uc3m | Universidad Carlos III de Madrid

Grado Universitario en Ingeniería de Sistemas Audiovisuales
2017-2018

Trabajo Fin de Grado

"Aplicaciones piloto de Internet de las cosas"

Alejandro Arribas Palomo

Tutor

Carlos García Rubio

Leganés, Julio 2018

RESUMEN

El objetivo principal del proyecto es desarrollar dos aplicaciones piloto de Internet de las cosas (en inglés, *Internet of Things, abreviado IoT*) capaces de controlar dispositivos restringidos, en este caso, dos bombillas, gracias al protocolo de aplicación restringida, CoAP.

El trabajo se desarrolla dentro del entorno de la Raspberry Pi donde se ha instalado el sistema operativo Raspbian. Desde ahí, se ha desarrollado el código en lenguaje Python para la manipulación de las bombillas a partir de la puerta de enlace que es el intermediario en la transmisión de los mensajes.

Las aplicaciones que se han diseñado son un *Mando Remoto* y un *Temporizador*. El primer prototipo se crea para facilitar al usuario la configuración de las distintas funcionalidades de las bombillas, pudiendo ser de manera individual o colectiva. Las funcionalidades que se pueden controlar son el encendido o apagado, el ajuste de intensidad y, por último, el color, todo ello mediante la Raspberry Pi como control remoto.

La segunda aplicación es el *Temporizador*, cuya funcionalidad se centra en programar el encendido y apagado de una o ambas bombillas dependiendo el tiempo de programación elegida por el usuario, que puede ser entre 3 opciones: programación diaria, semanal o mensual. El usuario establece la fecha y la hora a la que se debe llevar a cabo el programa con el fin de simular que se encuentra en el domicilio para evitar posibles robos en la vivienda.

Antes de comenzar el desarrollo software, se analizó cómo se transmiten los mensajes desde la puerta de enlace, conectada a la red mediante un cable ethernet, a las bombillas. La puerta de enlace está formada por un sistema llamado, Trådfri, con el cual se consigue la transmisión de paquetes entre ambos dispositivos a través del protocolo de comunicación inalámbrico ZigBee.

También, se investigó como se genera la comunicación entre la Raspberry Pi y la puerta de enlace. Esto se consigue mediante los protocolos de comunicación de la Raspberry Pi, donde se ha utilizado CoAP y el protocolo de seguridad de capa de transporte de datagramas (DTLS).

Por último, se realizó tanto el presupuesto para llevar a cabo el proyecto, en el cual, se exponen los costes del personal y material, como la conclusión donde se analiza si se han cumplido o no los objetivos marcados.

Palabras clave: Raspberry Pi; IoT; CoAP.

ÍNDICE DE CONTENIDOS

RESUMEN II
ÍNDICE DE FIGURAS VI
ÍNDICE DE TABLAS VII
1. INTRODUCCIÓN 1
1.1. Motivación 1
1.2. Objetivos 1
1.3. Entorno socio-económico 2
1.4. Legislación y marco regulador 4
1.5. Estructura 5
2. ESTADO DEL ARTE 7
2.1. IoT 7
2.1.1. Patrón de comunicación dispositivo a dispositivo 8
2.2. Protocolos de comunicación en IoT 9
2.2.1. ZigBee 9
2.2.2. CoAP 10
2.2.3. DTLS 18
2.2.4. LWM2M 21
2.3. Raspberry PI 22
2.3.1. Raspbian 23
2.3.2. Python 25
3. INSTALACIÓN E INICIACIÓN AL ENTORNO 26
3.1. Equipamiento 26
3.1. Instalación del entorno 31
3.1.1 Instalar el kit Trådfri Led 31
3.1.2 Montaje de la Raspberry Pi 32
3.1.3 Instalación del sistema operativo 33
3.1.4 Configuraciones y actualizaciones 35
3.2. Iniciación al entorno 36
3.2.1 Pruebas con librería CoAP 36
3.2.2 Pruebas con solicitudes en Python 38
3.2.3 Aplicación de prueba con Rainbow Hat 41
4. APLICACIÓN 1: MANDO REMOTO 44
4.1. Introducción y objetivos 44

4.2. Análisis e implementación 45
5. APLICACIÓN 2: TEMPORIZADOR **58**
5.1. Introducción y objetivos 58
5.2. Análisis e implementación 59
6. PLANIFICACIÓN Y PRESUPUESTO **76**
6.1. Planificación 76
6.2. Presupuesto 78
7. CONCLUSIONES **80**
7.1. Conclusión 80
7.2. Opinión Personal 81
7.3. Líneas futuras 82
8. ENGLISH SUMMARY **83**
9. GLOSARIO **89**
10. REFERENCIAS **90**
APÉNDICE 1. MANUAL DE INSTALACIÓN **1**
APÉNDICE 2. MANUAL DE USUARIO **2**
Aplicación 1: Mando Remoto 2
Aplicación 2: Temporizador 5

ÍNDICE DE FIGURAS

Ilustración 1: IoT surgió entre 2008 y 2009 7
Ilustración 2: Modelo de comunicación dispositivo a dispositivo. 8
Ilustración 3: Transmisión de mensajes de tipo Confirmable..... 12
Ilustración 4: Transmisión de mensaje de tipo Non-Confirmable..... 12
Ilustración 5: Intercambio de mensajes con cookie para evitar denegar el servicio. 19
Ilustración 6: Solución a la pérdida de paquetes gracias al temporizador. 20
Ilustración 7: Partes de la Raspberry Pi 22
Ilustración 8: Logotipo de Raspberry Pi..... 23
Ilustración 9: Raspbian - Combinación de Raspberry Pi y Debian..... 23
Ilustración 10: Contenido del kit Trådfri Led..... 26
Ilustración 11: Material del Starter Kit 30
Ilustración 12: Resultado del montaje de la Raspberry Pi, PIbow Case y el Rainbow Hat 30
Ilustración 13: Dispositivos auxiliares para el control de la Raspberry Pi 31
Ilustración 14: Dos modos de descarga (NOOBS y RASPBIAN) 33
Ilustración 15: Enlaces de descarga del sistema operativo..... 34
Ilustración 16: Imagen tras abrir WIN32 Disk Imager 34
Ilustración 17: Mensaje para que el usuario seleccione la bombilla que desee. 2
Ilustración 18: Mensaje que indica al usuario que está en la primera opción del menú. 3
Ilustración 19: Nivel de intensidad que tiene la bombilla/s. 3
Ilustración 20: "Normal" es el color que se ha establecido en la bombilla/s 4
Ilustración 21: Elección del día que se quiere establecer el Temporizador. 6
Ilustración 22: Elección del mes en que se quiere establecer el Temporizador 6
Ilustración 23: Elección de la hora inicial en que se quiere establecer el Temporizador. 6
Ilustración 24: Elección del minuto inicial en que se quiere establecer el Temporizador. ... 6
Ilustración 25: Elección de la hora final en que se quiere establecer el Temporizador. 6
Ilustración 26: Elección del minuto inicial en que se quiere establecer el Temporizador. ... 6
Ilustración 27: Temporizador activado. 6

ÍNDICE DE TABLAS

Tabla 1: Código de posibles respuestas en caso de éxito. **16**
Tabla 2: Código de posibles respuestas por error del cliente **16**
Tabla 3: Código de posibles respuestas por error del servidor. **17**
Tabla 4: Comandos para actualizar los repositorios y el sistema **35**
Tabla 5: Comando para el acceso al menú oculto **35**
Tabla 6: Comando de solicitud para el apagado de una bombilla **36**
Tabla 7: Comandos de solicitudes para el ajuste de intensidad **38**
Tabla 8: Comandos de solicitudes para el ajuste del color **38**
Tabla 9: Comandos para la recepción del estado de las bombillas **38**
Tabla 10: Contenido del nano texto tradfri.cfg **39**
Tabla 11: Comando de solicitud para el encendido de la primera bombilla **39**
Tabla 12: Comando para registrar un usuario API y recibir la clave **40**
Tabla 13: Tradfri.cfg actualizado **40**
Tabla 14: Comando con el cual recibe el estado de las bombillas **40**
Tabla 15: Comando para la configuración de ambas bombillas **40**
Tabla 16: Comando para la instalación del software Rainbow Hat **42**
Tabla 17: Funciones para cambiar el estado de las bombillas **42**
Tabla 18: Función del botón A **43**
Tabla 19: Función para conocer que botón se ha pulsado **47**
Tabla 20: Funciones para activar y desactivar las bombillas mediante los botones B y C respectivamente. **48**
Tabla 21: Función para ajustar la intensidad dependiendo la bombilla escogida **51**
Tabla 22: Solicitud mediante el protocolo CoAP en lenguaje Python **52**
Tabla 23: Función colour - cambia el color de la bombilla escogida. **55**
Tabla 24: Función tradfriActions.tradfri_color_light() **56**
Tabla 25: Funciones para el control de los leds en el Rainbow Hat **59**
Tabla 26: Implementación del código para solventar el error al seleccionar el mes incorrecto **63**
Tabla 27: Sentencia para encender un punto decimal **66**
Tabla 28: Función para ejecutar el temporizador diario **67**
Tabla 29: Fragmento de código para comprobar si el día introducido es lunes **70**
Tabla 30: Función para ejecutar el temporizador semanal **72**
Tabla 31: Función para ejecutar el temporizador mensual **75**
Tabla 32: Planificación del proyecto mediante el Diagrama de Gantt **76**

Tabla 33: Presupuesto del personal **78**
Tabla 34: Presupuesto del material **79**
Tabla 35: Resumen de costes **79**
Tabla 36: Comando para descargar el repositorio el cual contiene el software de ambas aplicaciones. **1**
Tabla 37: Comando para ejecutar la aplicación Mando Remoto **1**
Tabla 38: Comando para ejecutar la aplicación Temporizador **1**

1. INTRODUCCIÓN

1.1. Motivación

Internet de las cosas es una de las tecnologías más prometedoras en la actualidad. Este término se refiere a la interconexión digital de objetos cotidianos con Internet[1].

Poco a poco, las industrias van adoptando esta tecnología con el fin de recopilar información como, por ejemplo, captar la temperatura o humedad que hay en tiempo real en el medio ambiente a través de sensores u optimizar tiempo, como diseñar una aplicación que realice un inventario y muestre la ubicación de donde se encuentra cada dispositivo.

Además de las empresas, en los hogares se está introduciendo un elevado número de objetos que son capaces de conectarse a la red y que el usuario puede interactuar con ellos. Esta tecnología permite acciones como controlar remotamente la puerta del garaje, la iluminación, etc. Hay objetos como por ejemplo la nevera, que muestran al usuario el número de alimentos que contiene en su interior y avisa si alguno de estos se ha agotado para que el usuario sepa la inexistencia de ellos.

Esta tecnología como se puede observar, permite crear multitud de aplicaciones dependiendo del objeto, por ejemplo, en el caso de la iluminación hay bombillas que pueden ser controladas inalámbricamente para realizar distintas acciones como apagar o encender la luz, ajustar la intensidad o cambiar el color. Estas características permiten al programador crear aplicaciones innovadoras para mejorar la forma de vida.

En definitiva, Internet de las cosas es un concepto enriquecedor para conseguir vivir en un mundo inteligente.

1.2. Objetivos

El objetivo principal del proyecto es desarrollar dos aplicaciones piloto de Internet de las cosas con el fin de poder controlar inalámbricamente a través de la Raspberry Pi dispositivos CoAP, concretamente dos bombillas. Para ello, se utiliza una puerta de enlace que actúa como intermediaria entre ambos dispositivos, recibiendo y enviando información a través de los protocolos de comunicación ZigBee (ver sección 2.2.1) y CoAP sobre DTLS (ver sección 2.2.2 y 2.2.3).

[1] Fuente vía - https://es.wikipedia.org/wiki/Internet_de_las_cosas

Para conseguir el objetivo se realizó un tutorial para aprender a controlar las bombillas a través de aplicaciones en lenguaje de programación Python.

Los pasos del proyecto para cumplir el objetivo son:

- **Desarrollar el entorno:** La primera etapa se centró en desarrollar el entorno que se describe en la página web[2]. Esta página muestra una introducción para entender cómo se envían los comandos desde la terminal de la Raspberry Pi hasta la puerta de enlace[3]. En la página lo enseñan de dos modos, enviando solicitudes directamente mediante el protocolo de comunicación CoAP o a través de comandos en Python. Ambos procesos tienen el mismo fin, controlar las bombillas[4].

- **Implementar una aplicación de mando remoto:** Es la primera aplicación piloto que se desarrolló. El objetivo principal era diseñar un mando remoto para controlar todas las configuraciones posibles de las bombillas. Es decir, tanto el encendido y apagado como la intensidad y el color. Esta aplicación es similar al mando inalámbrico que contiene el kit Trådfri LED[5], pero con mejoras ya que es más dinámico y es posible elegir la bombilla que desea configurar.

- **Implementar una aplicación temporizador:** Es la segunda y última aplicación que se implementó. El objetivo de este prototipo es fomentar la seguridad en el hogar cuando el usuario no se encuentra en ella. La funcionalidad que se destaca es poder programar la bombilla o el conjunto de bombillas en la hora y fecha establecidas por el usuario.

1.3. Entorno socio-económico

La tecnología de Internet de las cosas hace referencia a la posibilidad de que cualquier objeto o cosa pueda estar conectado a Internet y, a su vez, a nosotros.

Al principio, los teléfonos inteligentes, ordenadores y tabletas, eran los únicos conectados a la red, pero, esto ha evolucionado, el número de objetos conectados a Internet ha crecido exponencialmente hasta el punto de que distintos objetos como, por ejemplo, el frigorífico, el termostato o, como en este proyecto, las bombillas, estén conectados a ella realizando acciones imprescindibles por si solas. Las acciones que llevan a cabo estos objetos sirven para mejorar la calidad de vida de las personas.

[2] Fuente vía - https://shop.pimoroni.com/
[3] Fuente vía - https://www.ikea.com/es/es/catalog/products/40337806/
[4] Fuente vía - https://www.ikea.com/es/es/catalog/products/60338452/
[5] Fuente vía - https://www.ikea.com/es/es/catalog/products/80338960/

Actualmente, se estima que el número de objetos conectados a Internet está en torno a los 8.400 millones de dispositivos, provocando un gasto económico en las empresas de 905.000 millones y una inversión de 680.750 millones de euros dedicada al desarrollo de aplicaciones para el consumidor [1]. En el año 2020, se cree que el número de objetos interconectados entre sí y con los seres humanos ascenderá a los 20.400 millones [1], una cifra realmente abrumadora, puesto que se producirá un gran impacto social en la manera de convivir las personas con los objetos.

El principal problema que surgió cuando se desarrolló Internet de las cosas, es que Internet solo tenía reservados 32 bits para utilizarse en direcciones IP. El protocolo de Internet versión 4, IPv4 [2], solo aceptaba 4 billones de direcciones. Esa cifra no era suficiente para conectar tal cantidad de dispositivos. Por ese motivo, se recurrió a varias soluciones como utilizar Traducción de Direcciones de Red (NAT - *Network Address Translation*) o implementar el protocolo de Internet versión 6 (IPv6 - *Protocol Internet version 6*).

El concepto de NAT se refiere a la capacidad de poder conectar a Internet un conjunto de dispositivos a una única dirección IP, otorgando a cada uno ellos una dirección IP privada dentro de la misma red. Así, si dentro de una red hay un número elevado de dispositivos solo ocuparían una dirección IP no tantas direcciones como dispositivos haya.

La segunda solución es, IPv6 [3], la cual aumenta el tamaño de la dirección IP de 32 bits que se tenía antes con el protocolo IPv4 a 128 bits. Esto significa que este protocolo es capaz de proporcionar 670 mil billones de direcciones haciendo posible así la interconexión de todos los dispositivos que se requiera.

Ya con el protocolo IPv6 implementado en los dispositivos, se espera que en los próximos años la tecnología IoT crezca a pasos agigantados. Esta evolución traerá consigo enormes cantidades de datos e información que estará transmitiéndose por la red libremente.

Internet de las cosas, es una tecnología innovadora y eficiente, gracias a ello se está implantando en las empresas consiguiendo grandes beneficios debido a que se está mejorando la eficiencia y la productividad a la vez que se reducen los costes.

El impacto económico que provocará esta tecnología en una década, se estima que estará entre los 8 billones de dólares y creará 4,5 millones de puestos de trabajo en Europa [4].

Hasta el momento, las startups y las grandes empresas tecnológicas son las que han apostado en la innovación de IoT, pero ambas necesitan ayuda mutua. Las startups dedicadas a dicha tecnología tienen pocos recursos para su investigación y las empresas tecnológicas necesitan ideas y modelos disruptivos que pueden encontrar en las startups. Estados Unidos y Europa están a la cabeza como los dos continentes con más empresas emergentes dedicadas a Internet de las cosas.

Estados Unidos invierte en empresas emprendedoras casi 25 mil millones de euros [5] mientras que Reino Unido que es el país europeo que más invierte en startups, no llega a los 2 mil millones de euros, con lo que se podría decir que si Europa quiere ser competitiva con el resto del mundo en esta tecnología debería realizar una mayor inversión.

1.4. Legislación y marco regulador

Dentro de unos años, la mayoría de los objetos de nuestro alrededor estarán interconectados entre sí. Las personas podrán ejecutar acciones o adquirir la información que requieran en cualquier momento sobre el objeto que desee.

El problema fundamental es que vivimos en una sociedad donde todavía se puede vulnerar la red de Internet por ciberataques con lo que se podría obtener datos e información personal de las personas y cosas.

Debido a esto, la Unión Europea ha regulado una serie de campos como son la estandarización, privacidad y protección de datos, ciberseguridad y cibercriminalidad, infraestructura e I+D+i.

Respecto a la privacidad y protección de los datos, se ha publicado el nuevo Reglamento (UE) 2016/679 del Parlamento Europeo y del Consejo [6], aprobando la libre circulación y la protección de los datos de las personas.

La ciberseguridad se trata en la Directiva (UE) 2016/1148 del Parlamento Europeo y del Consejo [7], el 6 de julio de 2016, la cual aplica una serie de medidas para garantizar la seguridad en las redes y establecer en la Unión Europea un nivel común de ciberseguridad y mejorar ante posibles ataques cibernéticos.

La cibercriminalidad se trata en la Directiva 2013/40/UE [8], donde se establecen unas normas sobre las posibles infracciones penales y sanciones aplicables a los ataques contra los sistemas de información.

En cuanto a la estandarización, la Directiva 2014/53/UE [9] trata sobre la coordinación sobre los Estados Miembros en el ámbito de la comercialización de equipos radioeléctricos.

Y, por último, respecto a la infraestructura para el crecimiento de la tecnología de Internet de las cosas se ha impulsado la propuesta de Comunidades Conectadas para facilitar la comunicación entre sí de distintos municipios y, además, a diferentes agentes locales de banda ancha y operadores.

1.5. Estructura

Este es el último apartado de la introducción donde se expone una breve descripción de cada uno de los capítulos de este documento.

1. **Introducción:** En esta sección, se explica qué me ha llevado a elegir este proyecto y cuál es el objetivo del mismo. También incluye un analisis social, económico y legal de las bases del proyecto, así como, las normativas vigentes. Y por último, la estructura que presenta este documento.

2. **Estado del arte:** En este apartado se analiza en detalle todo acerca de Internet de las cosas y sus protocolos de comunicación. Además, se explica el software de la Raspberry Pi y el lenguaje de programación que se ha llevado a cabo para el desarrollo del proyecto.

3. **Análisis y diseño:** En esta parte del documento se describen todos los componentes que se han utilizado y cómo ha sido la iniciación al entorno.

4. **Aplicación 1: Mando Remoto:** En esta sección se explica la primera aplicación que se ha implementado, en ella se expondrá el procedimiento que se ha seguido, la arquitectura del software y el resultado.

5. **Aplicación 2: Temporizador:** En este apartado se explica la segunda y última aplicación desarrollada. Se explicará el procedimiento seguido, la arquitectura del software y el resultado.

6. **Planificación y presupuesto:** En este punto se realiza la planificación del proyecto y una estimación del presupuesto del mismo.

7. **Conclusiones:** En esta sección se exponen las conclusiones y la opinión personal respecto a mi apreciación del proyecto.

8. **English Summary:** En este apartado se realiza un resumen del trabajo en Inglés.

9. **Glosario:** Esta sección está formada por un índice de todos los acrónimos que se han usado durante la redacción del trabajo.

10. **Referencias:** Apartado en el cuál se exponen detalladamente todas las referencias utilizadas en el documento.

Apéndice 1. Manuales de Instalación: este apartado se realiza un manual de instalación para que el usuario siga paso a paso el procedimiento y logre instalar tanto el software como el hardware correctamente.

Apéndice 2. Manuales de Usuario: Sección donde se realiza un manual para el usuario con el fin de explicar el correcto funcionamiento de cada una de las aplicaciones.

2. ESTADO DEL ARTE

En este apartado, se va a exponer un breve análisis de las tecnologías y los protocolos de comunicación que se utilizan en el proyecto.

2.1. IoT

En el año 1999, el británico Kevin Ashton [10] perteneciente al Instituto de Tecnología de Massachusetts (MIT), aplicó por primera vez la expresión "*Internet of Things*" para explicar cómo se conectaban los dispositivos a Internet por medio de sensores. Este proceso lo describió a través del sistema RFID[6] que se basa en transmitir la identificación del objeto por radiofrecuencia.

Internet de las cosas, es un término que se refiere al marco en el que la capacidad de cómputo y la conectividad de Internet se propaga a través de sensores, cosas u objetos, permitiendo intercambiar, generar y consumir datos entre ellos sin interactuar en exceso los usuarios. Cada dispositivo dispone de una dirección IP que le identifica. Con esta IP, el dispositivo puede transmitir y recibir información con el fin de poder monitorear en todo momento el estado del dispositivo para optimizar los procesos y la productividad.

Internet de las cosas surgió cuando el número de objetos conectados a la red era superior al número de las personas que habitaban en el mundo. Para obtener este valor se dividió el número de personas que habitan en la Tierra entre los dispositivos que hay conectados a la red y según IBSG de Cisco [11], estima que Internet de las cosas surgió entre el año 2008 y 2009 puesto que, en 2003, como se puede observar en la siguiente ilustración, los dispositivos no superaban a las personas.

En la siguiente ilustración se muestra el gráfico de los dispositivos que habrá conectados por persona en el futuro, así como, los conectados en 2003:

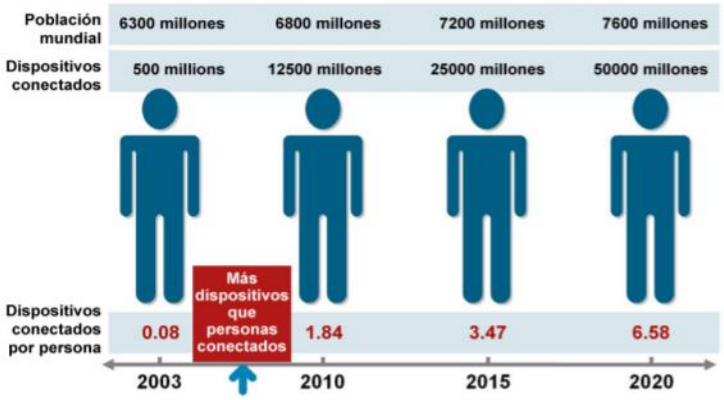

Ilustración 1: IoT surgió entre 2008 y 2009 – Fuente: ISBG de Cisco, abril 2011 [10]

[6] Fuente vía - https://es.wikipedia.org/wiki/RFID

2.1.1. Patrón de comunicación dispositivo a dispositivo

Este patrón de comunicación dispositivo a dispositivo [12] establece entre dos o más dispositivos una interconexión entre sí para transmitir información sin necesidad de un servidor de aplicaciones como intermediario.

La comunicación entre los dispositivos se suele realizar a través de redes IP o Internet [12], pero, cuando se establece comunicación directa como ocurre en este proyecto, se utilizan protocolos como Bluetooth[7], Z-Wave[8] o ZigBee[9]. ZigBee es el protocolo de comunicación entre la puerta de enlace y las bombillas inteligentes. Ambos dispositivos están conectados entre sí y esto sirve para que la puerta de enlace mediante este protocolo transmita hasta las bombillas la acción que deben de realizar respecto a su estado.

Este modo de comunicación se utiliza para aplicaciones con transmisión de paquetes con tamaño de datos reducidos como son el caso de dispositivos IoT, para automatizar dispositivos en el hogar, ya sea por bombillas inteligentes como es el caso, medidor de temperatura, cierre de puertas de garaje, etc.

Hay un problema en el modo de comunicación y es que ambos dispositivos no pueden utilizar distintos protocolos de comunicación, es decir, la interoperabilidad en este punto es muy importante puesto que para conectar dos dispositivos entre sí y poder transmitir información, deben utilizar un protocolo común. Hay incompatibilidad entre objetos que utilizan el protocolo Z-Wave, por ejemplo, con los dispositivos con ZigBee.

Ilustración 2: Modelo de comunicación dispositivo a dispositivo.

[7] Fuente vía - https://www.bluetooth.com/
[8] Fuente vía - http://www.z-wave.com/
[9] Fuente vía - http://www.zigbee.org/

2.2. Protocolos de comunicación en IoT

Para llevar a cabo la comunicación entre dispositivos inteligentes conectados a Internet, se utilizan distintos tipos de protocolos.

2.2.1. ZigBee

ZigBee [13] es un protocolo de comunicación inalámbrica creado para aplicaciones de seguridad y automatización en el hogar.

ZigBee Alliance [14] fue quien instauró este protocolo. La alianza se estableció en el año 2002 y está formada por un conjunto de empresas que trabajan con el fin de desarrollar, mejorar y crear estándares para aplicarlos en dispositivos que serán imprescindibles en un futuro dentro de la tecnología de Internet de las cosas.

Este protocolo está basado en el estándar 802.15.4 [15] generado por el Instituto de Ingenieros Eléctricos y Electrónicos (IEEE), para desarrollar aplicaciones alimentadas con baterías en las que su tiempo de vida es largo y donde la transferencia de volumen de datos es reducida. Además, estás aplicaciones poseen conectividad simple, es decir, permite conexiones “punto a punto” o “punto a multipunto”.

ZigBee [16] transmite información a una tasa de velocidad que está entre 20 y 250 Kbps con un alcance de 30 a 100 metros. Además, operan en la banda de frecuencia de 2.4 GHz. ZigBee destaca por su bajo coste y su bajo consumo de energía. Este bajo consumo es debido a que después de transmitir la información al nodo, el protocolo se queda en modo suspensión para ahorrar energía.

Los dispositivos basados en ZigBee [16] deben de ser interoperables, es decir, se puede intercambiar información entre ellos incluso cuando los mensajes están encriptados por motivos de seguridad.

