Universidad de Valladolid

UNIVERSIDAD DE VALLADOLID

ESCUELA DE INGENIERIAS INDUSTRIALES

Grado en Ingeniería Electrónica Industrial y Automática

Control Domótico basado en el protocolo MQTT

Autor:

de la Cal Calleja, Javier

Tutor:

Mazaeda Echevarría, Rogelio
Ingeniería de Sistemas y Automática

Valladolid, julio de 2019.

AGRADECIMIENTOS

Me gustaría empezar mostrando mi agradecimiento a Rogelio Mazaeda, mi tutor en el presente trabajo, por su comprensión y ayuda a lo largo de este periodo académico. Sus consejos, reflexiones e indicaciones han sido indispensables para la realización de este.

Quiero hacer una mención especial a mi familia que siempre estuvo apoyándome y creyeron en mi en todo momento. Gracias a ellos he conseguido todo lo que tengo.

Para terminar agradecer todo el apoyo a Estela, compañeros y amigos que han hecho que todo esto fuera posible.

RESUMEN

En la sociedad actual el manejo de la información tiene una importancia muy significativa. La generación, comunicación, transformación y almacenamiento de los datos forma parte esencial de la vida moderna. En este sentido, conceptos como el de Internet de las Cosas (IoT) cobra una importancia notable en muchos aspectos y también en la domótica.

La existencia de múltiples dispositivos inteligentes en el ámbito del hogar hace de las aplicaciones domóticas un campo de prueba excelente para estos nuevos conceptos. En este trabajo nos proponemos el desarrollo de un termostato inteligente que utilice el protocolo MQTT para brindar una solución basada en eventos y escalable.

La solución brindada utiliza las ventajas del MQTT como protocolo flexible y sencillo que permite coordinar el funcionamiento de diferentes dispositivos inteligentes. Esta solución ofrece una interfaz con el usuario mediante una aplicación Android para teléfonos inteligentes.

PALABRAS CLAVE

MQTT, Domótica, Internet de las Cosas, WiFi, Control basado en eventos.

ABSTRACT

In today's society the use of information has a significant importance. Generation, communication, transformation and data storage are part of the modern life. On this matter, concepts such as the Internet of Things (IoT) have a great significance in many different aspects including home automation.

The existence of multiple smart devices in the home setting turns the home automation devices into an excellent testing ground field to these new concepts. This dissertation proposes the development of a Smart Thermostat that uses the MQTT protocol in order to give a scalable solution based on events.

The provided solution uses the advantages of the MQTT as a flexible and simple protocol that allows the coordination of the functioning of different smart devices. This solution offers an interface with the user by an application for Android smartphones.

KEYWORDS

MQTT, Domotics, Internet of Things, WiFi, Event-Based Control.

ÍNDICE DE CONTENIDO

ÍNDICE DE ILUSTRACIONES

ÍNDICE DE TABLAS

1. INTRODUCCIÓN

1.1. Antecedentes y justificación del proyecto

En los últimos años hay ciertos factores que están ganando importancia dentro de la sociedad tales como la información, la comunicación, la instantaneidad y la eficiencia, por poner algunos ejemplos. Gran parte de la culpa la tiene el avance de la tecnología, y en concreto las Tecnologías de la Información y la Comunicación (TIC). Es por eso por lo que el Internet de las Cosas (IoT) esté ganando protagonismo, y más aun dentro del concepto de domótica.

El internet de las cosas (IoT) es un paradigma basado en la interconexión de dispositivos, a los cuales se les proporciona una conexión a internet para que puedan interaccionar entre ellos. Estos dispositivos pueden ser sensores, actuadores, objetos cotidianos, etc. En definitiva, cualquier cosa podría adquirir una conexión a internet y establecer una comunicación que nos permita enviar y recibir información extrayendo un beneficio de ello[1].

El IoT puede ser aplicado en prácticamente infinidad de situaciones. Ejemplo de ello son la agricultura y ganadería, permitiendo una gestión más eficiente de cultivos y ganado, las ciudades y edificios, las cuales adquieren el nombre de *Smart Cities* y *Smart Buildings* respectivamente, pudiendo controlar los aparcamientos, tráfico y estado de la estructura entre otros, las casas con la domótica y sin olvidarnos de las aplicaciones industriales, donde el Internet Industrial de las Cosas (IIoT) junto con la industria 4.0 y el *Big Data* permiten a los usuarios tomar decisiones con una mayor seguridad y precisión.

La domótica es el conjunto de tecnologías que controlan y automatizan una casa, permitiendo una gestión eficiente de la energía y comunicación entre el sistema y las personas facilitando la vida de éstas[2]. Por lo que tiene sentido incluir el paradigma de IoT dentro de la domótica para aprovechar al máximo sus ventajas y llevarlas a un siguiente nivel. Si bien la domótica nos permite controlar nuestra casa de una manera eficiente, al dotarla de conexión a internet, el IoT nos permite obtener información y ejercer un control en tiempo real desde cualquier lugar que posea una conexión a internet.

Históricamente la tendencia ha sido hacia la creación de aplicaciones monolíticas de gran tamaño y complejidad, donde los diferentes módulos o tareas suelen estar fuertemente conectados. Este alto grado de conexión se hace evidente en el uso de las llamadas a funciones síncronas como mecanismo básico de interacción entre los diferentes módulos. Esto hace que los distintos elementos que conforman la aplicación estén integrados de una forma que es difícilmente modificable, dando por resultado soluciones poco escalables. Entendiendo el término escalable como la capacidad que poseen las soluciones de aumentar sus prestaciones, utilizando de forma eficiente de

más recursos de cómputo cuando éstos sean puestos a su disposición para hacer frente a demandas de trabajo o rendimiento mayores. Estas aplicaciones tradicionales monolíticas no suelen dar facilidades para la su reconfiguración dinámica, una vez que ya se han desplegado en los equipos finales de computo. Son aplicaciones frágiles, puesto que basta con que cambie un elemento no previsto para que todo el sistema se derrumbe. Sin embargo, las necesidades de los sistemas están cambiando, y ahora para aumentar la complejidad y a la vez la resiliencia, robustez y escalabilidad es importante cambiar de filosofía. Las aplicaciones de ahora tienen que ser no-bloqueantes a través de una comunicación asíncrona, utilizando tal vez, nuevas soluciones de control, como por ejemplo el control basado en eventos.

En la línea de las nuevas necesidades en las aplicaciones informáticas y gracias a que los avances tecnológicos lo permiten toma importancia el manifiesto de Sistemas Reactivos[3], una propuesta para satisfacer la nueva filosofía de los sistemas, que define cuatro aspectos clave que deben poseer éstos: responsividad, resiliencia, elasticidad y que sean orientados a mensajes. Estos sistemas tienen bajos tiempos de respuesta lo que permite que los errores se detecten rápidamente y al ser resilientes, pueden seguir siendo responsivos incluso después de suceder el fallo. Los fallos se aíslan en el componente en el que sucede permitiendo que si ocurre un fallo se pueda recuperar sin comprometer al sistema como un todo. Al ser elásticos son responsivos sin depender de la carga del sistema, adaptando los recursos para llevar a cabo la tarea. En cierto modo, un agente escalable es a su vez elástico, ya que al crecer sus prestaciones adapta los recursos para hacer frente a esa mayor demanda. Son también orientados a mensajes, siendo estos asíncronos, lo que permite que haya un bajo acoplamiento y que la comunicación sea no-bloqueante, es decir, los dispositivos consumen los recursos solo cuando éstos están activos.

Tiempo atrás, las aplicaciones eran frágiles, síncronas y estaban fuertemente acopladas lo que derivaba en sistemas de gran complejidad. Esto era debido a que dichas aplicaciones eran construidas en base a la tecnología disponible, donde la comunicación entre los distintos elementos o agentes era cableada y requería diseñar muy bien de antemano los sistemas y prever las posibles modificaciones o ampliaciones de este, y en consecuencia las comunicaciones son rígidas. Sin embargo, con el avance de la tecnología y con ello la mejora de la comunicación inalámbrica y la aparición del WiFi, se consigue una menor rigidez en el ámbito físico. Además, permite una mayor dispersión geográfica de los dispositivos y poder conectar elementos que hace años sería impensable, es decir, dotar de conexión a internet a las cosas (IoT), teniendo una mayor posibilidad de interacción entre agentes, obteniendo una mayor escalabilidad y menor acoplamiento. Es por todo esto que es necesario que existan protocolos de comunicación que se adapten a las nuevas exigencias.

En este contexto han surgido una serie de protocolos de comunicación que dan soporte a las necesidades anteriormente descritas y encajen en el paradigma del IoT, tales como *Advanced Message Queuing Protocol* (AMQP), *Constrained Application Protocol* (CoAP), *Data Distribution Service* (DDS), *Message Queue Telemetry Transport* (MQTT) entre otros. En la Tabla 1 se pueden apreciar las principales diferencias entre los mencionados protocolos, cortesía del instituto nacional de ciberseguridad[4].

	Transporte	Modelo	Ámbito de aplicación	Conocimiento del contenido	Datos principales	Seguridad	Prioridad de los datos	Tolerancia a fallos
AMQP	TCP/IP	Intercambio de mensajes punto a punto	D2D D2C C2C	Ninguno	Codificados	TLS	Ninguno	Específica de la implementación
CoAP	UDP/IP	Petición/Respuesta (REST)	D2D	Ninguno	Codificados	DTLS	Ninguno	Descentralizado
DDS	UDP/IP (unicast + mcast) TCP/IP	Publicación/Suscripción Petición/Respuesta	D2D D2C C2C	Enrutamiento basado en el contenido, consultas	Declarados codificados	TLS, DTLS, DDS	Prioridades de transporte	Descentralizado
MQTT	TCP/IP	Publicación/Suscripción	D2C	Ninguno	No definidos	TLS	Ninguno	El nodo central (broker) es el punto único de fallo (SPoF)

Tabla 1. Comparación de protocolos IoT.

MQTT es un protocolo de mensajería asíncrona que disocia al emisor y al receptor tanto en espacio como en tiempo, y a pesar de que su nombre diga "*queue*" no tiene nada que ver con colas de mensajería que tradicionalmente se conocen en informática, aunque sí que es posible generar colas de mensajes con otros mecanismos que se verán más adelante en profundidad. Utiliza un modelo de publicación y suscripción, el cual es no-bloqueante, lo que permite que no sea necesaria una red muy fiable. Es un protocolo liviano y flexible, lo que hace que pueda implementarse en dispositivos con recursos limitados y se adapte a situaciones con diferente demanda de recursos. La comunicación es posible gracias a un intermediario de mensajes entre los distintos clientes, que puede ser cualquier cosa capaz de enviar o recibir mensajes. El cliente publica un mensaje en un tema o *topic*, que se lo envía al intermediario, o también llamado *broker*, el cual redirige este mensaje al cliente o clientes que estén suscritos a ese *topic*. Puesto que los mensajes están organizados por temas proporciona una jerarquía a la estructura del sistema.[5]

Se aprecian concordancias entre el protocolo MQTT y el manifiesto de los Sistemas Reactivos, y a su vez dicho protocolo encaja muy bien con el paradigma del IoT. Es por esto por lo que es el elegido para el desarrollo del presente trabajo. Igualmente, el protocolo MQTT está cogiendo peso últimamente como se puede observar en la Ilustración 1, creciendo su uso frente a otros protocolos.

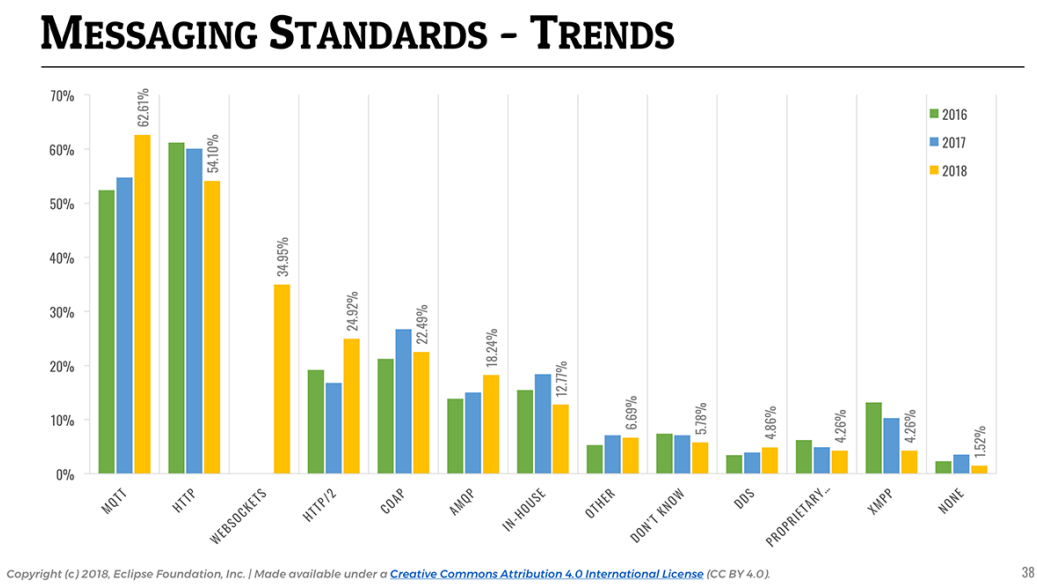

Ilustración 1. Tendencias de uso de protocolos de IoT.

Es por lo mencionado en el presente apartado, que se va a desarrollar un dispositivo domótico basado en IoT, siguiendo el manifiesto de los Sistemas Reactivos y utilizando como protocolo de comunicación: MQTT.

1.2. Objetivos del proyecto

El propósito del presente proyecto es el desarrollo de un termostato inteligente como aplicación del protocolo *Message Queue Telemetry Transport* (MQTT) en un dispositivo domótico y la utilización de conceptos relacionados con el IoT. Se pretende construir una red de comunicación Machine-to-Machine (M2M) entre un sensor de temperatura, un actuador, que accione la caldera, y un control principal, que gestione las funcionalidades del dispositivo, permitiendo la interacción mediante una aplicación móvil implementada en Android y una página web para la configuración del sensor y actuador.

Se trabajará con distintas tecnologías y lenguajes de programación para demostrar su flexibilidad y aplicación en dispositivos con recursos limitados, viendo cómo encaja dicho protocolo en el paradigma del IoT y el manifiesto de los Sistemas Reactivos y sigue la línea de los avances de la tecnología actualmente cumpliendo con la demanda de los usuarios en materia de eficiencia y confort.

Se tiene el objetivo de plantar las bases de un sistema domótico que tenga la capacidad de crecer con el tiempo, añadiéndole otros sensores y brindando más funcionalidades que satisfagan las necesidades cotidianas de una persona, para hacerle la vida más fácil. En resumen, que sea escalable.

1.3. Descripción del sistema

El dispositivo domótico llevará a cabo el control de temperatura de una casa, como prototipo de un producto comercial y lo que esto conlleva: generalidad, configuración fácil, escalabilidad, etc. No será una solución *ad hoc* a un problema determinado con unas características marcadas, sino que tendrá una configurabilidad y podrá ser instalado en cualquier casa.

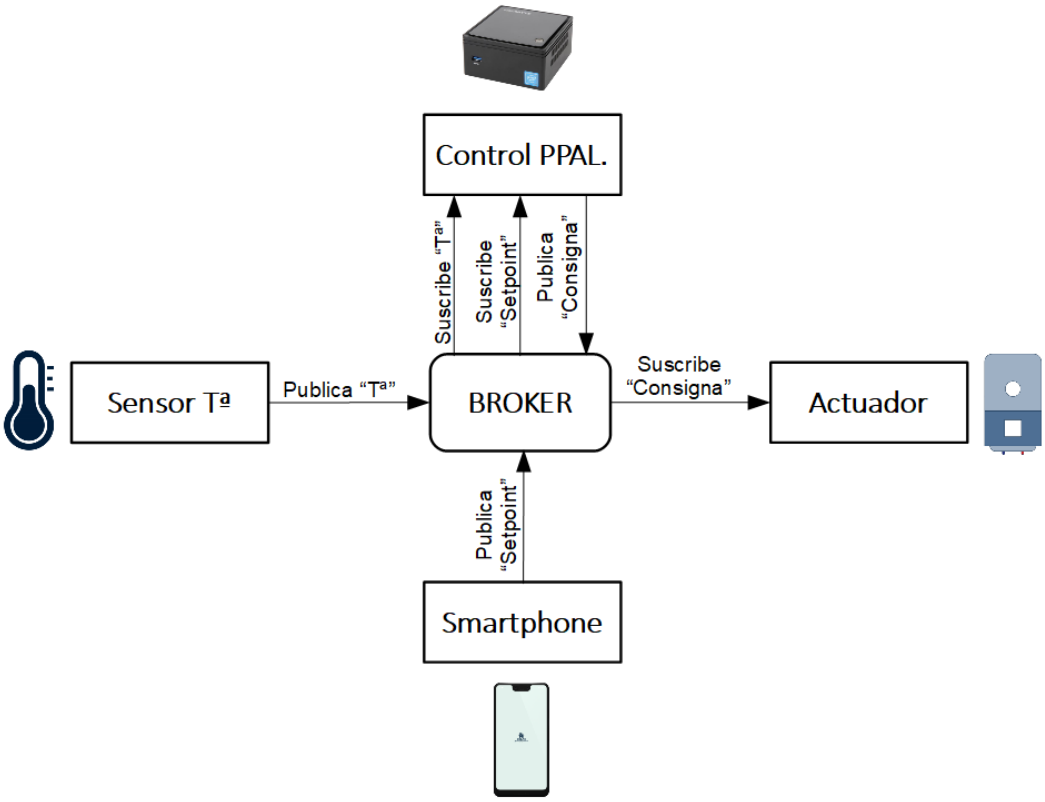

Ilustración 2. Estructura básica de funcionamiento del termostato.

Se plantearán las bases para que el producto sea escalable y se le puedan añadir más funcionalidades según avance el tiempo. Para ello el sistema constará de un controlador central que será el encargado de supeditar el resto de los dispositivos, de tal forma que estos dispositivos no sabrán de la existencia del resto, es decir estarán desacoplados y se comunicarán asíncronamente.

Para conseguir esto, los dispositivos se conectarán a internet a través de WiFi y utilizarán el protocolo MQTT para comunicarse e intercambiar información. Gracias a que MQTT se basa en el modelo de publicación y suscripción, se consigue un intercambio de mensajes clasificados por temáticas. Existen agentes que publican información sobre esas temáticas acordadas previamente y otros agentes interesados en dichas temáticas se suscriben a ellas. Esto se consigue gracias a que, a modo de enlace entre el agente que publica y el agente que se suscribe, existe un intermediario que filtra los

mensajes en función de las temáticas y redirige los mensajes del publicador al suscriptor de una forma dinámica, elástica y proporcionando un bajo acoplamiento entre dichos agentes.

En la Ilustración 2 podemos ver la estructura básica de funcionamiento del termostato, en la cual aparecen los elementos principales del sistema domótico, tales como un controlador, un sensor de temperatura, un actuador y un teléfono inteligente, que publican mensajes o se suscriben a temáticas y tienen un intermediario que filtra dichos mensajes en función de los temas, el *broker*.

El sensor de temperatura es el encargado de medir y enviar la temperatura al control principal, que será el encargado de comparar esa temperatura con el valor de referencia (*setpoint),* que ha fijado el usuario, y en función de esa comparación decide si enciende o apaga la caldera enviándole una consigna al actuador.

Se prevé una interfaz con el usuario a través de una aplicación en un teléfono móvil ya que la gran mayoría de la población posee uno. Nos servirá para la configuración y control del sistema en tiempo real.

Para el sensor de temperatura y el actuador se utilizarán microcontroladores que tengan la capacidad de conectarse a una red WiFi, ya que no consumirán muchos recursos y necesitan la conexión WiFi para acceder a internet. Estos dispositivos son capaces de implementar los clientes de esos dos elementos y conectarse a internet.

Estos dispositivos necesitan las credenciales de la red WiFi de casa, para facilitárselos se prevé de una interfaz de configuración para que pueda introducir dichos parámetros. De esta forma se le da generalidad y configurabilidad al producto.

El controlador central albergará el control principal y el intermediario o *broker*, aunque se discutirá en líneas futuras las posibilidades de dónde ubicar a este último. El control principal será el único que sepa de la existencia del resto de elementos y sabrá qué hacer en cada caso. Será el encargado de gestionar todas las funcionalidades del sistema: calendario de funcionamiento semanal, modos de funcionamiento, etc.

1.4. Estructura de la memoria

Con el fin de facilitar la comprensión del presente documento se van a explicar brevemente los distintos apartados en los que se divide la memoria.

- INTRODUCCION: se enmarca y justifica el proyecto en el contexto en el que se va a desarrollar el trabajo, marcando los objetivos esenciales y una descripción del sistema que se va a construir.

- ASPECTOS PREVIOS: se describe el marco teórico de las tecnologías, técnicas y elementos que toman protagonismo en la elaboración del proyecto.

- SOLUCIÓN DOMÓTICA: se describe la solución que se va a desarrollar, como funciona, que se quiere conseguir, etc.

- DECISIONES DE IMPLEMENTACIÓN: se explica cómo se ha implementado la solución que se ha elegido para este proyecto.

- LINEAS FUTURAS: se plantearán las líneas a seguir en un futuro, como enfocarlo para poder desarrollarlo a lo largo del tiempo. En resumidas cuentas, lo que se pretende conseguir cuando el sistema esté completo.

- CONCLUSIONES: se exponen las ideas alcanzadas tras la realización del proyecto.

- BIBLIOGRAFÍA: se enumeran las fuentes bibliográficas consultadas para desarrollar el proyecto.

- ANEXOS: contiene el código de todos los programas que se han implementado para llevar a cabo el proyecto.

2. ASPECTOS PREVIOS

2.1. MQTT

2.1.1. Introducción y un poco de historia

Message Queue Telemetry Transport (MQTT) es un protocolo de mensajería asíncrona usado para la comunicación machine-to-machine (M2M) en el paradigma del internet de las cosas (IoT). Pese a lo que dice su nombre, no tiene nada que ver con las colas de mensajes y es un protocolo basado en un modelo de publicación y suscripción, muy simple, ligero y flexible, perfecto para dispositivos con recursos limitados.

Fue inventado por Dr. Andy Stanford-Clark de IBM y Arlen Nipper de Arcom (ahora Eurotech), a finales de los 90 cuando querían conectar sensores de los oleoductos a través de satélites y necesitaban un protocolo que consumiera poca batería y funcionará en un ancho de banda pequeño. Desde el año 2014 forma parte de OASIS, un consorcio internacional que vela por el desarrollo de estándares, convirtiéndose en un estándar abierto del mismo.

2.1.2. Porque MQTT y no otros protocolos

MQTT es un protocolo simple, liviano y flexible que permite implementarse en dispositivos con recursos limitados y conectarse a redes con ancho de banda limitada o alta latencia. Su flexibilidad hace que se comporte bien ante diferentes escenarios o demanda de recursos.

Esto encaja perfectamente con el paradigma IoT, puesto que la mayoría de los objetos que se dotan de conexión a internet poseen recursos limitados y las redes a las que se conectan pueden fallar, o tener latencias altas.

Otros protocolos famosos como HTTP o AMQP no han triunfado en el IoT por diversas razones. En el caso de HTTP es un protocolo cliente-servidor unidireccional, el cliente hace una petición y el servidor le responde, y puesto que los dispositivos IoT son los clientes no pueden recibir información de forma pasiva. En redes con latencia o congestionadas este protocolo tendría problemas puesto que es pesado y es sincronizado. Si hablamos de AMQP, en gran medida podría encajar con las características del IoT puesto que es asíncrono, sin embargo, está diseñado para sistemas de alto rendimiento donde los recursos no son limitados y las redes suelen ser fiables.

2.1.3. El modelo publicación/suscripción

El modelo publicación y suscripción permite que MQTT sea no-bloqueante, desacoplando al emisor y receptor. El emisor es el que publica los mensajes sobre un tema o *topic* y el receptor es el que está suscrito a cierto *topic* y por consiguiente recibe los mensajes. Esto es posible gracias a un intermediario, llamado "*bróker*" que filtra los mensajes que le llegan de los "*publicadores*" y los reparte a los "*suscriptores*" según el *topic*.

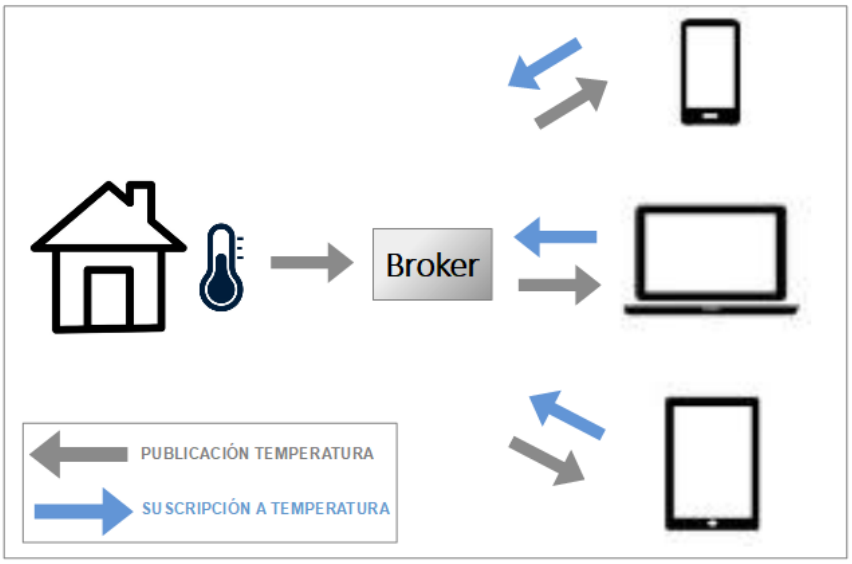

Ilustración 3. Ejemplo de modelo publicación y suscripción.

El desacople entre emisor y receptor se produce en tres dimensiones:

- Tiempo, ya que el cliente que publica y el que está suscrito no necesariamente deben estar funcionando al mismo tiempo.
- Espacio, es decir, el publicador y el suscriptor no necesitan conocerse entre ellos. Únicamente deben conocer la dirección IP del bróker y el *topic* de suscripción.
- Sincronización, puesto que las acciones que transcurren en los clientes no se ven afectadas al publicar o recibir un mensaje de un *topic* suscrito.

Otro factor importante de este modelo es la escalabilidad, ya que el bróker puede procesar las operaciones por eventos y realizar varias operaciones en paralelo. Esta característica es apoyada por la estructura de *topics* jerarquizada que se verá en un apartado más adelante.

2.1.4. Cliente MQTT

Cuando en el apartado anterior hablábamos de publicadores y suscriptores, básicamente estábamos hablando de clientes que tenían la función de publicar mensajes o estar suscritos, pero además un cliente puede ser publicador y suscriptor a al mismo tiempo.

Un cliente MQTT puede ser cualquier dispositivo que se conecte a un bróker MQTT a través de internet, desde uno con recursos limitados como puede ser un microcontrolador hasta un ordenador que cumpla sobradamente.

La principal ventaja de los clientes MQTT es que es muy sencillo implementarlos, permitiendo su uso en dispositivos con pocos recursos. Además, las librerías de clientes MQTT tienen una amplia variedad de lenguajes

para su uso (C++, Java, Arduino, etc.), lo que permite que dicho protocolo sea muy flexible.

2.1.5. Broker MQTT

El bróker es la pieza clave del modelo de publicación y suscripción, al cual se conectan los clientes para enviar y recibir los mensajes. Es el intermediario encargado de recibir los mensajes, filtrarlos en base al *topic*, y despacharlos a los clientes que estén suscritos a dicho *topic*, y además tiene la responsabilidad de autenticar a los clientes que tratan de conectarse a él.

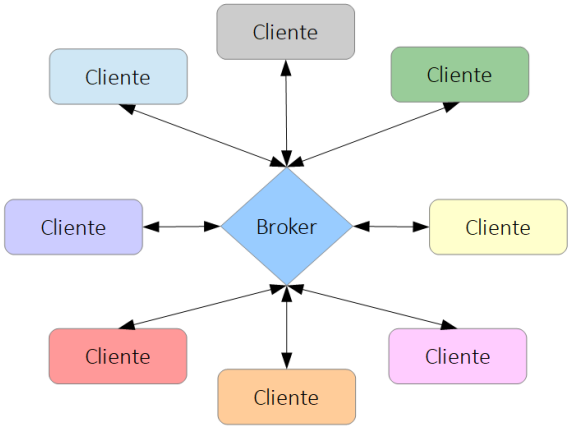

Ilustración 4. Estructura en estrella del protocolo MQTT.

Como se ha comentado anteriormente este protocolo disocia al emisor y receptor en tiempo, y para ello el bróker tiene la posibilidad de almacenar mensajes cuando el suscriptor no esté disponible y poder entregarlos cuando vuelva a entregarlos, permitiendo una calidad de servicio (QoS).

2.1.6. Establecimiento de la conexión

MQTT está basado en TCP/IP y esta implementado en las capas superiores del modelo OSI de comunicación, como se puede ver en la Ilustración 5.

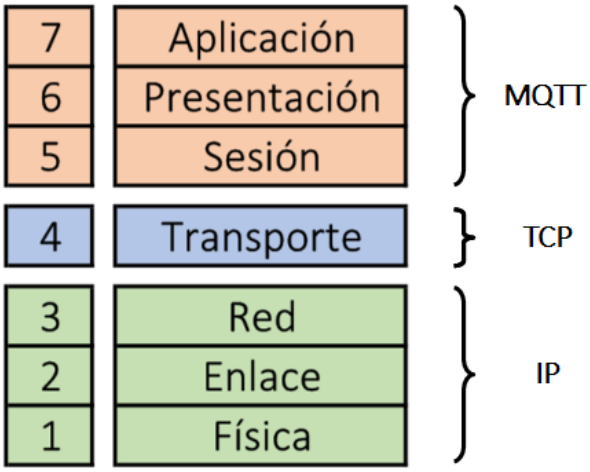

Ilustración 5. MQTT en el modelo OSI.

Este protocolo proporciona un desacople en el espacio, es decir, que el emisor y receptor no saben de la existencia del otro, esto sucede porque la conexión se establece siempre de cliente a bróker y no de cliente a cliente. Para ello el cliente inicia la comunicación enviándole un mensaje de conexión (CONNECT) al bróker y el éste le responde con un mensaje que le informa del estado de la conexión (CONNACK).

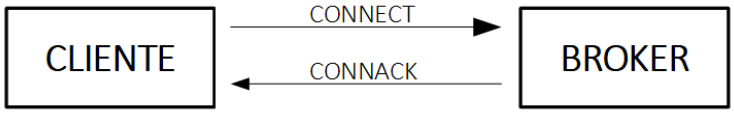

Ilustración 6. Establecimiento de la conexión MQTT.

El mensaje CONNECT contiene un id del cliente a modo de identificador para el bróker y le indica al bróker si va a ser una sesión persistente o no. En caso de serla, el bróker guarda las suscripciones y los mensajes perdidos de las suscripciones con QoS. En el caso de que se necesite autenticación por parte del broker, el mensaje contiene el usuario y la contraseña, sin embargo, estos datos son texto plano y no contienen ningún tipo de cifrado, lo que lo hacen inseguro. Para solucionar esto se recomienda el uso de TLS o certificados SSL.

El mensaje de CONNACK responde con un código (*Tabla 2*) que indica el estado de la conexión e informa si ese cliente ya había iniciado una sesión permanente o no.

Código devuelto	Significado
0	Conexión aceptada.
1	Conexión rechazada, versión de protocolo erróneo.
2	Conexión rechazada, identificador rechazado.
3	Conexión rechazada, servidor no disponible.
4	Conexión rechazada, usuario o contraseña incorrectos.
5	Conexión rechazada, no autorizado.

Tabla 2. Código estado de la conexión MQTT (CONNACK).

2.1.7. Topic

El *topic* o tema es el identificador que le sirve al bróker para filtrar y distribuir los mensajes a los clientes que estén suscritos a él, es decir, el bróker cuando recibe un mensaje de un cliente (publicador) coteja el *topic* del mensaje entrante con los *topic* de los clientes (suscritores) que se han suscrito a algún *topic* y redirige el mensaje a los clientes suscritos a dicho *topic*.

Los *topic* deben tener como mínimo un carácter, distinguen entre minúsculas y mayúsculas, y pueden contener espacios, aunque no se recomienda usarlos.

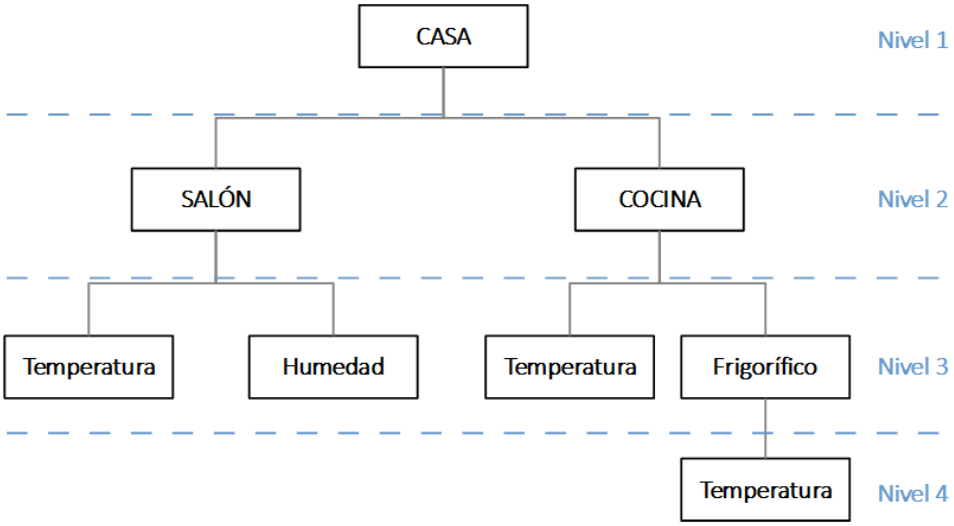

Ilustración 7. Ejemplo de estructura de topics.

Una cosa muy interesante es la posibilidad de jerarquizar los *topic*, separándolos con una barra (‘/’), lo que permite crear una estructura de *topic* escalable, permitiendo añadir o eliminar elementos de la red de comunicación sin afectar a los otros elementos. Como ejemplo de ello, se puede ver en la Ilustración 6 una estructura correspondiente a una casa que se miden distintos parámetros del salón y cocina, de la que resultan los siguientes *topics*:

- ‘CASA/SALÓN/Temperatura’
- ‘CASA/SALÓN/Humedad’
- ‘CASA/COCINA/Temperatura’
- ‘CASA/COCINA/Frigorífico/Temperatura’

A la hora de suscribirse existen dos comodines para sustituir un nivel de jerarquía o varios, según se desee. Estos comodines solo pueden ser usados por los suscriptores y no por los clientes que publican mensajes en un *topic*, el cual debe ser conciso.

El símbolo ‘+’ es un comodín que sustituye un nivel de la jerarquía dentro de la estructura creada, y recibiría todos los mensajes que engloben el comodín. Por ejemplo: ‘CASA/+/Temperatura’, sustituye a ‘CASA/SALÓN/Temperatura’ y ‘CASA/COCINA/Temperatura’ y recibiría sendos datos. Sin embargo, no sustituye a ‘CASA/COCINA/Frigorífico/Temperatura’ puesto que tiene un nivel más (‘Frigorífico’) antes de ‘Temperatura’.

El símbolo ‘#’es el comodín que sustituye varios niveles de la jerarquía. Solo puede ir ubicado al final del *topic*, lo que significa que los niveles que sustituye siempre serán los inferiores. Por ejemplo: ‘CASA/#’ coincidirá con todos los *topics*. ‘CASA/COCINA/#’, coincidirá con los *topic* ‘CASA/COCINA/Temperatura’ y ‘CASA/COCINA/Frigorífico/Temperatura’.

Cabe mencionar que el bróker posee una jerarquía de *topic* para las estadísticas internas del mismo. Ejemplo de estos son:

- '$SIS/broker/clients/connected'
- '$SIS/broker/messages/sent'
- '$SIS/broker/version'

2.1.8. Calidad de servicio (QoS)

La calidad del servicio o Quality of Service (QoS) es la garantía de entrega de los mensajes, es decir, QoS define como se va a intentar de asegurar el cliente o el bróker de que se ha recibido el mensaje. Esto es útil puesto que nos permite dar fiabilidad a la entrega de mensajes sin preocuparnos en si la red es fiable para los clientes que se hayan conectado al menos una vez al broker.

MQTT fija tres niveles de QoS. Cuanto mayor es el nivel de QoS más alta es la fiabilidad, pero implica mayor latencia y necesita mayor ancho de banda.

El nivel más bajo es el 'QoS 0' también llamado *"A lo sumo una vez"*. Con este nivel el mensaje se envía una vez y no se recibe confirmación de entrega, por lo que también se le conoce como *"fire and forget"*, dispara y olvida. Es el nivel que proporciona un menor esfuerzo en la entrega y el que necesita menos recursos.

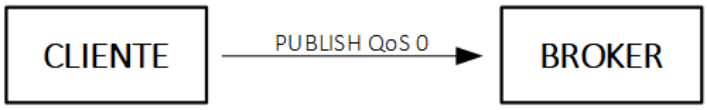

Ilustración 8. Publicación con QoS 0 – "A lo sumo una vez".

Un nivel más arriba está el 'QoS 1' también conocido como *"Al menos una vez"*. En este nivel el mensaje enviado tiene la garantía de que será entregado al receptor, para ello el emisor enviará el mensaje tantas veces como sea necesario hasta recibir una respuesta con la confirmación de la entrega (PUBACK). Con este QoS las comunicaciones se ralentizan y se incrementan los recursos usados.

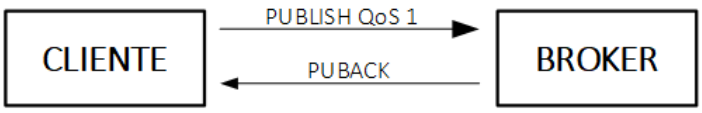

Ilustración 9. Publicación con QoS 1 – "Al menos una vez".

En último lugar se encuentra el 'QoS 2' que se le conocido como *"Exactamente una vez"*. Es el nivel más alto y asegura que el mensaje se entregue una sola vez. Para ello hace uso de un par de confirmaciones entre el emisor y el receptor del mensaje. Esto aumenta la sobrecarga y disminuye el rendimiento del sistema.

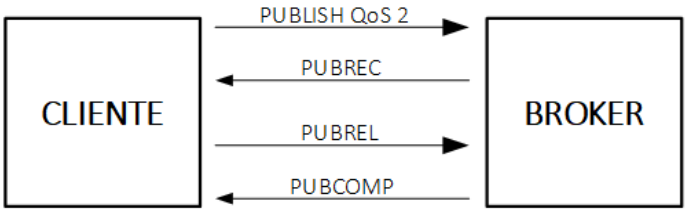

Ilustración 10. Publicación con QoS 2 – "Exactamente una vez".

Cabe destacar que existe una degradación entre el publicador y el suscriptor si tienen distinto QoS. Como el publicador envía un mensaje con un QoS y el suscritor se suscribe con otro QoS, que puede ser igual o distinto que el del publicador, quiere decir que el cliente está eligiendo el máximo QoS con el que se enviará un mensaje o se recibirá. Por ejemplo, si un cliente publica con 'QoS 2' y otro cliente está suscrito con 'QoS 0', éste recibirá el mensaje con 'QoS 0', pero si un segundo cliente está suscrito al mismo *topic* con 'QoS 2', entonces lo recibirá con 'QoS 2'. Por otro lado, si un cliente publica con 'QoS 2' y otro cliente está suscrito con 'QoS 0', éste lo recibirá con 'QoS 0'.

2.1.9. Mensajes retenidos

Esta funcionalidad permite que el bróker almacene o retenga el último mensaje publicado en un *topic*, incluso después de haberlo enviado a todos los clientes conectados. Si se publica un mensaje en un *topic* con esta función activada, el bróker guardará este mensaje y cuando un cliente se suscriba a dicho *topic* el bróker le enviará el mensaje que tiene almacenado.

Publicar un mensaje como "mensaje retenido" tiene sentido cuando un nuevo cliente se conecta y quiere obtener la información de forma inmediata, sin tener que esperar a que el emisor publique de nuevo un mensaje.

Se podría decir que estos mensajes son el *"last known good"*, es decir, el último dato con valor, que no tiene que ser el valor actual, pero si el último con importancia. Por ejemplo, si un sensor de humedad en una casa publica cada cierto tiempo el valor medido con un mensaje retenido sobre un *topic*, aunque el valor de la humedad este variando entre publicaciones, el valor importante es el publicado, el que el bróker retiene. Y cuando un cliente nuevo quiera obtener esa información no tendrá que esperar hasta que vuelva a publicar un dato y recibirá el último dato publicado, el almacenado en el bróker.

2.1.10. Otros datos de interés

Como ya se ha comentado brevemente en el apartado "*2.1.6. Establecimiento de la conexión"*, en la conexión del cliente puede establecer si la sesión va a ser limpia o persistente. En caso de que el cliente posea pocos recursos o desee obtener los mensajes incluso después de periodos de desconexión, éste establecerá una sesión persistente para que el bróker almacene las suscripciones y cree una cola de mensajes donde guardará los mensajes

mientras el cliente esta desconectado y se los entregará cuando se vuelva a conectar. En caso de ser un cliente que no necesite que los mensajes sean almacenados establecerá una sesión limpia, y el bróker no almacenará nada.

Otro aspecto interesante que ofrece el protocolo MQTT es el *"Last Will and Testament"* (LWT), una funcionalidad que proporciona un aviso cuando ocurre una desconexión inesperada de un cliente. Esto se configura con el mensaje CONNECT en el establecimiento de la conexión y se fija un *topic*, un mensaje, el QoS y si va a ser un mensaje retenido. Todo esto lo guarda la bróker una vez establecida la conexión del cliente, y cuando se desconecte de forma imprevista el bróker publicara el mensaje LWT en el *topic* indicado y con las características especificadas.

2.2. MOSQUITTO

Mosquitto[6] es un bróker muy ligero de código abierto, escrito en C, que implementa las últimas versiones del protocolo MQTT, es muy flexible ya que permite su implementación desde un dispositivo con recursos limitados hasta un ordenador con recursos de sobra.

3. SOLUCIÓN DOMÓTICA

3.1. Introducción

El presente trabajo se centra en el desarrollo de un dispositivo domótico acorde con el paradigma IoT, de acuerdo con el manifiesto de Sistemas Reactivos y usando MQTT como protocolo de comunicación. Se le ha querido dar un enfoque de producto comercial e ir más allá del sentido meramente académico, buscándole significado e intencionalidad al proyecto que se va a desarrollar. Es por eso por lo que se partirá de un concepto global de lo que se quiere conseguir y se irán tomando decisiones para definir el producto de una forma justificada.

Un producto domótico comercial debe formar un sistema domótico completo para facilitar la vida de las personas, ayudándolas en tareas cotidianas y ofreciéndolas un confort, brindándoles un mayor control sobre el funcionamiento de la casa y tratando de lograrlo sin perder de vista la eficiencia de recursos energéticos.

Para dotar de versatilidad al sistema es importante que sea escalable y permita añadir o quitar elementos con facilidad, como pueden ser más sensores o actuadores. Igualmente debe ser sencillo su configuración y uso para poder llegar a un público más amplio.

El desarrollo del producto se apoyará en los pilares del manifiesto de Sistemas Reactivos: responsivo, resiliente, elástico y orientado a mensajes.

Otro aspecto importante es el precio, el cual debe ser razonable y rentable para que el cliente quede satisfecho con la compra.

3.2. Visión general

Se tiene como propósito el desarrollo de un sistema domótico que controle distintos apartados de una casa. Este sistema ofrecerá la posibilidad de controlar un aspecto o varios, cumpliendo con el concepto de escalabilidad. El sistema permitirá añadir y eliminar elementos de una forma sencilla sin alterar el funcionamiento global del mismo, porque son elementos levemente acoplados (*loose-coupled* en la terminología anglosajona), o lo que es lo mismo, trabajarán de una forma asíncrona entre ellos. El usuario tendrá la posibilidad de controlar la calefacción de su casa en función de la temperatura de una sala o varias, añadir otros sensores, como por ejemplo el de humedad, controlar las persianas, luces, entre otras posibilidades.

Cuando el usuario opta por adquirir cualquier tipo de solución domótica pretende hacer su vida más fácil y cómoda, brindándole una mayor sensación de control. Ya sea porque dicho elemento le va a ayudar con la toma de decisiones ofreciéndole información útil en tiempo real, o porque va a hacer el trabajo por él. El dispositivo no debe complicar la vida del usuario con una

instalación complicada, interfaces de usuario poco intuitivas, cableados engorrosos, etc.

Hacer la vida más fácil y cómoda engloba dos conceptos importantes, el confort y la eficiencia de recursos. El confort es el bienestar o comodidad de un individuo, que buscará el mayor posible, pero siempre buscando una eficiencia de los recursos. Usando como ejemplo el control de temperatura de una vivienda, un usuario que tiene frío pondrá la calefacción con una consigna alta para alcanzar una temperatura que le haga entrar en calor, y con ello obtener un confort, pero una vez se alcance una temperatura optima reducirá esa consigna para ahorrar en los recursos utilizados y no derrochar energía. En otras palabras, el equilibrio entre confort y eficiencia ayuda a hacer la vida más fácil y cómoda a las personas.

El modelo tiene que ofrecer versatilidad, es decir, brindar al usuario una amplia gama de elementos para poder controlar su casa y lograr ese confort y eficiencia mencionados anteriormente. Los elementos o dispositivos pueden ser cualquiera que ofrezca una solución domótica, facilitando la vida de las personas en casa y pueden ser controladores de las luces, las persianas, control de temperatura de la vivienda, control de accesos a la vivienda, etc. El usuario tiene la posibilidad de elegir en función de sus necesidades que dispositivo o dispositivos utiliza en su solución domótica gracias a que el modelo es escalable. Por ejemplo, se puede instalar un control de las persianas como única solución, más adelante el usuario decide añadir un control de accesos y un sistema de vigilancia y seguridad y en un futuro implementar un control de temperatura y así sucesivamente hasta poder complementar el sistema domótico, escalarlo y alcanzar una automatización mayor.

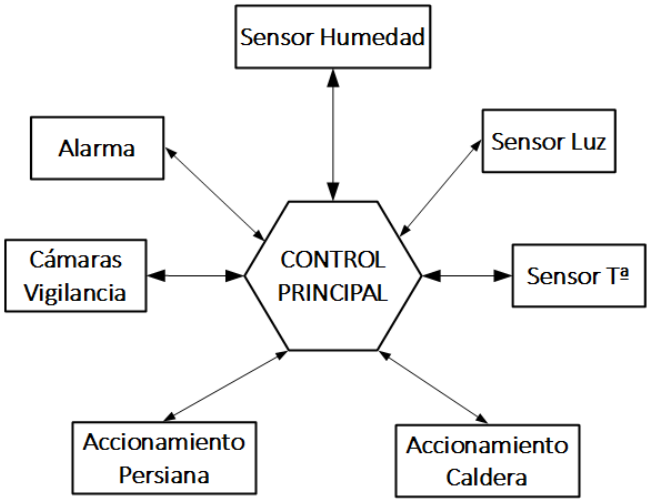

Ilustración 11. Ejemplo de una solución domótica escalable y desacoplada.

Algo que permite escalabilidad del sistema es lograr que los elementos que forman el sistema domótico estén desacoplados, es decir, que cada elemento

o dispositivo funcione por sí mismo sin saber de la existencia del resto de elementos, y si falla, que no comprometa al resto del sistema. Esto se consigue gracias a un elemento principal que será el que recibe información del resto de elementos y los controla, interpretando y estudiando la información recibida y estableciendo un control apropiado a cada situación. El control principal será el único conocedor de los integrantes del sistema, y en función de esto sabrá cómo comportarse. Gracias a esto se consigue una asincronía entre los dispositivos involucrados en el sistema, es decir, trabajarán de una forma complementaria entre ellos, pero nunca crearán una dependencia entre ellos del tipo que sea. En la Ilustración 11 podemos observar un ejemplo de una solución obtenida del sistema escalable y desacoplado. En esa solución se podrían eliminar y/o añadir agentes sin alterar el funcionamiento del resto.

Puesto que los elementos que forman el sistema domótico tienen que comunicarse entre sí y proporcionar información al usuario, cobra sentido el hecho de que se conecten a internet para crear una red de comunicación que cumpla con las necesidades del sistema. Este hecho haría que además de pertenecer a la rama de la domótica, estaría incluido en el paradigma del IoT.

Para establecer una conexión a internet existen diferentes medios o tecnologías, diferenciados en dos grandes grupos, cableada e inalámbrica. Por un lado, dentro del grupo de conexiones cableadas estaría la conexión con cable coaxial, par trenzado (UTP) y fibra óptica. Por otro lado, dentro del grupo de conexiones inalámbricas se encuentra la conexión por satélite, redes móviles (GSM, GPRS, 3G, 4G, etc.), WiFi, etc.

Nosotros vamos a optar por la conexión WiFi para acceder a internet ya que partiendo de que esta tecnología está bastante extendida en la actualidad, el número de usuarios que la utilizan está en auge, siendo raro encontrar una casa que tenga conexión a internet y no WiFi. La elección de esta tecnología simplifica la instalación física, ya que no sería necesario un cableado para la conexión a internet.

Una característica decisiva a la hora de adquirir un dispositivo o dispositivos, además de cumplir con las necesidades cotidianas de una vivienda, es el precio. El coste del producto debe ser asequible para llegar a un mayor público. Las tecnologías que se van a emplear en el desarrollo del sistema son de bajo coste ya que los avances de la tecnología hacen posible encontrar hardware a precios cada vez más bajos, y más aún si hablamos de que los elementos utilizados en domótica no necesitan de una precisión muy alta, ya que las acciones a realizar no son criticas como podrían serlo en la industria, dónde los precios son más elevados. En cuanto al software existe una amplia variedad de posibilidades de código abierto que permiten no incrementar el precio en este aspecto.

En definitiva, para el usuario el factor económico es importante por dos motivos, precio de la solución y dinero que se va a ahorrar gracias al ahorro energético, es decir, la eficiencia de recursos. Si comparamos el precio de la solución con los beneficios esperables, en términos de conveniencia y ahorro energético, el balance puede ser muy positivo.

La forma de tradicional de enfrentar la mejora del confort en el hogar se basa en controlar de forma individual las variables físicas que se relaciona con dicha sensación de sentirse a gusto. En muchos casos, se fija una magnitud física que el usuario prevé que le va a reportar un confort, y si no se logra ese bienestar deseado debe modificar de nuevo la variable física hasta encontrar el grado de satisfacción deseado. Sin embargo, lo que realmente importa es el confort y no tanto lograr esa magnitud física introducida, es por eso por lo que es interesante considerar un control de la sensación de confort del usuario. Si hablamos del control de temperatura, el usuario tiene calor o frio, y según la forma tradicional de dicho control el usuario fija el valor de los grados que desea tener en su casa y el lazo de control se ejecuta en función de esa variable introducida para lograr satisfacer al usuario, es decir, el usuario traduce su sensación de temperatura a una única magnitud física. Resulta atractivo evolucionar ese control tradicional hacia otro en el que, gracias a otras magnitudes físicas, como la humedad, la existencia o no de corrientes de aire, etc. además de la opinión del usuario con su sensación subjetiva, obtenga más información del ambiente que le rodea así como la opinión de la persona, y gracias a ello pueda realizar un control mucho más complejo y preciso adaptándose a los gustos, preferencias y sensaciones de cada usuario y alcanzar su confort, que es el fin último.

El sistema domótico completo es muy amplio y no es posible abarcar la solución completa en el presente proyecto. Se va a partir del control de la calefacción de una casa en función de la temperatura, pero se tratará de enfatizar el carácter de arquitectura abierta y escalable de la solución brindada.

3.3. ¿Por qué MQTT?

Queremos que un controlador central sea el que gobierne el resto de los dispositivos. Que éstos sean independientes y estén desacoplados entre ellos, hasta tal punto que no sepan de la existencia del resto. Los elementos tienen una función marcada y saben qué hacer en caso de fallo sin comprometer al sistema como un todo. Se comunican con el controlador central y él gestiona la información recibida y ejecuta el control apropiado en cada situación. Lo que hace que la interacción entre todos los elementos se produzca de forma asíncrona. Además, se pretende que se puedan añadir o quitar dispositivos sin verse afectado el funcionamiento global, es decir, que sea escalable y versátil.

A partir de estos requerimientos, el sistema domótico a construir utilizará el protocolo *Message Queue Telemetry Transport (MQTT),* ya que es una opción muy válida para satisfacer la comunicación *M2M* entre los elementos del sistema.

Se ven ciertas similitudes entre la estructura presentada del sistema domótico y la estructura de la red de comunicación del protocolo MQTT. Se busca una estructura en estrella con el control centralizado para permitir que los dispositivos o clientes no tengan la necesidad de conocerse para interactuar, es decir, tengan un bajo acoplamiento.

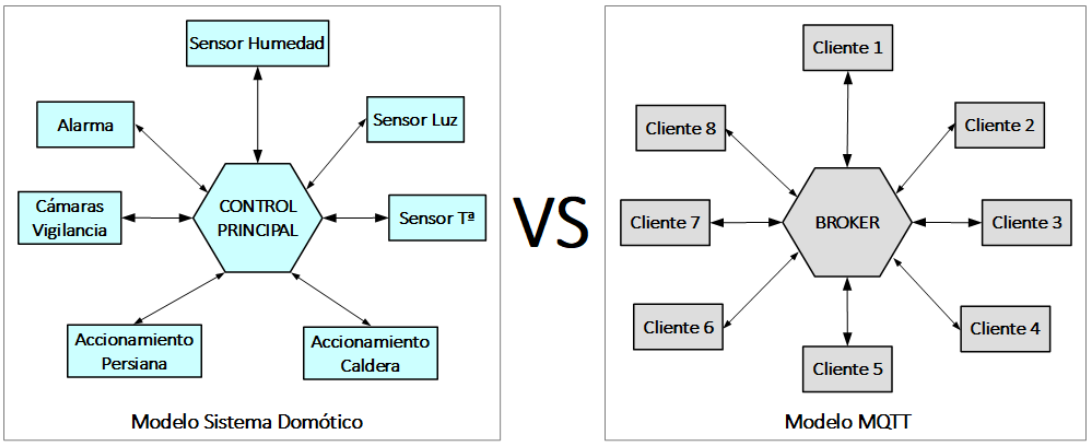

Ilustración 12. Comparación del modelo domótico propuesto y el MQTT.

Aunque la estructura ofrecida por el protocolo MQTT es igual a simple vista como se puede apreciar en la Ilustración 12, existen ciertos matices que hay que resaltar. Hay que diferenciar que los elementos que forman la estructura del sistema domótico son independientes físicamente y los elementos del modelo MQTT son virtuales, es decir, son elementos software, los cuales puede que tengan asignado un dispositivo físico diferente para cada uno o es posible que varios elementos software se ejecuten compartiendo el mismo dispositivo físico. Esta aclaración nos sirve para entender que el control principal o controlador central domótico tiene que gestionar el control de las funcionalidades del sistema domótico y también soportar el *broker* MQTT, es decir, el agente de control que gestiona dichas funcionalidades es cliente del *broker* MQTT. El resto de los dispositivos domóticos serán también clientes del *broker* MQTT. De esta forma el *broker* filtrará los mensajes recibidos por los clientes y redistribuirá la información a sus clientes. Que el destino de los mensajes que se publican el intermediario sea el control principal (cliente MQTT que implementa las funcionalidades del sistema) y/o dispositivos (resto de clientes MQTT), se consigue gracias a una estructura de *topics* y a un protocolo superior o programa que gestionará las conexiones y quien es quien en el sistema.

La interacción entre los elementos que participan en el sistema tiene que ser no-bloqueante. Esto se consigue gracias al modelo de publicación y suscripción orientado a mensajes en el que se basa el protocolo MQTT, que logra disociar al emisor y al receptor tanto en espacio como en tiempo y hace que sean elementos *loose*, o sea, levemente acoplados.

El modelo de publicación y suscripción de MQTT permite que los dispositivos del sistema domótico (sensor de temperatura, accionador de la caldera, accionador de las persianas, etc.) sean independientes y tengan su función marcada. Cada dispositivo tendrá fijado su funcionamiento y solamente dependerá de que el *broker* esté disponible para entablar una conexión con él, pero no dependerá de otros dispositivos de una forma estricta, es decir, tendrán un mínimo de dependencia funcional: se quiere lograr un control de temperatura y cada elemento es necesario para un funcionamiento del sistema como un todo.

Los dispositivos tienen que ser resilientes, es decir, responderán ante un fallo de una forma elegante y seguirán siendo responsivos tras el fallo. Esto también favorece a la independencia de los dispositivos, ya que pueden fallar y recuperarse sin comprometer a los demás elementos. En caso de fallo replicaran donde haga falta cumpliendo con la responsividad, delegando en el control principal el tratamiento del error y se irán a un estado seguro que no comprometa su integridad ni la del resto del entorno o personas. Esto es gracias al modelo de publicación y suscripción, que forma la estructura que vimos en la Ilustración 12 y al estar orientado a mensajes.

Se ha mencionado que el protocolo MQTT está orientado a mensajes. Esto favorece la asincronía de la comunicación y que la sobrecarga del sistema sea menor ya que los recursos serán consumidos solo cuando estos existan o estén activos.

Una de las propiedades más reseñables que debe tener un sistema domótico es la escalabilidad, que facilite ser versátil y se adapte a las necesidades del usuario. En cuanto a este aspecto el protocolo MQTT gracias a la jerarquía de *topics* permite establecer una estructura por niveles y con ello ayudar a la escalabilidad del sistema.

Al ser escalable es posible que el número de dispositivos en una casa puede variar a lo largo del tiempo, o puede que los dispositivos que haya en cada casa sean distintos. Por esta razón el sistema debe funcionar igual cuando haya pocos dispositivos como cuando haya muchos, es decir, debe mantenerse responsivo ante diferentes situaciones con demandas de recursos diversas.

Se puede apreciar que el protocolo MQTT ayuda a que en el sistema domótico se cumplan los aspectos que aparecen en el manifiesto de Sistemas Reactivos.

Es un sistema responsivo incluso después de los fallos, por lo que también es resiliente, es elástico, lo que ayuda a la escalabilidad, y está orientado a mensajes que facilita la asincronía. Esto permite que sea un sistema no-bloqueante.

El protocolo MQTT es además de código abierto lo que permite su uso sin ningún coste y de esta forma se abarata el precio final.

3.4. Solución adoptada

En el presente proyecto se va a desarrollar un termostato capaz de conectarse a internet para permitir que el usuario pueda interactuar con él a tiempo real desde cualquier parte. Se pretende desarrollar un producto configurable, en otras palabras, no se quiere construir una solución *ad hoc* para una determinada situación y que quede restringida a sus características particulares, sino que se busca un resultado comercial. Darle un significado al trabajo de fin de grado y utilizarlo para llegar a una solución útil.

Esta solución tiene como intención satisfacer una necesidad cotidiana, como es el control de la temperatura de una casa, con el objetivo de hacer la vida más fácil al usuario y proporcionarle confort y eficiencia de recursos.

El termostato como tal consta de un controlador central o control principal, que gobierna las funcionalidades principales del sistema, un sensor, que mide la temperatura ambiente y se la envía al controlador central, y un actuador, que accionará la calefacción cuando sea demandada por el control principal. La interfaz de usuario que permite controlar todo el sistema en tiempo real será una aplicación móvil en un smartphone.

Como se aprecia en la Ilustración 13, se pueden diferenciar claramente cuatro partes en la estructura que forma el sistema del termostato: control principal, sensor, actuador, dispositivo móvil. Cada uno tendrá una función marcada e independiente del resto, esto quiere decir que cada dispositivo no tendrá conocimiento sobre la existencia del resto.

Se van a obtener mediciones de temperatura periódicas y el controlador central cotejará la temperatura medida con el valor de referencia que se desea tener, el cual es fijado por el usuario a través de un smartphone. El *setpoint* se fija de forma manual o automática gracias a los intervalos de un calendario que contiene los *setpoint* para cada franja horaria. En función de la comparación entre la temperatura deseada y la temperatura medida, el controlador central encenderá o apagará la calefacción.

Para entablar una comunicación entre los dispositivos se va a emplear el protocolo MQTT, como se viene comentando en anteriores apartados, ya que nos permite hacer el sistema escalable, elástico, responsivo y resiliente, y está orientado a mensajes.

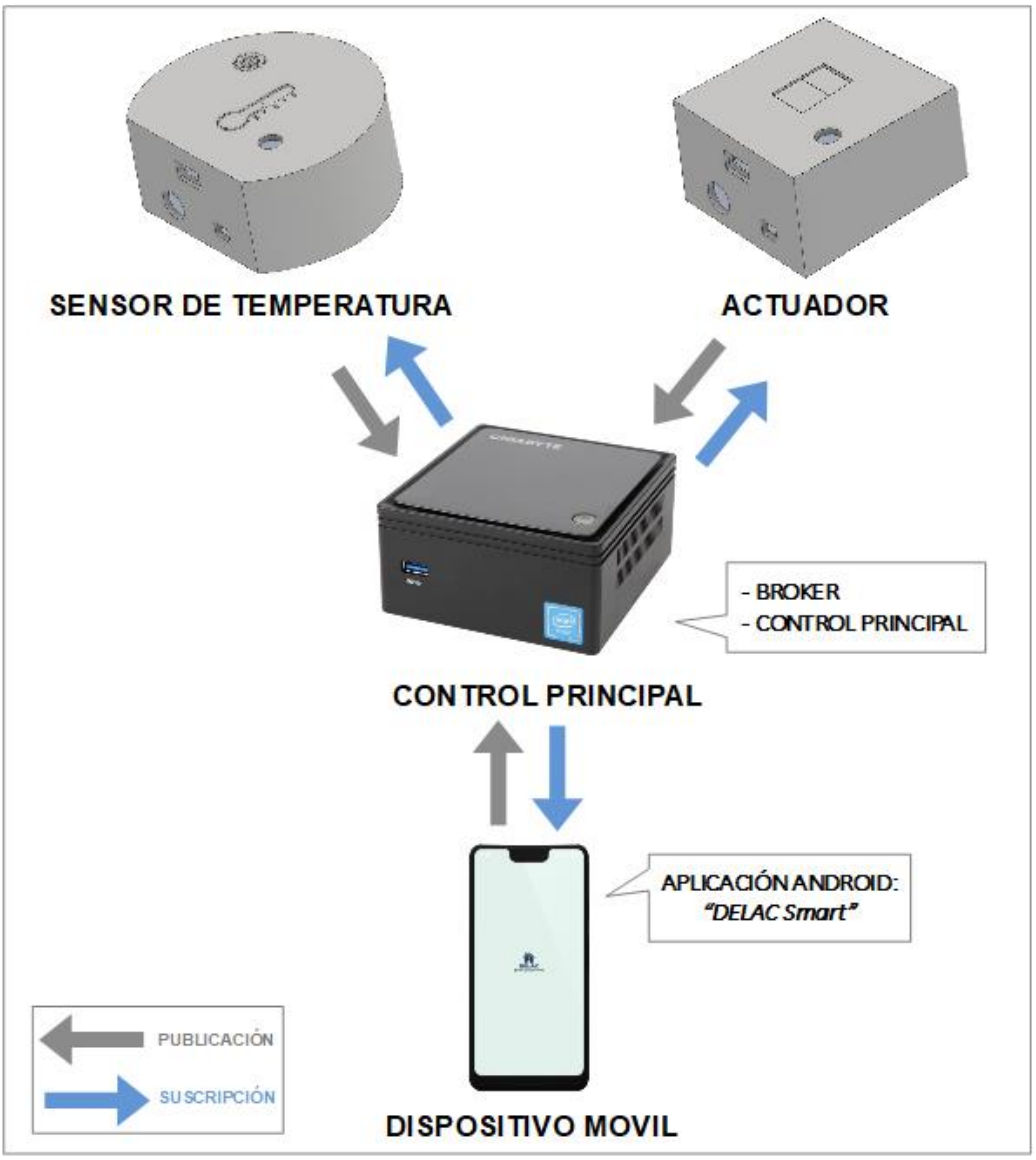

Ilustración 13. Termostato. Estructura del sistema.

Esta comunicación se logra porque todos los elementos están conectados a internet a través de la red WiFi e intercambian mensajes gracias al modelo de publicación y suscripción, en el cual está basado el protocolo MQTT.

El protocolo MQTT se basa en los elementos *broker* y clientes. Los clientes serán cada uno de los agentes que forman parte del sistema, y es por eso por lo que el sensor, actuador, control principal y el smartphone son clientes MQTT y se conectaran al *broker* MQTT para poder comunicarse.

El *broker* tiene que ejecutarse en un dispositivo que opere 24 horas al día y 7 días a la semana, es decir tener plena disponibilidad para que los clientes se conecten a él y puedan enviar o recibir mensajes. En nuestro caso el controlador central siempre estará operativo, puesto que será el encargado de contener el *broker* MQTT.

3.4.1. Controlador central

El controlador central es el elemento esencial del sistema, en el cual hay que diferenciar los dos elementos que contiene: *broker* MQTT y control principal. Este dispositivo se conectará a internet a través del WiFi de casa y en consecuencia tanto el *broker* como el control principal tendrán acceso a internet.

El *broker* MQTT se ejecutará en el controlador central porque es un elemento imprescindible, sin él no se concibe el sistema como tal, y siempre está disponible ya que estará encendido 24/7, lo que permite que cualquier cliente se conecte en el momento que quiera.

La función del *broker* MQTT es ser el intermediario de los clientes que se hayan conectado, estén conectados o se vayan a conectar a él, de tal forma que permita una comunicación asíncrona y facilite que los dispositivos estén levemente acoplados. Será el encargado de filtrar los mensajes que se publiquen en función de los *topics* y distribuirlos a los clientes que se hayan suscrito a dichas etiquetas.

Estos *topics* se organizarán con una estructura jerárquica que organiza la comunicación y la hace más limpia. Incrementa la posibilidad de escalar el sistema y facilita añadir o quitar elementos del sistema domótico, característica que debe tener nuestra solución.

