TRABAJO FIN DE GRADO

INGENIERÍA DE TECNOLOGÍAS DE TELECOMUNICACIÓN

Diseño e implementación de un sensor IoT low-cost para comunicaciones infrarrojas en un entorno domótico

Autor
Ramón Fernández Gualda

Directores
Jorge Navarro Ortiz
Sandra Sendra Compte

Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación

—

Granada, junio de 2017

TRABAJO FIN DE GRADO

INGENIERÍA DE TECNOLOGÍAS DE TELECOMUNICACIÓN

Diseño e implementación de un sensor IoT low-cost para comunicaciones infrarrojas en un entorno domótico

Autor
Ramón Fernández Gualda

Directores
Jorge Navarro Ortiz
Sandra Sendra Compte

Departamento de Teoría de la Señal, Telemática y Comunicaciones

—

Granada, junio de 2017

Diseño e implementación de un sensor IoT low-cost para comunicaciones infrarrojas en un entorno domótico

Ramón Fernández Gualda

Palabras clave: ESP8266, Internet of Things (IoT), NodeMCUv1.0, Wemos D1 Mini, Arduino, Protocolos de señales infrarrojas (IR), Message Queue Telemetry Transport (MQTT)

Resumen

Continuamente estamos invadidos por una gran variedad de aparatos electrónicos en el ámbito doméstico como televisores, aires acondicionados, sistemas de calefacción, cadenas de música o reproductores DVD. Todas ellas acompañadas por supuesto de su sistema de comunicaciones infrarrojas, es por ello que se aprecia un aumento de controles remoto en el hogar que permiten dicha comunicación.

Por otro lado, es destacable la normalización de la disponibilidad y uso de *smartphones* ofreciendo actualmente modelos con una gran potencia de cálculo que sigue creciendo exponencialmente con el tiempo. Tal es el caso que en las soluciones domóticas se busca la integración de dichos aparatos de manera que permitan el control y gestión de los dispares equipos que forman parte de la "vivienda inteligente". La ejecución de los comandos que administran el funcionamiento de los dispositivos del hogar se realizarán por medio de aplicaciones previamente instaladas.

Es por ello que este proyecto se focaliza en el desarrollo de una solución *low-cost* capacitada para la gestión de aquellos sistemas que permiten una comunicación por infrarrojos, como por mandos a distancia, dirigida desde un terminal móvil por aplicación que permita el envío de las instrucciones a llevar a cabo inclusive fuera del lugar de ejecución.

Se adentra pues, a un exhaustivo estudio de los diferentes protocolos de infrarrojos empleados en el mercado que facilitará el entendimiento de los mismos, descubrimiento de las bases de datos existentes que almacenan todo tipo de información relativa al funcionamiento de cada uno de los protocolos dependiendo de la estructura y una indagación sobre el chip ESP8266, del cual se aprovechará la versión ESP-12-E con varias versiones de módulos con capacidades suficientes para cumplir las necesidades IoT que se abordarán a un coste muy competente.

Por último se desea mostrar un diseño e implementación del circuito compuesto tanto de transmisor y receptor de señales IR que posibilitará las acciones anteriores de manera que se interactúe entre el *smartphone* y los

dispositivos domésticos IR. Será imprescindible el papel del servidor privado con protocolo MQTT, protocolo *lightweight* que facilitará la comunicación con paradigma Publicador - Suscriptor con el cual se trabajará dando a conocer su utilidad.

Design and implementation of a low-cost IoT sensor for infrared communications in a domotic environment

Ramón Fernández Gualda

Keywords: ESP8266, Internet of Things (IoT), NodeMCUv1.0, Wemos D1 Mini Arduino, Infrared Signal Protocols (IR), Message Queue Telemetry Transport (MQTT)

Abstract

We are continually invaded by a large variety of electronic devices in the domestic environment such as televisions, air conditioners, heating systems, stereo systems or DVD players. All of them are accompanied, of course, by an infrared communications system. This is why there is an increase in remote controls in the homethat allow such communication.

On the other hand, it is remarkable the standardization of the availability and use of smartphones offering current models with a great computing power that continues growing exponentially over time. Such is the case that in the domotic solutions they search for the integration of these devices in a way that allows the control and management of the disparate equipments that form part of the "Smart house". The execution of the commands that manage the operation of the home devices will be carried out through previously installed applications.

This is why this project focuses on the development of a low-cost solution capable of managing those systems that allow infrared communication, such as remote control, directed from a mobile terminal per application that allows the sending of the instructions to be carried out even outside the place of execution.

Therefore, an exhaustive study of the different infrared protocols used on the market that will facilitate the understanding of the same, discovery of the existing databases that store all types of information relative to the operation of each one of the protocols depending on the brand and an inquiry on the ESP8266 chip, which will take advantage of the ESP-12-E versión with some versions of modules with enough capacities to obey the IoT requirements that will be exposed at a very competent cost.

Finally, it is desirable to show a design and implementation of the composite circuit of both transmitter and receiver of IR signals that will enable the previous actions so that it interacts between the smartphone and the IR home devices. The role of the private server with protocol MQTT will be essential, lightweight protocol that will facilitate the communication with

paradigm Publisher - Subscriber with which it will work to make its utility known.

Yo, **Ramón Fernández Gualda**, alumno de la titulación de Grado en Ingeniería de Tecnologías de Telecomunicación de la **Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación de la Universidad de Granada**, con DNI XXX, autorizo la ubicación de la siguiente copia de mi Trabajo Fin de Grado en la biblioteca del centro para que pueda ser consultada por las personas que lo deseen.

Fdo: Ramón Fernández Gualda

Granada, 21 de junio de 2017 .

D. **Jorge Navarro Ortiz**, Profesor del Área de Ingeniería Telemática del Departamento de teoría de la Señal, Telemática y Comunicaciones de la Universidad de Granada.

Dª. **Sandra Sendra Compte**, Profesora del Área de Ingeniería Telemática del Departamento de teoría de la Señal, Telemática y Comunicaciones de la Universidad de Granada.

Informan:

Que el presente trabajo, titulado ***Diseño e implementación de un sensor IoT low-cost para comunicaciones infrarrojas en un entorno domótico***, ha sido realizado bajo su supervisión por **Ramón Fernández Gualda**, y autoriza la defensa de dicho trabajo ante el tribunal que corresponda.

Y para que conste, expiden y firman el presente informe en Granada a 21 de junio de 2017.

Los directores:

Jorge Navarro Ortiz **Sandra Sendra Compte**

Agradecimientos

A mis Padres, por su apoyo y sacrificio incondicional. Vosotros me enseñasteis los valores que hoy tengo. A mi Hermano con mayúscula. A Daniel, mi maestro, por guiarme en este arduo camino. A todos aquellos amigos y personas que han contribuído a que hoy en día sea un poquito más que ayer, como persona y como futuro profesional en este campo tan amplio y bonito de la Ingeniería.

A mi tutor Jorge Navarro, por su constante consejo, siempre dispuesto a ayudar, implicado en el trabajo y ofreciendo siempre una solución a cada problema. Gracias por hacer esto posible.

Constancia, esfuerzo y disciplina.

Índice general

Índice de figuras

Índice de cuadros

Siglas

ADC Conversión Analógica Digital. 51

AGC Control Automático de Ganancia. 35

CAGR Tasa de Crecimiento Anual Compuesto. 4

CPU Unidad Central de Proceso. 51

GND *Ground*. 85

GPIO *General Purpose Input/Output*. 51

I2C *Inter-Integrated Circuit*. 51

IDE Entorno de Desarrollo Integrado. 51

IEEE Instituto de Ingeniería Eléctrica y Electrónica. 51

IoT *Internet of Things*. 3

IP *Internet Protocol*. 7

IR Radiación Infrarroja. 7

LED Diodo Emisor de Luz. 12

LSB Bit Menos Significativo. 40

M2M *Machine To Machine*. 4

MQTT *Message Queue Telemetry Transport*. 22

MSB Bit Más Significativo. 39

NRZ No Retorno a Cero. 44

PCB *Printed Circuit Board*. 27

QoS *Quality of Service*. 62

QSPI *Queued Serial Peripheral Interface*. 51

RAM *Random Access Memory*. 25

ROI Retorno de Inversión. 6

RST Reset. 50

RTL *Resistor Transistor Logic*. 91

SCE Sistema de Cableado Estructurado. 2

SDR *Software Defined Radio*. 91

SoC *System on Chip*. 61

SPI *Serial Peripheral Interface*. 51

SRAM *Static Random Access Memory*. 59

SSL *Secure Sockets Layer*. 61

STOMP *Streaming Text Oriented Messaging Protocol*. 61

TCP Protocolo de Control de Transmisión. 62

TIC Tecnologías de la Información y Comunicación. 5

TLS *Transport Layer Security*. 61

UART *Universal Asynchronous Receiver-Transmitter*. 51

VCR Videograbador. 46

WEP *Wired Equivalent Privacy*. 51

Wi-Fi *Wireless Fidelity*. 12

WLAN *Wireless Local Area Network*. 6

WPA *Wi-Fi Protected Access*. 51

XMPP *Extensible Messaging and Presence Protocol*. 61

Capítulo 1

Introducción

En este capítulo se pretende realizar una introducción que ayude a comprender el contexto de los aspectos que se pretende abarcar.

Primeramente se redactará un apartado que explica las motivaciones que han llevado a realizar este proyecto tratando de exponer y evaluar la situación actual en el mercado de las tecnologías involucradas en el mismo como puede ser la implementación de la domótica y la comunicación por infrarrojos de manera que se de a conocer la importancia de este estudio.

Asimismo, se expondrán en los apartados posteriores los objetivos principales en los que se enfoca el presente proyecto y la planificación que se ha seguido para finalmente explicar la estructura detallada por capítulos a modo de resumen de todo el trabajo.

1.1. Contexto y motivación

Uno de los principales motivos que llevan a realizar este trabajo es debido al creciente uso de la domótica en nuestro día a día.

La domótica, como se conoce actualmente, es aquel campo de la tecnología que se aplica en el hogar, cuyos hogares son conocidos como smart home. Sus objetivos clave son la comodidad, el ahorro energético, la seguridad, facilitar la permanente comunicación, o la accesibilidad para favorecer la autonomía de aquellas personas con limitaciones funcionales donde se pretende tener un impacto positivo. Todo ello mediante la automatización y el manejo remoto de dispositivos a nuestro alcance.

El término "domótica" [1]está compuesto por dos palabras: *domus*, que significa *casa* en latín, y *tica*, de *automática*, o que funciona por sí sola, del griego, y está definida en el diccionario de la Real Academia como "el conjunto de sistemas que automatizan las diferentes instalaciones de una vivienda". Incluso, aunque este término se ha extrapolado a cualquier tipo de

instalaciones, eso no sería correcto, puesto que para instalaciones de edificios mucho más complejos que las destinadas a viviendas, ya existe otro término conocido como Inmótica.

La domótica tiene su origen la década de los setenta, donde se llevó a cabo una serie de investigaciones en las cuales dieron luz a los primeros dispositivos de automatización de edificios basados en la tecnología X-10. Durante los años venideros el público mostró un importante interés en la búsqueda de la vivienda ideal en la que primeramente, solo se realizó la instalación de sistemas comerciales con algunos dispositivos automáticos en Estados Unidos que generalmente se encargaban de la regulación de la temperatura ambiente de oficinas. Más tarde, con el auge de los ordenadores personales a finales de los ochenta, fue necesaria la instalación en los edificios de los SCE (Sistema de Cableado Estructurado) que permitiesen la comunicación en una red de área local transportando datos. Esto fue aprovechado para permitir el envío de voz o dispositivos de seguridad y control, por lo que las construcciones o edificios que contaban con una instalación de un SCE se denominaban "edificios inteligentes".

En España no se implanta hasta inicios de los noventa donde se empiezan a dar a conocer dicha tecnología, pero no es hasta hace relativamente unos pocos años atrás cuando se solidifica en la sociedad el concepto. Aunque aún es relativamente pequeña la cantidad de hogares inteligentes, su presencia va en aumento para contribuir al aumento de la calidad de vida.

La continua expansión tecnológica de este campo en estos últimos años conlleva a un aumento de nuevas ideas y proyectos en la iluminación, climatización, en seguridad, o en comunicación, que da lugar a un amplio abanico de posibilidades en su implementación que aseguran la prosperidad de las smart homes en los años venideros. Cambios no solo aplicables a aquellos lugares donde vivimos sino en lugares de trabajo u ocio. Es un hecho ver como al igual que es de vital importancia el suministro eléctrico y de agua corriente en las casas, también lo será el de asegurar unas instalaciones mínimas domotizadas.

La normalización de estos términos permitirá a compañías de la construcción, que incluyan en sus proyectos esta tecnología, una mayor competitividad en el mercado, por lo que una gran cantidad de personal como arquitectos, instaladores o diseñadores verán de interés familiarizarse con la domótica.

De hecho uno de los aspectos que más han ralentizado el uso de la domótica ha sido sus inicios, pues poca gente estaba dispuesta a pagar costes adicionales por ella, pero debido a su normalización y a la actual reducción de precios, se ha conseguido que ya nadie pueda imaginarse su hogar sin este tipo de tecnología.

La cantidad de funciones que actualmente se permiten en un hogar digital es debido en gran medida al progreso continuo tecnológico sufrido en los sistemas de telecomunicaciones, la expansión de internet y la aparición del IoT (*Internet of Things*), quienes otorgan la capacidad desmesurada de crear, transmitir, procesar y ejecutar la información que se desea llevar a cabo, de tal manera que nos permite la conexión entre diferentes dispositivos en el hogar o incluso servicios interactivos.

Es posible encontrar una enorme lista de aplicaciones dependiendo del uso de los dispositivos domóticos: Por ejemplo para la climatización y consumo energético está disponible la programación del encendido y apagado de aparatos como calderas, aire acondicionado, toldos o iluminación, contadores electrónicos que gestionan la información acerca del consumo energético; para el ocio y comfort se puede encontrar la rápida y fácil accesibilidad a internet desde cualquier parte del hogar, juegos en red, visión de canales de televisión, control de aparatos electrónicos del hogar desde un PC o desde el móvil mediante internet; para seguridad se encuentra implantación de sensores de vigilancia como micrófonos o cámaras, la gestión para los avisos en caso de intrusión o avería como alarmas, y el control del acceso a la residencia; para servicios comunitarios está el control de la iluminación de zonas comunes, servicios web para la comunidad de propietarios o la gestión de alarmas de seguridad y alarmas técnicas. Como se ha comentado al principio, estas aplicaciones serían solo unos pequeños ejemplos de la enorme lista que hay en este amplio campo (véase Figura 1.1).

Figura 1.1: Aplicaciones de la domótica [1]

Según un estudio realizado por la consultora sueca Berg Insight [2](especializada

en estudios de mercado del ámbito M2M/IoT) ha declarado que 91 millones de casas en Europa y Norteamérica serán inteligentes para 2020.

Además, considera que hoy en día un correcto sistema de casa inteligente debe añadir una interfaz de usuario mediante una aplicación de *smartphone* o un portal web. Se diferencian incluso hasta seis categorías diferentes de sistemas domóticos como son: Sistemas de gestión energética y climatización; Sistemas de seguridad y control de acceso; Sistemas de control de iluminación, ventanas y aparatos; Electrodomésticos; Sistemas audiovisuales y de ocio; Y por último, sistemas asistenciales.

También se informa acerca del enorme crecimiento en el mercado de las *smart homes* durante el año 2015, las cuales aumentaron un 62 % suponiendo un total de 16,9 millones de bases instaladas a finales de año, siendo 2,8 millones sistemas multifunción o de instalación para la casa completa, mientras que el resto (14,1 millones) estaban destinadas a soluciones que tenían como fin una función específica. Solo en América del Norte se registró un total de 12,7 millones de hogares inteligentes, que suponen un 9.7 % del total de las casas del país, situando al susodicho como el mercado de las viviendas inteligentes más avanzado del mundo.

Entre 2015 y 2020 se estima que el número de hogares con tecnología domótica será de unos 46,2 millones. Los ingresos obtenidos de este mercado alcanzaron la cifra de 5,4 millones de € en 2015 mientras que para 2020 se espera unas ganancias de hasta 21,2 millones de €.

El mercado Europeo se verá dentro de 2 o 3 años en la misma situación que América del Norte en términos de implementación y madurez del mercado. Se han recogido cifras de hasta 6,6 millones de sistemas de hogares inteligentes solo en 2015 frente a los 3,3, millones del año anterior. Alrededor de 0,8 millones de esos sistemas eran multifunción o de casa completa, mientras que los otros 5,8 millones eran soluciones concretas. Se pronostica así un un crecimiento del CAGR (Tasa de Crecimiento Anual Compuesto) del 54 % durante los siguientes cinco años de la cantidad de viviendas inteligentes en Europa, dando lugar a 44,9 millones de *smart houses* en 2020, cuyos ingresos equivaldrían a 12.800 millones de euros para esa fecha.

En la siguiente tabla de la Figura 1.2 se muestra de manera más gráfica [2] como el número de hogares (en millones), tanto en Europa como en Norteamérica, va en continuo aumento y como con el tiempo, los dos mercados van tomando cierta similitud, más concretamente para 2020.

Aquellas soluciones puntuales ya implantadas han significado un 58 % de los ingresos combinados entre Europa y América del Norte en comparación con los sistemas de hogar completo. Entre estas soluciones específicas planteadas más exitosas se encuentran termostatos inteligentes, sistemas de seguridad, bombillas inteligentes, cámaras y sistemas de audio en habita-

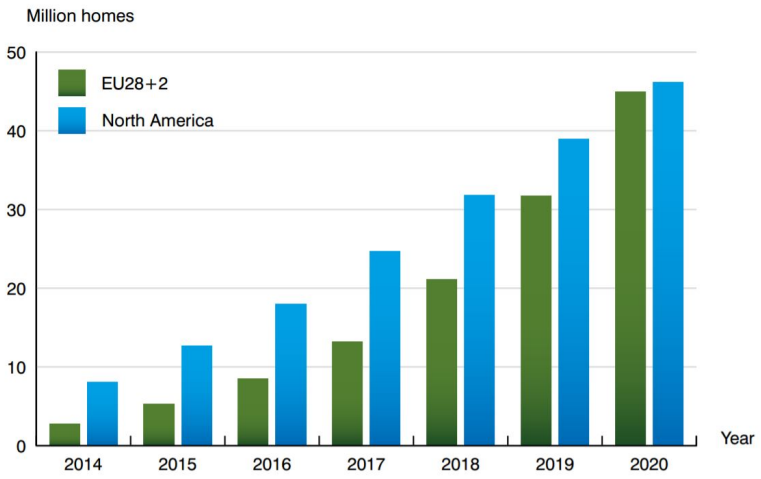

Figura 1.2: Evolución del número total de smart homes

ciones. Para estos productos, los mayores proveedores que los comercializan son Philips Lightning, Honeywell, Danfoss, Belkin, Chamberlain, Kwikset y Assa Abloy, aunque cada vez más hay nuevas empresas que quieren unirse al mercado. Para lo sistemas de hogar completo las compañías tradicionales son Creston, Control4, Gira y Jung. Los principales proveedores en América del Norte, en general son: Comcast, ADT y Vivint; Y en Europa nos encontramos con: Verisure, RWE, Deutsche Telekom y Loxone.

Si bien las aplicaciones en los móviles como interfaz de usuario están a la orden del día, también es cierto que en el futuro se pronostica que será necesario la unificación de plataformas o aplicaciones en una sola interfaz debido al problema de la creación de aplicaciones individuales por dispositivo. Una solución se puede encontrar en la voz de manera que con ella se conecten y controlen una amplia gama de dispositivos y servicios por comandos de voz.

Amazon, gigante de la industria TIC, ya ha mostrado una solución de esto con Alexa, la cual es una asistente controlada por voz que ha tenido una buena acogida y la plataforma *HomeKit* de Apple mediante Siri junto con empresas como Microsoft y Google que ya están terminando de preparar sus ideas.

Otro punto a favor para la domótica es, como ya se ha comentado con anterioridad, el ahorro de consumo energético. Tal es así que la Unión Europea ha financiado el proyecto llamado "DOMOTIC" [3] mediante el "Programa LIFE" que se encarga de impulsar el desarrollo de proyectos innovadores relacionados con la política comunitaria de medio ambiente. Dicho proyecto se basa en la demostración de la reducción de emisiones de dióxido de carbono mediante el uso de tecnologías inteligentes como la Domótica e Inmótica

puesto que supone una prioridad para la Unión Europea en esta materia. Teniendo sobre todo en cuenta aspectos como que la climatización e iluminación suponen el 40 % del consumo de energía que se genera en la UE son los causantes de las emisiones del dióxido se hace comprender que no es aspecto nada despreciable. Sin embargo, una correcta medida de eficiencia energética llevada a cabo en el sector podría suponer un ahorro energético del 74 %.

En dicho proyecto se han realizado pruebas en tres edificios: dos centros educativos (Centro de Educación Secundaria San Valero y el campus de la Universidad de San Jorge) ambos en la Comunidad Autónoma de Aragón y el espacio de Propuestas Ambientales Educativas (PRAE) en la Comunidad Autónoma de Castilla y León; donde se ha demostrado que mediante la automatización de los edificios se puede mejorar la eficiencia energética en hasta el 63,9 % anual, con un ahorro económico de 162.000€ al año, con un ROI (Retorno de Inversión) de 4 años. Por ello como resultados se obtienen un ahorro de entre el 40 % y 50 % en el consumo de la energía eléctrica, un ahorro del gasóleo para la calefacción del 30 % de media, la detección de fugas y funcionamientos anómalos en las instalaciones ahorrando futuras pérdidas a más largo plazo y un alto ahorro económico.

Otro aspecto importante que no se debe olvidar y de gran relevancia en este proyecto y que en conjunción con la domótica tendrá un papel primordial, será el de las comunicaciones infrarrojas.

La enorme cantidad de aplicaciones a las que se les ha dado uso y que hoy en día se utiliza es amplio, como los equipos de visión nocturna, o como medio en la fibra óptica, o incluso terapias hipertérmicas y tratamiento del cáncer, pero un uso muy común es el que se encuentra con los controles remoto.

Es un hecho que, si se recapacita un poco, enseguida nos cercioramos de que las comunicaciones infrarojas nos rodean dentro del hogar. La mayoría de los dispositivos del hogar como TV, Aire Acondicionado, DVD, TDT o aparatos de música, reciben comandos a través de uno de estos controles remoto que se basan en infrarojos.

Aunque dentro del hogar se pueden ver muchos tipos de comunicaciones inalámbricas como los sistemas de telefonía celular, comunicación móvil satelital y redes de área local inalámbricas (WLANs), aparte de la anterior mencionada, tienen una serie de diferencias tales como las frecuencias de operación, anchos de banda, velocidades de transmisión, esquemas de acceso a la red, cobertura o movilidad.

Respecto a los sistemas de radio frecuencia, los sistemas de comunicaciones infrarrojas cuentan con un ancho de banda muy grande y no está regulado en ninguna parte del planeta.

Además son inmunes a interferencias radioeléctricas, y al no atravesar paredes, cada cuarto representaría una celda que no interfiere con las demás celdas, aceptando así una alta densidad de reúso del sistema. Esto además hace más complicado que sea susceptible a captarse mediante escuchas clandestinas. Este confinamiento favorece incluso si se quiere mejorar la relación S/N, con el simple hecho de aumentar la potencia de la señal que se transmite puesto que no se causarán problemas de interferencia entre celdas vecinas.

De esta manera se logra comprender porqué se recurre tanto a este tipo de comunicación inalámbrica dentro del hogar.

1.2. Objetivos principales

Teniendo en cuenta todo el contexto anterior, entre los objetivos principales que persigue este proyecto es el de la creación de una solución domótica *low-cost* capaz de gestionar una gran variedad de protocolos infrarrojos, mediante una plataforma universal, para una exitosa comunicación entre los dispositivos del hogar que permiten dicha comunicación y los terminales móviles de usuario, tablets u ordenadores personales de una determinada red IP, desde donde se ejecutaran dichas órdenes.

En el caso de que se añada un nuevo dispositivo que no realice ninguna acción ante el envío de una orden, el sistema propuesto tendrá la capacidad suficiente para aprender las nuevas órdenes por las que se rige el nuevo dispositivo, para después poder controlarlo de manera remota.

Esta automatización en el envío de comandos se pretende implementar de manera fiable mediante un protocolo estandarizado con una plataforma sencilla de mensajes que haga de pasarela entre el *smartphone* y el circuito diseñado capaz de controlar remotamente los distintos dispositivos receptores de señales IR. Esta sencilla solución pretende abaratar costes en el contexto de la domótica.

Otros aspectos secundarios que se aprecian en este proyecto es el de entender el funcionamiento de los diferentes protocolos IR de las diferentes marcas, así como la familiarización en la realización del envío de este tipo de señales infrarrojas de distancia cercana a modo general.

1.3. Estructura de la memoria

En esta sección se define la estructura del proyecto, se van a mostrar los diferentes capítulos que al componen seguida de una breve descripción de ese capítulo. Esto ayudará al lector a realizar un breve reconocimiento de la sección que se desee investigar. En total existen ocho capítulos con su correspondiente bibliografía:

Capítulo 1: Introducción

En el capítulo número uno se realiza un contextualización de los sucesos que han dado lugar a la motivación para la realización de este trabajo. Además se explican los objetivos principales que se pretenden llevar a cabo, explicando por último aquella bibliografía en la que se basa principalmente el proyecto, siendo la que más veces se ha investigado.

Capítulo 2: Estado del arte

En el mercado actualmente se encuentran algunos dispositivos que realizan funciones parecidas a las que se van a implementar en el proyecto, por eso se cree importante el análisis y comparación de esos productos para valorar respecto de ellos que funcionalidades adicionales pueden ofrecernos este nuevo diseño o mejoras que pueden influir en el desarrollo de un producto más completo.

Capítulo 3: Especificación de requisitos

En este tercer capítulo llamado "Especificación de requisitos" se argumenta qué finalidad consigue la solución propuesta y mediante qué procedimiento se va a emplear, estos dos explicaciones vienen en los enunciados de requisitos funcionales y requisitos no funcionales.

Capítulo 4: Planificación y estimación de costes

Se proporciona una planificación temporal seguida en el desarrollo del proyecto, con todas sus fases.

Además, este apartado está creado exclusivamente para el análisis de todo el material, tanto software, como hardware, como humano que se ha invertido para hacer factible el dimensionamiento de la solución, de modo que el lector interesado en la realización del mismo sea consciente de todas las herramientas que se necesitan, así como un precio estimado desglosado y total.

Capítulo 5: Tecnologías implicadas

Se pretende establecer las bases del fundamento teórico para poner al día al lector de todas las características fundamentales de las distintas tecnologías que cubren este proyecto como pueden ser el infrarrojo y sus protocolos asociados entre otros.

Capítulo 6: Diseño e implementación

Apartado en el cual se explica el diseño llevado a cabo como solución explicado al detalle, incluyendo además otras variaciones del circuito para

transmitir con más potencia, la instalación de servicios necesarios para la comunicación dentro del hogar, y la composición del mensaje que se enviará al control remoto que lo interpretará y ejecutará según este.

Capítulo 7: Pruebas realizadas y validación

En dicho capítulo se tiene en cuenta el apartado anterior, donde una vez completado el diseño, se realizarán una serie de pruebas que certificarán que se cumple con la misión del proyecto, además se analizarán otros parámetros del mismo a modo de información adicional en la que el lector, teniendo en cuenta esa información podrá elegir según su criterio, qué diseño le sería más conveniente o interesante.

Capítulo 8: Conclusiones y trabajos futuros

Este capítulo cierra el proyecto. Incluye un resumen con una vista general a todo lo realizado en él y una serie de reflexiones que han dado lugar durante su realización al igual que posibles mejoras propuestas y que serían de interés tener en cuenta en casos futuros.

Bibliografía

Como toque final, se muestran todas aquellas bibliografías que han servido de consulta a lo largo de todo el proyecto.

1.4. Principales fuentes bibliográficas

Aquí se muestran las principales fuentes de mayor relevancia para la realización y desarrollo de este proyecto, utilizadas como apoyo y motivación para el mismo. También, al final de la memoria habrá una ampliación de toda la bibliografía partícipe de la misma.

La base de conocimiento sobre la composición y forma de los distintos protocolos de infrarrojos no ha sido posible encontrarlos recogidos en un solo libro. Para poder obtener toda esa información, ha sido preciso la búsqueda de los distintos fabricantes, ya que son protocolos propietarios, de sus *datasheets* u hojas técnicas donde se indica con la mayor precisión las características que envuelve al formato de la señal IR que se envía.

- Neil Kolban (Nov 2016). *"Kolban's Book on the ESP32 and ESP8266"* [En línea]. Disponible en: `https://leanpub.com/ESP8266_ESP32`

Capítulo 2

Estado del arte

Principalmente, esta sección tratará de mostrar algunos productos que pueden encontrarse en el mercado como solución para la comunicación con dispositivos finales del hogar que reciben comandos del usuario o cliente, exponiendo sus características, utilización y coste. Por otro lado, se comparará estos con la solución propia diseñada argumentando sus ventajas respecto de ellas y de porqué supone una mejora.

A continuación se describen los principales dispositivos descubiertos en el mercado y que son de mayor relevancia.

Xiaomi IR

Figura 2.1: Dispositivo Xiaomi IR

Este dispositivo (Figura 2.1) es un controlador remoto universal de señales IR [4] que puede sustituir al control remoto de un dispositivo del hogar ordinario. Es capaz de establecer conexión con una gran variedad de dispositivos

finales del hogar que reciban infrarrojos y todo controlado mediante nuestro terminal móvil conectado a internet, que hará uso de una aplicación desde la que se ejecuta los comandos deseados incluso estando el *smartphone* fuera del hogar. Su estructura redonda permite un ángulo de visión de 360 grados (conseguida gracias a una estructura de 6 LEDs apuntando a todas las direcciones), una distancia de control remoto de 20 metros, con un componente transmisivo de luz negra capaz de reducir la reflexión difusa mejorando la penetración infrarroja. En cuanto a su alimentación trae un cable USB para conectarlo a la corriente. Este dispositivo recibe los comandos del móvil aún cuando se encuentra fuera del hogar ya que Xiaomi IR cuenta con conexión Wi-Fi.

El único sistema operativo de móvil que permite es Android 4.4 o superior. Utiliza una aplicación concretamente que tiene una interfaz de usuario para la elección del comando a ejecutar, se llama "Mi *Smart Home*". Por último, su precio es de 20€.

