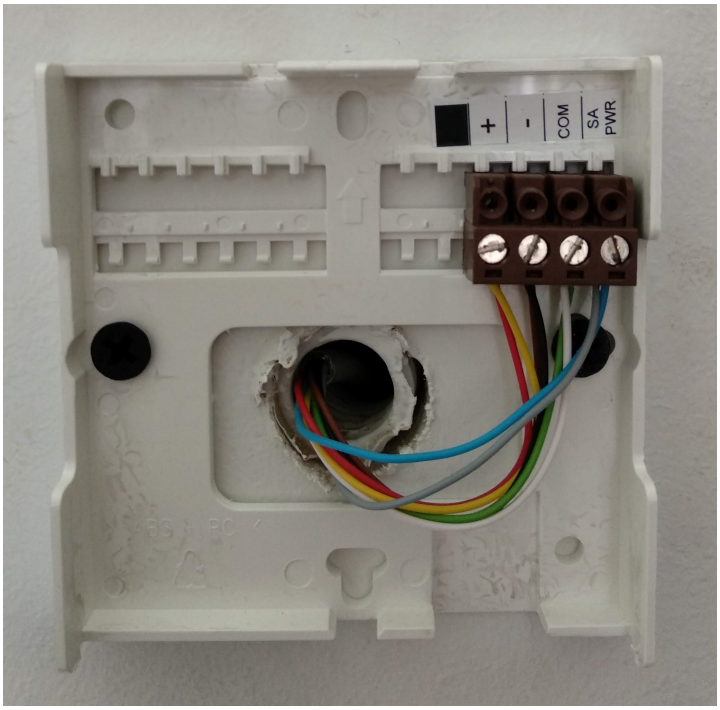


Figura 2.2 - Conector mando de control NS-ATP7002

Como se puede apreciar el conector del mando de control cuenta con los siguientes pines: pin +, pin -, pin COM y por último el pin SA PWR. Estos pines son utilizados para establecer una comunicación con el actuador central, así como para alimentar la circuitería que el propio mando incorpora. Esta información parcialmente se encuentra disponible en

los manuales antes comentados [26], [27] y experimentalmente se ha comprobado, como se explicará en el capítulo 4 de esta memoria.

Otro aspecto importante que ha condicionado el diseño del prototipo implementado ha sido las limitaciones del SA Bus relativas a la corriente máxima capaz de proporcionar. Como se puede observar en la figura 2.3 la corriente máxima es de 240 mA y si se excede esta corriente se pueden causar problemas de comunicación en la instalación o provocar una desconexión de los dispositivos conectados al bus.

SA Bus Device Limits

The SA Bus is limited to 10 devices total to ensure good communication on the bus and is limited to four NS sensors because only four unique addresses can be set on the sensors. Due to change of value (COV) limitations, it is best to limit the number of IOMs on the SA Bus to four. The SA bus is also limited by the total power consumption of the devices connected on the bus. Exceeding the total power consumption limit can cause poor bus communication and cause devices to go offline.

Table 8 provides the power consumption of devices commonly connected to the SA bus.

Important: The total power consumption for the SA Bus is limited to 210 mA on the SA Bus modular jack and 240 mA on the SA Bus terminal block. Exceeding the total power consumption limit can cause poor bus communication and cause devices to go offline.

Table 8: Power Consumption by Common SA Bus Devices

SA Bus Device	Power Consumption on the SA Bus
Discharge Air Sensors (NS-DTN70x3-0)	12 mA
Balancing Sensors (NS-ATV7003-0)	25 mA
Network Sensors without display	13 mA
Network Sensors with display no RH	21 mA
Network Sensors with display and RH	27 mA
CO2 Network Sensors (NS-BCN7004-0)	28 mA (non-isolated) or 5 mA (isolated)
ZFR1811 Wireless Field Bus Router	90 mA
ZFR1812 Wireless Field Bus Router	90 mA
DIS1710 Local Controller Display	90 mA - may be a temporary load

MS/TP Communications Bus Technical Bulletin 28

Figura 2.3 - Limitaciones del Bus de comunicaciones [27]

Como se describirá en el capítulo 4 de esta memoria se ha llegado a la conclusión de que entre los pines COM y SA PWR existe una tensión de 16 V que puede proporcionar suficiente corriente para alimentar el prototipo diseñado en este proyecto.

A la vista de la información disponible y después de los ensayos realizados encaminados a conocer el funcionamiento del mando de control NS-ATP7002, se ha decidido intervenir eléctricamente en referido componente para de este modo poder controlarlo de forma remota. Dicha intervención ha consistido en realizar básicamente 3 conexiones:

1. Circuito de *bypass* del interruptor de encendido y apagado. Aprovechando la buena accesibilidad del interruptor de encendido/apagado (Figura 2.4) junto con la poca corriente que pasa a través de él (94,6 µA), se ha decidido realizar un

bypass para de este modo proceder de forma externa al apagado o encendido del módulo. Este *bypass* no inhabilita el encendido o apagado manual el cual sigue operativo, pero ofrece la posibilidad de hacerlo funcionar de forma automática mediante un sistema electrónico de apertura y cierre (relé SSR, optoacoplador, etc).

Figura 2.4 – Bypass realizado sobre el pulsador de encendido/apagado del mando NS-ATP7002

2. Alimentación del prototipo implementado. Para conseguir alimentar el prototipo se han aprovechado las conexiones existentes en el mando de control a través de las cuales este dispositivo se conecta al bus de comunicaciones. En la figura 2.5 se muestran las conexiones utilizadas por el mando de control para recibir alimentación y para comunicarse con el actuador central.

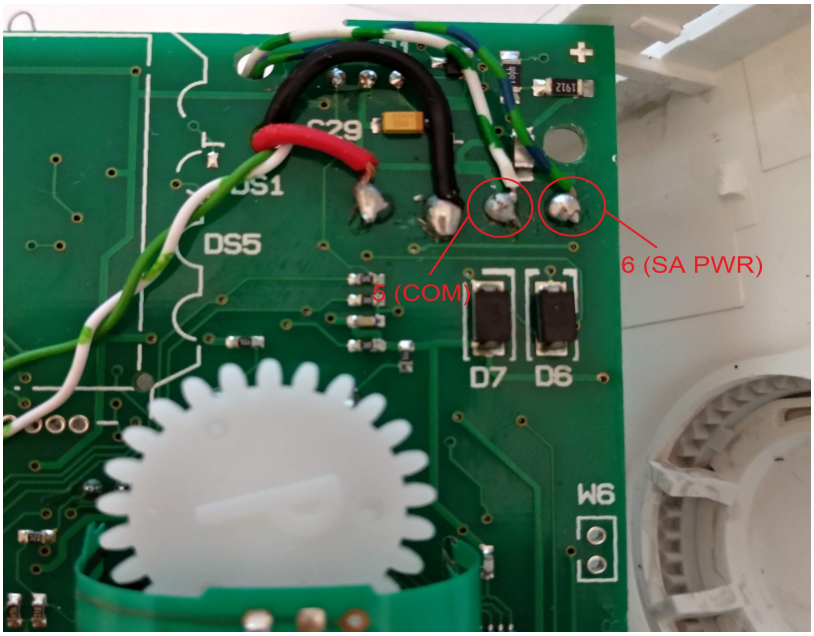


Figura 2.5 - Conexiones existentes en el mando de control NS-ATP7002

3. Verificación del estado (encendido/apagado) del sistema. Para poder verificar el estado del sistema de climatización se ha procedido a monitorizar la tensión entre los terminales del led incorporado en el mando de control. En la figura 2.6 se muestran los terminales del led. En el caso de que el sistema de climatización esté encendido la tensión en extremos del led es de 1.8 V así, esta tensión se monitoriza a través de la entrada analógica del módulo NodeMCU como se comentará más adelante.

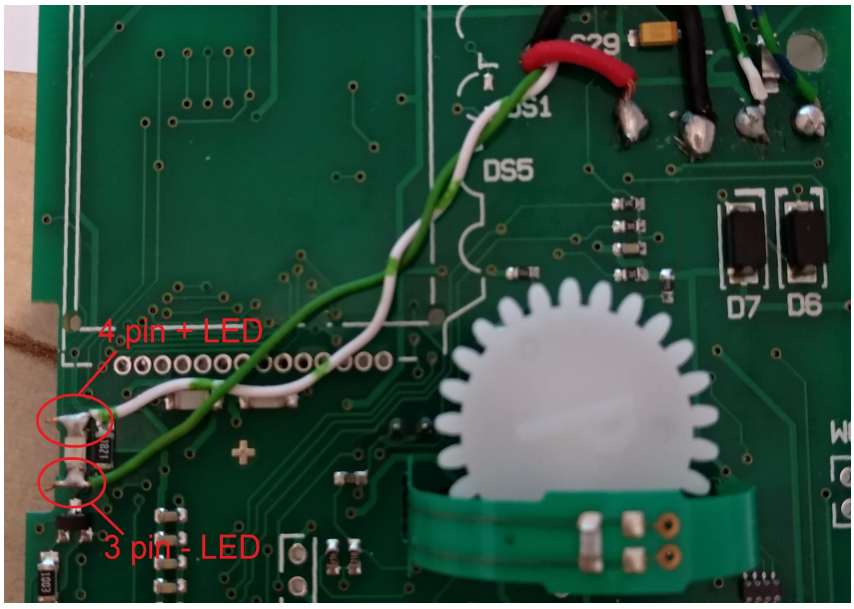


Figura 2.6 - Conexiones terminales LED

Para adaptar el mando de control de la climatización y para un mejor acceso a los terminales y conexiones anteriormente citados se ha añadido un tira de 6 pines a la caja del mando (Figura 2.7 y Figura 2.8).

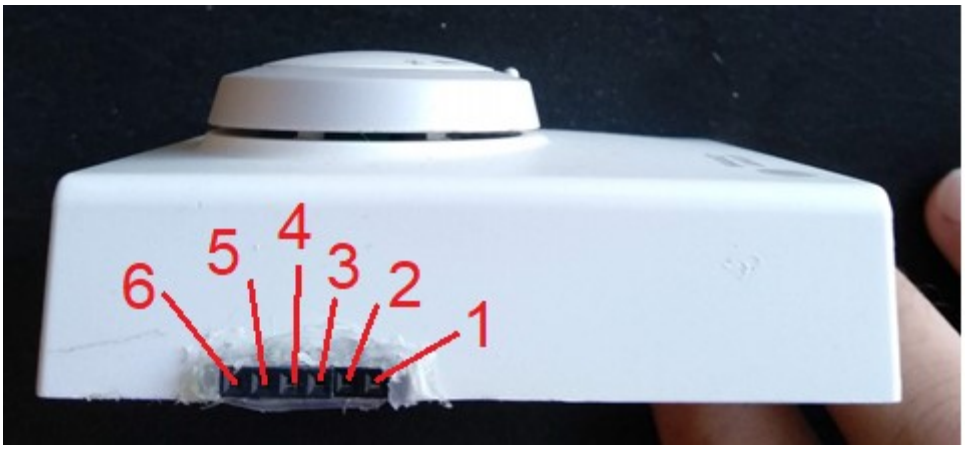


Figura 2.7 - Tira 6 Pines Mando Climatización (Exterior)

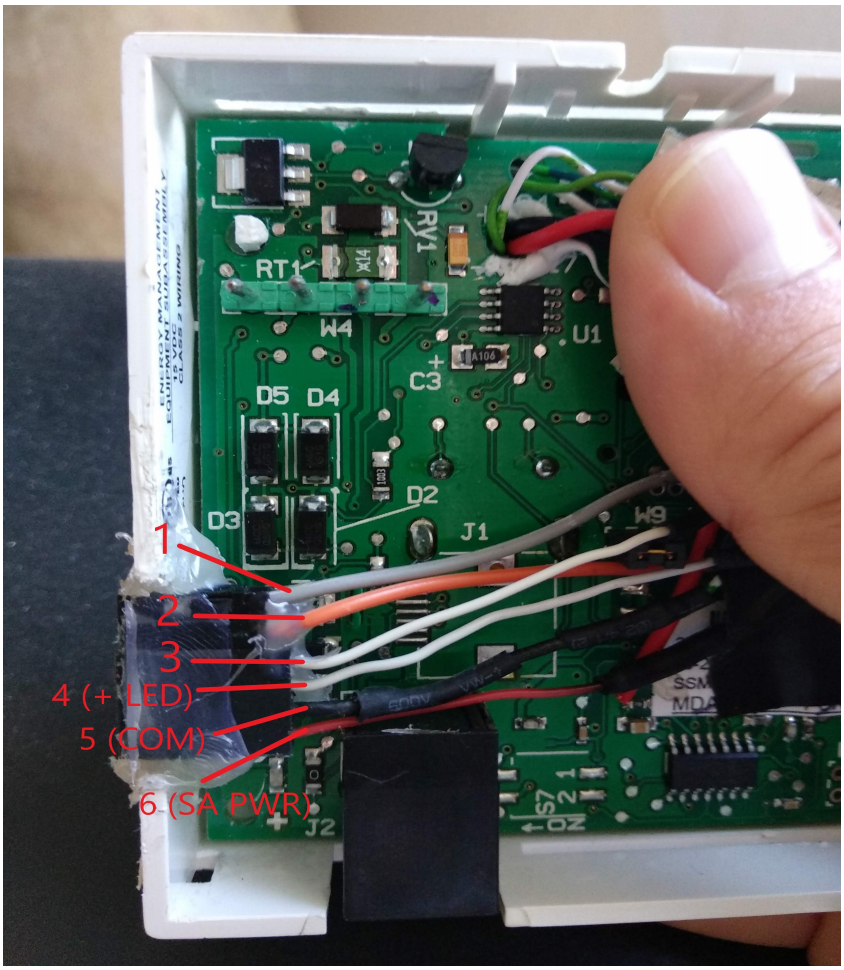


Figura 2.8 - Tira 6 Pines Mando Climatización (Interior)

La funcionalidad de los pines que aparecen en la figura 2.7 y 2.8 que se han añadidos al mando de control es la siguiente:

- Los pines 1 y 2 están conectados a los extremos del pulsador incorporado en el mando de control y se utilizan para realizar el *bypass* antes comentado a través sistema electrónico de apertura y cierre.
- Los pines 3 y 4 están conectados a los terminales del led incorporado en el mando de control. El pin 3 está unido al cátodo y el pin 4 al ánodo del diodo led. Aunque se dispone de los dos terminales (añado y cátodo) solo el pin 4 está en uso y conectado a la entrada del conversor analógico digital del NodeMCU. Dicho sistema de desarrollo no incorpora entradas diferenciales en su conversor analógico digital, en su defecto todas las entradas están referenciadas a la tensión negativa de alimentación. Esta tensión de alimentación negativa como se verá más adelante, es común tanto para el mando de control como para el NodeMCU.
- Los pines 5 y 6 señalados en la figura 2.7 y 2.8 son los encargados de proporcionar alimentación al prototipo montado. El pin 5 está unido al terminal COM del mando de control y realiza la función de tierra de alimentación a la vez que de referencia para el conversor analógico digital con el que se monitoriza la tensión en extremos del led. A su vez, el pin 6 está conectado al terminal SA PWR del mando de control.

.

2.2. Partes constitutivas del prototipo

2.2.1. Sistema de alimentación – Conversor DC/DC

Con la utilización de un conversor DC/DC se ha conseguido alcanzar unos de los requerimientos de este proyecto que era no trabajar con baterías. Al existir una tensión de alimentación DC en los terminales de conexión del mando de control cabe la posibilidad de utilizar dos tipos de circuitos integrados a la hora de reducir la tensión de 16 V a una tensión adecuada para alimentar el NodeMCU. La primera de las opciones seria utilizar un regulador lineal (por ejemplo, un integrado lineal LM7805), la segunda (que ha sido la elegida) consistiría en utilizar un conversor DC/DC. El conversor DC/DC ofrece una ventaja clave para el diseño del prototipo con respecto al conversor lineal. Dicha ventaja es su alta eficiencia, como se verá en el capítulo 4 esto permite que la corriente que se demanda al SA Bus sea mínima y pase inadvertida si se compara con la corriente máxima de 240 mA que es capaz de suministrar el actuador central del sistema de climatización.

El conversor utilizado en este proyecto es un conversor DC/DC reductor no aislado que incorpora un potenciómetro de precisión para ajustar la tensión de salida desde los 16 V hasta los 4.3 V utilizados para alimentar el sistema de desarrollo NodeMCU. No aislado quiere decir que el terminal de tierra de entrada está unido internamente con el terminal de tierra de salida del conversor DC/DC.

Las partes constitutivas más importantes del conversor DC/DC son las siguientes:

- El componente principal es el integrado LM2596 quien se encarga de regular la tensión y es capaz de conducir una corriente máxima de 3 A.
- Un condensador de entrada necesario para estabilizar el rizado de la señal de entrada y un condensador de salida que estabiliza el rizado de la tensión a la salida del conversor.
- Un diodo rectificador tipo *Schottky* utilizado para permitir el flujo de corriente en directa e impedirla en inversa.
- También incorpora una bobina utilizada para almacenar energía y que actúa de filtro junto con el condensador de salida para de esta manera minimizar el rizado.
- Por último dispone de un potenciómetro de precisión para ajustar la tensión de salida.

En la figura 2.9 y 2.10 se muestran las partes constitutivas del conversor DC/DC utilizado.

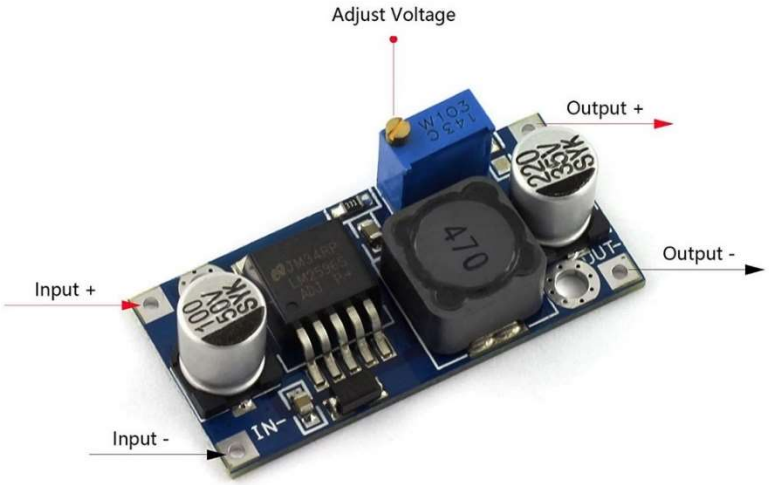


Figura 2.9 - Conversor DC-DC LM2596

Los pines de entrada, input (+) e input (-), se conectarán a los pines COM y SA PWR de mando de control Johnson Controls NS-APT7002. Por otro lado, los pines de salida, output (+) e output (-) se conectan a los terminales de alimentación de la placa de desarrollo NodeMCU.

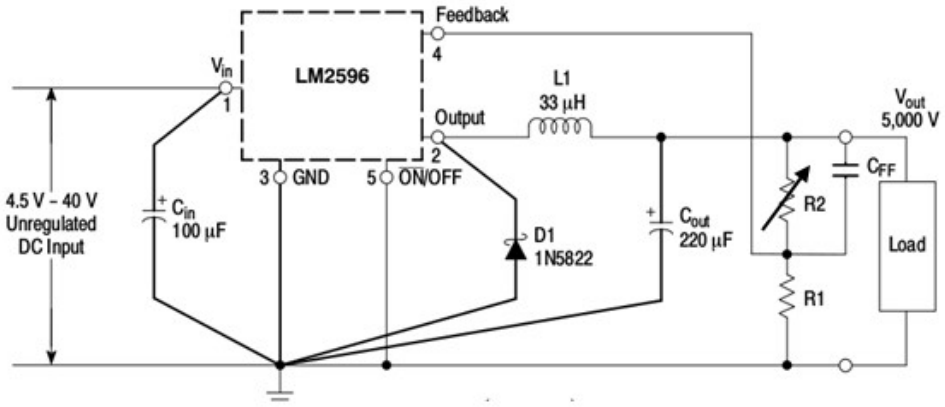


Figura 2.10 -Esquema partes constitutivas conversor DC/DC

Principales Características Técnicas:

- Voltaje de entrada: 4.5 V – 40 V (DC)
- Voltaje salida: 1.25 V – 35 V (DC)
- Corriente de salida: 3 A (máximo).
- Potencia de salida: 50 – 70 W
- Eficiencia de conversión: 92% (máxima)
- Frecuencia de trabajo: 150 kHz

- Temperatura de trabajo: -40ºC - +85ºC

2.2.2. NodeMCU

El propósito general de este sistema embebido NodeMCU en el proyecto que nos ocupa es realizar la comunicación vía WiFi con el servidor MQTT y almacenar el *schetch* desarrollado para que a través de las salidas/entradas del módulo se actué sobre un sistema electrónico de apertura y cierre y se automatice el enciendo o apagado del sistema de climatización. Se ha optado por la utilización de este sistema embebido y no un Arduino UNO con una shield WiFI (por ejemplo) por las siguientes razones:

- El módulo NodeMCU lleva integrada la interfaz inalámbrica WiFi.
- Es un sistema de reducido tamaño.
- Por su bajo consumo (según especificaciones técnicas del fabricante).
- Logística (se dispone de este módulo) y lo más importante contrastada experiencia tras exitosos proyectos realizados con anterioridad.

NodeMCU [22] es una placa o kit de desarrollo totalmente abierta, a nivel de hardware y de software, además de ser un dispositivo de bajo coste. El kit lleva incorporado un módulo ESP-12E el cual reúne la memoria Flash para almacenar los programas o *sketchs* y la antena WiFi. Dentro del módulo ESP-12E se encuentra un chip ESP8266 que se suele llamar SoC (del término en inglés *System on a Chip*) el cual incorpora en su interior un microcontrolador o MCU. Este módulo es el encargado de facilitar el acceso a los pines y demás conectores del SoC, además de los pines del microcontrolador. Internamente los pines del módulo ESP-12E están cableados hasta los pines del SoC ESP8266. El microcontrolador integrado dentro del ESP8266 se conoce como Tensilica L-106 de 32-bit y es el encargado de gestionar todas las entradas, salidas y cálculos necesarios para hacer funcionar los programas cargados en su memoria.

A continuación en la figura 2.10 se muestra un esquema general de este tipo de kit de desarrollo.

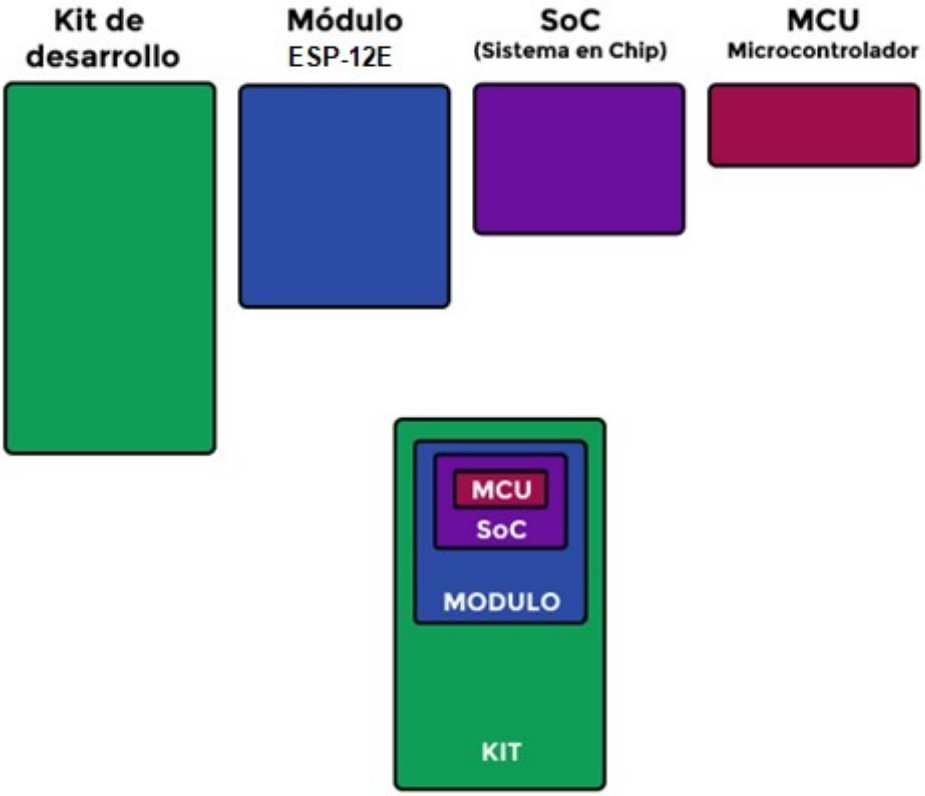


Figura 2.11 - Esquema General Kit de Desarrollo [22]

Principales Características:

- Consumo de Corriente 10 microA – 170 mA
- Módulo ESP-12E
- SoC ESP8266
- Procesador Tensilica L106 32-bit
- Frecuencia Procesador: 80MHz (160MHz máx.)
- Memoria Flash 4 MByte
- Memoria RAM hasta 128 MByte
- Chip USB CH340G (convertidor TTL a USB)
- Conectividad WiFi 2.4 GHz 802.11 b/g/n
- Soporta Seguridad WPA y WPA2
- Soporta tres modos de operación:
    - STA – Station (Cliente)
    - AP – Acces Point (Punto de Acceso)
    - STA+AP – Station + Acces Point

- Protocolo TCP/IP integrado
- 13 puertos GPIO, 3.3 V, 40mA máx.
- Pines D0 – D8, S1 – S3 pueden ser usados como GPIO, PWM, I2C
- 1 entrada analógica ADC (Analog Digital Converter) de 10 bits (pin A0)
- 3 pines de 3.3 V
- 5 pines de GND (Ground)
- 1 pin para alimentación externa (Vin)
- Conector MicroUSB integrado

Otro factor importante son los puertos GPIO de los cuales dispone y el conversor analógico digital utilizado para la monitorización del led incorporado en el mando de control. El puerto GPIO04 (D2) y el puerto GPIO14 (D5) se utilizarán para conectarlos a un sistema electrónico de apertura y cierre que forma parte del prototipo como se describirá más adelante. En la figura 2.12 se puede observar la distribución de los pines disponibles en el NodeMCU.

El NodeMCU se alimentará mediante las patillas “Vin” y “GND” a través de dos cables provenientes del conversor DC-DC y este a su vez proporciona alimentación al sistema electrónico de apertura y cierre a través de los pines GND y 3.3V.

La entrada analógica (pin A0) del módulo es la encargada de monitorizar la tensión del led existente en el mando de climatización. Para monitorizar la tensión se conecta el cable positivo proveniente del led incorporado en el mando de control NS-ATP7002 al pin A0 del módulo NodeMCU. La monitorización del led se utiliza para conocer en todo momento el estado de la climatización (encendida o apagada). Puesto que la tensión existente entre el pin positivo del led del mando de control y el pin de tierra (GND) en estado apagado es de 5 voltios, y la tensión máxima admitida por el pin analógico es la tensión de alimentación, en este caso 4.3 voltios. Se ha realizado un circuito divisor de tensión para reducir la tensión a la entrada del pin ADC ya que no es recomendable que ésta sea mayor a la admitida (4,3 voltios).

En el plano número 1 se pueden ver con más detalle las conexiones realizadas entre los distintos componentes del prototipo.

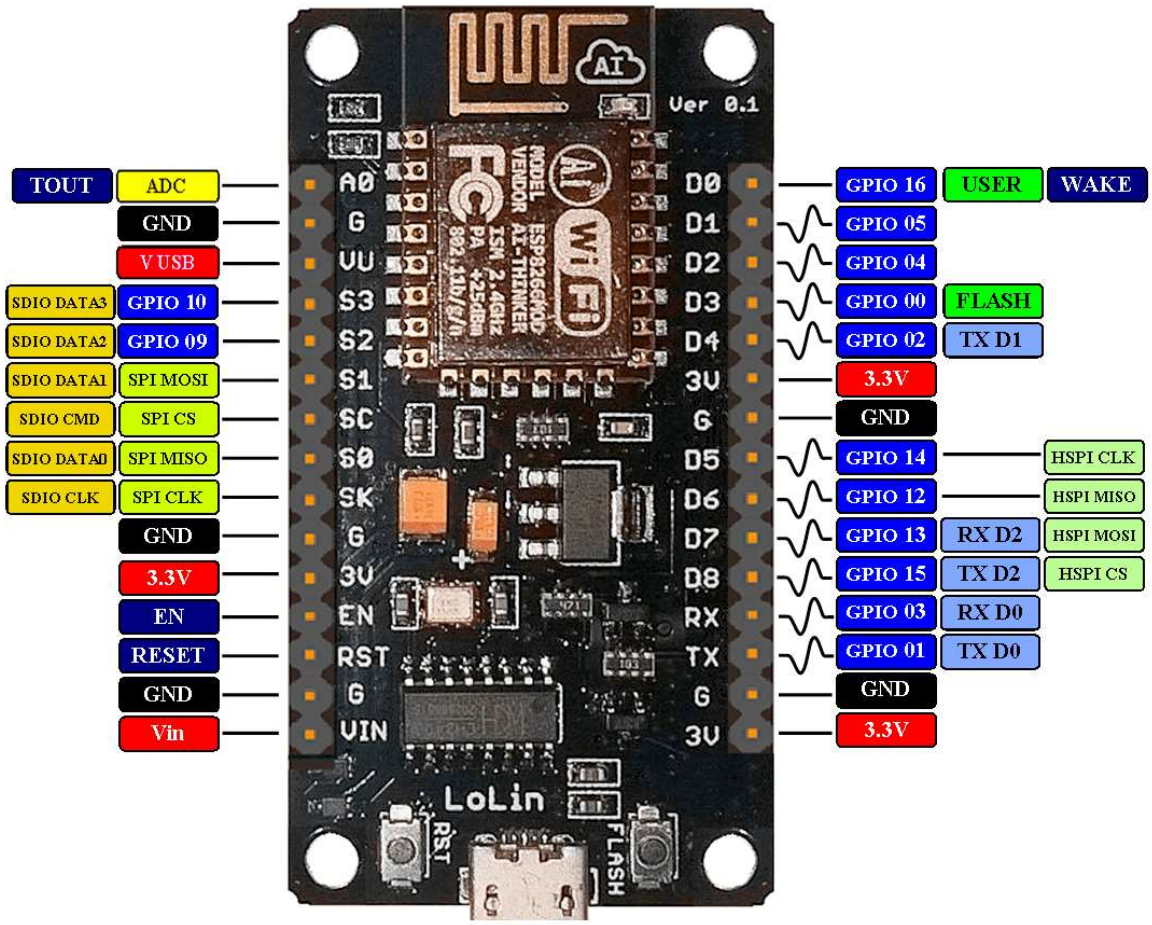


Figura 2.12 - Distribución de los pines disponibles en el NodeMCU [41]

2.2.3. Sistema electrónico de apertura y cierre

El sistema electrónico de apertura y cierre utilizado en el proyecto que nos ocupa es un módulo comercial de reducido tamaño que integra un relé capaz de conmutar la corriente y la tensión en extremos del pulsador incorporado en el mando de control NS-ATP7002.

Para realizar el *bypass* entre los terminales del pulsador del mando de control se ha utilizado un sistema de la marca Adafruit [24] modelo *FeatherWing*. Esta plataforma dispone de dos modelos de sistemas *FeatherWing*, el primero de ellos incorpora un relé de enclavamiento (*Latching*) y el segundo incorpora un relé sin enclavamiento (*Non-Latching*). El módulo que incorpora el relé sin enclavamiento dispone de un pin SET, un pin COM que está mecánicamente conectado al pin NC (*Normally Close*) y un pin NO (*Normally Open*) el cual se encuentra desconectado. Cuando se activa el pin SET mediante un pulso de tensión alto, el relé conmuta sus salidas y el pin COM se conecta con el pin NO, además se desconecta el pin NC. Cuando el relé está activo se utilizan alrededor de 50 mA para

mantener la bobina encendida y hay que tener en cuenta que cuando se pierde la alimentación o se deja de excitar el pin SET el relé volverá al estado abierto.

El segundo modelo y el utilizado en este trabajo fin de grado tiene integrado un relé de enclavamiento (Figura 2.12). Este módulo necesita utilizar dos pines para abrir o cerrar los contactos del relé, pero a diferencia del primer módulo este consume menos energía. Los dos pines son SET y UNSET. Cuando se activa el pin SET enviando un pulso alto de tensión durante 10 milisegundos, el relé conmuta sus salidas y se conecta el pin COM con el pin NO quedándose estos contactos cerrados. Aquí es donde se conectan terminales del pulsador del mando de control de la climatización (entre el pin COM y el NO). Cuando se activa el pin UNSET mediante un pulso alto durante 10 milisegundos, el relé conmuta de nuevo para así abrir sus contactos.

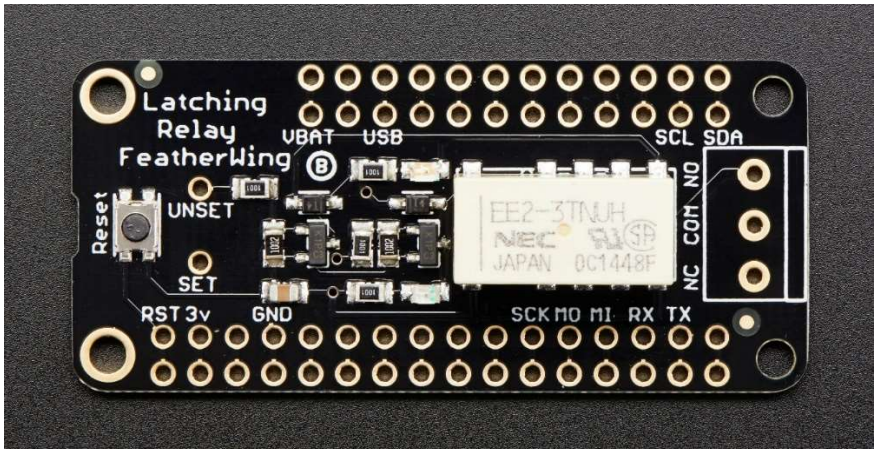


Figura 2.13 - Relé de Enclavamiento FeatherWing de Adafruit [24]

En este proyecto se ha utilizado el relé de enclavamiento debido a que las salidas digitales del módulo NodeMCU no pueden proporcionar una corriente de salida de 50 mA. Mediante la utilización del relé de enclavamiento solo se tendría que mandar un pulso alto durante 10 milisegundos para que se excite la bobina y el relé conmute sus salidas para de este modo activar el sistema de climatización.

Mediante el pin D2 (GPIO 04) del módulo NodeMCU actuamos sobre el pin SET del relé, para que este cierre sus contactos y estos realicen el *bypass* en extremos del pulsador del mando de la climatización. A través de esta acción los contactos del relé permanecen cerrados, por lo que mediante el pin D5 (GPIO 14) actuamos sobre el pin UNSET del relé para que éste vuelva a poner los contactos en abierto (NO – Normally Open). De esta manera realizamos la función de pulsador y no de interruptor.

2.2.4. Raspberry Pi

En este proyecto se ha utilizada un sistema embebido Raspberry Pi 3 Model B (Figura 2.13), este es el primer modelo de Raspberry Pi de tercera generación que incorpora conectividad inalámbrica WiFi y Bluetooth. Una Raspberry Pi según Raspberry Pi Fundation [25] “es un pequeño y asequible ordenador que puede usarse para aprender a programar”. Está destinado principalmente al desarrollo de pequeños prototipos y proyectos de IoT. Se ha elegido el modelo 3 B debido a que dispone de conectividad inalámbrica sin la necesidad de conectar ningún dispositivo externo para la creación de una red inalámbrica WiFi. También se ha decantado por la utilización de este sistema embebido ya que es un dispositivo de pequeñas dimensiones y bajo consumo comparado por ejemplo con un ordenador personal. Este sistema permite la instalación de software de código abierto necesario para la creación de los diferentes servidores utilizados en este proyecto. Este software se detallará en el capítulo 3 de esta memoria.