La red ZigBee [17] está formado por nodos que se conectan entre sí para transmitir la información entre ellos.

Cada nodo pertenece a una de estas tres categorías:

- **Coordinador ZigBee:** Es el nodo fundamental puesto que es el que se controla la red y quién establece la comunicación entre los dispositivos.

- **Router ZigBee:** Esta categoría es un punto inferior al coordinador y su función es interconectar los dispositivos para llevar a cabo el correcto funcionamiento de la información que se transmita.

- **Dispositivo final ZigBee:** Es el último punto en la transmisión. El dispositivo final es el que recibe la instrucción que debe realizar y está conectado solamente con el que le transmite la información. Este nodo es el único que mientras no reciba datos puede estar ausente ahorrando energía.

La topología de red en la que pueden estar estructurados los nodos son: estrella, malla o grupo de árboles.

- **Red en estrella:** El nodo central es el coordinador que seleccionará un identificador para la red. Todos los dispositivos finales o router se comunican directamente con el coordinador que será quién transmita la información.

- **Red en malla:** En este tipo de red, el nodo central también es el coordinador. Los nodos están comunicados entre sí con el fin de transmitir la información hasta el último nodo. En este tipo de red, los router pueden actuar como coordinadores. Los dispositivos finales reciben la información, pero no pueden transmitirla como los router.

- **Red en grupo de árboles:** Esta red forma una estructura tipo árbol con varias estrellas. Cada estrella formaría un grupo y el conjunto de ellas es lo que se le denomina grupo de árboles. Al haber distintas estrellas, en el primer árbol está el coordinador y en el resto, están los router transmitiendo la información a los dispositivos finales.

La red en malla es la tipología más utilizada en el protocolo ZigBee puesto que es la más eficiente. Esto es gracias a que, si algún nodo entre medias está desconectado, se busca una ruta alternativa para transmitir la información del nodo origen al nodo destino.

2.2.2. CoAP

The Constrained Application Protocol (CoAP) [18] es un protocolo especializado en dispositivos inalámbricos restringidos de baja potencia. Se diseñó para aplicaciones en el que la comunicación se produce entre máquinas, en inglés, *machine to machine* (M2M), dentro de una red restringida.

CoAP implementa un protocolo de transferencia web que se basa en REST (en inglés, *REpresentational State Transfer)*. REST es una arquitectura software que se basa en el protocolo de transferencia de hipertexto *(HTTP – Hipertext Transfer Protocol)* para transmitir y recibir datos entre cliente y servidor.

Los métodos que se utilizan para realizar peticiones con REST son: *POST*, *GET*, *DELETE* y *PUT*. Estos métodos son los mismos que los que se utiliza en HTTP y no se encapsulan. Para identificar los recursos REST utiliza URIs que son identificadores de recursos

uniformes (en inglés, *Uniform Resource Identifier*) que, a partir de una secuencia de caracteres, identifican correctamente el recurso especificado.

Los principales objetivos del protocolo CoAP son:

- Diseñar un protocolo de web global.
- Desarrollar un subconjunto de REST con HTTP para la utilización en programas máquina a máquina (M2M).

Características

Las características principales de este protocolo son:

- Intercambio de mensajes asíncronos.
- Soporte a través del identificador de recursos uniforme (URI) y el tipo de contenido.
- Vinculación a la seguridad de la capa de transporte de datos, DTLS.
- Proxy simple y capacidad de almacenamiento en caché.
- Protocolo web que satisface las condiciones de M2M en entornos restringidos.
- Instaurar proxies a partir de mapear el protocolo HTTP sin estado, y así poder acceder a los recursos de CoAP mediante el protocolo HTTP.
- Reducción de sobrecargas TCP (en inglés, *Transmission Control Protocol*) por medio de una vinculación UDP (en inglés, *User Datagram Protocol*) en opciones de fiabilidad unicast y apoyo multicast de peticiones.

Modelo de mensajería

CoAP utiliza un modelo de mensajería basado en el protocolo de transporte UDP o a través de DTLS para el intercambio de mensajes entre "Host".

El protocolo de transporte UDP no es fiable en la transmisión de paquetes, puesto que, pueden aparecer duplicados, perder los mensajes o llegar fuera de orden. El mensaje está formado por un encabezado binario de 4 bytes en el que puede ir seguido de opciones binarias y el payload, que es la carga útil del mensaje.

Entre host se pueden intercambiar gran cantidad de mensajes por segundo, debido a eso, para evitar que se dupliquen y asegurar la fiabilidad del mensaje, se añade un ID de 16 bits.

Al enviar una solicitud de tipo *confirmable* (CON), se activa el tiempo de espera hasta que el destinatario envía un mensaje de asentimiento (ACK) con el mismo ID que la

solicitud para indicar a qué mensaje hace referencia. El mensaje se retransmitirá si el contador de tiempo es menor que el tiempo máximo de retransmisión. Si se supera este valor, se anula la retransmisión de más mensajes puesto que al no recibir respuesta, el proceso informa de que ha fallado la comunicación entre ambos nodos. Si el destinatario no puede procesar la solicitud, no enviará ACK, sino que enviará un *reset* (RST), un mensaje de reinicio.

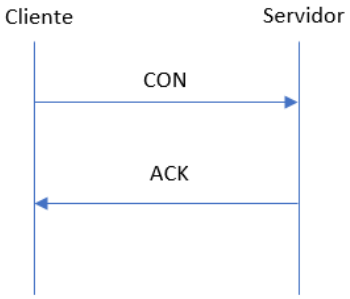

Ilustración 3: Transmisión de mensajes de tipo Confirmable.

Por el contrario, si el mensaje de solicitud es de tipo *non-confirmable* (NON), no se envía un ACK, sino que el destinatario envía un RST si el mensaje ha sido recibido. En el mensaje de solicitud también se inserta el ID del mensaje.

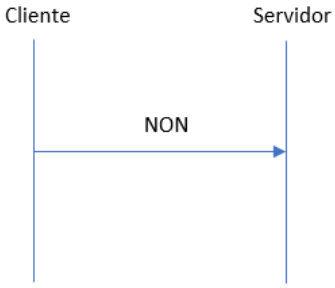

Ilustración 4: Transmisión de mensaje de tipo Non-Confirmable.

Formato de un mensaje

El formato de un mensaje CoAP está constituido en tres partes:

- **Cabecera CoAP:** La cabecera es un conjunto de datos necesarios para reconocer un mensaje y transmitir datos característicos.

  Los campos de la cabecera son:

- ***Versión (Ver):*** Campo que se muestra con 2 bits sin signo. Indica el número de versión del protocolo CoAP. Si el mensaje contiene un número de versión desconocido se debe de ignorar el campo.

- ***Tipo (T):*** Número entero sin signo formado por 2 bits. Este campo indica que tipo de mensaje de solicitud envía el mensaje cliente, *Confirmable* (0), *Non-confirmable* (1), *ACK* (2) o *Reset* (3).

- ***Longitud del Token (TKL):*** El token está formado por 4 bits sin signo. La longitud del campo *Token* comprende de 0 a 8 bytes. Los valores entre 9 y 15 están reservados. El token se utiliza en el campo de la cabecera para relacionar la respuesta con su determinada solicitud.

- ***Código:*** Este campo contiene 8 bits sin signo. Estos bits están divididos en dos partes: una clase a la que le constituyen 3 bits y los detalles formados por los 5 bits restantes.
  Dependiendo de si el mensaje es una solicitud o una respuesta entonces el campo contiene el método o el código de respuesta respectivamente.

- ***ID:*** Último campo de la cabecera formado por 16 bits sin signo. Este campo se utiliza para detectar si se ha duplicado un mensaje de solicitud y para relacionar los mensajes CON o NON con las respuestas ACK o RST.

- **Opciones:** Campo que se envía a continuación de la cabecera. Se envía el número de opciones, si el valor es cero es que no hay ninguna opción.

- **Carga útil:** La longitud de la carga útil depende del tamaño del datagrama, es decir, el cuerpo del mensaje.

Opciones CoAP

Los mensajes de solicitud y respuesta pueden contener un conjunto de opciones. Cada opción está definida por su longitud y valor. Las opciones están calculadas a partir de la suma del número de opciones que se han definido en el mensaje anterior, que deben de establecerse en el orden definido más un delta de codificación.

El formato de las opciones en el mensaje es:

- **Opción Delta:** Número de 4 bits sin signo. El valor debe estar entre el 0 y 12. Esta opción Delta indica la diferencia entre el número de opción actual con el número del mensaje anterior.

- **Longitud:** Número entero de 4 bits sin signo. Se refiere a la longitud del valor de la opción.

- **Valor:** Una secuencia de exactamente la longitud de opción en bytes. El formato de la longitud y el valor depende de la opción.

Las opciones principales son:

- **Uri-Host, Uri-Port, Uri-Path y Uri-Query:** Estas opciones se utilizan para indicar al servidor que recurso está identificando el nodo cliente. Estas opciones especifican el host destino, el puerto, la ruta del recurso y la consulta del mismo respectivamente.

- **Content-Format:** Opción donde se indica el formato con el que se representa la carga útil del mensaje.

- **Accept:** Indica que Content-Format es válido para el cliente.

- **Max-Age:** Es el tiempo que el cliente tiene que esperar hasta que recibe la respuesta del servidor.

- **E-Tag:** La etiqueta de entidad es generada por el servidor del recurso y sirve como el identificador local del mismo. Se utiliza para diferenciar las representaciones de recursos que se modifican con el tiempo.

Solicitud/Respuesta CoAP

CoAP ofrece un tipo de interacción solicitud/respuesta entre host o nodos. Host son los puntos finales donde se realiza la comunicación mediante este protocolo. Este tipo de interacción es muy similar al protocolo de transferencia de hipertexto (HTTP) con la excepción, de que CoAP utiliza el protocolo de transporte UDP para realizar el intercambio de mensajes de manera asíncrona.

El proceso para solicitar, mediante el protocolo CoAP, una o más acciones de un recurso a un servidor se realiza a través de los métodos de peticiones que son idénticos a lo de HTTP. El nodo que solicita la acción es el cliente y solicita uno o más recursos contenidos dentro del servidor. Una vez enviada la solicitud, el servidor enviará una respuesta para

indicar al cliente si la petición ha tenido éxito o si, por el contrario, se ha producido un error.

Si el cliente quiere realizar una petición, este proceso se llevará a cabo mediante una solicitud CoAP de tipo *Confirmable* (CON) o *Non-Confirmable* (NON). Esta solicitud está formada por un método que se efectúa sobre un recurso, el identificador del mismo, la carga útil y metadatos referentes a la solicitud.

Los métodos que se utilizan para la solicitud son:

- **GET:** Este método se basa en obtener la información del recurso que se ha solicitado mediante la URI. Si se obtiene correctamente la información se recibirá en la respuesta el código 2.05 o 2.03 indicando que el método y su contenido es válido.

- **POST:** Con este método el cliente solicita al recurso identificado que se procese la representación contenida. Esto puede provocar cambios o consecuencias en el servidor. Si se ha creado un nuevo recurso el código de respuesta es 2.01 y en la respuesta debería de incluir la URI del recurso nuevo que se ha generado. Si no se ha creado, sino que se ha eliminado el código será 2.04, y, por el contrario, si la consecuencia de la solicitud es la eliminación del recurso entonces el código de respuesta es 2.02.

- **PUT:** El cliente solicita con este método que el recurso identificado se reemplace con los datos recibidos en la carga útil o si por el contrario no está creado, que se cree. En este caso si se ha modificado el recurso, en la respuesta se recibirá el código 2.04 y si al no existir ningún recurso con esa identificación y se ha tenido que crear, se enviará el código 2.01.

- **DELETE:** Este último método solicita al servidor que se elimine el recurso con el identificador que se ha enviado en la solicitud. Si se ha eliminado con éxito, en el mensaje de respuesta debería de recibir el cliente el código 2.05 para informarle de que se ha realizado correctamente la acción especificada.

En el caso de que el método de solicitud escogido no sea válido, se obtendrá una respuesta de tipo 4.05. Esta respuesta indica que el método solicitado no está permitido.

Después de recibir una solicitud por parte del cliente, el nodo que actúa como servidor generará una respuesta CoAP, que para indicar que es la respuesta de una solicitud se utiliza un token que ha sido generado por el cliente en la solicitud. Este token es una secuencia de 0 a 8 bytes, y es como el identificador de la solicitud que se ha enviado.

Para identificar la respuesta se utiliza el campo Código en el encabezado del protocolo de comunicación CoAP. Este código dependerá del resultado de la solicitud. Es decir, a partir del resultado de la solicitud en el servidor, se enviará una respuesta con el formato "c.dd" donde la "c" es la clase de respuesta que se envía y la "dd" son los detalles de la respuesta.

Hay tres tipos de clases diferentes para el código de respuesta:

- **Éxito:** La "c" obtiene valor 2 indicando que la solicitud fue recibida y procesada correctamente. Los tipos de respuesta que hay en esta clase son:

CÓDIGO DE RESPUESTA	
2.01	Creado
2.02	Eliminado
2.03	Válido
2.04	Modificado
2.05	Conforme

Tabla 1: Código de posibles respuestas en caso de éxito.

- **Error por parte del cliente:** En este caso, la letra "c" toma el valor 4. La solicitud enviada contiene un error de sintaxis.

CÓDIGO DE RESPUESTA	
4.00	Solicitud Incorrecta
4.01	Cliente no autorizado
4.02	Opción inválida
4.03	Prohibido
4.04	No encontrado
4.05	Método no permitido
4.06	No aceptable
4.12	Falló en la precondición
4.13	Entidad de solicitud demasiado grande
4.15	Contenido no admitido

Tabla 2: Código de posibles respuestas por error del cliente

- **Error por parte del servidor:** El número correspondiente a la "c" en este error, es el 5. Indica que el servidor no proceso una solicitud que está correcta.

CÓDIGO DE RESPUESTA	
5.01	No implementado
5.02	Mala puerta de enlace
5.03	Servicio no disponible
5.04	Tiempo de espera de puerta de enlace
5.05	Proxying no admitido

Tabla 3: Código de posibles respuestas por error del servidor.

En este intercambio de mensajes entre el cliente y el servidor se utiliza en ciertas ocasiones la técnica "*Piggybacked*", que consiste en que el servidor responde al cliente enviándole una respuesta de tipo ACK siempre y cuando la solicitud sea de tipo *Confirmable*. La respuesta se envía independientemente de si la solicitud que ha generado el cliente ha sido exitosa o no.

URI de CoAP

Para identificar y localizar los recursos se utiliza esquemas de URI con "coap" si son solicitudes con CoAP o "coaps" si el protocolo CoAP está protegido con DTLS. Además, proporciona un lugar para situar el recurso identificado. En el proyecto, el servidor atenderá las solicitudes CoAP que están protegidas por el protocolo de seguridad DTLS.

El esquema URI con este protocolo es el siguiente:
coaps-URI = "coaps:" "//" host [":" port] path- abempty ["?" query]

En el apartado de host, se indica una dirección IP para acceder al servidor CoAP. Si el host tiene un nombre registrado, el punto final puede utilizar DNS (en inglés, Domain Name System), que es un sistema de nombres de dominio para traducir los nombres en identificadores y así, obtener la dirección del host. El host no puede estar vacío sino la solicitud no sería válida.

El siguiente valor que se debe introducir en la URI es el puerto UDP en el que se encuentra el servidor CoAP. En este caso, en el que la solicitud está protegida con DTLS, el puerto predeterminado es el 5684. Le prosigue la ruta que es donde se identifica el recurso escogido. Está formada por un conjunto de valores separados por el carácter "/".

Enlace DTLS sobre CoAP

Como DTLS se diseñó para generar seguridad en la transmisión de información entre puntos finales. Por ello, los dispositivos pueden utilizar uno de los cuatro modelos de seguridad.

Estos modos de seguridad son: *NoSec*, *PreSharedKey*, *RawPublicKey* y *Certificate*.

En este proyecto se utilizó el modelo de seguridad *PreSharedKey* [19] (PSK) que consiste en una autentificación mediante claves pre compartidas. Estas claves se comparten entre los puntos finales tanto del cliente como del servidor para proteger la comunicación entre ellos.

Este modelo de seguridad solo se puede llevar a cabo si DTLS está habilitado. Se crea esta autenticación cuando se quiere realizar una conexión con un punto final. Al iniciar la conexión, el sistema genera una clave adecuada dependiendo a que nodo se quiere conectar y a partir de ese nodo, se implementa una conexión mediante el protocolo DTLS utilizando el modo de seguridad PSK.

2.2.3. DTLS

El protocolo *Datagram Transport Layer Security* (DTLS) [20] está diseñado para que durante la comunicación entre nodos se garantice la protección de datos. Es decir, la función principal de DTLS es evitar que se obtenga información o se reciban mensajes de nodos exteriores sin permiso dentro de una comunicación protegida entre el cliente y servidor. DTLS está basado en el protocolo TLS (en inglés, *Transport Layer Security*) cuya función es proteger la comunicación y el tráfico de ambos nodos en la red.

Durante la transmisión de datagramas entre el cliente y el servidor, la comunicación no sufre ningún retraso en los mensajes pero sí se pueden producir otros inconvenientes. El paquete de datos que forma el datagrama puede verse afectado por lo que se conoce como paquete recibido fuera de orden. Esto quiere decir, que el paquete o los paquetes no han sido entregados al destinatario en el mismo orden que se han enviado.

Otros dos inconvenientes habituales en la transmisión de datagramas en este protocolo, son las pérdidas de los paquetes de datos y el exceso de datos en la carga útil en un datagrama.

El protocolo TLS no se puede utilizar directamente para el transporte de datagramas debido a los inconvenientes que se producen en la transmisión. El objetivo del protocolo DTLS es diseñar un protocolo idéntico a TLS salvo por mínimas modificaciones para que sea capaz de realizar el transporte de datagramas correctamente.

Hay tres cambios significativos en los mensajes de saludo entre el protocolo DTLS y TLS:

- Intercambio de cookies sin estado para evitar la denegación de servicio.
- Tiempo de espera y retransmisión.
- Fragmentación y re-ensamblaje de mensajes.

Respecto al primer cambio, la seguridad de los datagramas puede verse afectada por dos ataques que pueden provocar que el servicio de conexión entre dos nodos se pierda.

El primer ataque que se puede producir es el llamado *"ataque de amplificación"*, donde el cliente transmite al destinatario el mensaje *"ClientHello"* para iniciar la conexión. Sin embargo, el servidor envía un mensaje de tamaño excesivo provocando un desbordamiento en el cliente.

El segundo ataque es el llamado *"consumo de recursos"*, donde el intruso transmite un número elevado de mensajes *"ClientHello"* con la finalidad de consumir los recursos del servidor puesto que con cada mensaje de conexión el servidor creará una nueva sesión y le asignará un recurso.

Para solucionar estos dos problemas, el protocolo DTLS incluye un mensaje llamado *"HelloVerifyRequest"* el cual se transmite en respuesta al mensaje del cliente *"ClientHello"*. *"HelloVerifyRequest"* es un mensaje que se transmite al cliente junto con una cookie asociada. Esta cookie se utiliza para comprobar que nadie ha utilizado esta dirección y, además, como el mensaje no es de un tamaño elevado, el cliente retransmitirá el mensaje *"ClientHello"* con la cookie anexa impidiendo enviar más mensajes de inicio de conexión del servidor. Después de estos mensajes, el servidor continúa la comunicación con un *"ServerHello"*.

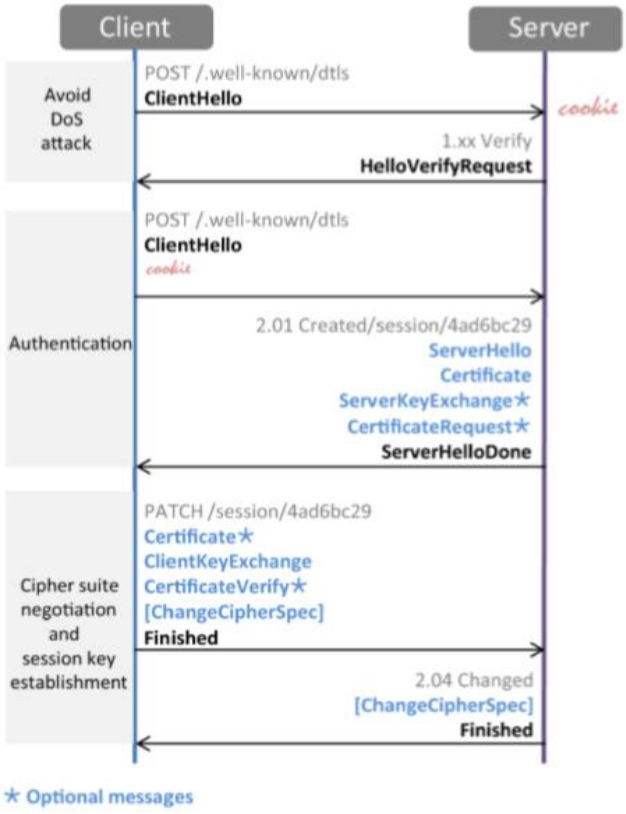

Ilustración 5: Intercambio de mensajes con la cookie para impedir la denegación del servicio. [21]

El otro problema es acerca de *"handshake"*. En el protocolo TLS los mensajes de transmisión y recepción entre el cliente y el servidor deben seguir un orden. En el transporte de datagramas no se produce dicho orden debido a que como se ha comentado anteriormente, los mensajes no llegan en el mismo orden que se enviaron y, además, hay mensajes que se pierden antes de llegar al destinatario.

El protocolo DTLS soluciona estos inconvenientes de la siguiente forma:

- **Pérdida de paquetes:** En este caso, el protocolo soluciona la pérdida de paquetes con un temporizador. Este temporizador se activa al enviar el mensaje inicial "*ClientHello"*, si el mensaje de respuesta "*HelloVerifyRequest*" enviado por el servidor se pierde, entonces al expirar el temporizador se retransmite el mensaje del cliente "*ClientHello*". El servidor también contiene un temporizador para retransmitir su mensaje cuando se expira el temporizador.

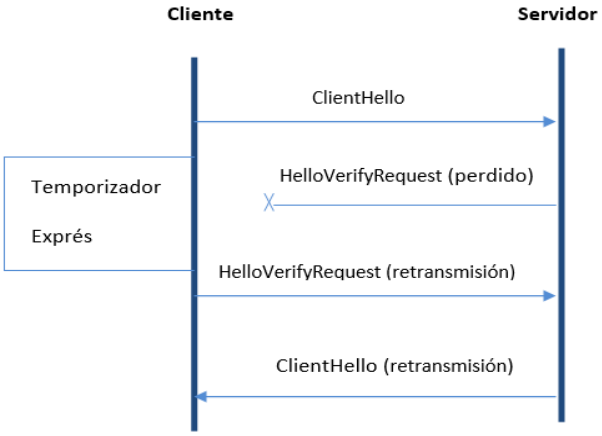

Ilustración 6: Solución a la pérdida de paquetes gracias al temporizador.

- **Reordenación:** Para llevar a cabo con éxito la reordenación de paquetes, DTLS inserta en cada mensaje de saludo un número de secuencia específico para cada uno y si el destinatario al recibir el paquete comprueba mediante el número que es un paquete desordenado, entonces, dicho paquete se quedará en espera hasta que lleguen los paquetes que le preceden y así, se consigue ordenar correctamente los datagramas enviados en la comunicación entre nodos.

Los mensajes que se utilizan para establecer conexión son de mayor tamaño que cualquier otro datagrama y por ese motivo, se produce lo que se conoce como fragmentación IP.

- **Tamaño del mensaje:** Los mensajes de saludo entre ambos nodos tanto en el protocolo TLS como en DTLS suelen ser de un tamaño elevado en comparación con el datagrama inicial del protocolo UDP.

La solución propuesta por el protocolo DTLS es fragmentar el mensaje en varios registros donde cada registro se insertará en un datagrama IP individual. El destinatario una vez que reciba los mensajes fragmentados puede desfragmentarlos para recuperar el mensaje original.

A continuación, se va a explicar el protocolo de registro que es idéntico al protocolo TLS salvo por la introducción de dos nuevos campos, el número de secuencia y época.

- **Época:** Es un campo numérico constituido por 16 bits. Ambos nodos tanto el cliente como el servidor utilizan este número para decretar que estado de cifrado se ha llevado a cabo para la protección de la carga útil. El valor de época se incrementa cuando se envía un mensaje *"ChangeCipherSpec".*

- **Número de Secuencia:** Este campo está formado por 48 bits, y se inserta en los registros para que no se produzca la reproducción de mensajes. Este número se incrementa de uno en uno para cada registro y se restaurará a su valor inicial cero cuando se transmite el estado de cifrado en una renegación de la sesión.

2.2.4. LWM2M

M2M, se basa en la comunicación entre dos dispositivos terminales cliente/servidor a partir de una conexión inalámbrica o cableada sin necesidad de que el usuario interactúe como intermediario.

Internet de las cosas se centra en este entorno de comunicación porque el coste de la operación es menor que otros tipos de comunicación, hay un número elevado de puntos finales y el punto clave, porque como en el concepto Internet de las cosas el tamaño de los datos que se transmiten no es elevado, esta comunicación es perfecta para ello.

El protocolo LWM2M [22] (*LightWeightM2M*), fue publicado por OMA (*Open Mobile Alliance*), una organización constituida para editar, desarrollar y realizar acciones sobre normativas técnicas en el ámbito de la telefonía móvil. El protocolo se basa en estándares de seguridad IETF[10](*Internet Engineering Task Force*) y su arquitectura está diseñada a partir de REST.

Este protocolo se centra en la gestión de dispositivos para las tecnologías de M2M e Internet de las cosas que es capaz de controlar remotamente los dispositivos a partir de CoAP como modo de transporte. Con este protocolo se consigue la comunicación entre el teléfono inteligente a través de la App *"IKEA Trådfri"*[11] y la puerta de enlace [23].

[10] Fuente vía - https://www.ietf.org/standards/rfcs/
[11] Fuente vía - https://play.google.com/store/apps/details?id=com.ikea.tradfri.lighting&hl=es

LWM2M aparte de gestionar dispositivos, es capaz de transmitir paquetes de datos de Internet al punto final, el servidor.

2.3. Raspberry PI

Este proyecto se realiza a través del hardware Raspberry Pi [24]. Aunque en el mercado existen distintos modelos, el utilizado para este trabajo es la Raspberry Pi 3 Modelo B[12], la última versión del producto. La Fundación Raspberry creó la Raspberry Pi sacando al mercado el primer hardware el 29 de marzo de 2012, y el mismo día, pero cuatro años más tarde, surgió la Raspberry Pi 3 Modelo B.

Este hardware es una pequeña CPU que contiene diversas mejoras en comparación con la versión anterior Raspberry Pi 2 Modelo B.

