



UNIVERSIDAD DE JAÉN
Escuela Politécnica Superior de Jaén

Trabajo Fin de Grado

MONITORIZACIÓN Y CONTROL DEL CONSUMO ELÉCTRICO EN VIVIENDAS INTELIGENTES

Alumno: Francisco Arjona Gutiérrez

Tutor: Prof. D. Antonio Cano Ortega

Co-tutor: Prof. D. Francisco José Sánchez Sutil

Dpto: Departamento de Ingeniería Eléctrica

CANO
ORTEGA
ANTONIO -
26009131H

Firmado
digitalmente por
CANO ORTEGA
ANTONIO -
26009131H
Fecha: 2023.06.07
19:02:42 +02'00'

SANCHEZ
SUTIL
FRANCISCO
JOSE -
26004797P

Firmado digitalmente
por SANCHEZ SUTIL
FRANCISCO JOSE -
26004797P
Fecha: 2023.06.07
18:58:48 +02'00'

RESUMEN

Debido al gran uso de las tecnologías inalámbricas en nuestro día a día, era inevitable pensar que no se le iba a dar un uso doméstico con afán de facilitarnos la vida cotidiana y controlar el gasto energético que necesita la vida en el hogar, mejorando así el impacto económico y medioambiental.

Este proyecto busca utilizar estas tecnologías en la domótica de un hogar mediante dispositivos comerciales de bajo coste. Estos dispositivos han sido programados para actuar en función de un hogar bajo ciertas circunstancias y escenarios para controlar el uso energético de distintos aparatos electrónicos, siendo estas circunstancias modificables a antojo del usuario mediante una app móvil mientras se lleva un registro energético.

Este registro se lleva a cabo utilizando el almacenamiento en la nube, un recurso que cada vez está más a la orden del día y nos va a permitir en todo momento tener un control desde fuera del hogar.

ABSTRACT

Due to the great use of wireless technologies in our day-to-day life, it was inevitable to think that it was not going to be used for domestic purposes with the aim of facilitating our daily lives and controlling the energy expenditure that life at home needs, thus improving the economic and environmental impact.

This project seeks to use these technologies in home automation through low-cost commercial devices. These devices have been programmed to work according to a home under certain circumstances and scenarios to control the energy use of different electronic devices, these circumstances being modifiable at the user's whim through a mobile app while keeping an energy record.

This record is carried out using cloud storage, a resource that is becoming more and more the order of the day and will allow us to have control from outside the home at all times.

Índice

RESUMEN	2
ABSTRACT	2
1. INTRODUCCIÓN	6
1.1 Contexto y motivación	6
1.2 Objetivos	8
2. ANTECEDENTES	8
2.1 Tecnologías inalámbricas	8
2.2 Actuadores	10
2.3 Sensores	12
3. HERRAMIENTAS RELACIONADAS	14
3.1 Wi-Fi	15
3.2 Firebase	15
3.3 MIT App Inventor	16
3.4 Arduino IDE	17
3.5 KiCad.....	17
3.6 Wemos D1 mini.....	18
3.7 Dimmer.....	19
3.8 Relé.....	20
3.9 Shield de humedad y temperatura	20
3.10 LDR.....	20
3.11 Multímetro	21
3.12 Alimentación	21
3.13 Otros materiales.....	22
4. METODOLOGÍA.....	22
4.1 Creación de proyectos.....	23
4.2 Módulos.....	35
4.3 Diseño de PCBs.....	46
5. PRUEBAS Y RESULTADOS DEL PROYECTO	56
6. CONCLUSIONES	67
BIBLIOGRAFÍA.....	68
ANEXO	71
Presupuesto	71
Programación	73

Índice de ilustraciones

Ilustración 2.1. Señales de ondas analógica y digital.....	13
Ilustración 3.1. Placa Wemos D1 mini.....	18
Ilustración 3.2. Guía visual de los pines de la placa Wemos D1 mini.....	19
Ilustración 3.3. Conexión del módulo PZEM-004T.....	21
Ilustración 4.1. Preferencias de Arduino IDE.....	23
Ilustración 4.2. Pantalla de bienvenida de Firebase.....	24
Ilustración 4.3. Pantalla de selección de Google Analytics en Firebase.....	24
Ilustración 4.4. Recorte de pantalla del menú de Firebase.....	25
Ilustración 4.5. Botón de selección de creación de nueva base de datos en Firebase.....	25
Ilustración 4.6. Selección de ubicación de la base de datos de Firebase.....	26
Ilustración 4.7. Preferencias de seguridad de la base de datos de Firebase.....	26
Ilustración 4.8. Pantalla de Fingerprints.....	27
Ilustración 4.9. Ubicación de la base de datos.....	27
Ilustración 4.10. Extracto de código de FirebaseHttpClient.h.....	28
Ilustración 4.11. Base de datos en Firebase del proyecto.....	28
Ilustración 4.12. Información detallada de las habitaciones en Firebase (parte 1).....	29
Ilustración 4.13. Información detallada de las habitaciones en Firebase (parte 2).....	29
Ilustración 4.14. Creación de un nuevo proyecto en MIT App Inventor.....	30
Ilustración 4.15. Selección de nombre del proyecto en MIT App Inventor.....	30
Ilustración 4.16. Pantalla de Login de la aplicación móvil.....	31
Ilustración 4.17. Pantalla de Inicio de la aplicación móvil.....	31
Ilustración 4.18. Pantalla de Ajustes Smart Home de la aplicación móvil.....	32
Ilustración 4.19. Pantalla de Salón de la aplicación móvil.....	33
Ilustración 4.20. Pantalla de Cuarto de baño de la aplicación móvil.....	34
Ilustración 4.21. Pantalla de ConsumoNow de la aplicación móvil.....	34
Ilustración 4.22. Cara A del módulo de Iluminación.....	35
Ilustración 4.23. Cara B del módulo de Iluminación.....	36
Ilustración 4.24. Diagrama de flujo de la iluminación.....	37
Ilustración 4.25. Cara A del módulo del sensor de iluminación.....	38
Ilustración 4.26. Cara B del módulo del sensor de iluminación.....	38
Ilustración 4.27. Diagrama de flujo del sensor de iluminación.....	39
Ilustración 4.28. Cara A del módulo de climatización.....	41
Ilustración 4.29. Cara B del módulo de climatización.....	41
Ilustración 4.30. Diagrama de flujo de climatización.....	42
Ilustración 4.31. Cara A del módulo del enchufe inteligente.....	43
Ilustración 4.32. Cara B del módulo del enchufe inteligente.....	44
Ilustración 4.33. Diagrama de flujo del enchufe inteligente.....	45
Ilustración 4.34. Editor de símbolos de KiCad (PZEM-004T).....	46
Ilustración 4.35. Editor de huellas de KiCad (PZEM-004T).....	47
Ilustración 4.36. Propiedades del símbolo de KiCad.....	48
Ilustración 4.37. Esquema de conexionado eléctrico del enchufe inteligente.....	48
Ilustración 4.38. Esquema de conexionado eléctrico de la climatización.....	49
Ilustración 4.39. Esquema de conexionado eléctrico de la iluminación.....	49
Ilustración 4.40. Esquema de conexionado eléctrico del sensor de luminosidad.....	50
Ilustración 4.41. Layout Editor del conexionado eléctrico del enchufe inteligente.....	50
Ilustración 4.42. Modelo 2D del enchufe inteligente – cara B.....	51
Ilustración 4.43. Modelo 2D del enchufe inteligente – cara A.....	51
Ilustración 4.44. Layout Editor del conexionado eléctrico de la climatización.....	52
Ilustración 4.45. Modelo 2D de la climatización – cara A.....	52
Ilustración 4.46. Modelo 2D de la climatización – cara B.....	52
Ilustración 4.47. Layout Editor del conexionado eléctrico de la iluminación.....	53
Ilustración 4.48. Modelo 2D de la iluminación – cara B.....	53
Ilustración 4.49. Modelo 2D de la iluminación – cara A.....	53
Ilustración 4.50. Layout Editor del conexionado eléctrico del sensor de luminosidad.....	54

Ilustración 4.51. Modelo 2D del sensor de luminosidad – cara B.....	54
Ilustración 4.52. Modelo 2D del sensor de luminosidad – cara A.....	55
Ilustración 5.1. Esquema de conexionado del módulo PZEM-004T con Arduino.....	56
Ilustración 5.2. Configuración de la prueba con el secador de pelo	56
Ilustración 5.3. Activación del modo automático en Firebase.....	58
Ilustración 5.4. Activación de las demás variables de LuzBath en Firebase.....	59
Ilustración 5.5. Apagado de luminaria por valores altos del sensor en Firebase.....	59
Ilustración 5.6. Estado de las variables antes de la prueba con el enchufe en Firebase.....	60
Ilustración 5.7. Activación de TeleSalon en Firebase.....	61
Ilustración 5.8. Activación del resto de variables y cálculo de la potencia de la tele en..... Firebase.....	61
Ilustración 5.9. Cambio de la potencia contratada para realizar la prueba	62
Ilustración 5.10. Desactivación de las variables debido a una potencia consumida..... elevada.....	62
Ilustración 5.11. Pestaña ‘Ajustes Smart Home’ de la app con el salón en menor prioridad.....	63
Ilustración 5.12. Gráfica del comportamiento de los enchufes con respecto a la potencia.....	65
Ilustración 5.13. Gráfica de la potencia en el minuto 17:52 de la prueba en ConsumoNow de la app.....	66

Índice de tablas

Tabla 3.1. Configuración de pines de la placa Wemos D1 mini.....	19
Tabla 4.1. Estado del aire acondicionado según los relés.....	40
Tabla 5.1. Datos obtenidos de la prueba con el secador de pelo.....	57
Tabla 5.2. Valores de potencia de cada enchufe en cada minuto.....	64
Tabla A.1. Presupuesto del módulo de iluminación.....	71
Tabla A.2. Presupuesto del módulo del sensor de iluminación.....	71
Tabla A.3. Presupuesto del módulo del enchufe inteligente.....	72
Tabla A.4. Presupuesto del módulo de climatización.....	72

1. INTRODUCCIÓN

1.1 Contexto y motivación

Se va a comenzar introduciendo los pilares básicos en los que se va a centrar en este proyecto de control energético utilizando la domótica en el hogar. Uno de los muchos puntos clave que van a ser incluidos en este proyecto es el llamado IoT (“Internet of Things” – “Internet de las cosas”), que se explicarán más adelante y será la base para que nuestra casa se convierta en una Smart Home.

El avance significativo de la tecnología y del uso integrado cada vez mayor en la vida cotidiana ha servido como influencia para basar este proyecto, intentando buscar soluciones al alcance de cualquier bolsillo y fácil de entender para cualquier público.

Además, este proyecto busca también en generar un interés en estas tecnologías para que sea propicio un avance en estas y vaya siendo cada vez más accesible y ofrezca unas mayores ventajas.

El término de Smart Home (hogar inteligente en español) se puede definir como aquellos hogares que incorporen un sistema capaz de controlar y automatizar muchas de las tareas que dependían de la acción humana. Con este sistema, también se podrá recopilar y analizar datos para mejorar la eficiencia energética y con ello, una mejora en el ahorro económico del hogar. En general, este tipo de hogares ofrecen unas mejoras en cuanto al hogar convencional como:

- **Mejora de la eficiencia energética**, ya que estos funcionan teniendo en cuenta las necesidades y pudiéndose programar para aprovechar al máximo los recursos tales como la luz diurna o apagar luces cuando no haya presencia alguna.
- **Mejora en el impacto económico**. Al mejorar la eficiencia energética, por ende, se reduce el impacto de la factura de luz.
- **Mejora en la comodidad** al poder controlar todos estos sistemas desde un móvil con tecnología inalámbrica y, al tenerlo de forma automática, poder dejar de estar pendientes de muchas tareas cotidianas del hogar.
- **Mejora en la seguridad**. Como mejora incluida en el anterior punto, el hecho de poder controlar el hogar de forma inalámbrica hace que, mediante el uso de sensores y cámaras, se puede tener controlado el hogar y despreocuparse cuando se esté fuera ya que desde el smartphone personal se podrá ver en todo momento el estado actual del hogar.
- **Mejora en la accesibilidad**, al poder dar el control, igualmente de forma inalámbrica, a personas que, por ejemplo, tengan una movilidad reducida. Igualmente, el hecho de automatizar un hogar hará de este para personas mayores un lugar más comprensible y fácil para ellos.

Como se ha comentado antes, la base de estas Smart Home es el llamado IoT o Internet de las cosas (“Internet of Things” en inglés). IoT ha surgido

recientemente como una de las tecnologías más importantes del siglo XXI. La comunicación continua entre personas, procesos y cosas ahora es posible gracias a la capacidad de conectar elementos comunes, como electrodomésticos, automóviles, termostatos y luces, a Internet a través de dispositivos integrados. Los elementos físicos pueden compartir y recopilar datos con poca interacción humana gracias a la computación de bajo costo, la nube, los grandes datos, el análisis y las tecnologías móviles. Los sistemas digitales pueden registrar, monitorear y modificar cada interacción entre entidades conectadas en el entorno hiperconectado de hoy. Los mundos físico y digital interactúan y funcionan juntos sin problemas.

Aunque el concepto de IoT ha existido por un largo tiempo, apenas se ha convertido en realidad gracias a una variedad de avances tecnológicos recientes. Algunas de sus características son:

- **Acceso a tecnología de sensores asequibles y de bajo consumo.** Más fabricantes ahora pueden usar la tecnología IoT gracias a sensores confiables y de precio razonable.
- **Conectividad.** Ahora es sencillo vincular sensores a la nube y otros dispositivos para una transferencia de datos efectiva gracias a la abundancia de protocolos de red para Internet.
- **Plataformas de computación en la nube.** Las plataformas en la nube están cada vez más disponibles, lo que permite a las empresas y los consumidores obtener la plataforma que necesitan para ascender sin tener que manejarlo todo.
- **Modelado analítico y aprendizaje automático.** Con mejoras en el análisis y el aprendizaje automático, así como el acceso a los grandes y variados volúmenes de datos almacenados en la nube, las empresas pueden adquirir conocimientos de forma más rápida y sencilla. Por un lado, el desarrollo de estas tecnologías relacionadas empuja los límites de IoT, mientras que, por otro lado, los datos generados por IoT también hacen crecer estas tecnologías.
- **IA (Inteligencia Artificial) conversacional.** El procesamiento del lenguaje natural (PLN) se ha puesto a disposición de los dispositivos IoT gracias a los avances en redes neuronales, tales como los asistentes digitales personales Alexa, Cortana y Siri, y ha mejorado su atractivo, viabilidad y asequibilidad para uso doméstico.

Esta agrupación e interconexión de dispositivos y objetos a través de una red (bien sea privada o Internet, la red de redes) donde son visibles y pueden interaccionar entre ellos. Esta interacción de máquina a máquina (interacción M2M -machine to machine-) requiere un protocolo que está sujeto a una serie de condiciones o requisitos, entre ellos una gran cantidad de dispositivos conectados, escalabilidad, mantener un acoplamiento débil entre dispositivos, requerir poca potencia de procesamiento para poder adaptarse a dispositivos con poca capacidad de cálculo, interoperabilidad para poder para funcionar en una gran variedad de dispositivos y lenguajes de programación, soportando una gran cantidad de comunicaciones simultáneas, y siendo posible una cantidad de

cosas diferentes. La información importante se intercambiará en un entorno seguro y, en el caso de la domótica, esta información controlará incluso los dispositivos físicos.

1.2 Objetivos

El objetivo principal de este proyecto es crear e implementar prototipos encargados de distintas funciones de un hogar inteligente a la vez que se lleva un control sobre el consumo. Además, en este trabajo se van a cumplir los siguientes objetivos:

- Hacer un estudio del estado del arte de las tecnologías implicadas en una Smart Home, optando así a la más adecuada para el proyecto.
- Una introducción a las tecnologías usadas por los prototipos del proyecto para manejar la información.
- Creación de prototipos utilizando dispositivos de bajo coste y entornos de programación de código abierto.
- Desarrollo de una aplicación móvil para la gestión de los prototipos.
- Programación de los microcontroladores Arduino para la gestión de los prototipos en base a los datos obtenidos por sensores y app.
- Subida y bajada de datos a la base de datos en la nube.
- Seguimiento del consumo desde la base de datos en internet y la app.
- Pruebas con el prototipo y obtención de datos.

2. ANTECEDENTES

Aunque la tecnología del hogar esté avanzada, ha pasado por diversos cambios desde sus inicios hasta llegar a ser lo que se conoce hoy en día. A continuación, se expondrán las diferentes tecnologías inalámbricas que se han utilizado, así como los diversos sensores y actuadores que han permitido este auge de la domótica.

2.1 Tecnologías inalámbricas

2.1.1 Wi-Fi

No se puede empezar a hablar de tecnologías inalámbricas sin mencionar primero a la tecnología Wi-Fi. Esta tecnología consiste en la transmisión de datos mediante ondas de radio y permite la conexión de multitud de dispositivos a Internet a través de un punto de acceso o conectarse entre ellos debido a la gran cantidad de dispositivos compatibles.

Esta tecnología utiliza un estándar IEEE 802.11ax desde el año 2019 (también conocido como Wifi 6) para transmitir las ondas en las frecuencias de 2.4 GHz y 5GHz.

Aunque esta tecnología tenga un gran alcance que otras y sea la preferida a la hora de interconectar dispositivos de domótica, el elevado uso de los aparatos electrónicos con esta tecnología de comunicación hace que se sature y vaya más despacio. Aún así, sigue siendo de las opciones más usadas en la domótica ya que asegura una compatibilidad total con los dispositivos sin necesidad de cables o de ninguna infraestructura.

2.1.2 Bluetooth

Se hace referencia a Bluetooth como el protocolo de comunicación creado específicamente para dispositivos de baja potencia ya que solo necesita un rango de emisión pequeño y se basa en transceptores económicos.

Sin embargo, esta tecnología solo cuenta con un alcance medio de unos 10 metros en interior o 100 metros en el exterior, pudiendo ser mayor en las versiones más recientes.

Aun así, el consumo de energía es muy pequeño y viéndose reducido en cada versión actualizada. Ofreciendo una alta accesibilidad, esta tecnología opera en la frecuencia de 2.4GHz, teniéndose en cuenta en el mundo de la domótica para conectar diversos dispositivos pequeños relacionados con IoT.

2.1.3 Zigbee

Zigbee es un protocolo de comunicación que permite a los dispositivos conectarse entre ellos sin tener que conectarse a una red wifi ya que opera en una frecuencia de ondas de radio de baja energía, lo que le permite tener una gran resistencia ante perturbaciones.

Otras de sus muchas características es que utiliza una tecnología de encriptación AES-128 para garantizar su seguridad, incluso con la posibilidad de conexión de hasta 64000 dispositivos en un rango comprendido entre los 10 y los 75 metros. A pesar de coexistir en la misma frecuencia que las anteriores, su funcionamiento no se ve afectado debido a su baja tasa de transmisión y a las características del estándar IEE 802.15.4.

Zigbee ofrece, además, un catálogo de unos 2500 dispositivos de 400 fabricantes distintos.

Este protocolo es ideal para la automatización de edificios que incluyan gran cantidad de dispositivos conectados ya que, si son pocos, tecnologías antes mencionadas serían más que suficientes.

2.1.4 Z-Wave

Protocolo de comunicación similar a Zigbee, empleando ondas de radio de baja potencia y mostrando buena resistencia a perturbaciones y a crecimiento de la instalación.

Sin embargo, lo que diferencia este protocolo de Zigbee es que este no posee un estándar abierto, reduciendo la compatibilidad entre dispositivos de diferentes marcas.

Z-Wave también utiliza una encriptación AES-128 y trabaja en un rango de 100 metros con posibilidad de conexión de hasta 232 dispositivos. Ofrece un catálogo de 2400 dispositivos aproximadamente de 700 fabricantes, siendo el líder del mercado mundial en el control inalámbrico.

2.1.5 Insteon (Tecnología X10)

Los sistemas domóticos X10 utilizan como medio de transmisión la red eléctrica y la radiofrecuencia para enviar las órdenes de control.

Es uno de los protocolos más antiguos, siendo poco fiable y careciendo de flexibilidad, por lo que, en 1997 surge Insteon.

Insteon utiliza dispositivos X10 LINC mejorados, siendo más fiables y flexibles y ofreciendo la simplicidad y bajo coste de su predecesor. La frecuencia es parecida a la de X10 (120 KHz), siendo esta de 131.65 KHz. Además de la frecuencia, también se diferencia con su antecesora en que los repetidores son de difusión simultánea y el protocolo Insteon es mucho más rápido. Los dispositivos Insteon son de doble vía y se respaldan mutuamente, detectando así cualquier error.

2.2 Actuadores

Los actuadores son dispositivos que reciben una orden de un regulador o controlador y modifican el estado de un elemento de un sistema. Los actuadores más comunes en la domótica son

2.2.1 Motores

Son capaces de girar mientras convierten la energía eléctrica en energía mecánica. La parte estacionaria de este motor está formada por un electroimán que es generado por el campo magnético que ejerce la fuerza sobre la parte móvil. Este último está compuesto por varias bobinas y se denomina rotor.

En la domótica, son bastante incluidos ya que activan accionamientos lineales o rotatorios como puede ser cierre y apertura de puertas, ventanas o compuertas de ventilación y motores de lavavajillas, lavadoras y demás electrodomésticos.

2.2.2 Dimmers

Son dispositivos capaces de regular la intensidad de luz. Estos dispositivos son de lo más común en la domótica ya que, regular correctamente las distintas bombillas del hogar para su iluminación, mejora la eficiencia energética considerablemente.

2.2.3 Relés

Es un interruptor eléctrico o electrónico basado en un circuito de acoplamiento que sirve como una interfaz de protección de aislamiento contra posibles problemas eléctricos al tiempo que permite el uso de potencias considerables para la carga a través

de señales de control modestas en la entrada. Suelen utilizarse para automatizar electrodomésticos ya que muchos artículos del hogar son incompatibles con varios protocolos de comunicación, por lo que esta automatización se basa en el control del suministro eléctrico.

2.2.4 Contactores

Están diseñados para funcionar como unos interruptores automáticos, muy parecido al relé, pero con corrientes y voltajes más grandes. Los contactores primarios son los que se utilizan para operar los circuitos de potencia de los conjuntos, como los que suministran electricidad a motores eléctricos o electrodomésticos.

2.2.5 Luminaria

Uno de los sistemas de una casa que más energía consume es la iluminación, por lo que es importante instalar luminarias adecuadas al espacio y utilizar tecnología que consuma la menor cantidad de energía. También puedes hacer un uso eficiente de las luces instalando sensores en los pasillos para que solo se enciendan cuando pase la gente o utilizando dimmers en las habitaciones o pasillos. Ser capaz de seleccionar la cantidad correcta de luz también conduce a una mayor comodidad al usar los controles de luminosidad.

2.2.6 Toldos y persianas

Las persianas y toldos en un sistema domótico están automatizados y pueden ser controlados por el controlador de la vivienda que, a menudo, están hechos de un motor tubular. Además de los interruptores junto a las persianas y toldos, también pueden incluir sensores de fin de carrera, controles por radiofrecuencia o mandos a distancia.

2.2.7 Aire acondicionado y calefacción

Se puede dividir en dos categorías: un sistema central que controla normalmente el flujo de aire de toda la casa, o un sistema particular que regula habitaciones específicas. El uso de aire, agua o electricidad puede afectar el sistema de calefacción. El aspecto más crucial del sistema de aire acondicionado de una casa es la zonificación, o dividir la casa en secciones con las temperaturas adecuadas en cada una. Si bien es importante ajustar cada uno de estos componentes para mayor comodidad y evitar el desperdicio de energía, este no es el factor más crítico. Si la calefacción es eléctrica, se utilizan calentadores. Estos aparatos calientan el ambiente utilizando el efecto Joule, en el que la potencia se pierde en forma de calor cuando la corriente pasa por una resistencia.

2.2.8 Cerraduras electrónicas

Junto con otros elementos de control de acceso que identifican al usuario, detectan o avisan de la intención de entrar, son actuadores utilizados para facilitar la entrada a viviendas y edificios. Los sistemas más básicos y ampliamente utilizados se

basan en electroimanes que se montan en la puerta o el marco y permiten que el pestillo se libere en respuesta a una señal eléctrica.

2.2.9 Electroválvulas

Son los dispositivos que habilitan o inhabilitan el paso de fluidos (agua, gas, gasóleo, etc.) a las múltiples instalaciones y maquinaria que de ellos dependen, como agua potable, agua caliente, calefacción, riego, piscinas y electrodomésticos. . . Son muy importantes porque, por ejemplo, con ellos se pueden elegir qué estancias de la casa calentar o, en caso de fuga, cerrar el suministro principal de agua de la casa.

2.2.10 Señalización luminosa y acústica

Este tipo de actuadores se utiliza para dar alarmas o informas de algún evento que está sucediendo en la vivienda como persuadir a posibles intrusos, usarse para funciones de aviso horario, presencia de visitas, llamadas telefónicas u otro tipo de eventualidades.

2.3 Sensores

Un sensor es una herramienta que puede generar una señal que se puede traducir en función de la cantidad de energía o física que desea monitorear. En otras palabras, un sensor es un objeto que tiene la capacidad de detectar, medir y proporcionar una salida asociada con la medición. Dos componentes cruciales componen los sensores: la entrada, que entra en contacto con la señal física a medir, y la salida, que proporciona información sobre la variable que se mide. La señal de salida puede contener la misma cantidad física que la señal de entrada o una diferente; por ejemplo, un sensor que mide la luminosidad puede producir una señal de salida eléctrica o luminosa.

En el caso de los sensores, toman datos de un proceso o del mundo físico y transforman la magnitud en una señal eléctrica para que los datos puedan ser analizados por el controlador o en otro lugar. Las señales eléctricas en los sensores se pueden dividir en dos categorías:

- Señal analógica: la salida del sensor puede presentar cualquiera valor, que es el valor medido.
- Señal digital: la salida del sensor solo puede presentar un valor alto o un valor bajo, siendo estos el 1 y 0 lógicos.

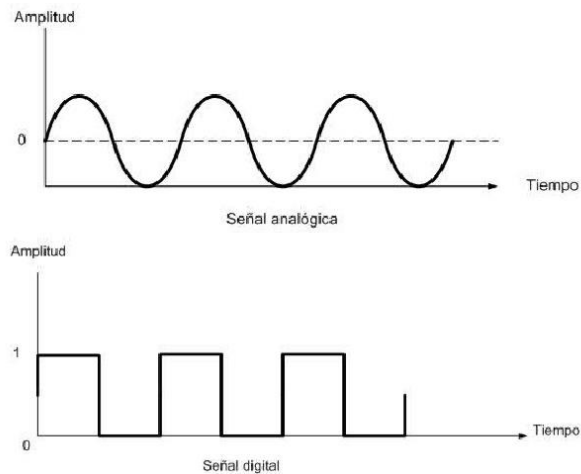


Ilustración 2.1. Señales de ondas analógica y digital

En la imagen se observa que el valor de la señal en la analógica puede tomar cualquiera de los valores comprendidos entre su máximo y mínimo mientras que, en la digital, solo puede tomar uno de los dos valores.

En cuanto a características, las más importantes que hacen que un sensor sea mejor son:

- Repetitividad: capacidad de dar el mismo resultado cuando la entrada es la misma, independientemente del tiempo o de la persona que lo mida.
- Exactitud: diferencia menor posible entre el valor medido y el valor real.
- Sensibilidad: similitud entre la salida y la entrada que ha provocado el cambio.
- Linealidad: ajuste de las medidas tomadas a una recta teórica.

Dependiendo de las medidas que se realizan o el objetivo final del sensor, estos se clasifican en diversos tipos:

2.3.1 Sensores de presencia

Son capaces de detectar cualquier movimiento. Entre sus posibles tareas se encuentra la de encender o apagar cualquier luminaria de la zona donde se coloque y la apertura o cierre de puertas automáticas. Otro uso común que se les da es para detectar cualquier tipo de intrusión, estando conectado generalmente a una alarma que alertará de alguna presencia no deseada.

2.3.2 Sensores de temperatura

Dan un registro de la temperatura de donde estén situados. Pueden medir tanto la temperatura de un dispositivo como la ambiental. En el uso de la domótica es muy importante para un uso eficiente de la energía tanto en calefacción como climatización.

2.3.3 Sensores de luminosidad

Detectan los niveles de luz presentes en el entorno. Muy importantes en la domótica para activar o desactivar la iluminación según la cantidad de luz diurna presente, por ejemplo. Funcionan con una resistencia variable según el nivel de luz.

2.3.4 Sensores de humedad

Suelen ser utilizados en la domótica para la parte exterior de las casas, sobre todo, para dar una información climatológica del ambiente y control sobre el suelo (bien para el césped o regadío). También se incluyen para climatizar las salas correctamente, conjugando con la temperatura de la habitación.

2.3.5 Sensores de presión

Suelen estar ligados a sistemas de bombeo de agua en sistemas de aire acondicionado y calefacción en casas donde estas instalaciones estén presentes. También son utilizados para sistemas de presencia.

2.3.6 Sensores de identificación

Desde sensores que leen tarjetas magnéticas con identificadores personales o llaves especiales hasta paneles numéricos, incluso controles biométricos con escáneres oculares, reconocimiento de voz y lectura de huella dactilar.

2.3.7 Detectores de compuestos en el aire

Los más conocidos entre estos detectores son los detectores de humos, los cuales están conectados a una alarma y un sistema contra incendios que evitan que se provoque un fuego al detectar la presencia de humo presente en el aire.

Además de detectar humo, existen detectores capaces de medir la calidad del aire y detectar fugas de cualquier gas nocivo como butano o propano o detectar concentraciones altas en el aire de CO debido a la combustión de cualquier chimenea, por ejemplo.