En la Ilustración 14 podemos observar la forma básica que tendría la estructura de *topics* de la solución que hemos adoptado. Sobre esta estructura es fácil añadir y eliminar elementos, haciéndola crecer sin comprometer al resto. Se puede añadir más dispositivos debajo de “CASA”, es decir, al mismo nivel que los que se aprecian en dicha ilustración. se pueden añadir más casas, que tendrán asociados sus propios dispositivos, posibilitando su crecimiento y escalabilidad.

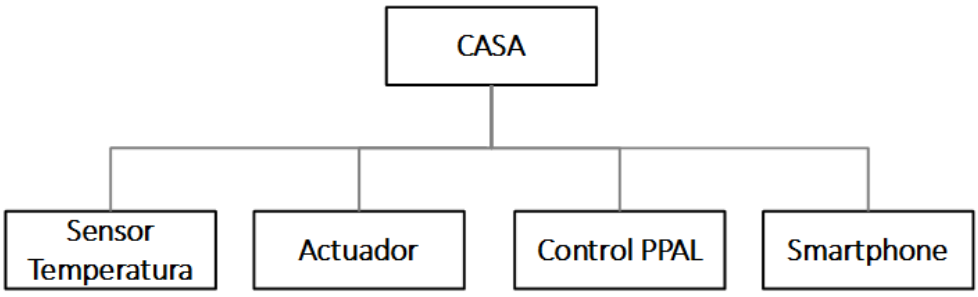

Ilustración 14. Estructura básica de topics de la solución.

Además del *broker* MQTT, el controlador central implementa el control de las funcionalidades del dispositivo, es decir, es el “cerebro” del sistema. Este *software* es cliente del *broker* e interactúa con él para establecer una comunicación con el resto de los clientes de los que necesita información.

El control principal, en rasgos generales, es el encargado de recibir el valor de la temperatura ambiente por parte del sensor y el valor de la temperatura de referencia a la que tiene que encenderse la caldera, que la fija el usuario, compararlos y actuar sobre la caldera encendiéndola o apagándola. Este funcionamiento básico queda representado en el esquema de la Ilustración 15.

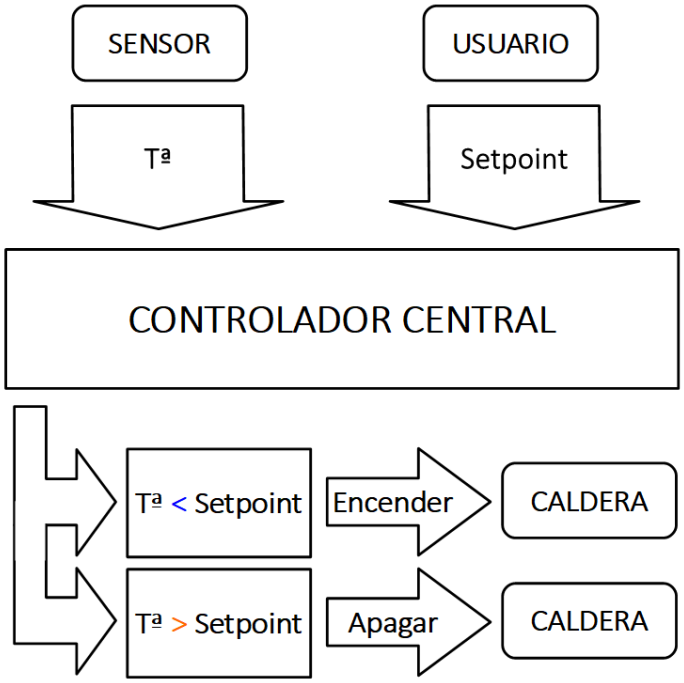

Ilustración 15. Esquema funcionamiento simple del termostato.

El fin último es el control de la temperatura de una casa de la forma más eficiente y ofreciéndole confort al usuario. Para lograr esto, y que cobre sentido la instalación del termostato inteligente, se le dota de ciertas funcionalidades que le hagan la vida más fácil.

Para poder llevar a cabo estas funcionalidades necesita de otros dispositivos, que se explicarán en detalle más adelante, pero es necesario saber que existen para entender el funcionamiento del control principal. Los dispositivos de los que se habla son: un sensor de temperatura, un actuador, y una interfaz de usuario. El sensor mide la temperatura y la envía periódicamente, el actuador es el encargado de encender o apagar la calefacción y la interfaz de usuario nos permitirá interactuar con el sistema, o lo que es lo mismo, configurar las funcionalidades ofrecidas y observar el estado actual del sistema.

Se parte del control básico de la calefacción en función de la temperatura y un *setpoint* introducido manualmente por el individuo, y se irá incrementando la complejidad con un mayor número de funcionalidades que permitan facilitarle la vida y automatizar la casa no solo a los efectos de garantizar la temperatura óptima (confort), sino que también le permita ahorrar dinero y no derrochar recursos innecesariamente (eficiencia).

El control básico de temperatura es el que encontramos en un termostato al uso, que regula la temperatura en función de los grados que se hayan fijado manualmente y si se quiere modificar la temperatura de referencia hay que hacerlo *in situ*. Para darle más versatilidad, una de las funcionalidades que incluye el control principal es un calendario.

El calendario será semanal, es decir, se programará el funcionamiento de los días de una semana. Cada día tendrá unos intervalos de funcionamiento, y el usuario configurará la hora de inicio y la temperatura deseado para cada intervalo.

	Intervalo 1	Intervalo 2	Intervalo 3	Intervalo 4	Intervalo 5	Intervalo 6
LUNES	8:00h	10:00h	14:00h	16:30h	19:00h	22:00h
	21° C	17° C	23° C	18° C	21° C	17° C

Tabla 3. Ejemplo de configuración de un día del calendario.

En la Tabla 3 podemos ver un ejemplo de cómo sería la configuración de un día del calendario que nos permite programar distintos intervalos de funcionamiento. Con esto logramos que haya distintos *setpoint* o valores de referencia a lo largo del día y con ello que aumente la eficiencia del dispositivo al adaptar la temperatura en cada momento del día. Esto será igual para cada día de la semana, que tendrá sus propios intervalos con sus valores de referencia.

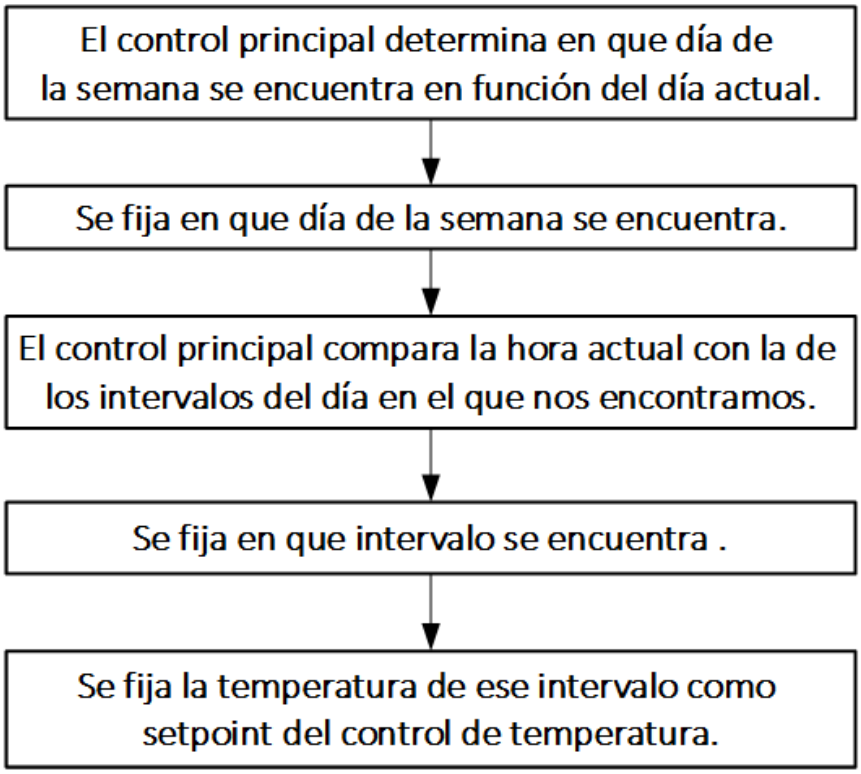

Ilustración 16. Diagrama de la funcionalidad de calendario.

El control principal determinará en que día del calendario se encuentra en función del día actual, y una vez sepa que día de la semana es, comparará la hora actual con la hora de inicio de cada intervalo para saber en cual está y fijar la temperatura de referencia que corresponda. Ese valor de referencia que se fija en función de la hora y el día, será el que el control principal tomará como *setpoint* para regular la temperatura. Esto lo podemos ver de una forma más gráfica en el diagrama que se muestra en la Ilustración 16.

Complementaria a la funcionalidad del calendario, el control principal ofrece la funcionalidad de poder seleccionar varios modos de funcionamiento, que se clasifican en función del modo de trabajo, estación del año, si se está en casa o no, y si se está de vacaciones o no.

El termostato podrá funcionar de forma manual, semi-automática o automática, a esta funcionalidad la llamamos "Modo de trabajo".

- En el modo automático el control principal regulará la temperatura en función de los *setpoint* programados en los intervalos del calendario.

- El usuario puede modificar dicho *setpoint* de forma manual para que en un momento puntual sea más alto o bajo en función de sus necesidades, este modo es el llamado semi-automático. Cuando se alcance la hora de inicio del siguiente intervalo, se fijará el setpoint de ese intervalo y sustituirá al fijado manualmente y se volverá al modo automático. Por ejemplo, si en un momento dado la temperatura de referencia fijada por un intervalo en el modo automático es de 23ºC y el usuario tiene calor, puede bajar ese *setpoint* para que se apague la calefacción y así no seguir calentando la casa.

- El modo manual, como su propio nombre indica, el usuario fija un setpoint directamente y ese valor prevalecerá por encima del programado en el calendario.

Si el individuo sale de casa, ya sea por un periodo corto de tiempo (menos de un día) o más prolongado (varios días), el control principal brinda la posibilidad de programar un periodo de tiempo en el cual la temperatura se regulará en función de un valor prefijado, para ahorrar recursos durante ese tiempo que se va a estar fuera.

El valor prefijado que será el *setpoint* para el control de temperatura durante la ausencia del usuario se llamará "Temperatura eco". Este nombre viene de temperatura ecológica, ya que ahorra energía y recursos.

Si el periodo en el que el usuario va a ser por unas horas podrá activar el "Modo fuera de casa".

- Si no se activa, quiere el control interpreta que estas en casa y se rige por el modo de trabajo que este seleccionado: automático o manual.

- En caso de activarlo, la interfaz de usuario solicitará una hora de regreso a casa. Hasta esa hora el control fijará como *setpoint* la "Temperatura eco" que se habrá configurado anteriormente. Cuando llegue la hora de regreso se volverá al modo que se estuviera previamente a la activación del "Modo fuera de casa".

Si el periodo de tiempo que el usuario va a pasar fuera es de varios días, entonces puede activar el "Modo vacaciones".

- Si no se activa, se considerará que se está en casa y el control se regirá por el modo de trabajo que esté seleccionado previamente: automático o manual.

- En caso de activarlo, la interfaz de usuario solicitará un día de regreso a casa. Hasta ese día el control fijará como setpoint la "Temperatura eco" que se habrá configurado anteriormente. Cuando llegue el día de regreso se volverá al modo en que se estuviera previamente a la activación del "Modo vacaciones".

Cuando llegan las estaciones calurosas, la caldera no tiene necesidad de funcionar. Se prevé de un "Modo estación del año", que constará de modo invierno y modo verano.

- En el modo invierno, el control dependerá del resto de modos de funcionamiento para controlar la temperatura.

- En el modo verano, se apagará la caldera y no hará caso del resto de modos de funcionamiento.

Como se aprecia, estos modos de funcionamiento le brindan eficiencia al dispositivo, por lo cual el usuario obtendrá un ahorro en su factura. Además, todas estas funcionalidades le ayudan al usuario a automatizar su casa, que se despreocupe del control de la calefacción de su casa, porque solo necesitará configurar una vez el sistema, y salvo en momentos puntuales que necesite modificar el funcionamiento normal, los modos de fuera de casa, vacaciones y modo manual, permiten hacer la vida más fácil a la persona que lo use, brindándole un confort y una eficiencia.

Otra de las funcionalidades que se ofrece es "Estadísticas", donde podemos encontrar información útil. El usuario podrá ver cuando ha estado encendida la caldera y cuánto tiempo, cuál ha sido la temperatura máxima y mínima que

se ha alcanzado, etc. Con esta información se ayuda a la toma de decisiones del usuario, para determinar la configuración más eficiente.

A modo de seguridad, el control principal tiene que ofrecer un modo seguro, por lo que en el caso de que no reciba temperatura durante un periodo de tiempo razonable, el control principal interpretará que se ha desconectado el sensor, por lo cual no tiene posibilidad de saber que temperatura hay en casa. En ese caso el control principal llevará la caldera a un estado seguro, es decir, le enviará un mensaje al actuador para que apague la caldera. De esta forma, si no se tiene suficiente información la caldera estará apagada y no consumirá recursos.

3.4.2. Sensor de temperatura

El sensor de temperatura es uno de los dispositivos del sistema domótico que forman parte del termostato inteligente. Su función es obtener la temperatura del ambiente y enviársela al controlador central para que él realice el control de la calefacción.

Este dispositivo es independiente, es decir, su funcionamiento no depende de otros para funcionar. Únicamente necesitará tener acceso al *broker* MQTT para poder comunicarse y enviar la temperatura. El sensor no sabe de la existencia de otros elementos del sistema, por esa razón, hará lo que tenga que hacer por sí mismo. Por ejemplo, la temperatura será enviada pero el sensor desconoce si otros dispositivos la reciben.

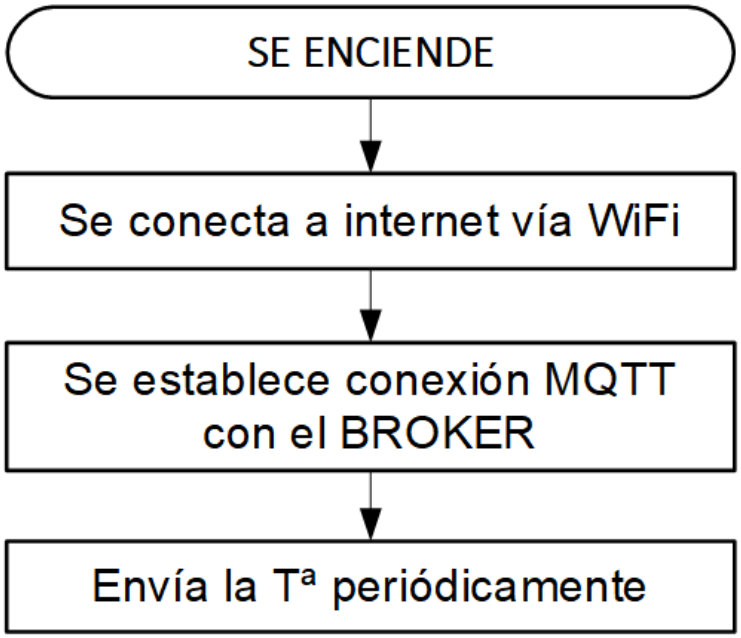

Ilustración 17. Diagrama de funcionamiento básico del sensor.

Lo que se muestra en el diagrama de la Ilustración 17 es el procedimiento que hace el sensor cuando pasa de estar apagado a encendido, que será similar al

que se hace en el actuador. En el podemos diferenciar dos partes. Una sería el proceso de conexión, en primer lugar, a internet, y seguidamente la conexión MQTT. Y por otro lado está el funcionamiento del sensor, es decir, realizar mediciones de temperatura y enviarlas.

El dispositivo se conectará a internet a través de la red WiFi de la casa. Como se le intenta dar una configurabilidad al producto, las credenciales (usuario y contraseña) de la red WiFi se las facilitará el usuario a través de una interfaz creada a ese efecto, en vez de ir preprogramadas en el prototipo. De esta forma se consigue que el dispositivo pueda ser usado en cualquier casa o si se produce algún cambio en la red de la casa poder conectar el sensor a internet de nuevo.

Con la conexión a internet no es suficiente para comunicarse con el controlador central, necesita un protocolo de comunicación. Como se ha comentado anteriormente el elegido es el protocolo MQTT. Y en la estructura MQTT, el sensor es un cliente del *broker* implementado en el controlador central. Por consiguiente, el sensor de temperatura tiene que conectarse al *broker* para poder enviar y recibir mensajes del controlador principal.

Una vez se haya establecido la conexión MQTT, el sensor estará preparado para enviar y recibir información, por ejemplo, la temperatura medida.

Con la intención de darle robustez al dispositivo, el sensor almacenará las credenciales de la red WiFi a la que se ha conectado. De esta forma si se pierde conexión a internet, al conocer el usuario y la contraseña de la red WiFi, intentará nuevamente establecer una conexión a internet. Gracias a esto el usuario no tiene que estar introduciendo las credenciales cada vez que tenga que conectarse a internet. De la misma forma y con la misma intención, si se perdiera la conexión con el *broker* se intentará conectarse de nuevo.

Tal y como se menciona anteriormente el objetivo del sensor es obtener mediciones periódicas de la temperatura ambiente de una casa y enviárselas al *broker* para que los dispositivos que necesiten ese valor puedan consumirlo, como por ejemplo el controlador principal. Es decir, este dispositivo tiene dos funciones paralelas en el tiempo, por un lado, medir la temperatura, y por el otro, enviar el valor medido.

La medición del valor de la temperatura ambiente se realiza periódicamente cada cierto tiempo, el cual es un parámetro que podrá fijar el usuario. Es decir, este parámetro es el tiempo de rastreo, el cual una vez transcurrido realiza una nueva medición y así sucesivamente. Esta medición se realiza independiente del envío del valor, o sea, el sensor mide periódicamente en función del tiempo de rastreo se haya enviado o no el valor de la temperatura.

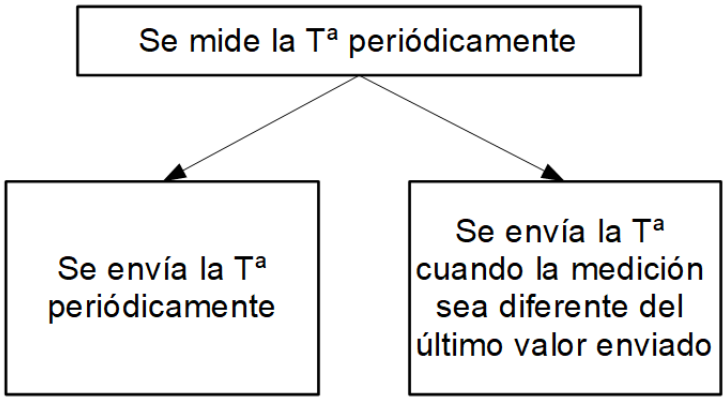

Ilustración 18. Medición y envío de la Tª del sensor.

El envío del valor de la temperatura medida se restringe a dos condicionantes, por un lado, un tiempo máximo de envío, por otro el incremento de temperatura, también llamado delta "Δ". En la Ilustración 18 podemos apreciar lo comentado en este párrafo.

Para el control principal sería suficiente una lectura de la temperatura, mientras ésta no varíe. Por ejemplo, si la temperatura es de 23ºC en el ambiente y es estable, con que el sensor envíe una vez este valor valdría, es decir, seria innecesario que el sensor estaría enviando continuamente 23ºC. Sin embargo, es necesario enviar la temperatura cada cierto tiempo (tiempo máximo de envío) para asegurarnos de que el ultimo valor que posee el control principal y el ultimo valor medido por el sensor son iguales y no se ha perdido ningún mensaje debido a un fallo. Este tiempo nos servirá además para comprobar el estado de la conexión, es decir, si sigue estando activa entre el control principal y el sensor.

Como se ha comentado, la temperatura si no se envía por tiempo se envía en función del incremento de temperatura que se produzca. Esto quiere decir que si no hay cambios y la temperatura se mantiene constante, se enviará por tiempo. Sin embargo, si se produce un cambio de temperatura se envía el nuevo valor medido. Este cambio es un incremento en valor absoluto, o sea, se considera tanto si disminuye como si aumenta el valor. Este incremento de temperatura o delta "Δ" tiene que ser significativo y lo fijará el usuario. Para enviar la temperatura, la diferencia entre el ultimo valor medido y el ultimo valor enviado tiene que ser mayor de la delta "Δ" que ha configurado el individuo. Si es menor que esa delta "Δ" se considerará que no se ha producido una variación significativa, y por lo tanto que no sea importante.

Este funcionamiento permite al controlador llevar a cabo un control basado en eventos, ya que el sensor envía la temperatura de forma discreta cuando se produzca un incremento de temperatura, es decir, cuando se produzca un evento. Gracias a esto, se logra que las redes de comunicación no se saturen,

ahorrando en el ancho de banda y en consecuencia se ahorrara energía, ya que los microcontroladores que se usan tanto para el sensor de temperatura como actuador el momento de máximo gasto energético son en el envío de información.

3.4.3. Actuador

El actuador es otro de los dispositivos del sistema domótico que forman parte del termostato inteligente. Es el encargado de accionar la caldera, encendiéndola o apagándola, a petición del controlador central.

De igual manera que el sensor de temperatura, este dispositivo tiene un funcionamiento independiente del resto, esto quiere decir que sabe qué hacer en base a sus criterios. El actuador depende de tener acceso al *broker* MQTT para comunicarse y del valor de la consigna que recibe del control principal.

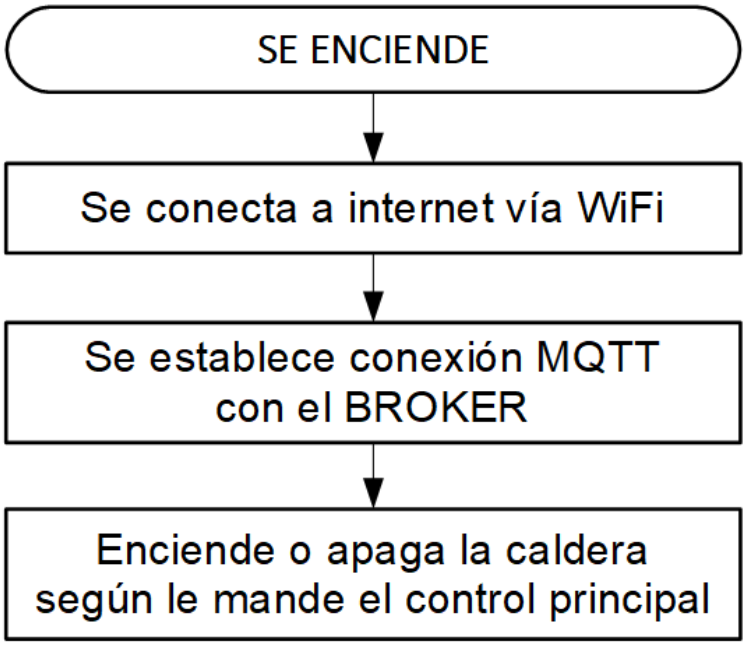

Ilustración 19. Diagrama de funcionamiento básico del actuador.

En el diagrama de funcionamiento que vemos en la Ilustración 19 se aprecia el proceso que sigue el actuador para encenderse. Se pueden diferenciar dos partes. Por un lado, el proceso de conexión, que es igual que el que se produce en el sensor de temperatura. Este proceso de conexión consta de dos partes también, una es la conexión a internet y la segunda es la conexión MQTT. Por otro lado, está el funcionamiento del actuador propiamente dicho, que sería encender o apagar la calefacción en función de lo que mande el control principal.

En el actuador el proceso de conexión a internet y al *broker* MQTT es igual al del sensor de temperatura. Para conectarse a la red WiFi de casa y establecer una conexión a internet hay que facilitarles las credenciales de la red, que se

las introduce el usuario, y de esta forma conseguiremos que el dispositivo sea configurable y pueda ser usado en cualquier casa.

Como ya se ha comentado el protocolo que se va a utilizar para comunicarse con el resto de los dispositivos es el protocolo MQTT. El actuador es un cliente, y una vez se conecte a internet, se conectará al *broker*.

Una vez se haya establecido la conexión MQTT, el actuador estará preparado para enviar y recibir información, por ejemplo, recibir la consigna de la caldera o enviar el estado de la misma.

El actuador almacenará las credenciales de la red WiFi a la que se ha conectado por última vez. De esta forma en el caso de un fallo de conexión a internet, el dispositivo tiene la posibilidad de volver a reintentar la conexión, puesto que conoce el usuario y la contraseña de la red WiFi. Gracias a esto, el usuario no tiene que estar introduciendo las credenciales cada vez que se quiera conectar a internet. De la misma forma y con la misma intención, si se perdiera la conexión con el *broker*, el actuador intentará conectarse de nuevo. Y con esto se consigue darle robustez al sistema.

La función del actuador es accionar la caldera, es decir, encenderla o apagarla en función de la orden que le manda el control principal. El actuador estará esperando hasta que el control principal le mande que encienda o apague la caldera.

Otra de las funciones del actuador es informar del estado de la caldera periódicamente, es decir, si la calefacción está encendida o apagada. Con este mensaje el cliente que reciba conocerá el estado actual de la caldera y podrá estar informado a tiempo real. El estado se envía independientemente de recibir o no recibir la señal de mando de la caldera.

Con el fin de proporcionar seguridad y eficiencia, el actuador tiene un modo seguro y en caso de cualquier fallo la caldera se apagará. Si la conexión a internet falla, se pierde la conexión con el *broker*, etcétera, el actuador apagará la caldera para ahorrar recursos y proporcionar seguridad. En el caso de perder la conexión, no se recibirán mensajes y por tanto no se sabrá si el control principal está mandando la señal de mando para que se encienda o se apague, y ante la duda prevalece el modo seguro y se apaga la calefacción.

3.4.4. Teléfono móvil

El teléfono móvil es el elemento del sistema domótico que nos permitirá interactuar con los dispositivos, obtener información en tiempo real, configurarlos y ejercer un control sobre la caldera, es decir, será el intermediario que nos proporcione la interfaz de usuario. En la Ilustración 20 vemos como el teléfono móvil interactúa con el sistema domótico.

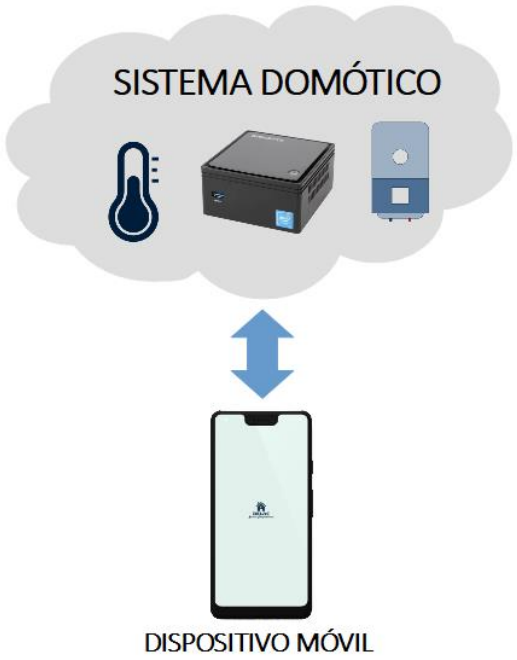

Ilustración 20. Interacción dispositivo móvil con el sistema domótico.

Utilizaremos un smartphone porque es un aparato tecnológico que actualmente lo utiliza gran parte de la población y está familiarizado con su uso. Con una aplicación de sencilla instalación y uso, cualquier persona será capaz de interactuar con el producto domótico que se está desarrollando.

Un factor importante para usar el teléfono como intermediario entre el sistema y el usuario es porque tiene la capacidad de conectarse a internet, ya sea a través de la red móvil (3G, 4G, etc.), o a través de la red WiFi, por ejemplo, el WiFi de su casa.

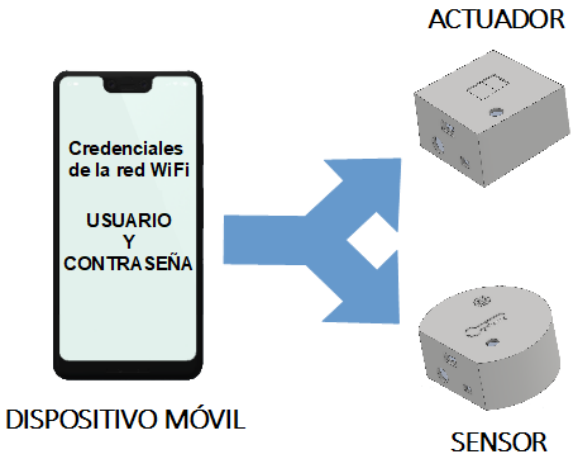

Ilustración 21. Introducción de las credenciales WiFi al sensor y actuador.

Por un lado, el teléfono inteligente nos servirá de interfaz para configurar la red WiFi a la que se tienen que conectar el sensor y el actuador, tal y como podemos ver en la Ilustración 21. En los apartados anteriores del actuador y sensor se

había comentado que necesitan que les facilitemos las credenciales de la red WiFi en lugar de ir predefinidas en la programación del dispositivo. Nos apoyaremos en el teléfono como interfaz de configuración para introducir esos parámetros en los dispositivos. De esta forma le damos configurabilidad a los dispositivos y ayudamos a que el sistema tenga forma de producto comercial.

Por otro lado, el smartphone tendrá una aplicación con la que el usuario podrá ver información del sistema domótico en tiempo real, configurar ciertos parámetros y en general controlar la calefacción de la casa. En otras palabras, la aplicación del teléfono móvil será la parte tangible de las funcionalidades que se han descrito en el apartado del controlador principal, entre otras cosas.

La aplicación será un cliente MQTT, por lo que a través de la conexión a internet que tiene el teléfono móvil establecerá una conexión con el *broker* para poder comunicarse con los dispositivos del sistema domótico y de esta forma interactuar con ellos.

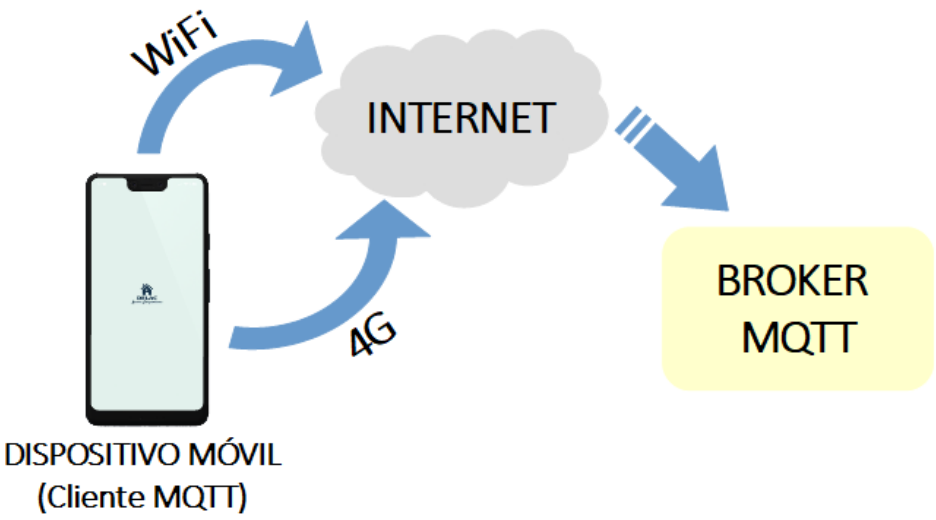

Ilustración 22.Conexión del teléfono al broker a través de internet

El fin de la aplicación es hacer de intermediario entre el sistema y el usuario, es decir, que el usuario pueda controlar la calefacción de su casa a través del teléfono móvil. La aplicación le brindara a al usuario datos útiles para facilitar el control y configuración.

Una de las cosas que el usuario podrá hacer con la aplicación es observar el estado del sistema, es decir, ver la temperatura que hay en su casa, ver si la calefacción está encendida o apagada, entre otras, esté donde esté y en tiempo real.

Además, el usuario podrá usar las funcionalidades que ofrece el control principal y que se han explicado en el apartado del controlador central. Se podrá programar el calendario de funcionamiento, ver las estadísticas y

configurar los parámetros de los dispositivos, así como los modos de funcionamiento del control principal.

Un resumen de lo que se puede hacer con el dispositivo móvil:

- Introducir las credenciales WiFi al sensor y al actuador.
- Ver la temperatura ambiente de la casa
- Ver el estado actual de la caldera.
- Modificar el setpoint.
- Programar el calendario de funcionamiento.
- Ver las estadísticas del sistema.
- Configurar los parámetros de los dispositivos.
- Configurar los modos de funcionamiento del control principal (automático o manual, si sale fuera de casa, etc.)

4. DECISIONES DE IMPLEMENTACIÓN

Una vez se ha explicado lo que se quiere conseguir y que se puede observar en el apartado anterior. En este apartado se va a exponer las decisiones que se toman para desarrollar el sistema domótico de control de temperatura que se toma como objetivo de este trabajo de fin de grado.

Estas decisiones se pueden dividir en dos grandes grupos: las decisiones de *hardware* y las de *software*.

4.1. HARDWARE

4.1.1. Controlador central

Entendemos controlador central como el dispositivo encargado de llevar a cabo las tareas principales de control del sistema domótico. En nuestro caso, el proyecto se centra en el control de temperatura de una casa, por lo que el controlador central gestionará dicho control, además, gestionará las funcionalidades que el sistema le brinda al usuario.

Este dispositivo tiene que poder llevar a cabo las tareas que tenga adjudicadas y para ello debe disponer de los recursos suficientes. Sin embargo, las tareas que debe desempeñar no serán críticas ni de gran magnitud, por lo que es posible implementarlas en múltiples placas de desarrollo, que gracias al avance de la tecnología en un pequeño espacio tienen la potencia suficiente para ejecutar aplicaciones importantes, además, a un precio reducido.

El controlador central debe tener un tamaño comedido, para que no sea engorrosa su colocación dentro de la vivienda. Además, la instalación debe ser sencilla para no complicarle la vida al usuario, ya que lo que se busca es el resultado opuesto. Así que cuanto menor sea el cableado mejor, y es por eso que el dispositivo que se emplee para el controlador central deba poseer una conexión WiFi a internet.

En un primer lugar se pensó utilizar la placa de desarrollo *Raspberry Pi 3 Model 3*[7], ya que posee la potencia y recursos suficientes para poder llevar a cabo el desarrollo e implementación del control principal en ella. Además, posee conexión inalámbrica, lo que reduce el cableado. Sin embargo, pese a tener una interfaz similar a la de un ordenador, ya que implementa el sistema operativo *Raspbian*[8] basado en *Debian*, resulta un poco pesado trabajar directamente con dicho dispositivo a la hora de desarrollar el programa del control principal.

Finalmente se optó por usar un mini-pc de la marca *GIGABYTE*, como el de la Ilustración 23, el cual es un ordenador de reducido tamaño y recursos limitados dentro de la computación, pero sobradamente suficientes para el desarrollo de nuestro proyecto. Posee conexión inalámbrica WiFi a internet, lo cual encaja

con las especificaciones. En nuestro caso el ordenador tiene instalado el sistema operativo *Linux UBUNTU 18.04 LTS*.

Esta opción no se elige como propuesta final de cara a presentarlo a un cliente, puesto que es un producto comercial ya en sí mismo como ordenador, pero la elegimos porque cumple con nuestras necesidades y simula lo que sería el controlador central que iría incluida en nuestra solución domótica.

Ilustración 23. Mini-PC GIGABYTE.

Una de las principales razones por la que nos decantamos por la utilización de este dispositivo es porque nos facilita las tareas de desarrollo, ya que, al ser un ordenador, nos proporciona todas las ventajas de un ordenador común y por consiguiente nos permite desarrollar el programa que se encargará de gestionar todas las funcionalidades del control de temperatura de nuestro sistema domótico de una forma más fluida, ya que posee más recursos que una placa de desarrollo y soporta aplicaciones más pesadas, como son los IDE de programación. Además, gracias a su tamaño reducido nos resulta útil para simular la instalación en casa.

En un futuro se optará por el diseño de un controlador que se adapte específicamente a las características propias del sistema en vez de usar uno genérico. Sin embargo, esto no entra en el objeto del presente trabajo y por eso utilizaremos el ordenador *GIGABYTE.*

4.1.2. Periféricos

Entenderemos como periféricos todos aquellos dispositivos que se le puedan añadir al controlador central para formar el sistema domótico. En nuestro caso como el sistema domótico parte de un control de temperatura, los periféricos

serán el actuador y el sensor de temperatura. Ambos dispositivos compartirán características hardware, y por ello los englobamos en el mismo apartado.

El sensor de temperatura y el actuador llevarán a cabo las tareas de recopilación de la temperatura del ambiente y accionamiento de la caldera respectivamente. Son tareas que no requieren de grandes recursos por lo que pueden ser implementadas en cualquier placa de desarrollo. Estos dispositivos tienen que poder conectarse a internet a través del WiFi para poder establecer una comunicación con el resto de los elementos del sistema.

En ambos dispositivos periféricos se va a utilizar la placa de desarrollo *NodeMCU*[9], la cual es totalmente abierta, tanto a nivel de *software* como de *hardware* lo que abarata costes. Este dispositivo se puede apreciar en la Ilustración 24.

Ilustración 24. Placa de desarrollo NodeMCU.

El *NodeMCU* es un kit o placa de desarrollo que nos facilita la vida a la hora de prototipar, programar y construir nuestros circuitos. Una placa de desarrollo es más que un microcontrolador, estos kits incorporan un chip que contiene el microcontrolador o MCU del inglés *microcontroller unit*. En la Ilustración 25 se puede ver representado esto de lo que estamos hablando.

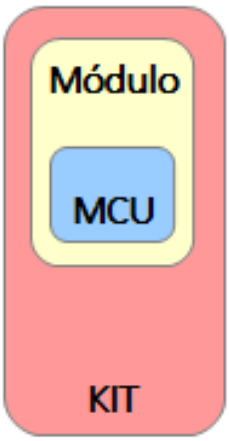

Ilustración 25. Componentes placa o kit de desarrollo.

El MCU o microcontrolador que emplea el *NodeMCU* es el ESP8266, el cual tiene todo lo necesario para funcionar como si fuese un ordenador, destacando que permite conexión WiFi. Las principales características son las siguientes[10]:

- El microcontrolador es de bajo consumo de 32 bits.
- Posee un módulo WiFi que posee la certificación de la *Wi-Fi Alliance*, por lo que usa el protocolo IEEE 802.11b/g/n, y trabaja a una frecuencia de 2.4GHz.
- Tiene una entrada analógica de 10 bits y 17 pines de propósito general de entrada y salida (GPIO).
- Permite comunicación I2C, SPI, UART entre otras e incorpora PWM.
- No tiene memoria *flash* para almacenar los programas.
- Se alimenta a 3.3V.

Ilustración 26. Microcontrolador ESP8266.

Este microcontrolador está dentro del módulo WiFi ESP-12 que usa el *NodeMCU,* y que posee una memoria *flash* para almacenar los programas y una antena para el módulo WiFi. Lo podemos observar en la Ilustración 27.

Ilustración 27. Módulo WiFi ESP-12.

Este módulo facilita el acceso a los pines en mayor medida que como estaban en el ESP8266 e incorpora una antena y memoria *flash*, sin embargo, resulta complicado su programación y por eso en vez de utilizar este módulo se utilizará el *NodeMCU* para el actuador y el sensor de temperatura.

El kit de desarrollo incluye el módulo ESP-12, pero añade ciertos elementos que nos facilitan su programación y en definitiva su uso. Este kit de desarrollo se puede ver en la Ilustración 24, en el cual se puede apreciar como en un

extremo incluye el módulo WiFi ESP-12, pero a mayores incorpora los siguientes elementos:

- Conversor Serie-USB para facilitar la programación de la placa.
- Pines de alimentación de 3.3V y 5V para los sensores.
- LEDs de estado de la placa.
- Botón de *reset*.
- Un fácil acceso a los pines.

En definitiva, usaremos la placa de desarrollo *NodeMCU* para implementar los dispositivos periféricos puesto que es muy sencilla su programación a través de un cable USB, tiene conexión WiFi, posee los recursos suficientes y tiene múltiples pines para conectar los sensores y elementos electrónicos que sean necesarios. Además, tiene un precio reducido por ser una placa abierta, y nos conviene a la hora de reducir costes.

Hay varios modelos, pero nosotros usaremos el *NodeMCU Lolin V3*, y su esquema de pines detallado lo podemos ver en la

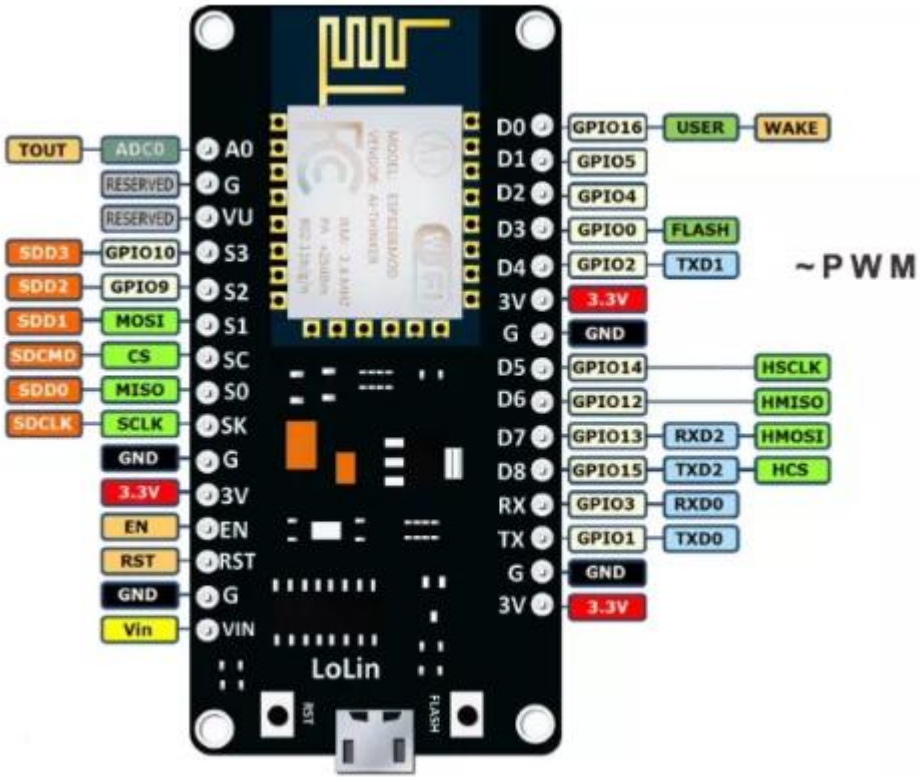

Ilustración 28. Esquema detallado de pines del NodeMCU Lolin V3.

El *NodeMCU* será el elemento principal de los dos periféricos que se van a implementar, el actuador y el sensor de temperatura, será el que contenga el programa y ejecute las tareas que se les encomiende. Los dos prototipos tendrán características parecidas, puesto que usan la misma placa de

desarrollo, sin embargo, los elementos electrónicos y sensores serán diferentes en cada caso, ya que cada periférico tiene un propósito diferente. A continuación, se detallan cada uno en profundidad y por separado.

4.1.2.1. Sensor de temperatura

El sensor de temperatura es el periférico que se encarga de tomar medidas periódicamente de la temperatura ambiente, la procesa y se la envía al *broker* para que los clientes que necesiten dicha información puedan obtenerla.

Como se ha comentado antes, el elemento de procesamiento que llevará a cabo las tareas a desempeñar por el sensor de temperatura, será la placa de desarrollo *NodeMCU*. En ella se implementará el programa, es decir, será el "cerebro" de nuestro sensor de temperatura. A ella se le conectaran los elementos electrónicos y eléctricos necesarios para construir un dispositivo completo.

Los elementos necesarios serán los siguientes:

- NodeMCU.
- Sensor de temperatura.
- Resistencias.
- Diodos LED.
- Pulsador.
- Condensadores.
- Fuente de alimentación 5V y/o 3.3V.
- Cables.
- Placas de prototipado.

El sensor medirá la temperatura del ambiente, por lo que necesitamos un sensor de temperatura. Para nuestro prototipo usaremos el circuito integrado LM35[11], que podemos ver representado en la Ilustración 29.

Ilustración 29. Sensor de temperatura LM35.

Este sensor esta calibrado directamente en grados centígrados y nos proporciona una salida analógica de tensión lineal proporcional a los grados centígrados, en el cual 10mV corresponden a 1º C. Mide un rango de temperatura desde los -55º C hasta los 150º C y opera entre 4V y 30V. Como nuestra placa de desarrollo posee una entrada analógica (A0), la usaremos para conectar dicho sensor. Se le añade un condensador de 100nF en paralelo

en los bornes de alimentación del sensor para que no se produzcan caídas de tensión en el LM35 y la señal que ofrece sea más fiable.

El sensor de temperatura tendrá tres diodos LED para indicar el estado en el que se encuentra el dispositivo periférico. Serán de tres colores diferentes, amarillo, verde y rojo. Será necesario añadirle una resistencia a cada uno.

El circuito tendrá un pulsador para cambiar de modo en el sensor de temperatura, pero esto se explicará más adelante en el apartado de *software*.

En el circuito habrá dos tensiones diferentes, por un lado, estará la tensión de alimentación del *NodeMCU*, que será de 5V, y por otro lado, está la tensión de alimentación para el resto de los componentes eléctricos y electrónicos, que será de 3.3V. Esta tensión de alimentación de los componentes se puede coger de los pines que proporciona el propio *NodeMCU* o de una fuente externa. En nuestro caso se utiliza una fuente de alimentación externa que nos proporciona las dos tensiones y se alimenta al circuito como se puede ver en la Ilustración 30. En la entrada de alimentación de la placa de desarrollo se coloca un condensador para evitar que las caídas de tensión alteren el correcto funcionamiento de este.

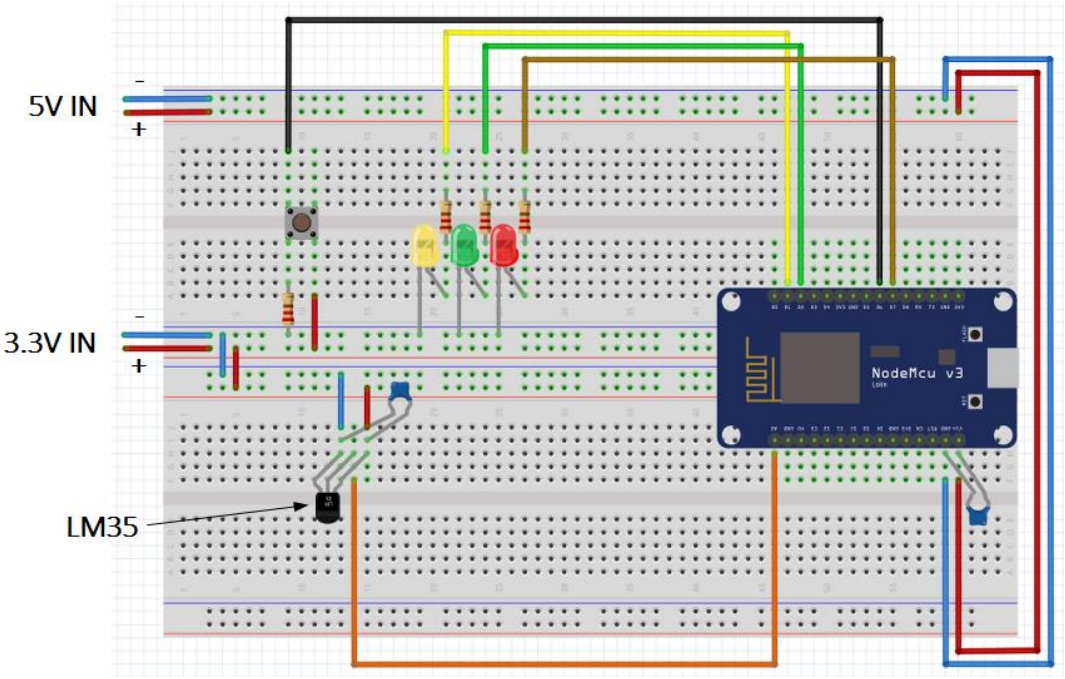

Ilustración 30. Circuito del sensor de temperatura.

4.1.2.2. Actuador

El actuador es el periférico encargado de accionar la calefacción de la casa en función de la señal de mando, a la que está suscrita, y la enciende o apaga.

De la misma forma que con el sensor de temperatura, el actuador usa la placa de desarrollo *NodeMCU* como elemento principal de computo, para

implementar el programa que llevará a cabo las tareas que se le asignen a este dispositivo. A dicha placa se le conectan elementos eléctricos y electrónicos para conseguir el funcionamiento deseado.

Los elementos necesarios serán los siguientes:

- NodeMCU.
- Relé SRD-05VDC-SL-C.
- Resistencias.
- Diodos LED.
- Pulsador.
- Condensadores.
- Fuente de alimentación 5V y/o 3.3V.
- Cables.
- Placas de prototipado.

El actuador tiene que accionar la calefacción, y para ello utilizaremos el relé SRD-05VDC-SL-C[12]. Con el circuito de control se le dará la señal de mando al relé y este será el encargado de accionar la caldera. Para facilitar el montaje, se usará un módulo que incorpora el relé, además facilita el acceso a los pines y viene con unas bornas para conectar la caldera. Dicho modulo se puede apreciar en la Ilustración 31.

Ilustración 31. Módulo relé SRD-05VDC-SL-C.

A modo de indicador luminoso del estado de la caldera, si está encendida o no, se conecta un diodo LED con su respectiva resistencia. De esta forma con un simple vistazo sabremos si está encendida o no.

El actuador también tendrá tres diodos LED para indicar el estado en el que se encuentra el dispositivo periférico. Serán de tres colores diferentes, amarillo, verde y rojo. Será necesario añadirle una resistencia a cada uno.

El circuito tendrá un pulsador para cambiar de modo en el actuador, pero esto se explicará más adelante en el apartado de *software*.

El montaje del circuito del prototipo tiene como resultado un esquema que podemos apreciar en la Ilustración 32. En esa imagen podemos ver el conexionado de los elementos eléctricos y electrónicos que se han necesitado.

Al igual que en el sensor de temperatura se coloca un condensador a la entrada de la alimentación del *NodeMCU* para que en caso de haber caídas de tensión no afecten el funcionamiento de la placa de desarrollo. La alimentación será de 5V para la placa y de 3.3V para el resto de las componentes eléctricos y electrónicos. Los 3.3V se obtienen de una fuente de alimentación externa para nuestro prototipo, pero podrían obtenerse de los pines que ofrece el *NodeMCU*.

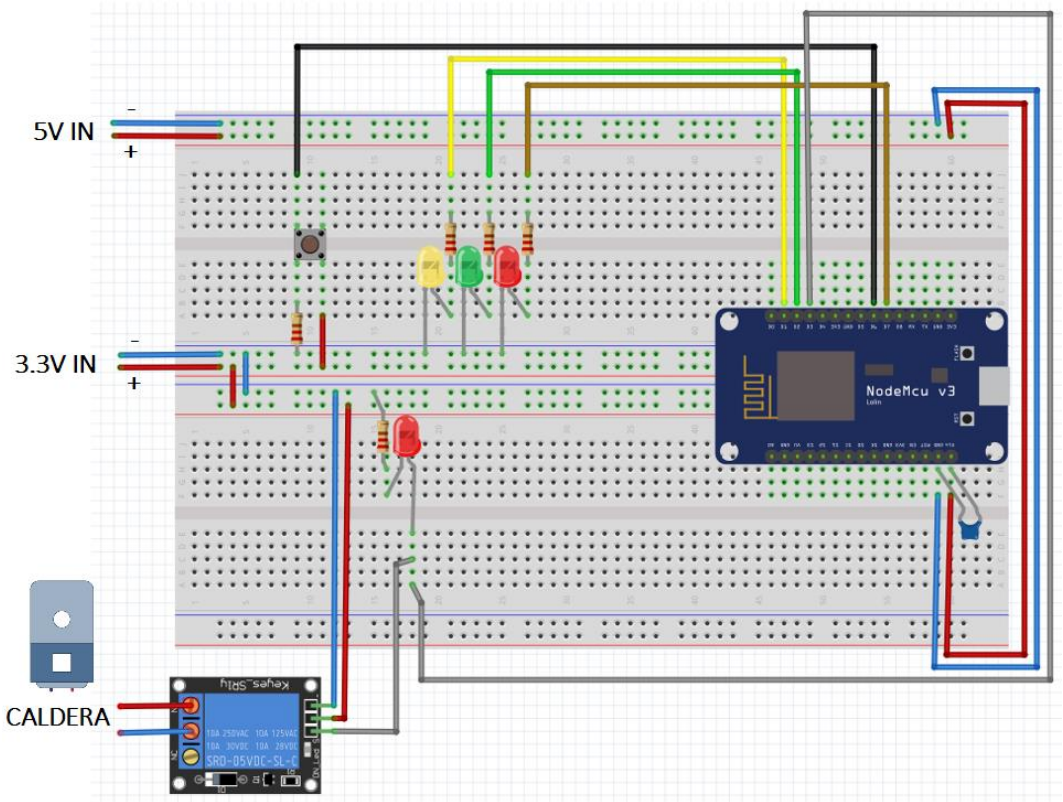

Ilustración 32. Circuito del actuador.

4.1.3. Teléfono móvil inteligente

El teléfono inteligente es el elemento que tendrá instalada una aplicación que sirve de interfaz de usuario para interactuar con los dispositivos del sistema domótico, configurarlos, obtener información valiosa de ellos en tiempo real que sirva para tomar decisiones y tener el control de la casa.

En el mercado hay una amplia variedad de modelos de teléfonos inteligentes, pudiéndose clasificar por rango de precios, especificaciones técnicas, sistema operativo que emplean, etc.

Puesto que la aplicación no será muy pesada y no demandará muchos recursos, las especificaciones técnicas del dispositivo no necesariamente tienen que ser muy altas y cualquier teléfono actual tiene la capacidad de soportar dicha aplicación.

Para este proyecto se va a utilizar un teléfono con sistema operativo *Android* puesto que es el dispositivo que se tiene actualmente, sin embargo, se podría usar un teléfono con iOS o cualquier sistema operativo.

El teléfono en el que se va a implementar la aplicación es un Xiaomi Mi 9 con sistema operativo Android ya que es del que se dispone, pero la aplicación que se va a desarrollar podría ser instalada en cualquier teléfono *Android*.

Ilustración 33. Teléfono móvil Xiaomi Mi 9.

4.1.4. Arquitectura de red

Todos los dispositivos tienen que comunicarse para intercambiar información en forma de mensajes para poder llevar a cabo las tareas que tienen asignadas. Para ello se conectan a la red WiFi de casa y se utilizara el protocolo MQTT, el cual consta de uno o varios clientes y un intermediario entre estos, conocido como *broker*.

Un aspecto importante es definir la red de comunicación entre ellos de tal forma que tengan acceso al *broker* en todo momento, puesto que es la pieza clave para la comunicación. Filtra los mensajes de los publicadores y los redirige a los clientes suscriptores.

En nuestro caso el *broker* del que hablamos ira instalado en el ordenador que hace las veces de controlador central. Como el ordenador está en la red WiFi de casa todos los dispositivos que estén conectados en la misma red podrán verle. Este ordenador tendrá asignada una dirección IP local, y por consiguiente el *broker* tendrá esa misma dirección para identificarse. De la misma forma que el ordenador, el sensor de temperatura y el actuador estarán conectados a la red WiFi de casa y tendrán su propia dirección IP local. Estos dispositivos podrán acceder al *broker* con la dirección IP del ordenador que es la misma que la de este.

En definitiva, si se conoce la dirección IP local del ordenador en la red WiFi de casa, conocemos la dirección del *broker*, y los dispositivos o clientes que quieran conectarse a él podrán hacerlo siempre y cuando estén en la misma red WiFi.

En la Ilustración 34 vemos como los dispositivos interactúan con el *broker* gracias a la conexión WiFi de la casa donde están instalados.

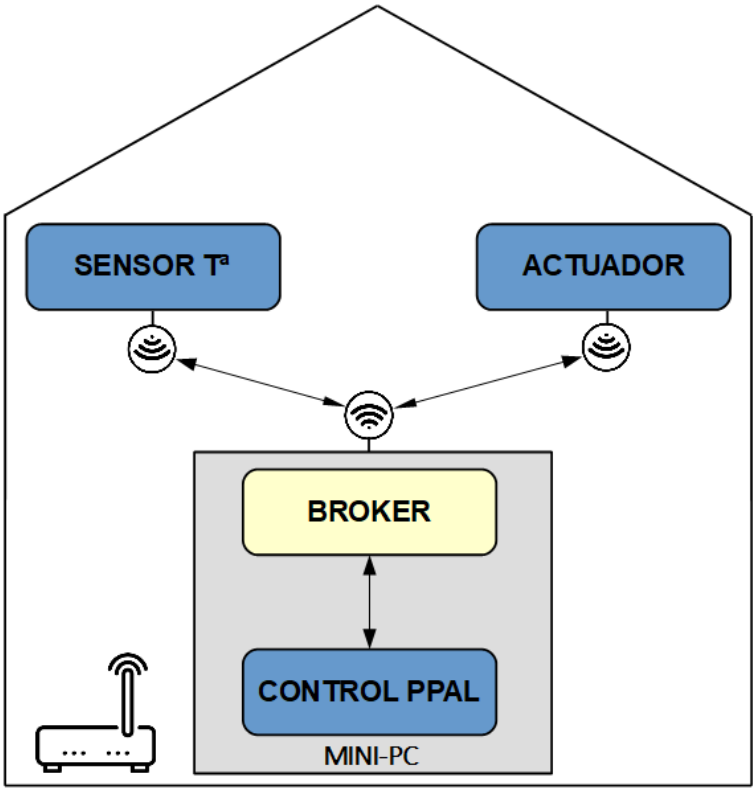

Ilustración 34. Arquitectura de red de la casa.

Hasta aquí no se había mencionado el teléfono inteligente en la arquitectura de red puesto que este dispositivo puede estar conectado a la red WiFi de casa, en el caso de que el usuario lo use dentro de casa, o a las redes móviles (3G, 4G, etc.) u otra red WiFi distinta, en el caso de estar fuera de casa.

Si el usuario se encuentra en casa y tiene su teléfono conectado a la red WiFi doméstica donde están conectados el resto de los dispositivos, la aplicación instalada en el teléfono tendrá acceso al *broker* de la misma forma que el actuador o el sensor de temperatura.

En el caso de que el usuario se encuentre fuera de casa y quiera hacer uso de la aplicación para consultar cualquier información o realizar alguna operación de control sobre el sistema, no podrá acceder conociendo la dirección IP local del ordenador y en consecuencia, del *broker*, ya que esa dirección no es visible del *router* para afuera, es decir no es accesible para dispositivos que no estén conectados a la red del *router*.

La dirección que es visible para cualquier dispositivo conectado a internet es la dirección IP pública del *router*. Esta dirección la asigna la compañía con la que se contrate los servicios de internet, es inequívoca y accesible desde cualquier parte del mundo. Por lo que la aplicación del teléfono móvil tendrá que conocer esta dirección pública para acceder a la red de la casa, pero todavía no tendrá acceso al *broker*.

Para tener acceso a la dirección IP local del ordenador desde cualquier parte del mundo es necesario hacer una traducción de la dirección de red, o lo que es lo mismo un NAT (del inglés *Network Address Translation*). Se conoce que el puerto por el que se tramita la comunicación MQTT es el 1883, por lo que el NAT redirigirá de la IP pública del *router* a la IP local del ordenador cuando reciba una petición por el puerto 1883. Para esto hay que acceder al menú de configuración del *router* y asignar dicha redirección de puertos, tal y como se puede observar en la Ilustración 35.

Ilustración 35. Redirección de puertos (NAT) en el router.

Una vez esté configurado el *router* con dicha redirección de las direcciones cuando se accede por el puerto 1883, que es el del protocolo MQTT, el teléfono móvil se conectará a internet desde cualquier parte del mundo y conociendo la IP pública del *router* tendrá acceso al *broker,* ya que el *router* se encarga de redirigir la conexión hacia la IP local del ordenador, como se ve en la Ilustración 36.

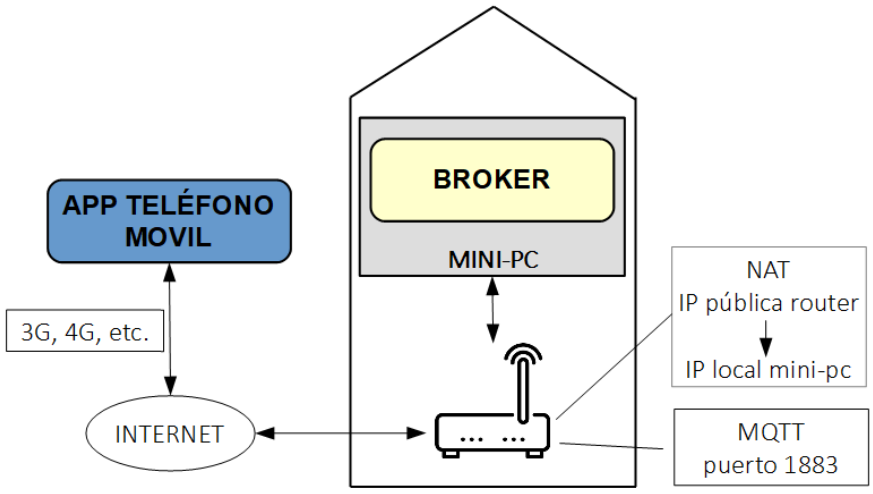

Ilustración 36. Arquitectura de red con el teléfono móvil.

4.2. SOFTWARE

4.2.1. MQTT

4.2.1.1. Broker

El *bróker* es el intermediario encargado de filtrar los mensajes recibidos por los clientes publicadores en función de los temas de publicación y redirigirlos a los clientes suscriptores de dichos temas.

Para nuestro proyecto utilizaremos Mosquitto[6] como intermediario del protocolo MQTT. Mosquitto es un *broker* muy ligero y de código abierto que soporta las últimas versiones del protocolo MQTT. Se puede instalar en diversos sistemas operativos, como Windows, Linux, Debian, etc. y ser usado en casi cualquier dispositivo, ya sea un ordenador o dispositivos con recursos más limitados, por ejemplo, una *Raspberry Pi*.

El dispositivo donde se ejecutará el *broker* Mosquitto tiene que estar disponible y accesible durante todo el día, es por eso por lo que se instala en el ordenador que realiza la función de controlador central. Este ordenador tiene como sistema operativo *Linux UBUNTU 18.04 LTS* y le descargaremos e instalaremos Mosquitto directamente desde la terminal del ordenador como se puede ver en la Ilustración 37.

```
apt-get install mosquitto
```

Ilustración 37. Instalar Mosquitto a través del terminal de Ubuntu.

Este *broker* utiliza el puerto 1883 para las comunicaciones, el cual no está encriptado. Para utilizar comunicaciones más seguras se usa el 8883. Nosotros utilizaremos el 1883 porque para nuestro prototipo no se ha implementado la encriptación de mensajes.

Para acceder al *broker* y establecer una conexión con los clientes tienen que conocer su dirección IP, que es el identificativo del que se sirven para estos para conectarse al intermediario. Como se ha comentado el nuestro se ejecuta en el ordenador, puesto que nuestro *broker* tendrá la misma dirección IP de este.

4.2.1.2. Clientes

Los clientes de nuestro sistema domótico son cuatro: sensor de temperatura, actuador, aplicación del teléfono móvil y el software encargado del control en el ordenador, es decir, el control principal. Sin embargo, el sensor de temperatura y el actuador utilizan la misma placa de desarrollo, por lo que son tres los tipos de cliente que cohabitan en el proyecto.

El sensor de temperatura y el actuador utilizan la misma placa de desarrollo *NodeMCU*, por lo que la explicación del cliente MQTT será aplicable para ambos dispositivos.

Para programar la placa de desarrollo se utilizará el entorno de desarrollo de Arduino, que se puede descargar de su web[13]. Este IDE posee de varias librerías que implementan clientes MQTT, por lo que resulta muy sencillo implementar un cliente en el *NodeMCU*. La librería que nosotros vamos a utilizar para poder implementar el cliente MQTT tanto en el actuador como en el sensor es: *PubSubClient*. Esta librería es compatible con ESP8266[14], por lo que también es compatible con *NodeMCU*.

En la Ilustración 38 podemos ver el gestor de librerías, dónde se descarga e instala la librería *PubSubClient*. Para implementar el cliente MQTT tenemos que incluirla en el código de nuestro programa y el dispositivo estará disponible para enviar y/o recibir mensajes.

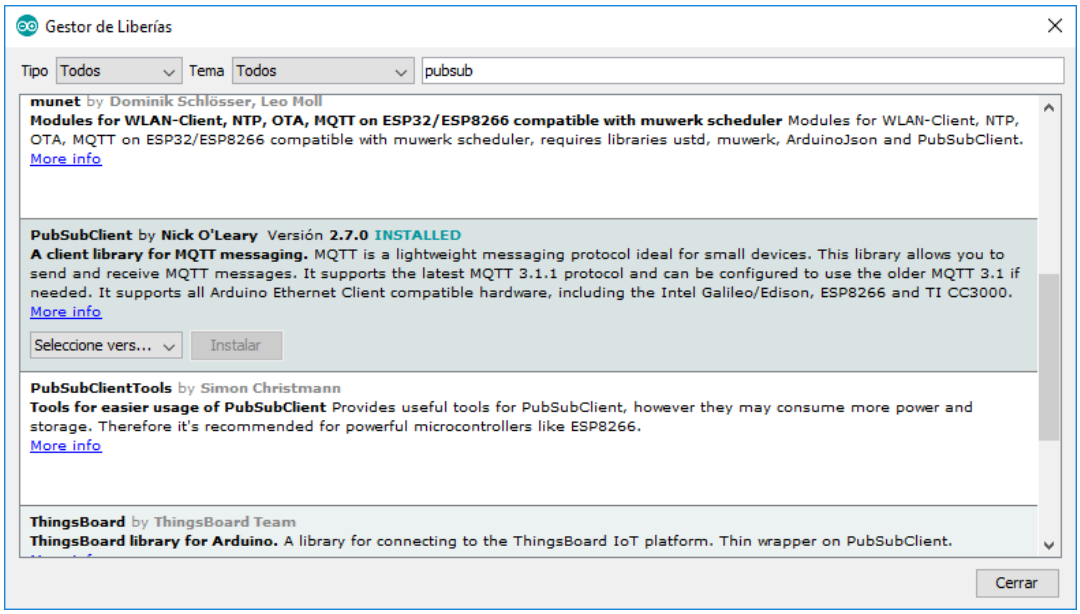

Ilustración 38. Instalación librería PubSubClient en el IDE de Arduino.