Es un dispositivo muy potente con una plataforma que permite la conexión con todos los aparatos del hogar, pero sin embargo, como punto a favor de nuestra solución encontramos el precio, una menor cantidad de componentes con una plataforma mucho más flexible en cuanto al tipo de mensaje que se envía desde el móvil al dispositivo que actuará de control remoto, un peso más ligero y consumo muy inferior, puesto que la potencia transmitida es la suficiente para abarcar el rango de una habitación, son algunos de los aspectos positivos que se aprecian de nuestra solución. También como punto positivo, cabe mencionar que no existe ninguna limitación en cuanto a sistema operativo, es válido para cualquier móvil con capacidad para la instalación de una aplicación.

Existen muchos más modelos de diferentes marcas con la misma funcionalidad que este producto de Xiaomi, como el AnyMote Home cuyo precio asciende a 90€, pero se ha preferido hacer mención de este en concreto porque respecto de otros modelos, su precio es mucho más económico y pertenece a una marca muy reconocida por el abaratamiento de costes de sus productos.

Otro dispositivo parecido es Orvibo AllOne IR [5] (véase la Figura 2.2). Por el módico precio de 30€, Orvibo ha desarrollado un producto capaz de controlar hasta 8 aparatos mediante señales infrarrojas que consiste en un control remoto con forma de "platillo volante" como se ha aprecia en la imagen. Este control remoto actúa de hub desde el cual los comandos que se ordenan desde nuestro móvil son ejecutadas. Válido tanto para iOS y Android mediante una aplicación llamada Wiwo en la que después de un previo registro y configuración dará acceso remoto al Orvibo AllOne incluso desde fuera de nuestra red local para su gestión y acción desde fuera del hogar.

Figura 2.2: Dispositivo Orvibo AllOne IR

Aunque resulta interesante, se encuentra limitado en cuanto a la cantidad de dispositivos a los que permite acceso, y la distancia de alcance se ve relativamente disminuída, siendo superior el alcance en cualquiera de las propuestas en el diseño de la solución sugerida, sin contar con el inconveniente del precio.

Dispone de una base de datos actualizable por Internet dispuesta a ofrecer cobertura a todos los protocolos propietarios posibles, facilitando un modo de aprendizaje si dicho dispositivo del hogar no se encuentra registro en la base de datos. Sus dimensiones son de 11cm de diámetro y 3cm de grosor, alimentado mediante un conector hembra micro USB a 5VDC. Su forma redonda permite cubrir hasta un campo de visión de 360º con un alcance de aproximadamente 2 metros.

Además de estos dispositivos, comienzan a aparecer equipos de aire acondicionado que son controlables por Wi-Fi. Por ejemplo, Samsung[6] ofrece una gama de aires (Figura 2.3) acondicionados caracterizados por controlar sus funciones mediante Wi-Fi integrado, desde un teléfono móvil con sistema Android, tanto fuera como desde dentro del hogar. Resulta útil si, por ejemplo, hay necesidad de dejar a punto la temperatura antes de llegar al hogar. Mediante una orden en nuestro terminal móvil desde fuera de casa podemos accionar el aire acondicionado para que todo esté listo a la llegada.

Hace uso de una aplicación en el móvil conocida como "*Smart Air Conditioner*" de manera que este hace la función de un cómodo control remoto sustituyendo al tradicional mando a distancia incluído en un dispositivo aire acondicionado.

Esta novedad de Samsung ofrece tres posibles modelos: El climatizador *F-H7709 Triangle Design Wi-Fi Virus Doctor*, el *F-H7700* y el *U.Interior Split de pared compatible FJM Smart Home.*

Cabe resaltar su elevado coste, que aunque varía según la potencia, el

Figura 2.3: Climatizador Samsung

precio mínimo por uno de estos modelos sería de 200€ hasta un máximo que ronda los 900€, precio nada desdeñable.

Es obvio a la hora de comparar con nuestro proyecto, que lo que más llamativo resulta es la enorme diferencia de precio, incluir en este dispositivo la tecnología Wi-Fi ha aumentado significativamente su valor, mientras que con la solución planteada ni siquiera es necesario comprar un nuevo climatizador que permita dicha comunicación, se podría seguir utilizando el anterior mediante infrarrojos. Otra de las proezas que destacan en nuestro proyecto, además de lo anterior, es que contiene una plataforma universal, permitiendo mucha más flexibilidad en cuanto a número de protocolos IR permitidos. Sería muy laborioso tener que almacenar en memoria más aplicaciones en el dispositivo móvil por cada marca de dispositivo que se encuentre en el hogar. Otro dato a tener en cuenta es que se desconoce el protocolo utilizado por dicha empresa. Al ser privado y mediante comunicación Wi-Fi, se desconoce la base de datos que albergan los comandos que se emiten.

Otro tipo de control a distancia de comandos IR parecido, es un pequeño módulo STB[7] que se conecta al teléfono móvil mediante el puerto jack, convirtiendo así al *smartphone* en un control remoto universal sin la necesidad de tener un mando para cada dispositivo del hogar.

Como se aprecia en la Figura 2.4, en el otro extremo del módulo se encuentra el LED IR que emitirá las señales oportunas. El sistema operativo del móvil puede ser tanto Android como iOS, que permita la descarga de la aplicación que mostrará la interfaz de usuario en forma de mando en la pantalla táctil del *smartphone*, transformando la pulsación de un botón en un comando IR transmitido mediante el LED infrarrojo del STB.

Aunque el precio de la unidad puede llegar a asemejarse al de nuestra solución, con un par de euros, vemos que la limitación en cuanto a conexión es grande. Mientras nuestra solución permite accionar los dispositivos finales desde fuera del hogar, el módulo STB conectado al *smartphone* solo se limita

Figura 2.4: Módulo STB IR

al envío de señales infrarrojas, por lo que su conexión solo abarca aquella distancia que permita la potencia de transmisión del infrarrojo.

Capítulo 3

Especificación de requisitos

En esta sección se realizará una especificación tanto los requisitos funcionales como los no funcionales que se deben cumplimentar para el correcto desarrollo del proyecto, por lo que la solución que se exponga debe cumplir con dichos requisitos.

Se ha decidido dividir en estos dos subgrupos para que se identifiquen con mayor claridad y precisión todos los aspectos a llevar a cabo mediante documentación sobre el contenido del trabajo en la fase del diseño.

3.1. Requisitos funcionales

Se definen los requisitos funcionales como aquellas características que formarán parte de la solución, es decir, aquellos requisitos que debe cumplir que verifiquen el correcto trabajo del diseño. A continuación se enumeran dichos requisitos:

- **Emisión de señales infrarrojas mediante un dispositivo:**

  El dispositivo que se pretende desarrollar debe posibilitar la transmisión de señales infrarrojas en las cuales se transportarán las órdenes que el cliente desee ejecutar, que dependiendo de la marca, el protocolo tendrá una estructura u otra por lo que el mensaje transportado deberá incluir para su correcta recepción posterior. Hay que recordar que dicha plataforma soportará una amplia variedad de protocolos IR.

- **Recepción de señales infrarrojas a través del dispositivo:** Es muy probable que, en términos prácticos, se introduzca un nuevo dispositivo en el hogar con un protocolo propietario que impida el conocimiento de la base de datos que utilice, o que haga uso de otro protocolo que aún no esté incluído en la actual base de datos. Es por ello que para no poner límites en la plataforma, se permitirá el aprendizaje de códigos de nuevos protocolos activando el modo “recepción”

almacenando los nuevos comandos correctamente en la base de datos para obtener con el tiempo una solución más completa y mejorada.

Como aspecto secundario, a la captación del nuevo código a aprender, el dispositivo en escucha intentará adivinar de qué protocolo se trata para obtener más información acerca del mismo.

- **Creación de dispositivo con capacidad de conexión con dispositivos del hogar:**

  Teniendo en cuenta los dos requisitos anteriores, se pretende alcanzar una solución que unificando las funcionalidades anteriores permita controlar mediante comunicación infrarroja los dispositivos finales del hogar, como aires acondicionados, TVs, reproductores de audio o video (y un largo etcétera), actuando de control remoto. Se ha decantado por la comunicación infrarroja al ser ese tipo de tecnología la que acepta casi todos los dispositivos mencionados anteriormente.

  Dicho dispositivo será distinto al tradicional al tener doble funcionalidad, tanto emisión como recepción de comandos IR.

- **Dispositivo en constante comunicación con base de datos mediante una red IP:**

  El resultado de una comunicación continua con la base de datos facilita el almacenamiento de código en el dispositivo, que solo actuará de pasarela encargándose de añadir las cabeceras correspondientes en el mensaje que será recibido desde la base de datos.

  Una conexión con una red IP permite que pueda ser controlado por equipos terceros que incluso se encuentren fuera de la vivienda ampliando el rango de comunicación.

- **Utilización de un protocolo estándar "Publicador - Suscriptor":**

  Como requisito interesante, se aconseja la utilización de un protocolo estándar "Publicador-Suscriptor" de poco peso para el envío y recepción de mensajes que permita al dispositivo su integración en otras soluciones como por ejemplo en sistemas domóticos.

  Se buscará que facilite su desarrollo en el tipo de mensajes, ofreciendo flexibilidad en cuanto a la estructura del mensaje que gestiona.

- **Interpretación de los mensajes del protocolo ligero:**

  El dispositivo de pasarela además de cumplir con las funcionalidades necesarias que permiten la comunicación con los dispositivos finales,

debe de ser capaz de interpretar los mensajes recibidos del dispositivo móvil o dispositivo que actúe de cliente mediante el protocolo ligero empleado para posteriormente ejecutar el envío de una determinada señal infrarroja o un comando RAW genérico.

3.2. Requisitos no funcionales

Los requisitos no funcionales son aquellas condiciones que la solución debe cumplir que se deben llevar a cabo durante las fases de diseño e implementación.

Como requisito software se necesita:

- **Arduino**

  El dispositivo pasarela debe permitir programarse mediante este software para que pueda realizar la interpretación, con un código determinado, de los diferentes mensajes que recibirá del protocolo *lightweight* seleccionando el tipo de señal infrarroja que más tarde recibirá el dispositivo final.

  Es un software extensamente conocido que ha desarrollado una serie de librerías que facilitan la automatización de la programación de este tipo de comunicaciones.

Después también existe otro requisito hardware importante:

- **Utilización del chip ESP8266**

  Este chip proviene de la idea del tutor. Las justificaciones de su uso se deben a que es un componente a un bajo precio muy llamativo, tiene una antena Wi-Fi que permite dicha conexión, lo que lo mantendría comunicado con la base de datos y en constante escucha con el cliente y fácilmente programable con Arduino.

Capítulo 4

Planificación y estimación de costes

Este capítulo se realizará un desglose de la planificación en la elaboración del proyecto, así como los costes estimados de cualquier tipo asociados al mismo.

En primer lugar se explican las diferentes fases del desarrollo del trabajo desde el inicio, incluyendo una descripción de cada una de las diferentes partes. También se realizará una estimación del tiempo dedicado a cada fase y para una visualización más completa de lo anterior se muestra una representación gráfica de los apartados con un diagrama de Gantt.

Hasta aquí abarcaría la planificación, mientras que seguidamente se reconocerán los recursos implicados de cualquier tipo, ya sean hardware, software y humanos.

La última parte concluirá con un desglose de la estimación del coste del proyecto teniendo en cuenta tantos aquellos recursos humanos como materiales.

4.1. Planificación

En dicho apartado se expondrán los diferentes secciones o fases de la planificación del proyecto para mostrar con claridad el desarrollo que se ha seguido acompañado de una descripción en cada uno que detalla el proceso del mismo.

Las secciones que forman parte de la planificación para la realización de este proyecto son:

- **Estudio del estado del arte**

  Parte de la planificación que se centra en la extracción de la mayor cantidad de información posible acerca de las tecnologías y herramientas

relacionadas con este tipo de estudio que dará las bases a un correcto desarrollo del mismo.

En esta primera fase se ha procedido a un análisis de los diferentes modos de estructura de los mensajes de infrarrojos, métodos de envío y tipos de comandos. Un conocimiento previo acerca de las comunicaciones infrarrojas que ha ayudado a verificar que es la más apropiada, en vez de por ejemplo el uso de las comunicaciones mediante Bluetooth. También se indaga acerca de qué tipo de dispositivo sería más conveniente utilizar a modo de pasarela de acuerdo al precio, tipo de programación, espacio y conexiones permitidas de acuerdo a la búsqueda de la posible solución más óptima.

- **Familiarización con el chip ESP8266**

  La siguiente fase consiste en la toma de contacto con el chip ESP8266. Se profundizará en el aprendizaje de este herramienta seleccionada, así como conocer sus características u hoja técnica para más tarde indagar acerca de qué módulos disponibles en el mercado actual integran el chip, cual se adaptaría mejor a nuestro proyecto, estudio de las arquitecturas correspondientes y su funcionamiento descubriendo todas y cada una de las posibilidades que nos ofrecen. La elección del módulo es determinante ya que debe de tener capacidad para tener conexión tanto por red IP como por infrarrojo.

- **Conocimiento del protocolo MQTT**

  Para la continuación del proyecto será imprescindible la elección del protocolo de tipo "Publicador-Suscriptor", que en este caso será MQTT (*Message Queue Telemetry Transport*). Por tanto, es necesario una recopilación de información para la comprensión de su funcionamiento y gestión de mensajes. También se diseñará una estructura de los mensajes a enviar, que será la que reciba el dispositivo mediante el ESP8266.

- **Pruebas de programación con software Arduino**

  Se hará uso de las librerías habilitadas para el chip ESP8266. Se realizan pruebas con este software que hace uso del lenguaje de programación C. En esta fase se configurará el dispositivo para, una vez definidos los mensajes MQTT, que sea capaz de interpretarlos y en función de ello realizar una orden u otra dependiendo de qué tipo de protocolo se quiera enviar, si es un código RAW o qué tipo de frecuencia portará la señal IR.

- **Diseño e implementación del circuito final**

  Esta fase será a la que más tiempo se le ha dedicado, puesto que todos los anteriores conocimientos serán puestos en práctica para la implementación de la solución final. El desarrollo del circuito de la solución

contendrá la parte de emisión y la parte de recepción de comandos, y se justificará el escenario planteado en cada parte del circuito. Además debe adecuarse a las órdenes que se le exigen al circuito enviando correctamente el comando sugerido por lo que el desarrollo del *firmware* deberá ordenar al módulo su puesta en modo de emisión, y en caso de aprendizaje, activar su modo en escucha.

- **Fase de pruebas**

  Una vez terminado el diseño e implementación se establece un período de pruebas en el que la solución será sometida a diferentes pruebas que determinen el correcto funcionamiento, el cumplimiento de los objetivos y datos adicionales del mismo que ofrezca. Algunas pruebas parciales son: Con diferentes mandos se puede probar la veracidad del protocolo que emite dicho mando, también si el dispositivo aprende correctamente los comandos. En la emisión de infrarrojos, dependiendo del circuito empleado se puede hallar la distancia o rango que cubre el infrarrojo, y si el cliente MQTT ejecuta el comando que eligió en un principio. Todas estas pruebas conseguirán detectar posibles fallos y ayudarán a determinar si se ha completado con éxito el proyecto.

- **Elaboración de una memoria técnica del proyecto**

  Por último, y junto a la fase de "Diseño e implementación", será una de las fases a las que también se ha necesitado una importante cantidad de tiempo. Será la parte en donde todo lo relativo al proyecto pasa a ser documentado en una memoria técnica que recoge los conocimiento teóricos, los aspectos técnicos, los aspectos concretos del diseño puesto en marcha, los resultados obtenidos y las conclusiones que confirman la validez del trabajo.

Como ya se comentó al principio del Capítulo 4, una vez conocidas las diferentes fases y descritas, se dará paso a la representación gráfica de la planificación dividida en cada una de las secciones, que como ya se advirtió se mostrará con el diagrama de Gantt. A continuación se muestra el resultado obtenido en la Figura 4.1.

Por último queda mostrar una tabla en la que tanto esta como el anterior diagrama reflejan en diferentes apartados temporales la cantidad de tiempo invertido en este trabajo. Para ser más exactos dicha tabla publica por estimación la cantidad de horas empleadas para cada tarea. (véase Cuadro 4.1).

Aquí finalizaría la planificación del desarrollo del proyecto, en la cual hay que tener en cuenta que la temporización de cada tarea puede variar con el tiempo que realmente ha sido necesario utilizar debido a posibles intentos de ampliación de alguna etapa.

Figura 4.1: Diagrama de Gantt

Fase	Descripción	Tiempo estimado
Fase 1	Estudio del estado del arte	40 horas
Fase 2	Familiarización con el chip ESP8266	30 horas
Fase 3	Conocimiento del protocolo MQTT	15 horas
Fase 4	Pruebas de programación con software Arduino	55 horas
Fase 5	Diseño e implementación del circuito final	65 horas
Fase 6	Fase de pruebas	15 horas
Fase 7	Elaboración de una memoria técnica del proyecto	80 horas

Cuadro 4.1: Estimación temporal del proyecto

4.2. Recursos empleados

En esta sección se realizará una clasificación de todos los recursos implicados, tanto humanos, como hardware, como software. Cada uno de ellos expuestos en una subsección diferente que procurará identificarlos, de manera que haga más factible poder realizar una estimación de costes posterior.

4.2.1. Recursos humanos

Aquí se muestra el personal que ha tenido participación en el proyecto:

- D. Jorge Navarro Ortiz, Profesor Contratado Doctor de la Universidad de Granada en el Departamento de Teoría de la Señal, Telemática y Comunicaciones, en calidad de tutor del proyecto.
- Dª. Sandra Sendra Compte, Profesora Ayudante Doctor de la Universidad de Granada en el Departamento de Teoría de la Señal, Telemática y Comunicaciones, en calidad de tutora del proyecto.
- Ramón Fernández Gualda, alumno del Grado de Ingeniería de Tecnologías de Telecomunicación y autor del presente proyecto.

4.2.2. Recursos hardware

A continuación se identifican los diferentes recursos hardware que se han empleado:

- Ordenador portátil Toshiba Satellite P50-B-118, con procesador Intel Core i7-4710HQ a 2,50GHz, con una memoria RAM DE 8GB y disco duro de 1TB de capacidad. Este PC será utilizado para subir los diferentes programas al chip ESP8266 creados y para la creación del servidor privado que necesita el protocolo tipo “Publicador - Suscriptor”.

- Línea de acceso a Internet, que será la que nos permita la conexión Wi-Fi entre los clientes (el dispositivo pasarela y el *smartphone*) con el *broker* que gestiona los mensajes principalmente.

- *Smartphone* iPhone 5S, con sistema operativo iOS 10.3.1. Dispositivo mediante el cual el usuario puede lanzar su petición o mensaje como cliente.

- *NodeMCU v1.0* con ESP8266, módulo *low-cost* que será el encargado tanto de enviar las señales IR a los diferentes equipos del hogar como de recibir para el aprendizaje de comandos.

- *Wemos D1 Mini* con ESP8266, otra alternativa de módulo *low-cost* al igual que el anterior, que permite la comunicación mediante infrarrojo y por Wi-Fi mediante su antena.

- Para la exposición de todos los componentes electrónicos utilizados en el proyecto se ha decidido representarlos en Cuadro 4.2.

Elemento	Cantidad
Placa *Protoboard*	1
Mini Placa *Protoboard* PB016	1
RTL2832U R820T2 DVB-T *Tuner Dongle*	1
Medidor de energía USB	1
TSOP4838	3
MB102 Regulador de tensión 5V - 3.3V	1
Cable 5V a fuente de alimentación	1
Cable micro USB	1
Cables para placa *Protoboard*	20
Transistor NPN 2N2222	2
LED luminoso	5
LED IR emisor	5

Cuadro 4.2: Lista de componentes que forman parte del proyecto

4.2.3. Recursos software

Como recursos software que se han utilizado destacan los siguientes:

- Sistema operativo *Windows 10 Home (64 bits)*, instalado en el ordenador portátil, donde se ha instalado el servidor de gestión de mensajes y programado los diferentes tareas para nuestro módulo.

- *Oracle VM VirtualBox*, haciendo uso de una máquina virtual con sistema operativo *Ubuntu 14.04 LTS (64 bits)* equipado con todo el software capaz de ejecutar las funciones sobre captación de señales IR del *RTL-SDR*.

- *Arduino*, lenguaje de programación basado en C que irá actualizando las diferentes funciones que se deseen aplicar sobre el *NodeMCU v1.0*.

- *Eclipse Mosquitto*, que es un software gratuito que nos permitirá la conexión por mensajes MQTT.

- *Mqtt*: Aplicación para el móvil que hará que este actúe como cliente MQTT.

- *GanttPro*: Programa gratuito dedicado exclusivamente para la elaboración de diagramas de Gantt que ha permitido la creación del mismo en este trabajo.

- *Fritzing*: Herramienta software de código abierto que ayuda a la realización de esquemas de circuitos relacionados con Arduino y diseñar el PCB final.

4.3. Estimación de costes

Ahora se dedicará un apartado a la estimación del gasto o coste necesario para llevar a cabo dicho proyecto, teniendo previamente en cuenta todos y cada uno de los tipos de recursos detallando uno por uno el precio de cada recurso.

Todos aquellos recursos software han sido en su totalidad gratuitos, es por ello que no se harán mención de ellos en el siguiente apartado 4.3 sobre la estimación de costes.

4.3.1. Recursos humanos

Al ser esto un proyecto, el cálculo del coste de los recursos humanos se valorará en función de la cantidad de horas invertidas en el mismo. Se ha creado a continuación una tabla como referencia a dichas horas empleadas en cada una de las secciones.

Cabe mencionar que a la hora de realizar una estimación se partirá desde las dos siguientes premisas:

- Un Ingeniero recién graduado en Tecnologías de la Telecomunicación se estima que obtiene un salario de 25€/hora.

- Un Profesor Doctor de la Universidad de Granada se estima que obtiene un salario de 50 € / hora. Teniendo en cuenta tanto las tutorías realizadas a lo largo del curso como la revisiones del trabajo, se considera que ha habido una inversión de 15 horas.

La realización del Cuadro 4.3 pretende mostrar a modo general la cantidad de horas que se han dedicado para cada una de las secciones que componen la planificación del proyecto con su respectivo coste, concluyendo así con el precio total estimado:

Fase	Descripción	Horas	Coste estimado
Fase 1	Estudio del estado del arte	40	1000€
Fase 2	Familiarización con el chip ESP8266	30	750€
Fase 3	Conocimiento del protocolo MQTT	15	375€
Fase 4	Pruebas de programación con Arduino	55	1375€
Fase 5	Implementación del circuito final	65	1625€
Fase 6	Fase de pruebas	15	375€
Fase 7	Elaboración de la memoria técnica	80	2000€
Tutorías		15	750€
TOTAL:			**8250€**

Cuadro 4.3: Estimación del coste de los recursos humanos

Para una visualización más gráfica se representan los anteriores datos en la siguiente Figura 4.2.

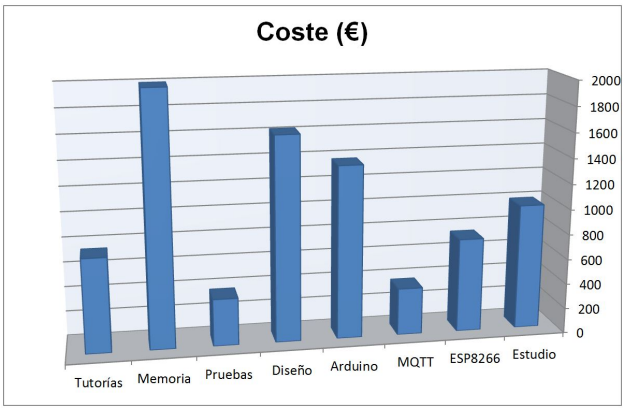

Figura 4.2: Gráfico del coste de los recursos humanos según fase

4.3.2. Recursos hardware

A continuación se mostrará un desglose del material físico empleado, recogido en el Cuadro 4.4.

Elemento	€/Unid.	Unids.	Coste
Portátil Toshiba Satellite P50-B-118	143,17	1	143,17€
NodeMCU v1.0	2,5	1	2,5€
Wemos D1 Mini	2,5	1	2,5€
Smartphone iPhone 5S	47,5	1	47,5€
Línea de acceso a Internet	30	10 meses	300€
Placa *Protoboard*	5	1	5€
Mini Placa *Protoboard* PB016	2,8	2	5,6€
RTL2832U R820T2 DVB-T *Tuner Dongle*	6,6	1	6,6€
Medidor de energía USB	3	1	3€
TSOP4838	1	3	1€
MB102 Regulador de tensión 5V - 3.3V	7	1	7€
Cable 5V a fuente de alimentación	8	1	8€
Cable micro USB	1,5	2	3€
Cables para placa *Protoboard*	0,1	20	2€
Transistor NPN 2N2222	0,1	2	0,2€
LED luminoso	0,05	5	0,25€
LED IR emisor	0,2	5	1€
TOTAL:	**538,82€**		

Cuadro 4.4: Lista de componentes que forman parte del proyecto

Exclusivamente para el portátil Toshiba y el terminal iPhone 5S se ha considerado que su tiempo de vida es de 3 años, cuyos precios totales suponen 859€ y 285€ respectivamente, pero el tiempo de vida útil es de 6 meses, que es el tiempo que se tendrá en cuenta a la hora del presupuesto y de ahí su precio menor en el Cuadro 4.4.

4.4. Presupuesto final

Una vez se tiene identificados todos los tipos de recursos que se van a utilizar y declarado los costos de los mismos, en el Cuadro 4.5 se va a recoger a modo global el presupuesto total estimado de este Trabajo Fin de Grado como resultado de la suma de esos recursos.

En resumen, el precio final estimado de este Trabajo Fin de Grado es de 8788,82€.

Como puede apreciarse en el Cuadro 4.5 la mayoría de los costes recaen en los recursos humanos. Esto demuestra que no hace falta disponer de una

Recursos	Coste
Recursos humanos	8250€
Recursos hardware	538,82€
TOTAL:	**8788,82€**

Cuadro 4.5: Presupuesto estimado final

gran suma de dinero para el hadware sino que se requiere de las capacidades personales que permitan el desarrollo de este tipo de estudios.

Cabe mencionar que la estimación temporal no es exacta, sino que es una aproximación a la planificación seguida que ha permitido la correcta realización del trabajo.

Capítulo 5

Tecnologías implicadas

Tanto las herramientas hardware y software, como los protocolos involucrados van a ser expuestos en este capítulo a fin de dar a conocer su comportamiento mediante un desarrollo teórico.

Para la comprensión del proyecto, es muy importante enfatizar sobre el conocimiento de la tecnología de comunicación infrarroja, del cual primeramente se hará un importante estudio que ayudará a su comprensión antes de pasar al diseño e implementación.

Como herramienta hardware principal, se enfocará el discurso en la exposición del chip que se va a emplear, llamado ESP8266, el cual va a hacer posible la realización de las diferentes pruebas que más adelante se detallarán.

Se especificarán las funciones de las herramientas software, fundamentales para agregar las tareas que se pretenden llevar a cabo. Entre ellas destacamos el software Arduino, y por otro lado tenemos Mosquitto, para la creación del servidor MQTT.

5.1. Comunicaciones Infrarrojas

Toda onda electromagnética se caracteriza entre otras cosas por su frecuencia o longitud de onda. Es por ello que se desarrolló lo que se denomina como espectro electromagnético, donde se encuentra situado el infrarrojo en un rango de longitudes de onda de 800nm a 1000μm), y que se representa en la Figura 5.1.

Los infrarrojos[8] se dividen en:
- Infrarrojo cercano (800nm a 2500nm).
- Infrarrojo medio (de 2.5μm a 50μm).
- Infrarrojo lejano (de 50μm a 1000μm).

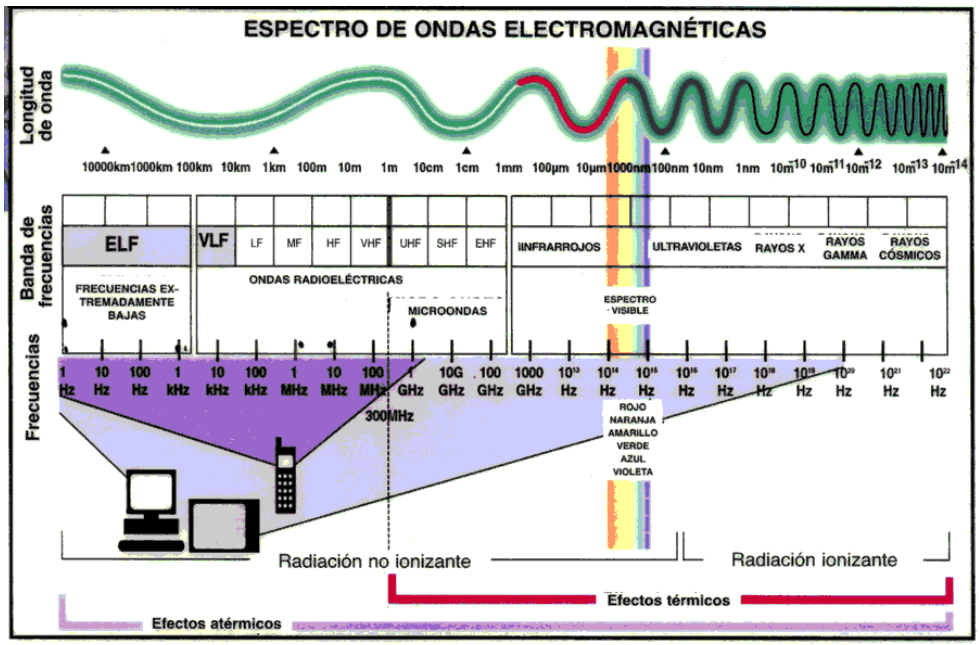

Figura 5.1: Esquema del espectro de ondas electromagnéticas

Concretamente, la longitud de onda de los rayos infrarrojos es de 850 – 900 nm, la cual utilizamos para la transmisión y que como ventaja respecto a las de radio es que transmiten a mayor frecuencia, por lo que su espectro no está tan limitado y se puede obviar la restricción de su ancho de banda a falta de regulación en ese espectro.