2.3.8 Detectores de fluidos

Pensados por si hay alguna fuga de agua, evitan que se siga inundando la zona cortando el suministro de agua y activando sistemas de evacuación de agua (en el caso de que los hubiese).

También son importantes en la domótica porque pueden detectar cuando está lloviendo para recoger los toldos, bajar persianas o desactivar el regadío de las plantas.

3. HERRAMIENTAS RELACIONADAS

A continuación, se va a detallar las tecnologías relacionadas con este proyecto y con la domótica en sí, así explicar la razón por la cual se han elegido y son las más convenientes.

3.1 Wi-Fi

De las tecnologías mencionadas en el anterior apartado, la que se ha utilizado en el proyecto y la más utilizada en el mundo de la domótica para este tipo de proyectos es la tecnología Wi-Fi.

Para el proyecto, se ha valorado el uso de las demás tecnologías inalámbricas. El uso de la tecnología X10 no era viable desde un principio ya que encarecería mucho el proyecto al tener que adquirir los dispositivos necesarios para transmitir la información por la red eléctrica.

El estándar usado con Z-Wave, al no ser público, encarecería igualmente el proyecto mientras que, por otro lado, se desean de tecnologías de bajo coste.

La tecnología Zigbee se podría utilizar para el proyecto, pero aumentaría la complejidad de este por lo que la decisión se queda entre las tecnologías inalámbricas más fáciles de manejar a nivel usuario.

Finalmente, se opta por elegir la tecnología Wi-Fi ya que la tecnología Bluetooth no es suficiente para abarcar toda la Smart Home y, además, al utilizar una base de datos en tiempo real, la cual explicaremos más adelante, es necesario la conexión Wi-Fi. Esta también permitirá tanto cambiar valores como activar desde fuera de casa los dispositivos conectados.

3.2 Firebase

Los desarrolladores de apps pueden crear aplicaciones iOS, Android y web mediante la plataforma de desarrollo de aplicaciones patrocinada por Google conocida como Firebase. Varias herramientas están disponibles en Firebase para monitorear análisis, informar y resolver errores de aplicaciones, así como desarrollar experimentos de marketing y productos.

Entre los servicios que brinda Firebase se encuentran los siguientes:

- **Analytics:** Google Analytics para Firebase ofrece informes gratuitos e ilimitados sobre hasta 500 eventos diferentes. Analytics proporciona información sobre el comportamiento de los usuarios en las aplicaciones de iOS y Android, lo que permite una mejor toma de decisiones sobre la mejora del rendimiento y la promoción de la aplicación.
- **Authentication:** Firebase Authentication facilita el desarrollo de un sistema de autenticación seguro para los desarrolladores y mejora las experiencias de inicio de sesión e incorporación de los usuarios. Con soporte para autenticación telefónica, cuentas de correo electrónico y contraseña, Google, Facebook, GitHub, Twitter y más, esta función proporciona una solución de identidad completa.

- **Crashlytics:** Firebase Crashlytics es un informe de fallas en tiempo real que ayuda a los desarrolladores a rastrear, priorizar y resolver problemas de estabilidad que reducen el calibre de sus productos. Con Crashlytics, los desarrolladores pueden dedicar más tiempo a crear funciones para sus aplicaciones que a administrar y solucionar fallas.
- **Cloud Messaging:** Firebase Cloud Messaging (FCM) es una aplicación de mensajería multiplataforma gratuita que permite a las empresas transmitir y recibir mensajes en iOS, Android y la web con confiabilidad.
- **Performance:** la herramienta Firebase Performance Monitoring proporciona a los desarrolladores información sobre las características de rendimiento de las aplicaciones de iOS y Android, lo que les permite identificar áreas y momentos en los que se podría mejorar el rendimiento de sus aplicaciones.
- **Test lab:** Es una infraestructura basada en la nube para probar aplicaciones. Los desarrolladores pueden probar rápida y fácilmente sus aplicaciones de iOS o Android en varias combinaciones de hardware. En la consola de Firebase, pueden ver los resultados, incluidos vídeos, imágenes y registros.
- **Realtime Database:** Firebase Realtime Database es una base de datos NoSQL alojada en la nube que permite a los usuarios sincronizar y almacenar datos en tiempo real. Cuando una aplicación se desconecta, todos los clientes aún pueden acceder a los datos, ya que se sincronizan constantemente.

Este último servicio es el que se usará en el proyecto como una base de datos gratuita y en la nube, además de poder compartir datos con todos los dispositivos presentes en cada una de las habitaciones del hogar. Otra ventaja a tener en cuenta a la hora de decantarse por el uso de esta base de datos es la compatibilidad que tiene con el resto de tecnologías presentes en este proyecto como van a ser MIT App Inventor y Arduino.

3.3 MIT App Inventor

Todo el mundo, incluidos los niños, pueden crear aplicaciones completamente funcionales para teléfonos inteligentes y tabletas utilizando MIT App Inventor, un entorno de programación visual fácil de usar. En menos de 30 minutos, aquellos que no estén familiarizados con MIT App Inventor pueden lanzar su primera aplicación sencilla. Además, esta tecnología basada en bloques hace posible desarrollar aplicaciones sofisticadas y altamente efectivas mucho más rápido que con los entornos de programación convencionales. Al permitir que todos, especialmente los jóvenes, pasen del consumo de tecnología a la producción de tecnología, la iniciativa MIT App Inventor tiene como objetivo democratizar el desarrollo de software.

MIT App Inventor tiene una interfaz relativamente sencilla porque solo tiene dos pestañas: diseñador y bloques. En *diseñador* se seleccionan los componentes que se quieren que incluya cada ventana de la aplicación, y los bloques, donde se programan arrastrando bloques conectados a los componentes que se tienen incorporados. A

medida que se desarrolla la aplicación, MIT App Inventor da la opción de probarla usando un emulador, USB o AI Companion, un programa que se puede descargar desde Google Play Store, o también ingresando un código para ver cómo funciona en un smartphone.

Además, MIT App Inventor incluye una característica que se puede incluir en la app. Se trata de FirebaseDB, la cual permitirá conectarse a nuestra base de datos en Firebase y tener un acceso tanto de lectura y escritura a nuestros datos.

3.4 Arduino IDE

Arduino IDE o entorno de desarrollo integrado de Arduino (Arduino Integrated Development Environment en inglés) es una plataforma de código abierto basada en hardware y software fácil de usar. Está destinado y creado para simplificar el uso, el diseño y la programación de microcontroladores. En el Instituto de Diseño de Interacción Ivrea, Arduino se desarrolló como una herramienta de creación rápida de prototipos para personas con poca experiencia. La fuerza del concepto Arduino reside en su código fuente abierto y su comunidad, que comparte miles de proyectos, lo que facilita adquirir nuevas habilidades, perfeccionar las existentes y crear nuevos proyectos utilizando lo aprendido. Su lenguaje de programación se basa en Java y funciona en Windows, Mac OS X y Linux.

También, Arduino posee su propio editor web para programar y almacenar tus sketches en la nube, además de tener disponibles todas las librerías y placas de Arduino.

3.5 KiCad

KiCad es un software libre destinado al diseño esquemáticos de circuitos electrónicos y que estos pasen a placas de circuito impreso (del inglés: Printed Circuit Board, PCB).

Los componentes principales del software incluyen herramientas para producir esquemas, diagramas de circuitos impresos, simulaciones SPICE, nomenclatura, dibujos, archivos Gerber y vistas tridimensionales de la PCB y sus partes.

La aplicación multiplataforma KiCad funciona con Windows, Linux y Apple MacOS. Está disponible bajo la licencia GNU GPL de código abierto.

3.6 Wemos D1 mini

Wemos D1 mini es una placa de desarrollo basada en el SoC (System on Chip) ESP8266, el cual es un chip de bajo costo que incluye un microcontrolador y capacidad de conexión wifi. Siendo sus dimensiones de 34.2 mm x 25.6 mm y un peso de 3 g, es una de las placas más pequeñas basadas en el ESP8266.

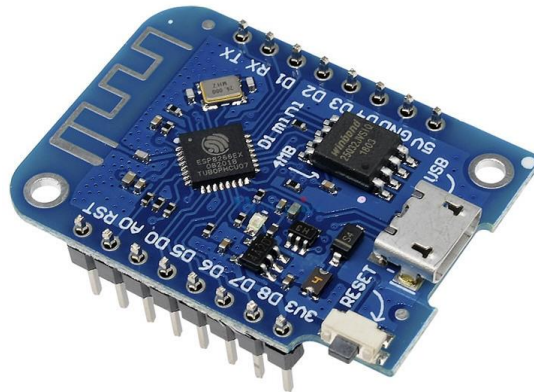


Ilustración 3.1. Placa Wemos D1 mini

Esta placa tiene un procesador de 32 bits una memoria flash de 4MB. Esta, a su vez, es la versión más pequeña de la placa Wemos D1.

Como la Wemos D1 mini es muy fácil de utilizar y compatible con Arduino IDE, es ideal para incluirla en proyectos orientados al IoT ya que también, como se ha mencionado antes, posee un bajo costo. Además, permite aumentar la funcionalidad mediante la conexión de Shields, como pueden ser: controlador de motores, módulo con relé, pantalla Oled, sensores de temperatura o humedad.

La versión de la placa que se utiliza en este proyecto es la V3.1.0, que se puede ver en la ilustración 3.1. y tiene las siguientes características:

- Frecuencia del reloj: 80MHz/160MHz
- Memoria flash: 4MB
- Tensión funcionamiento: 3.3V
- Entradas y salidas digitales: 11, todos (salvo el D0) con PWM, interrupciones, e I2C
- Entradas analógicas: 1 (Max. 3.2V)
- Conector Micro-USB

La enumeración de pines y el resumen de su función principal vienen explicados en la tabla 3.1., junto con la guía ilustrativa de la configuración de pines de la ilustración 3.2.:

Pin	ESP-8266 Pin	Función
TX	TXD	TXD
RX	RXD	RXD
A0	A0	Analog input (max 3.2V)
D0	GPIO16	IO
D1	GPIO5	IO, PWM, Interrupt, I2C, SCL
D2	GPIO4	IO, PWM, Interrupt, I2C, SDA
D3	GPIO0	IO, 10k Pull-up, PWM, Interrupt, I2C
D4	GPIO2	IO, 10k Pull-up, PWM, Interrupt, I2C, BUILTIN_LED
D5	GPIO14	IO, PWM, Interrupt, I2C, SCK
D6	GPIO12	IO, PWM, Interrupt, I2C, MISO
D7	GPIO13	IO, PWM, Interrupt, I2C, MOSI
D8	GPIO15	IO, 10k Pull-down, PWM, Interrupt, I2C, SS
G	GND	Ground
5V	-	5V
3V3	3.3V	3.3V
RST	RST	Reset

Tabla 3.1. Configuración de pines de la placa Wemos D1 mini

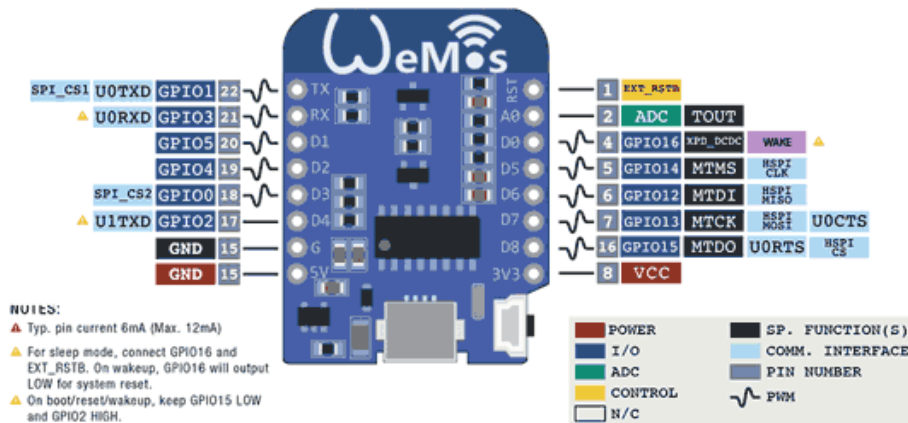


Ilustración 3.2. Guía visual de los pines de la placa Wemos D1 mini

3.7 Dimmer

El dimmer es un dispositivo capaz de ajustar el nivel de luz de la luminaria presente en una Smart home. El dimmer se conecta a la placa Wemos D1 mini mediante dos pines digitales. Uno de ellos controla el paso por cero y, el otro (DIM/PSM) controla el nivel de corriente.

Los niveles lógicos que admite son de 5V y de 3.3V, por lo que lo hace ideal para utilizarlo con la placa Wemos D1 mini.

3.8 Relé

Este interruptor eléctrico nos permitirá controlar la climatización del hogar. Unos de los shields de Arduino es un relé de 5V.

Este relé soporta en la configuración NC (Normally Close) en continua una tensión de 30V y una corriente de 3A y, en alterna, 250V y 3A. En su configuración NO (Normally Open) soporta en continua 30V y 3A y, en alterna, 250V y 3A.

Aunque para el proyecto es una opción viable, hay módulos comerciales con, por ejemplo, dos relés de 5V y 30A, capaces de soportar una potencia mayor, como la que se necesitaría para controlar la climatización de una habitación de una Smart home, al requerir esta de mucha potencia en comparación a la pensada en el proyecto.

3.9 Shield de humedad y temperatura

Otro shield del que dispone Arduino es el SHT30 Shield, capaz de darnos datos de la humedad y temperatura del ambiente.

Algunas de las características de este shield son:

- Tamaño pequeño y compacto
- Bajo coste
- Precisión de $\pm 3\%RH$ y $\pm 0.3^{\circ}C$
- Comunicación I2C
- Dos direcciones seleccionables (D1 y D2)

3.10 LDR

Para medir la luminosidad de las habitaciones, se usará una LDR, pues es un sensor fácil de obtener y económico, además de dar unos valores de luminosidad bastantes fiables.

Para explicar rápidamente que es una LDR (Light-Dependent Resistor), es un componente electrónico que posee una resistencia que varía en función de la cantidad de luz que recibe. A más luz recibe, su resistencia disminuye (normalmente). Su funcionamiento se basa en el efecto fotoeléctrico al incidir luz en un semiconductor como el sulfuro de cadmio, CdS.

3.11 Multímetro

De los elementos más importantes en este proyecto, este módulo PZEM-004T tiene la función de medir parámetros eléctricos como el voltaje, la corriente, la potencia activa y la energía.

Comunicando este dispositivo por el puerto serie con Arduino, se pueden obtener valores de la potencia de nuestro hogar y de la potencia que se esté usando en cada momento y ajustarlas para adecuarnos a cada situación.

Las conexiones se realizan de la misma forma que si se utilizara una pinza amperimétrica para tomar las medidas. Uno de los hilos del circuito a medir se pasa por el toroidal para medir la intensidad, y el PZEM se conecta en paralelo con los puntos donde se quiera medir la tensión. En nuestro caso, bastará simplemente con abrir la pinza. La conexión esquemática viene representada en la ilustración 3.3.

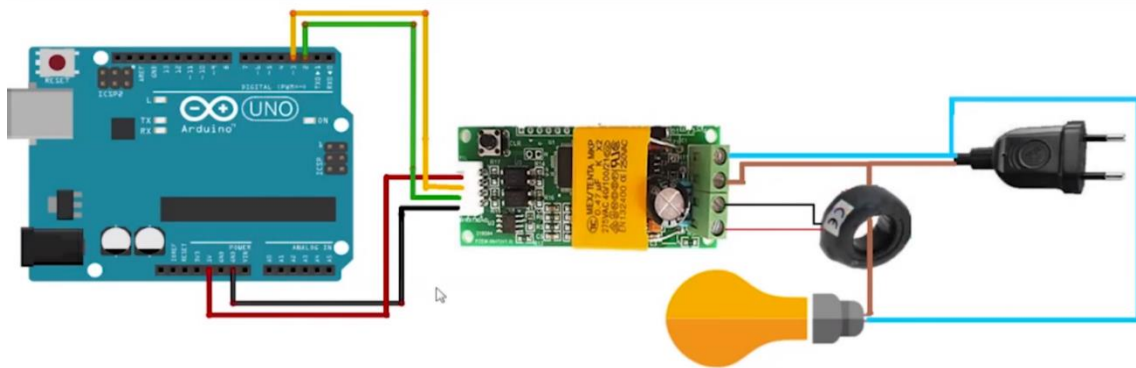


Ilustración 3.3. Conexión del módulo PZEM-004T

Las especificaciones de este multímetro son:

- Voltaje de funcionamiento: 80 ~ 260 VCA
- Voltaje de prueba: 80 ~ 260 VCA
- Energía clasificada: 100A / 22000W
- Frecuencia de trabajo: 45-65Hz
- Precisión de medición: 1.0
- Tamaño de la placa: 7,3 x 3 x 2,3 cm

3.12 Alimentación

Este prototipo ha utilizado la alimentación micro-USB que trae incorporada y por la que se cargan nuevos programas. Sin embargo, para un funcionamiento autónomo y sin tener que estar dependiendo de estar enchufado a un ordenador, el cual debe de estar encendido continuamente, se tendrá que añadir al prototipo una fuente de alimentación capaz de darnos una corriente continua de 5V, que es lo que necesita el Wemos D1 mini.

3.13 Otros materiales

Además de lo dicho anteriormente, también se hace uso de luces leds para simular el encendido de distintos relés durante la fase de prototipo, sus resistencias, el cableado necesario del prototipo, así como la placa Protoboard para realizar todas las conexiones sobre ella.

4. METODOLOGÍA

Este trabajo comienza con la idea de controlar los distintos aparatos en el hogar según su potencia, por lo que necesitamos medir esta. Para ello, se piensa en que la plataforma donde se lleve a cabo el proyecto sea una cuyas librerías se encuentren fácilmente a disposición del público y tenga recursos de fácil uso, por lo que Arduino es la mejor opción.

Tras ello, se diseña la configuración que va a tener este trabajo en una supuesta casa, que se ha diseñado en base a las necesidades de una casa real y en las que se den todos los casos que se han pensado para el proyecto. Con respecto al control manual de los dispositivos domóticos del hogar que contendrá el proyecto, desarrollamos una aplicación móvil para el dispositivo smartphone, desde donde se puede controlar el hogar y visualizar datos recogidos.

En este proyecto, puesto que los dispositivos que se van a utilizar simulan los que se utilizarían en realidad, pero no son iguales, tanto la potencia que consumen, así como las variables que los controlan, serán también distintas por lo que es importante definir las bien desde el principio. La programación viene integrada en un mismo dispositivo Arduino para que este sea capaz de controlar varias funciones, aunque lo ideal sería separar la programación bien por habitaciones, en el caso de que, en la habitación, los medios a controlar sean capaces de ser soportados por un mismo dispositivo; o por módulos, puesto que estos se encargarían de controlar distintos apartados como puede ser un grupo de electrodomésticos o de iluminaciones diferentes.

El dispositivo Arduino se conecta a una red Wi-Fi para poder acceder a la base de datos Firebase, en el que consultará el último estado de las variables a controlar constantemente, ya que pueden ser modificadas en todo momento por la app móvil, a la vez que modificarlas según la medición de los sensores y del estado de los distintos actuadores. Por otro lado, App Inventor se encarga de actualizar el estado en Firebase de cada una de las variables que han sido modificadas desde la app móvil, además de cargar en la app el estado de ellas cada vez que se cambia entre una pantalla y otra.

4.1 Creación de proyectos

La programación básica en el IDE de Arduino no incluye la tarjeta ESP8266 por lo que se ha descargado añadiendo una URL en el gestor de tarjetas. Esto permitirá poder buscarla en el gestor, descargarla e instalarla.

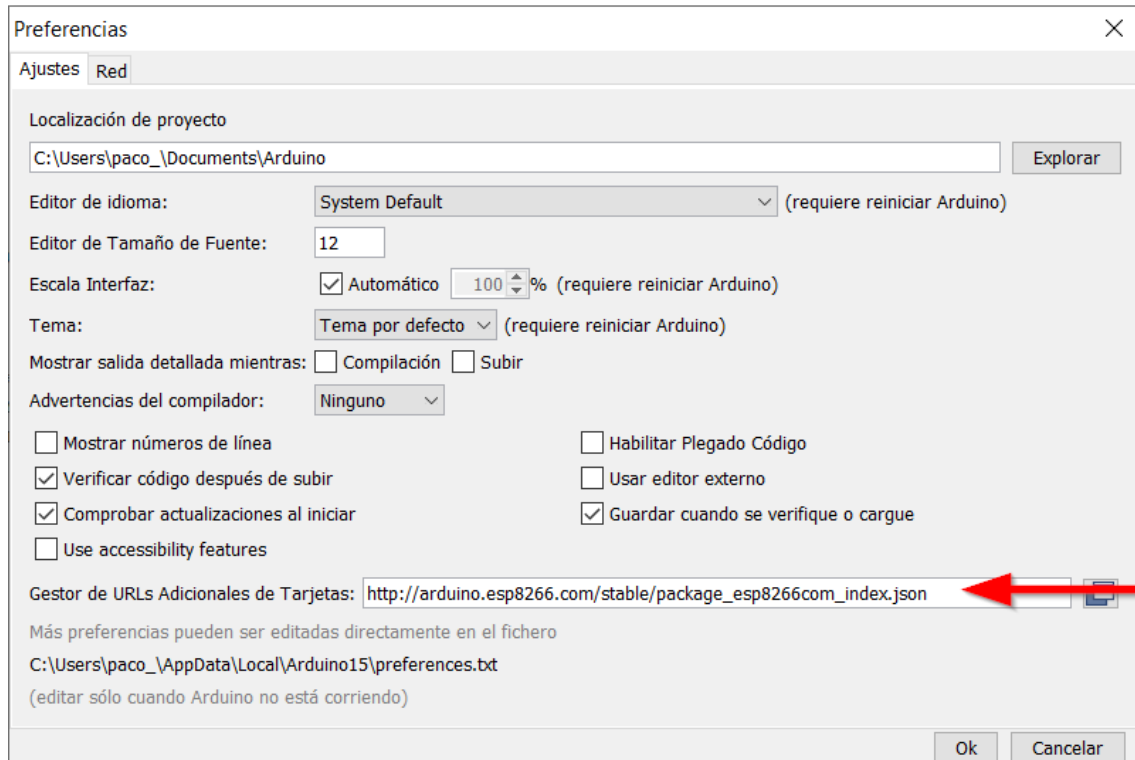


Ilustración 4.1. Preferencias de Arduino IDE

Una vez instalada, aparecerá en la pestaña Herramientas, donde se podrá buscar la tarjeta, LOLIN(WEMOS) D1 R2 & mini en nuestro caso.

La programación del IDE Arduino estará leyendo y actualizando periódicamente las variables que derivan de sensores, así como internas del código para el correcto funcionamiento del algoritmo de control. Otras variables solo podrán ser alteradas desde la app móvil o cambiando directamente la base de datos de Firebase.

Para crear el proyecto de Firebase, podrá ser de ayuda cualquier guía o vídeo colgado en la web pues hay multitud de estos en línea. Aun así, es fácil e intuitivo. En primer lugar, se creará un proyecto al que se le pondrá un nombre. En nuestro caso, será el nombre control del consumo TFG.

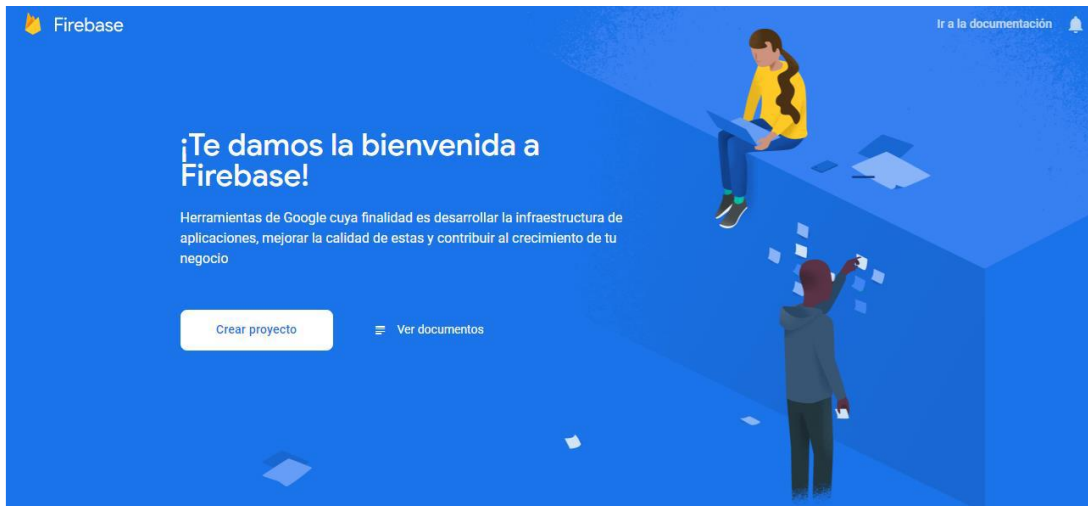


Ilustración 4.2. Pantalla de bienvenida de Firebase

Seleccionamos si queremos habilitar Google Analytics.

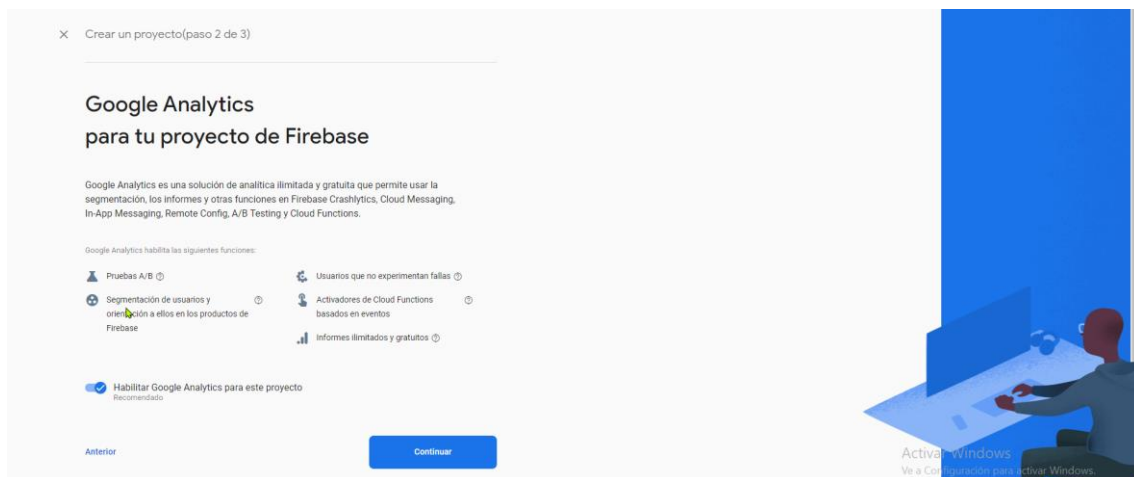


Ilustración 4.3. Pantalla de selección de Google Analytics en Firebase

Seleccionamos para trabajar en la parte de Realtime Database y escogemos las reglas de seguridad.