Especificaciones Técnicas:

- Procesador
    - Chipset Broadcom BCM2387
    - ARM Cortex-A53 1.2 GHz de cuatro núcleos
- GPU: Broadcom VideoCore IV Dual Core 400MHz
- RAM: 1 GB LPDDR (900 MHZ)
- Conectividad:
    - Ethernet 10/100 Base T
    - WiFi 802.11 b/g/n 2.4 GHz
    - Bluetooth 4.1 (Classic / Bluetooth Low Energy)
- Almacenamiento: microSD
- Puertos:
    - HDMI
    - 4 x USB 2.0
    - Ethernet
    - Conector Cámara (Camara Serial Interface – CSI)
    - Conector de la interfaz Serial (Display Serial Interface - DSI)
    - Jack audio – video
- GPIO (General Purpose Input / Output): 40 pines

Figura 2.14 - Raspberry Pi Model 3 B

El sistema Raspberry Pi tiene instalado un sistema operativo basado en GNU/Linux llamado Raspbian. En el Anexo 3 se puede ver en detalle el proceso de instalación del Sistema Operativo. También almacena en su memoria una infraestructura LAMP (Linux, Apache, MySQL y PHP), un servidor MQTT denominado Mosquitto, un túnel (Ngrok) para acceder a la aplicación web desde cualquier parte de la red de internet y un punto de acceso WiFi utilizado para la comunicación entre el módulo NodeMCU y el servidor MQTT.

En el apartado de software se explicará en detalle todo el funcionamiento de la aplicación web y de los distintos programas utilizados en el proyecto.

3. Capítulo 3: Diseño del software

En este capítulo se describirá el software utilizado para llevar a cabo este trabajo fin de grado.

Se ha decidido implementar una red MQTT que trabaje junto a una red HTTP. En la red MQTT el sistema NodeMCU trabaja como un cliente MQTT que recibe mensajes desde el servidor MQTT a través de la red WiFi. En la red HTTP hay una arquitectura cliente-servidor en la cual se encuentra instalada una infraestructura LAMP con un servidor web Apache encargado de atender las peticiones provenientes de los diferentes clientes web. En la figura 3.1 se muestra un diagrama de la arquitectura de redes implementadas.

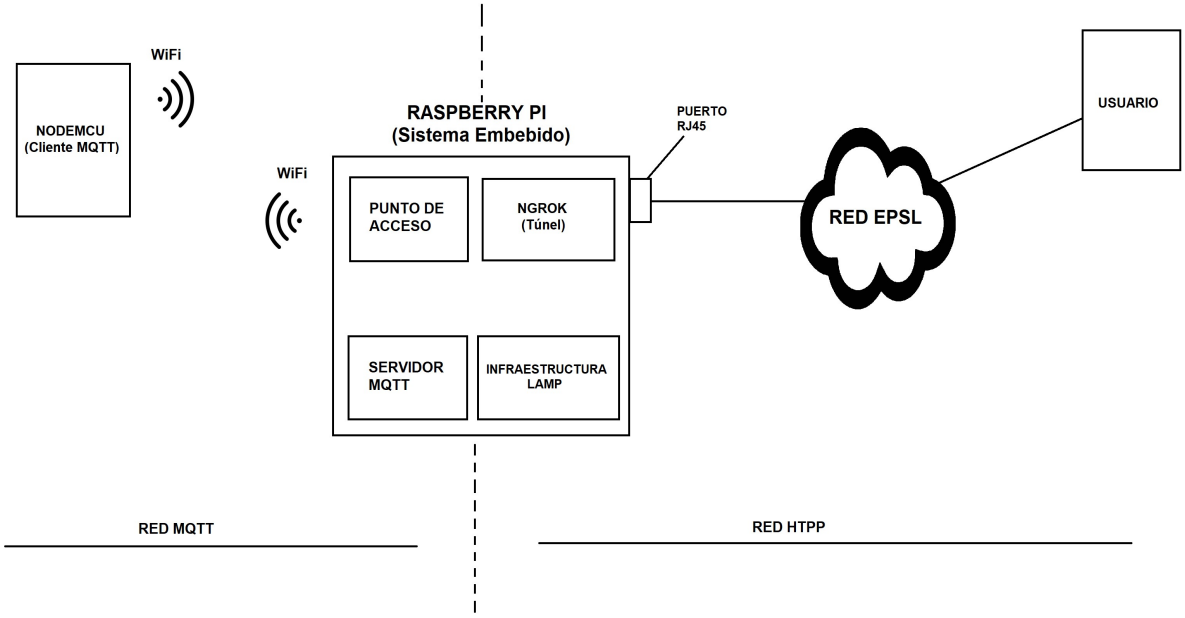


Figura 3.1 - Diagrama de la red implementada en el proyecto

En las siguientes secciones se detallarán las diferentes aplicaciones software utilizadas en el proyecto así como la aplicación web diseñada y el programa principal del sistema NodeMCU.

3.1 Aplicación Web – Interfaz de Usuario

Como interfaz de usuario y pensada para hacer más sencilla la automatización del encendido/apagado del sistema de climatización en uno de los despachos de la Escuela Politécnica Superior de Linares se ha diseñado una aplicación web alojada en la memoria del sistema Raspberry Pi. Esta aplicación interactúa con el servidor MQTT permitiendo al usuario controlar remotamente el sistema de climatización.

A continuación, en la figura 3.2 y 3.3 se representa el diagrama de flujo de la aplicación web implementada.

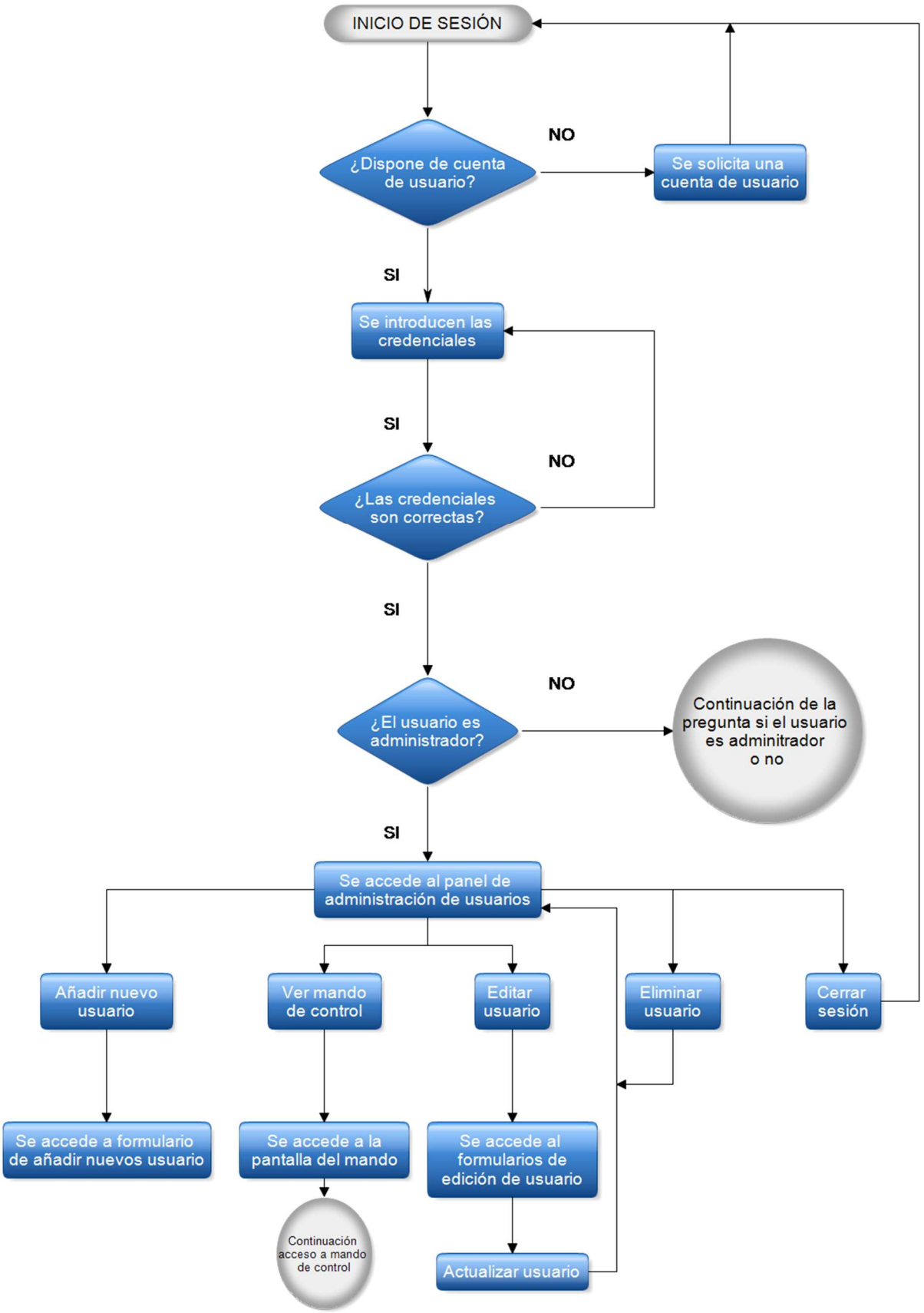


Figura 3.2 - Diagrama de flujo aplicación web

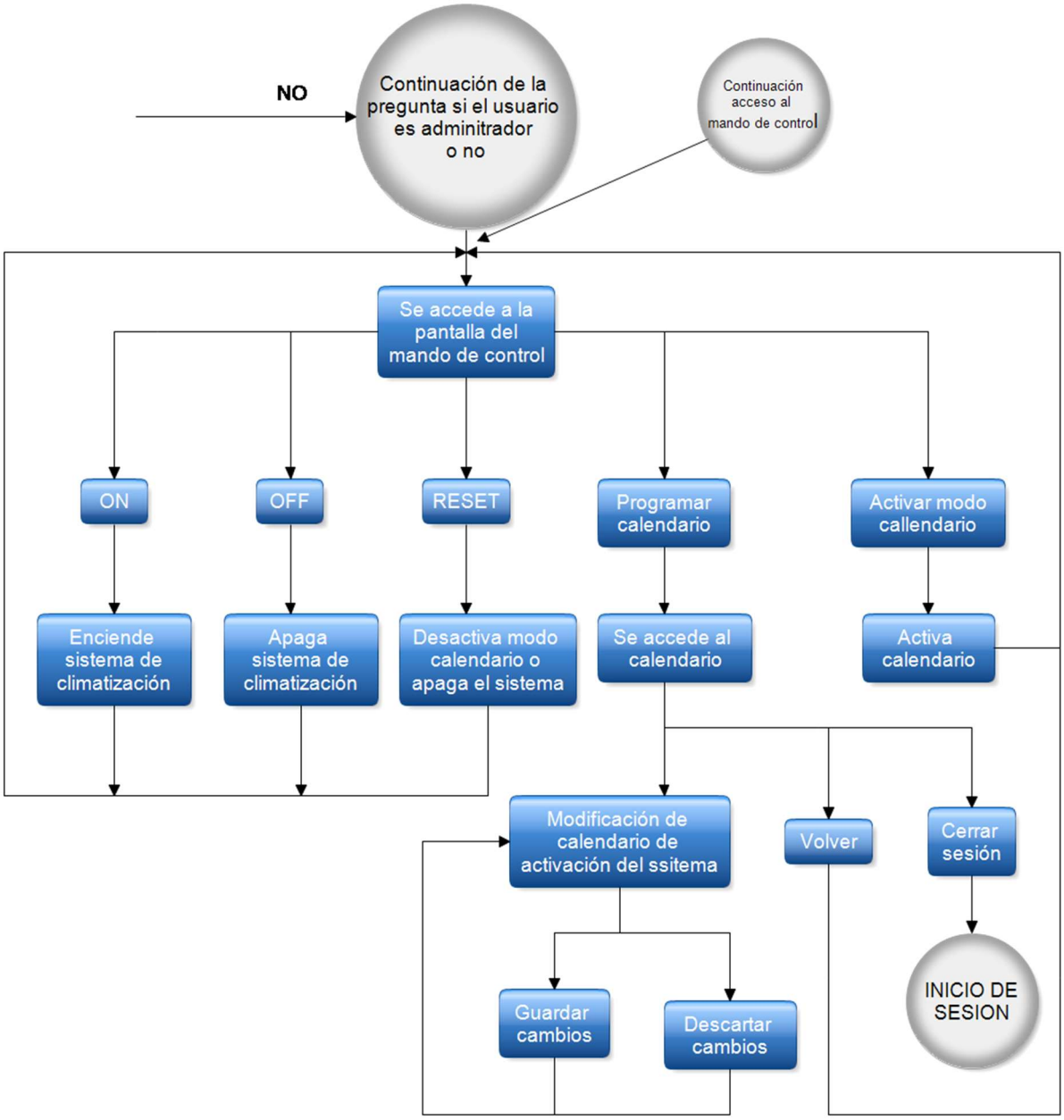


Figura 3.3 - Continuación diagrama de flujo aplicación web

A continuación se realiza una descripción de la aplicación web viendo las diferentes pantallas de las cuales dispone. Para acceder a la aplicación es necesario disponer de una conexión a Internet y desde cualquier navegador web acceder a la siguiente URL: https://onlineremotecontrol.ngrok.io . Tras esto, se necesitará disponer de una cuenta de usuario para poder iniciar sesión en la aplicación. Para este proyecto se ha creído conveniente la utilización de dos perfiles de usuario. El primero de ellos, es un perfil de administrador pensado para que los usuarios que utilicen el control remoto del sistema de climatización no se equivoquen a la hora de introducir el número de su despacho. Esto es debido a que si se introduce un número erróneo de despacho no se podrá controlar el encendido/apagado del sistema climático. A la misma vez, si se introduce el número de despacho de otro usuario es posible controlar el sistema de climatización de dicho despacho. Por lo tanto, el administrador es el encargado de la creación de nuevas cuentas de usuarios previamente solicitadas. El segundo de los perfiles, es un perfil de usuario normal el cual solo dispone de acceso al control del sistema climático de su propio despacho.

En la siguiente figura (figura 3.4) se puede observar la pantalla de inicio de sesión de la aplicación y seguidamente en la figura 3.5 se mostrará el formulario que un usuario normal puede utilizar para solicitar una nueva cuenta.

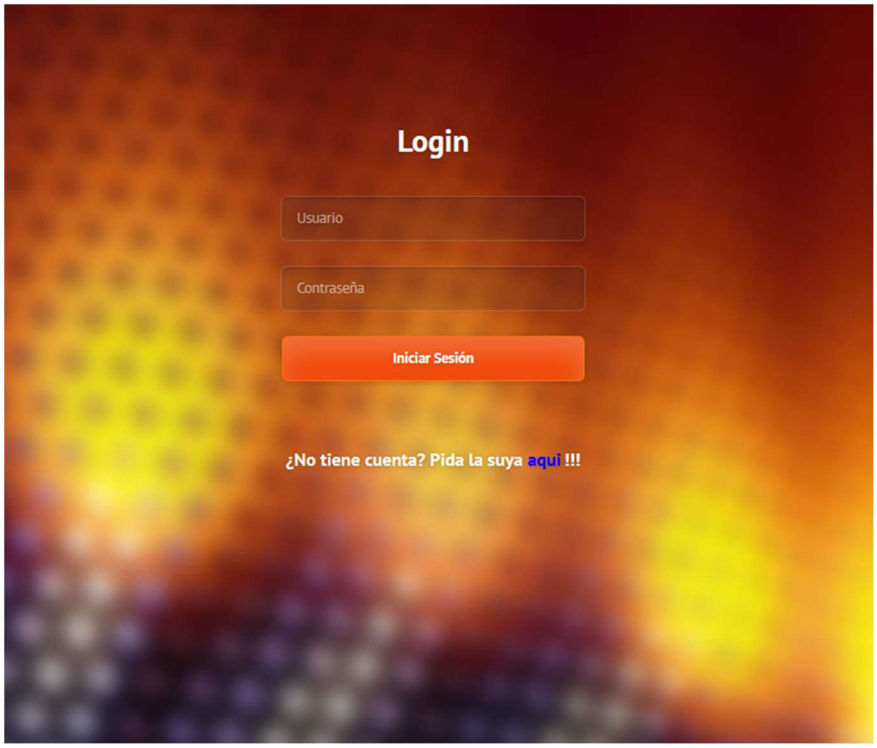


Figura 3.4 - Pantalla inicio de sesión aplicación web

Figura 3.5 - Formulario solicitud cuenta de usuario

A continuación, en la figura 3.6 se muestra el panel de administración de un usuario con perfil de administrador.

Figura 3.6 - Panel administración de usuarios

Un usuario con perfil de administrador puede editar los usuarios existentes en la base de datos, eliminar un usuario o añadir una nueva cuenta. Para editar un usuario se pulsa el botón de “EDITAR” mostrado en la figura 3.6 y una vez pulsado se abre un formulario donde el administrador puede actualizar los datos de un usuario registrado. En la figura 3.7 se muestra el formulario de edición de datos.

Figura 3.7 - Formulario de edición de un usuario registrado

Para eliminar un usuario basta con pulsar el botón “ELIMINAR” mostrado en la figura 3.6. Si el administrador quiere añadir un nuevo usuario este debe hacer clic en el botón “Añadir Usuario” ubicado en la parte superior de la pantalla de administración y rellenar el formulario pertinente. En la figura 3.8 se muestra el formulario para añadir un nuevo usuario.

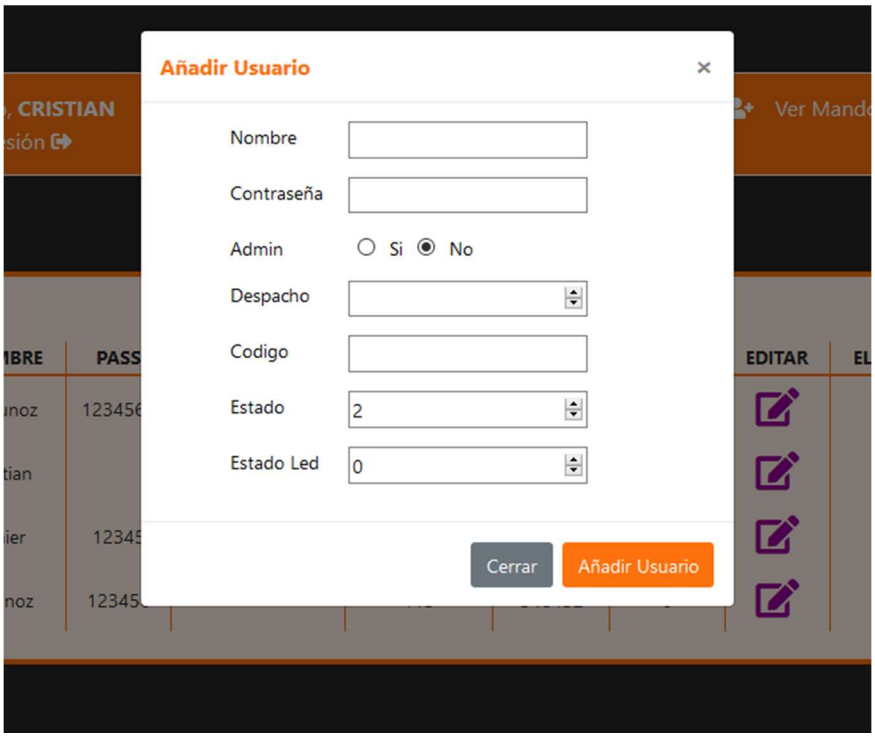


Figura 3.8 - Formulario para añadir nuevo usuario

Por último, un administrador también puede acceder a la pantalla del mando de control para así poder controlar remotamente el sistema de climatización ubicado en su despacho. La pantalla del mando de control tiene las mismas funcionalidades para un administrador que para un usuario normal. En la siguiente figura (figura 3.9) se muestra la pantalla principal del mando de control tanto para un usuario normal como para un administrador. Se ha creído conveniente utilizar 3 modos de funcionamiento de la aplicación. Estos modos son:

- Modo ON: el sistema de climatización se enciende.
- Modo OFF el sistema de climatización se apaga.
- Modo calendario: el sistema de climatización se enciende o apaga de forma automática según un horario preestablecido en la aplicación.

Figura 3.9 - Pantalla principal mando de control

En esta pantalla el usuario dispone de 5 botones. Estos botones son: ON/OFF, Activar Calendario, Programar Calendario, RESET y por último el botón de Cerrar Sesión.

La primera vez que un usuario acceda a la aplicación ésta detectará automáticamente el estado del sistema de climatización instalado en el despacho del usuario. Como se puede observar en la figura 3.9 en la parte superior izquierda del mando aparece un indicador del estado del sistema de climatización en cada momento, este indicador cambiar de color (verde o rojo) dependiendo si el sistema de climatización está encendido o apagado. Por lo tanto, si cuando el usuario accede por primera vez a la aplicación y el sistema de climatización este apagado, aparecerá el botón de ON junto con el indicador en color rojo. Cuando el usuario pulse el botón de ON la aplicación mandará automáticamente un mensaje al servidor MQTT instalado en el sistema Raspberry Pi y éste servidor devolverá el mensaje recibido al sistema NodeMCU. El sistema NodeMCU actuará

en consecuencia sobre el pulsador del mando de control NS-ATP7002 del sistema de climatización y encenderá el sistema. En este momento si el sistema se enciende aparecerá el indicador de color verde junto con el botón de OFF para así poder apagar el sistema climático.

Cuando el usuario pulsa el botón de “Programar Calendario” se abrirá una pantalla donde el usuario podrá configurar un horario de encendido y apagado automático del sistema de climatización. El calendario está programado de tal manera que se puedan seleccionar franjas de 30 en 30 minutos para el encendido del sistema de climatización de lunes a viernes con un horario de 8:00 hasta las 21:30. Una vez seleccionado el horario el usuario podrá guardar dicho su configuración o descartarla y volver a la pantalla principal del mando. A continuación en la figura 3.10 se puede observar la pantalla del horario de encendido y apagado del sistema de climatización.

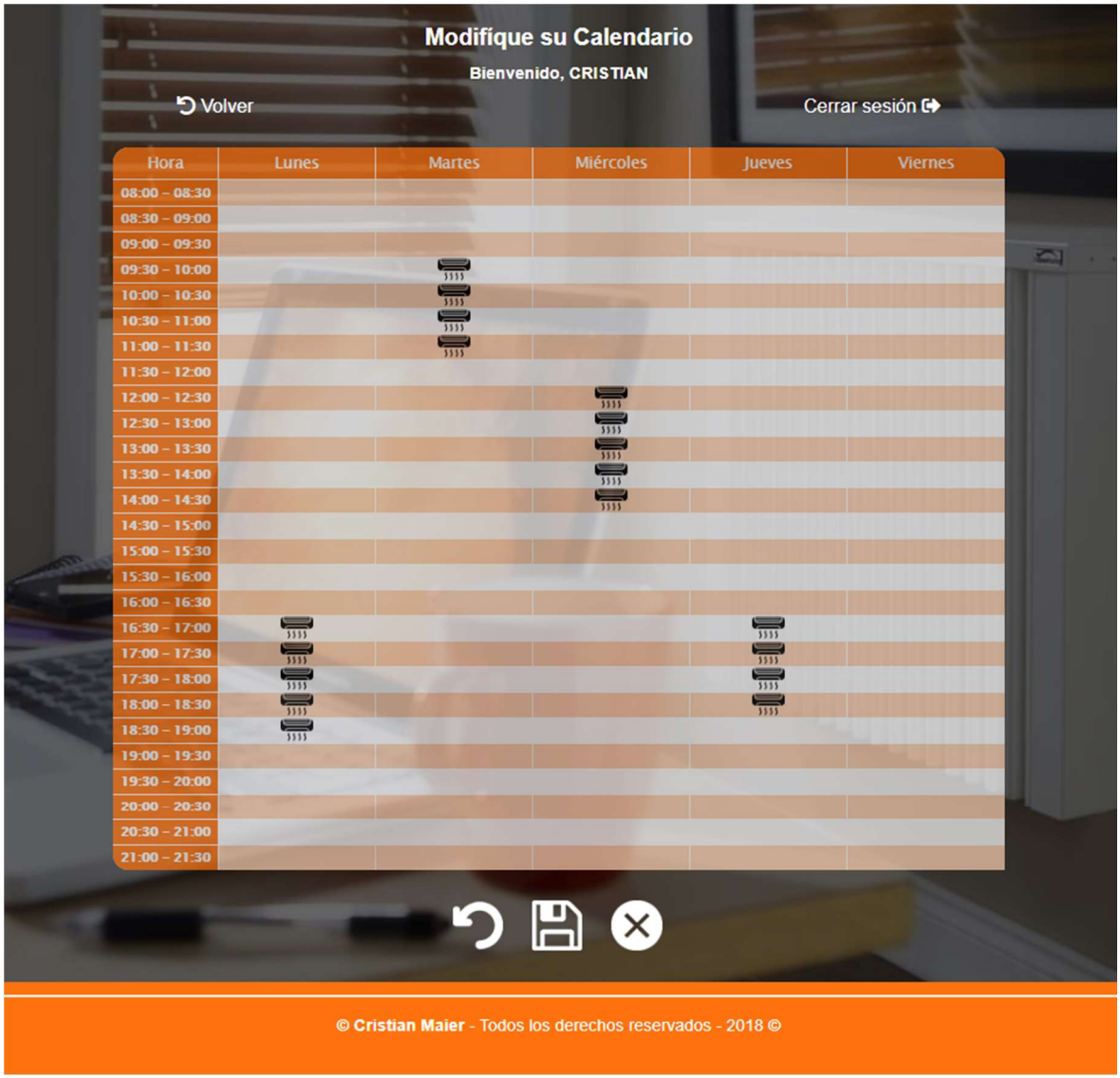


Figura 3.10 - Pantalla horario de encendido y apagado del sistema climático

Una vez guardado el horario el usuario podrá volver a la pantalla principal del mando de control y activar el calendario para que el sistema funcione automáticamente sin necesidad de pulsar el botón de ON o de OFF. En la siguiente figura (figura 3.11), se muestra el mando de control con el modo calendario activado.

Figura 3.11 - Pantalla mando de control con modo calendario activado

Por último, cuando se pulsa el botón de RESET y el modo calendario está activado, este modo se desactivará, de este modo el control del sistema de climatización pasará a modo manual, para así encender o apagar el sistema de climatización pulsando los botones de ON u OFF.

3.2 Programa NodeMCU

En este trabajo fin de grado se ha desarrollado un programa mediante el entorno de desarrollo de Arduino que posteriormente se ha cargado en la memoria flash del módulo NodeMCU a través de la conexión USB que incorpora dicho módulo. Este programa se encarga de realizar la conexión con el punto de acceso WiFi configurado en el sistema Raspberry Pi y de realizar la comunicación con el servidor MQTT. El programa gestiona las entradas/salidas del módulo NodeMCU para que éste actúe sobre los pines SET y UNSET incorporados en el relé de enclavamiento utilizado en el diseño del prototipo. La gestión se realiza atendiendo a los mensajes enviados desde la aplicación web al servidor MQTT. Este servidor MQTT envía dichos mensajes a través de la red WiFi al sistema NodeMCU para su posterior procesamiento.

A continuación en la figura 3.12 se puede observar el diagrama de flujo del programa y en el anexo 8 se puede consultar en detalle el código fuente.

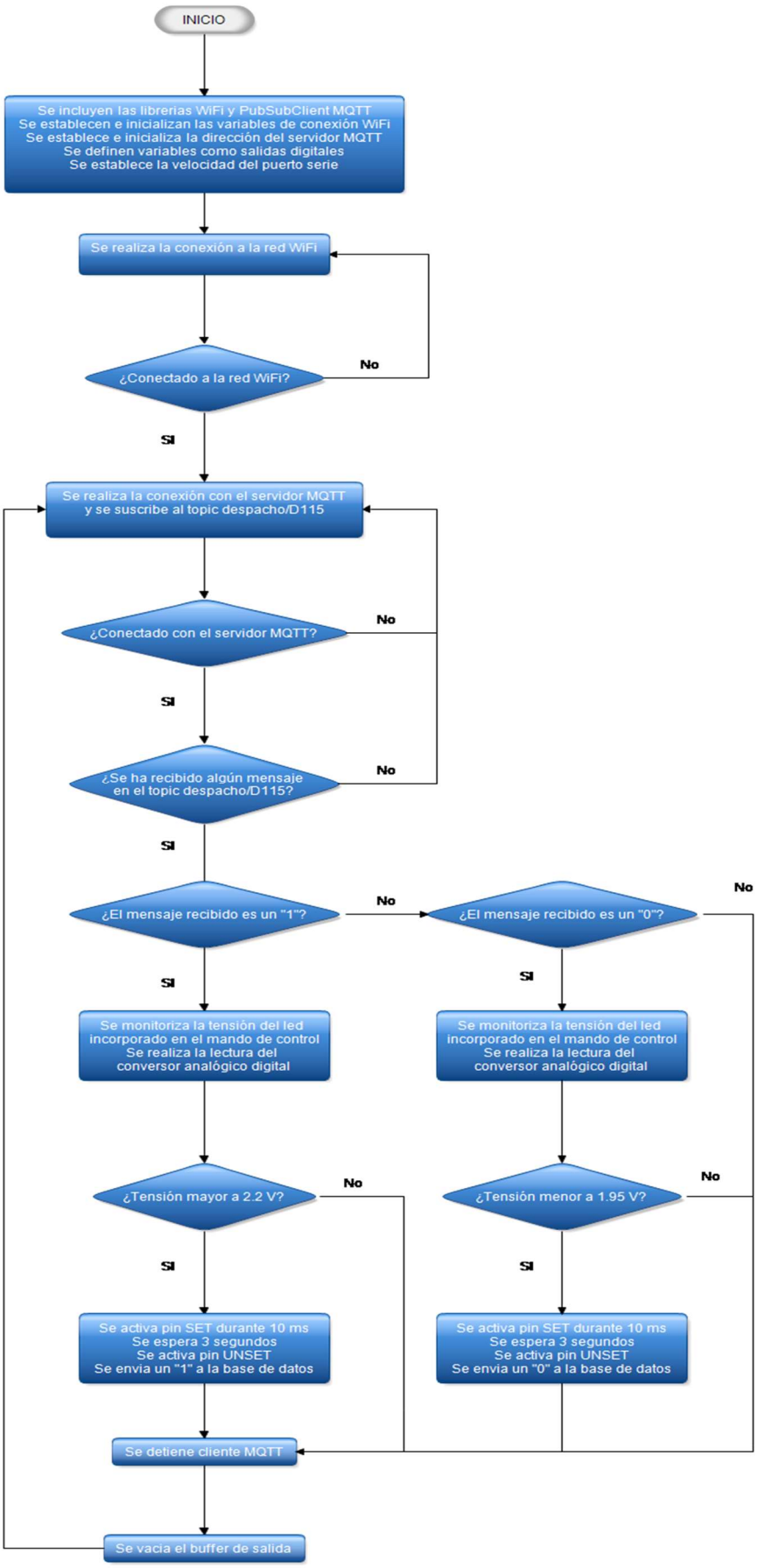


Figura 3.12 - Diagrama de flujo programa NodeMCU

3.3 Arduino IDE

Arduino IDE [28] (Integrated Development Environment – Entorno de Desarrollo Integrado) es un software de código abierto utilizado para escribir y transferir código tanto a las placas originales de Arduino como a otras placas de desarrollo compatibles (NodeMCU) con este IDE. El entorno está escrito en lenguaje Java y está basado en Processing [29] y otros softwares de código abierto. Admite a su vez lenguajes de programación como C o C++. Este entorno de desarrollo se puede ejecutar en Windows, MacOs y Linux. El código fuente para el IDE está publicado bajo licencia GNU – Licencia Publica General.

En este proyecto se ha utilizado el IDE de Arduino para desarrollar el programa principal del prototipo, que posteriormente se ha transferido a la memoria del módulo NodeMCU. Este entorno se ha instalado en un ordenador con sistema operativo Windows.

En el Anexo 1 de este documento se puede ver en detalle el proceso de instalación del entorno de desarrollo Arduino.

3.4 Raspbian – Sistema operativo Raspberry Pi

Raspbian es el sistema operativo recomendado y oficial de la Fundación Raspberry [35]. Está optimizado para el hardware de cualquier modelo de Raspberry Pi y se basa en una distribución gratuita de GNU/Linux llamada Debian.

Raspbian viene preinstalado con una gran cantidad de software para educación, programación y uso general. Entre este software se puede encontrar Python, Scratch, Sonic Pi, Java, Mathematica, etc [38].

Dispone de dos versiones, una de ella más completa con entorno gráfico y otra más reducida sin entorno gráfico.

- Raspbian Stretch PIXEL [36] (Pi *Improved Xwindows Environment Lightweight*): es la versión completa de Raspbian con entorno gráfico, es la versión con escritorio. Dispone de menús, ventanas, iconos, fondos de pantalla, etc.
- Raspbian Stretch Lite [36]: es la versión de Raspbian sin entorno gráfico. Esta versión utiliza el modo de consola de comandos sin gráficos.

En este trabajo fin de grado se ha utilizado la versión Raspbian Strech con escritorio debido a que ha sido la primera vez en utilizar una Raspberry Pi y no se disponen de conocimientos avanzados de Linux.

En el Anexo 3 se puede consultar en detalle el proceso de instalación del Sistema Operativo Raspbian.

3.5 Infraestructura LAMP

Se denomina LAMP a un conjunto de software libre y de código abierto utilizado normalmente para crear una plataforma empleada en desarrollar, desplegar y hacer accesibles aplicaciones web estáticas o dinámicas.

LAMP es el acrónimo compuesto por las cuatro iniciales de sus componente: Linux, Apache, MySQL y PHP.

- Linux es el sistema operativo utilizado en la infraestructura.
- Apache es un servidor web HTTP de código abierto [37]. En él se encuentra alojada la aplicación web (interfaz de usuario) diseñada para este proyecto.
- MySQL es un sistema gestor de bases de datos encargado de clasificar y proporcionar acceso a las bases de datos donde la aplicación web diseñada puede almacenar información. MySQL puede ser sustituido por MariaDB dentro de la infraestructura LAMP, su funcionamiento es muy parecido.
- PHP (Hypertext Preprocessor) es un lenguaje de programación enfocado a la programación de *scripts* del lado del servidor y adecuado para el desarrollo web. Fue uno de los primeros lenguajes del lado del servidor que se podían incorporar dentro de documentos HTML. PHP puede ser sustituido por PERL o Python dentro de la infraestructura LAMP.

Toda esta infraestructura hace posible la creación y accesibilidad de la aplicación web implementada en este trabajo fin de grado.

En el Anexo 4 se puede consultar en detalle el proceso de instalación de la infraestructura LAMP dentro de la Raspberry Pi.

3.6 MQTT

Message Queuing Telemetry Transport (MQTT) [2] es un protocolo de transporte de mensajes Cliente-Servidor basado en publicaciones y suscripciones de los clientes a los denominados *topics* (temas). Se ejecuta sobre TCP/IP o sobre otros protocolos de red que proporcionan conexiones bidireccionales, sin pérdida y ordenadas. El puerto TCP/UDP 1883 está reservado por la IANA (*Internet Assigned Numbers Authority*) para su uso con MQTT. El puerto 8883 también está registrado para usar MQTT sobre SSL.

En Octubre de 2014 MQTT se convirtió en un estándar OASIS [9] [10] (*Organization for the Advancement of Structured Information Standards*) y en Enero de 2016 ISO [11] (*International Organization for Standardization*) lo aprobó oficialmente como un estándar (ISO / IEC 20922) [12].

Este protocolo está diseñado para dispositivos con restricciones (RAM, CPU, etc.) y redes de bajo ancho de banda, alta latencia o poco confiables. Este tipo de red es la utilizada en el proyecto que nos ocupa ya que es una red inalámbrica WiFi entre una Raspberry Pi y un sensor NodeMCU.

MQTT se utiliza para la comunicación M2M (Maquina a Maquina, del término en Inglés *Machine to Machine*) en el IoT, donde se envían cantidades pequeñas de información y por tanto no se necesita gran ancho de banda. Para este proyecto el protocolo MQTT es idóneo puesto que es necesaria la comunicación M2M entre la Raspberry Pi y módulo NodeMCU.