El teléfono móvil que se usa para este proyecto es un *Xiaomi* que tiene sistema operativo Android, por lo que se usará el entorno de desarrollo que proporciona Google para construir sus aplicaciones. Este entorno de desarrollo o IDE es *Android Studio* y se puede descargar desde la página *Developers*[15].

En este entorno de programación vamos a usar *Eclipse Paho Android Service*[16] que es una librería escrita en Java que permite implementar un cliente MQTT en las aplicaciones de Android. Para añadir la librería a nuestro proyecto de *Android Studio* debemos añadir las líneas de código que se muestran en la Ilustración 39 en las dependencias del archivo build.gradle.

```
implementation 'org.eclipse.paho:org.eclipse.paho.android.service:1.1.1'
implementation 'org.eclipse.paho:org.eclipse.paho.client.mqttv3:1.2.0'
```

Ilustración 39. Añadir la librería Eclipse Paho Android Service al proyecto.

Para poner en marcha el servicio del cliente MQTT es necesario añadir la línea de código que se muestra en la Ilustración 40 para declararlo en el archivo AndroidManifest.xml.

```
<service android:name="org.eclipse.paho.android.service.MqttService" />
```

Ilustración 40. Declaración del servicio MQTT en Android Studio.

Para que pueda enviar y recibir mensajes es necesario otorgarle a la aplicación de ciertos permisos. Estos permisos son necesarios para que la aplicación funcione correctamente y pueda tener acceso a ciertas funcionalidades del teléfono, y son los que se muestran en la

```
<uses-permission android:name="android.permission.WAKE_LOCK" />
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
<uses-permission android:name="android.permission.READ_PHONE_STATE" />
```

Ilustración 41. Permisos necesarios para Eclipse Paho Android Service.

El agente encargado del control principal se implementa en el mini-pc. Se va a usar el entorno de desarrollo *Code::Blocks*, ya que es una plataforma libre y soporta el lenguaje de programación C++, que es el que se va a usar para desarrollar el programa del control principal.

En primer lugar, vamos a descargar e instalar las librerías de desarrollador para clientes MQTT a través de la línea de comandos del terminal como se puede ver en la Ilustración 42.

```
apt-get install libmosquitto-dev
```

Ilustración 42. Instalar librerías desarrollador de clientes MQTT.

Una vez instaladas las librerías, es necesario incluir la librería que necesitamos en el proyecto de *Code::Blocks* en C++. Para ello hay que configurar el compilador para que incorpore esa librería como se puede ver en la Ilustración 43.

Como la librería *libmosquitto* se desarrolló en C y nosotros queremos implementar el programa en C++, vamos a crear una clase envoltorio que herede de la clase en C y exponga los métodos C++ para un uso más fácil de la misma. En esta librería existe una clase básica en C++ llamada *mosquittopp*, y es la que vamos a coger como base para heredarla y personalizarla a nuestro gusto con nuestras necesidades para implementar la clase envoltorio que finalmente usaremos.

En la Ilustración 44 podemos ver como creamos una nueva clase envoltorio que llamaremos *mosquittoMQTT* y que hereda de la clase *mosquittopp.*

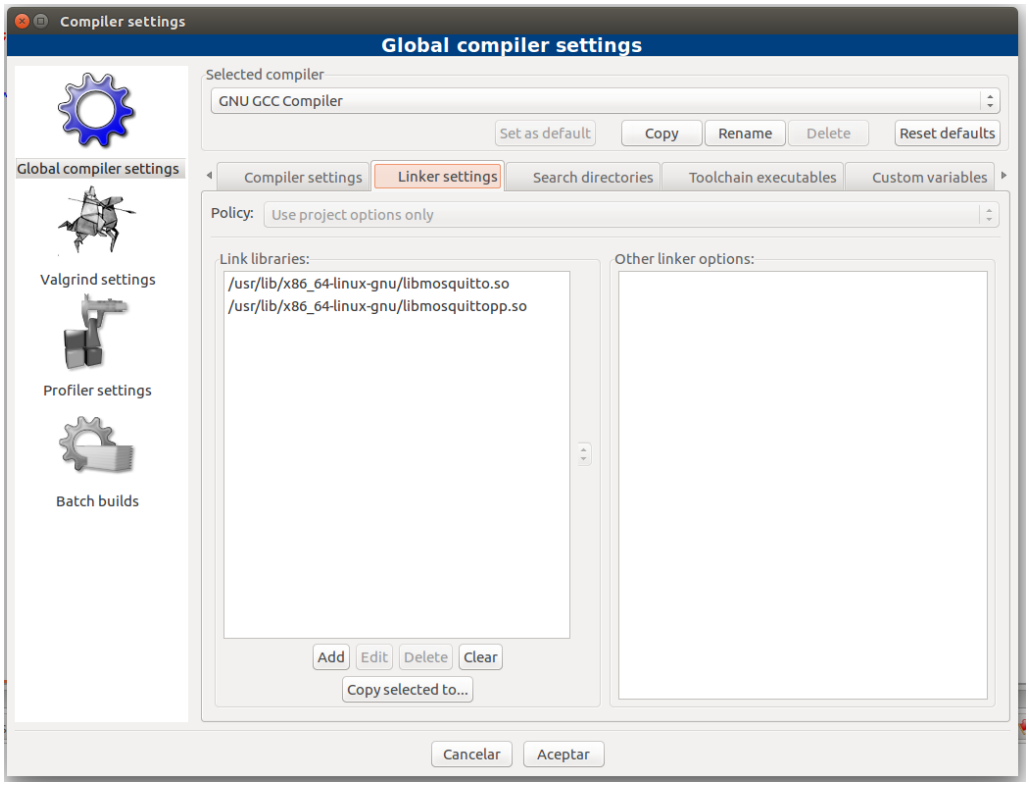

Ilustración 43. Incorporar librerías cliente MQTT en Code::Blocks.

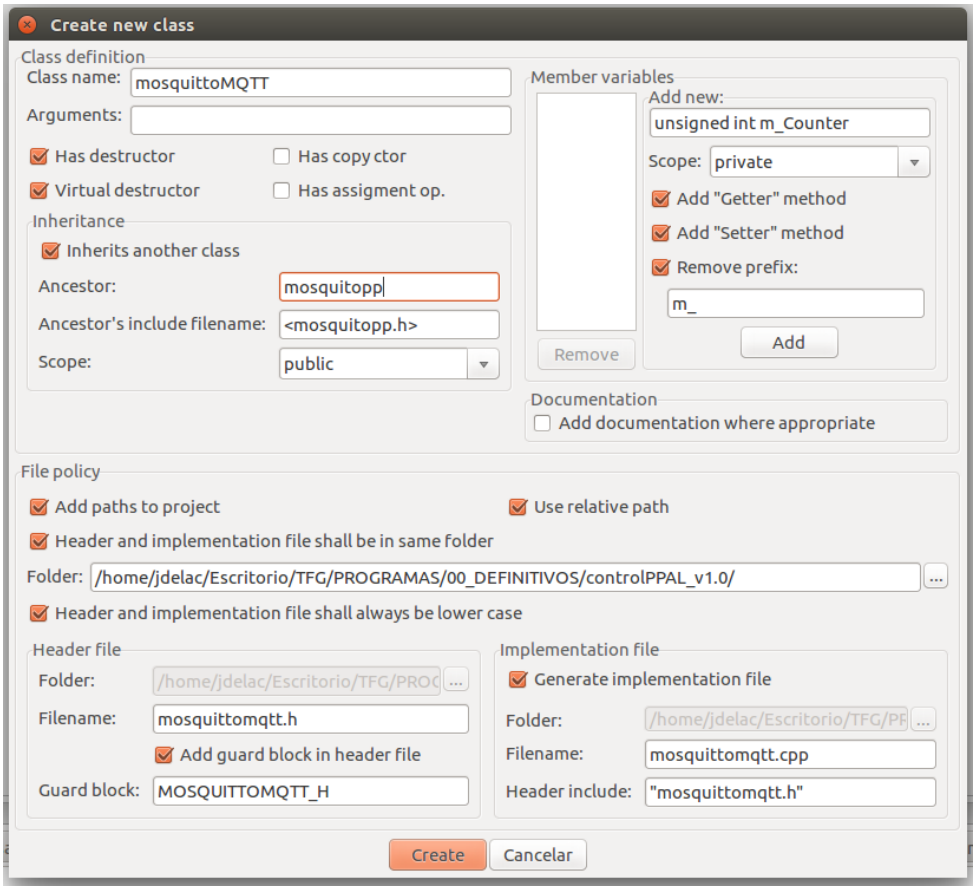

Ilustración 44. Creación de la nueva clase envoltorio en C++ de MQTT.

4.2.1.3. Estructura de topics

Como ya se ha comentado, el protocolo MQTT utiliza temas o *topics* para que el *broker* pueda filtrar los mensajes y estos lleguen a los clientes que están suscritos a dichas temáticas. Es importante crear una estructura que permita escalar el sistema, permitiendo crecer o degradarse en un futuro sin comprometer al resto de dispositivos.

Los *topics* que se usan en este proyecto se muestran en la Ilustración 45. En esa estructura se puede ver que los temas están jerarquizados y divididos en tres grandes grupos, que a su vez se pueden volver a dividir en subgrupos como es el caso del "CONTROL_PPAL". Está diseñada para que se puedan añadir más dispositivos en un futuro de una forma elegante sin comprometer a los demás.

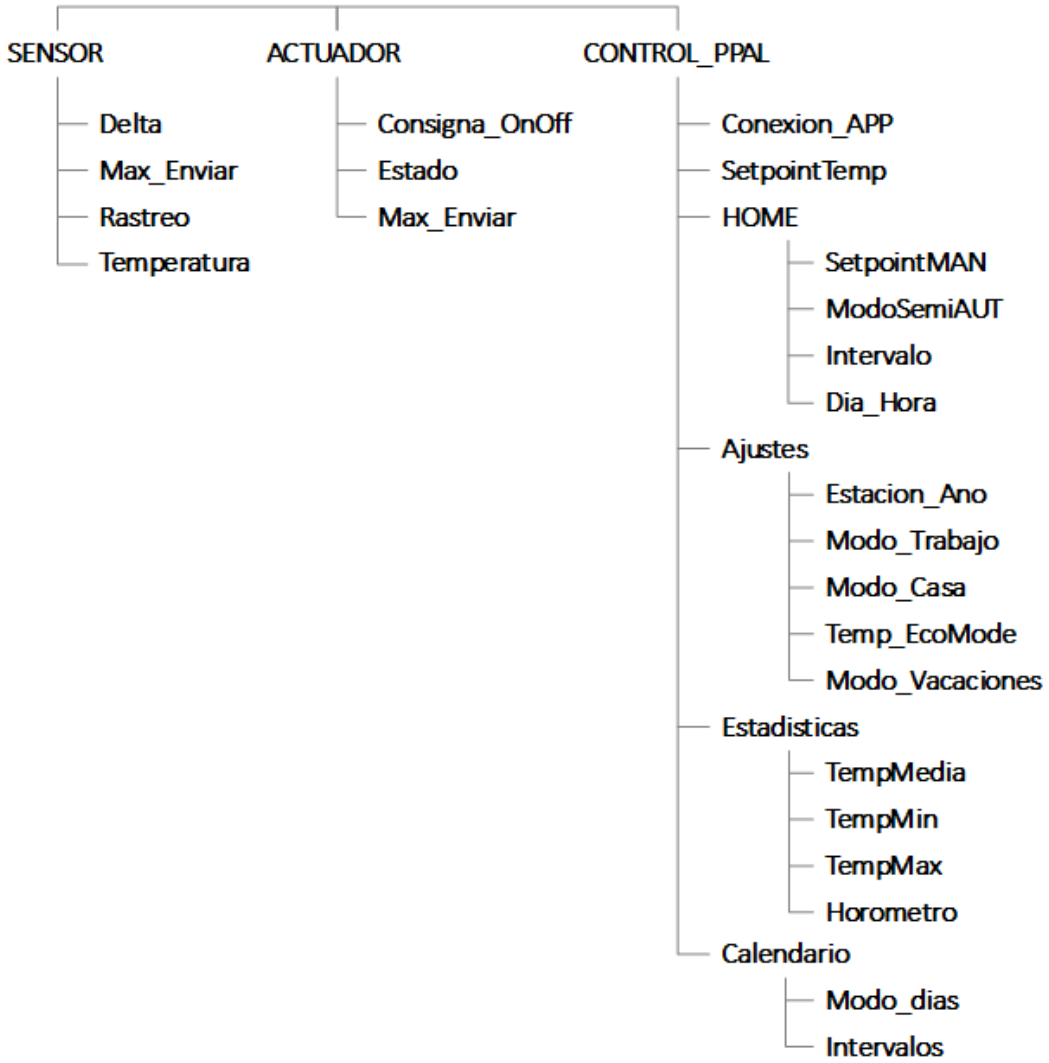

Ilustración 45. Estructura de topics.

4.2.2. Control principal

El control principal es un agente software encargado del control de temperatura del sistema domótico, que está implementado en el ordenador. Recibirá la

temperatura del sensor y el *setpoint* de la aplicación Android del teléfono móvil y en función de esos valores encenderá o apagará la caldera.

El programa del control principal se va a desarrollar en el lenguaje de programación C++, para lo cual nos hemos servido de libros como *"C++ Concurrency in Action"*[17] y *"Real-Time C++: Efficient Objetc-Oriented and Template Microcontroller Programming"*[18]. El entorno de desarrollo que se ha elegido es *Code::Blocks*, que es un IDE de código abierto, y soporta múltiples compiladores.

El proyecto de *Code::Blocks* se llamará *controlPPAL* y en él encontraremos tres archivos importantes: main.cpp, mosquittomqtt.h y mosquittomqtt.cpp.

En los archivos mosquittomqtt se define la clase envoltorio que integra la librería de libmosquitto en C++ para implementar el cliente MQTT, que es mostquittopp. En el archivo mosquittomqtt.h se definen las variables que va a usar esta clase y los métodos que se van a emplear. En el archivo mosquittomqtt.cpp se implementan los métodos que se han definido en el archivo .h. Esta clase será la encargada de conectarse con el *broker*, suscribirse a los *topics* y gestionar la llegada de mensajes de las temáticas a las que se ha suscrito. En el archivo main.cpp se implementa el funcionamiento principal del programa.

El programa lo que hace nada más iniciarse es establecer la conexión con el *broker*, que en nuestro caso como esta en el mismo ordenador que el programa le podemos hacer la llamada con *localhost*, es decir, servidor local o *broker* local. La conexión se establece por el puerto 1883. Una vez se establece la conexión con el *broker* se suscribe a los *topics*:

- CONTROL_PPAL/Conexión_APP
- CONTROL_PPAL/SetpointTemp
- SENSOR/Temperatura
- CONTROL_PPAL/Home/ModoSemiAUT
- CONTROL_PPAL/Estadísticas/TempMedia
- CONTROL_PPAL/Estadísticas/TempMin
- CONTROL_PPAL/Estadísticas/TempMax
- CONTROL_PPAL/Estadísticas/Horometro
- CONTROL_PPAL/Ajustes/Estacion_Ano
- CONTROL_PPAL/Ajustes/Modo_Trabajo
- CONTROL_PPAL/Ajustes/Modo_Casa
- CONTROL_PPAL/Ajustes/Temp_EcoMode
- CONTROL_PPAL/Ajustes/Modo_Vacaciones
- CONTROL_PPAL/Calendario/Modo_dias
- CONTROL_PPAL/Calendario/Intervalos

Una vez se terminan las suscripciones el programa está a la espera de recibir los valores de la temperatura, *setpoint* y configuración de los modos de funcionamiento.

El programa comparará la temperatura con el *setpoint*, en caso de ser mayor que este la calefacción se apagará y en caso de ser menor se encenderá. Para enviar la orden de mando sobre el actuador se publica un 0 para apagar y un 1 para encender sobre el *topic*:

- ACTUADOR/Consigna_OnOff

Cuando se enciende la caldera el programa contabilizará el tiempo que está en funcionamiento y publicará el valor cada minuto en el siguiente *topic:*

- CONTROL_PPAL/Estadísticas/Horometro

Una de las funcionalidades del programa es que el *setpoint* varíe con el tiempo en función de un calendario programado. Se reciben los *setpoint* y las horas de comienzo de cada intervalo, además, como cada día puede tener dos modos de funcionamiento programados: diario y festivo; también se recibe el modo seleccionado para cada día. El programa utilizara estos datos para saber que *setpoint* debe fijar en función del día, hora y el modo configurado. Una vez se establezca el intervalo y modo en el que nos encontramos actualmente se publicará en el *topic*:

- CONTROL_PPAL/Home/Intervalo

La hora de referencia es la del ordenador, por lo que, se envía a los clientes que la necesiten publicando en el siguiente *topic*:

- CONTROL_PPAL/Home/Dia_Hora

Otras funcionalidades son las de los modos de funcionamiento, manual o automático, si estas dentro o fuera de casa, etc., los cuales se explican más detalladamente en el apartado 3. El programa comparará en qué modo se encuentra y funcionará de una forma u otra.

Cada vez que se reciba un valor de temperatura comparará con los valores máximo y mínimo, y si es mayor que el máximo actualizará esa variable y si es menor que el mínimo actualizará la otra, pero si está comprendido entre esos valores esas variables se quedaran como tal. También se calculará el valor de la temperatura media. Se publicarán los valores una vez se hayan comparado con el máximo y mínimo y se haya calculado la media en los siguientes *topic*s:

- CONTROL_PPAL/Estadísticas/TempMedia
- CONTROL_PPAL/Estadísticas/TempMin
- CONTROL_PPAL/Estadísticas/TempMax

4.2.3. Periféricos

Los dispositivos periféricos son todos aquellos que se añaden al controlador principal para formar el sistema domótico. En nuestro caso se lleva a cabo un termostato inteligente, por lo que los periféricos son el sensor de temperatura y el actuador.

Los dos dispositivos utilizan la misma placa de desarrollo *NodeMCU* por lo que el desarrollo de los programas para ambos será en el mismo IDE. Además, compartirán parte del software, el cual será idéntico con matices y se diferenciará en la funcionalidad específica a la que está destinado el uno y otro.

Para desarrollar los programas de estos dispositivos se va a usar el entorno de desarrollo de Arduino, el cual es de código abierto y la programación es similar al C. El *NodeMCU* integra el ESP8266 y Arduino lo soporta, sin embargo, no viene por defecto incorporada en el IDE y hay que descargar e incluir la tarjeta en el entorno de desarrollo. Como se ve en la Ilustración 46, en el gestor de URLs adicionales de tarjetas dentro de preferencias se incluye la siguiente dirección:

http://arduino.esp8266.com/stable/package_esp8266com_index.json

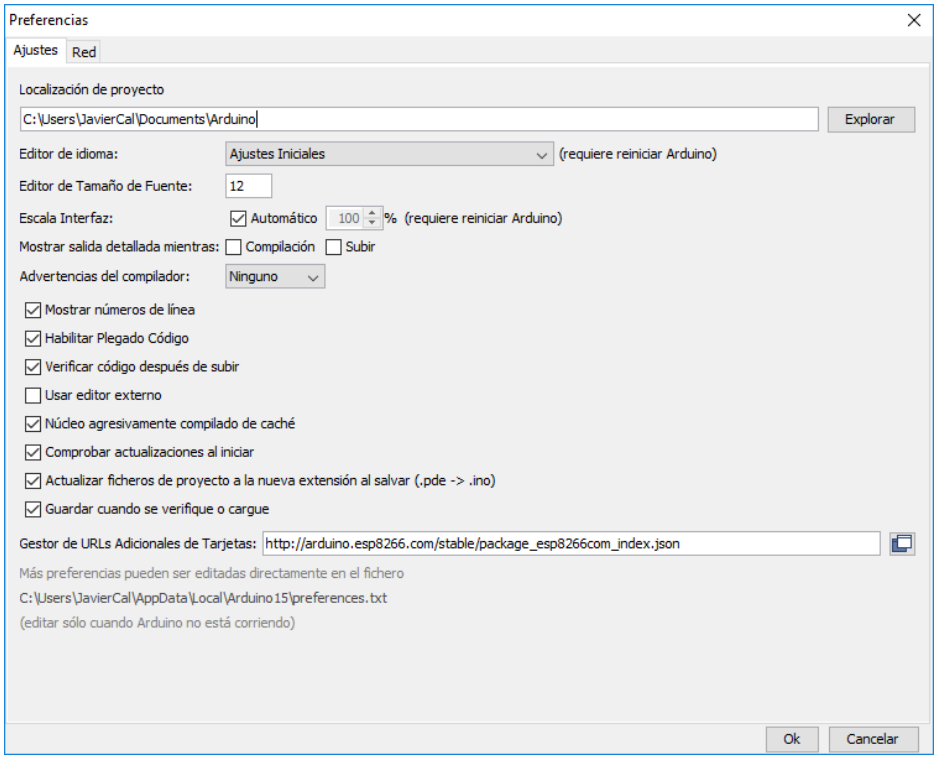

Ilustración 46. Gestor de URLs adicionales de tarjetas.

Una vez se haya instalado hay que descargar la tarjeta desde el gestor de tarjetas en herramientas, como se puede ver en la Ilustración 47. Una vez

se instale esto el entorno de desarrollo está listo para poder empezar con la programación.

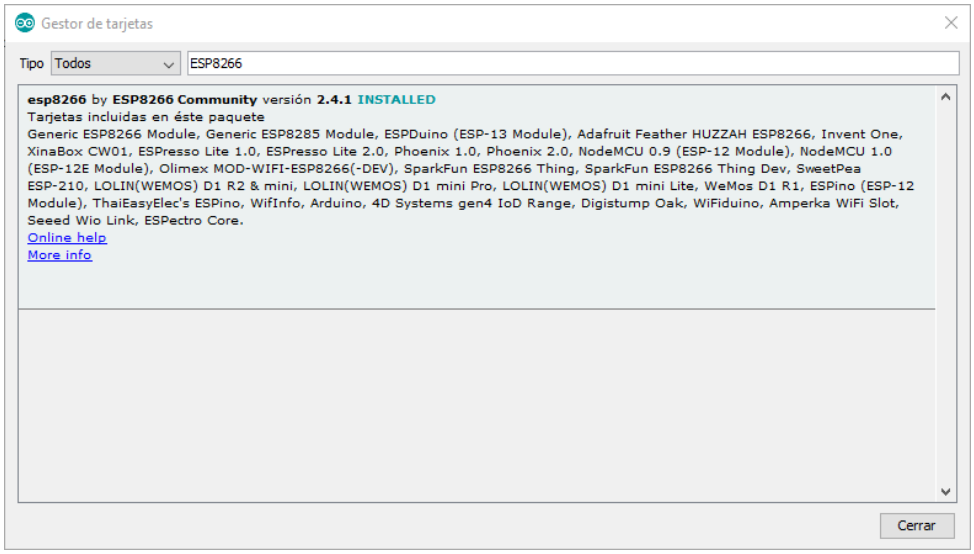

Ilustración 47. Añadir ESP8266 en el gestor de tarjetas Arduino.

Los dos dispositivos para comunicarse con el sistema domótico a través del protocolo MQTT tienen que conectarse a internet. Como la placa de desarrollo tiene conectividad WiFi, la utilizarán como medio de conexión a internet. Para ello es necesario que la placa conozca las credenciales de la red WiFi a la que se van a conectar los dispositivos. Se podrían introducir programáticamente en el código y la placa se conectaría a internet a través de la red fijada, sin embargo, esto no es muy versátil y provoca problemas si se cambia de usuario y contraseña porque tendría que volver a introducirlos en el programa y volver a cargarlo en la placa.

Para solventar el problema de facilitarle las credenciales de la red WiFi se implementa un procedimiento de conexión de la placa en el que el usuario podrá introducir las credenciales de la red WiFi que desee y así acceder a internet.

Lo primero que realiza la placa al encenderse es comprobar si tiene almacenado o no en la memoria EEPROM las credenciales de alguna red WiFi introducida anteriormente. Si no tiene ningún valor almacenado en ella la placa generara un punto de acceso (AP) WiFi. Este punto de acceso o red WiFi que genera el dispositivo tendrá unas credenciales conocidas, que irían indicadas en el propio dispositivo.

Credenciales punto de acceso dispositivos		
Dispositivo	SSID	Contraseña
Sensor de temperatura	TempSensor_WiFi_setup	0123456789
Actuador	Actuador_WiFi_setup	0123456789

Tabla 4. Credenciales punto de acceso periféricos.

Buscando las redes WiFi que aparecen en la Tabla 4 e introduciendo la contraseña correspondiente que aparece también en dicha tabla, estaremos conectados al punto de acceso que están generando estos dispositivos.

Para facilitar la conexión al punto de acceso y que no haga falta buscar las redes WiFi que generan los dispositivos, se generan dos códigos QR, que almacenan la información de las credenciales de dichos puntos de acceso.

Cuando el usuario quiera conectarse al punto de acceso le bastará con escanear los códigos QR y el teléfono móvil se conectará automáticamente a la red WiFi que contiene dicho código.

Los códigos QR se generan con la página web "Unitag QR"[19], y con ellos se diseña una etiqueta que iría pegada en los dispositivos, tanto en el sensor como en el actuador.

La etiqueta con el código QR para el sensor de temperatura se muestra en la Ilustración 48.

Ilustración 48. Etiqueta código QR del sensor de temperatura.

La etiqueta con el código QR para el actuador se muestra en la Ilustración 49.

Ilustración 49. Etiqueta código QR del actuador.

Las placas *NodeMCU* funcionarán como servidor web y alojarán una página web de configuración de la conexión WiFi, donde se podrá introducir las credenciales de la red WiFi de casa a la que se quiere conectar. Cada dispositivo tendrá su propia página de configuración que será distinta estéticamente, pero que el funcionamiento es el mismo. En la Ilustración 50 se muestra como quedan las páginas web para el actuador y para el sensor de temperatura. Para acceder a esta página web bastará con estar conectado al punto de acceso del sensor o del actuador e introducir en cualquier navegador web del móvil la dirección http://192.168.4.1. Esta dirección es así porque es la dirección IP que la tarjeta se asigna a si misma cuando crea el AP. Para facilitar el acceso a dichas páginas web se crea un portal cautivo en las placas de los dispositivos, que cuando se conecten a las redes WiFi que han generado el sensor y/o el actuador, dichas páginas web aparecerán automáticamente, y no hará falta conocer la dirección IP de los dispositivos.

Ilustración 50. Páginas web de configuración del Actuador y Sensor.

Podemos apreciar que las dos páginas web contienen los mismos elementos, pero con una estética diferente para cada una de ellas. En estas páginas podremos introducir las credenciales WiFi de la red de nuestra casa para que

el sensor y el actuador se conecten a internet y puedan comunicarse con el sistema domótico. Los elementos que aparecen son: SSID, contraseña y *broker*. El SSID es el identificativo de la red WiFi de la casa y en la página se muestra una lista de las redes disponibles para seleccionar la que corresponda a la de nuestra casa. La contraseña es la de la red WiFi de nuestra casa. El identificativo del *broker* será la dirección IP local que tiene el ordenador, y por consiguiente el *broker* de nuestro sistema. Al darle al botón de enviar las credenciales se enviarán a la placa *NodeMCU*.

Una vez haya recibido las credenciales WiFi de la red a la que se quiere que se conecten los dispositivos (sensor y actuador) se almacenarán en la memoria EEPROM, la cual es una memoria no volátil y dichos valores permanecerán guardados incluso después de apagar el dispositivo. Una vez guardados las credenciales la placa eliminará el punto de acceso que había creado y cambia el modo del módulo WiFi a modo estación (STA) para conectarse a internet.

En el modo STA la placa tratará de conectarse a la red WiFi que se le ha facilitado. Si no consigue conectarse con esas credenciales en un periodo de 30 segundos volverá a crear el punto de acceso para que el usuario vuelva a introducir los datos de la red WiFi para volver a intentar la conexión a internet.

Una vez se conecte a la red WiFi de casa el dispositivo establecerá la conexión MQTT con el *broker*. Tras esto el actuador y/o sensor estarán conectados con el sistema domótico y podrán llevar a cabo las tareas que tengan asignadas en sus programas.

Si cuando la placa se enciende tiene credenciales en la memoria EEPROM, intentará conectarse a la red WiFi que tiene almacenada en el modo STA, saltándose el paso del modo AP que crea el punto de acceso WiFi.

Ambos dispositivos tienen un pulsador que sirve para volver a configurar la red WiFi a la que se quiere que se conecte el dispositivo, ya sea el sensor y/o el actuador. Si presionamos el pulsador, parará de lo que esté haciendo y volverá a estar en el modo AP, el cual sirve para configurar la red WiFi a través del punto de acceso y las páginas web. Cuando la placa reciba las nuevas credenciales WiFi, sobrescribirá las que estaban anteriormente almacenadas en la memoria EEPROM y de esta forma quedarán guardadas las últimas.

Si estamos en el modo AP, es decir, con el punto de acceso WiFi creado y a la espera de recibir las credenciales, podemos borrar los datos que haya almacenados en la memoria EEPROM presionando el pulsador tres veces.

Para resumir el proceso que se ha explicado se muestra el esquema del ciclo de encendido del *NodeMCU* en la Ilustración 51. Tanto el del sensor de temperatura como el del actuador tienen el mismo ciclo de funcionamiento.

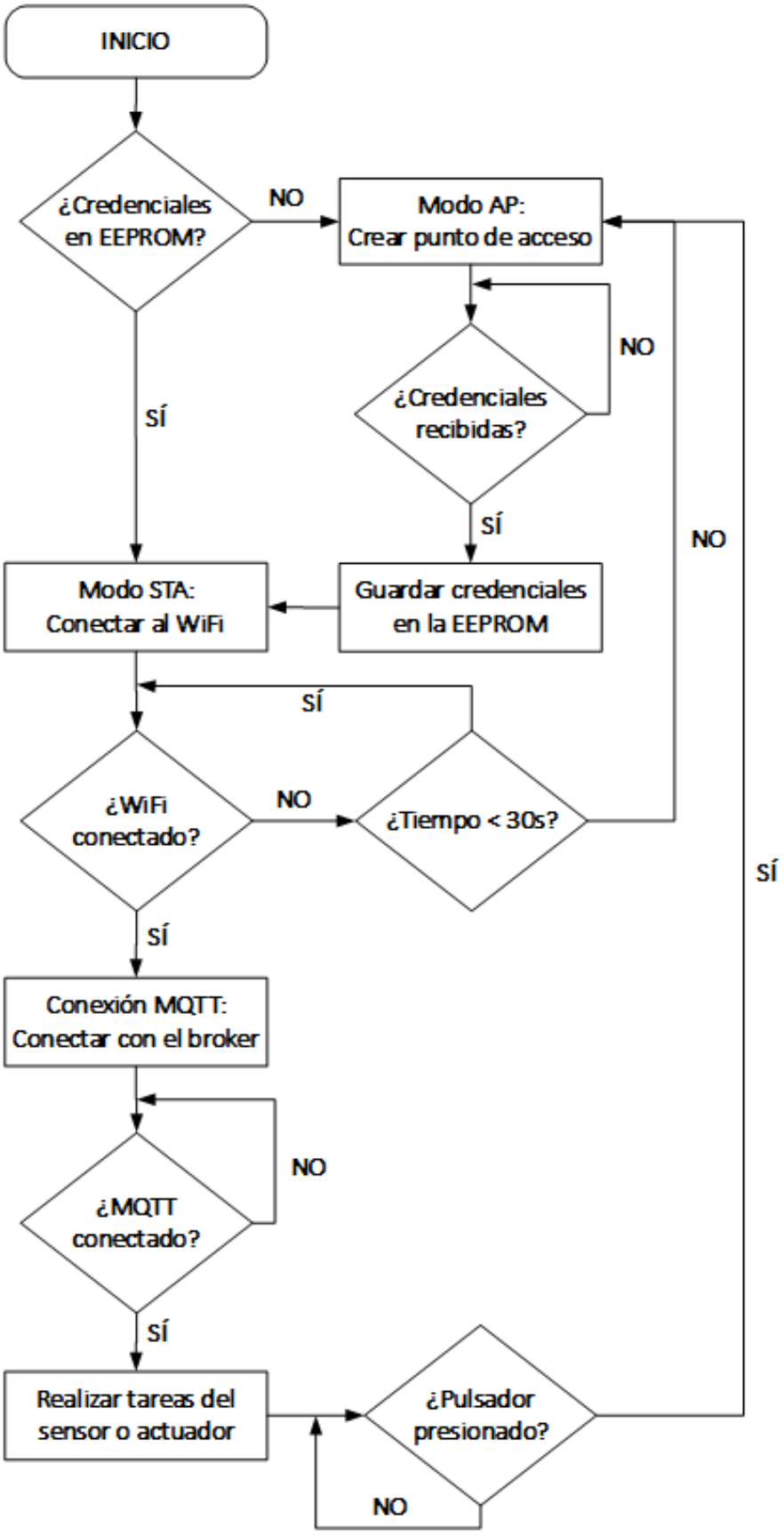

Ilustración 51. Ciclo de encendido y conexión del NodeMCU.

4.2.3.1. Sensor de temperatura

El sensor de temperatura es el encargado de realizar mediciones periódicamente y enviarlas cuando se produzca un cambio o cada cierto tiempo, lo que ocurra antes.

Cuando el dispositivo ya se ha conectado a internet a través de la red WiFi y ha conseguido establecer una conexión MQTT con el *broker*, se suscribirá a los siguientes *topics*:

- SENSOR/Delta
- SENSOR/Max_enviar
- SENSOR/Rastreo

La temperatura se lee periódicamente, es decir, cada cierto tiempo. Las mediciones tienen que representar fielmente la realidad, por lo que ese tiempo de rastreo, que el usuario tiene la posibilidad de configurar, tienen que ser lo más alto posible, teniendo como limitante que con las mediciones no se pierda información. En nuestro caso el tiempo de rastreo predefinido será de un segundo.

Para enviar la temperatura medida el sensor publicará un mensaje en el siguiente *topic*:

- SENSOR/Temperatura

El código del programa completo se encuentra en el apartado de Anexos.

4.2.3.2. Actuador

El actuador es el encargado de accionar la caldera, encendiéndola o apagándola en función de la señal de mando que recibe.

Cuando el dispositivo ya se ha conectado a internet a través de la red WiFi y ha conseguido establecer una conexión MQTT con el *broker*, se suscribirá a los siguientes *topics*:

- ACTUADOR/Consigna_OnOff
- ACTUADOR/Max_enviar

El actuador informará del estado en el que se encuentra, encendido o apagado, publicando un mensaje en el topic:

- ACTUADOR/Estado

El código del programa completo se encuentra en el apartado de Anexos.

4.2.4. Teléfono móvil inteligente

El teléfono móvil será el dispositivo que nos sirve para interactuar con el sistema. Gracias a él podemos acceder a las páginas web de configuración de

la red WiFi del sensor y actuador, pero, además se instalará una aplicación Android, ya que el dispositivo que se usa tiene ese sistema operativo. Dicha aplicación será la encargada de mostrarnos información del sistema y poder ejercer un control sobre este.

Para desarrollar la aplicación se usará el entorno de desarrollo *Android* Studio que ofrece Google, propietaria del sistema operativo Android. Este IDE es gratuito y está disponible para varios sistemas operativos, como *Windows*, *MacOS,* etc. Nosotros usaremos la versión de Windows. La programación puede ser en varios lenguajes como Java, Kotlin, C++, etc. Nosotros desarrollaremos la aplicación en Java porque hay una mayor comunidad de programadores en Java y a la hora de buscar información es más fácil.

Una aplicación Android se basa en las *Activity*, que son las pantallas que vemos en la app del móvil. Estas actividades se dividen en dos partes la clase java y el *layout* en xml. La clase java es la parte que se encarga del funcionamiento de la aplicación, que se programa en Java, y los *Layout* es la parte visual de la aplicación, es decir, lo que vemos cuando abrimos la app. En un *Activity* normalmente hay siempre una clase java y un *layout*, en el primero se programará las funciones que van a llevar a cabo y en la segunda se programa la estética de la interfaz que se muestra en el teléfono móvil.

Además de los *Activity*, están los *Fragment*, que no llegan a ser una actividad, pero funcionan como estas y también constan de una clase java y un *layout.* Estos fragmentos son trozos de aplicación que pueden ser llamados desde un *Activity* y permiten versatilidad a la hora de desarrollar la aplicación.

Las Activity de nuestro proyecto son:

- BienvenidaActivity
- DispositivosActivity

Los Fragment de nuestro proyecto son:

- HomeFragment
- CalendarioFragment
- EstadisticasFragment
- AjustesFragment

La estructura para nuestro proyecto es la que se muestra en la Ilustración 52, donde podemos ver como se divide en carpetas. Esta vista de la estructura es la vista *app*, que no es la forma de distribución de las carpetas en las que se guarda, pero facilita su distribución a la hora de programar y es más intuitivo en para el funcionamiento de la aplicación.

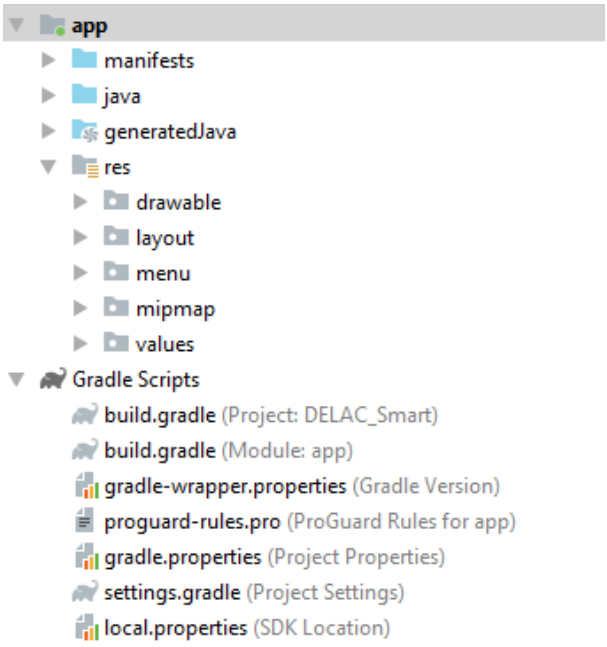

Ilustración 52. Estructura proyecto en Android Studio.

La carpeta de *manifests* contiene el archivo "AndroidManifest.xml", como se puede ver en la Ilustración 53, la cual contiene información esencial sobre la aplicación.

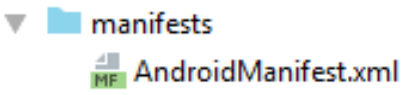

Ilustración 53. Carpeta manifests Android Studio.

En dicho archivo se declaran los permisos para acceder a ciertas funcionalidades del teléfono, que necesita la aplicación para funcionar correctamente. Se define también la versión mínima de Android API que requiere la aplicación. Se definen los componentes de la aplicación, tales como actividades (*activity*), servicios, etc.

También se define qué actividad será la principal y se mostrará al iniciar la aplicación. En nuestro caso, una vez se inicia la aplicación se mostrará una pantalla que se llama en inglés *splash screen*. Esta actividad tiene como objetivo servir de presentación de la aplicación mientras cargan las funciones principales de la app, es decir, sirve de bienvenida para el usuario mientras cargan el resto de las pantallas y así no tiene que esperar con una pantalla incompleta. Contiene una imagen y el título de la aplicación.

Además, se fija el nombre que tendrá la aplicación, el icono que tendrá esta y el tema de personalización que se usará. En nuestro caso la aplicación la llamaremos *DELAC Smart*, que proviene de mi apellido: de la Cal, y utilizamos la palabra inglesa *Smart* que significa inteligente. El icono estará formado por

una casa y un termómetro haciendo referencia a que sirve para controlar la temperatura de la casa. El icono quedaría como se muestra en la

Ilustración 54. Icono de la aplicación Android: DELAC Smart.

La carpeta java, la cual se muestra en la Ilustración 55, contiene las clases Java de las *Activity* o actividades que comentamos antes. Estos archivos de programa son los encargados de llevar a cabo las funcionalidades de la aplicación, es como si fuesen el "cerebro" de esta. Además de las actividades se encuentran los *Fragment*, que como su propio nombre indica son fragmentos de la aplicación. Las clases java de las *Activity* y de los *Fragment* heredan el nombre de las propias actividades y fragmentos.

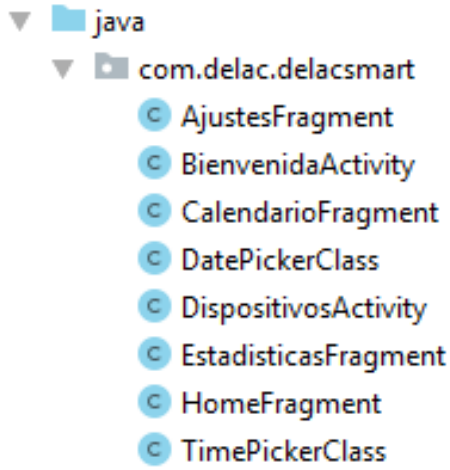

Ilustración 55. Carpeta java Android Studio.

Las clases java de los *Activity* son:

- BienvenidaActivity
- DispositivosActivity

Las clases java de los *Fragment* son:

- HomeFragment
- CalendarioFragment

- EstadísticasFragment
- AjustesFragment

Clases adicionales:

- DatePickerClass
- TimePickerClass

La clase java *BienvenidaActivity* es la encargada de mostrar la pantalla de bienvenida o *Splash Screen*, y cuando estén cargadas todas las funcionalidades de la aplicación lanzar la *Activity DispositivosActivity*. La actividad *BienvenidaActivity* no tiene un *layout* ligado y la pantalla se implementa en su clase java y es la que se muestra en la Ilustración 56.

Ilustración 56. Pantalla de bienvenida "Splash Screen".

La clase java *DispositivosActivity* es la encargada de ejecutar las funcionalidades principales de la aplicación. Esta actividad tendrá un menú de navegación para cambiar entre las pestañas: home, calendario, estadísticas y ajustes. Cuando se pulse sobre los distintos iconos se mostrará el *fragment* correspondiente a cada pestaña, que respectivamente son: *HomeFragment*, *CalendarioFragment*, *EstadísticasFragment* y *AjustesFragment.*

Además de gestionar el menú de navegación y lanzar el fragmento necesario en cada caso, la clase *DispositivosActivity* se encarga de llevar a cabo los métodos del protocolo MQTT. En esta clase se establecerá la conexión con el *broker* MQTT y cuando esté conectada se suscribirá a los siguientes *topics*:

- SENSOR/Temperatura
- ACTUADOR/Estado
- CONTROL_PPAL/SetpointTemp
- CONTROL_PPAL/HOME/ModoSemiAUT
- CONTROL_PPAL/HOME/Dia_Hora
- CONTROL_PPAL/HOME/Intervalo

Una vez se ha suscrito a los temas indicados la aplicación estará a la espera de recibir mensajes, y cuando reciba alguno la clase los gestionará y hará la tarea o tareas que correspondan.

Cuando haya que publicar un mensaje *DispositivosActivity* será la encargada de hacerlo. Los *topic* sobre los que la aplicación publica son:

- CONTROL_PPAL/HOME/SetpointMAN
- CONTROL_PPAL/Calendario/Modo_dias
- CONTROL_PPAL/Calendario/Intervalos
- CONTROL_PPAL/Ajustes/Estacion_Ano
- CONTROL_PPAL/Ajustes/Modo_Trabajo
- CONTROL_PPAL/Ajustes/Modo_Casa
- CONTROL_PPAL/Ajustes/Temp_EcoMode
- CONTROL_PPAL/Ajustes/Modo_Vacaciones
- SENSOR/Delta
- SENSOR/Max_Enviar
- SENSOR/Rastreo
- ACTUADOR/Max_Enviar

Si la aplicación no consigue establecer una conexión MQTT con el *broker* o pierde la conexión por cualquier motivo, esta clase hará que se muestre un cuadro de diálogo para que se reintente la conexión MQTT.

La clase *HomeFragment* es la encargada de ejecutar las tareas específicas del fragment home. En él se mostrará la temperatura que está midiendo el sensor de temperatura, y se podrá modificar el *setpoint* de forma manual o visualizar el valor que se ha configurado en la pestaña de calendario, se mostrará el estado de la caldera y los modos en los que se encuentra, es decir, si está en modo automático o no, si se encuentra en casa o fuera, etc.

En la clase *CalendarioFragment* se implementan los menús para la configuración de los intervalos de funcionamiento, las temperaturas de

referencia y las horas de inicio de estos. Además, se podrá elegir el modo de programación para cada día entre diario y festivo.

La clase *EstadisticasFragment* mostrará información estadística de lo que ocurre en el sistema demótico, como el tiempo de funcionamiento de la caldera, temperatura máxima y mínima, etc.

El fragmento *AjustesFragment* implementa los menús de configuración de los modos de la aplicación, modo trabajo, modo casa, modo vacaciones, etc., y también los ajustes del sensor de temperatura y del actuador.

Las clases *TimePickerClass* y *DatePickerClass no tienen* ningún *layout* asociado, sin embargo, la parte visual está incluida y programada en estas clases java y mostrará los cuadros de dialogo que se muestran en la Ilustración 57 y la Ilustración 58 respectivamente. Estas clases son las encargadas de que se pueda seleccionar la hora de llegada y la fecha de llegada en dichos cuadros de dialogo cuando se active el modo fuera de casa o modo vacaciones.

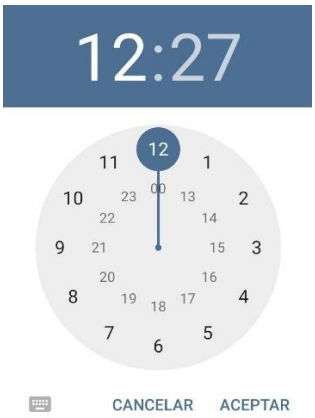

Ilustración 57. Cuadro de diálogo para seleccionar la hora.

Ilustración 58. Cuadro de diálogo para seleccionar la fecha.

La carpeta res, que se muestra en la Ilustración 59, contiene las carpetas que tienen los recursos necesarios para la personalización de la aplicación.

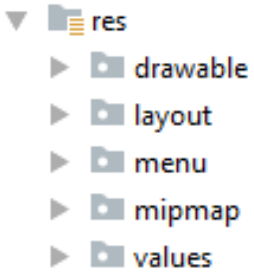

Ilustración 59. Carpeta res Android Studio.

La carpeta *drawable* y *mipmap* contienen las imágenes, iconos, archivos dibujables que usa la aplicación. La carpeta menú contiene un archivo que define la personalización del menú de la aplicación, el cual se puede ver en la Ilustración 60.

Ilustración 60. Menú de navegación DELAC Smart.

La carpeta *layout* contiene los *layout* programados en xml correspondientes a las *Activity* y los *Fragment*, *que* dan forma a las pantallas de la aplicación que se muestran en el teléfono. Estos archivos son los que se muestran en la Ilustración 61

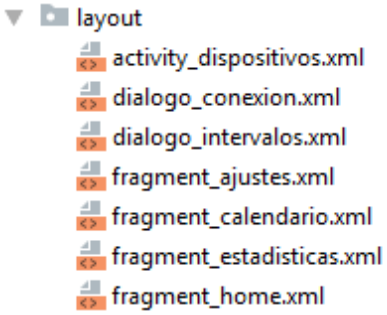

Ilustración 61. Carpeta layout Android Studio.

Solo hay un *layout* ligado a una *Activity* y es el correspondiente a la actividad *DispositivosActivity*:

- activity_dispositivos.xml

Los *layout* ligados a los *Fragment*:

- fragment_home.xml
- fragment_calendario.xml

- fragment_estadisticas.xml
- fragment_ajustes.xml

Otros *layout* adicionales:

- dialogo_conexion.xml
- dialogo_intervalos.xml

El *layout activity_dispositivos.xml* es el archivo que genera la pantalla de la actividad *DispositivosActivity*, que se muestra en la Ilustración 62. Este layout está formado por la *action bar*, la cual contiene el nombre de la aplicación (DELAC Smart) y va colocada en la parte superior de la pantalla con fondo de color azul, un contenedor de *fragments* y una barra de navegación en la parte inferior para interactuar entre las distintas pantallas de home, calendario, etc.

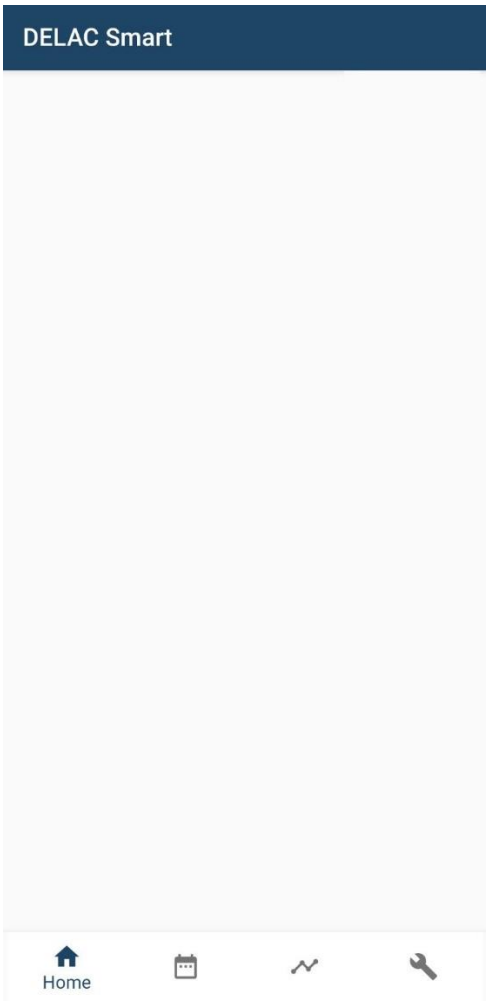

Ilustración 62. Layout de la Activity DispositivosActivity.

El contenedor de *fragmetns* es toda la parte blanca central comprendida entre la *action bar* y la barra de navegación. En este contenedor se visualizarán los *fragment* en función de dicho menú de navegación.

El *layout* del fragment *HomeFragment* es el archivo *fragment_home.xml*. Este *layout* es el encargado de generar la parte visible de dicho fragmento en la pantalla del móvil, el cual podemos observar en la Ilustración 63. Este fragmento se visualizará en el contenedor de *fragments* del layout *dispositivos_activity.xml.*

Ilustración 63. Layout del Fragment HomeFragment.

El *layout* del fragment *CalendarioFragment* es *fragment_calendario.xml*. Este *layout* es el encargado de generar la parte visible de dicho fragmento en la pantalla del móvil, el cual podemos observar en la Ilustración 64. Este fragmento se visualizará en el contenedor de *fragments* del layout *dispositivos_activity.xml.*

Ilustración 64. Layout del Fragment CalendarioFragment.

El *layout* del fragment *EstadisticasFragment* es *fragment_estadisticas.xml*. Este *layout* es el encargado de generar la parte visible de dicho fragmento en la pantalla del móvil, el cual podemos observar en la Ilustración 65. Este fragmento se visualizará en el contenedor de *fragments* del layout *dispositivos_activity.xml.*

Ilustración 65. Layout del Fragment EstadisticasFragment.

El *layout* del fragment *AjustesFragment* es el archivo *fragment_ajustes.xml*. Este *layout* es el encargado de generar la parte visible de dicho fragmento en la pantalla del móvil, el cual podemos observar en la Ilustración 63. Este fragmento se visualizará en el contenedor de *fragments* del layout *dispositivos_activity.xml.*

Ilustración 66.Layout del Fragment AjustesFragment.

El *layout dialogo_conexion.xml* es el encargado de visualizar el cuadro de dialogo que se muestra en la Ilustración 67 cuando ocurre un fallo de conexión MQTT con el *broker*. Este cuadro nos permite reintentar la conexión.

Ilustración 67. Cuadro de diálogo de fallo de conexión MQTT.

El *layout dialogo_intervalos.xml* es el cuadro de dialogo que aparece cuando se quiere configurar un intervalo en el calendario. En el podremos configurar la hora de inicio y la temperatura de referencia para ese intervalo. Dicho cuadro de dialogo es el que podemos ver en la Ilustración 68.

Ilustración 68. Cuadro de diálogo de configuración de los intervalos.

En la carpeta *values* se encuentran los archivos donde se configuran el tema que usa la aplicación, llamado *AppTheme*, el cual consta de tres colores para personalizar la aplicación: *colorPrimary, colorPrimaryDark, colorAccent*. Nosotros hemos querido darle predominancia al color azul, por lo cual los colores en valor hexadecimal son:

- colorPrimary: #1E4466.
- colorPrimaryDark: #001D3B.
- colorAccent: #4e6f94.

La carpeta *Gradle Scripts,* que se muestra en la Ilustración 69, contiene los archivos *build.gradle*, en los cuales se almacena información importante para la compilación del proyecto, como la versión de Android SDK, tanto la ideal como la mínima para que funcione la aplicación. También se define la versión de la aplicación y se incluyen las referencias a librerías externas, por ejemplo, nosotros incluimos aquí las librería para el cliente MQTT.

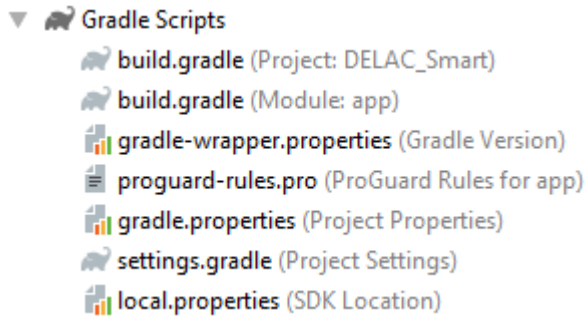

Ilustración 69. Carpeta Gradle Scripts Android Studio

Todos los archivos del proyecto de la aplicación se pueden encontrar en los anexos de la memoria.

5. LINEAS FUTURAS

En este proyecto se ha llevado a cabo el control de temperatura de una casa construyendo un termostato inteligente como ejemplo de sistema domótico dentro del paradigma del IoT y usando el protocolo MQTT para intercomunicar los distintos dispositivos que forman el control. Con el sistema que se ha desarrollado se facilita la vida del usuario, brindándole un confort y un mayor control sobre el entorno.

El confort se obtiene gracias a un control basado en eventos de la temperatura. El usuario fija un *setpoint* de la temperatura relacionado con las expectativas de confort de este. En un futuro sería interesante considerar en evolucionar el control para que ese confort no dependa únicamente de la variable de la temperatura, sino que también incluya otros parámetros como la humedad, opinión del usuario, etc.

Se le podría preguntar al usuario como se encuentra, y que el sistema fuese aprendiendo con él para saber cómo le gusta que esté su casa y comprender las sensaciones de la persona y ajustar la calefacción, aire acondicionado, y otros a su gusto para brindarle el confort deseado. Un ejemplo de interfaz de usuario para evaluar el confort de una persona es el que se ve en la Ilustración 70 , la cual se ha obtenido del artículo "*Personal comfort models – A new paradigm in thermal comfort for occupant-centric environmental control"*[20].

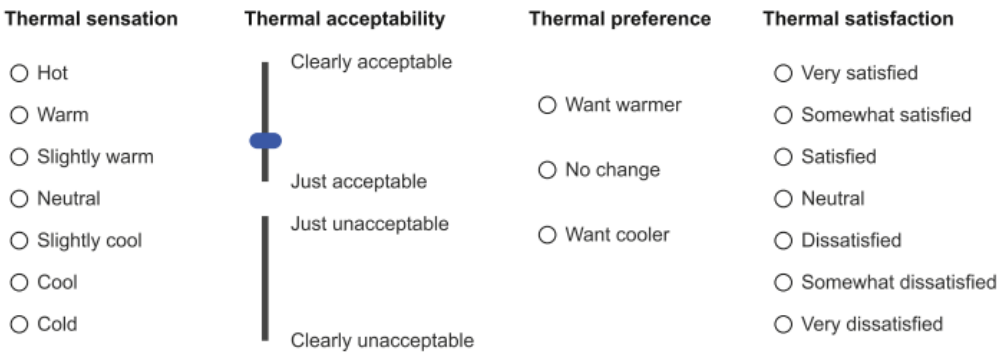

Ilustración 70. Interfaz de usuario para evaluar el confort.

Como hemos comentado el sistema domótico que se ha desarrollado en este proyecto únicamente controla la temperatura de la casa, sin embargo, el concepto de sistema domótico alberga mucho más que únicamente el control de la temperatura. En el futuro se incluirían más aspectos de la casa a controlar, como puede ser la iluminación, la seguridad, con el control de cámaras, alarmas antirrobo, alarmas antincendios, etc. Al fin y al cabo, cualquier cosa que se imagine podría llegar a ser automatizada e incluida en el sistema domótico. Cuando el sistema esté completo cobrará más sentido la estructura que se ha creado para el sistema domótico implementado en el presente trabajo.

En este proyecto únicamente se ha desarrollado los prototipos de sensor de temperatura y actuador, sin embargo, en un futuro habría que diseñar la PCB del circuito que integran y una envolvente para ellos. Un ejemplo de una envolvente para el sensor de temperatura seria la que se puede ver en la Ilustración 71.

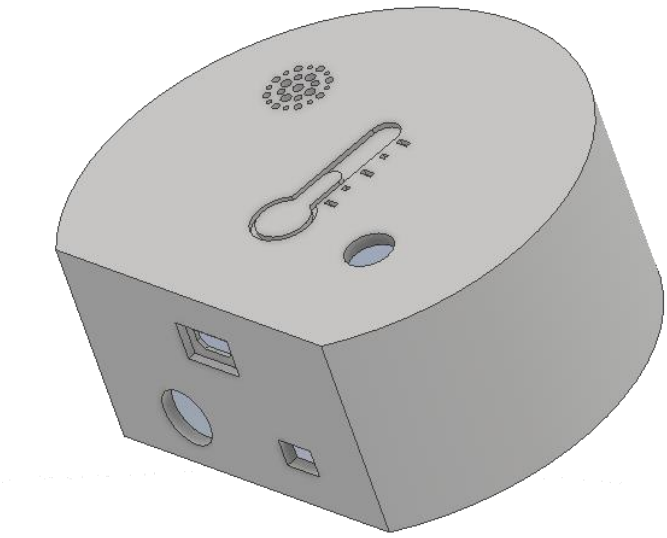

Ilustración 71. Envolvente Sensor de Temperatura.

La envolvente ejemplo del actuador seria la que se ve en la Ilustración 72

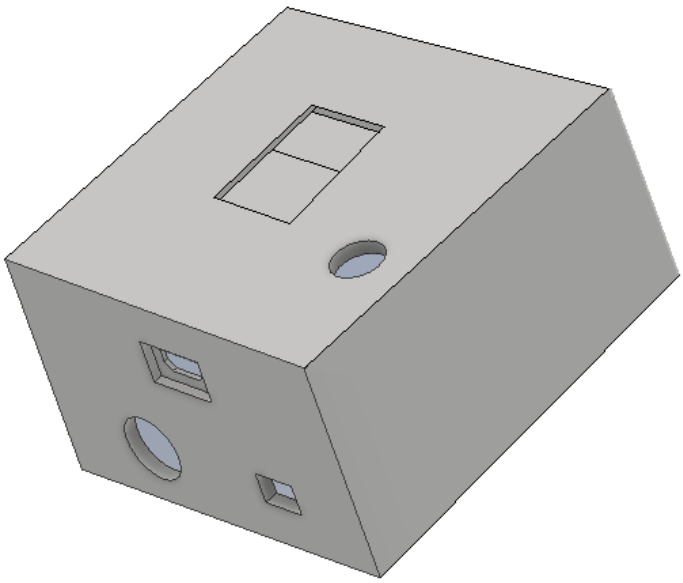

Ilustración 72. Envolvente Actuador.

Otro aspecto para evolucionar en el futuro sería el controlador central. Ahora se ha utilizado un mini-pc, pero en un futuro habría que realizar el diseño completo de este. En este dispositivo se implementaría el control de todos los aspectos de la casa que se vaya a automatizar.

En dicho mini-pc también se ejecuta el *broker* del protocolo MQTT, en un futuro sería interesante debatir cual sería el lugar idóneo para colocar dicho intermediario. Quizás en un servidor de la empresa que gestione todos los usuarios que se registren para poder gestionar sus dispositivos.

En cuanto a la interfaz de usuario, en nuestro caso se optó por la implementación de una aplicación en Android, sin embargo, en un futuro habría que ofrecer una solución para todas las plataformas disponibles, como por ejemplo iOS.

Para nuestra solución la interfaz de usuario únicamente incorpora el control de temperatura que se ha diseñado, en un futuro sería interesante diseñar la aplicación para que hubiera una pantalla de inicio donde se crearían "casas" que estarían ligadas a un usuario que se haya registrado previamente, donde se añadirían los dispositivos y se asociarían a ellas. En esa pantalla se incluirían todos los dispositivos de la casa y daría acceso a la pantalla de cada uno. Un ejemplo de la pantalla de la que se habla es la que se ve en la Ilustración 73.

Ilustración 73. Pantalla de inicio de la aplicación DELAC Smart.

La configuración de la conexión WiFi de estos dispositivos (sensor y actuador) se ha hecho con un portal cautivo que muestra la página web de configuración. En un futuro para simplificar y hacer el proceso más sencillo, el procedimiento de configuración y añadirlo a la casa deseada se hará con la aplicación *DELAC Smart*.

El apartado de estadísticas está implementado de una forma rudimentaria y es uno de los aspectos a mejorar, pudiendo desarrollar una base de datos que almacene la información obtenida por los dispositivos.

En conclusión, las líneas futuras apuntan a pulir el producto hasta conseguir que sea comercial y pudiera venderse al usuario final. Hacerle más robusto, funcional, intuitivo e ir mejorándolo de forma continua en función de las necesidades que la sociedad demanda y siguiendo la línea de los avances tecnológicos.

6. CONCLUSIONES

Se ha conseguido desarrollar un sistema domótico dentro del paradigma del IoT, el cual facilita la vida del usuario, le brinda confort y mayor control sobre su entorno. Este sistema utiliza el protocolo MQTT para coordinar de forma asíncrona el funcionamiento de varios dispositivos para conseguir una solución de control de temperatura de una casa que planta las bases de un control domótico más complejo.

Cabe destacar que se ha tenido un especial interés en no dejarlo en una mera prueba de concepto sino avanzar lo posible en la definición de producto. Para ello se ha implementaron las páginas de configuración WiFi para los dispositivos periféricos, se almacenaron las credenciales en la memoria EEPROM, y para poder conectarse desde fuera de casa se llevó a cabo un redireccionamiento NAT.

Otro punto de interés de este trabajo ha sido en plantar las bases de un control escalable en el tiempo, tanto en número de dispositivos como en complejidad del sistema. En un futuro se seguirá avanzando en él hasta construir un producto comercial en la dirección del control de confort.

A lo largo del trabajo se ha podido observar como el protocolo MQTT satisface las necesidades que teníamos a la hora de brindar una solución domótica e IoT. Este protocolo posibilita la comunicación entre dispositivos de una forma no-bloqueante y que estos sean elementos levemente acoplados.

Para desarrollar el termostato inteligente nos hemos servido de diferentes tecnologías, dispositivos con distintos recursos, entornos y lenguajes de programación, lo que demuestra que el protocolo MQTT es muy versátil y puede ser usado en multitud de situaciones.

Por otro lado, se sabe que los teléfonos inteligentes tienes aplicaciones que exceden los meramente relacionados con la comunicación. Una de estas aplicaciones bien pudiera ser el control domótico. En nuestro caso hemos usado un teléfono inteligente como intermediario entre el medio y el humano, implementando una aplicación Android que sirve de interfaz de usuario. El desarrollo de dicha aplicación nos ha hecho valorar el trabajo que hay detrás de ellas y comprender como funcionan intrínsecamente.

Por último, quiero destacar que la elección de este proyecto se tomó principalmente para utilizar el *Trabajo de Fin de Grado* como una herramienta de aprendizaje e introducción en el mundo de la domótica e Internet de las Cosas. Temas que están de actualidad y cada vez cobran más importancia en el sector de la ingeniería. En este aspecto se ha cumplido el objetivo que se tenía aprendiendo sobre domótica, IoT, MQTT, Android, C++, HTML, entre otros conceptos y tecnologías.

7. BIBLIOGRAFÍA

[1] «¿Qué es IoT (Internet Of Things)? | Deloitte España». [En línea]. Disponible en: https://www2.deloitte.com/es/es/pages/technology/articles/IoT-internet-of-things.html. [Accedido: 04-may-2019].

[2] «Qué es Domótica - CEDOM | Asociación Española de Domótica e Inmótica». [En línea]. Disponible en: http://www.cedom.es/sobre-domotica/que-es-domotica. [Accedido: 04-may-2019].

[3] J. Bonér, D. Farley, R. Kuhn, y Thompson. Martin, «El Manifiesto de Sistemas Reactivos». [En línea]. Disponible en: https://www.reactivemanifesto.org/es. [Accedido: 05-may-2019].

[4] «IoT: protocolos de comunicación, ataques y recomendaciones | INCIBE-CERT». [En línea]. Disponible en: https://www.incibe-cert.es/blog/iot-protocolos-comunicacion-ataques-y-recomendaciones. [Accedido: 05-may-2019].

[5] «Conociendo MQTT». [En línea]. Disponible en: https://www.ibm.com/developerworks/ssa/library/iot-mqtt-why-good-for-iot/index.html. [Accedido: 05-may-2019].

[6] «Eclipse Mosquitto». [En línea]. Disponible en: https://mosquitto.org/. [Accedido: 16-may-2019].

[7] «Teach, Learn, and Make with Raspberry Pi – Raspberry Pi». [En línea]. Disponible en: https://www.raspberrypi.org/. [Accedido: 18-jun-2019].

[8] «FrontPage - Raspbian». [En línea]. Disponible en: https://www.raspbian.org/. [Accedido: 18-jun-2019].

[9] «NodeMcu -- An open-source firmware based on ESP8266 wifi-soc.» [En línea]. Disponible en: https://www.nodemcu.com/index_en.html. [Accedido: 07-may-2019].

[10] «ESP8266EX Datasheet», 2018.

[11] «LM35 LM35 Precision Centigrade Temperature Sensors», 1999.

[12] «RELAY SRD-05VDC-SL-C».

[13] «Arduino - Home». [En línea]. Disponible en: https://www.arduino.cc/. [Accedido: 24-jun-2019].