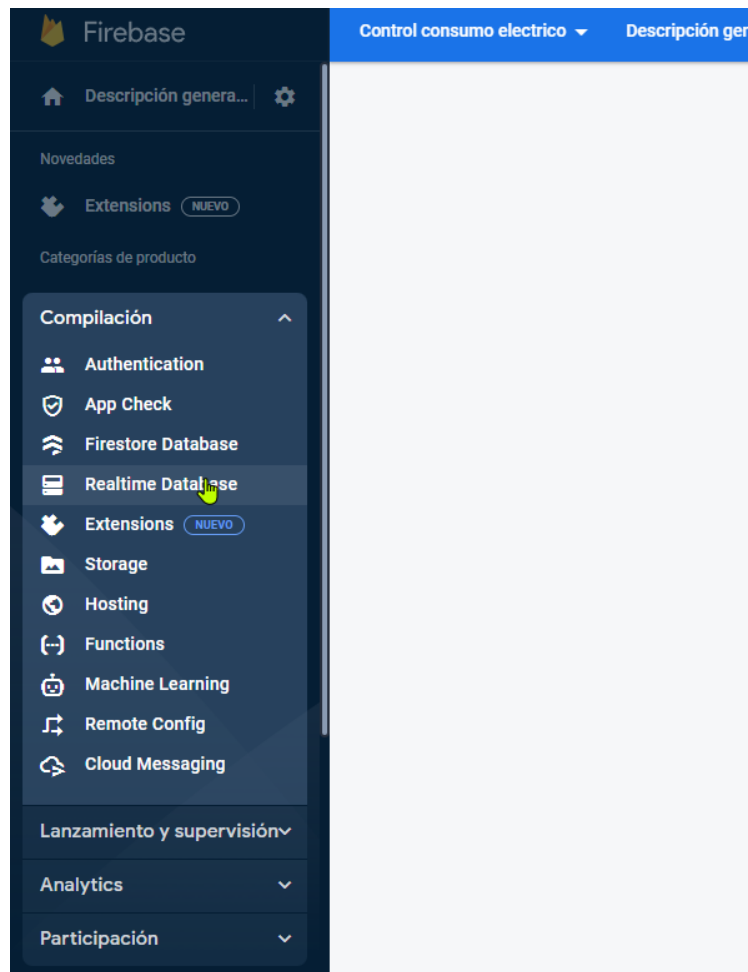


Ilustración 4.4. Recorte de pantalla del menú de Firebase

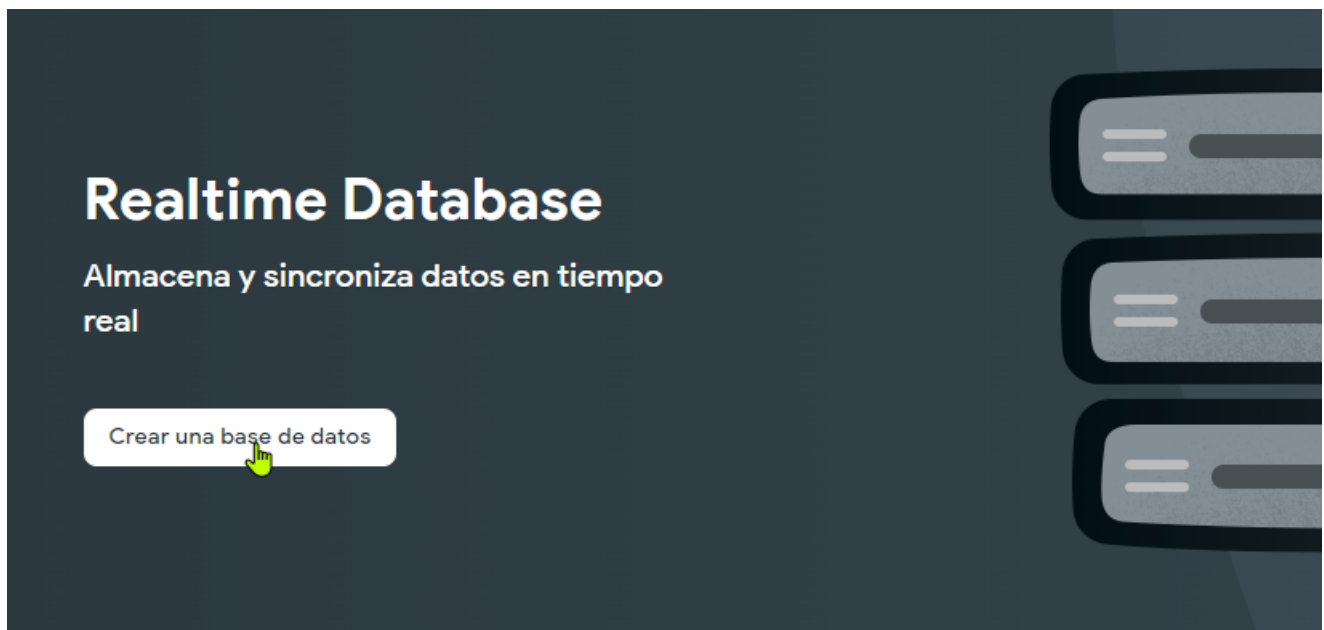
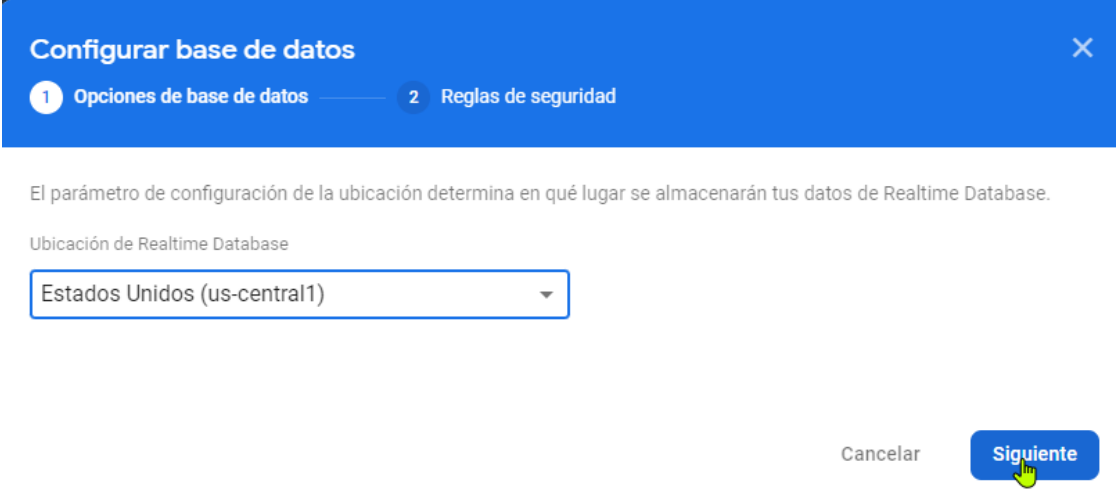


Ilustración 4.5. Botón de selección de creación de nueva base de datos en Firebase



Configurar base de datos ✕

1 Opciones de base de datos — 2 Reglas de seguridad

El parámetro de configuración de la ubicación determina en qué lugar se almacenarán tus datos de Realtime Database.

Ubicación de Realtime Database

Estados Unidos (us-central1) ▼

Cancelar Siguiente

Ilustración 4.6. Selección de ubicación de la base de datos de Firebase



Configurar base de datos ✕

1 Opciones de base de datos — 2 Reglas de seguridad

Cuando definas la estructura de los datos, deberás crear reglas para protegerlos.
[Más información](#)

☐ Comenzar en modo bloqueado

De forma predeterminada, tus datos son privados. El acceso de lectura/escritura de los clientes solo se otorgará como se indica en tus reglas de seguridad.

☒ Comenzar en modo de prueba

Para permitir una configuración rápida, los datos se abren de forma predeterminada. Sin embargo, debes actualizar las reglas de seguridad en un plazo de 30 días a fin de habilitar el acceso de lectura/escritura a largo plazo para los clientes.

```
{
  "rules": {
    ".read": "now < 1670194800000", // 2022-12-5
    ".write": "now < 1670194800000", // 2022-12-5
  }
}
```

! Las reglas de seguridad predeterminadas del modo de prueba permiten que cualquier usuario con acceso a tu referencia de base de datos pueda ver, editar y borrar todos los datos durante los siguientes 30 días.

Cancelar Habilitar

Ilustración 4.7. Preferencias de seguridad de la base de datos de Firebase

Para la correcta conexión de Firebase con Arduino, habrá que visitar la página para hallar la 'huella digital' de la base de datos.

Privacy
Research
Other



Fingerprints

Is your employer, school, or Internet provider

eavesdropping on your **secure** connections?

342 sets of fingerprints checked per day
2,664,184 sets of fingerprints checked for our visitors

Secure browser connections can be intercepted and decrypted by authorities who spoof the authentic site's certificate. But **the authentic site's fingerprint CANNOT be duplicated!**

Domain Name	Certificate Name	EV	Security Certificate's Authentic Fingerprint
www.grc.com	grc.com	—	C3:FB:91:85:CC:6B:4C:7D:E7:18:ED:D8:00:D2:84:E7:6E:97:06:07
www.facebook.com	*.facebook.com	—	53:EA:92:FD:D7:0C:4F:27:B6:A4:9D:20:59:E5:E1:45:D8:E5:41:37
www.paypal.com	www.paypal.com	—	B4:F9:B2:3D:92:D4:97:AB:21:98:9F:49:17:AB:AD:F7:03:F3:98:0B
twitter.com	twitter.com	—	DA:26:F7:2E:AC:AC:75:CC:EE:79:41:40:44:10:20:84:96:97:AE:B0
www.blogger.com	*.blogger.com	—	B1:13:85:4C:73:3C:BB:9A:8E:E3:5A:68:25:82:A8:0B:84:01:E5:98
www.linkedin.com	www.linkedin.com	—	E8:B3:49:E7:DE:6C:56:87:4D:5E:CC:2F:F0:54:C7:FD:AA:3C:37:05
www.yahoo.com	yahoo.com	—	83:51:5E:8E:98:3C:E9:51:0C:D3:45:1D:50:94:9D:D4:FB:BF:B2:99
wordpress.com	*.wordpress.com	—	68:E8:C3:E6:33:46:72:78:38:16:D6:C6:45:B4:C3:FB:83:82:C8:0C
www.wordpress.com	*.wordpress.com	—	68:E8:C3:E6:33:46:72:78:38:16:D6:C6:45:B4:C3:FB:83:82:C8:0C

Each site's authentic security certificate fingerprint (shown above) was just now obtained by GRC's servers from each target web server. If your web browser sees a different fingerprint for the same certificate (carefully verify the Certificate Name is identical) that forms strong evidence that something is intercepting your web browser's secure connections and is creating fraudulent site certificates.

Custom Site Fingerprinting

In addition to the well-known web sites listed above, GRC's web server can obtain and display the "fingerprint" of any HTTPS-capable public web server's secure connection certificate. Simply enter the domain name of the server you wish to fingerprint, then press Enter or click the "Fingerprint Site" button:

https://

Ilustración 4.8. Pantalla de Fingerprints



eavesdropping on your **secure** connections?

202 custom fingerprints checked per day
943,794 custom fingerprints checked for our visitors

Secure browser connections can be intercepted and decrypted by authorities who spoof the authentic site's certificate. But **the authentic site's fingerprint CANNOT be duplicated!**

Domain Name	Certificate Name	EV	Security Certificate's Authentic Fingerprint
control-consumo-electrico-default-rtdb.firebaseio.com	*.us-central1.firebaseiodatabase.app	—	05:54:AC:F9:6D:9B:E0:9C:10:B4:09:B1:E2:7C:12:AF:B4:C8:6C:EA

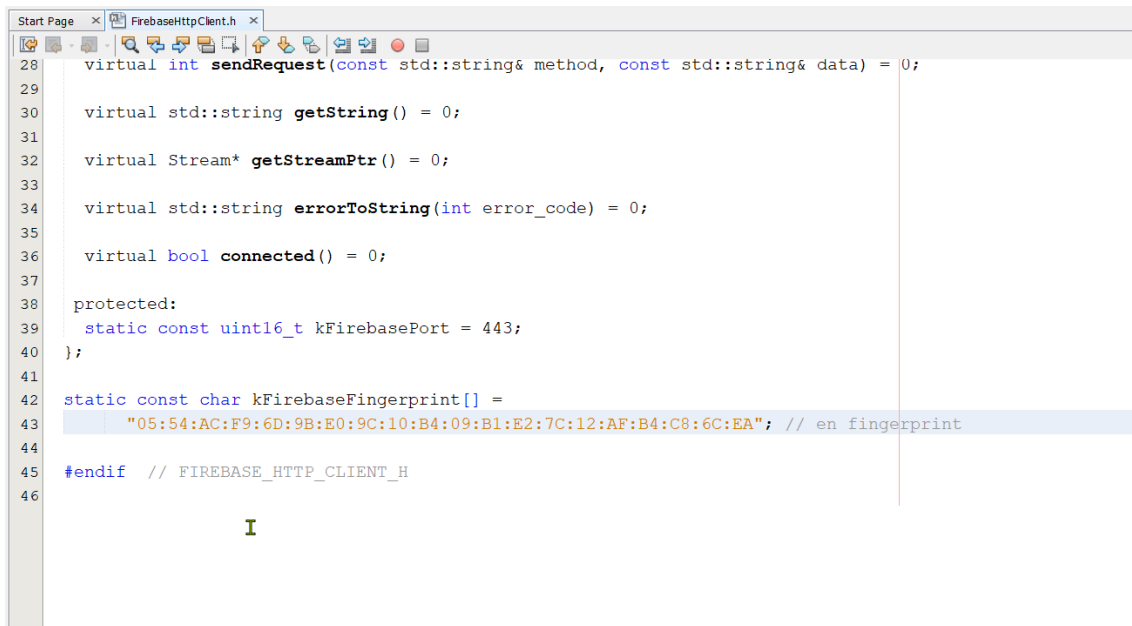
Each site's authentic security certificate fingerprint (shown above) was just now obtained by GRC's servers from each target web server. If your web browser sees a different fingerprint for the same certificate (carefully verify the Certificate Name is identical) that forms strong evidence that something is intercepting your web browser's secure connections and is creating fraudulent site certificates.

Custom Site Fingerprinting

GRC's web server can obtain and display the "fingerprint" of any HTTPS-capable public web server's secure connection certificate. Simply enter the domain name of the server you wish to fingerprint, then press Enter or click the "Fingerprint Site" button:

Ilustración 4.9. Ubicación de la base de datos

Esta nos dará la dirección física donde se encuentra. La copiamos y abrimos el archivo *FirebaseHttpClient.h* y, al final de todo el código, reemplazamos la dirección que haya por la obtenida.



```

28 virtual int sendRequest(const std::string& method, const std::string& data) = 0;
29
30 virtual std::string getString() = 0;
31
32 virtual Stream* getStreamPtr() = 0;
33
34 virtual std::string errorToString(int error_code) = 0;
35
36 virtual bool connected() = 0;
37
38 protected:
39     static const uint16_t kFirebasePort = 443;
40 };
41
42 static const char kFirebaseFingerprint[] =
43     "05:54:AC:F9:6D:9B:E0:9C:10:B4:09:B1:E2:7C:12:AF:B4:C8:6C:EA"; // en fingerprint
44
45 #endif // FIREBASE_HTTP_CLIENT_H
46

```

Ilustración 4.10. Extracto de código de *FirebaseHttpClient.h*

La base de datos ha sido organizada por habitaciones, dejando fuera de ellas algunos datos que serán consultados por Arduino. Dentro de las habitaciones, se encuentran los diferentes dispositivos de los que se dispondrá en cada habitación. Dado que esto es un prototipo y la programación sería similar entre habitaciones para controlar los diferentes dispositivos, solo se ha optado por realizar la programación de ‘Salón’ y ‘Baño’ para comprobar cómo funcionarían las principales funciones del prototipo.

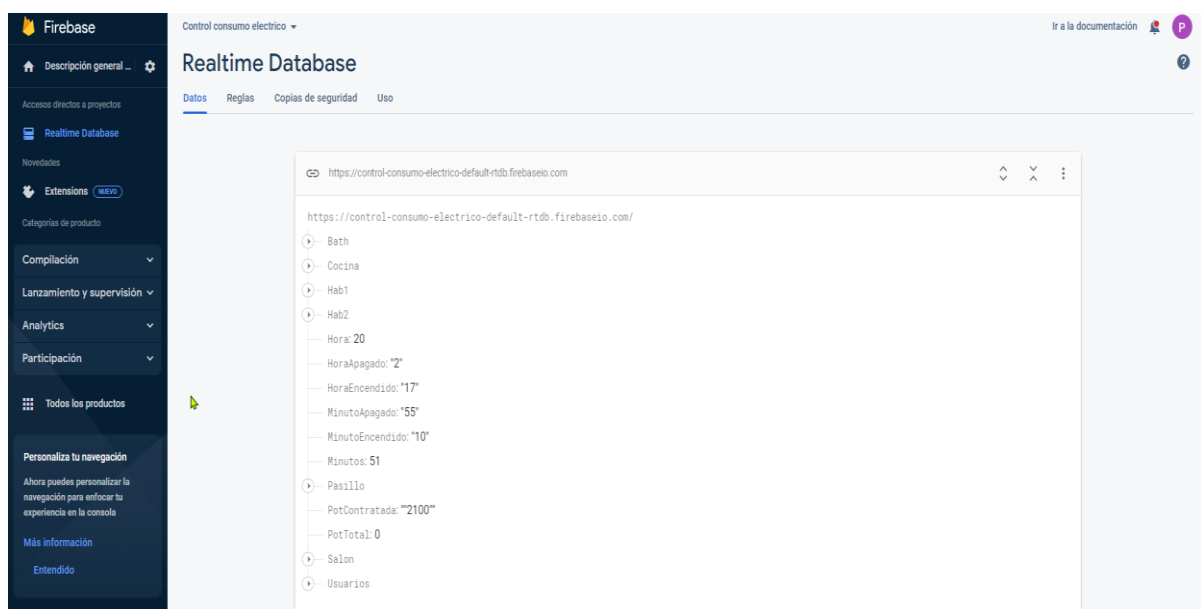


Ilustración 4.11. Base de datos en Firebase del proyecto

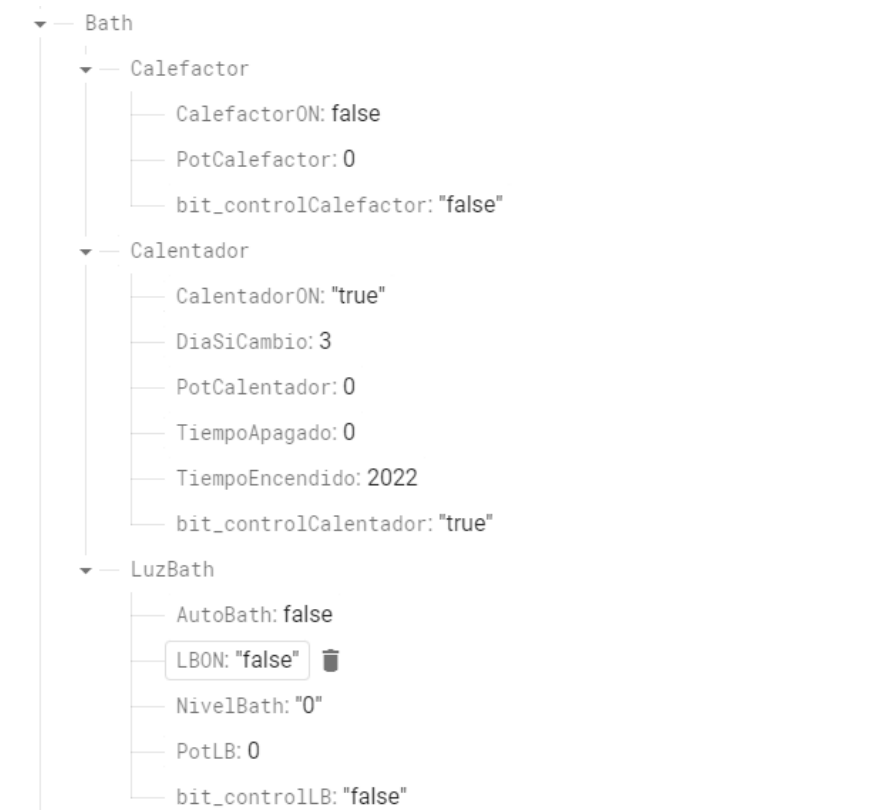


Ilustración 4.12. Información detallada de las habitaciones en Firebase (parte 1)



Ilustración 4.13. Información detallada de las habitaciones en Firebase (parte 2)

Otro proyecto que se ha tenido que crear es en la plataforma de MIT App Inventor para poder desarrollar la app móvil que controle el prototipo. Para ello, tan solo hay que darle a crear un nuevo proyecto, seleccionar un nombre y ya estaríamos listos para comenzar a programar nuestra app.

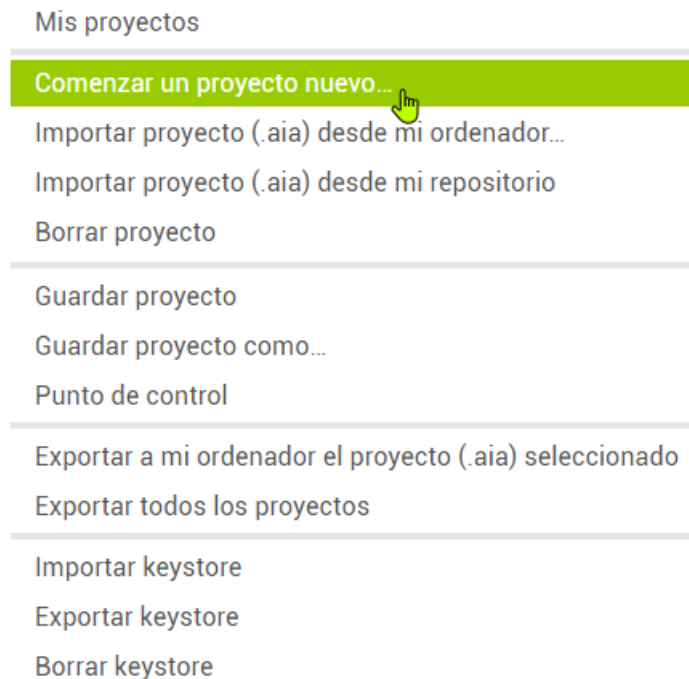


Ilustración 4.14. Creación de un nuevo proyecto en MIT App Inventor



Ilustración 4.15. Selección de nombre del proyecto en MIT App Inventor

La app está compuesta por elementos fáciles que permiten una fácil navegación entre ella, siendo bastante intuitiva.

Conforme abrimos la app, la primera ventana que nos encontramos es la de inicio de sesión. En esta ventana, se pueden registrar nuevos usuarios o introducir algún usuario ya registrado previamente.

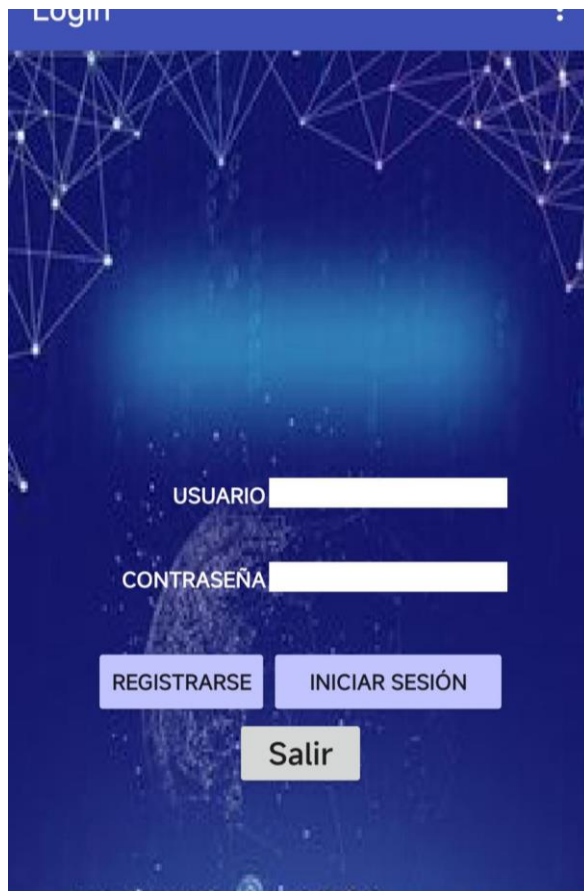


Ilustración 4.16. Pantalla de Login de la aplicación móvil

Si hemos iniciado sesión correctamente, pasaremos a la pantalla de Inicio, donde se podrá elegir entre ajustar nuestra Smart Home o ver una gráfica con el consumo en tiempo real.

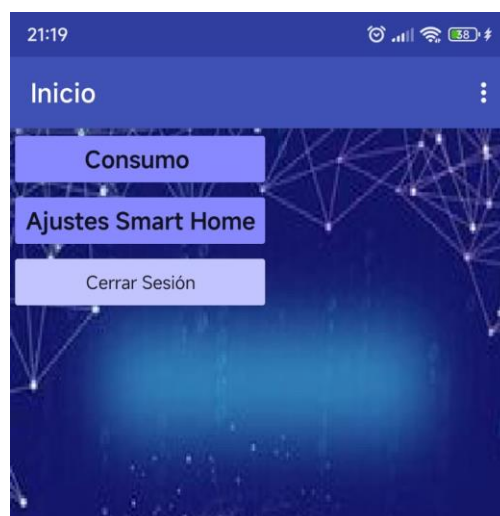


Ilustración 4.17. Pantalla de Inicio de la aplicación móvil

Si entramos en Ajustes Smart Home, podremos seleccionar cualquier habitación para cambiar el estado de los dispositivos y cambiar el orden de prioridad de estas para que desconectasen sus dispositivos al superarse cierta potencia contratada.



Ilustración 4.18. Pantalla de Ajustes Smart Home de la aplicación móvil

Solo en la pantalla de Ajustes Smart Home, podremos cambiar la potencia contratada en el hogar, así como la hora de encendido y apagado, la cual utilizará el modo automático de las habitaciones que lo necesiten.

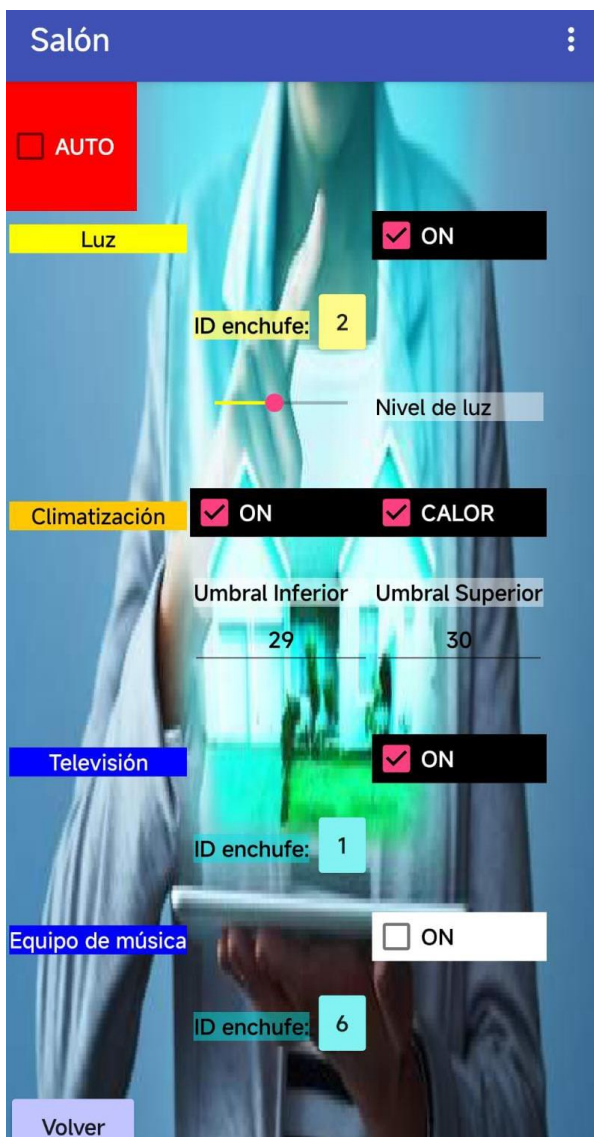


Ilustración 4.19. Pantalla de Salón de la aplicación móvil

Dentro de las habitaciones se pueden controlar la luz, ajustar umbrales de temperatura de la climatización, poner en modo automático o encender y apagar cualquier otro dispositivo. Además, puedes cambiar el número identificador (ID) de cada dispositivo para que estos puedan funcionar en otras habitaciones.

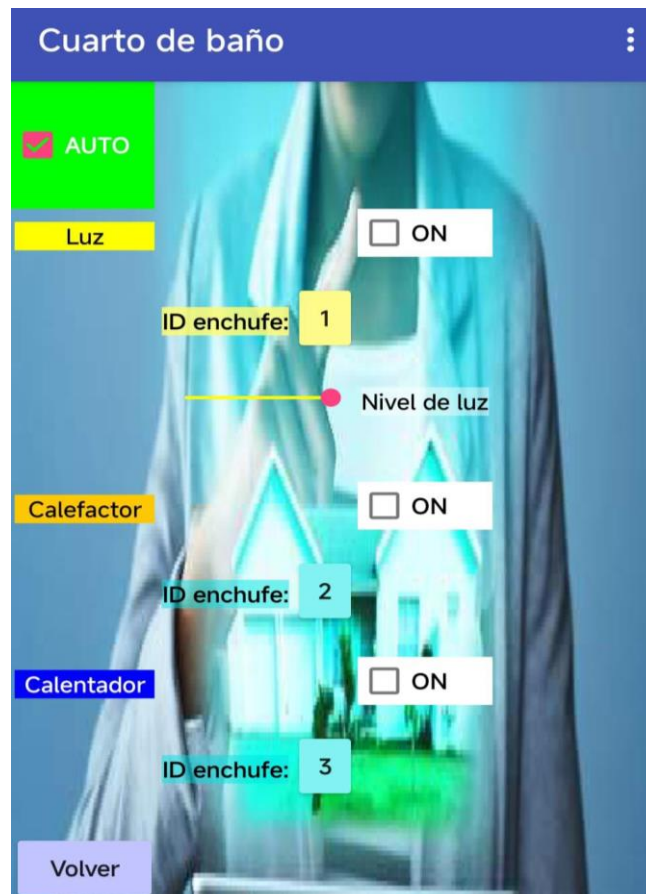


Ilustración 4.20. Pantalla de Cuarto de baño de la aplicación móvil

Aquí ahora podemos ver un ejemplo de la gráfica de barras con la potencia de lo encendido según las imágenes anteriores.

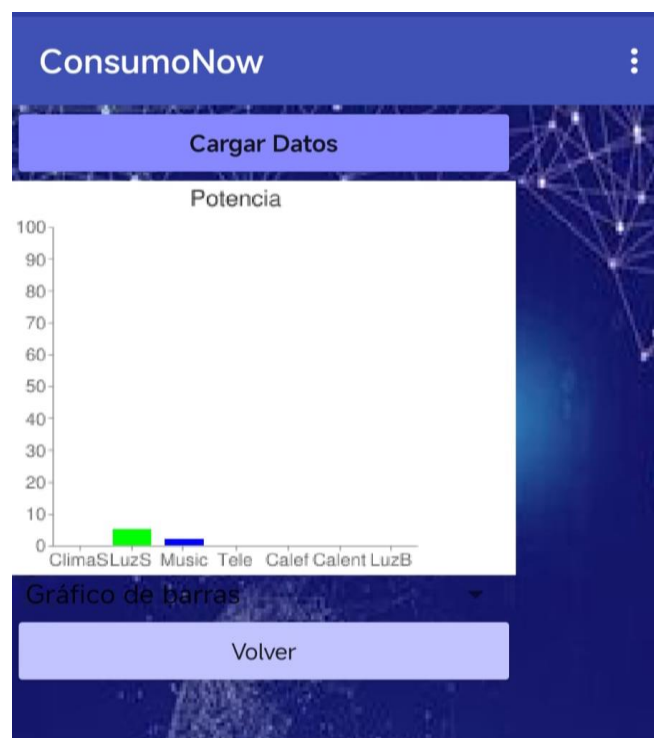


Ilustración 4.21. Pantalla de ConsumoNow de la aplicación móvil

4.2 Módulos

Todo el equipo está compuesto por cuatro módulos distintos encargados cada uno de una tarea. Estos módulos de control son: iluminación, sensor de iluminación, climatización y enchufe inteligente. Cada módulo cuenta con su propia programación, estando dispuestos en una PCB, especialmente diseñada en función de los dispositivos necesarios para su correcto funcionamiento.