La arquitectura de MQTT sigue una topología en estrella (Figura 3.13), donde existe un nodo central que realiza la función de servidor MQTT (*Broker*). El *Broker* es el encargado de gestionar la red y enviar los mensajes a cada cliente.

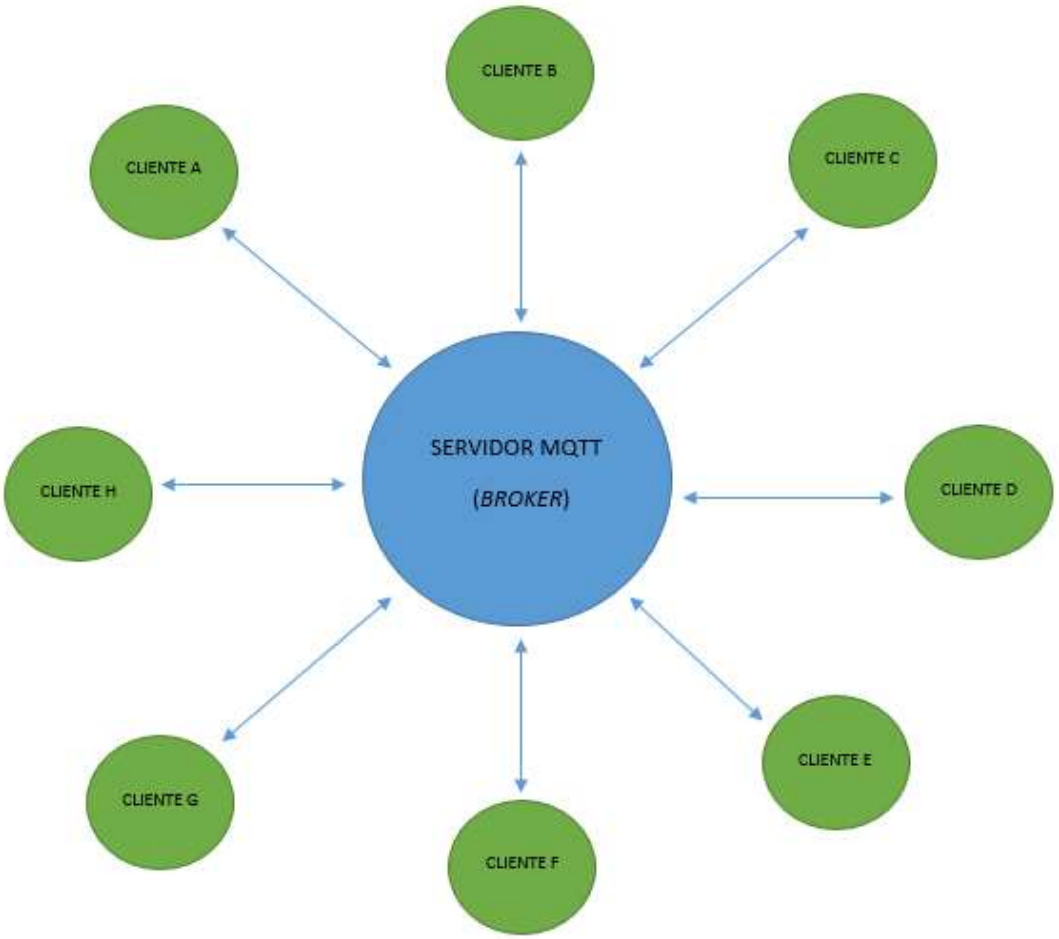


Figura 3.13 - Arquitectura MQTT

Dentro de la arquitectura MQTT es muy importante el concepto “tema” (de ahora en adelante *topic*), ya que la comunicación entre los clientes MQTT y el *Broker* se basa en estos *topics*.

Un *topic* se representa mediante una cadena de texto o cifras y tiene una estructura jerárquica, cada jerarquía se separa por un barra inclinada “/”.

El *topic* contiene la información de enrutamiento para el *Broker*, así cada cliente que desea recibir un mensaje se suscribe a un determinado *topic* y el *Broker* es el encargado de entregar el mensaje con el *topic* correspondiente. Por lo tanto, los clientes no tienen porque comunicarse entre ellos, ellos solo se comunican a través de *topics* y el *Broker* es quien entrega los mensajes a cada cliente.

Un ejemplo de *topic* utilizado para enviar la temperatura (Figura 3.14) del salón de la casa podría ser: “**casa / salón / temperatura”**.

Un cliente podría suscribirse a un tema en concreto o podría usar un comodín. El signo más (+) es un comodín de un solo nivel y solo permite valores arbitrarios para una jerarquía, si se necesita suscribirse a más de un nivel hay que usar el comodín multinivel (#). El comodín multinivel permite suscribirse a todos los niveles jerárquicos subyacentes, por ejemplo **casa/#**, se estaría suscribiendo a todos los *topics* que comienzan por casa.

La suscripción al *topic* **casa/+/temperatura** daría como resultado la suscripción a todos los *topics* de temperatura, como por ejemplo casa/**habitación**/temperatura, casa/**cocina**/temperatura, etc. El símbolo + se sustituirá por cada nivel que tenga como nivel superior casa y como nivel inferior temperatura.

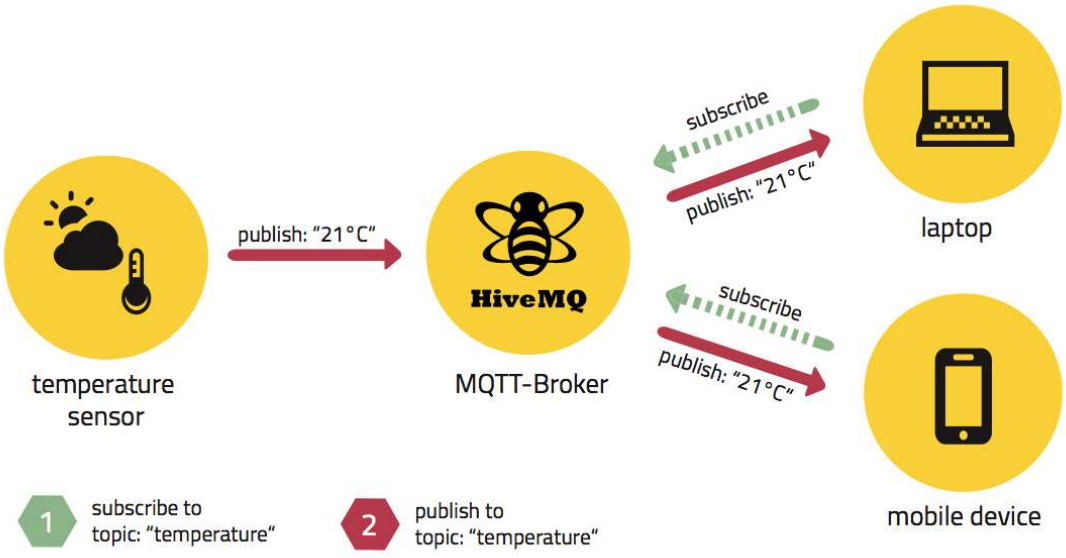


Figura 3.14 - Publicación/Suscripción MQTT [14]

En este proyecto los diferentes usuarios conectados a la aplicación web se comportan como clientes MQTT que publican mensajes al *topic* "despacho / nº _despacho" y el prototipo diseñado se comporta como cliente MQTT que se suscribe a este *topic*. Los *topics* creados por los usuarios se almacenan en el interior del *Broker*,

3.7 Mosquitto - Servidor MQTT

Mosquitto [30] es un servidor de mensajes de código abierto que implementa el protocolo MQTT versiones 3.1 y 3.1.1. Es un servidor liviano y adecuado para cualquier dispositivo, desde un mini ordenador con una sola placa de baja potencia como es una Raspberry Pi utilizada en este proyecto hasta servidores completos con capacidad de

gestionar gran cantidad de información. Mosquitto es parte de la Fundación Eclipse [31] y es un proyecto de IoT Eclipse [32]. Se encuentra bajo licencia EPL (*Eclipse Public License*) / EDL (*Eclipse Distribution License*). El proyecto Mosquitto proporciona una librería en lenguaje C para la implementación de clientes MQTT, y los conocidos clientes MQTT mosquitto_pub e mosquitto_sub.

En el proyecto que nos ocupa se ha utilizado el cliente MQTT mosquitto_pub para publicar mensajes mediante la aplicación web y el cliente mosquitto_sub para que el módulo NodeMCU se suscriba al topic “despacho / nº_despacho” comentado en el apartado anterior.

A pesar de que existen diversos servidores MQTT instalables [33], como por ejemplo, Apache ActiveMQ, HiveMQ, IBM MessageSight, Moquette, por mencionar algunos, para este proyecto se ha elegido el servidor MQTT Mosquito debido a la fácil implementación y a la gran cantidad de información y documentación relacionada con su funcionamiento e infinidad de ejemplos para su uso. También existen servidores MQTT en la nube [34], como por ejemplo, Flespi, HiveMq, Bevywise, etc. y hasta el propio Mosquitto dispone de servicios en la nube, pero se ha deseado disponer de un servidor MQTT propio instalado en una Raspberry Pi ya que de esta forma se puede realizar una mejor administración y no se tendría que depender de proveedores externos.

En este Trabajo fin de Grado el servidor Mosquitto es el encargado de recibir los mensajes enviados desde la aplicación web diseñada y enviarlos al sensor NodeMCU, para que éste mediante los pines conectados al relé actúe sobre el mando de control de la climatización.

En el Anexo 5 de este documento se puede ver en detalle la instalación de este servidor en la Raspberry Pi.

Figura 3.15 - Logo Mosquitto [30]

3.8 AP – Punto Acceso WiFi

En este Trabajo fin de Grado se ha utilizado la Raspberry Pi como punto de acceso inalámbrico, ejecutando una red independiente utilizando las funciones inalámbricas incorporadas en su hardware [39]. Se ha instalado el software de punto de acceso junto con un software de servidor DCHP para que la Raspberry Pi proporcione comunicación inalámbrica a la placa de desarrollo NodeMCU y ésta se comunique con el servidor MQTT.

En el Anexo 6 se puede consultar en detalle la configuración e instalación del software necesario para que el sistema embebido Raspberry Pi funcione como Punto de Acceso.

3.9 Ngrok – Túnel a Localhost

Ngrok [40] es una herramienta capaz de exponer un servidor web, alojado en una maquina local, a Internet a través de túneles seguros. Esta herramienta coloca un Firewall/NAT entre la maquina local (localhost) e Internet.

Para crear el túnel seguro solo es necesario proporcionarle a la aplicación Ngrok el puerto en el que está escuchando el servidor web local, normalmente es el puerto 80, el predeterminado para HTTP. La aplicación crea un túnel mediante una URL mostrando lo que se tenga en la carpeta principal del servidor web previamente instalado. Ngrok dispone de una versión gratuita donde la URL (subdominio) utilizada para acceder al servidor web local se genera de forma aleatoria cada vez que se arranca el servicio, y otra versión de pago donde se pueden configurar subdominios permanentes.

En este trabajo fin de grado se ha utilizado Ngrok para acceder desde fuera de la red local de la Politécnica Superior de Linares a la aplicación web alojada en el servidor web de la Raspberry Pi.

Se ha decidido utilizar esta herramienta debido a que para poder acceder a la aplicación web sin utilizar un túnel se debería de abrir un puerto en el router de la propia universidad y esto no es posible ya que se podría comprometer la seguridad de la red y de los usuarios en la universidad. De esta forma no se compromete la seguridad porque la aplicación Ngrok solo proporciona acceso al servidor alojado en la Raspberry Pi. Todo el tráfico entrante y saliente utiliza certificados TLS (Transport Layer Security) y los túneles están protegidos mediante autenticación de usuario y contraseña.

En el Anexo 7 se puede ver en detalle la instalación de Ngrok en la Raspberry Pi.

4. Capítulo 4: Resultados Experimentales. Pruebas y Ensayos

En este capítulo se detallarán las pruebas realizadas para conocer el funcionamiento del mando de control NS-ATP7002 así como las pruebas llevadas a cabo para comprobar el correcto funcionamiento del prototipo diseñado.

Para conocer en detalle el funcionamiento del mando de control individual del sistema de climatización instalado en uno de los despachos de la Escuela Politécnica Superior de Linares se ha procedido a desmontarlo. Así, empotrado en la pared se ha encontrado un conector de 4 pines (figura 4.1). Los pines del conector incorporan 4 pares de cables provenientes del bus de comunicación del sistema de climatización.

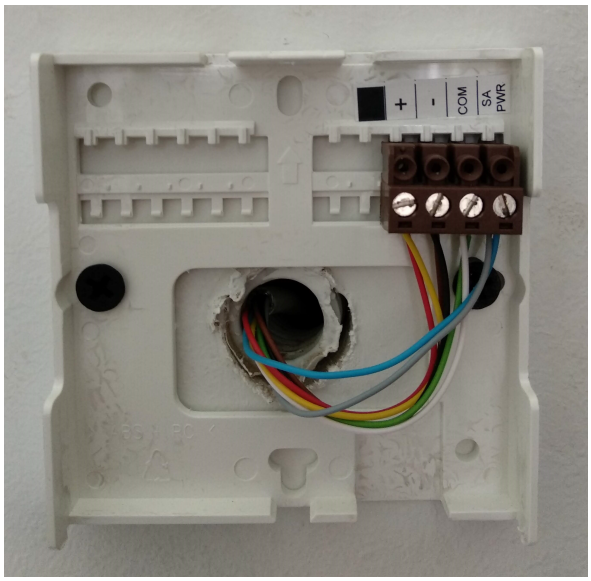


Figura 4.1 - Conector del mando de control empotrado en la pared

Tras un estudio de la documentación técnica disponible del fabricante y para conocer en detalle si es posible conectar el prototipo al bus de comunicación y alimentarlo de forma correcta, se ha procedido a la medición de las tensiones entre los diferentes pines. Las mediciones se han realizado tanto con el sistema apagado como encendido sin producirse variaciones significativas en los valores de tensión medidos. En la figura 4.2 se pueden observar las tensiones medidas entre los 4 pines disponibles en el conector.

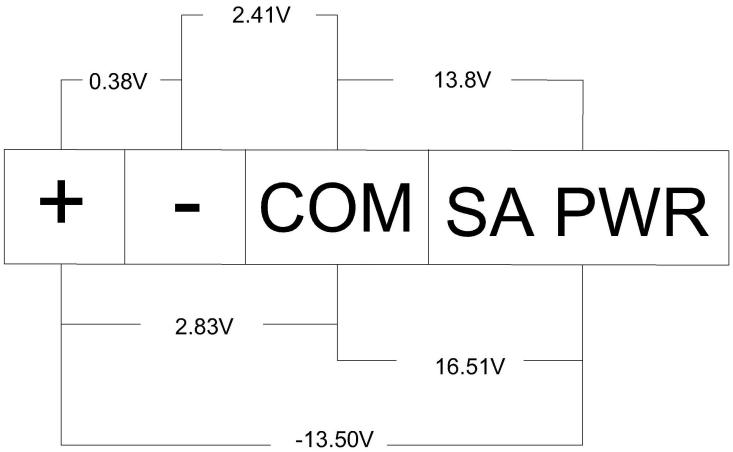


Figura 4.2 - Tensiones entre terminales del conector del mando de control NS-ATP7002

Según el manual de instalación del mando de control entre los pines COM y SA PWR debe de haber al menos 15 V para alimentar el mando. Como se puede observar en la figura 4.2 la tensión entre estos pines es de 16.51 V, por lo tanto, hay una tensión de alimentación adecuada para alimentar el prototipo. Se han comprobado las tensiones con un osciloscopio y se han apreciado señales de comunicación sobre valores de tensión DC entre el pin – y el pin COM, también entre el pin + y el pin COM. Estas señales son ondas cuadradas y se utilizan para comunicar el mando de control con el actuador central.

Para comprobar que las tensiones entre los pines del conector no varían al conectar una carga y que de este modo el actuador central es capaz de alimentar el prototipo, se ha conectado una resistencia (330 Ω) y un led (verde) entre los diferentes pines. Al conectar la resistencia y el led entre los pines SA PWR y – se descubrió que la tensión entre estos pines disminuye desde los 13.8 V hasta la tensión de 8.01 V. Al mismo tiempo que se realizaba este ensayo se apreció que la tensión entre los pines + y COM aumentaba desde los 2.83 V a los 7.98 V. Este ensayo se repitió igualmente conectando la carga entre los extremos de los pines + y COM obteniéndose el mismo resultado, una de las tensiones aumenta y la otra disminuye.

Llegado a este punto se ha decidido que para tener una correcta alimentación para el prototipo y no intervenir las comunicaciones entre el mando de control y el actuador central es necesario utilizar la señal de alimentación existente entre los pines SA PWR y COM ya que es la única que permanece constante. Además, con arreglo el manual de instalación del mando de control y el boletín técnico del protocolo MS/TP [27], ésta alimentación proviene del actuador central, siendo capaz de proporcionar una corriente máxima de 240 mA.

De igual modo se ha medido la corriente que circula por el pulsador del mando de control cuando se actúa sobre este y la tensión entre los extremos del led incorporado en el mando. La corriente que circula por el pulsador cuando se actúa sobre el es de 94.6 μA y la tensión entre los extremos del led al estar este encendido es de 1.86 V.

Una vez decidido de que pines se obtendrá la alimentación del prototipo, se ha procedido a realizar diferentes pruebas para asegurarse de que el bus de comunicación es capaz de suministrar la corriente necesaria para un correcto funcionamiento del prototipo. Estas pruebas han consistido en la conexión de diferentes cargas (resistencias de potencia) entre los extremos de los pines COM y SA PWR.

A continuación se describen los ensayos realizados junto con los resultados obtenidos.

Ensayo 1: conexión de resistencias de potencia con el sistema de climatización encendido

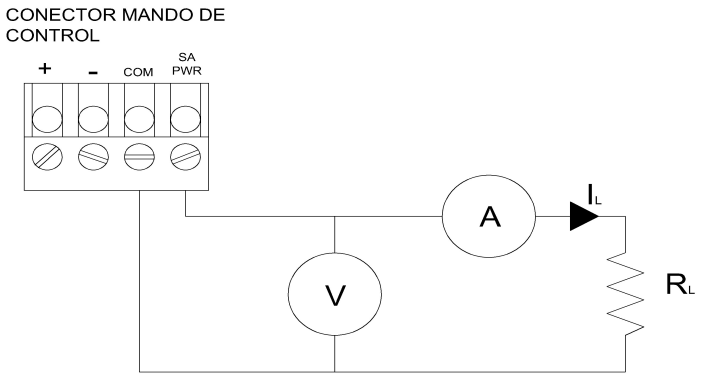


Figura 4.3 - Ensayo 1: conexión resistencias de potencia

Previamente se han realizado unos cálculos para conocer de antemano los valores de resistencias y potencias disipadas necesarias para el circuito de ensayo de la figura 4.3.

$$R = \frac{V}{I} \Rightarrow R_L(I_L = 50\ \text{mA}) = \frac{16}{50 * 10^{-3}} \cong 320\ \Omega$$

$$P = V * I \Rightarrow P = 16 * 50 * 10^{-3} = 0.8\ W$$

$$R_L(I_L = 240\ \text{mA}) = \frac{16}{240 * 10^{-3}} \cong 67\ \Omega$$

$$P = 16 * 240 * 10^{-3} = 3.84\ W$$

La resistencia más cercana a 320 Ω disponible ha sido una resistencia de 470 Ω, por lo tanto, se ha calculado la intensidad teórica de esta resistencia:

$$I_L = \frac{16}{470} \cong 34\ mA$$

$$P = 16 * 34 * 10^{-3} = 0.544\,W$$

En la tabla 4.1 se muestran los resultados obtenidos tras el ensayo. Como se puede apreciar la tensión entre extremos de los pines COM y SA PWR es capaz de proporcionar hasta 214 mA si bien la tensión de alimentación cae 1.1 V. esta caída de tensión es asumible y no afecta al funcionamiento dl prototipo como se verá a continuación.

R_L (Ω)	I_L (mA)	V_{in} (V)
470	34	15.7
150	100	15.3
68	214	14.6

Tabla 4.1 - Resultados obtenidos tras la ejecución del primero de los ensayos destinado a conocer la corriente máxima proporcionada entre los pines COM y SA PWR

Ensayo 2: **test funcional del conversor DC/DC (Vout = 3.7 V)**

Mediante este ensayo se ha realizado una prueba funcional del conversor DC/DC. Se ha tratado básicamente de conectar la entrada del mismo a los terminales COM y SA PWR y hacerlo trabajar bajo diferentes estados de carga.

La primera prueba se ha realizado actuando sobre el potenciómetro del mismo y fijando con una tensión de salida del mismo de 3.7 V. Se ha de recordar que el NodeMCU necesita una tensión de al menos 3.3 V para funcionar.

Para medir la tensión de entrada al conversor DC/DC se ha conectado el canal 1 de un osciloscopio y para medir la corriente se ha utilizado un amperimetro en serie con el terminal SA PWR (terminal positivo de entrada del conversor). Así mismo, para medir la tensión de salida se ha conectado el segundo canal del osciloscopio a los terminales de salida del conversor DC/DC y para medir la corriente se ha conectado otro amperímetro en serie con el terminal positivo de salida. En la figura 4.4 se pueden ver las conexiones realizadas.

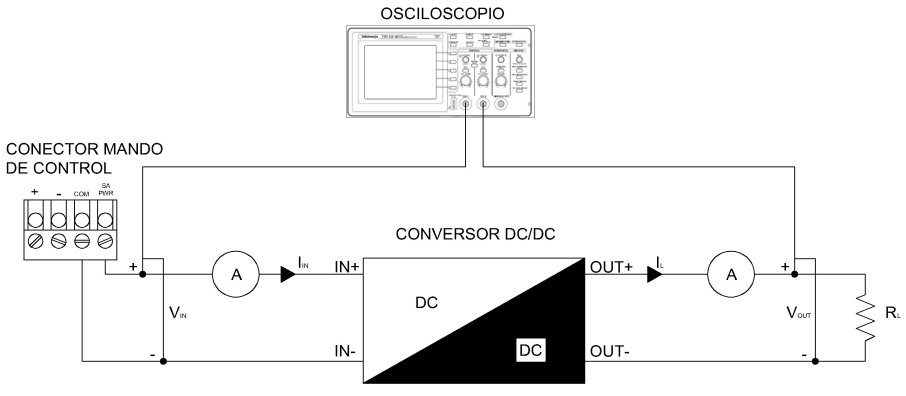


Figura 4.4 - Conexiones ensayo 2: conversor DC/DC y resistencias

Previamente se han realizado los cálculos teóricos para estimar el valor y potencia disipada de las resistencias de carga utilizadas.

$$R = \frac{V}{I} \Rightarrow R_L(I_L = 20\ \text{mA}) = \frac{3.7}{20 * 10^{-3}} \cong 185\ \Omega$$

$$P = V * I \Rightarrow P = 3.7 * 20 * 10^{-3} = 0.074\ W$$

$$R_L(I_L = 220\ \text{mA}) = \frac{3.7}{220 * 10^{-3}} \cong 16.82\ \Omega$$

$$P = 3.7 * 220 * 10^{-3} = 0.81\ W$$

A continuación se muestra en tabla 4.2 los resultados de las pruebas realizadas y las resistencias de potencia conectadas.

V_{in} (V)	V_{out} (V)	R_L (Ω)	I_{in} (mA)	I_L (mA)
15.9	3.65	180	12	20
15.8	3.52	100	18	40
15.6	3.45	33	43	110
15.5	3.53	22	58	150
15.4	2.80	16	60	192

Tabla 4.2 - Resultados medidas con conversor DC/DC (V_{out} 3.7 V) y resistencias de potencia

A la vista de los resultados se ha llegado a la conclusión que si se ajusta la tensión de salida del conversor a 3.7 V al demandarle al circuito una corriente próxima a 200 mA hay una caída de tensión de 0.85 V. Por lo tanto, no se podría alimentar el sistema de desarrollo NodeMCU porque la tensión de salida del conversor baja de los 3.3 V mínimos necesarios para alimentar el sistema embebido.

Ensayo 3: **test funcional del conversor DC/DC (Vout = 4.3 V)**

En la segunda prueba realizada con el conversor DC/DC se ha fijado una tensión de salida del mismo de 4.3 V para así solucionar el problema de la caída de tensión de la red visto en el ensayo anterior.

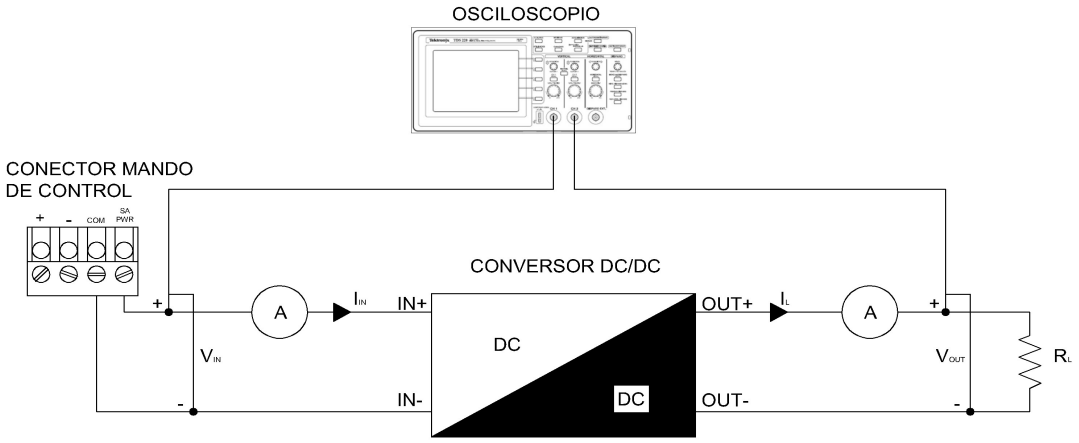


Figura 4.5 - Conexiones ensayo 3: conversor DC/DC y resistencias

Previamente se han realizado los cálculos teóricos para estimar el valor y potencia disipada de las resistencias de carga utilizadas.

$$R = \frac{V}{I} \Rightarrow R_L(I_L = 20\ \text{mA}) = \frac{4.3}{20 * 10^{-3}} \cong 215\ \Omega$$

$$P = V * I \Rightarrow P = 4.3 * 20 * 10^{-3} = 0.086\ W$$

$$R_L(I_L = 240\ \text{mA}) = \frac{4.3}{240 * 10^{-3}} \cong 17.92\ \Omega$$

$$P = 4.3 * 240 * 10^{-3} = 1.032\ W$$

A continuación se muestra la tabla 4.3 con los resultados de las pruebas realizadas y las resistencias de potencia conectadas.

V_{in} (V)	V_{out} (V)	R_L (Ω)	I_{in} (mA)	I_L (mA)
15.95	4.31	220	12	20
15.9	4.3	180	14	24
15.8	4.2	100	20	45
15.6	4.05	33	48	125
15.5	3.8	22	62	185
15.4	3.65	16	71	230

Tabla 4.3 - Resultados medidas con conversor DC/DC (V_{out} 4.3 V) y resistencias de potencia

Como se pueden observar en las tablas anteriormente expuestas la tensión de entrada al conversor proveniente de los pines COM y SA PWR no baja de los 15 V, que justamente corresponde a la tensión mínima de alimentación del mando de control según el manual técnico de instalación. Tampoco disminuye la tensión de salida del conversor

DC/DC, ésta tensión no baja de los 3.3 V mínimos necesarios para que el sistema de desarrollo NodeMCU funcione correctamente. En otras palabras, gracias al conversor DC/DC se consigue satisfacer una demanda de hasta 230 mA con una tensión de salida de 3.65 V sin perturbar la red de climatización y pasando prácticamente desapercibido.

También se ha de hacer notar que el sistema de climatización no se ha apagado en ningún momento mientras se han realizado las pruebas anteriores.

Ensayo 4: test funcional del conversor DC/DC junto con un módulo NodeMCU

En esta prueba se ha conectado la salida del conversor DC/DC al terminal de alimentación del módulo NodeMCU. Al sistema NodeMCU se la ha incorporado un pequeño programa capaz de hacer parpadear cada 3 segundos el led incorporado en su placa. Esta prueba se ha realizado para conocer la tensión y la corriente que circula por el terminal de entrada y de salida del conversor DC/DC cuando se conecta el módulo NodeMCU. Para medir la tensión de entrada al conversor DC/DC se ha conectado el canal 1 de un osciloscopio y para medir la corriente se ha conectado un amperímetro en serie con el terminal SA PWR (terminal positivo de entrada del conversor). Así mismo, para medir la tensión de salida se ha conectado un segundo canal del osciloscopio a los terminales de salida del conversor DC/DC y para medir la corriente se ha conectado otro amperímetro en serie con el terminal positivo de salida. En la figura 4.6 se pueden ver un esquemático con las conexiones realizadas.

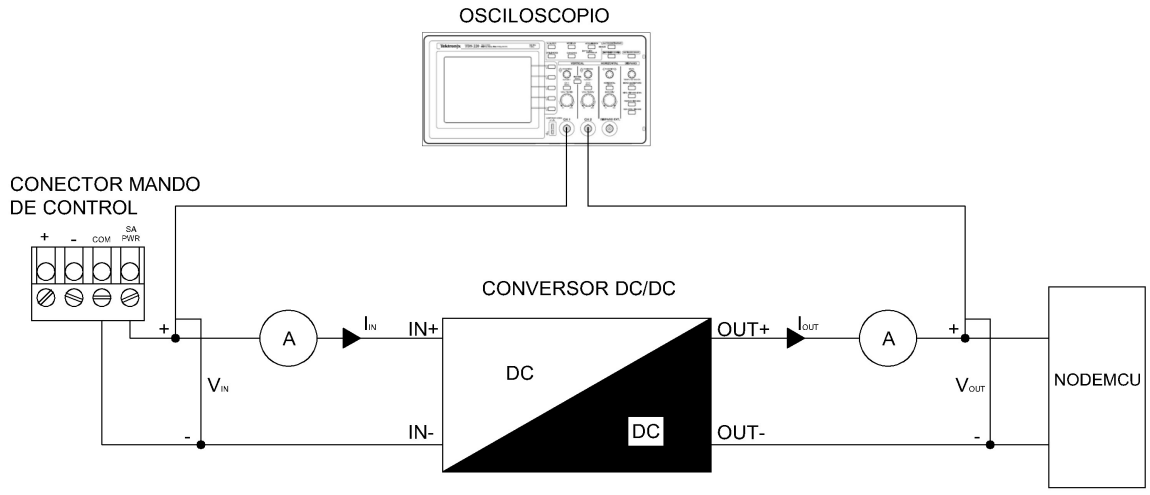


Figura 4.6 - Conexiones prueba 4: conversor DC/DC y NodeMCU

Se han obtenido los siguientes resultados:

	V_{in} (V)	I_{in} (mA)	V_{out} (V)	I_{out} (mA)
Led apagado	15.6	35	4.05	80
Led encendido	15.6	35.7	4.03	86

Tabla 4.4 - Resultados de medidas con conversor DC/DC y NodeMCU

A la vista de los resultados obtenidos la corriente demandada por el NodeMCU no llega a superar los 86 mA y la que es más importante, la demanda de corriente requerida del conector empotrado se queda en tan solo 35.7 mA.

Ensayo 5: test funcional del prototipo

Finalmente se ha realizado la última prueba conectando el prototipo a la alimentación del mando de control. Para asegurar de que el sistema funciona correctamente y para conocer en todo momento la tensión tanto de entrada al conversor como de salida se ha conectado un osciloscopio midiendo tensión a través de sus dos canales. El primer canal del osciloscopio se ha conectado entre los terminales COM y SA PWR del mando de control (estos terminales son los terminales de entrada del conversor DC/DC) y el segundo canal se ha conectado a los terminales de salida del conversor DC/DC. También se han conectado dos amperímetros para conocer en todo momento la corriente que circula por el circuito. Uno de los amperímetros se ha conectado en serie con el terminal SA PWR (terminal de entrada al conversor DC/DC) y el segundo amperímetro se ha conectado al terminal positivo de salida del conversor, también en serie. En la siguiente figura (figura 4.8) se pueden ver un esquemático con las conexiones realizadas.

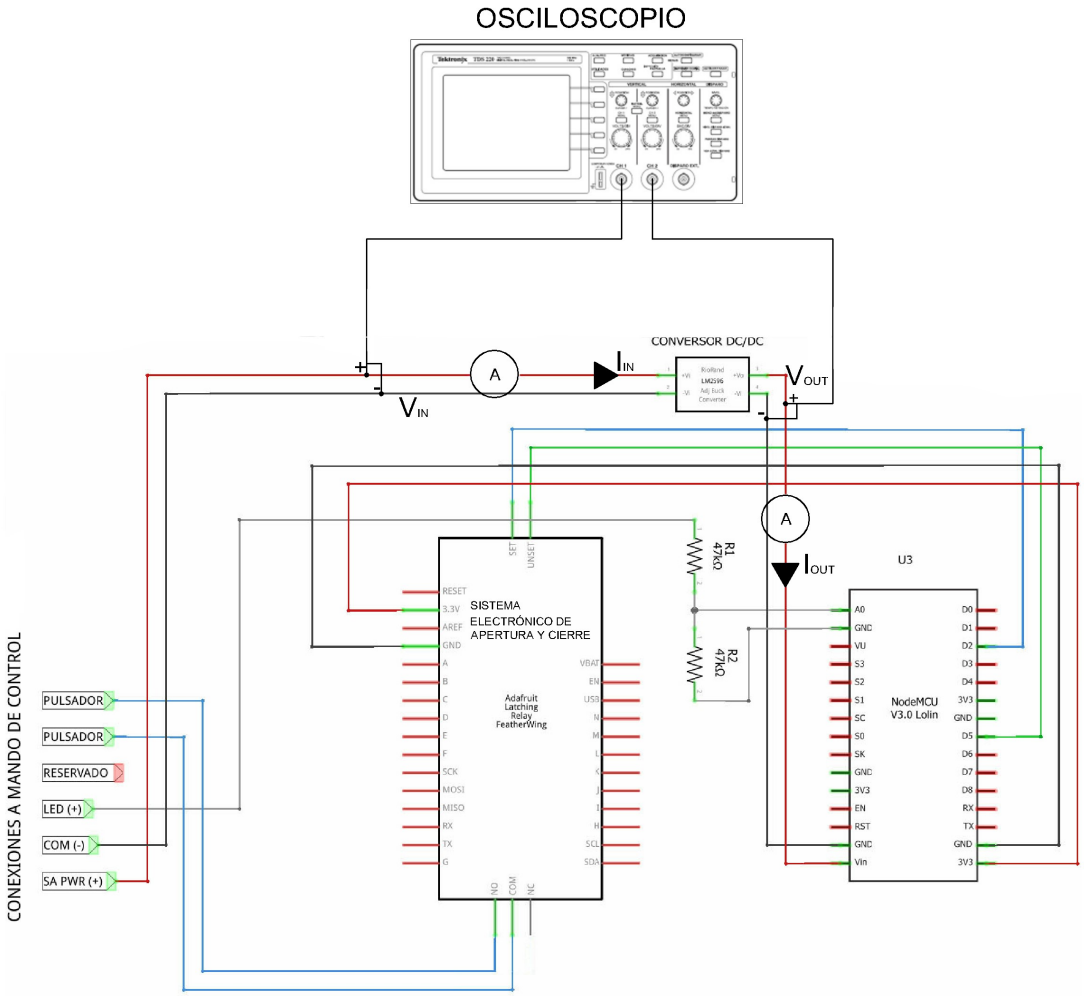


Figura 4.7 - Conexiones prueba 6: prototipo con mando de control

A continuación (tabla 4.5) se exponen los resultados de las mediciones con el prototipo funcionando.

V_{in} (V)	I_{in} (mA)	V_{out} (V)	I_{out} (mA)
15.7	33.4	4.1	76

Tabla 4.5 - Resultados tras la realización del ensayo 5

Una vez más la demanda de corriente (I_{in}) está muy por debajo de los valores máximos que el conector empotrado es capaz de proporcionar.