Aunque hay diversos modos [12] en la transmisión de estas señales, tales como el punto a punto, casi-difuso y el modo difuso, aquel que se usaría para este proyecto sería el modo punto a punto ya que en el casi-difuso, la forma de emisión es radial, que quiere decir en todas las direcciones, esto se puede conseguir transmitiendo hacia superficies reflectantes, y tampoco es de interés el modo difuso al ser igual e cuanto a la forma de emisión pero permitiendo múltiples reflexiones. El modo punto a punto se enfoca a una sola dirección, por lo que tanto emisor como receptor deben posicionarse enfrentados y cuya focalización en solo una pequeña región del espacio permite una distancia superior para una potencia dada respecto de los demás modos.

Es amplia la cantidad de ventajas que ofrece el uso de esta tecnología: Se puede hablar de su bajo costo en la mayoría de controles remotos que hay en el mercado; su implementación no requiere apenas lugar y es simple, no tiene problemas de interferencias electromagnéticas, la alimentación que requiere precisa de poca potencia, posee una gran variedad de estándares

a utilizar fácilmente implementables, no produce ningún efecto negativo en los ojos y facilita el uso de muchos equipos.

La luz infrarroja además contiene una característica, y es que, cualquier ser vivo cuya temperatura sea superior a 0 Kelvin (o -273,15 grados Celsius), es capaz de emitir radiación infrarroja. Como bien explica la ley de Wien[10], existe una relación inversa entre la longitud de onda en la que se produce el pico de emisión de un cuerpo negro y su temperatura, encontrando en los objetos a temperaturas normales un máximo de emisión en el infrarrojo. Para aplicaciones como los equipos de visión nocturna se puede aprovechar para la visualización de objetos cuando la cantidad de luz es pobre, consiguiendo que aquellos cuerpos con mayor temperatura sean los más luminosos. También se utiliza en controles remoto esta tecnología, que es el campo en el que se quiere profundizar en este proyecto, ya que hay que tener en cuenta que los sistemas infrarrojos de comunicaciones son inmunes a interferencias y ruido de tipo radioeléctrico.

Para el caso de las aplicaciones como los mandos a distancia (o telecomandos) [9], esa emisión IR de los cuerpos representa un problema, por lo que para que esa radiación no interfiera con el mando a distancia IR, la señal se modula con una portadora. Conseguir esto no es difícil, ya que solo se debe de enviar la portadora en forma de tren de ondas estable junto con la información que se desea transmitir. La mayoría de las señales radioeléctricas enviadas a través del aire utilizan este principio como las ondas radioeléctricas cuya longitud de onda es más larga que la infrarroja.

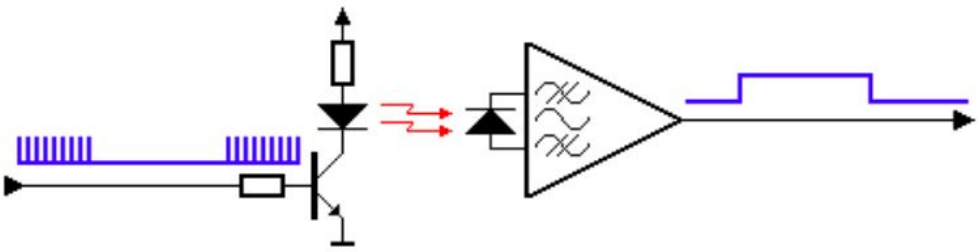

Figura 5.2: Modulación de una señal IR

Como nos muestra la Figura 5.2, a la izquierda aparece un emisor que se encarga de emitir la señal infrarroja mediante 1 diodo de infrarrojo, cuya luz no podrá visualizarse. Hay que tener en cuenta que la modulación de los diodos IR comprende un rango desde algunos pares de decenas de KHz

hasta los 100MHz, cuya transmisión se basa en la numeración binaria. Y a la derecha se encuentra el receptor que demodula la señal correcta. Generalmente esa numeración binaria se distingue en marcas y espacios. Los espacios serían la señal que hay por defecto o nulo envío en el que el estado del transmisor es apagado, de manera que no se envía luz en este momento. En la zona de las marcas es en la cual se envía ese tren de impulsos del que se se hacía mención antes a una determinada frecuencia (que normalmente las frecuencias de la portada abarcan entre 30kHz y 60kHz). La frecuencia más utilizada es la de 38kHz. En el lado del receptor los espacios representan un flanco en alto mientras que las marcas serían tomadas en cuenta como un flanco en bajo.

La correcta recepción será posible siempre y cuando anteriormente el dispositivo permanezca a la espera, luego hará uso de un leve consumo de energía para su posterior captación.

Por supuesto mencionar que el pequeño radio de emisión que ofrece la comunicación infrarroja es un pequeño factor de seguridad en contra de la captación malintencionada de información, teniendo en cuenta además que no es capaz de atravesar paredes, de manera que se puede ahorrar la implementación de laboriosos algoritmos de cifrado utilizados en sistemas de RF.

Control remoto

Se habla de control remoto a aquel dispositivo electrónico cuya función consiste en la capacidad de permitirnos realizar operaciones remotas sobre un aparato electrónico que hará de receptor (véase Figura 5.3).

Hoy en día dicho artilugio posee una gran popularidad en el mercado, al haberse implantado en la mayoría de aparatos electrónicos del hogar como son los televisores, DVDs, aires acondicionados, ordenadores, garajes, equipos de música, etc. Cabe decir que se podría decir que casi todos los mandos a distancia hacen uso de la tecnología infrarroja (IR) para establecer dicha comunicación. Normalmente estos controles están equipados con un pequeño diodo que emite señales de tipo infrarrojo cercano el cual no es visible para el ojo humano, pero que el receptor si es capaz de detectar. Este diodo está caracterizado por su cavidad de cerámica, la cual hay que hacer un breve comentario sobre su resistencia al medio, ya que este tipo de material es más resistente a daños físicos y por tanto, dota al control remoto de un extra añadido.

La distancia de emisión está relacionada con la cantidad de corriente que se administra al emisor. Las distancias más pequeñas se conseguirán con la mínima corriente (unos 100mA), mientras que su alcance se verá en aumento si llegamos a alcanzar hasta el Amperio. Es por ello que a la hora de elegir

la configuración, se deberá decantar por una relación distancia-energía de acuerdo a las exigencias del sistema. Otro factor a tener en cuenta es el uso de baterías o pilas, ya que a mayor potencia, menor tiempo de vida útil suministrarán. Dicho LED posee como norma una determinada cantidad de disipación de potencia media la cual si se supera deja de conducir. Gracias a que los impulsos que conducen los LED son muy cortos, el diodo puede conducir hasta esas cantidades de corriente de manera que no se compromete el funcionamiento del mismo. El valor que no debe de exceder bajo ningún criterio es el valor de pico máximo de corriente. Además el LED IR para que conduzca, debe de tener una caída de voltaje nominal de aproximadamente 1.1V.

El funcionamiento del mismo es más que simple. Contiene un circuito en el que con la presión de un botón, este se cierra, se analiza el botón pulsado, se envía al diodo la señal solicitada y esta la envía a una determinada frecuencia. Si el dispositivo que está a la espera de la función se posiciona en la misma frecuencia será capaz de entender la orden emitida.

Motivos por lo que se ha implantado la luz infrarroja en controles remotos se debe a:

- Su frecuencia no tiene secuelas en los tejidos vivos (tiene menos impacto que la luz visible).
- Es invisible, luego no percibimos su uso visualmente (su longitud de onda se encuentra fuera del espectro visible).
- Tiene poco alcance, aunque los dispositivos en los que habitualmente se usa se encuentran muy cerca (unos pocos metros).

Receptores infrarrojos

Para los receptores de infrarrojos [11] codificados incluyen el elemento que es sensible al infrarrojo, como una lente, o filtro de espectro y la lógica que captura las señales a una determinada frecuencia.

En la Figura 5.4 que representa el diagrama de bloques, este se compone de, empezando por la izquierda, del diodo que capturará la señal IR, después el circuito AGC la limita obteniendo así un nivel de pulso constante (independientemente de la distancia al control remoto), luego se envía al filtro paso que se sintoniza a la frecuencia portadora del mando a distancia, para por último pasar al decodificador. Este procura detectar la presencia de la portadora de manera que si se encuentra se enviará un flanco en bajo.

El ejemplo anterior forma parte de la estructura del diodo receptor TSOP4838, cuya conexión se explica en la figura de abajo y cuya conexiones son tres: la primera de las patillas va conectada a la señal de salida, la segunda a tierra, y la tercera al voltaje de alimentación, que puede ser hasta de 5.5 V (visto desde la Figura 5.6).

Figura 5.3: Control remoto

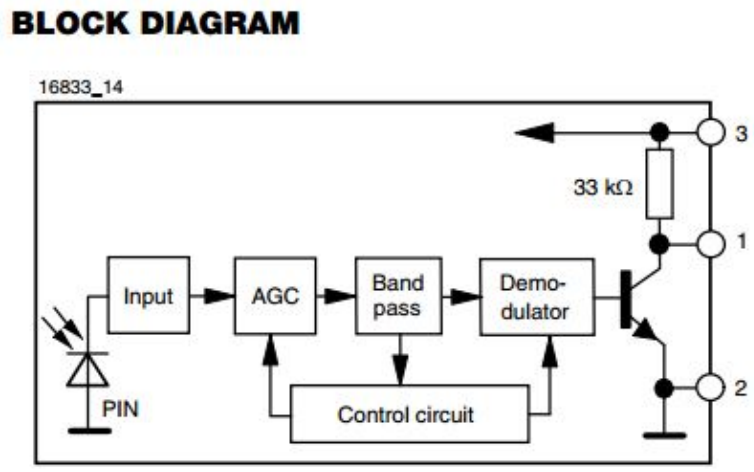

Figura 5.4: Diagrama de bloque de un receptor infrarrojo

TSOP4838 es un diodo receptor IR a muy bajo precio que físicamente se puede asociar como aparece en la Figura 5.6

Dicho modelo proviene de la marca Vishay. Esta empresa, Siemens y Telefunken son los principales proveedores en Europa. Siemens comercializa la serie SFH506-xx, donde xx representa la frecuencia portadora (30, 33,

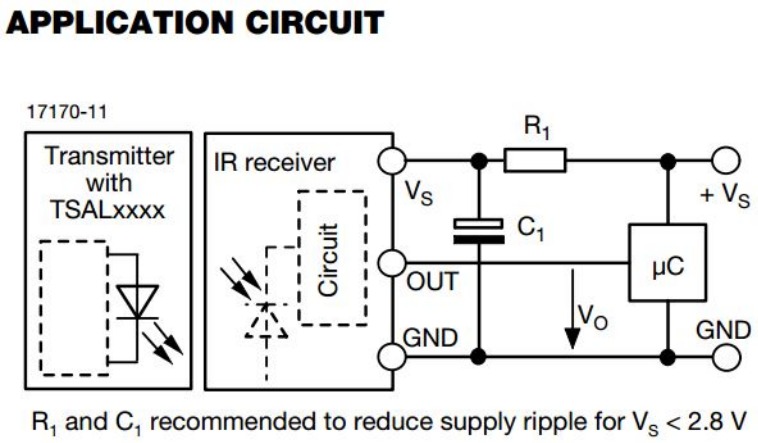

Figura 5.5: Circuito de aplicación

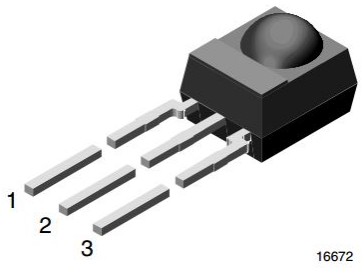

Figura 5.6: Receptor infrarrojo TSOP4838

36, 40 o 56kHz), mientras que Vishay tiene la serie de productos Vishay TSOP12xx, TSOP48xx y TSOP62xx.

Es posible el aumento de la potencia de la señal transmitida si el circuito de emisión es más potente, obteniendo así mayores distancias en el radio que cubre la señal infrarroja, y por supuesto, y al ser este el caso con emisores y detectores direccionales, mayor aún si tanto el emisor como receptor están correctamente direccionados con el mejor ángulo de visión del receptor.

Debido a la cantidad de normas de emisión de IR para mandos a distancia, cada compañía decidió crear su propio protocolo. Es por ello que hoy en día se dispone de una enorme cantidad de protocolos de infrarrojos, casi por marca.

5.2. Protocolos de infrarrojos

Este apartado pretende hacer una breve descripción de los protocolos más importantes que actualmente se encuentran en el mercado explicando

sus características principales.

Protocolo RC-5

Es un protocolo desarrollado por Philips [13], ampliamente utilizado, aunque más tarde se desarrolló el protocolo sucesor RC-6. Las características principales son las que aparecen en el Cuadro 5.1.

Longitud de bits de dirección	5
Longitud de bits de comando	6
Longitud de bits de comando para RC5X	7
Tiempo de bit constante	1.778ms (64 ciclos)
Tipo de modulación	Codificación *Bi-phase*
Frecuencia portadora	36kHz

Cuadro 5.1: Características principales del protocolo de RC5

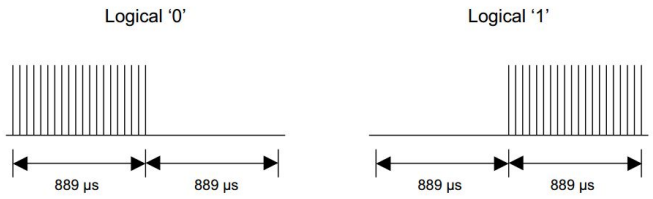

Figura 5.7: Modulación en el protocolo RC5

La modulación bifásica también es denominada como codificación Manchester (véase Figura 5.7). Como aparece en la tabla, el tiempo de bit es constante siendo de 1.778ms, con la mitad del tiempo compuesto por una ráfaga de la portadora de 36kHz y la otra mitad compuesto por un espacio. El 0 lógico situaría la ráfaga en la primera mitad del tiempo del bit, y el 1 lógico haría el método inverso (en la segunda parte). El ciclo de trabajo de la portadora para este protocolo sería de 1/3 o 1/4.

La estructura del mensaje que sigue dicho protocolo sería tal cual indica la Figura 5.8:

La parte inicial está compuesta por dos impulsos que son dos 1 lógicos. En el caso del protocolo extendido de RC5 sólo se utiliza un bit de inicio. El segundo bit de inicio formaría a pasar parte del campo de comando siendo un total de 7 bits. Cabe mencionar que el valor de este segundo campo convertido al campo de comando debe de estar invertido.

El tercer bit que se observa en la gráfica anterior es de conmutación. Cada vez que se suelta el botón, se invierte. Así es como los receptores detectan si la pulsación es continua.

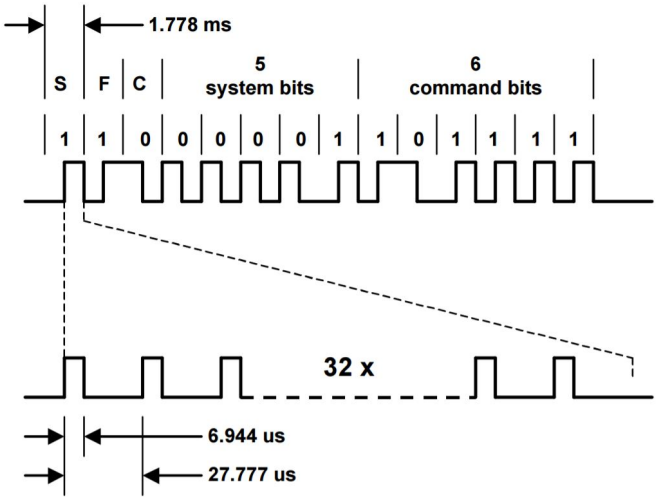

Figura 5.8: Forma de onda de la salida en el protocolo RC5

Los siguientes bits son del campo de dirección y son 5, y se ordenan empezando por el MSB, y después se envía el campo de comando con la misma ordenación, pero de 6 bits. El mensaje total ocupará 14 bits, con una duración total de 25ms.

Si la tecla se mantiene pulsada, el sistema enviará el mensaje repetido cada 114ms. El bit de conmutación mantendrá el mismo número lógico tantas veces como sea repetido.

Protocolo SIRC

Para este protocolo [14], creado por SONY, existen tres posibles versiones: la primera es de 12 bits, la segunda de 15, y la tercera de 20 bits en las que en todas se guardan 7 bits para el campo de comando. Las versiones se diferencian en el tamaño de bits del campo dirección. Para la versión de 12 bits hay 5 bits de dirección, para la de 15 bits hay 8, y por último la de 20 bits, que tiene 7 bits de comando y los 8 bits restantes son bits extendidos (véase Cuadro 5.2).

En este caso, un 1 lógico representa una ráfaga de 1.2ms mientras que para el 0 lógico dura 0.6ms, siempre separadas en los dos casos por intervalos de 0.6ms (véase Figura 5.9) . El ciclo de trabajo de la portadora recomendado es 1/4 o 1/3.

Una vez definida la modulación, se pasa a conocer la estructura del mensaje en SIRC en la Figura 5.10.

En este ejemplo ilustrado se muestra la versión de 12 bits, que empieza por un pulso de inicio que dura 2.4ms con su respectivo espacio de 0.6ms

Longitud de bits de dirección (versión 12 bits)	5
Longitud de bits de comando (versión 12 bits)	7
Longitud de bits de dirección (versión 15 bits)	8
Longitud de bits de comando (versión 15 bits)	7
Longitud de bits de dirección (versión 20 bits)	5
Longitud de bits de comando (versión 20 bits)	7
Longitud de extensión (versión 20 bits)	8
Intervalo de bit	1.8ms (1) o 1.2ms (0)
Tipo de modulación	Con intervalo de pulso
Frecuencia portadora	40kHz

Cuadro 5.2: Características principales del protocolo SIRC

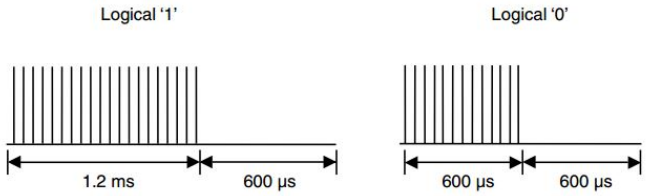

Figura 5.9: Modulación en el protocolo SIRC

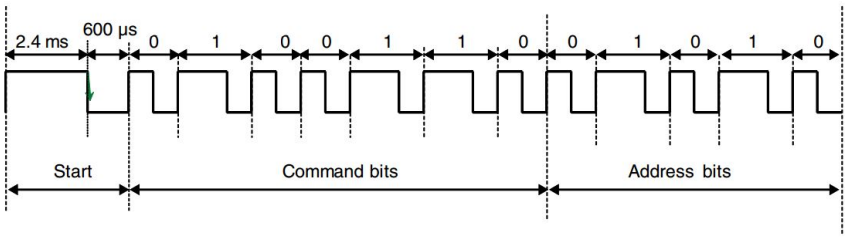

Figura 5.10: Forma de onda de la salida en el protocolo SIRC

que ajusta la ganancia del receptor IR. Después se transmite el comando de 7 bits, y por último la dirección de 5 bits. Tanto comando como dirección se enviarán empezando por el LSB.

Para una pulsación continua del botón del mando, los comandos se repetirán cada 45ms.

Protocolo JVC

La marca JVC tiene su propio protocolo [15]que usa para la mayoría de sus dispositivos.

Las características principales serán expuestas en el Cuadro 5.3.

Longitud de bits de dirección	8
Longitud de bits de comando	8
Tipo de modulación	Con intervalo del pulso
Frecuencia portadora	37.9KHz
Intervalo de bit	1.055ms (0) o 2.11ms (1)

Cuadro 5.3: Características principales del protocolo JVC

Gracias al intervalo del ancho del pulso, el sistema reconoce esos pulsos como un "0" lógico o como un "1" lógico. Cada pulso es de unos 0.527ms, que en una frecuencia de portadora de 38KHz equivaldría a unos 20 ciclos, por lo tanto el reconocimiento de un "1" lógico vendría dado por la detección de un pulso de 0.527ms en un intervalo de 2.11ms (80 ciclos) y un "0" lógico sería identificado por un pulso de 0.527ms en un intervalo de 1.05ms (40 ciclos). El ciclo de trabajo de la portadora que se recomienda es de 1/3. A continuación se muestra una representación gráfica en la Figura 5.11.

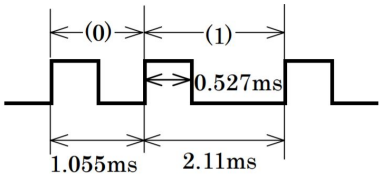

Figura 5.11: Ancho del pulso e intervalo del pulso en JVC

Una vez explicada su modulación, han de precisarse las características oportunas de un mensaje de este protocolo en la Figura 5.12

Figura 5.12: Forma de onda de la salida en JVC

Dicho mensaje está compuesto de la siguiente manera como aparece en la imagen de arriba. El término "LSB" hace referencia al bit menos significativo, mientras que "MSB" representará el bit más significativo. Que aparezca primeramente "LSB" (a la izquierda), antes que "MSB" nos advierte de que será el primer valor a transmitir. No hay que olvidarse de que el inicio de la trama está marcado por una ráfaga AGC de 8.4ms (320 ciclos) y un espacio

de 4.2ms (160 ciclos), seguidos de la dirección y el comando. El tiempo total de la transmisión es variable porque los tiempos de bits son variables.

Por último cabe mencionar que este protocolo realiza una serie de repeticiones cada 50 - 60ms mientras que el botón se mantiene pulsado, pero las repeticiones no incluyen la cabecera de inicio de 8.4ms ni el espacio adyacente de 2.4ms. Es así como el receptor entiende si es una tecla pulsada por primera vez o está pulsada.

Protocolo NEC

NEC Electronics, empresa a la que se le atribuye la creación de este protocolo, actualmente es conocida como Renesas. El protocolo NEC[16] cumple las características mostradas en el Cuadro 5.4.

Longitud de bits de dirección	8
Longitud de bits de comando	8
Modo extendido disponible	8 bits de dirección adicionales
Tipo de modulación	Con intervalo del pulso
Frecuencia portadora	37.5kHz
Intervalo de bit	1.125ms (0) o 2.25ms (1)

Cuadro 5.4: Características principales del protocolo NEC

Los datos son decodificados gracias a la "distancia de pulso", siendo cada pulso de 560μs en una frecuencia de portadora de 38kHz (aproximadamente 21 ciclos). Un "1" lógico tarda 2.25ms en transmitir un pulso, mientras que el "0" lógico sólo necesita 1.125ms. Al igual que el anterior protocolo, el ciclo de trabajo del portador recomendado es de 1/4 o 1/3. En la Figura 5.13 se aprecia lo anterior más claramente.

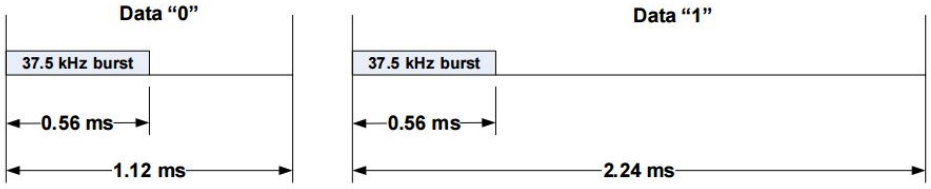

Figura 5.13: Ancho del pulso e intervalo del pulso en NEC

Una vez entendido, lo anterior, pasa a explicarse la estructura que tendría un mensaje de dicho protocolo.

La Figura 5.17 muestra dicha estructura. El mensaje empieza por una ráfaga AGC de 9ms que permite que los receptores ajusten la ganancia, la cual es seguida de un espacio de 4,5ms, para después incluir la dirección

Figura 5.14: Forma de onda de la salida en NEC

y el comando. Hay que tener en cuenta el modo de envío de la dirección y comando, puesto que ambas se transmiten dos veces, siendo la segunda vez invertida en cada uno de los bits. Esto permite verificar que el mensaje enviado es correcto. El tiempo total de transmisión es constante porque cada bit se repite con su longitud invertida. En caso de no utilizar la duplicación con los valores invertidos, se puede ampliar la dirección y el comando a 16 bits cada uno.

Acerca de la pulsación continua de un comando, dicha orden solo se transmite una vez. Siempre que la pulsación dure más de 108ms se transmitirá un código de repetición que a diferencia del comando anteriormente explicado será un pulso AGC de 9ms acompañado de un espacio de 2,24ms y una ráfaga de 560μs. Las dos figuras (Figura 5.15 y 5.16) lo muestran gráficamente.

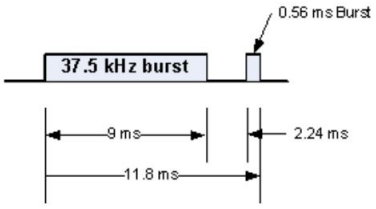

Figura 5.15: Formato de transmisión de repetición en NEC

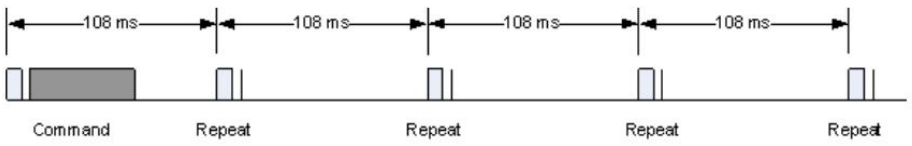

Figura 5.16: Tiempos de repetición en NEC

Debido a la popularidad alcanzada por el protocolo las direcciones fueron agotadas, por lo que sacrificando los bits de verificación se extendió habilitando de 256 valores a 65000 diferentes. El rango de direcciones se extiende de 8 bits a 16 bits mientras que el resto de la forma de la estructura del protocolo se mantiene.

Con esta ampliación el tiempo total del mensaje no sería constante. Dependería del número total de 1 y 0 que se encuentran en el mensaje. Para que el tiempo se mantenga constante, el número 1 en el campo de dirección debe de ser 8 (siendo el número 0 también 8). Esto reduce el número máximo de direcciones diferentes a aproximadamente 13000.

La redundancia del comando se conserva, siendo 256 los comandos permitidos por dirección (Figura 5.17).

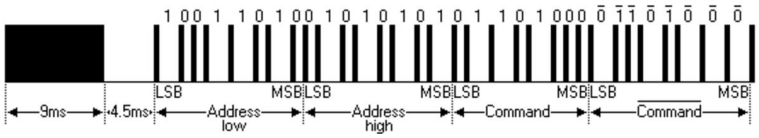

Figura 5.17: Formato de onda de la salida en NEC extendido

Siempre que el byte bajo sea el inverso exacto del byte en alto no será una dirección extendida válida, por lo que todos los valores de direcciones en el protocolo extendido no son posibles usarlos (256 valores no válidos).

Protocolo Nokia NRC17

El protocolo Nokia Remote Control[17] utiliza 17 bits para transmitir los comandos IR, lo que da a entender el nombre del protocolo. Fue diseñado para la electrónica de consumo de Nokia como televisores o vídeo e incluso otras marcas como Finlux y Salora hicieron uso de él. Algunas características del mismo se muestran en el Cuadro 5.5.

Longitud de bits de dirección	4
Longitud de bits de comando	8
Longitud de bits de subcódigo	4
Tiempo de bit constante	1ms
Tipo de modulación	Codificación *Bi-phase*
Frecuencia portadora	38kHz

Cuadro 5.5: Características principales del protocolo NRC17

La modulación en este protocolo es bifásica, (o NRZ - No Retorno a Cero) con una frecuencia portadora de 38kHz. Como aparece en la Figura 5.18, los bits tienen un tiempo constante de 1ms, estando la mitad del bit compuesto por una ráfaga y la otra mitad vacía. Un 1 lógico sería una ráfaga en la primera mitad del bit, mientras que el 0 lógico sería el inverso (ráfaga solo en la segunda mitad del bit). La relación pulso/pausa de la frecuencia portadora de 38kHz es de 1/4, reduciendo el consumo de la energía.

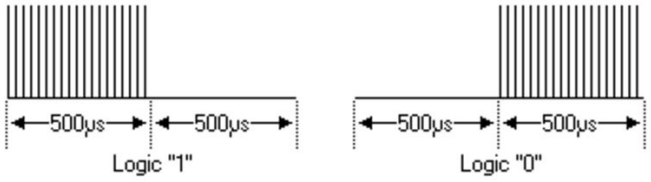

Figura 5.18: Modulación en el protocolo NRC17

La Figura 5.19 explica perfectamente la estructura que sigue dicho protocolo. La primera parte está compuesta por un pre-pulso formada por una ráfaga de 0.5m y una pausa de 2,5ms que suman un total de 3ms. Lo siguiente será el bit de inicio compuesto por un 1 lógico que sirve para que el receptor sincronice el tiempo. Los próximos 8 bits serían el comando empezando con LSB, más tarde la dirección de 4 bits y el subcódigo, usado también en ocasiones como extensión de la dirección con sus 4 bits. La suma total del tiempo del mensaje es de 20ms.

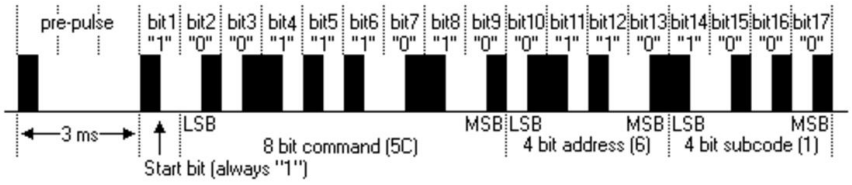

Figura 5.19: Formato de onda de salida en NRC17

Cuando se pulsa una tecla, se envía un mensaje de inicio, pero no será hasta 40ms más tarde cuando realmente se envíe el mensaje. En caso de que se mantenga pulsado, este se repite cada 100ms, y una vez soltada, se envñia un mensaje de parada. Como el receptor ve los mensajes de inicio y de parada, toma esa secuencia como una sola, eliminando así errores de repetición (véase Figura 5.20).

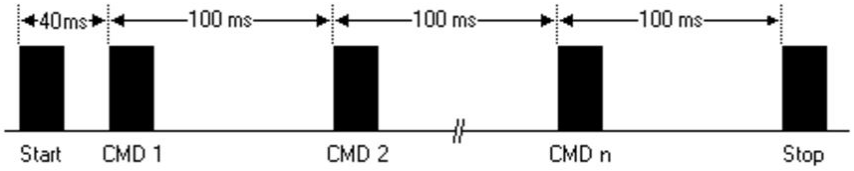

Figura 5.20: Caso de repetición de comando en NRC17

Protocolo Sharp

Sharp utiliza también su protocolo propietario[18] en algunos dispositivos como VCRs o cadenas de música entre otros. A continuación, las características aparecen en el Cuadro 5.6.