[14] «Arduino Client for MQTT». [En línea]. Disponible en: https://pubsubclient.knolleary.net/. [Accedido: 24-jun-2019].

[15] «Android Developers». [En línea]. Disponible en: https://developer.android.com/. [Accedido: 24-jun-2019].

[16] «Eclipse Paho - MQTT and MQTT-SN software». [En línea]. Disponible en: https://www.eclipse.org/paho/clients/android/. [Accedido: 24-jun-2019].

[17] A. Williams, *C++ Concurrency in Action*. 2012.

[18] C. Kormanyos, *Real-Time C++: Efficient Object-Oriented and Template Microcontroller Programming*. 2015.

[19] «Generador de códigos QR diseños - Gratuito - Unitag». [En línea]. Disponible en: https://www.unitag.io/es/qrcode. [Accedido: 27-jun-2019].

[20] J. Kim, S. Schiavon, y G. Brager, «Personal comfort models – A new paradigm in thermal comfort for occupant-centric environmental control», *Build. Environ.*, vol. 132, pp. 114-124, mar. 2018.

8. ANEXOS

8.1. Programa Sensor de Temperatura

SensorTemperatura.ino

```
1. /******************* DECLARACIÓN DE
   LIBRERIAS*******************************/
2. #include "ESP8266WiFi.h" //Libreria para la tarjeta de desarrollo nodeMCU
3. #include "WiFiClient.h"
4. #include "ESP8266WebServer.h"
5. #include "DNSServer.h" //Libreria para servidor DNS. Utilizada para el
   portal cautivo.
6. #include "PubSubClient.h"  //Libreria MQTT para arduino
7. #include "EEPROM.h" //Libreria para la memoria EEPROM
8.
9. /******************* DECLARACION DE
   VARIABLES*******************************/
10.  //Topics
11.  const char* topic_deltaSensor = "SENSOR/Delta";
12.  const char* topic_MaxEnviarSensor = "SENSOR/Max_enviar";
13.  const char* topic_RastreoSensor = "SENSOR/Rastreo";
14.  const char* topic_tempActual = "SENSOR/Temperatura";
15.
16.  //Variables mensaje
17.  long lastMsg = 0;
18.  long lastMsg2 = 0;
19.  char msg[50];
20.
21.  //Asignacion GPIO a los LED
22.  int LED_AP = 5;
23.  int LED_WiFi = 4;
24.  int ONLed = 13;
25.  int pulsador_reset = 12;
26.
27.  //Variable intentos de conexion wifi STA, cada intento corresponde a
   medio segundo.
28.  int intentos = 0;
29.
30.  //Variable de estado del dispositivo:
31.  //  0-Crear AP
32.  //  1-AP creado, esperando recibir las credenciales
33.  //  2-Credenciales recibidas, se pasa a modo STA y se conecta al WiFi
34.  //  3-Conectado al WiFi, el sensor llevará a cabo sus tareas especificas
35.  int estado=0;
36.
37.  //Contador de pulsos del pulsador
38.  volatile int cont = 0;
39.
40.  //Variable sensor temperatura LM35
41.  int pinLM35 = 0;
42.  float tempC; //Valor de la temperatura obtenida del sensor
43.  float old_tempC=0; //Último valor enviado(publicado)
44.
45.  //Variables recibidas
46.  long rastreo = 2000;
47.  long max_envio = 10000;
48.  float delta = 1.0;
49.
50.  //Variables para la construccion de la pagina web
```

```
51.  String cabecera =
52.  "<!DOCTYPE HTML >"
53.  "<html>"
54.  "  <head>"
55.  "    <title>TempSensor - WiFi Setup</title>"
56.  "    <meta http-equiv=\"Content-Type\" content=\"text/html; charset=utf-
     8\">"
57.  "    <meta name=\"viewport\" content=\"width=device-width, initial-
     scale=1\">"
58.  "  </head>"
59.  "  <body style=\"background-color:#F1EFC8\";>"
60.  "    <div>"
61.  "      <div style=\"background-color:#C70039\"; align=\"middle\">"
62.  "        <h2
     style=\"color:#F1EFC8\";><br>🌡CONTROL<br>DE<br>TEMPERATURA<br></h2>"
63.  "        <h2 style=\"color:#F1EFC8\";>[Sensor de Temperatura]<br><br>WiFi
     SetUp<br><br></h2>"
64.  "      </div>";
65.
66.  String lista_ssid = "";
67.
68.  String pssw_broker =
69.  "          <tr>"
70.  "             <td>‣</td>"
71.  "             <th align=\"left\">CONTRASEÑA:</th>"
72.  "          </tr>"
73.  "          <tr>"
74.  "             <td></td>"
75.  "             <td><input name=\"password\" type=\"password\"
     id=\"myInput\"></td>"
76.  "             <td align=\"center\"><input type=\"checkbox\"
     onclick=\"myFunction()\"></td>"
77.  "             <td align=\"center\"> 👁</td>"
78.  "          </tr>"
79.  "          <tr>"
80.  "             <td>‣</td>"
81.  "             <th align=\"left\">BROKER:</th>"
82.  "          </tr>"
83.  "          <tr>"
84.  "             <td></td>"
85.  "             <td><input name=\"broker\" type=\"text\"></td>"
86.  "             <td></td>"
87.  "             <td></td>"
88.  "          </tr>"
89.  "    </table><br>";
90.
91.  String botones =
92.  "            <div align=\"middle\">"
93.  "             <input type=\"submit\" value=\"Enviar\">"
94.  "             <input type=\"reset\" value=\"Borrar\">"
95.  "             <br><br>"
96.  "             <p>*En caso de no aparecer la red WiFi deseada, refresque
     la pagina refresque la pagina o escanee de nuevo pulsando:[ ↺ ].</p>"
97.  "            </div>";
98.
99.  String callback_submit =
100. "      <div  align=\"middle\">"
101. "        <h3>✓ Credenciales enviados.</h3><br><br><br><br><br><br>"
102. "      </div>";
103.
```

```
104. String pie_pagina =
105. "           <div style=\"background-color:#FFA630\"; align=\"middle\">"
106. "             <p><i><br>Configuración de la conexión<br><br></i></p>"
107. "           </div>"
108. "         </form>"
109. "       </div>"
110. "     </div>"
111. "  <script>"
112. "  function myFunction() {"
113. "    var x = document.getElementById(\"myInput\");"
114. "    if (x.type === \"password\") {"
115. "      x.type = \"text\";"
116. "    } else {"
117. "      x.type = \"password\";"
118. "    }"
119. "  }"
120. "  </script>"
121. "  </body>"
122. "</html>";
123.
124. //Datos conexión punto de acceso (AP) WiFi
125. const char ssid_ap[] = "TempSensor_WiFi_setup";
126. const char password_ap[] = "0123456789";
127.
128. //Variables almacenamiento credenciales red WiFi de casa
129. char ssid[50];
130. char password[50];
131.
132. //Datos del broker MQTT
133. char mqtt_server[50];
134.
135. //Declaración servidor DNS
136. const byte DNS_PORT = 53;
137. DNSServer dnsServer;
138.
139. //Declaracion servidor
140. ESP8266WebServer server(80);
141.
142. //Cliente wifi
143. WiFiClient espClient;
144. PubSubClient client(espClient);
145.
146. /******************* DECLARACIÓN DE
     FUNCIONES*****************************/
147.
148. //Función WIFI AP punto de acceso wifi
149. void setup_wifi_ap(){
150.   boolean result_ap = false; //Variable resultado de intentar crear
     access point;
151.   digitalWrite(LED_WiFi, LOW);
152.   digitalWrite(LED_AP, HIGH);
153.   Serial.print("\n\n---> MODO WIFI AP\n\n");
154.
155.   while(result_ap==false){
156.     WiFi.disconnect(); //Desconectamos el WiFi
157.     WiFi.mode(WIFI_AP); //Establecemos el modo de WiFi como Access Point
158.     result_ap = WiFi.softAP(ssid_ap,password_ap); //Creamos el punto de
     acceso wifi
159.     delay(100);
160.     if(result_ap == true){
161.       Serial.println("Access Point creado con exito\n");
```

```
162.       Serial.print("Direccion IP Access Point - por defecto: ");
   //Imprime la dirección IP
163.       Serial.println(WiFi.softAPIP());
164.       Serial.print("Direccion MAC Access Point: ");
   //Imprime la dirección MAC
165.       Serial.println(WiFi.softAPmacAddress());
166.
167.       dnsServer.start(DNS_PORT, "*",IPAddress(192, 168, 4, 1)); //Arranco
   el servidor DNS para el portal cautivo
168.                                                                  //en la
   dirección IP 192.168.4.1 del NodeMCU
169.       server.onNotFound([]() {
170.         escanear_redes();
171.       });
172.
173.       //Declaramos las llamadas a las funciones de las peticiones HTTP
174.       server.on("/", handleRoot);
175.       server.on("/", escanear_redes);
176.       server.on("/submit", handleSubmit);
177.       server.on("/escanear", escanear_redes); //Escanea las redes wifi
   disponibles
178.       server.begin();
179.       Serial.println("Servidor HTTP creado");
180.       estado = 1; //Se cambia el estado a 1. El AP ha sido creado y se
   mantiene a la espera de las credenciales.
181.     }
182.     else
183.     {
184.       Serial.println("Fallo al crear el Access Point");
185.
   Serial.print("\nReintentando");delay(200);Serial.print(".");delay(200);Seri
   al.print(".");delay(200);Serial.println(".");
186.     }
187.   }
188. }//Fin setup_wifi_ap()
189.
190. //Función de envío de pagina web cuando se esta en AP mode
191. void handleRoot()
192. {
193.   server.send(200, "text/html", cabecera + lista_ssid + pssw_broker +
   botones + pie_pagina);
194. }//Fin handleRoot()
195.
196. //Función de recepción y almacenamiento de SSID y PASSWORD y BROKER de la
   pag web
197. void handleSubmit(){
198.   String SSIDvalue;
199.   String PASSWORDvalue;
200.   String BROKERvalue;
201.
202.   Serial.println("\n**DATOS RECIBIDOS PAGINA WEB**\n");
203.
204.   if (server.args() > 0 ) {
205.     for ( int i = 0; i < server.args(); i++ ) {
206.       if (server.argName(i) == "ssid") {
207.          SSIDvalue = server.arg(i);
208.          SSIDvalue.toCharArray(ssid, 50);
209.          Serial.print("SSID recibido: ");
210.          Serial.println(ssid);
211.       }else if(server.argName(i) == "password"){
212.          PASSWORDvalue = server.arg(i);
```

```
213.          PASSWORDvalue.toCharArray(password, 50);
214.          Serial.print("PASSWORD recibida: ");
215.          Serial.println(password);
216.       }else if(server.argName(i) == "broker"){
217.          BROKERvalue = server.arg(i);
218.          BROKERvalue.toCharArray(mqtt_server, 50);
219.          Serial.print("BROKER recibido: ");
220.          Serial.println(mqtt_server);
221.       }
222.     }
223.   }
224.
225.   Serial.println("\n*****************************\n");
226.
227.   //Cuando se reciben los datos, se lanza una pagina web que muestra que
   los datos han sido recibidos
228.   server.send(200, "text/html", cabecera + callback_submit +
   pie_pagina);
229.   delay(100);
230.   guardarCredenciales(); //Se guardan los credenciales en la memoria
   EEPROM
231.   cont=0; //Ponemos el contador de pulsos del pulsador a cero
232.   estado=2; //Se cambia a estado 2. Las credenciales han sido recibidas y
   se pasa al modo STA para conectarse al WiFi
233. }//Fin handleSubmit()
234.
235. //Función que escanea las redes WIFI disponibles y genera su parte de
   código html
236. void escanear_redes(){
237.   int n = WiFi.scanNetworks(); //Número de redes encontradas
238.   if(n == 0){ //Si no hay redes encontradas muestra un mensaje de que no
   hay redes disponibles
239.     Serial.println("No se han encontrado redes WiFi disponibles");
240.     lista_ssid = "No se han encontrado redes WiFi disponibles";
241.   }else{ //Si encuentra redes WiFi, compone la parte de codigo html para
   la pagina web
242.     lista_ssid =
243.     "      <div  align=\"middle\">"
244.     "        <h3>Por favor, complete los credenciales:</h3>"
245.     "        <form action=\"http://192.168.4.1/submit\" method=\"post\">"
246.     "        <table>"
247.     "        <tr>"
248.     "           <td>‣</td>"
249.     "           <th align=\"left\">SSID:</th>"
250.     "           <td></td>"
251.     "           <td></td>"
252.     "        </tr>"
253.     "        <tr>"
254.     "           <td></td>"
255.     "           <td><select name=\"ssid\">";
256.
257.     for(int i = 0; i < n; i++){ //Genera la lista de redes WiFi
258.       lista_ssid = (lista_ssid) + "<option value= \"" + WiFi.SSID(i) +
   "\">" + WiFi.SSID(i)+ "</option>";
259.       delay(10);
260.     }
261.     //Se forma la parte de la página web de la lista de redes
262.     lista_ssid = (lista_ssid) + "</select></td>";
263.     lista_ssid = (lista_ssid) + "<td align=\"center\"><button
   type=\"submit\"
   formaction=\"http://192.168.4.1/escanear\">↺</button></td>";
```

```
264.     lista_ssid = (lista_ssid) + "<td align=\"center\"> *</td></tr>";
265.
266.     handleRoot(); //Llama a la función que genera y envía al navegador la
   pagina completa
267.   }
268. }//Fin escanear_redes()
269.
270. //Función WIFI STA conexión a red wifi externa
271. void setup_wifi(){
272.   digitalWrite(LED_AP, HIGH);
273.   bool estado_LED_WiFi = LOW;
274.   long lastMsg = 0;
275.   Serial.print("\n\n---> MODO WIFI STA\n\n");
276.
277.   Serial.print("Conectando a... ");
278.   Serial.println(ssid);
279.
280.   WiFi.softAPdisconnect(); //Desconectamos el modo AP del wifi
281.   WiFi.disconnect(); //Desconectamos el wifi
282.
283.   WiFi.mode(WIFI_STA); //Establecemos modo wifi STA
284.   WiFi.begin(ssid, password); //Conexión a la red WiFi
285.
286.   //Mientras no se ha conectado al WiFi espera 30 segundos
287.   while (WiFi.status() != WL_CONNECTED){
288.     Serial.print(".");
289.     delay(500);
290.     intentos ++;
291.     digitalWrite(LED_WiFi, estado_LED_WiFi);
292.     estado_LED_WiFi = !estado_LED_WiFi; //Se cambia el estado para que el
   led parpadee
293.     //Cada intento equivale a medio segundo: 60/2= 30 segundos
294.     if(intentos > 60){ //Si supera los 30 segundos, es decir los 60
   intentos volverá al modo AP
295.       Serial.println("\n¡No se pudo conectar!\nEs posible que los
   credenciales sean invalidos.");
296.       Serial.println("*NOTA: Entrara en el modo AP de nuevo. Configure la
   red WIFI nuevamente.");
297.       estado = 0; //Se cambia estado 0, que generará de nuevo el punto de
   acceso AP
298.       intentos = 0; //Pone el contador de tiempo a cero
299.       return; //Se sale de la función para que no ejecute el codigo de
   abajo
300.     }
301.   }
302.
303.   //Si se consigue conectar al WiFi se ejecutara el siguiente código:
304.
305.   digitalWrite(LED_WiFi, HIGH); //Se enciende el led de conexión que
   indica que esta conectado al WiFi
306.
307.   Serial.println("");
308.   Serial.print("Conectado a la red:");
309.   Serial.println(ssid);
310.   Serial.print("Direccion IP: ");
311.   Serial.println(WiFi.localIP());
312.   Serial.println("\n");
313.   estado = 3; // Se cambia al estado 3. Se ha conectado al WiFi y el
   sensor hará sus tareas.
314.   intentos = 0;
315.
```

```
316. }//Fin setup_wifi()
317.
318. //Función cargar credenciales de la EEPROM
319. int cargarCredenciales() {
320.   EEPROM.begin(512);
321.   EEPROM.get(0, ssid);
322.   EEPROM.get(0 + sizeof(ssid), password);
323.   EEPROM.get(0 + sizeof(ssid) + sizeof(password), mqtt_server);
324.   char ok[4 + 1]; //Esta variable nos sirve para comprobar si hay
   credenciales almacenadas
325.   EEPROM.get(0 + sizeof(ssid) + sizeof(password) + sizeof(mqtt_server),
   ok);
326.   EEPROM.end();
327.
328.   //Si no coindice la variable ok con la cadena de caracateres no hay
   nada almacenado
329.   if (String(ok) != String("OKEY")) {
330.     ssid[0] = 0;
331.     password[0] = 0;
332.     mqtt_server[0] = 0;
333.     return 0; //Devuelve 0 cuando no hay credenciales almacenadas
334.   }
335.
336.   //Si hay credenciales almacenadas sobreescribe las variables de
   programa de estas
337.   Serial.println("Credenciales recuperados:");
338.   Serial.println(ssid);
339.   Serial.println(strlen(password) > 0 ? "********" : "<no password>");
340.   Serial.println(mqtt_server);
341.   return 2; //Devuelve 2 cuando hay credenciales almacenadas
342. }//Fin cargarCredenciales()
343.
344. //Función que guarda los credenciales en la EEPROM
345. void guardarCredenciales() {
346.   EEPROM.begin(512);
347.   EEPROM.put(0, ssid);
348.   EEPROM.put(0 + sizeof(ssid), password);
349.   EEPROM.put(0 + sizeof(ssid) + sizeof(password), mqtt_server);
350.   char ok[4 + 1] = "OKEY"; //Esta variable nos sirve para comprobar si
   hay credenciales almacenadas
351.   EEPROM.put(0 + sizeof(ssid) + sizeof(password) + sizeof(mqtt_server),
   ok);
352.   EEPROM.commit();
353.   EEPROM.end();
354. }//Fin guardarCredenciales()
355.
356. //Función que elimina los credenciales almacenados en la EEPROM
357. void eliminarCredenciales() {
358.   bool estado_LED_AP = LOW;
359.   EEPROM.begin(512);
360.   //Para borrar las credenciales sobreescribimos la memoria EEPROM con
   ceros
361.   for (int i = 0; i < 512; i++){
362.     EEPROM.write(i, 0);
363.   }
364.   //Parpadea para dar una señal visual de que se ha borrado la EEPROM
365.   for (int j=0; j<6; j++){
366.     digitalWrite(LED_AP, estado_LED_AP);
367.     delay(500);
368.     estado_LED_AP = !estado_LED_AP;
369.   }
```

```
370.   Serial.println("Credenciales borrados de la memoria EEPROM");
371.   EEPROM.end();
372. }//Fin eliminarCredenciales()
373.
374. //Función para resetear el NodeMCU (con la interrupcion del pulsador) y
   volver al estado de creacion punto de acceso AP
375. void Reset(){
376.   estado=0; //Se pasa al estado 0, de creación del punto de acceso AP.
377. }//Fin Reset()
378.
379. //Función que cuenta las veces que pulsas el pulsador (interrupcion en el
   modo AP)
380. void contador(){
381.   cont++;
382. }//Fin contador()
383.
384. //Función que recibe los mensajes de los topic suscritos
385. void callback(char* topic, byte* payload, unsigned int length) {
386.   String s; //Auxiliar para la conversión del payload
387.
388.   Serial.print("Mensaje recibido [");
389.   Serial.print(topic);
390.   Serial.print("]: ");
391.
392.   if(strcmp(topic,topic_deltaSensor) == 0){ //con strcmp comparo los dos
   string
393.     payload[length] = '\0';
394.     s = String((char*)payload);
395.     delta  = s.toFloat();
396.     Serial.println(delta);
397.   }else if(strcmp(topic,topic_MaxEnviarSensor) == 0){
398.     payload[length] = '\0';
399.     s = String((char*)payload);
400.     max_envio = s.toInt();
401.     Serial.println(max_envio);
402.   }else if(strcmp(topic, topic_RastreoSensor)== 0){
403.     payload[length] = '\0';
404.     s = String((char*)payload);
405.     rastreo = s.toInt();
406.     Serial.println(rastreo);
407.   }
408. }//Fin callback()
409.
410. //Función en la que el cliente se reconecta en caso de desconectarse o no
   estar conectado
411. bool reconnect() {
412.   digitalWrite(LED_AP, HIGH);
413.   // Creain ID de cliente aleatorio
414.   String clientId = "ESP8266Client-";
415.   clientId += String(random(0xffff), HEX);
416.   // Bucle hasta que se conecte al broker
417.   while (!client.connected()) {
418.     Serial.print("Esperando conexion MQTT...");
419.
420.     // Attempt to connect
421.     if (client.connect(clientId.c_str())) {
422.       digitalWrite(LED_AP, LOW);
423.       Serial.println("conectado");
424.       return true;
425.     } else {
426.       Serial.print("failed, rc=");
```

```
427.       Serial.print(client.state());
428.       Serial.println(" intentar en 5 segundos");
429.       // Wait 5 seconds before retrying
430.       delay(5000);
431.     }
432.
433.     if(estado == 0){ //Si pulso el pulsador de cambio de modo
434.                     // salgo del while y del reconnect para poder cambiar
   al modo AP
435.       return false;
436.     }
437.   }
438. }//Fin reconnect()
439.
440. //Función que se suscribe a los topic preconfigurados
441. void subscripcion_topics(){
442.   Serial.println("Me suscribo a los TOPIC preconfigurados");
443.
444.   client.subscribe(topic_deltaSensor); // Diferencia de variacion entre
   el ultimo valor enviado y la ultima lectura del sensor (temp)
445.   client.subscribe(topic_MaxEnviarSensor); // Tiempo máximo de envío de
   la temperatura si no envia la tª por modificacion con respecto al delta
446.   client.subscribe(topic_RastreoSensor); //Tiempo de obtención de un
   valor del sensor.
447. }//Fin subscripcion_topics
448.
449. //Función lectura temperatura
450. void leer_temperatura(){
451.   tempC = analogRead(pinLM35); //Leemos el valor analógico del sensor
   LM35
452.   tempC = (tempC*330.0)/1024.0;//Calculamos la temperatura a partir del
   valor analógico con Vin=3.3V
453.
454.   // Envia el dato al puerto serial
455.   Serial.print("Dato leido de temp: ");
456.   Serial.println(tempC);
457. }//Fin leer_temperaturas
458.
459. //Función que compara dos variables tipo float en funcion de
   epsilon(numero de decimales que compara)
460. //Devuelve un 1 si el primero es mayor que el segundo
461. //Devuelve un 2 si el segundo es mayor que el primero
462. //Devuelve un 0 si son iguales
463. int compara_mayor(float primero, float segundo, int epsilon){
464.   int primer_comp, segun_comp,valor;
465.
466.   primer_comp = primero * epsilon;
467.   segun_comp = segundo * epsilon;
468.
469.   if(primer_comp>segun_comp){
470.     valor = 1;
471.   }else if(primer_comp<segun_comp){
472.     valor = 2;
473.   }else{
474.     valor = 0;
475.   }
476.   return valor;
477. }//Fin compara_mayor()
478.
479. /********************************** DECLARACIÓN DEL SETUP()
   ****************************************/
```

```
480. void setup() {
481.   delay(1000);
482.   Serial.begin(115200);
483.   randomSeed(micros());
484.
485.   //Define los pines que se van a usar como salidas o entradas
486.   pinMode(LED_AP, OUTPUT);
487.   pinMode(ONLed, OUTPUT);
488.   pinMode(LED_WiFi, OUTPUT);
489.   pinMode(pulsador_reset, INPUT);
490.
491.   Serial.println("\t SENSOR Tª ENCENDIDO");
492.   digitalWrite(ONLed, HIGH);
493.   Serial.println ("\t Comprobacion EEPROM...");
494.   estado = cargarCredenciales(); //Comprueba si hay credenciales
   almacenadas en la EEPROM y asigna el estado del dispositivo
495.   if(estado == 0){
496.     Serial.println("EEPROM vacia. Nada que recuperar");
497.   }
498. }//Fin setup()
499.
500. /********************************** DECLARACION DEL LOOP()
   ****************************************/
501. void loop() {
502.   //En función de la variable estado se determina en que modo se
   encuentra el dispositivo
503.   switch(estado){
504.     //Modo 0 - Modo creación AP: El dispositivo crea el punto de acceso
   AP
505.     case 0:
506.       setup_wifi_ap();
507.       break;//Fin case 0
508.
509.     //Modo 1 - Modo AP: El punto de acceso está creado, se lanza la
   pagina web y se espera recibir las credenciales
510.     case 1:
511.
   attachInterrupt(digitalPinToInterrupt(pulsador_reset),contador,FALLING);
   //Se declará la interrupción para borrar la EEPROM
512.
513.       //Si el pulsador se presiona 3 veces se borra la EEPROM
514.       if(cont>=3){
515.         Serial.println("PULSADOR ACCIONADO 3 VECES!!");
516.         eliminarCredenciales();
517.         cont=0;
518.         detachInterrupt(digitalPinToInterrupt(pulsador_reset)); //Se
   elimina la interrupción para borrar la EEPROM
519.       }
520.
521.       dnsServer.processNextRequest(); //Portal cautivo
522.       server.handleClient();
523.       detachInterrupt(digitalPinToInterrupt(pulsador_reset)); //Se
   elimina la interrupción para borrar la EEPROM
524.       break;//Fin case 1
525.
526.     //Modo 2 - Modo STA: Credenciales recibidas, se conecta a la red WiFi
   introducida por el usuario
527.     case 2:
528.       setup_wifi();
529.
530.       break;//Fin case 2
```

```
531.
532.     //Modo 3 - Modo funcionamiento dispositivo. El sensor medirá y
   enviará la temperatura
533.     case 3:
534.
   attachInterrupt(digitalPinToInterrupt(pulsador_reset),Reset,FALLING);//Acti
   va interrupcion que accede al modo configuracion wifi (modo AP)
535.       client.setServer(mqtt_server, 1883); //Crea el cliente MQTT y le
   indica a que broker se tiene que conectar y en que puerto
536.       client.setCallback(callback); //Setea el callback de mensajes
   recibidos de los topic
537.
538.       //Si el cliente no se conecta reintenta la conexión
539.       if(!client.connected()){
540.         bool aux = reconnect();
541.         if(!aux){
542.           detachInterrupt(digitalPinToInterrupt(pulsador_reset));//Al
   salir del estado 3 desactivo la interrupcion
543.           break;
544.         }
545.         subscripcion_topics();//Me subscribo a los topic preconfigurados
   si se ha conectado al broker
546.       }
547.       client.loop();
548.
549.       //Leer temperatura
550.       long now = millis();
551.       if(now - lastMsg > rastreo){
552.         leer_temperatura();
553.         lastMsg = now;
554.       }
555.
556.       //Enviar temperatura por tiempo o por incremento de temperatura en
   valor absoluto
557.       long now2 = millis();
558.       float resta = fabs(tempC - old_tempC);
559.
560.       if((compara_mayor(delta,resta,1000)== 2) || (now2 - lastMsg2 >
   max_envio)){
561.
562.         Serial.println("\nResta temperaturas: ");
563.         Serial.print(resta);
564.         Serial.println("\n");
565.         old_tempC = tempC;
566.         int aux=old_tempC*1000000;
567.
568.         snprintf (msg, 75, "%d", aux/1000000);
569.         Serial.print("\nTemperatura enviada: ");
570.         Serial.print(msg);
571.         Serial.println("\n");
572.         client.publish(topic_tempActual, msg, true);
573.         lastMsg2 = now2;
574.       }
575.
576.       if(WiFi.status() != WL_CONNECTED){
577.         Serial.println("Conexion WIFI perdida...Se reintentara de
   nuevo...");
578.         detachInterrupt(digitalPinToInterrupt(pulsador_reset));//Al salir
   del estado 3 desactivo la interrupcion
579.         estado = 2;
580.       }
```

```
581.        break;//Fin case 3
582.    }
583.
584. }//Fin loop()
```

8.2. Programa Actuador

Actuador.ino

```
1. /******************** DECLARACIÓN DE
   LIBRERIAS*******************************/
2. #include "ESP8266WiFi.h" //Libreria para la tarjeta de desarrollo nodeMCU
3. #include "WiFiClient.h"
4. #include "ESP8266WebServer.h"
5. #include "DNSServer.h" //Libreria para servidor DNS. Utilizada para el
   portal cautivo.
6. #include "PubSubClient.h"  //Libreria MQTT para arduino
7. #include "EEPROM.h" //Libreria para la memoria EEPROM
8.
9. /******************** DECLARACION DE
   VARIABLES*******************************/
10.  //Topics
11.  const char* topic_Consigna = "ACTUADOR/Consigna_OnOff";
12.  const char* topic_actuadorEstado = "ACTUADOR/Estado";
13.  const char* topic_MaxEnviarActuador = "ACTUADOR/Max_enviar";
14.
15.  //Variables mensaje
16.  long lastMsg = 0;
17.  long lastMsg2 = 0;
18.  char msg[50];
19.
20.  //Asignacion GPIO a los LED
21.  int LED_AP = 5;
22.  int LED_WiFi = 4;
23.  int ONLed = 13;
24.  int Salida_Actuador = 0;
25.  int pulsador_reset = 12;
26.
27.  //Variable intentos de conexion wifi STA, cada intento corresponde a
   medio segundo
28.  int intentos = 0;
29.
30.  //Variable consigna recibida;
31.  int recibido = 0;
32.
33.  //Variable de estado del dispositivo:
34.  //  0-Crear AP
35.  //  1-AP creado, esperando recibir las credenciales
36.  //  2-Credenciales recibidas, se pasa a modo STA y se conecta al WiFi
37.  //  3-Conectado al WiFi, el actuador llevará a cabo sus tareas
   especificas
38.  int estado=0;
39.
40.  //Contador de pulsos del pulsador
41.  volatile int cont = 0;
42.
43.  //Variables recibidas
44.  int ON_OFF = 0;
45.  long max_envio = 60000;
46.
47.  int ON_OFF_old = 0;
```

```
48.  int cambiar=0;
49.
50.  //Variables para la construccion de la pagina web
51.  String cabecera =
52.  "<!DOCTYPE HTML >"
53.  "<html>"
54.  "  <head>"
55.  "    <title>TempSensor - WiFi Setup</title>"
56.  "    <meta http-equiv=\"Content-Type\" content=\"text/html; charset=utf-
     8\">"
57.  "    <meta name=\"viewport\" content=\"width=device-width, initial-
     scale=1\">"
58.  "  </head>"
59.  "  <body style=\"background-color:#F3F4F4\";>"
60.  "    <div>"
61.  "      <div style=\"background-color:#292D3C\"; align=\"middle\">"
62.  "        <h2
     style=\"color:#F3F4F4\";><br>🌡CONTROL<br>DE<br>TEMPERATURA<br></h2>"
63.  "        <h2 style=\"color:#F3F4F4\";>[Actuador]<br><br>WiFi
     SetUp<br><br></h2>"
64.  "      </div>";
65.
66.  String lista_ssid = "";
67.
68.  String pssw_broker =
69.  "          <tr>"
70.  "             <td>‣</td>"
71.  "             <th align=\"left\">CONTRASEÑA:</th>"
72.  "          </tr>"
73.  "          <tr>"
74.  "             <td></td>"
75.  "             <td><input name=\"password\" type=\"password\"
     id=\"myInput\"></td>"
76.  "             <td align=\"center\"><input type=\"checkbox\"
     onclick=\"myFunction()\"></td>"
77.  "             <td align=\"center\"> 👁</td>"
78.  "          </tr>"
79.  "          <tr>"
80.  "             <td>‣</td>"
81.  "             <th align=\"left\">BROKER:</th>"
82.  "          </tr>"
83.  "          <tr>"
84.  "             <td></td>"
85.  "             <td><input name=\"broker\" type=\"text\"></td>"
86.  "             <td></td>"
87.  "             <td></td>"
88.  "          </tr>"
89.  "    </table><br>";
90.
91.  String botones =
92.  "            <div align=\"middle\">"
93.  "             <input type=\"submit\" value=\"Enviar\">"
94.  "             <input type=\"reset\" value=\"Borrar\">"
95.  "             <br><br>"
96.  "             <p>*En caso de no aparecer la red WiFi deseada, refresque
     la pagina refresque la pagina o escanee de nuevo pulsando:[ ↺ ].</p>"
97.  "           </div>";
98.
99.  String callback_submit =
100. "      <div  align=\"middle\">"
```

```
"        <h3>✓ Credenciales enviados.</h3><br><br><br><br><br><br>"
"      </div>";

String pie_pagina =
"           <div style=\"background-color:#419D78\"; align=\"middle\">"
"             <p><i><br>Configuración de la conexión<br><br></i></p>"
"           </div>"
"        </form>"
"      </div>"
"    </div>"
"  <script>"
"  function myFunction() {"
"    var x = document.getElementById(\"myInput\");"
"    if (x.type === \"password\") {"
"      x.type = \"text\";"
"    } else {"
"      x.type = \"password\";"
"    }"
"  }"
"  </script>"
"  </body>"
"</html>";

//Datos conexion punto de acceso (AP) WiFi
const char ssid_ap[] = "Actuador_WiFi_setup";
const char password_ap[] = "0123456789";

//Variables almacenamiento credenciales red WiFi de casa
char ssid[50];
char password[50];

//Datos del broker MQTT
char mqtt_server[50];

//Declaración servidor DNS
const byte DNS_PORT = 53;
DNSServer dnsServer;

//Declaracion servidor
ESP8266WebServer server(80);

//Cliente wifi
WiFiClient espClient;
PubSubClient client(espClient);

/******************* DECLARACIÓN DE
   FUNCIONES*****************************/

//Función WIFI AP punto de acceso wifi
void setup_wifi_ap(){
  boolean result_ap = false; //Variable resultado de intentar crear
   access point;
  digitalWrite(LED_WiFi, LOW);
  digitalWrite(LED_AP, HIGH);
  Serial.print("\n\n---> MODO WIFI AP\n\n");

  while(result_ap==false){
    WiFi.disconnect(); //Desconectamos el WiFi
    WiFi.mode(WIFI_AP); //Establecemos el modo de WiFi como Access Point
    result_ap = WiFi.softAP(ssid_ap,password_ap); //Creamos el punto de
   acceso wifi
```

```
159.     delay(100);
160.     if(result_ap == true){
161.       Serial.println("Access Point creado con exito\n");
162.       Serial.print("Direccion IP Access Point - por defecto: ");
   //Imprime la dirección IP
163.       Serial.println(WiFi.softAPIP());
164.       Serial.print("Direccion MAC Access Point: ");
   //Imprime la dirección MAC
165.       Serial.println(WiFi.softAPmacAddress());
166.
167.       dnsServer.start(DNS_PORT, "*",IPAddress(192, 168, 4, 1)); //Arranco
   el servidor DNS para el portal cautivo
168.                                                                  //en la
   dirección IP 192.168.4.1 del NodeMCU.
169.       server.onNotFound([]() {
170.         escanear_redes();
171.       });
172.
173.       //Declaramos las llamadas a las funciones de las peticiones HTTP
174.       server.on("/", handleRoot);
175.       server.on("/", escanear_redes);
176.       server.on("/submit", handleSubmit);
177.       server.on("/escanear", escanear_redes); //Escanea las redes wifi
   disponibles
178.       server.begin();
179.       Serial.println("Servidor HTTP creado");
180.       estado = 1; //Se cambia el estado a 1. El AP ha sido creado y se
   mantiene a la espera de las credenciales.
181.     }
182.     else
183.     {
184.       Serial.println("Fallo al crear el Access Point");
185.
   Serial.print("\nReintentando");delay(200);Serial.print(".");delay(200);Seri
   al.print(".");delay(200);Serial.println(".");
186.     }
187.   }
188. }//Fin setup_wifi_ap()
189.
190. //Función de envio de pagina web cuando se esta en AP mode
191. void handleRoot()
192. {
193.   server.send(200, "text/html", cabecera + lista_ssid + pssw_broker +
   botones + pie_pagina);
194. }//Fin handleRoot()
195.
196. //Función de recepcion de SSID y PASSWORD y BROKER de la pag web
197. void handleSubmit(){
198.   String SSIDvalue;
199.   String PASSWORDvalue;
200.   String BROKERvalue;
201.
202.   Serial.println("\n**DATOS RECIBIDOS PAGINA WEB**\n");
203.
204.   if (server.args() > 0 ) {
205.     for ( int i = 0; i < server.args(); i++ ) {
206.       if (server.argName(i) == "ssid") {
207.          SSIDvalue = server.arg(i);
208.          SSIDvalue.toCharArray(ssid, 50);
209.          Serial.print("SSID recibido: ");
210.          Serial.println(ssid);
```

```
211.       }else if(server.argName(i) == "password"){
212.          PASSWORDvalue = server.arg(i);
213.          PASSWORDvalue.toCharArray(password, 50);
214.          Serial.print("PASSWORD recibida: ");
215.          Serial.println(password);
216.       }else if(server.argName(i) == "broker"){
217.          BROKERvalue = server.arg(i);
218.          BROKERvalue.toCharArray(mqtt_server, 50);
219.          Serial.print("BROKER recibido: ");
220.          Serial.println(mqtt_server);
221.       }
222.     }
223.   }
224.
225.   Serial.println("\n*****************************\n");
226.
227.   //Cuando se reciben los datos, se lanza una pagina web que muestra que
   los datos han sido recibidos
228.   server.send(200, "text/html", cabecera + callback_submit +
   pie_pagina);
229.   delay(100);
230.   guardarCredenciales(); //Se guardan los credenciales en la memoria
   EEPROM
231.   cont=0; //Ponemos el contador de pulsos del pulsador a cero
232.   estado=2; //Se cambia a estado 2. Las credenciales han sido recibidas y
   se pasa al modo STA para conectarse al WiFi
233. }//Fin handleSubmit()
234.
235. //Función que escanea las redes WIFI disponibles y genera su parte de
   código html
236. void escanear_redes(){
237.   int n = WiFi.scanNetworks(); //Número de redes encontradas
238.   if(n == 0){ //Si no hay redes encontradas muestra un mensaje de que no
   hay redes disponibles
239.     Serial.println("No se han encontrado redes WiFi disponibles");
240.     lista_ssid = "No se han encontrado redes WiFi disponibles";
241.   }else{ //Si encuentra redes WiFi, compone la parte de codigo html para
   la pagina web
242.     lista_ssid =
243.     "      <div  align=\"middle\">"
244.     "        <h3>Por favor, complete los credenciales:</h3>"
245.     "        <form action=\"http://192.168.4.1/submit\" method=\"post\">"
246.     "        <table>"
247.     "        <tr>"
248.     "           <td>‣</td>"
249.     "           <th align=\"left\">SSID:</th>"
250.     "           <td></td>"
251.     "           <td></td>"
252.     "        </tr>"
253.     "        <tr>"
254.     "           <td></td>"
255.     "           <td><select name=\"ssid\">";
256.
257.     for(int i = 0; i < n; i++){//Genera la lista de redes WiFi
258.       lista_ssid = (lista_ssid) + "<option value= \"" + WiFi.SSID(i) +
   "\">" + WiFi.SSID(i)+ "</option>";
259.       delay(10);
260.     }
261.     //Se forma la parte de la página web de la lista de redes
262.     lista_ssid = (lista_ssid) + "</select></td>";
```

```
263.     lista_ssid = (lista_ssid) + "<td align=\"center\"><button
   type=\"submit\"
   formaction=\"http://192.168.4.1/escanear\">↺</button></td>";
264.     lista_ssid = (lista_ssid) + "<td align=\"center\"> *</td></tr>";
265.
266.     handleRoot(); //Llama a la función que genera y envía al navegador la
   pagina completa
267.   }
268. }//Fin escanear_redes()
269.
270. //Función WIFI STA conexión a red wifi externa
271. void setup_wifi(){
272.   digitalWrite(LED_AP, HIGH);
273.   bool estado_LED_WiFi = LOW;
274.   long lastMsg = 0;
275.   Serial.print("\n\n---> MODO WIFI STA\n\n");
276.
277.   Serial.print("Conectando a... ");
278.   Serial.println(ssid);
279.
280.   WiFi.softAPdisconnect(); //Desconectamos el modo AP del wifi
281.   WiFi.disconnect(); //Desconectamos el wifi
282.
283.   WiFi.mode(WIFI_STA); //Establecemos modo wifi STA
284.   WiFi.begin(ssid, password); //Conexión a la red WiFi
285.
286.   //Mientras no se ha conectado al WiFi espera 30 segundos
287.   while (WiFi.status() != WL_CONNECTED){
288.     Serial.print(".");
289.     delay(500);
290.     intentos ++;
291.     digitalWrite(LED_WiFi, estado_LED_WiFi);
292.     estado_LED_WiFi = !estado_LED_WiFi; //Se cambia el estado para que el
   led parpadee
293.     //Cada intento equivale a medio segundo: 60/2= 30 segundos
294.     if(intentos > 60){ //Si supera los 30 segundos, es decir los 60
   intentos volverá al modo AP
295.       Serial.println("\n¡No se pudo conectar!\nEs posible que los
   credenciales sean invalidos.");
296.       Serial.println("*NOTA: Entrara en el modo AP de nuevo. Configure la
   red WIFI nuevamente.");
297.       estado = 0; //Se cambia estado 0, que generará de nuevo el punto de
   acceso AP
298.       intentos = 0; //Pone el contador de tiempo a cero
299.       return; //Se sale de la función para que no ejecute el codigo de
   abajo
300.     }
301.   }
302.
303.   //Si se consigue conectar al WiFi se ejecutara el siguiente código:
304.
305.   digitalWrite(LED_WiFi, HIGH); //Se enciende el led de conexión que
   indica que esta conectado al WiFi
306.
307.   Serial.println("");
308.   Serial.print("Conectado a la red:");
309.   Serial.println(ssid);
310.   Serial.print("Direccion IP: ");
311.   Serial.println(WiFi.localIP());
312.   Serial.println("\n");
```

```
313.   estado = 3; // Se cambia al estado 3. Se ha conectado al WiFi y el
   sensor hará sus tareas.
314.   intentos = 0;
315.
316. }//Fin setup_wifi()
317.
318. //Función cargar credenciales de la EEPROM
319. int cargarCredenciales() {
320.   EEPROM.begin(512);
321.   EEPROM.get(0, ssid);
322.   EEPROM.get(0 + sizeof(ssid), password);
323.   EEPROM.get(0 + sizeof(ssid) + sizeof(password), mqtt_server);
324.   char ok[4 + 1]; //Esta variable nos sirve para comprobar si hay
   credenciales almacenadas
325.   EEPROM.get(0 + sizeof(ssid) + sizeof(password) + sizeof(mqtt_server),
   ok);
326.   EEPROM.end();
327.
328.   //Si no coindice la variable ok con la cadena de caracateres no hay
   nada almacenado
329.   if (String(ok) != String("OKEY")) {
330.     ssid[0] = 0;
331.     password[0] = 0;
332.     mqtt_server[0] = 0;
333.     return 0; //Devuelve 0 cuando no hay credenciales almacenadas
334.   }
335.
336.   //Si hay credenciales almacenadas sobreescribe las variables de
   programa de estas
337.   Serial.println("Credenciales recuperados:");
338.   Serial.println(ssid);
339.   Serial.println(strlen(password) > 0 ? "********" : "<no password>");
340.   Serial.println(mqtt_server);
341.   return 2; //Devuelve 0 cuando no hay credenciales almacenadas
342. }//Fin cargarCredenciales()
343.
344. //Función que guarda los credenciales en la EEPROM
345. void guardarCredenciales() {
346.   EEPROM.begin(512);
347.   EEPROM.put(0, ssid);
348.   EEPROM.put(0 + sizeof(ssid), password);
349.   EEPROM.put(0 + sizeof(ssid) + sizeof(password), mqtt_server);
350.   char ok[4 + 1] = "OKEY"; //Esta variable nos sirve para comprobar si
   hay credenciales almacenadas
351.   EEPROM.put(0 + sizeof(ssid) + sizeof(password) + sizeof(mqtt_server),
   ok);
352.   EEPROM.commit();
353.   EEPROM.end();
354. }//Fin guardarCredenciales()
355.
356. //Función que elimina los credenciales almacenados en la EEPROM
357. void eliminarCredenciales() {
358.   bool estado_LED_AP = LOW;
359.   EEPROM.begin(512);
360.   //Para borrar las credenciales sobreescribimos la memoria EEPROM con
   ceros
361.   for (int i = 0; i < 512; i++){
362.     EEPROM.write(i, 0);
363.   }
364.   //Parpadea para dar una señal visual de que se ha borrado la EEPROM
365.   for (int j=0; j<6; j++){
```

```
366.     digitalWrite(LED_AP, estado_LED_AP);
367.     delay(500);
368.     estado_LED_AP = !estado_LED_AP;
369.   }
370.   Serial.println("Credenciales borrados de la memoria EEPROM");
371.   EEPROM.end();
372. }//Fin eliminarCredenciales()
373.
374. //Función para resetear el NodeMCU (con la interrupcion del pulsador) y
   volver al estado de creacion punto de acceso AP
375. void Reset(){
376.   estado=0; //Se pasa al estado 0, de creación del punto de acceso AP.
377.   ON_OFF=0;
378.   actuador();
379. }//Fin Reset()
380.
381. //Función que cuenta las veces que pulsas el pulsador (interrupcion en el
   modo AP)
382. void contador(){
383.   cont++;
384. }//Fin contador()
385.
386. //Función que recibe los mensajes de los topic suscritos
387. void callback(char* topic, byte* payload, unsigned int length) {
388.   String s; //Auxiliar para la conversión del payload
389.
390.   Serial.print("Mensaje recibido [");
391.   Serial.print(topic);
392.   Serial.print("]: ");
393.
394.   if(strcmp(topic, topic_Consigna) == 0){ //con strcmp comparo los dos
   string
395.     payload[length] = '\0';
396.     s = String((char*)payload);
397.     ON_OFF  = s.toInt();
398.     Serial.println(ON_OFF);
399.     if(ON_OFF==ON_OFF_old){
400.       cambiar = 0;
401.     }else{
402.       ON_OFF_old = ON_OFF;
403.       cambiar=1;
404.     }
405.     recibido = 1;
406.   }else if(strcmp(topic,topic_MaxEnviarActuador) == 0){
407.     payload[length] = '\0';
408.     s = String((char*)payload);
409.     max_envio = s.toInt();
410.     Serial.println(max_envio);
411.   }
412. }//Fin callback()
413.
414. //Función en la que el cliente se reconecta en caso de desconectarse o no
   estar conectado
415. bool reconnect() {
416.   digitalWrite(LED_AP, HIGH);
417.   // Creain ID de cliente aleatorio
418.   String clientId = "ESP8266Client-";
419.   clientId += String(random(0xffff), HEX);
420.   // Bucle hasta que se conecte al broker
421.   while (!client.connected()) {
422.     Serial.print("Esperando conexion MQTT...");
```

```
423.
424.     // Attempt to connect
425.     if (client.connect(clientId.c_str())) {
426.       digitalWrite(LED_AP, LOW);
427.       Serial.println("conectado");
428.       return true;
429.     } else {
430.       Serial.print("failed, rc=");
431.       Serial.print(client.state());
432.       Serial.println(" intentar en 5 segundos");
433.       digitalWrite(Salida_Actuador, LOW); //Si pierdo la conexión pongo
   el actuador en el estado seguro(parada)
434.       // Wait 5 seconds before retrying
435.       delay(5000);
436.     }
437.
438.     if(estado == 0){ //Si pulso el pulsador de cambio de modo
439.                     // salgo del while y del reconnect para poder cambiar
   al modo AP
440.       return false;
441.     }
442.   }
443. }//Fin reconnect()
444.
445. //Función que se suscribe a los topic preconfigurados
446. void subscripcion_topics(){
447.   Serial.println("Me suscribo a los TOPIC preconfigurados");
448.
449.   client.subscribe(topic_Consigna); // Orden de encendido o apagado de la
   calefacción
450.   client.subscribe(topic_MaxEnviarActuador); // Tiempo máximo de envío
   del estado de la caldera
451.
452. }//Fin subscripcion_topics
453.
454. //Función que enciendo o apaga la salida del  actuador
455. void actuador(){
456.
457.   if(ON_OFF == 1){
458.     digitalWrite(Salida_Actuador, HIGH); //Enciende la caldera
459.   }else if(ON_OFF == 0){
460.     digitalWrite(Salida_Actuador, LOW); //Apaga la caldera
461.   }
462. }
463.
464. /********************************** DECLARACIÓN DEL SETUP()
   ****************************************/
465. void setup() {
466.   delay(1000);
467.   Serial.begin(115200);
468.   randomSeed(micros());
469.
470.   //Define los pines que se van a usar como salidas o entradas
471.   pinMode(LED_AP, OUTPUT);
472.   pinMode(ONLed, OUTPUT);
473.   pinMode(LED_WiFi, OUTPUT);
474.   pinMode(pulsador_reset, INPUT);
475.   pinMode(Salida_Actuador, OUTPUT);
476.
477.   Serial.println("\t ACTUADOR ENCENDIDO");
478.   digitalWrite(ONLed, HIGH);
```

```
479.   Serial.println ("\t Comprobacion EEPROM...");
480.   estado = cargarCredenciales(); //Comprueba si hay credenciales
   almacenadas en la EEPROM y asigna el estado del dispositivo
481.   if(estado == 0){
482.     Serial.println("EEPROM vacia. Nada que recuperar");
483.   }
484. }//Fin setup()
485.
486.
487. /********************************** DECLARACION DEL LOOP()
   ****************************************/
488. void loop() {
489.   //Comprueba si hay credenciales almacenadas en la EEPROM y asigna el
   estado del dispositivo
490.   switch(estado){
491.     //Modo 0 - Modo creación AP: El dispositivo crea el punto de acceso
   AP
492.     case 0:
493.       setup_wifi_ap();
494.       break;//Fin case 0
495.
496.     //Modo 1 - Modo AP: El punto de acceso está creado, se lanza la
   pagina web y se espera recibir las credenciales
497.     case 1:
498.
   attachInterrupt(digitalPinToInterrupt(pulsador_reset),contador,FALLING);//S
   e declará la interrupción para borrar la EEPROM
499.
500.       //Si el pulsador se presiona 3 veces se borra la EEPROM
501.       if(cont>=3){
502.         Serial.println("PULSADOR ACCIONADO 3 VECES!!");
503.         eliminarCredenciales();
504.         cont=0;
505.         detachInterrupt(digitalPinToInterrupt(pulsador_reset)); //Se
   elimina la interrupción para borrar la EEPROM
506.       }
507.
508.       dnsServer.processNextRequest(); //Portal cautivo
509.       server.handleClient();
510.       detachInterrupt(digitalPinToInterrupt(pulsador_reset)); //Se
   elimina la interrupción para borrar la EEPROM
511.       break;//Fin case 1
512.
513.     //Modo 2 - Modo STA: Credenciales recibidas, se conecta a la red WiFi
   introducida por el usuario
514.     case 2:
515.       setup_wifi();
516.
517.       break;//Fin case 2
518.
519.     //Modo 3 - Modo funcionamiento dispositivo. El actuador encenderá o
   apagará la caldera
520.     case 3:
521.
   attachInterrupt(digitalPinToInterrupt(pulsador_reset),Reset,FALLING);//Acti
   va interrupcion que accede al modo configuracion wifi (modo AP)
522.       client.setServer(mqtt_server, 1883); //Crea el cliente MQTT y le
   indica a que broker se tiene que conectar y en que puerto
523.       client.setCallback(callback); //Setea el callback de mensajes
   recibidos de los topic
524.
```

```
525.       //Si el cliente no se conecta reintenta la conexión
526.       if(!client.connected()){
527.         bool aux = reconnect();
528.         if(!aux){
529.           detachInterrupt(digitalPinToInterrupt(pulsador_reset));//Al
   salir del estado 3 desactivo la interrupcion
530.           break;
531.         }
532.         subscripcion_topics();//Me subscribo a los topic preconfigurados
   si se ha conectado al broker
533.       }
534.       client.loop();
535.
536.       //Acciona la salida del actuador
537.       actuador();
538.
539.       //Modo seguro si no recibe valor de la consigna en 2 minutos.
540.       long now2 = millis();
541.       if(now2-lastMsg2 > 120000){
542.         ON_OFF=0;
543.         actuador();
544.         lastMsg2 = now2;
545.         Serial.print("\nMODO SEGURO");
546.
547.       }else if(recibido ==1){
548.         recibido = 0;
549.         lastMsg2 = now2;
550.       }
551.
552.       //Envia estado
553.       long now = millis();
554.       if(now-lastMsg > max_envio || cambiar == 1){
555.         if(ON_OFF==1){
556.           snprintf (msg, 75, "1");
557.         }else if(ON_OFF==0){
558.           snprintf (msg, 75, "0");
559.         }
560.         Serial.print("\nEstado de la caldera enviado: ");
561.         Serial.print(msg);
562.         Serial.println("\n");
563.         client.publish(topic_actuadorEstado,msg,true);
564.         lastMsg = now;
565.         cambiar = 0;
566.       }
567.
568.       if(WiFi.status() != WL_CONNECTED){
569.         Serial.println("Conexion WIFI perdida...Se reintentara de
   nuevo...");
570.         detachInterrupt(digitalPinToInterrupt(pulsador_reset));//Al salir
   del estado 3 desactivo la interrupcion
571.         ON_OFF=0;
572.         actuador();
573.         estado = 2;
574.       }
575.       break;//Fin case 3
576.   }//Fin switch()
577.
578. }//Fin loop()
```