Una vez realizada la última prueba se ha procedido a conectar el prototipo al mando de control (figura 4.9) mediante el conector mecanizado en el mando de control bajo ensayo (Figura 2.7 y 2.8). Para ello se han insertado los pines (macho) disponibles en la caja del prototipo a la tira de pines (hembra) incorporada en el mando de control. Hay que tener en cuenta que si el prototipo se conecta con el sistema de climatización encendido el led

incorporado en el mando se apaga pero el sistema de climatización sique funcionando. Para solucionar este problema se ha de tomar la precaución de conectar el prototipo una vez el mando este amagado y nunca con este en funcionamiento.

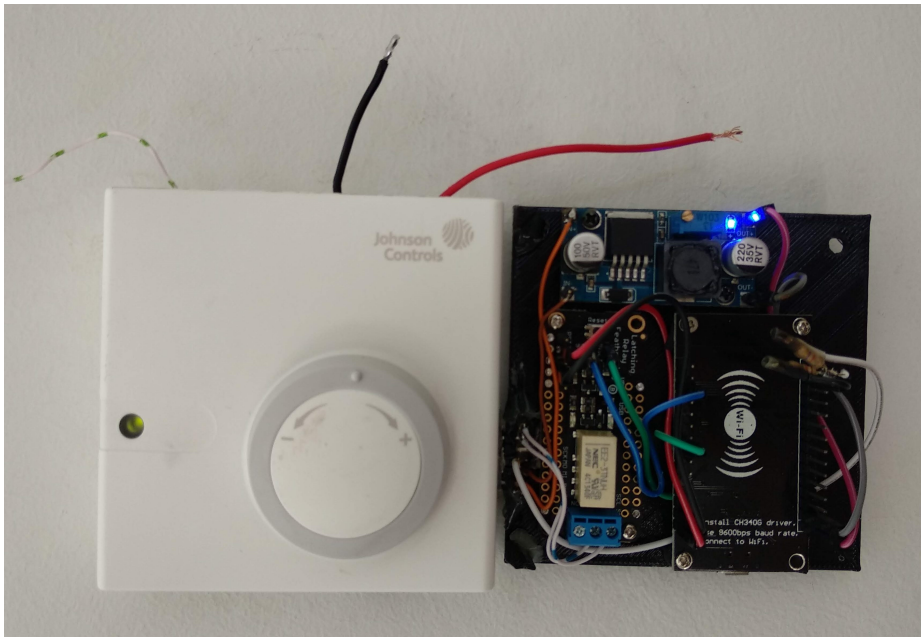


Figura 4.8 - Conexión final del prototipo con el mando de control

Con arreglo a las pruebas realizadas se ha llegado a la conclusión de que el prototipo puede funcionar según lo esperado permitiendo el control remoto desde cualquier parte de la red de Internet del sistema de climatización de los despachos de la Escuela Politécnica Superior de Linares.

Capítulo 5: Conclusiones

En este trabajo fin de grado se ha conseguido construir un prototipo de bajo coste capaz de controlar el mando individual del sistema de climatización instalado en el edificio departamental de la Escuela Politécnica Superior de Linares. Tras una revisión del estado del arte en lo que a sistemas domóticos respecta el proyecto incorpora tanto en hardware como en software innovadoras tecnologías: sistemas embebidos y arquitecturas software utilizadas en IoT, respectivamente. De este modo, el prototipo diseñado junto con la interfaz de usuario, permiten controlar remotamente el encendido y apagado del sistema de climatización así como la posibilidad de configurar un horario semanal automático sin la necesidad de intervenir manualmente sobre el mando de control del sistema climático. Todos los componentes utilizados en este proyecto son de reducido tamaño y tienen un consumo mínimo de energía. Por otro lado, se ha conseguido incorporar todos los componentes en una caja construida con una impresora 3D. Con este prototipo se ha conseguido cumplir el objetivo principal de este proyecto, así como todos los requisitos mencionados en la sección 1.3 de esta memoria.

A continuación y de forma esquemática se van a listar las conclusiones más importantes que se han extraído de este trabajo fin de grado.

- Se ha conseguido que el usuario pueda controlar el sistema climático de los despachos de la Escuela Politécnica Superior de Linares mediante dispositivos móviles y dispositivos de escritorio.
- Los elementos hardware constitutivos son de bajo coste y comerciales.
- El sistema embebido que actúa directamente sobre el mando de control se alimenta del conector del propio mando de la climatización, con lo que no es necesario la utilización de baterías o alimentación externa.
- Se ha conseguido implementar una arquitectura software basada en el protocolo MQTT para establecer una comunicación segura entre el prototipo que controla el mando de la climatización y la aplicación web implementada.
- Se ha conseguido disponer de una gestión de usuarios mediante el uso de bases de datos.
- El prototipo implementado se ha montado y probado con éxito en uno de los despachos del edificio departamental.

Capítulo 6: Líneas futuras

Tras la realización de este trabajo fin de grado cabe mencionar una serie de mejoras que podría aumentar las prestaciones del sistema implementado y descrito en esta memoria. Entre esas actuaciones destinadas a aumentar las prestaciones del equipo se han de citar las siguientes:

- En cuanto a mejoras hardware cabe la posibilidad de diseñar e implementar dos shields capaces de realizar las tareas del conversor DC/DC y del sistema electrónico de apertura y cierre. El sistema NodeMCU incorpora las conexiones necesarias para poder montar dichos circuitos de manera apilable reduciéndose considerablemente el tamaño del prototipo. Teniendo en cuenta la poca corriente de alimentación demandada, sería viable construir de forma sencilla y compacta una shield que incorporando un conversor DC/DC reductor, fuese capaz de alimentar la plataforma de desarrollo NodeMCU. Igualmente las corrientes de apertura y cierre puestas en juego en la conmutación del mando de control, permiten diseñar una shield de dimensiones reducidas capaz de llevar a cabo -de forma más eficiente que el relé utilizado- la tarea de apagado y encendido del mando de control. La implementación de estos dos componentes en una PCB supondría además de una reducción en el tamaño y una más que probable disminución del coste final del prototipo.
- En cuanto a las mejoras software se podría realizar una búsqueda más a fondo con el fin de incorporar algún tipo de periférico al NodeMCU con el objetivo de poderse conectar directamente a la red wifi de la Universidad de Jaén. Esto supondría un ahorro considerable dado que la plataforma Raspberry Pi ya no sería necesaria.

ANEXOS

Anexo 1: Instalación Arduino IDE – Entorno de Desarrollo Arduino

Para instalar el entorno de desarrollo de arduino en un equipo Windows se han de seguir los pasos que se detallan a continuación:

1. Se ha de acceder a la página web oficial de Arduino: www.arduino.cc , en la pestaña de Software y hacer clic sobre “DOWNLOADS”. (Figura A.1.1)

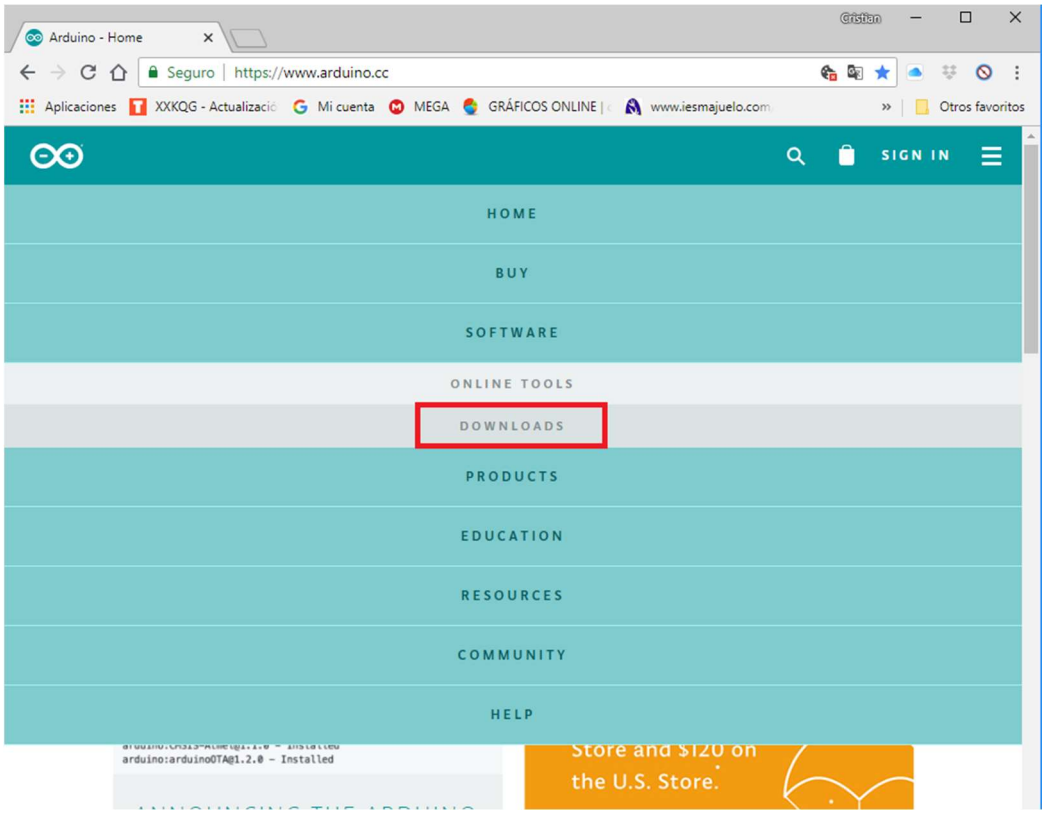


Figura A.1. 1 - Página web Arduino

2. Se ha de descargar el archivo de instalación haciendo clic sobre el instalador de Windows. (Figura A.1.2)

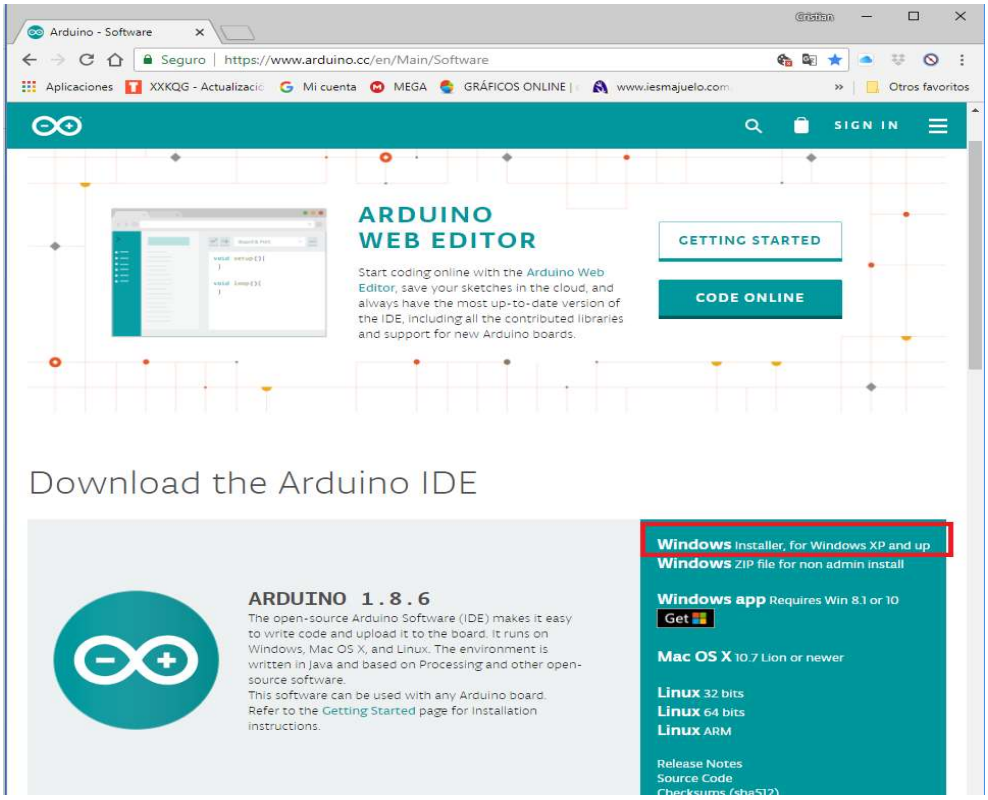


Figura A.1. 2 - Pagina web Arduino, Archivo de instalación.

3. Se ha de acceder a la carpeta de descargas del sistema operativo y ejecutar el archivo de instalación descargado.
4. Se han de aceptar las condiciones haciendo clic en “I Agree”. (Figura)

Figura A.1. 3 - Aceptación de condiciones

5. Se han de dejar todas las opciones marcas por defecto y debe prestar especial atención en la opción de “Install USB Drivers” (Figura A.1.4), ya que habilitar esta opción resulta esencial para que la placa Arduino se pueda comunicar con el PC via USB. Se presiona “Next” y tras esto se presiona el botón “Install” (Figura A.1.5).

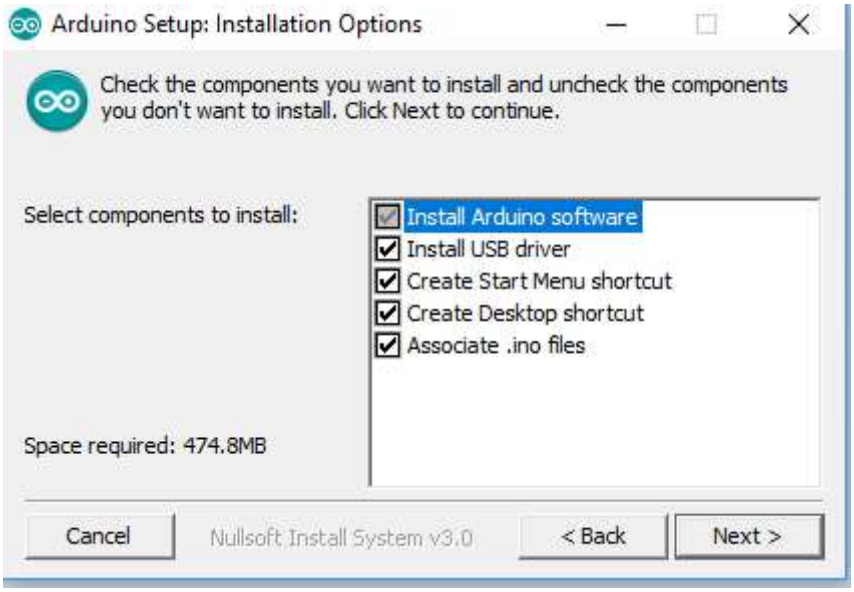


Figura A.1. 4 - Selección de opciones de instalación

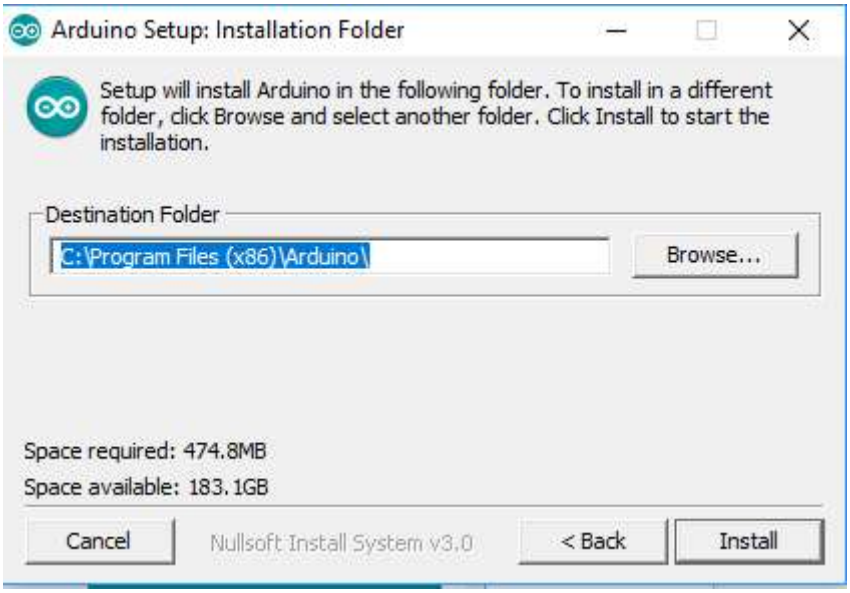


Figura A.1. 5 - Selección carpeta de instalación

6. Se ha de esperar hasta que el proceso de instalación acabe (figura A.1.6) y tras esto, se ha de presionar el botón “Close” para cerrar el asistente de instalación.

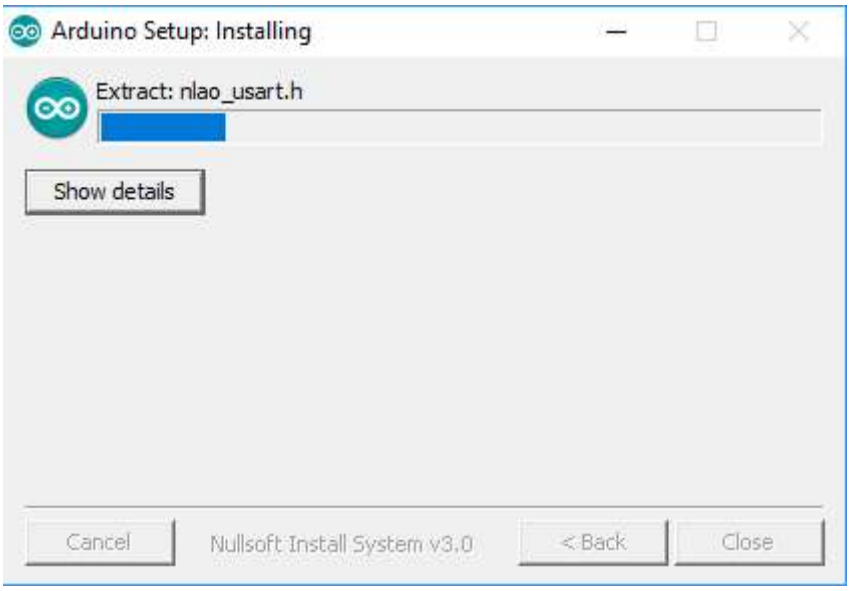


Figura A.1. 6 - Proceso de instalación Arduino IDE

7. Se ha de iniciar el IDE de Arduino haciendo clic sobre el acceso directo creado en el escritorio.
8. Se ha de añadir el soporte para trabajar con la tarjeta NodeMCU, ya que por defecto solo aparecen las tarjetas con soporte oficial de Arduino. Para ello se accede al repositorio de GitHub (www.github.com/esp8266/arduino) donde se encuentran los archivos para las tarjetas ESP8266. Del repositorio de GitHub se necesita la URL del archivo .json, este archivo permite al IDE de arduino incorporar las librerías que son necesarias para programar las tarjetas con el SoC ESP8266, entre las que se encuentra el NodeMCU. Por lo tanto, se copia la URL del archivo (Figura A.1.7).

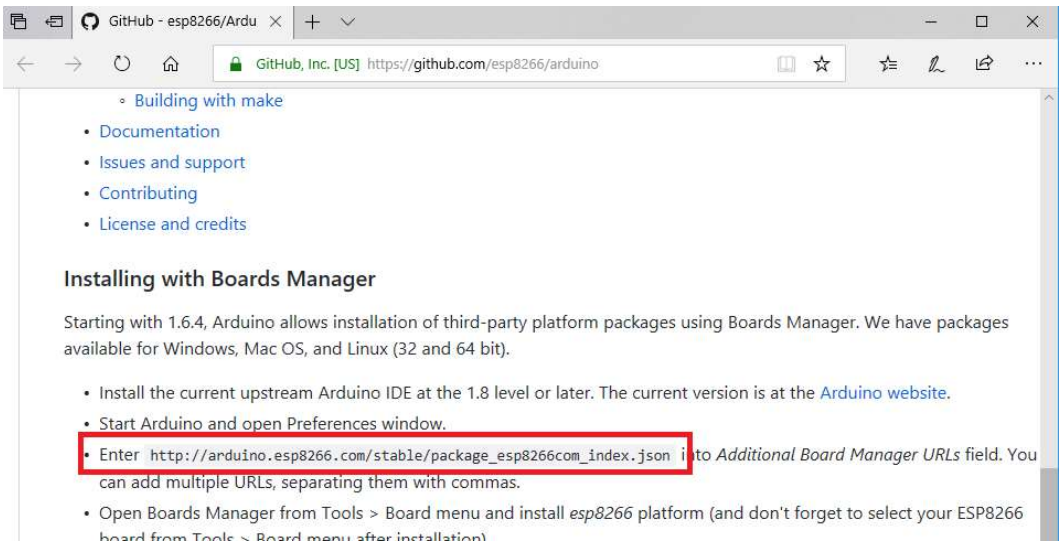


Figura A.1. 7 - Repositorio GitHub ESP8266: URL archivo .json

9. Se ha de acceder a la aplicación de arduino, hacer clic sobre “Archivo” → “Preferencias” y en el campo “Gestor de URLs Adicionales de Tarjetas” copiar la URL de archivo .json. (Figura A.1.8)

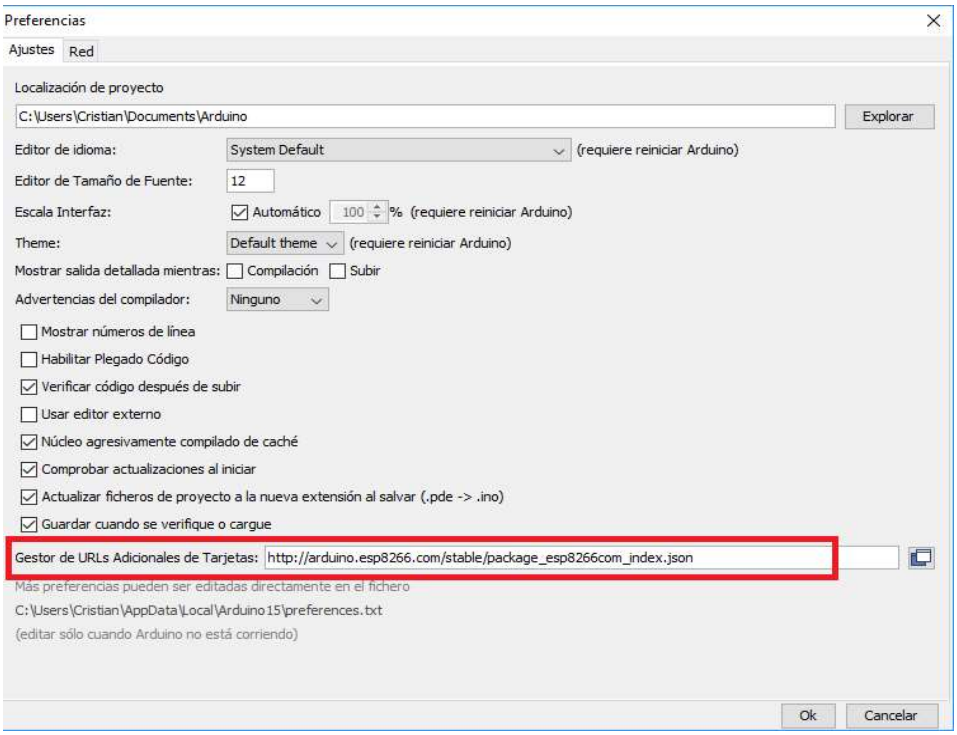


Figura A.1. 8 - Pantalla preferencias aplicación Arduino

10. Se ha de acceder a la pestaña “Herramientas”, hacer clic en la opción “Placa” y por ultimo hacer clic en “Gestor de tarjetas…” para añadir una nueva tarjeta (Figura A.1.9).

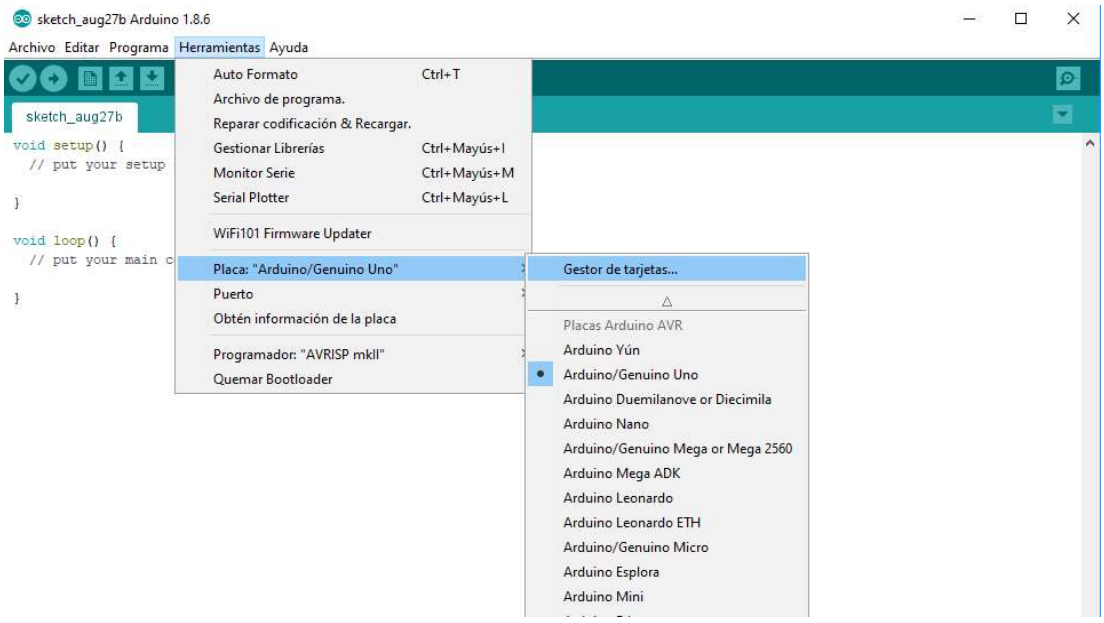


Figura A.1. 9 - Selección gestor de tarjetas

11. Se ha de insertar en el buscador del gestor de tarjetas el nombre del paquete que incorpora la tarjeta NodeMCU. Exactamente el paquete que se ha de buscar es “esp8266”. Por último se ha de seleccionar el paquete “esp8266 by ESP8266 Community” y se ha de pulsar en instalar.(Figura A.1.10)

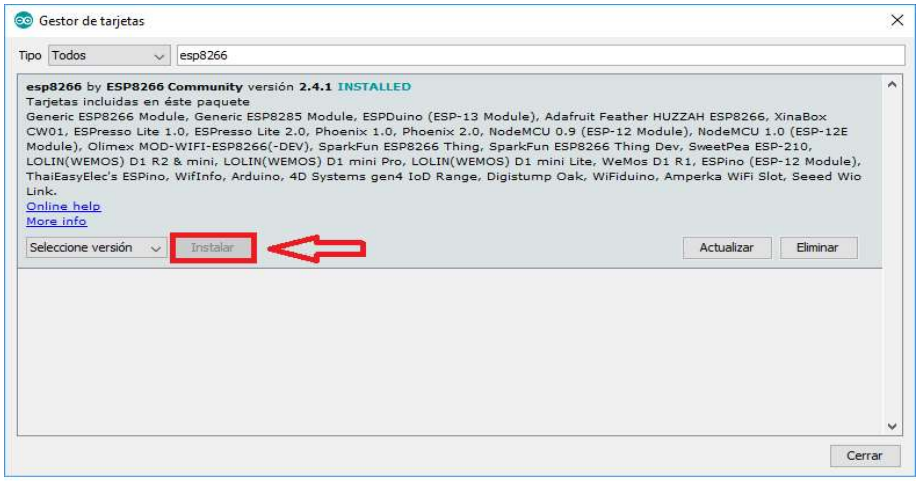


Figura A.1. 10 - Instalación paquete de tarjetas ESP8266

12. Una vez instalado el soporte para las tarjetas NodeMCU se accede a la pestaña “Herramientas”, se hace clic sobre la opción “Placa” y se selecciona la nueva tarjeta con la que se desea trabajar, en este caso NodeMCU 1.0 (ESP-12E Module), tal y como se muestra en la figura A.1.11

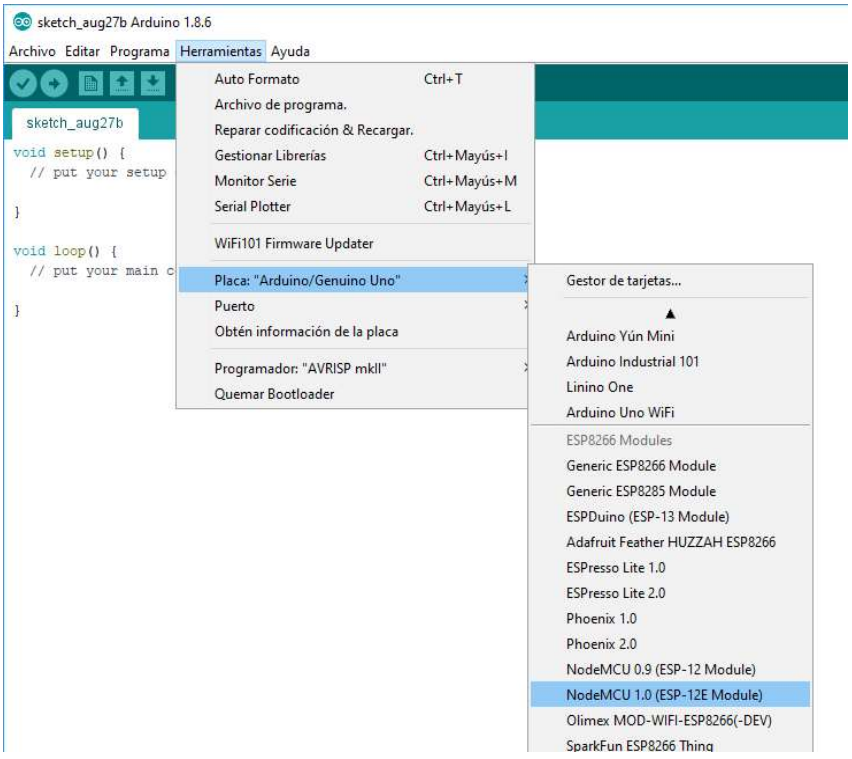


Figura A.1. 11 - Selección tarjeta NodeMC

Anexo 2: Instalación Librería PubSubClient de MQTT en Arduino IDE

Para que la tarjeta NodeMCU trabaje como un cliente MQTT y se pueda comunicar con el servidor MQTT se necesita instalar la librería PubSubClient, proceso que se realiza desde el IDE de Arduino. Para ello se deben seguir los siguientes pasos:

1. Se ha de iniciar la aplicación Arduino.
2. Se ha de hacer clic en la pestaña “Programa”, seleccionar la opción “Incluir Librerías” y por ultimo hacer clic en “Gestionar Librerías”. (Figura A.2.1)

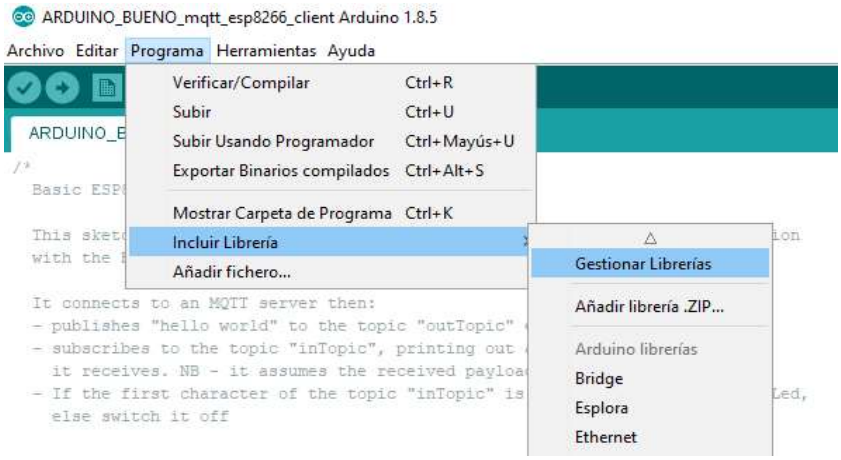


Figura A.2. 1 Selección de opción incluir librerias

3. Se ha de introducir en el buscador del gestor de librerías el nombre de la librería que se desea instalar, concretamente se ha de introducir “PubSubClient”. Tras esto, se ha de seleccionar la librería “PubSubClient by Nick O’Leary” y se presiona el botón “Instalar”, tal y como se muestra en la figura A.2.2

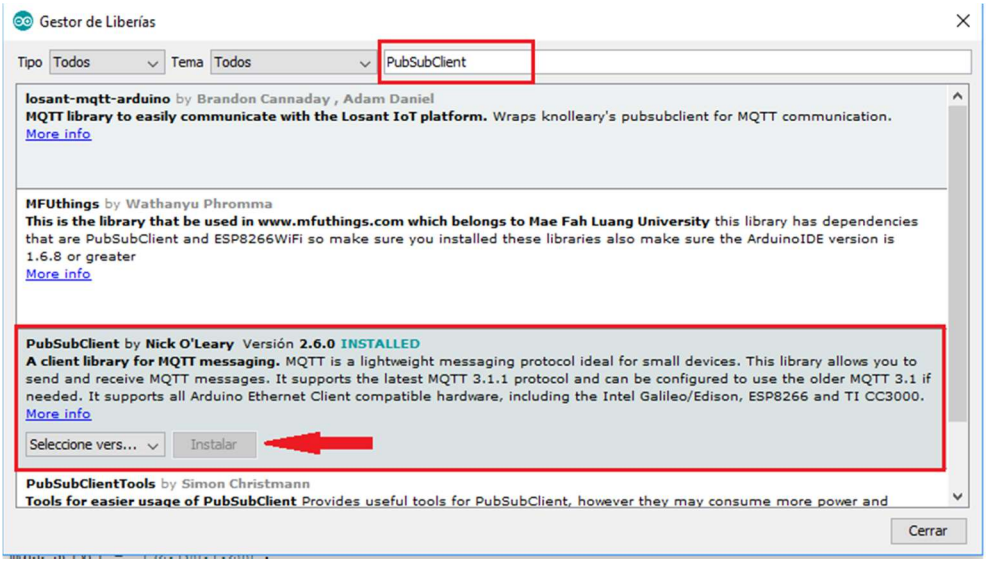


Figura A.2. 2 - Instalación librería PubSubClient de MQTT

4. Una vez instalada la librería se puede acceder a diversos ejemplos para crear clientes MQTT. Para acceder a los ejemplos se hace clic sobre la pestaña “Archivo”, se selecciona la opción “Ejemplos”, se busca la librería “PubSubClient” instalada anteriormente y se elige el ejemplo que se desea abrir. (Figura A.2.3)

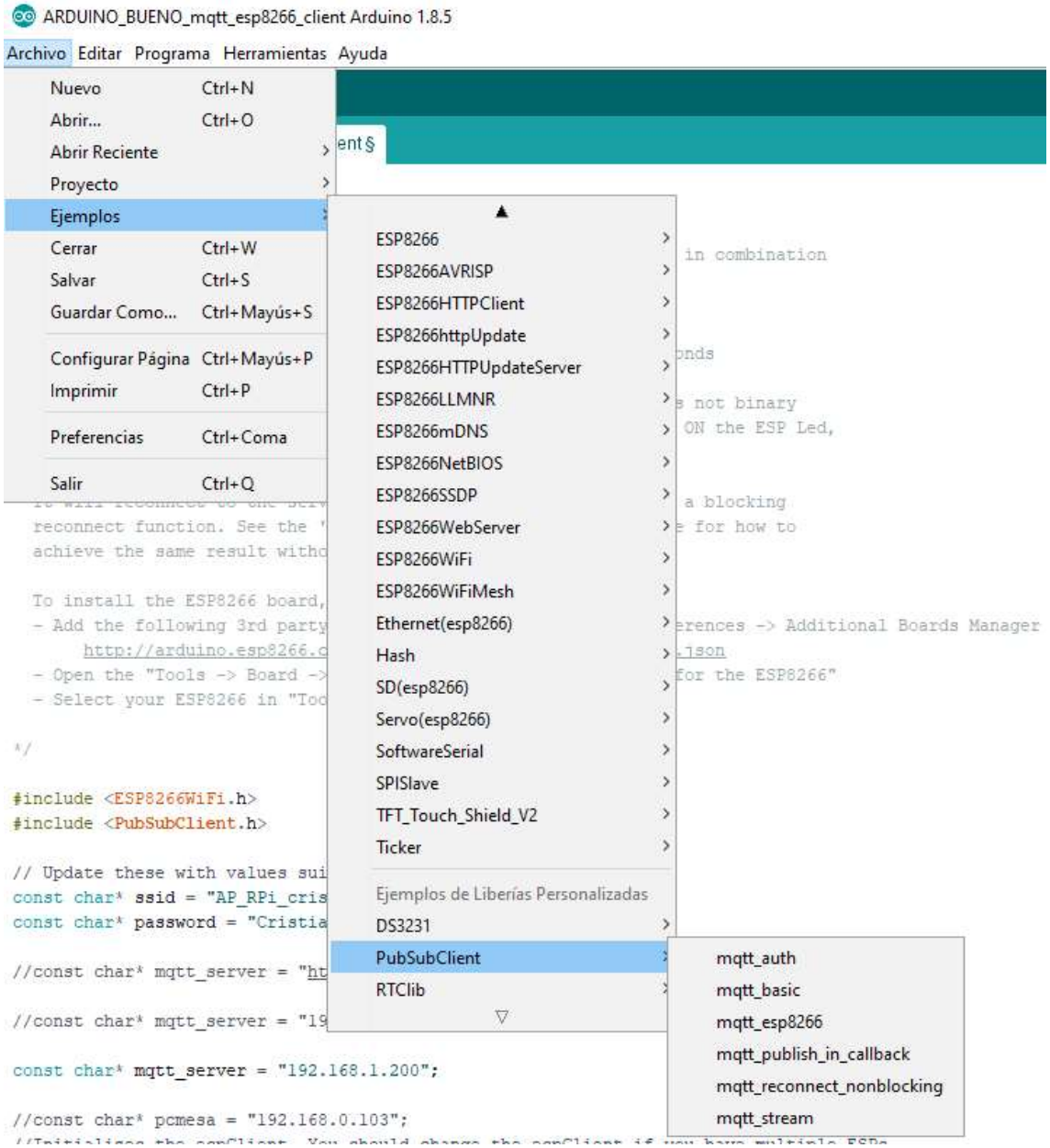


Figura A.2. 3 - Selección ejemplos librería MQTT

Anexo 3: Instalación Sistema Operativo Raspbian

Para instalar el sistema operativo (de aquí en adelante SO) en una Raspberry Pi se ha de contar al menos de una tarjeta MicrosSD de 8 GB de memoria y clase 6 virgen, ya que esta será el soporte físico encargado de almacenar el sistema operativo. También se podría usar una tarjeta MiscroSD de mayor capacidad y clase (p.e 16 GB clase 10).