Longitud de bits de dirección	5
Longitud de bits de comando	8
Intervalo de bit	2ms (1) o 1ms (0)
Tipo de modulación	Con intervalo de pulso
Frecuencia portadora	38kHz

Cuadro 5.6: Características principales del protocolo de Sharp

Para la modulación, las ráfagas de un pulso duran 320μs, que a una frecuencia de 38kHz suponen unos 12 ciclos aproximadamente (véase Figura 5.21). Un 1 lógico tarda 2ms en transmitirse, mientras que el 0 lógico tarda 1ms. El ciclo de trabajo de la portadora es 1/4 o 1/3.

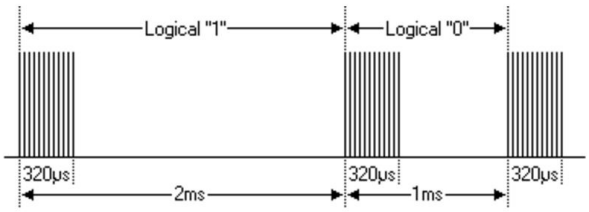

Figura 5.21: Modulación en el protocolo Sharp

A continuación se muestra la estructura de un mensaje de este protocolo en la Figura 5.22. Como se aprecia, en la primera parte del mensaje se envía la dirección empezando por el LSB, hasta 5 bits, después el comando de 8 bits comenzado por el LSB, finalizando con el bit de "Expansión" y el de "*Check*".

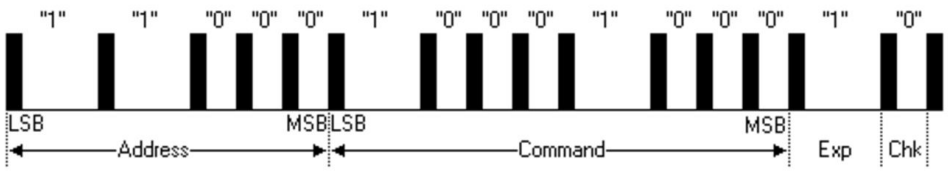

Figura 5.22: Forma de onda de la salida en el protocolo Sharp

Una secuencia completa no está formada solo por el mensaje anterior sino por dos mensajes (Figura 5.23). El primero que se transmite es exactamente

el que se ha descrito anteriormente, y el segundo se envía con un retardo de 40ms más tarde, que contiene la misma información con un pequeño cambio. Dicho cambio consiste en que los bits del mensaje, excepto los del campo de dirección, están invertidos. Esto sirve como comprobación de un envío satisfactorio del mensaje.

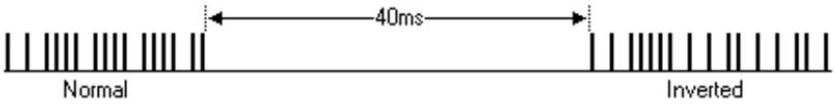

Figura 5.23: Repetición de envío del comando en el protocolo Sharp

Protocolo RC-MM

Dicho protocolo[19] también ha sido definido por Philips como un protocolo multimedia de infrarrojos, cuyas características vienen recogidas en el Cuadro 5.7.

Longitud de bits del mensaje	12 o 24
Tiempo de repetición	27.778ms (36 mensajes/seg)
Rango de tiempo de mensajes	3.5ms a 6.5ms
Tipo de modulación	Con intervalo de pulso
Frecuencia portadora	36kHz

Cuadro 5.7: Características principales del protocolo de RC-MM

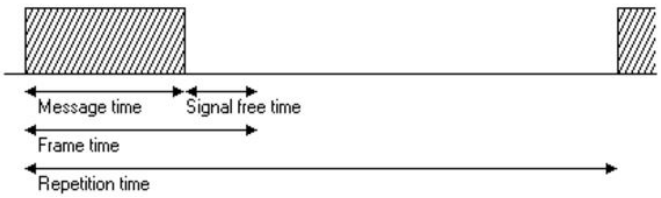

Figura 5.24: Forma de onda de la salida en el protocolo RC-MM

En la Figura 5.24 se muestran los tiempos de transmisión. El tiempo de mensaje es el que tiempo que tarda en transmitirse el mensaje. Ese tiempo puede ser de 3.5ms a 6.5ms, según los datos introducidos y el protocolo. El tiempo libre de señal es aquel tiempo que no se envía ninguna señal para evitar confusiones. Philips recomienda que ese tiempo sea de 1ms, pero si se utilizan cerca otras señales del protocolo RC-5 o RC-6 es aconsejable ampliarlo hasta 3,36ms. El tiempo de trama sería la suma de los dos anteriores,

llegando a sumar aproximadamente 10ms por trama. El tiempo de repetición recomendado es de 27.778, lo que alcanza hasta a 36 mensajes por segundo.

El ciclo de trabajo de la portadora es de 1/3 o 1/4, y cada mensaje es precedido de un impulso inicial de 416.7μs seguido de un espacio de 277.8μs. Después es seguido por 12 o 24 bits de datos. Los tiempos de codificación vienen recogidos en la Figura 5.25.

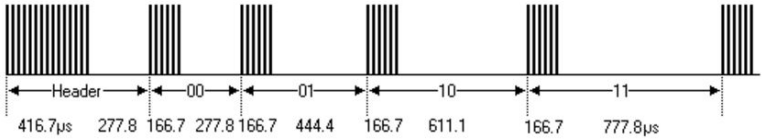

Figura 5.25: Modulación en el protocolo RC-MM

Protocolo RCA

Dicho protocolo[20] es parecido a NEC, implantado en mandos infrarrojos como el de XBOX. Sus características aparecen en el Cuadro 5.8.

Longitud de bits del mensaje	12
Longitud de bits de comando	8
Longitud de bits de dirección	4
Tiempo de bit	2.5ms (1) o 1.5ms (0)
Tipo de modulación	Con intervalo de pulso
Frecuencia portadora	56kHz

Cuadro 5.8: Características principales del protocolo de RCA

Para la codificación, un 1 lógico está formado por una ráfaga de 500μ al principio (unos 28 ciclos) mientras que el 0 lógico la ráfaga dura la misma cantidad de tiempo, pero en un intervalo de tiempo más corto (en 1.5ms) (véase Figura 5.26).

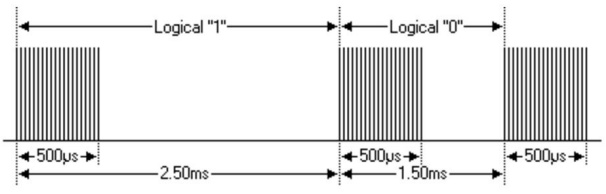

Figura 5.26: Modulación en el protocolo RCA

La estructura de un mensaje de este protocolo se compone como indica la Figura 5.27). Este sería la típica estructura de un mensaje de este tipo,

que como bien se aprecia, los diferentes campos se empezará con el envío del bit MSB. El inicio está compuesto por la ráfaga AGC de 4ms, seguido del campo dirección y comando. Cabe resaltar que estos dos campos se envían dos veces, y que al igual que con el protocolo NEC, la segunda vez será trasmitida de manera invertida. Esta parte repetida puede ser ignorada si no se desea chequear.

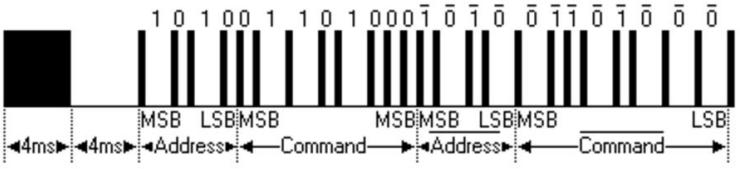

Figura 5.27: Forma de onda de la salida en el protocolo RCA

La repetición de un mismo comando al estar pulsado supone que se transmita cada 64ms.

Protocolo X-Sat Mitsubishi

A continuación, en el Cuadro 5.9, se exponen las características principales de este protocolo[21].

Longitud de bits de comando	8
Longitud de bits de dirección	8
Tiempo de bit	2ms (1) o 1ms (0)
Tipo de modulación	Con intervalo de pulso
Frecuencia portadora	38kHz

Cuadro 5.9: Características principales del protocolo X-Sat Mitsubishi

Cada ráfaga de la portadora de 38kHz dura 526μs (aproximadamente 20 ciclos), que se transforma en un 1 lógico si se transmite en un intervalo de 2ms o se transmite en un 0 lógico si ese intervalo es de 1ms. El ciclo de trabajo de la portadora es de 1/4 a 1/3 (en Figura 5.28).

El mensaje típico en este protocolo está formada por la estructura deglosada en la Figura 5.29. El envío de los bits se realiza empezando por el bit LSB. El mensaje de inicio es una ráfaga AGC de 8ms, para que los receptores ajusten la ganancia. Después continua con un espacio de 4ms, seguido de el campo dirección y el comando. El tiempo total del mensaje es variable, esto se debe a que los tiempos de bits son variables.

Por último, la repetición de un comando hace que esté se envíe cada 60ms como muestra la Figura 5.30.

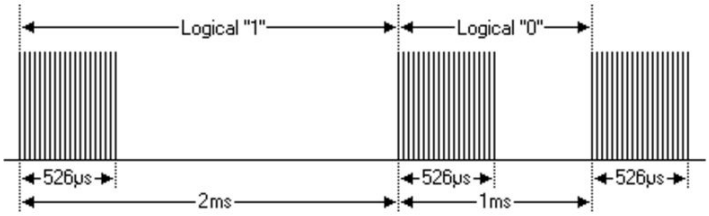

Figura 5.28: Modulación en el protocolo X-Sat Mitsubishi

Figura 5.29: Forma de onda de la salida en el protocolo X-Sat Mitsubishi

Figura 5.30: Repetición de envío del comando en el protocolo X-Sat Mitsubishi

5.3. ESP8266

Este chip (Figura 5.31) va a ser clave para la solución de este proyecto como herramienta hardware al ser un requisito principal, y el exigido por el tutor. Es realmente llamativo por su gran variedad de aplicaciones con un tamaño tan pequeño a un bajo costo.

ESP8266[22], diseñado desde un principio para el IoT (*Internet of Things*), hace uso de la tecnología Wi-Fi mediante su antena permitiendo a otros microcontroladores una conexión inalámbrica dentro del hogar. Aunque se diseñó hace unos años, se ha ido haciendo eco de él más tarde debido a la falta de documentación en inglés.

La alimentación que exige es de 3.3V de alguna fuente. Administrarle un voltaje de 5V puede significar la pérdida del chip. El pin "RST" debe de conectarse a 3.3V dependiendo de la versión del *firmware*.

Las características principales de este dispositivo se encuentran en el Cuadro 5.10.

Actualmente hay muchas maneras programar el chip como MicroPython (que implementa Python), ESPlorer (que ofrece la posibilidad de la utiliza-

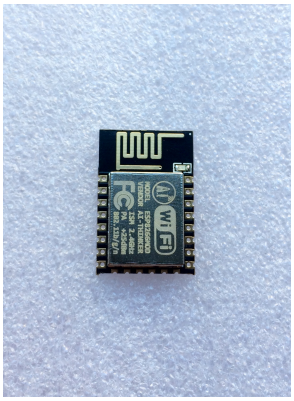

Figura 5.31: Chip ESP8266

CPU RISC de 32 bits	Tensilica Xtensa LX106
Reloj	80 MHz
Memoria interna	64KB (instrucciones) 96KB (datos)
Memoria externa flash QSPI	512 - 4MB (ampliable a 16MB)
IEEE 802.11 b/g/n Wi-Fi 2.4GHz	Con autenticación WEP, WPA/WPA2
Pines GPIO	16
Otros	SPI,I2C, UART y conversor ADC de 10 bits

Cuadro 5.10: Características principales del chip ESP8266

ción de comandos LUA[26], Microphyton y comandos AT) y Arduino (el cual está basado en C) entre otros. Los comandos Hayes [25] (o AT) es un lenguaje con un estándar abierto de comandos para configurizar y parametrizar dispositivos. Estos comandos tienen el prefijo "AT" que quiere decir "Atención". Por otra parte, Lua es un lenguaje de programación diseñado para una programación procedimental con utilidades para la descripción de datos con gran soporte para la programación orientada a objetos, programación funcional y orientada a datos. Está fuertemente ligada a la programación de este chip, por lo que es frecuente su uso, sin embargo se ha decantado por el lenguaje de Arduino.

Finalmente se ha apostado por el uso del software Arduino, por una serie de ventajas como que al estar basado en C, se puede programar en un IDE (*Integrated Development Environment*) ya experimentado y también que facilita la programación al ser casi idéntica que un Arduino. La creación de una serie de librerías idénticas de Wi-Fi que las de Arduino, ha permitido la reutilización del código para este chip. Se facilita mucho trabajo con las librerías creadas del ESP8266 en Arduino. Haber programado todo en otro entorno que haga uso de comandos AT o LUA, por ejemplo, conllevaría una

ralentización bastante considerable a la hora de programar, ya que previamente se requeriría el aprendizaje del mismo.

Concretamente para el modelo que se va a emplear, que es el ESP-12[23], el mapa de *pinout* que lo representa sería el indicado en la Figura 5.32. Contiene 20 pines activos, un LED, una antena de PCB, *shield* y sus dimensiones son de 24x16mm.

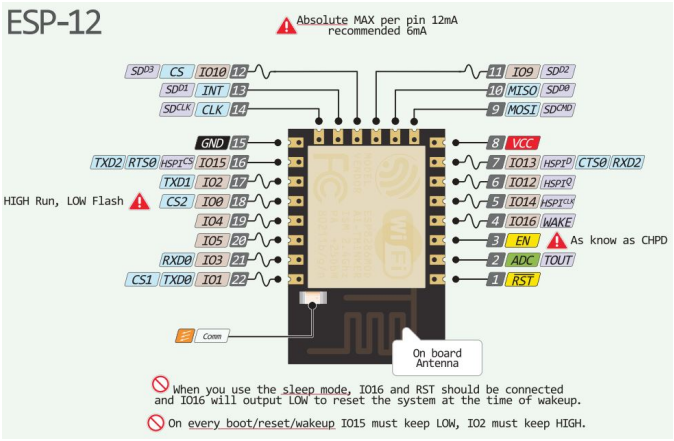

Figura 5.32: *Pinout* del chip ESP8266

Algunas de las funciones básicas del mismo son: `pinMode()`, `digitalReal()`, `digitalWrite()`, `analogWrite()` o `analogRead()` . Los pines GPIO pueden leerse con el mismo número de pin. Quiere decir que para leer el GPIO6, el comando se llamará `digitalRead(6)`. Los pines I/O digitales tienen protección de sobretensiones e inicialmente están configurados para entrada. Los pines GPIO0 hasta el GPIO15 pueden ser de entrada, salida o entrada *Pull Up*.

El comando `analogRead(A0)` lee el valor del canal ADC conectado. El ADC analógico que posee es de 10 bits, lo que da unos posibles 1024 muestras diferentes y su rango comprende de 0V a 1V.

Otro aspecto a tener en cuenta es que, al elegir un GPIO como salida, por ejemplo para conducir un LED, la corriente máxima es de 12mA. Una corriente superior a esa podría dañar el dispositivo.

Antes del ESP-12 [24] ha existido una enorme variedad de encapsulados y placas (Figura 5.33). Puede que a la hora de escoger se deban tener en cuenta aspectos como la necesidad de una cantidad de memoria mayor, o la correcta verificación de la homologación, ya que hay algunos modelos que no cumplen con la normativa del país. Sin embargo, módulos como el ESP-07 y el ESP-12, entre otros, permiten realizar circuitos en los que le ESP8266 no solo se encarga de las comunicaciones Wi-Fi sino que también actúa de

microcontrolador que hace la gestión del dispositivo. El ESP-02, ESP-04, ESP-05, ESP-06, ESP-08, ESP-09 y el ESP-10 no cuentan con antena, será por tanto necesario agregarle mediante su conector una antena externa.

Pero sin embargo, se ha elegido este chip por tener las funcionalidades necesarias para llevar acabo el diseño, tales como antena, se puede programar y existe numerosa documentación y desarrollo de librerías para su programación.

Su sucesor es el chip ESP32, que apareció en 2016, que además de Wi-Fi soporta Bluetooth con un aumento de la potencia con respecto el ESP8266, pero que en ningún momento pretende sustituirlo. Debido al aumento de características que permite este, será el cliente el que decida entre uno u otro de acuerdo a sus necesidades.

Figura 5.33: Distintos encapsulados

Otro de los grandes motivos por los que se ha elegido ESP8266, antes que cualquiera de los anteriores encapsulados o su sucesor ESP-13, se debe a una serie de módulos hardware desarrollados para este chip, y cuyas explicaciones y características vienen dadas en los siguientes subapartados. Hablamos del módulo *NodeMCU v1.0* y el *Wemos D1 Mini*.

NodeMCU v1.0

NodeMCU va a ser una de las placas de desarrollo que integran el microcontrolador comentado y que se presentará como una de las soluciones de nuestro proyecto como el dispositivo que actuará de pasarela y que comunicará al cliente con el dispositivo final del hogar (véase Figura 5.34).

Es un dispositivo versátil, relativamente potente y a un precio por el que muchos se decantan para desarrollar proyectos IoT. También "NodeMCU" es una plataforma IoT de código abierto haciendo referencia al *firmware* que hace uso de comandos LUA que sustituye al laborioso kit de desarrollo del ESP8266. Tiene una memoria flash de almacenamiento de 4MB y una memoria RAM de 128KB.

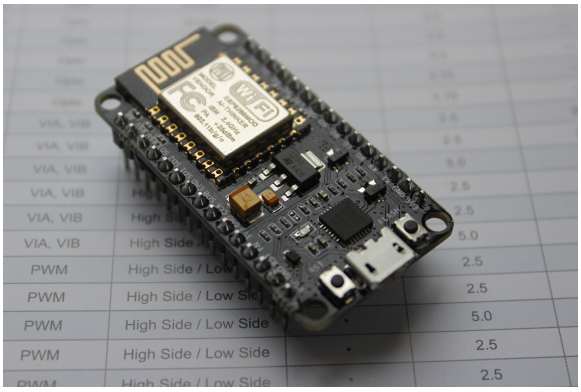

Figura 5.34: *NodeMCU v1.0*

Su placa tiene unas dimensiones de 49x24.5mm (la segunda generación), y para la programación del dispositivo solo se necesita un cable micro USB y un ordenador, ofreciendo entre otras ventajas, la posibilidad de programar desde el IDE de Arduino. Dicho cable micro USB suele alimentarse de 5V, pero como el ESP-12 se alimenta solo de 3.3V, esta placa incluye un regulador de tensión para que finalmente se alimente de 3.3V. El consumo residual del regulador es de 8mA. Cabe mencionar que el cable USB no es el único modo de alimentar la placa. *NodeMCU* dispone de un puerto "Vi" el cual podemos alimentar con una tensión de 5V hasta 12V inclusive sin necesidad del USB. Además el *pitch* es de 2mm, lo que facilita su implementación en una placa de tipo *Protoboard*.

Principalmente los fabricantes son: Amica, LoLin y Doit.

El *Pinout* de la versión empleada del módulo *NodeMCU* sería la mostrada en la Figura 5.35.

Todos los pines están cableados hasta el interfaz de *NodeMCU*, disponiendo de 11 pines de I/0 y el puerto de entrada analógica A0. Los pines GPIO del 6 al 11 se suelen usar para conectar la memoria flash. Si se hace uso de ellos, es probable el reinicio inesperado del *NodeMCU*. Además dispone de dos botones con los nombres de "RST" y "FLASH". "RST" iría directamente conectado al pin de "RESET", y "FLASH" se conecta a GPIO0 que activa la carga del *firmware*. Para nuestro proyecto, estos botones estarán en segundo plano puesto que son innecesarios con el IDE de Arduino.

Acerca del puerto analógico A0, se ha realizado un estudio en el cual se ha comprobado la cantidad de muestras digitales que es capaz de generar. Dicha prueba ha dado como estimación que el puerto A0 es capaz de generar 10000 muestras por segundo (10kHZ). Tiene un rango de posibles valores comprendidos entre 0V y 1V. Con esa velocidad de 10kHz, claramente no cumple el Teorema de Nyquist, el cual afirma que la frecuencia de muestreo

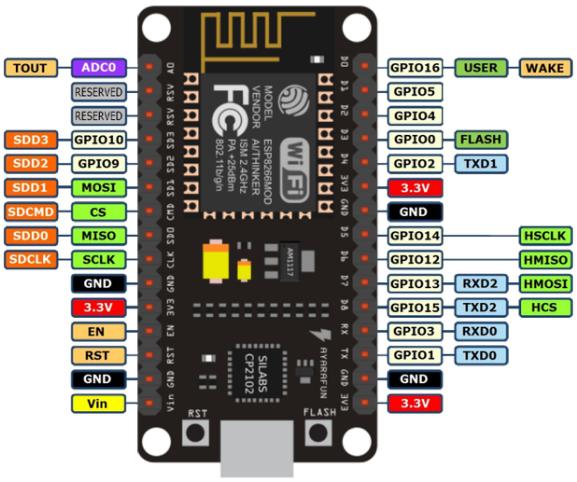

Figura 5.35: *Pinout* del módulo *NodeMCU v1.0*

debe de ser mayor o igual a doble de la frecuencia de la señal (que en términos infrarrojos se mueve en torno a 38kHz - 42kHz). Es por ello que pobremente generaría 1 de cada 4 muestras de la señal.

Aunque dicho IDE permite unas velocidades de carga de 921kbps, velocidades que realmente interesan para la reducción del tiempo de carga, es cierto que dicha velocidad se verá limitada por la calidad del cable USB, que en caso de no ser la suficiente no permitirá la subida del programa, por lo que será necesario ajustar manualmente la velocidad hasta que el cable lo permita. Cuando se conecta al PC el sistema operativo realiza la instalación automática de los *drivers*.

Existen varias versiones de la placa *NodeMCU*:

- La primera generación lleva la numeración de v0.9. Una placa *NodeMCU* de un característico color amarillo cuyas dimensiones son de 47x31mm. Dichas características complican su uso en una placa *Protoboard* normal al ocupar los diez pines horizontales. El microcontrolador que usa es el ESP-12.

- La segunda generación V2 tiene la numeración V1.0 y da solución el inconveniente del tamaño de su antecesora permitiendo el desarrollo de proyectos en placas *Protoboard*. El nuevo tamaño (que simplemente se basa en la reducción del ancho y no del alto) es justo el que permite la disponibilidad de una fila de pines a cada uno de los lados. Por último, cabe añadir que el microcontrolador es mejorado al ESP-12E.

- Por último, también se desarrolla la tercera generación, aunque no se considera como tal (de hecho su numeración es la misma que la de la

segunda generación, v1.0) por el fabricante LoLin, ya que no representa una mejora de su antecesora, solamente unos pequeños cambios. El tamaño vuelve a aumentar, imposibilitándolo para el correcto funcionamiento en placas *Protoboard* aunque a cambio ofrece un puerto USB más robusto.

En definitiva, una vez conocidas todas las versiones disponibles (véase Figura 5.36) en el mercado, se ha tomado la decisión de usar la segunda generación, el *NodeMCU v1.0* gracias a su tamaño que hace posible su diseño en placas estándares y su mejora en el chip ESP12-E.

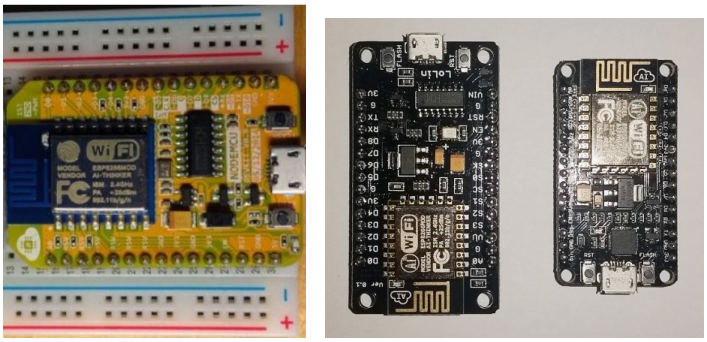

Figura 5.36: A la izquierda la primera versión, en medio la tercera y a la derecha la segunda

Wemos D1 Mini

El título de este subapartado refleja el nombre de la otra placa que se presenta como alternativa a *NodeMCU* dada su equiparable capacidad y funcionalidades (véase Figura 5.37), también gobernado por el chip ESP8266 para la gestión de la comunicación Wi-Fi. Tiene una memoria interna de 32KB para instrucciones y 96KB para datos[27].

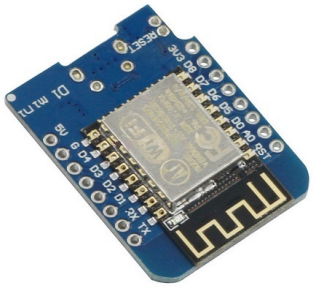

Figura 5.37: *Wemos D1 Mini*

Su alimentación también es mediante cable micro USB y por lo tanto, de 5V, con un solo botón de "RESET", una memoria flash con un almacenamiento de 4MB, pero con un tamaño más reducido al *NodeMCU* disponiendo solo de 16 patillas. Todas las versiones de *Wemos D1* son programables mediante el puerto micro USB.

El *Pinout* de la versión elegida del módulo se encuentra en la Figura 5.38.

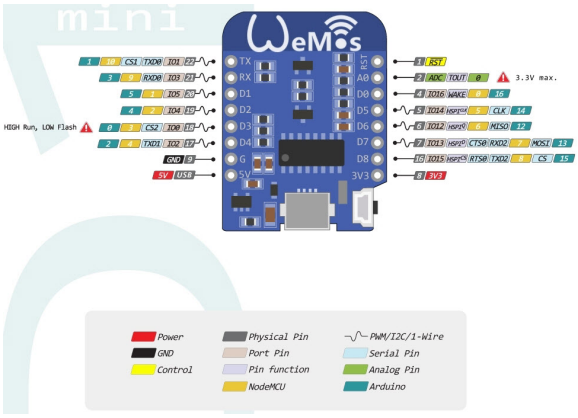

Figura 5.38: *Pinout* del módulo *Wemos D1 Mini*

Hay varias versiones de *Wemos D1*:

- Por un lado se encuentran las primeras placas *Wemos D1*: *Wemos D1* y *Wemos D1 R2*. Con 11 entradas/salidas digitales, 1 entrada analógica. Programables mediante conector micro USB y una distribución de pines tipo Arduino UNO y su velocidad del reloj es de 80MHz. Su inconveniente es el tamaño, con unas dimensiones considerables (53.4x68.6mm), cuya versión del ESP8266 es el ESP-12F. La única diferencia entre las dos es que la última incluye una modificación para que junto con Arduino se le pueda cargar *NodeMCU*.

- Más tarde se fabricaron las versiones *Wemos D1 Mini* y *Wemos D1 Pro Mini*. Ideales por su reducción de tamaño con respecto a las anteriores versiones que también ofrece los 12 pines, con un tamaño reducido de 25x34.2mm. En este caso la versión del ESP8266 es mejorada con respecto a *Wemos D1* siendo el ESP-12S.

Estas características tan similares al *NodeMCU* permiten que en este estudio se pueda realizar un diseño paralelo con las dos placas, concretamente el *NodeMCU v1.0* y el *Wemos D1 mini*. Sus similares prestaciones y forma de programar facilitarán que la solución ofrecida tenga flexibilidad en cuanto a hardware se refiere.

5.4. Arduino

Como herramienta software se implementará el conocido software Arduino[28] (logo en la Figura 5.39). Arduino es una plataforma de hardware libre que además dispone de su propio entorno de desarrollo que hoy en día se utiliza para infinidad de placas. El IDE de Arduino contiene un lenguaje muy similar al ya conocido lenguaje C, con lo que simplifica y facilita su uso.

Figura 5.39: Logo del software Arduino

Lo que lo ha llevado a la fama no es solo el hardware Arduino creado (consistente en una placa con un microcontrolador) y el interesante software sino también la amplia comunidad que alberga y apoya este desarrollo en continuo movimiento publicando nuevo contenido días tras día. Por estos tres pilares, Arduino ha alcanzado un importante reconocimiento.

El software Arduino está compuesto por un editor de código, un compilador, un depurador y constructor de interfaz gráfica que en conjunto forma el entorno de aplicación, todo en un programa de aplicación. Incorpora una gestión de librerías, la gestión de placas y los avisos de actualización de versiones de librerías y cores. Los programas de Arduino están compuestos por un solo fichero con la extensión ".ino". El fichero principal debe de estar dentro de una carpeta con el mismo nombre que el fichero. Se puede destacar de dicho software su sencillez y facilidad de uso. Por supuesto está disponible para todos los sistemas operativos de ordenador.

El lenguaje de programación al que se asemeja es C++ pero Arduino ofrece librerías que ayudan a la programación de los pines de entrada y salida, así como otras librerías para operaciones específicas. Otro hecho que refleja que C++ y Arduino no son el mismo lenguaje, sino que solo se asemejan, es que en la estructura del programa no se utiliza la función `main()` sino que se usan las funciones `setup()` y `loop()`. Abajo de la Figura 5.40 en negro aparece la consola de error, arriba, de izquierda a derecha, los botones en verde son: verificar (comprueba si el programa está bien escrito y puede funcionar), subir (carga el programa tras compilarlo), nuevo *sketch* (crea un programa nuevo), abrir *sketch*, guardar (salva el programa en el ordenador)

y a la derecha del todo, para abrir el Monitor Serie (donde abre una ventana de comunicación con la placa en donde se visualizan las respuestas que este da). En medio se encuentra el editor, y la pestaña en blanco refleja el nombre del *sketch* actualmente abierto.

Figura 5.40: IDE del software Arduino

Algunas de las ventajas de las nuevas versiones es que la detección de la placa que conectamos se hace de manera automática, realiza un control y visualización de la memoria Flash y SRAM que ocupa un *sketch* o proyecto y cada vez que se carga y se sube el proyecto a la placa se realiza un autoguardado del *sketch*.

El Monitor Serie es una de las opciones que incorpora Arduino y que será de gran utilidad para verificar y comprobar el correcto funcionamiento de nuestro código paso a paso. Esto es posible porque muestra los datos enviados por Arduino a través del puerto serie al que va conectado la placa.

Cuando se carga un *sketch* en el ESP8266 con el IDE de Arduino, el software primero borra de la placa el anterior *sketch* que hubiese y después

carga el nuevo en la memoria flash del ESP8266. También está el gestor de tarjetas, que muestra a qué placas existe soporte para programarlas, instalando aquellas que aun no han sido añadidas.