El primer avance respecto a las anteriores es que es la Raspberry Pi 3 Modelo B es más rápida y potente puesto que contiene un procesador de cuatro núcleos a 1,2 GHz. Mientras que la versión anterior, la Raspberry Pi 2 Modelo B tenía un procesador de cuatro núcleos de 900 MHz. Aparte de esta característica, para poder realizar aplicaciones IoT, la Raspberry Pi 3 Modelo B[13] está preparada para ello puesto que contiene dos herramientas nuevas como son la conexión inalámbrica WI-FI y el Bluetooth 4.1 de bajo consumo.

Aparte de las mejoras que trae el nuevo modelo de Raspberry Pi, la placa contiene otras características importantes como cuatro puertos USB, puerto HDMI, conexión a la red por cable, etc. En la siguiente ilustración se muestran dónde están situados todos los componentes de la placa:

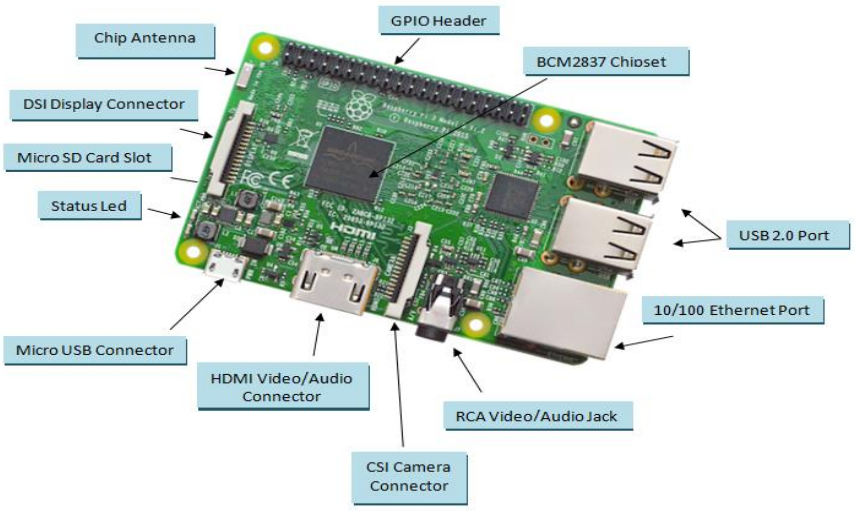

Ilustración 7: Partes de la Raspberry Pi

[12] Fuente vía - https://www.raspberrypi.org/products/
[13] Fuente vía - https://developer.android.com/things/hardware/raspberrypi

El logotipo de la Raspberry Pi lo creó Paul Beech a elección de todos los usuarios:

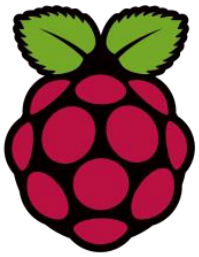

Ilustración 8: Logotipo de Raspberry Pi. Fuente: https://www.raspberrypi.org

Una vez analizadas las características de la Raspberry Pi 3 Modelo B, se va a proceder a explicar el sistema operativo que se insertó en la placa y el lenguaje de programación con el que desarrolló ambas aplicaciones.

2.3.1. Raspbian

Raspbian[14] es el sistema operativo que se instaló en este proyecto para la Raspberry Pi y es el recomendado por la Fundación Raspberry. Raspbian utiliza internamente el sistema operativo GNU/Linux basado en software libre, es decir, se puede utilizar todo el software interno para cualquier finalidad. El nombre del sistema operativo Raspbian surge de la unión entre Raspberry y Debian que es la sociedad que sustenta el sistema operativo GNU/Linux libre.

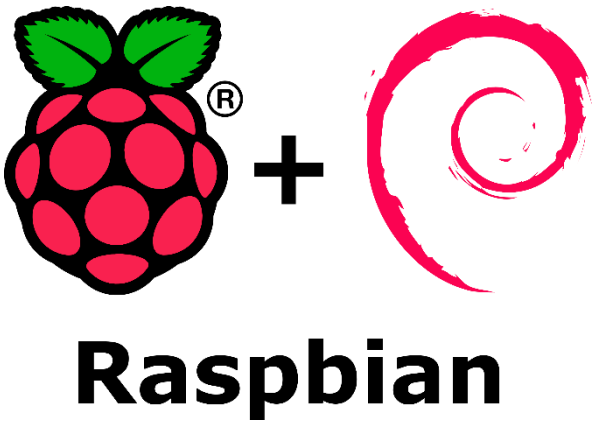

Ilustración 9: Raspbian - Combinación de Raspberry Pi y Debian. Fuente: https://www.pistachitos.com

En la página oficial de Raspberry Pi se puede descargar el sistema operativo de dos modos:

[14] Fuente vía - https://www.raspberrypi.org/downloads/raspbian/

- **NOOBS**

  NOOBS[15] es un instalador que contiene el sistema operativo Raspbian. Es lo más utilizado por los usuarios principiantes puesto que es un asistente que te ayuda a instalar Raspbian en la Raspberry Pi.

  Dependiendo donde se compre la Raspberry Pi, la tarjeta microSD puede contener este asistente instalado ya en su interior para ayudar al usuario a instalar Raspbian con facilidad. Este instalador no solo funciona para Raspbian, también se pueden instalar otros sistemas operativos que previamente se han descargado de Internet.

- **Raspbian**

  La otra opción es no instalar el sistema operativo con el asistente, sino que el usuario lo instala personalmente. Esta opción es para usuarios avanzados debido a que lo que te descargas es un .zip que contiene una imagen del sistema operativo que mediante un programa externo hay que escribir en la tarjeta microSD.

  Hay dos modelos de Raspbian que se pueden instalar:

  - Raspbian[16]: El sistema operativo Raspbian genérico que contiene el entorno gráfico para poder utilizarlo como un escritorio. Es decir, el usuario podrá visualizar el sistema operativo con las herramientas básicas, menú, fondo de pantalla, etc.

  - Raspbian Lite: Es la otra opción de descargarse Raspbian, pero esta no contiene el entorno gráfico, con lo que su tamaño es más reducido. Es decir, el usuario deberá de controlar la Raspberry Pi mediante el terminal de Linux. Esta opción está enfocada para usuarios con alto conocimiento en Linux que vayan a trabajar con la Raspberry en modo servidor.

El proyecto se realizó en lenguaje de programación Python. Raspbian contiene en su interior el compilador para este lenguaje.

Al escribir en la terminal “sudo raspi-config” se abre un menú oculto en el que se pueden ajustar diversas configuraciones. En el proyecto se modificó la opción de expandir la partición para que conseguir que ocupase toda la tarjeta de memoria.

[15] Fuente vía - https://www.raspberrypi.org/documentation/installation/noobs.md
[16] Fuente vía - https://www.raspberrypi.org/downloads/raspbian/

2.3.2. Python

Python [25] es un lenguaje de programación creado por Guido Van Rossum como sustitución al lenguaje ABC. Actualmente, la organización Python Software Foundation[17] (PSF) se dedica a impulsar, proteger y avanzar en el desarrollo del mismo.

Python es un tipo de lenguaje interpretado, esto produce un ahorro de tiempo al programador puesto que no se requiere la compilación del código para poder ejecutarlo. Su característica principal es que, aunque es un lenguaje potente, el tiempo de aprendizaje es corto.

Este lenguaje es de alto nivel y destaca por su sintaxis formal y su tipado dinámico. Gracias a ambas características se consigue implementar programas complejos entre tres y cinco veces más reducidos en extensión de líneas de código que otros lenguajes de programación como Java o C. Esto produce un ahorro de tiempo en la implementación del algoritmo, aunque la ejecución del mismo no sea excesivamente rápida.

Python es un lenguaje multiparadigma[18], este concepto se basa en la forma de programación de los usuarios. Con esta característica se puede llevar a cabo una programación orientada a objetos, una programación imperativa y una programación funcional. Además, Python es un lenguaje legible ya que el código se estructura mediante sangrías para tener el programa más organizado con el fin de poder entender su contenido más fácilmente.

Python se desarrolló para el sistema operativo Unix, pero actualmente, es un lenguaje multiplataforma ya que es compatible con multitud de sistemas operativos como Mac Os, Windows, Linux, etc.

Por último, cabe comentar que Python cuenta internamente con una multitud de librerías que se pueden importar. Esto enriquece al lenguaje ya que puede captar un número elevado de funciones adicionales.

[17] Fuente vía - https://www.python.org/psf/
[18] Fuente vía - https://www.python.org/doc/

3. INSTALACIÓN E INICIACIÓN AL ENTORNO

3.1. Equipamiento

Antes de iniciarse en el mundo de Internet de las cosas y empezar a documentar como se comenzó a desarrollar el tutorial de aprendizaje de *Pimoroni*[19], que es el primer paso para conseguir el objetivo del proyecto, lo primero que se llevó a cabo fue investigar el funcionamiento de cada uno de los dispositivos que se van a utilizar para sacarle la máxima rentabilidad a la hora de desarrollar un software con el fin mejorar la calidad de vida de las personas.

Los dispositivos que se han utilizado son los siguientes:

- **Kit TRÅDFRI LED**

  Este kit contiene en su interior una puerta de enlace, un mando remoto y dos bombillas LED E27.

Ilustración 10: Contenido del kit Trådfri Led

La **puerta de enlace**[20] o también llamada, pasarela, es un dispositivo con distintas características las cuales dependiendo de su estado se pueden mostrar con un led blanco encendido, en intermitente o apagado.

  - **Encendido/Apagado:** Si la puerta de enlace está conectada, es decir, el dispositivo está alimentado, entonces se mostrará el led activado al lado

[19] Fuente vía - https://learn.pimoroni.com/tutorial/sandyj/controlling-ikea-tradfri-lights-from-your-pi
[20] Fuente vía - https://www.ikea.com/es/es/manuals/tradfri-regulador-iluminacion-inalambrico__AA-1947721-2_pub.pdf

del símbolo de encendido sobre la superficie de la tapa de la pasarela. Por el contrario, si esta desconectada no se encenderá el led porque al dispositivo no le estará llegando corriente.

- **Internet:** Esta funcionalidad tiene 3 estados. Si el led está encendido fijamente es debido a que el dispositivo está conectado a la red mediante el cable ethernet. Si el estado del led es intermitente es porque hay un problema en la conexión a Internet, y, en cambio, si el led está apagado es porque no hay conexión a Internet ya sea porque no se ha conectado el cable ethernet desde el dispositivo al router WI-FI del hogar o porque dicho router está desconectado.

- **Red:** Es otra característica de la puerta de enlace. El led estará activado si dentro de la red Trådfri hay dispositivos conectados. Aunque las bombillas estén apagadas si están emparejadas con el mando remoto entonces el led estará encendido indicando que está listo para usarse. Si el estado del led en vez de ser fijo, parpadea, quiere decir que no hay ningún dispositivo Trådfri conectado, un ejemplo claro, cuando el mando remoto está apagado y con ello el grupo de bombillas del hogar. Y, en el último caso, el led no estará activado cuando no hay ningún dispositivo Trådfri asociado con la pasarela, es decir, no está creada ninguna red Trådfri.

- **Acoplamiento:** Por último, aunque no tiene led que muestre ningún estado sobre la superficie de la puerta de enlace, la pasarela tiene una última funcionalidad que se realiza una única vez en el momento que se instala por primera vez el dispositivo junto con el mando remoto. Está función es crear la red Trådfri. El mando remoto se debe acercar aproximadamente en torno a 2 centímetros de la pasarela y se pulsa el botón de acoplamiento que se encuentra en la parte posterior del mando durante 10 segundos para emparejar ambos dispositivos. Con la pasarela solo se empareja el mando remoto, esté último es quien se encarga de emparejar las bombillas.

Otro dispositivo que contiene el kit es el **mando remoto**[21]. Este dispositivo está formado por 5 botones en su cara principal y un botón, que es el botón de emparejamiento, en el dorso, junto a la pila.

Los cinco botones de la cara superior del mando sirven para la configuración de todos los dispositivos que forman la red Trådfri. Los dispositivos que forman esta red son la puerta de enlace, el mando remoto y las bombillas asociadas.

El botón central sirve para encender o apagar las fuentes de iluminación. También,

[21] Fuente vía - https://www.ikea.com/es/es/manuals/tradfri-mando-a-distancia__AA-1895962-1_pub.pdf

este botón tiene la función de sincronizar las bombillas que hayan perdido dicha funcionalidad, esto se consigue, dejándolo pulsado un mínimo de 3 segundos para poner la bombilla con su configuración por defecto.

Los botones < y > se utilizan para cambiar el color de las bombillas y así, poder crear distintos ambientes en el dormitorio según la escena que se escoja. Los 3 tipos de colores o escenas a los que se pueden optar son: frio, normal y cálido.

Los otros dos botones que se encuentran en el lado opuesto a los botones de cambio de espectro, sirven para aumentar o disminuir la intensidad gradualmente.

Si se necesita reiniciar el mando remoto, se debe de pulsar cuatro veces el botón de emparejamiento en un intervalo de cinco segundos.

Para finalizar, el último dispositivo del kit son las **bombillas**[22] que son un tipo de luces tipo led que consumen 85% menos de energía que las convencionales y tienen un tiempo de vida estimado de veinte veces más duraderas.

Este tipo de bombilla tiene unas características especiales, ya que como se ha explicado anteriormente en el mando remoto, estas bombillas no solo se usan para proporcionar luz en cualquier lugar del hogar, sino que nos da la posibilidad de modificar su intensidad y su tonalidad creando un ambiente idóneo ante distintas situaciones. Por ejemplo, si el usuario está viendo una película la mejor opción sería poner las luces en un tono cálido con poca intensidad, pero, sin embargo, si el usuario necesita una luz más intensa en un momento determinado quizá la mejor opción sería poner las bombillas en un tono frio con una intensidad alta para dar luminosidad al lugar donde se encuentra.

La tonalidad de las bombillas fluctúa entre 2200 Kelvin que es la escena a la que se le toma el nombre como “Cálida” proporcionando un color anaranjado, 2700 Kelvin cuando la escena es “Normal” obteniendo un tono blanco cálido y 4000 Kelvin, siendo la escena “Frio” y el color es blanco frio.

- **Raspberry Pi 3 Starter kit for Android Things**[23]

  Este kit contiene una Raspberry Pi 3, el Rainbow Hat, una edición especial de Pibow Coupé, una fuente de alimentación para la Raspberry Pi y una tarjeta microSD de 16Gb.

[22] Fuente vía - https://www.ikea.com/es/es/manuals/tradfri-bombilla-led-e-lumenes__AA-1950947-1_pub.pdf
[23] Fuente vía - https://shop.pimoroni.com/products/rainbow-hat-for-android-things

La **Raspberry Pi 3 Modelo B**, es una pequeña CPU formada por 4 núcleos de 64 bits a 1,2GHz. Este pequeño ordenador tiene 4 puertos USB, un puerto HDMI, la posibilidad de conectarse a la red de manera inalámbrica, ya que este es uno de los pocos modelos capaces de conectarse mediante Wi-Fi y también, contiene un puerto ethernet para conectarse por cable. Además, tiene conexión Bluetooth que a diferencia de las versiones antiguas de este hardware carecían de esta característica y, por último, cabe destacar la conexión de 40 pines los cuales hace posible la adaptación de una interfaz física, como el Rainbow Hat que se ha utilizado en este proyecto, aunque hay más interfaces físicas aparte de este.

Respecto al **Rainbow Hat**[24], es una interfaz física que se sitúa sobre los 40 pines de la Raspberry Pi y con ella, se puede realizar distintos tipos de proyectos gracias a sus propiedades y características.

Este dispositivo está formado por 4 displays de 14 segmentos, los cuales se representan mediante un led en color verde. En la parte superior de los displays, se sitúan 7 leds multicolores que sirven para crear distintas funcionalidades en los proyectos que se desarrollen. Y en la parte inferior, hay 3 botones los cuales sobre los mismos tienen un led cada uno de distinto color, azul, verde y rojo respectivamente.

Esta interfaz también está compuesta por sensores de temperatura y presión, y un zumbador piezoeléctrico para generar sonidos mediante el protocolo PWM donde se da valor a las frecuencias y eso las convierte en notas.

Principalmente, Rainbow Hat es un hardware rico para los usuarios creativos ya que, mediante sus sensores, leds y displays hace posible crear proyectos innovadores.

El **Pibow Coupé** no es más que una edición especial de un tipo de carcasa o cubierta con el fin de proteger la Raspberry Pi.

Y, por último, la Raspberry Pi funciona mediante diferentes sistemas operativos, aunque el que se ha utilizado en este trabajo es Raspbian. Este software se instala en la **microSD** de 16Gb y el entorno donde se desarrolla es Linux.

[24] Fuente vía - https://www.raspberrypi.org/magpi/rainbow-hat-review/

Ilustración 11: Material del Starter Kit – Fuente: https://developer.android.com/things/get-started/kits

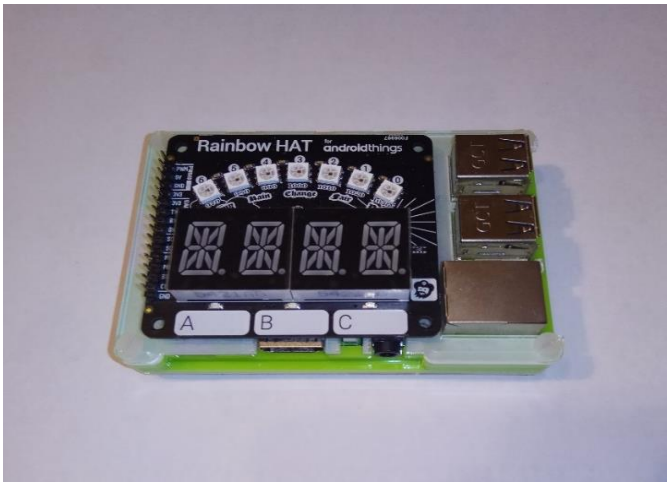

Ilustración 12: Resultado final del montaje de la Raspberry Pi, PIbow Case y el Rainbow Hat

- **Monitor, Ratón y Teclado**

  Para poder controlar la Raspberry Pi con su sistema operativo, Raspbian, se requiere de un monitor que irá conectado al puerto HDMI de la Raspberry. En mi caso, el monitor es de la marca Samsung del año 2005 y no lleva incorporado una conexión HDMI sino VGA con lo que se necesitó comprar un conversor de VGA a HDMI para poder utilizarlo.

  Con respecto al ratón y el teclado, que son dispositivos hardware, van conectados a los puertos USB de la Raspberry Pi.

Ilustración 13: Dispositivos auxiliares para el control de la Raspberry Pi

3.1. Instalación del entorno

En este apartado se expone los pasos que se han de seguir para instalar todos los dispositivos con el fin de poder ejecutar correctamente ambas aplicaciones.

3.1.1 Instalar el kit Trådfri Led

El primer paso para comenzar la instalación del kit Trådfri es insertar una pila botón de litio en el dorso del mando remoto. Para la apertura de la tapa se debe de hacer palanca con un utensilio plano.

La puerta de enlace es el primer dispositivo que se instala. Este dispositivo tiene dos conexiones, una de alimentación y otra de conexión de red la cual debe estar conectada mediante un cable ethernet al router. La puerta de enlace o pasarela tiene en la parte superior tres leds. Si se han seguido los pasos correctamente los leds de alimentación y conexión a la red deberían estar iluminados, el led restante es para indicar que el dispositivo esta emparejado con el mando remoto.

La red Trådfri se crea emparejando el mando remoto con la puerta de enlace. Para llevar a cabo el emparejamiento, se debe de acercar el mando remoto a la pasarela y no se debe de estar a una distancia superior a 2 cm. Cuando el mando este situado a dicha distancia de la puerta de enlace, se debe pulsar durante 10 segundos el botón que se encuentra en el dorso del mando remoto junto a la pila. Si se ha emparejado con éxito, el led de la puerta de enlace parpadea dos veces hasta quedarse fijo. Después de este proceso, la pasarela debe estar ya instalada y lista para utilizarse.

Una vez instalada la puerta de enlace, se debe proceder a emparejar las bombillas.

Primero, se deben de ubicar cada una en una lámpara. Una vez instaladas, se procede a emparejarlas con el mando remoto y a su vez, con la puerta de enlace. Para poder emparejar las bombillas deben de estar encendidas. El usuario debe colocar el mando remoto en una posición que no supere los 5 centímetros de distancia con la bombilla y manteniendo el botón del dorso del mando remoto pulsado se visualizará una luz roja en la cara frontal del mismo y al transcurrir 10 segundos la bombilla empezará a atenuarse y parpadeará indicando que el emparejamiento ha sido exitoso. Para emparejar la bombilla restante se debe de seguir el mismo procedimiento.

Una vez que ya se tienen todos los dispositivos conectados y emparejados ya estarían todos conectados a la misma red Trådfri.

En el caso de que se necesite desemparejar los dispositivos que estén conectados se hará el mismo procedimiento, pero la bombilla parpadeará dos veces en vez de una.

3.1.2 Montaje de la Raspberry Pi

El segundo paso es montar la Raspberry Pi[25] junto con su estuche protector. Para ello, se debe coger las dos placas de plástico y despegar las películas blancas que se encuentran adheridas en ambas capas. Todas las capas contienen un número en la parte superior izquierda menos la última capa que no está numerada. El usuario debe de buscar la capa con el número “0” que será la que se utiliza como base para la estructura. A continuación, se debe de colocar la capa “1” sobre la capa “0” y la capa “2” sobre la capa “1”. El usuario debe asegurarse de que todas las capas se colocan con el número en la esquina superior izquierda. Después, se debe de colocar la Raspberry Pi sobre la capa “2”. Una vez colocada la Raspberry Pi, para acabar de protegerla se coloca la capa “3” sobre la placa y a continuación, la capa no numerada sobre esta última.

Para fijar las capas con la Raspberry Pi se colocan cuatro pernos de plástico en las esquinas y se usará la llave incluida en el kit para apretar las cuatro tuercas con el fin de fijar los pernos.

Después, para acabar con el montaje queda colocar el hardware Rainbow Hat sobre el conector macho de 40 pines que posee la Raspberry Pi. El conector de la interfaz Rainbow Hat es de tipo hembra con lo que se debe colocar para que encajen los 40 pines y seguidamente presionar hacia abajo.

Por último, se debe de escoger el adaptador de salida correcto para el alimentador de corriente. Una vez escogido el adaptador, se debe deslizar sobre el adaptador de corriente hasta que encaje y se escuche un clic. Este sonido indicará que se ha colocado correctamente sobre el alimentador.

[25] Fuente vía - https://androidthings.withgoogle.com/#!/kits/raspberry-pi-3-starter-kit

3.1.3 Instalación del sistema operativo

Para instalar el sistema operativo en la Raspberry Pi, primero se debe de insertar la tarjeta microSD en el adaptador que contiene el starter Kit y después, introducirlo en la ranura del ordenador.

Una vez dentro, se debe formatear la tarjeta, para ello, se abre el administrador de discos del ordenador y se identifica el disco extraíble (la tarjeta microSD). Cuando ya esté identificado se elimina el volumen y la partición que contenga hasta dejarla vacía por completo. Después de formatearla, se crea un nuevo volumen.

El siguiente paso es descargar el sistema operativo Raspbian para la Raspberry Pi. Para ello, se debe de acceder a la página oficial de la Fundación Raspberry Pi (https://www.raspberrypi.org) y pinchar en la pestaña de descargas. Ahí aparecerán dos modos de descargar el sistema operativo Raspbian, a través del asistente NOOBS o mediante el enlace de Raspbian.

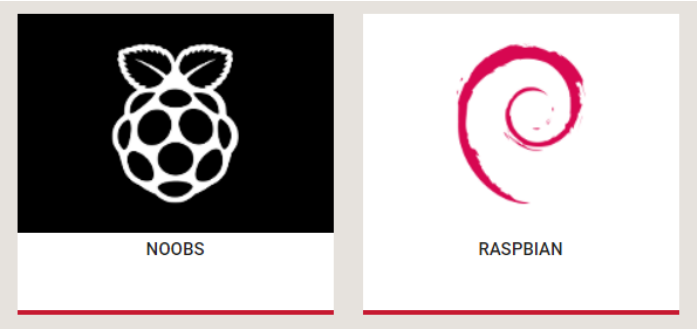

Ilustración 14: Dos modos de descarga (NOOBS y RASPBIAN)

El usuario puede descargar el sistema operativo con la opción que prefiera, en este caso, se va a explicar cómo se instala a partir del enlace Raspbian, es decir, sin el asistente.

Al pinchar en Raspbian aparecen dos modos de descarga, el primero con entorno gráfico que significa que el sistema operativo se muestra como un escritorio o, por el contrario, sin entorno gráfico, es decir, utilizando comandos. Se descarga Raspbian Lite si el nivel en Linux es avanzado puesto que la Raspberry se ejecutará como un servidor, si no es así, se debería descargar Raspbian.

Ilustración 15: Enlaces de descarga del sistema operativo

La descarga se puede llevar a cabo de dos modos, mediante torrent o por .zip. Una vez que se ha descargado el archivo se debe de descomprimir. Dentro de la carpeta aparecerá un archivo .img que es una imagen del sistema operativo. Para insertarlo en la Raspberry Pi, primero, hay que escribir la imagen mediante el programa WIN32Disk, se puede descargar el programa a través del siguiente enlace: https://sourceforge.net/projects/win32diskimager/.

Al abrir el programa WIN32DISK se observa la siguiente imagen:

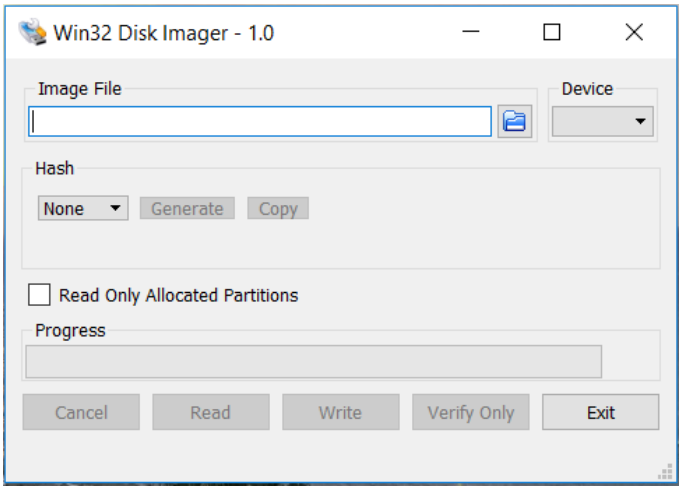

Ilustración 16: Imagen tras abrir WIN32 Disk Imager

En el bloque "*Image File*" se debe de adjuntar el archivo .img que se acaba de descargar y en la pestaña "*Device*" se selecciona el disco extraíble que será la tarjeta microSD.
A continuación, se pulsa el botón "*Write*" para escribir la imagen en el disco que se ha seleccionado. Al finalizar, saldrán ventanas emergentes que se deben cerrar indicando varios errores, esto es debido a que Windows no entiende el contenido que se ha escrito en la microSD.