Existen varios sistemas operativos para Raspberry Pi dependiendo de las necesidades de cada usuario. Algunos SO vienen preparados para tareas específicas, por ejemplo RetroPie que se usa para la emulación de videojuegos o OpenELEC, si lo que se desea montar un MediaCenter.

En este proyecto se va a instalar SO Raspbian que es el sistema operativo oficial y de propósito general recomendado en la página web del fabricante.

Para realizar la instalación del sistema operativo en la tarjeta SD antes mencionada desde un ordenador con sistema operativo Windows se han seguir los siguientes pasos:

1. Se ha de visitar la página oficial de descargas de Raspberry Pi. https://www.raspberrypi.org/downloads/ (Figura A.3.1)

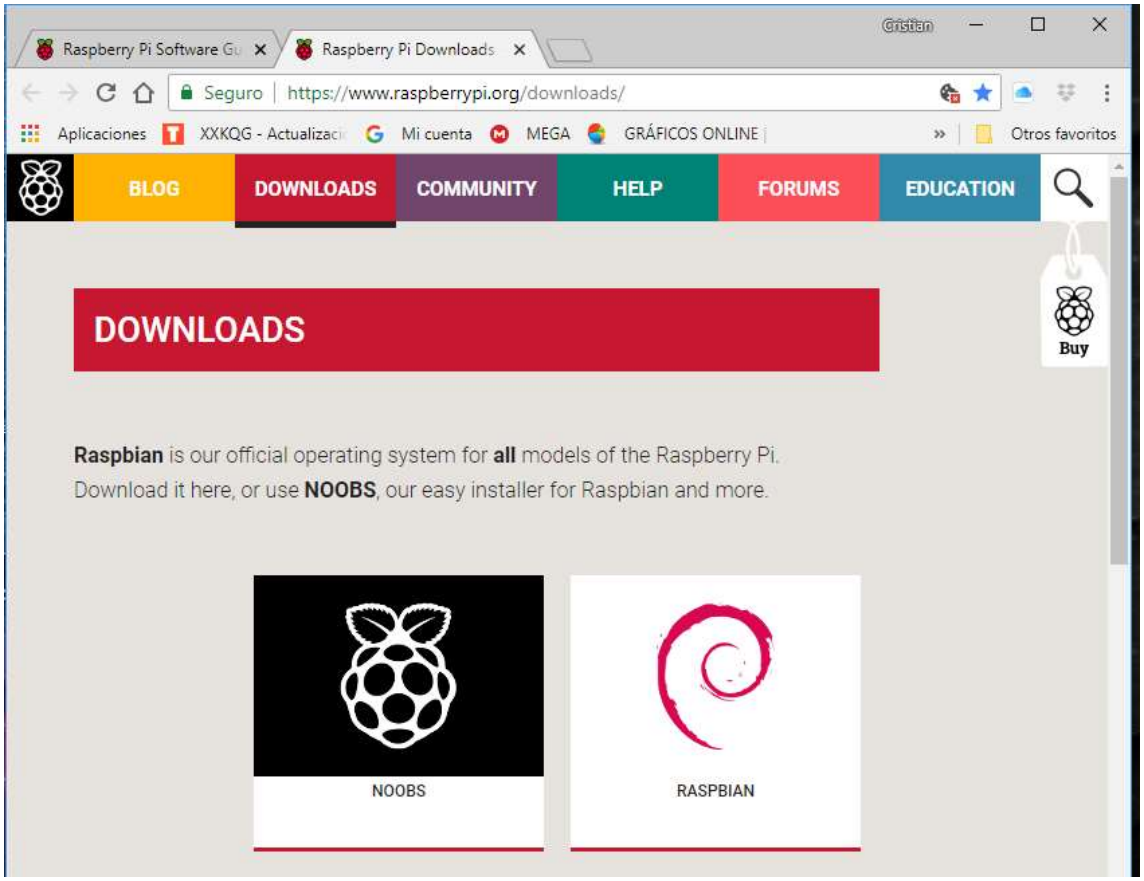


Figura A.3. 1 Página web de Raspberry Pi

2. Se ha de hacer clic en RASPBIAN y después se ha de hacer clic en el botón “Donwload ZIP” debajo de “Raspbian Stretch with Desktop” (figura A.3.2), ya que es la opción de SO que cuenta con entorno de escritorio resultando más cómoda para el usuario.

Figura A.3. 2 - Descarga Raspbian Stretch

3. Una vez descargado Raspbian se descomprime el archivo .zip y se guarda la imagen del sistema (archivo .img).
4. Se ha de insertar una tarjeta MicroSD en el ordenador utilizando para ello un adaptador MicroSD a SD si fuera necesario. NOTA: El proceso anteriormente descrito no funciona con un adaptador MicrosSD a USB.
5. Se ha de formatear la tarjeta MicroSD, para ello se utiliza el programa “SD Card Formatter” (Figura A.3.3). Este programa se puede descargar desde la siguiente URL https://www.sdcard.org/downloads/formatter_4/index.html

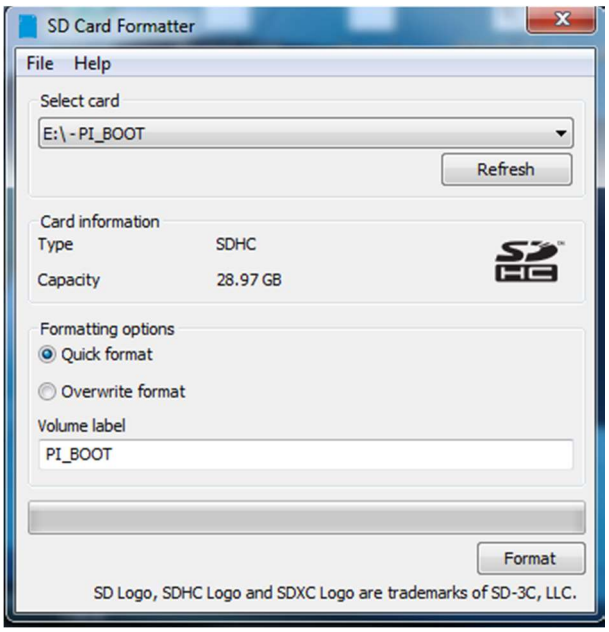


Figura A.3. 3 - Pantalla principal aplicación SD Card Formatter

6. Se ha de descargar el programa Win32DiskImager, ya que es el programa utilizado para grabar la imagen del sistema operativo en la tarjeta MicroSD. Este programa se puede descargar de https://sourceforge.net/projects/win32diskimager/
7. Se ha de iniciar el programa y hacer clic en el icono de la tarjeta. (Figura A.3.4)
8. En la ventana emergente se busca y selecciona la imagen de Raspbian descargada anteriormente.
9. En “Device” se necesita seleccionar la letra que el sistema (Windows) le ha asignado a la tarjeta MicroSD. (Figura A.3.4)
10. Se ha de hacer clic en el botón “Write” para comenzar a grabar el sistema operativo. (Figura A.3.4)

Figura A.3. 4 - Selección e instalación del SO en la tarjeta MicroSD

11. Se ha de confirmar el proceso de instalación (figura A.3.5) y proporcionar permisos de usuario si así fuese necesario.

Figura A.3. 5 - Confirmación del proceso de instalación

12. Se ha de esperar mientras el programa finalice el proceso de grabación. (Figura A.3.6). Al finalizar aparecerá una ventana emergente indicando que el proceso ha terminado y se podrá extraer la tarjeta. (Figura A.3.7)

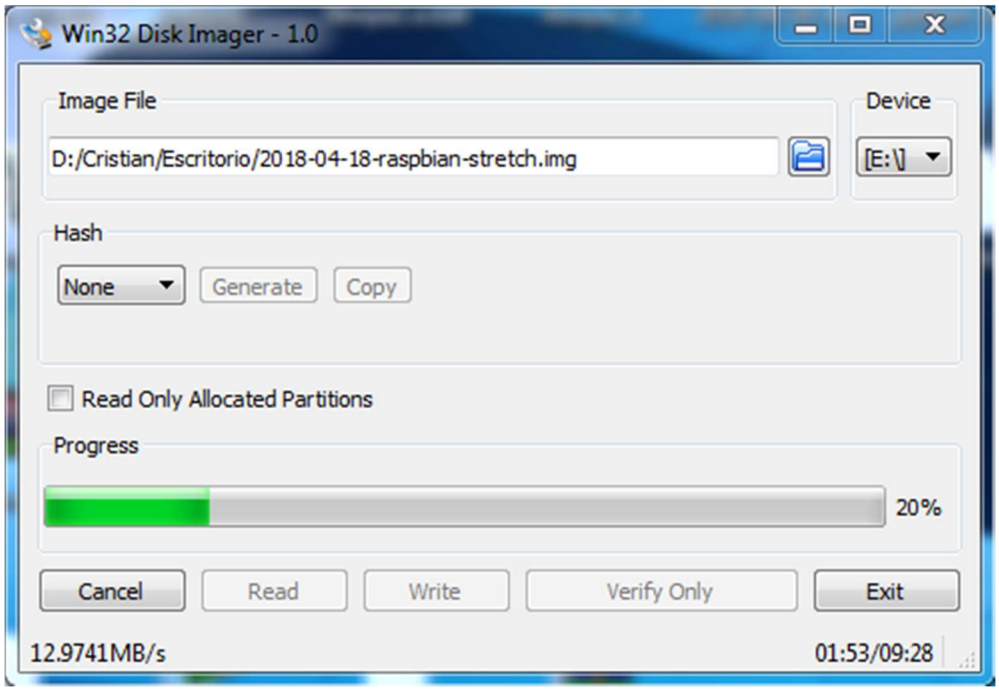


Figura A.3. 6 - Proceso de instalación

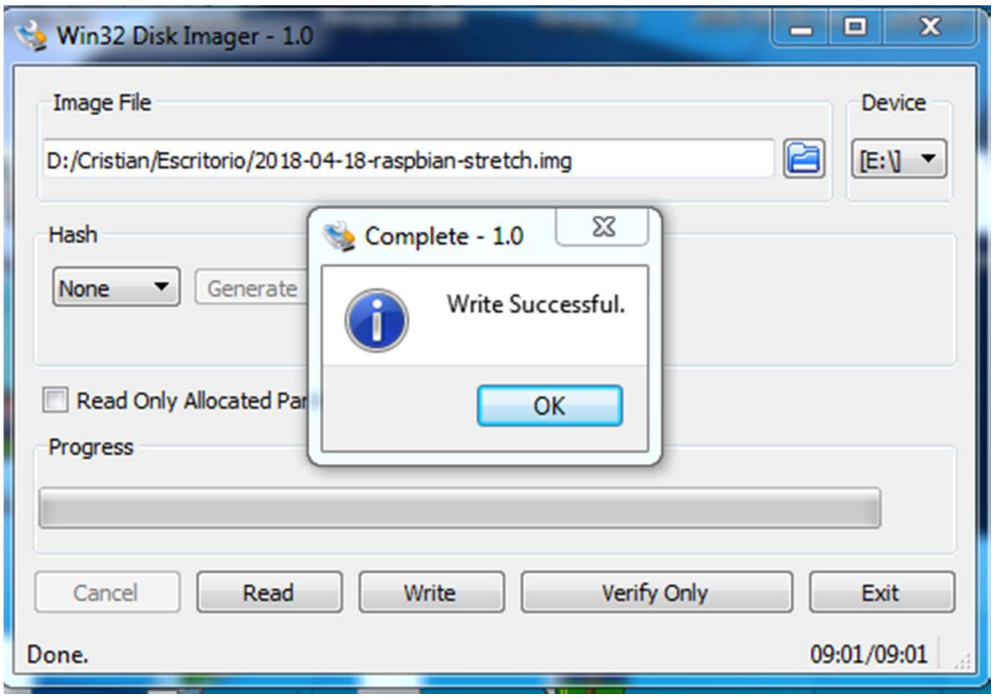


Figura A.3. 7 - Pantalla informativa indicando que le proceso de instalación se ha realizado con éxito

13. Por último, se ha de conectar la tarjeta MicroSD en el interior de la Raspberry Pi y realizar el primer inicio del sistema operativo, proceso que se detalla de forma sucinta a continuación:

Primer Arranque de Raspberry Pi

Antes de iniciar por primera vez el dispositivo Raspberry Pi es necesario realizar los siguientes pasos:

1. Se ha de introducir la tarjeta Micro SD preparada anteriormente en el tarjetero habilitado a tal efecto en el Raspberry Pi.
2. Se ha de conectar un teclado con puerto USB.
3. Se ha de conectar una pantalla con puerto HDMI.
4. Al ser un sistema operativo con interfaz gráfica y disponer de escritorio se necesita conectar también un ratón con puerto USB.
5. Se ha de conectar un cable de red en el puerto RJ-45 para poder tener acceso a internet.
6. Por último se procede a alimentar el dispositivo a través del puerto microUSB con un cargador o fuente de alimentación de 5 V y 2 A.

Al encenderse por primera vez el equipo se mostrará por pantalla el proceso de carga del sistema operativo. Una vez el arranque finalice se mostrará el escritorio del sistema operativo Raspbian. (Figura A.3.8).

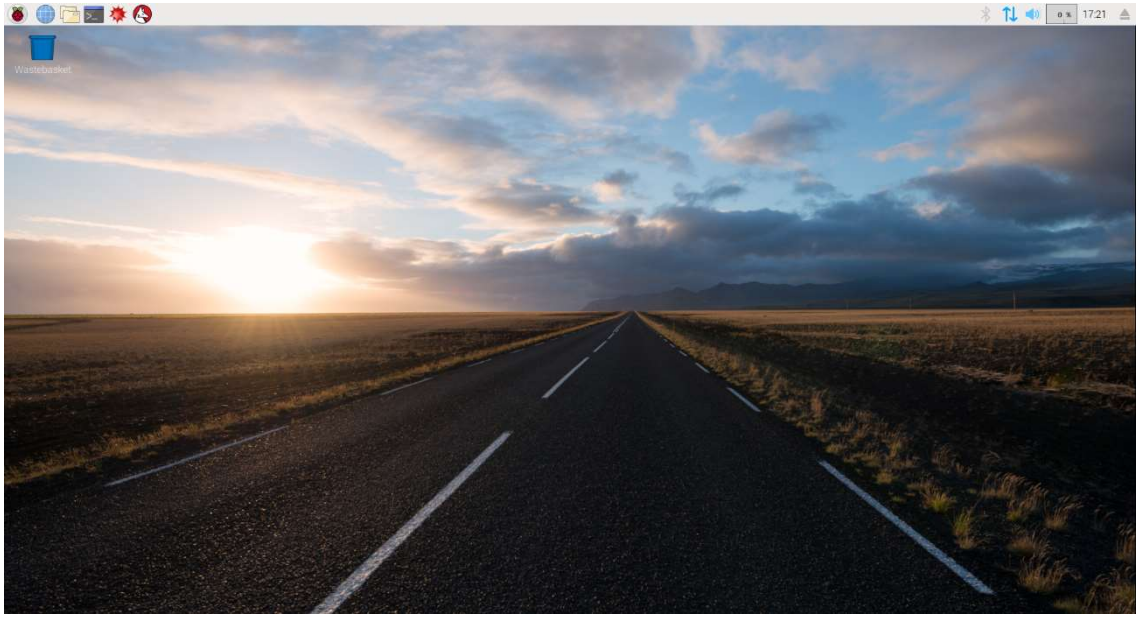


Figura A.3. 8 - Captura de pantalla del entorno de escritorio Raspbian

Anexo 4: Instalación Infraestructura LAMP

Una infraestructura LAMP es el acrónimo de:

- L: Linux, que corresponde al sistema operativo.
- A: Apache, que corresponde al servidor web.
- M: MySQL / MariaDB, que es el gestor de bases de datos.
- P: PHP / Perl / Python, que corresponde al lenguaje de programación, normalmente usado en el lado del servidor.

Antes de comenzar con la instalación de los distintos componentes de la infraestructura LAMP siempre es recomendable actualizar la Raspberry Pi y el sistema operativo. Para ello se ejecuta en la terminal (Figura A.4.1) los siguientes comandos:

Para actualizar la Raspberry Pi:

```
sudo rpi-update
```

Para actualizar el sistema:

```
sudo apt update
```

y tras esto:

```
sudo apt upgrade
```

Cuando se ejecuta el comando *upgrade* (Figura A.4.2) y el sistema pregunta si se desea continuar, se pulsa la tecla s, para confirmar y después la tecla *enter* para seguir con la descarga.

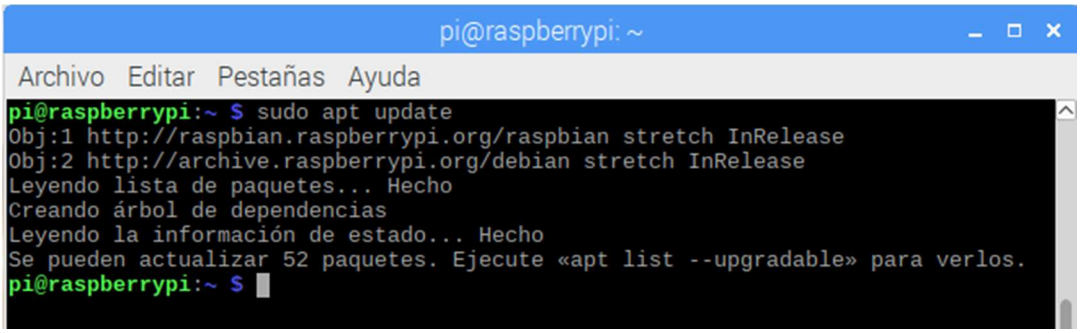


Figura A.4. 1 - Ejecución del comando update en un terminal de la Raspberry Pi

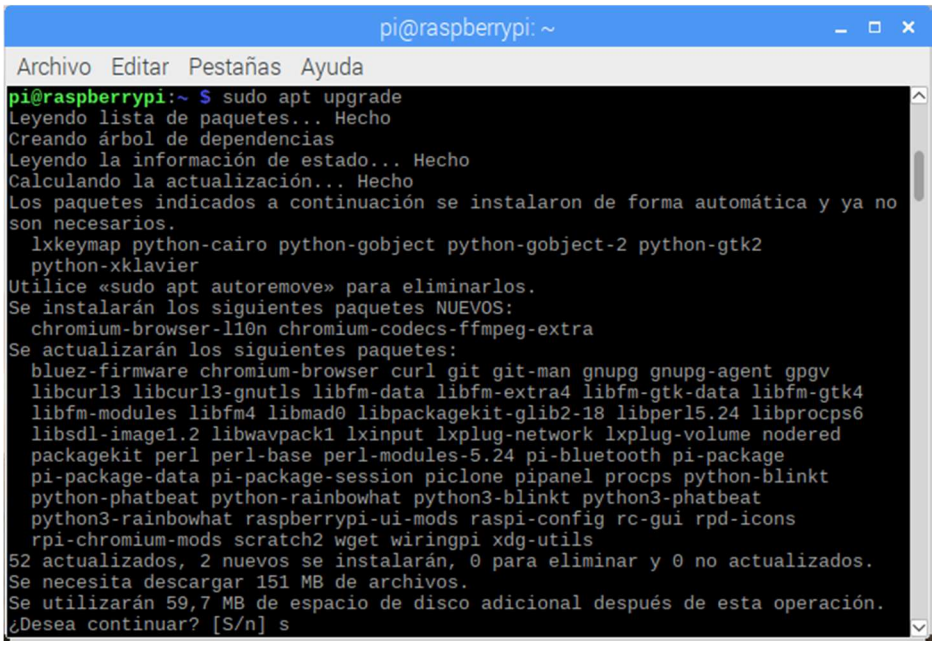


Figura A.4. 2 - Ejecución del comando upgrade en un terminal de la Raspberry Pi

Una vez actualizado el sistema se procede a asignar una dirección IP estática a la Raspberry Pi para que sea más fácil acceder de forma remota vía SSH, VNC o a la hora de configurar el dispositivo como servidor Web.

Configuración IP estática

Para configurar una IP estática en el SO Raspbian se debe editar el fichero "dhcpcd.conf" con la ayuda del siguiente comando desde el terminal:

sudo nano /etc/dhcpcd.conf

Dentro de este fichero se pueden observar unas líneas de código comentadas con el símbolo # las cuales incluyen un ejemplo de una configuración de red con IP estática. (Figura A.4.3).

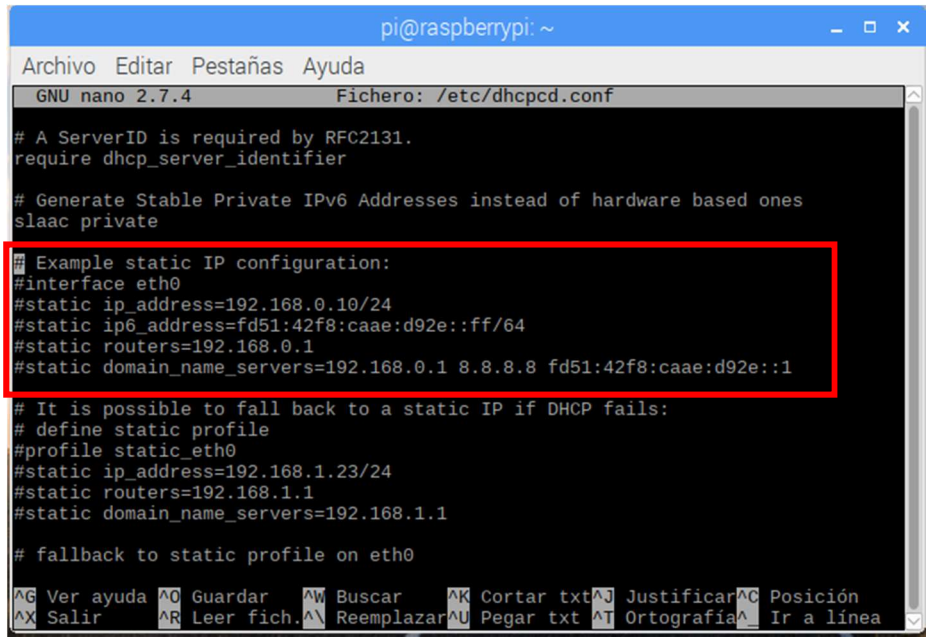


Figura A.4. 3 - Archivo configuración IP estática

Para asignar una dirección IP estática propia se elimina el comentario del fragmento anterior y se modifica para que quede de la siguiente manera:

Interface eth0
Static ip_address=192.168.0.200/24
Static routers=192.168.0.1
Static domain_name_servers=192.168.0.1 8.8.8.8

Donde cada uno de los parámetros hace referencia a:

- Interface = Nombre de la interfaz que se desea configurar
- static ip_address = Dirección fija deseada (dejar el /24 al final ya que es la máscara de subred)
- static routers = Dirección del gateweay (del router)
- static domain_name_servers = Dirección del servidor DNS (normalmente la IP del router). Si es necesario más de un servidor DNS se pueden añadir separándolos por un espacio.

A continuación se guarda el archivo pulsando “Ctrl+O”, se sale del editor nano mediante “Ctrl+X”. Tras esto se reinicia la Raspberry Pi para aplicar la nueva configuración. Para ello se ejecuta el siguiente comando desde el terminal:

sudo reboot

Para comprobar si la configuración se ha realizado correctamente se ejecuta el siguiente comando:

ifconfig eth0

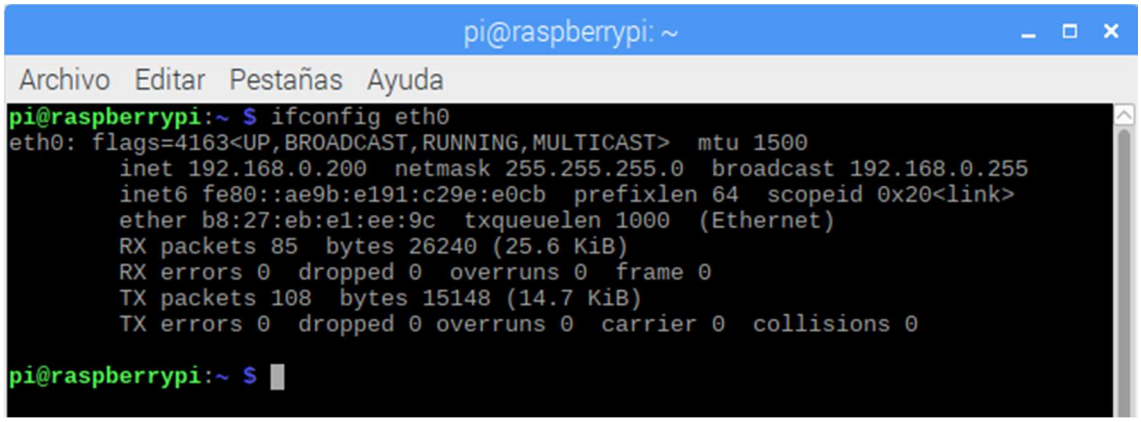


Figura A.4. 4 - Resultado de ejecutar el comando ifconfig eth0 tras configurar el sistema con una IP estática

Como se puede observar en la Figura A.4.4 el proceso de configuración se ha realizado correctamente y la dirección IP es la deseada.

Instalación del servidor web Apache

Para realizar la instalación del servidor web Apache se ejecuta el siguiente comando desde el terminal:

sudo apt-get install apache2 -y

Una vez realizada la instalación se puede comprobar el funcionamiento de Apache abriendo un navegador web y escribir en la barra de direcciones http://localhost o ejecutar el comando:

sudo systemctl status apache2

Tras ejecutar este comando el sistema devuelve por pantalla el estado (activo/inactivo) del servidor web Apache.

Se procede a cambiar la raíz del documento de inicio de Apache, para disponer de todos los permisos necesarios a la hora de crear archivos php, html, etc. Para ello se crea una carpeta en el directorio raíz "/home/pi" llamada "Public_html", después se ejecutan los siguientes comandos en el terminal:

sudo nano /etc/apache2/sites-enables/000-default.conf

En el archivo 000-default.conf (que a continuación aparecerá por pantalla) se busca el término “DocumentRoot” (Figura A.4.5) y se sustituye la ruta que viene por defecto agregando el siguiente código:

```
DocumentRoot /home/pi/Public_html

<Directory /home/pi/Public_html>
        Options Indexes FollowSymLinks
        AllowOverride all
        Require all granted
</Directory>
```

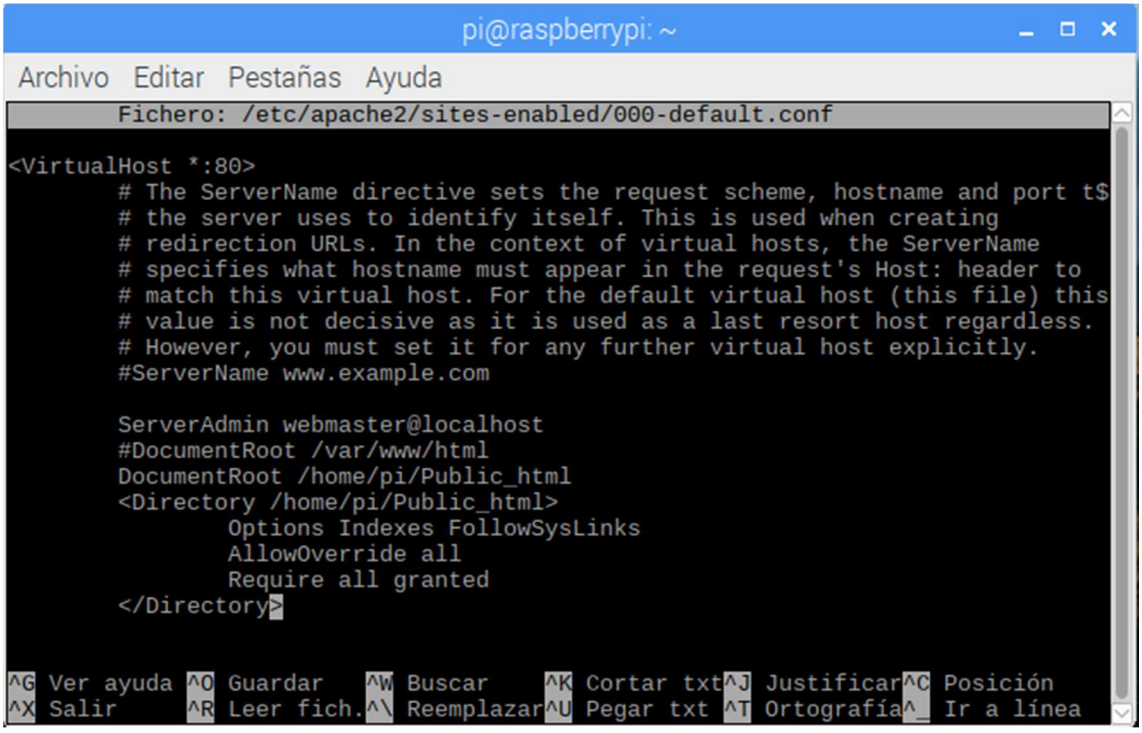


Figura A.4. 5 - Archivo de configuración 000-default.conf una vez editado

Por último se procede a cambiar el usuario de ejecución (run user) que tiene el servidor web Apache por defecto, para ello se ejecuta el comando:

```
sudo nano /etc/apache2/envvars
```

Se busca el término “APACHE_RUN_USER=www-data” y se sustituye por el usuario deseado, quedando el código como se muestra en la Figura A.4.6.

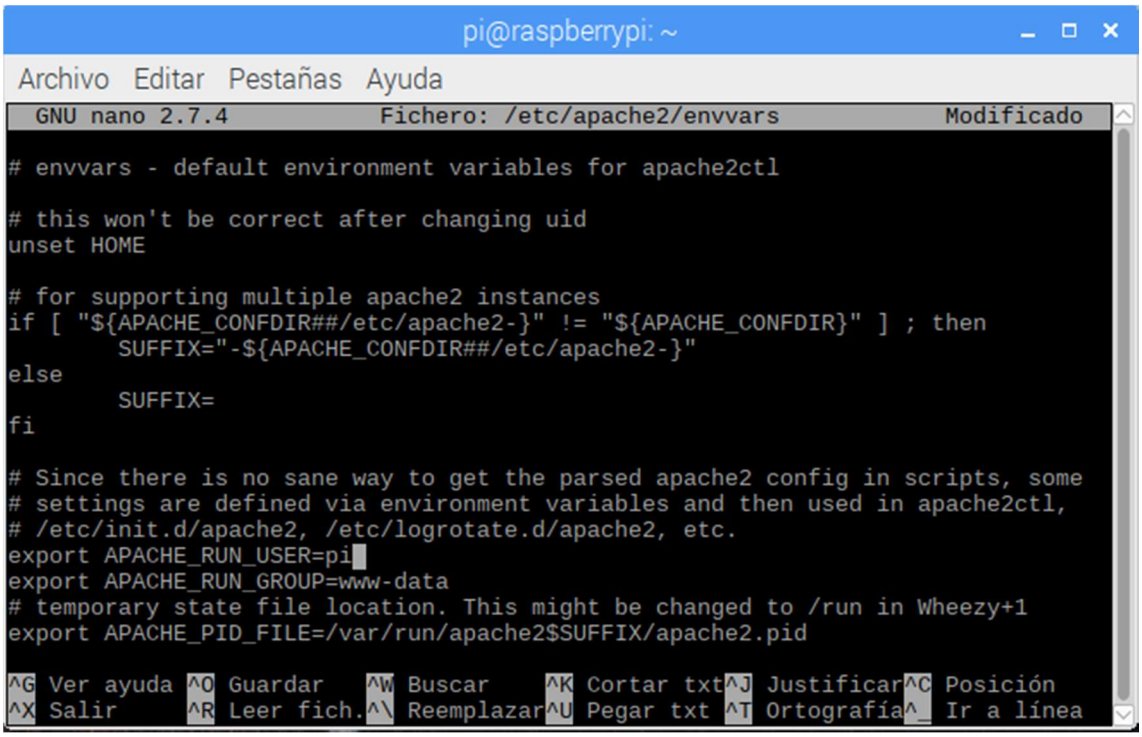


Figura A.4. 6 - Modificación usuario de ejecución del servidor web Apache

Finalmente se reinicia el servidor web Apache mediante el siguiente comando:

```
sudo systemctl restart apache2
```

Instalación del gestor de bases de datos MySQL

Para instalar el gestor de bases de datos MySQL se ejecuta el siguiente comando desde el terminal:

```
sudo apt-get install mysql-client mysql-server
```

Para comprobar su funcionamiento se utiliza el comando:

```
sudo systemctl status mysql
```

Tras este comando el sistema devuelve el estado (activo) del gestor de bases de datos MySQL.