En resumen, el proyecto está formado por distintos módulos distribuidos por las habitaciones y controlando el consumo energético y actuando en consecuencia con los actuadores correspondientes.

Iluminación

Componentes

El módulo de iluminación estaría compuesto por el Wemos D1 mini, una fuente de alimentación, un dimmer y la PZEM. Estos elementos estarían dispuestos sobre una PCB diseñada para el correcto funcionamiento de todo el módulo.

Si tenemos en cuenta más elementos externos a la hora de ver correctamente el funcionamiento del módulo, estos serían un portalámparas y una luminaria regulable, ya sea una luminaria LED o una luminaria incandescente.



Ilustración 4.22. Cara A del módulo de Iluminación

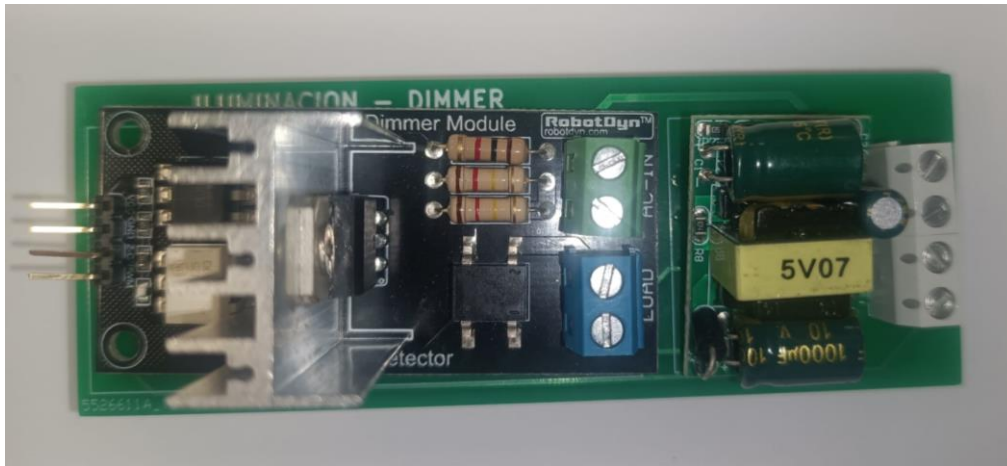


Ilustración 4.23. Cara B del módulo de Iluminación

Librerías

Para el correcto funcionamiento del código del prototipo en Arduino IDE, se han tenido que incluir las siguientes librerías:

<FirebaseArduino.h>: permite la lectura y escritura entre Arduino IDE y Firebase.

<ESP8266HTTPClient.h>.

<ESP8266WiFi.h>.

configuran la conexión vía Wi-Fi.

<ArduinoJson.h>: necesaria para la comunicación con Firebase.

<RBDdimmer.h>:

<dimmable_light.h>: marcan el correcto funcionamiento del dimmer.

<PZEM004Tv30.h>: configura el módulo PZEM para obtener la potencia consumida.

<Time.h>.

<TimeSib.h>.

<WiFiUdp.h>.

necesarias para obtener la hora local a través de internet.

Programación

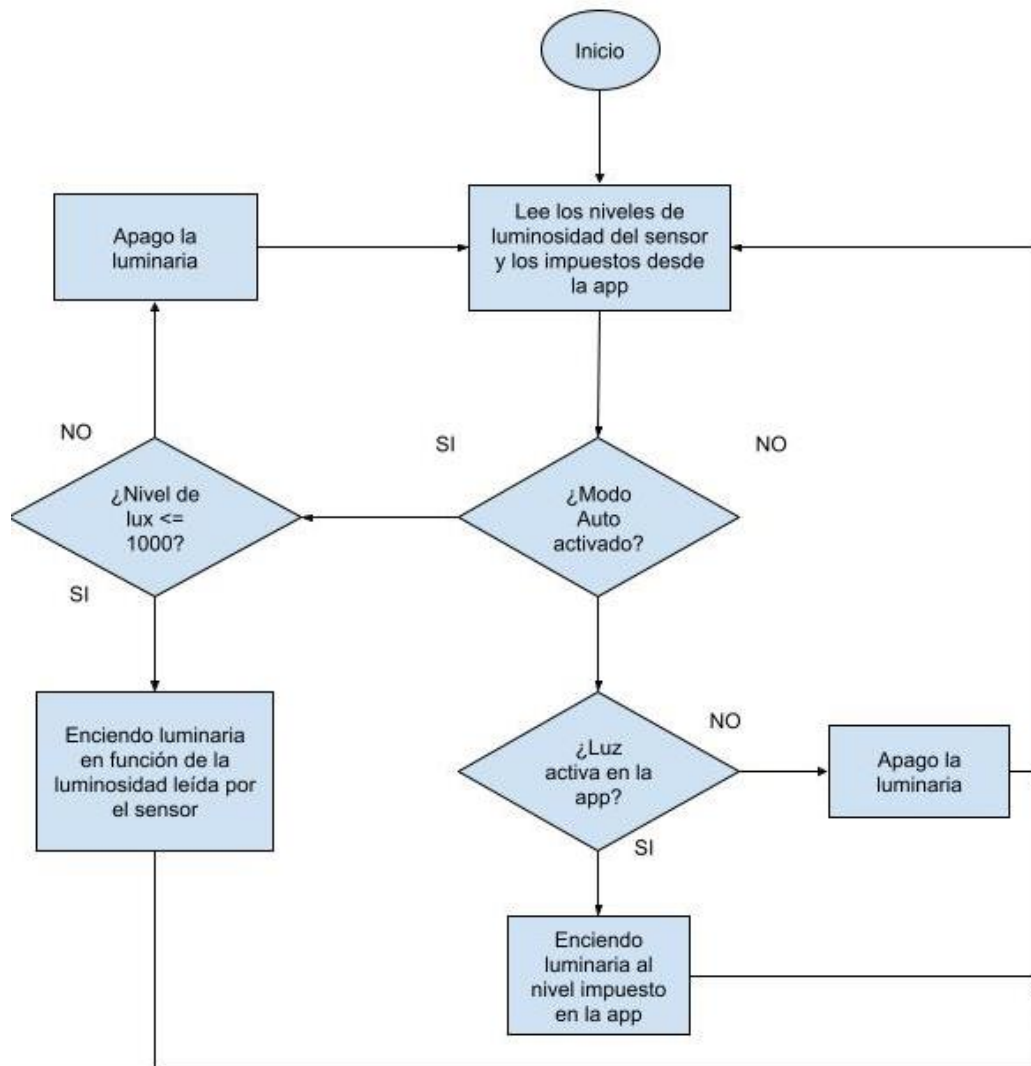


Ilustración 4.24. Diagrama de flujo de la iluminación

Funcionamiento

Arduino IDE se encarga de leer al principio las variables de Firebase para comprobar de qué módulo de iluminación se trata. Una vez que fija el ID del módulo, realiza una comparativa entre la potencia que se tiene contratada y la actual que estaría consumiendo la iluminación para elegir en qué modo trabajar. Tiene dos distintos modos según la cantidad de potencia total con respecto a la contratada.

- Hasta un 60% de la potencia contratada, trabajaría de modo normal, sin ninguna restricción de acorde a lo pensado a la hora de realizar la programación. Si el modo Auto está activado, se activará o no según la cantidad de luz presente en la habitación. Con el modo Auto desactivado, podremos controlar el estado de la luminaria, así como su nivel de luz, desde la app.

- En un rango superior al 60%, el módulo comprobaría si el dispositivo pertenece a la habitación en menor orden de prioridad y, si es el caso, apagaría toda luminaria puesto que no se desea que esté encendida. Tras esto, realiza una breve espera para comprobar que se ha bajado del 60%. Si no, comprobaría la siguiente habitación con el menor orden de prioridad.

Sensor de iluminación

Componentes

El módulo del sensor estaría compuesto por el Wemos D1 mini, una fuente de alimentación y el sensor BH1750.

Este módulo no cuenta con el medidor PZEM ya que el consumo energético es despreciable respecto al del hogar.



Ilustración 4.25. Cara A del módulo del sensor de iluminación

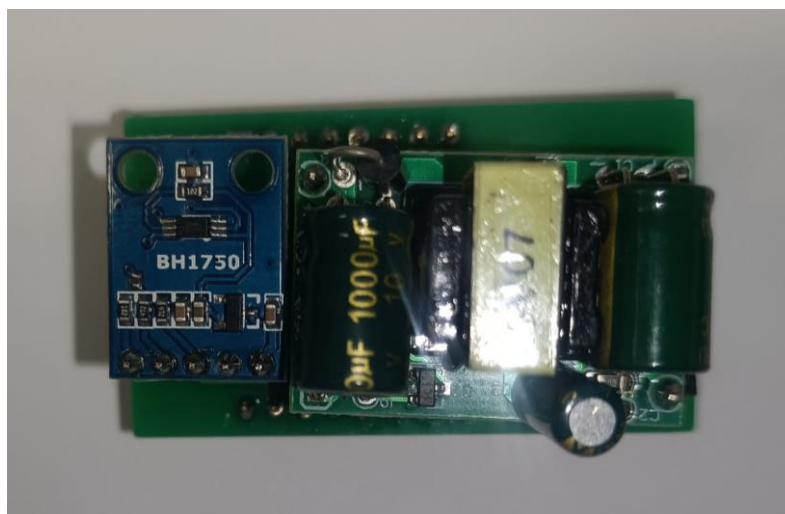


Ilustración 4.26. Cara B del módulo del sensor de iluminación

Librerías

Para el correcto funcionamiento del código del prototipo en Arduino IDE, se han tenido que incluir las siguientes librerías:

<FirebaseArduino.h>: permite la lectura y escritura entre Arduino IDE y Firebase.

<ESP8266HTTPClient.h>
<ESP8266WiFi.h> } configuran la conexión vía Wi-Fi.

<ArduinoJson.h>: necesaria para la comunicación con Firebase.

<BH1750.h>
<Wire.h> } configuran el sensor y su funcionamiento.

Programación

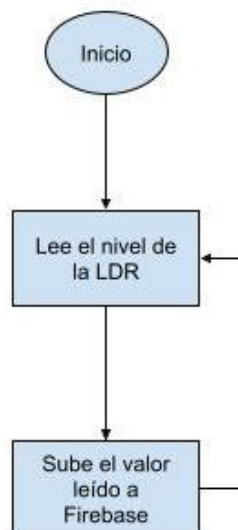


Ilustración 4.27. Diagrama de flujo del sensor de iluminación

Funcionamiento

El sensor solo se va a encargar de tomar datos de luminosidad y subirlos a Firebase donde se irán leyendo para el funcionamiento del módulo de iluminación, previamente explicado.

Al principio se fija el ID del módulo, toma los datos, los sube a la habitación correspondiente y vuelve a empezar.

El motivo por el que el módulo del sensor está construido aparte es para evitar falsear las medidas de la luminosidad al estar cerca de donde se situaría la luminaria.

Climatización

Componentes

El módulo de climatización a implementar en la vivienda dispone de un Wemos D1 mini acoplado con un módulo Shield SHT30, una fuente de alimentación, dos relés cuyos funcionamientos se explicarán más adelante; y el módulo PZEM para medir el consumo energético.

Estos dos relés mencionados anteriormente, se encargarían de cambiar entre Modo Frío o Modo Calor en el aire acondicionado split inverter (así controlaremos tanto el frío como el calor con la misma máquina) y el otro relé que se encargaría de encender o apagar este.

Para explicar esta conclusión, se ha analizado la conexión y funcionamiento real de estos aires acondicionados. Estos hacen que el equipo siempre quede en modo frío cuando la válvula inversora no está habilitada. Por lo tanto, todo lo que se requiere para que el equipo funcione en modo frío es que esté encendido, y para que funcione en modo calor, la válvula inversora debe estar recibiendo corriente (así activaría el ciclo de calor). De lo contrario, el equipo operaría en modo frío.

Así que, el esquema del funcionamiento del aire acondicionado en función de los relés quedaría de la siguiente manera:

	Relé Calor		
		ON	OFF
Relé ON	ON	Modo calor	Modo frío
	OFF	Apagado	

Tabla 4.1. Estado del aire acondicionado según los relés



Ilustración 4.28. Cara A del módulo de climatización

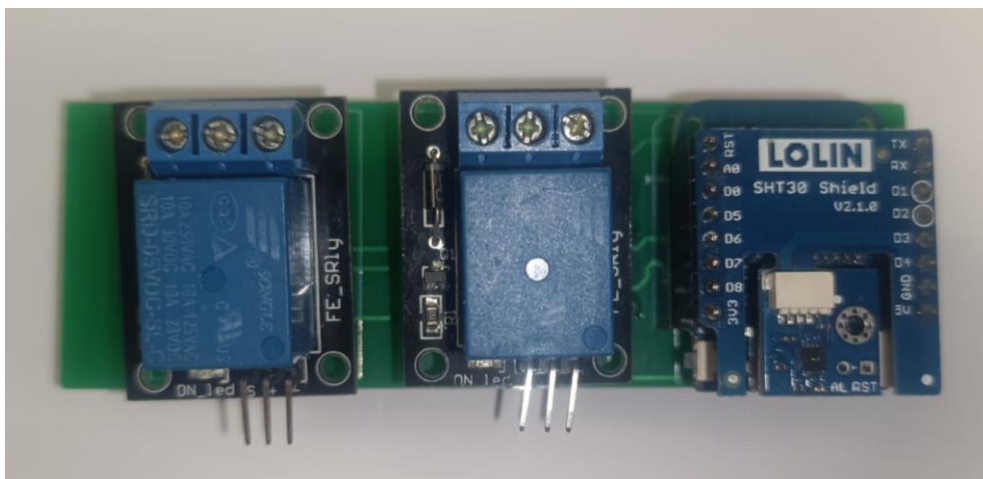


Ilustración 4.29. Cara B del módulo de climatización

Librerías

Para el correcto funcionamiento del código del prototipo en Arduino IDE, se han tenido que incluir las siguientes librerías:

<FirebaseArduino.h>: permite la lectura y escritura entre Arduino IDE y Firebase.

<ESP8266HTTPClient.h>.

<ESP8266WiFi.h>.

configuran la conexión vía Wi-Fi.

<ArduinoJson.h>: necesaria para la comunicación con Firebase.

<WEMOS_SHT3X.h>: configura el módulo SHT30 para obtener la temperatura.

<PZEM004Tv30.h>: configura el módulo PZEM para obtener la potencia consumida.

<Time.h>.

<TimeSib.h>.

<WiFiUdp.h>.

} necesarias para obtener la hora local a través de internet.

Programación

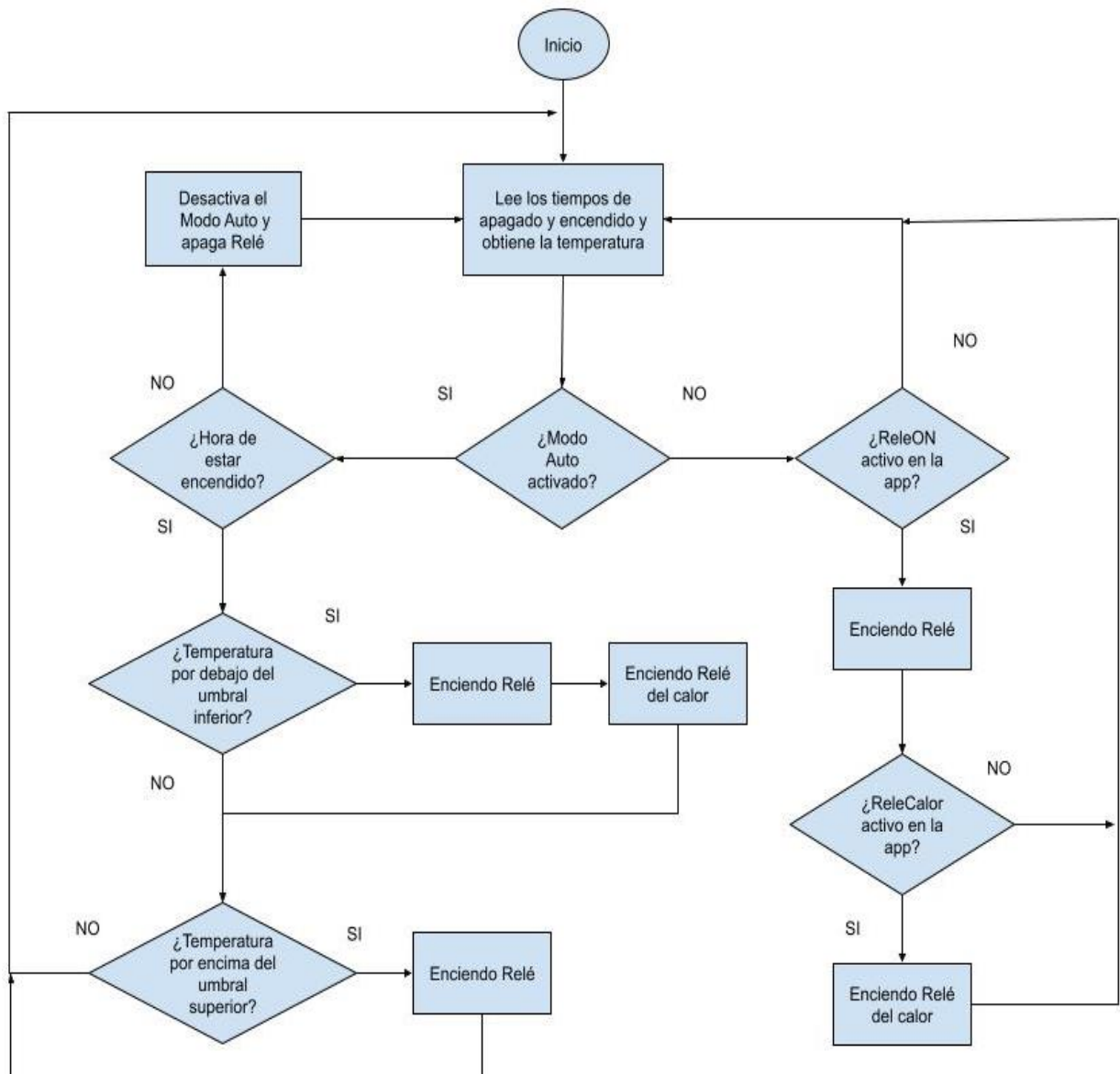


Ilustración 4.30. Diagrama de flujo de la climatización

Funcionamiento

Al principio se comprueban las variables de Firebase para asignar de qué módulo de climatización se trata. Una vez que fija el ID del módulo, obtiene la temperatura y, con las horas de encendido y apagado que se imponen desde la app, comprueba si tiene que desactivar el modo Auto. Todo esto lo hace trabajando de dos formas según el nivel de potencia utilizada en el hogar de la siguiente forma:

- Hasta un 60% de la potencia contratada, trabajaría de modo normal, sin ninguna restricción de acorde a lo pensado a la hora de realizar la programación. Si el modo Auto está activado, se activará el frío, el calor o ninguno según la comparación de temperatura ambiente con la de los umbrales impuestos desde la app. Con el modo Auto desactivado, podremos controlar el estado de la climatización a través de la app.
- En un rango superior al 60%, el prototipo empezaría a comprobar si el dispositivo pertenece a la habitación que se le ha dado menor orden de prioridad y, si es el caso, apagaría la calefacción puesto que no se desea que esté encendida. Tras esto, realiza una breve espera para comprobar que se ha bajado del 60%. Si no, comprobaría la siguiente habitación con el menor orden de prioridad.

Enchufe inteligente

Componentes

El módulo de enchufe inteligente está compuesto por un Wemos D1 mini, un relé, la fuente de alimentación y la PZEM.

Estos elementos, al igual que en el resto de módulos, vienen distribuidos en una PCB intentando la mayor eficiencia en cuanto a tamaño que ocupan se refiere.



Ilustración 4.31. Cara A del módulo del enchufe inteligente

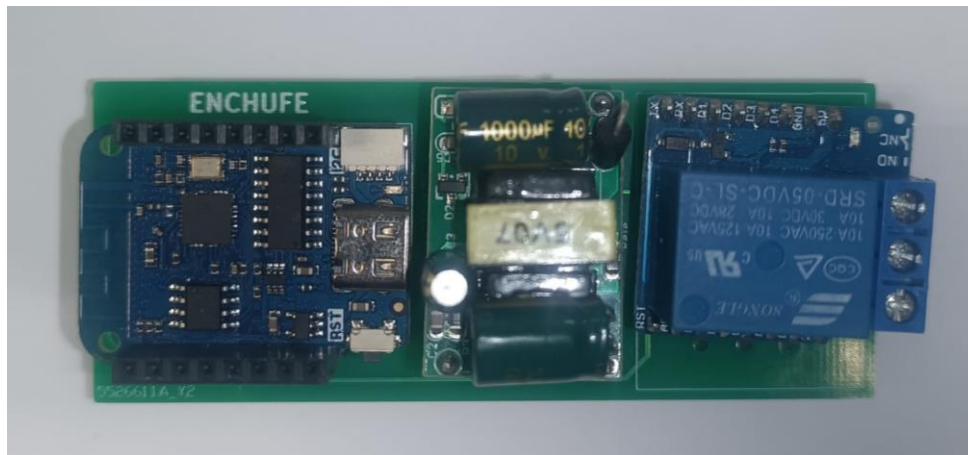


Ilustración 4.32. Cara B del módulo del enchufe inteligente

Librerías

Para el correcto funcionamiento del código del prototipo en Arduino IDE, se han tenido que incluir las siguientes librerías:

<FirebaseArduino.h>: permite la lectura y escritura entre Arduino IDE y Firebase.

<ESP8266HTTPClient.h>.

<ESP8266WiFi.h>.

configuran la conexión vía Wi-Fi.

<ArduinoJson.h>: necesaria para la comunicación con Firebase.

<PZEM004Tv30.h>: configura el módulo PZEM para obtener la potencia consumida.

<Time.h>.

<TimeSib.h>.

<WiFiUdp.h>.

necesarias para obtener la hora local a través de internet.

Programación

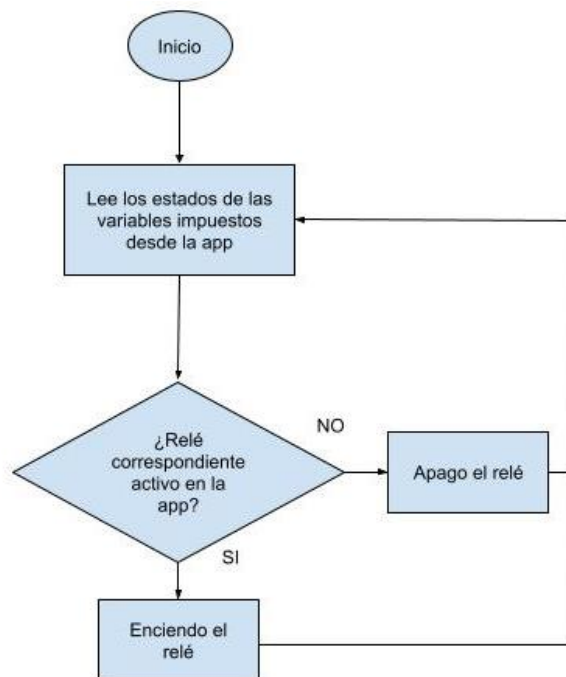


Ilustración 4.33. Diagrama de flujo del enchufe inteligente

Funcionamiento

Arduino IDE leerá las variables de Firebase y determinará el enchufe en el cual tiene que trabajar. Una vez que fija el ID del módulo, realiza la comparativa entre la potencia previamente mencionada en los otros módulos para determinar el modo en el que trabajar. Estos son los dos distintos modos según la cantidad de potencia total con respecto a la contratada:

- Hasta un 60% de la potencia contratada, trabajaría de modo normal, sin ninguna restricción y tal cual la programación. En función del estado de la variable en la app y, por tanto, en la base de datos, se activará o desactivará el módulo que le corresponda.

Especial mención a que se puede hacer una programación de funcionamiento especial si un aparato necesitase un funcionamiento distinto al de cualquier otro enchufe como puede ser un calentador o un frigorífico, por ejemplo. Como el calentador entra dentro de una de las habitaciones que se han programado, tiene una programación especial que, además, tenga en cuenta de si lleva 1 hora encendido, se apague automáticamente y, si lleva 6 horas apagado, se encienda. Si se cambiase manualmente el estado desde la app, cambiaría el tiempo de apagado o encendido de este, según al estado que cambie.

- En un rango superior al 60%, el módulo empezaría a comprobar si el dispositivo pertenece a la habitación que tiene menor orden de prioridad y, en ese caso, apagaría el dispositivo que gobierna el enchufe. Tras esto, realiza una breve espera para comprobar que se ha bajado del 60%. Si no, comprobaría la siguiente habitación con el menor orden de prioridad.

En el caso de que fuese algún dispositivo con cualquier funcionalidad especial como puede ser un frigorífico, se le añadiría en código que, si fuese este, no se apagase ya que, por ejemplo, este dispositivo debe permanecer siempre funcionando. En el caso del calentador, no se ha considerado que tenga un especial funcionamiento ya que, si la habitación está en menor orden de prioridad, es que no nos preocupa que esté o no encendido.

4.3 Diseño de PCBs

Como se ha estado comentando anteriormente, cada módulo del proyecto ha sido diseñado con unos elementos específicos que se han visto ordenados sobre placas de circuito impreso (PCBs).

Para ello, se han diseñado cuatro PCBs distintas para cada módulo utilizando el programa KiCad.

En KiCad no existe la huella de elementos como la fuente de alimentación o algunos otros por lo que han tenido que ser creadas tomando medidas reales y dibujándolas.

Primero, se crea el símbolo en una nueva librería. Esto servirá posteriormente para el conexionado eléctrico.

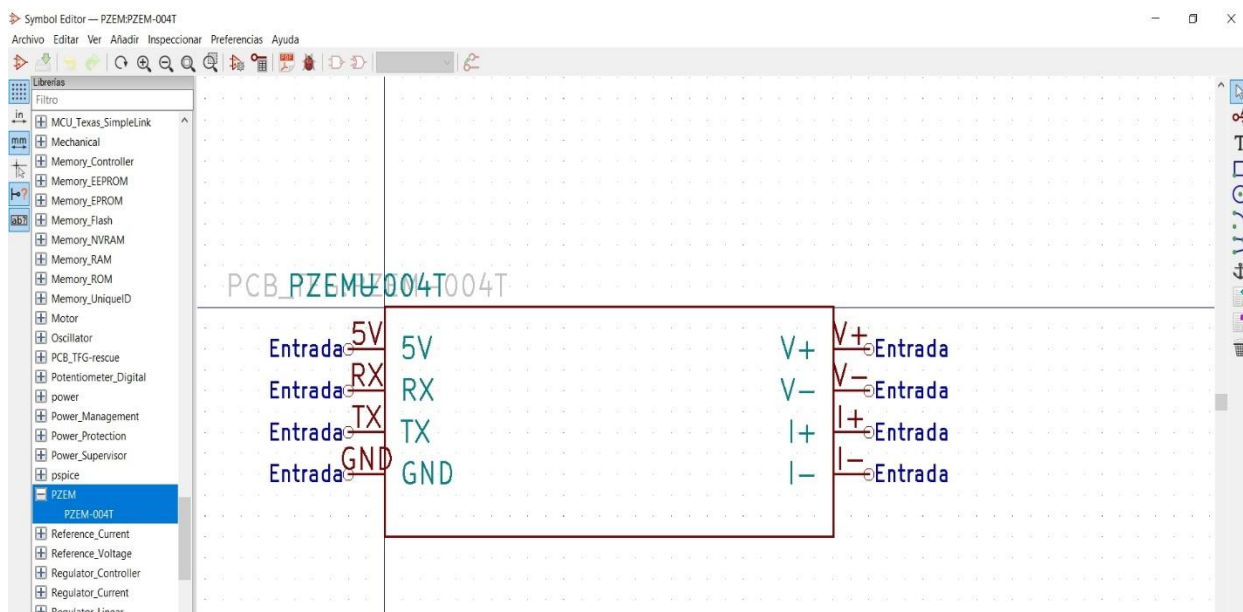


Ilustración 4.34. Editor de símbolos de KiCad (PZEM-004T)

Con las medidas reales tomadas, se dibuja la huella. Esto sería el lugar físico que ocuparía en la PCB a la hora de colocar los dispositivos.