Después de todo este proceso, la tarjeta microSD contiene el sistema operativo de la Raspberry Pi con lo que ya se puede expulsar la tarjeta del ordenador.

3.1.4 Configuraciones y actualizaciones

Antes de encender la Raspberry Pi se debe insertar la tarjeta microSD en ella y conectar el ratón, teclado y monitor para poder manejar y controlar el hardware.

Después de realizar las conexiones, se encenderá la placa y se visualizará como se inicia el sistema operativo Raspbian en la Raspberry. Cuando se muestre el escritorio, se podrá configurar los distintos ajustes de la Raspberry Pi y se deberá conectar mediante WI-FI o a través de cable ethernet al mismo router donde está conectada la puerta de enlace.

Una vez que se tiene acceso a Internet en la Raspberry Pi, es conveniente actualizar cada cierto tiempo el sistema operativo y los repositorios. Para ello, se debe pinchar en el cuarto icono empezando por la izquierda, el icono representa un terminal de Linux.

Cuando ya se ha conseguido abrir el terminal, el usuario debe escribir *"sudo apt-get update"* para actualizar el sistema operativo a la última versión. Una vez finalizado, el usuario debe escribir *"sudo apt-get upgrade"* para actualizar los repositorios.

ACTUALIZACIÓN DE LOS REPOSITORIOS	`sudo apt-get upgrade`
ACTUALIZACIÓN DEL SISTEMA	`sudo apt-get update`

Tabla 4: Comandos para actualizar los repositorios y el sistema

Cuando se ha finalizado ambos procesos, se debe ajustar las configuraciones básicas que se aconsejan modificar la primera vez que se ejecuta el sistema operativo. Estas configuraciones son el ajuste de idioma que por defecto está en inglés, el teclado, etc. Para llevar a cabo estos ajustes, el usuario tiene que clicar en el primer botón de la izquierda donde se muestra el logotipo de la Raspberry Pi. El siguiente paso es clicar en "*Preferences"* y a continuación, en "*Raspberry Pi configuration"*. Dentro de este submenú, en la pestaña de "*Localisation"* se ajusta el idioma del sistema y el idioma del teclado. El usuario en este menú de configuración también tiene la opción de modificar el nombre de la Raspberry, adjudicarle una contraseña, indicar como quiere que inicie sesión, etc.

Por último, aparte del menú de ajustes que aparece en la pantalla principal de la Raspberry Pi, hay otro menú oculto al que se accede mediante el comando:

MENÚ OCULTO	`sudo raspi-config`

Tabla 5: Comando para el acceso al menú oculto

3.2. Iniciación al entorno

Una vez que se ha explicado las funciones y las características de los dispositivos y se ha realizado la instalación de los mismos, se va a describir detenidamente como se inició en el entorno de Internet de las Cosas.

El objetivo de este apartado es desarrollar el tutorial de *Pimoroni* para aprender a controlar las bombillas mediante comandos CoAP sobre DTLS o a través de solicitudes de Python, ambos enviados desde la Raspberry Pi hasta la puerta de enlace y desde ahí, con el protocolo de comunicación inalámbrico ZigBee se transmite la información hasta las bombillas para su ejecución.

3.2.1 Pruebas con librería CoAP

El primer paso que se llevó a cabo fue la instalación de la librería "*libcoap*" con DTLS incluido, específicamente, la referida al cliente, es decir, coap-client que se utiliza para enviar comandos a la pasarela.

Para poder enviar solicitudes a la puerta de enlace mediante comandos coap-client se necesita la dirección IP y la clave de la pasarela. Para ello, se utilizó un analizador de red para identificar las direcciones IP de los dispositivos conectados a la red y la clave, se obtuvo del dorso del dispositivo que está formada por 16 dígitos.

Una vez que se instaló la librería, el siguiente paso que se llevó a cabo fue enviar una solicitud para apagar una bombilla mediante un comando coap-client por la terminal de la Raspberry Pi. Este comando está formado por información del host final y una carga útil.

SOLICITUD PARA APAGAR UNA BOMBILLA
`coap-client -m put -u "Cliente_identity" -k "Password_pasarela" -e '{"3311": [{"5850": 0 }] }' "coaps://192.168.X.XX:5684/15001/65537"`

Tabla 6: Comando de solicitud para el apagado de una bombilla

La información del host se declara con esta sentencia: *"coaps://192.168.X.XX:5684/15001/65537"*, donde se modificó la dirección IP por la que se obtuvo a través de la aplicación referenciada a la puerta de enlace. El código 15001 sirve para controlar las bombillas de manera individual, si en lugar de ese valor fuese el 15004, se controlaría de manera colectiva el conjunto de bombillas. A continuación, se introduce el valor 65537, indicando que el ajuste que se va a aplicar con este comando se va a realizar en la primera bombilla, si en vez de en la primera, se quiere llevar a cabo en la segunda se pondrá el código 65538.

Eso se lleva a cabo en el caso en que se quiera enviar información individualmente a una bombilla, si, por el contrario, fuese al grupo, después del 15004 se tendría que poner el código del grupo que se obtendrá más adelante.

La carga útil es el resto del comando enviado, *coap-client –m put –u "Cliente_identity" –k "PASSWORD_PASARELA" –e '{ "3311": [{ "5850": 0 }] }'*.

El comando comienza indicando el protocolo que se está utilizando para la conexión de ambos dispositivos, como se ha instalado la librería "*libcoap*" referida al cliente, se escribe coap-client al inicio de la sentencia seguido de *"–m put"* ya que se trata de una solicitud de envío. Después, se escribe "*–u "Cliente_identity""* que es el nombre de usuario al que se le asigna la responsabilidad del envío de la sentencia. A continuación, se le pasa la clave de la puerta de enlace *–k "PASSWORD_PASARELA"* donde hay que introducir sus 16 dígitos. Y por último, la acción que se quiere realizar, *"–e '{ "3311": [{ "5850": 0 }] }'"*, en este caso, el apagado de una de las bombillas ya que el código 3311 hace referencia a un atenuador y el código 5850 a un conmutador al que dependiendo el valor que se le pase como parámetro la bombilla se encenderá o se apagará. En este caso, como el valor es un 0 se apagará, por el contrario, si fuese un 1 se encendería.

Teniendo la primera bombilla encendida, se envió el comando por la terminal con el fin de apagarla, pero se produjo un error mostrando en el terminal el código 4.01.

Lo primero, se investigó que significaba el error 4.01 y como se podía resolver. Para comprobar si era problema con la comunicación, primeramente, se realizó un ping a la dirección IP de la pasarela y efectivamente, se comprobó que la comunicación era correcta. El segundo paso, fue investigar en la documentación de los protocolos de comunicación que se utilizan en el proyecto y se descubrió que el error 4.01 hacía referencia a un error por parte del cliente. El cliente no estaba autorizado para realizar operaciones sobre las bombillas.

Gracias a esta información, se percató de que el error se estaba produciendo en el usuario ya que el nombre no podía ser "Cliente_identity", por eso, se modificó el nombre a "Alex" y funcionó a la perfección.

Ahora que ya se podía enviar solicitudes, se probó si aparte del encendido y apagado también se modificaba la intensidad y el color de las bombillas. Para comprobarlo, se enviaron los siguientes comandos y se observó que se modificó el estado de la bombilla con éxito.

SOLICITUD PARA EL AJUSTE DE INTENSIDAD
`coap-client -m put -u "Client" -k "Password_pasarela" -e '{"3311": [{"5851": 127 }] }' "coaps://192.168.X.XX:5684/15001/65537"`

Tabla 7: Comandos de solicitudes para el ajuste de intensidad

SOLICITUD PARA EL AJUSTE DEL COLOR
`coap-client -m put -u "Client" -k "Password_pasarela" -e '{"3311": [{"5706": "cold" }] }' "coaps://192.168.X.XX:5684/15001/65537"`

Tabla 8:Comandos de solicitudes para el ajuste del color

Además, mediante comandos de la librería CoAP, aparte de enviar solicitudes, se puede realizar consultas sobre el estado de las bombillas para visualizar si están encendidas o apagadas, evaluar el nivel de intensidad y que escena se está generando.

La consulta se puede realizar de manera colectiva o individual:

COMANDO PARA LA RECEPCIÓN DEL ESTADO DE UNA BOMBILLA
`coap-client -m get -u "Client" -k "Password_pasarela" "coaps://192.168.X.XX:5684/15001/65537"`
COMANDO PARA LA RECEPCIÓN DEL ESTADO DE DOS BOMBILLAS
`coap-client -m get -u "Client" -k "Password_pasarela" "coaps://192.168.X.XX:5684/15004/GrupoId"`

Tabla 9: Comandos para la recepción del estado de las bombillas

3.2.2 Pruebas con solicitudes en Python

La siguiente tarea se basó fundamentalmente en realizar solicitudes de Python a través de la terminal de la Raspberry Pi para configurar las bombillas o recibir información sobre el estado de las mismas. Es decir, mismo procedimiento que el anterior, pero en este caso, no se utilizó la librería "*libcoap*" sino un contenedor de Python.

Lo primero, se instaló "*pip",* que es un administrador de los paquetes de Python que sirve para gestionar e instalar los paquetes de dicho lenguaje. Después de su instalación, se instaló el paquete "*tqdm*", que procede del árabe "*taqadum*" y significa progreso, es decir, muestra barras de progreso al solicitar información de las bombillas inteligentes. Con este paquete se consigue que la información se muestre de manera más dinámica y visual.

Por último, se clono el repositorio de "*GitHub*", para tener almacenada en la Raspberry su carpeta "*ikea-smartlight*" la cual contenía distintos scripts de Python que son los que se utilizarán para la configuración de las bombillas a partir de solicitudes.

Una vez instalado todos los paquetes necesarios y el correspondiente directorio, el siguiente paso fue crear un nano texto. Este archivo se creó en el editor de texto del sistema operativo Raspbian. En el archivo se escribió información sobre la dirección IP de la puerta de enlace, y la clave de 16 dígitos de la misma.

En esta parte, en vez de pasar dicha información directamente por el comando como se realizó en la tarea anterior, ahora al script que llamemos por la terminal pedirá acceso mediante el nano texto "*tradfri.cfg*". El nano texto quedó de la siguiente manera:

ARCHIVO TRADFRI.CFG

`[tradfri]`
`hubip = Dirección IP de la pasarela`
`securityid = Clave 16 dígitos`

Tabla 10: Contenido del nano texto tradfri.cfg

Ya con el archivo "*tradfri.cfg*" creado, se envió por primera vez una solicitud de envió con Python a través de la terminal para encender la primera bombilla. El comando que se lanzó fue:

COMANDO PARA ENCENDER LA BOMBILLA Nº1
`python tradfri-lights.py -l 65537 -a power -v on`

Tabla 11: Comando de solicitud para el encendido de la primera bombilla

El comando está formado por una llamada al script de Python, "*tradfri-lights.py"*, que es el código que se utiliza para la configuración de las distintas acciones que se pueden realizar en una bombilla individual. A continuación, le precede "*–l 65537"* donde la letra "l" especifica la bombilla que se desea controlar y el valor 65537 indica la bombilla. Respecto al siguiente código, "*-a power –v on"*, indica la acción que se va a llevar a cabo, en este caso, encender o apagar la bombilla y el "*–v on"* hace referencia al valor, como se envió un "*on*" se debería haber encendido la bombilla, pero la primera vez que se realizó esta acción no resultó así. Al ejecutar el comando, apareció el siguiente error: "*ValueError: No JSON object could be decoded"*.

Lo primero que se hizo fue analizar los procedimientos que se han llevado a cabo con el fin de encontrar el error. Principalmente, se centró en dos aspectos, en que la información del nano texto no estuviera cogiendo los datos correctamente y por ese motivo, la pasarela no estuviese aceptando la petición o que el tutorial de la página estuviese desactualizado y el comando de solicitud ya no se enviase de ese modo.

Investigando detenidamente en Internet, en el repositorio de "*github"*[26], se comprobó que era una mezcla de ambas opciones, es decir, la página sí que estaba desactualizada porque en el nano texto ya no se debe escribir la "*securityid*" debido a que a partir de la versión

[26] Fuente vía - https://github.com/hvanderlaan/ikea-smartlight

1.1.15 de la pasarela ya no se permite este uso. La versión de la puerta de enlace que se está usando para el desarrollo del proyecto es la 1.3.1 con lo que es superior a la permitida para la utilización de la *"securityid"*. Ahora, en lugar de poner la clave de 16 dígitos de la pasarela, se tuvo que enviar un comando registrando un usuario de API, y se recibió una clave, "*apikey"*. Esos dos valores son lo que se deberían de insertar en el nano texto en lugar de la "*securityid*".

COMANDO PARA RECIBIR LA CLAVE APIKEY
`coap-client -m post -u "Client" -k "CodigoSeguridadPasarela" -e '{"9090": "UsuarioAPI"}'  "coaps://IP_ADDRESS:5684/15011/9063"`

Tabla 12: Comando para registrar un usuario API y recibir la clave

ARCHIVO TRADFRI.CFG ACTUALIZADO

`[tradfri]`
`hubip = Dirección IP de la pasarela`
`apiuser = Nombre de usuario`
`apikey = Clave recibida`

Tabla 13: Archivo Tradfri.cfg actualizado

Aparte de modificar el archivo *"tradfri.cfg"*, como los scripts de Python que se descargaron del repositorio tomaban las variables *"hubip"* y *"securityid"* como parámetros de los métodos para solicitar el permiso a la puerta de enlace, se tuvo que cambiar la variable *"securityid"* por *"apiuser"* y *"apikey"* en todos los archivos donde recibía este campo.

Con todos los cambios ya realizados, se envió el comando para encender la bombilla número uno y funcionó a la perfección. Además, se envió el mismo comando, pero con acciones diferentes, como "*brightness*" y un valor comprendido entre 0 y 100 para cambiar la intensidad de la bombilla o el color que es la otra acción disponible cuyos valores posibles son *"cold"*, *"normal"* o *"warm"*.

A continuación, se enviaron dos comandos más. El primero, se basa en obtener el estado de cada bombilla e información relevante como el nombre asociado al grupo de bombillas y su código para poder controlar ambas al mismo tiempo.

COMANDO PARA RECIBIR EL ESTADO DE LAS BOMBILLAS
`python tradfri-status.py`

Tabla 14: Comando con el cual recibe el estado de las bombillas

COMANDO PARA CONFIGURAR EL GRUPO DE BOMBILLAS
`python tradfri-groups.py –g XXXXXX –a power –v on`

Tabla 15: Comando para la configuración de ambas bombillas

Este último comando, sirve para configurar ambas bombillas a la vez a diferencia del comando donde se configura las bombillas individualmente (Tabla 11), es decir, la acción que se le pasa por parámetro se realizará de manera grupal, no como en el caso anterior que era de manera individual. Las dos únicas diferencias entre ambos comandos son la clave grupal que se le pasa por parámetro en vez de la clave individual de la bombilla, y el script de Python.

3.2.3 Aplicación de prueba con Rainbow Hat

Para finalizar el tutorial de *Pimoroni*, se evaluó los requisitos del último apartado. Esta sección, describe un procedimiento basado en el encendido y apagado de las bombillas mediante el sensor de luz ambiental de la interfaz física enviro pHAT. Es decir, la función de la práctica era implantar el código en Python proporcionado en la página e insertarlo y ejecutarlo dentro de un script de la Raspberry Pi. Una vez ejecutado, con el conmutador de la lámpara encendido se podía apagar y encender la luz dependiendo de la cantidad de luz ambiental que entrase al sensor, para ello, se cubría con la mano el sensor cuando se quisiese apagar la bombilla y se retiraba la mano de él cuando se quisiera encender.

El problema que se presentó es que en el Starter kit de la Raspberry Pi no estaba incluido este hardware, sino que contenía el hardware Rainbow Hat.

Rainbow Hat es una interfaz que se conecta a los 40 pines de la Raspberry Pi igual que el hardware enviro pHAT, pero tienen funcionalidades diferentes y una de ellas es, que no contiene el sensor de luz ambiental para la realización de este apartado. A pesar de este problema, se inventó una pequeña aplicación para aprender a controlar el Rainbow Hat el cual estaba ya implantado sobre la Raspberry Pi.

La aplicación consistía en apagar y encender las bombillas, es decir, se buscó realizar la misma función que se expone en el tutorial, pero con la modificación de que en vez de utilizar el sensor para el encendido y apagado de las bombillas se utilizó dos de los tres botones del Rainbow Hat.

La funcionalidad que tomó la aplicación no es más que si se pulsaba el botón A se encendía la bombilla y el led correspondiente al botón y, además, en el display se mostraba el mensaje de texto “LUZ ON”. Si, por el contrario, se pulsaba el botón B, se apagaba la bombilla, se encendía el led del botón indicando que se había pulsado el botón y se mostraba el mensaje “LUZ OFF”.

Para llevar a cabo la aplicación, lo primero que se realizó fue la instalación del software de Rainbow Hat para poder utilizar la interfaz en las aplicaciones que se implementen. El comando que se envió para proceder a la instalación fue el siguiente:

COMANDO PARA LA INSTALACIÓN DEL SOFTWARE RAINBOW HAT
`curl https://get.pimoroni.com/rainbowhat | bash`

Tabla 16: Comando para la instalación del software Rainbow Hat

A continuación, se reinició la Raspberry Pi para que se estableciesen los paquetes instalados. En el directorio del hardware se había creado una carpeta con nombre *"Pimoroni"*. Dentro de ella, se encontraba la carpeta *"Rainbow Hat"*, la cual contenía documentación y un conjunto de scripts para la realización de pequeñas aplicaciones con el fin de conocer cómo se llevan a cabo las distintas funcionalidades de la interfaz física.

La implementación de la aplicación comenzó insertando los paquetes *"os"*, que sirve para poder gestionar archivos dentro de los directorios del sistema operativo y el paquete *"configParser"* cuya función es entre otras, analizar archivos. A partir de estos dos paquetes se desarrolló el código para leer el nano texto "*tradfri.cfg*" y guardar las variables de la dirección IP de la pasarela, el usuario de API y la clave. Aparte de estos dos paquetes, también, se añadió la biblioteca Rainbow Hat para poder realizar las funciones que fuesen precisas con la interfaz y el paquete "*tradfriActions*", importante para el control de las bombillas.

En la aplicación se implementaron cuatro funciones. Dos de ellas, se utilizan para encender y apagar la luz de la bombilla pasándoles por parámetros las variables del archivo "*tradfri.cfg*" y, además, el código de la bombilla que se va a configurar para que la puerta de enlace realice dicha función. En el interior de las funciones, se realizó una llamada a un script externo llamado" *tradfriActions*", que está ubicado dentro de la carpeta de *"ikea-smartlight"* que se creó al instalar el administrador de paquetes de Python, "*pip"*. Este script contiene diferentes funciones que hacen posible la configuración de las distintas acciones para una bombilla o grupo de bombillas mediante llamadas con coap-client dependiendo a la función a la que se llame.

En la aplicación, se invocó concretamente la función encargada del estado de la luz de una bombilla, que gracias a importar el paquete de *"tradfriActions"* se pudo utilizar funciones fuera de nuestro archivo de Python.

FUNCIÓN ENCARGADA DE ENCENDER LA BOMBILLA

`def switch_on(hubip, apiuser, apikey, bulbid):`
`  tradfriActions.tradfri_power_light(hubip, apiuser, apikey,`
`  bulbid, 'on')`

FUNCIÓN ENCARGADA DE APAGAR LA BOMBILLA

`def switch_off(hubip, apiuser, apikey, bulbid):`
`  tradfriActions.tradfri_power_light(hubip, apiuser, apikey,`
`  bulbid, 'off')`

Tabla 17: Funciones para cambiar el estado de las bombillas

Las otras dos funciones son dos escuchadores a los botones A y B donde dentro de ambas se llama a las funciones de encender *"switch_o*n" y apagar *"switch_off"* respectivamente. Dentro de ellas, se implementó el código para activar el botón que se pulse y también, el mensaje que se muestra por el display dependiendo de si la bombilla estuviese encendida o apagada. Esta parte del código, se consigue mediante invocaciones a métodos a partir de la biblioteca de la interfaz Rainbow Hat que gracias a ella se puede dar funcionalidad al hardware.

FUNCIÓN DEL BOTÓN A

```
@rainbowhat.touch.A.press()
def touch_a(channel):
        rainbowhat.lights.rgb(1,0,0)
        switch_on(hubip, apiuser, apikey, bulbid)
        rainbowhat.display.print_str("LUZ-")
        rainbowhat.display.show()
        time.sleep(1.0)
        rainbowhat.display.print_str("--ON")
        rainbowhat.display.show()
        time.sleep(1.0)
```

Tabla 18: Función del botón A

La estructura del botón B es idéntica a la del botón A, pero el contenido es distinto. En este caso se llama al método "switch_off" para apagar la bombilla y el mensaje que se muestra en la interfaz si se ha pulsado el botón es "LUZ OFF".

Con esto se finalizaría el proceso de aprendizaje para poder controlar dispositivos CoAP a través de la Raspberry Pi con lenguaje de programación Python.

4. APLICACIÓN 1: MANDO REMOTO

4.1. Introducción y objetivos

Una vez finalizado el proceso de aprendizaje de la página web de Pimoroni donde se ha aprendido a controlar inalámbricamente las bombillas inteligentes con el protocolo de comunicación CoAP y mediante programas en Python, se comenzó a discurrir unas posibles aplicaciones piloto con el fin de facilitar tareas cotidianas de las personas o permitirles realizar acciones que actualmente no estén dentro de su alcance.

Al realizar en el último apartado del tutorial la aplicación de encender y apagar las bombillas mediante la Raspberry Pi como si fuese un conmutador, pero de manera remota, se pensó en un prototipo de mando a distancia mediante el cual sea posible controlar el conjunto de bombillas situadas en distintos puntos del hogar. Además de activar y desactivar las bombillas, con el *Mando Remoto* se podría configurar el resto de acciones posibles, como son el ajuste de intensidad o el cambio de color dependiendo la escena que se requiera en ese momento.

Al final se decidió implementar este prototipo de aplicación por dos motivos, el primero porque es práctico, ya que mediante el hardware de desarrollo se pueden manejar y configurar completamente, de manera sencilla y rápida, todas las bombillas que estén asociadas a la red Trådfri, y lo segundo, porque es una aplicación novedosa por el mero hecho de que gracias a la interfaz física que está insertada en la Raspberry Pi, se puede crear un mando remoto no muy común en el mercado dándole distintas funcionalidades a los componentes de la interfaz creando así, un mando remoto dinámico y característico.

El objetivo principal de la aplicación era implementar un *Mando Remoto* que se controle a través de la Raspberry Pi y que estaría implementado mediante un programa en lenguaje de programación Python.

La aplicación consta de un mensaje inicial para dar la bienvenida al usuario, acto seguido, el programa le pide al mismo elegir que bombilla o que grupo de bombillas desea configurar. Una vez que se ha seleccionado la bombilla/s, la Raspberry Pi muestra en el display el menú de la aplicación, el cual consta de tres acciones posibles a realizar en las bombillas si el usuario lo desea. Estas acciones son, encender o apagar la luz, el control de la intensidad lumínica donde se puede ajustar en un rango de 0 a 100% y, por último, cambiar el color de la bombilla/s generando distintos efectos en el lugar donde estén ubicadas y así ajustarse a las necesidades del usuario. Cuando el usuario ha acabado de configurar la bombilla/s puede volver al menú para seleccionar otra acción que ya había configurado antes o, por el contrario, el programa le da la posibilidad de sustituir la bombilla que está controlando en ese momento, por cualquier otra conectada a la red Trådfri.

Aparte del menú que es la funcionalidad más importante de la aplicación, gracias a la interfaz física, durante el mensaje inicial y el ajuste de las distintas acciones en el Rainbow Hat se iluminan los siete leds de distinto modo según lo que este configurando en ese momento el usuario, proporcionándole más dinamismo al *Mando Remoto*. Aparte, cada botón tendrá un sonido peculiar que se generará mediante el piezoeléctrico que está contenido dentro de la interfaz.

4.2. Análisis e implementación

Una vez que se ha explicado de manera general la funcionalidad del *Mando Remoto*, en este sub-apartado, se va a exponer paso a paso mediante tablas, explicaciones e ilustraciones cómo se desarrolla la aplicación de inicio a fin.

Lo primero de todo, para comenzar el código y que no se generé ningún problema al intentar programar alguna función específica de la cual sea necesaria algún paquete en concreto, se insertaron las bibliotecas principales para la implementación del código. Esto se realizó igual que como se comentó anteriormente en la aplicación de prueba y se llamaron a los mismos paquetes mediante el comando import.

Posteriormente, se implementó el código para proceder a la lectura del archivo de texto "*tradfri.cfg"* mediante una invocación a la clase "*ConfigParser*[27]*"* que no es más que un analizador de archivos donde en este programa se utiliza para leer el fichero "*tradfri.cfg"* y se obtuvo mediante una invocación al método "*get()"* las variables *"(hubip, apiuser, apikey)"* necesarias para que la puerta de enlace de permiso para controlar las bombillas.

La aplicación da comienzo visualmente con un mensaje inicial "*HOLA*", para dar la bienvenida al usuario en el que cada letra se muestra en cada uno de los cuatro displays de la Raspberry Pi. En paralelo, a través de los siete leds que están en la parte superior de los displays se muestra una secuencia intermitente de colores desde el exterior hasta el interior creando un mando luminoso y dinámico.

El hardware de desarrollo solo posee cuatro displays en donde solo es posible mostrar una letra, número o símbolo en cada uno para ofrecer un mensaje al usuario. Para optimizar dicha interfaz, se investigó el modo de mostrar un texto superior a cuatro letras como si fuese un texto barrido. Para ello, se probó distintas formas de llevarlo a cabo y resultó que con un bucle "*for()"* se conseguía recorrer el mensaje letra a letra hasta completarse el texto. Al desplazamiento de las letras se le añadió un tiempo de 0.2 segundos para que el texto no se visualizase demasiado ralentizado y fuese más dinámico.

Una vez que se ha conseguido esta funcionalidad, lo siguiente en mostrar por la Raspberry Pi después del mensaje inicial es un mensaje extenso donde se le indica al usuario que

[27] Fuente vía - https://docs.python.org/2/library/configparser.html

bombilla o bombilla/s desea configurar. El mensaje es el siguiente: "*PULSE A PARA CONFIGURAR LA BOMBILLA 1* B CONFIGURA BOMBILLA 2 * C CONFIGURA AMBAS"*. A continuación de este texto, aparece un mensaje fijo "*ABoC*" sobre los cuatro displays donde indica que opción quiere escoger el usuario. El mensaje se muestra a propósito de este modo para mostrar las letras A, B y C sobre los botones A, B y C respectivamente y así, facilitar al usuario la elección.