Una vez instalado el gestor de base de datos es necesario instalar el programa “mysql secure” utilizado para mejorar la seguridad del gestor MySQL. Este proceso se realizará ejecutando el siguiente comando desde el terminal:

```
sudo mysql_secure_installation
```

Después de ejecutar el comando anterior se han de llevar a cabo los siguientes pasos para realizar una correcta configuración de mysql:

- Se ha de pulsar la tecla “intro” para introducir la contraseña del usuario “root”.
- Se ha de pulsar “Y” para crear una contraseña para el usuario “root”.
- Se ha de escribir dos veces la nueva contraseña del usuario “root”.
- Se ha de pulsar “Y” para borrar el usuario anónimo por defecto.
- Se ha de pulsar “Y” para deshabilitar el acceso remoto del usuario “root”.
- Se ha de pulsar “Y” para borrar la base de datos “test” que viene por defecto.
- Se ha de pulsar “Y” para volver a cargar la tabla de los privilegios y que los cambios realizados hasta el momento tengan efecto inmediato. (Figura A.4.7)

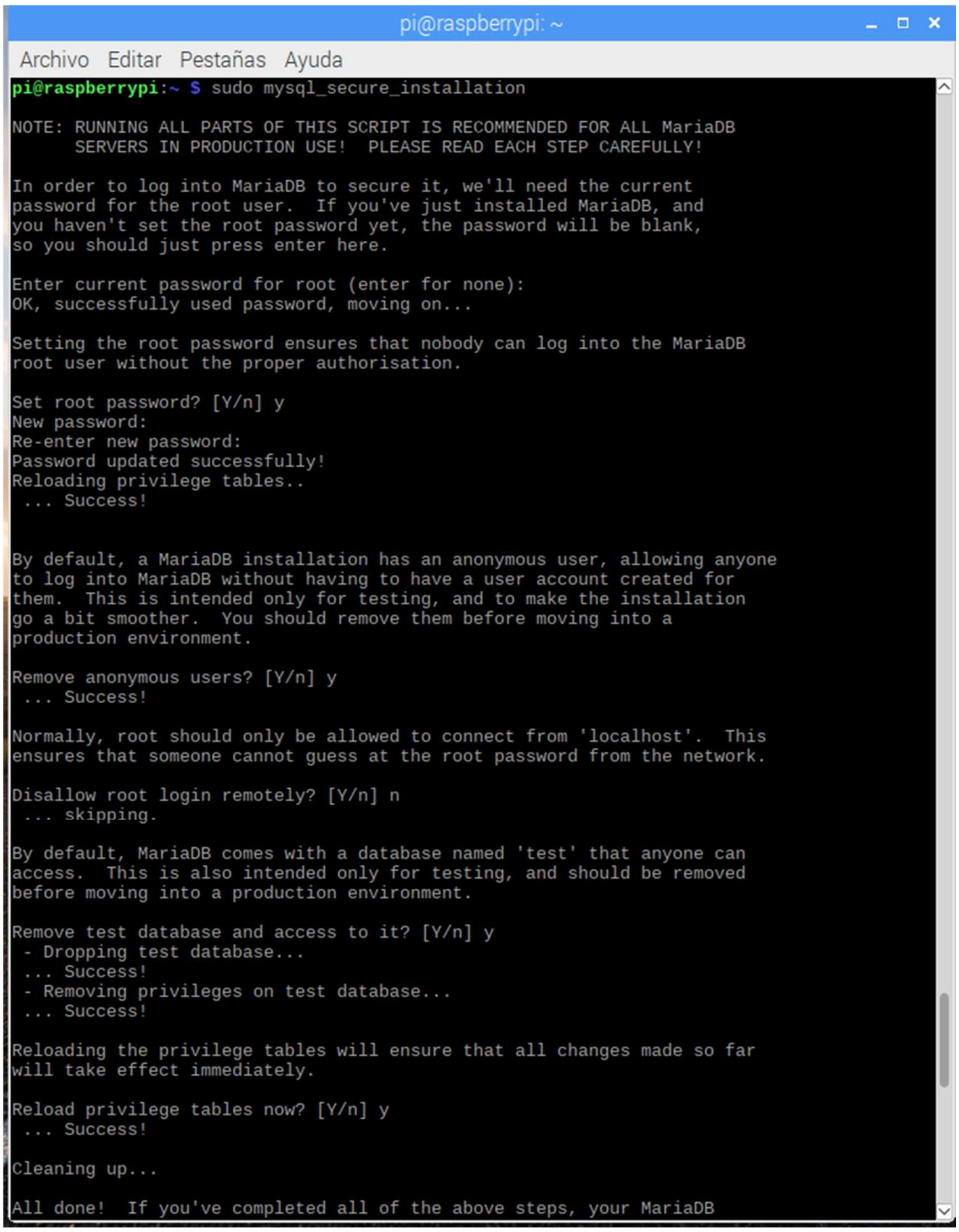


Figura A.4. 7 - Configuración de MySQL Secure

El siguiente paso (Figura A.4.8) consiste en crear un nuevo usuario para no tener que iniciar sesión con el usuario “phpmyadmin” o con el usuario “root” ya que son los usuarios que se incluyen por defecto en la configuración y pueden acarrear problemas de seguridad. Para ello en el terminal de Raspbian se escribe el siguiente comando:

sudo mysql –u root –p

Se introduce la contraseña del usuario “root” configurada en el paso anterior.

Si se desea utilizar el usuario “pi”, usuario por defecto de la Raspberry Pi, para la base de datos y para phpmyadmin, solo se necesita otorgar todos los privilegios sobre toda la base de datos para dicho usuario. Para ello se ejecuta el siguiente comando:

grant all privileges on mybd.* to pi@localhost identified by ‘contraseña’;

Para crear un usuario nuevo dentro de Mysql/MariaDB se ejecuta:

create user ‘cristian’@’localhost’ identified by ‘contraseña;

En el paso anterior se ha creado el usuario cristian y asignado una contraseña.

Por último se le otorga todos los privilegios al usuario cristian mediante el siguiente comando:

grant all privileges on *.* to ‘cristian’@’localhost’;

flush privileges;

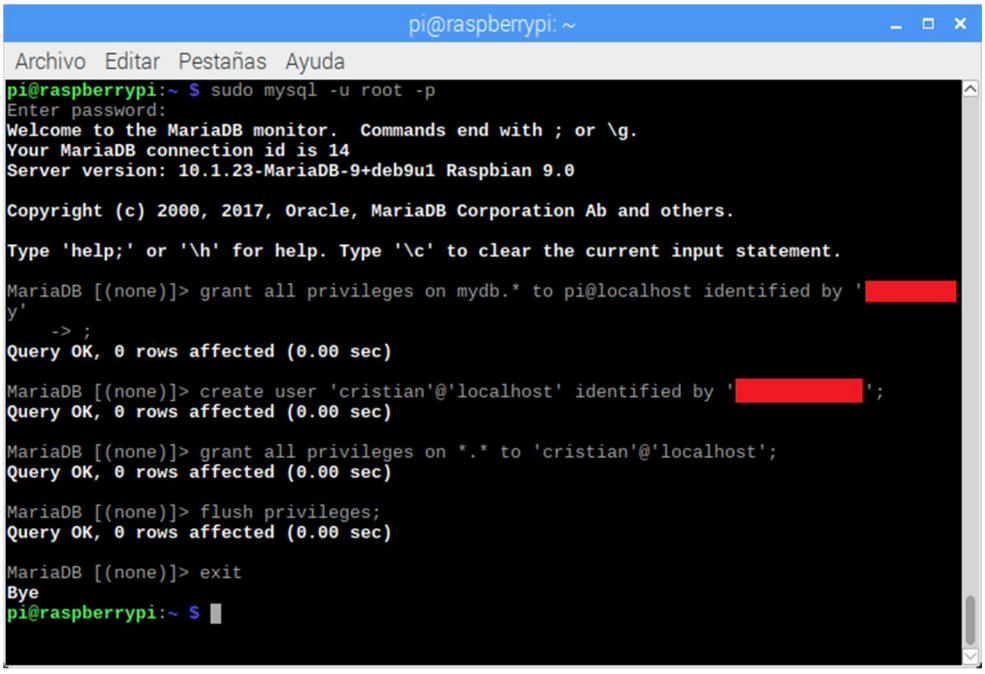


Figura A.4. 8 - Creación de un nuevo usuario para MySQL

Instalación PHP: Hypertext Preproccesor

Para instalar el intérprete de código php se ha de ejecutar en el terminal de Raspbian el siguiente comando:

sudo apt-get install php libapache2-mod-php –y

Para comprobar el funcionamiento de php se crea un fichero php (info.php) dentro de la carpeta “/home/Public_html” y se agrega en su interior código php que proporciona información sobre la versión de PHP instalada en el equipo.

sudo nano /home/Public_html/info.php

Dentro del archivo se agrega el siguiente código:

```
<?php
    Phpinfo();
?>
```

Tras esto se de iniciar el navegador web y se debe accede a la URL http://localhost/info.php. (Figura A.4.9)

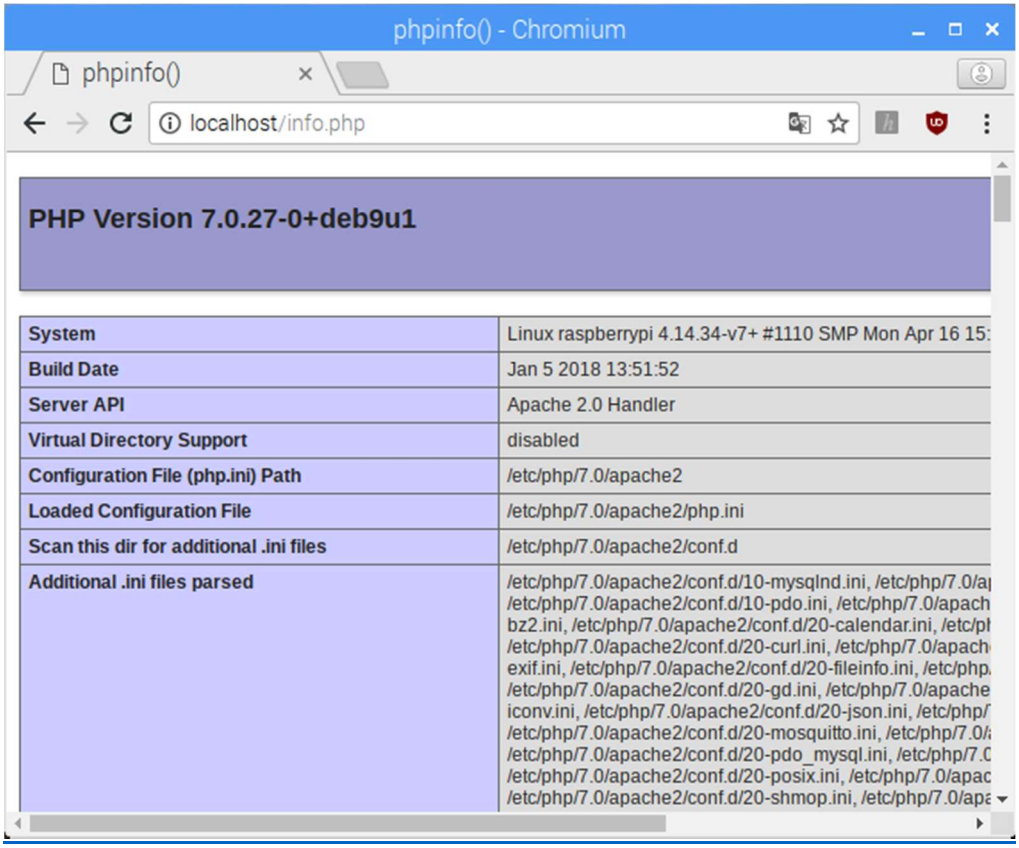


Figura A.4. 9 - Archivo info.php visto desde el navegador Chromium

Instalación Phpmyadmin

PhpMyAdmin es una herramienta escrita en PHP encargada de manejar la administración de MySQL a través de páginas web.

Para instalar phpmyadmin se ejecuta el siguiente comando desde el terminal:

sudo apt-get install phpmyadmin

Durante la instalación de phpmyadmin aparecerá una ventana donde es necesario seleccionar el tipo de servidor web instalado. Se selecciona Apache pulsando la barra espaciadora del teclado y después con la ayuda del tabulador se selecciona <Aceptar> y se pulsa la tecla intro. (Figura A.4.10)

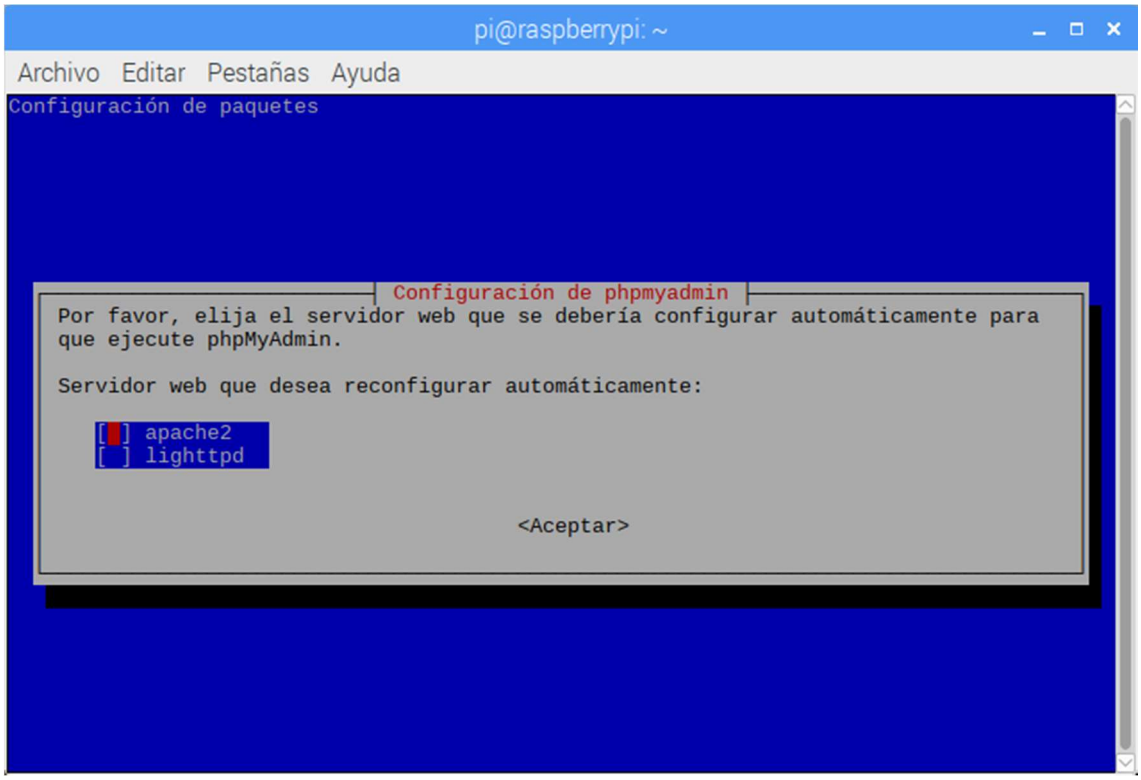


Figura A.4. 10 - Selección servidor para phpmyadmin

A continuación se pregunta si se desea configurar la base de datos con “dbconfig-common”, se confirma seleccionando <Sí>.

Por último en el menú de configuración (Figura A.4.11) se necesita introducir una contraseña para phpmyadmin, se teclea la contraseña dos veces y se selecciona <Aceptar>.

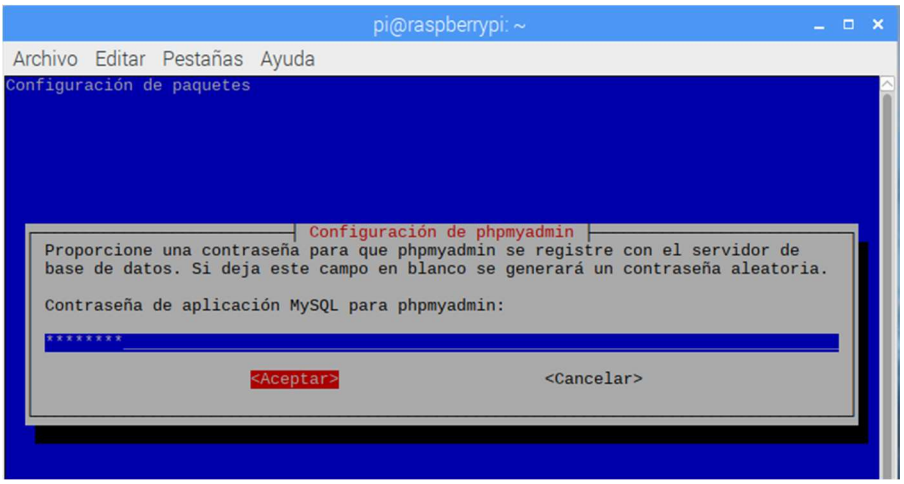


Figura A.4. 11 - Proceso de configuración de la contraseña para Phpmyadmin

Para comprobar que phpmyadmin se ha instalado correctamente se inicia el navegador web y se teclea en la barra de direcciones http://localhost/phpmyadmin. (Figura A.4.12)

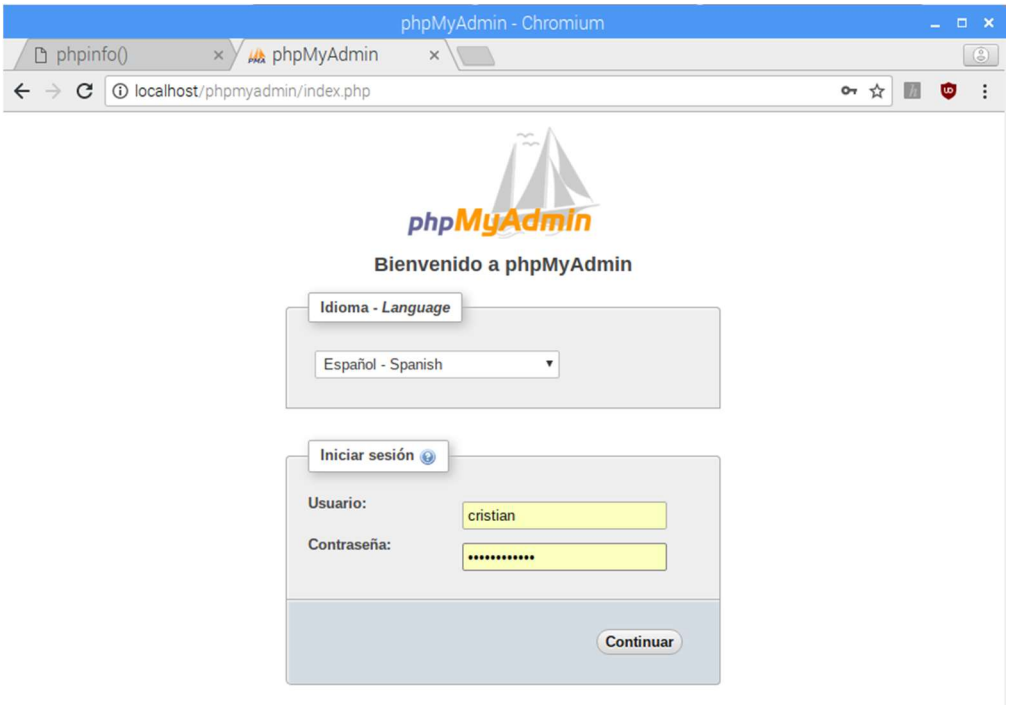


Figura A.4. 12 - Inicio sesión PhpMyAdmin

Se puede comprobar como si se intenta iniciar sesión con el usuario “root” no es posible, ya que a la hora de configurar mysql se ha deshabilitado esa opción. Esto es una nueva función del gestor de bases de datos MariqDB, es la última actualización que ha incluido phpMyAdmin y añade una seguridad adicional.

Anexo 5: Instalación del servidor MQTT Mosquitto

Para instalar el servidor (también llamado *broker*) MQTT Mosquitto en la plataforma de desarrollo Raspberry Pi, es necesario abrir la consola de comandos del sistema operativo Raspbian y ejecutar los siguientes comandos:

```
Sudo apt-get update

Sudo apt-get install mosquitto mosquitto-clients
```

Estos comandos actualizan el sistema de gestión de paquetes (APT) e instalan el servidor Mosquitto.

Si al ejecutar los comandos anteriores sucede algún problema o el servidor no se instala correctamente entonces, se necesita seguir los siguientes pasos:

1. Importar la clave de firma del repositorio de paquetes Mosquitto. Para ello se ejecuta el siguiente comando:
   ```
   Sudo wget http://repo.mosquitto.org/debian/mosquitto-repo.gpg.key
   ```
2. Añadir la firma al sistema de gestión de paquetes (APT):
   ```
   Sudo apt-get add mosquitto-repo.gpg.key
   ```
3. Añadir los repositorios de Mosquitto al sistema de gestión de paquetes y después descargar la lista de repositorios:
   ```
   Cd /etc/apt/sources.list.d/
   Sudo wget http://mosquitto.org/debian/mosquitto-stretch.list
   ```
4. Finalmente, instalar el servidor Mosquitto a través de los siguientes comandos:
   ```
   Sudo apt-get update
   Sudo apt-get install mosquitto mosquitto-clients
   ```

Pruebas Realizadas

Para comprobar que la instalación del servidor Mosquitto se ha realizado correctamente se ha llevado a cabo una prueba en la que un cliente se suscribe a un determinado *topic* y otro cliente publica un mensaje sobre ese *topic*. Para simular a los dos clientes MQTT se han utilizado dos terminales de comandos.

Un cliente se suscribirá al topic “test/mosquitto” y el segundo cliente publicara un mensaje sobre el mismo topic. La prueba será exitosa si el mensaje enviado por el publicador se ve reflejado en el terminal del suscriptor.

Ensayo funcional del cliente suscriptor

Para iniciar el suscriptor, se ejecuta en el terminal del sistema de desarrollo Raspberry Pi (Figura A.5.1):

mosquitto_sub –d –t ‘test/mosquitto’

La opción “–d” se utiliza para habilitar el modo de depuración y la opción “–t” permite especificar el topic.

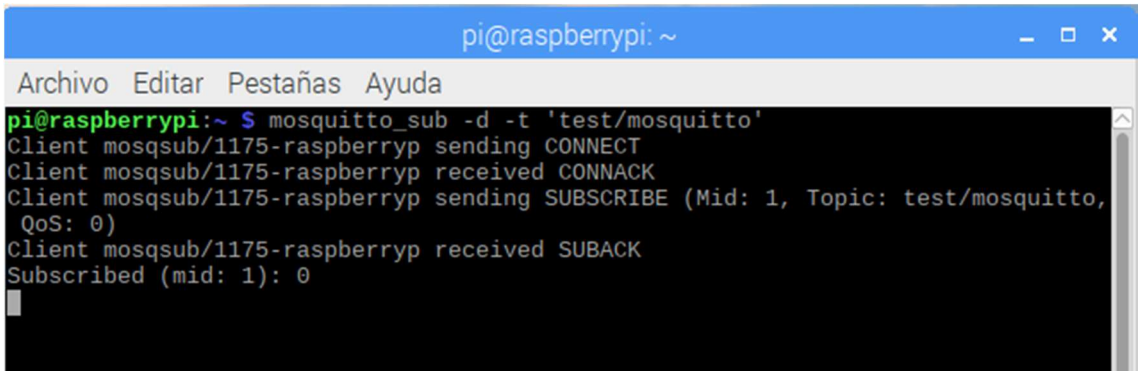


Figura A.5. 1 - Terminal del cliente suscriptor

Ensayo funcional del cliente publicador

Para publicar un mensaje se ejecuta el comando:

mosquitto_pub –d –t ‘test/mosquitto’ –m ‘Este es un mensaje de prueba’

La opción “–m” se utiliza para agregar el mensaje (Figura A.5.2).

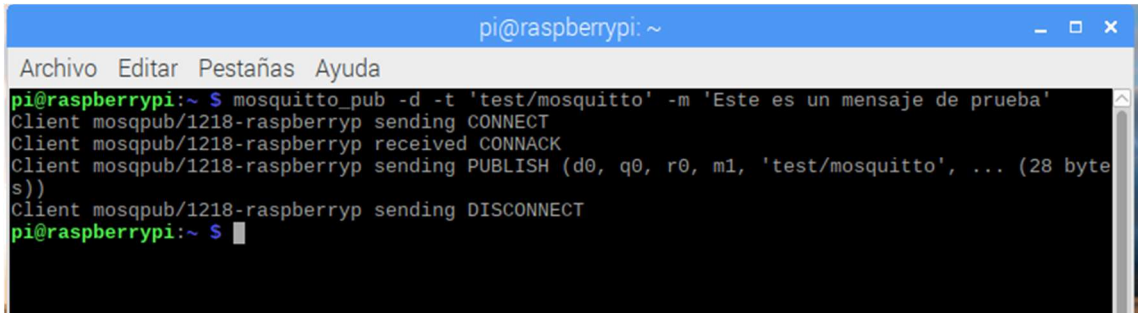


Figura A.5. 2 - Terminal del cliente publicador

Como se puede observar en la figura A.5.3 la prueba ha tenido éxito ya que se ha recibido el mensaje enviado en el terminal del cliente suscriptor.

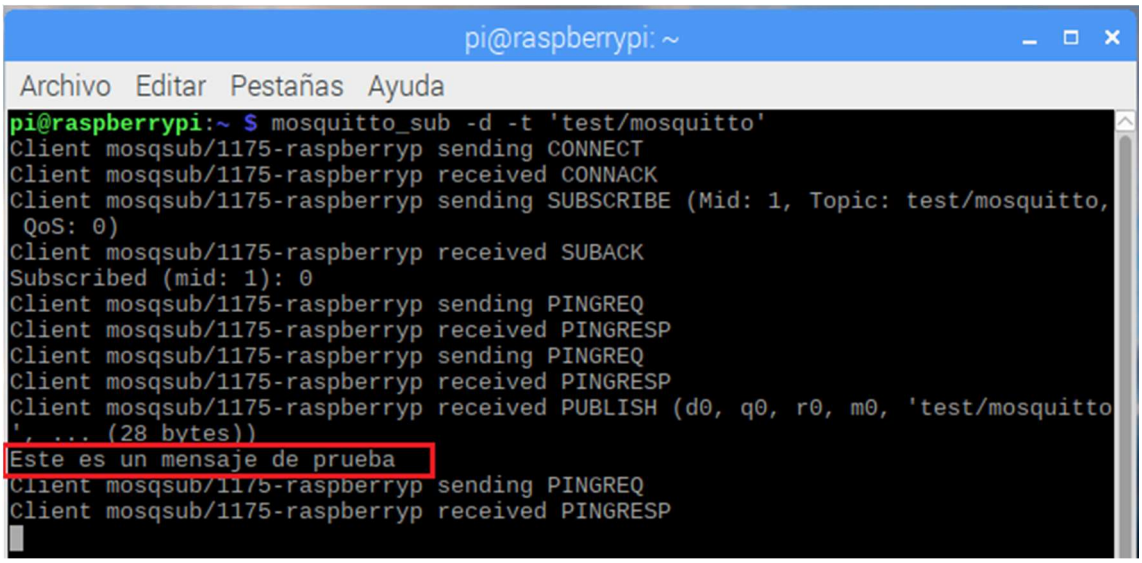


Figura A.5. 3 - Mensaje recibido por parte del cliente suscriptor

Instalación librerías PHP Mosquitto

Para poder conectar a través de código PHP con el *broker* MQTT Mosquitto es preciso tener instalado y habilitado el módulo de desarrollo de las librerías Mosquitto PHP. Para ello se realizan los siguientes pasos:

1. Se ha de instalar librerías Mosquitto desde el terminal mediante el comando:

   sudo apt install libmosquitto-dev
2. Se ha de crear el archivo “mosquitto.ini” dentro del directorio “mods-available” y añadir la siguiente línea en el interior del archivo: “extension=mosquitto.so”

   sudo nano /etc/php/7.0/mods-available/mosquitto.ini

   extension=mosquitto.so
3. Se ha de guardar el archivo y salir del editor de textos.
4. Se ha de comprobar si la librería se ha instalado correctamente ejecutando el comando:

   dpkg –l | grep mosquitto.

   Como se puede observar en la figura A.5.4 resultado de la ejecución del ultimo comando desde el terminal, en la primera línea se indica que la librería se encuentra.

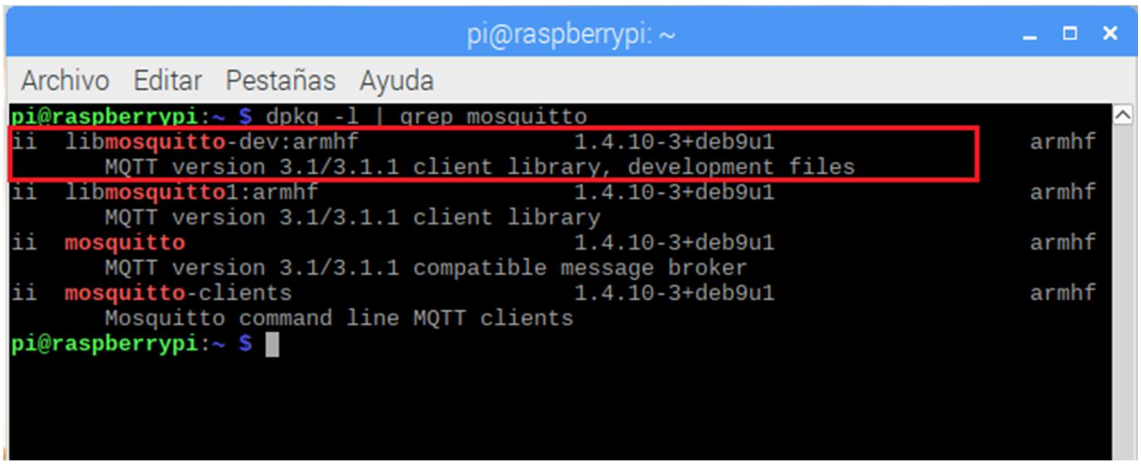


Figura A.5. 4 - Resultado de la ejecucion en el terminal del comando dpkg –l | grep mosquitto

5. Se ha de habilitar el módulo Mosquitto ejecutando el siguiente comando:

   sudo phpenmod mosquitto

6. Por último es necesario reiniciar el servidor web apache:

   sudo service apache2 restart

Con la configuración descrita anteriormente ya se puede iniciar una comunicación con el *broker* Mosquitto mediante archivos PHP. Para realizar una prueba se ha ejecutado un archivo php llamado “pubon.php”, el cual envía el mensaje “Hola desde PHP” sobre el *topic* “test/php”, (Figura A.5.5, A.5.6 y A.5.7) para ello se ha ejecutado el siguiente comando:

php /home/pi/Public_html/pubon.php

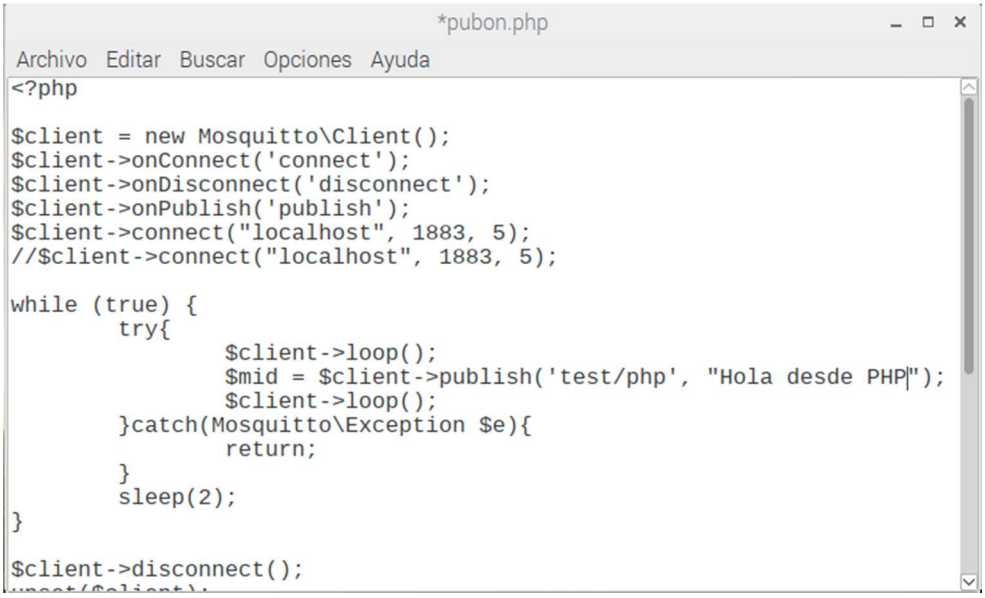


Figura A.5. 5 - Archivo pubon.php

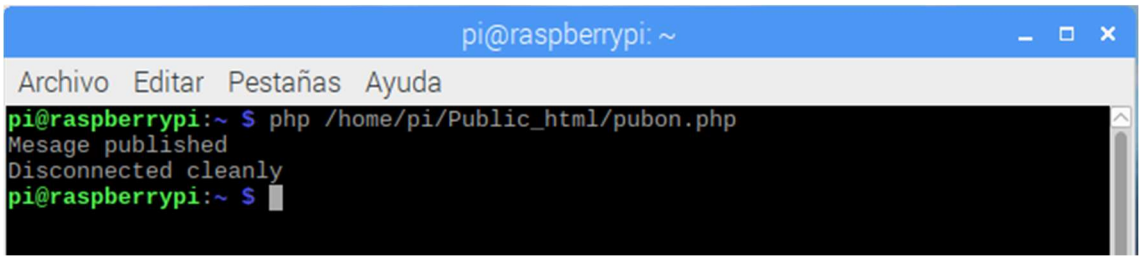


Figura A.5. 6 - Comando para ejecutar archivo php

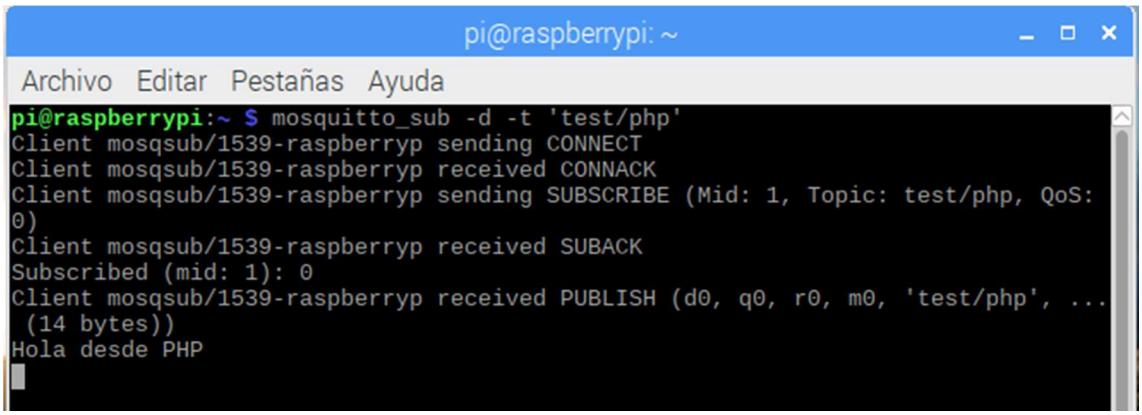


Figura A.5. 7 - Mensaje recibido desde PHP

Anexo 6: Instalación AP – Punto de Acceso

Raspberry Pi se puede utilizar como punto de acceso inalámbrico para ejecutar una red independiente. Para ello, Raspberry Pi necesitará tener instalado un software de AP y un software de servidor DHCP, para proporcionar a los dispositivos de conexión una dirección de red (IP).

Para iniciar la instalación del software, se abre una terminal de comando y se ejecuta el siguiente comando:

```
sudo apt-get install dnsmasq hostapd
```

Dado que los archivos de configuración del software instalado anteriormente no están debidamente configurados, es necesario detener los servicios de dnsmasq y hostapd. Para ello:

```
sudo systemctl stop dnsmasq

sudo systemctl stop hostapd
```

Para un correcto funcionamiento del punto de acceso se necesitan realizar las siguientes configuraciones:

Configuración IP Estática Puerto Inalámbrico wlan0

Se entiende que se está configurando una red independiente para que actúe como un servidor DHCP, por lo que la Raspberry Pi necesita tener una dirección IP estática asignada al puerto inalámbrico.