Las variables que se utilizan en los programas son las variables globales y las locales. Las variables globales son declaradas al principio del programa, antes que el comando `setup()`. En esta posición tan temprana cualquier función puede verla y utilizarla, y es importante recordar que ocupa un espacio de memoria permanente de manera que si se abusa de ellas se puede estar originando un uso ineficiente de la memoria. Las variables locales se definen dentro de las funciones, de manera que solo esa función puede verla y modificarla. También existen las variables constantes como son: *HIGH/-LOW* que son los niveles en bajo y en alto de posibles señales de entrada y salida; *INPUT/OUTPUT* para definir como entrada o salida; *false/true* que a diferencia de las anteriores estas se escriben en minúscula.

Se diferencian los siguientes tipos de variables en Arduino recogidos en el Cuadro 5.11.

Tipos de datos	Memoria	Rango de valores
boolean	1 byte	0 o 1 (*True o False*)
byte / unsigned char	1 byte	0 - 225
char	1 byte	-128 - 127
int	2 bytes	-32768 - 32767
word / unsigned int	2 bytes	0 - 65535
long	2 bytes	-2147483648 - 2147483647
unsigned long	4 bytes	0 - 4294967295
float/double	4 bytes	-3,4028235E+38 - 3,4028235E+38
string	1 byte + x	*Array* de caracteres
array	1 byte + x	Colección de variables

Cuadro 5.11: Tipos de variables en Arduino

Los *arrays* son conjuntos de valores a los que se accede con un índice. Cualquier valor puede ser recogido haciendo uso del nombre de la matriz y el número del índice. En la matriz, el primer valor tiene índice 0, que quiere decir que su posición es 0. Un *array* tiene que ser declarado y de manera opcional, se asignan valores a cada posición.

Un *string* (*char array*) es un *array* de tipo *char*, variable de la que se hará uso continuo en nuestro *sketch*. Un *string* (Objeto) es una clase que trata con cadenas de texto de manera fácil con métodos y funciones que ofrece la clase, aunque en general los *Strings* suelen usar de manera intensa la memoria.

Contiene operadores: aritméticos, compuestos y de comparación; estructuras de control: estructuras de decisión y estructuras de repetición; y también están las funciones definidas por usuario. Estas últimas son bloques de códigos con un nombre que se ejecutan cada vez que se llama a la función. Ejemplos de funciones también `setup()` y `loop()`, en las cuales la primera solo se ejecuta una sola vez, al principio, y la segunda se ejecuta en bucle.

Más funciones básicas que son necesarias para el código del proyecto pueden ser las de los puertos digitales de entrada y salida con: `pinMode(pin, modo)`, `digitalWrite(pin,valor)`, y `int digitalRead(pin);` y también están las de tiempo como el *delay(ms).*

En definitiva, Arduino ofrece la facilidad de poder programar en un lenguaje que ya conocemos y que podemos aplicar inmediatamente sin la necesidad de gastar tiempo adicional en aprender un lenguaje nuevo que probablemente no se necesite en ocasiones futuras. Arduino, al ser capaz de gestionar una amplia variedad de placas para su programación, es con toda seguridad una herramienta software que se emplee en el futuro para otro tipo de proyectos que requieran de otro tipo de SoC (*System on Chip*), luego resulta muy útil su aprendizaje para adaptarse antes a él. Estas serían las premisas por las que se ha decidido que hacer uso de este software como parte de la ayuda en el diseño e implementación de la solución final complementaría con mayor éxito el trabajo.

5.5. MQTT

MQTT[29] será otra de las herramientas software en las que se basará este proyecto y que tendrá un papel fundamental. Será el protocolo de comunicación utilizado por el servidor privado con el dispositivo pasarela de manera que este lo interprete de acuerdo a un orden establecido (véase Figura 5.41). Una de las opciones que permite MQTT es la de ser cifrado (MQTT *Secure*), con encriptación SSL o TLS, con su respectiva sobrecarga de la red, por ello la opción más sencilla será la de no tener ningún tipo de autenticación. Existen otro protocolos similares como STOMP (Protocolo orientado a texto de mensajería en *streaming*) o XMPP (Protocolo de mensajes de aplicaciones Web), que como bien indican sus nombres, el primer está enfocado a mensajería en *streaming*, y el segundo solo orientado a aplicaciones Web.

Se define como un protocolo estandarizado [30] del tipo "Publicador - Suscriptor" de mensajería simple en el Internet de las Cosas y en el M2M (*Machine to Machine*). Es asíncrono y ligero, lo cual implica un consumo muy pequeño de ancho de banda y bajas necesidades de recursos ideal para su uso con sensores o pequeños dispositivos empotrados, por lo que con estas características puede presumir de necesitar un consumo realmente bajo.

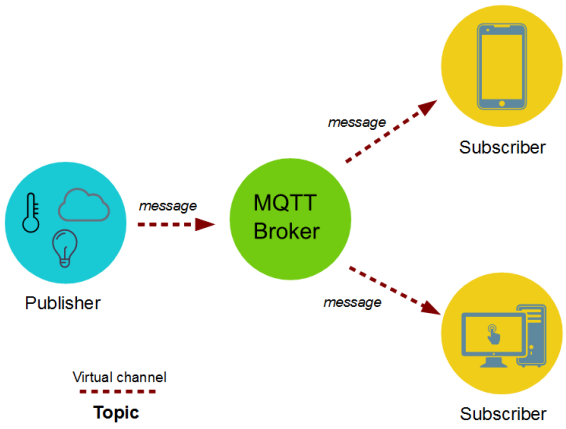

Figura 5.41: Escenario básico del protocolo MQTT

La arquitectura de MQTT se basa en la topología estrella, con un nodo principal llamado *broker* o servidor y los restantes nodos que se denominan clientes. Este servidor permite hasta 1000 clientes y gestiona tanto la red manteniéndola activa, como la transmisión de mensajes pues por él se propaga cada uno de ellos. Teniendo en cuenta el tipo de topología y las funciones del *broker*, está claro que la comunicación solo puede ser de uno a uno, o de uno a muchos. Ese mismo servidor puede estar instalado en un ordenador, un servidor coorporativo, *hosting*, o una Raspberry Pi por ejemplo.

Un método de MQTT son los tópicos o temas. Aquí será el lugar donde el cliente publica el mensaje, y aquellos quienes deseen recibir todos los mensajes de dicho tópico, deberán anteriormente subscribirse. Su estructura es jerárquica o ramificada (véase Figura 5.41), y cada jerarquía está separada con e signo "/". Cuando se escribe "#" detrás de un tópico a la hora de subscribir un nodo, este realizará una lectura de todos los subtópicos que se encuentren dentro de este. A modo de seguridad, los tópicos deben de ser lo más personalizados posible de manera que no se produzcan conflictos entre dispositivos y se reduzca la posibilidad de poder publicar en dicho tópico por alguien ajeno a la red.

Este protocolo utiliza un puerto estándar, que es el puerto TCP/IP número 1883 reservado para su uso. Además también está el puerto número 8883 para el caso de MQTT *Secure*.

Otro parámetro en MQTT es la calidad de servicio, que implica el esfuerzo que realizará el *broker* en asegurar la recepción del mensaje. Los mensajes se pueden enviar a cualquier nivel de calidad de servicio y a su vez los clientes tienen la posibilidad de subscribirse a tópicos de cualquier nivel de QoS (*Quality of Service*). Eso significa que el cliente, aun estando subscrito a un tema con un nivel alto de calidad de servicio, es posible subscribirse a ese

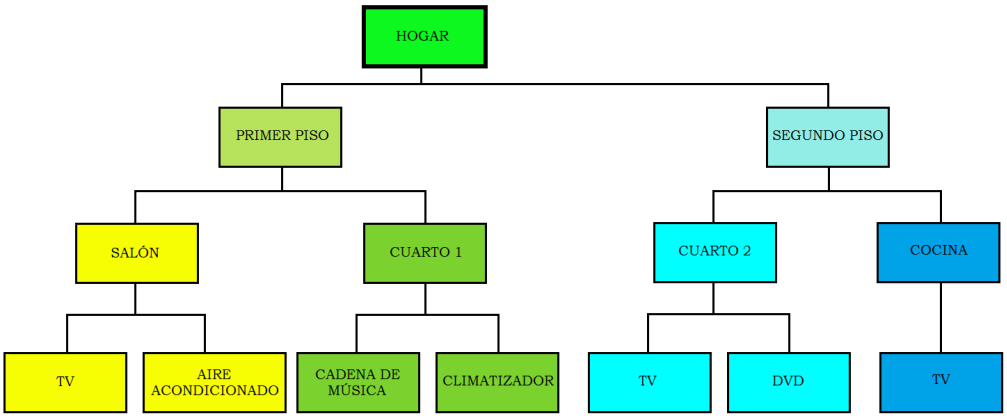

Figura 5.42: Topología en estrella seguida por los tópicos de MQTT

mismo tema modificando el nivel de QoS a otro nivel más bajo. Un nivel más alto implica mayor fiabilidad, pero también mayor latencia y más requisitos de mayor ancho de banda. El nivel 0 realiza una entrega del mensaje sin confirmación, en el nivel 1 la entrega requiere de una confirmación, y en el nivel 2 dicha entrega del mensaje realiza una *four step handshake* como confirmación de recepción.

Además, se incluye una funcionalidad que permite la retención de los mensajes. La retención de mensajes no es más que el servidor ofrece la posibilidad de mantener el mensaje aun cuando este ya ha sido enviado a todos los subscriptores. Si un cliente nuevo se subscribe al tópico después, este va a recibir ese último mensaje que ha sido retenido en el servidor. Es útil cuando se trata de un tema con poca frecuencia de actualización, permitiendo a nuevos clientes que no tengan que esperar mucho tiempo para recibir información, sino que reteniendo será posible una actualización instantánea.

Uno de los más importantes programas de MQTT, es Mosquitto [38], del cual más adelante en el capítulo 7 de Diseño e implementación se utilizará. Este software es de código abierto, disponible su instalación tanto para Windows como para Linux entre otros. Este software facilita la creación de un *broker* MQTT en nuestro sistema con su respectivo puerto 1883 en escucha.

Tanto este programa como el protocolo en sí, han sido elegidos para formar parte del proyecto no solo por todo lo anterior mencionado, sino porque MQTT da vía libre a la composición de los mensajes que se envían en sí, es decir, que a la hora de la implementación no hay que adaptarse a ningún tipo de estructura del mensaje puesto que MQTT sólo se encarga de habilitar la comunicación. Seremos nosotros los encargados (en el siguiente capítulo) de dar forma a esos mensajes que serán interpretados por el ESP8266 como

se le ha ordenado.

Capítulo 6

Diseño e implementación

En el presente capítulo se pretenden abordar los procedimientos que se han llevado a cabo para poder diseñar e implementar la solución de este proyecto.

Una vez tratados y expuestos todos los términos teóricos en lo que se envuelve la solución, mediante un esquema general se detallará el camino a seguir. No sin antes realizar una breve descripción acerca de los tipos de representación de señales de infrarrojas.

Más tarde, además se explicará el diagrama de flujos por el que se rige el chip ESP8266, las librerías que se emplean que ayudarán al chip a su correcta conexión y comunicación. Además, es imprescindible mostrar cual es la estructura seguida de un mensaje MQTT y los diferentes tipos de circuitos llevados a cabo en el chip ESP8266.

Conceptos previos

Antes de adentrarnos en la implementación como tal, es interesante saber identificar y representar las señales de infrarrojos. Es necesario conocer los distintos formatos en los que se representan las señales IR producidas al enviar un comando mediante cualquier control remoto. Se diferencian en: comandos PRONTO-HEX, comandos HEX y los códigos RAW, con sus respectivas definiciones.

La forma más básica en la que se recibe un comando suele ser en un código RAW. Un código RAW en las señales infrarrojas representa mediante una cadena de números la duración en el tiempo de los flancos en alto y en bajo de la señal infrarroja que se ha recibido. La longitud del mismo depende del protocolo. Como ejemplo, a continuación para dar a entender su forma se mostrará un código RAW recibido, cuyos valores se dan en formato decimal:

{4500, 4500, 500, 1750, 500, 1750, 500, 1700, 550, 600, 500, 600, 550}

En donde el primer número sería el intervalo que transcurre con el flanco en alto, y el segundo intervalo pasaría a formar parte del intervalo inverso, y así sucesivamente. En caso de que en la representación del código RAW aparezca un signo tal como "+" o "-" delante de cada número, querrá reflejar el nivel del flanco, siendo en alto para el caso del signo "+" y en bajo para el caso del signo "-". Por otra parte, cada valor de la cadena representa en microsegundos el intervalo en el que se mantiene ese flanco.

Anteriormente se ha mencionado que existen otras vías para la representación de la señal infrarroja. Otro caso son los comandos PRONTO-HEX. Dicho código significa lo mismo que el anterior código RAW, ya que es una cadena de valores que consiste en la representación del tiempo o duración del flanco de la señal infrarroja en alto o en bajo. La diferencia reside en el tipo de formato, ya que esta vez se realiza en hexadecimal. Por tanto, se puede realizar una conversión de código RAW a código PRONTO-HEX realizando una conversión de formato decimal a hexadecimal. Para el caso contrario, se realizaría el procedimiento inverso. Como última noticia sobre estos dos códigos, cabe afirmar que para su composición no es necesario conocer de antemano el tipo de protocolo, ya que no influye en su formación.

Por lo tanto, la conversión [31] del mensaje del código RAW anteriormente expuesto a código PRONTO-HEX sería:

{1194 11C6 1F4 6D6 1F4 6D6 1F4 6A4 226 258 1F4 258 226}

Por último, dentro de estos conceptos previos, queda exponer el tipo de representación de señales IR mediante código HEX. Como bien indica su nombre, su formato es en hexadecimal. Es importante tener en cuenta que su composición, a diferencia de los anteriores, es necesario conocer previamente el protocolo en el que se basa. Esto facilitará al dispositivo poder analizar, según el código RAW obtenido, la formación de los números lógicos, para más tarde agruparlos de cuatro en cuatro formando un hexadecimal.

En resumen, una vez obtenido el código RAW, teniendo en cuenta la modulación del protocolo utilizado, un "1" lógico tendrá una duración y un "0" lógico tendrá su respectiva duración que se obtendrán según la composición de los intervalos del código RAW, obteniéndose:

{1 1 1 0 0 0 0 0 1 1 1 0 0 0 0 0 0 1 0 0 0 0 0 0}

Que sería la cadena en binario, que para obtenerse su código HEX, solo es necesario agruparlos de cuatro en cuatro obteniéndose el número hexadecimal:

{E0E040}

Para realizar la conversión inversa, al igual que en el anterior caso es necesario saber el protocolo empleado. Esto permitirá poder demodular de la numeración lógica al intervalo en el dominio del tiempo.

Diseño a seguir

Este proyecto no pretende desarrollar una solución completa a un problema, puesto que será la suma de varios proyectos llevados a cabo por el profesor Doctor Jorge Navarro Ortiz, los que se complementarán para dar forma a una solución que es capaz de cubrir dicho dilema.

La solución de este proyecto seguirá un esquema el cual todo el proceso se lleva a cabo dentro del hogar (como se refleja en la Figura 6.1). Tanto cliente visto como usuario final como el otro extremo que sería el dispositivo con el que se pretende comunicar, ambos se encontraran en el mismo espacio. A diferencia de un control remoto habitual que solo cubre el radio al que este alcanza limitándose solo a ese espacio y de manera direccional, esta solución permite poder accionar el aparato eléctrico desde cualquier parte de la casa a la cual el radio de cobertura de Wi-Fi alcanza. Es por tanto que mientras que tanto usuario como dispositivo pasarela y servidor MQTT se encuentren bajo dicho radio, será factible dicha implementación.

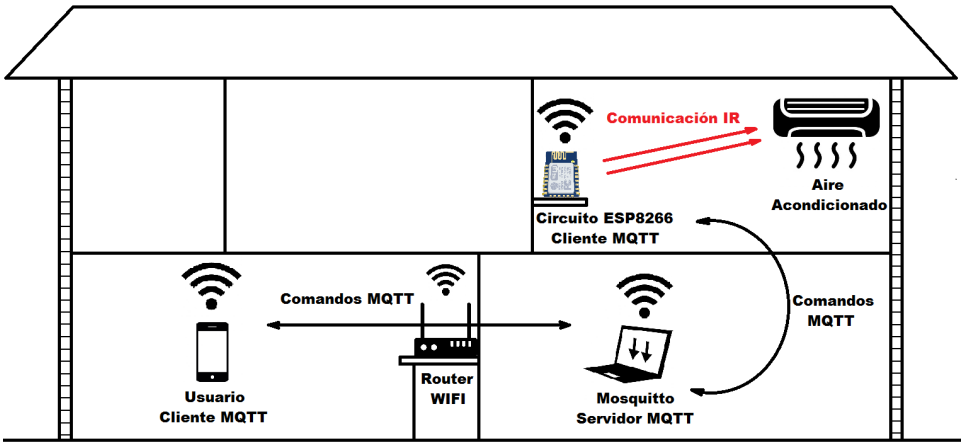

Figura 6.1: Escenario del actual diseño

El usuario, desde una aplicación haría de cliente MQTT, enviando un mensaje al *broker* mediante comandos MQTT, dando se publicaría en el tópico elegido, donde seguidamente el ESP8266 recibiría el mensaje al estar subscrito en ese mismo tópico y lo interpretaría enviando la señal infrarroja idónea al dispositivo final, que en este caso se ha optado por un aire acondicionado.

Aun así, esta solución, permite un alcance superior, puesto que se han empleado unas herramientas tanto hardware como software que posibilitan una ampliación del escenario. Dicha ampliación podría acaparar un desarrollo el cual el usuario final tiene la capacidad de comunicarse con los dispositivos del hogar ubicándose fuera de la casa.

Además, para facilitar la comprensión del comportamiento del dispositivo pasarela, que puede ser tanto el *NodeMCU v1.0* o el *Wemos D1 Mini* con sus respectivos circuitos, se ha creado un diagrama de flujos que muestra de manera más gráfica sus estados (véase Figura 6.2).

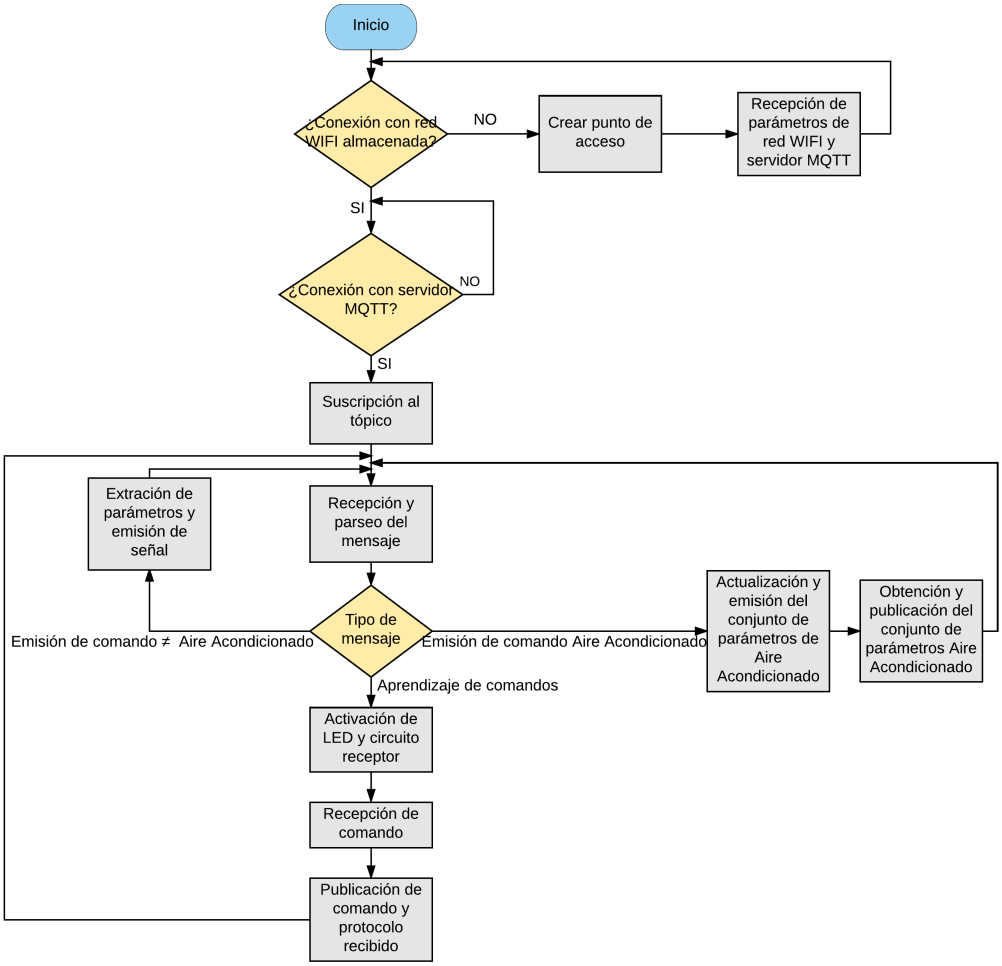

Figura 6.2: Diagrama de flujo del dispositivo pasarela

El dispositivo nada más encenderse, comprueba que tenga acceso a alguna red Wi-Fi. Si anteriormente ha formado parte de alguna red Wi-Fi, primeramente intentará la conexión con los parámetros ya almacenados. En caso de que no sea posible esa conexión, creará un punto de acceso Wi-Fi

con un nombre formado por "ESP" y el número de identificación del chip. El objetivo de la creación de este punto de acceso no es más que el cliente se conecte con su dispositivo o *smartphone* para introducirle los parámetros de la red Wi-Fi a la que debe acceder (dirección IP y contraseña). Además cabe recordar que se necesitará especificar la dirección IP y el puerto del servidor MQTT.

Una vez enviados los parámetros desde el terminal del cliente y recibidos por el dispositivo pasarela, eliminará el punto de acceso e intentará la conexión siguiendo los parámetros. En caso de que no estén bien escritos, o no sean correctos, o la red no esté activa, la conexión fallaría y volvería a crear un punto de acceso. En caso afirmativo, intentaría la conexión con el servidor MQTT, de manera que si es errónea, realiza un bucle de intento de conexión cada 5 segundos. Si la conexión se realiza con éxito, el ESP8266 se subscribe al tópico donde se van a publicar los mensajes, donde quedaría a la espera de la recepción de los mismos.

Cuando se produce la recepción de un mensaje, como primer paso, el ESP8266 identifica mediante el parseo los campos del protocolo y del tipo del mensaje. Dependiendo de estos parámetros, se realizará una opción u otra.

Para la opción, el cual se ha elegido el modo de emisión de una señal infrarroja para un dispositivo diferente de un climatizador. El chip extraerá los parámetros del mensaje y se limitará a enviar la señal IR correspondiente al protocolo exigido de manera que una vez finalizado, el dispositivo pasarela volverá al estado de espera a la recepción de comandos.

La segunda opción posible reside en la emisión de una señal infrarroja destinada a un aire acondicionado. El modo de emisión de señales IR se ha separado entre esta opción y la anterior debido a las diferencias en cuanto a la estructura de los mensajes. Mientras que para un dispositivo que no es un climatizador solo se envía la información referente a la tecla pulsada, en el caso de un aire acondicionado cada comando pulsado envía la información con todos los parámetros de funcionamiento de la configuración que se desea establecer.

Es por ello que en esta nueva opción se actualiza la información de la tecla pulsada y se emite el conjunto de parámetros del aire acondicionado. Seguidamente ejecuta una publicación del conjunto de parámetros en el tópico al que inicialmente se subscribió. Finalmente y al igual que en el anterior caso, cumplida su tarea, vuelve al estado de espera de recepción de un nuevo mensaje.

La última opción que permite el dispositivo pasarela será la de activar su modo de recepción cuando el mensaje MQTT que reciba le ordene el aprendizaje de un nuevo comando. Es por ello que a la hora de realización

de la orden, este activa un LED luminoso que da la señal al usuario de que el circuito está a la espera de la recepción de la nueva señal infrarroja. Una vez que el usuario interactúa con el dispositivo presionando la tecla a aprender del control remoto, el chip reciba el código RAW y publica en el tópico dicho código RAW, el código HEX (en caso de identificar el protocolo), y el protocolo detectado, entre los parámetros más importantes de ese nuevo mensaje publicado. Al igual que en los anteriores casos, su estado vuelve a la espera de un nuevo mensaje.

Librerías de Arduino utilizadas

- **WiFiManager.h:**

  Librería [36] que permite la conexión a una red Wi-Fi cuyos parámetros (SSID y contraseña) pueden ser establecidos dinámicamente, sin necesidad de tener que compilar el código en el chip para cambiar de red. Desde un terminal el cliente puede introducir los nuevos parámetros a esa red Wi-Fi con la creación de un punto de acceso. Dicho punto de acceso posibilita rescatar todos aquellos parámetros que se deseen, entre ellos, para este proyecto se han elegido los referentes a la dirección IP y el puerto del servidor MQTT. Como funcionalidad adicional, ofrece un escaneo de las posibles redes Wi-Fi que se encuentran cerca para advertir al cliente si se puede acceder a la red deseada por el mismo.

- **FS.h:**

  El objetivo de esta librería consiste en la escritura y lectura de la memoria flash externa del ESP8266. Concretamente en nuestro caso será de utilidad para almacenar y posteriormente leer los parámetros SSID, contraseña de la red Wi-Fi, dirección IP y puerto del servidor MQTT.

- **PubSubClient.h:**

  Librería [37] enfocada a la gestión de MQTT. Ofrece la posibilidad de conectarse a un servidor en concreto a través de un puerto. Además, permite la suscripción y publicación de tópicos. Dispone de una función de *callback* que permite ejecutar el código deseado al recibir un mensaje.

- **IRremoteESP8266.h:**

  Librería [35] que contiene toda la información relativa al envío y recepción de señales infrarrojas dependiendo del protocolo. Entre sus funciones existe la posibilidad de introducir diferentes parámetros como: el código RAW, el comando HEX, el comando en decimal, el número

de bits del mensaje, la frecuencia de transmisión infrarroja o el número de veces que se debe repetir la señal; de manera que procesando esos datos, adapta la señal infrarroja oportuna, a esos ajustes, que después emitirá. Esto es posible ya que implementa funciones creadas exclusivamente para el envío de señales IR generadas por código RAW o por un comando HEX. En este último formato solo disponible para algunos protocolos tales como NEC, SONY, JVC, SHERWOOD, COOLIX, WHYNTER, LG, RC5, RC6, DISH, SHARP, SAMSUNG, DENON o PANASONIC.

Ahora se detallará con varios ejemplos de varios protocolos. El primer código muestra cómo se programa el envío de varias señales IR. En primer lugar se indica a la librería el puerto que se quiere conectar a los LEDs infrarrojos y después se añaden los comandos oportunos que generan la señal final teniendo en cuenta los parámetros de entrada, que podrán ser como en este caso, comandos HEX, o el número de bits que ocupa el mensaje, o la frecuencia de transmisión de la señal, o el número de veces de que se repite la señal. El comando `sendNEC()` por tanto solo necesita el código HEX y el número de bits, al igual que el comando `sendSony()`.

```
1
2  IRsend irsend(D1); //an IR led is connected to GPIO5 pin D1
3
4  void setup()
5  {
6    irsend.begin();
7    Serial.begin(9600);
8  }
9
10 void loop() {
11   Serial.println("NEC");
12   irsend.sendNEC(0x00FFE01F, 36);
13   delay(2000);
14   Serial.println("Sony");
15   irsend.sendSony(0xa90, 12);
16   delay(2000);
17 }
```

El siguiente ejemplo muestra cómo la librería es capaz mediante sus funciones, preparar el circuito para recibir señales:

```
1
2  IRrecv irrecv(D2);
3
4  void  setup ( )
5  {
6    Serial.begin(9600);   // Status message will be sent to PC at 9600
         baud
7    irrecv.enableIRIn();  // Start the receiver
8  }
9
```

```
10 void  loop ( )
11 {
12   decode_results  results;        // Somewhere to store the results
13
14   if (irrecv.decode(&results)) {  // Grab an IR code
15     dumpInfo(&results);           // Output the results
16     dumpRaw(&results);            // Output the results in RAW format
17     dumpCode(&results);           // Output the results as source
           code
18     Serial.println("");           // Blank line between entries
19     irrecv.resume();              // Prepare for the next value
20   }
21 }
```

Se inicializa el puerto por el que entrarán los datos capturados por el circuito, después almacena el código IR, se muestra en fomato RAW, y si identifica el protocolo, además de declararlo envía su formato HEX.

También permite que el dispositivo active el modo de recepción detectando el código RAW emitido. En caso de que en su librería se encuentre el protocolo reconocido a esa señal, además de identificarlo y mostrarlo al usuario, también añadirá ese comando en formato hexadecimal. Visto la implementación física en el circuito, la librería es encargada de enviar el tren de impulsos a través del puerto al que se conectan los diodos de emisión infrarroja activando y desactivando estos de forma que generen las señales IR deseadas. En el caso del circuito de recepción, tiene capacidad de detectar los pulsos de la señal recibida en el puerto de entrada del ESP8266 conectado al puerto de salida del circuito receptor.

- **IRDaikinESP.h, IRMitsubishiAC.h y IRKelvinator.h:**

  Estas tres librerías se han colocado en un mismo apartado debido a su parecido funcionamiento. Se encargarán del envío de señales IR destinadas a dispositivos de aire acondicionado. Como ya se hizo mención anteriormente, este tipo de señales incluye todos los parámetros del estado actual de climatizador, luego es por ello que requieren otro tipo de librería que soporte dicha estructura. Contiene un vector en el que se almacena los parámetros del aire acondicionado y contiene funciones para enviarlos a través del puerto, y también permite consultarlos.

  En el siguiente ejemplo va a quedar claro como respecto a la librería antes vista para dispositivos diferentes a climatizadores, esta mantendrá una mayor complejidad. Concretamente representa el envío de una señal destinada a un aire acondicionado de la marca Daikin.

```
1
2 IRDaikinESP dakinir(D1);
3
4 void setup(){
```

```
5   dakinir.begin();
6   Serial.begin(115200);
7 }
8
9
10 void loop(){
11   Serial.println("Sending...");
12
13   // Set up what we want to send. See IRDaikinESP.cpp for all the
         options.
14   dakinir.on();
15   dakinir.setFan(1);
16   dakinir.setMode(DAIKIN_COOL);
17   dakinir.setTemp(25);
18   dakinir.setSwingVertical(0);
19   dakinir.setSwingHorizontal(0);
20
21   // Now send the IR signal.
22   dakinir.send();
23
24   delay(5000);
25 }
```

Igualmente va a tener un comienzo de inicialización del puerto a transmitir, y a continuación fija los parámetros que se desean configurar, y por último, se envía el vector completo con todos los parámetros, y aquellos que no hayan sido modificados se enviarán con sus valores por defecto.