Dependiendo de la opción escogida, el usuario podrá controlar una bombilla individualmente si pulsa el botón A o B, o, por el contrario, podrá configurar ambas a la vez si ha pulsado la C. La implementación en Python de esta parte de la aplicación se llevó a cabo a través de una invocación a un método escuchador que está contenido dentro de la biblioteca de la interfaz física, Rainbow Hat.

A partir del valor de la variable "*channel"* se conoce que botón se ha pulsado. Si "*channel"* es igual a 0, se ha pulsado el botón A por lo que solo se tuvo que pasar por parámetro de la función que se realizará posteriormente el identificador de la bombilla que se ha escogido, en este caso, el identificador de la primera bombilla es 65537. En el caso de que "*channel"* sea igual a 1, se habrá pulsado el botón B y se implementa el mismo código, pero con el identificador de la otra bombilla, es decir, 65538. Y, por último, si "*channel"* es 2, el botón elegido es la C y el identificador es el del grupo de bombillas. En el array *bombillas[]*, es donde está contenido los tres valores del identificador.

Después de seleccionar la bombilla que se va a configurar, se muestra por el display el siguiente texto: *"MENU"*. Este mensaje indica al usuario que está dentro del menú del *Mando Remoto*.

En la siguiente tabla se visualiza el código que se ha explicado teóricamente:

FUNCIÓN PARA OBTENER QUE BOTÓN SE HA PULSADO

```
@rainbowhat.touch.press()
        def touch(channel):

             luz_boton_on(channel)
             luz_boton_off()

             if channel==0:

                bulbid=bombillas[0]
                display_mensaje("MENU",0)
                menu_luces(bulbid)

             elif channel==1:
```

```
                    bulbid=bombillas[1]
                    display_mensaje("MENU",0)
                    menu_luces(bulbid)

                else:

                    bulbid=bombillas[2]
                    display_mensaje("MENU",0)
                    menu_luces(bulbid)
```

Tabla 19: Función para conocer que botón se ha pulsado

El menú está constituido por tres opciones:

1. **Encendido y apagado.**

   Esta acción se basa básicamente en encender o apagar la luminosidad en las bombillas dependiendo de lo que el usuario estime oportuno. Si desea iluminar una estancia del hogar encenderá la bombilla y en cambio, la apagará para no consumir electricidad si no va a estar dentro de ella.

   Siguiendo el proceso de la aplicación, si se presiona el botón A el usuario entrará dentro del menú en el cual podrá configurar tres acciones en las bombillas. Estas acciones son encender o apagar la bombilla/s, ajustar la intensidad y cambiar el color.

   Después de presionar el botón, se visualiza en el display de la Raspberry Pi el mensaje, "1.LUZ", indicando al usuario la opción del menú en la que se encuentra. Esta acción activa o desactiva la bombilla dependiendo el estado en el que esté. Para activar la bombilla se pulsa el botón B y para desactivarla se presiona el botón C, así se proporcionó distintas funcionalidades para los tres botones disponibles.

   La implementación del código de esta acción no fue compleja porque anteriormente se ha desarrollado para la aplicación de prueba un programa similar donde se lleva a cabo está configuración. Lo único complejo de esta parte del código es entender cómo se realizan las llamadas a las funciones (Tabla 20) para el encendido y apagado de las bombillas sabiendo las variables que se le tienen que pasar por parámetro y también, comprender el script *"tradfriActions"* para conocer con que método se consigue está acción.

   El siguiente fragmento de código incluye el desarrollo en Python de las funcionalidades de los botones B y C:

FUNCIONES DE ENCENDER Y APAGAR LAS BOMBILLAS

```
@rainbowhat.touch.B.press()
def press_b(channel):

    global estado
    luz_boton_on(channel)
    luz_boton_off()

    switch_on(hubip, apiuser, apikey, bulbid)
    display_mensaje("ON  ",0)
    estado=1
    leds_luces(estado)

@rainbowhat.touch.C.press()
def press_c(channel):

     global estado
     luz_boton_on(channel)
     luz_boton_off()

     switch_off(hubip, apiuser, apikey, bulbid)
     display_mensaje("OFF ",0)
     estado=0
     leds_luces(estado)
```

Tabla 20: Funciones para activar y desactivar las bombillas mediante los botones B y C respectivamente.

Como se puede observar, el código está formado por dos escuchadores *"@rainbowhat.touch.B.press()"* y *"@rainbowhat.touch.C.press()"* que sirven para detectar cuando se ha pulsado el botón B o el botón C respectivamente.

En el caso de que se pulse el botón B, como en su método se realiza una invocación a la función *"switch_on"* (Tabla 20), se producirá el encendido de la bombilla. En esta invocación, se le pasa por parámetro las claves de la puerta de enlace y el identificador de la bombilla con el fin de comunicar a la pasarela la acción que debe ejecutar. Aparte de activarse la bombilla, para sacar la máxima rentabilidad a la interfaz, después de ejecutar la acción, en el display se muestra el mensaje de texto "*ON*" y se atribuye una secuencia característica a los siete leds. La funcionalidad que se les otorga es realizar una secuencia creciente de encendido desde el led de la posición cero hasta el seis, mostrando una acción similar a la que se ha llevado a cabo en la bombilla. Es decir, la bombilla ha pasado de estar desactivada a estar activada, por consiguiente, los leds modifican su estado de desactivado ha activado progresivamente con color azul hasta el punto de que todos los leds luzcan por igual.

Y, por el contrario, si se pulsa el botón C como en el interior de su función se realiza una invocación, en este caso, a la función *"switch_off"* (Tabla 20) entonces se desactivará la luminosidad de la bombilla o bombillas dependiendo de la elección que haya escogido el usuario anteriormente.

Como al encender la bombilla la interfaz física tiene una determinada funcionalidad, para este caso contrario también la tendrá. Lo que se ha programado para este suceso, es la visualización del mensaje "*OFF*" y la desactivación de todos los leds uno a uno de modo decreciente, es decir, desde el led seis hasta el led cero. Esta función de la interfaz realiza una acción semejante a la bombilla, puesto que transcurre de estar encendida a estar desactivada igual que los leds del hardware Rainbow Hat.

2. **Intensidad lumínica:**

La segunda opción del menú es el ajuste de la intensidad lumínica en la bombilla/s. Con esta configuración se consigue manejar la intensidad en un rango de 0 - 100%. Esta es una de las nuevas funciones que las bombillas Trådfri pueden realizar en comparación a las bombillas convencionales. En estas últimas, por lo general, solo es posible el encendido y apagado, sin embargo, con las bombillas Trådfri al ser de tipo led se pueden realizar otras acciones como es, en este caso, el ajuste de intensidad.

El usuario puede reducir la intensidad para generar menos luminosidad en la estancia y ahorrar energía, o, si, por el contrario, necesita una mayor iluminación solo tendría que aumentar la intensidad de la bombilla hasta el grado en que se ajuste a sus necesidades.

Una vez explicado en que consiste esta acción, se continúa desarrollando el programa del *Mando Remoto*. Estando en el apartado de "*1.LUZ"*, si se presiona el botón A de nuevo, entonces, el programa pasaría a este ajuste el cual se mostraría en el display del Rainbow Hat con el siguiente mensaje, "*2.INT*", indicando que es la sección para el ajuste de la intensidad lumínica.

Dentro de este apartado, se puede pulsar el botón B o C para controlar la intensidad. Este ajuste solo está comprendido como se ha comentado anteriormente en un rango de 0-100%, que mediante los botones B y C se aumentará o disminuirá dicho valor en un rango de 10%.

Esto se llevó a cabo para que el usuario si estaba en el valor 100% no tuviera que pulsar el botón B cien veces hasta bajar al 0%, sino que fuese más interactivo y eficiente.

El programa por defecto comienza con un valor intermedio de 50%. A partir de ese valor, el usuario interacciona con la Raspberry Pi para reducir o aumentar la intensidad. Si se quiere reducir esta propiedad a 0% por ejemplo, entonces deberá presionar el botón B cinco veces, ya que el valor disminuye en un rango de 10%.

Hay que tener en cuenta que cuando la intensidad de la bombilla es cero, entonces, la bombilla se apaga ya que se le estaría aplicando una energía nula sobre ella. Además, si estando en el valor cero el usuario vuelve a pulsar la B, en la Raspberry Pi se muestra un mensaje de error en forma de *"scroll"* indicando que el valor cero en la intensidad es la mínima cantidad que la variable puede obtener.Por el contrario, si el usuario presiona el botón C aumentará la intensidad. En este caso cuando se llega al valor cien no se apaga sino al revés, la bombilla estaría en su máximo rendimiento. Si el usuario presionase de nuevo el botón C teniendo una intensidad del 100%, se mostraría un mensaje de error indicando que el valor es el máximo que se puede establecer.

Para esta segunda acción también se le otorga funcionalidad a la interfaz física Rainbow Hat, pero diferente a la primera.

Lo primero que se hizo fue investigar que se podía aplicar a los leds para que tuviesen una función similar al proceso de ajuste de la intensidad. Como la función de este proceso no es más que manejar la intensidad disminuyéndola o aumentándola en un rango fijo de 0 a 100%, entonces, se pensó que la mejor manera de representar está acción en la interfaz física era aplicar en los leds un limitador.

A cada uno de los siete leds se le aplica un color distinto dependiendo del rango donde se encuentren. En el caso de que la intensidad sea igual a 0%, la bombilla se apagará y todos los leds estarán desactivados, si la intensidad es igual a 10% se activará el led seis al cual se le asigna el color azul verdoso claro. Cuando el nivel de intensidad es del 20% o 30% entonces aparte de estar activado el led seis, el led cinco también lo estará con un color verde claro. Para un porcentaje del 40%, al led cuatro se le aplica el color verde, como es un limitador, los leds anteriores también estarán activados. Si el nivel de la intensidad es del 50%, nivel que se aplica por defecto a la bombilla nada más encenderla, o el 60%, el led número tres, toma un valor verde amarillento. Hasta este porcentaje la bombilla trabaja a un rendimiento intermedio o bajo dependiendo del nivel de intensidad que se le aplique. Sin embargo, la energía que se le aplica a la bombilla para trabajar con un rendimiento alto es mayor, con lo que dicho nivel tiene que ser representado en el limitador para que el usuario sea consciente de ello. Es decir, si la bombilla tiene un nivel de intensidad del 70%, al led dos se le asigna el color naranja. Si el usuario presiona una o dos veces el botón C hasta llegar a un porcentaje del 80% o 90% entonces al led uno se le asigna un color con tono rojizo indicando que la

bombilla está llegando a su nivel máximo de intensidad. Y, por último, si el nivel de intensidad es del 100% entonces la bombilla estará trabajando en su máximo rendimiento y el color para el led cero será rojo. Aparte de activar el led cero, también deberán estar activados todos los l restantes con el color mencionado anteriormente.

A continuación, se muestra un fragmento del código necesario para realizar esta acción:

FUNCIÓN PARA AJUSTAR LA INTENSIDAD

```
def intensity (hubip, apiuser, apikey, bulbid, value):

     if bulbid!=131073:

            tradfriActions.tradfri_dim_light(hubip, apiuser,
            apikey, bulbid, value)

     else:

            tradfriActions.tradfri_dim_group(hubip, apiuser,
            apikey, bulbid, value)
```

Tabla 21: Función para ajustar la intensidad dependiendo la bombilla escogida

Esta función que se ha implementado (Tabla 24) es la más importante para que se realice el ajuste de intensidad en la bombilla. Dentro del método donde se ha desarrollado el código para realizar esta acción, se implementó dos escuchadores a los botones B y C para disminuir o aumentar la intensidad. Dentro de ambos botones, se produce una llamada a la función *"intensity()"*, la función (Tabla 24), la cual dependiendo de la bombilla que haya elegido el usuario controlar ya sea una bombilla especifica o un grupo de bombillas se realiza una invocación a un método u otro.

En el caso de que el usuario haya elegido manipular una única bombilla entonces mientras se está ejecutando el programa se introducirá en la llamada al método *"tradfriActions.tradfri_dim_light()"*, la cual está dentro de otro script diferente al proyecto del *Mando Remoto* pero gracias al paquete insertado arriba llamado *"import tradfriActions"* se puede invocar funciones de ese script dentro del código. Este método necesita como parámetros las variables de (Tabla 16) necesarias para el acceso a la puerta de enlace y otras dos variables más como el identificador de la bombilla y el valor de la intensidad que se ha seleccionado.

Este método, *"tradfriActions.tradfri_dim_light()"*, lo que hace internamente es programar en lenguaje Python las solicitudes que se envían a la puerta de enlace mediante el protocolo de comunicación CoAP (Tabla 10).

A continuación, se puede observar en la siguiente tabla una parte del método para visualizar como se realiza:

SOLICITUD MEDIANTE EL PROTOCOLO COAP EN LENGUAJE PYTHON

`tradfriHub='coaps://{}:5684/15001/{}'.format(hubip, lightbulbid)`
 	
`payload = '{ "3311" : [{ "5851" : %s }] }' % int(dim)`
 	
`api = '{} -m put -u "{}" -k "{}" -e \'{}\' "{}"'.format(coap, apiuser, apikey, payload, tradfriHub)`

Tabla 22: Solicitud mediante el protocolo CoAP en lenguaje Python

La solicitud se divide en la carga útil donde se le pasa el valor de la intensidad al código 5851, que es el que actúa como regulador de la intensidad dependiendo del valor que obtiene y el punto final, *"tradfriHub"*, que es donde se va a realizar el ajuste de intensidad, para ello, necesita la dirección IP que será la de la puerta de enlace y la bombilla donde ejecutará la acción. Por último, se crea la variable *"api"*, que es la que realiza la comunicación entre la Raspberry Pi y la puerta de enlace y en este caso, el servicio que se está pidiendo es la regulación de la intensidad.

Si, por el contrario, el usuario ha seleccionado el grupo de bombillas, entonces dentro de la condición (Tabla 24), se introduce dentro del else ya que el identificador es igual al 131073. Ahí se invoca a la función *"tradfriActions.tradfri_dim_group()"* que es distinta a la que se invoca para una bombilla individual porque está solo funciona cuando se le pasa el identificador del grupo de bombillas. Internamente, la implementación del código es similar a la de *"tradfriActions.tradfri_dim_light()"*, ya que la función que realiza es idéntica a mostrada anteriormente (Tabla 25).

3. **Color:**

La tercera opción del menú es la elección del color de la bombilla. El usuario puede escoger entre tres tipos de tonalidades diferentes, como son: cálido, frio y normal.

Estos colores generan cada uno un ambiente o escena distinta en el lugar donde está ubicada la bombilla/s. El tono cálido está pensado para crear un ambiente acogedor. Respecto al color frío, tiene un tono azulado y está pensado para aplicarse a zonas donde hay escasa iluminación, y, por último, el tono normal o neutro, es parecido a la luz natural, con lo que se puede utilizar para iluminar cualquier estancia.

Este ajuste es posible gracias a que este tipo de bombillas son de tipo LED. Es decir, aparte de realizar las mismas funciones que una bombilla convencional se le pueden aplicar otras funcionalidades como el ajuste de intensidad o el cambio de color para permitir al usuario moldear su vivienda como el desee sin tener tanta restricción como las bombillas genéricas.

Una vez que se ha definido en que consiste esta última parte del menú, se continúa desarrollando el funcionamiento del *Mando Remoto* desde la última parte que se ha explicado.

Anteriormente, se explicó que una vez que se ha elegido la bombilla que el usuario desea configurar, el programa se introduce en el menú. Este menú se basa en tres acciones, encendido o apagado, ajuste de intensidad y el color. Por defecto, el menú comienza en el encendido o apagado de la bombilla/s, pero se puede cambiar la acción a realizar sobre ella pulsando el botón A hasta llegar a la que desee. Si el usuario pulsa solo una vez el botón A, entonces la acción a realizar sobre la bombilla será el ajuste de intensidad, pero para este caso que se va a explicar a continuación, el usuario deberá presionar el botón A una vez más para llegar a la última opción del menú, el color.

Lo primero de todo, cuando el usuario llega a la opción del ajuste del color, lo que se muestra en la Raspberry Pi es el mensaje de texto *"3.COL"* indicando que el programa se encuentra parado en esta sección para su posible configuración. A partir de aquí, mediante el botón B la persona que maneje la Raspberry Pi podrá cambiar el color entre tres opciones, cálido, normal o frio.

En el hardware dependiendo del color seleccionado para que el usuario sepa el tono que tiene la bombilla en ese instante, en la interfaz de la Raspberry Pi se muestra en formato texto la opción que se ha seleccionado. Es decir, en el caso en que el tono escogido sea el cálido, se muestra el texto "*WARM*" en los 4 displays disponibles del Rainbow Hat, cuyo significado es cálido en inglés. Si el tono elegido es normal, en la interfaz física se muestra "*NORM*" ocupando los 4 displays e indicando abreviadamente la tonalidad escogida. Y, por último, si por el contrario no se elige ninguna de las otras dos opciones y se prefiere un tono frío, en el Rainbow Hat se muestra el texto "*COLD*" que es su traducción en inglés.

Dentro de esta acción, hay un inconveniente a tener en cuenta. El problema es el

siguiente, si la bombilla está apagada ya sea bien porque se ha apagado mediante el botón C en la primera opción del menú o porque se ha configuración un nivel de intensidad del 0%, entonces el color no se podrá configurar.
Si la bombilla está apagada y el usuario dentro de la opción del color pulsa el botón B para cambiar el color, el programa está desarrollado para que muestre un mensaje de error y salga aparezca en formato *"scroll"* el siguiente mensaje: *"ERROR. PRIMERO DEBE ENCENDER LA LUZ".*

Para este último caso del menú, los leds del Rainbow Hat también tendrán una funcionalidad particular. Como se ha hecho con las otras dos acciones, en este caso, también se buscaba aplicar una particularidad a los leds para que el usuario visualizase el mismo proceso que se va a realizar en la bombilla, pero en la interfaz física. Para ello, se ideó activar todos los leds del Rainbow Hat cuando se pulsase el botón B mostrándose los siete leds del mismo color que la bombilla. Es decir, si el color elegido es cálido, aparte de mostrarse el texto en el display e iluminar la bombilla con dicho color, para que el usuario se haga una idea del color por si no la tiene a la vista la bombilla, mediante los siete leds lo podrá visualizar, ya que se ha programado para que los todos leds se iluminen con el mismo color que la bombilla, en este caso, que es el tono cálido, los leds toman un color anaranjado. Si el tono elegido es neutro, el color de los leds será un tono blanco amarillento. Y, por último, si se escoge el tono frio, los leds se activarán con un color blanco azulado, igual que la bombilla.

Una vez expuesto cómo se desarrolla el programa en la Raspberry Pi, se va a explicar cómo se ha implementado el código para llevar a cabo el programa. Para esta acción solo es necesario implementar el escuchador del botón B ya que es el único botón que se utiliza. Dentro de este método, se realizó un condicional, en el que se le pasa una condición indicando que, si el contador del color es igual a cero, el color es cálido. Si, por el contrario, el contador es uno, la bombilla se iluminará con el tono normal y si el contador toma valor dos, el color seleccionado será frio. Dentro de este último, se pondrá el contador a cero para que en el caso de que se vuelva a pulsar el botón B se repita la secuencia empezando de nuevo por el primero.

Dentro del condicional, para iluminar la bombilla se hace una llamada al método *"colour()"* expuesto a continuación:

FUNCIÓN PARA CAMBIAR EL COLOR DE LAS BOMBILLAS

```
def colour(hubip, apiuser, apikey, bulbid, value):

      if bulbid!=131073:

            if value=="warm":

                tradfriActions.tradfri_color_light(hubip,
                apiuser, apikey, bulbid,"warm")

             elif value=="normal":

                tradfriActions.tradfri_color_light(hubip,
                apiuser, apikey, bulbid,"normal")

             elif value=="cold":

                tradfriActions.tradfri_color_light(hubip,
                apiuser, apikey, bulbid,"cold")
        else:

             for x in range(2):

                bulbid=bombillas[x]

                if value=="warm":

                    tradfriActions.tradfri_color_light(hubip,
                    apiuser, apikey, bulbid, "warm")

                elif value=="normal":

                    tradfriActions.tradfri_color_light(hubip,
                    apiuser, apikey, bulbid, "normal")

                elif value=="cold":

                    tradfriActions.tradfri_color_light(hubip,
                    apiuser, apikey, bulbid, "cold")
```

Tabla 23: Función colour - cambia el color de la bombilla escogida.

Este método actúa igual que los explicados anteriormente para el ajuste de intensidad y para el encendido o apagado de las bombillas, pero en este caso, se implementa para la modificación del color.

La función *"colour()"* recibe como parámetros las variables (Tabla 16), el identificador de la bombilla que se ha seleccionado y el valor del color elegido. Dentro del método hay una primera condición en el que dependiendo de si se ha elegido una bombilla o el grupo de bombillas el programa lleva a cabo una función u otra.

En el caso de que se haya elegido una sola bombilla, se invoca a un método del *"tradfriActions"* para iluminar esa bombilla individualmente. Este método es *"tradfriActions.tradfri_color_light()"* en el que se le pasa los mismo parámetros que a la función *"colour()"* pero en el valor de "*value*" se le asigna directamente el string del color seleccionado.

El script *"tradfriActions"*, como su propio nombre indica es donde se desarrolla el código para realizar las posibles acciones de las luces Trådfri. Una de las acciones es la del color y se implementa de manera similar que para el ajuste de intensidad. Es decir, mediante el lenguaje Python se escribe la solicitud para la comunicación con la pasarela, que es posible, gracias al protocolo CoAP. Como se puede observar (Tabla 27) se utiliza las tres mismas variables, *"tradfriHub"*, *"payload"* y *"api"*, que en el ajuste de intensidad. El valor de las variables *"tradfriHub"* y *"api"* son idénticas, la única que cambia es la carga útil que tendrá un valor distinto por cada color.

FUNCIÓN tradfriActions.tradfri_color_light()

```
tradfriHub='coaps://{}:5684/15001/{}'.format(hubip,lightbulbid)

    if value == 'warm':
        payload = '{ "3311" : [{ "5709" : %s, "5710": %s }] }'
                  %("33135","27211")
    elif value == 'normal':
        payload = '{ "3311" : [{ "5709" : %s, "5710": %s }] }'
                  %("30140","26909")
    elif value == 'cold':
        payload = '{ "3311" : [{ "5709" : %s, "5710": %s }] }'
                  %("24930", "24684")

api = '{} -m put -u "{}" -k "{}" -e \'{}\' "{}"'.format(coap,
      apiuser, apikey,payload,tradfriHub)
```

Tabla 24: Función tradfriActions.tradfri_color_light()

Sin embargo, para el caso de que el usuario haya seleccionado el grupo de bombillas, al no tener una función propia para ello en el script *"tradfriActions"*, se investigó la manera de cambiar el color de ambas bombillas a la vez.

Resultó que la mejor opción era realizar un bucle *"for()"* cambiando el color primero de una bombilla individualmente y acto seguido, de la otra bombilla. Se llevó a cabo una prueba para ver si se notaba algún retardo en el cambio de color en ambas bombillas y se observó, que el cambio era inmediato en ambas y no se visualizó ningún retardo con lo que se consiguió satisfactoriamente la acción.

Una vez explicadas las tres acciones que se pueden realizar en las bombillas desde el menú, el *Mando Remoto* no finaliza aquí.

Dentro del ajuste del color, el botón C hasta el momento no se utilizaba para llevar a cabo ninguna acción, sin embargo, se ha implementado un código para que cuando se pulse el botón aparezca un mensaje de texto en forma de *"scroll"* indicando lo siguiente: *"FIN DEL MENU".* A continuación del mensaje, el programa vuelve a la opción de elegir bombilla por si el usuario desea configurar otra bombilla diferente a la actual o si, por el contrario, se ha equivocado al escoger de bombilla. Esta acción le da la opción de corregir el error sin tener que ejecutar de nuevo el script del *Mando Remoto*.

Cuando finaliza el mensaje de *"FIN DEL MENU",* acto seguido, aparece el mensaje *"ELIGE BOMBILLA".* En este caso, el mensaje extenso donde se explicaba para que servía cada botón ya no aparece porque el usuario debería ya conocer la funcionalidad del programa y así, de esta forma, la elección de la nueva bombilla es más dinámico puesto que no hay que esperar a que imprima por el display del Rainbow Hat todo el mensaje. Después de seleccionar la bombilla, el programa vuelve a entrar al menú para su configuración.

El *Mando Remoto* nunca finaliza, tiene la misma función que el mando remoto que venía en el kit Trådfri y solo se apaga cuando se desconecta la Raspberry Pi. Esto se hace con el fin de que el usuario pueda manipular las bombillas en el momento que desee sin tener que empezar de cero.

5. APLICACIÓN 2: TEMPORIZADOR

5.1. Introducción y objetivos

En este apartado, se va a exponer la segunda aplicación que se creó, el *Temporizador*. Se analizará porque se ha elegido desarrollar esta aplicación y cuáles son sus ventajas. Además, se explica cómo se ha desarrollado e implementado el código para su posterior uso.

Después de crear el *Mando Remoto*, una aplicación que se llevó a cabo para facilitar al usuario controlar las bombillas y ajustar todas las configuraciones disponibles que desee, se pensó en idear una aplicación práctica con el fin de satisfacer las necesidades de las personas.

En este artículo[28] se expone como las personas están expuestas a sufrir un robo en su vivienda durante el periodo vacacional o en fin de semana. En esta franja, es cuando el índice de robo aumenta considerablemente debido a que el propietario no se encuentra en el domicilio. En algunos casos, como bien se explica en el artículo, las personas influyen en este hecho puesto que facilitan información relevante de donde se encuentran.

Para intentar solventar este problema, surgió el *Temporizador*, una aplicación que se basa fundamentalmente en proporcionar seguridad en la vivienda mientras el usuario está ausente. Esta aplicación no protege la vivienda como tal, sino que simula que hay alguien en el interior de la vivienda consiguiendo disminuir el índice de delitos de allanamiento de morada.

Esto se consigue gracias a que la aplicación está diseñada para que el propietario antes de salir del domicilio, durante un periodo corto o largo de tiempo, active este software estableciendo un periodo de días y una determinada hora a la que la luz de las bombillas Trådfri se enciendan, simulando que hay alguien en el hogar y así, evitar posibles hurtos. Además, se analizó que como no existe actualmente en el mercado ningún software dedicado a este aspecto para la Raspberry Pi y las luces Trådfri, podría ser un punto de inflexión de cara a patentar el software e introducirlo al mercado.