Para configura una IP estática en la interfaz inalámbrica es necesario editar el archivo de configuración “dhcpcd.conf”, para ello se ejecuta el siguiente comando:

```
sudo nano /etc/dhcpcd.conf
```

Se añaden las siguientes líneas:

```
interface wlan0

static ip_address = 192.168.4.1/24
```

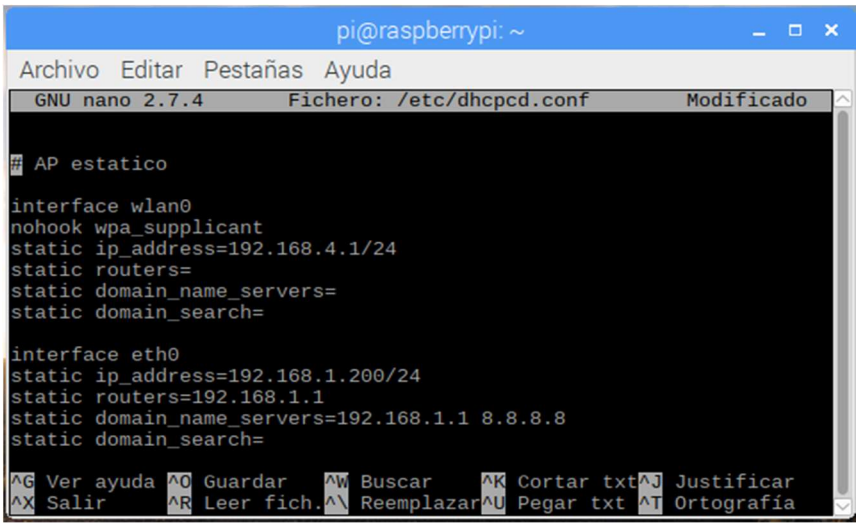


Figura A.6. 1 - Configuracion IP estatica wlan0

Una vez añadido el código es necesario reiniciar el servicio dhcpcd para aplicar la configuración:

sudo service dhcpcd restart

Configuración del Servidor DHCP (dnsmasq)

De forma predeterminada el archivo de configuración dnsmasq.conf contiene mucha información innecesaria, por lo que es más fácil comenzar desde cero. Para esto cambiamos el nombre del archivo de configuración y editamos uno nuevo.

sudo mv /etc/dnsmasq.conf /etc/dnsmasq.conf.original

sudo nano /etc/dnsmasq.conf

Se añade la siguiente información en el archivo de configuración dnsmasq:

interface = wlan0

dhcp-range = 192.168.4.2, 192.168.2.20, 255.255.255.0, 24h

De este modo el servidor DHCP proporciona direcciones IP entre 192.168.4.2 y 192.168.4.20, con un tiempo de alquiler de 24 horas. Si se quiere proporcionar servicios DHCP para otros dispositivos de red, por ejemplo dispositivos de red cableada (eth0), se puede agregar más secciones con el encabezado de interfaz apropiado, con el rango de direcciones que se pretende proporcionar a esa interfaz.

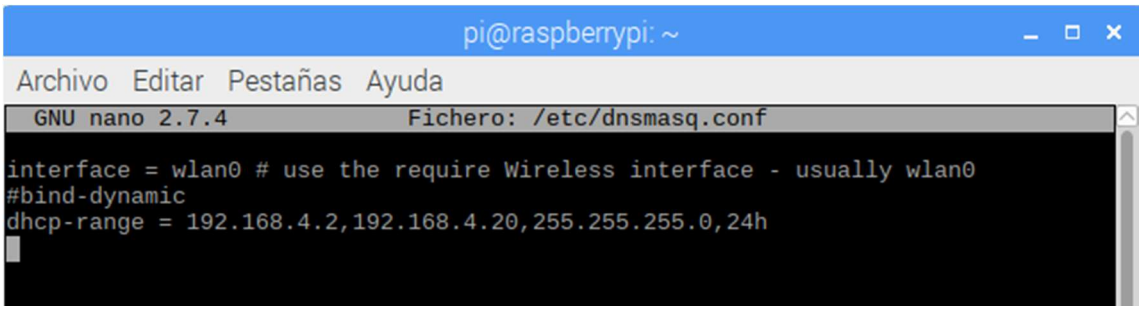


Figura A.6. 2 - Configuración dnsmaq

Configuración del Software del Punto de Acceso (hostapd)

Es necesario editar el archivo de configuración hostapd.conf para agregar diversos parámetros de red. Después de la instalación inicial este será un archivo nuevo/vacío.

Sudo nano /etc/hostapd/hostapd.conf

Al abrir el archivo se agrega la siguiente información (en el campo ssid (nombre de red) y wpa_passphrase (contraseña) se configura según preferencia):

interface = wlan0
driver = nl80211
ssid = NombreDeLaRed
hw_mode = g
channel = 7
wmm_enabled = 0
macaddr_acl = 0
auth_algs = 1
ignore_broadcast_ssid = 0
wpa = 2
wpa_passphrase = AardvarkBadgerHedgehog
wpa_key_mgmt = WPA-PSK
wpa_pairwise = TKIP
rsn_pairwise = CCMP

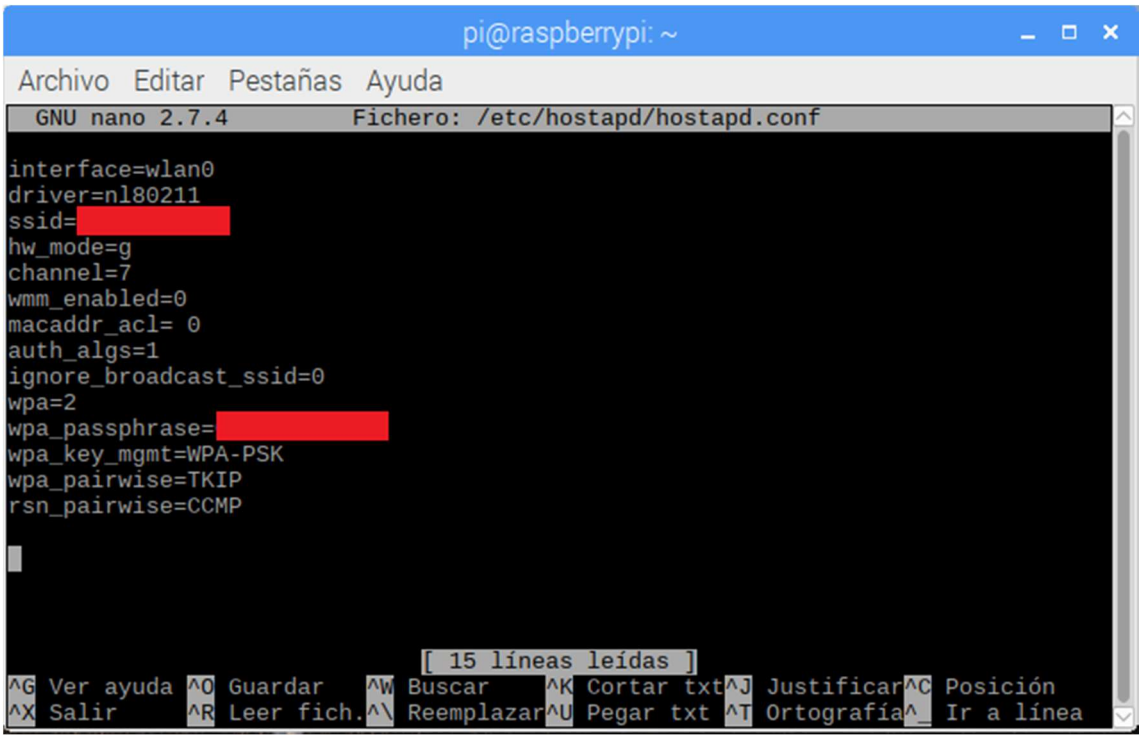


Figura A.6. 3 - Configuración parámetros de red inalámbrica

En este paso es necesario comunicarle al sistema la ubicación del archivo de configuración, para ello se ejecuta el siguiente comando:

sudo nano /etc/default/hostapd

Se busca la línea #DAEMON_CONF y se reemplaza con la siguiente información:

DAEMON_CONF = “/etc/hostapd/hostapd.conf”

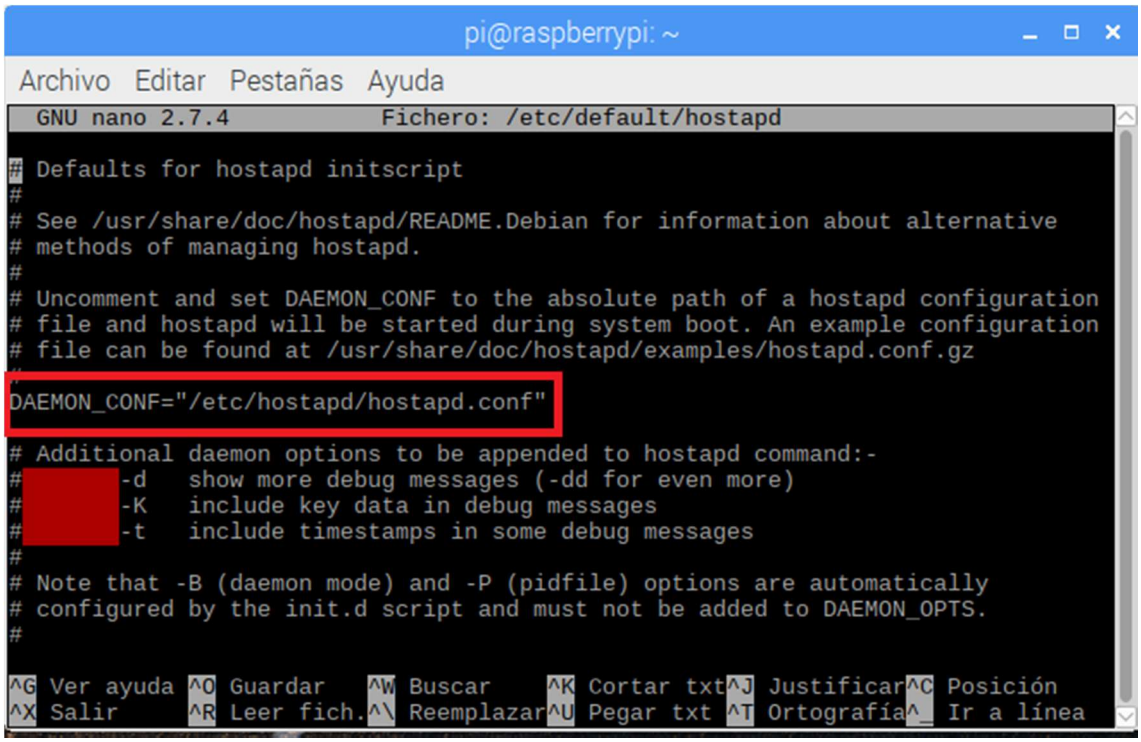


Figura A.6. 4 - Sustitución de la ubicación del archivo de configuración hostapd

Añadir enrutamiento y enmascaramiento de tráfico (“masquerade”)

Es necesario editar el archivo sysctl.conf y eliminar el comentario (#) de la siguiente línea: Net.ipv4.ip_forward = 1. Para ello se ejecuta:

sudo nano /etc/sysctl.conf

Se añade enmascaramiento para el tráfico saliente en la interfaz eth0:

sudo iptables –t nat –A POSTROUTING –o eth0 –j MASQUERADE

Se guarda la regla añadida de iptables:

sudo sh –c “iptables-save> /etc/iptables.ipv4.nat”

Se edita el archivo “rc.local” y se añade la siguiente información encima de la línea “exit 0” para instalar la regla anterior en el arranque del sistema:

sudo nano /etc/rc.local

ip tables-restore</etc/iptables.ipv4.nat

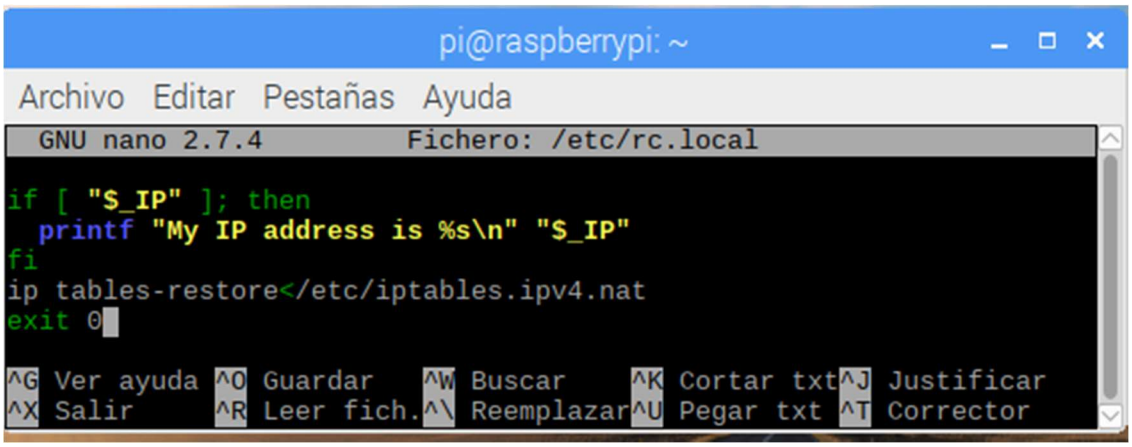


Figura A.6. 5 - Inserción regla de enmascaramiento de trafico al arranque del sistema

Por último se reinicia por completo la Raspberry Pi para que se apliquen todos los cambios.

```
sudo reboot
```

Usando un dispositivo inalámbrico, se procede a buscar redes inalámbricas. El SSID de red que se especificó en la configuración de hostapd ahora debería estar presente y debería poder accederse con la contraseña especificada.

En este punto ha ocurrido un error ya que la red WiFi configurada no se ha podido encontrar. Para solucionar este problema es necesario deshabilitar el servicio wpa_Supplicant en la interfaz inalámbrica wlan0. Esto es debido a que wpa_supplicant es un servicio para autenticar clientes en un punto de acceso y hostapd es un servicio que ejecuta la interfaz inalámbrica como punto de acceso que recibirá solicitudes de autenticación de los clientes. Ejecutar wpa_supplicant y hostapd en una misma interfaz no tiene sentido y el punto de acceso configurado no funcionara.

Para deshabilitar el servicio es necesario acceder al archivo de configuración de dhcpcd.conf y añadir “nohook wpa_supplicant” después de la interfaz wlan0, dejo el ejemplo en la siguiente línea:

```
sudo nano /etc/dhcpcd.conf
interface wlan0
nohook wpa_supplicant
static_ip_address=192.168.4.1/24
```

Otro procedimiento para deshabilitar el servicio es acceder a “/etc/wpa_supplicant/wpa_supplicant.conf” y editar el archivo comentando la línea que

comienza con “ctrl_interface”, o comentar todas las líneas que hay dentro del archivo de configuración wpa_supplicant.conf.

Por último ya se pueden iniciar de nuevo los servicios detenidos al principio de la configuración ejecutando los siguientes comandos:

sudo service hostapd start

sudo service dnsmasq start

En este punto final de la configuración la Raspberry Pi actúa como un punto de acceso inalámbrico y otros dispositivos pueden asociarse con ella.

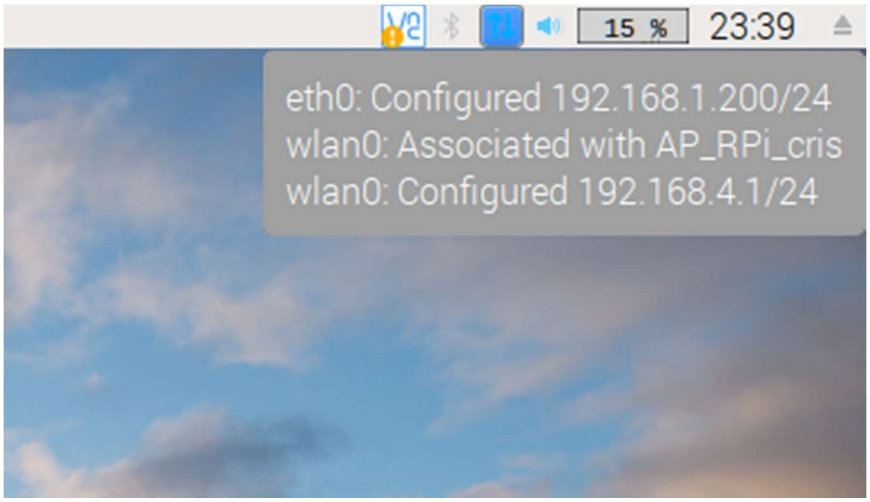


Figura A.6. 6 - Punto de acceso configurado con éxito

Anexo 7: Instalación Ngrok – Túnel

Para proceder a instalar el túnel Ngrok se accede a la página oficial de descargas de Ngrok a través del navegador de la Raspberry Pi para iniciar la descarga de la aplicación: https://ngrok.com/download

Ngrok permite realizar la descarga de la aplicación sin disponer de una cuenta de usuario, pero al instalar e iniciar el servicio el túnel creado caducará al cabo de 8 horas. Por lo tanto, para que el túnel no expire se ha preferido obtener una cuenta de usuario haciendo clic en “SIGN UP” y rellenar los datos solicitados por la página de Ngrok.

Una vez creada la cuenta de usuario e iniciado la sesión, la página detectará automáticamente el sistema operativo instalado en el dispositivo Raspberry Pi, en este caso Linux (ARM), y se procede a realizar la descarga.

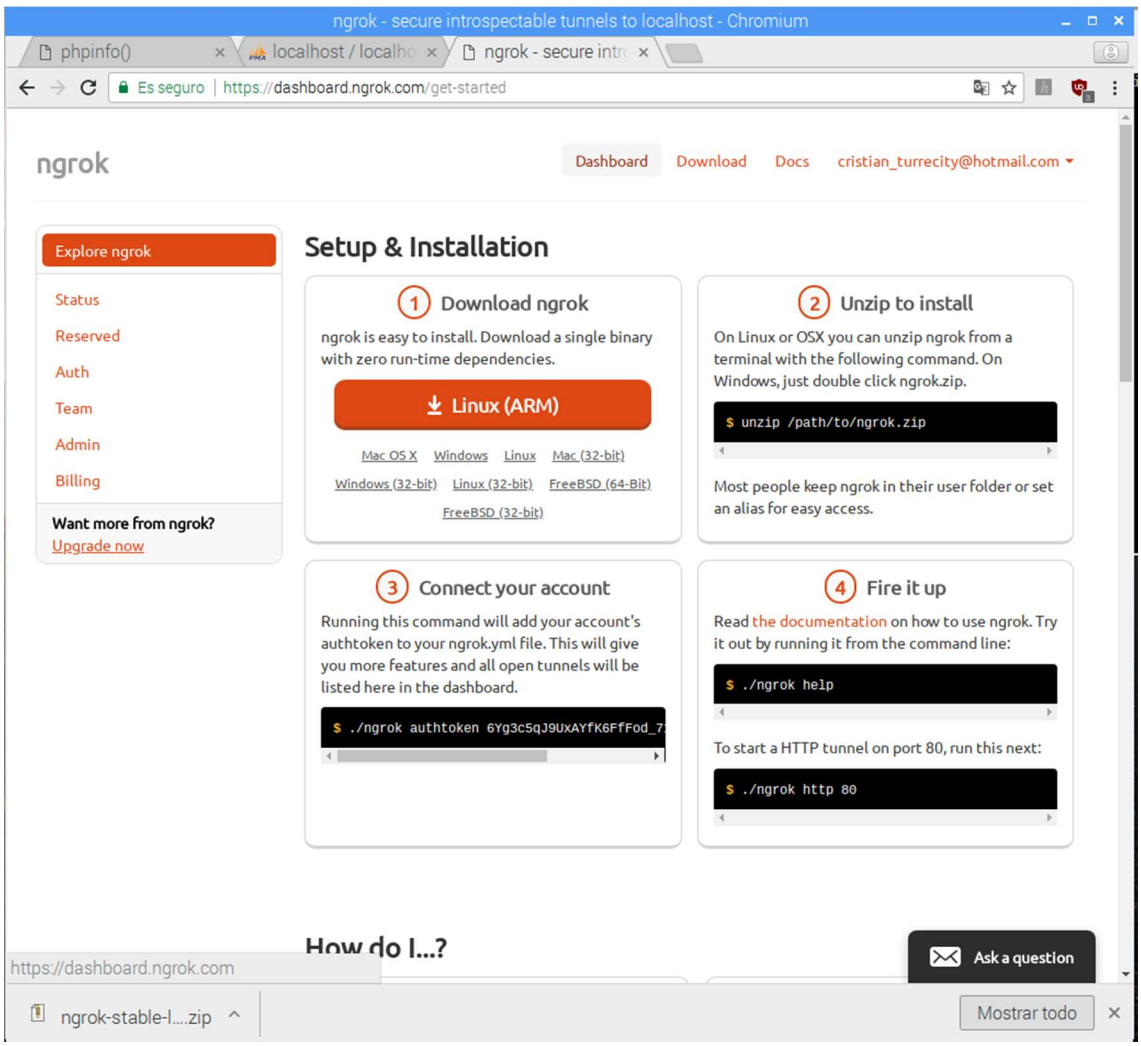


Figura A.7. 1 - Pagina web de descargas Ngrok

Descargado el archivo de instalación se descomprime utilizando el siguiente comando desde la terminal de la Raspberry Pi:

unzip /path/to/ngrok.zip

nzip /home/pi/Downloads/ngrok-stable-linux-arm.zip

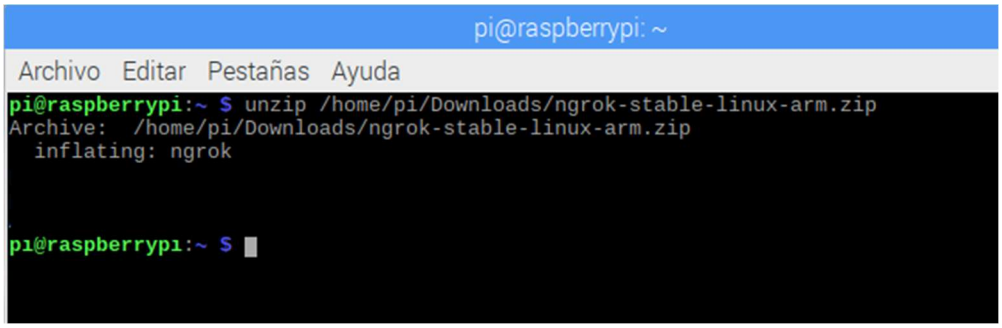


Figura A.7. 2 - Descompresión archivo instalación

Conectados a la cuenta de Ngrok se hace clic en “Dashboard” y después clic en “Auth” para ver el código “authtoken” del túnel. Este código se utilizará para que la aplicación instalada en la Raspberry Pi quede vinculada con la cuenta de usuario creada anteriormente.

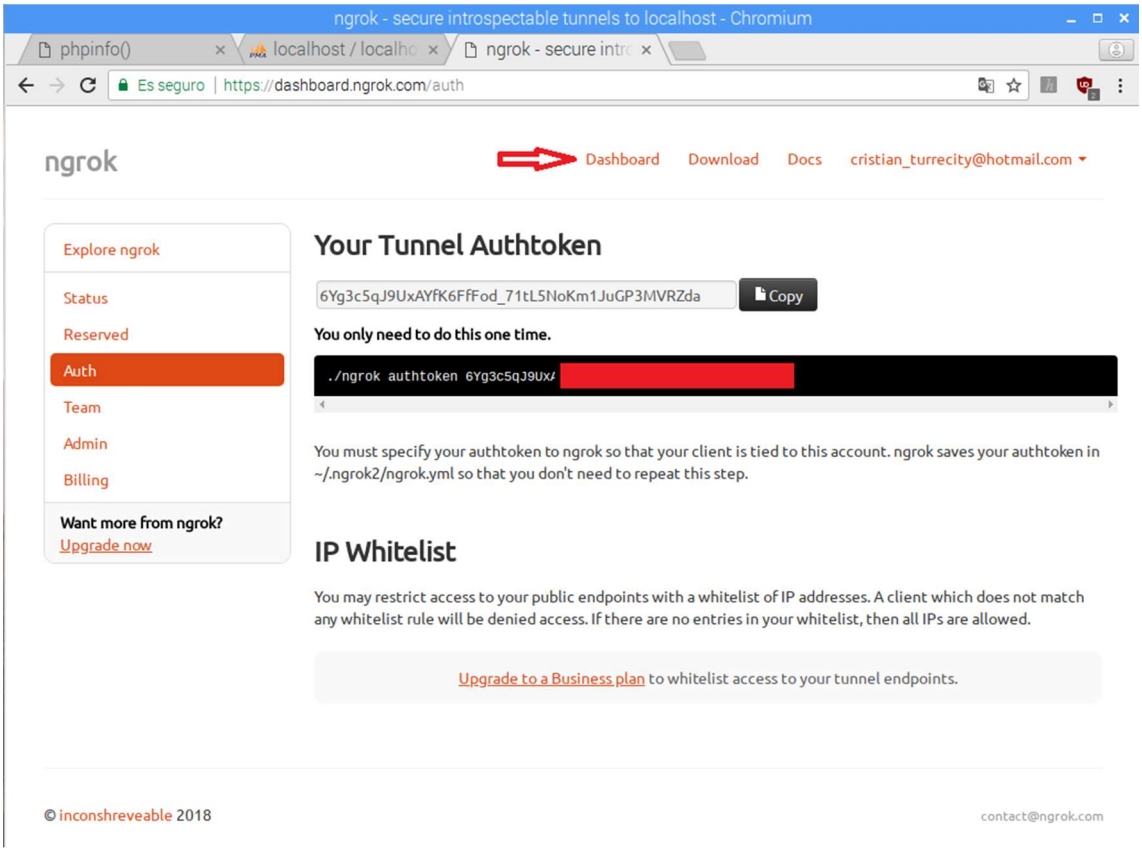


Figura A.7. 3 - Panel de usuario Ngrok

Se copia el código y se añade al archivo de configuración ngrok.yml, este proceso solo es necesario realizarlo una vez, no es necesario añadir el código cada vez que se inicia ngrok. Para ello se ejecuta desde la terminal el comando:

./ngrok authtoken <YOUR-AUTH-TOKEN”>

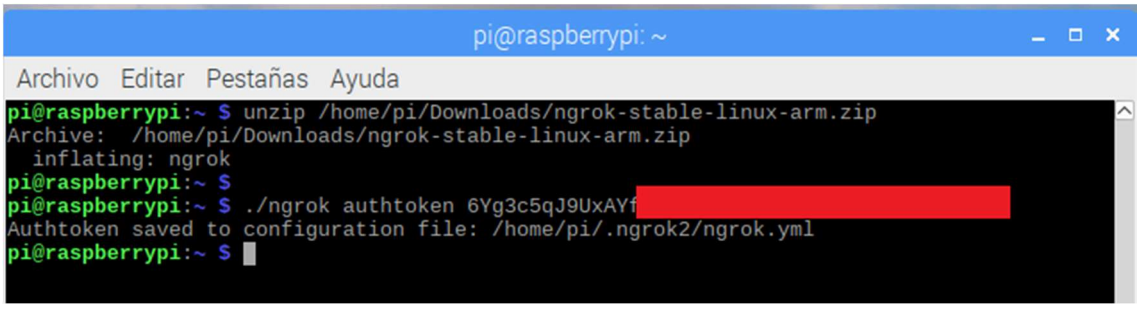


Figura A.7. 4 - Inserción código de autenticación

Por último ya es posible iniciar el servicio Ngrok y crear un túnel HTTP en el puerto 80. Cuando se inicia Ngrok, éste mostrará una interfaz de usuario con la URL pública del túnel y otra información de estado y métricas sobre las conexiones realizadas a través del túnel.

./ngrok http 80

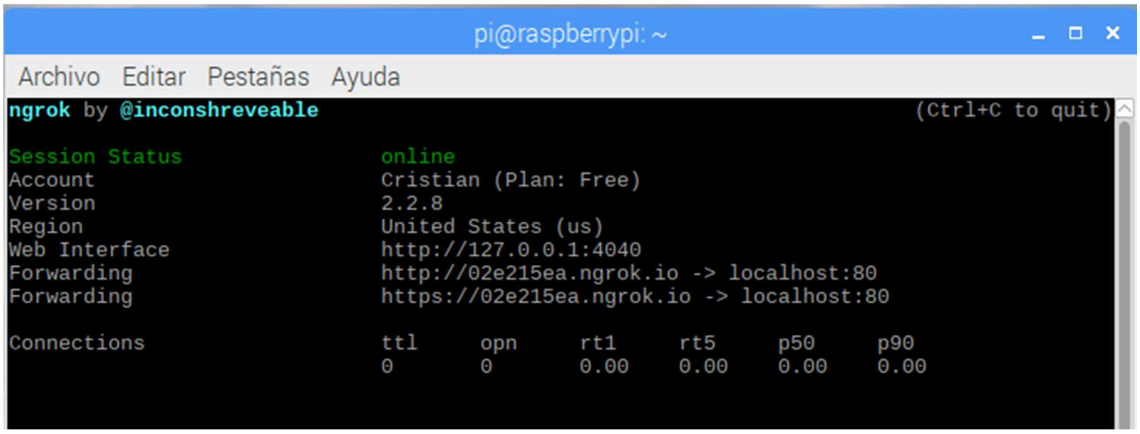


Figura A.7. 5 - Servicio Ngrok iniciado con cuenta de usuario

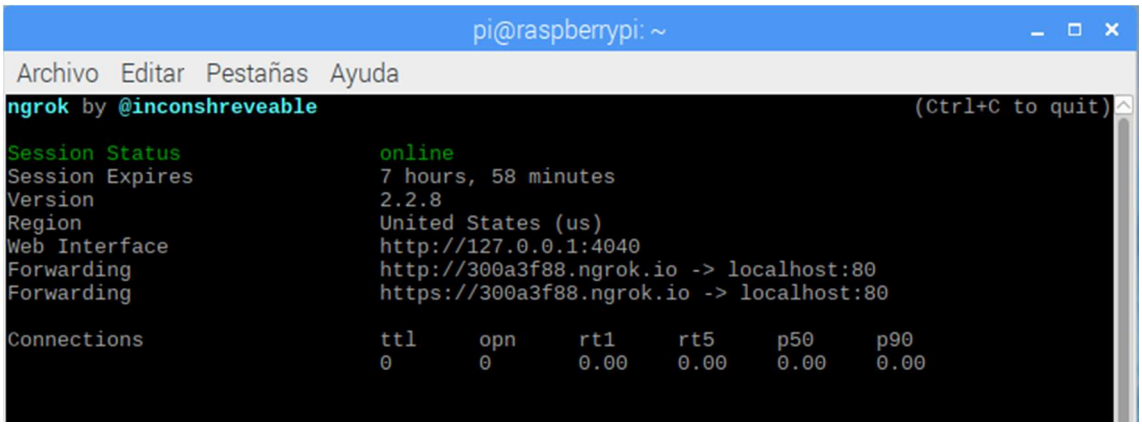


Figura A.7. 6 - Servicio Ngrok iniciado sin cuenta de usuario

Ngrok proporciona otra interfaz de usuario web en tiempo real en la que se puede inspeccionar todo el tráfico HTTP que se ejecuta a través de sus túneles. Después de iniciar Ngrok solo se tiene que acceder a la URL http://localhost:4040 o http://127.0.0.1:4040 mediante un navegador web para inspeccionar los detalles de la solicitud. En la interfaz web se mostrarán todos los detalles de la solicitud y de la respuesta así como, el tiempo, la duración, los encabezados, los parámetros de consulta, carga útil de la solicitud y los bytes brutos enviados por el cable.