Diagramas de secuencias de los mensajes MQTT

A fin de explicar los distintos modos que existen en cuanto a la comunicación de todos los componentes que forman parte de la solución, se ha ideado una serie de diagramas secuenciales que explican paso a paso el orden que se lleva a cabo en el envío de mensajes MQTT en la comunicación entre ellos. De esta manera se diferencian tres importantes modelos distintos de comunicación. El primero, como muestra la Figura 6.3, sería el caso del envío de mensajes MQTT correspondiente a la emisión de una señal infrarroja destinada a un dispositivo final diferente a un climatizador o aire acondicionado, correspondiente a la siguiente figura. Cabe mencionar que dichos envíos de mensajes MQTT se realizan bajo el mismo tópico.

El usuario envía la petición en MQTT, pasando por el *broker*, que se limita a su retransmisión, y el cliente ESP8266 recibe esa petición. En este caso no es necesario recibir ninguna contestación por mensaje MQTT del cliente ESP8266, puesto que solo se limita a recibir el mensaje, interpretarlo y enviar la señal infrarroja adecuada.

El segundo caso a diferenciar sería, como muestra la Figura 6.4, dentro del modo de emisión de señal infrarroja, el correspondiente a aquella señal destinada a un dispositivo aire acondicionado, que a diferencia del anterior,

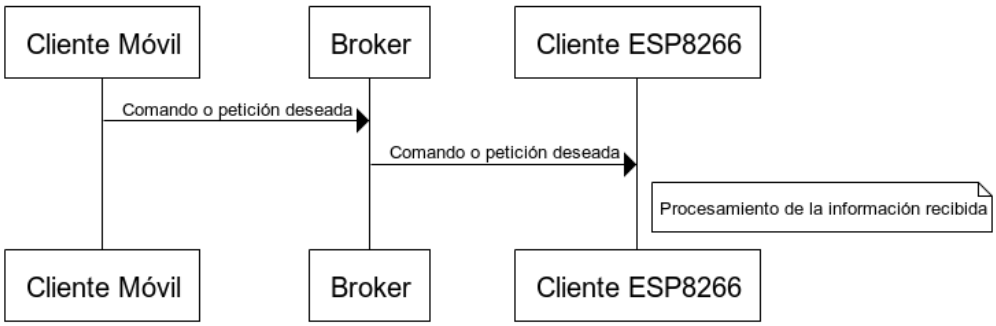

Figura 6.3: Diagrama de secuencias para emisión de comandos a dispositivos diferentes a climatizadores

el usuario sí recibe una contestación en el tópico con la declaración de cada uno de los parámetros actuales del climatizador. El cliente final envía la petición MQTT al *broker* para su retransmisión, donde el cliente ESP8266 la recibirá, interpretará, y por último, realizará una respuesta con otro mensaje MQTT que incluye todos los parámetros actuales que definen el estado del aire acondicionado. El envío de la declaración completa de los parámetros actuales se debe a que, al igual que los controles remoto de estos sistemas ofrecen una visualización del estado actual del aire acondicionado, es de suma importancia añadir esa información en el diseño, facilitando pues la configuración futura teniendo en cuenta todos los aspectos que se llevan a cabo en su utilización. Antes de la siguiente petición, el usuario deberá consultar el tópico para tener en cuenta los valores actuales que debe incluir en el mensaje MQTT. La siguiente figura lo aclara.

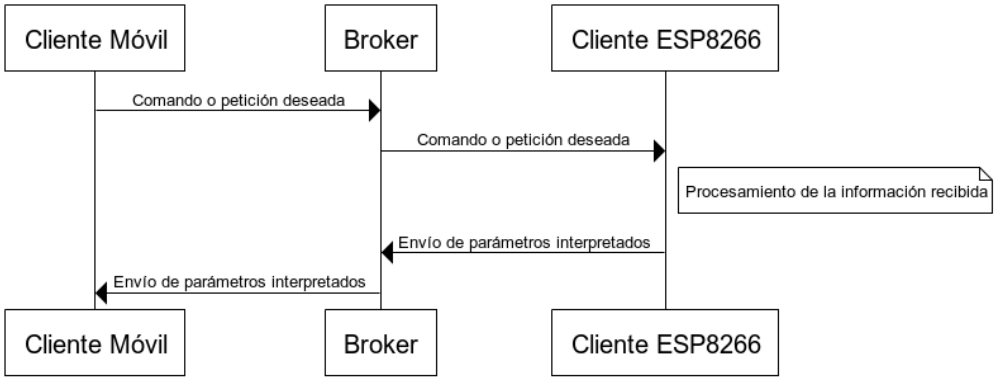

Figura 6.4: Diagrama de secuencias para emisión de comandos a aires acondicionados

Tanto, en el primer caso, como en este anterior y el que viene a continuación, toda la comunicación se realiza mediante el mismo tópico.

Por último, reside el caso de activación del modo de recepción. En este caso primeramente un cliente móvil será el encargado del envío de la petición MQTT, que pasa por el *broker*, para acabar en el cliente ESP8266. A partir de ahí el circuito activa su modo de recepción a la espera del comando infrarrojo, hasta que finalmente lo recibe y envía la información recabada mediante mensaje MQTT al *broker*. Como se ha mandado mediante el tópico habitual, todos los clientes suscritos serán capaces de visualizarlo, pero será de especial interés aquel cliente que tenga la capacidad de base de datos para su posterior adición a los posibles comandos a emitir posteriormente, pues la recepción a esa base de datos es el objetivo de que el ESPP8266 vuelva a enviar un mensaje MQTT como contestación.

Todo ello queda representado en la Figura 6.5.

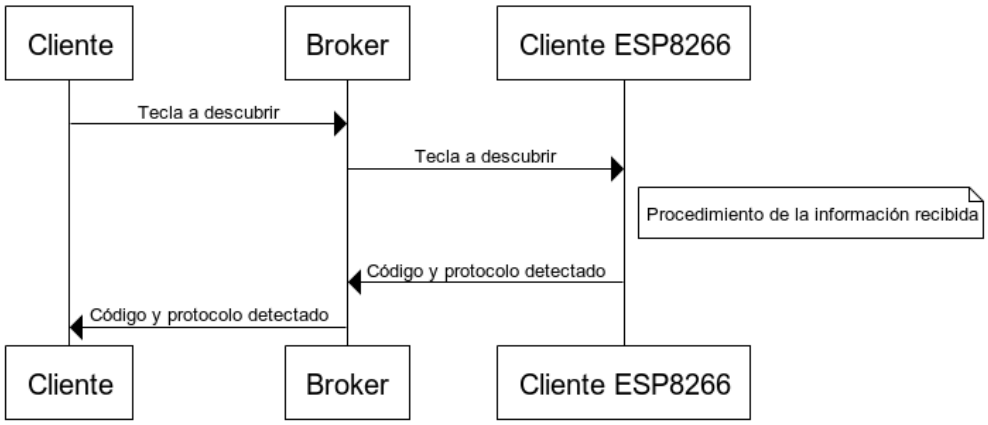

Figura 6.5: Diagrama de secuencias para la recepción de comandos

Diseño de los mensajes MQTT

La libertad que ofrece MQTT a la hora de diseñar un mensaje es una ventaja a aprovechar, luego en este subapartado se va a llevar a cabo la exposición del diseño de los diferentes mensajes que se van a crear y que el dispositivo pasarela interpretará en su recepción.

La capacidad del dispositivo pasarela está influenciada por este diseño, ya que todo aquello que se va a mostrar y describir a continuación será lo que estimule al ESP8266 a actuar de una forma u otra. La programación de dicho dispositivo está sometida a la comprensión de los mensajes entrantes MQTT, en los que se contienen la orden a ejercer. Técnicamente, dicha programación consiste en el parseo del mensaje MQTT recibido, lo cual no es más que la extracción de los diferentes campos existentes en el mensaje.

Adentrándose en el análisis del mensaje, la primera extracción del parseo

reside en los dos primeros campos del mensaje. Dichos campos son los de "Protocolo" y "Tipo" en segundo lugar. Dependiendo de como sean estos, los restantes campos tendrán un significado u otro, ya que los dos primeros campos instruyen al dispositivo pasarela qué tipo de parámetros se incluyen en los siguientes campos.

Como se ha mencionado en reiteradas veces, el circuito final del dispositivo consistirá una parte presidida por el circuito emisor, y la otra parte, que estará compuesta por los componentes del circuito receptor. Es por ello que el segundo campo llamado "Tipo" del mensaje MQTT, será el parámetro clave que indique al dispositivo que debe activarse su modo emisor, o en cambio su modo receptor. Debido a la gran usabilidad que se le da al mensaje MQTT, este no formará una estructura fija ni su tamaño (para un mismo tipo) será igual, aunque sí seguirá una misma estructura. Además en sus distintos campos podrán escribirse palabras o números, dependiendo de lo que corresponda. Los campos que forman parte del mensaje MQTT están delimitados por un espacio entre ellos.

Como norma general, se ha decidido que el primer campo del mensaje, será el protocolo, el cual estará formado por una palabra (en mayúscula) que indica el protocolo, y el segundo campo, el de "Tipo" será solamente un número. Los restantes campos (igualmente separados por espacios) tendrán un significado u otro dependiendo de estos dos primeros.

En primer lugar se desglosará el formato del mensaje MQTT que dará lugar a accionar el circuito receptor. La estructura de este tipo de mensaje que es ejecuta por el cliente final aparece en el Cuadro 6.1.

Protocolo	Tipo	Mando	Tecla

Cuadro 6.1: Estructura del mensaje MQTT para recepción

El cliente MQTT desde su dispositivo móvil mandará el mensaje MQTT al ESP8266 recibido mediante la comunicación Wi-Fi al tópico correspondiente. Esa orden es parseada primeramente por los dos primeros campos. Para el caso del circuito receptor, el valor del segundo campo "Tipo" será 9. El primer campo indicará el protocolo que el usuario piensa que puede pertenecer. El tercer campo es el de "Mando", que será interesante especificarlo para poder distinguir dispositivos del mismo protocolo. Esto es debido a que hay compañías como por ejemplo Samsung, que para diferentes dispositivos del hogar impulsados por infrarrojos contienen diferentes bases de datos para los comandos, por lo tanto, para dos dispositivos que pertenezcan a la misma marca, es habitual que los comandos que gestionan no sean idénticos o hagan la misma acción. Por último, el cuarto campo "Tecla" especifica la tecla que se va a escuchar.

A continuación se muestran algunos ejemplos de envío de este mensaje tal cual aparece en el Cuadro 6.2.

SAMSUNG	9	TV	ON
PHILIPS	9	Proyector1	MUTE

Cuadro 6.2: Estructura del mensaje MQTT para recepción

Estos serán los parámetros que el cliente deberá introducir para este tipo de mensaje. Es importante que el cliente se asegure de introducir bien los parámetros manualmente para identificar la tecla del dispositivo al que se quiere aprender. Su finalidad al fin y al cabo está en que sea la base de datos la que reciba claramente la información para poder almacenarla de manera exitosa. Aún así, ese sería el mensaje iniciado por el usuario final, aún falta enviar el código interceptado por la pulsación de la tecla. Esa información adicional la añadiría el chip ESP8266 y que enviaría al tópico mediante un mensaje con otro valor en el campo "Tipo" (cuyo valor no ejerce ninguna acción en el ESP8266 después de su lectura a su aparición al tópico). Por tanto, la estructura del nuevo mensaje final que crearía en este caso el chip ESP8266 con toda la información acerca del aprendizaje de un nuevo comando sería definida en el Cuadro 6.3.

Protocolo	Tipo	Mando	Tecla	RAW	HEX	Protocolo Detectado

Cuadro 6.3: Estructura del mensaje final MQTT para recepción

El primer campo contiene la misma información que el mensaje descrito anteriormente, aunque ahora el segundo campo ("Tipo") tendría un valor de 8, para distinguirlo del anterior y que el ESP8266 a su lectura en el tópico no produzca ninguna acción. Los dos siguientes campos de "Mando" y "Tecla" serían los elegidos manualmente por el cliente siendo idénticos al mensaje MQTT anterior, y en "RAW" aparecería todo el código RAW captado por el circuito en el momento de la pulsación de la tecla. El siguiente campo "HEX" es creado mediante el código RAW y se proporciona en el mensaje. Por último, el campo existente e el de "Protocolo Detectado" mediante el cual se visualiza aquel protocolo (con el nombre en mayúscula) que el circuito ha identificado mediante sus librerías. En caso de ser un nuevo protocolo que no consta en sus librerías, aparecerá "UNKNOWN", como no identificado. Cuando sí se detecta un protocolo pero el usuario ha marcado en el primer campo un protocolo diferente, se deberá dar prioridad al protocolo indicado en el último campo pues será el que las librerías del chip han identificado.

Un ejemplo que podemos obtener se representa en el Cuadro 6.4

RTLSDR	8	Mando	ON	Código RAW	16757325	NEC

Cuadro 6.4: Ejemplo de mensaje final MQTT para recepción

Donde en el campo de código RAW se encontraría un código como este: [9050,4500, 600,550, 600,550, 650,550, 600,550, 600,550, 650,550, 600,550, 600,550, 650,1650, 600,1650, 600,1650, 650,1650, 600,1650, 600,1700, 650,1650, 600,1650, 600,1700, 650,550, 600,1700, 650,1650, 600,550, 650,550, 600,1650, 650,550, 600,600, 600,1650, 600,550, 600,550, 650,1650, 600,1650, 600,600, 600,1650, 600,1588600, 9050,4500, 600,550, 600,550, 650,550, 600,550, 600,550, 650,550, 600,550, 600,550, 650,1650, 600,1650, 600,1650, 650,1650, 600,1700, 600,1650, 650].

Es en este mensaje donde se incluye toda la información para que el gestor de la base de datos pueda recoger todo aquello que necesita para su almacenamiento, y que conocerá de antemano que el destino de ese mensaje es él por la creación de un único campo "Tipo" exclusivo.

La segunda parte del diseño de los mensajes MQTT es más extensa y se va a definir a continuación, pero implica un diseño más amplio debido a la enorme cantidad que ofrecen las librerías para los distintos tipo de envío de señales infrarrojas.

La clave residirá como en el caso anterior, en los dos primeros campos del mensaje MQTT. Se diferenciarán según el campo "Tipo". Para el primero de los casos, si "Tipo" equivale a 0, el mensaje a enviar será un código RAW, cuya estructura seguirá el patrón del Cuadro 6.5.

Protocolo	Tipo	RAW	Frecuencia (kHz)

Cuadro 6.5: Estructura del mensaje final MQTT para emisión de código RAW

Donde todos los anteriores campos tienen la misma definición que los mensajes descritos en párrafos atrás, y el que falta por comentar, el campo de "Frecuencia (kHz)" es aquel campo donde el usuario indica la frecuencia de transmisión de la señal infrarroja (típicamente es 38kHz). Luego un pequeño ejemplo aparcería en el Cuadro 6.6.

KNEISSEL	0	Código RAW	38

Cuadro 6.6: Ejemplo de mensaje final MQTT para emisión de código RAW

Donde el código en el que aparece "código RAW" estaría escrito una

cadena de números tal como el anterior caso. Y el último campo donde aparece escrito "38" representa la frecuencia en kHz (38kHz). Este tipo incluye una excepción, y es que en el caso de que el Protocolo sea "GC", no será necesario especificar la frecuencia de transmisión, pues ya la incluye en la librería.

El siguiente tipo ha sido creado para el envío de señales infrarrojas mediante comandos HEX, luego el valor del mismo será de 1. La estructura que sigue se muestra en el Cuadro 6.7.

Protocolo	Tipo	Código HEX	Número de bits	Número de repeticiones

Cuadro 6.7: Estructura del mensaje MQTT para emisión de código HEX

Esta estructura solo se llevará a cabo si en el primer campo de "Protocolo" está escrito alguno de los siguientes: NEC, SONY, JVC, SHERWOOD. El campo "Número de bits" indica al chip el tamaño del mensaje infrarrojo de ese protocolo, y el número de repeticiones indica la cantidad de veces que hay que repetir el comando, simulando una pulsación continua de una tecla del control remoto. Este último campo no es obligatorio en todos los anteriores protocolos, por ejemplo para NEC, si este valor se omite, por defecto no lo repetirá, SONY en caso de omisión lo repite una sola vez y en SHERWOOD, dado su campo ausente, se repetirá una sola vez pode defecto. Un ejemplo de envío de este tipo de mensaje se ha plasmado en el Cuadro 6.8.

SONY	1	0xa90	12	2

Cuadro 6.8: Ejemplo de mensaje MQTT para emisión de código HEX

Otro grupo de protocolos con los que también es posible utilizar este tipo de envío son: COOLIX, WHYNTER, LG, RC5, RC6, DISH, SHARP, SAMSUNG, DENON. La diferencia que hay respecto de los anteriores es que en la estructura del mensaje no se incluirá el último campo, es decir, solo se envía una sola vez, sin repeticiones posibles, como muestra el Cuadro 6.9.

Protocolo	Tipo	Código HEX	Número de bits

Cuadro 6.9: Estructura del otro mensaje MQTT con código HEX

Si el campo que aparece en "Protocolo" es PANASONIC, excepcionalmente la estructura contendrá los parámetros del Cuadro 6.10. Donde el campo "Dirección" refleja aquella parte de la señal infrarroja que incluye la información relativa a qué dispositivo va dirigida, en formato decimal.

Protocolo	Tipo	Código HEX	Dirección

Cuadro 6.10: Estructura del mensaje MQTT con código HEX en protocolo PANASONIC

Todos estos protocolos anteriormente mencionados son los únicos capacitados para usar ese tipo de comando (HEX). Cualquier otro protocolo que se intente mandar con un comando HEX no será enviado por el circuito emisor.

A continuación damos paso a un caso en concreto, cuya protocolo es SHARP. Dicho protocolo ofrece la posibilidad de mandar la señal infrarroja solo introduciendo sus campos "*Address*" y "*Command*". En concreto, esto será posible mediante la implementación de otro valor en el campo de "Tipo", teniendo un valor de 2. La estructura sería la acordada en el Cuadro 6.11. Estos dos nuevos campos contienen el comando a ejecutar (campo "Comando") y el dispositivo al que va dirigido (campo "Dirección").

Protocolo	Tipo	Dirección	Comando

Cuadro 6.11: Estructura del mensaje MQTT en protocolo SHARP

Cabe mencionar que los dos campos serán colocados en formato decimal, como aparece en el Cuadro 6.12.

SHARP	2	4	104

Cuadro 6.12: Ejemplo de mensaje MQTT en protocolo SHARP

Por último, como adición también es posible el envío de señales infrarrojas a aires acondicionados como DAIKI, KELVINATOR y MITSUBISHI, mediante un vector *char* donde se incluyen todos sus parámetros, pero de este método solo se hará una mención a su estructura, puesto que existe otra librería más detallada y exclusiva acerca de estos tres protocolos. El "Tipo" creado tiene valor de 3 y en cualquier caso, la estructura de mensaje MQTT en caso de poseer el vector *char* se muestra en el Cuadro 6.13. Al igual que casos previos, si en el campo "Protocolo" no aparece cualquiera de los tres mencionados, será imposible su envío posterior.

Realmente, la comunicación habitual que se va a realizar en caso de que los dispositivos finales sean aires acondicionados, y concretamente para los protocolos de DAIKIN, MITSUBISHI y KELVINATOR será de la manera que se va a desglosar abajo, y será mediante el "Tipo" con valor 4. La

Protocolo	Tipo	Vector char

Cuadro 6.13: Estructura del mensaje MQTT en caso de posesión del vector char

estructura de los mensajes será idéntica a los tres protocolos como indica el Cuadro 6.14.

Protocolo	Tipo	Orden	Valor

Cuadro 6.14: Estructura del mensaje MQTT para DAIKIN

Puesto que la diferencia entre ellos reside en el significado de los distintos valores de los campos "Orden" y "Valor", pasamos a detallar el protocolo DAIKIN.

El nuevo campo de "Orden" tendrá un amplio rango de modalidades las cuales se describirán en el Cuadro 6.15 para su rápida comprensión.

Orden	Significado
0	POWER
1	AUX
2	TEMP
3	FAN
4	MODE
5	SWING VERTICAL
6	SWING HORIZONTAL

Cuadro 6.15: Distintos significados del campo "Orden" en DAIKIN

Todos estos parámetros son modificables en climatizadores cotidianos. Botones como: POWER, para apagarlo o encenderlo, AUX, que ajusta el ruido del aire acondicionado, TEMP, para regular la temperatura, FAN que fija la velocidad, MODE que define el modo, y los SWINGS que ajustan la dirección del aire.

El campo "Valor" definirá el estado del campo "Orden" puesto que este último indica el parámetro a cambiar, y el de "Valor" indica el cómo. Por ello en el Cuadro 6.16 se explican todos los casos.

Finalmente, se van a mostrar dos ejemplos acerca de este protocolo (véase Cuadro 6.17.

La estructura del caso del protocolo MITSUBISHI sufre unas variaciones

Orden	Valor	Significado
POWER	0	OFF
	1	ON
AUX	0	POWERFULL
	1	SILENT
TEMP	{15,...,30}	Temperatura
FAN	0	AUTOMÁTICO
	1-5	VELOCIDAD
MODE	0	COOL
	1	HEAT
	2	FAN
	3	AUTO
	4	DRY
SWING VERTICAL	0	NO
	1	SI
SWING HORIZONTAL	0	NO
	1	SI

Cuadro 6.16: Distintos significados del campo “Valor” en DAIKIN

Protocolo	Tipo	Orden	Valor
DAIKIN	4	0	1
DAIKIN	4	2	20

Cuadro 6.17: Ejemplo de mensaje MQTT para DAIKIN

respecto a DAIKIN ínfimas, pero que requieren de la creación de una nueva tabla para el campo de “Orden” (véase Cuadro 6.18). El valor VANE es un sinónimo del valor SWING del protocolo DAIKIN.

Orden	Significado
0	POWER
1	VANE
2	TEMP
3	FAN
4	MODE

Cuadro 6.18: Distintos significados del campo “Orden” en MITSUBISHI

Y a continuación se detallan los diferentes valores del campo “Valor” en el correspondiente Cuadro 6.19.

Orden	Valor	Significado
POWER	0	OFF
	1	ON
VANE	0	AUTO
	1	AUTO-MOVE
TEMP	{15,...,30}	Temperatura
FAN	0	AUTO
	1	SILENT
MODE	0	AUTO
	1	COOL
	2	DRY
	3	HEAT

Cuadro 6.19: Distintos significados del campo "Valor" en MITSUBISHI

Por último, y más completo, queda el protocolo KELVINATOR, el cual, la estructura del mensaje MQTT sigue siendo idéntica pero discrepa en los significados de los valores de los campos "Orden" y "Valor", luego el Cuadro 6.20 indicará los significados del campo "Orden".

Orden	Significado
0	POWER
1	QUIET
2	TEMP
3	FAN
4	MODE
5	SWING VERTICAL
6	SWING HORIZONTAL
7	ION-FILTER
8	LIGHT
9	TURBO

Cuadro 6.20: Distintos significados del campo "Orden" en KELVINATOR

Cabe mencionar que el valor de QUIET es otro sinónimo al valor de AUX en DAIKIN, y el ION-FILTER es un valor que mediante su activación purifica el aire.

El correspondiente Cuadro 6.21 explica los diferentes significados de los valores del campo "Valor".

Hay que entender *TRUE* y *FALSE* como ON y OFF. Teniendo en cuenta todo lo anterior, ya es posible mandar correctamente cualquier comando a

Orden	Valor	Significado
POWER	0	OFF
	1	ON
QUIET	0	FALSE
	1	TRUE
TEMP	{15,...,30}	Temperatura
FAN	0	AUTOMÁTICO
	1-5	VELOCIDAD
MODE	0	AUTO
	1	COOL
	2	DRY
	3	FAN
	4	HEAT
SWING VERTICAL	0	FALSE
	1	TRUE
SWING HORIZONTAL	0	FALSE
	1	TRUE
ION-FILTER	0	FALSE
	1	TRUE
LIGHT	0	FALSE
	1	TRUE
TURBO	0	FALSE
	1	TRUE

Cuadro 6.21: Distintos significados del campo “Valor” en KELVINATOR

un aire acondicionado. Era necesario crear un tipo aparte ya que este tipo de comandos difieren en gran medida al resto de dispositivos. Para una misma pulsación de una tecla nunca se manda el mismo mensaje. Esto se debe a que ese mensaje no contiene solo la acción a ejecutar sino el resto de parámetros que conforman el estado del aire acondicionado.

Por último, en el caso de estas tres librerías, como se ha hecho hincapié en la explicación de los diagramas de secuencias, será aquí donde el chip ESP8266 realice un envío de mensaje MQTT publicando en el tópico la tecla pulsada con el resto de parámetros del estado del climatizador. Este nuevo mensaje tendrá un valor de “Tipo” de 5, el cual el ESP8266 después de leerlo, no realizará ninguna acción posterior, puesto que su objetivo es solo informativo. Tanto la estructura como un ejemplo de mensaje MQTT informativo se muestra en el Cuadro 6.22. Este ejemplo está diseñado para el caso de DAIKIN. Si enviásemos un comando de cualquiera de los otros dos protocolos, los mensajes informativos MQTT contendrían sus correspondientes campos.

Protocol.	Tipo	Power	Aux	Temp	Fan	Mode	SwingVer	SwingHor
DAIKIN	5	ON	SILENT	20	3	FAN	ON	OFF

Cuadro 6.22: Ejemplo de mensaje MQTT informativo para DAIKIN

Implementación física del circuito

Uno de las últimas secciones que conforman este capítulo es la implementación física del circuito que implica aquellos componentes electrónicos que han permitido la transmisión de las señales infrarrojas y la recepción. No se ha requerido un amplio abanico de componentes ni complejos, su diseño es muy sencillo, de hecho, es posible encontrar varios diseños posibles para conseguir transmitir y recibir señales.

A continuación se va a explicar un posible diseño de un circuito emisor de señales infrarrojas. Dicho circuito es el más simple de todos pues solo consta de dos LEDs infrarrojos en serie, conectados al *NodeMCU v1.0* o al *Wemos D1 Mini* (dependiendo del módulo que se utilice). La patilla "-" de los LEDs deberá apuntar a GND, de manera que uno de los dos LEDs tenga dicha patilla en tierra, y las patillas "+" deberán apuntar al puerto del módulo que será el módulo por el que se envíen las señales digitales.

En la Figura 6.6 se ha representado para los dos módulos. Este sería el circuito más simple. El puerto que se ha elegido para enviar las señales es el puerto D8, que también ha sido declarado en la programación con el software Arduino. Como el puerto tiene una tensión de salida de 3.3V cuando el flanco está en alto, y los LEDs solo necesitan unos 1,5V aproximadamente cada uno, podrán conducir a la vez quitando el cortocircuito. Para su correcto funcionamiento solo faltaría alimentarlo con el puerto microUSB que incluyen estos módulos.

Otra alternativa a este circuito simple es la de poner los diodos en paralelo, dividiéndose la corriente. La tensión se sigue manteniendo, por lo que pueden conducir, y es aplicable a los dos módulos. Como es exactamente igual, se mostrará solo en uno de los módulos (con *Wemos D1 Mini*), mediante la Figura 6.7.

Los puertos asignados han sido los mismos que en circuito simple (D8 y GND). la única diferencia que reside en los dos circuitos es en la colocación de los LEDs.

Hay otros diseños de circuitos que son capaces de transmitir a mayor potencia, y por lo tanto, mayor longitud, en el que para ello se necesitaría una tensión de 5V [32], que propiciará una mayor corriente en el circuito. *NodeMCU v1.0* tiene una desventaja respecto de *Wemos D1 Mini* en este caso ya que *Wemos* tiene un puerto con una tensión de salida de 5V, pero

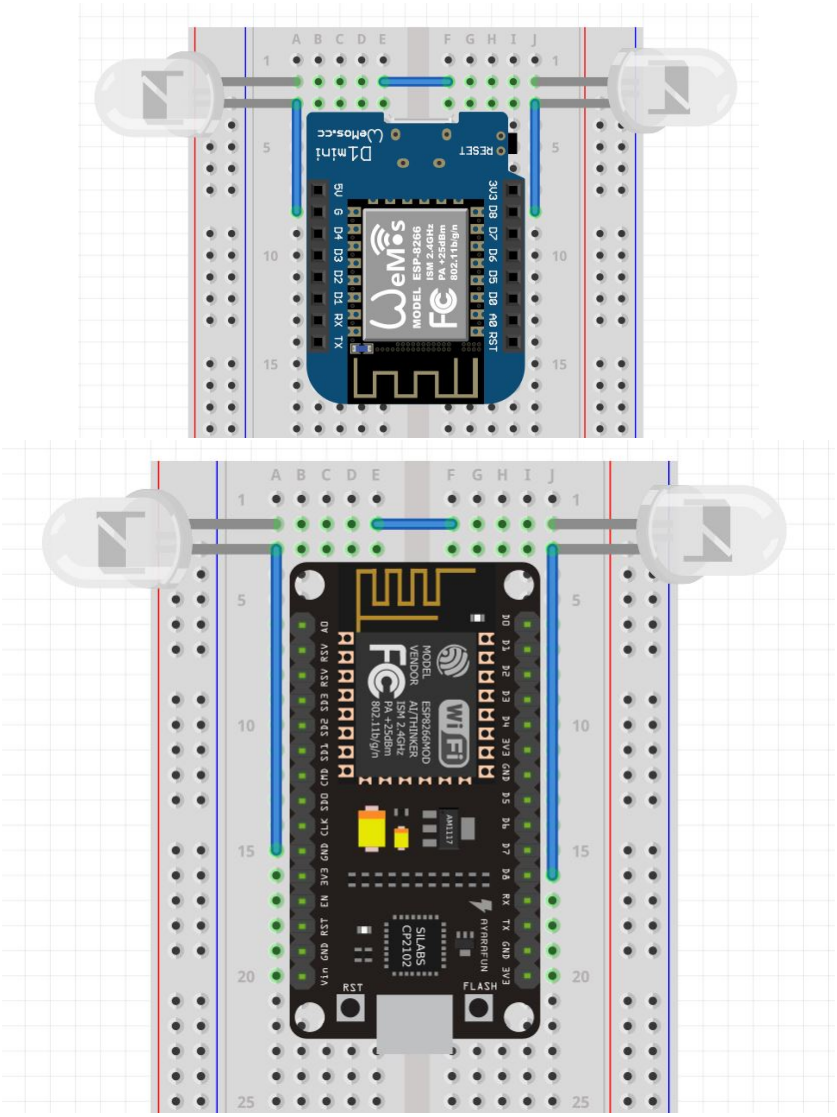

Figura 6.6: Posibles circuitos simples emisores

NodeMCU no, luego para este se requerirá de un circuito externo que lo alimente de 5V. El circuito para *NodeMCU* será necesario para una alimentación de 5V para un incremento de la potencia de emisión, representado en la Figura 6.8.