Para obtener el máximo rendimiento de la aplicación y, con ello cubrir las necesidades de las personas, el software contiene tres tipos diferentes de temporizador:

- **Temporizador diario:** Consiste en establecer el temporizador para encender y apagar las bombillas en un día y a una hora determinada por el usuario.

[28] Fuente vía - http://www.lasexta.com/noticias/sociedad/los-expertos-aconsejan-no-anunciar-en-redes-sociales-el-lugar-y-la-fecha-de-las-vacaciones-para-evitar-los-robos-a-domicilio_2016070357790c2d6584a859c7718199.html

- **Temporizador semanal:** Durante los siete días de la semana, entendiendo el comienzo de esta en lunes, se producirá una repetición del alumbrado de las bombillas en el periodo fijado.

- **Temporizador mensual:** Repetición de la hora programada para el encendido de las bombillas durante el primer día de mes hasta concluir el mismo.

5.2. Análisis e implementación

Una vez explicado el funcionamiento y el objetivo del *Temporizador*, en esta sección, se describirá paso a paso el desarrollo del programa y el modo en el que se ha implementado.

Para comenzar, en el script del programa igual que se hizo en la aplicación del *Mando Remoto*, se insertaron los paquetes y las librerías necesarias para la correcta ejecución del programa. Estos paquetes son los mismos que se han comentado en las dos aplicaciones anteriores, salvo la librería "*datetime*" que se insertó con el comando *"import"* y se utilizó para obtener la fecha y hora del sistema y así, poder llevar a cabo el *Temporizador*. A continuación, para poder obtener las variables del archivo de texto *"tradfri.cfg"*, se realizó la misma implementación que en la aplicación del *Mando Remoto*, es decir, mediante la invocación a la clase *"ConfigParser"*.

Después de que se han introducido los paquetes y se han obtenido las variables (*"hubip"*, *"apiuser"* y *"apikey"*) necesarias para la puerta de enlace, se puede implementar el resto del programa.

El programa comienza con el mensaje de bienvenida *"HOLA"*, dedicado a la persona que está ejecutando la aplicación y en paralelo, se ejecuta en los siete leds del Rainbow Hat un barrido de luces los cuales se activan progresivamente desde el led seis hasta el cero cada uno de un color y al llegar al led cero se desactivan uno a uno desde el led seis. Acto seguido, se realiza el mismo procedimiento, pero a la inversa empezando en el cero y acabando en el led seis y una vez que se han desactivado los siete leds, parpadean tres veces todos los leds a la vez con los colores correspondientes de cada uno de ellos indicando que se ha activado la aplicación del *Temporizador*.

Las funciones más importantes para el control de los leds de la interfaz física son:

FUNCIONES PARA EL CONTROL DE LOS LEDS EN EL RAINBOW HAT

`rh.rainbow.set_all(R,G,B)`
`rh.rainbow.set_pixel(x, R,G,B)`
`rh.rainbow.show()`

Tabla 25: Funciones para el control de los leds en el Rainbow Hat

Estas funciones sirven para activar los leds y mostrarlos en la interfaz. En el caso de la primera sentencia, *"rh.rainbow.set_all (R, G, B)"*, es una función que se basa en encender todos los leds a la vez con el color que se le pase por los parámetros de entrada. Los colores de los leds se determinan por la composición de los colores primarios (RGB) que reciben por parámetro. Si por el contrario, en vez de encender todos los leds del mismo color, se quiere encender algún led en particular y asignarle algún color especifico, entonces, se utiliza la función *"rh.rainbow.set_pixel(x,R,G,B)"*. Como el propio nombre indica, en la primera sentencia se realiza un *"set_all"* refiriéndose a establecer la acción en todos los leds, sin embargo, en esta nueva sentencia, *"set_pixel"*, se tiene que introducir en el primer parámetro el pixel que se quiere controlar, y en los otros tres restantes, el valor de las componentes (RGB) para pintar el led en ese determinado color. Por último, la sentencia *"rh.rainbow.show()"* sirve para poder mostrar en el Rainbow Hat la acción de cualquiera de las otras dos funciones, sin este método, la implementación del código estaría correcto, pero no se visualizaría el resultado en la interfaz, que es el objetivo.

Como se puede observar, todas las funciones que se implementan para realizar cualquier acción sobre los leds del Rainbow Hat tiene que llamarse a partir de la biblioteca *"Rainbow"*.

Después de esta breve explicación sobre cómo se configura los leds, el siguiente paso del programa una vez que se ha mostrado el mensaje inicial con el juego de luces en los leds, es preguntar al usuario que bombilla desea escoger para realizar el temporizador sobre ella. Este ajuste se produce debido a que normalmente las bombillas estarán ubicadas en diferentes sitios con lo que se pensó que el usuario debía de escoger que bombilla/s quería encender exactamente.

El mensaje que se muestra en el display de la interfaz, es un mensaje tipo barrido: *"ELIGE LA BOMBILLA* A->BOMBILLA 1* B->BOMBILLA 2* C->AMBAS BOMBILLAS"*. Al finalizar, se muestra el mensaje *"ABoC"*, el cual estará fijo hasta que el usuario no seleccione que bombilla/s quiere utilizar en el *Temporizador*. La elección de la bombilla o las bombillas se realiza de la misma manera que se explicó en la aplicación del *Mando Remoto*, es decir, si se pulsa el botón A se seleccionará una de las dos bombillas, si se presiona el botón B entonces se escogería la otra bombilla, y si, por el contrario, no se selecciona ninguna de estas dos opciones y se pulsa el botón C, se elegirá controlar ambas bombillas a la vez.

La implementación del código de esta parte del programa se realizó igual que para el *Mando Remoto* (Tabla 22), salvo la línea de código en el que se imprimía por pantalla el mensaje de texto *"MENU"*.

Respecto a los siete leds del Rainbow Hat continúan fijos cada uno con un determinado color, pero cambiaran su estado dependiendo del botón que pulse el usuario. Si la opción elegida es la A, es decir, se selecciona una única bombilla, entonces para personalizar la interfaz con esta configuración se muestra un parpadeo del bloque de leds desde el número tres hasta el seis mientras el resto de leds permanecen desactivados. En el caso de que se marque la opción B, se realiza el mismo proceso, pero, al contrario, el bloque que se muestra es el de la parte de la derecha a partir del led número tres. Por último, si el usuario elige la opción C, al seleccionar ambas bombillas parpadean todos los leds en conjunto en vez de mostrar un único bloque indicando que se ha seleccionado una de las dos bombillas.

Una vez que ya se ha elegido la bombilla/s, el programa continúa mostrando un mensaje en formato *"scroll"* para indicar al usuario que tipo de temporizador desea escoger. En la aplicación, hay tres tipos de temporizadores, diario, semanal y mensual, y se seleccionará uno de estos tres mediante los botones A, B, C respectivamente.

- **Temporizador Diario**

  Este tipo de temporizador está diseñado para establecer en el día especificado, el periodo de tiempo en que el usuario desea activar las bombillas. La bombilla solo se enciende en ese rango de tiempo, después, el programa se apaga y con ello las bombillas.

  Lo importante en esta aplicación es indicar correctamente el día, semana o mes dependiendo del tipo de temporizador que se haya escogido y el periodo de tiempo en que se desea tener las bombillas activadas. El usuario tiene libertad de asignar la hora y el día que estime oportuno y puede estar o no en la vivienda, aunque la aplicación está diseñada para que se establezca cuando el usuario no se encuentra en el interior con el fin de proporcionar seguridad en el hogar.

  Una vez resumido el objetivo principal de la aplicación, se va a explicar cómo se ha desarrollado e implementado el temporizador diario.

  Después de seleccionar este temporizador mediante el botón A de la Raspberry Pi, a través de la interfaz Rainbow Hat, se muestra una secuencia de luces con los siete leds.

  Durante todo el transcurso de la aplicación, los leds se iluminan cada uno con un color establecido al principio del programa. Estos son el azul, verde, rojo, naranja, rosa, azul cielo y amarillo. Para este caso, como el temporizador es diario se va a representar en la interfaz una secuencia en la cual se activa y se desactiva cada led uno a uno desde el número seis hasta el cero y después, parpadearán dos veces todos los leds a la vez. Con esta representación se intenta mostrar al usuario que

solo está disponible el temporizador para un día especifico por eso después de activar el led se desactiva antes de encender el siguiente. Además, con el procedimiento de activar y desactivar todos los leds se intenta mostrar al usuario que tiene la opción de escoger el día que desee de lunes a domingo, es decir, cualquiera de los siete días de la semana como se representa con los siete leds del Rainbow Hat. En el display se muestra el mensaje *"1.DIA"* indicando el tipo de temporizador escogido.

Después de mostrar a través de la interfaz la secuencia de leds, para establecer el *Temporizador* se tienen que seguir siete etapas donde en cada etapa que se avance en el programa se iluminará un led en el Rainbow Hat.

Los pasos para llevar a cabo el temporizador son:

1. ***Elección del día:*** El primer paso para configurar correctamente el temporizador diario es elegir el día que se quiere establecer. En el Rainbow Hat, el led cero está activado indicando que el usuario se encuentra en el primer paso y en el display se visualiza el siguiente mensaje, *"D- 1"*. La *"*D*"* indica la palabra *"*Día*"* y el *"*1*"* indica el valor del día en que se quiere encender la bombilla. Con los botones B y C se disminuye o aumenta respectivamente dicho valor hasta establecer el día que el usuario desee. Si se intenta asignar un valor que no está comprendido entre los días uno y treinta y uno, entonces por la interfaz se muestra un mensaje de error indicando que el día no es correcto. Si se selecciona un día comprendido entre ese rango para establecer el valor asignado solo hace falta pulsar el botón A y aparte de guardar dicho valor, el programa continuo al siguiente paso.

2. ***Elección del mes:*** Al pulsar el botón A, el programa avanza a la siguiente etapa con lo que se ilumina un led más en la interfaz. En el display se muestra el mensaje, *"ELIGE MES"* y acto seguido, se establece en el display el mensaje *"M- 1"*. La *"*M*"* de este mensaje se refiere a la palabra "Mes" y el número uno al mes en el que se quiere establecer el temporizador. Este valor no puede sobrepasarse de 12, es decir, del mes de diciembre, ni puede ser inferior al 1 correspondiéndose al mes de enero. Si el valor no está entre ese rango se muestra por pantalla un mensaje de error indicando al usuario que el valor que quiere establecer no es posible y dependiendo de si está seleccionando un valor superior a 12 o inferior a 1, el programa mostrará en el mensaje a través de la interfaz que el valor es el máximo o el mínimo respectivamente.

Para esta asignación al igual que para el primer paso se utiliza el botón B y C para disminuir o aumentar el valor y una vez elegido el mes, se guarda el valor pulsando el botón A.

Hay un inconveniente a tener en cuenta en esta fase. Dependiendo el día que se ha seleccionado, hay restricción a la hora de elegir el mes puesto que hay meses que no tienen treinta y un días, o como el caso de febrero, que tiene solo veintiocho días.

En la siguiente tabla, se muestra la implementación para solventar este error:

FRAGMENTO DE CÓDIGO PARA COMPROBAR SI EL DÍA ES VÁLIDO

```
t=datetime.now()
mes_sistema=t.month
year=t.year

if (mes==4 or mes==6 or mes==9 or mes==11) and
dia==31:

      mensaje="EL DIA NO ES VALIDO    "
      scroll_mensaje(mensaje)
      dia_activado=False
      mes_activado=False
      elegir_dia(dia_activado)

 elif mes==2 and dia>28:

       mensaje="EL DIA NO ES VALIDO    "
       scroll_mensaje(mensaje)
       dia_activado=False
       mes_activado=False
       elegir_dia(dia_activado)
```

Tabla 26: Implementación del código para solventar el error al seleccionar el mes incorrecto

El problema fundamental es que, si el día seleccionado es el día treinta y uno, los meses de abril (4), junio (6), septiembre (9) o noviembre (11) no pueden seleccionarse para activar la bombilla puesto que solo tienen treinta días. Con el mes de febrero (2) pasa exactamente igual, salvo que este mes tiene veintiocho días.

Conociendo este inconveniente lo que se implemento fue un condicional (Tabla 29) en el que, si el día era igual a treinta y uno, y se seleccionaba uno de los meses de abril, junio, septiembre o noviembre entonces la interfaz muestra un mensaje de error indicando que el día seleccionado no es válido, y

en vez de pasar a la siguiente fase, vuelve al paso anterior para seleccionar un día correcto. Para el caso del mes de febrero, el procedimiento es el mismo pero la condición que se le impone es que si el día es superior a veintiocho entonces el programa muestra el mensaje de error y vuelve el usuario a seleccionar un día correcto en el caso de que se quiera elegir febrero.

3. ***Elección de la hora inicial:*** Este paso como el propio título indica, sirve para elegir la hora inicial a la que la bombilla se va a activar. Después de tener ya elegido que día en concreto se quiere utilizar el temporizador, a continuación, se elegirá el periodo de tiempo que la bombilla/s estará encendida. Para empezar a asignar este rango de tiempo se empieza seleccionando la hora inicial.

   Si continuamos con el desarrollo del programa, al presionar el botón A se avanza un paso más y con ello, se activa el led dos junto con los otros dos leds que ya estaban encendidos anteriormente.

   En el Rainbow Hat se muestra el mensaje "*SELECCIONE HORA INICIAL*", y al finalizar, se muestra el mensaje fijo "*H- 0*". El primer mensaje sirve como instrucción al usuario para saber que debe de elegir en este apartado y el segundo mensaje se utiliza para interactuar el hardware con el usuario para la selección de la hora.

   La hora inicial se elige pulsando el botón C donde el valor incrementara de uno en uno hasta llegar al número veintitrés, si se pulsa nuevamente se empieza la cuenta en el 0, es decir, en las doce de la noche en adelante.

   Una vez que se ha escogido la hora inicial para guardar el valor y pasar a la siguiente fase, a diferencia de los pasos anteriores, hay que pulsar el botón B. Se implementó de este modo porque la siguiente fase se considera como un sub-apartado de esta última.

4. ***Elección del minuto inicial:*** Para concretar la hora exacta en la que la bombilla se activa, hace falta seleccionar el minuto inicial.

   Al pulsar el botón B para guardar el valor de la hora inicial, se ilumina un nuevo led y se muestra en el display el mensaje *"SELECCIONE EL MINUTO INICIAL"* y a continuación, *"M- 0"*. La letra "M" en este mensaje significa "Minuto", no mes como en la fase anterior. Por defecto, el minuto será el 0, es decir, si no lo configuramos la bombilla se activa a en punto dependiendo la hora seleccionada. Por el contrario, si se quiere modificar este valor, se pulsa el botón C para incrementarlo y aumentara de cinco en cinco hasta el

minuto cincuenta y cinco. Si estando en este minuto se pulsa de nuevo el botón C, la secuencia comenzará de nuevo en el 0.
En este caso, para pasar al siguiente paso y así guardar el minuto inicial, se pulsa el botón A.

5. ***Elección de la hora final:*** Una vez que se ha seleccionado la hora y el minuto inicial, falta asignarle valor a la hora y minuto final en el que la bombilla o grupo de bombillas se desactivarán.

   El procedimiento para este caso es el mismo, según se pulsa el botón A el led cuatro y todos los anteriores se activarán indicando que el usuario está en el paso cinco para establecer el temporizador. Además, se muestra en el display el mensaje: *"SELECCIONE HORA FINAL"* y al finalizar, se muestra *"H- 0"*. La letra "H" como se explicó anteriormente indica la hora, en este caso, la hora final en el que las bombillas se desactivarán y junto a ella, su valor, por defecto empieza en el 0, es decir, a las doce de la noche. Si se quiere modificar este valor hay que realizar el mismo procedimiento que en la fase de seleccionar la hora inicial en el que mediante el botón C se aumenta el número hasta seleccionar la hora que desee el usuario.

   Una vez escogido el valor, para pasar a la siguiente fase se debe de pulsar el botón B.

6. ***Elección del minuto final:*** Para concluir el periodo de tiempo en el que la bombilla/s va a estar encendida se tiene que asignar valor al minuto final.

   El funcionamiento de la fase del minuto inicial y el minuto final es idéntico. Se presiona el botón B y al realizar esta acción, se activa con ello el led cinco. Además, se muestra por los cuatro displays de la interfaz el mensaje de texto: *"SELECCIONE MINUTO FINAL*" y acto seguido, *"M- 0"*. La letra "M" hace referencia a la palabra "Minuto" y el 0, al valor del minuto que se le quiera asignar. Con el botón C de la interfaz se podrá modificar ese valor en intervalos de cinco en cinco.

   Para almacenar ese valor y guardar la hora final se debe de pulsar el botón A.

7. ***Establecido el temporizador:*** Al finalizar la elección del minuto final, la configuración del temporizador diario concluye.

   Después de introducir todos los datos, se llega a la última fase del *Temporizador* que para indicárselo al usuario el programa enciende el último led que faltaba por activarse. Con todos los leds ya encendidos, en el display de la Raspberry Pi se muestra el mensaje: *"TEMPORIZADOR DIARIO*

ACTIVADO". A continuación de este mensaje, el programa le muestra al usuario el día y la hora a la que se ha establecido el temporizador. A partir de este mensaje, el usuario compara si los datos que él ha introducido coinciden con los que se muestran por el display. Esta configuración se implementó así, para que el usuario comprobase antes de abandonar la vivienda si se iba a aplicar de forma correcta el temporizador y si no es así, que tuviese la opción de volver a programarlo.

El programa después de mostrar los datos que se han introducido, como el día y el periodo de tiempo en el que la bombilla/s va a estar encendida, mostrará la hora en tiempo real en los cuatro displays de la Raspberry Pi. La separación de las horas y los minutos en la interfaz no se visualiza con dos puntos ":" debido a que los displays del Rainbow Hat no contemplan esta posibilidad con lo que se realiza mediante un único punto, por ejemplo, *"22.34"*.

Como se puede observar en la Ilustración 13, cada display está formado por 14 segmentos y un punto en la parte inferior derecha de cada uno de ellos. Para establecer la hora con el formato que se ha explicado anteriormente, se debe de encender el punto inferior derecho del segundo display. Esto se lleva a cabo mediante la siguiente sentencia:

SENTENCIA PARA ENCENDER EL PUNTO DECIMAL
`rainbowhat.display.set_decimal(1,True)`

Tabla 27: Sentencia para encender un punto decimal

Con la biblioteca *"rainbowhat"* se accede mediante instancias al punto decimal que se desea activar. La función para activar el led del punto decimal es, *"set_decimal"* la cual recibe dos parámetros, en el primero, se le debe pasar un número entre el cero y el tres donde el cero es el punto decimal del display empezando por la izquierda y el número tres, el punto decimal del último display de la derecha. Con el segundo parámetro, se le indica si el punto decimal elegido se debe de activar o no, para ello, se utiliza una variable booleana en el que si recibe true el led se encenderá y si, por el contrario, es false, el led se desactivará.

En este caso, se implementa esta sentencia para encender el punto decimal del segundo display y así, mostrar mediante la Raspberry Pi un reloj. El reloj se mantendrá en el display hasta la finalización del temporizador diario.

Una vez que se ha encendido y apagado el temporizador en las horas fijadas, el temporizador diario finaliza con el mensaje, *"FIN DEL TEMPORIZADOR"*.

A continuación, se muestra como se ha implementado el código para llevar a cabo esta última fase y una breve explicación del mismo:

FUNCIÓN PARA EJECUTAR EL TEMPORIZADOR DIARIO

```
def p_diaria():

    global bulbid, activado

    mensaje="TEMPORIZADOR DIARIO ACTIVADO
             DIA->{dia}/{mes}/{year}".format(dia=dia,
             mes=mes, year=year)

    scroll_mensaje(mensaje)
    rainbowhat.display.clear()
    ajustar_tiempo()

    while activado==True:

        t=datetime.now()
        hora_sistema=t.hour
        minuto_sistema=t.minute

        if(t.hour==hora_inicial and
           t.minute==minuto_inicial):

           switch_on(hubip, apiuser, apikey, bulbid)

           while activado==True:

               t=datetime.now()
               hora_sistema=t.hour
               minuto_sistema=t.minute
               tiempo_sistema(hora_sistema,minuto_sistema):

               if(t.hour==hora_final and
                  t.minute==minuto_final):

                   switch_off(hubip,apiuser, apikey,bulbid)
                   activado=False

            tiempo_sistema(hora_sistema,minuto_sistema)

   final()
```

Tabla 28: Función para ejecutar el temporizador diario

Primero, se comenzó implementando el mensaje que indica al usuario que el temporizador se ha activado. Dentro de ese mismo mensaje, se imprime el día en que se ha establecido el temporizador que mediante el comando *".format"*, que formatea las variables de la cadena de texto con las variables globales del script. Este mensaje se representa en modo barrido para poder visualizar la cadena de texto completa. A continuación, se llama a la función *"ajustar_tiempo()"* la cual se basa en mostrar la hora de inicio y la hora final del temporizador de forma correcta. Con la forma correcta se refiere a que cuando los minutos tanto iniciales como finales son de una sola cifra, en el display debe de visualizarse un valor de la variable minutos de dos cifras añadiendo un cero a la izquierda.

Después de implementar el código para mostrar la información principal, se generó un bucle *"while()"* para que cuando la hora indicada por el usuario sea igual a la hora recibida por el sistema entonces se produzca el encendido de la bombilla o bombillas que se hayan elegido.

Después de encenderse, al cumplirse la condición, el programa entra en otro bucle *"while()"* en el que estará ejecutándose continuamente hasta que la condición se cumpla. En esta condición se evalúa la hora del sistema con la hora final que le ha asignado el usuario. Cuando finaliza el bucle se le llama a la función *"final()"* para mostrar en el display el texto para finalizar la aplicación.

- **Temporizador Semanal:**

El temporizador semanal es la segunda opción que puede escoger el usuario. Dentro de este apartado se va a explicar el objetivo fundamental y el modo en el que se desarrolla y se ejecuta en la Raspberry Pi.

Este temporizador se creó porque la función de esta aplicación se basa en programar el encendido y apagado de la bombilla o bombillas cuando el usuario no se encuentra en la vivienda. El problema surge cuando el propietario está ausente un periodo de tiempo superior a un día, el temporizador diario no serviría debido a que solo cubre un día especifico. Por este aspecto, se desarrolló este tipo de temporizador con el que es posible repetir la misma función que se lleva a cabo en el temporizador diario, pero durante una semana repitiéndose a la misma hora fijada de lunes a domingo.

El usuario debería de escoger el temporizador semanal en el caso de que se fuese a ausentar entre dos a siete días puesto que es la opción que mejor le conviene.

Después de explicar la finalidad del temporizador semanal, se va a dar paso a describir el desarrollo del programa cuando el usuario elige la segunda opción con el botón B.

Al pulsar el botón, los siete leds de la interfaz Rainbow Hat muestran una secuencia luminosa activándose uno a uno desde el led seis al led cero sin desactivarlos, al contrario que en el temporizador diario. Esta secuencia, representa la actividad que lleva a cabo el temporizador puesto que su objetivo es encender la bombilla en un periodo de tiempo establecido desde el lunes hasta el domingo, es decir, los siete días de la semana que es el mismo número de leds que contiene la interfaz. En paralelo, se muestra en el display el texto *"2.SEM"* indicando al usuario la opción que ha seleccionado.

A continuación, el programa está implementado para que realice los mismos pasos que en el temporizador diario, con lo que el usuario tendrá que seleccionar el día, mes, hora inicial, minuto inicial, hora final y minuto final para activar el temporizador semanal.

La estructura del programa en Python es muy similar al temporizador diario salvo por un aspecto, seleccionar el día de activación.

En el primer paso, cuando la Raspberry Pi solicita al usuario ingresar el día en el que se desea activar la bombilla, el propietario no puede seleccionar el día actual salvo que este sea lunes. El programa está implementado para reconocer el número de la semana del año que quiere el usuario encender la bombilla, para ello, el propietario debe seleccionar siempre el lunes de la semana que quiere programar el temporizador, aunque en realidad el día actual no sea lunes.

El número de la semana se utiliza para repetir el proceso de encendido de la bombilla en el periodo establecido por el usuario mientras que dicho número sea el mismo, cuando cambie, es decir, el lunes siguiente a las 00:00h también cambiará simultáneamente el número de la semana con lo que el temporizador ya no funcionaria.

Para resumir, en este temporizador se pueden producir dos sucesos al programar la bombilla:

- **Semana completa:** En este proceso se desarrolla el temporizador durante los siete días de la semana. Este suceso ocurre cuando el usuario selecciona un lunes posterior al de su número de semana, o también, si el día actual es lunes y selecciona ese mismo día, el programa se ejecutará durante toda esa semana.

- **Semana parcial:** En este caso, toma el nombre de semana parcial porque como su propio nombre indica el temporizador no se desarrolla durante todos los días de la semana. Esto es debido a que el usuario ha seleccionado el lunes de la propia semana en que se encuentra sin que el día actual fuese lunes. Es decir, si, por ejemplo, el día en que se encuentra el usuario es el 28 de marzo de 2018, al seleccionar el lunes 26 de marzo de 2018, el temporizador se ejecutará desde ese miércoles hasta el domingo, que es el último día que cubre ese número de semana.

En la siguiente tabla se muestra como se ha implementado en código Python la explicación anterior:

FRAGMENTO DE CÓDIGO PARA COMPROBAR SI EL DÍA INTRODUCIDO ES LUNES

```
if semanal==True:

      fecha_semanal=datetime(year,mes,dia,0,0,0)
      tupla_valores=datetime.isocalendar(fecha_semanal)
      dia_semana=tupla_valores[2]
      n_semana=tupla_valores[1]

      if dia_semana!=1:

          mensaje="EL DIA INTRODUCIDO NO ES LUNES"
          scroll_mensaje(mensaje)
          dia_activado=False
          mes_activado=False
          elegir_dia(dia_activado)
```

Tabla 29: Fragmento de código para comprobar si el día introducido es lunes

Para comprobar si el día introducido es lunes, se implementó este fragmento de código en Python que sirve para comprobar si el temporizador es semanal.

Una vez comprobado, se obtiene la fecha a partir de los parámetros de entrada (año, mes y día), y gracias a la clase *"isocalendar"*, que está contenida dentro del módulo *"datetime"*, se obtiene una tupla con tres valores (año, número de semana y día de la semana).

Una tupla es una lista de datos que no se puede modificar después de haberla creado. Esta tupla nos devuelve el día de la semana en formato de número entero y mediante un condicional, se comprueba si el día de la semana es distinto a 1. Es decir, como el 1 equivale a lunes, cuando el día sea el primer día de la semana seguirá con el programa, por el contrario, si el día seleccionado no es lunes

entonces se imprimirá un mensaje de texto en la Raspberry Pi indicando que el día introducido no es lunes y volverá a pedir al usuario que introduzca un día correcto. A continuación, el programa pide al usuario los siguientes cinco pasos restantes igual que en el temporizador diario.