8.3. Programa control principal

8.3.1. Main

main.cpp

```
1. /*****************************************************************************
   *********/
2. /*
   */
3. /*                                  TRABAJO FIN DE GRADO
   */
4. /*                                    Control principal
   */
5. /*
   */
6. /*****************************************************************************
   *********/
7.
8. #include <iostream>
9. #include "mosquittomqtt.h"
10.
11.  using namespace std;
12.
13.  float tempC_media; //Variable temperatura media
14.  int tempMin;
15.  int tempMax;
16.  string str_tempC_media;
17.  string str_tempMin;
18.  string str_tempMax;
19.
20.  //Variables de la fecha
21.  int dia=0;
22.  int hora=6;
23.  int mins=0;
24.  int mins_old=0;
25.  int seg=0;
26.  int seg_old=0;
27.
28.  int intervalo = 0;//Intervalo del dia
29.  int intervalo_old = 7; // Intervalo antiguo del dia
30.
31.  //Variables del totalizador de horas de funcionamiento
32.  int estadisticas_horas_acumuladas = 0;
33.  int estadisticas_mins_acumulados = 0;
34.  bool estadisticas_permisivo_contador = 0;
35.  string horometro;
36.
37.  //Variables del setpoint
38.  int setpointTempC=15;
39.  int setpointTempC_old=0;
40.
41.  //Variables de modo semi automatico
42.  int modoSemiAUT = 0;
43.  int modoSemiAUT_old = 0;
44.
45.  //Variable axuliar para actualizar el sp
46.  int actualizar_SP = 1;
47.
48.  bool on = 0, off = 1; // para que solo accione la calefaccion una vez
49.
```

```
50.  string Dia_Hora;
51.  string intervalo_str;
52.
53.
54.  /*DECLARACIÓN DE LA ESTRUCTURA "TEMPERATURA"*/
55.  struct Nodo
56.  {
57.      int lectura;
58.      string timestamp;
59.      /*Sobrecargamos el operador '<' para que a la hora de comparar sepa
     que compara*/
60.      bool operator <(Nodo & n)
61.      {
62.          return lectura <  n.lectura;
63.      }
64.  };
65.
66.  /**************************************************************************
     ************/
67.  /* Nombre: void Insertar(list<T> &lista,  Nodo &temp)
     */
68.  /* Descripción: Inserta un valor en la lista enlazada segun el criterio
     FIFO        */
69.  /* Argumentos: Lista y temperatura nueva a insertar.
     */
70.  /* Valor Devuelto: Ninguno
     */
71.  /**************************************************************************
     ************/
72.  template<typename T>
73.  void Insertar(list<T> &lista,  Nodo &temp)
74.  {
75.      if(lista.empty()) // Si lista esta vacia.
76.      {
77.           lista.emplace( lista.begin(), temp ); // Insertamos el primer
     valor.
78.      }
79.      else
80.      {
81.          lista.push_back(temp); // Insertamos un valor segun el criterio
     FIFO.
82.      }
83.  } // Fin de la funcion Insertar().
84.
85.  /**************************************************************************
     ************/
86.  /* Nombre: void Borrar(list<T> &lista)
     */
87.  /* Descripción: Borrar un valor de la lista enlazada segun el criterio
     FIFO.        */
88.  /* Argumentos: Lista
     */
89.  /* Valor Devuelto: Ninguno
     */
90.  /**************************************************************************
     ************/
91.  template<typename T>
92.  void Borrar(list<T> &lista)
93.  {
94.      if(lista.size()>=1) // Si el tamaño de la lista es mayor o igual a 1.
95.      {
```

```
96.          lista.pop_front(); // Borramos un valor segun el criterio FIFO.
97.      }
98.  } //Fin de la funcion Borrar()
99.
100. /*************************************************************************
   ************/
101. /* Nombre: void Imprimir(list<T> &lista)
   */
102. /* Descripción: Imprime por pantalla los valores contenidos en la lista
   enlazada    */
103. /* Argumentos: Lista.
   */
104. /* Valor Devuelto: Ninguno
   */
105. /*************************************************************************
   ************/
106. template<typename T>
107. void Imprimir(list<T> &lista)
108. {
109.     typename list<T> ::iterator it; // Iterador que recorrerá la lista.
110.     int n=1; // Declaramos un contador "n" que nos indica la posicion en
   la que se encuentran
111.              //los valores en la cola.
112.
113.     cout<<"\n\t             ------------------------------------------
   "<<endl;
114.     cout<<"\t            | LECTURA DE LOS DATOS CONTENIDOS EN LA COLA
   |"<<endl;
115.     cout<<"\t             ------------------------------------------
   \n"<<endl;
116.
117.     if (lista.size()>0) // Si el tamaño de la lista es mayor que cero.
118.     {
119.         cout<<"\t\t ---------------------------------------------------
   "<<endl;
120.         cout<<"\t\t|  Núm  |Temperatura(°C)|          Tiempo
   |"<<endl;
121.
122.         for(it=lista.begin(); it!=lista.end(); it++) // Con el iterador
   recorremos la lista y vamos imprimiento
123.         {                                            // los valores que
   va adquiriendo 'it'.
124.
125.             cout<<"\t\t|-------|---------------|-------------------------
   ---|"<<endl;
126.             cout<<"\t\t|  -"<<n<<"-\t|      " << it->lectura;
127.             cout<< " \t| "<<it->timestamp ;//<< "|"<< endl;
128.
129.             n++; // Incremeta en una unidad, a medida que se imprimen los
   datos.
130.         }
131.
132.     cout<<"\t\t ---------------------------------------------------
   "<<endl;
133.     }
134.     else // Si la lista no contiene ningun valor.
135.     {
136.         cout<<"\n\t\t\t       ¡LISTA VACÍA!"<<endl;
137.     }
138. } // Fin de la funcion Imprimir().
139.
```

```
140. /*************************************************************************
   ************/
141. /* Nombre: void MaxMin(list<T> &lista)
   */
142. /* Descripción: Compara los valores de la lista enlazada y nos dice cual
   es máximo  */
143. /*              y mínimo.
   */
144. /* Argumentos: Lista.
   */
145. /*Valor Devuelto: Ninguno
   */
146. /*************************************************************************
   ************/
147. template<typename T>
148. void MaxMin(list<T> &lista,mosquittoMQTT* mqttHdl)
149. {
150.
151.     typename list<T> :: iterator it; // Iterador que recorrera la lista.
152.     if(lista.size()==1) // Si el tamaño de la lista en igual a 1.
153.     {
154.         /*Se sabe que el maximo y el minimo son el mismo valor puesto que
   el
155.         tamaño de la lista es 1*/
156.
157.         /*Con la llamada al algoritmo 'min_element' obtenemos el minimo
   valor
158.         contenido en la lista*/
159.         it = min_element(lista.begin(),lista.end());
160.         cout<<"\n\t\t       ------------------------------------------
   "<<endl;
161.         cout<<"\t\t      | LA TEMPERATURA MÁXIMA Y MÍNIMA SON IGUALES
   |"<<endl;
162.         cout<<"\t\t       ------------------------------------------
   \n"<<endl;
163.
164.         cout<<"\t\t    ------------------------------------------
   "<<endl;
165.         cout<<"\t\t   |Temperatura(°C)|           Tiempo
   |"<<endl;
166.         cout<<"\t\t   |---------------|--------------------------
   |"<<endl;
167.         cout<<"\t\t   |     " << it->lectura << "\t   | ";
168.         cout<<it->timestamp ;//<< "|"<< endl;
169.
170.         cout<<"\t\t    ------------------------------------------
   "<<endl;
171.         str_tempMax = to_string(it->lectura) + " °C";
172.         str_tempMin = to_string(it->lectura) + " °C";
173.
174.     }
175.     else // Si la lista tiene un tamaño distinto de uno
176.     {
177.         /*Con la llamada al algoritmo 'max_element' obtenemos el mayor
   valor
178.         contenido en la lista*/
179.         it= max_element(lista.begin(),lista.end());
180.
181.         cout<<"\n\t                  ---------------------"<<endl;
182.         cout<<"\t                | TEMPERATURA MÁXIMA |"<<endl;
183.         cout<<"\t                 ---------------------\n"<<endl;
```

```
184.
185.         cout<<"\t\t    ------------------------------------------
   "<<endl;
186.         cout<<"\t\t   |Temperatura(°C)|           Tiempo
   |"<<endl;
187.         cout<<"\t\t   |---------------|--------------------------
   |"<<endl;
188.         cout<<"\t\t   |     " << it->lectura << "\t   | ";
189.         cout<<it->timestamp ;//<< "|"<< endl;
190.
191.         cout<<"\t\t    ------------------------------------------
   "<<endl;
192.         str_tempMax = to_string(it->lectura) + " °C";
193.
194.         /*Con la llamada al algoritmo 'min_element' obtenemos el minimo
   valor
195.         contenido en la lista*/
196.         it = min_element(lista.begin(),lista.end());
197.
198.         cout<<"\n\t\t                    ---------------------"<<endl;
199.         cout<<"\t\t                  | TEMPERATURA MÍNIMA |"<<endl;
200.         cout<<"\t\t                   ---------------------\n"<<endl;
201.
202.         cout<<"\t\t    ------------------------------------------
   "<<endl;
203.         cout<<"\t\t   |Temperatura(°C)|           Tiempo
   |"<<endl;
204.         cout<<"\t\t   |---------------|--------------------------
   |"<<endl;
205.         cout<<"\t\t   |     " << it->lectura << "\t   | ";
206.         cout<<it->timestamp ;//<< "|"<< endl;
207.
208.         cout<<"\t\t    ------------------------------------------
   "<<endl;
209.
210.         str_tempMin = to_string(it->lectura) + " °C";
211.
212.     }
213.     mqttHdl->publish(NULL, mqttHdl-
   >topic_TempMax,str_tempMax.size(),str_tempMax.c_str(),0,true);
214.     mqttHdl->publish(NULL, mqttHdl-
   >topic_TempMin,str_tempMin.size(),str_tempMin.c_str(),0,true);
215. } // Fin de la funcion MaxMin().
216.
217. /*****************************************************************
   ************/
218. /* Nombre: void Tiempo()
   */
219. /* Descripción: Calcula la diferencia en segundos entre el año 2000 y la
   actualidad */
220. /* Argumentos: Ninguno.
   */
221. /* Valor Devuelto: Segundos.
   */
222. /*****************************************************************
   ************/
223. string Tiempo()
224. {
225.     int segundos; // Variable que almacena el tiempo.
226.     string fecha_hora;
227.     string dia_str;
```

```
228.     time_t timer; // Variable de la libreria ctime que nos indica el
   tiempo transcurrido desde 1970.
229.     segundos=time(&timer);
230.     fecha_hora = ctime (&timer);//Convertimos el tiempo en una string
231.     hora = atoi(fecha_hora.substr(11,2).c_str());
232.     mins = atoi(fecha_hora.substr(14,2).c_str());
233.     seg = segundos;
234.     dia_str = fecha_hora.substr(0,3);
235.     if(dia_str.compare("Mon")==0){
236.         dia = 0;
237.         dia_str = "Lun";
238.     }else if(dia_str.compare("Tue")==0){
239.         dia = 1;
240.         dia_str = "Mar";
241.     }else if(dia_str.compare("Wed")==0){
242.         dia = 2;
243.         dia_str = "Mie";
244.     }else if(dia_str.compare("Thu")==0){
245.         dia = 3;
246.         dia_str = "Jue";
247.     }else if(dia_str.compare("Fri")==0){
248.         dia = 4;
249.         dia_str = "Vie";
250.     }else if(dia_str.compare("Sat")==0){
251.         dia = 5;
252.         dia_str = "Sab";
253.     }else {
254.         dia = 6;
255.         dia_str = "Dom";
256.     }
257.
258.     Dia_Hora = dia_str + " " + to_string(hora);
259.     if(mins<10){
260.         Dia_Hora = Dia_Hora + ":0" + to_string(mins);
261.     }else{
262.         Dia_Hora = Dia_Hora + ":" + to_string(mins);
263.     }
264.
265.     return fecha_hora; // Devolvemos el valor de la variable segundos.
266. } // Fin de la funcion Tiempo().
267.
268. /*************************************************************************
   ************/
269. /* Nombre: float media()
   */
270. /* Descripción: Calcula el valor medio de los datos contenidos en la
   lista          */
271. /* Argumentos: Lista.
   */
272. /* Valor Devuelto: Media
   */
273. /*************************************************************************
   ************/
274. template<typename T>
275. float media(list<T> &lista){
276.
277.     typename list<T> :: iterator it; // Iterador que recorrera la lista.
278.     float sumatorio = 0;
279.     float media;
280.     int n = 0; // Contador tamaño de la lista
281.
```

```
282.     if(lista.size()<10)
283.     {
284.         cout<<"\n\t\t                    --------------------"<<endl;
285.         cout<<"\t\t                   | TEMPERATURA  MEDIA |"<<endl;
286.         cout<<"\t\t                    --------------------\n"<<endl;
287.
288.         cout<<"\n\t\t Valores insuficientes. \n\t\tMínimo 10 valores para
   calcular la media\n"<<endl;
289.
290.     }
291.     else
292.     {
293.         for(it=lista.begin(); it!=lista.end(); it++)
294.         {
295.             sumatorio = it->lectura + sumatorio;
296.             n++;
297.         }
298.
299.         media = sumatorio/n;
300.
301.         cout<<"\n\t\t                    --------------------"<<endl;
302.         cout<<"\t\t                   | TEMPERATURA  MEDIA |"<<endl;
303.         cout<<"\t\t                    --------------------"<<endl;
304.
305.         cout<<"\t\t                   |       "<<media<<"°C      |"<<endl;
306.         cout<<"\t\t                    --------------------\n"<<endl;
307.
308.     }
309.     return media;
310. }
311.
312. /*************************************************************************
   ************/
313. /* Nombre: void horometro_caldera()
   */
314. /* Descripción: Calcula el tiempo que lleva funcionando la caldera
   */
315. /* Argumentos: llamada, actuador.
   */
316. /* Valor Devuelto: Ninguno
   */
317. /*************************************************************************
   ************/
318. void horometro_caldera(int llamada, bool actuador){
319.     if(llamada == 0){
320.         if(actuador == 0){
321.             estadisticas_permisivo_contador = 0;
322.         }else{
323.             estadisticas_permisivo_contador = 1;
324.         }
325.     }else{
326.         if(estadisticas_permisivo_contador == 1){
327.             estadisticas_mins_acumulados ++;
328.             if(estadisticas_mins_acumulados > 59){
329.                 estadisticas_horas_acumuladas ++;
330.                 estadisticas_mins_acumulados = 0;
331.             }
332.         }
333.     }
334. }//fin horometro_caldera()
335.
```

```
336. /*************************************************************************
   ************/
337. /* Nombre: actuador()
   */
338. /* Descripción: enciende o apaga la salida del actuador en funcion de la
   temp       */
339. /* Argumentos: temp, setpoint, cliente mqtt
   */
340. /* Valor Devuelto: nada
   */
341. /*************************************************************************
   ************/
342. void actuador(int temp, int setpoint, mosquittoMQTT* mqttHdl){
343.     //if(compara_mayor(temp,setpoint,1000) == 2 && on == 0)
344.     if(temp<=setpoint && on == 0)
345.     {
346.         cout<<"\n\t\t **Se ENCIENDE la calefacción\n"<<endl;
347.         mqttHdl->publish(NULL, mqttHdl->topic_ConsignaOnOff,
   1,"1",0,true);
348.         on = 1;
349.         off = 0;
350.         horometro_caldera(0,1);
351.     }
352.     //if(compara_mayor(temp,setpoint,1000) == 1 && off == 0)
353.     if(temp > setpoint && off==0)
354.     {
355.         cout<<"\n\t\t **Se APAGA la calefacción\n"<<endl;
356.         mqttHdl->publish(NULL, mqttHdl-
   >topic_ConsignaOnOff,1,"0",0,true);
357.         on = 0;
358.         off = 1;
359.         horometro_caldera(0,0);
360.     }
361. }
362.
363. int intervalo_actual(int intervalos_hora[], int intervalos_mins[]){
364.     int posicion;
365.
366.     for(int i=0; i<6; i++){
367.         if(intervalos_hora[i]==hora){
368.             if(intervalos_mins[i]<=mins){
369.                 posicion = i;
370.                 break;
371.             }else if(i==0 && intervalos_mins[i]>mins){
372.                 posicion = 5;
373.                 break;
374.             }else{
375.                 posicion = i-1;
376.                 break;
377.             }
378.         }else if(intervalos_hora[i]<hora){
379.             if(i!=5 && intervalos_hora[i+1]>hora){
380.                 posicion = i;
381.                 break;
382.             }else if(i==5){
383.                 posicion = 5;
384.                 break;
385.             }
386.         }else if(i==0 && intervalos_hora[i]>hora){
387.             posicion = 5;
388.             break;
```

```
389.         }
390.     }
391.
392.     return posicion;
393. }
394.
395. /*PROGRAMA PRINCIPAL, MAIN*/
396. int main()
397. {
398.     /*DECLARACIÓN DE VARIABLES*/
399.     list<Nodo> lista; // Lista de temperaturas.
400.     Nodo valor; // Variable de tipo nodo para almacenar los valores
   introducidos por el usuario.
401.     Nodo temp_max; //Cariable de tipo nodo para temperatura maxima
402.     Nodo temp_min; //Variable de tipo nodo para temperatura minima
403.     tempC_media=0.00;
404.
405.     mosquittoMQTT * mqttHdl;
406.     mqttHdl = new mosquittoMQTT(NULL, "localhost", 1883);
407.
408.     setlocale(LC_CTYPE, "Spanish"); // Reconoce los caracteres de la
   ortografia del castellano.
409.
410.     /*Imprime los valores de tipo float con una precision decimal de 2.*/
411.     cout.setf(ios::fixed, ios::floatfield);
412.     cout.precision(2);
413.
414.     /*FIN DECLARACIÓN DE VARIABLES*/
415.
416.     while(1)
417.     {
418.         //Mantiene la conexion
419.         int res = mqttHdl->loop();
420.         if (res){
421.             mqttHdl->reconnect();
422.         }
423.         Tiempo();
424.
425.         if(mins!=mins_old){
426.             mqttHdl->publish(NULL, mqttHdl-
   >topic_DiaHora,Dia_Hora.size(),Dia_Hora.c_str(),0,true);
427.             horometro_caldera(1,0);
428.             if(estadisticas_horas_acumuladas < 10){
429.                 horometro = "0" +
   to_string(estadisticas_horas_acumuladas) + " horas y ";
430.             }else{
431.                 horometro = to_string(estadisticas_horas_acumuladas) + "
   horas y ";
432.             }
433.             if(estadisticas_mins_acumulados < 10){
434.                 horometro = horometro + "0" +
   to_string(estadisticas_mins_acumulados) + " minutos";
435.             }else{
436.                 horometro = horometro +
   to_string(estadisticas_mins_acumulados) + " minutos";
437.             }
438.             mqttHdl->publish(NULL, mqttHdl-
   >topic_Horometro,horometro.size(),horometro.c_str(),0,true);
439.
440.             if(mqttHdl->tempC<=setpointTempC)
441.             {
```

```
442.                 mqttHdl->publish(NULL, mqttHdl->topic_ConsignaOnOff,
   1,"1",0,true);
443.
444.             }
445.             if(mqttHdl->tempC > setpointTempC)
446.             {
447.                 mqttHdl->publish(NULL, mqttHdl-
   >topic_ConsignaOnOff,1,"0",0,true);
448.             }
449.
450.             mins_old = mins;
451.         }
452.
453.         //Calcula la temperatura media cuando recibe un dato nuevo
454.         if(mqttHdl->permit_tempC == 1)
455.         {
456.             //temperatura_media(mqttHdl->tempC);
457.             cout<<"\n\t             ------------------------------------
   "<<endl;
458.             cout<<"\t            | INTRODUCIR NUEVO VALOR EN LA COLA
   |"<<endl;
459.             cout<<"\t             ------------------------------------
   \n"<<endl;
460.             cout<<"\t"<<endl;
461.
462.             valor.lectura = mqttHdl->tempC;
463.             valor.timestamp=Tiempo(); // El tiempo se le asigna
   automaticamente en el instante que ha sido
464.                                       // introducido la lectura.
465.
466.             if(lista.size()<100) // Si la lista tiene menos de 10
   valores.
467.             {
468.                 Insertar(lista,valor); // Insertamos un valor segun el
   criterio FIFO.
469.             }
470.             else // Si la lista tiene 10 o mas valores.
471.             {
472.                 /*Introducimos y sacamos un  valor segun el criterio
   FIFO.*/
473.                 Insertar(lista, valor);
474.                 Borrar(lista);
475.             }
476.
477.             Imprimir(lista);
478.
479.             cout<<"\t\t Setpoint: " << setpointTempC << endl;
480.             cout<<"\t\t Permiso Setpoint: " << mqttHdl->permit_setpoint
   << endl;
481.             cout<<"\t\t Intervalo: " << intervalo << endl;
482.             cout<<"\t\t Intervalo old: " << intervalo_old << endl;
483.             cout<<"\t\t Setpoint almacenado en el array diario: " <<
   mqttHdl->Intervalos_calendario_DiarioTemp[intervalo] << endl;
484.             cout<<"\t\t Setpoint almacenado en el array festivo: " <<
   mqttHdl->Intervalos_calendario_FestivoTemp[intervalo] << endl;
485.             cout<<"\t\t Setpoint old: " << setpointTempC_old << endl;
486.             cout<<"\t\t Dia: " << dia << endl;
487.             cout<<"\t\t Hora: " << hora << endl;
488.             cout<<"\t\t Minutos: " << mins << endl;
489.             cout<<"\t\t Modo trabajo: " << mqttHdl->modoTrabajo << endl;
```

```
490.             cout<<"\t\t Modo dia: " << mqttHdl->ModosDias_calendario[dia]
   <<endl;
491.
492.
493.             if(lista.size()>=5) // Si la lista contiene algun valor.
494.             {
495.                 MaxMin(lista, mqttHdl);
496.
497.                 tempC_media=media(lista);
498.
499.                 str_tempC_media = to_string((int)tempC_media) + " °C";
500.                 mqttHdl->publish(NULL, mqttHdl-
   >topic_TempMedia,str_tempC_media.size(),str_tempC_media.c_str(),0,true);
501.             }
502.             else // Si la lista esta vacia.
503.             {
504.                 cout<<"\n\t\t                 ----------------------------
   --"<<endl;
505.                 cout<<"\t\t                | NO HAY VALORES QUE COMPARAR
   |"<<endl;
506.                 cout<<"\t\t                 ----------------------------
   \n"<<endl;
507.                 str_tempC_media = "-- °C";
508.                 mqttHdl->publish(NULL, mqttHdl-
   >topic_TempMedia,str_tempC_media.size(),str_tempC_media.c_str(),0,true);
509.                 str_tempMax = "--  °C";
510.                 mqttHdl->publish(NULL, mqttHdl-
   >topic_TempMax,str_tempMax.size(),str_tempMax.c_str(),0,true);
511.                 str_tempMin = "-- °C";
512.                 mqttHdl->publish(NULL, mqttHdl-
   >topic_TempMin,str_tempMin.size(),str_tempMin.c_str(),0,true);
513.             }
514.
515.             mqttHdl->permit_tempC = 0;
516.         }
517.
518.         if(lista.size()>=5)
519.         {
520.             if(mqttHdl->estacionAno<1){
521.
522.                 if(mqttHdl->modoTrabajo<1){
523.
524.                     if(mqttHdl->ModosDias_calendario[dia]<1){//Diario
525.                         intervalo = intervalo_actual(mqttHdl-
   >Intervalos_calendario_DiarioHora, mqttHdl-
   >Intervalos_calendario_DiarioMins);
526.                         if(mqttHdl->permit_setpoint==1){
527.                             setpointTempC = mqttHdl->setpoint_tempC_MAN;
528.
529.                             if(setpointTempC > mqttHdl-
   >Intervalos_calendario_DiarioTemp[intervalo]){
530.                                 modoSemiAUT = 1;
531.                             }else if(setpointTempC < mqttHdl-
   >Intervalos_calendario_DiarioTemp[intervalo]){
532.                                 modoSemiAUT = 2;
533.                             }
534.
535.                             if(setpointTempC==mqttHdl-
   >Intervalos_calendario_DiarioTemp[intervalo] || intervalo_old!=intervalo){
536.                                 modoSemiAUT = 0;
537.                                 mqttHdl->permit_setpoint = 0;
```

```
538.                                 actualizar_SP = 1;
539.                             }
540.
541.                             if(modoSemiAUT!=modoSemiAUT_old){
542.                                 mqttHdl->publish(NULL, mqttHdl-
    >topic_modoSemiAUT,to_string(modoSemiAUT).size(),to_string(modoSemiAUT).c_s
    tr());
543.                                 modoSemiAUT_old=modoSemiAUT;
544.                             }
545.                         }else{
546.                             setpointTempC = mqttHdl-
    >Intervalos_calendario_DiarioTemp[intervalo];
547.                             if(actualizar_SP == 1 ||
    intervalo_old!=intervalo || setpointTempC_old!=setpointTempC){
548.                                 mqttHdl->publish(NULL, mqttHdl-
    >topic_setpoint,to_string(setpointTempC).size(),to_string(setpointTempC).c_
    str());
549.                                 actualizar_SP = 0;
550.                                 setpointTempC_old = setpointTempC;
551.
552.                                 if(intervalo!=intervalo_old){
553.                                     intervalo_str = "D: " +
    to_string(intervalo+1);
554.                                     mqttHdl->publish(NULL, mqttHdl-
    >topic_Intervalo,intervalo_str.size(),intervalo_str.c_str());
555.                                     intervalo_old = intervalo;
556.                                 }
557.                             }
558.                         }
559.
560.                     }else{//Festivo
561.                         intervalo = intervalo_actual(mqttHdl-
    >Intervalos_calendario_FestivoHora, mqttHdl-
    >Intervalos_calendario_FestivoMins);
562.                         if(mqttHdl->permit_setpoint==1){
563.                             setpointTempC = mqttHdl->setpoint_tempC_MAN;
564.
565.                             if(setpointTempC > mqttHdl-
    >Intervalos_calendario_FestivoTemp[intervalo]){
566.                                 modoSemiAUT = 1;
567.                             }else if(setpointTempC < mqttHdl-
    >Intervalos_calendario_FestivoTemp[intervalo]){
568.                                 modoSemiAUT = 2;
569.                             }
570.
571.                             if(setpointTempC==mqttHdl-
    >Intervalos_calendario_FestivoTemp[intervalo] || intervalo_old!=intervalo){
572.                                 modoSemiAUT = 0;
573.                                 mqttHdl->permit_setpoint = 0;
574.                                 actualizar_SP = 1;
575.                             }
576.
577.                             if(modoSemiAUT!=modoSemiAUT_old){
578.                                 mqttHdl->publish(NULL, mqttHdl-
    >topic_modoSemiAUT,to_string(modoSemiAUT).size(),to_string(modoSemiAUT).c_s
    tr());
579.                                 modoSemiAUT_old=modoSemiAUT;
580.                             }
581.
582.                         }else{
```

```
583.                                 setpointTempC = mqttHdl-
   >Intervalos_calendario_FestivoTemp[intervalo];
584.                                 if(actualizar_SP == 1 ||
   intervalo_old!=intervalo || setpointTempC_old!=setpointTempC){
585.                                     mqttHdl->publish(NULL, mqttHdl-
   >topic_setpoint,to_string(setpointTempC).size(),to_string(setpointTempC).c_
   str());
586.                                     actualizar_SP = 0;
587.                                     setpointTempC_old = setpointTempC;
588.
589.                                     if(intervalo!=intervalo_old){
590.                                         intervalo_str = "F: " +
   to_string(intervalo+1);
591.                                         mqttHdl->publish(NULL, mqttHdl-
   >topic_Intervalo,intervalo_str.size(),intervalo_str.c_str());
592.                                         intervalo_old = intervalo;
593.                                     }
594.                                 }
595.                             }
596.                         }
597.
598.                     }else{
599.                         if(mqttHdl->permit_setpoint==1){
600.                             setpointTempC = mqttHdl->setpoint_tempC_MAN;
601.                         }
602.                     }
603.                 }else{
604.                     setpointTempC = -1;
605.                     setpointTempC_old = setpointTempC;
606.                     if(mqttHdl->permit_estacionAno == 1){
607.                         mqttHdl->publish(NULL, mqttHdl-
   >topic_setpoint,1,"0");
608.                         mqttHdl->permit_estacionAno = 0;
609.                     }
610.                 }
611.
612.                 actuador(mqttHdl->tempC,setpointTempC, mqttHdl);
613.             }
614.
615.         }
616.         return 0;
617. }
```

```
1.  /*********************************************************************
   **********/
2. /*
   */
3. /*                                  TRABAJO FIN DE GRADO
   */
4. /*                                    Control principal
   */
5. /*
   */
6. /*********************************************************************
   *********/
7.
8. #include <iostream>
9. #include "mosquittomqtt.h"
10.
11.  using namespace std;
```

```
12.
13.  float tempC_media; //Variable temperatura media
14.  int tempMin;
15.  int tempMax;
16.  string str_tempC_media;
17.  string str_tempMin;
18.  string str_tempMax;
19.
20.  //Variables de la fecha
21.  int dia=0;
22.  int hora=6;
23.  int mins=0;
24.  int mins_old=0;
25.  int seg=0;
26.  int seg_old=0;
27.
28.  int intervalo = 0;//Intervalo del dia
29.  int intervalo_old = 7; // Intervalo antiguo del dia
30.
31.  //Variables del totalizador de horas de funcionamiento
32.  int estadisticas_horas_acumuladas = 0;
33.  int estadisticas_mins_acumulados = 0;
34.  bool estadisticas_permisivo_contador = 0;
35.  string horometro;
36.
37.  //Variables del setpoint
38.  int setpointTempC=15;
39.  int setpointTempC_old=0;
40.
41.  //Variables de modo semi automatico
42.  int modoSemiAUT = 0;
43.  int modoSemiAUT_old = 0;
44.
45.  //Variable axuliar para actualizar el sp
46.  int actualizar_SP = 1;
47.
48.  bool on = 0, off = 1; // para que solo accione la calefaccion una vez
49.
50.  string Dia_Hora;
51.  string intervalo_str;
52.
53.
54.  /*DECLARACIÓN DE LA ESTRUCTURA "TEMPERATURA"*/
55.  struct Nodo
56.  {
57.      int lectura;
58.      string timestamp;
59.      /*Sobrecargamos el operador '<' para que a la hora de comparar sepa
     que compara*/
60.      bool operator <(Nodo & n)
61.      {
62.          return lectura <  n.lectura;
63.      }
64.  };
65.
66.  /*************************************************************************
     ************/
67.  /* Nombre: void Insertar(list<T> &lista,  Nodo &temp)
     */
68.  /* Descripción: Inserta un valor en la lista enlazada segun el criterio
     FIFO        */
```

```
69.  /* Argumentos: Lista y temperatura nueva a insertar.
   */
70.  /* Valor Devuelto: Ninguno
   */
71.  /**********************************************************************
   ************/
72.  template<typename T>
73.  void Insertar(list<T> &lista,  Nodo &temp)
74.  {
75.      if(lista.empty()) // Si lista esta vacia.
76.      {
77.           lista.emplace( lista.begin(), temp ); // Insertamos el primer
   valor.
78.      }
79.      else
80.      {
81.          lista.push_back(temp); // Insertamos un valor segun el criterio
   FIFO.
82.      }
83.  } // Fin de la funcion Insertar().
84.
85.  /**********************************************************************
   ************/
86.  /* Nombre: void Borrar(list<T> &lista)
   */
87.  /* Descripción: Borrar un valor de la lista enlazada segun el criterio
   FIFO.        */
88.  /* Argumentos: Lista
   */
89.  /* Valor Devuelto: Ninguno
   */
90.  /**********************************************************************
   ************/
91.  template<typename T>
92.  void Borrar(list<T> &lista)
93.  {
94.      if(lista.size()>=1) // Si el tamaño de la lista es mayor o igual a 1.
95.      {
96.          lista.pop_front(); // Borramos un valor segun el criterio FIFO.
97.      }
98.  } //Fin de la funcion Borrar()
99.
100. /**********************************************************************
   ************/
101. /* Nombre: void Imprimir(list<T> &lista)
   */
102. /* Descripción: Imprime por pantalla los valores contenidos en la lista
   enlazada    */
103. /* Argumentos: Lista.
   */
104. /* Valor Devuelto: Ninguno
   */
105. /**********************************************************************
   ************/
106. template<typename T>
107. void Imprimir(list<T> &lista)
108. {
109.     typename list<T> ::iterator it; // Iterador que recorrerá la lista.
110.     int n=1; // Declaramos un contador "n" que nos indica la posicion en
   la que se encuentran
111.             //los valores en la cola.
```

```
112.
113.     cout<<"\n\t              ------------------------------------------
   "<<endl;
114.     cout<<"\t            | LECTURA DE LOS DATOS CONTENIDOS EN LA COLA
   |"<<endl;
115.     cout<<"\t             ------------------------------------------
   \n"<<endl;
116.
117.     if (lista.size()>0) // Si el tamaño de la lista es mayor que cero.
118.     {
119.         cout<<"\t\t ---------------------------------------------------
   "<<endl;
120.         cout<<"\t\t|  Núm  |Temperatura(°C)|          Tiempo
   |"<<endl;
121.
122.         for(it=lista.begin(); it!=lista.end(); it++) // Con el iterador
   recorremos la lista y vamos imprimiento
123.         {                                            // los valores que
   va adquiriendo 'it'.
124.
125.             cout<<"\t\t|-------|---------------|-------------------------
   ---|"<<endl;
126.             cout<<"\t\t|  -"<<n<<"-\t|      " << it->lectura;
127.             cout<< " \t| "<<it->timestamp ;//<< "|"<< endl;
128.
129.             n++; // Incremeta en una unidad, a medida que se imprimen los
   datos.
130.         }
131.
132.     cout<<"\t\t ---------------------------------------------------
   "<<endl;
133.     }
134.     else // Si la lista no contiene ningun valor.
135.     {
136.         cout<<"\n\t\t\t       ¡LISTA VACÍA!"<<endl;
137.     }
138. } // Fin de la funcion Imprimir().
139.
140. /*************************************************************************
   ************/
141. /* Nombre: void MaxMin(list<T> &lista)
   */
142. /* Descripción: Compara los valores de la lista enlazada y nos dice cual
   es máximo  */
143. /*              y mínimo.
   */
144. /* Argumentos: Lista.
   */
145. /*Valor Devuelto: Ninguno
   */
146. /*************************************************************************
   ************/
147. template<typename T>
148. void MaxMin(list<T> &lista,mosquittoMQTT* mqttHdl)
149. {
150.
151.     typename list<T> :: iterator it; // Iterador que recorrera la lista.
152.     if(lista.size()==1) // Si el tamaño de la lista en igual a 1.
153.     {
154.         /*Se sabe que el maximo y el minimo son el mismo valor puesto que
   el
```

```
155.         tamaño de la lista es 1*/
156.
157.         /*Con la llamada al algoritmo 'min_element' obtenemos el minimo
   valor
158.         contenido en la lista*/
159.         it = min_element(lista.begin(),lista.end());
160.         cout<<"\n\t\t       ------------------------------------------
   "<<endl;
161.         cout<<"\t\t      | LA TEMPERATURA MÁXIMA Y MÍNIMA SON IGUALES
   |"<<endl;
162.         cout<<"\t\t       ------------------------------------------
   \n"<<endl;
163.
164.         cout<<"\t\t    ------------------------------------------
   "<<endl;
165.         cout<<"\t\t   |Temperatura(°C)|           Tiempo
   |"<<endl;
166.         cout<<"\t\t   |---------------|--------------------------
   |"<<endl;
167.         cout<<"\t\t   |     " << it->lectura << "\t   | ";
168.         cout<<it->timestamp ;//<< "|"<< endl;
169.
170.         cout<<"\t\t    ------------------------------------------
   "<<endl;
171.         str_tempMax = to_string(it->lectura) + " °C";
172.         str_tempMin = to_string(it->lectura) + " °C";
173.
174.     }
175.     else // Si la lista tiene un tamaño distinto de uno
176.     {
177.         /*Con la llamada al algoritmo 'max_element' obtenemos el mayor
   valor
178.         contenido en la lista*/
179.         it= max_element(lista.begin(),lista.end());
180.
181.         cout<<"\n\t                  --------------------"<<endl;
182.         cout<<"\t                 | TEMPERATURA MÁXIMA |"<<endl;
183.         cout<<"\t                  --------------------\n"<<endl;
184.
185.         cout<<"\t\t    ------------------------------------------
   "<<endl;
186.         cout<<"\t\t   |Temperatura(°C)|           Tiempo
   |"<<endl;
187.         cout<<"\t\t   |---------------|--------------------------
   |"<<endl;
188.         cout<<"\t\t   |     " << it->lectura << "\t   | ";
189.         cout<<it->timestamp ;//<< "|"<< endl;
190.
191.         cout<<"\t\t    ------------------------------------------
   "<<endl;
192.         str_tempMax = to_string(it->lectura) + " °C";
193.
194.         /*Con la llamada al algoritmo 'min_element' obtenemos el minimo
   valor
195.         contenido en la lista*/
196.         it = min_element(lista.begin(),lista.end());
197.
198.         cout<<"\n\t\t                  --------------------"<<endl;
199.         cout<<"\t\t                 | TEMPERATURA MÍNIMA |"<<endl;
200.         cout<<"\t\t                  --------------------\n"<<endl;
201.
```

```
202.         cout<<"\t\t    ------------------------------------------
   "<<endl;
203.         cout<<"\t\t   |Temperatura(°C)|           Tiempo
   |"<<endl;
204.         cout<<"\t\t   |---------------|--------------------------
   |"<<endl;
205.         cout<<"\t\t   |     " << it->lectura << "\t   | ";
206.         cout<<it->timestamp ;//<< "|"<< endl;
207.
208.         cout<<"\t\t    ------------------------------------------
   "<<endl;
209.
210.         str_tempMin = to_string(it->lectura) + " °C";
211.
212.     }
213.     mqttHdl->publish(NULL, mqttHdl-
   >topic_TempMax,str_tempMax.size(),str_tempMax.c_str(),0,true);
214.     mqttHdl->publish(NULL, mqttHdl-
   >topic_TempMin,str_tempMin.size(),str_tempMin.c_str(),0,true);
215. } // Fin de la funcion MaxMin().
216.
217. /*************************************************************************
   ************/
218. /* Nombre: void Tiempo()
   */
219. /* Descripción: Calcula la diferencia en segundos entre el año 2000 y la
   actualidad */
220. /* Argumentos: Ninguno.
   */
221. /* Valor Devuelto: Segundos.
   */
222. /*************************************************************************
   ************/
223. string Tiempo()
224. {
225.     int segundos; // Variable que almacena el tiempo.
226.     string fecha_hora;
227.     string dia_str;
228.     time_t timer; // Variable de la libreria ctime que nos indica el
   tiempo transcurrido desde 1970.
229.     segundos=time(&timer);
230.     fecha_hora = ctime (&timer);//Convertimos el tiempo en una string
231.     hora = atoi(fecha_hora.substr(11,2).c_str());
232.     mins = atoi(fecha_hora.substr(14,2).c_str());
233.     seg = segundos;
234.     dia_str = fecha_hora.substr(0,3);
235.     if(dia_str.compare("Mon")==0){
236.         dia = 0;
237.         dia_str = "Lun";
238.     }else if(dia_str.compare("Tue")==0){
239.         dia = 1;
240.         dia_str = "Mar";
241.     }else if(dia_str.compare("Wed")==0){
242.         dia = 2;
243.         dia_str = "Mie";
244.     }else if(dia_str.compare("Thu")==0){
245.         dia = 3;
246.         dia_str = "Jue";
247.     }else if(dia_str.compare("Fri")==0){
248.         dia = 4;
249.         dia_str = "Vie";
```

```
250.     }else if(dia_str.compare("Sat")==0){
251.         dia = 5;
252.         dia_str = "Sab";
253.     }else {
254.         dia = 6;
255.         dia_str = "Dom";
256.     }
257.
258.     Dia_Hora = dia_str + " " + to_string(hora);
259.     if(mins<10){
260.         Dia_Hora = Dia_Hora + ":0" + to_string(mins);
261.     }else{
262.         Dia_Hora = Dia_Hora + ":" + to_string(mins);
263.     }
264.
265.     return fecha_hora; // Devolvemos el valor de la variable segundos.
266. } // Fin de la funcion Tiempo().
267.
268. /*************************************************************************
   ************/
269. /* Nombre: float media()
   */
270. /* Descripción: Calcula el valor medio de los datos contenidos en la
   lista          */
271. /* Argumentos: Lista.
   */
272. /* Valor Devuelto: Media
   */
273. /*************************************************************************
   ************/
274. template<typename T>
275. float media(list<T> &lista){
276.
277.     typename list<T> :: iterator it; // Iterador que recorrera la lista.
278.     float sumatorio = 0;
279.     float media;
280.     int n = 0; // Contador tamaño de la lista
281.
282.     if(lista.size()<10)
283.     {
284.         cout<<"\n\t\t                    --------------------"<<endl;
285.         cout<<"\t\t                  | TEMPERATURA  MEDIA |"<<endl;
286.         cout<<"\t\t                   --------------------\n"<<endl;
287.
288.         cout<<"\n\t\t Valores insuficientes. \n\t\tMínimo 10 valores para
   calcular la media\n"<<endl;
289.
290.     }
291.     else
292.     {
293.         for(it=lista.begin(); it!=lista.end(); it++)
294.         {
295.             sumatorio = it->lectura + sumatorio;
296.             n++;
297.         }
298.
299.         media = sumatorio/n;
300.
301.         cout<<"\n\t\t                    --------------------"<<endl;
302.         cout<<"\t\t                  | TEMPERATURA  MEDIA |"<<endl;
303.         cout<<"\t\t                   --------------------"<<endl;
```

```
304.
305.         cout<<"\t\t                    |       "<<media<<"°C      |"<<endl;
306.         cout<<"\t\t                     --------------------\n"<<endl;
307.
308.     }
309.     return media;
310. }
311.
312. /*************************************************************************
     ************/
313. /* Nombre: void horometro_caldera()
     */
314. /* Descripción: Calcula el tiempo que lleva funcionando la caldera
     */
315. /* Argumentos: llamada, actuador.
     */
316. /* Valor Devuelto: Ninguno
     */
317. /*************************************************************************
     ************/
318. void horometro_caldera(int llamada, bool actuador){
319.     if(llamada == 0){
320.         if(actuador == 0){
321.             estadisticas_permisivo_contador = 0;
322.         }else{
323.             estadisticas_permisivo_contador = 1;
324.         }
325.     }else{
326.         if(estadisticas_permisivo_contador == 1){
327.             estadisticas_mins_acumulados ++;
328.             if(estadisticas_mins_acumulados > 59){
329.                 estadisticas_horas_acumuladas ++;
330.                 estadisticas_mins_acumulados = 0;
331.             }
332.         }
333.     }
334. }//fin horometro_caldera()
335.
336. /*************************************************************************
     ************/
337. /* Nombre: actuador()
     */
338. /* Descripción: enciende o apaga la salida del actuador en funcion de la
     temp       */
339. /* Argumentos: temp, setpoint, cliente mqtt
     */
340. /* Valor Devuelto: nada
     */
341. /*************************************************************************
     ************/
342. void actuador(int temp, int setpoint, mosquittoMQTT* mqttHdl){
343.     //if(compara_mayor(temp,setpoint,1000) == 2 && on == 0)
344.     if(temp<=setpoint && on == 0)
345.     {
346.         cout<<"\n\t\t **Se ENCIENDE la calefacción\n"<<endl;
347.         mqttHdl->publish(NULL, mqttHdl->topic_ConsignaOnOff,
     1,"1",0,true);
348.         on = 1;
349.         off = 0;
350.         horometro_caldera(0,1);
351.     }
```

```
352.     //if(compara_mayor(temp,setpoint,1000) == 1 && off == 0)
353.     if(temp > setpoint && off==0)
354.     {
355.         cout<<"\n\t\t **Se APAGA la calefacción\n"<<endl;
356.         mqttHdl->publish(NULL, mqttHdl-
   >topic_ConsignaOnOff,1,"0",0,true);
357.         on = 0;
358.         off = 1;
359.         horometro_caldera(0,0);
360.     }
361. }
362.
363. int intervalo_actual(int intervalos_hora[], int intervalos_mins[]){
364.     int posicion;
365.
366.     for(int i=0; i<6; i++){
367.         if(intervalos_hora[i]==hora){
368.             if(intervalos_mins[i]<=mins){
369.                 posicion = i;
370.                 break;
371.             }else if(i==0 && intervalos_mins[i]>mins){
372.                 posicion = 5;
373.                 break;
374.             }else{
375.                 posicion = i-1;
376.                 break;
377.             }
378.         }else if(intervalos_hora[i]<hora){
379.             if(i!=5 && intervalos_hora[i+1]>hora){
380.                 posicion = i;
381.                 break;
382.             }else if(i==5){
383.                 posicion = 5;
384.                 break;
385.             }
386.         }else if(i==0 && intervalos_hora[i]>hora){
387.             posicion = 5;
388.             break;
389.         }
390.     }
391.
392.     return posicion;
393. }
394.
395. /*PROGRAMA PRINCIPAL, MAIN*/
396. int main()
397. {
398.     /*DECLARACIÓN DE VARIABLES*/
399.     list<Nodo> lista; // Lista de temperaturas.
400.     Nodo valor; // Variable de tipo nodo para almacenar los valores
   introducidos por el usuario.
401.     Nodo temp_max; //Cariable de tipo nodo para temperatura maxima
402.     Nodo temp_min; //Variable de tipo nodo para temperatura minima
403.     tempC_media=0.00;
404.
405.     mosquittoMQTT * mqttHdl;
406.     mqttHdl = new mosquittoMQTT(NULL, "localhost", 1883);
407.
408.     setlocale(LC_CTYPE, "Spanish"); // Reconoce los caracteres de la
   ortografia del castellano.
409.
```

```
410.     /*Imprime los valores de tipo float con una precision decimal de 2.*/
411.     cout.setf(ios::fixed, ios::floatfield);
412.     cout.precision(2);
413.
414.     /*FIN DECLARACIÓN DE VARIABLES*/
415.
416.     while(1)
417.     {
418.         //Mantiene la conexion
419.         int res = mqttHdl->loop();
420.         if (res){
421.             mqttHdl->reconnect();
422.         }
423.         Tiempo();
424.
425.         if(mins!=mins_old){
426.             mqttHdl->publish(NULL, mqttHdl-
   >topic_DiaHora,Dia_Hora.size(),Dia_Hora.c_str(),0,true);
427.             horometro_caldera(1,0);
428.             if(estadisticas_horas_acumuladas < 10){
429.                 horometro = "0" +
   to_string(estadisticas_horas_acumuladas) + " horas y ";
430.             }else{
431.                 horometro = to_string(estadisticas_horas_acumuladas) + "
   horas y ";
432.             }
433.             if(estadisticas_mins_acumulados < 10){
434.                 horometro = horometro + "0" +
   to_string(estadisticas_mins_acumulados) + " minutos";
435.             }else{
436.                 horometro = horometro +
   to_string(estadisticas_mins_acumulados) + " minutos";
437.             }
438.             mqttHdl->publish(NULL, mqttHdl-
   >topic_Horometro,horometro.size(),horometro.c_str(),0,true);
439.
440.             if(mqttHdl->tempC<=setpointTempC)
441.             {
442.                 mqttHdl->publish(NULL, mqttHdl->topic_ConsignaOnOff,
   1,"1",0,true);
443.
444.             }
445.             if(mqttHdl->tempC > setpointTempC)
446.             {
447.                 mqttHdl->publish(NULL, mqttHdl-
   >topic_ConsignaOnOff,1,"0",0,true);
448.             }
449.
450.             mins_old = mins;
451.         }
452.
453.         //Calcula la temperatura media cuando recibe un dato nuevo
454.         if(mqttHdl->permit_tempC == 1)
455.         {
456.             //temperatura_media(mqttHdl->tempC);
457.             cout<<"\n\t             ------------------------------------
   "<<endl;
458.             cout<<"\t            | INTRODUCIR NUEVO VALOR EN LA COLA
   |"<<endl;
459.             cout<<"\t             ------------------------------------
   \n"<<endl;
```

```
460.             cout<<"\t"<<endl;
461.
462.             valor.lectura = mqttHdl->tempC;
463.             valor.timestamp=Tiempo(); // El tiempo se le asigna
   automaticamente en el instante que ha sido
464.                                       // introducido la lectura.
465.
466.             if(lista.size()<100) // Si la lista tiene menos de 10
   valores.
467.             {
468.                 Insertar(lista,valor); // Insertamos un valor segun el
   criterio FIFO.
469.             }
470.             else // Si la lista tiene 10 o mas valores.
471.             {
472.                 /*Introducimos y sacamos un  valor segun el criterio
   FIFO.*/
473.                 Insertar(lista, valor);
474.                 Borrar(lista);
475.             }
476.
477.             Imprimir(lista);
478.
479.             cout<<"\t\t Setpoint: " << setpointTempC << endl;
480.             cout<<"\t\t Permiso Setpoint: " << mqttHdl->permit_setpoint
   << endl;
481.             cout<<"\t\t Intervalo: " << intervalo << endl;
482.             cout<<"\t\t Intervalo old: " << intervalo_old << endl;
483.             cout<<"\t\t Setpoint almacenado en el array diario: " <<
   mqttHdl->Intervalos_calendario_DiarioTemp[intervalo] << endl;
484.             cout<<"\t\t Setpoint almacenado en el array festivo: " <<
   mqttHdl->Intervalos_calendario_FestivoTemp[intervalo] << endl;
485.             cout<<"\t\t Setpoint old: " << setpointTempC_old << endl;
486.             cout<<"\t\t Dia: " << dia << endl;
487.             cout<<"\t\t Hora: " << hora << endl;
488.             cout<<"\t\t Minutos: " << mins << endl;
489.             cout<<"\t\t Modo trabajo: " << mqttHdl->modoTrabajo << endl;
490.             cout<<"\t\t Modo dia: " << mqttHdl->ModosDias_calendario[dia]
   <<endl;
491.
492.
493.             if(lista.size()>=5) // Si la lista contiene algun valor.
494.             {
495.                 MaxMin(lista, mqttHdl);
496.
497.                 tempC_media=media(lista);
498.
499.                 str_tempC_media = to_string((int)tempC_media) + " °C";
500.                 mqttHdl->publish(NULL, mqttHdl-
   >topic_TempMedia,str_tempC_media.size(),str_tempC_media.c_str(),0,true);
501.             }
502.             else // Si la lista esta vacia.
503.             {
504.                 cout<<"\n\t\t                  ----------------------------
   --"<<endl;
505.                 cout<<"\t\t                | NO HAY VALORES QUE COMPARAR
   |"<<endl;
506.                 cout<<"\t\t                 ----------------------------
   \n"<<endl;
507.                 str_tempC_media = "-- °C";
```

```
508.                 mqttHdl->publish(NULL, mqttHdl-
   >topic_TempMedia,str_tempC_media.size(),str_tempC_media.c_str(),0,true);
509.                 str_tempMax = "--  °C";
510.                 mqttHdl->publish(NULL, mqttHdl-
   >topic_TempMax,str_tempMax.size(),str_tempMax.c_str(),0,true);
511.                 str_tempMin = "-- °C";
512.                 mqttHdl->publish(NULL, mqttHdl-
   >topic_TempMin,str_tempMin.size(),str_tempMin.c_str(),0,true);
513.             }
514.
515.             mqttHdl->permit_tempC = 0;
516.         }
517.
518.         if(lista.size()>=5)
519.         {
520.             if(mqttHdl->estacionAno<1){
521.
522.                 if(mqttHdl->modoTrabajo<1){
523.
524.                     if(mqttHdl->ModosDias_calendario[dia]<1){//Diario
525.                         intervalo = intervalo_actual(mqttHdl-
   >Intervalos_calendario_DiarioHora, mqttHdl-
   >Intervalos_calendario_DiarioMins);
526.                         if(mqttHdl->permit_setpoint==1){
527.                             setpointTempC = mqttHdl->setpoint_tempC_MAN;
528.
529.                             if(setpointTempC > mqttHdl-
   >Intervalos_calendario_DiarioTemp[intervalo]){
530.                                 modoSemiAUT = 1;
531.                             }else if(setpointTempC < mqttHdl-
   >Intervalos_calendario_DiarioTemp[intervalo]){
532.                                 modoSemiAUT = 2;
533.                             }
534.
535.                             if(setpointTempC==mqttHdl-
   >Intervalos_calendario_DiarioTemp[intervalo] || intervalo_old!=intervalo){
536.                                 modoSemiAUT = 0;
537.                                 mqttHdl->permit_setpoint = 0;
538.                                 actualizar_SP = 1;
539.                             }
540.
541.                             if(modoSemiAUT!=modoSemiAUT_old){
542.                                 mqttHdl->publish(NULL, mqttHdl-
   >topic_modoSemiAUT,to_string(modoSemiAUT).size(),to_string(modoSemiAUT).c_s
   tr());
543.                                 modoSemiAUT_old=modoSemiAUT;
544.                             }
545.                         }else{
546.                             setpointTempC = mqttHdl-
   >Intervalos_calendario_DiarioTemp[intervalo];
547.                             if(actualizar_SP == 1 ||
   intervalo_old!=intervalo || setpointTempC_old!=setpointTempC){
548.                                 mqttHdl->publish(NULL, mqttHdl-
   >topic_setpoint,to_string(setpointTempC).size(),to_string(setpointTempC).c_
   str());
549.                                 actualizar_SP = 0;
550.                                 setpointTempC_old = setpointTempC;
551.
552.                                 if(intervalo!=intervalo_old){
553.                                     intervalo_str = "D: " +
   to_string(intervalo+1);
```

```
554.                                     mqttHdl->publish(NULL, mqttHdl-
   >topic_Intervalo,intervalo_str.size(),intervalo_str.c_str());
555.                                     intervalo_old = intervalo;
556.                                 }
557.                             }
558.                         }
559.
560.                     }else{//Festivo
561.                         intervalo = intervalo_actual(mqttHdl-
   >Intervalos_calendario_FestivoHora, mqttHdl-
   >Intervalos_calendario_FestivoMins);
562.                         if(mqttHdl->permit_setpoint==1){
563.                             setpointTempC = mqttHdl->setpoint_tempC_MAN;
564.
565.                             if(setpointTempC > mqttHdl-
   >Intervalos_calendario_FestivoTemp[intervalo]){
566.                                 modoSemiAUT = 1;
567.                             }else if(setpointTempC < mqttHdl-
   >Intervalos_calendario_FestivoTemp[intervalo]){
568.                                 modoSemiAUT = 2;
569.                             }
570.
571.                             if(setpointTempC==mqttHdl-
   >Intervalos_calendario_FestivoTemp[intervalo] || intervalo_old!=intervalo){
572.                                 modoSemiAUT = 0;
573.                                 mqttHdl->permit_setpoint = 0;
574.                                 actualizar_SP = 1;
575.                             }
576.
577.                             if(modoSemiAUT!=modoSemiAUT_old){
578.                                 mqttHdl->publish(NULL, mqttHdl-
   >topic_modoSemiAUT,to_string(modoSemiAUT).size(),to_string(modoSemiAUT).c_s
   tr());
579.                                 modoSemiAUT_old=modoSemiAUT;
580.                             }
581.
582.                         }else{
583.                             setpointTempC = mqttHdl-
   >Intervalos_calendario_FestivoTemp[intervalo];
584.                             if(actualizar_SP == 1 ||
   intervalo_old!=intervalo || setpointTempC_old!=setpointTempC){
585.                                 mqttHdl->publish(NULL, mqttHdl-
   >topic_setpoint,to_string(setpointTempC).size(),to_string(setpointTempC).c_
   str());
586.                                 actualizar_SP = 0;
587.                                 setpointTempC_old = setpointTempC;
588.
589.                                 if(intervalo!=intervalo_old){
590.                                     intervalo_str = "F: " +
   to_string(intervalo+1);
591.                                     mqttHdl->publish(NULL, mqttHdl-
   >topic_Intervalo,intervalo_str.size(),intervalo_str.c_str());
592.                                     intervalo_old = intervalo;
593.                                 }
594.                             }
595.                         }
596.                     }
597.
598.                 }else{
599.                     if(mqttHdl->permit_setpoint==1){
600.                         setpointTempC = mqttHdl->setpoint_tempC_MAN;
```

```
                    }
                }
            }else{
                setpointTempC = -1;
                setpointTempC_old = setpointTempC;
                if(mqttHdl->permit_estacionAno == 1){
                    mqttHdl->publish(NULL, mqttHdl-
>topic_setpoint,1,"0");
                    mqttHdl->permit_estacionAno = 0;
                }
            }

            actuador(mqttHdl->tempC,setpointTempC, mqttHdl);
        }

    }
    return 0;
}
```

```
/*****************************************************************************
**********/
/*
*/
/*                                  TRABAJO FIN DE GRADO
*/
/*                                    Control principal
*/
/*
*/
/*****************************************************************************
*********/

#include <iostream>
#include "mosquittomqtt.h"

using namespace std;

float tempC_media; //Variable temperatura media
int tempMin;
int tempMax;
string str_tempC_media;
string str_tempMin;
string str_tempMax;

//Variables de la fecha
int dia=0;
int hora=6;
int mins=0;
int mins_old=0;
int seg=0;
int seg_old=0;

int intervalo = 0;//Intervalo del dia
int intervalo_old = 7; // Intervalo antiguo del dia

//Variables del totalizador de horas de funcionamiento
int estadisticas_horas_acumuladas = 0;
int estadisticas_mins_acumulados = 0;
bool estadisticas_permisivo_contador = 0;
string horometro;
```

```
36.
37.  //Variables del setpoint
38.  int setpointTempC=15;
39.  int setpointTempC_old=0;
40.
41.  //Variables de modo semi automatico
42.  int modoSemiAUT = 0;
43.  int modoSemiAUT_old = 0;
44.
45.  //Variable axuliar para actualizar el sp
46.  int actualizar_SP = 1;
47.
48.  bool on = 0, off = 1; // para que solo accione la calefaccion una vez
49.
50.  string Dia_Hora;
51.  string intervalo_str;
52.
53.
54.  /*DECLARACIÓN DE LA ESTRUCTURA "TEMPERATURA"*/
55.  struct Nodo
56.  {
57.      int lectura;
58.      string timestamp;
59.      /*Sobrecargamos el operador '<' para que a la hora de comparar sepa
     que compara*/
60.      bool operator <(Nodo & n)
61.      {
62.          return lectura <  n.lectura;
63.      }
64.  };
65.
66.  /**************************************************************************
     ************/
67.  /* Nombre: void Insertar(list<T> &lista,  Nodo &temp)
     */
68.  /* Descripción: Inserta un valor en la lista enlazada segun el criterio
     FIFO        */
69.  /* Argumentos: Lista y temperatura nueva a insertar.
     */
70.  /* Valor Devuelto: Ninguno
     */
71.  /**************************************************************************
     ************/
72.  template<typename T>
73.  void Insertar(list<T> &lista,  Nodo &temp)
74.  {
75.      if(lista.empty()) // Si lista esta vacia.
76.      {
77.           lista.emplace( lista.begin(), temp ); // Insertamos el primer
     valor.
78.      }
79.      else
80.      {
81.          lista.push_back(temp); // Insertamos un valor segun el criterio
     FIFO.
82.      }
83.  } // Fin de la funcion Insertar().
84.
85.  /**************************************************************************
     ************/
```

```
86.  /* Nombre: void Borrar(list<T> &lista)
   */
87.  /* Descripción: Borrar un valor de la lista enlazada segun el criterio
   FIFO.        */
88.  /* Argumentos: Lista
   */
89.  /* Valor Devuelto: Ninguno
   */
90.  /**************************************************************************
   ************/
91.  template<typename T>
92.  void Borrar(list<T> &lista)
93.  {
94.      if(lista.size()>=1) // Si el tamaño de la lista es mayor o igual a 1.
95.      {
96.          lista.pop_front(); // Borramos un valor segun el criterio FIFO.
97.      }
98.  } //Fin de la funcion Borrar()
99.
100. /**************************************************************************
   ************/
101. /* Nombre: void Imprimir(list<T> &lista)
   */
102. /* Descripción: Imprime por pantalla los valores contenidos en la lista
   enlazada    */
103. /* Argumentos: Lista.
   */
104. /* Valor Devuelto: Ninguno
   */
105. /**************************************************************************
   ************/
106. template<typename T>
107. void Imprimir(list<T> &lista)
108. {
109.     typename list<T> ::iterator it; // Iterador que recorrerá la lista.
110.     int n=1; // Declaramos un contador "n" que nos indica la posicion en
   la que se encuentran
111.              //los valores en la cola.
112.
113.     cout<<"\n\t             ------------------------------------------
   "<<endl;
114.     cout<<"\t            | LECTURA DE LOS DATOS CONTENIDOS EN LA COLA
   |"<<endl;
115.     cout<<"\t             ------------------------------------------
   \n"<<endl;
116.
117.     if (lista.size()>0) // Si el tamaño de la lista es mayor que cero.
118.     {
119.         cout<<"\t\t ---------------------------------------------------
   "<<endl;
120.         cout<<"\t\t|  Núm  |Temperatura(°C)|          Tiempo
   |"<<endl;
121.
122.         for(it=lista.begin(); it!=lista.end(); it++) // Con el iterador
   recorremos la lista y vamos imprimiento
123.         {                                            // los valores que
   va adquiriendo 'it'.
124.
125.             cout<<"\t\t|-------|---------------|-------------------------
   ---|"<<endl;
126.             cout<<"\t\t|  -"<<n<<"-\t|      " << it->lectura;
```

```
127.             cout<< " \t| "<<it->timestamp ;//<< "|"<< endl;
128.
129.             n++; // Incremeta en una unidad, a medida que se imprimen los
   datos.
130.         }
131.
132.     cout<<"\t\t ----------------------------------------------------
   "<<endl;
133.     }
134.     else // Si la lista no contiene ningun valor.
135.     {
136.         cout<<"\n\t\t\t       ¡LISTA VACÍA!"<<endl;
137.     }
138. } // Fin de la funcion Imprimir().
139.
140. /*********************************************************************
   ************/
141. /* Nombre: void MaxMin(list<T> &lista)
   */
142. /* Descripción: Compara los valores de la lista enlazada y nos dice cual
   es máximo  */
143. /*              y mínimo.
   */
144. /* Argumentos: Lista.
   */
145. /*Valor Devuelto: Ninguno
   */
146. /*********************************************************************
   ************/
147. template<typename T>
148. void MaxMin(list<T> &lista,mosquittoMQTT* mqttHdl)
149. {
150.
151.     typename list<T> :: iterator it; // Iterador que recorrera la lista.
152.     if(lista.size()==1) // Si el tamaño de la lista en igual a 1.
153.     {
154.         /*Se sabe que el maximo y el minimo son el mismo valor puesto que
   el
155.         tamaño de la lista es 1*/
156.
157.         /*Con la llamada al algoritmo 'min_element' obtenemos el minimo
   valor
158.         contenido en la lista*/
159.         it = min_element(lista.begin(),lista.end());
160.         cout<<"\n\t\t       ------------------------------------------
   "<<endl;
161.         cout<<"\t\t      | LA TEMPERATURA MÁXIMA Y MÍNIMA SON IGUALES
   |"<<endl;
162.         cout<<"\t\t       ------------------------------------------
   \n"<<endl;
163.
164.         cout<<"\t\t    ------------------------------------------
   "<<endl;
165.         cout<<"\t\t   |Temperatura(°C)|           Tiempo
   |"<<endl;
166.         cout<<"\t\t   |---------------|--------------------------
   |"<<endl;
167.         cout<<"\t\t   |     " << it->lectura << "\t   | ";
168.         cout<<it->timestamp ;//<< "|"<< endl;
169.
```

```
170.         cout<<"\t\t    ------------------------------------------
   "<<endl;
171.         str_tempMax = to_string(it->lectura) + " °C";
172.         str_tempMin = to_string(it->lectura) + " °C";
173.
174.     }
175.     else // Si la lista tiene un tamaño distinto de uno
176.     {
177.         /*Con la llamada al algoritmo 'max_element' obtenemos el mayor
   valor
178.         contenido en la lista*/
179.         it= max_element(lista.begin(),lista.end());
180.
181.         cout<<"\n\t                  --------------------"<<endl;
182.         cout<<"\t                 | TEMPERATURA MÁXIMA |"<<endl;
183.         cout<<"\t                  --------------------\n"<<endl;
184.
185.         cout<<"\t\t    ------------------------------------------
   "<<endl;
186.         cout<<"\t\t   |Temperatura(°C)|           Tiempo
   |"<<endl;
187.         cout<<"\t\t   |---------------|--------------------------
   |"<<endl;
188.         cout<<"\t\t   |     " << it->lectura << "\t   | ";
189.         cout<<it->timestamp ;//<< "|"<< endl;
190.
191.         cout<<"\t\t    ------------------------------------------
   "<<endl;
192.         str_tempMax = to_string(it->lectura) + " °C";
193.
194.         /*Con la llamada al algoritmo 'min_element' obtenemos el minimo
   valor
195.         contenido en la lista*/
196.         it = min_element(lista.begin(),lista.end());
197.
198.         cout<<"\n\t\t                  --------------------"<<endl;
199.         cout<<"\t\t                 | TEMPERATURA MÍNIMA |"<<endl;
200.         cout<<"\t\t                  --------------------\n"<<endl;
201.
202.         cout<<"\t\t    ------------------------------------------
   "<<endl;
203.         cout<<"\t\t   |Temperatura(°C)|           Tiempo
   |"<<endl;
204.         cout<<"\t\t   |---------------|--------------------------
   |"<<endl;
205.         cout<<"\t\t   |     " << it->lectura << "\t   | ";
206.         cout<<it->timestamp ;//<< "|"<< endl;
207.
208.         cout<<"\t\t    ------------------------------------------
   "<<endl;
209.
210.         str_tempMin = to_string(it->lectura) + " °C";
211.
212.     }
213.     mqttHdl->publish(NULL, mqttHdl-
   >topic_TempMax,str_tempMax.size(),str_tempMax.c_str(),0,true);
214.     mqttHdl->publish(NULL, mqttHdl-
   >topic_TempMin,str_tempMin.size(),str_tempMin.c_str(),0,true);
215. } // Fin de la funcion MaxMin().
216.
```

```
217. /**************************************************************************
   ************/
218. /* Nombre: void Tiempo()
   */
219. /* Descripción: Calcula la diferencia en segundos entre el año 2000 y la
   actualidad */
220. /* Argumentos: Ninguno.
   */
221. /* Valor Devuelto: Segundos.
   */
222. /**************************************************************************
   ************/
223. string Tiempo()
224. {
225.     int segundos; // Variable que almacena el tiempo.
226.     string fecha_hora;
227.     string dia_str;
228.     time_t timer; // Variable de la libreria ctime que nos indica el
   tiempo transcurrido desde 1970.
229.     segundos=time(&timer);
230.     fecha_hora = ctime (&timer);//Convertimos el tiempo en una string
231.     hora = atoi(fecha_hora.substr(11,2).c_str());
232.     mins = atoi(fecha_hora.substr(14,2).c_str());
233.     seg = segundos;
234.     dia_str = fecha_hora.substr(0,3);
235.     if(dia_str.compare("Mon")==0){
236.         dia = 0;
237.         dia_str = "Lun";
238.     }else if(dia_str.compare("Tue")==0){
239.         dia = 1;
240.         dia_str = "Mar";
241.     }else if(dia_str.compare("Wed")==0){
242.         dia = 2;
243.         dia_str = "Mie";
244.     }else if(dia_str.compare("Thu")==0){
245.         dia = 3;
246.         dia_str = "Jue";
247.     }else if(dia_str.compare("Fri")==0){
248.         dia = 4;
249.         dia_str = "Vie";
250.     }else if(dia_str.compare("Sat")==0){
251.         dia = 5;
252.         dia_str = "Sab";
253.     }else {
254.         dia = 6;
255.         dia_str = "Dom";
256.     }
257.
258.     Dia_Hora = dia_str + " " + to_string(hora);
259.     if(mins<10){
260.         Dia_Hora = Dia_Hora + ":0" + to_string(mins);
261.     }else{
262.         Dia_Hora = Dia_Hora + ":" + to_string(mins);
263.     }
264.
265.     return fecha_hora; // Devolvemos el valor de la variable segundos.
266. } // Fin de la funcion Tiempo().
267.
268. /**************************************************************************
   ************/
```

```
269. /* Nombre: float media()
   */
270. /* Descripción: Calcula el valor medio de los datos contenidos en la
   lista          */
271. /* Argumentos: Lista.
   */
272. /* Valor Devuelto: Media
   */
273. /**************************************************************************
   ************/
274. template<typename T>
275. float media(list<T> &lista){
276.
277.     typename list<T> :: iterator it; // Iterador que recorrera la lista.
278.     float sumatorio = 0;
279.     float media;
280.     int n = 0; // Contador tamaño de la lista
281.
282.     if(lista.size()<10)
283.     {
284.         cout<<"\n\t\t                    ---------------------"<<endl;
285.         cout<<"\t\t                   | TEMPERATURA  MEDIA |"<<endl;
286.         cout<<"\t\t                    ---------------------\n"<<endl;
287.
288.         cout<<"\n\t\t Valores insuficientes. \n\t\tMínimo 10 valores para
   calcular la media\n"<<endl;
289.
290.     }
291.     else
292.     {
293.         for(it=lista.begin(); it!=lista.end(); it++)
294.         {
295.             sumatorio = it->lectura + sumatorio;
296.             n++;
297.         }
298.
299.         media = sumatorio/n;
300.
301.         cout<<"\n\t\t                    ---------------------"<<endl;
302.         cout<<"\t\t                   | TEMPERATURA  MEDIA |"<<endl;
303.         cout<<"\t\t                    ---------------------"<<endl;
304.
305.         cout<<"\t\t                   |       "<<media<<"°C      |"<<endl;
306.         cout<<"\t\t                    ---------------------\n"<<endl;
307.
308.     }
309.     return media;
310. }
311.
312. /**************************************************************************
   ************/
313. /* Nombre: void horometro_caldera()
   */
314. /* Descripción: Calcula el tiempo que lleva funcionando la caldera
   */
315. /* Argumentos: llamada, actuador.
   */
316. /* Valor Devuelto: Ninguno
   */
317. /**************************************************************************
   ************/
```

```
318. void horometro_caldera(int llamada, bool actuador){
319.     if(llamada == 0){
320.         if(actuador == 0){
321.             estadisticas_permisivo_contador = 0;
322.         }else{
323.             estadisticas_permisivo_contador = 1;
324.         }
325.     }else{
326.         if(estadisticas_permisivo_contador == 1){
327.             estadisticas_mins_acumulados ++;
328.             if(estadisticas_mins_acumulados > 59){
329.                 estadisticas_horas_acumuladas ++;
330.                 estadisticas_mins_acumulados = 0;
331.             }
332.         }
333.     }
334. }//fin horometro_caldera()
335.
336. /*************************************************************************
   ************/
337. /* Nombre: actuador()
   */
338. /* Descripción: enciende o apaga la salida del actuador en funcion de la
   temp       */
339. /* Argumentos: temp, setpoint, cliente mqtt
   */
340. /* Valor Devuelto: nada
   */
341. /*************************************************************************
   ************/
342. void actuador(int temp, int setpoint, mosquittoMQTT* mqttHdl){
343.     //if(compara_mayor(temp,setpoint,1000) == 2 && on == 0)
344.     if(temp<=setpoint && on == 0)
345.     {
346.         cout<<"\n\t\t **Se ENCIENDE la calefacción\n"<<endl;
347.         mqttHdl->publish(NULL, mqttHdl->topic_ConsignaOnOff,
   1,"1",0,true);
348.         on = 1;
349.         off = 0;
350.         horometro_caldera(0,1);
351.     }
352.     //if(compara_mayor(temp,setpoint,1000) == 1 && off == 0)
353.     if(temp > setpoint && off==0)
354.     {
355.         cout<<"\n\t\t **Se APAGA la calefacción\n"<<endl;
356.         mqttHdl->publish(NULL, mqttHdl-
   >topic_ConsignaOnOff,1,"0",0,true);
357.         on = 0;
358.         off = 1;
359.         horometro_caldera(0,0);
360.     }
361. }
362.
363. int intervalo_actual(int intervalos_hora[], int intervalos_mins[]){
364.     int posicion;
365.
366.     for(int i=0; i<6; i++){
367.         if(intervalos_hora[i]==hora){
368.             if(intervalos_mins[i]<=mins){
369.                 posicion = i;
370.                 break;
```

```
371.             }else if(i==0 && intervalos_mins[i]>mins){
372.                 posicion = 5;
373.                 break;
374.             }else{
375.                 posicion = i-1;
376.                 break;
377.             }
378.         }else if(intervalos_hora[i]<hora){
379.             if(i!=5 && intervalos_hora[i+1]>hora){
380.                 posicion = i;
381.                 break;
382.             }else if(i==5){
383.                 posicion = 5;
384.                 break;
385.             }
386.         }else if(i==0 && intervalos_hora[i]>hora){
387.             posicion = 5;
388.             break;
389.         }
390.     }
391.
392.     return posicion;
393. }
394.
395. /*PROGRAMA PRINCIPAL, MAIN*/
396. int main()
397. {
398.     /*DECLARACIÓN DE VARIABLES*/
399.     list<Nodo> lista; // Lista de temperaturas.
400.     Nodo valor; // Variable de tipo nodo para almacenar los valores
   introducidos por el usuario.
401.     Nodo temp_max; //Cariable de tipo nodo para temperatura maxima
402.     Nodo temp_min; //Variable de tipo nodo para temperatura minima
403.     tempC_media=0.00;
404.
405.     mosquittoMQTT * mqttHdl;
406.     mqttHdl = new mosquittoMQTT(NULL, "localhost", 1883);
407.
408.     setlocale(LC_CTYPE, "Spanish"); // Reconoce los caracteres de la
   ortografia del castellano.
409.
410.     /*Imprime los valores de tipo float con una precision decimal de 2.*/
411.     cout.setf(ios::fixed, ios::floatfield);
412.     cout.precision(2);
413.
414.     /*FIN DECLARACIÓN DE VARIABLES*/
415.
416.     while(1)
417.     {
418.         //Mantiene la conexion
419.         int res = mqttHdl->loop();
420.         if (res){
421.             mqttHdl->reconnect();
422.         }
423.         Tiempo();
424.
425.         if(mins!=mins_old){
426.             mqttHdl->publish(NULL, mqttHdl-
   >topic_DiaHora,Dia_Hora.size(),Dia_Hora.c_str(),0,true);
427.             horometro_caldera(1,0);
428.             if(estadisticas_horas_acumuladas < 10){
```

```
429.                 horometro = "0" +
   to_string(estadisticas_horas_acumuladas) + " horas y ";
430.             }else{
431.                 horometro = to_string(estadisticas_horas_acumuladas) + "
   horas y ";
432.             }
433.             if(estadisticas_mins_acumulados < 10){
434.                 horometro = horometro + "0" +
   to_string(estadisticas_mins_acumulados) + " minutos";
435.             }else{
436.                 horometro = horometro +
   to_string(estadisticas_mins_acumulados) + " minutos";
437.             }
438.             mqttHdl->publish(NULL, mqttHdl-
   >topic_Horometro,horometro.size(),horometro.c_str(),0,true);
439.
440.             if(mqttHdl->tempC<=setpointTempC)
441.             {
442.                 mqttHdl->publish(NULL, mqttHdl->topic_ConsignaOnOff,
   1,"1",0,true);
443.
444.             }
445.             if(mqttHdl->tempC > setpointTempC)
446.             {
447.                 mqttHdl->publish(NULL, mqttHdl-
   >topic_ConsignaOnOff,1,"0",0,true);
448.             }
449.
450.             mins_old = mins;
451.         }
452.
453.         //Calcula la temperatura media cuando recibe un dato nuevo
454.         if(mqttHdl->permit_tempC == 1)
455.         {
456.             //temperatura_media(mqttHdl->tempC);
457.             cout<<"\n\t             ------------------------------------
   "<<endl;
458.             cout<<"\t            | INTRODUCIR NUEVO VALOR EN LA COLA
   |"<<endl;
459.             cout<<"\t             ------------------------------------
   \n"<<endl;
460.             cout<<"\t"<<endl;
461.
462.             valor.lectura = mqttHdl->tempC;
463.             valor.timestamp=Tiempo(); // El tiempo se le asigna
   automaticamente en el instante que ha sido
464.                                       // introducido la lectura.
465.
466.             if(lista.size()<100) // Si la lista tiene menos de 10
   valores.
467.             {
468.                 Insertar(lista,valor); // Insertamos un valor segun el
   criterio FIFO.
469.             }
470.             else // Si la lista tiene 10 o mas valores.
471.             {
472.                 /*Introducimos y sacamos un  valor segun el criterio
   FIFO.*/
473.                 Insertar(lista, valor);
474.                 Borrar(lista);
475.             }
```

```
476.
477.             Imprimir(lista);
478.
479.             cout<<"\t\t Setpoint: " << setpointTempC << endl;
480.             cout<<"\t\t Permiso Setpoint: " << mqttHdl->permit_setpoint
   << endl;
481.             cout<<"\t\t Intervalo: " << intervalo << endl;
482.             cout<<"\t\t Intervalo old: " << intervalo_old << endl;
483.             cout<<"\t\t Setpoint almacenado en el array diario: " <<
   mqttHdl->Intervalos_calendario_DiarioTemp[intervalo] << endl;
484.             cout<<"\t\t Setpoint almacenado en el array festivo: " <<
   mqttHdl->Intervalos_calendario_FestivoTemp[intervalo] << endl;
485.             cout<<"\t\t Setpoint old: " << setpointTempC_old << endl;
486.             cout<<"\t\t Dia: " << dia << endl;
487.             cout<<"\t\t Hora: " << hora << endl;
488.             cout<<"\t\t Minutos: " << mins << endl;
489.             cout<<"\t\t Modo trabajo: " << mqttHdl->modoTrabajo << endl;
490.             cout<<"\t\t Modo dia: " << mqttHdl->ModosDias_calendario[dia]
   <<endl;
491.
492.
493.             if(lista.size()>=5) // Si la lista contiene algun valor.
494.             {
495.                 MaxMin(lista, mqttHdl);
496.
497.                 tempC_media=media(lista);
498.
499.                 str_tempC_media = to_string((int)tempC_media) + " °C";
500.                 mqttHdl->publish(NULL, mqttHdl-
   >topic_TempMedia,str_tempC_media.size(),str_tempC_media.c_str(),0,true);
501.             }
502.             else // Si la lista esta vacia.
503.             {
504.                 cout<<"\n\t\t                  ----------------------------
   --"<<endl;
505.                 cout<<"\t\t                | NO HAY VALORES QUE COMPARAR
   |"<<endl;
506.                 cout<<"\t\t                 ----------------------------
   \n"<<endl;
507.                 str_tempC_media = "-- °C";
508.                 mqttHdl->publish(NULL, mqttHdl-
   >topic_TempMedia,str_tempC_media.size(),str_tempC_media.c_str(),0,true);
509.                 str_tempMax = "--  °C";
510.                 mqttHdl->publish(NULL, mqttHdl-
   >topic_TempMax,str_tempMax.size(),str_tempMax.c_str(),0,true);
511.                 str_tempMin = "-- °C";
512.                 mqttHdl->publish(NULL, mqttHdl-
   >topic_TempMin,str_tempMin.size(),str_tempMin.c_str(),0,true);
513.             }
514.
515.             mqttHdl->permit_tempC = 0;
516.         }
517.
518.         if(lista.size()>=5)
519.         {
520.             if(mqttHdl->estacionAno<1){
521.
522.                 if(mqttHdl->modoTrabajo<1){
523.
524.                     if(mqttHdl->ModosDias_calendario[dia]<1){//Diario
```

```
525.                         intervalo = intervalo_actual(mqttHdl-
   >Intervalos_calendario_DiarioHora, mqttHdl-
   >Intervalos_calendario_DiarioMins);
526.                         if(mqttHdl->permit_setpoint==1){
527.                             setpointTempC = mqttHdl->setpoint_tempC_MAN;
528.
529.                             if(setpointTempC > mqttHdl-
   >Intervalos_calendario_DiarioTemp[intervalo]){
530.                                 modoSemiAUT = 1;
531.                             }else if(setpointTempC < mqttHdl-
   >Intervalos_calendario_DiarioTemp[intervalo]){
532.                                 modoSemiAUT = 2;
533.                             }
534.
535.                             if(setpointTempC==mqttHdl-
   >Intervalos_calendario_DiarioTemp[intervalo] || intervalo_old!=intervalo){
536.                                 modoSemiAUT = 0;
537.                                 mqttHdl->permit_setpoint = 0;
538.                                 actualizar_SP = 1;
539.                             }
540.
541.                             if(modoSemiAUT!=modoSemiAUT_old){
542.                                 mqttHdl->publish(NULL, mqttHdl-
   >topic_modoSemiAUT,to_string(modoSemiAUT).size(),to_string(modoSemiAUT).c_s
   tr());
543.                                 modoSemiAUT_old=modoSemiAUT;
544.                             }
545.                         }else{
546.                             setpointTempC = mqttHdl-
   >Intervalos_calendario_DiarioTemp[intervalo];
547.                             if(actualizar_SP == 1 ||
   intervalo_old!=intervalo || setpointTempC_old!=setpointTempC){
548.                                 mqttHdl->publish(NULL, mqttHdl-
   >topic_setpoint,to_string(setpointTempC).size(),to_string(setpointTempC).c_
   str());
549.                                 actualizar_SP = 0;
550.                                 setpointTempC_old = setpointTempC;
551.
552.                                 if(intervalo!=intervalo_old){
553.                                     intervalo_str = "D: " +
   to_string(intervalo+1);
554.                                     mqttHdl->publish(NULL, mqttHdl-
   >topic_Intervalo,intervalo_str.size(),intervalo_str.c_str());
555.                                     intervalo_old = intervalo;
556.                                 }
557.                             }
558.                         }
559.
560.                     }else{//Festivo
561.                         intervalo = intervalo_actual(mqttHdl-
   >Intervalos_calendario_FestivoHora, mqttHdl-
   >Intervalos_calendario_FestivoMins);
562.                         if(mqttHdl->permit_setpoint==1){
563.                             setpointTempC = mqttHdl->setpoint_tempC_MAN;
564.
565.                             if(setpointTempC > mqttHdl-
   >Intervalos_calendario_FestivoTemp[intervalo]){
566.                                 modoSemiAUT = 1;
567.                             }else if(setpointTempC < mqttHdl-
   >Intervalos_calendario_FestivoTemp[intervalo]){
568.                                 modoSemiAUT = 2;
```

```
569.                                 }
570.
571.                                 if(setpointTempC==mqttHdl-
   >Intervalos_calendario_FestivoTemp[intervalo] || intervalo_old!=intervalo){
572.                                     modoSemiAUT = 0;
573.                                     mqttHdl->permit_setpoint = 0;
574.                                     actualizar_SP = 1;
575.                                 }
576.
577.                                 if(modoSemiAUT!=modoSemiAUT_old){
578.                                     mqttHdl->publish(NULL, mqttHdl-
   >topic_modoSemiAUT,to_string(modoSemiAUT).size(),to_string(modoSemiAUT).c_s
   tr());
579.                                     modoSemiAUT_old=modoSemiAUT;
580.                                 }
581.
582.                             }else{
583.                                 setpointTempC = mqttHdl-
   >Intervalos_calendario_FestivoTemp[intervalo];
584.                                 if(actualizar_SP == 1 ||
   intervalo_old!=intervalo || setpointTempC_old!=setpointTempC){
585.                                     mqttHdl->publish(NULL, mqttHdl-
   >topic_setpoint,to_string(setpointTempC).size(),to_string(setpointTempC).c_
   str());
586.                                     actualizar_SP = 0;
587.                                     setpointTempC_old = setpointTempC;
588.
589.                                     if(intervalo!=intervalo_old){
590.                                         intervalo_str = "F: " +
   to_string(intervalo+1);
591.                                         mqttHdl->publish(NULL, mqttHdl-
   >topic_Intervalo,intervalo_str.size(),intervalo_str.c_str());
592.                                         intervalo_old = intervalo;
593.                                     }
594.                                 }
595.                             }
596.                         }
597.
598.                     }else{
599.                         if(mqttHdl->permit_setpoint==1){
600.                             setpointTempC = mqttHdl->setpoint_tempC_MAN;
601.                         }
602.                     }
603.                 }else{
604.                     setpointTempC = -1;
605.                     setpointTempC_old = setpointTempC;
606.                     if(mqttHdl->permit_estacionAno == 1){
607.                         mqttHdl->publish(NULL, mqttHdl-
   >topic_setpoint,1,"0");
608.                         mqttHdl->permit_estacionAno = 0;
609.                     }
610.                 }
611.
612.                 actuador(mqttHdl->tempC,setpointTempC, mqttHdl);
613.             }
614.
615.         }
616.         return 0;
617. }
```