Como era habitual, la alimentación de los módulos se realizaba mediante el cable microUSB, pero a falta de una salida de 5V, se ha requerido su alimentación mediante un regulador de tensión de 5V externo (MB102), que irá conectado al puerto "Vi" del *NodeMCU*. Será desde ese puerto del que el módulo será alimentado para ofrecer la corriente necesaria al resto del

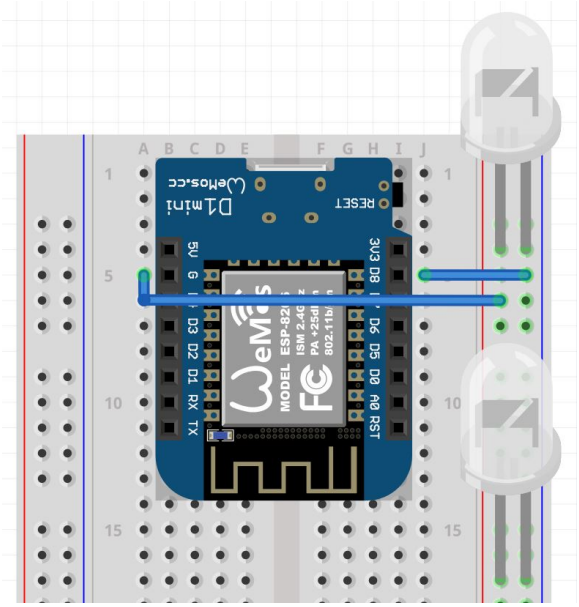

Figura 6.7: Alternativa de circuito simple emisor

circuito. Además de los dos diodos IR en serie, también será necesario la utilización de un transistor, concretamente el NPN 2N2222 [34]. Sería un circuito mucho más potente en la emisión de señales infrarrojas permitiendo más alcance que en los casos de circuitos anteriores.

Por último, el circuito de recepción estaría formado por siguientes componentes descritos en el esquemático de la Figura 6.9. Se ha asignado un puerto de diferente donde actuará de entrada a los datos y que irá directamente conectado el diodo IR receptor. El receptor IR es el modelo TSOP4838 [33] con tres anclajes. El primero va al puerto comentado, el de en medio se destina al puerto de 3.3V y el último irá directo a GND. Además se ha complementado con un LED luminoso. El objetivo de este LED consiste en poder advertir cuando se encuentra en modo recepción. Cuando el diodo está encendido, el cliente sabe que es momento de mandar la señal infrarroja a aprender, puesto que el circuito la captará. Una vez enviada la señal infrarroja, automáticamente el diodo se apagará, indicando que la señal IR ha sido recibida.

Antes de finalizar la implementación física, es conveniente recordar la alimentación del dispositivo pasarela, pues es un aspecto que influye en su diseño y ubicación. Hay dos posible maneras de alimentarlo, una sería mediante baterías y la otra mediante el cable microUSB por el puerto serie del módulo, que tanto *NodeMCU v1.0* como *Wemos D1 Mini* disponen.

Sin embargo, la vía más adecuada para su alimentación reside en el

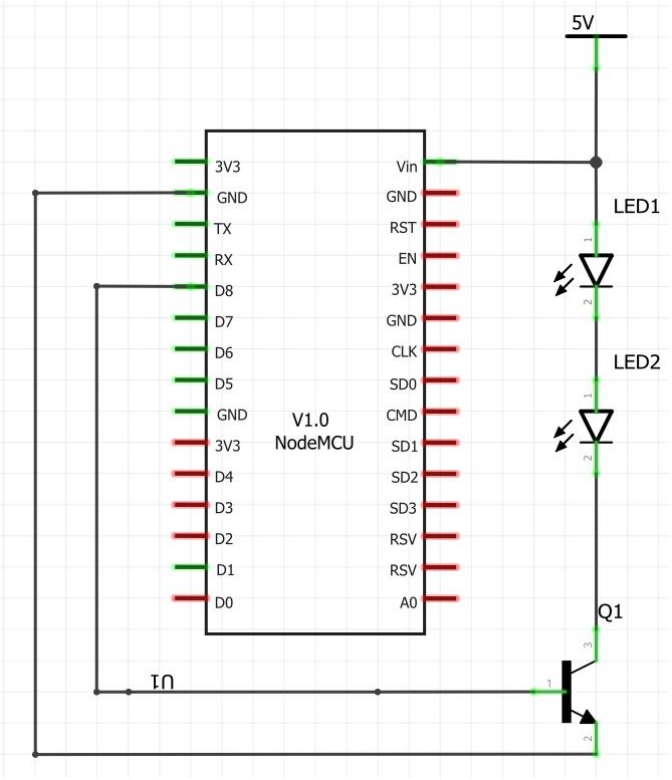

Figura 6.8: Circuito de emisión con alimentación de 5V

uso del cable microUSB antes que las baterías. El consumo de energía mediante baterías implica programar el dispositivo pasarela en modo ahorro de energía (tipo *deep sleep*). Cada cierto intervalo de tiempo asignado previamente se despertaría para leer el tópico y comprobar el último mensaje que hay. De este modo la batería prolongaría su durabilidad, en cualquier otro caso su consumo aumentaría considerablemente. Desgraciadamente, este tipo de programación provocaría dos inconvenientes. El primero sería por supuesto que no ofrecería un servicio interactivo, pues solo despierta cada cierto tiempo, ya sea cada treinta segundos, un minuto o dos. Otro gran inconveniente que presenta es que, en un tópico de MQTT sólo se almacena el último mensaje, luego, si se intenta mandar dos comandos consecutivos en un espacio de tiempo inferior a su tiempo de *sleep*, a la hora de activarse de nuevo leerá el último, sin conocer en ningún momento el primer mensaje. El usuario se vería limitado a poder ejecutar solo un comando por tiempo de *sleep*. Esta acción resulta menos pesada si se hace desde fuera, cuando por ejemplo se intenta accionar un aire acondicionado para su pleno rendimiento a la llegada del hogar, pero en caso de estar dentro de la vivienda resultaría demasiado comprometedor el simple ejemplo de tener que esperar treinta segundos o un minuto solo para cambiar el canal de televisión.

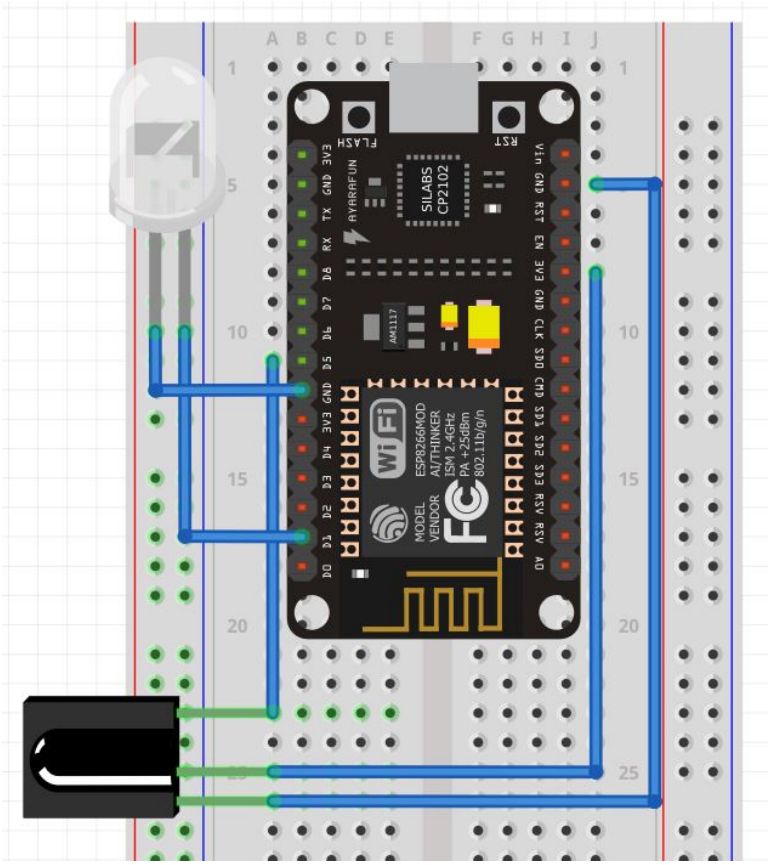

Figura 6.9: Circuito de recepción

Si el sistema estuviese alimentado por cable, no sería necesario programarlo en modo ahorro de energía, sino que siempre podría estar leyendo constantemente el tópico, ofreciendo un servicio en tiempo real sin la pérdida de ningún mensaje y su uso sería igual de válido tanto para ejecutar sus comandos dentro como fuera de casa.

Implementación del servicio MQTT

Por último, para acabar este capítulo solo falta explicar la implementación del servicio MQTT, cuya actuación será fundamental para la comunicación mediante mensajes MQTT.

Se ha diseñado un servidor privado, ya que su dirección IP es privada, por tanto, ningún usuario ajeno al hogar podría conectarse a menos que forme parte de la red Wi-Fi. Concretamente es un servidor Mosquitto, previamente descargado e instalado [39] de su correspodiente web oficial para Windows (aunque también hay una amplia gamas de sistemas operativos

disponibles) en nuestro ordenador. El tópico creado para esta ocasión será denominado como el nombre de identificación del chip ESP8266. De esta manera nadie podrá subscribirse fácilmente al tópico y publicar en él. Cada módulo contiene su propia identificación independientemente. Al no ser MQTT seguro, no será necesario escribir los campos de usuario y contraseña en aquellos casos en los que se pregunte por ellos. Su instalación es prácticamente automática, por lo que no será necesaria un tutorial con los pasos a seguir. Únicamente recordar que una vez instalado, es necesario ajustar la configuración desde la sección "Servicios" de Windows, tal cual aparece en la Figura 6.10.

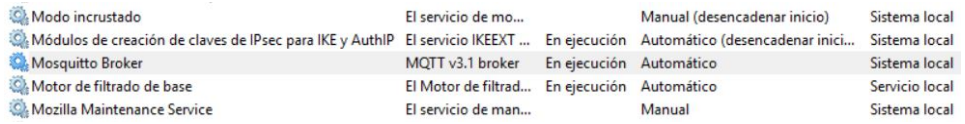

Figura 6.10: Configuración en "Servicios" de Windows

Una vez seguido esos pasos, por último solo queda comprobar que el puerto 1883 se encuentre en escucha. Esto se realizará mediante el "cmd" de Windows, escribiendo el comando: `netstat -an` (véase Figura 6.11).

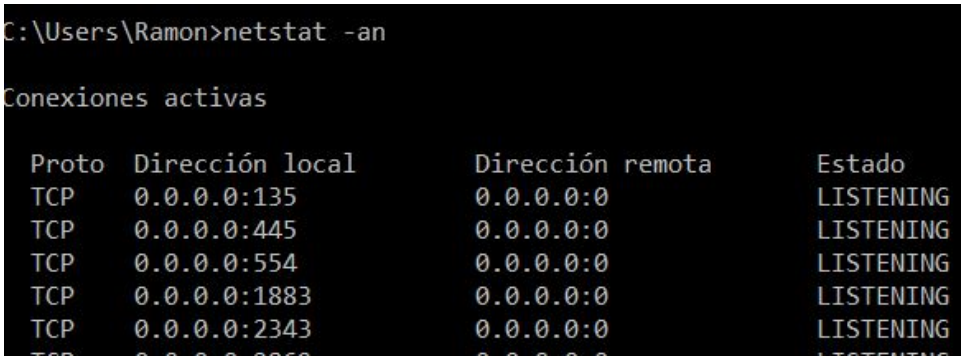

Figura 6.11: Comando en "cmd" de Windows

El móvil actuará como cliente MQTT, mediante una aplicación con la que podrá mandar los mensajes MQTT. Dicha aplicación se llama "Mqtt" y permite tanto suscribirse como publicar en un tópico determinado. El otro cliente MQTT será el ESP8266, que actuará de una manera u otra dependiendo del contenido de los mensajes que anteriormente se ha diseñado.

Con todo esto, ya estaría todo el diseño preparado para la fase de pruebas y validación.

Capítulo 7

Pruebas realizadas y validación

En dicho capítulo se pondrá a prueba la implementación de la solución sometiéndola a diferentes fases que determinan la calidad del resultado describiendo el proceso que se llevará a cabo.

Durante este capítulo se irán recogiendo todos los datos almacenados que se capten en las diferentes pruebas. Por supuesto era fundamental hacer una comprobación al dispositivo pasarela en un testeo que demostrase que su programación ofrece los resultados deseados, y que el circuito completo cumple sus funciones.

Transformación a comando HEX de una señal infrarroja

En primer lugar, se va a realizar un análisis de las señales infrarrojas que los controles remotos emiten. El dispositivo receptor que se va a utilizar es el RTL_SDR *Dongle*. Este dispositivo usado como SDR permite interceptar señales, entre ellas, las infrarrojas, mediante software preparado concienzudamente para ello. Esa captación se produce porque incluye un sensor IR muy parecido al TSP382. Dicho software puede devolver los datos de intervalos de los pulsos IR. El *dongle* RTL_SDR es comúnmente utilizado para observar transmisiones de TV. Contiene un mando infrarrojo con el que poder comunicarse, y mediante una máquina de Linux, se instala el *driver* del RTL_SDR llamado dvb_usb_rtl28xxu que permite la recepción de infrarrojos.

Esta prueba va a precisar de un control remoto para una televisión de la marca SAMSUNG, que presionando la tecla ON se analizará la señal recibida y mediante una base de datos de infrarrojos se comprobará si el comando es el correcto, y por lo tanto, sabremos identificarlo. Si el RTL_SDR ya está preparado[40], se introduce por el puerto USB del portátil y se sincroniza con la máquina virtual. Una vez esté todo listo, solamente hay que abrir el

Terminal de Linux y escribir el siguiente comando:

```
wimunet@wimunet-VirtualBox:~$ rtl_ir -d 0 -t
Found 1 device(s):
  0:  Realtek, RTL2838UHIDIR, SN: 00000001

Using device 0: Generic RTL2832U OEM
Found Rafael Micro R820T tuner
pulse 1, duration 1840 usec
pulse 0, duration 1780 usec
pulse 1, duration 240 usec
pulse 0, duration 660 usec
pulse 1, duration 240 usec
pulse 0, duration 660 usec
pulse 1, duration 240 usec
pulse 0, duration 660 usec
pulse 1, duration 240 usec
pulse 0, duration 220 usec
pulse 1, duration 220 usec
pulse 0, duration 220 usec
pulse 1, duration 240 usec
pulse 0, duration 220 usec
pulse 1, duration 220 usec
pulse 0, duration 220 usec
pulse 1, duration 240 usec
pulse 0, duration 220 usec
pulse 1, duration 220 usec
pulse 0, duration 680 usec
pulse 1, duration 220 usec
pulse 0, duration 680 usec
pulse 1, duration 200 usec
pulse 0, duration 680 usec
pulse 1, duration 220 usec
pulse 0, duration 240 usec
pulse 1, duration 220 usec
pulse 0, duration 220 usec
pulse 1, duration 220 usec
pulse 0, duration 240 usec
pulse 1, duration 220 usec
pulse 0, duration 220 usec
pulse 1, duration 220 usec
pulse 0, duration 240 usec
pulse 1, duration 200 usec
pulse 0, duration 240 usec
pulse 1, duration 220 usec
pulse 0, duration 680 usec
pulse 1, duration 220 usec
pulse 0, duration 240 usec
pulse 1, duration 200 usec
pulse 0, duration 240 usec
pulse 1, duration 220 usec
pulse 0, duration 240 usec
pulse 1, duration 200 usec
pulse 0, duration 240 usec
pulse 1, duration 220 usec
pulse 0, duration 220 usec
pulse 1, duration 220 usec
pulse 0, duration 240 usec
pulse 1, duration 220 usec
pulse 0, duration 680 usec
pulse 1, duration 200 usec
```

```
pulse 0, duration 240 usec
pulse 1, duration 220 usec
pulse 0, duration 680 usec
pulse 1, duration 220 usec
pulse 0, duration 680 usec
pulse 1, duration 220 usec
pulse 0, duration 680 usec
pulse 1, duration 220 usec
pulse 0, duration 680 usec
pulse 1, duration 220 usec
pulse 0, duration 680 usec
pulse 1, duration 200 usec
pulse 0, duration 700 usec
pulse 1, duration 200 usec
pulse 0, duration 2540 usec
pulse 0, duration 1280 usec
```

Todo el resultado que muestra el RTL_SDR representa la duración en el tiempo que tiene cada uno de los flancos de la seña infrarroja, que al fin y al cabo es la definición de un código RAW. Cabe mencionar que RTL_SDR aplica un factor de reducción del 2,5 aproximadamente al tiempo de los pulsos de subida y bajada, luego los intervalos reales serían mayores. La Figura 7.1 muestra una lista con los intervalos que requiere cada valor.

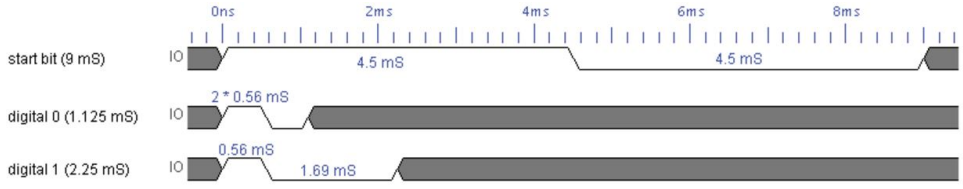

Figura 7.1: Lista de valores para protocolo Samsung

Observándolos detenidamente (y teniendo en cuenta los datos escalados) se aprecia la similitud entre los resultados. Calculando los "0"s y "1"s lógicos a partir del primer "240" que sería desde donde se empieza (los dos primeros valores solo representan el *start bit*) descubrimos la siguiente secuencia:

1 1 1 0 0 0 0 0 1 1 1 0 0 0 0 0 0 1 0 0 0 0 0 0 1 0 1 1 1 1 1 1

Para convertirlo a un comando HEX para faltaría agruparlas de 4 en 4, por lo que finalmente se obteiene:

0xE0E40BF

Finalmente se conoce el comando HEX, de 32 bits. La última parte de los datos recabados del RTL_SDR estaba formada de un "0" muy largo, el cual se ha asociado a un posible ruido final.

Para poder simular el envío de este comando mediante el chip ESP8266 que se ha programado con el software Arduino, tendría que enviarse mediante el siguiente ejemplo:

```
IRsend irsend;

void setup(){
  pinMode (3, OUTPUT);  //output as used in library
}
void loop() {
  irsend.sendSAMSUNG(0xE0E040BF,32);
  delay (5000);
}
```

Finalmente, mediante bases de datos en internet [41] [42] [43] [44] se ha comprobado el análisis anterior y se ha verificado que el comando HEX calculado coincide exactamente con el comando que debería enviarse en este protocolo para ese dispositivo final.

Simulación de envío de señal infrarroja

A continuación se va a poner a prueba la solución implantada mediante el envío de una señal infrarroja. En concreto primero se analizará el correcto funcionamiento en la emisión de señales infrarrojas.

El servidor Mosquitto en el ordenador debe estar preparado en primer lugar, para permitir la comunicación de mensajes MQTT, tal cual se detalló al final del capítulo anterior.

El segundo paso será la de preparar el dispositivo pasarela, el cual debe configurarse mediante el móvil del usuario para su conexión a la red Wi-Fi del hogar. Por lo cual, se conecta el circuito montado al ordenador, para poder descubrir el nombre del tópico mediante el Monitor Serie de Arduino. Esta preparación con Arduino solo será necesaria la primera vez que se configure a la red Wi-Fi, pues una vez hecho los parámetros se guardan para futuros casos en los que se desconecte el circuito de la luz.

Además el diseño final del circuito constará de la parte del circuito que permite la emisión de señales infrarrojas y la recepción. Concretamente para este diseño se ha utilizado el *NodeMCU v1.0* con una mini placa con cableado conectado por debajo del mismo (pues la altura del *NodeMCU* en la placa lo permite) que simplifica enormemente el tamaño de su diseño (véase Figura 7.2). Cabe mencionar que el circuito emisor utilizado será el primer circuito descrito en el capítulo anterior, por lo que se utilizará una entrada de 3.3V, reduciéndose el diseño considerablemente.

Una vez conectado el dispositivo, y abierto el Monitor Serie aparecerá lo que se refleja en la Figura 7.3. En este punto, se visualiza arriba de la imagen el nombre del tópico que coincidirá exactamente con el número de identificación del tópico y lo siguiente serán los pasos que sigue el dispositivo para la creación del punto de acceso.

Una vez creado dicho punto, será el cliente mediante el terminal móvil el

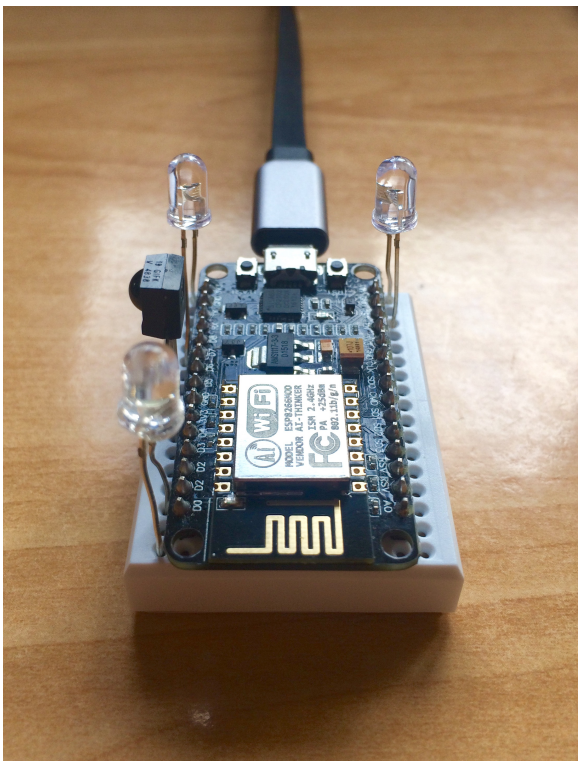

Figura 7.2: Circuito final

que deberá indicar los parámetros al ESP8266 de la red Wi-Fi a la que debe de conectarse, junto a la dirección IP y el puerto del servidor MQTT. En el móvil se escanea todos lo puntos de acceso Wi-Fi que hay, y se selecciona el punto de acceso que tenga el nombre del número de identificación del chip (esto se deducirá rápidamente al empezar con las letras de "ESP", seguido de una cadena de números). Hecho esto la red abrirá una ventana emergente con una siguiente interfaz (véase Figura 7.4). Dicha imagen muestra la ventana, con una dirección IP, arriba del todo, por defecto a la que en caso de que la ventana emergente no aparezca automáticamente por defecto, siempre se podrá acceder escribiéndose en el navegador de internet del móvil.

Tanto desde la primera opción en azul que aparece como desde la segunda, se podrán escribir todos los parámetros anteriormente comentados (en el primer caso antes de introducirlos escanea todas las redes Wi-Fi para advertir de las actualmente disponibles y en el segundo caso se omite esa acción), para finalmente dar lugar a la Figura 7.5.

Una vez se presione la ventana *save* dará lugar a la notificación en el terminal que indica que los parámetros han sido satisfactoriamente introducidos (véase Figura 7.6).

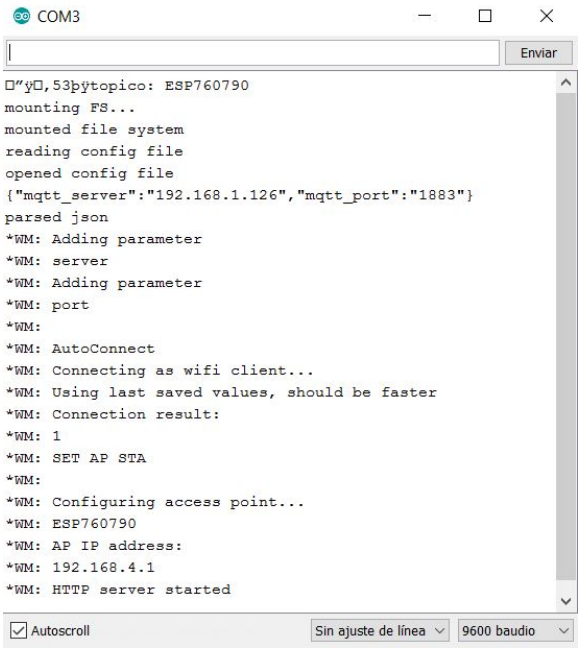

Figura 7.3: Creación del punto de acceso

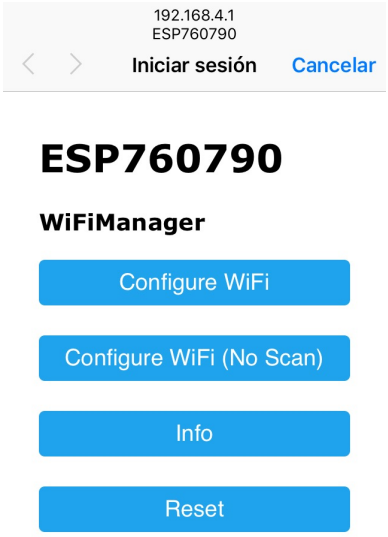

Figura 7.4: Ventana del punto de acceso en terminal

Mientras tanto, en el Monitor Serie de Arduino, se dan las advertencias mientras que se han realizado los anteriores pasos (desde que se introducen los parámetros hasta que se conecta exitosamente a la red Wi-Fi y a al servidor MQTT) que indica la Figura 7.7. La primera parte es el *feedback* que

Figura 7.5: Configuración de parámetros en ventana emergente

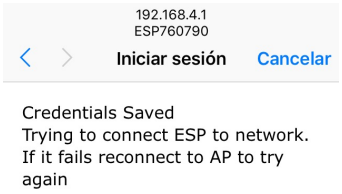

Figura 7.6: Confirmación de entrada de parámetros introducida

el ESP8266 envía para afirmar que se están introduciendo los parámetros en la ventana emergente, después muestra los parámetros introducidos, hace un intento de conexión al nuevo punto de acceso sugerido y finalmente se conecta. Por último y abajo del todo, muestra su correcta conexión al servidor MQTT y su puerto.

Una vez preparado el dispositivo pasarela, queda conectar el cliente MQTT que ejecutará los mensajes MQTT. Luego desde el terminal y con la aplicación anteriormente instalada, introducimos en su interfaz de entrada todos aquellos parámetros que permiten su conexión con el servidor (véase Figura 7.8).

Como el protocolo utilizado no es MQTT seguro, no es necesario autentificarse en el mismo con usuario y contraseña. Una vez escrito los parámetros, se accederá a la aplicación y al envío del mensaje, no sin antes escribir el tópico al que se desea publicar lo que indica la Figura 7.9.

En estas dos ventanas que aparecen dentro de la aplicación, en la de arriba se muestran los mensajes que pasa por el tópico y el nombre del tópico en cuestión (arriba del todo), y en la segunda ventana de abajo se

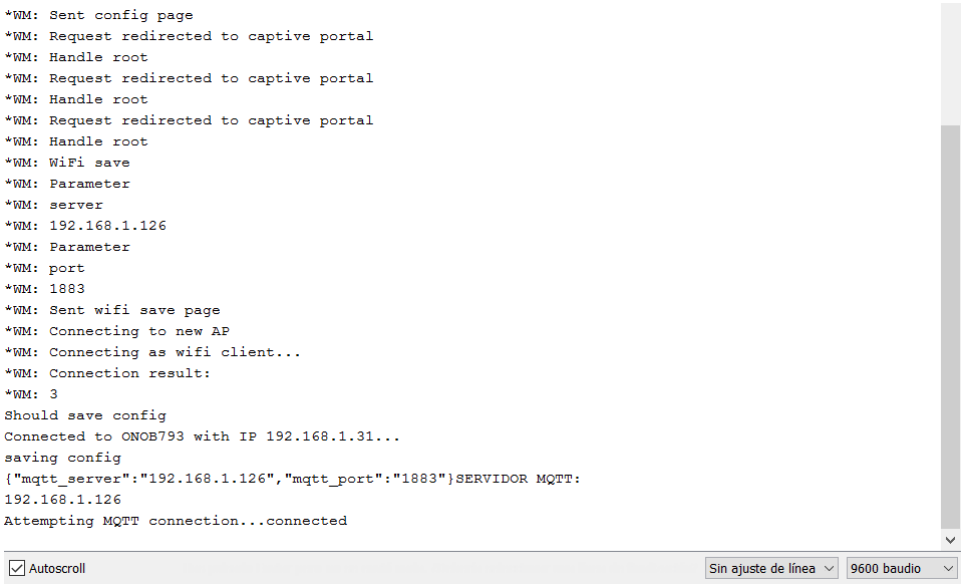

Figura 7.7: Visualización desde el Monitor Serie de Arduino

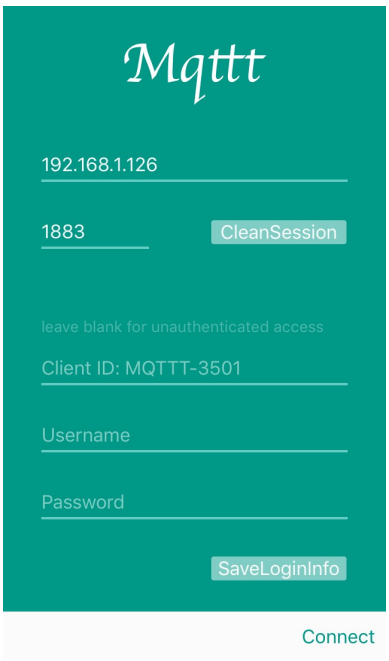

Figura 7.8: Interfaz de entrada de la aplicación para MQTT

muestra el mensaje a mandar y el tópico en el que se publica (encima de ese mensaje). Para publicarlo sólo hay que presionar en la esquina inferior derecha llamada *Publish*.