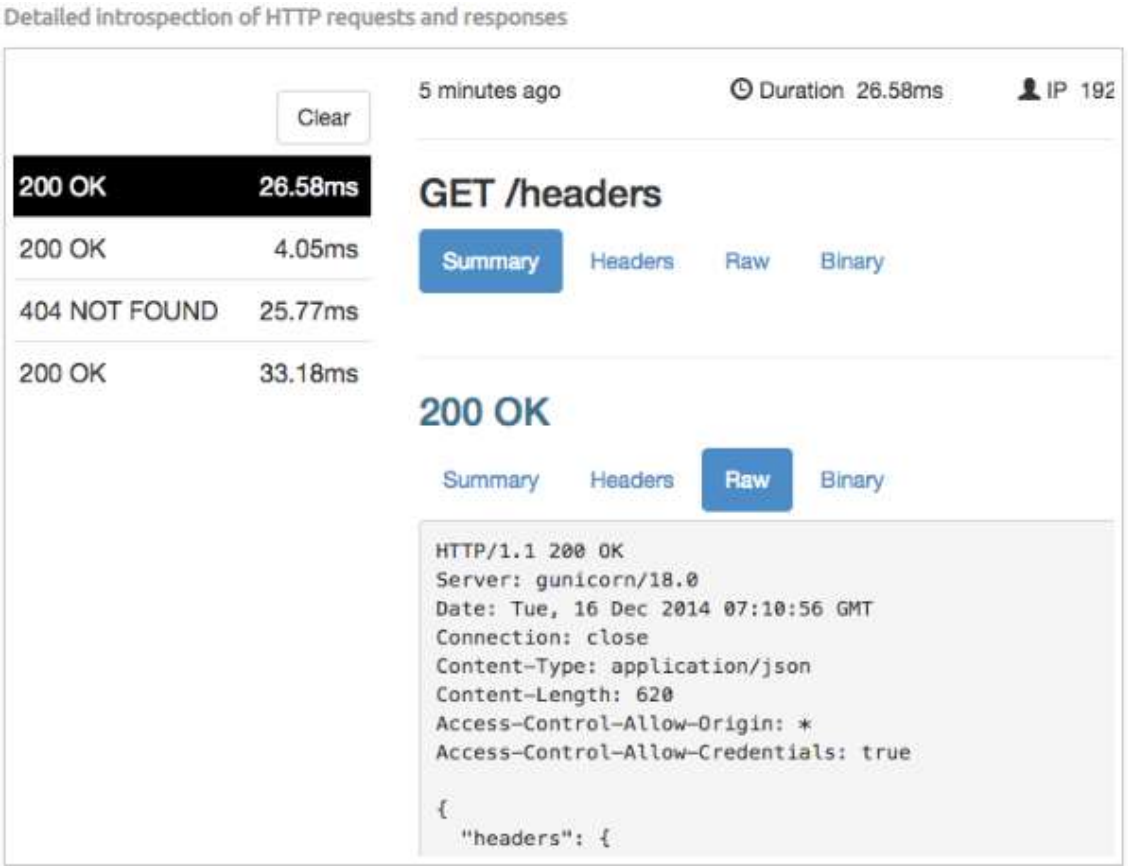


Figura A.7. 7 - Detalles de una inspección HTTP. Petición y respuesta

Anexo 8: Código fuente programa NodeMCU

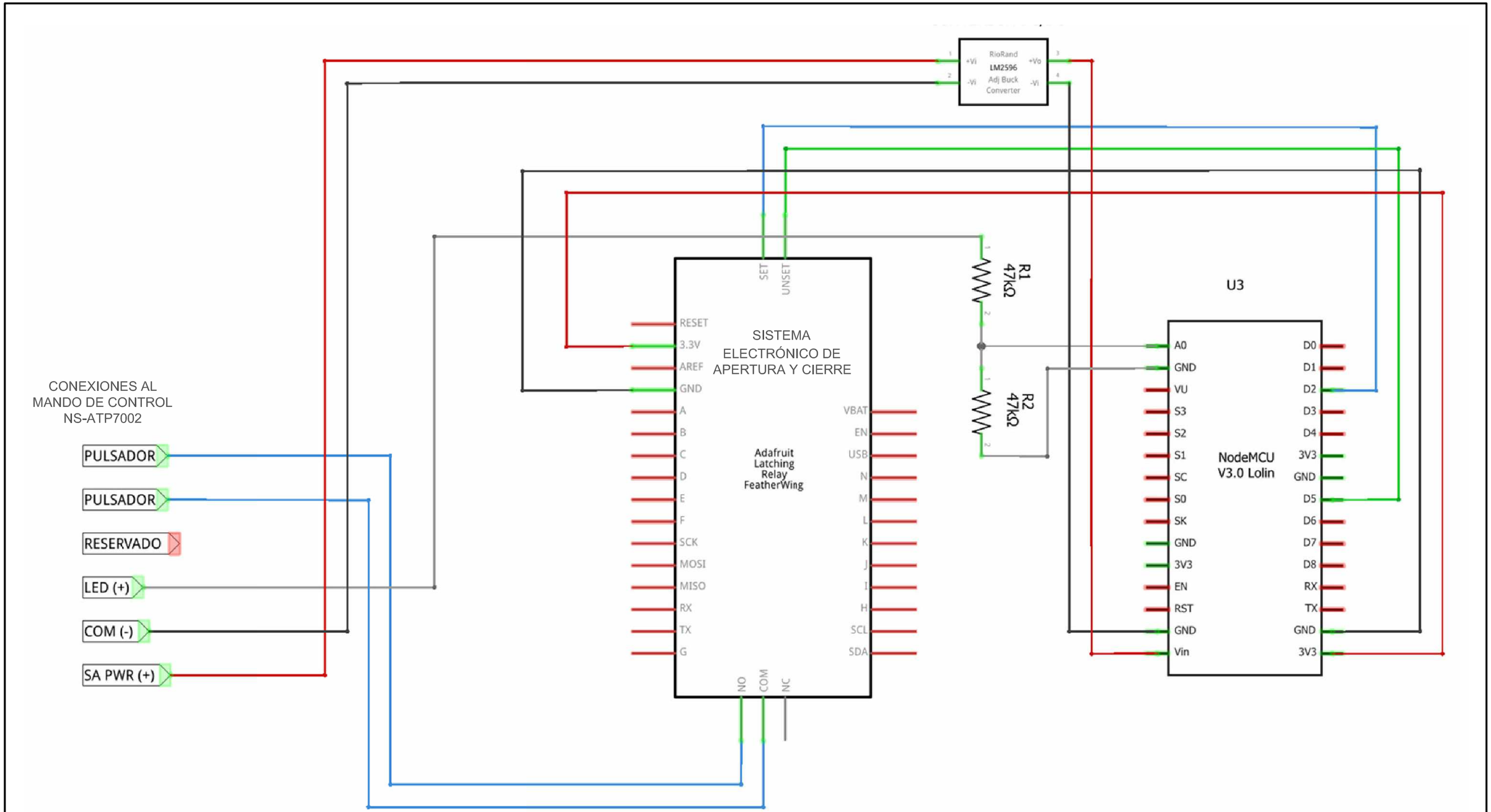
```
/*
  Este sketch demuestra las capacidades de la biblioteca PubSubClient en
combinación
  con la placa ESP8266 - NodeMCU

  Se conecta a un servidor (broker) MQTT y después:
  - se suscribe al tema (topic) "despacho/D115", despacho done estará
montado el prototipo
  Nota: se supone que las cargas útiles recibidas (mensajes) son cadenas
  de caracteres y no cadenas binarias.
  - Si el primer carácter recibido del topic "despacho/D115" es un 1,
activa una de las
  salidas GPIO del NodeMCU para que ésta active el relé, en caso
contrario, si el carácter
  recibido es un 0 se desactiva el relé.

 En el caso de que la conexión con el servidor MQTT se pierda, el cliente
volverá a realizar
 la conexión utilizando una función de reconexión con bloqueo.

*/
// Librerias WiFi para el módulo ESP8266 (NodeMCU)
#include <ESP8266WiFi.h>
// Librerias cliente MQTT
#include <PubSubClient.h>

// Parámetros del Punto de Acceso utilizado en el proyecto.
const char* ssid = "AP_RPi_cris"; // Nombre de la red WiFi a la que se
conecta.
const char* password = "Cristian2407"; // Contraseña de la red WiFi.

// Datos del Servidor MQTT (dirección IP o dominio)
const char* mqtt_server = "192.168.1.200";

// Inicialización del cliente WiFi y cliente MQTT:
WiFiClient espClient;
PubSubClient client(espClient);

// Definición de variables:
int SET = 4; // GPIO 04 - D2, se conecta al pin SET del relé.

int UNSET = 14; // GPIO 14 - D5, se conecta con el pin UNSET del relé.

// Función de obligatoria declaración en cualquier sketch.
void setup() {

    pinMode(SET,OUTPUT); // se define el pin como salida.
    digitalWrite(SET,LOW); // se inicializa el pin apagado (estado bajo).

    pinMode(UNSET,OUTPUT); // se define el pin como salida.
    digitalWrite(UNSET,LOW); // se inicializa el pin apagado (estado
bajo).

    // Iniciando comunicación Serie con un velocidad en baudios de 115200
    Serial.begin(115200); // Se abre el puerto serie y se fija la
velocidad
    setup_wifi();

    // Conexión con el servidor MQTT puerto 1883
```

```
    client.setServer(mqtt_server, 1883);
    // La función callback es la que recibe los mensajes y controla los
pines GPIO
    client.setCallback(callback);
}

// Inicialización de la conexión con el punto de acceso WiFi
void setup_wifi() {

    delay(10);
    // Comenzamos conectándonos a una red WiFi
    Serial.println();
    Serial.print("Connecting to ");
    Serial.println(ssid);

    WiFi.begin(ssid, password);

    while (WiFi.status() != WL_CONNECTED) {
      delay(500);
      Serial.print(".");
    }
    Serial.println("");
    Serial.println("WiFi connected");
    Serial.print("ESP IP address: ");
    Serial.println(WiFi.localIP());
    Serial.print("MQTT Server IP: ");
    Serial.println(mqtt_server);
}

// La función callback es la que recibe los mensajes y controla los pines
GPIO del módulo NodeMCU
// Esta función se ejecuta cuando algún dispositivo publica un mensaje
sobre un topic al que está
// suscrito el módulo ESP8266.
//
void callback(String topic, byte* message, unsigned int length) {
  Serial.print("Message arrived on topic: ");
  Serial.print(topic);
  Serial.print(".Mensaje: ");
  String messageTemp;

  // Se realiza un bucle para comprobar los caracteres del mensaje
recibido
  for (int i = 0; i < length; i++){
    // los caracteres recibidos se guardan en la variable messageTemp
    messageTemp += (char)message[i];
  }
  Serial.println(messageTemp);

  // Si se ha recibido un mensaje en el topic despacho/D115, se comprueba
si el mensaje
  // es un 1 o un 0. Se actua sobre el relé dependiendo del mensaje
recibido.

    if (topic == "despacho/D115"){
    Serial.print("Cambiando estado de la climatizacion a: ");

    // se comprueba si el mensaje recibido es un 1.
    if(messageTemp == "1"){
      // mediante la entrada analogica se monitoriza la tensión del led
incorporado en el
```

```
      // mando de climatización
      int sensorValue = analogRead(A0); // lectura del ADC
      float voltage = sensorValue * 3.3 / 1023.0; // escalamos a valores
de voltaje entre 0 y 3.3 V

      if(espClient.connect(mqtt_server, 80)>0){
        // se comprueba si la tensión es mayor que 2.2, esto significa
que la climatización está apagada
        // y se procede a su encendido.
        if (voltage > 2.2) {
          // si es asi, se activa el pin SET a traves de la salida GPIO
04 durante 10 milisegundos
          digitalWrite(SET,HIGH); // D2 se conecta con el SET de relé
          delay(10); // solo es necesario activar la salida GPIO durante
10 milisegundos para activar la bobina
          digitalWrite(SET,LOW); // despues de los 10 ms se desactiva la
salida GPIO 04
          // en este paso el relé ha cerrado sus contacto y se está
actuando sobre el pulsador del mando.
          // Se deja pulsado el pulsador durante 3 segundos para asegurar
que se enciende el sistema climático.
          delay(3000);
          // Despues de los 3 segundos se activa el pin GPIO 14 para
actuar sobre el pin UNSET del relé y éste
          // vuelva a abrir sus contacto y deje de pulsar el pulsador del
mando, ya que si no conmutan los
          // contactos del relé se quedaria siempre pulsado el pulsador.
          digitalWrite(UNSET,HIGH); // D5 se conecta con el pin UNSET del
relé
          delay(10); // se actua sobre el relé durante 10 milisegundos
          digitalWrite(UNSET,LOW); // se vuelve a desactivar el GPIO 14
para no actuar sobre el relé
          Serial.println("ON");
          // en este paso ya se ha activado el sistema de climatización y
se envia un 1 a la base de datos
          // para que en la aplicación web conozca en todo momento el
estado del sistema climático.
          int estadoCalefaccion = 1;
          espClient.print("GET /callenri/recibeEsp8266.php?valor=");
          espClient.print(estadoCalefaccion);
          espClient.println(" HTTP/1.0");
          espClient.println("User-Agent: Arduino 1.0");
          espClient.println();
          Serial.println("Conectado");
        }
      }else {
         Serial.println("Fallo en la conexion");
         }
    }
    // el proceso cuando se recibe un mensaje 0 es análogo a cuando se
recibe un mensaje 1
    else if(messageTemp == "0"){
      int sensorValue = analogRead(A0); // lectura del ADC
      float voltage = sensorValue * 3.3 / 1023.0; // escalamos a valores
de voltaje entre 0 y 3.3 V
      if(espClient.connect(mqtt_server, 80)>0){
        if(voltage < 1.95){
          digitalWrite(SET,HIGH); // D2 - GPIO 04
          delay(10);
          digitalWrite(SET,LOW);
          delay(3000);
```

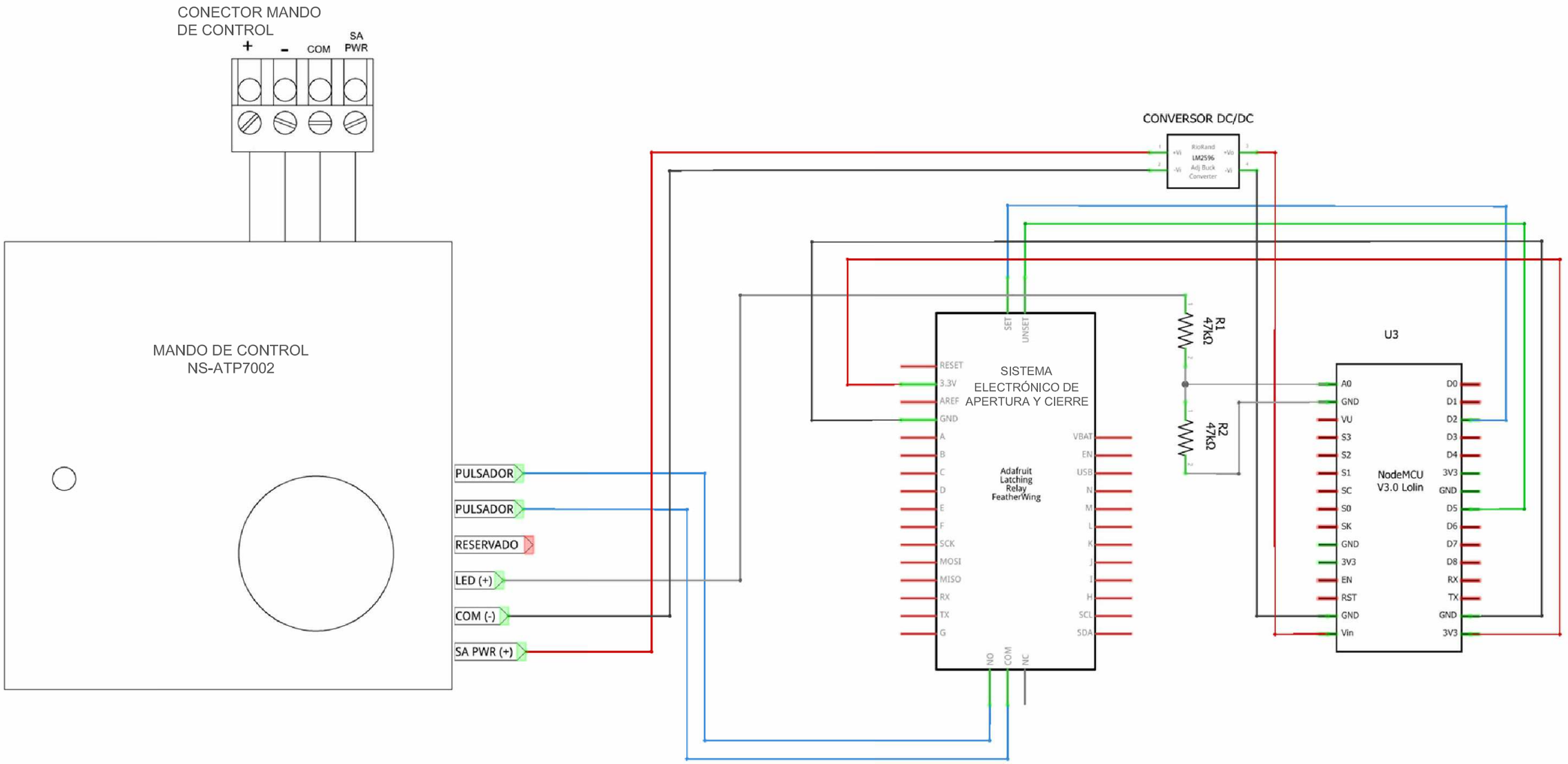
```
          digitalWrite(UNSET,HIGH); // D5 - GPIO 14
          delay(10);
          digitalWrite(UNSET,LOW);
          Serial.println("OFF");
          int estadoCalefaccion = 0;
          // en este caso se manda un 0 a la base de datos para saber que
el sistema está apagado.
          espClient.print("GET /callenri/recibeEsp8266.php?valor=");
          espClient.print(estadoCalefaccion);
          espClient.println(" HTTP/1.0");
          espClient.println("User-Agent: Arduino 1.0");
          espClient.println();
          Serial.println("Conectado");
         }
      }else {
         Serial.println("Fallo en la conexion");
        }
    }
    espClient.stop();
    espClient.flush();
  }

}

// Esta función reconecta el cliente ESP con el servidor MQTT
void reconnect() {
  // Se realiza un bucle hasta que se vuelva a conectar el cliente con el
servidor.
  while (!client.connected()) {
    Serial.print("Attempting MQTT connection...");
    // intentando conectar
    if (client.connect("ESP8266Client")) {
      Serial.println("connected");
      // Una vez conectado vuelve a suscribirse al topic despacho/D115
      client.subscribe("despacho/D115");
    } else {
      Serial.print("failed, rc=");
      Serial.print(client.state());
      Serial.println(" try again in 5 seconds");
      // Wait 5 seconds before retrying
      delay(5000);
    }
  }
}

// Para este proyecto no se debe cambiar nada en la función loop.
// Básicamente asegura que el módulo ESP8266 esté conectado con el
servidor MQTT
void loop() {

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

}
```

PLANOS



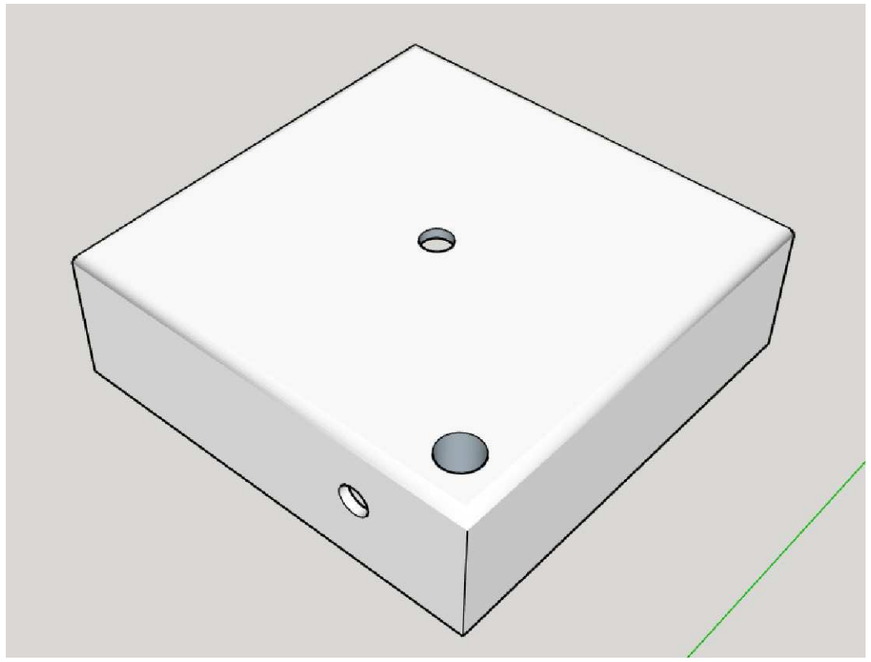
	FECHA	NOMBRE	FIRMA	ESCUELA POLITÉCNICA SUPERIOR LINARES
DIBUJADO	13/08/18	Cristian Maier		
COMPROBADO				

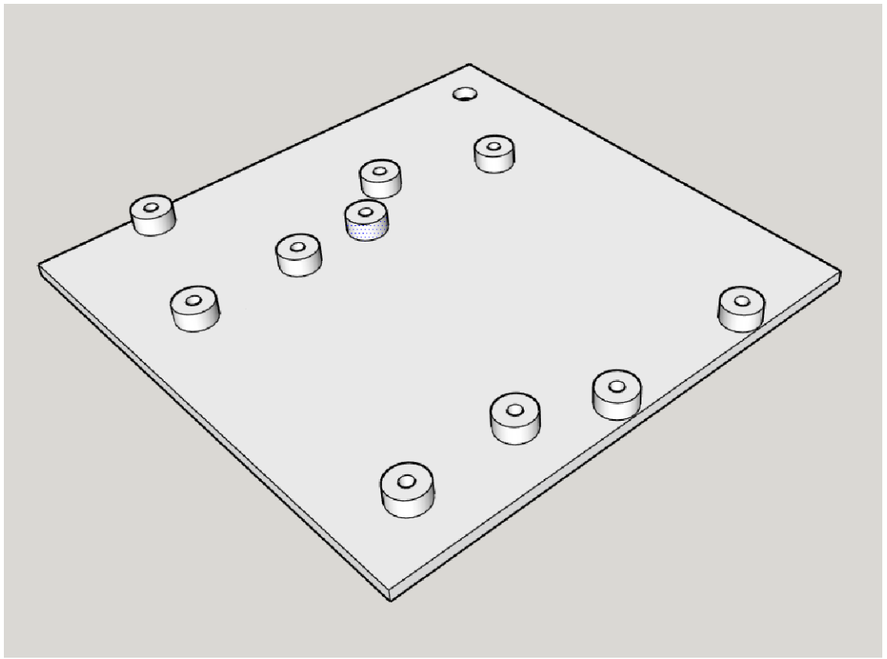
ESCALA:	Esquema de conexiones del prototipo	Nº PLANO 1
		SUSTITUYE A:
		SUSTITUIDO POR:

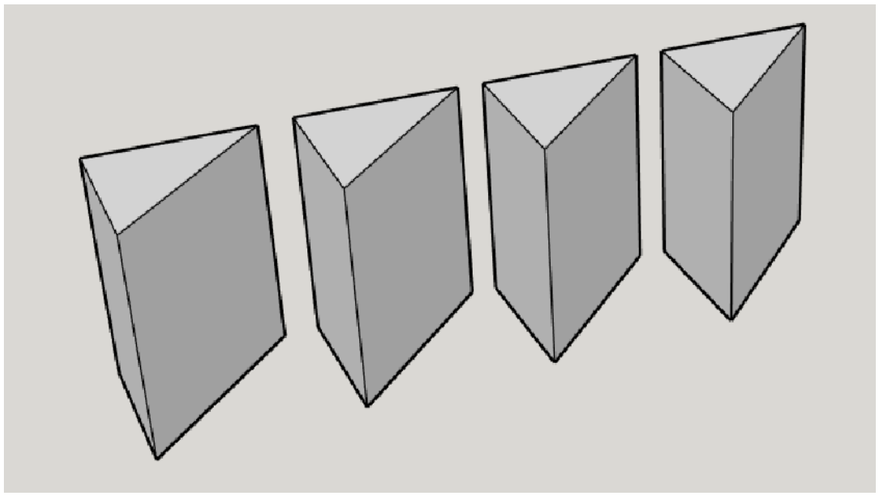


	FECHA	NOMBRE	FIRMA	ESCUELA POLITÉCNICA SUPERIOR LINARES
DIBUJADO	13/08/18	Cristian Maier		
COMPROBADO				

ESCALA:	Esquema de conexiones generales	Nº PLANO 2
		SUSTITUYE A:
		SUSTITUIDO POR:







	FECHA	NOMBRE	FIRMA	ESCUELA POLITÉCNICA SUPERIOR LINARES
DIBUJADO	13/08/18	Cristian Maier		
COMPROBADO				

ESCALA:	Caja de prototipo	Nº PLANO 3
		SUSTITUYE A:
		SUSTITUIDO POR:

ESTADO DE MEDICIONES

1. Adaptación mando control

Nº Orden	Concepto	Uds.
01	Conector multipolar hembra recto, 6 pines 1 fila	1
02	Kit de cable conector para placas de prueba, 10 piezas	1
P.EP2.3	Hora de soldadura (Oficial 1ª)	0.5
P.EP2.1	Hora de comprobación (Técnico)	0.25

2. Prototipo

Nº Orden	Concepto	Uds.
03	Kit de desarrollo NodeMCU Lolin V3	1
04	Conversor de tensión DC/DC LM2596	1
05	Relé en miniatura. Latching Relay Featherwing de Adafruit	1
06	Conector multipolar macho recto, 6 pines 1 fila. Orificio pasante, terminación soldada	1
02	Kit de cable conector para placas de prueba, 10 piezas	1
07	Tornillo mecánico, acabado galvanizado brillante, acero, avellanado	10
08	Resistencia fija de película de carbono, 47 kΩ, ±5%, 0.25 W	2
09	Caja de plástico. Impresa con impresora 3D	1
P.EP2.3	Hora de soldadura (Oficial 1ª)	0.5
P.EP2.1	Hora de comprobación (Técnico)	0.25

3. Otros materiales

Nº Orden	Concepto	Uds.
10	Kit de desarrollo Premium Raspberry Pi 3 (Raspberry Pi Model 3 B, carcasa Premium, fuente de alimentación micro USB, cable HDMI, tarjeta microSD de 8GB)	1
11	Latiguillo de Ethernet Cat6, 1m, Recto, Amarillo, Funda PVC	1

PRESUPUESTO

1. Precios simples

1.1. Partida de materiales

Nº Orden	Concepto	Código RS	Precio unitario (€)
01	Conector multipolar hembra recto, 6 pines 1 fila	602-7951	1,573
02	Kit de cable conector para placas de prueba, 10 piezas	791-6450	2,41
03	Kit de desarrollo NodeMCU Lolin V3		2,83
04	Conversor de tensión DC/DC LM2596		1,82
05	Relé en miniatura. Latching Relay Featherwing de Adafruit	124-5527	6,82
06	Conector multipolar macho recto, 6 pines 1 fila. Orificio pasante, terminación soldada	702-0121	0,836
07	Tornillo mecánico, acabado galvanizado brillante, acero, avellanado	553-396	0,021
08	Resistencia fija de película de carbono, 47 kΩ, ±5%, 0.25 W	135-976	0,029
09	Caja de plástico. Impresa con impresora 3D		60,50
10	Kit de desarrollo Premium Raspberry Pi 3 (Raspberry Pi Model 3 B, carcasa Premium, fuente de alimentación micro USB, cable HDMI, tarjeta microSD de 8GB)	123-7157	61,41
11	Latiguillo de Ethernet Cat6, 1m, Recto, Amarillo, Funda PVC	411-520	1,90

1.2. Partida de equipos y maquinaria

Cantidad (h)	Concepto	Precio unitario (€)
1	Soldador	0,35

1.3. Partida de mano de obra

Cantidad (h)	Concepto	Precio unitario (€)
1	Técnico	19,43
1	Oficial de primera	14,81

2. Precios auxiliares

2.1. Salario técnico

Concepto		Importe (€)
Salario base		12.111,06
Plus de convenio		781,32
Pagas extraordinarias		2.148,73
Paga de participación en beneficios		537,18
Base cotización a la segundad social		15.578,29
Cotización a la segundad social (35% sobre la base)		5.452,40
Dietas		1.162,96
	Total	37.771,94
	Importe por hora	19,43

2.2. Salario oficial de primera

Concepto		Importe (€)
Salario base		9.409,08
Plus de convenio		781,32
Pagas extraordinarias		1.698,40
Paga de participación en beneficios		424,60
Base cotización a la segundad social		11.338,72
Cotización a la segundad social (35% sobre la base)		3.968,55
Dietas		1.162,96
	Total	28.783,63
	Importe por hora	14,81

2.3. Una hora de soldadura

Cantidad	Concepto	Precio unitario (€)	Subtotal (€)
1 h	Oficial de primera	14,81	14,81
1 h	Soldador	0,35	0,35
0.05 kg	Estaño	50,20	2,51
		Total	17.67

3. Precios descompuestos

3.1. Adaptación mando control

Nº Orden	Concepto	Uds.	Precio unitario (€)	Subtotal (€)
01	Conector multipolar hembra recto, 6 pines 1 fila	1	1,573	1,573
02	Kit de cable conector para placas de prueba, 10 piezas	1	2,41	2,41
P.EP2.3	Hora de soldadura (Oficial 1ª)	0.5	17.67	8,835
P.EP2.1	Hora de comprobación (Técnico)	0.25	19,43	4,857
			Costes directos	17,67
			Costes indirectos: 13% sobre costes directos	2.30
			Total	19,97

3.2. Prototipo

Nº Orden	Concepto	Uds.	Precio unitario (€)	Subtotal (€)
03	Kit de desarrollo NodeMCU Lolin V3	1	2,83	2,83
04	Conversor de tensión DC/DC LM2596	1	1,82	1,82
05	Relé en miniatura. Latching Relay Featherwing de Adafruit	1	6,82	6,82
06	Conector multipolar macho recto, 6 pines 1 fila. Orificio pasante, terminación soldada	1	0,836	0,836
02	Kit de cable conector para placas de prueba, 10 piezas	1	2,41	2,41
07	Tornillo mecánico, acabado galvanizado brillante, acero, avellanado	10	0,021	0,21
08	Resistencia fija de película de carbono, 47 kΩ, ±5%, 0.25 W	2	0,029	0,058
09	Caja de plástico. Impresa con impresora 3D	1	60,50	60,50
P.EP2.3	Hora de soldadura (Oficial 1ª)	0.5	17.67	8,835

P.EP2.1	Hora de comprobación (Técnico)	0.25	19,43	4,857
			Costes directos	86,77
			Costes indirectos: 13% sobre costes directos	11.28
			Total	98.05

3.3. Otros materiales

Nº Orden	Concepto	Uds.	Precio unitario (€)	Subtotal (€)
10	Kit de desarrollo Premium Raspberry Pi 3 (Raspberry Pi Model 3 B, carcasa Premium, fuente de alimentación micro USB, cable HDMI, tarjeta microSD de 8GB)	1	61,41	61,41
11	Latiguillo de Ethernet Cat6, 1m, Recto, Amarillo, Funda PVC	1	1,90	1,90
			Costes directos	63,31
			Costes indirectos: 13% sobre costes directos	8,23
			Total	71,54

3.4. Diseño y desarrollo aplicación web

Nº Orden	Concepto	Uds. (h)	Precio unitario (€)	Subtotal (€)
	Diseño aplicación web. (Técnico)	45	19,43	874,35
	Desarrollo aplicación web (Técnico)	75	19,43	1.457,25
			Costes directos	2.331,60
			Costes indirectos: 13% sobre costes directos	303,11
			Total	2.634,71

3.5. Desarrollo programa (sketch) NodeMCU

Nº Orden	Concepto	Uds. (h)	Precio unitario (€)	Subtotal (€)
	Desarrollo programa NodeMCU (Técnico)	10	19,43	194,30
			Costes directos	194,30
			Costes indirectos: 13% sobre costes directos	25,26
			Total	219,56

3.6. Diseño caja prototipo

Nº Orden	Concepto	Uds. (h)	Precio unitario (€)	Subtotal (€)
	Diseño caja prototipo (Técnico)	1,5	19,43	29,15
			Costes directos	29,15
			Costes indirectos: 13% sobre costes directos	3,79
			Total	32,94

4. Presupuestos

4.1. Presupuesto de ejecución material

Uds.	Concepto	Precio unitario (€)	Subtotal (€)
1	Adaptación mando control	19,97	19,97
1	Prototipo	98.05	98.05
		Costes directos	118,02
		Costes indirectos: 5% sobre costes directos	15,34
		Total presupuesto ejecución material	133,36

El presupuesto de ejecución material asciende a la cantidad de CIENTO TREINTA Y TRES EUROS con TREINTA Y SEIS CENTÍMOS

4.2. Presupuesto de diseño y desarrollo

Uds.	Concepto	Precio unitario (€)	Subtotal (€)
1	Diseño y desarrollo aplicación web	3732,50	2.634,71
1	Desarrollo programa NodeMCU	219,56	219,56
1	Diseño caja prototipo	32,94	32,94
		Costes directos	2.887,21
		Costes indirectos: 5% sobre costes directos	144.36
		Total presupuesto de diseño y desarrollo	3.031,57

El presupuesto de diseño y desarrollo asciende a la cantidad de CUATRO MIL CIENTO OCHENTA Y CUATRO con VEINTICINCO CENTÍMOS.

4.3. Presupuesto total

Concepto	Subtotal (€)
Ejecución material del sistema	133,36
Diseño y desarrollo	3.031,57
Presupuesto total sin IVA	3.164,93
IVA (21%)	664,63
Presupuesto total IVA inc.	**3.829,56**

El presupuesto total del proyecto asciende a la cantidad de **TRES MIL OCHOCIENTOS VEINTINUEVE con CINCUENTA Y SEIS CENTÍMOS**.

ESTUDIO BÁSICO DE SEGURIDAD Y SALUD

1. Objeto del presente estudio básico de seguridad y salud

El presente Estudio Básico de Seguridad y Salud (E.B.S.S) tiene como objeto servir de base para que las personas que participen en el desarrollo y ejecución de todas las operaciones a las que hace referencia el proyecto en que se encuentra incluido este estudio, las lleven a efecto en las mejores condiciones que puedan alcanzarse a garantizar el mantenimiento de la salud y la integridad física.

2. Relación puntual de los trabajos a realizar

- Estudio y desarrollo de las partes software y hardware.
- Realización y montaje de las placas del prototipo.

3. Identificación de riesgos en cada fase de ejecución

Durante la ejecución de los trabajos se plantea la realización de las siguientes fases con la identificación de los riesgos que conllevan:

3.1. Estudio y desarrollo de las partes software y hardware

Durante esta fase de ejecución la persona encargada de realizarla puede estar sometida a radiaciones peligrosas procedentes de monitores de PC, mala calidad en la iluminación o ventilación y descargas eléctricas debido a un mal estado de la red o deficiencias en los equipos eléctricos utilizados.

3.2. Realización y montaje de las placas del prototipo

Durante la ejecución de esta fase, los riesgos a tener presentes se engloban en la posibilidad de entrar en contacto con atmósferas tóxicas e inhalación de humo procedentes del soldador, que pueden provocar irritaciones y afecciones respiratorias. Además hay que tener presentes la posibilidad de sufrir traumatismos, lesiones abiertas, cortes y quemaduras debido al uso de destornilladores, alicates, tijeras, cúter y soldadores de estaño.

4. Relación de medios técnicos previstos con identificación de riesgos

Se describen, a continuación, los medios técnicos que se prevé utilizar para el desarrollo de este proyecto identificando sus posibles riesgos.

- **Equipo informático**: radiaciones del monitor, choque eléctrico o lesiones no traumáticas en miembros superiores.
- **Equipo de soldadura**: quemaduras, inhalación de sustancias tóxicas y riesgo de incendio.
- **Destornilladores, alicates, tijeras y cúter**: golpes, cortes y caídas al suelo.

5. Tipos de energía a emplear

A continuación se relacionan los diferentes tipos de energía a utilizar con identificación de los riesgos que su utilización deriva.

- **Electricidad**: quemaduras físicas y químicas, contactos eléctricos directos e indirectos, exposición a fuentes luminosas peligrosas o incendios.
- **Radiación ultravioleta**: exposición a fuentes luminosas peligrosas.
- **Reacción química**: quemaduras, atmosferas irritantes y tóxicas e inhalación de sustancias tóxicas.
- **Energía mecánica**: enganches, cortes y caídas.

6. Materiales peligrosos

En este apartado se identifican los materiales a emplear considerados peligrosos, relacionando los riesgos que supone su utilización.

Estaño: atmosferas tóxicas, irritantes y riesgo de inhalación de humo tóxico.

7. Medidas de prevención de riesgos

7.1. Medidas de protección generales

A nivel general, para evitar los posibles riesgos descritos con anterioridad y de forma colectiva, se procederá a realizar los trabajos que sean necesarios en el proceso de desarrollo y ejecución del proyecto en lugares que cumplan con las siguientes características:

- Iluminación (natural o artificial) abundante y en el grado adecuado.
- Correcta ventilación.
- Fuentes de suministro de energía en perfecto uso.
- La maquinaria deberá de tener en perfecto estado sus protecciones.

7.2. Equipos de protección individual (EPI)

- Atmósferas tóxicas e irritantes: gafas de seguridad para uso básico y mascarilla respiratoria de filtro para humos de soldadura.
- Quemaduras físicas: guantes de protección frente a abrasión.
- Exposiciones a fuentes luminosas peligrosas: gafas de seguridad contra radiaciones y protectores de pantalla para luz ultravioleta.

BIBLIOGRAFÍA

[1] https://nodemcu.readthedocs.io/en/dev/

[2] http://mqtt.org/

[3] http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/os/mqtt-v3.1.1-os.html#_Toc398718009

[4] https://www.iso.org/standard/69466.html

[5] https://www.ibm.com/uk-en/information-technology/andy-stanford-clark

[6] https://www.linkedin.com/in/arlen-nipper-42281057

[7] http://mqtt.org/tag/arlen-nipper

[8] https://www.youtube.com/watch?v=s9nrm8q5eGg

[9] https://www.iso.org/organization/306174.html

[10] https://www.oasis-open.org/

[11] https://www.iso.org/home.html

[12] https://www.iso.org/standard/69466.html

[13] http://public.dhe.ibm.com/software/dw/webservices/ws-mqtt/mqtt-v3r1.html

[14] https://www.hivemq.com/blog/how-to-get-started-with-mqtt

[15]https://www.gsma.com/mobileeconomy/wp-content/uploads/2018/05/The-Mobile-Economy-2018.pdf

[16] https://www.gsma.com/mobileeconomy/

[17] https://www.ikerlan.es/

[18] https://www.ikerlan.es/lineas-de-especializacion/area/sistemas-embebidos-confiables

[19] https://www.certsi.es/blog/introduccion-los-sistemas-embebidos

[20] https://www.tutorialspoint.com/embedded_systems/es_overview.htm

[21] http://linuxemb.wikidot.com/tesis-c2

[22] https://programarfacil.com/podcast/nodemcu-tutorial-paso-a-paso/

[23] https://www.arduino.cc/

[24] https://www.adafruit.com/

[25] https://www.raspberrypi.org/

[26] http://cgproducts.johnsoncontrols.com/MET_PDF/241009417.PDF

[27] http://cgproducts.johnsoncontrols.com/MET_PDF/12011034.pdf

[28] https://www.arduino.cc/en/main/software#

[29] https://processing.org/

[30] https://mosquitto.org/

[31] http://www.eclipse.org/

[32] https://iot.eclipse.org/

[33] https://github.com/mqtt/mqtt.github.io/wiki/server-support

[34] https://github.com/mqtt/mqtt.github.io/wiki/public_brokers
[35] https://www.raspberrypi.org/
[36] https://www.raspberrypi.org/downloads/raspbian/
[37] https://www.apache.org/
[38] https://www.raspberrypi.org/documentation/usage/
[39] https://www.raspberrypi.org/documentation/configuration/wireless/access-point.md
[40] https://ngrok.com/
[41] http://www.esploradores.com/comparacion-de-placas-nodemcu_/
[42] https://ngrok.com/docs
[43] https://es.rs-online.com/web/p/kits-de-desarrollo-de-control-de-potencia/1245527/
[44] http://cgproducts.johnsoncontrols.com/MET_PDF/12011968.PDF
[45]https://www.johnsoncontrols.com/-/media/jci/be/united-states/hvac-controls/thermostats/specs/1900891.pdf
[46] https://www.johnsoncontrols.com/buildings/hvac-controls/control-sensors/networked-sensors
[47] http://www.handsontec.com/pdf_learn/esp8266-V10.pdf