En la última fase, después de establecer todos los datos anteriores, se activa el temporizador mostrando en el display el mensaje: *"TEMPORIZADOR SEMANAL ACTIVADO"*, además, mostrará el número de la semana del año en que ha elegido establecer el programa y el día que se ha establecido.

En la siguiente tabla, se expone la función que se ha implementado para poder ejecutar el temporizador semanal.

FUNCIÓN PARA EJECUTAR EL TEMPORIZADOR SEMANAL

```
def p_semanal():

    global activado, n_semana, dia_semana, bulid

    mensaje="TEMPORIZADOR SEMANAL ACTIVADO
             SEMANA->{n_semana}DIA->{dia}/{mes}/{year}
           ".format(n_semana=n_semana,dia=dia,mes=mes,year=year)

     scroll_mensaje(mensaje)
     rainbowhat.display.clear()
     ajustar_tiempo()
     activo=False
     numero=n_semana
     contador=0

     while activado==True:

           t=datetime.now()
           fecha_semanal=datetime(year,mes,dia,0,0,0)
           tupla_valores=datetime.isocalendar(fecha_semanal)
           n_semana=tupla_valores[1]
           hora_sistema=t.hour
           minuto_sistema=t.minute

           if (n_semana==numero and t.hour==hora_inicial and
                t.minute==minuto_inicial):

                switch_on(hubip, apiuser, apikey, bulbid)
                activo=True
                contador=1

                while activo==True:
```

```
                    t=datetime.now()
                   hora_sistema=t.hour
                   minuto_sistema=t.minute
                   tiempo_sistema(hora_sistema,minuto_sistema)

                   if(n_semana==numero and t.hour==hora_final
                      and t.minute==minuto_final):

                       switch_off(hubip,apiuser,
    apikey,bulbid)
                       activo=False

     elif (n_semana==numero and contador==1):

           activado=False

     tiempo_sistema(hora_sistema,minuto_sistema)
final()
```

Tabla 30: Función para ejecutar el temporizador semanal

Después de mostrar por display los mensajes informativos en relación a los datos que se han establecido, se llamó a la función *"ajustar_tiempo()"*, igual que en el temporizador diario, sirve para mostrar tanto la hora y minuto inicial como la hora y minuto final de forma satisfactoria. A continuación, se realizó un bucle *"while()"* para cuando el temporizador semanal estuviese activado, recogiese todos los datos necesarios para la activación de la bombilla o bombillas en el periodo establecido por el usuario. El bucle se genera continuamente hasta que activado sea false. Dentro del *"while()"*, hay un condicional en el que, si cumple las condiciones de que el número de la semana, la hora inicial y el minuto inicial coinciden con los valores que ha establecido el usuario, entonces, se encenderá la bombilla hasta que el número de la semana, hora y minuto final coincida con las que asigno el usuario para finalizar el periodo de activación. Cuando el número de la semana ya no sea igual y el contador sea igual a uno entonces finalizará el temporizador mediante la función final.

- **Temporizador Mensual:**

Por último, se explica el temporizador mensual que es la tercera opción que puede elegir el usuario mediante el botón C.

En verano, la mayoría de las personas eligen disfrutar sus vacaciones fuera de sus hogares. Ese periodo vacacional, por lo general, no suele ser un periodo corto de una semana o inferior con lo que se pensó en implementar está último temporizador debido a este tipo de circunstancias.

La función de este temporizador es encender y apagar la bombilla o el grupo de bombillas en el periodo establecido por el usuario durante el mes completo o lo que reste de mes desde que lo inicie el propietario.

Para el temporizador mensual también habrá dos características dependiendo el mes en que el usuario quiera iniciar el programa:

- **Mes completo:** Esta opción se produce cuando el usuario quiere establecer el temporizador mensual para un mes completo. Esto se puede llevar a cabo eligiendo un mes posterior al que se encuentra el propietario o activando el temporizador si se encuentra en el primer día de mes. Por ejemplo, si el usuario está en el mes de junio y quiere programar este temporizador para el mes de julio entonces el programa se llevará a cabo durante todo el mes completo.

- **Mes parcial:** Este caso se produce cuando se establece el mismo mes en el que está el usuario, es decir, si la persona que está programando el temporizador mensual se encuentra en el día 7 de abril, si selecciona el mes cuatro, lo que estará programando será el temporizador mensual desde el día que lo active, en este caso, el día 7 de abril hasta que finalice el mes de abril el día 30.

Después de explicar en qué consiste este temporizador, se va a exponer como se muestra el proceso en la Raspberry Pi y como se ha implementado.

Al seleccionar el botón C, el Rainbow Hat muestra en el display el mensaje *"3.MES"* indicando que se ha seleccionado este tipo de temporizador. Simultáneamente, los leds de la interfaz se activan parpadeando tres veces todos en bloque. Para este caso, se implementó así puesto que lo que se va a activar es el mes completo o parte de él hasta su fin. A continuación, se le pide al usuario los datos necesarios para establecer el temporizador. A diferencia del temporizador diario y semanal, el mensual comienza en el segundo paso, elegir el mes en el que se quiere activar el programa.

Como se ha explicado anteriormente, no se elige día porque el programa dependiendo el mes que elija el usuario puede establecerse el temporizador durante todo el mes completo o durante un tiempo parcial.

El resto de los pasos posteriores para seleccionar el periodo de tiempo que estará la bombilla encendida durante el día son idénticos a los otros dos temporizadores. En la última fase, se muestra mediante el display los datos que se han seleccionado y posteriormente, se ejecuta el temporizador.

En la siguiente tabla se muestra como se implementó esta última fase:

FUNCIÓN PARA EJECUTAR EL TEMPORIZADOR MENSUAL

```
def p_mensual():

    global activado, n_semana, dia_semana, bulbid

    mensaje="TEMPORIZADOR MENSUAL ACTIVADO EN EL
             MES->{mes}/{year}".format(mes=mes, year=year)

    scroll_mensaje(mensaje)
    rainbowhat.display.clear()
    ajustar_tiempo()
    activo=False
    contador=0

    while activado==True:

          t=datetime.now()
          mes_sistema=t.month
          hora_sistema=t.hour
          minuto_sistema=t.minute

          if (mes_sistema==mes and t.hour==hora_inicial and
             t.minute==minuto_inicial):

              switch_on(hubip, apiuser, apikey, bulbid)
              activo=True
              contador=1

               while activo==True:

                    t=datetime.now()
                    hora_sistema=t.hour
                    minuto_sistema=t.minute
                    mes_sistema=t.month
                    tiempo_sistema(hora_sistema,minuto_sistema)

                    if(mes_sistema == mes and
                       t.hour==hora_final
                       and t.minute==minuto_final):

                        switch_off(hubip,apiuser, apikey,bulbid)
                        activo=False

            elif (mes_sistema!=mes and contador==1):
```

```
                    activado=False


            tiempo_sistema(hora_sistema, minuto_sistema)
    final()
```

Tabla 31: Función para ejecutar el temporizador mensual

Lo primero que indica es que el temporizador mensual se ha activado y acto seguido, muestra el mes en que se va a realizar el programa. A continuación, se implementó un bucle en el que está obteniendo continuamente los datos acerca del mes, hora y minuto inicial.

Mediante una condicional comprueba si los datos que ha seleccionado el usuario coincide con el mes, hora y minuto en tiempo real. Si coincide, entonces se activa la bombilla/s y al igual que antes, el programa se introduce dentro de otro bucle que recogerá los datos del sistema y hasta que esos datos no coincidan con el tiempo final fijado por el usuario no se apagará la bombilla. Cuando el mes del sistema ya no coincida con el mes que se ha establecido anteriormente y el contador sea igual a 1 entonces se finalizará el temporizador habiendo realizado con éxito su función. Para entrar en el condicional una de las condiciones que se tiene que dar es que contador sea igual a uno, esto se realizó porque sin esta condición si el usuario eligiese un mes futuro entonces se finalizaría el programa sin ejecutarse el temporizador correctamente.

6. PLANIFICACIÓN Y PRESUPUESTO

En este capítulo se analiza la planificación que se ha llevado a cabo durante el proyecto y el presupuesto requerido para poder realizarlo.

6.1. Planificación

Para representar de forma adecuada la planificación del proyecto, se ha analizado mediante el siguiente Diagrama de Gantt:

		SEMANAS																		
Etapas	**Duración**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**
Planteamiento inicial y selección de objetivos	**2**																				
Adquisición del material	**2**																				
Instalación e iniciación al entorno	**3**																				
Mando remoto	**4**																				
Temporizador	**5**																				
Realización de pruebas y análisis de resultados	**3**																				
Documentación del proyecto	**20**																				

Tabla 32: Planificación del proyecto mediante el Diagrama de Gantt

En el Diagrama de Gantt se muestran las etapas que constituyen el proyecto y la duración de las mismas. El período de duración del proyecto ha sido de 20 semanas.

A continuación, se va a describir brevemente cada una de las etapas del proyecto:

- **Planteamiento inicial y selección de objetivos**

Este apartado es la primera etapa del cronograma. En esta fase se investigó acerca de los temas más influyentes del trabajo como son Internet de las cosas, los protocolos de comunicación, Python, etc.

Además, en este punto se definieron los objetivos que iba a tener el proyecto y el modo en que se iba a llevar a cabo para conseguirlos.

- **Adquisición del material**

En esta etapa se llevó a cabo la búsqueda del material para el desarrollo y las pruebas del proyecto con el fin de cumplir los objetivos establecidos anteriormente. Una vez encontrados los dispositivos con la mejor relación calidad-precio se adquirieron.

- **Instalación e iniciación al entorno**

Esta actividad se basó en familiarizarse con los dispositivos que se habían adquirido anteriormente. Lo primero que se llevó a cabo fue la instalación de todos los dispositivos y, a continuación, se analizó su funcionamiento y sus limitaciones.

También se investigó el entorno donde se iba a desarrollar el proyecto y se realizaron diversas pruebas con el fin de observar cómo funcionan los protocolos de comunicación y las tecnologías adquiridas.

- **Mando Remoto**

En esta etapa se desarrolló la primera aplicación piloto, el *Mando Remoto*. Esta fase es una de las más importantes y duraderas del proyecto puesto que se basa en la implementación del primer software. Con este prototipo se cumple uno de los objetivos del proyecto.

- **Temporizador**

El *Temporizador* es la segunda y última aplicación piloto que se desarrolló. Esta fase es más duradera debido a que la implementación del código fue más costosa que la del *Mando Remoto*. Con esta aplicación se cumplió otro de los objetivos establecidos.

- **Realización de pruebas y análisis de resultados**

Después de desarrollar ambas aplicaciones piloto se llevaron a cabo un número elevado de pruebas exhaustivas con el fin de solventar los posibles problemas que pudiesen provocar. Para ello, se modificaban los parámetros de entrada pensando en cómo lo haría un usuario para que no tuviese ningún inconveniente.

En esta etapa, una vez que se realizaron las pruebas y se comprobó que ambos programas eran consistentes, se analizaron los resultados y los objetivos marcados.

- **Documentación del proyecto**

Por último, el proyecto se ha ido documentando paralelamente a su ejecución. En el documento se redacta el desarrollo de todo el proceso del trabajo, las tecnologías y los protocolos utilizados y los resultados obtenidos.

6.2. Presupuesto

En este apartado se definen los costes tanto de las personas involucradas en el proyecto como de los dispositivos utilizados.

- **Presupuesto del personal**

Apellidos Nombre	Categoría	Dedicación (personas mes)	Coste persona mensual	Coste Total (Euros)
García Rubio Carlos	Ingeniero Senior	0,22	4.289,54	943,70
Arribas Palomo Alejandro	Ingeniero Junior	2,05	2.694,39	5.523,50
	Persona mes	***2,27***	***Total***	***6.467,2 €***

Tabla 33: Presupuesto del personal

Este presupuesto se obtiene considerando que una persona al mes trabaja 132 horas. El proyecto dura un total de 300 horas comprendidas entre 20 semanas.

- **Presupuesto del material**

Descripción	Coste (Euros)	% Uso dedicado al proyecto	Dedicación (meses)	Período depreciación	Coste imputable
Starter Kit Raspberry Pi 3	90,74	100	5	60	6,80
Kit Trådfri LED	79,99	100	5	60	6,00
Monitor	69,99	100	5	60	5,83

Teclado	12,50	100	5	60	1,041
Ratón	8,29	100	5	60	0,69
Ordenador portátil	683	100	5	60	56,90
Router	62,10	100	5	60	5,17
TOTAL					***82,43***

Tabla 34: Presupuesto del material

El coste imputable se obtiene a partir de la siguiente fórmula:

$$Coste\ imputable = Coste\ de\ adquisición * \%\ Uso * (\frac{Dedicación}{Período\ de\ depreciación})$$

- **Resumen de costes**

Presupuesto Coste Totales	Presupuesto Coste Totales
Personal	6.467,2
Amortización	82,43
Subcontratación de tareas	0
Costes de funcionamiento	0
Costes indirectos	1.309,93
TOTAL	***7.859,56***

Tabla 35: Resumen de costes

7. CONCLUSIONES

7.1. Conclusión

IoT es una de las tecnologías con más crecimiento en los últimos años. Internet de las cosas es una red en la cual están interconectados entre sí un número elevado de dispositivos con el fin de poder transmitir información entre ellos, realizar acciones sobre objetos, etc.

Cada día hay más dispositivos que son capaces de conectarse a Internet y con ello, de poder ser controlados remotamente por otras máquinas. Este incremento se ha conseguido en parte gracias a la implantación que se está llevando a cabo con el protocolo de Internet, IPv6. Con este protocolo es posible identificar a todos los dispositivos con Internet otorgándoles una dirección IP a cada uno.

En este proyecto se ha querido reproducir un entorno de la tecnología de Internet de las cosas mediante dispositivos CoAP, en este caso, son dos bombillas inteligentes las cuales son controladas inalámbricamente a través de la Raspberry Pi modelo B.

Este tipo de bombillas están conectadas a Internet a través de la puerta de enlace y al ser de tipo led, el usuario puede interactuar con ellas ajustando su intensidad y modificando su color.

Los objetivos del proyecto eran investigar de primera mano el funcionamiento de la tecnología de Internet de las cosas con el fin de poder desarrollar dos aplicaciones piloto a través del lenguaje de programación Python. Las aplicaciones que se han llevado a cabo son el *Mando Remoto* y el *Temporizador*.

Al iniciarse en el entorno de Internet de las cosas surgieron ciertos problemas con el protocolo CoAP sobre DTLS por motivos de autenticación, pero se resolvieron investigando a cerca de los protocolos en distintas RFCs [5][6][8]. Otro problema fue controlar el encendido y apagado de las bombillas dependiendo de la cantidad de luz que se introduzca en los sensores de iluminación. El problema es que la interfaz Rainbow Hat no contiene este tipo de sensores con lo que no se pudo realizar esta configuración. La interfaz que contiene los sensores de iluminación es Enviro pHAT.

Los objetivos marcados al principio del proyecto se han cumplido satisfactoriamente puesto que la función de ambas aplicaciones ha sido el esperado.

El objetivo de la primera aplicación era desarrollar una aplicación llamada *Mando Remoto* cuya funcionalidad fuese similar al mando remoto que viene incorporado en el kit Trådfri Led, pero dándole más dinamismo a través de la Raspberry Pi. Esta aplicación se llevó a

cabo para aprender a controlar todos los ajustes que se podían realizar en las bombillas y así conseguir reproducir el mismo software que se vende en el mercado, pero mejorado puesto que es más dinámico. El programa mejora en varios aspectos como elegir que bombilla quieres configurar, cambiar de bombilla si el usuario lo requiere, programa más visual, etc. Por estos motivos, el objetivo de la primera aplicación se ha conseguido con éxito.

El segundo objetivo era implementar la aplicación *Temporizador*. Es una aplicación que se desarrolló con el fin de garantizar cierta protección en el hogar cuando el usuario no se encuentre en el interior. Evaluando el resultado de la aplicación se considera que se ha cumplido el objetivo puesto que la finalidad de crear un temporizador se cumple, y, además, se ha implementado dentro del programa tres tipos de temporizadores (diario, semanal o mensual) ofreciendo facilidades a las necesidades del usuario.

Como conclusión se podría decir que ambas aplicaciones cumplen con lo previsto e incluso se han mejorado con respecto a la idea principal.

7.2. Opinión Personal

La razón por la que elegí este proyecto fue porque me causó gran interés investigar y desarrollar un trabajo acerca de Internet de las cosas. Con esta oportunidad he podido adquirir conocimientos de una tecnología que está en continuo crecimiento y dentro de unos años, tendrá gran repercusión en el mercado tecnológico.

El proyecto se centra en desarrollar dos aplicaciones piloto que consigan controlar unas bombillas inteligentes a través de la Raspberry Pi. Creo que haber aceptado este proyecto es un gran paso para encauzar mi futuro en el sector tecnológico ya que he comprobado de primera mano cómo sería la gestión de un proyecto similar al realizado.

Además, el proyecto se ha realizado en lenguaje de programación Python. Este es un lenguaje que no había estudiado durante mi etapa en la Universidad y adquirir este conocimiento de cara al futuro, me haría crecer profesionalmente, por lo que, aparte de llamarme la atención hacer el trabajo acerca de IoT, tenía su reto extra, aprender un lenguaje de programación nuevo como es Python.

Por estos motivos, he realizado un trabajo que me ha llenado personalmente y donde estoy seguro que no se va a quedar aquí. Mi objetivo personal es continuar investigando esta tecnología fuera de la Universidad para llevarlo a cabo en mi futuro hogar, no solo para la iluminación, sino también, para otros objetos, como pueden ser, controlar el termostato, la puerta del garaje, las persianas, pero todo habiéndolo desarrollado personalmente.

7.3. Líneas futuras

En este apartado se explica las posibles mejoras que se le puede añadir al trabajo para que las aplicaciones sean más eficientes y productivas.

Uno de los puntos clave a mejorar es el ahorro de energía. La idea es que después de que el usuario configure el *Mando Remoto* o el *Temporizador*, en vez de que la Raspberry Pi se quede encendida mostrando el menú en el primer caso o la hora en el segundo, está se quede en suspensión, pero sin dejar de ejecutarse el programa internamente. De esta forma los displays y los leds de la Raspberry Pi no estarían encendidos constantemente y el usuario ahorraría energía.

La aplicación *Temporizador* sirve para que desde el exterior del hogar de la impresión de que hay alguien en su interior y así reducir posibles robos en la vivienda. Este porcentaje se podría reducir incluso más si se desarrollase un programa idéntico al del *Temporizador*, pero para otro objeto inteligente del hogar como, por ejemplo, las persianas. Así, el usuario aparte de programar las bombillas también lo podría hacer con las persianas, y ambas se podrían ir alternando durante el día para dar impresión de que alguien se encuentra dentro.

Como última mejora, pensando desde el punto de vista del usuario, sería poder controlar a distancia las distintas configuraciones de las bombillas, tanto el encendido y apagado como la intensidad y el color. Esto se podría llevar a cabo, por ejemplo, enviando un mensaje a través del correo electrónico poniendo palabras clave que la Raspberry Pi pueda descifrar y acto seguido ejecutar la acción. Con esta mejora se conseguiría controlar a distancia cualquiera de las dos aplicaciones sin la necesidad de hacerlo a través de la Raspberry Pi.

8. ENGLISH SUMMARY

In this chapter, it is explained a summary of the project for explaining the issue and the objetives of it.

The Project is based on developing two pilot applications relating to Internet of Things. It has been used a Raspberry Pi 3, a Gateway and two intelligents light bulbs Trådfri Led in order that the software was possible.

IoT is a technology that is increasing for last decade. It refers to the possibility of every object can be connected to Internet and be able to communicate, share, generate and consume data among them without the necessity of an user.

Internet of Things was born when the number of objects connected to network was bigger than the amount of people in the world. Everything that is connected by Internet Protocol (IP) needs an address IP as identifier. The number of objects capable of connecting were so high that the protocol IPv4 couldn't admit so many IPs. So that, it was created a new versión, IPv6 that makes possible the connection of every device you need.

To make possible the Project, the kit Starter Raspberry Pi 3 for Android Things and the kit Trådfri Led were bought.

The kit Raspberry Pi 3 is the hardware that makes the work. It is an improved version with a little, faster and powerful CPU. It includes a quad-core processor. It runs at 1.2 GHz. It has been choosen because it is prepared for the development of IoT applications, due to its Wireless networking. The WI-FI connection will be able to control both intelligent bulbs sending a request to the gateway. Moreover, the Rainbow HAT is a hardware connected to the Raspberry Pi. Due to its characteristics it allows create many applications. It is composed by four displays, some leds and three buttons that makes possible interact with the user.

The kit Tradfri Led is composed by two intelligent bulbs. Not only are they used to light up, but they also can be configured to control the intensity and choose the color among three options Cold, Warm and Normal. The kit includes a remote control to configure the light bulbs. It is formed by five buttons to control the settings from a distance.
Inside this kit, it is found the gateway that connects up the wire ethernet with the router and make possible the Internet connection.

After explaining the devices that have been needed to the Project, the objectives of it will be analyzed.

- **Environment Introduction**

The first objective was started with the Internet of Things environment in relation to the Tradfri Led light bulbs. Different tests were carried out in order to check the bulbs control and so analyzing the communication among the Raspberry Pi, the gateway and the light bulbs.

When the user makes an action over the light bulbs through the Raspberry PI is sending a request for the gateway with the communication protocol CoAP on DTLS. The Constrained Application Protocol was designed to establish a restricted communication between two machines. In this Project, the protocol CoAp is used as client because Raspberry Pi is the applicant that request the action from the server.

If the client wants to make a request, it will bring about through a request that can be Confirmable (CON) or Non-Confirmable (NON). The request is formed by a method which is headed for a resource, the resource identification, the payload and metadata related to the request.

The methods that has been used are:

- **GET:** It is based on getting information from the request resource (URI-Host).

- **POST:** The client can apply from the request that has been identified previously that the representation enclosed was processed.

- **PUT:** It is able to apply that the resource identifier by the request replaces with the data that has been got from the payload.

- **DELETE:** The resource identified can be deleted by the identifier has been sent in the request.

After sending the request, the server process and sends a CoAP answer to the client. In order to connect the request with the answer it is used a token which is previously created in the request.

The answer has a format ‘c.dd’ where the ‘c’ indicates the kind of answer that is sent and ‘dd’ the answer details.

There are three types of anwers:

- **Success:** 'c' gets a code value 2, indicating that the request has been successfully received and precessed.

- **Client Error:** 'c' gets a code value 4, in case that the request that was sent have an error.

- **Server Error:** 'c' gets a code value 5, if the server doesn't understand or doesn't process a correct request.

The CoAP protocol protects the request with the Datagram Transport Layer Security protocol. The DTLS is responsible for the security on the data transmission between two host. In order to identify the resources, CoAP uses URI schemes as "coaps".

After explaining the communication between the Raspberry Pi 3 and the Gateway, the next step will be analysing how the order is transmited to the bulbs from the Gateway.
To be succesful on this we need the ZigBee protocol.

ZigBee is a Wireless protocol used in IoT (Internet of Things). It is based on the 802.15.4 standard and it is used in applications where the transmission of the data volumen is reduced and there is a simple conectivity.

ZigBee is a Wireless protocol used in IoT (Internet of Things). It is based on the 802.15.4 standard and it is used in applications where the transmission of the data volumen is reduced and there is a simple connectivity.
Simple connectivity makes reference to connections from 'point to point' or from 'point to multipoint', as it has been realized in this Project, from the Gateway to bulbs.

Not only is the low cost a characteristic of this protocol but it is also the energy-efficient light bulbs because after transmitting the data to the final URI-host, the protocol suspends its activity to save energy.

The Interoperability or the capacity that two or more devices have for changing data between them and use those data with a certain objetive is other ZigBee protocol characteristic.

To sum up, ZigBee is a protocol that is used for intelligent devices responsible for home automation.

The two applications that has been created so as to the Project was succesful are a Remote Control and a Timer. To create both applications it has been used the programming language Python.

Python is a programming language. It is easy to learn because it has an efficient high-level data structures. Besides, Python is an ideal language for developping scripts on easy applications as it has been used to create this Project.

The programmes that are developped by Python use to be more reduced than the programmes that use a 'C' or a 'Java' language. This is because Phyton can generate complex operations with an only command. Commands are typed with an indentation instead of squared brackets and it is not necessary typed variables.

- **Remote Control**

After finishing the knowledge process from the web page Pimoroni, it was designed the first pilot application: the Remote Control. This application has been created in order to manage to replicate and improve the remote control that is including in the kit Trådfri Led.

The remote control program was developped for two reason. First, it is a practical application because the user can control the light bulbs from it. Secondly, it is a new application due to not only does the remote control switch on and switch off the light bulbs but it also controls the intensity and the color of them.

The remote control program was developped for two reason.
First, it is a practical application because the user can control the light bulbs from it. Secondly, it is a new application due to not only does the remote control switch on and switch off the light bulbs but it also controls the intensity and the color of them. Moreover, as the Project has been created over the Rainbow Hat hardware, the remote control will be more dynamic than a standard one because of the displays and the seven leds the remote control will be more visual.

Running the remote control, the applications starts with an initial messages that welcomes the user. Next, the program ask for which light bulb does the user want to control or if the user wants to control several light bulbs at the same time. This can be done by the three Raspberry Pi buttons. Comparing to the remote control including in the kit Trådfri Led this is a better applications because the original can't offer that possibility.

Once that the user has chosen the light bulb the program runs to the application menu.
Inside the menu there are three options to control the light bulbs.

Through the first option it can chose between switch on and switch off the light bulbs, secondly, it can select the light intensity and finally it can modify the color.

While the original remote control only allows to control all the light bulbs at the same times, the new application lets the user control the light bulbs one by one or all together. Therefore, a saving of electrical energy can be produced.

During the running of the program the light bulbs would paint in a characteristic way in order to the user finds a connection between the light bulbs response and the secuency that are showed in leds.

- **Timer**

The following objetives was designed an innovative application to satisfy the users necessities.

Holiday and weekends are the periods were the robbery considerably increases because the user don't use to be at his/her residence. For this reason, it was designed a timer, an application to providing security at home when the user wasn't in.
The principal function is making believe that there is someone at home all the time. So as to, the user must actívate the timer before leaving the residence, setting the light bulbs switched on and switched off connected to the Trådfri network during a determine period of time.

For this reason, it has been created three classes of timers in order to the user can chose the most suitable for his/her necessities.

- **Daily timer:** This kind of timer has been created to be used when the user goes out for a day or a few hours. To active the daily timer, the user has to select previously the day, the month and the time he is going to be out. The Trådfri led light bulbs run the day and the time the user will has setted.