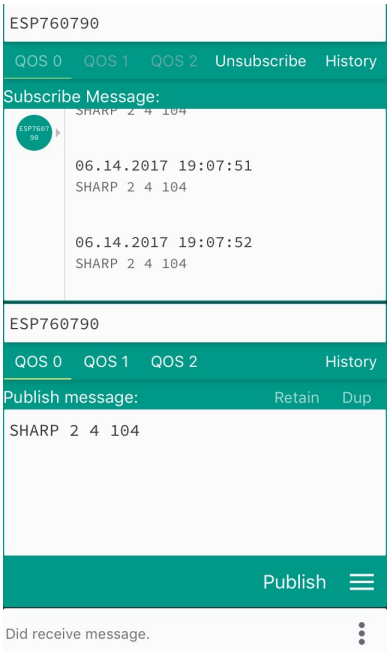

Figura 7.9: Ingreso de mensaje MQTT de emisión desde la aplicación

Dicha acción llega al chip ESP8266, que activa su circuito emisor y seguidamente nos cercioramos de que el dispositivo final al que se envía (que en este caso es una cadena de música) se enciende (ya que el mensaje mostrado representa la tecla ON de dicho aparato, el cual es la tecla más habitual en este tipo de pruebas).

Un dato importante en el que se ha visto interesante hacer mención es el del tiempo de respuesta, pues este es casi automático, del orden del segundo, luego no se aprecia diferencia apenas con un control remoto normal teniendo en cuenta el tiempo de respuesta desde el *smartphone* hasta el aparato del hogar. Pero si hay que comentar que el tiempo entre envío de comando y comando debe ser mínimo de 3 segundos, para que al dispositivo le de tiempo a procesar la información.

Mediante la aplicación del cliente MQTT instalada en el móvil se podía visualizar los diferentes mensajes publicados en el tópico, pero además existe otra alternativa a esa visualización de los mensajes enviados al tópico mediante nuestro servidor en el computador, y es que, mediante el "cmd" e introduciendo el comando de la Figura 7.10, es posible su visualización. Ese comando permite la subscripción al tópico que se le indique a continuación de esa sencilla manera.

Incluso, además de poder visualizar los diferentes mensajes MQTT que viajan por el tópico, es posible publicar mensajes MQTT desde el cmd de

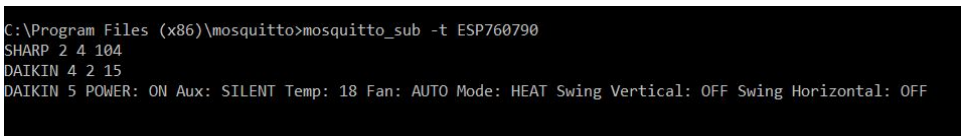

Figura 7.10: Mensajes MQTT publicados en el tópico desde el cmd

Windows, que para ello, el comando (compuesto del tópico al que publicar y el mensaje que se desea enviar) que se debe utilizar se muestra en la Figura 7.11.

Figura 7.11: Mensaje MQTT emitido en el tópico desde el cmd

Hay que tener en cuenta que la ejecución de estos comandos solo surtirán efecto si anteriormente se ha accedido desde el cmd de Windows a la carpeta donde se instaló el software Mosquitto. Por defecto, la ruta de dicha carpeta será: C:\Program Files (x86)\mosquitto

Distancias alcanzadas por los diferentes circuitos emisores

Otro de los parámetros que se han considerado interesantes es el de la distancia de emisión de las señales infrarrojas. Al poseer varios tipos de circuitos de emisión con más y menos potencia, hay que valorar las diferentes distancias que ofrecen.

En el primer diseño de emisión del que se hizo referencia en el capítulo anterior, consistente en la alimentación mediante cable micro-USB del módulo y por tanto, alimentado con 3V, cuyos diodos IR se situaban en serie, se ha logrado una distancia de emisión de 4 metros.

En el segundo caso, con el mismo tipo de alimentación, pero con los diodos infrarrojos en paralelo, se ha logrado una alcance de casi 3 metros (2.9 metros). En parte debido a que por ese paralelismo en los LEDs, la corriente que pasa por cada uno se divide.

Para el último caso, el que se utilizaba una alimentación superior al ser de 5V, se ha encontrado con un alcance de 6,1 metros. Por lo tanto, se logra mayor alcance si se transmite mayor corriente (sin llegar a la máxima permitida por los diodos).

Simulación de recepción de señal infrarroja

Para finalizar este tópico, al igual que se ha mostrado y descrito cómo realizar un envío de un mensaje de manera exitosa, también se expondrá la

simulación de recepción de señal infrarroja.

Dando por hecho que el servidor Mosquitto está listo, el dispositivo pasarela con el circuito también (siendo igual que en la emisión), se pasará a hacer el uso de la parte del circuito receptora, que será la que capte las señales IR. Se conecta el dispositivo al ordenador abriendo el Monitor Serie de Arduino que puede mostrar el mensaje MQTT que se manda una vez recibido. También puede desde la aplicación apreciarse la nueva notificación en el tópico. Que incluso para realizar una prueba resulta más interesante de usar para más tarde copiar el código e intentar su reenvío.

Si todo está preparado ya (incluso la suscripción al correspondiente tópico), comenzaremos con el envío del mensaje MQTT que activará el modo receptor del circuito desde la aplicación en el *smartphone*. Un posible mensaje podría ser el elegido en la Figura 7.12. El mensaje descrito en la mensaje refleja el deseo del cliente de aprender la tecla ON de un mando del RTL-SDR que funciona con el protocolo NEC.

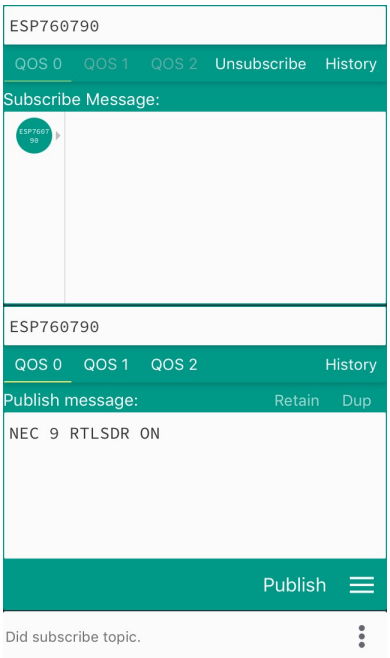

Figura 7.12: Ingreso de mensaje MQTT de recepción desde la aplicación

Una vez se publique el mensaje, automáticamente el circuito encenderá el LED luminoso que se encuentra en él dando a entender que ahora el circuito se encuentra en modo de escucha. Éste se mantendrá encendido hasta que la señal infrarroja sea detectada. Una vez que la señal infrarroja se reciba el LED se apagará. Hecho eso el chip ESP8266 enviará un mensaje

MQTT al tópico indicando en él el código RAW que se ha capturado. Puede visualizarse tanto desde el Monitor Serie como desde la aplicación. Desde este último, el mensaje sería visto como aparece en la Figura 7.13.

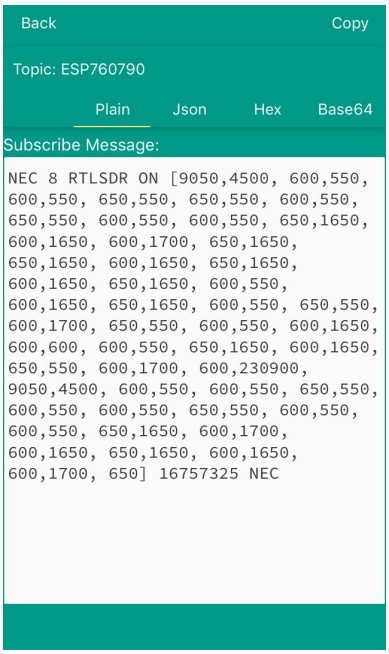

Figura 7.13: Código RAW recibido dentro del mensaje MQTT

Esta imagen representa el mensaje MQTT recibido en el tópico, que sería de interés a la base de datos, para su posterior almacenamiento y futuro uso. Cabe tener en cuenta que el aprendizaje interesa para el caso de señales infrarrojas para dispositivos que no son aires acondicionados. Para los climatizadores, la pulsación de un botón no representa la misma señal infrarroja en todos los casos porque el mensaje que lleva inscrito contiene todos los parámetros del estado del aire acondicionado, y no tienen que ser idénticos en todas las situaciones. Más bien, para un aprendizaje de señales infrarrojas de nuevos aires acondicionados, interesa incluir en el chip las librerías que lo habiliten desde un principio.

Volviendo al anterior resultado en la recepción, si queremos comprobar que el código RAW recibido es válido solo hay que copiarlo e introducirlo en un mensaje de emisión de señal IR con su respectivo campo de “Tipo” (valor 0), y publicarlo en el tópico. En este caso, si se desconoce la frecuencia de transmisión, sería conveniente empezar a probar con el valor más utilizado, el cual es el de 38kHz.

Consumo de energía de los diseños

Por último, se comentarán los valores de energía que consumen cada uno de los módulos utilizados en el diseño de alimentación de 3.3V alimentado mediante cable micro-USB. Además, el circuito alimentado por 5V es de interés apreciar su diferencia de consumo respecto de los dos anteriores por su incremento de voltaje, y por tanto, de corriente. Se han analizado tres tipos de casos en cada uno: el primero consistirá en dejar en reposo el dispositivo, y será el valor más importante para tener en cuenta pues será el que represente la mayoría de tiempo cuando se encuentre en su puesta en marcha; en segundo lugar se halla el modo de envío de comandos, que concretamente se realizarán 20 órdenes por minuto. Esta cantidad se ha considerado suficiente ya que representa un comando cada 3 segundos, luego no es algo muy común en un cliente final y teniendo en cuenta que como se explicó anteriormente el dispositivo sólo es capaz de procesar correctamente una orden cada 3 segundos, representaría el caso límite de máximo consumo. Y de menor interés, y por tanto ubicado en el último puesto, se expone el consumo en el modo de aprendizaje del circuito. Es el que menos probabilidad de aparición tiene y se han ejecutado unas 20 órdenes por minuto. Se ha decidido agrupar todos estos resultados en tres Cuadros (Cuadro 7.1, Cuadro 7.2 y Cuadro 7.3) que den a conocer los valores de forma más clara. La primera será para el módulo *Wemos D1 Mini*, la segunda para *NodeMCU v1.0* y la tercera para el *NodeMCU v1.0* con un circuito de alimentación de 5V.

Modo de funcionamiento	Consumo en 1 minuto (mAh)
Modo reposo	0.01428571
Modo envío de comandos	0.01666667
Modo recepción de comandos	0.02

Cuadro 7.1: Tabla de consumo del módulo *Wemos D1 Mini*

Modo de funcionamiento	Consumo en 1 minuto (mAh)
Modo reposo	0.01503759
Modo envío de comandos	0.01626016
Modo recepción de comandos	0.02105263

Cuadro 7.2: Tabla de consumo del módulo *NodeMCU v1.0*

Modo de funcionamiento	Consumo en 1 minuto (mAh)
Modo reposo	0.01639344
Modo envío de comandos	0.01769912
Modo recepción de comandos	0.01960784

Cuadro 7.3: Tabla de consumo del módulo *NodeMCU v1.0* con alimentación de 5V

Capítulo 8

Conclusiones y trabajos futuros

8.1. Conclusiones

Llegado a este punto del Trabajo Fin de Grado, se da por finalizado la solución propuesta en este proyecto alcanzada de manera satisfactoria. Es un proyecto que pretendía junto con otra serie de proyectos llevados a cabo por el mismo profesor complementarse para ofrecer una solución final y completa en cuanto a la integración de nuevos sistemas en la domótica, todo a un coste muy competitivo.

Un gran punto a favor de este nuevo sistema reside en el posible ahorro de costes. Esto es posible gracias al rápido conocimiento de los módulos empleados en el mercado que son el cerebro de la solución como son *NodeMCU v1.0* y *Wemos D1 Mini*.

De la solución final como suma de los trabajos, en concreto el presente tiene como objetivo permitir la comunicación con aquellos dispositivos del hogar capaces de ser controlados mediante señales infrarrojas, todo ello implantado con un protocolo de bajo requerimiento de ancho de banda capaz de enviar mensajes cortos diseñado para dispositivos IoT de bajas prestaciones, que más tarde podría complementarse con una sólida interfaz de usuario basada en la plataforma openHAB. Esto complementa el sistema puesto que al usuario se familiariza más cómodamente con una aplicación en el *smartphone* donde se ubica la base de datos tan mencionada en este trabajo.

Junto con los argumentos anteriores que expresan la necesidad de la adición de dicha plataforma, se une una necesidad más, que radica en el campo de la seguridad. Mientras que MQTT en su versión común se encuentra totalmente desprotegida de posibles capturas de mensajes, openHAB facilita una comunicación con autentificación, método que evitaría escuchas indeseadas,

no por parte de cualquier usuario puesto que la comunicación inalámbrica como 3G, 4G o Wi-Fi no es fácil de descifrar, sino por parte de la operadora que abastece el acceso a Internet, evitando regalar información muy valiosa.

Las principales aportaciones de este proyecto son:

- Como más importante, la gestión de un dispositivo pasarela con una plataforma flexible y universal con la que no sólo pueda tratar con aparatos del hogar del mismo uso (sólo para TV) como reflejaría un control remoto universal disponible en el mercado, sino con dispositivos con diferentes funcionalidades, ya sean TV, TDT, DVD, equipo de música o aire acondicionado, entre otros.
- Una alternativa válida para dispositivos del actual IoT, pues el protocolo estandarizado MQTT está diseñado para dispositivos pequeños sin gran procesamiento de cálculo. Permite el envío de mensajes pequeños que no requieren de un gran ancho de banda. La estructura que se ha seguido en este trabajo presenta una forma efectiva de actuar, simple y concisa, válida para un caso práctico.
- Aunque el diseño que se ha llevado a cabo tiene su contexto dentro del hogar, incluyendo al usuario final, las herramientas utilizadas permiten una comunicación desde su contexto más amplio, luego no se limita a un espacio cerrado. Su alcance abarca hasta aquellas zonas donde la tecnología móvil (como 3G o 4G) hace presencia.
- El hallazgo del funcionamiento de los diferentes protocolos de señales infrarrojas no es tarea fácil teniendo en cuenta la gran cantidad de protocolos existentes en la actualidad. A modo de base de datos, se ha incluído una gran cantidad de información de muchos de estos protocolos siendo necesaria su búsqueda en datasheets de cada marca, al ser protocolos propietarios.
- Este proyecto no sólo contribuye con una solución a la hora de implantar un sistema de comunicación con dispositivos finales del hogar, sino que ofrece una serie de alternativas con diferentes módulos como *NodeMCU v1.0* o *Wemos D1 Mini* e incluso diferentes circuitos con sus respectivos resultados tanto de consumo, potencia y distancia. Aquella persona interesada en poner en marcha esta solución es mediante su criterio personal la que decida qué alternativa será la elegida para su implementación.

8.2. Vías futuras

Todos los objetivos propuestos para este Trabajo Fin de Grado han sido completados y realizados con éxito, sin embargo, este proyecto permite imaginar nuevas mejoras futuras o implementaciones adicionales.

Las mejoras que se mencionan son descritas a continuación:

- Las librerías utilizadas en el software Arduino definen la estructura de señales infrarrojas de una gran variedad de dispositivos de diferentes funciones. Contienen una gran variedad de protocolos que habilitan la comunicación, aunque la incorporación de nuevos dispositivos en el hogar puede dar lugar al no reconocimiento de las señales existentes en la base de datos. Es por ello que se decidió que el dispositivo pasarela pudiese aprender nuevos comandos e incluirlos en su sistema, dando por hecho el constante valor para una determinada tecla.

  Existe la excepción en los comandos de los climatizadores, pues estos no presentan un valor constante en el mensaje IR que se transmite, debido a su mayor complejidad en la estructura. Cada caso puede representar diferentes valores para la pulsación de una misma tecla, puesto que depende de los restantes parámetros que representan el estado del aire acondicionado.

  Esto impide que ante desconocidos protocolos de climatizadores aun por descubrir, se descarte el uso del modo aprendizaje del dispositivo pasarela. Específicamente para estos dispositivos sería interesante la adición de nuevas librerías en su programación a modo de extensión. Un ejemplo de librería a incluir sería: *arduino-heatpumpir*. Dicha librería facilita la comunicación con protocolos como los de Daikin, Fujitsu, Hisense, Hitachi, Hyundai, Midea, Mitsubishi, Panasonic, Samsung, Sharp y Toshiba.

- La seguridad de los mensajes MQTT podría ser reforzada con MQTT seguro. Aunque openHAB sostiene una plataforma con autenticación, MQTT dejaría de depender de la seguridad de esta si fuese posible su implementación. Existen actualmente librerías de Arduino que sostienen MQTT seguro, pero no han llegado a establecer de manera tan automática la conexión como sucede en MQTT normal, luego habrá que esperar a que se sedimente esta herramienta para encontrar nuevas soluciones más seguras.

  Incluso el servidor Mosquitto puede instalarse para mantener MQTT seguro, con su puerto correspondiente, que en dicho caso pasaría a ser el 8883, y la aplicación al móvil del usuario final debería ser diferente cuyo objetivo sea el de poder utilizar MQTT seguro.

- Así como los controles remotos disponen de sensores térmicos para mantener informado al momento de la temperatura y humedad del ambiente, es posible de manera adicional conectar uno de estos sensores como por ejemplo el sensor DHT22 o el DHT11 (de menor precisión

al anterior mencionado) a otro de los numerosos puertos que disponen los módulos elegidos en este proyecto, ya que ofrecería información adicional útil para el uso del sistema.

- Como es sabido, el chip ESP8266 no ha sido el último desarrollado. Hoy en día podemos encontrar dos nuevos modelos en el mercado que le suceden, como se comentó en capítulos anteriores, comericalizados los últimos años. De ellos se esperan nuevas gamas de módulos que realicen las funciones que cometían en sus anteriores versiones incluyendo las mejoras de los nuevos chips, como son otro tipo de tecnologías de comunicación como Bluetooth y mayores prestaciones de memoria y procesamiento. Aún carecen de documentación oficial en inglés, pero pronto se dará a conocer en los próximos años.

8.3. Valoración personal

Para finalizar el Trabajo de Fin de Grado, se ha desarrollado este apartado que trata acerca de las reflexiones personales y de la cual se dará una valoración final una vez finalizado el proyecto.

Como bien indica su nombre, se llega al final de una etapa de mucha dedicación y constancia, donde el aprendizaje esta a la orden del día, teniendo en cuenta que el aprendizaje no finaliza una vez acabada esta etapa obteniendo el Grado en Ingeniería de Tecnologías de Telecomunicación, sino que durante el transcurso de toda nuestra carrera laboral descubriremos que se dará en multitud de ocasiones siendo necesario reinventarse una y otra vez.

Una clave muy importante para este proyecto a comentar ha sido su pronta aparición, pues la propuesta se acordó llevar a cabo a finales de Febrero del año pasado, tiempo que ha permitido su investigación durante el verano y su desarrollo a lo largo de todo el curso, procurando no apurar la realización para la etapa final. De hecho, ha dado la libertad de poder realizar multitud de pruebas en el transcurso y descartar aquellas inválidas o inviables y evitar sustos inesperados de última hora.

A nivel personal, este Grado ha supuesto una enorme adquisición de conocimientos y aptitudes que tras este largo período se han ido recopilando y que consecuentemente se ha demostrado con la realización de este trabajo final de manera exitosa y a tiempo, cumpliendo con los objetivos preestablecidos, buscando siempre la máxima eficiencia posible. Un concepto importante que no se puede olvidar y que ha estado muy presente es la capacidad de abstracción, imprescindible para su realización.

El proyecto ha impulsado el uso de software fácilmente aplicable al mercado como Arduino, conocimiento de chips de bajo coste y gracias a esta

etapa finalizada he podido aprender varios lenguajes de programación altamente exigidos en el mercado laboral, entre otras muchas cosas. Pronto dará lugar a la que realmente es la carrera más difícil de afrontar: la salida a la vida profesional, y que mejor que sentirse bien formado para dar comienzo a ella.

El autor también quiere confesar su satisfacción por todo lo anteriormente descrito, tanto por los conocimientos adquiridos, como por la ayuda facilitada de los docentes de la Escuela Técnica de Ingeniería de Informática y Telecomunicaciones de la Universidad de Granada, de la que se siente orgulloso, ya sin todos estos ingredientes no hubiese sido posible.

Bibliografía

[1] "La domótica como solución de futuro, Madrid vive ahorrando energía", Consejería de Economía e Innovación Tecnológica, Comunidad de Madrid, 2007. Disponible en https://www.fenercom.com/pdf/publicaciones/la-domotica-como-solucion-de-futuro-fenercom.pdf (accedida el 20 de junio de 2017).

[2] Johan Svanberg, "Smart Homes and Home Automation", Berg Insight's M2M Research Series, 2015. Disponible en http://www.berginsight.com/ReportPDF/ProductSheet/bi-sh4-ps.pdf (accedida el 20 de junio de 2017).

[3] "LIFE+DOMOTIC", Layman's Report, Departamento Proyectos Europeos y Cooperación, 2014. Disponible en http://ec.europa.eu/environment/life/project/Projects/index.cfm?fuseaction=home.showFile&rep=file&fil=LIFE09_ENV_ES_000493_LAYMAN_EN_ES.pdf (accedida el 20 de junio de 2017).

[4] Web de Xiaomi IR. Disponible en https://xiaomi-mi.com/sockets-and-sensors/xiaomi-mi-smart-home-all-in-one-media-control-center/ (accedida el 20 de junio de 2017).

[5] Web de Orvibo AllOne IR. Disponible en http://www.orvibo.com/en/product/126.html (accedida el 20 de junio de 2017).

[6] Web de Samsung climatizador F-H6724 Triangle Design Wi-Fi, 6.8 kW. Disponible en http://www.samsung.com/es/air-conditioners/wall-mount-ar09kswsbwkn/AR24KSWSAWKNEU/ (accedida el 20 de junio de 2017).

[7] Web de STB IR. Disponible en https://es.aliexpress.com/item/Free-shipping-and-dropshipping-Air-Conditioner-DVD-Projector-TV-home-appliances-universal-smart-remote-control-switch/32471405865.html (accedida del 20 de junio de 2017).

[8] José Luis Zamorano Flores. Gloria Fca. Serrano Moya, "Sistemas infrarrojos de comunicaciones inalámbricas", Área de Comunicaciones

del Depto. de Electrónica de la UAM Azcapotzalco. Disponible en http://www.izt.uam.mx/newpage/contactos/anterior/n47ne/infra.pdf (accedida el 20 de junio de 2017).

[9] San Bergmans, "IR Remote Control Theory", 2013. Disponible en http://www.sbprojects.com/knowledge/ir/index.php (accedida el 20 de junio de 2017).

[10] Miguel Ángel Padrón y Ángel Plaza, "Sobre el problema de la radiación del cuerpo negro",Technical Report. nº 011 - 1999, Instituto Universitario de Microelectrónica Aplicada de las Palmas de Gran Canaria, 1999. Disponible en http://www.dma.ulpgc.es/profesores/personal/aph/ficheros/investigacion/ficheros/cuerpo_negro.pdf (accedida el 20 de junio de 2017).

[11] Javier de Lope Asiaín, "Emisión y recepción de infrarrojos", Departamento de Inteligencia Artificial, Facultad de Informática, Universidad Politécnica de Madrid, Nov 2001. Disponible en http://www.dia.fi.upm.es/~jdlope/docs/delope01a.pdf (accedida el 20 de junio de 2017).

[12] Oswaldo B. González Hernández, "Estudio de la aplicación de técnicas de modulación OFDM para comunicaciones ópticas guiadas en el canal infrarrojo", Tesis Doctoral, Universidad de la Laguna, 2005. Disponible en ftp://h3.bbtk.ull.es/ccppytec/cp192.pdf (accedida el 20 de junio de 2017).

[13] Paul Seerden, "AN10210 Using the Philips 87LPC76x microcontroller as a remote control transmitter", Application Note, Philips, May 2003. Disponible en http://datasheet.datasheetarchive.com/originals/library/Datasheet-024/DSA00415600.pdf (accedida el 20 de junio de 2017).

[14] "Implementing receivers for infrared remote control protocols using STM32F10xxx microcontrollers", Application Note, ST, Feb 2012. Disponible en http://www.st.com/content/ccc/resource/technical/document/application_note/c7/d1/63/f7/80/06/41/a4/CD00267896.pdf/files/CD00267896.pdf/jcr:content/translations/en.CD00267896.pdf (accedida el 20 de junio de 2017).

[15] "Infrared remote control code for JVC models", JVC, Ago 2006. Disponible en http://support.jvc.com/consumer/support/documents/RemoteCodes.pdf (accedida el 20 de junio de 2017).

[16] "Recommendations for Implementing an IR Blaster on MC1321x and MC9S08QE128 - Based IEEE® 802.15.4 Wireless Nodes",Application Note, Freescale Semiconductor, 2010. Disponible en http://datasheet.datasheetarchive.com/originals/library/Datasheets-SW1/DSASW0017886.pdf (accedida el 20 de junio de 2017).

[17] San Bergmans, "Nokia NRC17 Protocol" Mar 2016. Disponible en http://www.sbprojects.com/knowledge/ir/nrc17.php (accedida el 20 de junio de 2017).

[18] San Bergmans, "Sharp Protocol" Mar 2016. Disponible en http://www.sbprojects.com/knowledge/ir/sharp.php (accedida el 20 de junio de 2017).

[19] San Bergmans, "Philips RC-MM Protocol" Mar 2016. Disponible en http://www.sbprojects.com/knowledge/ir/rcmm.php (accedida el 20 de junio de 2017).

[20] San Bergmans, "RCA Protocol" Mar 2016. Disponible en http://www.sbprojects.com/knowledge/ir/rca.php (accedida el 20 de junio de 2017).

[21] San Bergmans, "X-Sat / Mitsubishi Protocol" Mar 2016. Disponible en http://www.sbprojects.com/knowledge/ir/xsat.php (accedida el 20 de junio de 2017).

[22] "ESP8266EX Datasheet", Espressif, 2017. Disponible en http://espressif.com/sites/default/files/documentation/0a-esp8266ex_datasheet_en.pdf (accedida el 20 de junio de 2017).

[23] "Wifi en Arduino", Nov 2016. Disponible en https://aprendiendoarduino.wordpress.com/tag/esp32/ (accedida el 20 de junio de 2017).

[24] "Modules", ESP8266 Community Wiki, May 2017. Disponible en http://www.esp8266.com/wiki/doku.php?id=esp8266-module-family (accedida el 20 de junio de 2017).

[25] "Comandos Hayes", Facultad de Ingeniería de la Universidad de Buenos Aires. Disponible en materias.fi.uba.ar/6637/tps/practica_hayes.doc (accedida el 20 de junio de 2017).

[26] Roberto Ierusalimschy, Luiz Henrique de Figueiredo, Waldemar Celes, "Manual de Referencia de Lua 5.1", Sep 2011. Disponible en https://www.lua.org/manual/5.1/es/manual.html (accedida el 20 de junio de 2017).

[27] Web de Wemos D1 Mini. Disponible en https://www.wemos.cc/ (accedida el 20 de junio de 2017).

[28] "Samsung RAW IR Codes", 2015. Disponible en https://aprendiendoarduino.wordpress.com/category/software/ (accedida el 20 de junio de 2017).

[29] Web de MQTT. Disponible en http://mqtt.org/ (accedida el 20 de junio de 2017).

[30] Gaston C Hillar, "MQTT Essentials - A Lightweight IoT Protocol", Libro, 1º Edición, Abr 2017. Disponible en https://www.packtpub.com/application-development/mqtt-essentials-lightweight-iot-protocol (accedida el 20 de junio de 2017).

[31] Conversión de código RAW a HEX, Sep 2016. Disponible en https://www.reddit.com/r/arduino/comments/2yb2hk/samsung_raw_ir_codes/ (accedida el 20 de junio de 2017).

[32] "WiFi IR Blaster", 2016. Disponible en https://www.hackster.io/BuddyC/wifi-ir-blaster-af6bca (accedida el 20 de junio de 2017).

[33] "IR Receiver Modules for Remote Control Systems", Vishay Semiconductors, Jul 2016. Disponible en http://www.vishay.com/docs/82459/tsop48.pdf (accedida el 20 de junio de 2017).

[34] "P2N2222A", Amplifier Transistors, On Semiconductor, Ene 2013. Disponible en http://www.onsemi.com/pub/Collateral/P2N2222A-D.PDF (accedida el 20 de junio de 2017).

[35] "IRremote ESP8266 Library", Aportación de GitHub, Jun 2017. Disponible en https://github.com/markszabo/IRremoteESP8266 (accedida el 20 de junio de 2017).

[36] "WiFiManager", Aportación de GitHub, Jun 2017. Disponible en https://github.com/tzapu/WiFiManager (accedida el 20 de junio de 2017).

[37] "Arduino Client for MQTT", PubSubClient, Aportación de GitHub, Jun 2017. Disponible en https://github.com/knolleary/pubsubclient (accedida el 20 de junio de 2017).

[38] "Mosquitto", Eclipse. Disponible en https://mosquitto.org/ (accedida el 20 de junio de 2017).

[39] "Step by step installing and configuring Mosquitto with Windows 7", Eclipse, Jun 2015. Disponible en https://sivatechworld.wordpress.com/2015/06/11/step-by-step-installing-and-configuring-mosquitto-with-windows-7/ (accedida el 20 de junio de 2017).

[40] "Receiving IR signals with RTL-SDR dongles", Eclipse, Jul 2016. Disponible en https://medium.com/@rxseger/receiving-ir-signals-with-rtl-sdr-dongles-5a8658a44b90 (accedida el 20 de junio de 2017).

[41] "IR Remote Code", Aportación de GitHub, Feb 2016. Disponible en https://github.com/lepiaf/IR-Remote-Code/blob/master/README.md (accedida el 20 de junio de 2017).

[42] "Samsung remote IR-codes", Jun 2011. Disponible en http://arduinostuff.blogspot.com.es/2011/06/samsung-remote-ir-codes.html?m=1 (accedida el 20 de junio de 2017).

[43] "Find IR codes". Disponible en http://irdb.tk/find/ (accedida el 20 de junio de 2017).

[44] "Infrared Hex Code Database". Disponible en http://www.remotecentral.com/cgi-bin/codes/ (accedida el 20 de junio de 2017).