8.3.2. Mosquittomqtt.h

mosquittomqtt.h

```
#ifndef MOSQUITTOMQTT_H
#define MOSQUITTOMQTT_H

#include <mosquittopp.h>

#include <iostream>
#include <fstream>
#include <stdlib.h>
#include <cstring>
 #include <list>
 #include <time.h>
 #include <locale.h>
 #include <algorithm>

 class mosquittoMQTT : public mosqpp::mosquittopp
 {
     public:

         int tempC; //Variable de la temperatura
         int setpoint_tempC_MAN; //Variable del setpoint en manual

         //Variables auxiliares
         bool permit_tempC;
         bool permit_setpoint;
         bool permit_estacionAno;

         int ModosDias_calendario [7]; //Variable modo de los dias de la semana (Diario o Festivo)

         //Variables de los intervalos de diario y festivo:
         int Intervalos_calendario_DiarioTemp [6];
         int Intervalos_calendario_DiarioHora [6];
         int Intervalos_calendario_DiarioMins [6];
         int Intervalos_calendario_FestivoTemp [6];
         int Intervalos_calendario_FestivoHora [6];
         int Intervalos_calendario_FestivoMins [6];

         //Variables de los modos de funcionamiento:
         int estacionAno;
         int modoTrabajo;
         int modoCasa;
         int modoVacaciones;
         int tempEcoMode;

         //Topics
         const char* topic_tempActual = "SENSOR/Temperatura";
         const char* topic_ConsignaOnOff = "ACTUADOR/Consigna_OnOff";
         const char* topic_actuadorEstado = "ACTUADOR/Estado";
         const char* topic_MaxEnviarActuador = "ACTUADOR/Max_enviar";
         const char* topic_ConexionAPP = "CONTROL_PPAL/Conexion_APP";
         const char* topic_setpoint = "CONTROL_PPAL/SetpointTemp";
         const char* topic_setpoint_MAN = "CONTROL_PPAL/Home/SetpointMAN";
         const char* topic_modoSemiAUT = "CONTROL_PPAL/Home/ModoSemiAUT";
         const char* topic_DiaHora = "CONTROL_PPAL/Home/Dia_Hora";
         const char* topic_Intervalo = "CONTROL_PPAL/Home/Intervalo";
         const char* topic_ModosDias = "CONTROL_PPAL/Calendario/Modo_dias";
```

```
56.          const char* topic_Intervalos =
   "CONTROL_PPAL/Calendario/Intervalos";
57.          const char* topic_Horometro =
   "CONTROL_PPAL/Estadisticas/Horometro";
58.          const char* topic_TempMin = "CONTROL_PPAL/Estadisticas/TempMin";
59.          const char* topic_TempMax = "CONTROL_PPAL/Estadisticas/TempMax";
60.          const char* topic_TempMedia =
   "CONTROL_PPAL/Estadisticas/TempMedia";
61.          const char* topic_estacionAno =
   "CONTROL_PPAL/Ajustes/Estacion_Ano";
62.          const char* topic_modoTrabajo =
   "CONTROL_PPAL/Ajustes/Modo_Trabajo";
63.          const char* topic_modoCasa = "CONTROL_PPAL/Ajustes/Modo_Casa";
64.          const char* topic_temEcoMode =
   "CONTROL_PPAL/Ajustes/Temp_EcoMode";
65.          const char* topic_modoVacaciones =
   "CONTROL_PPAL/Ajustes/Modo_Vacaciones";
66.
67.          //Constructor y destructor de la clase
68.          mosquittoMQTT(const char *id, const char *host, int port);
69.          ~mosquittoMQTT();
70.
71.          //Definicion de los metodos MQTT
72.          void on_connect(int rc);
73.          void on_message(const struct mosquitto_message *message);
74.  };
75.
76.  #endif // MOSQUITTOMQTT_H
```

8.3.3. Mosquittomqtt.cpp

mosquittomqtt.cpp

```
1. #include "mosquittomqtt.h"
2.
3. mosquittoMQTT::mosquittoMQTT(const char *id, const char *host, int
   port):mosquittopp(id)
4. {
5.     //Constructor
6.     mosqpp::lib_init(); //Se inicializa libmosquitto
7.
8.   int keepalive = 120;
9.   connect(host, port, keepalive);               //Conectar con el broker
   MQTT
10.  }
11.
12.  mosquittoMQTT::~mosquittoMQTT()
13.  {
14.      //Destructor
15.      loop_stop(); //Se destruye el thread
16.      mosqpp::lib_cleanup();
17.  }
18.
19.  //Metodo que se llama cuando se ha establecido la conexion con el broker
20.  void mosquittoMQTT::on_connect(int rc)
21.  {
22.          std::cout<<"Conectado con el codigo: "<< rc<<".\n"<< std::endl;
23.      std::cout<<"_____________________________\n"<< std::endl;
24.          if (rc == 0)
25.          {
26.              //Suscripcion de los topic
27.           subscribe(NULL, topic_ConexionAPP);
28.                  subscribe(NULL, topic_tempActual);
29.                  subscribe(NULL, topic_setpoint_MAN);
30.                  subscribe(NULL, topic_estacionAno);
31.                  subscribe(NULL, topic_modoTrabajo);
32.                  subscribe(NULL, topic_modoCasa);
33.                  subscribe(NULL, topic_temEcoMode);
34.                  subscribe(NULL, topic_modoVacaciones);
35.                  subscribe(NULL, topic_ModosDias);
36.                  subscribe(NULL, topic_Intervalos);
37.
38.          }
39.  }
40.
41.  //Metodo que se llama cuando se recibe un mensaje de los topics a los que
   se esta suscrito
42.  void mosquittoMQTT::on_message(const struct mosquitto_message *message)
43.  {
44.      std::cout<<"\nRecibido: \n"<<"\t Topic: " << message->topic << "\n\t
   Mensaje: "<< (char *) message->payload << "\n"<< std::endl;
45.
46.      std::string t(message->topic);
47.      std::string p = std::string((char *)message->payload);
48.
49.      if(strcmp(t.c_str(),topic_ConexionAPP) == 0){
50.
51.      }else if(strcmp(t.c_str(),topic_tempActual) == 0){
52.          tempC = std::atoi (p.c_str());
53.          permit_tempC = 1;
```

```
54.      }else if(strcmp(t.c_str(),topic_setpoint_MAN) == 0){
55.          setpoint_tempC_MAN = std::atoi (p.c_str());
56.          permit_setpoint = 1;
57.      }else if(strcmp(t.c_str(), topic_ModosDias)==0){
58.          p=p.substr(3);
59.          for(int i=0; i<7; i++){
60.              ModosDias_calendario[i] = std::atoi(p.substr(i+i,1).c_str());
61.          }
62.      }else if(strcmp(t.c_str(),topic_Intervalos)==0){
63.          int d = p.find("D-");
64.          int f = p.find("F-");
65.          int t = 0;
66.          int h = 3;
67.          int m = 6;
68.          std::string diario = p.substr(d+2,54);
69.          std::string festivo = p.substr(f+2,54);
70.          for(int i=0; i<6;i++){
71.              Intervalos_calendario_DiarioTemp[i] =
   std::atoi(diario.substr(i+t,2).c_str());
72.              Intervalos_calendario_DiarioHora[i] =
   std::atoi(diario.substr(i+h,2).c_str());
73.              Intervalos_calendario_DiarioMins[i] =
   std::atoi(diario.substr(i+m,2).c_str());
74.              Intervalos_calendario_FestivoTemp[i] =
   std::atoi(festivo.substr(i+t,2).c_str());
75.              Intervalos_calendario_FestivoHora[i] =
   std::atoi(festivo.substr(i+h,2).c_str());
76.              Intervalos_calendario_FestivoMins[i] =
   std::atoi(festivo.substr(i+m,2).c_str());
77.              t = t + 8;
78.              h = h + 8;
79.              m = m + 8;
80.          }
81.      }else if(strcmp(t.c_str(), topic_estacionAno)==0){
82.          estacionAno = std::atoi(p.c_str());
83.          if(estacionAno == 1){
84.              permit_estacionAno = 1;
85.          }
86.      }else if(strcmp(t.c_str(), topic_modoTrabajo)==0){
87.          modoTrabajo = std::atoi(p.c_str());
88.      }else if(strcmp(t.c_str(), topic_modoCasa)==0){
89.          modoCasa = std::atoi(p.c_str());
90.      }else if(strcmp(t.c_str(), topic_modoVacaciones)==0){
91.          modoVacaciones = std::atoi(p.c_str());
92.      }else if(strcmp(t.c_str(), topic_temEcoMode)==0){
93.      }
94.
95.  }
```

8.4. Programa aplicación DELAC Smart

8.4.1. AndroidManifest.xml

```
1. <?xml version="1.0" encoding="utf-8"?>
2. <manifest xmlns:android="http://schemas.android.com/apk/res/android"
3.     package="com.delac.delacsmart">
4.
5.     <uses-permission android:name="android.permission.CAMERA" />
6.     <uses-permission android:name="android.permission.WAKE_LOCK" />
7.     <uses-permission android:name="android.permission.INTERNET" />
8.     <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"
   />
9.     <uses-permission android:name="android.permission.READ_PHONE_STATE" />
10.      <uses-permission
   android:name="android.permission.ACCESS_COARSE_LOCATION" />
11.     <uses-permission android:name="android.permission.ACCESS_WIFI_STATE"
   />
12.     <uses-permission android:name="android.permission.CHANGE_WIFI_STATE"
   />
13.
14.
15.     <application
16.         android:allowBackup="true"
17.         android:icon="@mipmap/ic_app_icon"
18.         android:label="@string/app_name"
19.         android:roundIcon="@mipmap/ic_app_icon_round"
20.         android:supportsRtl="true"
21.         android:theme="@style/AppTheme">
22.
23.         <activity
24.             android:name=".BienvenidaActivity"
25.             android:screenOrientation="portrait"
26.             android:theme="@style/BienvenidaTheme">
27.             <intent-filter>
28.                 <action android:name="android.intent.action.MAIN" />
29.
30.                 <category android:name="android.intent.category.LAUNCHER"
   />
31.             </intent-filter>
32.         </activity>
33.
34.         <activity
35.             android:name=".DispositivosActivity"
36.             android:screenOrientation="portrait" />
37.
38.         <service
   android:name="org.eclipse.paho.android.service.MqttService" />
39.     </application>
40.
41.  </manifest>
```

8.4.2. BienvenidaActivity

```
1. package com.delac.delacsmart;
2.
3. import android.content.Intent;
4. import android.os.Bundle;
5. import android.support.v7.app.AppCompatActivity;
6.
7. public class BienvenidaActivity extends AppCompatActivity {
8.     @Override
9.     protected void onCreate(Bundle savedInstanceState) {
10.          super.onCreate(savedInstanceState);
11.
12.          //Cuando se cargan la aplicación lanza la Activity
   DispositivosActivity
13.          Intent intent = new Intent(this, DispositivosActivity.class);
14.          startActivity(intent);
15.          finish();
16.      }
17.  }
```

8.4.3. DispositivosActivityy

```
1. package com.delac.delacsmart;
2.
3. import android.net.Uri;
4. import android.os.Bundle;
5. import android.support.annotation.NonNull;
6. import android.support.design.widget.BottomNavigationView;
7. import android.support.v4.app.DialogFragment;
8. import android.support.v4.app.Fragment;
9. import android.support.v4.app.FragmentManager;
10.  import android.support.v7.app.AlertDialog;
11.  import android.support.v7.app.AppCompatActivity;
12.  import android.view.LayoutInflater;
13.  import android.view.MenuItem;
14.  import android.view.View;
15.  import android.widget.Button;
16.  import android.widget.Toast;
17.
18.  import org.eclipse.paho.android.service.MqttAndroidClient;
19.  import org.eclipse.paho.client.mqttv3.IMqttActionListener;
20.  import org.eclipse.paho.client.mqttv3.IMqttDeliveryToken;
21.  import org.eclipse.paho.client.mqttv3.IMqttToken;
22.  import org.eclipse.paho.client.mqttv3.MqttCallback;
23.  import org.eclipse.paho.client.mqttv3.MqttClient;
24.  import org.eclipse.paho.client.mqttv3.MqttException;
25.  import org.eclipse.paho.client.mqttv3.MqttMessage;
26.
27.  import java.io.UnsupportedEncodingException;
28.
29.  public class DispositivosActivity extends AppCompatActivity implements
     HomeFragment.OnFragmentInteractionListener,
     CalendarioFragment.OnFragmentInteractionListener,
30.  EstadisticasFragment.OnFragmentInteractionListener,
     AjustesFragment.OnFragmentInteractionListener,
     DatePickerClass.OnDateSetListener, TimePickerClass.OnTimeSetListener{
31.
32.      private MqttAndroidClient client;
33.
34.      String topic_deltaSensor = "SENSOR/Delta";
35.      String topic_MaxEnviarSensor = "SENSOR/Max_enviar";
36.      String topic_RastreoSensor = "SENSOR/Rastreo";
37.      String topic_tempActual = "SENSOR/Temperatura";
38.
39.      String topic_actuadorEstado = "ACTUADOR/Estado";
40.      String topic_MaxEnviarActuador = "ACTUADOR/Max_enviar";
41.
42.      String topic_ConexionAPP = "CONTROL_PPAL/Conexion_APP";
43.
44.      String topic_setpoint = "CONTROL_PPAL/Setpoint_Temp";
45.
46.      String topic_setpoint_MAN = "CONTROL_PPAL/Home/Setpoint_MAN";
47.      String topic_modoSemiAUT = "CONTROL_PPAL/Home/Modo_SemiAUT";
48.      String topic_DiaHora = "CONTROL_PPAL/Home/Dia_Hora";
49.      String topic_Intervalo = "CONTROL_PPAL/Home/Intervalo";
50.
51.      String topic_tempMedia = "CONTROL_PPAL/Estadisticas/TempMedia";
52.      String topic_tempMin = "CONTROL_PPAL/Estadisticas/TempMin";
53.      String topic_tempMax = "CONTROL_PPAL/Estadisticas/TempMax";
54.      String topic_horametro = "CONTROL_PPAL/Estadisticas/Horometro";
```

```
55.
56.      String topic_ModosDias = "CONTROL_PPAL/Calendario/Modo_dias";
57.      String topic_Intervalos = "CONTROL_PPAL/Calendario/Intervalos";
58.      String topic_estacionAno = "CONTROL_PPAL/Ajustes/Estacion_Ano";
59.      String topic_modoTrabajo = "CONTROL_PPAL/Ajustes/Modo_Trabajo";
60.      String topic_modoCasa ="CONTROL_PPAL/Ajustes/Modo_Casa";
61.      String topic_temEcoMode = "CONTROL_PPAL/Ajustes/Temp_EcoMode";
62.      String topic_modoVacaciones = "CONTROL_PPAL/Ajustes/Modo_Vacaciones";
63.
64.      String tempSP;
65.
66.      int setpoint = 15;
67.      int temperaturaActual = 0;
68.      int estadoCaldera = 0;
69.      int estado_mqtt = 0;
70.      int modoSemiAUT = 0;
71.
72.      HomeFragment fragment_home =
   HomeFragment.newInstance(estado_mqtt,temperaturaActual,estadoCaldera);
73.      CalendarioFragment fragment_calendario = new CalendarioFragment();
74.      EstadisticasFragment fragment_estadisticas = new
   EstadisticasFragment();
75.      AjustesFragment fragment_ajustes = new AjustesFragment();
76.      FragmentManager fm = getSupportFragmentManager();
77.      Fragment active = fragment_home;
78.
79.      @Override
80.      protected void onCreate(Bundle savedInstanceState) {
81.          super.onCreate(savedInstanceState);
82.          setContentView(R.layout.activity_dispositivos);
83.
84.          BottomNavigationView navigation = (BottomNavigationView)
   findViewById(R.id.navigation);
85.
   navigation.setOnNavigationItemSelectedListener(mOnNavigationItemSelectedLis
   tener);
86.          fm.beginTransaction().add(R.id.fragment_container,
   fragment_ajustes, "4").hide(fragment_ajustes).commit();
87.          fm.beginTransaction().add(R.id.fragment_container,
   fragment_estadisticas, "3").hide(fragment_estadisticas).commit();
88.          fm.beginTransaction().add(R.id.fragment_container,
   fragment_calendario, "2").hide(fragment_calendario).commit();
89.          fm.beginTransaction().add(R.id.fragment_container,fragment_home,
   "1").commit();
90.
91.          //clienteMQTT();
92.
93.      }
94.
95.      @Override
96.      protected void onResume(){
97.          super.onResume();
98.          clienteMQTT();
99.      }
100.
101.     @Override
102.     protected void onPause(){
103.         super.onPause();
104.         //publicarMensaje(topic_ConexionAPP, "0");
105.     }
106.
```

```
107.     private BottomNavigationView.OnNavigationItemSelectedListener
   mOnNavigationItemSelectedListener = new
   BottomNavigationView.OnNavigationItemSelectedListener() {
108.
109.         @Override
110.         public boolean onNavigationItemSelected(@NonNull MenuItem item) {
111.             switch (item.getItemId()) {
112.                 case R.id.navigation_home:
113.
   fm.beginTransaction().hide(active).show(fragment_home).commit();
114.                     active = fragment_home;
115.                     return true;
116.                 case R.id.navigation_calendario:
117.
   fm.beginTransaction().hide(active).show(fragment_calendario).commit();
118.                     active = fragment_calendario;
119.                     return true;
120.                 case R.id.navigation_estadisticas:
121.
   fm.beginTransaction().hide(active).show(fragment_estadisticas).commit();
122.                     active = fragment_estadisticas;
123.                     return true;
124.                 case R.id.navigation_ajustes:
125.
   fm.beginTransaction().hide(active).show(fragment_ajustes).commit();
126.                     active = fragment_ajustes;
127.                     return true;
128.             }
129.
130.                 return false;
131.             }
132.         };
133.
134.     //Función para mostrar el selector de fecha
135.     public void mostrarDatePicker(){
136.         DialogFragment fragment_date = new DatePickerClass();
137.         fragment_date.show(getSupportFragmentManager(), "date picker");
138.     }
139.
140.     //Función para mostrar el selector de fecha
141.     public void mostrarTimePicker(){
142.         DialogFragment fragment_time = new TimePickerClass();
143.         fragment_time.show(getSupportFragmentManager(), "time picker");
144.     }
145.
146.     //Funcion que controla la comunicacion MQTT
147.     private void clienteMQTT(){
148.         String clientId = MqttClient.generateClientId();
149.         client=new MqttAndroidClient(this.getApplicationContext(),
   "tcp://62.42.147.97:1883",
150.                 clientId);
151.
152.         try {
153.             IMqttToken token = client.connect();
154.             token.setActionCallback(new IMqttActionListener() {
155.                 @Override
156.                 public void onSuccess(IMqttToken asyncActionToken) {
157.                     // We are connected
158.                     estado_mqtt=1;
159.                     fragment_home.cambiarEstadoMQTT(estado_mqtt);
```

```
160.                     Toast . makeText ( getApplicationContext() ,
   "Conexion establecida" , Toast.LENGTH_LONG).show();
161.                     publicarMensaje(topic_ConexionAPP,"1");
162.                     subscribirTopic(topic_tempActual);
163.                     subscribirTopic(topic_actuadorEstado);
164.                     subscribirTopic(topic_setpoint);
165.                     subscribirTopic(topic_modoSemiAUT);
166.                     subscribirTopic(topic_DiaHora);
167.                     subscribirTopic(topic_Intervalo);
168.                     subscribirTopic(topic_tempMedia);
169.                     subscribirTopic(topic_tempMin);
170.                     subscribirTopic(topic_tempMax);
171.                     subscribirTopic(topic_horametro);
172.                 }
173.
174.                 @Override
175.                 public void onFailure(IMqttToken asyncActionToken,
   Throwable exception) {
176.                     // Something went wrong e.g. connection timeout or
   firewall problems
177.                     estado_mqtt=0;
178.                     fragment_home.cambiarEstadoMQTT(estado_mqtt);
179.                     Toast . makeText ( getApplicationContext() ,
   "Conexion fallida" , Toast.LENGTH_LONG).show();
180.                     dialogoConexion();
181.                 }
182.             });
183.         } catch (MqttException e) {
184.             e.printStackTrace();
185.         }
186.
187.         client.setCallback(new MqttCallback() {
188.             @Override
189.             public void connectionLost(Throwable cause) {
190.                 clienteMQTT();
191.             }
192.
193.             @Override
194.             public void messageArrived(String topic, MqttMessage message)
   throws Exception {
195.                 //Toast . makeText ( getApplicationContext() , "Topic: "
   + topic , Toast.LENGTH_LONG).show();
196.
197.                 if(topic.compareTo(topic_tempActual)==0) {
198.                     temperaturaActual = Integer.parseInt(new
   String(message.getPayload()));
199.                     fragment_home.cambiarTempActual(temperaturaActual);
200.                     //Toast.makeText(getApplicationContext(), "Mensaje
   Recibido tempActual", Toast.LENGTH_LONG).show();
201.                 }else if(topic.compareTo(topic_actuadorEstado)==0){
202.                     //Toast.makeText(getApplicationContext(), "Mensaje
   Recibido actuadorEstado", Toast.LENGTH_LONG).show();
203.                     estadoCaldera = Integer.parseInt(new
   String(message.getPayload()));
204.                     fragment_home.cambiarEstadoCaldera(estadoCaldera);
205.                 }else if(topic.compareTo(topic_setpoint)==0){
206.                     setpoint = Integer.parseInt(new
   String(message.getPayload()));
207.                     fragment_home.cambiarTempSP(setpoint);
208.                 }else if(topic.compareTo(topic_modoSemiAUT)==0){
```

```
209.                     modoSemiAUT = Integer.parseInt(new
   String(message.getPayload()));
210.                     fragment_home.actualizarModoSemiAUT(modoSemiAUT);
211.                 }else if(topic.compareTo(topic_DiaHora)==0){
212.                     fragment_home.cambiarDiaHora(new
   String(message.getPayload()));
213.                 }else if(topic.compareTo(topic_Intervalo)==0){
214.                     fragment_home.cambiarIntervaloActual(new
   String(message.getPayload()));
215.                 }else if(topic.compareTo(topic_tempMedia)==0){
216.                     fragment_estadisticas.cambiarTempMedia(new
   String(message.getPayload()));
217.                 }else if(topic.compareTo(topic_tempMin)==0){
218.                     fragment_estadisticas.cambiarTempMin(new
   String(message.getPayload()));
219.                 }else if(topic.compareTo(topic_tempMax)==0){
220.                     fragment_estadisticas.cambiarTempMax(new
   String(message.getPayload()));
221.                 }else if(topic.compareTo(topic_horametro)==0){
222.                     fragment_estadisticas.cambiarHorametro(new
   String(message.getPayload()));
223.                 }
224.             }
225.             @Override
226.             public void deliveryComplete(IMqttDeliveryToken token) {
227.
228.             }
229.         });
230.     }
231.
232.     public void dialogoConexion(){
233.         LayoutInflater inflater = this.getLayoutInflater();
234.         View view = inflater.inflate(R.layout.dialogo_conexion, null);
235.         AlertDialog.Builder dialogo = new
   AlertDialog.Builder(DispositivosActivity.this);
236.         dialogo.setView(view);
237.         final AlertDialog alert = dialogo.create();
238.
239.         final Button Boton_aceptar = (Button)
   view.findViewById(R.id.aceptar_boton);
240.         final Button Boton_cancelar = (Button)
   view.findViewById(R.id.cancelar_boton);
241.
242.         Boton_aceptar.setOnClickListener(new View.OnClickListener() {
243.             @Override
244.             public void onClick(View v) {
245.                 clienteMQTT();
246.                 alert.dismiss();
247.             }
248.         });
249.
250.         Boton_cancelar.setOnClickListener(new View.OnClickListener() {
251.             @Override
252.             public void onClick(View v) {
253.                 alert.dismiss();
254.             }
255.         });
256.         alert.show();
257.     }
258.
259.     private void subscribirTopic(String topic){
```

```
260.          int qos = 1;
261.
262.          try {
263.              client.subscribe(topic, qos);
264.          } catch (MqttException e) {
265.              e.printStackTrace();
266.          }
267.      }
268.
269.      public void publicarMensaje(String topic, String payload){
270.          byte[] encodedPayload = new byte[0];
271.          try {
272.              encodedPayload = payload.getBytes("UTF-8");
273.              MqttMessage message = new MqttMessage(encodedPayload);
274.              client.publish(topic, message);
275.              //Toast.makeText(getApplicationContext(), "Mensaje
    publicado", Toast.LENGTH_LONG).show();
276.          } catch (UnsupportedEncodingException | MqttException e) {
277.              e.printStackTrace();
278.          }
279.      }
280.
281.      //Funcion que modifica el valor del setpoint y lo publica
282.      public void publicarSetPoint() {
283.          publicarMensaje(topic_setpoint_MAN,tempSP);
284.
285.      }
286.
287.      //Función que publica el estado de los modos de cada dia de la semana
288.      public void publicarModosDias(int[] Dias_modo){
289.          String mensaje = "MD-";
290.          for(int i=0; i<7;i++){
291.              mensaje = mensaje + Integer.toString(Dias_modo[i]) +";";
292.          }
293.          //Toast.makeText(getApplicationContext(), mensaje,
    Toast.LENGTH_LONG).show();
294.          try {
295.              publicarMensaje(topic_ModosDias,mensaje);
296.          }catch (Exception e){
297.
298.          }
299.      }
300.
301.      public void publicarIntervalos(int[] DiarioTemp, int[] DiarioHora,
    int[] DiarioMin, int[] FestivoTemp, int[] FestivoHora, int[] FestivoMin){
302.          String mensaje = "D-";
303.          for(int i=0; i<6;i++){
304.
305.              if(DiarioTemp[i]<10){
306.                  mensaje = mensaje + "0" + Integer.toString(DiarioTemp[i])
    +"_";
307.              }else{
308.                  mensaje = mensaje + Integer.toString(DiarioTemp[i]) +"_";
309.              }
310.              if(DiarioHora[i]<10){
311.                  mensaje = mensaje + "0" + Integer.toString(DiarioHora[i])
    + ":";
312.              }else{
313.                  mensaje = mensaje + Integer.toString(DiarioHora[i]) +
    ":";
314.              }
```

```
315.             if(DiarioMin[i]<10){
316.                 mensaje = mensaje + "0" + Integer.toString(DiarioMin[i])
   +";";
317.             }else {
318.                 mensaje = mensaje + Integer.toString(DiarioMin[i]) +";";
319.             }
320.         }
321.         mensaje = mensaje + "F-";
322.         for(int k=0; k<6;k++){
323.             if(FestivoTemp[k]<10){
324.                 mensaje = mensaje + "0" +
   Integer.toString(FestivoTemp[k]) +"_";
325.             }else{
326.                 mensaje = mensaje + Integer.toString(FestivoTemp[k])
   +"_";
327.             }
328.             if(FestivoHora[k]<10){
329.                 mensaje = mensaje + "0" +
   Integer.toString(FestivoHora[k]) + ":";
330.             }else{
331.                 mensaje = mensaje + Integer.toString(FestivoHora[k]) +
   ":";
332.             }
333.             if(FestivoMin[k]<10){
334.                 mensaje = mensaje + "0" + Integer.toString(FestivoMin[k])
   +";";
335.             }else {
336.                 mensaje = mensaje + Integer.toString(FestivoMin[k]) +";";
337.             }
338.         }
339.
340.         Toast.makeText(getApplicationContext(), mensaje,
   Toast.LENGTH_LONG).show();
341.         publicarMensaje(topic_Intervalos,mensaje);
342.     }
343.
344.     //Funcion para publicar el valor de la Estación del año
345.     public void publicarEstacionAno(int estado){
346.         String mensaje = Integer.toString(estado);
347.         publicarMensaje(topic_estacionAno, mensaje);
348.         Toast.makeText(this, "Publicar estacion año",
   Toast.LENGTH_SHORT).show();
349.     }
350.
351.     //Funcion para publicar el valor de la Estación del año
352.     public void publicarModoTrabajo(int estado){
353.         String mensaje = Integer.toString(estado);
354.         publicarMensaje(topic_modoTrabajo, mensaje);
355.     }
356.
357.     //Funcion para publicar el valor del modo Casa
358.     public void publicarCasa(int estado){
359.         String mensaje = Integer.toString(estado);
360.         publicarMensaje(topic_modoCasa, mensaje);
361.     }
362.
363.     //Funcion para publicar el valor de la temperatura Eco Mode
364.     public void publicarTempEcoMode(String valor){
365.         publicarMensaje(topic_temEcoMode, valor);
366.     }
367.
```

```
368.     //Funcion para publicar el valor del modo Vacaciones
369.     public void publicarModoVacaciones(int estado){
370.         String mensaje = Integer.toString(estado);
371.         publicarMensaje(topic_modoVacaciones, mensaje);
372.     }
373.
374.     //Funcion para publicar el valor del delta de temperatura
375.     public void publicarTempDelta(String valor){
376.         publicarMensaje(topic_deltaSensor, valor);
377.     }
378.
379.     //Funcion para publicar el valor del tiempo maximo de envio del
    sensor
380.     public void publicarTiempoMaxEnviar_Sensor(String valor){
381.         publicarMensaje(topic_MaxEnviarSensor, valor);
382.     }
383.
384.     //Funcion para publicar el valor del tiempo de rastreo
385.     public void publicarTiempoRastreo(String valor){
386.         publicarMensaje(topic_RastreoSensor, valor);
387.     }
388.
389.     //Funcion para publicar el valor del tiempo maximo de envio del
    actuador
390.     public void publicarTiempoMaxEnvio_Actuador(String valor){
391.         publicarMensaje(topic_MaxEnviarActuador, valor);
392.     }
393.
394.     /**FUNCIONES DE INTERACCION CON LOS FRAGMENT**/
395.
396.     public void onFragmentInteraction_DatePicker(int year, int month, int
    day){
397.         fragment_ajustes.cambiar_Fecha(year,month,day);
398.     }
399.
400.     public void onFragmentInteraction_TimePicker(int hora, int minuto){
401.         fragment_ajustes.cambiar_Hora(hora, minuto);
402.     }
403.
404.     //Funcion que recibe los datos del fragment HOME
405.     public void onFragmentInteraction_Home(String f_tempSP){
406.         tempSP = f_tempSP;
407.         publicarSetPoint();
408.     }
409.
410.     //Funcion que recibe los datos del fragment CALENDARIO_ModosDias
411.     public void onFragmentInteraction_Calendario_ModosDias( int[]
    Dias_Modo){
412.         publicarModosDias(Dias_Modo);
413.     }
414.
415.     public void onFragmentInteracion_Calendario_Intervalos(int[]
    DiarioTemp, int[] DiarioHora, int[] DiarioMin, int[] FestivoTemp, int[]
    FestivoHora, int[] FestivoMin){
416.
    publicarIntervalos(DiarioTemp,DiarioHora,DiarioMin,FestivoTemp,FestivoHora,
    FestivoMin);
417.     }
418.
419.     public void onFragmentInteraction_Estadisticas (Uri uri){
420.
```

```
421.     }
422.     //Funcion que recibe los datos del fragment AJUSTES_EstacionAno
423.     public void onFragmentInteraction_Ajustes_EstacionAno (int valor){
424.         publicarEstacionAno(valor);
425.         fragment_home.cambiarEstacionAno_home(valor);
426.     }
427.     //Funcion que recibe los datos del fragment AJUSTES_ModoTrabajo
428.     public void onFragmentInteraction_Ajustes_ModoTrabajo(int valor){
429.         publicarModoTrabajo(valor);
430.         fragment_home.cambiarModoTrabajo_home(valor);
431.     }
432.     //Funcion que recibe los datos del fragment AJUSTES_Casa
433.     public void onFragmentInteraction_Ajustes_Casa(int valor){
434.         if (valor == 1){
435.             mostrarTimePicker();
436.         }
437.         fragment_home.cambiarModoEco_home(valor);
438.         fragment_home.cambiarModoCasaVacaciones(valor);
439.         publicarCasa(valor);
440.     }
441.     //Funcion que recibe los datos del fragment AJUSTES_ModoVacaciones
442.     public void onFragmentInteraction_Ajustes_ModoVacaciones(int valor){
443.         if(valor==1) {
444.             mostrarDatePicker();
445.             valor=2;
446.         }
447.         fragment_home.cambiarModoEco_home(valor);
448.         fragment_home.cambiarModoCasaVacaciones(valor);
449.         publicarModoVacaciones(valor);
450.     }
451.     //Funcion que recibe los datos del fragment AJUSTES_TempEcoMode
452.     public void onFragmentInteraction_Ajustes_TempEcoMode(String valor){
453.         publicarTempEcoMode(valor);
454.     }
455.     //Funcion que recibe los datos del fragment AJUSTES_TempDelta
456.     public void onFragmentInteraction_Ajustes_TempDelta(String valor){
457.         publicarTempDelta(valor);
458.     }
459.     //Funcion que recibe los datos del fragment
    AJUSTES_TiempoMaxEnviarSensor
460.     public void
    onFragmentInteraction_Ajustes_TiempoMaxEnviarSensor(String valor){
461.         publicarTiempoMaxEnviar_Sensor(valor);
462.     }
463.     //Funcion que recibe los datos del fragment AJUSTES_TiempoRastreo
464.     public void onFragmentInteraction_Ajustes_TiempoRastreo(String
    valor){
465.         publicarTiempoRastreo(valor);
466.     }
467.     //Funcion que recibe los datos del fragment
    AJUSTES_TiempoMaxEnviarActuador
468.     public void
    onFragmentInteraction_Ajustes_TiempoMaxEnviarActuador(String valor){
469.         publicarTiempoMaxEnvio_Actuador(valor);
470.     }
471. }
```

8.4.4. HomeFragment

```
1. package com.delac.delacsmart;
2.
3. import android.content.Context;
4. import android.graphics.drawable.Drawable;
5. import android.os.Bundle;
6. import android.support.v4.app.Fragment;
7. import android.view.LayoutInflater;
8. import android.view.View;
9. import android.view.ViewGroup;
10.  import android.widget.ImageButton;
11.  import android.widget.ImageView;
12.  import android.widget.ProgressBar;
13.  import android.widget.TextView;
14.
15.
16.  public class HomeFragment extends Fragment  {
17.
18.      private OnFragmentInteractionListener mListener;
19.
20.      TextView ConexionMQTT;
21.      TextView TempActual;
22.      TextView TempSP;
23.      TextView Tv_Dia_Hora;
24.      TextView Tv_Intervalo;
25.
26.      ImageButton BotonUP_sp;
27.      ImageButton BotonDOWN_sp;
28.
29.      ImageView I_ConexionMQTT;
30.      ImageView Img_fuego;
31.
32.      ImageView Img_EstacionAno_home;
33.      ImageView Img_ModoEco_home;
34.      ImageView Img_ModoCasaVacaciones_home;
35.      ImageView Img__flecha_down_SP;
36.      ImageView Img__flecha_up_SP;
37.      ImageView Img_ModoTrabajo_home;
38.
39.      ProgressBar mProgress;
40.      ProgressBar mProgressSP;
41.
42.      static String ESTADO_MQTT = "ESTADO_MQTT";
43.      static String TEMP_ACTUAL = "TEMP_ACTUAL";
44.      static String ESTADO_CALDERA = "ESTADO_CALDERA";
45.
46.      int SetPoint = 15;
47.      int f_estado_mqtt;
48.      int f_tempActual;
49.      int f_estadoCaldera;
50.
51.
52.      public static HomeFragment newInstance(int estado_mqtt, int
   tempActual, int estadoCaldera){
53.          HomeFragment f = new HomeFragment();
54.          Bundle bundle = new Bundle();
55.          bundle.putInt(ESTADO_MQTT, estado_mqtt);
56.          bundle.putInt(TEMP_ACTUAL,tempActual);
57.          bundle.putInt(ESTADO_CALDERA, estadoCaldera);
```

```
58.           f.setArguments(bundle);
59.           return f;
60.       }
61.
62.       @Override
63.       public void onCreate(Bundle savedInstanceState) {
64.           super.onCreate(savedInstanceState);
65.           if (getArguments() != null) {
66.               f_estado_mqtt = getArguments().getInt(ESTADO_MQTT);
67.               f_tempActual= getArguments().getInt(TEMP_ACTUAL);
68.               f_estadoCaldera = getArguments().getInt(ESTADO_CALDERA);
69.           }
70.       }
71.
72.
73.       @Override
74.       public View onCreateView(LayoutInflater inflater, ViewGroup
    container, Bundle savedInstanceState) {
75.           // Inflate the layout for this fragment
76.           View view = inflater.inflate(R.layout.fragment_home, container,
    false);
77.           ConexionMQTT = (TextView) view.findViewById(R.id.conexionMQTT);
78.           I_ConexionMQTT = (ImageView)
    view.findViewById(R.id.i_conexionMQTT);
79.           Img_ModoTrabajo_home = (ImageView)
    view.findViewById(R.id.img_ModoTrabajo_home);
80.           BotonUP_sp = (ImageButton) view.findViewById(R.id.botonUP_sp);
81.           BotonDOWN_sp = (ImageButton)
    view.findViewById(R.id.botonDOWN_sp);
82.           TempActual = (TextView) view.findViewById(R.id.tempActual);
83.           TempSP = (TextView) view.findViewById(R.id.tempSP);
84.           mProgressSP = (ProgressBar)
    view.findViewById(R.id.sp_Progressbar);
85.           mProgress = (ProgressBar)
    view.findViewById(R.id.circularProgressbar);
86.           Drawable PB_Circular =
    getResources().getDrawable(R.drawable.pb_temperatura);
87.           mProgress.setProgress(0);    // Main Progress
88.           mProgress.setMax(60); // Maximum Progress
89.           mProgress.setProgressDrawable(PB_Circular);
90.           Drawable PB_Circular_SP =
    getResources().getDrawable(R.drawable.pb_setpoint);
91.           mProgressSP.setProgress(15);
92.           mProgressSP.setMax(60);
93.           mProgressSP.setProgressDrawable(PB_Circular_SP);
94.           Img__flecha_down_SP = (ImageView)
    view.findViewById(R.id.img_flecha_down_SP);
95.           Img__flecha_up_SP = (ImageView)
    view.findViewById(R.id.img_flecha_up_SP);
96.           Tv_Dia_Hora = (TextView) view.findViewById(R.id.tv_Dia_Hora);
97.           Tv_Intervalo = (TextView) view.findViewById(R.id.tv_Intervalo);
98.           Img_fuego = (ImageView) view.findViewById(R.id.img_fuego);
99.           Img_EstacionAno_home = (ImageView)
    view.findViewById(R.id.img_EstacionAno_home);
100.          Img_ModoEco_home = (ImageView)
    view.findViewById(R.id.img_ModoEco_home);
101.          Img_ModoCasaVacaciones_home = (ImageView)
    view.findViewById(R.id.img_ModoCasaVacaciones_home);
102.
103.
104.
```

```
105.          cambiarEstadoMQTT(f_estado_mqtt);
106.          cambiarTempActual(f_tempActual);
107.          cambiarTempSP_btn();
108.          cambiarEstadoCaldera(f_estadoCaldera);
109.
110.          return view;
111.      }
112.
113.      @Override
114.      public void onAttach(Context context) {
115.          super.onAttach(context);
116.          if (context instanceof OnFragmentInteractionListener) {
117.              mListener = (OnFragmentInteractionListener) context;
118.          } else {
119.              throw new RuntimeException(context.toString() + " must
    implement OnFragmentInteractionListener");
120.          }
121.      }
122.
123.      @Override
124.      public void onDetach() {
125.          super.onDetach();
126.          mListener = null;
127.      }
128.
129.      public void cambiarEstadoMQTT(int estado){
130.          Drawable I_MQTT_Conectado =
    getResources().getDrawable(R.drawable.ic_mqtt_conectado_24dp);
131.          Drawable I_MQTT_Desconectado =
    getResources().getDrawable(R.drawable.ic_mqtt_desconectado_24dp);
132.          if(estado==0){
133.              I_ConexionMQTT.setImageDrawable(I_MQTT_Desconectado);
134.              ConexionMQTT.setText("DESCONECTADO");
135.              f_estado_mqtt=0;
136.          }else{
137.              I_ConexionMQTT.setImageDrawable(I_MQTT_Conectado);
138.              ConexionMQTT.setText("CONECTADO");
139.              f_estado_mqtt=1;
140.          }
141.      }
142.
143.      public void cambiarDiaHora(String DiaHora){
144.          Tv_Dia_Hora.setText(DiaHora);
145.      }
146.
147.      public void cambiarIntervaloActual(String Intervalo){
148.          Drawable IC_CALENDAR =
    getResources().getDrawable(R.drawable.nav_calendario_24dp);
149.          Img_ModoTrabajo_home.setImageDrawable(IC_CALENDAR);
150.          Tv_Intervalo.setText(Intervalo);
151.      }
152.
153.      public void cambiarTempActual(int temperatura){
154.          String s = Integer.toString(temperatura);
155.          if(temperatura>40){
156.              temperatura=40;
157.              s="40";
158.          }
159.          if(temperatura<0){
160.              temperatura=0;
161.              s="0";
```

```
162.         }
163.         s=s+"°C";
164.         TempActual.setText(s);
165.         mProgress.setProgress(temperatura);
166.     }
167.
168.     public void cambiarTempSP(int sp){
169.         String msg = Integer.toString(sp);
170.         msg = msg +"°C";
171.         TempSP.setText(msg);
172.         mProgressSP.setProgress(sp);
173.         SetPoint = sp;
174.     }
175.
176.     public void actualizarModoSemiAUT(int modoSemiAUT){
177.         Drawable I_DOWN =
   getResources().getDrawable(R.drawable.ic_arrow_down_24dp);
178.         Drawable I_UP =
   getResources().getDrawable(R.drawable.ic_arrow_up_24dp);
179.         if(modoSemiAUT == 0){
180.             Img__flecha_up_SP.setImageDrawable(null);
181.             Img__flecha_down_SP.setImageDrawable(null);
182.         }else if(modoSemiAUT == 1){
183.             Img__flecha_up_SP.setImageDrawable(I_UP);
184.             Img__flecha_down_SP.setImageDrawable(null);
185.         }else if(modoSemiAUT == 2){
186.             Img__flecha_up_SP.setImageDrawable(null);
187.             Img__flecha_down_SP.setImageDrawable(I_DOWN);
188.         }
189.     }
190.
191.     public void cambiarTempSP_btn(){
192.         BotonUP_sp.setOnClickListener(new View.OnClickListener() {
193.             @Override
194.             public void onClick(View v) {
195.                 SetPoint++;
196.                 if (SetPoint > 40) {
197.                     SetPoint = 40;
198.                 }
199.                 String msg = Integer.toString(SetPoint);
200.                 mListener.onFragmentInteraction_Home(msg);
201.                 cambiarTempSP(SetPoint);
202.             }
203.         });
204.
205.         BotonDOWN_sp.setOnClickListener(new View.OnClickListener() {
206.             @Override
207.             public void onClick(View v) {
208.                 SetPoint--;
209.                 if (SetPoint < 0) {
210.                     SetPoint = 0;
211.                 }
212.                 String msg = Integer.toString(SetPoint);
213.                 mListener.onFragmentInteraction_Home(msg);
214.                 cambiarTempSP(SetPoint);
215.             }
216.         });
217.     }
218.
219.     public void cambiarEstadoCaldera(int estado){
```

```
220.         Drawable IC_Fuego =
   getResources().getDrawable(R.drawable.ic_fuego);
221.         if(estado==0){
222.             Img_fuego.setImageDrawable(null);
223.         }else{
224.             Img_fuego.setImageDrawable(IC_Fuego);
225.         }
226.     }
227.
228.     public void cambiarModoTrabajo_home(int estado){
229.         Drawable IC_TRABAJO =
   getResources().getDrawable(R.drawable.ic_modo_manual_24dp);
230.         Drawable IC_CALENDAR =
   getResources().getDrawable(R.drawable.nav_calendario_24dp);
231.         if(estado==0){
232.             Img_ModoTrabajo_home.setImageDrawable(IC_CALENDAR);
233.         }else{
234.             Img_ModoTrabajo_home.setImageDrawable(IC_TRABAJO);
235.             Tv_Intervalo.setText(null);
236.         }
237.     }
238.
239.     public void cambiarEstacionAno_home(int estado){
240.         Drawable IC_INVIERNO =
   getResources().getDrawable(R.drawable.ic_modo_invierno_24dp);
241.         Drawable IC_VERANO =
   getResources().getDrawable(R.drawable.ic_modo_verano_24dp);
242.
243.         if(estado==0){
244.             Img_EstacionAno_home.setImageDrawable(IC_INVIERNO);
245.         }else {
246.             Img_EstacionAno_home.setImageDrawable(IC_VERANO);
247.         }
248.     }
249.
250.     public void cambiarModoEco_home(int estado){
251.         Drawable IC_ECO =
   getResources().getDrawable(R.drawable.ic_modo_eco_24dp);
252.         if(estado==0){
253.             Img_ModoEco_home.setImageDrawable(null);
254.         }else{
255.             Img_ModoEco_home.setImageDrawable(IC_ECO);
256.         }
257.     }
258.
259.     public void cambiarModoCasaVacaciones(int estado){
260.         Drawable IC_HOME =
   getResources().getDrawable(R.drawable.nav_home_24dp);
261.         Drawable IC_FUERA_HOME =
   getResources().getDrawable(R.drawable.ic_fuera_de_casa_24dp);
262.         Drawable IC_VACACIONES =
   getResources().getDrawable(R.drawable.ic_modo_vacaciones_24dp);
263.
264.         if(estado==0){
265.             Img_ModoCasaVacaciones_home.setImageDrawable(IC_HOME);
266.         }else if (estado == 1){
267.             Img_ModoCasaVacaciones_home.setImageDrawable(IC_FUERA_HOME);
268.         }else {
269.             Img_ModoCasaVacaciones_home.setImageDrawable(IC_VACACIONES);
270.         }
271.     }
```

```
272.
273.     public interface OnFragmentInteractionListener {
274.         void onFragmentInteraction_Home(String f_tempSP);
275.     }
276. }
```