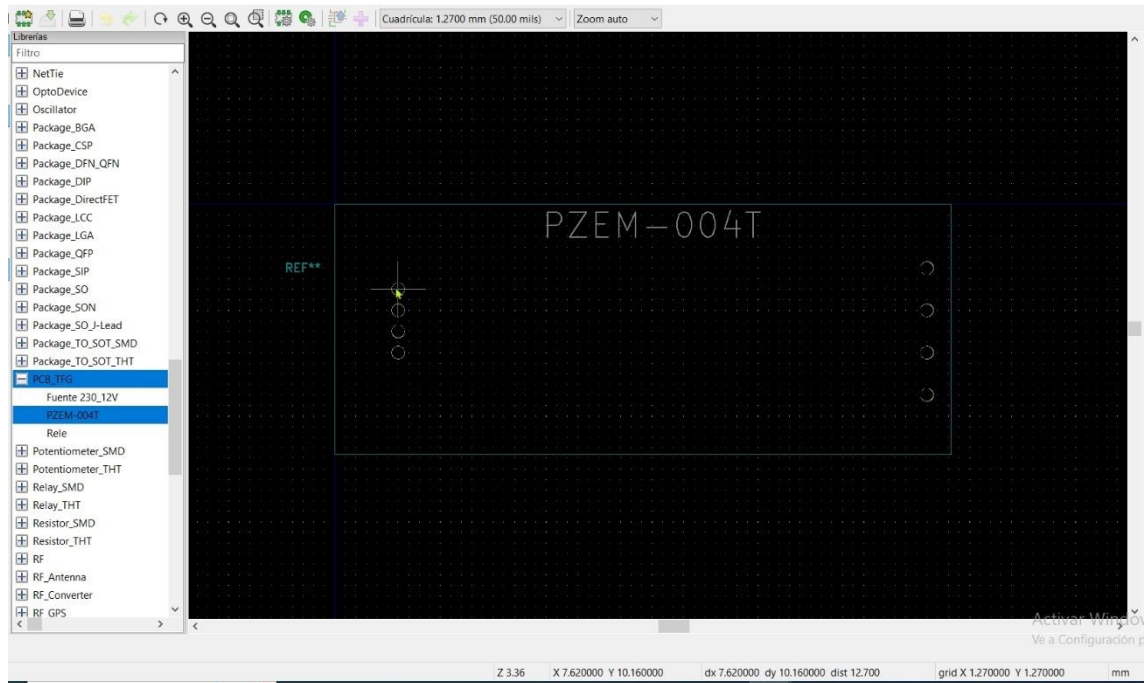


Ilustración 4.35. Editor de huellas de KiCad (PZEM-004T)

Una vez dibujados todos los componentes y sus respectivas huellas, se asocian los símbolos con las huellas.

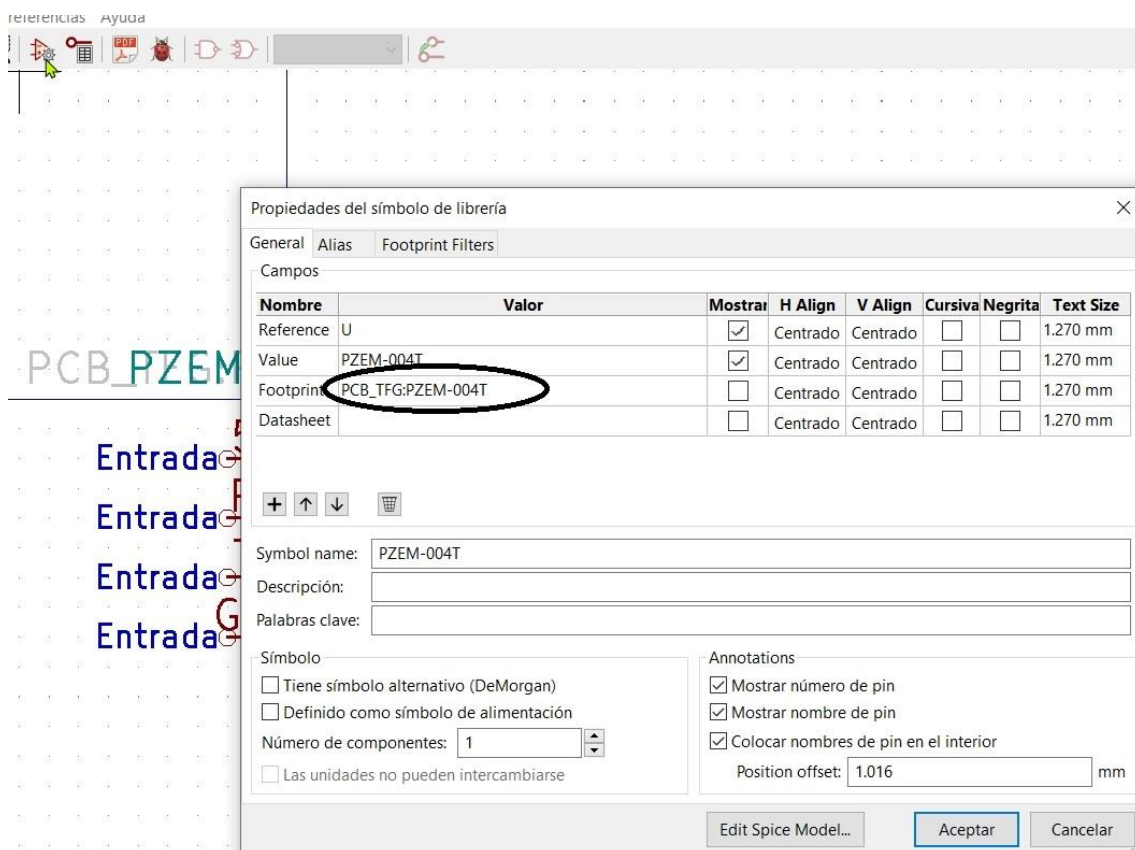


Ilustración 4.36. Propiedades del símbolo de KiCad

Luego, hay que hacer el esquema eléctrico del conexionado que llevará internamente la PCB previo a hacer las conexiones manuales. Estas imágenes muestran las conexiones eléctricas de los cuatro módulos:

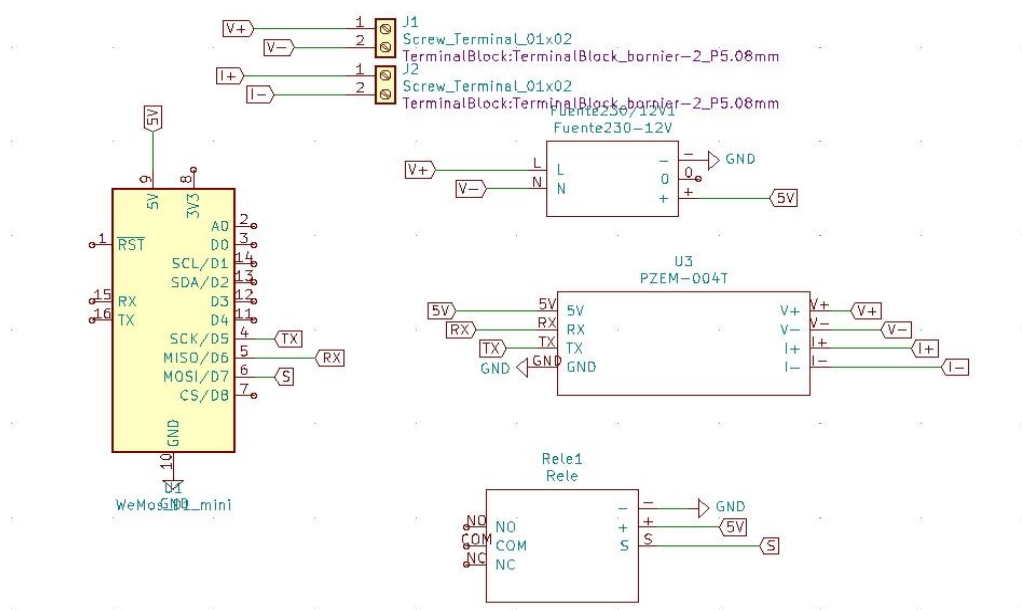


Ilustración 4.37. Esquema de conexionado eléctrico del enchufe inteligente

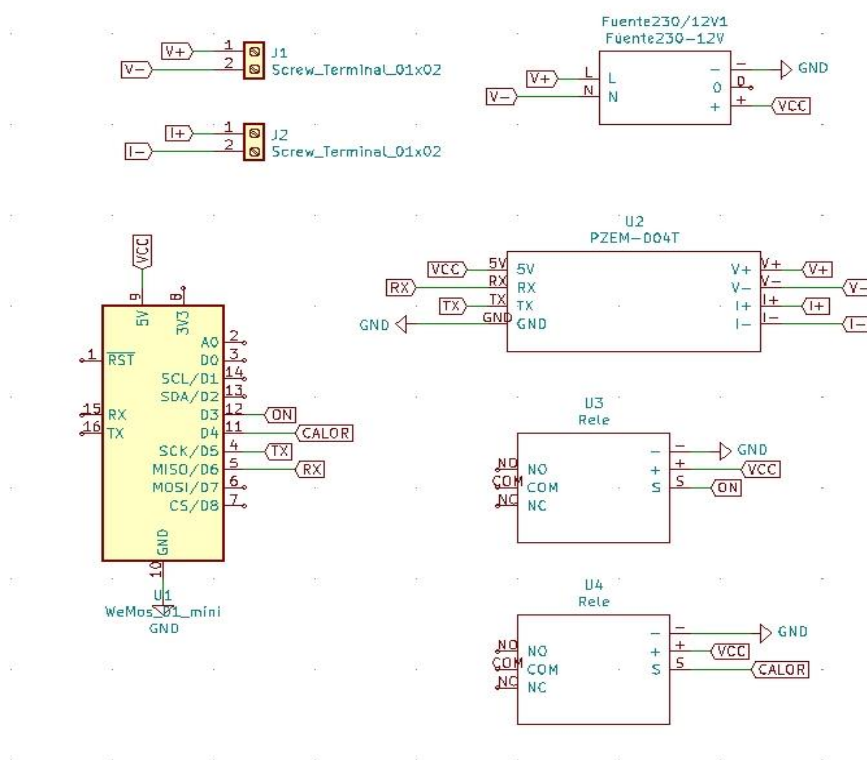


Ilustración 4.38. Esquema de conexionado eléctrico de la climatización

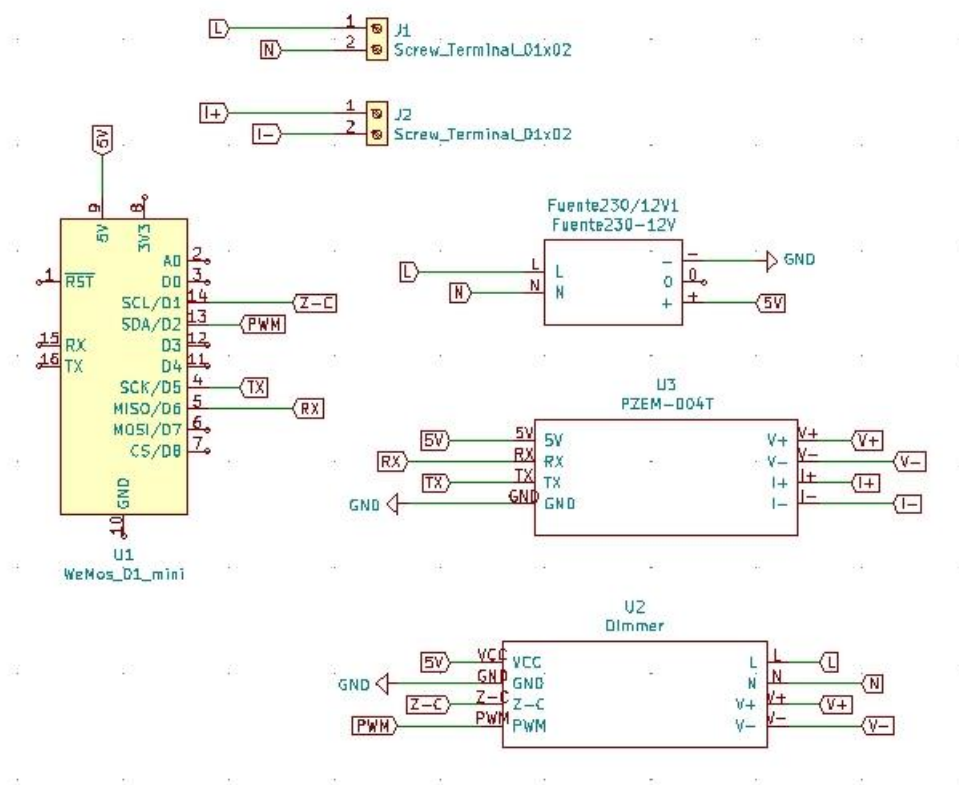


Ilustración 4.39. Esquema de conexionado eléctrico de la iluminación

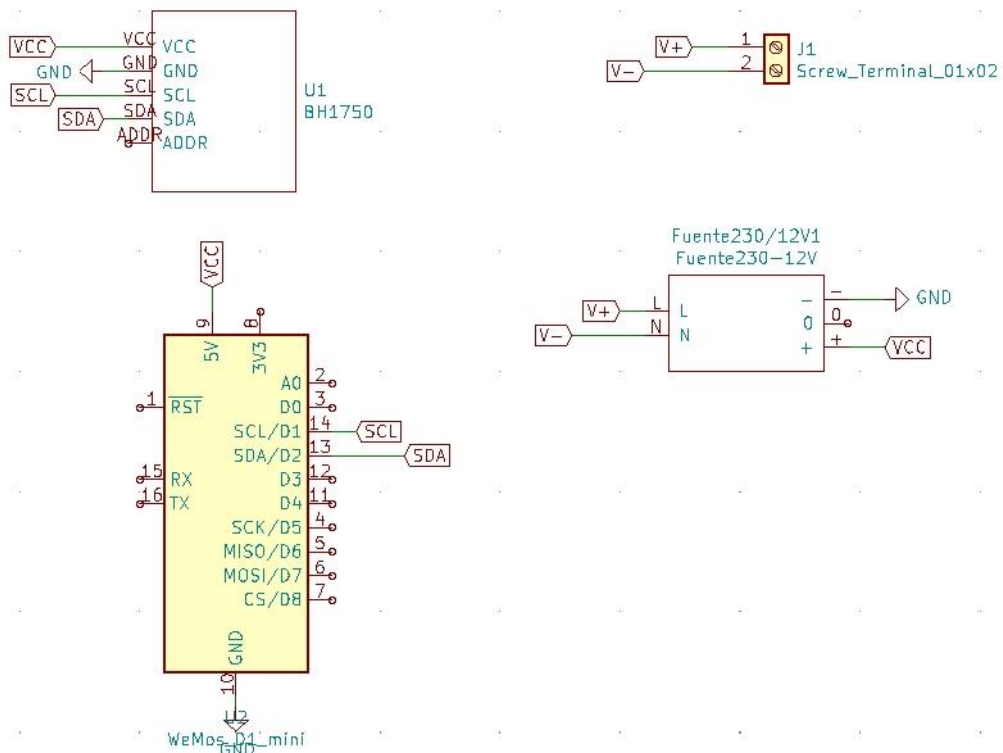


Ilustración 4.40. Esquema de conexionado eléctrico del sensor de luminosidad

Tras hacer el conexionado, se puede hacer el cableado virtual para concluir el diseño de las PCBs. A continuación, se muestran la organización final dada en 2D e imágenes 3D de la vista previa de las PCBs finales de cada módulo. También se incluye en las imágenes 3D un modelo del Wemos D1 mini.

Enchufe

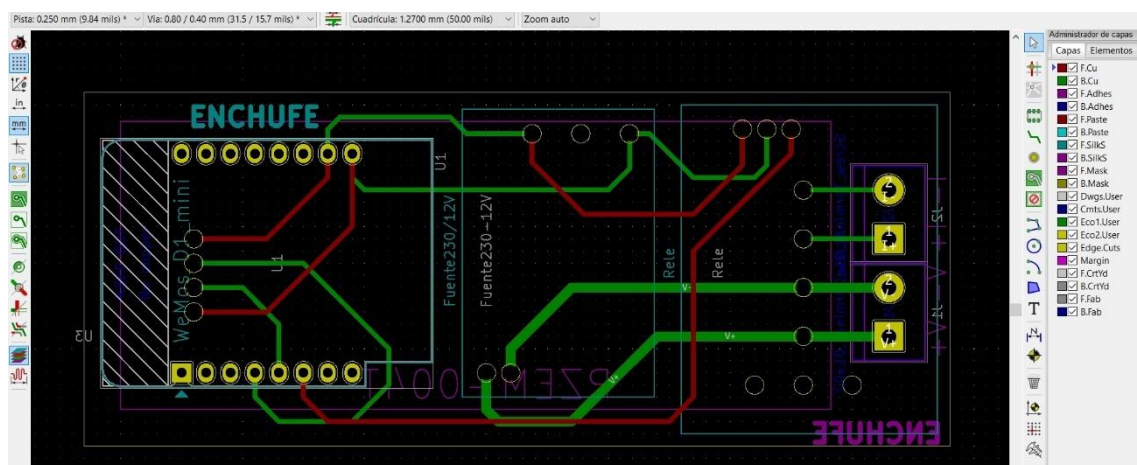


Ilustración 4.41. Layout Editor del conexionado eléctrico del enchufe inteligente

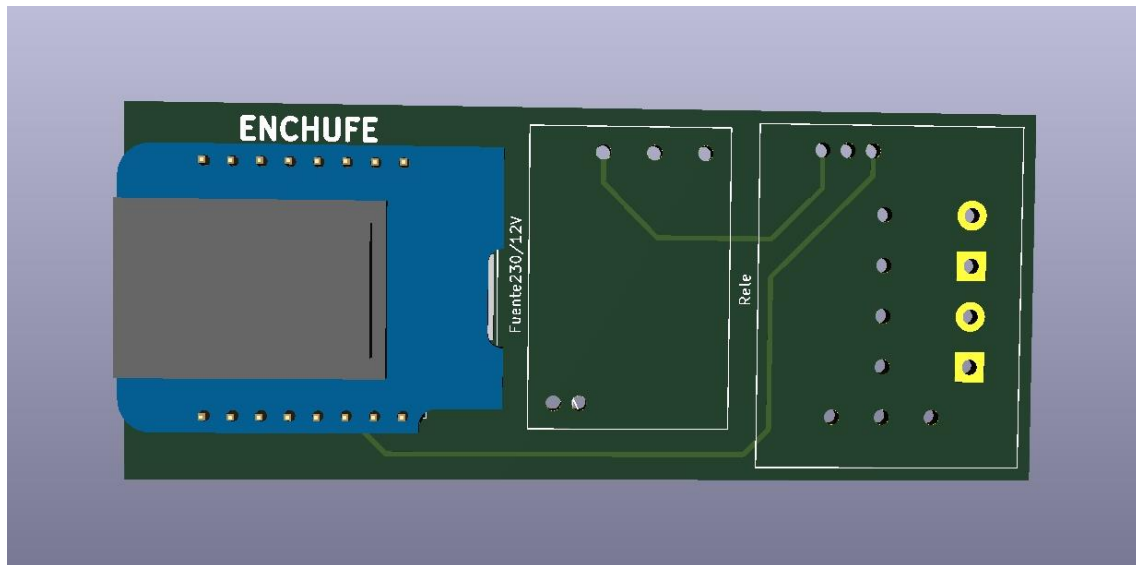


Ilustración 4.42. Modelo 2D del enchufe inteligente – cara B

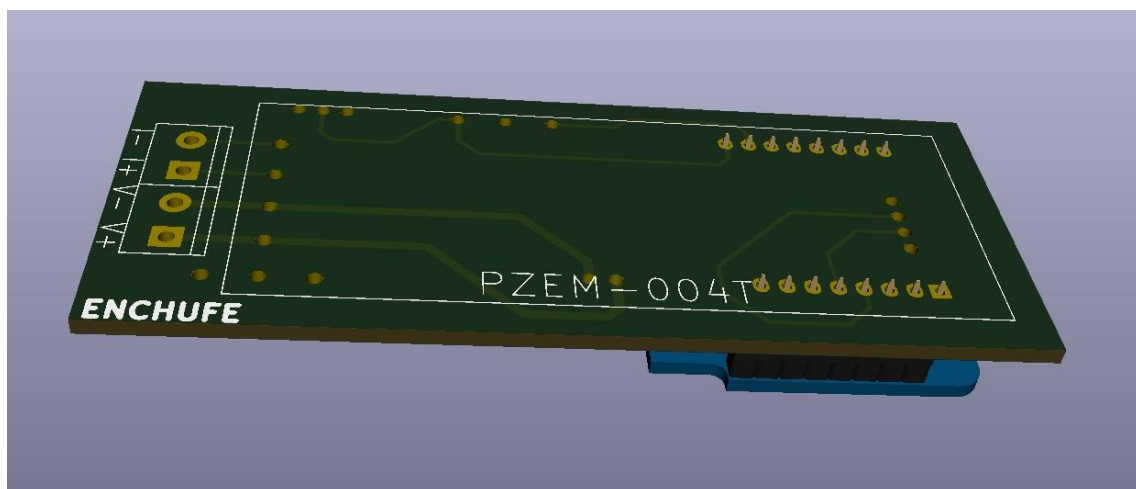


Ilustración 4.43. Modelo 2D del enchufe inteligente – cara A

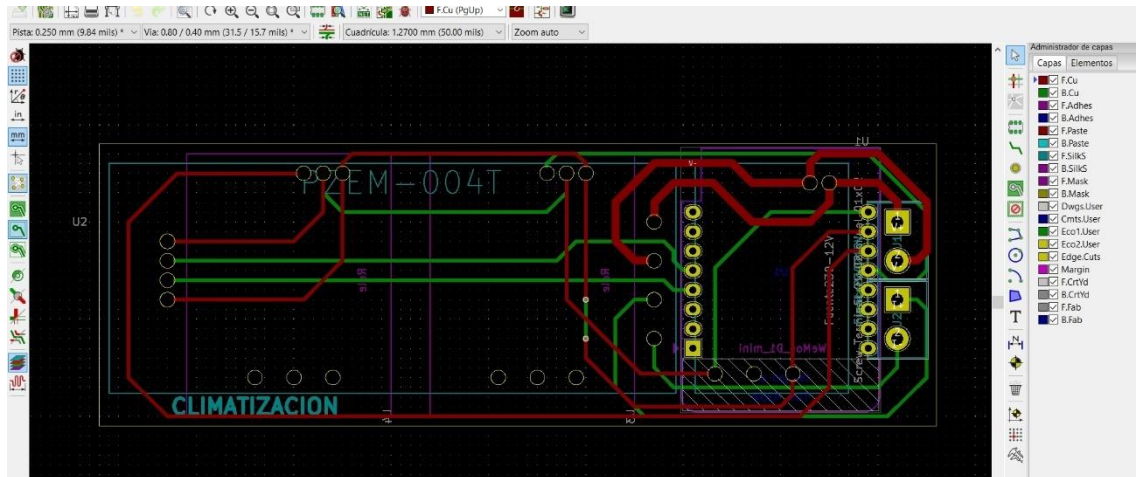


Ilustración 4.44. Layout Editor del conexionado eléctrico de la climatización

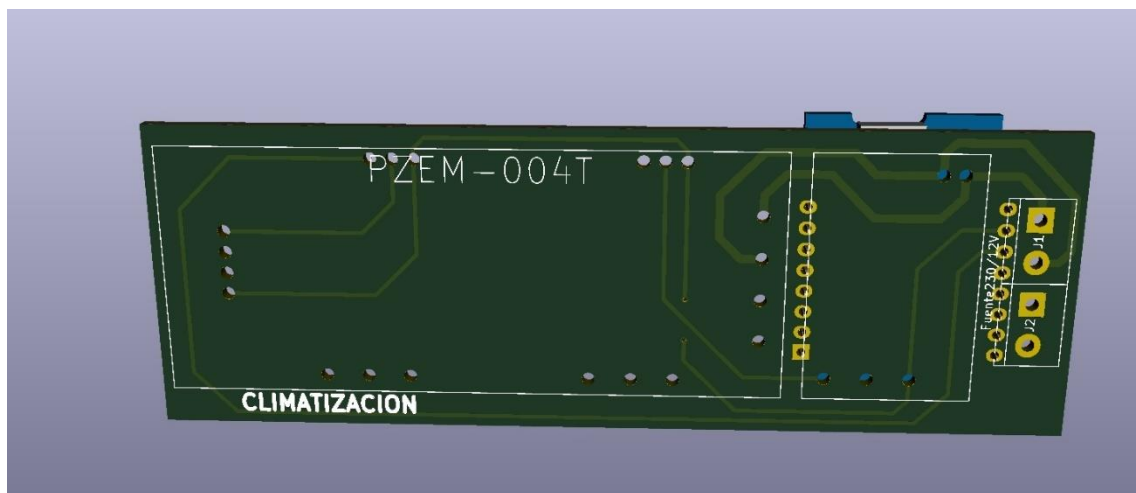


Ilustración 4.45. Modelo 2D de la climatización – cara A

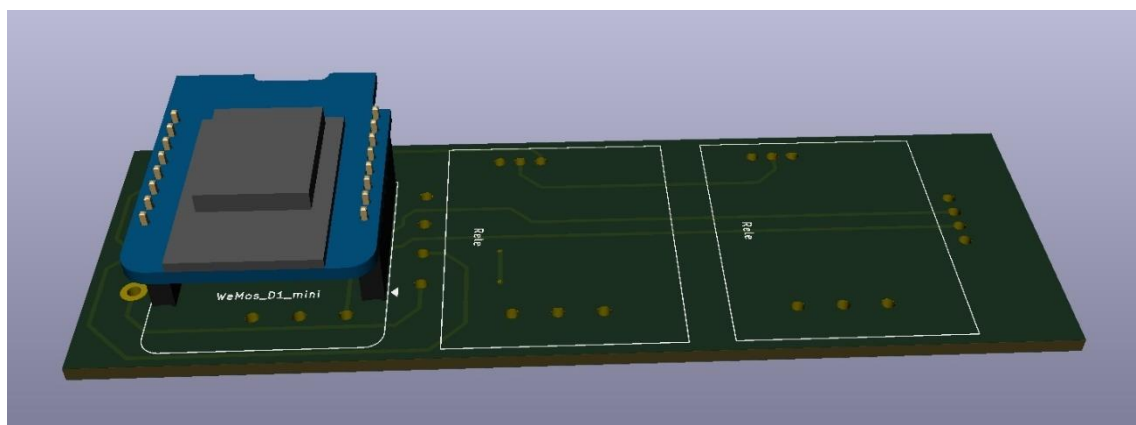


Ilustración 4.46. Modelo 2D de la climatización – cara B

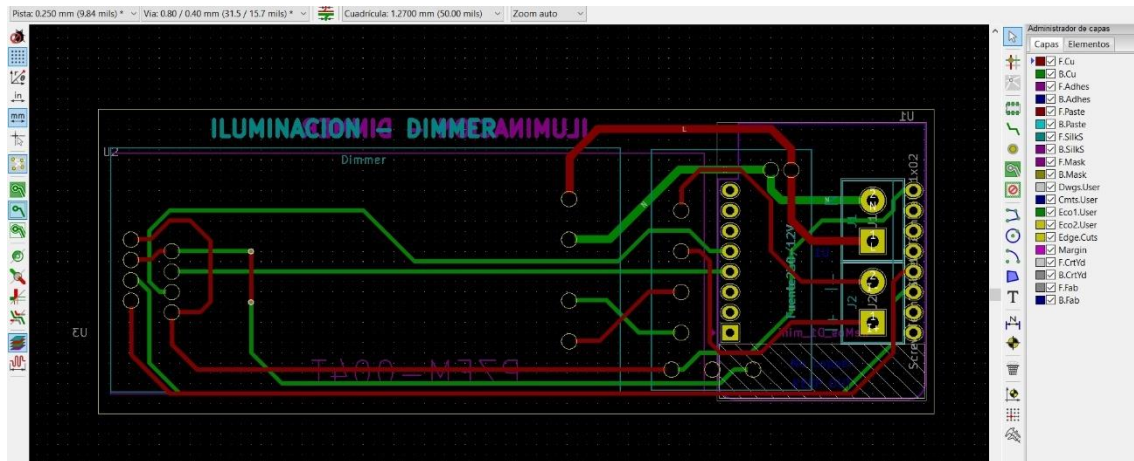


Ilustración 4.47. Layout Editor del conexionado eléctrico de la iluminación

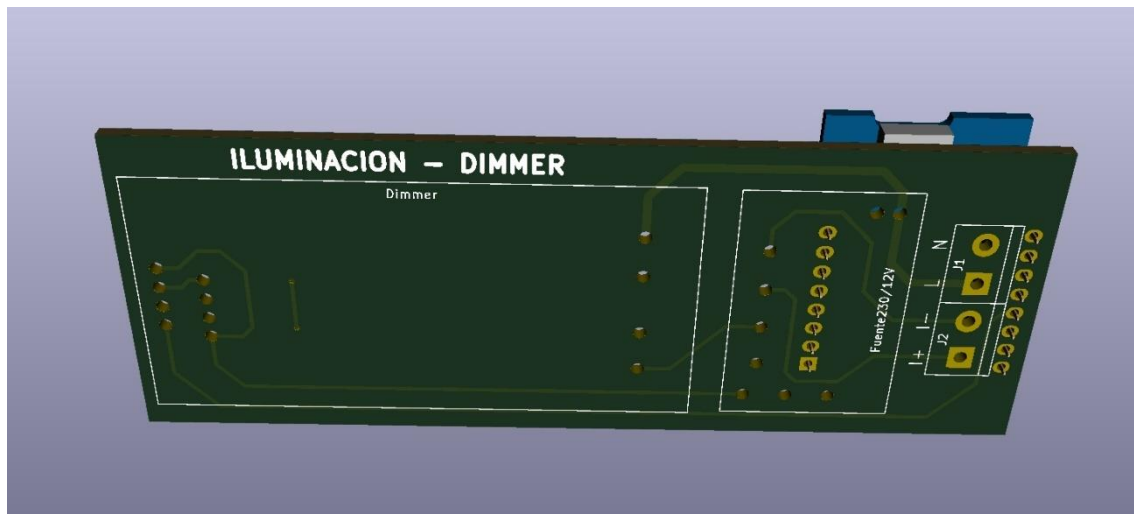


Ilustración 4.48. Modelo 2D de la iluminación – cara B

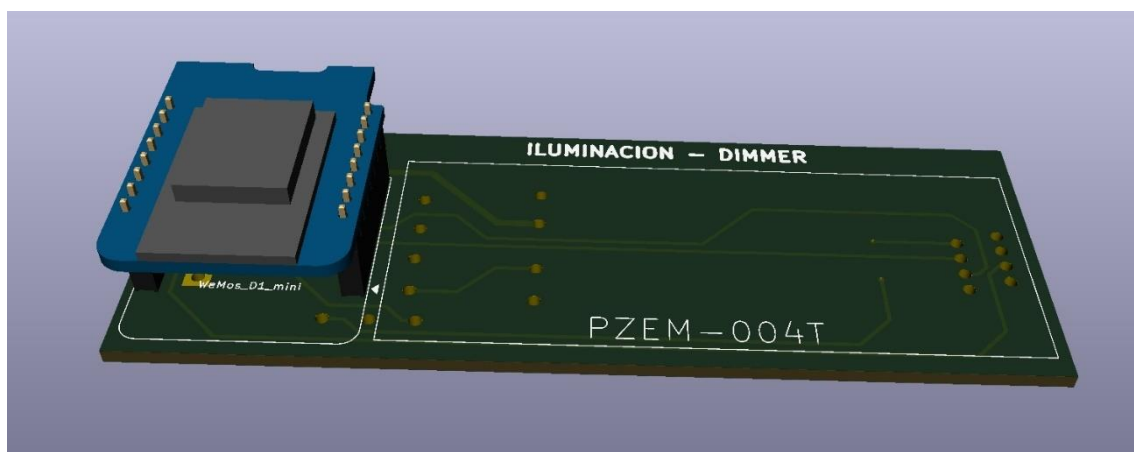


Ilustración 4.49. Modelo 2D de la iluminación – cara A

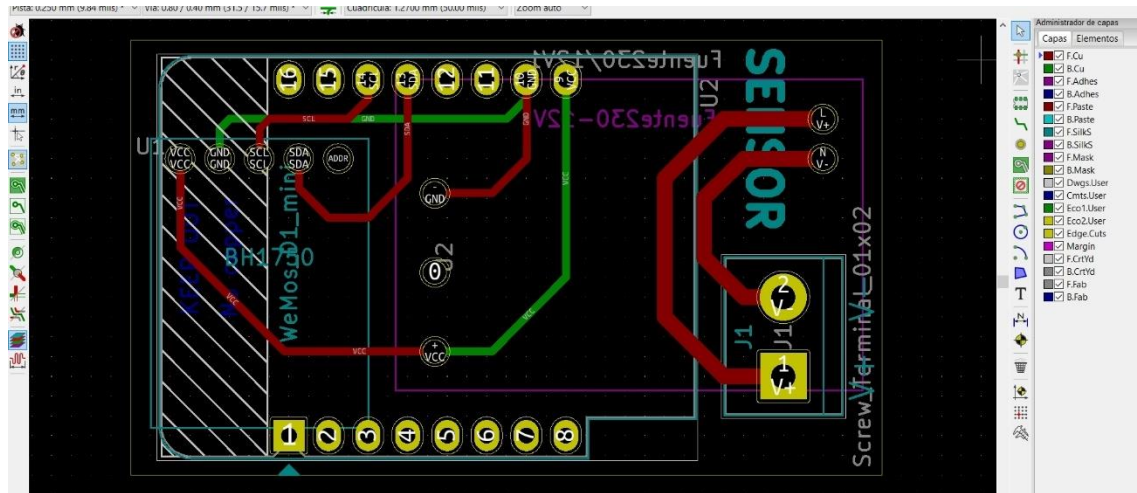


Ilustración 4.50. Layout Editor del conexionado eléctrico del sensor de luminosidad