- **Weekly timer:** The weekly timer function is based on setting the timer during a week. The user must set the period of light bulbs switched on and switched off from Monday to Sunday without the possibility of changing this configuration. This timer is advisable if the user are out for a weekend or a week holiday.

- **Monthly timer:** Finally, it was developped a monthly timer in case that the user need setting the light bulbs for more than a week. The user only has to select the month and the time in which he/she wants to switch on and switch off the light bulbs.

To conclude, the aim of this Project has been successfully achieve at the beginning. As far as I am concerned, this Project has been rewarded due to I have acquired knowledge about one of the most promising technological market in the field of the home automation. What is more, I have managed to apply that knowledge in a practical way in order to get two pilot applications with a particular purpose successfully.

9. GLOSARIO

IoT: Internet of Things

IPv4: Internet Protocol Version 4

IPv6: Internet Protocol Version 6

CoAP: Constrained Application Protocol

DTLS: Datagram Transport Layer Security

Tqdm: taqadum

RGB: Red Green Blue

IEEE: Institute of Electricals and Electronics Engineers

M2M: Machine to Machine

CPU: Central Processing Unit

DNS: Domain Name System

NAT: Network Address Translation

REST: REpresentational State Transfer

HTTP: Hipertext Transfer Protocol

URI: Uniform Resource Identifier

TCP: Transmission Control Protocol

UDP: User Datagram Protocol

ACK: Acknowledgement

TLS: Transport Layer Security

OMA: Open Mobile Alliance

LWM2M: Lightweigth M2M

IETF: Internet Engineering Task Force

RFID: Radio Frecuency Identification

PSF: Python Sotware Foundation

VGA: Video Graphics Array

PWM: Pulse-Width Modulation

API: Application Programming Interface

10. REFERENCIAS

[1] C. Ferrer, *"8.400 millones de dispositivos estarán conectados a Internet a finales de 2017", Blogthinkbig*, 18/2/2017. [En línea]. Disponible en: https://blogthinkbig.com/8-400-millones-de-dispositivos-estaran-conectados-a-internet-a-finales-de-2017

[2] J. Postel. *Internet Protocol*. RFC 791, 1981

[3] R. Hinden. *Internet Protocol, Version 6 (IPv6)*. RFC 2460, 1998.

[4] A. Noronha, R. Moriarty, K. O'Connell, N. Villa. *El valor de IoT: Como pasar de conectar cosas a obtener información*. Cisco, 2014.

[5] Efecom, *"La inversión en el Internet de las Cosas crecerá un 15% en 2017",* El Economista, 3/12/2016. [En línea]. Disponible en: http://www.eleconomista.es/empresas-finanzas/noticias/8003920/12/16/La-inversion-en-el-Internet-de-las-Cosas-crecera-un-15-en-2017.html

[6] *Reglamento (UE) 2016/679 del Parlamento Europeo y del Consejo*, UE 679-2016.

[7] *Directiva (UE) 2016/1148 del Parlamento Europeo y del Consejo*, UE 1148-2016.

[8] *Directiva 2013/40/UE del Parlamento Europeo y del Consejo*, UE 40-2013.

[9] *Directiva 2013/53/UE del Parlamento Europeo y del Consejo*, UE 53-2014.

[10] K. Rose, S. Eldridge, L. Chapin. *La Internet de las cosas*. Internet Society, 2015.

[11] D. Evans. *Internet of Things - La próxima evolución de Internet lo está cambiando todo*. Grupo de Soluciones para Internet (IBSG) de Cisco, 2011.

[12] D. McPherson. *RFC 7452 – Architectural Considerations in Smart Object Networking.* 2015.

[13] *ZigBee Specification*. ZigBee Standards Organization, 2012.

[14] R. Cobo. *Un premier sobre ZigBee.* Electro Industria. 2007.

[15] *IEEE Std 802.15.4 (Estándar IEEE para redes inalámbricas de baja velocidad),* 802.15.4 -2015.

[16] S. Farahani. *ZigBee Wireless Networks and Transceivers*. Newnes, 2008.

[17] J. Martín, D. Ruiz. *Protocolo ZigBee (IEE 802.15.4)*. Universidad de Alicante, Departamento de Tecnología Informática y Computación, 2007.

[18] Z. Shelby, K. Hartke, C. Bormann. *The Constrained Application Protocol (CoAP)* RFC 7252, 2014.

[19] P. Eronen, H. Tschofenig. *Pre-Shared Key Ciphersuites for Transport Layer Security (TLS)*. RFC 4279, 2005.

[20] N. Modadugu. *Datagram Tansport Layer Security Version 1.2*. RFC 6347, 2012.

[21] C. Hennebert, J.D. Santos, "*Security Protocols and Privacy Issues into 6LoWPAN Stack: A Synthesis*", IEEE Internet of Things Journal, vol. 1, no. 5, pp. 384-398, Oct.2014.

[22] G. Klas, F. Rodermund, Z. Shelby, S. Akhouri, J. Höller. *"Lightweight M2M": Enabling Device Management and Applications forthe Internet of Things.* Open Mobile Alliance, 2014.

[23] Hedda, *"Integración con Ikea Tradfri Gateway como puente para comunicar y controlar las luces, interruptores y sensores inteligentes basasdos en ZigBee de Ikea"* 27/3/2017. [En línea]. Disponible en: https://github.com/bwssytems/ha-bridge/issues/570

[24] E.Upton, G. Halfacree. *Raspberry Pi User Guide*. Wiley, 2012.

[25] Jr. F. L. Drake. *El tutorial de Python*. Python.org, 2009.

APÉNDICE 1. MANUAL DE INSTALACIÓN

En este apéndice se expone un manual de instalación para que el usuario sea capaz de descargar en su ordenador los programas para poder ejecutar a ambas aplicaciones.

Después de instalar previamente las bibliotecas *"libcoap"* y *"rainbowhat"* en el ordenador, se debe proceder a descargar el repositorio *"Tradfri-Ikea"* que contiene los software de ambos programas a través de la página de GitHub[29].

Para ello, se debe abrir un terminal en el sistema operativo Linux y escribir el comando:

COMANDO PARA DESCARGAR LOS PROGRAMAS DE AMBAS APLICACIONES
`sudo git clone --recursive https://github.com/arribas10/Tradfri`

Tabla 36: Comando para descargar el repositorio el cual contiene el software de ambas aplicaciones.

Al enviar este comando, se clonará el repositorio que hay en GitHub en su ordenador con lo que se ha debido de crear una carpeta llamada "Tradfri" en él. Para acceder a ella desde el terminal se utiliza el comando *"cd Tradfri"*.

Dentro de esta carpeta se encuentra una sub-carpeta con el nombre *"Tradfri-Ikea"* que contiene los dos scripts de Python para poder ejecutar las aplicaciones y un archivo de texto "Readme.txt" donde explica que el contenido de la carpeta y de los scripts.

En el interior de la carpeta *"Tradfri-Ikea"*, a la que se accede a partir del comando, "*cd Tradfri-Ikea"* se encuentran los softwares (*Mando_Remoto.py* y *Temporizador.py*) para que se utilicen si se quiere ejecutar la aplicación del *Mando Remoto* o el *Temporizador*.

Las aplicaciones no se pueden ejecutar a la vez, si el usuario quiere ejecutar la aplicación *Mando Remoto* debe de utilizar el siguiente comando:

COMANDO PARA EJECUTAR LA APLICACIÓN MANDO REMOTO
`python  Mando_Remoto.py`

Tabla 37: Comando para ejecutar la aplicación Mando Remoto

En caso contrario, si desea ejecutar la aplicación *Temporizador* debe de escribir el comando:

COMANDO PARA EJECUTAR LA APLICACIÓN TEMPORIZADOR
`python Temporizador.py`

Tabla 38: Comando para ejecutar la aplicación Temporizador

[29] Fuente vía - https://github.com/

APÉNDICE 2. MANUAL DE USUARIO

Este apartado es un manual de usuario donde se explica cómo se utiliza cada aplicación correctamente.

Aplicación 1: Mando Remoto

Cuando el usuario ejecuta la aplicación del *Mando Remoto*, la Raspberry Pi se enciende mostrando un mensaje inicial "*HOLA"*. A continuación, mostrará por los displays el mensaje: *"PULSE A PARA CONFIGURAR BOMBILLA 1*B CONFIGURA BOMBILLA 2*C CONFIGURA AMBAS"* y acto seguido, mostrará el mensaje *"ABoC"*. El usuario dependiendo la bombilla que desee configurar puede pulsar el botón A, B o si, por el contrario, quiere configurar las dos bombillas a la vez, deberá pulsar el botón C.

Ilustración 17: Mensaje para que el usuario seleccione la bombilla que desee.

Una vez seleccionado uno de los tres botones, el programa entrará en el menú para configurar la bombilla o bombillas escogidas.

El menú está formado por tres fases:

1. **Luz:** Es la primera fase del menú que el usuario se va a encontrar. El display mostrará el texto *"1.LUZ"* para indicar al usuario en qué fase está. Esta configuración sirve para que el usuario encienda o apague la bombilla mediante los botones B y C respectivamente. Si ya se ha ejecutado la acción que desee el usuario, para cambiar de fase y configurar otro parámetro, se debe pulsar el botón A.

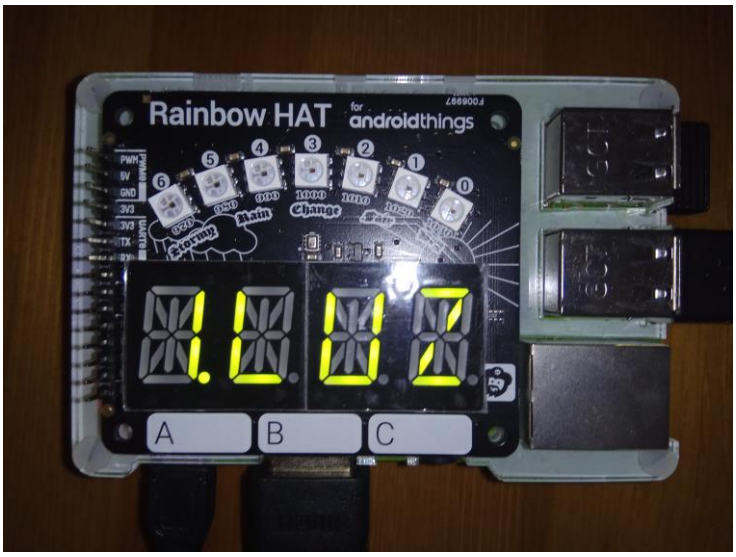

Ilustración 18: Mensaje que indica al usuario que está en la primera opción del menú.

2. **Intensidad:** El ajuste de la intensidad de las bombillas es la segunda configuración que se puede realizar en ellas. La Raspberry Pi al entrar en esta fase muestra el mensaje *"2.INT"*. La intensidad puede variar de 0% a 100%, en intervalos de 10 en 10. Para ello, el usuario con los botones B y C puede reducir o aumentar dicho nivel hasta ajustarse a sus necesidades.

Ilustración 19: Nivel de intensidad que tiene la bombilla/s.

3. **Color:** El color es la última acción que se puede configurar en las bombillas. Para pasar del ajuste de intensidad al color hay que pulsar el botón A. Una vez dentro de esta configuración, el hardware mostrará el mensaje *"3.COL"* en los cuatro displays que tiene disponible el Rainbow Hat. En esta sección, con el botón B el usuario puede variar el color de las bombillas entre tres opciones: *"Cold"*, *"Warm"* o *"Normal"*.

Ilustración 20: "Normal" es el color que se ha establecido en la bombilla/s

Una vez modificado el color que es el último ajuste que se puede realizar en las bombillas, el usuario tiene dos opciones:

La primera opción es pulsar el botón A, que llevaría al usuario nuevamente a la primera fase. Y la segunda opción, es pulsar el botón C, el programa volvería a empezar dando la opción al usuario de elegir una nueva bombilla para configurar.

Aplicación 2: Temporizador

El *Temporizador* es un programa para establecer como su propio nombre indica un temporizador para encender y apagar las bombillas en un intervalo de tiempo fijado.

Cuando el usuario ejecuta el programa, la aplicación comienza con un mensaje inicial *"HOLA"* dando la bienvenida al usuario. A continuación, le pide por pantalla a través del siguiente mensaje *"ELIGE LA BOMBILLA* A->BOMBILLA 1*B->BOMBILLA 2*C->AMBAS"*, que bombilla quiere utilizar para programar el temporizador. El usuario tiene que escoger a través de los botones del Rainbow Hat entre una de las tres opciones disponibles.

El siguiente paso es seleccionar el tipo de temporizador que el usuario desee programar. Para ello, los displays del Rainbow Hat mostrarán el siguiente mensaje: *"ELIGE ENTRE 3 OPCIONES* A TEMPORIZADOR DIARIO*B TEMPORIZADOR SEMANAL* C TEMPORIZADOR MENSUAL"*. Con los botones A, B o C, el usuario seleccionará el temporizador que prefiera.

Los tipos de temporizadores son:

- **Temporizador Diario:** Si el usuario pulsa el botón A seleccionará este tipo de temporizador para su configuración. El objetivo de este temporizador es establecer un intervalo de tiempo en un día determinado para que la bombilla o bombillas se enciendan y apaguen en dicho día. Una vez apagada finaliza el programa.

  Para establecer el temporizador hay que seguir una serie de pasos:

  1. **Día:** Es la primera tarea para establecer el temporizador. En este paso se debe de seleccionar el día que quiere el usuario programar las bombillas. En el Rainbow Hat se mostrará el mensaje *"D-1"* indicando el día que se va a seleccionar y mediante los botones B y C se puede aumentar o disminuir dicho número hasta elegir el día. Para guardar el valor y pasar al siguiente paso se debe pulsar el botón A.

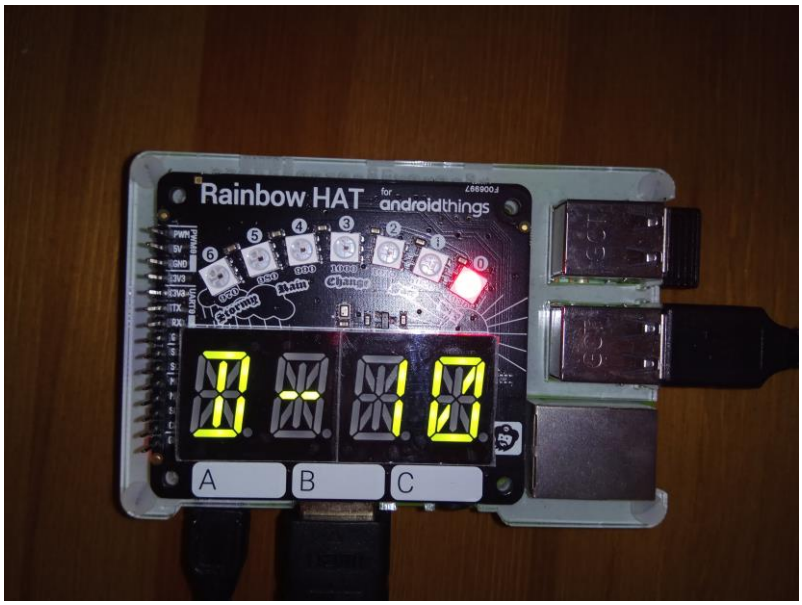

Ilustración 21: Elección del día que se quiere establecer el Temporizador.

2. **Mes:** Una vez elegido el día se debe seleccionar el mes en que se quiere comenzar el temporizador. En la interfaz, se muestra por defecto *"M- 1"* indicando que se va a seleccionar el mes uno a menos que el usuario no modifique dicho valor. Para modificarlo, se utilizan los botones B y C para aumentar o disminuir el valor. Para guardar el mes escogido por el usuario se presiona el botón A y acto seguido, se pasa al siguiente paso.

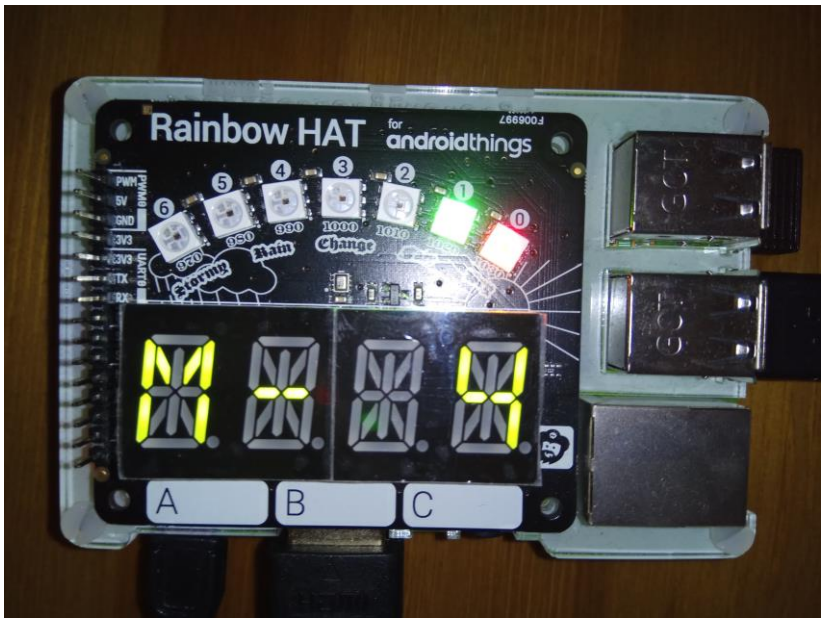

Ilustración 22: Elección del mes en que se quiere establecer el Temporizador

3. **Hora inicial:** Es el primer paso para establecer el intervalo de tiempo que estará la bombilla activada. El display mostrará el mensaje *"H- 1"* y el usuario deberá seleccionar la hora inicial a la que se quiere encender la bombilla en el día especificado. Para elegir la hora inicial se utiliza el botón C. Una vez que el usuario ha marcado la hora, para guardar el valor se debe pulsar el botón B y así entraremos en un sub-apartado para acabar de fijar la hora inicial.

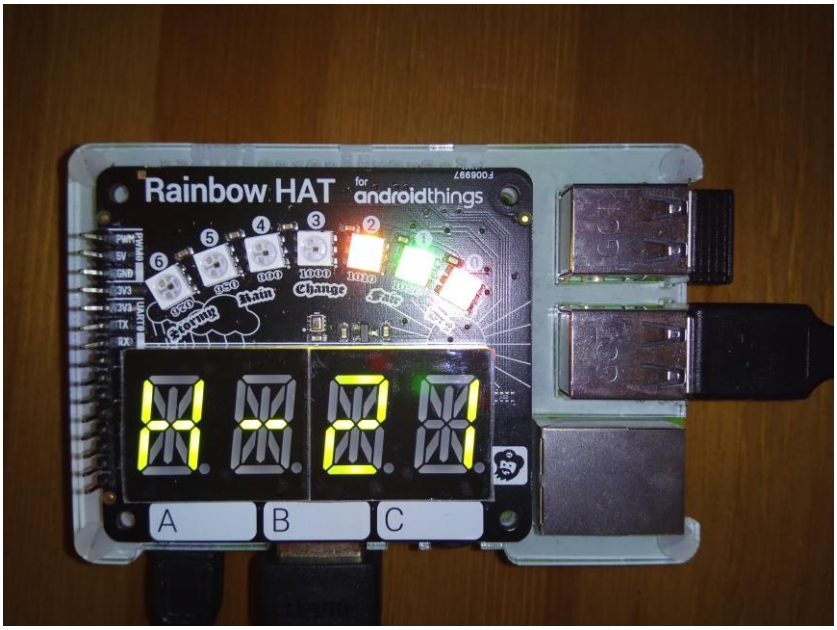

Ilustración 23: Elección de la hora inicial en que se quiere establecer el Temporizador.

4. **Minuto inicial:** El siguiente paso es establecer el minuto inicial. El display mostrará el mensaje *"M- 0"*. Para ello el usuario pulsará el botón C hasta marcar el minuto inicial que desee. Con cada pulsación en el botón C, los minutos incrementarán de diez en diez. Para guardar el valor y pasar al siguiente paso, se debe pulsar en este caso el botón A.

Ilustración 24: Elección del minuto inicial en que se quiere establecer el Temporizador.

5. **Hora final:** La siguiente etapa sigue el mismo procedimiento que con la hora inicial, pero en este caso, para establecer el fin del temporizador. El display mostrará el texto *"H- 1"* indicando la hora final a la que se va a apagar el temporizador, pero este valor el usuario lo puede modificar pulsando el botón C. Para guardar la hora escogida, se pulsa el botón B y además se pasará a la siguiente fase.

Ilustración 25: Elección de la hora final en que se quiere establecer el Temporizador.

6. **Minuto final:** El último parámetro que hay que establecer es el minuto final en que la bombilla/s se apagarán. Este valor se modifica igual que en el minuto inicial con el botón C y para guardar el valor escogido y activar definitivamente el temporizador se debe pulsar el botón A.

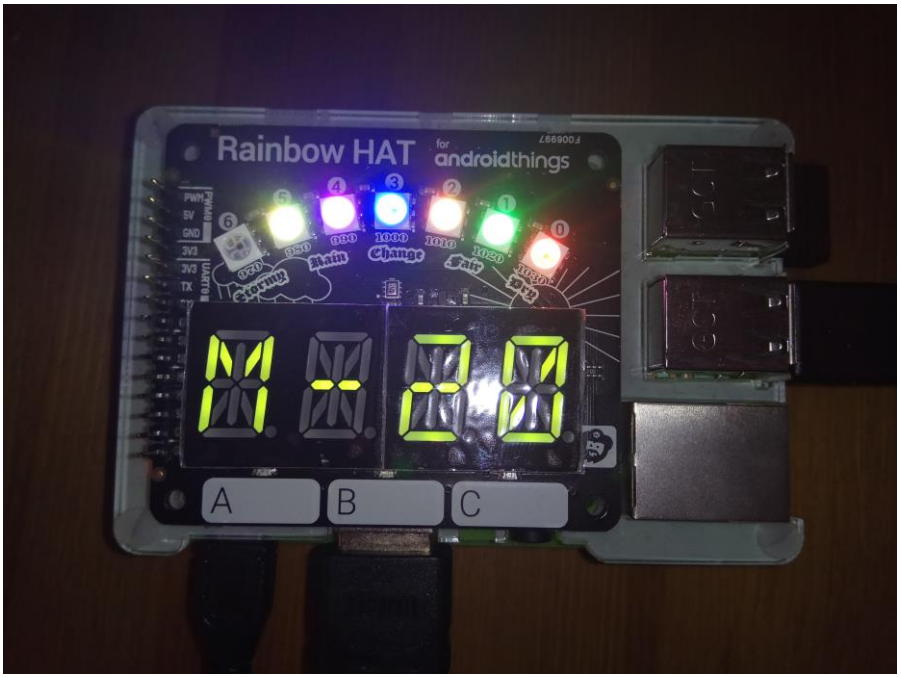

Ilustración 26: Elección del minuto inicial en que se quiere establecer el Temporizador.

7. **Temporizador activado:** Esta es la última etapa donde se muestran por el display del Rainbow Hat todos los datos introducidos por el usuario para establecer el temporizador. Al finalizar está representación de los valores, el temporizador se activará. La bombilla se encenderá cuando la hora, minuto, día y mes sea el mismo que lo escogido por el usuario. Cuando la hora y el minuto sean igual que la hora y minuto final establecidos por el usuario se apagará la bombilla y el programa finalizará.

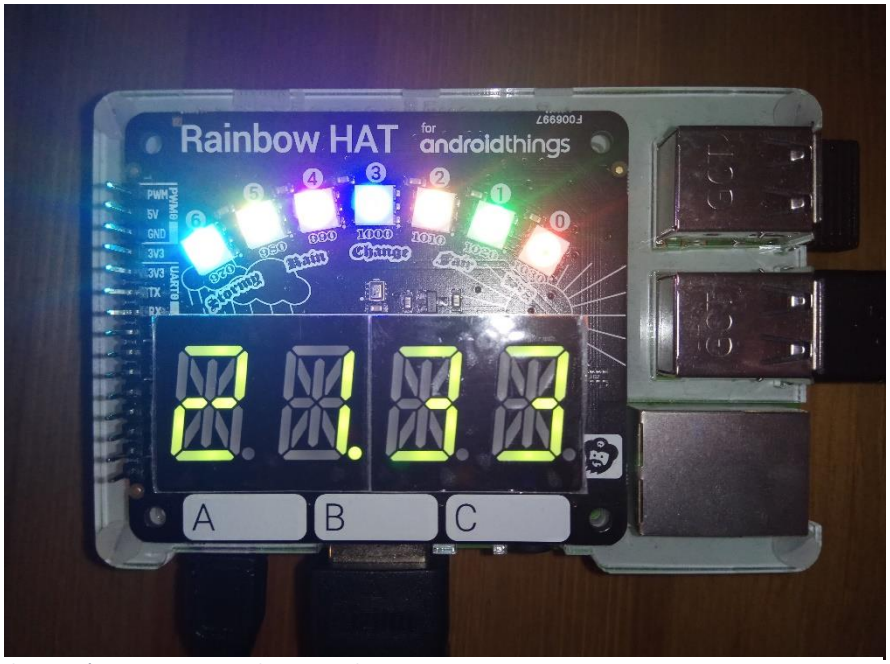

Ilustración 27: Temporizador activado.

- **Temporizador Semanal:** Si el usuario quiere programar un temporizador más de un día, este temporizador es una de las opciones que puede barajar.
  El objetivo es establecer el temporizador durante una semana considerando la misma desde el lunes a domingo. Este temporizador es útil en el caso de que el usuario esté fuera del hogar una semana, un fin de semana o un periodo de tiempo superior a un día e inferior a una semana.

  El temporizador semanal se establece siguiendo los mismos pasos que se llevan a cabo en el temporizador diario salvo por una condición. Esta condición se basa en que cuando se va a establecer en la primera etapa el día en que el usuario quiere fijar el inicio del temporizador el día seleccionado debe caer en lunes en el calendario. Si se elige otro día que no sea lunes al seleccionar el mes y verificarlo, el programa retorna hasta la primera etapa mostrando un error.

  El resto de los pasos siguen el mismo procedimiento que en el temporizador semanal.

- **Temporizador Mensual:** El temporizador mensual es la última opción que puede escoger el usuario para programar las bombillas. El objetivo de este temporizador es establecer un mes determinado por el usuario para encender y apagar la luz cada día durante el mes especificado. En el momento en que el mes se cumpla y se pase al siguiente el temporizador se apagará.

Los pasos que debe seguir el usuario para establecer el temporizador mensual son los mismos que en el temporizador diario salvo por una condición, en este caso no se establece el día puesto que las bombillas se van a programar en el mes que fije el usuario. Sabiendo esta modificación con respecto al temporizador diario, el usuario podrá establecer el temporizador mensual empezando por la etapa 2, es decir, eligiendo el mes que desee programar las bombillas.