8.4.5. CalendarioFragment

```
1.  package com.delac.delacsmart;
2.
3. import android.content.Context;
4. import android.content.SharedPreferences;
5. import android.os.Bundle;
6. import android.support.v4.app.Fragment;
7. import android.support.v7.app.AlertDialog;
8. import android.view.LayoutInflater;
9. import android.view.View;
10.  import android.view.ViewGroup;
11.  import android.widget.Button;
12.  import android.widget.CompoundButton;
13.  import android.widget.ImageButton;
14.  import android.widget.Switch;
15.  import android.widget.TextView;
16.
17.
18.
19.  public class CalendarioFragment extends Fragment implements
   View.OnClickListener {
20.
21.      private OnFragmentInteractionListener mListener;
22.
23.      public static final String PREFS_CALENDARIO = "VariablesCalendario";
24.      public static final String DIARIO_TEMP = "Diario_Temp";
25.      public static final String DIARIO_HORA = "Diario_Hora";
26.      public static final String DIARIO_MINS = "Diario_Mins";
27.      public static final String FESTIVO_TEMP = "Festivo_Temp";
28.      public static final String FESTIVO_HORA = "Festivo_Hora";
29.      public static final String FESTIVO_MINS = "Festivo_Mins";
30.      public static final String MODOS_DIAS = "Modos_dias";
31.
32.
33.      TextView Temp_D1;
34.      TextView Hora_D1;
35.      TextView Temp_D2;
36.      TextView Hora_D2;
37.      TextView Temp_D3;
38.      TextView Hora_D3;
39.      TextView Temp_D4;
40.      TextView Hora_D4;
41.      TextView Temp_D5;
42.      TextView Hora_D5;
43.      TextView Temp_D6;
44.      TextView Hora_D6;
45.      TextView Temp_F1;
46.      TextView Hora_F1;
47.      TextView Temp_F2;
48.      TextView Hora_F2;
49.      TextView Temp_F3;
50.      TextView Hora_F3;
51.      TextView Temp_F4;
52.      TextView Hora_F4;
53.      TextView Temp_F5;
54.      TextView Hora_F5;
55.      TextView Temp_F6;
56.      TextView Hora_F6;
57.
```

```
58.      Switch Switch_lunes;
59.      Switch Switch_martes;
60.      Switch Switch_miercoles;
61.      Switch Switch_jueves;
62.      Switch Switch_viernes;
63.      Switch Switch_sabado;
64.      Switch Switch_domingo;
65.
66.
67.
68.      int SetPoint = 15;
69.      int Hora_dialog;
70.      int Min_dialog;
71.      int modo = 0;
72.      int intervalo = 0;
73.     // int dia;
74.
75.      int long_array = 6;
76.      int[] diario_temp={21,18,21,18,22,17};
77.      int[] diario_hora={7,9,13,16,18,22};
78.      int[] diario_mins={30,0,0,0,0,0};
79.      int[] festivo_temp={21,18,21,18,22,17};
80.      int[] festivo_hora={10,12,14,16,18,22};
81.      int[] festivo_mins={0,0,0,0,0,0};
82.
83.      int[] dias_modo = {0,0,0,0,0,1,1};
84.      Boolean[] setear = {false,false,false,false,false,true,true};
85.
86.      String titulo;
87.      String setpoint_dialogo;
88.      String sHora_dialog;
89.      String sMin_dialog;
90.
91.      public CalendarioFragment() {
92.      }
93.
94.      @Override
95.      public void onCreate(Bundle savedInstanceState) {
96.          super.onCreate(savedInstanceState);
97.      }
98.
99.      @Override
100.     public View onCreateView(LayoutInflater inflater, ViewGroup
   container, Bundle savedInstanceState) {
101.         // Inflate the layout for this fragment
102.         View view = inflater.inflate(R.layout.fragment_calendario,
   container, false);
103.         Temp_D1 = (TextView) view.findViewById(R.id.temp_D1);
104.         Hora_D1 = (TextView) view.findViewById(R.id.hora_D1);
105.         Temp_D2 = (TextView) view.findViewById(R.id.temp_D2);
106.         Hora_D2 = (TextView) view.findViewById(R.id.hora_D2);
107.         Temp_D3 = (TextView) view.findViewById(R.id.temp_D3);
108.         Hora_D3 = (TextView) view.findViewById(R.id.hora_D3);
109.         Temp_D4 = (TextView) view.findViewById(R.id.temp_D4);
110.         Hora_D4 = (TextView) view.findViewById(R.id.hora_D4);
111.         Temp_D5 = (TextView) view.findViewById(R.id.temp_D5);
112.         Hora_D5 = (TextView) view.findViewById(R.id.hora_D5);
113.         Temp_D6 = (TextView) view.findViewById(R.id.temp_D6);
114.         Hora_D6 = (TextView) view.findViewById(R.id.hora_D6);
115.         Temp_F1 = (TextView) view.findViewById(R.id.temp_F1);
116.         Hora_F1 = (TextView) view.findViewById(R.id.hora_F1);
```

```
117.          Temp_F2 = (TextView) view.findViewById(R.id.temp_F2);
118.          Hora_F2 = (TextView) view.findViewById(R.id.hora_F2);
119.          Temp_F3 = (TextView) view.findViewById(R.id.temp_F3);
120.          Hora_F3 = (TextView) view.findViewById(R.id.hora_F3);
121.          Temp_F4 = (TextView) view.findViewById(R.id.temp_F4);
122.          Hora_F4 = (TextView) view.findViewById(R.id.hora_F4);
123.          Temp_F5 = (TextView) view.findViewById(R.id.temp_F5);
124.          Hora_F5 = (TextView) view.findViewById(R.id.hora_F5);
125.          Temp_F6 = (TextView) view.findViewById(R.id.temp_F6);
126.          Hora_F6 = (TextView) view.findViewById(R.id.hora_F6);
127.          Switch_lunes = (Switch) view.findViewById(R.id.switch_lunes);
128.          Switch_martes = (Switch) view.findViewById(R.id.switch_martes);
129.          Switch_miercoles = (Switch)
   view.findViewById(R.id.switch_miercoles);
130.          Switch_jueves = (Switch) view.findViewById(R.id.switch_jueves);
131.          Switch_viernes = (Switch) view.findViewById(R.id.switch_viernes);
132.          Switch_sabado = (Switch) view.findViewById(R.id.switch_sabado);
133.          Switch_domingo = (Switch) view.findViewById(R.id.switch_domingo);
134.
135.          return view;
136.      }
137.
138.      @Override
139.      public void onAttach(Context context) {
140.          super.onAttach(context);
141.          if (context instanceof
   CalendarioFragment.OnFragmentInteractionListener) {
142.              mListener =
   (CalendarioFragment.OnFragmentInteractionListener) context;
143.          } else {
144.              throw new RuntimeException(context.toString() + " must
   implement OnFragmentInteractionListener");
145.          }
146.      }
147.
148.      @Override
149.      public void onDetach() {
150.          super.onDetach();
151.          mListener = null;
152.      }
153.
154.      @Override
155.      public void onResume(){
156.          super.onResume();
157.
158.          SharedPreferences pref_calendario =
   this.getActivity().getSharedPreferences(PREFS_CALENDARIO,Context.MODE_PRIVA
   TE);
159.
160.          //Array int
161.          for(int i=0; i<long_array;i++){
162.              diario_temp[i] = pref_calendario.getInt(DIARIO_TEMP + "_" +
   i, diario_temp[i]);
163.              diario_hora[i] = pref_calendario.getInt(DIARIO_HORA + "_" +
   i, diario_hora[i]);
164.              diario_mins[i] = pref_calendario.getInt(DIARIO_MINS + "_" +
   i, diario_mins[i]);
165.
166.              festivo_temp[i] = pref_calendario.getInt(FESTIVO_TEMP + "_" +
   i, festivo_temp[i]);
```

```
167.             festivo_hora[i] = pref_calendario.getInt(FESTIVO_HORA + "_" +
   i, festivo_hora[i]);
168.             festivo_mins[i] = pref_calendario.getInt(FESTIVO_MINS + "_" +
   i, festivo_mins[i]);
169.         }
170.         for (int j=0;j<7;j++){
171.             dias_modo[j] = pref_calendario.getInt(MODOS_DIAS + "_" + j,
   dias_modo[j]);
172.         }
173.
174.         celda_presionada();
175.         cambiarLunes();
176.         cambiarMartes();
177.         cambiarMiercoles();
178.         cambiarJueves();
179.         cambiarViernes();
180.         cambiarSabado();
181.         cambiarDomingo();
182.
183.         setearSwitches();
184.
185.         for(int k=0; k<2;k++){
186.             for(int l=0;l<long_array;l++){
187.                 if(k==0) {
188.                     escribir_celda(k, l,
   diario_temp[l],diario_hora[l],diario_mins[l]);
189.                 }else{
190.
   escribir_celda(k,l,festivo_temp[l],festivo_hora[l],festivo_mins[l]);
191.                 }
192.             }
193.         }
194.     }
195.
196.     @Override
197.     public void onPause(){
198.         super.onPause();
199.
200.         SharedPreferences.Editor editor =
   this.getActivity().getSharedPreferences(PREFS_CALENDARIO,Context.MODE_PRIVA
   TE).edit();
201.
202.         //Array int
203.         for(int i=0; i<long_array;i++){
204.             editor.putInt(DIARIO_TEMP + "_" + i, diario_temp[i]);
205.             editor.putInt(DIARIO_HORA + "_" + i, diario_hora[i]);
206.             editor.putInt(DIARIO_MINS + "_" + i, diario_mins[i]);
207.
208.             editor.putInt(FESTIVO_TEMP + "_" + i, festivo_temp[i]);
209.             editor.putInt(FESTIVO_HORA + "_" + i, festivo_hora[i]);
210.             editor.putInt(FESTIVO_MINS + "_" + i, festivo_mins[i]);
211.         }
212.         for (int j=0; j<7 ; j++){
213.             editor.putInt(MODOS_DIAS + "_" + j, dias_modo[j]);
214.         }
215.         editor.commit();
216.     }
217.
218.
219.     public void mostrarInputDialog(){
```

```
220.         LayoutInflater layoutInflater =
   LayoutInflater.from(getContext());
221.         View promptView =
   layoutInflater.inflate(R.layout.dialogo_intervalos, null);
222.         final AlertDialog.Builder alertDialogBuilder = new
   AlertDialog.Builder(getContext());
223.         alertDialogBuilder.setView(promptView);
224.         final AlertDialog alert = alertDialogBuilder.create();
225.
226.         final TextView Titulo_dialog = (TextView)
   promptView.findViewById(R.id.titulo_dialog);
227.         final TextView TempSP_dialog = (TextView)
   promptView.findViewById(R.id.tempSP_dialog);
228.         final TextView Tv_hora_dialog = (TextView)
   promptView.findViewById(R.id.tv_hora_dialog);
229.         final TextView Tv_minutos_dialog = (TextView)
   promptView.findViewById(R.id.tv_minutos_dialog);
230.         final ImageButton BotonUP_SP_dialog = (ImageButton)
   promptView.findViewById(R.id.botonUP_sp_dialog);
231.         final ImageButton BotonDOWN_SP_dialog = (ImageButton)
   promptView.findViewById(R.id.botonDOWN_sp_dialog);
232.         final ImageButton BotonUP_hora_dialog = (ImageButton)
   promptView.findViewById(R.id.botonUP_hora_dialog);
233.         final ImageButton BotonDOWN_hora_dialog = (ImageButton)
   promptView.findViewById(R.id.botonDOWN_hora_dialog);
234.         final ImageButton BotonUP_min_dialog = (ImageButton)
   promptView.findViewById(R.id.botonUP_min_dialog);
235.         final ImageButton BotonDOWN_min_dialog = (ImageButton)
   promptView.findViewById(R.id.botonDOWN_min_dialog);
236.         final Button Boton_aceptar = (Button)
   promptView.findViewById(R.id.aceptar_boton);
237.         final Button Boton_cancelar = (Button)
   promptView.findViewById(R.id.cancelar_boton);
238.
239.         titulo = "Intervalo ";
240.
241.         switch (intervalo){
242.             case 0:
243.                 titulo +=  "1 de ";
244.                 break;
245.             case 1:
246.                 titulo +=  "2 de ";
247.                 break;
248.             case 2:
249.                 titulo +=  "3 de ";
250.                 break;
251.             case 3:
252.                 titulo +=  "4 de ";
253.                 break;
254.             case 4:
255.                 titulo +=  "5 de ";
256.                 break;
257.             case 5:
258.                 titulo +=  "6 de ";
259.                 break;
260.         }
261.
262.         switch (modo){
263.             case 0:
264.                 titulo +=  "DIARIO";
265.                 break;
```

```
266.                case 1:
267.                    titulo +=  "FESTIVO";
268.                    break;
269.            }
270.
271.            if(modo == 0){
272.                SetPoint = diario_temp[intervalo];
273.                Hora_dialog = diario_hora[intervalo];
274.                Min_dialog = diario_mins[intervalo];
275.            }else {
276.                SetPoint = festivo_temp[intervalo];
277.                Hora_dialog = festivo_hora[intervalo];
278.                Min_dialog = festivo_mins[intervalo];
279.            }
280.
281.            sHora_dialog = Integer.toString(Hora_dialog);
282.            sMin_dialog = Integer.toString(Min_dialog);
283.
284.            if(Hora_dialog<10){
285.                sHora_dialog = "0" + sHora_dialog;
286.            }
287.            if(Min_dialog<10){
288.                sMin_dialog = "0" + sMin_dialog;
289.            }
290.
291.            setpoint_dialogo = Integer.toString(SetPoint);
292.            setpoint_dialogo += "°C";
293.
294.            Titulo_dialog.setText(titulo);
295.            TempSP_dialog.setText(setpoint_dialogo);
296.            Tv_hora_dialog.setText(sHora_dialog);
297.            Tv_minutos_dialog.setText(sMin_dialog);
298.
299.            BotonUP_SP_dialog.setOnClickListener(new View.OnClickListener() {
300.                @Override
301.                public void onClick(View v) {
302.                    SetPoint++;
303.                    if (SetPoint > 40) {
304.                        SetPoint = 40;
305.                    }
306.                    setpoint_dialogo = Integer.toString(SetPoint);
307.                    setpoint_dialogo = setpoint_dialogo +"°C";
308.                    if(SetPoint<10){
309.                        setpoint_dialogo = "0" + setpoint_dialogo;
310.                    }
311.                    TempSP_dialog.setText(setpoint_dialogo);
312.                }
313.            });
314.
315.            BotonDOWN_SP_dialog.setOnClickListener(new View.OnClickListener()
    {
316.                @Override
317.                public void onClick(View v) {
318.                    SetPoint--;
319.                    if (SetPoint < 0) {
320.                        SetPoint = 0;
321.                    }
322.                    setpoint_dialogo = Integer.toString(SetPoint);
323.                    setpoint_dialogo = setpoint_dialogo +"°C";
324.                    if(SetPoint<10){
325.                        setpoint_dialogo = "0" + setpoint_dialogo;
```

```
326.                 }
327.                 TempSP_dialog.setText(setpoint_dialogo);
328.             }
329.         });
330.
331.         BotonUP_hora_dialog.setOnClickListener(new View.OnClickListener()
     {
332.             @Override
333.             public void onClick(View v) {
334.                 Hora_dialog++;
335.                 if(Hora_dialog>23){
336.                     Hora_dialog=23;
337.                 }
338.                 sHora_dialog = Integer.toString(Hora_dialog);
339.                 if(Hora_dialog<10){
340.                     sHora_dialog = "0" + sHora_dialog;
341.                 }
342.                 Tv_hora_dialog.setText(sHora_dialog);
343.             }
344.         });
345.
346.         BotonDOWN_hora_dialog.setOnClickListener(new
     View.OnClickListener() {
347.             @Override
348.             public void onClick(View v) {
349.                 Hora_dialog--;
350.                 if(Hora_dialog<0){
351.                     Hora_dialog=0;
352.                 }
353.                 sHora_dialog = Integer.toString(Hora_dialog);
354.                 if(Hora_dialog<10){
355.                     sHora_dialog = "0" + sHora_dialog;
356.                 }
357.                 Tv_hora_dialog.setText(sHora_dialog);
358.             }
359.         });
360.
361.         BotonUP_min_dialog.setOnClickListener(new View.OnClickListener()
     {
362.             @Override
363.             public void onClick(View v) {
364.                 Min_dialog++;
365.                 if(Min_dialog>59){
366.                     Min_dialog=59;
367.                 }
368.                 sMin_dialog = Integer.toString(Min_dialog);
369.                 if(Min_dialog<10){
370.                     sMin_dialog = "0" + sMin_dialog;
371.                 }
372.                 Tv_minutos_dialog.setText(sMin_dialog);
373.             }
374.         });
375.
376.         BotonDOWN_min_dialog.setOnClickListener(new
     View.OnClickListener() {
377.             @Override
378.             public void onClick(View v) {
379.                 Min_dialog--;
380.                 if(Min_dialog<0){
381.                     Min_dialog=0;
382.                 }
```

```
383.                 sMin_dialog = Integer.toString(Min_dialog);
384.                 if(Min_dialog<10){
385.                     sMin_dialog = "0" + sMin_dialog;
386.                 }
387.                 Tv_minutos_dialog.setText(sMin_dialog);
388.             }
389.         });
390.
391.         Boton_aceptar.setOnClickListener(new View.OnClickListener() {
392.             @Override
393.             public void onClick(View v) {
394.                 if(modo==0){
395.                     diario_temp[intervalo]=SetPoint;
396.
397.                     diario_hora[intervalo]=Hora_dialog;
398.                     diario_mins[intervalo]=Min_dialog;
399.                 }else {
400.                     festivo_temp[intervalo]=SetPoint;
401.
402.                     festivo_hora[intervalo]=Hora_dialog;
403.                     festivo_mins[intervalo]=Min_dialog;
404.                 }
405.
   escribir_celda(modo,intervalo,SetPoint,Hora_dialog,Min_dialog);
406.
   mListener.onFragmentInteracion_Calendario_Intervalos(diario_temp,diario_hor
   a,diario_mins,festivo_temp,festivo_hora,festivo_mins);
407.                 alert.dismiss();
408.             }
409.         });
410.
411.         Boton_cancelar.setOnClickListener(new View.OnClickListener() {
412.             @Override
413.             public void onClick(View v) {
414.                 alert.dismiss();
415.             }
416.         });
417.         alert.show();
418.     }
419.
420.
421.     public void celda_presionada(){
422.         Temp_D1.setOnClickListener(this);
423.         Hora_D1.setOnClickListener(this);
424.         Temp_D2.setOnClickListener(this);
425.         Hora_D2.setOnClickListener(this);
426.         Temp_D3.setOnClickListener(this);
427.         Hora_D3.setOnClickListener(this);
428.         Temp_D4.setOnClickListener(this);
429.         Hora_D4.setOnClickListener(this);
430.         Temp_D5.setOnClickListener(this);
431.         Hora_D5.setOnClickListener(this);
432.         Temp_D6.setOnClickListener(this);
433.         Hora_D6.setOnClickListener(this);
434.
435.         Temp_F1.setOnClickListener(this);
436.         Hora_F1.setOnClickListener(this);
437.         Temp_F2.setOnClickListener(this);
438.         Hora_F2.setOnClickListener(this);
439.         Temp_F3.setOnClickListener(this);
440.         Hora_F3.setOnClickListener(this);
```

```
441.         Temp_F4.setOnClickListener(this);
442.         Hora_F4.setOnClickListener(this);
443.         Temp_F5.setOnClickListener(this);
444.         Hora_F5.setOnClickListener(this);
445.         Temp_F6.setOnClickListener(this);
446.         Hora_F6.setOnClickListener(this);
447.     }
448.
449.     @Override
450.     public void onClick(View v){
451.         switch (v.getId()){
452.             //DIARIO
453.             case R.id.temp_D1:
454.                 modo = 0;
455.                 intervalo = 0;
456.                 mostrarInputDialog();
457.                 break;
458.
459.             case R.id.hora_D1:
460.                 modo = 0;
461.                 intervalo = 0;
462.                 mostrarInputDialog();
463.                 break;
464.
465.             case R.id.temp_D2:
466.                 modo = 0;
467.                 intervalo = 1;
468.                 mostrarInputDialog();
469.                 break;
470.
471.             case R.id.hora_D2:
472.                 modo = 0;
473.                 intervalo = 1;
474.                 mostrarInputDialog();
475.                 break;
476.
477.             case R.id.temp_D3:
478.                 modo = 0;
479.                 intervalo = 2;
480.                 mostrarInputDialog();
481.                 break;
482.
483.             case R.id.hora_D3:
484.                 modo = 0;
485.                 intervalo = 2;
486.                 mostrarInputDialog();
487.                 break;
488.
489.             case R.id.temp_D4:
490.                 modo = 0;
491.                 intervalo = 3;
492.                 mostrarInputDialog();
493.                 break;
494.
495.             case R.id.hora_D4:
496.                 modo = 0;
497.                 intervalo = 3;
498.                 mostrarInputDialog();
499.                 break;
500.
501.             case R.id.temp_D5:
```

```
502.                 modo = 0;
503.                 intervalo = 4;
504.                 mostrarInputDialog();
505.                 break;
506.
507.             case R.id.hora_D5:
508.                 modo = 0;
509.                 intervalo = 4;
510.                 mostrarInputDialog();
511.                 break;
512.
513.                 case R.id.temp_D6:
514.                 modo = 0;
515.                 intervalo = 5;
516.                 mostrarInputDialog();
517.                 break;
518.
519.             case R.id.hora_D6:
520.                 modo = 0;
521.                 intervalo = 5;
522.                 mostrarInputDialog();
523.                 break;
524.
525.             //FESTIVO
526.             case R.id.temp_F1:
527.                 modo = 1;
528.                 intervalo = 0;
529.                 mostrarInputDialog();
530.                 break;
531.
532.             case R.id.hora_F1:
533.                 modo = 1;
534.                 intervalo = 0;
535.                 mostrarInputDialog();
536.                 break;
537.
538.             case R.id.temp_F2:
539.                 modo = 1;
540.                 intervalo = 1;
541.                 mostrarInputDialog();
542.                 break;
543.
544.             case R.id.hora_F2:
545.                 modo = 1;
546.                 intervalo = 1;
547.                 mostrarInputDialog();
548.                 break;
549.
550.             case R.id.temp_F3:
551.                 modo = 1;
552.                 intervalo = 2;
553.                 mostrarInputDialog();
554.                 break;
555.
556.             case R.id.hora_F3:
557.                 modo = 1;
558.                 intervalo = 2;
559.                 mostrarInputDialog();
560.                 break;
561.
562.             case R.id.temp_F4:
```

```
563.                 modo = 1;
564.                 intervalo = 3;
565.                 mostrarInputDialog();
566.                 break;
567.
568.             case R.id.hora_F4:
569.                 modo = 1;
570.                 intervalo = 3;
571.                 mostrarInputDialog();
572.                 break;
573.
574.             case R.id.temp_F5:
575.                 modo = 1;
576.                 intervalo = 4;
577.                 mostrarInputDialog();
578.                 break;
579.
580.             case R.id.hora_F5:
581.                 modo = 1;
582.                 intervalo = 4;
583.                 mostrarInputDialog();
584.                 break;
585.
586.             case R.id.temp_F6:
587.                 modo = 1;
588.                 intervalo = 5;
589.                 mostrarInputDialog();
590.                 break;
591.
592.             case R.id.hora_F6:
593.                 modo = 1;
594.                 intervalo = 5;
595.                 mostrarInputDialog();
596.                 break;
597.         }
598.     }
599.
600.     //Función para escribir en la celda la temperatura
601.     public void escribir_celda(int fil, int col, int SP, int h,int m){
602.         String setpoint = Integer.toString(SP);
603.         setpoint = setpoint +"°C";
604.
605.         String hora = Integer.toString(h);
606.         String min = Integer.toString(m);
607.
608.         if(h<10){
609.             hora = "0" + hora;
610.         }
611.         if(m<10){
612.             min = "0" + min;
613.         }
614.         String time = hora + ":" + min;
615.         if(h>11){
616.             time += "PM";
617.         }else{
618.             time+="AM";
619.         }
620.
621.         switch (fil){
622.             case 0:
623.                 switch (col){
```

```
624.                     case 0:
625.                         Temp_D1.setText(setpoint);
626.                         Hora_D1.setText(time);
627.                         break;
628.                     case 1:
629.                         Temp_D2.setText(setpoint);
630.                         Hora_D2.setText(time);
631.                         break;
632.                     case 2:
633.                         Temp_D3.setText(setpoint);
634.                         Hora_D3.setText(time);
635.                         break;
636.                     case 3:
637.                         Temp_D4.setText(setpoint);
638.                         Hora_D4.setText(time);
639.                         break;
640.                     case 4:
641.                         Temp_D5.setText(setpoint);
642.                         Hora_D5.setText(time);
643.                         break;
644.                     case 5:
645.                         Temp_D6.setText(setpoint);
646.                         Hora_D6.setText(time);
647.                         break;
648.                 }
649.                 break;
650.             case 1:
651.                 switch (col){
652.                     case 0:
653.                         Temp_F1.setText(setpoint);
654.                         Hora_F1.setText(time);
655.                         break;
656.                     case 1:
657.                         Temp_F2.setText(setpoint);
658.                         Hora_F2.setText(time);
659.                         break;
660.                     case 2:
661.                         Temp_F3.setText(setpoint);
662.                         Hora_F3.setText(time);
663.                         break;
664.                     case 3:
665.                         Temp_F4.setText(setpoint);
666.                         Hora_F4.setText(time);
667.                         break;
668.                     case 4:
669.                         Temp_F5.setText(setpoint);
670.                         Hora_F5.setText(time);
671.                         break;
672.                     case 5:
673.                         Temp_F6.setText(setpoint);
674.                         Hora_F6.setText(time);
675.                         break;
676.                 }
677.
678.                 break;
679.
680.         }
681.
682.     }
683.
684.     public void cambiarLunes(){
```

```
685.         Switch_lunes.setOnCheckedChangeListener(new
   CompoundButton.OnCheckedChangeListener() {
686.
687.             public void onCheckedChanged(CompoundButton buttonView,
   boolean isChecked) {
688.                 if(Switch_lunes.isChecked()){
689.                     dias_modo[0]=1;
690.
691.                 }else{
692.                     dias_modo[0]=0;
693.                 }
694.
   mListener.onFragmentInteraction_Calendario_ModosDias(dias_modo);
695.             }
696.         });
697.     }
698.     public void cambiarMartes(){
699.         Switch_martes.setOnCheckedChangeListener(new
   CompoundButton.OnCheckedChangeListener() {
700.
701.             public void onCheckedChanged(CompoundButton buttonView,
   boolean isChecked) {
702.                 if(Switch_martes.isChecked()){
703.                     dias_modo[1]=1;
704.
705.                 }else{
706.                     dias_modo[1]=0;
707.                 }
708.
   mListener.onFragmentInteraction_Calendario_ModosDias(dias_modo);
709.             }
710.         });
711.     }
712.     public void cambiarMiercoles(){
713.         Switch_miercoles.setOnCheckedChangeListener(new
   CompoundButton.OnCheckedChangeListener() {
714.
715.             public void onCheckedChanged(CompoundButton buttonView,
   boolean isChecked) {
716.                 if(Switch_miercoles.isChecked()){
717.                     dias_modo[2]=1;
718.
719.                 }else{
720.                     dias_modo[2]=0;
721.                 }
722.
   mListener.onFragmentInteraction_Calendario_ModosDias(dias_modo);
723.             }
724.         });
725.     }
726.     public void cambiarJueves(){
727.         Switch_jueves.setOnCheckedChangeListener(new
   CompoundButton.OnCheckedChangeListener() {
728.
729.             public void onCheckedChanged(CompoundButton buttonView,
   boolean isChecked) {
730.                 if(Switch_jueves.isChecked()){
731.                     dias_modo[3]=1;
732.
733.                 }else{
734.                     dias_modo[3]=0;
```

```
735.                 }
736.
   mListener.onFragmentInteraction_Calendario_ModosDias(dias_modo);
737.             }
738.         });
739.     }
740.     public void cambiarViernes(){
741.         Switch_viernes.setOnCheckedChangeListener(new
   CompoundButton.OnCheckedChangeListener() {
742.
743.             public void onCheckedChanged(CompoundButton buttonView,
   boolean isChecked) {
744.                 if(Switch_viernes.isChecked()){
745.                     dias_modo[4]=1;
746.
747.                 }else{
748.                     dias_modo[4]=0;
749.                 }
750.
   mListener.onFragmentInteraction_Calendario_ModosDias(dias_modo);
751.             }
752.         });
753.     }
754.     public void cambiarSabado(){
755.         Switch_sabado.setOnCheckedChangeListener(new
   CompoundButton.OnCheckedChangeListener() {
756.
757.             public void onCheckedChanged(CompoundButton buttonView,
   boolean isChecked) {
758.                 if(Switch_sabado.isChecked()){
759.                     dias_modo[5]=1;
760.
761.                 }else{
762.                     dias_modo[5]=0;
763.                 }
764.
   mListener.onFragmentInteraction_Calendario_ModosDias(dias_modo);
765.             }
766.         });
767.     }
768.     public void cambiarDomingo(){
769.         Switch_domingo.setOnCheckedChangeListener(new
   CompoundButton.OnCheckedChangeListener() {
770.
771.             public void onCheckedChanged(CompoundButton buttonView,
   boolean isChecked) {
772.                 if(Switch_domingo.isChecked()){
773.                     dias_modo[6]=1;
774.
775.                 }else{
776.                     dias_modo[6]=0;
777.                 }
778.
   mListener.onFragmentInteraction_Calendario_ModosDias(dias_modo);
779.             }
780.         });
781.     }
782.
783.     public void setearSwitches(){
784.         for(int i=0;i<7;i++){
785.             if(dias_modo[i]==1){
```

```
786.                 setear[i]= true;
787.             }else {
788.                 setear[i]=false;
789.             }
790.         }
791.         Switch_lunes.setChecked(setear[0]);
792.         Switch_martes.setChecked(setear[1]);
793.         Switch_miercoles.setChecked(setear[2]);
794.         Switch_jueves.setChecked(setear[3]);
795.         Switch_viernes.setChecked(setear[4]);
796.         Switch_sabado.setChecked(setear[5]);
797.         Switch_domingo.setChecked(setear[6]);
798.     }
799.
800.
801.     public interface OnFragmentInteractionListener {
802.
803.         void onFragmentInteraction_Calendario_ModosDias(int[] Dias_Modo);
804.         void onFragmentInteracion_Calendario_Intervalos(int[] DiarioTemp,
    int[] DiarioHora, int[] DiarioMin, int[] FestivoTemp, int[] FestivoHora,
    int[] FestivoMin);
805.
806.
807.
808.     }
809. }
```

8.4.6. EstadisticasFragment

```
1. package com.delac.delacsmart;
2.
3. import android.content.Context;
4. import android.net.Uri;
5. import android.os.Bundle;
6. import android.support.v4.app.Fragment;
7. import android.view.LayoutInflater;
8. import android.view.View;
9. import android.view.ViewGroup;
10.  import android.widget.TextView;
11.
12.
13.  public class EstadisticasFragment extends Fragment {
14.
15.      private OnFragmentInteractionListener mListener;
16.
17.      TextView TempMedia;
18.      TextView TempMin;
19.      TextView TempMax;
20.      TextView Horometro;
21.
22.      public EstadisticasFragment() {
23.          // Required empty public constructor
24.      }
25.
26.      @Override
27.      public View onCreateView(LayoutInflater inflater, ViewGroup
   container, Bundle savedInstanceState) {
28.          // Inflate the layout for this fragment
29.          View view = inflater.inflate(R.layout.fragment_estadisticas,
   container, false);
30.          TempMedia = (TextView) view.findViewById(R.id.tempMedia);
31.          TempMin = (TextView) view.findViewById(R.id.tempMin);
32.          TempMax = (TextView) view.findViewById(R.id.tempMax);
33.          Horometro = (TextView) view.findViewById(R.id.horometro);
34.
35.          return view;
36.      }
37.
38.      @Override
39.      public void onAttach(Context context) {
40.          super.onAttach(context);
41.          if (context instanceof OnFragmentInteractionListener) {
42.              mListener = (OnFragmentInteractionListener) context;
43.          } else {
44.              throw new RuntimeException(context.toString() + " must
   implement OnFragmentInteractionListener");
45.          }
46.      }
47.
48.      @Override
49.      public void onDetach() {
50.          super.onDetach();
51.          mListener = null;
52.      }
53.
54.      public void cambiarTempMin(String tempMin){
55.         TempMin.setText(tempMin);
```

```
56.      }
57.
58.      public void cambiarTempMax(String tempMax){
59.          TempMax.setText(tempMax);
60.      }
61.
62.      public void cambiarTempMedia(String tempMedia){
63.          TempMedia.setText(tempMedia);
64.      }
65.
66.      public void cambiarHorametro(String horametro){
67.          Horometro.setText(horametro);
68.      }
69.
70.
71.      public interface OnFragmentInteractionListener {
72.          // TODO: Update argument type and name
73.          void onFragmentInteraction_Estadisticas(Uri uri);
74.      }
75.  }
```

8.4.7. AjustesFragment

```
1. package com.delac.delacsmart;
2.
3. import android.content.Context;
4. import android.content.DialogInterface;
5. import android.graphics.drawable.Drawable;
6. import android.os.Bundle;
7. import android.support.v4.app.Fragment;
8. import android.support.v7.app.AlertDialog;
9. import android.view.KeyEvent;
10.  import android.view.LayoutInflater;
11.  import android.view.View;
12.  import android.view.ViewGroup;
13.  import android.widget.CompoundButton;
14.  import android.widget.EditText;
15.  import android.widget.ImageView;
16.  import android.widget.Switch;
17.  import android.widget.TableLayout;
18.  import android.widget.TableRow;
19.  import android.widget.TextView;
20.
21.  public class AjustesFragment extends Fragment {
22.
23.      private OnFragmentInteractionListener mListener;
24.
25.      Switch SwitchEstacionAno;
26.      Switch SwitchModoTrabajo;
27.      Switch SwitchCasa;
28.      Switch SwitchModoVacaciones;
29.
30.      EditText TempEcoMode;
31.      EditText TempDelta;
32.      EditText TiempoMaxEnviarSensor;
33.      EditText TiempoRastreo;
34.      EditText TiempoMaxEnviarActuador;
35.
36.      TextView TVFecha;
37.
38.      ImageView ImgFecha;
39.
40.      TableLayout TLFecha;
41.
42.      TableRow RowCasa;
43.      TableRow RowVacaciones;
44.
45.      int estacionAno = 0; //0 es invierno, 1 es verano
46.      int modoTrabajo = 0; //0 es automatico, 1 manual
47.      int modoCasa = 0; //0 es casa, 1 es fuera de casa
48.      int modoVacaciones = 0; //0 es OFF, 1 es ON
49.
50.      String tempEco = "15";
51.      String tempDelta ="1";
52.      String tiempoMaxEnviar_Sensor = "60";
53.      String tiempoRastreo = "2";
54.      String tiempoMaxEnviar_Actuador = "60";
55.      String old_tempEco = "15";
56.      String old_tempDelta ="1";
57.      String old_tiempoMaxEnviar_Sensor = "60";
58.      String old_tiempoRastreo = "2";
```

```
59.      String old_tiempoMaxEnviar_Actuador = "60";
60.
61.      String fecha;
62.      String hora;
63.
64.
65.      public AjustesFragment (){
66.      }
67.
68.      @Override
69.      public void onCreate(Bundle savedInstanceState) {
70.          super.onCreate(savedInstanceState);
71.      }
72.
73.      @Override
74.      public View onCreateView(LayoutInflater inflater, ViewGroup
   container, Bundle savedInstanceState) {
75.          // Inflate the layout for this fragment
76.          View view = inflater.inflate(R.layout.fragment_ajustes,
   container, false);
77.          SwitchEstacionAno = (Switch)
   view.findViewById(R.id.switch_EstacionAno);
78.          SwitchModoTrabajo = (Switch)
   view.findViewById(R.id.switchModoTrabajo);
79.          SwitchCasa = (Switch) view.findViewById(R.id.switchCasa);
80.          SwitchModoVacaciones = (Switch)
   view.findViewById(R.id.switchModoVacaciones);
81.          TempEcoMode = (EditText) view.findViewById(R.id.tempEcoMode);
82.          TempDelta = (EditText) view.findViewById(R.id.tempDelta);
83.          TiempoMaxEnviarSensor = (EditText)
   view.findViewById(R.id.tiempoMaxEnviarSensor);
84.          TiempoRastreo = (EditText) view.findViewById(R.id.tiempoRastreo);
85.          TiempoMaxEnviarActuador = (EditText)
   view.findViewById(R.id.tiempoMaxEnviarActuador);
86.          TLFecha = (TableLayout) view.findViewById(R.id.tlFecha);
87.          RowCasa = (TableRow) view.findViewById(R.id.rowCasa);
88.          RowVacaciones = (TableRow) view.findViewById(R.id.rowVacaciones);
89.          TVFecha = (TextView) view.findViewById(R.id.tvFecha);
90.          ImgFecha = (ImageView) view.findViewById(R.id.imgFecha);
91.
92.          cambiarEstacionAno();
93.          cambiarModoTrabajo();
94.          cambiarModoEco();
95.          cambiarCasa();
96.          cambiarModoVacaciones();
97.          cambiarTempDelta();
98.          cambiarTiempoMaxEnviarSensor();
99.          cambiarTiempoRastreo();
100.         cambiarTiempoMaxEnviarActuador();
101.
102.         return view;
103.     }
104.
105.     @Override
106.     public void onAttach(Context context) {
107.         super.onAttach(context);
108.         if (context instanceof OnFragmentInteractionListener) {
109.             mListener = (OnFragmentInteractionListener) context;
110.         } else {
111.             throw new RuntimeException(context.toString() + " must
   implement OnFragmentInteractionListener");
```

```
112.         }
113.     }
114.
115.     @Override
116.     public void onDetach() {
117.         super.onDetach();
118.         mListener = null;
119.     }
120.
121.     public void cambiarEstacionAno(){
122.         SwitchEstacionAno.setOnCheckedChangeListener(new
   CompoundButton.OnCheckedChangeListener() {
123.             @Override
124.             public void onCheckedChanged(CompoundButton buttonView,
   boolean isChecked) {
125.                 if(SwitchEstacionAno.isChecked()){
126.                     estacionAno = 1;
127.
   mListener.onFragmentInteraction_Ajustes_EstacionAno(estacionAno);
128.                 }else{
129.                     estacionAno = 0;
130.
   mListener.onFragmentInteraction_Ajustes_EstacionAno(estacionAno);
131.                 }
132.             }
133.         });
134.     }
135.
136.     public void cambiarModoTrabajo(){
137.         SwitchModoTrabajo.setOnCheckedChangeListener(new
   CompoundButton.OnCheckedChangeListener() {
138.
139.             public void onCheckedChanged(CompoundButton buttonView,
   boolean isChecked) {
140.                 if(SwitchModoTrabajo.isChecked()){
141.                     modoTrabajo=1;
142.
143.                 }else{
144.                     modoTrabajo=0;
145.                 }
146.
   mListener.onFragmentInteraction_Ajustes_ModoTrabajo(modoTrabajo);
147.             }
148.         });
149.     }
150.
151.     public void cambiarCasa(){
152.         SwitchCasa.setOnCheckedChangeListener(new
   CompoundButton.OnCheckedChangeListener() {
153.
154.             public void onCheckedChanged(CompoundButton buttonView,
   boolean isChecked) {
155.                 if(SwitchCasa.isChecked()){
156.                     modoCasa=1;
157.                     RowVacaciones.setVisibility(View.GONE);
158.                     TLFecha.setVisibility(View.VISIBLE);
159.                 }else{
160.                     modoCasa=0;
161.                     RowVacaciones.setVisibility(View.VISIBLE);
162.                     TLFecha.setVisibility(View.GONE);
163.                 }
```

```
164.                 mListener.onFragmentInteraction_Ajustes_Casa(modoCasa);
165.             }
166.         });
167.     }
168.
169.     public void cambiar_Hora(int Hora, int Minuto){
170.         Drawable CASA =
    getResources().getDrawable(R.drawable.ic_fuera_de_casa_24dp);
171.         hora = Integer.toString(Hora)+" : "+Integer.toString(Minuto);
172.         TVFecha.setText(hora);
173.         ImgFecha.setImageDrawable(CASA);
174.     }
175.
176.     public void cambiarModoEco(){
177.         TempEcoMode.setOnKeyListener(new View.OnKeyListener() {
178.             @Override
179.             public boolean onKey(View view, int keyCode, KeyEvent
    keyEvent) {
180.
    if((keyEvent.getAction()==KeyEvent.ACTION_DOWN)&&(keyCode==KeyEvent.KEYCODE
    _ENTER)){
181.                     tempEco = TempEcoMode.getText().toString();
182.                     int i = Integer.parseInt(tempEco);
183.                     if(i<0 || i>40){
184.                         AlertDialog.Builder dialogo = new
    AlertDialog.Builder(getContext());
185.                         dialogo.setTitle("Valor erroneo");
186.                         dialogo.setMessage("Introduzca un valor
    comprendido entre 0 y 40");
187.                         dialogo.setPositiveButton("Aceptar", new
    DialogInterface.OnClickListener() {
188.                             @Override
189.                             public void onClick(DialogInterface
    dialogInterface, int i) {
190.                                 TempEcoMode.setText(old_tempEco);
191.                             }
192.                         });
193.                         dialogo.show();
194.                     }else{
195.                         old_tempEco=tempEco;
196.
    mListener.onFragmentInteraction_Ajustes_TempEcoMode(tempEco);
197.                     }
198.                     return true;
199.                 }
200.                 return false;
201.             }
202.         });
203.     }
204.
205.     public void cambiarModoVacaciones(){
206.         SwitchModoVacaciones.setOnCheckedChangeListener(new
    CompoundButton.OnCheckedChangeListener() {
207.
208.             public void onCheckedChanged(CompoundButton buttonView,
    boolean isChecked) {
209.                 if(SwitchModoVacaciones.isChecked()){
210.                     modoVacaciones=1;
211.                     RowCasa.setVisibility(View.GONE);
212.                     TLFecha.setVisibility(View.VISIBLE);
213.                 }else{
```

```
214.                     modoVacaciones=0;
215.                     RowCasa.setVisibility(View.VISIBLE);
216.                     TLFecha.setVisibility(View.GONE);
217.                 }
218.
   mListener.onFragmentInteraction_Ajustes_ModoVacaciones(modoVacaciones);
219.             }
220.         });
221.     }
222.
223.     public void cambiar_Fecha(int year,int month, int day){
224.         Drawable VACACIONES =
   getResources().getDrawable(R.drawable.ic_modo_vacaciones_24dp);
225.         fecha = Integer.toString(day)+" / "+Integer.toString(month) + " /
   " +Integer.toString(year);
226.         TVFecha.setText(fecha);
227.         ImgFecha.setImageDrawable(VACACIONES);
228.     }
229.
230.     public void cambiarTempDelta(){
231.         TempDelta.setOnKeyListener(new View.OnKeyListener() {
232.             @Override
233.             public boolean onKey(View view, int keyCode, KeyEvent
   keyEvent) {
234.
   if((keyEvent.getAction()==KeyEvent.ACTION_DOWN)&&(keyCode==KeyEvent.KEYCODE
   _ENTER)){
235.                     tempDelta = TempDelta.getText().toString();
236.                     int i = Integer.parseInt(tempDelta);
237.                     if(i==1 || i== 2 || i==3){
238.                         old_tempDelta=tempDelta;
239.
   mListener.onFragmentInteraction_Ajustes_TempDelta(tempDelta);
240.                     }else{
241.                         AlertDialog.Builder dialogo = new
   AlertDialog.Builder(getContext());
242.                         dialogo.setTitle("Valor erroneo");
243.                         dialogo.setMessage("Introduzca un Δtª igual a
   1°C, 2°C o 3°C");
244.                         dialogo.setPositiveButton("Aceptar", new
   DialogInterface.OnClickListener() {
245.                             @Override
246.                             public void onClick(DialogInterface
   dialogInterface, int i) {
247.                                 TempDelta.setText(old_tempDelta);
248.                             }
249.                         });
250.                         dialogo.show();
251.                     }
252.                     return true;
253.                 }
254.                 return false;
255.             }
256.         });
257.     }
258.
259.     public void cambiarTiempoMaxEnviarSensor(){
260.         TiempoMaxEnviarSensor.setOnKeyListener(new View.OnKeyListener() {
261.             @Override
262.             public boolean onKey(View view, int keyCode, KeyEvent
   keyEvent) {
```

```
263.
   if((keyEvent.getAction()==KeyEvent.ACTION_DOWN)&&(keyCode==KeyEvent.KEYCODE
   _ENTER)){
264.                     tiempoMaxEnviar_Sensor =
   TiempoMaxEnviarSensor.getText().toString();
265.                     int i = Integer.parseInt(tiempoMaxEnviar_Sensor);
266.                     if(i<=0){
267.                         AlertDialog.Builder dialogo = new
   AlertDialog.Builder(getContext());
268.                         dialogo.setTitle("Valor erroneo");
269.                         dialogo.setMessage("Introduzca un valor positivo,
   mayor que 0s");
270.                         dialogo.setPositiveButton("Aceptar", new
   DialogInterface.OnClickListener() {
271.                             @Override
272.                             public void onClick(DialogInterface
   dialogInterface, int i) {
273.
   TiempoMaxEnviarSensor.setText(old_tiempoMaxEnviar_Sensor);
274.                             }
275.                         });
276.                         dialogo.show();
277.                     }else{
278.
   old_tiempoMaxEnviar_Sensor=tiempoMaxEnviar_Sensor;
279.                         i=i*1000;
280.                         tiempoMaxEnviar_Sensor= Integer.toString(i);
281.
   mListener.onFragmentInteraction_Ajustes_TiempoMaxEnviarSensor(tiempoMaxEnvi
   ar_Sensor);
282.                     }
283.                     return true;
284.                 }
285.                 return false;
286.             }
287.         });
288.     }
289.
290.     public void cambiarTiempoRastreo(){
291.         TiempoRastreo.setOnKeyListener(new View.OnKeyListener() {
292.             @Override
293.             public boolean onKey(View view, int keyCode, KeyEvent
   keyEvent) {
294.
   if((keyEvent.getAction()==KeyEvent.ACTION_DOWN)&&(keyCode==KeyEvent.KEYCODE
   _ENTER)){
295.                     tiempoRastreo = TiempoRastreo.getText().toString();
296.                     int i = Integer.parseInt(tiempoRastreo);
297.                     if(i<=0){
298.                         AlertDialog.Builder dialogo = new
   AlertDialog.Builder(getContext());
299.                         dialogo.setTitle("Valor erroneo");
300.                         dialogo.setMessage("Introduzca un valor positivo,
   mayor que 0s");
301.                         dialogo.setPositiveButton("Aceptar", new
   DialogInterface.OnClickListener() {
302.                             @Override
303.                             public void onClick(DialogInterface
   dialogInterface, int i) {
304.                                 TiempoRastreo.setText(old_tiempoRastreo);
305.                             }
```

```
306.                         });
307.                         dialogo.show();
308.                     }else{
309.                         old_tiempoRastreo=tiempoRastreo;
310.                         i=i*1000;
311.                         tiempoRastreo= Integer.toString(i);
312.
   mListener.onFragmentInteraction_Ajustes_TiempoRastreo(tiempoRastreo);
313.                     }
314.                     return true;
315.                 }
316.                 return false;
317.             }
318.         });
319.     }
320.
321.     public void cambiarTiempoMaxEnviarActuador(){
322.         TiempoMaxEnviarActuador.setOnKeyListener(new View.OnKeyListener()
   {
323.             @Override
324.             public boolean onKey(View view, int keyCode, KeyEvent
   keyEvent) {
325.
   if((keyEvent.getAction()==KeyEvent.ACTION_DOWN)&&(keyCode==KeyEvent.KEYCODE
   _ENTER)){
326.                     tiempoMaxEnviar_Actuador =
   TiempoMaxEnviarActuador.getText().toString();
327.                     int i = Integer.parseInt(tiempoMaxEnviar_Actuador);
328.                     if(i<=0){
329.                         AlertDialog.Builder dialogo = new
   AlertDialog.Builder(getContext());
330.                         dialogo.setTitle("Valor erroneo");
331.                         dialogo.setMessage("Introduzca un valor positivo,
   mayor que 0s");
332.                         dialogo.setPositiveButton("Aceptar", new
   DialogInterface.OnClickListener() {
333.                             @Override
334.                             public void onClick(DialogInterface
   dialogInterface, int i) {
335.
   TiempoMaxEnviarActuador.setText(old_tiempoMaxEnviar_Actuador);
336.
337.                             }
338.                         });
339.                         dialogo.show();
340.                     }else{
341.
   old_tiempoMaxEnviar_Actuador=tiempoMaxEnviar_Actuador;
342.                         i=i*1000;
343.                         tiempoMaxEnviar_Actuador= Integer.toString(i);
344.
   mListener.onFragmentInteraction_Ajustes_TiempoMaxEnviarActuador(tiempoMaxEn
   viar_Actuador);
345.                     }
346.                     return true;
347.                 }
348.                 return false;
349.             }
350.         });
351.     }
352.
```

```
353.
354.     public interface OnFragmentInteractionListener {
355.         // TODO: Update argument type and name
356.         void onFragmentInteraction_Ajustes_EstacionAno(int valor);
357.         void onFragmentInteraction_Ajustes_ModoTrabajo(int valor);
358.         void onFragmentInteraction_Ajustes_Casa(int valor);
359.         void onFragmentInteraction_Ajustes_ModoVacaciones(int valor);
360.         void onFragmentInteraction_Ajustes_TempEcoMode(String valor);
361.         void onFragmentInteraction_Ajustes_TempDelta(String valor);
362.         void onFragmentInteraction_Ajustes_TiempoMaxEnviarSensor(String
    valor);
363.         void onFragmentInteraction_Ajustes_TiempoRastreo(String valor);
364.         void onFragmentInteraction_Ajustes_TiempoMaxEnviarActuador(String
    valor);
365.     }
366. }
```

8.4.8. TimePickerClass

```
1. package com.delac.delacsmart;
2.
3. import android.app.Dialog;
4. import android.app.TimePickerDialog;
5. import android.content.Context;
6. import android.os.Bundle;
7. import android.support.annotation.NonNull;
8. import android.support.v4.app.DialogFragment;
9. import android.widget.TimePicker;
10.
11.  import java.util.Calendar;
12.
13.  public class TimePickerClass extends DialogFragment {
14.
15.      private TimePickerClass.OnTimeSetListener mListener;
16.
17.
18.      @Override
19.      @NonNull
20.      public Dialog onCreateDialog(Bundle savedInstanceState) {
21.
22.          Calendar fechaActual = Calendar.getInstance();
23.          int hora = fechaActual.get(Calendar.HOUR_OF_DAY);
24.          int minuto  = fechaActual.get(Calendar.MINUTE);
25.
26.          return  new TimePickerDialog(getActivity(),timeSetListener, hora,
   minuto, true);
27.      }
28.
29.
30.
31.      private TimePickerDialog.OnTimeSetListener timeSetListener =
32.              new TimePickerDialog.OnTimeSetListener() {
33.                  public void onTimeSet(TimePicker view, int hora, int min)
   {
34.                      //String Year = Integer.toString(view.getYear());
35.                      //String Month = Integer.toString(view.getMonth());
36.                      //String Day =
   Integer.toString(view.getDayOfMonth());
37.
   mListener.onFragmentInteraction_TimePicker(view.getHour(),view.getMinute())
   ;
38.
39.                  }
40.              };
41.
42.      @Override
43.      public void onAttach(Context context) {
44.          super.onAttach(context);
45.          if (context instanceof
   AjustesFragment.OnFragmentInteractionListener) {
46.              mListener = (TimePickerClass.OnTimeSetListener) context;
47.          } else {
48.              throw new RuntimeException(context.toString() + " must
   implement OnFragmentInteractionListener");
49.          }
50.      }
51.
```

```
52.      @Override
53.      public void onDetach() {
54.          super.onDetach();
55.          mListener = null;
56.      }
57.
58.      public interface OnTimeSetListener {
59.          // TODO: Update argument type and name
60.          void onFragmentInteraction_TimePicker(int hora, int minuto);
61.      }
62.  }
```

8.4.9. DatePickerClass

```
1. package com.delac.delacsmart;
2.
3. import android.app.DatePickerDialog;
4. import android.app.Dialog;
5. import android.content.Context;
6. import android.os.Bundle;
7. import android.support.annotation.NonNull;
8. import android.support.v4.app.DialogFragment;
9. import android.widget.DatePicker;
10.
11.  import java.util.Calendar;
12.
13.  public class DatePickerClass extends DialogFragment {
14.
15.      private DatePickerClass.OnDateSetListener mListener;
16.
17.
18.      @Override
19.      @NonNull
20.      public Dialog onCreateDialog(Bundle savedInstanceState) {
21.
22.          final Calendar c = Calendar.getInstance();
23.          int year = c.get(Calendar.YEAR);
24.          int month = c.get(Calendar.MONTH);
25.          int day = c.get(Calendar.DAY_OF_MONTH);
26.
27.          return new DatePickerDialog(getActivity(), dateSetListener, year,
    month, day);
28.      }
29.
30.      private DatePickerDialog.OnDateSetListener dateSetListener =
31.              new DatePickerDialog.OnDateSetListener() {
32.                  public void onDateSet(DatePicker view, int year, int
    month, int day) {
33.                      //String Year = Integer.toString(view.getYear());
34.                      //String Month = Integer.toString(view.getMonth());
35.                      //String Day =
    Integer.toString(view.getDayOfMonth());
36.
    mListener.onFragmentInteraction_DatePicker(view.getYear(),(view.getMonth()+
    1), view.getDayOfMonth());
37.
38.                  }
39.              };
40.
41.      @Override
42.      public void onAttach(Context context) {
43.          super.onAttach(context);
44.          if (context instanceof
    AjustesFragment.OnFragmentInteractionListener) {
45.              mListener = (DatePickerClass.OnDateSetListener) context;
46.          } else {
47.              throw new RuntimeException(context.toString() + " must
    implement OnFragmentInteractionListener");
48.          }
49.      }
50.
51.      @Override
```

```
52.      public void onDetach() {
53.          super.onDetach();
54.          mListener = null;
55.      }
56.
57.      public interface OnDateSetListener {
58.          // TODO: Update argument type and name
59.          void onFragmentInteraction_DatePicker(int year, int month, int
   day);
60.      }
61.  }
```

8.4.10. activity_dispositivos.xml

```
1. <?xml version="1.0" encoding="utf-8"?>
2. <android.support.constraint.ConstraintLayout
   xmlns:android="http://schemas.android.com/apk/res/android"
3.     xmlns:app="http://schemas.android.com/apk/res-auto"
4.     xmlns:tools="http://schemas.android.com/tools"
5.     android:id="@+id/container"
6.     android:layout_width="match_parent"
7.     android:layout_height="match_parent"
8.     tools:context=".DispositivosActivity">
9.
10.      <FrameLayout
11.          android:id="@+id/fragment_container"
12.          android:layout_width="match_parent"
13.          android:layout_height="match_parent"
14.          android:layout_marginBottom="56dp"
15.          app:layout_constraintBottom_toTopOf="@+id/navigation"
16.          app:layout_constraintEnd_toEndOf="parent"
17.          app:layout_constraintStart_toStartOf="parent"
18.          app:layout_constraintTop_toTopOf="parent">
19.
20.      </FrameLayout>
21.
22.      <android.support.design.widget.BottomNavigationView
23.          android:id="@+id/navigation"
24.          android:layout_width="0dp"
25.          android:layout_height="wrap_content"
26.          android:background="?android:attr/windowBackground"
27.          app:itemBackground="@color/colorGris_claro"
28.          app:layout_constraintBottom_toBottomOf="parent"
29.          app:layout_constraintLeft_toLeftOf="parent"
30.          app:layout_constraintRight_toRightOf="parent"
31.          app:menu="@menu/navigation" />
32.
33.  </android.support.constraint.ConstraintLayout>
```

8.4.11. fragment_home.xml

```
1. <?xml version="1.0" encoding="utf-8"?>
2. <android.support.constraint.ConstraintLayout
   xmlns:android="http://schemas.android.com/apk/res/android"
3.     xmlns:app="http://schemas.android.com/apk/res-auto"
4.     xmlns:tools="http://schemas.android.com/tools"
5.     android:layout_width="match_parent"
6.     android:layout_height="match_parent"
7.     android:paddingBottom="2dp"
8.     tools:context=".HomeFragment">
9.
10.      <LinearLayout
11.          android:id="@+id/linearLayout3"
12.          android:layout_width="match_parent"
13.          android:layout_height="16dp"
14.          android:layout_marginStart="8dp"
15.          android:layout_marginEnd="8dp"
16.          android:gravity="right"
17.          android:orientation="horizontal"
18.          app:layout_constraintBottom_toBottomOf="parent"
19.          app:layout_constraintEnd_toEndOf="parent"
20.          app:layout_constraintStart_toStartOf="parent"
21.          app:layout_constraintTop_toTopOf="parent"
22.          app:layout_constraintVertical_bias="0.0">
23.
24.          <ImageView
25.              android:id="@+id/i_conexionMQTT"
26.              android:layout_width="16dp"
27.              android:layout_height="16dp" />
28.
29.          <TextView
30.              android:id="@+id/conexionMQTT"
31.              android:layout_width="wrap_content"
32.              android:layout_height="wrap_content"
33.              android:text="DESCONECTADO"
34.              android:textColor="#000000"
35.              android:textSize="12dp" />
36.
37.      </LinearLayout>
38.
39.      <TableLayout
40.          android:layout_width="wrap_content"
41.          android:layout_height="wrap_content"
42.          app:layout_constraintBottom_toBottomOf="@+id/constraintLayout"
43.          app:layout_constraintEnd_toEndOf="@+id/constraintLayout"
44.          app:layout_constraintHorizontal_bias="0.0"
45.          app:layout_constraintVertical_bias="0.0"
46.          app:layout_constraintStart_toStartOf="@+id/constraintLayout"
47.          app:layout_constraintTop_toTopOf="@+id/linearLayout3">
48.
49.          <TableRow
50.              android:layout_width="wrap_content"
51.              android:layout_height="wrap_content"
52.              android:layout_marginStart="8dp">
53.
54.              <TextView
55.                  android:id="@+id/tv_Dia_Hora"
56.                  android:layout_width="wrap_content"
57.                  android:layout_height="wrap_content"
```

```
58.                 android:textColor="#000000"
59.                 android:textSize="20dp" />
60.
61.         </TableRow>
62.
63.         <TableRow
64.             android:layout_width="wrap_content"
65.             android:layout_height="wrap_content">
66.
67.             <LinearLayout
68.                 android:layout_width="wrap_content"
69.                 android:layout_height="wrap_content"
70.                 android:orientation="horizontal">
71.
72.                 <ImageView
73.                     android:id="@+id/img_ModoTrabajo_home"
74.                     android:layout_width="30dp"
75.                     android:layout_height="30dp"
76.                      />
77.
78.                 <TextView
79.                     android:id="@+id/tv_Intervalo"
80.                     android:layout_width="wrap_content"
81.                     android:layout_height="wrap_content"
82.                     android:layout_gravity="center"
83.
84.                     android:textColor="#000000"
85.                     android:textSize="24dp" />
86.
87.             </LinearLayout>
88.
89.         </TableRow>
90.
91.
92.     </TableLayout>
93.
94.     <android.support.constraint.ConstraintLayout
95.         android:id="@+id/constraintLayout"
96.         android:layout_width="match_parent"
97.         android:layout_height="310dp"
98.
99.         android:layout_marginTop="32dp"
100.        app:layout_constraintEnd_toEndOf="parent"
101.        app:layout_constraintHorizontal_bias="0.0"
102.        app:layout_constraintStart_toStartOf="parent"
103.        app:layout_constraintTop_toBottomOf="@+id/linearLayout3">
104.
105.        <ProgressBar
106.
107.            android:id="@+id/sp_Progressbar"
108.            style="?android:attr/progressBarStyleHorizontal"
109.            android:layout_width="310dp"
110.            android:layout_height="310dp"
111.            android:layout_centerInParent="true"
112.            android:indeterminate="false"
113.            android:max="60"
114.            android:progress="15"
115.            android:progressDrawable="@drawable/pb_setpoint"
116.            android:secondaryProgress="40"
117.
118.            app:layout_constraintBottom_toBottomOf="parent"
```

```
119.             app:layout_constraintEnd_toEndOf="parent"
120.             app:layout_constraintStart_toStartOf="parent"
121.             app:layout_constraintTop_toTopOf="parent" />
122.
123.         <ProgressBar
124.             android:id="@+id/circularProgressbar"
125.             style="?android:attr/progressBarStyleHorizontal"
126.             android:layout_width="310dp"
127.             android:layout_height="300dp"
128.             android:layout_centerInParent="true"
129.             android:indeterminate="false"
130.             android:max="60"
131.             android:progress="23"
132.             android:progressDrawable="@drawable/pb_temperatura"
133.             app:layout_constraintBottom_toBottomOf="parent"
134.             app:layout_constraintEnd_toEndOf="parent"
135.             app:layout_constraintStart_toStartOf="parent"
136.             app:layout_constraintTop_toTopOf="parent" />
137.
138.
139.         <ImageView
140.             android:layout_width="70dp"
141.             android:layout_height="70dp"
142.             android:layout_centerInParent="true"
143.             android:layout_marginTop="8dp"
144.             android:background="@drawable/ic_termometro_rojo"
145.             app:layout_constraintBottom_toBottomOf="parent"
146.             app:layout_constraintEnd_toEndOf="parent"
147.             app:layout_constraintHorizontal_bias="0.498"
148.             app:layout_constraintStart_toStartOf="parent"
149.             app:layout_constraintTop_toTopOf="parent"
150.             app:layout_constraintVertical_bias="0.293" />
151.
152.         <TextView
153.             android:id="@+id/tempActual"
154.             android:layout_width="wrap_content"
155.             android:layout_height="wrap_content"
156.             android:layout_marginTop="8dp"
157.             android:text="0°C"
158.             android:textColor="#bb4d00"
159.             android:textSize="60dp"
160.             android:textStyle="bold"
161.
   app:layout_constraintBottom_toBottomOf="@+id/circularProgressbar"
162.             app:layout_constraintEnd_toEndOf="@+id/circularProgressbar"
163.             app:layout_constraintHorizontal_bias="0.505"
164.
   app:layout_constraintStart_toStartOf="@+id/circularProgressbar"
165.             app:layout_constraintTop_toTopOf="@+id/circularProgressbar"
166.             app:layout_constraintVertical_bias="0.631" />
167.
168.         <ImageView
169.             android:id="@+id/img_flecha_down_SP"
170.             android:layout_width="35dp"
171.             android:layout_height="35dp"
172.             android:layout_marginStart="8dp"
173.             android:layout_marginTop="12dp"
174.             android:layout_marginEnd="8dp"
175.             app:layout_constraintBottom_toBottomOf="@+id/tempSP"
176.             app:layout_constraintEnd_toStartOf="@+id/tempSP"
177.             app:layout_constraintHorizontal_bias="0.444"
```

```
178.             app:layout_constraintStart_toEndOf="@+id/botonDOWN_sp"
179.             app:layout_constraintTop_toTopOf="@+id/botonDOWN_sp" />
180.
181.         <ImageButton
182.             android:id="@+id/botonDOWN_sp"
183.             android:layout_width="60dp"
184.             android:layout_height="35dp"
185.             android:layout_marginTop="8dp"
186.             android:layout_marginEnd="8dp"
187.             android:background="@color/colorPrimaryDark"
188.             android:contentDescription="@null"
189.             app:layout_constraintBottom_toBottomOf="@+id/sp_Progressbar"
190.             app:layout_constraintEnd_toEndOf="@+id/sp_Progressbar"
191.             app:layout_constraintHorizontal_bias="0.049"
192.             app:layout_constraintStart_toStartOf="@+id/sp_Progressbar"
193.             app:layout_constraintTop_toTopOf="@+id/sp_Progressbar"
194.             app:layout_constraintVertical_bias="0.887"
195.
196.             app:srcCompat="@android:drawable/arrow_down_float" />
197.
198.         <TextView
199.             android:id="@+id/tempSP"
200.             android:layout_width="wrap_content"
201.             android:layout_height="wrap_content"
202.             android:layout_marginStart="8dp"
203.             android:layout_marginEnd="8dp"
204.             android:text="15°C"
205.             android:textColor="@color/colorPrimaryDark"
206.             android:textSize="45dp"
207.             android:textStyle="bold"
208.             app:layout_constraintBottom_toBottomOf="@+id/botonDOWN_sp"
209.             app:layout_constraintEnd_toStartOf="@+id/botonDOWN_sp"
210.             app:layout_constraintHorizontal_bias="0.505"
211.             app:layout_constraintStart_toEndOf="@+id/botonUP_sp"
212.             app:layout_constraintTop_toTopOf="@+id/botonDOWN_sp"
213.             app:layout_constraintVertical_bias="0.562" />
214.
215.         <ImageButton
216.             android:id="@+id/botonUP_sp"
217.             android:layout_width="60dp"
218.             android:layout_height="35dp"
219.             android:layout_marginStart="40dp"
220.             android:layout_marginTop="8dp"
221.             android:background="@color/colorPrimaryDark"
222.             android:contentDescription="@null"
223.             app:layout_constraintBottom_toBottomOf="@+id/sp_Progressbar"
224.             app:layout_constraintEnd_toEndOf="@+id/sp_Progressbar"
225.             app:layout_constraintHorizontal_bias="0.961"
226.             app:layout_constraintStart_toStartOf="@+id/sp_Progressbar"
227.             app:layout_constraintTop_toTopOf="@+id/sp_Progressbar"
228.             app:layout_constraintVertical_bias="0.887"
229.             app:srcCompat="@android:drawable/arrow_up_float" />
230.
231.         <ImageView
232.             android:id="@+id/img_flecha_up_SP"
233.             android:layout_width="35dp"
234.             android:layout_height="35dp"
235.             android:layout_marginStart="8dp"
236.             android:layout_marginTop="8dp"
237.             android:layout_marginEnd="8dp"
238.             android:layout_marginBottom="16dp"
```

```
239.             app:layout_constraintBottom_toBottomOf="@+id/botonUP_sp"
240.             app:layout_constraintEnd_toStartOf="@+id/botonUP_sp"
241.             app:layout_constraintHorizontal_bias="0.466"
242.             app:layout_constraintStart_toEndOf="@+id/tempSP"
243.             app:layout_constraintTop_toTopOf="@+id/tempSP" />
244.
245.     </android.support.constraint.ConstraintLayout>
246.
247.     <android.support.constraint.ConstraintLayout
248.         android:layout_width="match_parent"
249.         android:layout_height="350dp"
250.         app:layout_constraintEnd_toEndOf="parent"
251.         app:layout_constraintTop_toBottomOf="@+id/constraintLayout">
252.
253.         <ImageView
254.             android:id="@+id/imgCaldera"
255.             android:layout_width="318dp"
256.             android:layout_height="330dp"
257.             android:scaleType="fitCenter"
258.             app:layout_constraintBottom_toBottomOf="parent"
259.             app:layout_constraintEnd_toEndOf="parent"
260.             app:layout_constraintStart_toStartOf="parent"
261.             app:layout_constraintTop_toTopOf="parent"
262.             app:layout_constraintVertical_bias="0.0"
263.             app:srcCompat="@mipmap/caldera_color" />
264.
265.         <ImageView
266.             android:id="@+id/img_fuego"
267.             android:layout_width="75dp"
268.             android:layout_height="75dp"
269.             app:layout_constraintBottom_toBottomOf="parent"
270.             app:layout_constraintEnd_toEndOf="parent"
271.             app:layout_constraintStart_toStartOf="parent"
272.             app:layout_constraintTop_toTopOf="parent"
273.             app:layout_constraintVertical_bias="0.709" />
274.
275.         <ImageView
276.             android:id="@+id/img_EstacionAno_home"
277.             android:layout_width="30dp"
278.             android:layout_height="30dp"
279.             app:layout_constraintBottom_toBottomOf="@+id/imgCaldera"
280.             app:layout_constraintEnd_toEndOf="@+id/imgCaldera"
281.             app:layout_constraintStart_toStartOf="@+id/imgCaldera"
282.             app:layout_constraintTop_toTopOf="@+id/imgCaldera"
283.             app:layout_constraintVertical_bias="0.206"
284.             app:srcCompat="@drawable/ic_modo_invierno_24dp" />
285.
286.         <ImageView
287.             android:id="@+id/img_ModoEco_home"
288.             android:layout_width="30dp"
289.             android:layout_height="30dp"
290.             app:layout_constraintBottom_toBottomOf="@+id/imgCaldera"
291.             app:layout_constraintEnd_toEndOf="@+id/imgCaldera"
292.             app:layout_constraintStart_toStartOf="@+id/imgCaldera"
293.             app:layout_constraintTop_toTopOf="@+id/imgCaldera"
294.             app:layout_constraintVertical_bias="0.347" />
295.
296.         <ImageView
297.             android:id="@+id/img_ModoCasaVacaciones_home"
298.             android:layout_width="30dp"
299.             android:layout_height="30dp"
```

```
300.             app:layout_constraintBottom_toBottomOf="@+id/imgCaldera"
301.             app:layout_constraintEnd_toEndOf="@+id/imgCaldera"
302.             app:layout_constraintStart_toStartOf="@+id/imgCaldera"
303.             app:layout_constraintTop_toTopOf="@+id/imgCaldera"
304.             app:layout_constraintVertical_bias="0.488"
305.             app:srcCompat="@drawable/nav_home_24dp" />
306.
307.     </android.support.constraint.ConstraintLayout>
308.
309. </android.support.constraint.ConstraintLayout>
```