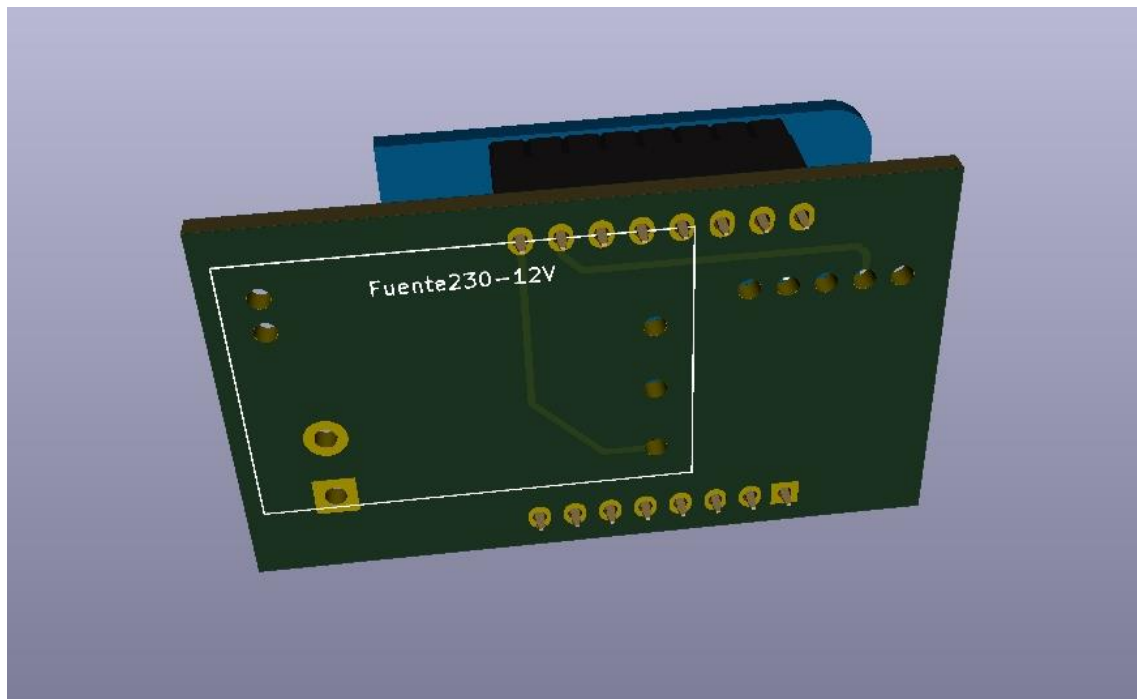


Ilustración 4.51. Modelo 2D del sensor de luminosidad – cara B

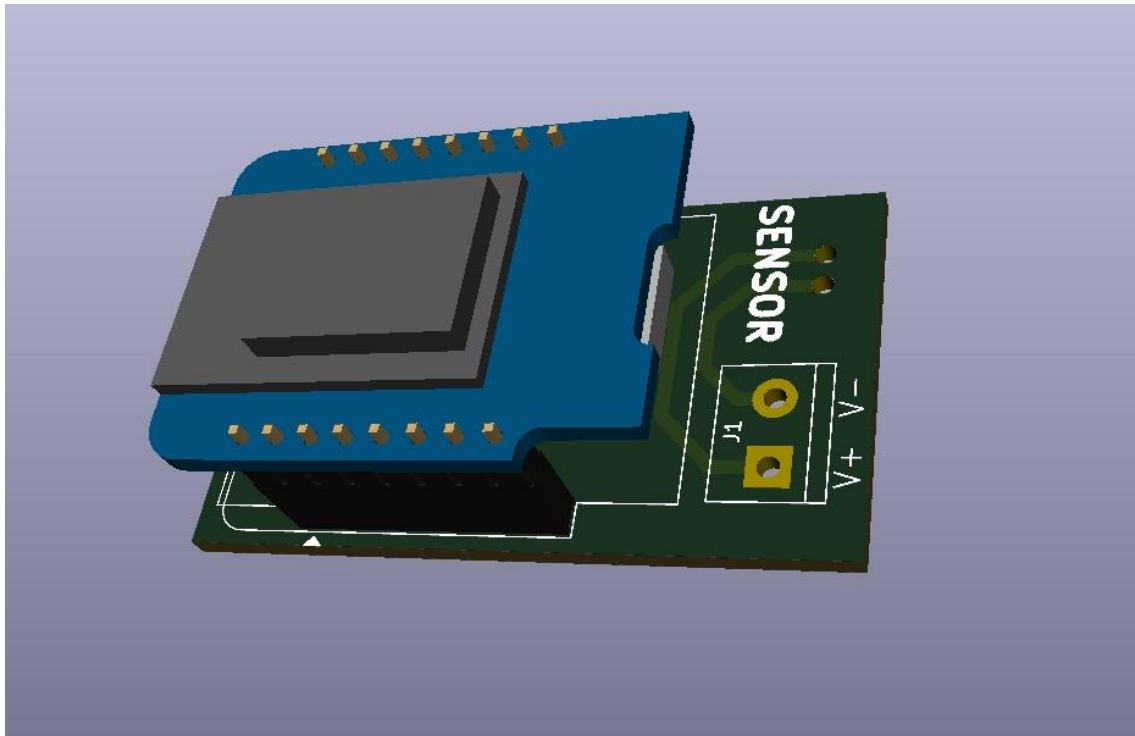


Ilustración 4.52. Modelo 2D del sensor de luminosidad – cara A

5. PRUEBAS Y RESULTADOS DEL PROYECTO

Para probar el funcionamiento de estos módulos con el medidor PZEM-004T, se han realizado primero unas pruebas con una luminaria incandescente y obteniendo solo la potencia con Arduino, se obtienen los siguientes datos durante unos 5 minutos de pruebas. La conexión se ha hecho siguiendo el siguiente esquema:

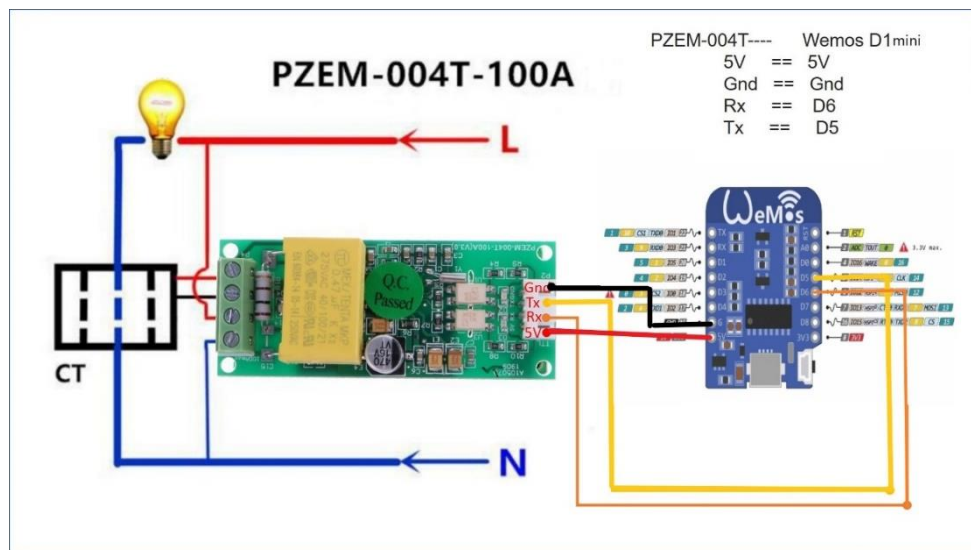


Ilustración 5.1. Esquema de conexionado del módulo PZEM-004T con Arduino



Ilustración 5.2. Configuración de la prueba con el secador de pelo

Hora	Potencia (W)	Estado (1=ON)
17:37:14	1880.5	1
17:37:26	1881.3	1
17:37:38	0.1	0
17:37:51	0	0
17:38:03	0	0
17:38:15	0	0
17:38:27	1881.0	1
17:38:40	1880.6	1
17:38:52	1880.4	1
17:39:04	0.1	0
17:39:17	0	0
17:39:29	0	0
17:39:41	1880.4	1
17:39:54	1880.9	1
17:40:06	0	0
17:40:19	0	0
17:40:32	1880.4	1
17:40:45	0	0
17:40:57	1881.4	1
17:41:09	0.1	0
17:41:21	0	0
17:41:33	1880.2	1
17:41:56	1880.2	1
17:42:08	0	0
17:42:20	0	0

Tabla 5.1. Datos obtenidos de la prueba con el secador de pelo

Una vez comprobado el funcionamiento del módulo PZEM, se pasan a realizar las pruebas relacionadas al funcionamiento del proyecto.

El prototipo está diseñado para calcular el nivel de potencia consumida por cada periférico de nuestro sistema y ser capaz de encender o apagar cuando se alcancen ciertos niveles de potencia consumida con respecto a la potencia contratada en el hogar, según los rangos establecidos en la programación en el Arduino IDE, como se ha explicado anteriormente.

Por otro lado, la app muestra en la pestaña de consumo, distintos tipos de gráficas que muestran el consumo de potencia actual de cada uno de los periféricos y así, aunque estemos en el rango de trabajo de por debajo del 60% del total de la potencia contratada, podemos optar por apagar manualmente, a través de la app, el periférico que deseemos.

Ya que, para realizar las pruebas con un rango de potencia consumida superior al 60%, se ha variado la potencia contratada del hogar y reduciéndola a una cercana a la consumida para forzar esas supuestas situaciones en el hogar.

Una de las pruebas consiste en encender una lámpara LED funcionando en modos automático y manual. Para ello, dependerá en el modo automático del nivel del sensor de luminosidad, separado de la luminaria para no falsear datos. Estos niveles se irán cambiando a voluntad haciendo incidir más luz en el sensor o tapándolo para que apenas reciba.

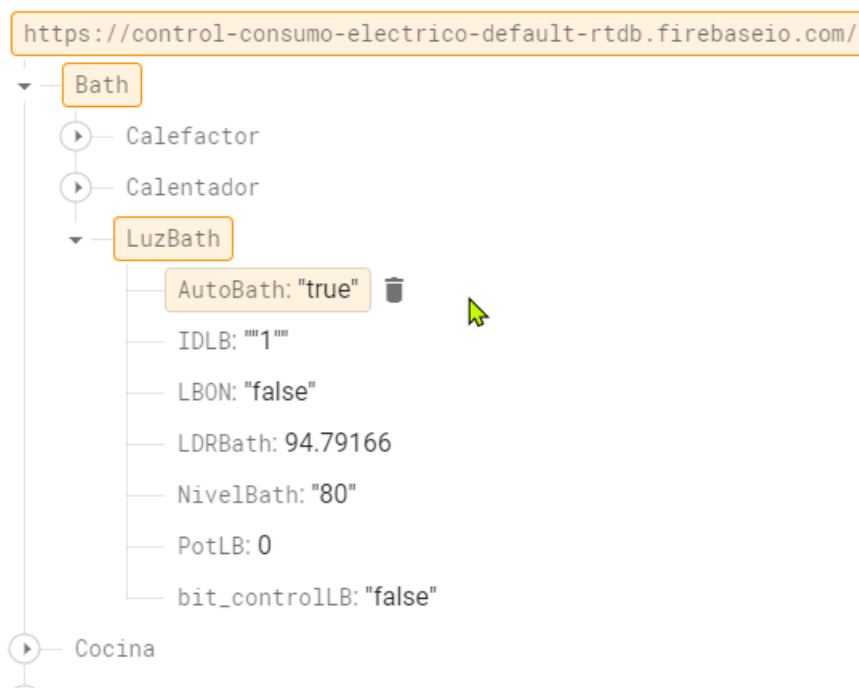


Ilustración 5.3. Activación del modo automático en Firebase

Tras poner el módulo de iluminación en modo automático, como el sensor da valores por debajo de 100 lux, se activará la luminaria con un nivel de intensidad al máximo.



Ilustración 5.4. Activación de las demás variables de LuzBath en Firebase

Para terminar la prueba, se hace incidir la luz de la linterna de un teléfono móvil, modificándose así la cantidad de lux captada por el sensor y, como excede el valor de 1200, desactiva la luminaria.

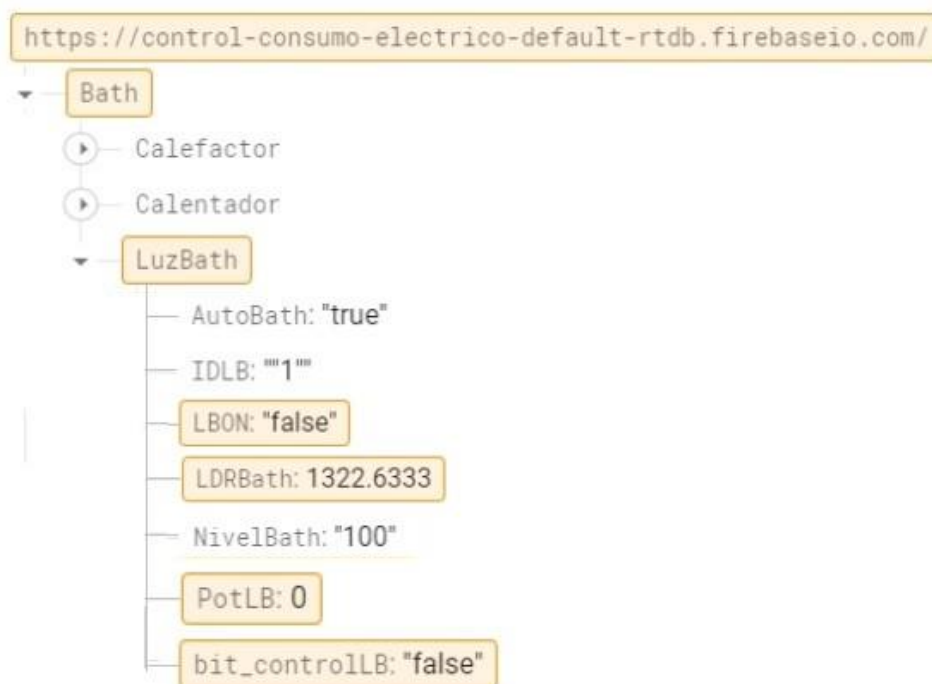


Ilustración 5.5. Apagado de luminaria por valores altos del sensor en Firebase

La otra prueba comprobará el funcionamiento del enchufe inteligente, siendo este capaz de desactivar su relé, aunque el módulo esté activado, cuando la habitación donde se encontraría está en menor prioridad que las demás.



Ilustración 5.6. Estado de las variables antes de la prueba con el enchufe en Firebase

Primero, se enciende la televisión del salón desde la app que, para la prueba, consistirá en encender la bombilla LED utilizada en la prueba anterior.



Ilustración 5.7. Activación de TeleSalon en Firebase

Tras activarla, medirá la potencia consumida y la sumará a la total.



Ilustración 5.8. Activación del resto de variables y cálculo de la potencia de la tele en Firebase

Para que el enchufe considere que se ha superado el 60% de potencia contratada en el hogar, se modificará desde la app para que sea cercana a la potencia consumida por la bombilla.

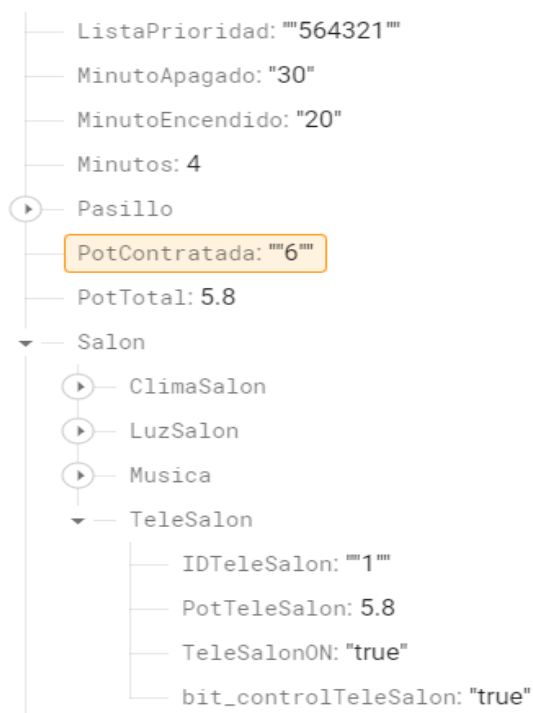


Ilustración 5.9. Cambio de la potencia contratada para realizar la prueba

Como el salón tiene el menor orden de prioridad, desconectará la televisión tras unos instantes y saliendo de ese rango superior al 60% de la potencia contratada.



Ilustración 5.10. Desactivación de las variables debido a una potencia consumida elevada

Para plasmar en una gráfica el comportamiento del enchufe en función de la potencia consumida alcanzada, se ha hecho una última prueba. Esta consiste en activar cada uno de los enchufes de las dos habitaciones que se han tenido en cuenta para el prototipo y ver el funcionamiento de este teniendo una de esas dos habitaciones en menor orden de prioridad que, en este caso, será el salón.

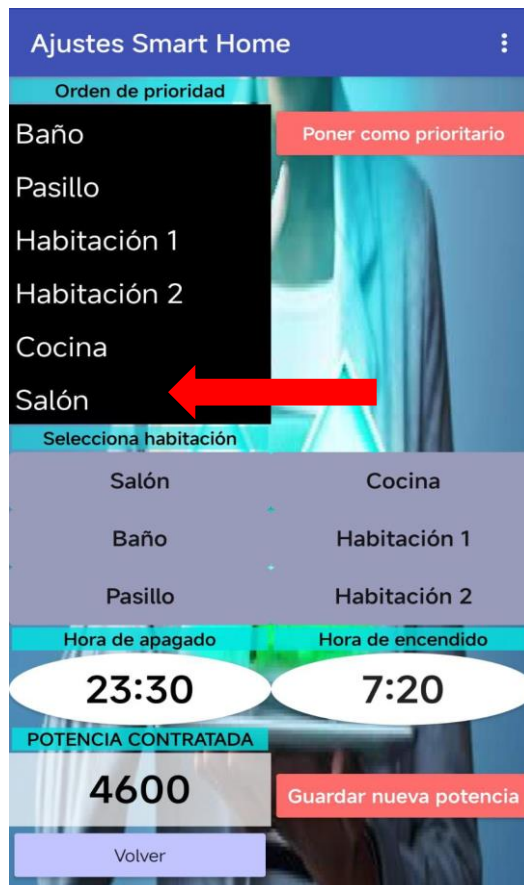


Ilustración 5.11. Pestaña 'Ajustes Smart Home' de la app con el salón en menor prioridad

Como también se aprecia en la imagen, se ha estipulado una potencia contratada en el hogar de 4600 W. Por lo tanto, cualquier valor igual o superior al 60% de esta cantidad, que son 2760 W, hará que el enchufe actúe y empiece a desconectar los dispositivos del salón.

A continuación, la siguiente tabla registra las potencias leídas en la base de datos a la hora de activar cada enchufe correspondiente. Esta vez, las lecturas de las potencias consumidas por cada enchufe se han impuesto como un valor fijo, al contrario que la anterior prueba con la iluminación cuyos valores de potencia leídos eran tal cual los proporcionaba el módulo PZEM-004T.

HORA	POT. TOTAL	Porcentaje	PotTeleSalon	PotMusica	PotCalef	PotCalen
17:38	0	0.0%	0	0	0	0
17:39	400	8.7%	400	0	0	0
17:40	400	8.7%	400	0	0	0
17:41	1400	30.4%	400	1000	0	0
17:42	1400	30.4%	400	1000	0	0
17:43	2900	63.0%	400	1000	1500	0
17:44	1900	41.3%	400	0	1500	0
17:45	1900	41.3%	400	0	1500	0
17:46	3150	68.5%	400	0	1500	1250
17:47	2750	59.8%	0	0	1500	1250
17:48	2750	59.8%	0	0	1500	1250
17:49	3750	81.5%	0	1000	1500	1250
17:50	2750	59.8%	0	0	1500	1250
17:51	2750	59.8%	0	0	1500	1250
17:52	4150	90.2%	400	1000	1500	1250
17:53	2750	59.8%	0	0	1500	1250
17:54	2750	59.8%	0	0	1500	1250

Tabla 5.2. Valores de potencia de cada enchufe en cada minuto

El proceso de obtención de estos datos y explicación de los sucesos es el siguiente:

- **17:38.** Todos los enchufes están desconectados.
- **17:39.** Se activa el enchufe de la televisión del salón (400 W).
- **17:41.** Se activa el enchufe del equipo de música del salón (1000 W).
- **17:43.** Se activa el enchufe del calefactor del baño (1500 W).
- **17:44.** Al haberse superado el 60% de la potencia contratada (63%) con los tres enchufes activados, procede a desconectar para bajar de este porcentaje. En este caso, desconecta el equipo de música.
- **17:46.** Se activa el calentador del baño (1250 W).
- **17:47.** Como se ha vuelto a superar el 60% (la potencia consumida es un 68.5% de la contratada) y el salón es el que está en menor orden de prioridad, desactiva la televisión del salón.
- **17:49.** Se vuelve a activar el enchufe del equipo de música.
- **17:50.** Como se ha alcanzado un valor del 81.5%, lo vuelve a desactivar.
- **17:52.** Se activan a la vez tanto el enchufe del equipo de música como el de la televisión del salón.
- **17:53.** Tras calcular que el nuevo porcentaje de potencia consumida es del 90.2%, desconecta ambos para que este baje del 60%.

Para clarificar todos estos sucesos, se añade la siguiente gráfica de los datos obtenidos, pudiendo observar que cuando el valor de potencia consumida sobrepasa del 60% de potencia contratada, inmediatamente actúa desactivando el enchufe que le corresponda.



Ilustración 5.12. Gráfica del comportamiento de los enchufes con respecto a la potencia

Por último, mencionar que en todo momento se puede monitorizar el consumo actual desde la app en la pantalla de 'ConsumoNow'. Además, se pueden seleccionar entre varios estilos de gráficos, así como elegir diferentes escalas por si quisiera comparar las potencias más pequeñas. Aún así, con el botón de 'Cargar Datos', la gráfica se escalará automáticamente en función del valor mayor.

Este ejemplo de gráfica es una captura de la aplicación justo en el momento en el que están los cuatro enchufes inteligentes activados y antes de que se desconecten, es decir, en la hora 17:52 de la prueba anterior descrita.

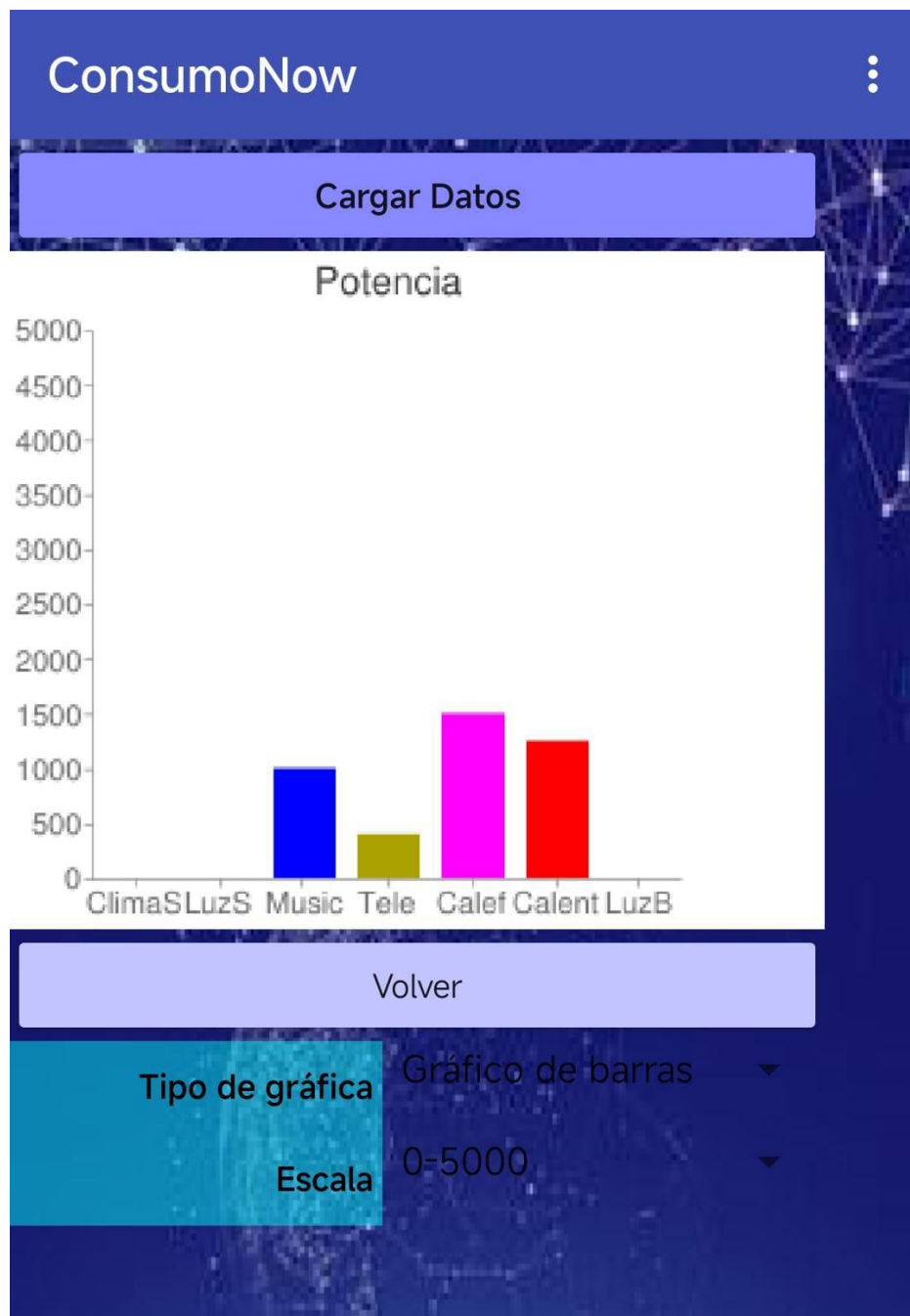


Ilustración 5.13. Gráfica de la potencia en el minuto 17:52 de la prueba en ConsumoNow de la app

6. CONCLUSIONES

Con este proyecto se ha intentado domotizar una vivienda a la vez que se lleva un control sobre el consumo de cada uno de los periféricos usados en la domótica del hogar. Todo esto usando materiales de bajo costo y utilizando herramientas como Arduino y MIT App Inventor, las cuales son de código abierto, sumado al plan inicial sin coste de Firebase, que es suficiente para la envergadura de nuestro proyecto, ha hecho que se cree un prototipo versátil, fácil de realizar y utilizar y bastante competente.