8.4.12. fragment_calendario.xml

```
1. <?xml version="1.0" encoding="utf-8"?>
2. <ScrollView
3.     xmlns:android="http://schemas.android.com/apk/res/android"
4.     xmlns:app="http://schemas.android.com/apk/res-auto"
5.     xmlns:tools="http://schemas.android.com/tools"
6.     android:orientation="vertical"
7.     android:layout_width="match_parent"
8.     android:layout_height="match_parent"
9.     android:background="@color/colorGris"
10.      app:layout_behavior="@string/appbar_scrolling_view_behavior"
11.      tools:context=".CalendarioFragment">
12.
13.      <LinearLayout
14.          android:layout_width="match_parent"
15.          android:layout_height="wrap_content"
16.          android:orientation="vertical">
17.
18.          <!--Tabla de DESCRIPCIÓN MODOS DIAS-->
19.          <TableLayout
20.              android:layout_width="match_parent"
21.              android:layout_height="wrap_content"
22.              android:background="@color/colorGris_main"
23.              >
24.
25.              <TableRow
26.                  android:layout_width="match_parent"
27.                  android:layout_height="match_parent">
28.
29.                  <ImageView
30.                      android:layout_width="wrap_content"
31.                      android:layout_height="wrap_content"
32.                      android:padding="7dp"
33.                      app:srcCompat="@drawable/ic_info_24dp" />
34.
35.                  <TextView
36.                      android:layout_width="wrap_content"
37.                      android:layout_height="wrap_content"
38.                      android:layout_marginStart="4dp"
39.                      android:layout_marginTop="3dp"
40.                      android:text="CALENDARIO"
41.                      android:textColor="@color/colorPrimaryDark"
42.                      android:textSize="24dp"
43.                      android:textStyle="bold|italic" />
44.
45.              </TableRow>
46.
47.              <TableRow
48.                  android:layout_width="match_parent"
49.                  android:layout_height="match_parent">
50.
51.                  <ImageView
52.                      android:layout_width="24dp"
53.                      android:layout_height="20dp" />
54.
55.                  <TextView
56.                      android:layout_width="wrap_content"
57.                      android:layout_height="wrap_content"
58.                      android:textColor="@color/colorPrimaryDark"
```

```
59.                     android:layout_marginStart="4dp"
60.                     android:layout_marginBottom="8dp"
61.                     android:textSize="16dp"
62.                     android:textStyle="italic"
63.                     android:text="Fije el modo de calendario, diario o
   festivo, de los\ndías de la semana:" />
64.
65.             </TableRow>
66.
67.         </TableLayout>
68.
69.         <!--Tabla de MODOS DIAS-->
70.         <TableLayout
71.             android:layout_width="match_parent"
72.             android:layout_height="wrap_content"
73.             android:background="@color/colorGris"
74.             >
75.
76.             <!--Fila LUNES-->
77.             <TableRow
78.                 android:layout_width="match_parent"
79.                 android:layout_height="match_parent">
80.
81.                 <ImageView
82.                     android:layout_width="24dp"
83.                     android:layout_height="24dp" />
84.
85.                 <TextView
86.                     android:layout_width="wrap_content"
87.                     android:layout_height="wrap_content"
88.                     android:layout_marginStart="18dp"
89.                     android:text="LUNES"
90.                     android:textColor="@color/colorPrimaryDark"
91.                     android:textSize="20dp"
92.                     android:textStyle="bold" />
93.
94.                 <TextView
95.                     android:layout_width="wrap_content"
96.                     android:layout_height="wrap_content"
97.                     android:textColor="@color/colorPrimaryDark"
98.                     android:layout_marginStart="14dp"
99.                     android:layout_marginEnd="4dp"
100.                    android:textSize="18dp"
101.                    android:text="[D] Diario" />
102.
103.                <Switch
104.                    android:id="@+id/switch_lunes"
105.                    android:layout_width="wrap_content"
106.                    android:layout_height="wrap_content" />
107.
108.                <TextView
109.                    android:layout_width="wrap_content"
110.                    android:layout_height="wrap_content"
111.                    android:textColor="@color/colorPrimaryDark"
112.                    android:layout_marginStart="4dp"
113.                    android:textSize="18dp"
114.                    android:text="[F] Festivo" />
115.
116.            </TableRow>
117.
118.            <!--Fila MARTES-->
```

```
119.            <TableRow
120.                android:layout_width="match_parent"
121.                android:layout_height="match_parent"
122.                android:layout_marginTop="@dimen/padding_between">
123.
124.                <ImageView
125.                    android:layout_width="24dp"
126.                    android:layout_height="24dp" />
127.
128.                <TextView
129.                    android:layout_width="wrap_content"
130.                    android:layout_height="wrap_content"
131.                    android:layout_marginStart="18dp"
132.                    android:text="MARTES"
133.                    android:textColor="@color/colorPrimaryDark"
134.                    android:textSize="20dp"
135.                    android:textStyle="bold" />
136.
137.                <TextView
138.                    android:layout_width="wrap_content"
139.                    android:layout_height="wrap_content"
140.                    android:textColor="@color/colorPrimaryDark"
141.                    android:layout_marginStart="14dp"
142.                    android:layout_marginEnd="4dp"
143.                    android:textSize="18dp"
144.                    android:text="[D] Diario" />
145.
146.                <Switch
147.                    android:id="@+id/switch_martes"
148.                    android:layout_width="wrap_content"
149.                    android:layout_height="wrap_content" />
150.
151.                <TextView
152.                    android:layout_width="wrap_content"
153.                    android:layout_height="wrap_content"
154.                    android:textColor="@color/colorPrimaryDark"
155.                    android:layout_marginStart="4dp"
156.                    android:textSize="18dp"
157.                    android:text="[F] Festivo" />
158.
159.            </TableRow>
160.
161.            <!--Fila MIERCOLES-->
162.            <TableRow
163.                android:layout_width="match_parent"
164.                android:layout_height="match_parent"
165.                android:layout_marginTop="@dimen/padding_between">
166.
167.                <ImageView
168.                    android:layout_width="24dp"
169.                    android:layout_height="24dp" />
170.
171.                <TextView
172.                    android:layout_width="wrap_content"
173.                    android:layout_height="wrap_content"
174.                    android:layout_marginStart="18dp"
175.                    android:text="MIÉRCOLES"
176.                    android:textColor="@color/colorPrimaryDark"
177.                    android:textSize="20dp"
178.                    android:textStyle="bold" />
179.
```

```
180.                 <TextView
181.                     android:layout_width="wrap_content"
182.                     android:layout_height="wrap_content"
183.                     android:textColor="@color/colorPrimaryDark"
184.                     android:layout_marginStart="14dp"
185.                     android:layout_marginEnd="4dp"
186.                     android:textSize="18dp"
187.                     android:text="[D] Diario" />
188.
189.                 <Switch
190.                     android:id="@+id/switch_miercoles"
191.                     android:layout_width="wrap_content"
192.                     android:layout_height="wrap_content" />
193.
194.                 <TextView
195.                     android:layout_width="wrap_content"
196.                     android:layout_height="wrap_content"
197.                     android:textColor="@color/colorPrimaryDark"
198.                     android:layout_marginStart="4dp"
199.                     android:textSize="18dp"
200.                     android:text="[F] Festivo" />
201.
202.             </TableRow>
203.
204.             <!--Fila JUEVES-->
205.             <TableRow
206.                 android:layout_width="match_parent"
207.                 android:layout_height="match_parent"
208.                 android:layout_marginTop="@dimen/padding_between">
209.
210.                 <ImageView
211.                     android:layout_width="24dp"
212.                     android:layout_height="24dp" />
213.
214.                 <TextView
215.                     android:layout_width="wrap_content"
216.                     android:layout_height="wrap_content"
217.                     android:layout_marginStart="18dp"
218.                     android:text="JUEVES"
219.                     android:textColor="@color/colorPrimaryDark"
220.                     android:textSize="20dp"
221.                     android:textStyle="bold" />
222.
223.                 <TextView
224.                     android:layout_width="wrap_content"
225.                     android:layout_height="wrap_content"
226.                     android:textColor="@color/colorPrimaryDark"
227.                     android:layout_marginStart="14dp"
228.                     android:layout_marginEnd="4dp"
229.                     android:textSize="18dp"
230.                     android:text="[D] Diario" />
231.
232.                 <Switch
233.                     android:id="@+id/switch_jueves"
234.                     android:layout_width="wrap_content"
235.                     android:layout_height="wrap_content" />
236.
237.
238.                 <TextView
239.                     android:layout_width="wrap_content"
240.                     android:layout_height="wrap_content"
```

```
241.                     android:textColor="@color/colorPrimaryDark"
242.                     android:layout_marginStart="4dp"
243.                     android:textSize="18dp"
244.                     android:text="[F] Festivo" />
245.
246.             </TableRow>
247.
248.             <!--Fila VIERNES-->
249.             <TableRow
250.                 android:layout_width="match_parent"
251.                 android:layout_height="match_parent"
252.                 android:layout_marginTop="@dimen/padding_between">
253.
254.                 <ImageView
255.                     android:layout_width="24dp"
256.                     android:layout_height="24dp" />
257.
258.                 <TextView
259.                     android:layout_width="wrap_content"
260.                     android:layout_height="wrap_content"
261.                     android:layout_marginStart="18dp"
262.                     android:text="VIERNES"
263.                     android:textColor="@color/colorPrimaryDark"
264.                     android:textSize="20dp"
265.                     android:textStyle="bold" />
266.
267.                 <TextView
268.                     android:layout_width="wrap_content"
269.                     android:layout_height="wrap_content"
270.                     android:textColor="@color/colorPrimaryDark"
271.                     android:layout_marginStart="14dp"
272.                     android:layout_marginEnd="4dp"
273.                     android:textSize="18dp"
274.                     android:text="[D] Diario" />
275.
276.                 <Switch
277.                     android:id="@+id/switch_viernes"
278.                     android:layout_width="wrap_content"
279.                     android:layout_height="wrap_content" />
280.
281.                 <TextView
282.                     android:layout_width="wrap_content"
283.                     android:layout_height="wrap_content"
284.                     android:textColor="@color/colorPrimaryDark"
285.                     android:layout_marginStart="4dp"
286.                     android:textSize="18dp"
287.                     android:text="[F] Festivo" />
288.
289.             </TableRow>
290.
291.             <!--Fila SÁBADO-->
292.             <TableRow
293.                 android:layout_width="match_parent"
294.                 android:layout_height="match_parent"
295.                 android:layout_marginTop="@dimen/padding_between">
296.
297.                 <ImageView
298.                     android:layout_width="24dp"
299.                     android:layout_height="24dp" />
300.
301.                 <TextView
```

```
302.                     android:layout_width="wrap_content"
303.                     android:layout_height="wrap_content"
304.                     android:layout_marginStart="18dp"
305.                     android:text="SÁBADO"
306.                     android:textColor="@color/colorPrimaryDark"
307.                     android:textSize="20dp"
308.                     android:textStyle="bold" />
309.
310.                 <TextView
311.                     android:layout_width="wrap_content"
312.                     android:layout_height="wrap_content"
313.                     android:textColor="@color/colorPrimaryDark"
314.                     android:layout_marginStart="14dp"
315.                     android:layout_marginEnd="4dp"
316.                     android:textSize="18dp"
317.                     android:text="[D] Diario" />
318.
319.                 <Switch
320.                     android:id="@+id/switch_sabado"
321.                     android:layout_width="wrap_content"
322.                     android:layout_height="wrap_content" />
323.
324.
325.                 <TextView
326.                     android:layout_width="wrap_content"
327.                     android:layout_height="wrap_content"
328.                     android:textColor="@color/colorPrimaryDark"
329.                     android:layout_marginStart="4dp"
330.                     android:textSize="18dp"
331.                     android:text="[F] Festivo" />
332.
333.             </TableRow>
334.
335.             <!--Fila DOMINGO-->
336.             <TableRow
337.                 android:layout_width="match_parent"
338.                 android:layout_height="match_parent"
339.                 android:layout_marginTop="@dimen/padding_between"
340.                 android:layout_marginBottom="@dimen/padding_between">
341.
342.                 <ImageView
343.                     android:layout_width="24dp"
344.                     android:layout_height="24dp" />
345.
346.                 <TextView
347.                     android:layout_width="wrap_content"
348.                     android:layout_height="wrap_content"
349.                     android:layout_marginStart="18dp"
350.                     android:text="DOMINGO"
351.                     android:textColor="@color/colorPrimaryDark"
352.                     android:textSize="20dp"
353.                     android:textStyle="bold" />
354.
355.                 <TextView
356.                     android:layout_width="wrap_content"
357.                     android:layout_height="wrap_content"
358.                     android:textColor="@color/colorPrimaryDark"
359.                     android:layout_marginStart="14dp"
360.                     android:layout_marginEnd="4dp"
361.                     android:textSize="18dp"
362.                     android:text="[D] Diario" />
```

```
363.
364.                 <Switch
365.                     android:id="@+id/switch_domingo"
366.                     android:layout_width="wrap_content"
367.                     android:layout_height="wrap_content" />
368.
369.                 <TextView
370.                     android:layout_width="wrap_content"
371.                     android:layout_height="wrap_content"
372.                     android:textColor="@color/colorPrimaryDark"
373.                     android:layout_marginStart="4dp"
374.                     android:textSize="18dp"
375.                     android:text="[F] Festivo" />
376.
377.             </TableRow>
378.
379.         </TableLayout>
380.
381.         <!--Tabla de DESCRIPCIÓN del CALENDARIO-->
382.         <TableLayout
383.             android:layout_width="match_parent"
384.             android:layout_height="wrap_content"
385.             android:background="@color/colorGris_main"
386.             >
387.
388.             <TableRow
389.                 android:layout_width="match_parent"
390.                 android:layout_height="match_parent">
391.
392.                 <ImageView
393.                     android:layout_width="wrap_content"
394.                     android:layout_height="wrap_content"
395.                     android:padding="7dp"
396.                     app:srcCompat="@drawable/ic_info_24dp" />
397.
398.                 <TextView
399.                     android:layout_width="wrap_content"
400.                     android:layout_height="wrap_content"
401.                     android:layout_marginStart="4dp"
402.                     android:layout_marginTop="3dp"
403.                     android:text="MODOS"
404.                     android:textAllCaps="false"
405.                     android:textColor="@color/colorPrimaryDark"
406.                     android:textSize="24dp"
407.                     android:textStyle="bold|italic" />
408.
409.             </TableRow>
410.
411.             <TableRow
412.                 android:layout_width="match_parent"
413.                 android:layout_height="match_parent">
414.
415.                 <ImageView
416.                     android:layout_width="24dp"
417.                     android:layout_height="20dp" />
418.
419.                 <TextView
420.                     android:layout_width="wrap_content"
421.                     android:layout_height="wrap_content"
422.                     android:textColor="@color/colorPrimaryDark"
423.                     android:layout_marginStart="4dp"
```

```
424.                     android:layout_marginBottom="8dp"
425.                     android:textSize="16dp"
426.                     android:textStyle="italic"
427.                     android:text="Fije la temperatura y los horarios de
   los intervalos\nde los modos: diario [D] y festivo [F]."/>
428.
429.             </TableRow>
430.
431.         </TableLayout>
432.
433.         <HorizontalScrollView
434.             android:layout_width="match_parent"
435.             android:layout_height="match_parent"
436.             app:layout_behavior="@string/appbar_scrolling_view_behavior">
437.
438.             <!--Tabla de CALENDARIO-->
439.             <TableLayout
440.                 android:layout_width="match_parent"
441.                 android:layout_height="wrap_content"
442.                 android:layout_marginStart="10dp"
443.                 android:layout_marginTop="10dp"
444.                 android:layout_marginBottom="10dp"
445.                 android:background="@color/colorGris">
446.
447.                 <!--Primera fila CABECERA-->
448.                 <TableRow
449.                     android:layout_width="match_parent"
450.                     android:layout_height="match_parent"
451.                     android:layout_marginEnd="20dp">
452.
453.                     <ImageView
454.                         android:layout_width="58dp"
455.                         android:layout_height="58dp"
456.
   android:background="@drawable/tabla_calendario_border"
457.                         android:padding="7dp"
458.                         app:srcCompat="@drawable/nav_calendario_24dp" />
459.
460.                     <TextView
461.                         android:layout_width="58dp"
462.                         android:layout_height="58dp"
463.
   android:background="@drawable/tabla_calendario_border"
464.                         android:gravity="center"
465.                         android:text="1"
466.                         android:textAlignment="center"
467.                         android:textAllCaps="true"
468.                         android:textColor="@color/colorPrimaryDark"
469.                         android:textSize="34dp"
470.                         android:textStyle="bold" />
471.
472.                     <TextView
473.                         android:layout_width="58dp"
474.                         android:layout_height="58dp"
475.
   android:background="@drawable/tabla_calendario_border"
476.                         android:gravity="center"
477.                         android:text="2"
478.                         android:textAlignment="center"
479.                         android:textAllCaps="true"
480.                         android:textColor="@color/colorPrimaryDark"
```

```
481.                         android:textSize="34dp"
482.                         android:textStyle="bold" />
483.
484.                     <TextView
485.                         android:layout_width="58dp"
486.                         android:layout_height="58dp"
487.
   android:background="@drawable/tabla_calendario_border"
488.                         android:gravity="center"
489.                         android:text="3"
490.                         android:textAlignment="center"
491.                         android:textAllCaps="true"
492.                         android:textColor="@color/colorPrimaryDark"
493.                         android:textSize="34dp"
494.                         android:textStyle="bold" />
495.
496.                     <TextView
497.                         android:layout_width="58dp"
498.                         android:layout_height="58dp"
499.
   android:background="@drawable/tabla_calendario_border"
500.                         android:gravity="center"
501.                         android:text="4"
502.                         android:textAlignment="center"
503.                         android:textAllCaps="true"
504.                         android:textColor="@color/colorPrimaryDark"
505.                         android:textSize="34dp"
506.                         android:textStyle="bold" />
507.
508.                     <TextView
509.                         android:layout_width="58dp"
510.                         android:layout_height="58dp"
511.
   android:background="@drawable/tabla_calendario_border"
512.                         android:gravity="center"
513.                         android:text="5"
514.                         android:textAlignment="center"
515.                         android:textAllCaps="true"
516.                         android:textColor="@color/colorPrimaryDark"
517.                         android:textSize="34dp"
518.                         android:textStyle="bold" />
519.
520.                     <TextView
521.                         android:layout_width="58dp"
522.                         android:layout_height="58dp"
523.
   android:background="@drawable/tabla_calendario_border"
524.                         android:gravity="center"
525.                         android:text="6"
526.                         android:textAlignment="center"
527.                         android:textAllCaps="true"
528.                         android:textColor="@color/colorPrimaryDark"
529.                         android:textSize="34dp"
530.                         android:textStyle="bold" />
531.
532.                 </TableRow>
533.
534.                 <!--Segunda fila DIARIO-->
535.                 <TableRow
536.                     android:layout_width="match_parent"
537.                     android:layout_height="match_parent">
```

```
538.
539.                     <TextView
540.                         android:layout_width="58dp"
541.                         android:layout_height="78dp"
542.
   android:background="@drawable/tabla_calendario_border"
543.                         android:gravity="center"
544.                         android:text="D"
545.                         android:textAlignment="center"
546.                         android:textAllCaps="true"
547.                         android:textColor="@color/colorPrimaryDark"
548.                         android:textSize="54dp"
549.                         android:textStyle="bold" />
550.
551.                     <!--Celda D1-->
552.                     <TableLayout
553.                         android:layout_width="match_parent"
554.                         android:layout_height="wrap_content"
555.
   android:background="@drawable/tabla_calendario_border">
556.
557.                         <TableRow
558.                             android:layout_width="match_parent"
559.                             android:layout_height="match_parent">
560.
561.                             <TextView
562.                                 android:id="@+id/temp_D1"
563.                                 android:layout_width="58dp"
564.                                 android:layout_height="50dp"
565.                                 android:clickable="true"
566.                                 android:gravity="center"
567.                                 android:text="16°C"
568.                                 android:textAlignment="center"
569.                                 android:textAllCaps="true"
570.
   android:textColor="@color/colorPrimaryDark"
571.                                 android:textSize="24dp" />
572.
573.                         </TableRow>
574.
575.                         <TableRow
576.                             android:layout_width="match_parent"
577.                             android:layout_height="match_parent">
578.
579.                             <TextView
580.                                 android:id="@+id/hora_D1"
581.                                 android:layout_width="58dp"
582.                                 android:layout_height="28dp"
583.
   android:background="@drawable/tabla_calendario_border"
584.                                 android:clickable="true"
585.                                 android:gravity="center"
586.                                 android:text="10:00 am"
587.                                 android:textAlignment="center"
588.                                 android:textAllCaps="true"
589.
   android:textColor="@color/colorPrimaryDark"
590.                                 android:textSize="12dp" />
591.
592.                         </TableRow>
593.
```

```
594.                     </TableLayout>
595.
596.                     <!--Celda D2-->
597.                     <TableLayout
598.                         android:layout_width="match_parent"
599.                         android:layout_height="wrap_content"
600.
   android:background="@drawable/tabla_calendario_border">
601.
602.                         <TableRow
603.                             android:layout_width="match_parent"
604.                             android:layout_height="match_parent">
605.
606.                             <TextView
607.                                 android:id="@+id/temp_D2"
608.                                 android:layout_width="58dp"
609.                                 android:layout_height="50dp"
610.                                 android:clickable="true"
611.                                 android:gravity="center"
612.                                 android:text="20°C"
613.                                 android:textAlignment="center"
614.                                 android:textAllCaps="true"
615.
   android:textColor="@color/colorPrimaryDark"
616.                                 android:textSize="24dp" />
617.
618.                         </TableRow>
619.
620.                         <TableRow
621.                             android:layout_width="match_parent"
622.                             android:layout_height="match_parent">
623.
624.                             <TextView
625.                                 android:id="@+id/hora_D2"
626.                                 android:layout_width="58dp"
627.                                 android:layout_height="28dp"
628.
   android:background="@drawable/tabla_calendario_border"
629.                                 android:clickable="true"
630.                                 android:gravity="center"
631.                                 android:text="14:00 pm"
632.                                 android:textAlignment="center"
633.                                 android:textAllCaps="true"
634.
   android:textColor="@color/colorPrimaryDark"
635.                                 android:textSize="12dp" />
636.
637.                         </TableRow>
638.
639.                     </TableLayout>
640.
641.                     <!--Celda D3-->
642.                     <TableLayout
643.                         android:layout_width="match_parent"
644.                         android:layout_height="wrap_content"
645.
   android:background="@drawable/tabla_calendario_border">
646.
647.                         <TableRow
648.                             android:layout_width="match_parent"
649.                             android:layout_height="match_parent">
```

```
650.
651.                               <TextView
652.                                   android:id="@+id/temp_D3"
653.                                   android:layout_width="58dp"
654.                                   android:layout_height="50dp"
655.                                   android:clickable="true"
656.                                   android:gravity="center"
657.                                   android:text="18°C"
658.                                   android:textAlignment="center"
659.                                   android:textAllCaps="true"
660.
   android:textColor="@color/colorPrimaryDark"
661.                                   android:textSize="24dp" />
662.
663.                           </TableRow>
664.
665.                           <TableRow
666.                               android:layout_width="match_parent"
667.                               android:layout_height="match_parent">
668.
669.                               <TextView
670.                                   android:id="@+id/hora_D3"
671.                                   android:layout_width="58dp"
672.                                   android:layout_height="28dp"
673.
   android:background="@drawable/tabla_calendario_border"
674.                                   android:clickable="true"
675.                                   android:gravity="center"
676.                                   android:text="17:00 pm"
677.                                   android:textAlignment="center"
678.                                   android:textAllCaps="true"
679.
   android:textColor="@color/colorPrimaryDark"
680.                                   android:textSize="12dp" />
681.
682.                           </TableRow>
683.
684.                       </TableLayout>
685.
686.                       <!--Celda D4-->
687.                       <TableLayout
688.                           android:layout_width="match_parent"
689.                           android:layout_height="wrap_content"
690.
   android:background="@drawable/tabla_calendario_border">
691.
692.                           <TableRow
693.                               android:layout_width="match_parent"
694.                               android:layout_height="match_parent">
695.
696.                               <TextView
697.                                   android:id="@+id/temp_D4"
698.                                   android:layout_width="58dp"
699.                                   android:layout_height="50dp"
700.                                   android:clickable="true"
701.                                   android:gravity="center"
702.                                   android:text="21°C"
703.                                   android:textAlignment="center"
704.                                   android:textAllCaps="true"
705.
   android:textColor="@color/colorPrimaryDark"
```

```
706.                                    android:textSize="24dp" />
707.
708.                            </TableRow>
709.
710.                            <TableRow
711.                                android:layout_width="match_parent"
712.                                android:layout_height="match_parent">
713.
714.                                <TextView
715.                                    android:id="@+id/hora_D4"
716.                                    android:layout_width="58dp"
717.                                    android:layout_height="28dp"
718.
   android:background="@drawable/tabla_calendario_border"
719.                                    android:clickable="true"
720.                                    android:gravity="center"
721.                                    android:text="19:30 pm"
722.                                    android:textAlignment="center"
723.                                    android:textAllCaps="true"
724.
   android:textColor="@color/colorPrimaryDark"
725.                                    android:textSize="12dp" />
726.
727.                            </TableRow>
728.
729.                        </TableLayout>
730.
731.                        <!--Celda D5-->
732.                        <TableLayout
733.                            android:layout_width="match_parent"
734.                            android:layout_height="wrap_content"
735.
   android:background="@drawable/tabla_calendario_border">
736.
737.                            <TableRow
738.                                android:layout_width="match_parent"
739.                                android:layout_height="match_parent">
740.
741.                                <TextView
742.                                    android:id="@+id/temp_D5"
743.                                    android:layout_width="58dp"
744.                                    android:layout_height="50dp"
745.                                    android:clickable="true"
746.                                    android:gravity="center"
747.                                    android:text="18°C"
748.                                    android:textAlignment="center"
749.                                    android:textAllCaps="true"
750.
   android:textColor="@color/colorPrimaryDark"
751.                                    android:textSize="24dp" />
752.
753.                            </TableRow>
754.
755.                            <TableRow
756.                                android:layout_width="match_parent"
757.                                android:layout_height="match_parent">
758.
759.                                <TextView
760.                                    android:id="@+id/hora_D5"
761.                                    android:layout_width="58dp"
762.                                    android:layout_height="28dp"
```

```
763.
   android:background="@drawable/tabla_calendario_border"
764.                                android:clickable="true"
765.                                android:gravity="center"
766.                                android:text="10:00 pm"
767.                                android:textAlignment="center"
768.                                android:textAllCaps="true"
769.
   android:textColor="@color/colorPrimaryDark"
770.                                android:textSize="12dp" />
771.
772.                            </TableRow>
773.
774.                        </TableLayout>
775.
776.                        <!--Celda D6-->
777.                        <TableLayout
778.                            android:layout_width="match_parent"
779.                            android:layout_height="wrap_content"
780.
   android:background="@drawable/tabla_calendario_border">
781.
782.                            <TableRow
783.                                android:layout_width="match_parent"
784.                                android:layout_height="match_parent">
785.
786.                                <TextView
787.                                    android:id="@+id/temp_D6"
788.                                    android:layout_width="58dp"
789.                                    android:layout_height="50dp"
790.                                    android:clickable="true"
791.                                    android:gravity="center"
792.                                    android:text="18°C"
793.                                    android:textAlignment="center"
794.                                    android:textAllCaps="true"
795.
   android:textColor="@color/colorPrimaryDark"
796.                                    android:textSize="24dp" />
797.
798.                            </TableRow>
799.
800.                            <TableRow
801.                                android:layout_width="match_parent"
802.                                android:layout_height="match_parent">
803.
804.                                <TextView
805.                                    android:id="@+id/hora_D6"
806.                                    android:layout_width="58dp"
807.                                    android:layout_height="28dp"
808.
   android:background="@drawable/tabla_calendario_border"
809.                                    android:clickable="true"
810.                                    android:gravity="center"
811.                                    android:text="10:00 pm"
812.                                    android:textAlignment="center"
813.                                    android:textAllCaps="true"
814.
   android:textColor="@color/colorPrimaryDark"
815.                                    android:textSize="12dp" />
816.
817.                            </TableRow>
```

```
818.
819.                     </TableLayout>
820.
821.
822.                 </TableRow>
823.
824.                 <!--Tercera fila FESTIVO-->
825.                 <TableRow
826.                     android:layout_width="match_parent"
827.                     android:layout_height="match_parent">
828.
829.                     <TextView
830.                         android:layout_width="58dp"
831.                         android:layout_height="78dp"
832.
   android:background="@drawable/tabla_calendario_border"
833.                         android:gravity="center"
834.                         android:text="F"
835.                         android:textAlignment="center"
836.                         android:textAllCaps="true"
837.                         android:textColor="@color/colorPrimaryDark"
838.                         android:textSize="54dp"
839.                         android:textStyle="bold" />
840.
841.                     <!--Celda F1-->
842.                     <TableLayout
843.                         android:layout_width="match_parent"
844.                         android:layout_height="wrap_content"
845.
   android:background="@drawable/tabla_calendario_border">
846.
847.                         <TableRow
848.                             android:layout_width="match_parent"
849.                             android:layout_height="match_parent">
850.
851.                             <TextView
852.                                 android:id="@+id/temp_F1"
853.                                 android:layout_width="58dp"
854.                                 android:layout_height="50dp"
855.                                 android:clickable="true"
856.                                 android:gravity="center"
857.                                 android:text="16°C"
858.                                 android:textAlignment="center"
859.                                 android:textAllCaps="true"
860.
   android:textColor="@color/colorPrimaryDark"
861.                                 android:textSize="24dp" />
862.
863.                         </TableRow>
864.
865.                         <TableRow
866.                             android:layout_width="match_parent"
867.                             android:layout_height="match_parent">
868.
869.                             <TextView
870.                                 android:id="@+id/hora_F1"
871.                                 android:layout_width="58dp"
872.                                 android:layout_height="28dp"
873.
   android:background="@drawable/tabla_calendario_border"
874.                                 android:clickable="true"
```

```
875.                                android:gravity="center"
876.                                android:text="10:00 am"
877.                                android:textAlignment="center"
878.                                android:textAllCaps="true"
879.
   android:textColor="@color/colorPrimaryDark"
880.                                android:textSize="12dp" />
881.
882.                            </TableRow>
883.
884.                        </TableLayout>
885.
886.                        <!--Celda F2-->
887.                        <TableLayout
888.                            android:layout_width="match_parent"
889.                            android:layout_height="wrap_content"
890.
   android:background="@drawable/tabla_calendario_border">
891.
892.                            <TableRow
893.                                android:layout_width="match_parent"
894.                                android:layout_height="match_parent">
895.
896.                                <TextView
897.                                    android:id="@+id/temp_F2"
898.                                    android:layout_width="58dp"
899.                                    android:layout_height="50dp"
900.                                    android:clickable="true"
901.                                    android:gravity="center"
902.                                    android:text="20°C"
903.                                    android:textAlignment="center"
904.                                    android:textAllCaps="true"
905.
   android:textColor="@color/colorPrimaryDark"
906.                                    android:textSize="24dp" />
907.
908.                            </TableRow>
909.
910.                            <TableRow
911.                                android:layout_width="match_parent"
912.                                android:layout_height="match_parent">
913.
914.                                <TextView
915.                                    android:id="@+id/hora_F2"
916.                                    android:layout_width="58dp"
917.                                    android:layout_height="28dp"
918.
   android:background="@drawable/tabla_calendario_border"
919.                                    android:clickable="true"
920.                                    android:gravity="center"
921.                                    android:text="14:00 pm"
922.                                    android:textAlignment="center"
923.                                    android:textAllCaps="true"
924.
   android:textColor="@color/colorPrimaryDark"
925.                                    android:textSize="12dp" />
926.
927.                            </TableRow>
928.
929.                        </TableLayout>
930.
```

```
931.                     <!--Celda F3-->
932.                     <TableLayout
933.                         android:layout_width="match_parent"
934.                         android:layout_height="wrap_content"
935.
   android:background="@drawable/tabla_calendario_border">
936.
937.                         <TableRow
938.                             android:layout_width="match_parent"
939.                             android:layout_height="match_parent">
940.
941.                             <TextView
942.                                 android:id="@+id/temp_F3"
943.                                 android:layout_width="58dp"
944.                                 android:layout_height="50dp"
945.                                 android:clickable="true"
946.                                 android:gravity="center"
947.                                 android:text="18°C"
948.                                 android:textAlignment="center"
949.                                 android:textAllCaps="true"
950.
   android:textColor="@color/colorPrimaryDark"
951.                                 android:textSize="24dp" />
952.
953.                         </TableRow>
954.
955.                         <TableRow
956.                             android:layout_width="match_parent"
957.                             android:layout_height="match_parent">
958.
959.                             <TextView
960.                                 android:id="@+id/hora_F3"
961.                                 android:layout_width="58dp"
962.                                 android:layout_height="28dp"
963.
   android:background="@drawable/tabla_calendario_border"
964.                                 android:clickable="true"
965.                                 android:gravity="center"
966.                                 android:text="17:00 pm"
967.                                 android:textAlignment="center"
968.                                 android:textAllCaps="true"
969.
   android:textColor="@color/colorPrimaryDark"
970.                                 android:textSize="12dp" />
971.
972.                         </TableRow>
973.
974.                     </TableLayout>
975.
976.                     <!--Celda F4-->
977.                     <TableLayout
978.                         android:layout_width="match_parent"
979.                         android:layout_height="wrap_content"
980.
   android:background="@drawable/tabla_calendario_border">
981.
982.                         <TableRow
983.                             android:layout_width="match_parent"
984.                             android:layout_height="match_parent">
985.
986.                             <TextView
```

```
                                    android:id="@+id/temp_F4"
                                    android:layout_width="58dp"
                                    android:layout_height="50dp"
                                    android:clickable="true"
                                    android:gravity="center"
                                    android:text="21°C"
                                    android:textAlignment="center"
                                    android:textAllCaps="true"
   android:textColor="@color/colorPrimaryDark"
                                    android:textSize="24dp" />

                            </TableRow>

                                <TableRow
                                    android:layout_width="match_parent"
                                    android:layout_height="match_parent">

                                    <TextView
                                        android:id="@+id/hora_F4"
                                        android:layout_width="58dp"
                                        android:layout_height="28dp"
   android:background="@drawable/tabla_calendario_border"
                                        android:clickable="true"
                                        android:gravity="center"
                                        android:text="19:30 pm"
                                        android:textAlignment="center"
                                        android:textAllCaps="true"
   android:textColor="@color/colorPrimaryDark"
                                        android:textSize="12dp" />

                                </TableRow>

                            </TableLayout>

                            <!--Celda F5-->
                            <TableLayout
                                android:layout_width="match_parent"
                                android:layout_height="wrap_content"
   android:background="@drawable/tabla_calendario_border">

                                <TableRow
                                    android:layout_width="match_parent"
                                    android:layout_height="match_parent">

                                    <TextView
                                        android:id="@+id/temp_F5"
                                        android:layout_width="58dp"
                                        android:layout_height="50dp"
                                        android:clickable="true"
                                        android:gravity="center"
                                        android:text="18°C"
                                        android:textAlignment="center"
                                        android:textAllCaps="true"
   android:textColor="@color/colorPrimaryDark"
                                        android:textSize="24dp" />
```

```
1043.                                   </TableRow>
1044.
1045.                                   <TableRow
1046.                                       android:layout_width="match_parent"
1047.                                       android:layout_height="match_parent">
1048.
1049.                                       <TextView
1050.                                           android:id="@+id/hora_F5"
1051.                                           android:layout_width="58dp"
1052.                                           android:layout_height="28dp"
1053.
   android:background="@drawable/tabla_calendario_border"
1054.                                           android:clickable="true"
1055.                                           android:gravity="center"
1056.                                           android:text="10:00 pm"
1057.                                           android:textAlignment="center"
1058.                                           android:textAllCaps="true"
1059.
   android:textColor="@color/colorPrimaryDark"
1060.                                           android:textSize="12dp" />
1061.
1062.                                   </TableRow>
1063.
1064.                               </TableLayout>
1065.
1066.                               <!--Celda F6-->
1067.                               <TableLayout
1068.                                   android:layout_width="match_parent"
1069.                                   android:layout_height="wrap_content"
1070.
   android:background="@drawable/tabla_calendario_border">
1071.
1072.                                   <TableRow
1073.                                       android:layout_width="match_parent"
1074.                                       android:layout_height="match_parent">
1075.
1076.                                       <TextView
1077.                                           android:id="@+id/temp_F6"
1078.                                           android:layout_width="58dp"
1079.                                           android:layout_height="50dp"
1080.                                           android:clickable="true"
1081.                                           android:gravity="center"
1082.                                           android:text="18°C"
1083.                                           android:textAlignment="center"
1084.                                           android:textAllCaps="true"
1085.
   android:textColor="@color/colorPrimaryDark"
1086.                                           android:textSize="24dp" />
1087.
1088.                                   </TableRow>
1089.
1090.                                   <TableRow
1091.                                       android:layout_width="match_parent"
1092.                                       android:layout_height="match_parent">
1093.
1094.                                       <TextView
1095.                                           android:id="@+id/hora_F6"
1096.                                           android:layout_width="58dp"
1097.                                           android:layout_height="28dp"
1098.
   android:background="@drawable/tabla_calendario_border"
```

```
1099.                                       android:clickable="true"
1100.                                       android:gravity="center"
1101.                                       android:text="10:00 pm"
1102.                                       android:textAlignment="center"
1103.                                       android:textAllCaps="true"
1104.
   android:textColor="@color/colorPrimaryDark"
1105.                                       android:textSize="12dp" />
1106.
1107.                                   </TableRow>
1108.
1109.                               </TableLayout>
1110.
1111.                           </TableRow>
1112.
1113.
1114.                       </TableLayout>
1115.
1116.                   </HorizontalScrollView>
1117.
1118.                   <!--Tabla de ayuda DESLIZAR-->
1119.                   <TableLayout
1120.                       android:layout_width="match_parent"
1121.                       android:layout_height="wrap_content"
1122.                       android:background="@color/colorGris"
1123.                       android:layout_marginBottom="10dp">
1124.
1125.                       <TableRow
1126.                           android:layout_width="match_parent"
1127.                           android:layout_height="match_parent">
1128.
1129.                           <ImageView
1130.                               android:layout_width="wrap_content"
1131.                               android:layout_height="wrap_content"
1132.                               android:padding="7dp"
1133.                               app:srcCompat="@drawable/ic_swap_horiz_24dp"
   />
1134.
1135.                           <TextView
1136.                               android:layout_width="wrap_content"
1137.                               android:layout_height="wrap_content"
1138.                               android:layout_marginStart="4dp"
1139.                               android:layout_marginTop="8dp"
1140.                               android:text="Desliza horizontalmente para
   configurar todos los\ncampos."
1141.                               android:textAllCaps="false"
1142.                               android:textColor="@color/colorPrimaryDark"
1143.                               android:textSize="13dp"
1144.                               android:textStyle="bold|italic" />
1145.
1146.                       </TableRow>
1147.
1148.
1149.
1150.                   </TableLayout>
1151.               </LinearLayout>
1152.
1153.           </ScrollView>
```

8.4.13. fragment_estadisticas.xml

```
1. <?xml version="1.0" encoding="utf-8"?>
2. <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3.     xmlns:tools="http://schemas.android.com/tools"
4.     android:layout_width="match_parent"
5.     android:layout_height="match_parent"
6.     tools:context=".EstadisticasFragment">
7.
8.     <!--Tabla de ESTADISTICAS-->
9.     <TableLayout
10.         android:id="@+id/ajustesPrincipales"
11.         android:layout_width="match_parent"
12.         android:layout_height="wrap_content"
13.         android:layout_marginTop="4dp"
14.         android:background="@color/colorGris_claro">
15.
16.         <!--Primera fila-->
17.         <TableRow
18.             android:layout_width="match_parent"
19.             android:layout_height="match_parent"
20.             android:background="@color/colorGris_main"
21.             android:padding="5dp"
22.             tools:visibility="visible">
23.
24.             <TextView
25.                 android:id="@+id/textView"
26.                 android:layout_width="181dp"
27.                 android:layout_height="wrap_content"
28.                 android:layout_marginStart="4dp"
29.                 android:text="ESTADISTICAS"
30.                 android:textAllCaps="true"
31.                 android:textColor="@color/colorPrimaryDark"
32.                 android:textSize="16dp"
33.                 android:textStyle="italic" />
34.         </TableRow>
35.
36.         <!--Segunda fila-->
37.         <TableRow
38.             android:layout_width="match_parent"
39.             android:layout_height="wrap_content"
40.             android:layout_gravity="center"
41.             android:layout_marginTop="1dp"
42.             android:background="@color/colorGris"
43.             android:padding="5dp">
44.
45.             <TextView
46.                 android:layout_width="123dp"
47.                 android:layout_height="wrap_content"
48.                 android:layout_marginStart="4dp"
49.                 android:text="Temperatura mínima:"
50.                 android:textColor="@color/colorPrimary"
51.                 android:textSize="16dp" />
52.
53.             <TextView
54.                 android:id="@+id/tempMin"
55.                 android:layout_width="36dp"
56.                 android:layout_height="wrap_content"
57.                 android:textColor="@color/colorPrimary"
58.                 android:textSize="16dp" />
```

```
59.
60.         </TableRow>
61.
62.         <!--Tercera fila-->
63.         <TableRow
64.             android:layout_width="match_parent"
65.             android:layout_height="match_parent"
66.             android:layout_gravity="center"
67.             android:layout_marginTop="1dp"
68.             android:background="@color/colorGris"
69.             android:padding="5dp">
70.
71.             <TextView
72.                 android:layout_width="wrap_content"
73.                 android:layout_height="wrap_content"
74.                 android:layout_marginStart="4dp"
75.                 android:text="Temperatura máxima:"
76.                 android:textColor="@color/colorPrimary"
77.                 android:textSize="16dp" />
78.
79.             <TextView
80.                 android:id="@+id/tempMax"
81.                 android:layout_width="36dp"
82.                 android:layout_height="wrap_content"
83.                 android:textColor="@color/colorPrimary"
84.                 android:textSize="16dp" />
85.         </TableRow>
86.
87.         <!--Tercera fila-->
88.         <TableRow
89.             android:layout_width="match_parent"
90.             android:layout_height="match_parent"
91.             android:layout_gravity="center"
92.             android:layout_marginTop="1dp"
93.             android:background="@color/colorGris"
94.             android:padding="5dp">
95.
96.             <TextView
97.                 android:layout_width="wrap_content"
98.                 android:layout_height="wrap_content"
99.                 android:layout_marginStart="4dp"
100.                android:text="Temperatura media:"
101.                android:textColor="@color/colorPrimary"
102.                android:textSize="16dp" />
103.
104.            <TextView
105.                android:id="@+id/tempMedia"
106.                android:layout_width="36dp"
107.                android:layout_height="wrap_content"
108.                android:textColor="@color/colorPrimary"
109.                android:textSize="16dp" />
110.        </TableRow>
111.
112.        <!--Quinta fila-->
113.        <TableRow
114.            android:layout_width="match_parent"
115.            android:layout_height="wrap_content"
116.            android:layout_gravity="center"
117.            android:layout_marginTop="1dp"
118.            android:background="@color/colorGris"
119.            android:padding="5dp">
```

```
120.
121.             <TextView
122.                 android:layout_width="115dp"
123.                 android:layout_height="wrap_content"
124.                 android:layout_marginStart="4dp"
125.                 android:text="Tiempo funcionamiento caldera:"
126.                 android:textColor="@color/colorPrimary"
127.                 android:textSize="16dp" />
128.
129.             <TextView
130.                 android:id="@+id/horometro"
131.                 android:layout_width="36dp"
132.                 android:layout_height="40dp"
133.                 android:textColor="@color/colorPrimary"
134.                 android:textSize="16dp" />
135.
136.
137.         </TableRow>
138.     </TableLayout>
139.
140. </LinearLayout>
```

8.4.14. fragment_ajustes.xml

```
1. <?xml version="1.0" encoding="utf-8"?>
2.     <ScrollView
3.     xmlns:android="http://schemas.android.com/apk/res/android"
4.     xmlns:app="http://schemas.android.com/apk/res-auto"
5.     xmlns:tools="http://schemas.android.com/tools"
6.     android:orientation="vertical"
7.     android:layout_width="match_parent"
8.     android:layout_height="match_parent"
9.     android:background="@color/colorGris_claro"
10.      app:layout_behavior="@string/appbar_scrolling_view_behavior"
11.      tools:context=".AjustesFragment">
12.
13.      <LinearLayout
14.          android:layout_width="match_parent"
15.          android:layout_height="wrap_content"
16.          android:orientation="vertical">
17.
18.          <!--Tabla de ajustes PRINCIPALES-->
19.          <TableLayout
20.              android:id="@+id/ajustesPrincipales"
21.              android:layout_width="match_parent"
22.              android:layout_height="wrap_content"
23.              android:layout_marginTop="4dp"
24.              android:background="@color/colorGris_claro">
25.
26.              <!--Primera fila-->
27.              <TableRow
28.                  android:layout_width="match_parent"
29.                  android:layout_height="match_parent"
30.                  android:background="@color/colorGris_main"
31.                  android:padding="5dp"
32.                  tools:visibility="visible">
33.
34.                  <TextView
35.                      android:id="@+id/textView"
36.                      android:layout_width="181dp"
37.                      android:layout_height="wrap_content"
38.                      android:layout_marginStart="4dp"
39.                      android:text="PRINCIPALES"
40.                      android:textAllCaps="true"
41.                      android:textColor="@color/colorPrimaryDark"
42.                      android:textSize="16dp"
43.                      android:textStyle="italic" />
44.              </TableRow>
45.
46.              <!--Segunda fila-->
47.              <TableRow
48.                  android:layout_width="match_parent"
49.                  android:layout_height="wrap_content"
50.                  android:layout_gravity="center"
51.                  android:layout_marginTop="1dp"
52.                  android:background="@color/colorGris"
53.                  android:padding="5dp">
54.
55.                  <TextView
56.                      android:layout_width="123dp"
57.                      android:layout_height="wrap_content"
58.                      android:layout_marginStart="4dp"
```

```
59.                     android:text="Estación año"
60.                     android:textColor="@color/colorPrimary"
61.                     android:textSize="16dp" />
62.
63.                 <TextView
64.                     android:layout_width="36dp"
65.                     android:layout_height="wrap_content"
66.                     android:textColor="@color/colorPrimary"
67.                     android:textSize="16dp" />
68.
69.                 <ImageView
   app:srcCompat="@drawable/ic_modo_invierno_24dp" />
70.
71.                 <Switch
72.                     android:id="@+id/switch_EstacionAno"
73.                     android:layout_width="wrap_content"
74.                     android:layout_height="wrap_content"/>
75.
76.                 <ImageView
77.                     android:layout_width="wrap_content"
78.                     app:srcCompat="@drawable/ic_modo_verano_24dp" />
79.             </TableRow>
80.
81.             <!--Tercera fila-->
82.             <TableRow
83.                 android:layout_width="match_parent"
84.                 android:layout_height="match_parent"
85.                 android:layout_gravity="center"
86.                 android:layout_marginTop="1dp"
87.                 android:background="@color/colorGris"
88.                 android:padding="5dp">
89.
90.                 <TextView
91.                     android:layout_width="wrap_content"
92.                     android:layout_height="wrap_content"
93.                     android:layout_marginStart="4dp"
94.                     android:text="Modo trabajo"
95.                     android:textColor="@color/colorPrimary"
96.                     android:textSize="16dp" />
97.
98.                 <TextView
99.                     android:layout_width="36dp"
100.                    android:layout_height="wrap_content"
101.                    android:textColor="@color/colorPrimary"
102.                    android:textSize="16dp" />
103.
104.                <TextView
105.                    android:layout_width="wrap_content"
106.                    android:layout_height="wrap_content"
107.                    android:layout_gravity="center"
108.                    android:text="AUT"
109.                    android:textColor="@color/colorPrimary"
110.                    android:textSize="16dp" />
111.
112.                <Switch
113.                    android:id="@+id/switchModoTrabajo"
114.                    android:layout_width="wrap_content"
115.                    android:layout_height="wrap_content" />
116.
117.                <TextView
118.                    android:layout_width="wrap_content"
```

```
119.                     android:layout_height="wrap_content"
120.                     android:layout_gravity="center"
121.                     android:text="MAN"
122.                     android:textColor="@color/colorPrimary"
123.                     android:textSize="16dp" />
124.             </TableRow>
125.
126.             <!--Cuarta fila-->
127.             <TableRow
128.                 android:layout_width="match_parent"
129.                 android:layout_height="wrap_content"
130.                 android:layout_gravity="center"
131.                 android:layout_marginTop="1dp"
132.                 android:background="@color/colorGris"
133.                 android:padding="5dp">
134.
135.                 <TextView
136.                     android:layout_width="115dp"
137.                     android:layout_height="wrap_content"
138.                     android:layout_marginStart="4dp"
139.                     android:text="Modo eco:"
140.                     android:textColor="@color/colorPrimary"
141.                     android:textSize="16dp" />
142.
143.                 <TextView
144.                     android:layout_width="36dp"
145.                     android:layout_height="wrap_content"
146.                     android:textColor="@color/colorPrimary"
147.                     android:textSize="16dp" />
148.
149.                 <TextView
150.                     android:layout_width="36dp"
151.                     android:layout_height="wrap_content"
152.                     android:textColor="@color/colorPrimary"
153.                     android:textSize="16dp" />
154.
155.                 <EditText
156.                     android:id="@+id/tempEcoMode"
157.                     android:layout_width="30dp"
158.                     android:layout_height="wrap_content"
159.                     android:ems="10"
160.                     android:gravity="center"
161.                     android:inputType="number"
162.                     android:shadowColor="@color/colorPrimary"
163.                     android:text="15"
164.                     android:textColor="@color/colorPrimary"
165.                     android:textSize="16dp" />
166.
167.                 <TextView
168.                     android:layout_width="wrap_content"
169.                     android:layout_height="wrap_content"
170.                     android:text="°C"
171.                     android:textColor="@color/colorPrimary"
172.                     android:textSize="16dp" />
173.
174.             </TableRow>
175.
176.             <!--Quinta fila-->
177.             <TableRow
178.                 android:id="@+id/rowCasa"
179.                 android:layout_width="match_parent"
```

```
180.                android:layout_height="wrap_content"
181.                android:layout_gravity="center"
182.                android:layout_marginTop="1dp"
183.                android:background="@color/colorGris"
184.                android:padding="5dp">
185.
186.                <TextView
187.                    android:layout_width="wrap_content"
188.                    android:layout_height="wrap_content"
189.                    android:layout_marginStart="4dp"
190.                    android:text="¿Estás en casa?"
191.                    android:textColor="@color/colorPrimary"
192.                    android:textSize="16dp" />
193.
194.                <TextView
195.                    android:layout_width="36dp"
196.                    android:layout_height="wrap_content"
197.                    android:textColor="@color/colorPrimary"
198.                    android:textSize="16dp" />
199.
200.                <TextView
201.                    android:layout_width="wrap_content"
202.                    android:layout_height="wrap_content"
203.                    android:layout_gravity="center"
204.                    android:text="SÍ"
205.                    android:textColor="@color/colorPrimary"
206.                    android:textSize="16dp" />
207.
208.                <Switch
209.                    android:id="@+id/switchCasa"
210.                    android:layout_width="wrap_content"
211.                    android:layout_height="wrap_content" />
212.
213.                <TextView
214.                    android:layout_width="wrap_content"
215.                    android:layout_height="wrap_content"
216.                    android:layout_gravity="center"
217.                    android:text="NO"
218.                    android:textColor="@color/colorPrimary"
219.                    android:textSize="16dp" />
220.            </TableRow>
221.
222.            <!--Sexta fila-->
223.            <TableRow
224.                android:id="@+id/rowVacaciones"
225.                android:layout_width="match_parent"
226.                android:layout_height="wrap_content"
227.                android:layout_gravity="center"
228.                android:layout_marginTop="1dp"
229.                android:background="@color/colorGris"
230.                android:padding="5dp">
231.
232.                <TextView
233.                    android:layout_width="wrap_content"
234.                    android:layout_height="wrap_content"
235.                    android:layout_marginStart="4dp"
236.                    android:text="Modo vacaciones"
237.                    android:textColor="@color/colorPrimary"
238.                    android:textSize="16dp" />
239.
240.                <TextView
```

```
241.                     android:layout_width="36dp"
242.                     android:layout_height="wrap_content"
243.                     android:textColor="@color/colorPrimary"
244.                     android:textSize="16dp" />
245.
246.                 <TextView
247.                     android:layout_width="wrap_content"
248.                     android:layout_height="wrap_content"
249.                     android:layout_gravity="center"
250.                     android:text="OFF"
251.                     android:textColor="@color/colorPrimary"
252.                     android:textSize="16dp" />
253.
254.                 <Switch
255.                     android:id="@+id/switchModoVacaciones"
256.                     android:layout_width="wrap_content"
257.                     android:layout_height="wrap_content" />
258.
259.                 <TextView
260.                     android:layout_width="wrap_content"
261.                     android:layout_height="wrap_content"
262.                     android:layout_gravity="center"
263.                     android:text="ON"
264.                     android:textColor="@color/colorPrimary"
265.                     android:textSize="16dp" />
266.             </TableRow>
267.
268.         </TableLayout>
269.
270.         <!--Tabla para ocultar y visibilizar fila-->
271.         <TableLayout
272.             android:id="@+id/tlFecha"
273.             android:layout_width="match_parent"
274.             android:layout_height="wrap_content"
275.             android:layout_gravity="center"
276.             android:background="@color/colorGris"
277.             android:padding="5dp"
278.             android:visibility="gone">
279.             <!--Única fila-->
280.             <TableRow
281.                 android:layout_width="match_parent"
282.                 android:layout_height="wrap_content"
283.                 android:gravity="center"
284.                 android:padding="5dp">
285.
286.                 <ImageView
287.                     android:id="@+id/imgFecha"
288.                     android:tint="@color/colorAccent"
289.                     app:srcCompat="@drawable/ic_modo_vacaciones_24dp" />
290.
291.                 <TextView
292.                     android:layout_width="wrap_content"
293.                     android:layout_height="wrap_content"
294.                     android:layout_marginStart="8dp"
295.                     android:text="Fuera hasta:  "
296.                     android:textColor="@color/colorAccent"
297.                     android:textSize="16dp" />
298.
299.                 <TextView
300.                     android:id="@+id/tvFecha"
301.                     android:layout_width="wrap_content"
```

```
302.                     android:layout_height="wrap_content"
303.                     android:gravity="center"
304.                     android:textColor="@color/colorAccent"
305.                     android:textSize="16dp" />
306.
307.
308.             </TableRow>
309.
310.         </TableLayout>
311.
312.         <!--Tabla de ajustes del TERMOMETRO-->
313.         <TableLayout
314.             android:id="@+id/ajustesTermometro"
315.             android:layout_width="match_parent"
316.             android:layout_height="wrap_content"
317.             android:background="@color/colorGris_claro">
318.
319.             <!--Primera fila-->
320.             <TableRow
321.                 android:layout_width="match_parent"
322.                 android:layout_height="match_parent"
323.                 android:background="@color/colorGris_main"
324.                 android:padding="5dp">
325.
326.                 <TextView
327.                     android:layout_width="181dp"
328.                     android:layout_height="wrap_content"
329.                     android:layout_marginStart="4dp"
330.                     android:text="TERMÓMETRO"
331.                     android:textAllCaps="true"
332.                     android:textColor="@color/colorPrimaryDark"
333.                     android:textSize="16dp"
334.                     android:textStyle="italic" />
335.             </TableRow>
336.
337.             <!--Segunda fila-->
338.             <TableRow
339.                 android:layout_width="match_parent"
340.                 android:layout_height="wrap_content"
341.                 android:layout_gravity="center"
342.                 android:layout_marginTop="1dp"
343.                 android:background="@color/colorGris"
344.                 android:padding="5dp">
345.
346.                 <TextView
347.                     android:layout_width="123dp"
348.                     android:layout_height="wrap_content"
349.                     android:layout_marginStart="4dp"
350.                     android:text="Delta temperatura (Δtª)"
351.                     android:textColor="@color/colorPrimary"
352.                     android:textSize="16dp" />
353.
354.                 <TextView
355.                     android:layout_width="36dp"
356.                     android:layout_height="wrap_content"
357.                     android:textColor="@color/colorPrimary"
358.                     android:textSize="16dp" />
359.
360.                 <TextView
361.                     android:layout_width="36dp"
362.                     android:layout_height="wrap_content"
```

```
363.                    android:textColor="@color/colorPrimary"
364.                    android:textSize="16dp" />
365.
366.                <EditText
367.                    android:id="@+id/tempDelta"
368.                    android:layout_width="30dp"
369.                    android:layout_height="wrap_content"
370.                    android:ems="10"
371.                    android:gravity="center"
372.                    android:inputType="number"
373.                    android:shadowColor="@color/colorPrimary"
374.
375.                    android:text="1"
376.                    android:textColor="@color/colorPrimary"
377.                    android:textSize="16dp" />
378.
379.                <TextView
380.                    android:layout_width="wrap_content"
381.                    android:layout_height="wrap_content"
382.                    android:text="°C"
383.                    android:textColor="@color/colorPrimary"
384.                    android:textSize="16dp" />
385.            </TableRow>
386.
387.            <!--Tercera fila-->
388.            <TableRow
389.                android:layout_width="match_parent"
390.                android:layout_height="wrap_content"
391.                android:layout_gravity="center"
392.                android:layout_marginTop="1dp"
393.                android:background="@color/colorGris"
394.                android:padding="5dp">
395.
396.                <TextView
397.                    android:layout_width="wrap_content"
398.                    android:layout_height="wrap_content"
399.                    android:layout_marginStart="4dp"
400.                    android:text="Tiempo máx. enviar"
401.                    android:textColor="@color/colorPrimary"
402.                    android:textSize="16dp" />
403.
404.                <TextView
405.                    android:layout_width="36dp"
406.                    android:layout_height="wrap_content"
407.                    android:textColor="@color/colorPrimary"
408.                    android:textSize="16dp" />
409.
410.                <TextView
411.                    android:layout_width="36dp"
412.                    android:layout_height="wrap_content"
413.                    android:textColor="@color/colorPrimary"
414.                    android:textSize="16dp" />
415.
416.                <EditText
417.                    android:id="@+id/tiempoMaxEnviarSensor"
418.                    android:layout_width="30dp"
419.                    android:layout_height="wrap_content"
420.                    android:ems="10"
421.                    android:gravity="center"
422.                    android:inputType="number"
423.                    android:shadowColor="@color/colorPrimary"
```

```
424.                    android:text="60"
425.                    android:textColor="@color/colorPrimary"
426.                    android:textSize="16dp" />
427.
428.                <TextView
429.                    android:layout_width="wrap_content"
430.                    android:layout_height="wrap_content"
431.                    android:layout_gravity="center"
432.                    android:text="s"
433.                    android:textColor="@color/colorPrimary"
434.                    android:textSize="16dp" />
435.            </TableRow>
436.
437.            <!--Cuarta fila-->
438.            <TableRow
439.                android:layout_width="match_parent"
440.                android:layout_height="wrap_content"
441.                android:layout_gravity="center"
442.                android:layout_marginTop="1dp"
443.                android:background="@color/colorGris"
444.                android:padding="5dp">
445.
446.                <TextView
447.                    android:layout_width="115dp"
448.                    android:layout_height="wrap_content"
449.                    android:layout_marginStart="4dp"
450.                    android:text="Tiempo rastreo"
451.                    android:textColor="@color/colorPrimary"
452.                    android:textSize="16dp" />
453.
454.                <TextView
455.                    android:layout_width="36dp"
456.                    android:layout_height="wrap_content"
457.                    android:textColor="@color/colorPrimary"
458.                    android:textSize="16dp" />
459.
460.                <TextView
461.                    android:layout_width="36dp"
462.                    android:layout_height="wrap_content"
463.                    android:textColor="@color/colorPrimary"
464.                    android:textSize="16dp" />
465.
466.                <EditText
467.                    android:id="@+id/tiempoRastreo"
468.                    android:layout_width="30dp"
469.                    android:layout_height="wrap_content"
470.                    android:ems="10"
471.                    android:gravity="center"
472.                    android:inputType="number"
473.                    android:shadowColor="@color/colorPrimary"
474.                    android:text="2"
475.                    android:textColor="@color/colorPrimary"
476.                    android:textSize="16dp" />
477.
478.                <TextView
479.                    android:layout_width="wrap_content"
480.                    android:layout_height="wrap_content"
481.                    android:layout_gravity="center"
482.                    android:text="s"
483.                    android:textColor="@color/colorPrimary"
484.                    android:textSize="16dp" />
```

```
485.
486.             </TableRow>
487.         </TableLayout>
488.
489.         <!--Tabla de ajustes del ACTUADOR-->
490.         <TableLayout
491.             android:id="@+id/ajustesActuador"
492.             android:layout_width="match_parent"
493.             android:layout_height="wrap_content"
494.             android:background="@color/colorGris_claro">
495.
496.             <!--Primera fila-->
497.             <TableRow
498.                 android:layout_width="match_parent"
499.                 android:layout_height="match_parent"
500.                 android:background="@color/colorGris_main"
501.                 android:padding="5dp">
502.
503.                 <TextView
504.                     android:layout_width="181dp"
505.                     android:layout_height="wrap_content"
506.                     android:layout_marginStart="4dp"
507.                     android:text="ACTUADOR"
508.                     android:textAllCaps="true"
509.                     android:textColor="@color/colorPrimaryDark"
510.                     android:textSize="16dp"
511.                     android:textStyle="italic" />
512.             </TableRow>
513.
514.             <!--Segunda fila-->
515.             <TableRow
516.                 android:layout_width="match_parent"
517.                 android:layout_height="wrap_content"
518.                 android:layout_gravity="center"
519.                 android:layout_marginTop="1dp"
520.                 android:background="@color/colorGris"
521.                 android:padding="5dp">
522.
523.                 <TextView
524.                     android:layout_width="wrap_content"
525.                     android:layout_height="wrap_content"
526.                     android:layout_marginStart="4dp"
527.                     android:text="Tiempo máx. enviar"
528.                     android:textColor="@color/colorPrimary"
529.                     android:textSize="16dp" />
530.
531.                 <TextView
532.                     android:layout_width="36dp"
533.                     android:layout_height="wrap_content"
534.                     android:textColor="@color/colorPrimary"
535.                     android:textSize="16dp" />
536.
537.                 <TextView
538.                     android:layout_width="36dp"
539.                     android:layout_height="wrap_content"
540.                     android:textColor="@color/colorPrimary"
541.                     android:textSize="16dp" />
542.
543.                 <EditText
544.                     android:id="@+id/tiempoMaxEnviarActuador"
545.                     android:layout_width="30dp"
```

```
546.                     android:layout_height="wrap_content"
547.                     android:ems="10"
548.                     android:gravity="center"
549.                     android:inputType="number"
550.                     android:shadowColor="@color/colorPrimary"
551.                     android:text="60"
552.                     android:textColor="@color/colorPrimary"
553.                     android:textSize="16dp" />
554.
555.                 <TextView
556.                     android:layout_width="wrap_content"
557.                     android:layout_height="wrap_content"
558.                     android:layout_gravity="center"
559.                     android:text="s"
560.                     android:textColor="@color/colorPrimary"
561.                     android:textSize="16dp" />
562.             </TableRow>
563.
564.
565.         </TableLayout>
566.     </LinearLayout>
567. </ScrollView>
```

8.4.15. dialogo_conexion.xml

```
1. <?xml version="1.0" encoding="utf-8"?>
2. <LinearLayout
3.     xmlns:android="http://schemas.android.com/apk/res/android"
4.     android:layout_width="300dp"
5.     android:layout_height="wrap_content"
6.     android:orientation="vertical"
7.     android:background="@drawable/edit_text_border"
8.     android:padding="@dimen/dialog_body">
9.
10.
11.      <TextView
12.          android:layout_width="wrap_content"
13.          android:layout_height="wrap_content"
14.          android:layout_alignParentTop="true"
15.          android:layout_centerHorizontal="true"
16.          android:gravity="center"
17.          android:layout_gravity="center_horizontal"
18.          android:textSize="18dp"
19.          android:text="FALLO DE CONEXION"
20.          android:textColor="@color/colorPrimaryDark"
21.          android:textStyle="bold"
22.          android:textAppearance="?android:attr/textAppearanceSmall" />
23.
24.      <TextView
25.          android:layout_width="wrap_content"
26.          android:layout_height="wrap_content"
27.          android:layout_alignParentTop="true"
28.          android:layout_centerHorizontal="true"
29.          android:gravity="center"
30.          android:paddingBottom="@dimen/padding_between"
31.          android:paddingTop="@dimen/padding_between"
32.          android:text="No se han podido obtener los datos solicitados. Ha
   sido imposible conectarse con el broker MQTT."
33.          android:textColor="@color/colorPrimaryDark"
34.          android:textAppearance="?android:attr/textAppearanceSmall" />
35.
36.      <TextView
37.          android:layout_width="wrap_content"
38.          android:layout_height="wrap_content"
39.          android:layout_alignParentTop="true"
40.          android:layout_centerHorizontal="true"
41.          android:gravity="center"
42.          android:paddingBottom="@dimen/padding_between"
43.          android:paddingTop="@dimen/padding_between"
44.          android:text="Presione ACEPTAR si desea reconectar:"
45.          android:textColor="@color/colorPrimaryDark"
46.          android:textAppearance="?android:attr/textAppearanceSmall" />
47.
48.      <LinearLayout
49.          android:layout_width="match_parent"
50.          android:layout_height="match_parent"
51.          android:gravity="center"
52.          android:orientation="horizontal"
53.          android:layout_marginTop="10dp">
54.
55.          <Button
56.              android:id="@+id/cancelar_boton"
57.              android:layout_width="110dp"
```

```
58.              android:layout_height="wrap_content"
59.              android:paddingBottom="@dimen/button_padding"
60.              android:paddingTop="@dimen/button_padding"
61.              android:backgroundTint="@color/colorPrimary"
62.              android:text="CANCELAR"
63.              android:textColor="@android:color/white" />
64.          <Button
65.              android:id="@+id/aceptar_boton"
66.              android:layout_width="110dp"
67.              android:layout_height="wrap_content"
68.              android:paddingBottom="@dimen/button_padding"
69.              android:paddingTop="@dimen/button_padding"
70.              android:backgroundTint="@color/colorPrimary"
71.              android:text="ACEPTAR"
72.              android:textColor="@android:color/white" />
73.      </LinearLayout>
74.  </LinearLayout>
```

8.4.16. dialogo_intervalos.xml

```
1. <?xml version="1.0" encoding="utf-8"?>
2. <LinearLayout
3.     xmlns:android="http://schemas.android.com/apk/res/android"
4.     xmlns:app="http://schemas.android.com/apk/res-auto"
5.     xmlns:tools="http://schemas.android.com/tools"
6.     android:layout_width="300dp"
7.     android:layout_height="wrap_content"
8.     android:orientation="vertical"
9.     android:background="@drawable/edit_text_border"
10.      android:padding="@dimen/dialog_body">
11.
12.
13.      <TextView
14.          android:id="@+id/titulo_dialog"
15.          android:layout_width="wrap_content"
16.          android:layout_height="wrap_content"
17.          android:layout_alignParentTop="true"
18.          android:layout_centerHorizontal="true"
19.          android:gravity="center"
20.          android:layout_gravity="center_horizontal"
21.          android:textSize="18dp"
22.          android:text="Intervalo 1 de DIARIO"
23.          android:textColor="@color/colorPrimaryDark"
24.          android:textStyle="bold"
25.          android:textAppearance="?android:attr/textAppearanceSmall" />
26.
27.      <TextView
28.          android:layout_width="wrap_content"
29.          android:layout_height="wrap_content"
30.          android:layout_alignParentTop="true"
31.          android:layout_centerHorizontal="true"
32.          android:gravity="center"
33.          android:paddingBottom="@dimen/padding_between"
34.          android:paddingTop="@dimen/padding_between"
35.          android:text="Introduce el setpoint de la temperatura:"
36.          android:textColor="@color/colorPrimaryDark"
37.          android:textAppearance="?android:attr/textAppearanceSmall" />
38.
39.
40.      <TableLayout
41.          android:layout_width="match_parent"
42.          android:layout_height="wrap_content">
43.
44.          <TableRow
45.              android:layout_width="wrap_content"
46.              android:layout_height="wrap_content"
47.              android:gravity="center_horizontal">
48.
49.              <ImageButton
50.                  android:id="@+id/botonDOWN_sp_dialog"
51.                  android:layout_width="41dp"
52.                  android:layout_height="24dp"
53.                  android:layout_marginTop="14dp"
54.                  android:background="@color/colorPrimaryDark"
55.                  android:contentDescription="@null"
56.                  app:srcCompat="@android:drawable/arrow_down_float" />
57.
58.              <TextView
```

```
59.                 android:id="@+id/tempSP_dialog"
60.                 android:layout_width="wrap_content"
61.                 android:layout_height="wrap_content"
62.                 android:layout_marginStart="16dp"
63.                 android:layout_marginEnd="16dp"
64.                 android:text="15°C"
65.                 android:textColor="@color/colorPrimaryDark"
66.                 android:textSize="40dp"
67.                 android:textStyle="bold" />
68.
69.             <ImageButton
70.                 android:id="@+id/botonUP_sp_dialog"
71.                 android:layout_width="41dp"
72.                 android:layout_height="24dp"
73.                 android:layout_marginTop="14dp"
74.                 android:background="@color/colorPrimaryDark"
75.                 android:contentDescription="@null"
76.                 app:srcCompat="@android:drawable/arrow_up_float" />
77.
78.         </TableRow>
79.
80.     </TableLayout>
81.
82.     <TextView
83.         android:layout_width="wrap_content"
84.         android:layout_height="wrap_content"
85.         android:layout_alignParentTop="true"
86.         android:layout_centerHorizontal="true"
87.         android:gravity="center"
88.         android:paddingBottom="@dimen/padding_between"
89.         android:paddingTop="@dimen/padding_between"
90.         android:text="Introduce la hora de inicio del intervalo:"
91.         android:textColor="@color/colorPrimaryDark"
92.         android:textAppearance="?android:attr/textAppearanceSmall" />
93.
94.     <!--Tabla SELECCIÓN HORAS-->
95.     <TableLayout
96.         android:layout_width="match_parent"
97.         android:layout_height="wrap_content">
98.
99.         <TableRow
100.            android:layout_width="wrap_content"
101.            android:layout_height="wrap_content"
102.            android:gravity="center_horizontal">
103.
104.            <!--Tabla botones HORAS-->
105.            <TableLayout
106.                android:layout_width="match_parent"
107.                android:layout_height="wrap_content">
108.
109.                <TableRow
110.                    android:layout_width="wrap_content"
111.                    android:layout_height="wrap_content"
112.                    android:gravity="center_horizontal">
113.
114.                    <ImageButton
115.                        android:id="@+id/botonUP_hora_dialog"
116.                        android:layout_width="41dp"
117.                        android:layout_height="24dp"
118.                        android:layout_marginTop="8dp"
119.                        android:background="@color/colorPrimaryDark"
```

```
120.                         android:contentDescription="@null"
121.                         app:srcCompat="@android:drawable/arrow_up_float"
   />
122.
123.                 </TableRow>
124.
125.                 <TableRow
126.                     android:layout_width="wrap_content"
127.                     android:layout_height="wrap_content"
128.                     android:gravity="center_horizontal">
129.
130.                     <ImageButton
131.                         android:id="@+id/botonDOWN_hora_dialog"
132.                         android:layout_width="41dp"
133.                         android:layout_height="24dp"
134.                         android:layout_marginTop="10dp"
135.                         android:background="@color/colorPrimaryDark"
136.                         android:contentDescription="@null"
137.
   app:srcCompat="@android:drawable/arrow_down_float" />
138.
139.                 </TableRow>
140.
141.             </TableLayout>
142.
143.             <TextView
144.                 android:id="@+id/tv_hora_dialog"
145.                 android:layout_width="wrap_content"
146.                 android:layout_height="wrap_content"
147.                 android:layout_alignParentTop="true"
148.                 android:layout_centerHorizontal="true"
149.                 android:clickable="true"
150.                 android:gravity="center"
151.                 android:paddingBottom="@dimen/padding_between"
152.                 android:paddingTop="@dimen/padding_between"
153.                 android:layout_marginStart="10dp"
154.                 android:text="21"
155.                 android:textSize="40dp"
156.                 android:textStyle="bold"
157.                 android:textColor="@color/colorPrimaryDark" />
158.
159.             <TextView
160.                 android:layout_width="wrap_content"
161.                 android:layout_height="wrap_content"
162.                 android:layout_alignParentTop="true"
163.                 android:layout_centerHorizontal="true"
164.                 android:clickable="true"
165.                 android:gravity="center"
166.                 android:paddingBottom="@dimen/padding_between"
167.                 android:paddingTop="@dimen/padding_between"
168.                 android:text=":"
169.                 android:textSize="40dp"
170.                 android:textStyle="bold"
171.                 android:textColor="@color/colorPrimaryDark" />
172.
173.             <TextView
174.                 android:id="@+id/tv_minutos_dialog"
175.                 android:layout_width="wrap_content"
176.                 android:layout_height="wrap_content"
177.                 android:layout_alignParentTop="true"
178.                 android:layout_centerHorizontal="true"
```

```
179.                 android:clickable="true"
180.                 android:gravity="center"
181.                 android:paddingBottom="@dimen/padding_between"
182.                 android:paddingTop="@dimen/padding_between"
183.                 android:text="21"
184.                 android:textSize="40dp"
185.                 android:textStyle="bold"
186.                 android:textColor="@color/colorPrimaryDark" />
187.
188.             <!--Tabla botones MINUTOS-->
189.             <TableLayout
190.                 android:layout_width="match_parent"
191.                 android:layout_height="wrap_content"
192.                 android:layout_marginStart="10dp">
193.
194.
195.                 <TableRow
196.                     android:layout_width="wrap_content"
197.                     android:layout_height="wrap_content"
198.                     android:gravity="center_horizontal">
199.
200.                     <ImageButton
201.                         android:id="@+id/botonUP_min_dialog"
202.                         android:layout_width="41dp"
203.                         android:layout_height="24dp"
204.                         android:layout_marginTop="8dp"
205.                         android:background="@color/colorPrimaryDark"
206.                         android:contentDescription="@null"
207.                         app:srcCompat="@android:drawable/arrow_up_float"
   />
208.
209.                 </TableRow>
210.
211.                 <TableRow
212.                     android:layout_width="wrap_content"
213.                     android:layout_height="wrap_content"
214.                     android:gravity="center_horizontal">
215.
216.                     <ImageButton
217.                         android:id="@+id/botonDOWN_min_dialog"
218.                         android:layout_width="41dp"
219.                         android:layout_height="24dp"
220.                         android:layout_marginTop="10dp"
221.                         android:background="@color/colorPrimaryDark"
222.                         android:contentDescription="@null"
223.
   app:srcCompat="@android:drawable/arrow_down_float" />
224.
225.                 </TableRow>
226.
227.             </TableLayout>
228.
229.         </TableRow>
230.
231.     </TableLayout>
232.
233.     <LinearLayout
234.         android:layout_width="match_parent"
235.         android:layout_height="match_parent"
236.         android:gravity="center"
237.         android:orientation="horizontal"
```

```
238.         android:layout_marginTop="10dp">
239.
240.         <Button
241.             android:id="@+id/cancelar_boton"
242.             android:layout_width="110dp"
243.             android:layout_height="wrap_content"
244.             android:paddingBottom="@dimen/button_padding"
245.             android:paddingTop="@dimen/button_padding"
246.             android:backgroundTint="@color/colorPrimary"
247.             android:text="CANCELAR"
248.             android:textColor="@android:color/white" />
249.         <Button
250.             android:id="@+id/aceptar_boton"
251.             android:layout_width="110dp"
252.             android:layout_height="wrap_content"
253.             android:paddingBottom="@dimen/button_padding"
254.             android:paddingTop="@dimen/button_padding"
255.             android:backgroundTint="@color/colorPrimary"
256.             android:text="ACEPTAR"
257.             android:textColor="@android:color/white" />
258.     </LinearLayout>
259. </LinearLayout>
```

8.4.17. build.gradle

```
1. apply plugin: 'com.android.application'
2.
3. android {
4.     compileSdkVersion 28
5.     defaultConfig {
6.         applicationId "com.delac.delacsmart"
7.         minSdkVersion 23
8.         targetSdkVersion 28
9.         versionCode 1
10.          versionName "1.0"
11.          testInstrumentationRunner
   "android.support.test.runner.AndroidJUnitRunner"
12.      }
13.      buildTypes {
14.          release {
15.              minifyEnabled false
16.              proguardFiles getDefaultProguardFile('proguard-android-
   optimize.txt'), 'proguard-rules.pro'
17.          }
18.      }
19.  }
20.
21.  dependencies {
22.      implementation fileTree(dir: 'libs', include: ['*.jar'])
23.      implementation 'com.android.support:appcompat-v7:28.0.0'
24.      implementation 'com.android.support.constraint:constraint-
   layout:1.1.3'
25.      testImplementation 'junit:junit:4.12'
26.      androidTestImplementation 'com.android.support.test:runner:1.0.2'
27.      androidTestImplementation
   'com.android.support.test.espresso:espresso-core:3.0.2'
28.      implementation 'com.android.support:design:28.0.0'
29.      implementation 'com.android.support:support-v4:28.0.0'
30.
31.
32.      implementation
   'org.eclipse.paho:org.eclipse.paho.android.service:1.1.1'
33.      implementation
   'org.eclipse.paho:org.eclipse.paho.client.mqttv3:1.2.0'
34.  }
```