La expansión del Internet de las Cosas ha servido como la principal fuerza impulsora detrás de este proyecto. Este ha sido increíblemente instructivo en el mundo académico, enseñando al autor nuevas facetas del Internet de las cosas, nuevas tecnologías y cómo usar herramientas que antes no conocía con facilidad. La automatización del hogar ha sido un entorno que ha requerido investigación para que realmente se pudiera usar a diario. Se ha avanzado en la programación tanto del ESP8266 como del Arduino IDE, así como el manejo de las herramientas utilizadas como MIT App Inventor y Firebase.

Estas pruebas han demostrado el correcto funcionamiento de subida y lectura de datos de nuestra base de datos en Firebase, el correcto cambio de modo desde nuestra app y una lectura correcta de la potencia consumida por nuestros módulos con el posterior comportamiento de estos en base al porcentaje de potencia consumida respecto a la contratada.

Como conclusión personal, este trabajo marca el fin del grado de electrónica no sin antes tener que poner en práctica muchos de los conocimientos vistos tales como programación, montaje de circuitos y diseño de PCBs. Un cúmulo de conocimientos exteriorizado en forma de un proyecto tangible y habiendo sido elegido por el gusto personal en el ámbito de la domótica del hogar.

BIBLIOGRAFÍA

Gracia, M. (s.f.). IoT - Internet of Things. URL:

<https://www2.deloitte.com/es/es/pages/technology/articles/loT-internet-of-things.html>

Bisha. (2020, 25 de abril). Monitorizando el consumo (por ejemplo el de nuestra vivienda) por cuatro duros: Construir un PZEM y vincularlo a domoticz. URL: <https://domotuto.com/monitorizando-el-consumo-de-nuestra-vivienda-construir-un-pzem-y-vincularlo-a-domoticz/>

Domótica integrada. (2018, 26 de junio). Claves y consejos para la automatización del hogar. URL: <https://domoticaintegrada.com/automatizacion-del-hogar/#:~:text=%C2%BFQu%C3%A9%20se%20puede%20automatizar%20en%20el%20hogar%3F,de%20riego%2C%20c%C3%A1maras%20de%20seguridad%E2%80%A6>

Z-Wave. (s.f.). Bienvenido a Z-Wave España. URL: <https://zwave.es/>

Certificados Energéticos. (s.f.). Viviendas inteligentes y eficiencia energética. URL: <https://www.certificadosenergeticos.com/viviendas-inteligentes-eficiencia-energetica/#:~:text=Una%20vivienda%20inteligente%20puede%20consumir,el%20ahorro%20energ%C3%A9tico%20que%20conlleva.>

Ibáñez Herrera, C. (2018). Sistema de autocontrol energético a nivel de usuario. URL: <https://biblus.us.es/bibing/proyectos/abreproy/91801/fichero/TFG-1801-IBA%C3%91EZ.pdf>

Rodríguez de Luis, E. (2019, 5 de marzo). ZigBee y Z-Wave: qué son, en qué se diferencian y que marcas de domótica son compatibles. URL: <https://www.xataka.com/seleccion/zigbee-z-wave-que-que-se-diferencian-que-marcas-domotica-compatibles>

Domingo, P. (2019, 4 de marzo).

Medidas eléctricas en corriente alterna con Arduino y un solo dispositivo (PZEM-004T). URL: <https://didactronica.com/medidas-electricas-en-corriente-alterna-con-arduino-y-un-solo-dispositivo-pzem-004t/>

García, R. (2020, 28 de febrero). Todo lo que permite hacer Bluetooth, la tecnología que vive a la sombra del WiFi. URL: <https://www.adslzone.net/reportajes/tecnologia/bluetooth/>

TotalEnergies. (2021, 24 de septiembre). ¿Cuál es la potencia eléctrica necesaria para mi hogar? URL: <https://www.totalenergies.es/es/hogares/blog/eficiencia-ahorro/cual-es-la-potencia-electrica-necesaria-para-mi-hogar>

Adeva, R. (2020, 9 de febrero). Tecnologías inalámbricas: diferencias y usos de WiFi, Bluetooth, Zigbee y Z-Wave. URL: <https://www.adslzone.net/reportajes/tecnologia/estandares-conexion-inalambrica/>

Roldán, A.M. (s.f.). Dispositivos electrónicos de potencia: Relés. URL:

https://www.ugr.es/~amroldan/enlaces/dispo_potencia/relés.htm

E.P.S Ingeniería de Gijón. (s.f.). Actuadores para domótica e Inmótica. URL:

<http://isa.uniovi.es/docencia/AutomEdificios/transparencias/actuadores.pdf>

Naylamp Mechatronics. (s.f.). Wemos D1 mini ESP8266. URL:

<https://naylampmechatronics.com/esp8266-wemos-d1-mini-esp8266.html>

Wemos.cc. (2019). D1 Boards. URL: <https://docs.wemos.cc/en/latest/d1/index.html>

Sistemas Domóticos. (s.f.). Sistemas domóticos existentes. URL:

https://domoticasistemas.com/tienda/tutoriales/1_sistemas-existentes-tipos-y-estandares.html

Fabiuz7. (2019, 5 de noviembre). Dimable Light for Arduino. URL:

<https://github.com/fabiuz7/Dimmable-Light-Arduino>

Benítez, L. (2021, julio). ¿Qué son las antiguas potencias normalizadas y los nuevos tramos de potencia contratada? URL: <https://www.holaluz.com/blog/cuales-son-las-potencias-normalizadas-actualmente/>

Wemos. (2020, 3 de enero). WeMos D1 & D1 mini Arduino Examples. URL:

https://github.com/wemos/D1_mini_Examples

Esp8266. (s.f.). ESP8266 core for Arduino. URL: <https://github.com/esp8266/Arduino>

FirebaseExtended. (s.f.). Arduino samples for Firebase. URL:

<https://github.com/FirebaseExtended/firebase-arduino>

PaulStoffregen. (s.f.). Time library for Arduino. URL:

<https://github.com/PaulStoffregen/Time>

Gibson Research Corporation. (s.f.). Fingerprints. URL:

<https://www.grc.com/fingerprints.htm>

AliExpress. (s.f.). Módulo de fuente de alimentación Ac dc 220V a 5v 3w HLK-PM01.

URL:

https://es.aliexpress.com/i/33022245121.html?cv=173210&af=442763&aff_platform=link-c-tool&afref=https%3A%2F%2Fwww.google.com%2F&sk=hbH1Nkk&aff_trace_key=d88701e7715948f38474c720b020192a-1591460771876-01336-hbH1Nkk&dp=edb56314e0aa1a0ab425301ffc1de705&terminal_id=a0cfdaf722c34a85bfaafece8ca46b44&aff_request_id=d88701e7715948f38474c720b020192a-1591460771876-01336-hbH1Nkk

Amazon. (s.f.). A0127 PZEM-004T - Multímetro de tensión de corriente (80-260 V, 100

A, transformador de núcleo dividido). URL: [https://www.amazon.es/A0127-PZEM-](https://www.amazon.es/A0127-PZEM-004T-mult%C3%ADmetro-Transformador-Dividido/dp/B07NQ66S82/ref=psdc_1951052031_t2_B08D7PTJ7D)

[004T-mult%C3%ADmetro-Transformador-Dividido/dp/B07NQ66S82/ref=psdc_1951052031_t2_B08D7PTJ7D](https://www.amazon.es/A0127-PZEM-004T-mult%C3%ADmetro-Transformador-Dividido/dp/B07NQ66S82/ref=psdc_1951052031_t2_B08D7PTJ7D)

Fernández, R. (s.f.). Cómo mostrar un gráfico con Google chart api. URL:

https://docs.google.com/document/d/1La_Y4ELmM52YE7-YHd_vMOKBsXHHE05x-5jJrtDNVYo/edit

Castañó Giraldo, S. (2020, 3 de agosto). RELEVADOR con ARDUINO [Manejo de CARGAS Grandes]. URL:

https://www.google.com/search?q=como+funciona+un+rele+arduino&sxsrf=ALiCzsbZq6oGE4UHx95vvFCL3JXwJUJV8w%3A1667582962753&ei=8ktIY5DNLbCFhbIPrJaloAU&oq=como+funciona+un+rele+ar&gs_l=Exgnd3Mtd2l6LXNlcnC4AQP4AQEqAggAMgUQABiABDIGEAAYFhgeMgYQABgWGB4yBhAAGBYHjIGEAAAYFhgewgKEAAAYRxiWBBiwA8ICBxAAGLADGEPcAgQQABhDkAYKSJ4XUMAFWJYLcAF4AcgBAJABAjGbjGgAfgCqgEDMC4z4gMEIEEYAOIDBCBGGACIBgE&sclient=gws-wiz-serp#fpstate=ive&vld=cid:e8841687,vid:81FGI6VVell

Sanjeev, A. (2018, 21 de marzo). How to Use an LDR Sensor With Arduino. URL:

<https://maker.pro/arduino/tutorial/how-to-use-an-ldr-sensor-with-arduino>

Rodríguez Diego, M. (2017, septiembre). Gestión domótica de una casa unifamiliar basada en Arduino. URL: <https://uvadoc.uva.es/bitstream/handle/10324/26328/TFG-P-709.pdf?sequence=1>

AliExpress. (s.f.). Fotorresistor LDR. URL:

https://es.aliexpress.com/item/32867519074.html?spm=a2g0o.productlist.0.0.370212182D5C7l&s=p&ad_pvid=202006071211393364899503279790003088512_2&algo_pvid=8dfa33da-065c-492e-9936-c3e8188af7da&algo_expid=8dfa33da-065c-492e-9936-c3e8188af7da-1&btsid=0ab6fa8115915570992194101e80bc&ws_ab_test=searchweb0_0,searchweb201602_,searchweb201603

AliExpress. Lolin Official Store. (s.f.). LOLIN D1 mini V3.1.0. URL:

<https://es.aliexpress.com/item/32529101036.html>

Pina Beatove, A. (2017, noviembre). DISEÑO DE UN SISTEMA DOMÓTICO CON COMUNICACIÓN WIFI. URL: <https://zaguan.unizar.es/record/69927/files/TAZ-TFG-2017-4220.pdf>

AliExpress. (s.f.). Cargador Micro USB de red por enchufe. URL:

https://es.aliexpress.com/item/4000675631448.html?spm=a2g0o.productlist.0.0.6d9045b3wiANiD&algo_pvid=df5f08b9-397e-4cdd-bc30-18dc156ed4d3&algo_expid=df5f08b9-397e-4cdd-bc30-18dc156ed4d3-30&btsid=0ab6f81615915574310281637e053d&ws_ab_test=searchweb0_0,searchweb201602_,searchweb201603

Firebase. (s.f.). Firebase. URL: <https://firebase.google.com/>

MIT App Inventor. (2012-2020). MIT App Inventor. URL: <http://ai2.appinventor.mit.edu/>

Arduino. (s.f.). Arduino. URL: <https://www.arduino.cc/>

ANEXO

Presupuesto

A continuación, se muestran en unas tablas el desglose del coste de cada módulo de nuestro proyecto:

Iluminación

Componente	Unidades	Precio unitario	Coste total
Wemos D1 mini	1 ud.	3.88 €	3.88 €
PZEM-004T	1 ud.	5.19 €	5.19 €
Anillo toroidal	1 ud.	1.49 €	1.49 €
Fuente de alimentación	1 ud.	1.85 €	1.85 €
AC Light Dimmer Module	1 ud.	4.63 €	4.63 €
PCB	1 ud.	(20 PCBs) 27.49 €	2.75 €
Conector	3 uds.	(10 piezas) 1.19 €	0.36 €
Pines para soldar	36 uds.	(40 piezas) 1.67 €	1.51 €
Horas ingeniero	130 h	15.38 €	1,999.40 €
		TOTAL	2,021.06 €

Tabla A.1. Presupuesto del módulo de iluminación

Sensor

Componente	Unidades	Precio unitario	Coste total
Wemos D1 mini	1 ud.	3.88 €	3.88 €
Fuente de alimentación	1 ud.	1.85 €	1.85 €
Sensor de luminosidad BH1750	1 ud.	1.56 €	1.56 €
PCB	1 ud.	(20 PCBs) 27.49 €	2.75 €
Conector	1 uds.	(10 piezas) 1.19 €	0.12 €
Pines para soldar	25 uds.	(40 piezas) 1.67 €	1.05 €
Horas ingeniero	30 h	15.38 €	461.40 €
		TOTAL	472.61 €

Tabla A.2. Presupuesto del módulo del sensor de iluminación

Enchufe

Componente	Unidades	Precio unitario	Coste total
Wemos D1 mini	1 ud.	3.88 €	3.88 €
PZEM-004T	1 ud.	5.19 €	5.19 €
Anillo toroidal	1 ud.	1.49 €	1.49 €
Fuente de alimentación	1 ud.	1.85 €	1.85 €
Relé	1 ud.	1.28 €	1.28 €
PCB	1 ud.	(20 PCBs) 27.49 €	2.75 €
Conector	2 uds.	(10 piezas) 1.19 €	0.24 €
Pines para soldar	31 uds.	(40 piezas) 1.67 €	1.30 €
Horas ingeniero	90 h	15.38 €	1,384.20 €
		TOTAL	1,402.18 €

Tabla A.3. Presupuesto del módulo del enchufe inteligente

Climatización

Componente	Unidades	Precio unitario	Coste total
Wemos D1 mini	1 ud.	3.88 €	3.88 €
PZEM-004T	1 ud.	5.19 €	5.19 €
Anillo toroidal	1 ud.	1.49 €	1.49 €
Fuente de alimentación	1 ud.	1.85 €	1.85 €
Sensor de temperatura SHT30	1 ud.	2.65 €	2.65 €
Relé	2 ud.	1.28 €	2.56 €
PCB	1 ud.	(20 PCBs) 27.49 €	2.75 €
Conector	2 uds.	(10 piezas) 1.19 €	0.24 €
Pines para soldar	34 uds.	(40 piezas) 1.67 €	1.42 €
Horas ingeniero	70 h	15.38 €	1,076.60 €
		TOTAL	1,098.63 €

Tabla A.4. Presupuesto del módulo de climatización

Programación

- Código Arduino IDE – Enchufe

```
1. #include <ESP8266WiFi.h>
2. #include <FirebaseArduino.h>
3. #include <ArduinoJson.h>
4. #include <ESP8266HTTPClient.h>
5. #include <PZEM004Tv30.h>
6. #include <WEMOS_SHT3X.h>
7. #include <Time.h>
8. #include <TimeSib.h>
9. #include "dimmable_light.h"
10. #include <WiFiUdp.h>
11. // #include <SoftwareSerial.h>
12.
13. #define FIREBASE_HOST "control-consumo-
    electrico-default-rtdb.firebaseio.com" //--> URL
    address of your Firebase Realtime Database.
14. #define FIREBASE_AUTH
    "ATu7ew99iTEgZLQIjeA76JgoVjCsrQKADbzbmF0U" //-->
    Your firebase database secret code.
15.
16. #define ON_Board_LED 2
17.
18. //-----SSID
    y CONTRASEÑA del Wi-Fi
19. const char* ssid = "IngElectrica"; //"Redmi K30
    Ultra"; //"MiFibra-E4BB"; //"BIBLIOTECA"; //
20. const char* password = "C@IngEle#U120"; //"pagal
    osdatos"; //"ZiFPczo7"; //"1357986420900"; //
21. //-----
22.
23. PZEM004Tv30 pzem(14,12, 0x01); //D5 (TX), D6
    (RX)
24. //IPAddress ip(192,168,1,1);
25.
26. //Enchufe
27. const int Rele = D7;
28.
29. int nivel, bucle;
30.
31. //NTP Server para el tiempo
32. static const char ntpServerName[] = "us.pool.
    ntp.org";
33. const int timeZone = 1;
```

```
34.   WiFiUDP Udp;
35.   unsigned int localPort = 8888;
36.   time_t getNtpTime();
37.   void sendNTPpacket(IPAddress &address);
38.   const int NTP_PACKET_SIZE = 48;
39.   byte packetBuffer[NTP_PACKET_SIZE];
40.
41.   time_t getNtpTime()
42.   {
43.       IPAddress ntpServerIP;
44.
45.       while (Udp.parsePacket() > 0) ;
46.       WiFi.hostByName(ntpServerName,
         ntpServerIP);
47.       sendNTPpacket(ntpServerIP);
48.       uint32_t beginWait = millis();
49.       while (millis() - beginWait < 1500) {
50.           int size = Udp.parsePacket();
51.           if (size >= NTP_PACKET_SIZE) {
52.               Udp.read(packetBuffer,
                 NTP_PACKET_SIZE);
53.               unsigned long secsSince1900;
54.               secsSince1900 = (unsigned long)packetB
         uffer[40] << 24;
55.               secsSince1900 |= (unsigned long)packetB
         uffer[41] << 16;
56.               secsSince1900 |= (unsigned long)packetB
         uffer[42] << 8;
57.               secsSince1900 |= (unsigned long)packetB
         uffer[43];
58.               return secsSince1900 - 2208988800UL + t
         imeZone * SECS_PER_HOUR;
59.           }
60.       }
61.       return 0;
62.   }
63.
64.   void sendNTPpacket(IPAddress &address)
65.   {
66.       memset(packetBuffer, 0, NTP_PACKET_SIZE);
67.       packetBuffer[0] = 0b11100011;
68.       packetBuffer[1] = 0;
69.       packetBuffer[2] = 6;
70.       packetBuffer[3] = 0xEC;
71.       packetBuffer[12] = 49;
72.       packetBuffer[13] = 0x4E;
73.       packetBuffer[14] = 49;
```

```
74.     packetBuffer[15] = 52;
75.     Udp.beginPacket(address, 123);
76.     Udp.write(packetBuffer, NTP_PACKET_SIZE);
77.     Udp.endPacket();
78. }
79.
80. void setup() {
81.     Serial.begin(115200);
82.     delay(500);
83.
84.     WiFi.begin(ssid, password);
85.     Serial.println("");
86.
87.     pinMode(ON_Board_LED, OUTPUT);
88.     digitalWrite(ON_Board_LED, HIGH); //-->
    Enciende el LED hasta que se conecta
89.
90.     //ESPERA A LA CONEXIÓN
91.     Serial.print("Connecting");
92.     while (WiFi.status() != WL_CONNECTED) {
93.         Serial.print(".");
94.         digitalWrite(ON_Board_LED, LOW);
95.         delay(250);
96.         digitalWrite(ON_Board_LED, HIGH);
97.         delay(250);}
98.
99.     //NTP
100.    Udp.begin(localPort);
101.    setSyncProvider(getNtpTime);
102.
103.
104.    digitalWrite(ON_Board_LED, HIGH); //--> Apaga
    el LED cuando se ha conectado
105.
106.    //Si se ha conectado correctamente, muestra
    la dirección IP
107.    Serial.println("");
108.    Serial.print("Conectado a: ");
109.    Serial.println(ssid);
110.    Serial.print("Dirección IP: ");
111.    Serial.println(WiFi.localIP());
112.    Serial.println();
113.
114.    //--CONFIGURACIÓN DE FIREBASE--
115.    Firebase.begin(FIREBASE_HOST, FIREBASE_AUTH);
116.
117.    //--CONFIGURACIÓN DE PINES--
```

```
118.   pinMode(Rele, OUTPUT);
119.   digitalWrite(Rele, LOW);
120.
121. }
122.
123. void loop() {
124.
125.   //--INICIALIZO LAS VARIABLES--
126.   float IDEnchufe = 1;
127.   float ListaPrioridad;
128.   float PotContratada;
129.   float PotTotal;
130.   float HoraEncendido, MinutoEncendido,
       HoraApagado, MinutoApagado;
131.   float Tiempo, TiempoSet1, TiempoSet2,
       TiempoEncendido = 0, TiempoApagado = 0,
       DiaSiCambio = 0, Dia;
132.   //BAÑO
133.   String CalefactorON,
       bit_controlCalefactor = "false", CalentadorON,
       bit_controlCalentador = "false";
134.   float PotCalefactor, PotCalentador,
       IDCalefactor, IDCalentador, newPotCalefactor,
       newPotCalentador;
135.   //COCINA
136.   // String ReleCalorCocina, ReleONCocina,
       HornoON, InduccionON, LavadoraON, LavavajillasON,
       MicroondasON, SecadoraON, TeleCocinaON,
       AutoCocina, LCON;
137.   // String bit_controlLC = "false",
       bit_controlClimaCocina = "false",
       bit_controlHorno = "false", bit_controlInduccion
       = "false", bit_controlLavadora = "false";
138.   // String bit_controlLavavajillas = "false",
       bit_controlMicroondas = "false",
       bit_controlSecadora = "false",
       bit_controlTeleCocina = "false";
139.   // float PotCalentador, PotClimaCocina,
       TemperaturaCocina, UmbralInfCocina,
       UmbralSupCocina;
140.   // float PotHorno, PotInduccion,
       PotLavadora, PotLavavajillas, PotMicroondas,
       PotSecadora, PotTeleCocina, LDRCocina,
       NivelCocina, PotLC;
141.   //PASILLO
142.   // bool ReleCalorPasillo, ReleONPasillo,
       AutoPasillo, LPON;
```

```
143.    // float PotClimaPasillo,
    TemperaturaPasillo, UmbralInfPasillo,
    UmbralSupPasillo, LDRPasillo, NivelPasillo,
    PotLP;
144.    //SALÓN
145.    String MusicaON, TeleSalonON,
    bit_controlMusica = "false",
    bit_controlTeleSalon = "false";
146.    float PotMusica, PotTeleSalon, IDTeleSalon,
    IDMusica, newPotMusica, newPotTeleSalon;
147.    //HABITACIÓN 1
148.    // bool ReleCalorHab1, ReleONHab1, AutoHab1,
    LH1ON, TeleHab1ON;
149.    // float PotClimaHab1, TemperaturaHab1,
    UmbralInfHab1, UmbralSupHab1, LDRHab1, NivelHab1,
    PotLH1, PotTeleHab1;
150.    //HABITACIÓN 2
151.    // bool ReleCalorHab2, ReleONHab2, AutoHab2,
    LH2ON, PCON;
152.    // float PotClimaHab2, TemperaturaHab2,
    UmbralInfHab2, UmbralSupHab2, LDRHab2, NivelHab2,
    PotLH2, PotPC;
153.
154.
155. //--SE CARGAN LAS VARIABLES BOOLEANAS--
156. CalefactorON = Firebase.getString("Bath/Calef
    actor/CalefactorON");
157. PotCalefactor = Firebase.getFloat("Bath/Calef
    actor/PotCalefactor");
158. bit_controlCalefactor = Firebase.getString("B
    ath/Calefactor/bit_controlCalefactor");
159. bit_controlCalentador = Firebase.getString("B
    ath/Calentador/bit_controlCalentador");
160. CalentadorON = Firebase.getString("Bath/Calen
    tador/CalentadorON");
161. IDCalefactor = ((Firebase.getString("Bath/Cal
    efactor/IDCalefactor")).substring(1)).toFloat();
162. IDCalentador = ((Firebase.getString("Bath/Cal
    entador/IDCalentador")).substring(1)).toFloat();
163. TiempoEncendido = Firebase.getFloat("Bath/Cal
    entador/TiempoEncendido");
164. TiempoApagado = Firebase.getFloat("Bath/Calen
    tador/TiempoApagado");
165. DiaSiCambio = Firebase.getFloat("Bath/Calenta
    dor/DiaSiCambio");
166. PotCalentador = Firebase.getFloat("Bath/Calen
    tador/PotCalentador");
```

```
167.
168. /* ReleCalorCocina =
    Firebase.getString("Cocina/ClimaCocina/ReleCalorCocina");
169.     ReleONCocina =
    Firebase.getString("Cocina/ClimaCocina/ReleONCocina");
170.     HornoON =
    Firebase.getString("Cocina/Horno/HornoON");
171.     InduccionON =
    Firebase.getString("Cocina/Induccion/InduccionON");
172.     LavadoraON =
    Firebase.getString("Cocina/Lavadora/LavadoraON");
173.     MicroondasON =
    Firebase.getString("Cocina/Microondas/MicroondasON");
174.     SecadoraON =
    Firebase.getString("Cocina/Secadora/SecadoraON");
175.     TeleCocinaON =
    Firebase.getString("Cocina/TeleCocina/TeleCocinaON");
176.     AutoCocina =
    Firebase.getString("Cocina/LuzCocina/AutoCocina");
177.     LCON =
    Firebase.getString("Cocina/LuzCocina/LCON");
178.
179.     ReleCalorPasillo =
    Firebase.getBool("Pasillo/ClimaPasillo/ReleCalorPasillo");
180.     ReleONPasillo =
    Firebase.getBool("Pasillo/ClimaPasillo/ReleONPasillo");
181.     AutoPasillo =
    Firebase.getBool("Pasillo/LuzPasillo/AutoPasillo");
182.     LPON =
    Firebase.getBool("Pasillo/LuzPasillo/LPON");*/
183.
184.     MusicaON = Firebase.getString("Salon/Musica/MusicaON");
185.     PotMusica = Firebase.getFloat("Salon/Musica/PotMusica");
186.     TeleSalonON = Firebase.getString("Salon/TeleSalonON");
```

```
187. PotTeleSalon = Firebase.getFloat("Salon/TeleSalon/PotTeleSalon");
188. bit_controlMusica = Firebase.getString("Salon/Musica/bit_controlMusica");
189. bit_controlTeleSalon = Firebase.getString("Salon/TeleSalon/bit_controlTeleSalon");
190. IDMusica = ((Firebase.getString("Salon/Musica/IDMusica")).substring(1)).toFloat();
191. IDTeleSalon = ((Firebase.getString("Salon/TeleSalon/IDTeleSalon")).substring(1)).toFloat();
192.
193. /* ReleCalorHab1 =
    Firebase.getBool("Hab1/ClimaHab1/ReleCalorHab1");
194. ReleONHab1 =
    Firebase.getBool("Hab1/ClimaHab1/ReleONHab1");
195. AutoHab1 =
    Firebase.getBool("Hab1/LuzHab1/AutoHab1");
196. LH1ON =
    Firebase.getBool("Hab1/LuzHab1/LH1ON");
197. TeleHab1ON =
    Firebase.getBool("Hab1/TeleHab1/TeleHab1ON");
198.
199. ReleCalorHab2 =
    Firebase.getBool("Hab2/ClimaHab2/ReleCalorHab2");
200. ReleONHab2 =
    Firebase.getBool("Hab2/ClimaHab2/ReleONHab2");
201. AutoHab2 =
    Firebase.getBool("Hab2/LuzHab2/AutoHab2");
202. LH2ON =
    Firebase.getBool("Hab2/LuzHab2/LH2ON");
203. PCON = Firebase.getBool("Hab2/PC/PCON");*/
204.
205. PotContratada = ((Firebase.getString("PotContratada")).substring(1)).toFloat();
206. PotTotal = Firebase.getFloat("PotTotal");
207. ListaPrioridad = ((Firebase.getString("ListaPrioridad")).substring(1)).toFloat();
208. HoraApagado = (Firebase.getString("HoraApagado")).toFloat();
209. MinutoApagado = (Firebase.getString("MinutoApagado")).toFloat();
210. HoraEncendido = (Firebase.getString("HoraEncendido")).toFloat();
211. MinutoEncendido = (Firebase.getString("MinutoEncendido")).toFloat();
212.
213. int sexta_posicion = ListaPrioridad % 10;
```

```
214.   int quinta_posicion = sexta_posicion % 10;
215.   int cuarta_posicion = quinta_posicion % 10;
216.   int tercera_posicion = cuarta_posicion % 10;
217.   int segunda_posicion = tercera_posicion % 10;
218.   int primera_posicion = segunda_posicion % 10;
219.
220.   if ((PotContratada*0.60) > PotTotal){
221.       //--BAÑO--
222.       if (IDEnchufe == IDCalefactor){
223.           if (CalefactorON == "true") {
224.               digitalWrite(Rele, HIGH);
225.               if (bit_controlCalefactor == "false") {
226.                   bit_controlCalefactor = "true";
227.                   Firebase.setString("Bath/Calefactor/b
it_controlCalefactor", bit_controlCalefactor);
228.                   delay(3000);
229.                   PotCalefactor = pzem.power();
230.                   Firebase.setFloat("Bath/Calefactor/Po
tCalefactor", PotCalefactor);
231.                   PotTotal = PotTotal + PotCalefactor;
232.                   Firebase.setFloat("PotTotal",
PotTotal);
233.               }
234.               newPotCalefactor = pzem.power();
235.               if (abs(newPotCalefactor - PotCalefacto
r) > 1) {
236.                   PotTotal = PotTotal - PotCalefactor;
237.                   PotCalefactor = newPotCalefactor;
238.                   Firebase.setFloat("Bath/Calefactor/Po
tCalefactor", PotCalefactor);
239.                   PotTotal = PotTotal + PotCalefactor;
240.                   Firebase.setFloat("PotTotal",
PotTotal);
241.               }
242.           } else {
243.               digitalWrite(Rele, LOW);
244.               PotTotal = PotTotal - PotCalefactor;
245.               Firebase.setFloat("PotTotal",
PotTotal);
246.               PotCalefactor = 0;
247.               Firebase.setFloat("Bath/Calefactor/PotC
alefactor", PotCalefactor);
248.               bit_controlCalefactor = "false";
249.               Firebase.setString("Bath/Calefactor/bit
_controlCalefactor",
bit_controlCalefactor);
250.           }
```

```
251.     }
252.
253.     if (IDEnchufe == IDCalentador){
254.         if (CalentadorON == "true") {
255.             digitalWrite(Rele, HIGH);
256.             TiempoApagado = 0;
257.             Firebase.setFloat("Bath/Calentador/Tiem
poApagado", TiempoApagado);
258.             if (bit_controlCalentador == "false") {
259.                 TiempoEncendido = hour()*100 + minute
();
260.                 Firebase.setFloat("Bath/Calentador/Ti
empoEncendido", TiempoEncendido);
261.                 bit_controlCalentador = "true";
262.                 Firebase.setString("Bath/Calentador/b
it_controlCalentador", bit_controlCalentador);
263.                 bucle = 2;
264.                 delay(3000);
265.                 PotCalentador = pzem.power();
266.                 Firebase.setFloat("Bath/Calentador/Po
tCalentador", PotCalentador);
267.                 PotTotal = PotTotal + PotCalentador;
268.                 Firebase.setFloat("PotTotal",
PotTotal);
269.             }
270.             newPotCalentador = pzem.power();
271.             if (abs(newPotCalentador - PotCalentado
r) > 1) {
272.                 PotTotal = PotTotal - PotCalentador;
273.                 PotCalentador = newPotCalentador;
274.                 Firebase.setFloat("Bath/Calentador/Po
tCalentador", PotCalentador);
275.                 PotTotal = PotTotal + PotCalentador;
276.                 Firebase.setFloat("PotTotal",
PotTotal);
277.             }
278.         } else {
279.             digitalWrite(Rele, LOW);
280.             TiempoEncendido = 0;
281.             Firebase.setFloat("Bath/Calentador/Tiem
poEncendido", TiempoEncendido);
282.             if (bit_controlCalentador == "true") {
283.                 TiempoApagado = hour()*100 + minute()
;
284.                 Firebase.setFloat("Bath/Calentador/Ti
empoApagado", TiempoApagado);
```

```
285.         bucle = 1;
286.     }
287.     PotTotal = PotTotal - PotCalentador;
288.     Firebase.setFloat("PotTotal",
    PotTotal);
289.     PotCalentador = 0;
290.     Firebase.setFloat("Bath/Calentador/PotC
    alentador", PotCalentador);
291.     bit_controlCalentador = "false";
292.     Firebase.setString("Bath/Calentador/bit
    _controlCalentador",
    bit_controlCalentador);
293.     }
294.
295.     Tiempo = hour()*100 + minute();
296.     Dia = day();
297.     if (Dia == DiaSiCambio){
298.         if (((Tiempo - TiempoEncendido) >= 100)
    && (bucle == 2)){ //1 hora encendido
299.             TiempoEncendido = 0;
300.             Firebase.setFloat("Bath/Calentador/Ti
    empoEncendido", TiempoEncendido);
301.             TiempoApagado = hour()*100 + minute()
    ;
302.             Firebase.setFloat("Bath/Calentador/Ti
    empoApagado", TiempoApagado);
303.             bucle = 1;
304.             CalentadorON = "false";
305.             Firebase.setString("Bath/Calentador/C
    alentadorON", CalentadorON);
306.             digitalWrite(Rele, LOW);
307.             PotTotal = PotTotal - PotCalentador;
308.             Firebase.setFloat("PotTotal",
    PotTotal);
309.             PotCalentador = 0;
310.             Firebase.setFloat("Bath/Calentador/Po
    tCalentador", PotCalentador);
311.             bit_controlCalentador = "false";
312.             Firebase.setString("Bath/Calentador/b
    it_controlCalentador", bit_controlCalentador);
313.             DiaSiCambio = Dia;
314.             Firebase.setFloat("Bath/Calentador/Di
    aSiCambio", DiaSiCambio);
315.         }
316.         if (((Tiempo - TiempoApagado) >= 600) &
    & (bucle == 1)){ //6 horas apagado
317.             TiempoApagado = 0;
```

```
318.         Firebase.setFloat("Bath/Calentador/TiempoApagado", TiempoApagado);
319.         TiempoEncendido = hour()*100 + minute();
320.         Firebase.setFloat("Bath/Calentador/TiempoEncendido", TiempoEncendido);
321.         bucle = 2;
322.         digitalWrite(Rele, HIGH);
323.         CalentadorON = "true";
324.         Firebase.setString("Bath/Calentador/CalentadorON", CalentadorON);
325.         DiaSiCambio = Dia;
326.         Firebase.setFloat("Bath/Calentador/DiaSiCambio", DiaSiCambio);
327.         if (bit_controlCalentador == "false")
328.         {
329.             bit_controlCalentador = "true";
330.             Firebase.setString("Bath/Calentador/bit_controlCalentador", bit_controlCalentador);
331.             delay(3000);
332.             PotCalentador = pzem.power();
333.             Firebase.setFloat("Bath/Calentador/PotCalentador", PotCalentador);
334.             PotTotal = PotTotal + PotCalentador;
335.             Firebase.setFloat("PotTotal", PotTotal);
336.             newPotCalentador = pzem.power();
337.             if (abs(newPotCalentador - PotCalentador) > 1) {
338.                 PotTotal = PotTotal - PotCalentador;
339.                 PotCalentador = newPotCalentador;
340.                 Firebase.setFloat("Bath/Calentador/PotCalentador", PotCalentador);
341.                 PotTotal = PotTotal + PotCalentador;
342.                 Firebase.setFloat("PotTotal", PotTotal);
343.             }
344.         }
345.     } else {
346.         if (((Tiempo + 2400) - TiempoEncendido) >= 100) && (bucle == 2){ //1 hora encendido
347.             TiempoEncendido = 0;
```

```
348.         Firebase.setFloat("Bath/Calentador/TiempoEncendido", TiempoEncendido);
349.         TiempoApagado = hour()*100 + minute();
350.         Firebase.setFloat("Bath/Calentador/TiempoApagado", TiempoApagado);
351.         CalentadorON = "false";
352.         Firebase.setString("Bath/Calentador/CalentadorON", CalentadorON);
353.         bucle = 1;
354.         digitalWrite(Rele, LOW);
355.         PotTotal = PotTotal - PotCalentador;
356.         Firebase.setFloat("PotTotal", PotTotal);
357.         PotCalentador = 0;
358.         Firebase.setFloat("Bath/Calentador/PotCalentador", PotCalentador);
359.         bit_controlCalentador = "false";
360.         Firebase.setString("Bath/Calentador/bit_controlCalentador", bit_controlCalentador);
361.         DiaSiCambio = Dia;
362.         Firebase.setFloat("Bath/Calentador/DiaSiCambio", DiaSiCambio);
363.     }
364.     if (((Tiempo + 2400) - TiempoApagado) >= 600) && (bucle == 1){ //6 horas apagado
365.         TiempoApagado = 0;
366.         Firebase.setFloat("Bath/Calentador/TiempoApagado", TiempoApagado);
367.         TiempoEncendido = hour()*100 + minute();
368.         Firebase.setFloat("Bath/Calentador/TiempoEncendido", TiempoEncendido);
369.         bucle = 2;
370.         digitalWrite(Rele, HIGH);
371.         CalentadorON = "true";
372.         Firebase.setString("Bath/Calentador/CalentadorON", CalentadorON);
373.         DiaSiCambio = Dia;
374.         Firebase.setFloat("Bath/Calentador/DiaSiCambio", DiaSiCambio);
375.         if (bit_controlCalentador == "false")
376.         {
377.             bit_controlCalentador = "true";
378.             Firebase.setString("Bath/Calentador/bit_controlCalentador", bit_controlCalentador);
379.             delay(3000);
```

```
379.         PotCalentador = pzem.power();
380.         Firebase.setFloat("Bath/Calentador/
PotCalentador", PotCalentador);
381.         PotTotal = PotTotal + PotCalentador
;
382.         Firebase.setFloat("PotTotal",
PotTotal);
383.     }
384.     newPotCalentador = pzem.power();
385.     if (abs(newPotCalentador - PotCalenta
dor) > 1) {
386.         PotTotal = PotTotal - PotCalentador
;
387.         PotCalentador = newPotCalentador;
388.         Firebase.setFloat("Bath/Calentador/
PotCalentador", PotCalentador);
389.         PotTotal = PotTotal + PotCalentador
;
390.         Firebase.setFloat("PotTotal",
PotTotal);
391.     }
392. }
393. }
394. }
395.
396. //-----
397.
398. //--SALÓN--
399.     if (IDEnchufe == IDMusica){
400.         if (MusicaON == "true"){
401.             digitalWrite(Rele, HIGH);
402.             if (bit_controlMusica == "false") {
403.                 bit_controlMusica = "true";
404.                 Firebase.setString("Salon/Musica/bit_
controlMusica", bit_controlMusica);
405.                 delay(3000);
406.                 PotMusica = pzem.power();
407.                 Firebase.setFloat("Salon/Musica/PotMu
sica", PotMusica);
408.                 PotTotal = PotTotal + PotMusica;
409.                 Firebase.setFloat("PotTotal",
PotTotal);
410.             }
411.             newPotMusica = pzem.power();
412.             if (abs(newPotMusica - PotMusica) > 1)
{
413.                 PotTotal = PotTotal - PotMusica;
```

```
414.         PotMusica = newPotMusica;
415.         Firebase.setFloat("Salon/Musica/PotMusica", PotMusica);
416.         PotTotal = PotTotal + PotMusica;
417.         Firebase.setFloat("PotTotal",
    PotTotal);
418.     }
419. } else {
420.     digitalWrite(Rele, LOW);
421.     PotTotal = PotTotal - PotMusica;
422.     Firebase.setFloat("PotTotal",
    PotTotal);
423.     PotMusica = 0;
424.     Firebase.setFloat("Salon/Musica/PotMusica", PotMusica);
425.     bit_controlMusica = "false";
426.     Firebase.setString("Salon/Musica/bit_controlMusica", bit_controlMusica);
427. }
428. }
429.
430. if (IDEnchufe == IDTeleSalon){
431.     if (TeleSalonON == "true"){
432.         digitalWrite(Rele, HIGH);
433.         if (bit_controlTeleSalon == "false") {
434.             bit_controlTeleSalon = "true";
435.             Firebase.setString("Salon/TeleSalon/bit_controlTeleSalon", bit_controlTeleSalon);
436.             delay(3000);
437.             PotTeleSalon = pzem.power();
438.             Firebase.setFloat("Salon/TeleSalon/PotTeleSalon", PotTeleSalon);
439.             PotTotal = PotTotal + PotTeleSalon;
440.             Firebase.setFloat("PotTotal",
    PotTotal);
441.         }
442.         newPotTeleSalon = pzem.power();
443.         if (abs(newPotTeleSalon - PotTeleSalon)
    > 1) {
444.             PotTotal = PotTotal - PotTeleSalon;
445.             PotTeleSalon = newPotTeleSalon;
446.             Firebase.setFloat("Salon/TeleSalon/PotTeleSalon", PotTeleSalon);
447.             PotTotal = PotTotal + PotTeleSalon;
448.             Firebase.setFloat("PotTotal",
    PotTotal);
449.         }
```

```
450.         } else {
451.             digitalWrite(Rele, LOW);
452.             PotTotal = PotTotal - PotTeleSalon;
453.             Firebase.setFloat("PotTotal",
PotTotal);
454.             PotTeleSalon = 0;
455.             Firebase.setFloat("Salon/TeleSalon/PotT
eleSalon", PotTeleSalon);
456.             bit_controlTeleSalon = "false";
457.             Firebase.setString("Salon/TeleSalon/bit
_controlTeleSalon", bit_controlTeleSalon);
458.         }
459.     }
460.     //-----
461. }
462. //
463. // CONTROLANDO POTENCIA SUPERIOR AL 60%
464. //
465.     else if ((PotContratada*0.60) <= PotTotal) {
466.         for (int i = 6; i >= 1; i--){ //No voy a
hacer la programación de las demás habitaciones,
solo de las dos que tengo en el prototipo
467.             if (i == 6){
468.                 if (6posicion == 2){
469.                     if (IDEnchufe == IDCalefactor){
470.                         if (CalefactorON == "true") {
471.                             digitalWrite(Rele, LOW);
472.                             CalefactorON = "false";
473.                             Firebase.setString("Bath/Calefact
or/CalefactorON", CalefactorON);
474.                             PotTotal = PotTotal - PotCalefact
or;
475.                             Firebase.setFloat("PotTotal",
PotTotal);
476.                             PotCalefactor = 0;
477.                             Firebase.setFloat("Bath/Calefacto
r/PotCalefactor", PotCalefactor);
478.                             bit_controlCalefactor = "false";
479.                             Firebase.setString("Bath/Calefact
or/bit_controlCalefactor",
bit_controlCalefactor);
480.                         }
481.                     }
482.                     if (IDEnchufe == IDCalentador){
483.                         if (CalentadorON == "true") {
484.                             digitalWrite(Rele, LOW);
485.                             CalentadorON = "false";
```

```
486.         Firebase.setString("Bath/Calentador/CalentadorON", CalentadorON);
487.         PotTotal = PotTotal - PotCalentador;
488.         Firebase.setFloat("PotTotal", PotTotal);
489.         PotCalentador = 0;
490.         Firebase.setFloat("Bath/Calentador/PotCalentador", PotCalentador);
491.         bit_controlCalentador = "false";
492.         Firebase.setString("Bath/Calentador/bit_controlCalentador", bit_controlCalentador);
493.     }
494. }
495. }
496. if (6posicion == 1){
497.     if (IDEnchufe == IDMusica){
498.         if (MusicaON == "true") {
499.             digitalWrite(Rele, LOW);
500.             MusicaON = "false";
501.             Firebase.setString("Salon/Musica/MusicaON", MusicaON);
502.             PotTotal = PotTotal - PotMusica;
503.             Firebase.setFloat("PotTotal", PotTotal);
504.             PotMusica = 0;
505.             Firebase.setFloat("Salon/Musica/PotMusica", PotMusica);
506.             bit_controlMusica = "false";
507.             Firebase.setString("Salon/Musica/bit_controlMusica", bit_controlMusica);
508.         }
509.     }
510.     if (IDEnchufe == IDTeleSalon){
511.         if (TeleSalonON == "true") {
512.             digitalWrite(Rele, LOW);
513.             TeleSalonON = "false";
514.             Firebase.setString("Salon/TeleSalon/TeleSalonON", TeleSalonON);
515.             PotTotal = PotTotal - PotTeleSalon;
516.             Firebase.setFloat("PotTotal", PotTotal);
517.             PotTeleSalon = 0;
518.             Firebase.setFloat("Salon/TeleSalon/PotTeleSalon", PotTeleSalon);
```

```
519.         bit_controlTeleSalon = "false";
520.         Firebase.setString("Salon/TeleSalon/bit_controlTeleSalon",
521.         bit_controlTeleSalon);
521.     }
522. }
523. }
524.     delay(1000);
525. }
526.     if (i == 5){
527.         if (5posicion == 2){
528.             if (IDEnchufe == IDCalefactor){
529.                 if (CalefactorON == "true") {
530.                     digitalWrite(Rele, LOW);
531.                     CalefactorON = "false";
532.                     Firebase.setString("Bath/CalefactorON", CalefactorON);
533.                     PotTotal = PotTotal - PotCalefactor;
534.                     or;
535.                     Firebase.setFloat("PotTotal", PotTotal);
536.                     PotCalefactor = 0;
537.                     Firebase.setFloat("Bath/Calefactor/PotCalefactor", PotCalefactor);
538.                     bit_controlCalefactor = "false";
539.                     Firebase.setString("Bath/Calefactor/bit_controlCalefactor",
540.                     bit_controlCalefactor);
541.                 }
542.             }
543.             if (IDEnchufe == IDCalentador){
544.                 if (CalentadorON == "true") {
545.                     digitalWrite(Rele, LOW);
546.                     CalentadorON = "false";
547.                     Firebase.setString("Bath/CalentadorON", CalentadorON);
548.                     PotTotal = PotTotal - PotCalentador;
549.                     or;
550.                     Firebase.setFloat("PotTotal", PotTotal);
551.                     PotCalentador = 0;
552.                     Firebase.setFloat("Bath/Calentador/PotCalentador", PotCalentador);
553.                     bit_controlCalentador = "false";
554.                     Firebase.setString("Bath/Calentador/bit_controlCalentador",
555.                     bit_controlCalentador);
```

```
552.         }
553.     }
554. }
555.     if (5posicion == 1){
556.         if (IDEnchufe == IDMusica){
557.             if (MusicaON == "true") {
558.                 digitalWrite(Rele, LOW);
559.                 MusicaON = "false";
560.                 Firebase.setString("Salon/Musica/
MusicaON", MusicaON);
561.                 PotTotal = PotTotal - PotMusica;
562.                 Firebase.setFloat("PotTotal",
PotTotal);
563.                 PotMusica = 0;
564.                 Firebase.setFloat("Salon/Musica/P
otMusica", PotMusica);
565.                 bit_controlMusica = "false";
566.                 Firebase.setString("Salon/Musica/
bit_controlMusica", bit_controlMusica);
567.             }
568.         }
569.         if (IDEnchufe == IDTeleSalon){
570.             if (TeleSalonON == "true") {
571.                 digitalWrite(Rele, LOW);
572.                 TeleSalonON = "false";
573.                 Firebase.setString("Salon/TeleSal
on/TeleSalonON", TeleSalonON);
574.                 PotTotal = PotTotal - PotTeleSalo
n;
575.                 Firebase.setFloat("PotTotal",
PotTotal);
576.                 PotTeleSalon = 0;
577.                 Firebase.setFloat("Salon/TeleSalo
n/PotTeleSalon", PotTeleSalon);
578.                 bit_controlTeleSalon = "false";
579.                 Firebase.setString("Salon/TeleSal
on/bit_controlTeleSalon",
bit_controlTeleSalon);
580.             }
581.         }
582.     }
583.     delay(1000);
584. }
585.     if (i == 4){
586.         if (4posicion == 2){
587.             if (IDEnchufe == IDCalefactor){
588.                 if (CalefactorON == "true") {
```

```
589.         digitalWrite(Rele, LOW);
590.         CalefactorON = "false";
591.         Firebase.setString("Bath/Calefact
or/CalefactorON", CalefactorON);
592.         PotTotal = PotTotal - PotCalefact
or;
593.         Firebase.setFloat("PotTotal",
PotTotal);
594.         PotCalefactor = 0;
595.         Firebase.setFloat("Bath/Calefacto
r/PotCalefactor", PotCalefactor);
596.         bit_controlCalefactor = "false";
597.         Firebase.setString("Bath/Calefact
or/bit_controlCalefactor",
bit_controlCalefactor);
598.     }
599. }
600.     if (IDEnchufe == IDCalentador){
601.         if (CalentadorON == "true") {
602.             digitalWrite(Rele, LOW);
603.             CalentadorON = "false";
604.             Firebase.setString("Bath/Calentad
or/CalentadorON", CalentadorON);
605.             PotTotal = PotTotal - PotCalentad
or;
606.             Firebase.setFloat("PotTotal",
PotTotal);
607.             PotCalentador = 0;
608.             Firebase.setFloat("Bath/Calentado
r/PotCalentador", PotCalentador);
609.             bit_controlCalentador = "false";
610.             Firebase.setString("Bath/Calentad
or/bit_controlCalentador",
bit_controlCalentador);
611.         }
612.     }
613. }
614.     if (4posicion == 1){
615.         if (IDEnchufe == IDMusica){
616.             if (MusicaON == "true") {
617.                 digitalWrite(Rele, LOW);
618.                 MusicaON = "false";
619.                 Firebase.setString("Salon/Musica/
MusicaON", MusicaON);
620.                 PotTotal = PotTotal - PotMusica;
621.                 Firebase.setFloat("PotTotal",
PotTotal);
```

```
622.         PotMusica = 0;
623.         Firebase.setFloat("Salon/Musica/PotMusica", PotMusica);
624.         bit_controlMusica = "false";
625.         Firebase.setString("Salon/Musica/bit_controlMusica", bit_controlMusica);
626.     }
627. }
628. if (IDEnchufe == IDTeleSalon){
629.     if (TeleSalonON == "true") {
630.         digitalWrite(Rele, LOW);
631.         TeleSalonON = "false";
632.         Firebase.setString("Salon/TeleSalon/TeleSalonON", TeleSalonON);
633.         PotTotal = PotTotal - PotTeleSalon;
634.         Firebase.setFloat("PotTotal", PotTotal);
635.         PotTeleSalon = 0;
636.         Firebase.setFloat("Salon/TeleSalon/PotTeleSalon", PotTeleSalon);
637.         bit_controlTeleSalon = "false";
638.         Firebase.setString("Salon/TeleSalon/bit_controlTeleSalon", bit_controlTeleSalon);
639.     }
640. }
641. }
642. delay(1000);
643. }
644. if (i == 3){
645.     if (3posicion == 2){
646.         if (IDEnchufe == IDCalefactor){
647.             if (CalefactorON == "true") {
648.                 digitalWrite(Rele, LOW);
649.                 CalefactorON = "false";
650.                 Firebase.setString("Bath/Calefactor/CalefactorON", CalefactorON);
651.                 PotTotal = PotTotal - PotCalefactor;
652.                 Firebase.setFloat("PotTotal", PotTotal);
653.                 PotCalefactor = 0;
654.                 Firebase.setFloat("Bath/Calefactor/PotCalefactor", PotCalefactor);
655.                 bit_controlCalefactor = "false";
```

```
656.         Firebase.setString("Bath/Calefact
or/bit_controlCalefactor",
bit_controlCalefactor);
657.     }
658. }
659.     if (IDEnchufe == IDCalentador){
660.         if (CalentadorON == "true") {
661.             digitalWrite(Rele, LOW);
662.             CalentadorON = "false";
663.             Firebase.setString("Bath/Calentad
or/CalentadorON", CalentadorON);
664.             PotTotal = PotTotal - PotCalentad
or;
665.             Firebase.setFloat("PotTotal",
PotTotal);
666.             PotCalentador = 0;
667.             Firebase.setFloat("Bath/Calentado
r/PotCalentador", PotCalentador);
668.             bit_controlCalentador = "false";
669.             Firebase.setString("Bath/Calentad
or/bit_controlCalentador",
bit_controlCalentador);
670.         }
671.     }
672. }
673.     if (3posicion == 1){
674.         if (IDEnchufe == IDMusica){
675.             if (MusicaON == "true") {
676.                 digitalWrite(Rele, LOW);
677.                 MusicaON = "false";
678.                 Firebase.setString("Salon/Musica/
MusicaON", MusicaON);
679.                 PotTotal = PotTotal - PotMusica;
680.                 Firebase.setFloat("PotTotal",
PotTotal);
681.                 PotMusica = 0;
682.                 Firebase.setFloat("Salon/Musica/P
otMusica", PotMusica);
683.                 bit_controlMusica = "false";
684.                 Firebase.setString("Salon/Musica/
bit_controlMusica", bit_controlMusica);
685.             }
686.         }
687.         if (IDEnchufe == IDTeleSalon){
688.             if (TeleSalonON == "true") {
689.                 digitalWrite(Rele, LOW);
690.                 TeleSalonON = "false";
```

```
691.         Firebase.setString("Salon/TeleSalon/TeleSalonON", TeleSalonON);
692.         PotTotal = PotTotal - PotTeleSalon;
693.         Firebase.setFloat("PotTotal", PotTotal);
694.         PotTeleSalon = 0;
695.         Firebase.setFloat("Salon/TeleSalon/PotTeleSalon", PotTeleSalon);
696.         bit_controlTeleSalon = "false";
697.         Firebase.setString("Salon/TeleSalon/bit_controlTeleSalon", bit_controlTeleSalon);
698.     }
699. }
700. }
701.     delay(1000);
702. }
703.     if (i == 2){
704.         if (2posicion == 2){
705.             if (IDEnchufe == IDCalefactor){
706.                 if (CalefactorON == "true") {
707.                     digitalWrite(Rele, LOW);
708.                     CalefactorON = "false";
709.                     Firebase.setString("Bath/Calefactor/CalefactorON", CalefactorON);
710.                     PotTotal = PotTotal - PotCalefactor;
711.                     Firebase.setFloat("PotTotal", PotTotal);
712.                     PotCalefactor = 0;
713.                     Firebase.setFloat("Bath/Calefactor/PotCalefactor", PotCalefactor);
714.                     bit_controlCalefactor = "false";
715.                     Firebase.setString("Bath/Calefactor/bit_controlCalefactor", bit_controlCalefactor);
716.                 }
717.             }
718.             if (IDEnchufe == IDCalentador){
719.                 if (CalentadorON == "true") {
720.                     digitalWrite(Rele, LOW);
721.                     CalentadorON = "false";
722.                     Firebase.setString("Bath/Calentador/CalentadorON", CalentadorON);
723.                     PotTotal = PotTotal - PotCalentador;
724.                     or;
```

```
724.         Firebase.setFloat("PotTotal",
    PotTotal);
725.         PotCalentador = 0;
726.         Firebase.setFloat("Bath/Calentado
    r/PotCalentador", PotCalentador);
727.         bit_controlCalentador = "false";
728.         Firebase.setString("Bath/Calentad
    or/bit_controlCalentador",
    bit_controlCalentador);
729.     }
730. }
731. }
732.     if (2posicion == 1){
733.         if (IDEnchufe == IDMusica){
734.             if (MusicaON == "true") {
735.                 digitalWrite(Rele, LOW);
736.                 MusicaON = "false";
737.                 Firebase.setString("Salon/Musica/
    MusicaON", MusicaON);
738.                 PotTotal = PotTotal - PotMusica;
739.                 Firebase.setFloat("PotTotal",
    PotTotal);
740.                 PotMusica = 0;
741.                 Firebase.setFloat("Salon/Musica/P
    otMusica", PotMusica);
742.                 bit_controlMusica = "false";
743.                 Firebase.setString("Salon/Musica/
    bit_controlMusica", bit_controlMusica);
744.             }
745.         }
746.         if (IDEnchufe == IDTeleSalon){
747.             if (TeleSalonON == "true") {
748.                 digitalWrite(Rele, LOW);
749.                 TeleSalonON = "false";
750.                 Firebase.setString("Salon/TeleSal
    on/TeleSalonON", TeleSalonON);
751.                 PotTotal = PotTotal - PotTeleSalo
    n;
752.                 Firebase.setFloat("PotTotal",
    PotTotal);
753.                 PotTeleSalon = 0;
754.                 Firebase.setFloat("Salon/TeleSalo
    n/PotTeleSalon", PotTeleSalon);
755.                 bit_controlTeleSalon = "false";
756.                 Firebase.setString("Salon/TeleSal
    on/bit_controlTeleSalon",
    bit_controlTeleSalon);
```

```
757.         }
758.     }
759. }
760.     delay(1000);
761. }
762.     if ((PotContratada*0.60) > PotTotal) {
763.         break;
764.     }
765. }
766.
767. /*
768.
769.
770.     float v = pzem.voltage();
771.     //if (v < 0.0)
772.         v = 0.0;
773.     Serial.print(v);
774.     Serial.print("V; ");
775.     */
776.
777.     Serial.print(hour());
778.     Serial.print(":");
779.     Serial.print(minute());
780.     Serial.print(":");
781.     Serial.println(second());
782.     int hora = hour();
783.     int minuto = minute();
784.     int segundo = second();
785.     Firebase.setInt("Hora",hora);
786.     Firebase.setInt("Minutos",minuto);
787.
788. //-----
789. /*
790.
791.     if (flag=="ON") {
792.         digitalWrite(ON_Board_LED, LOW);
793.     } else if (flag=="OFF") {
794.         digitalWrite(ON_Board_LED, HIGH);
795.     }
796.     Serial.println(flag);
797.
798.     float i = pzem.current();
799.     if (i >= 0.0){
800.         Serial.print(i);
801.         Serial.print("A; ");
802.     }
803.
```

```
804.     float p = (pzem.voltage() * pzem.current());
805.     if (p >= 0.0){
806.         Serial.print(p);
807.         Serial.print("W; ");
808.     }
809.
810.     float e = pzem.energy();
811.     if (e >= 0.0){
812.         Serial.print(e);
813.         Serial.print("Wh; ");
814.     }
815.
816.     Serial.println();
817.     Firebase.setFloat("Voltaje",v);
818.     Firebase.setFloat("Intensidad",i);
819.     Firebase.setFloat("Potencia",p);
820.     Firebase.setFloat("Energia",e);
821.
822.     */
823.
824.     // Conditions for handling errors.
825.     if (Firebase.failed()) {
826.         Serial.print("Setting /Value failed :");
827.         Serial.println(Firebase.error());
828.         delay(500);
829.         return;
830.     }
831.
832.     Serial.println("Setting successful");
833.     Serial.println();
834.     //-----*/
835.
836.     Serial.print("Bucle: ");
837.     Serial.println(bucle);
838.     Serial.print("nivel_dimmer: ");
839.     Serial.println(nivel_dimmer);
840.     Serial.print("Nivel PZEM: ");
841.     Serial.println((pzem.voltage() * pzem.current
        ())) );
842.     Serial.println (pzem.power());
843.     Serial.println(pzem.current());
844.     Serial.println(pzem.voltage());
845.     Serial.println(pzem.energy());
846.     Serial.print("ReleONSalon: ");
847.     Serial.println(ReleONSalon);
848.
849.
```

```

850.
851. // Conditions for handling errors.
852. if (Firebase.failed()) {
853.     Serial.print("Getting /Value failed :");
854.     Serial.println(Firebase.error());
855.     delay(500);
856.     return;
857. }
858. }
859.

```

- **Código Arduino IDE – Iluminación**

```

1. #include <ESP8266WiFi.h>
2. #include <FirebaseArduino.h>
3. #include <ArduinoJson.h>
4. #include <ESP8266HTTPClient.h>
5. #include <PZEM004Tv30.h>
6. #include <WEMOS_SHT3X.h>
7. #include <Time.h>
8. #include <TimeSib.h>
9. #include <RBDdimmer.h>
10. #include "dimmable_light.h"
11. #include <WiFiUdp.h>
12. #include <BH1750.h>
13. #include <Wire.h>
14. // #include <SoftwareSerial.h>
15.
16. #define FIREBASE_HOST "control-consumo-
    electrico-default-rtdb.firebaseio.com" //--> URL
    address of your Firebase Realtime Database.
17. #define FIREBASE_AUTH
    "ATu7ew99iTEgZLQIjeA76JgoVjCsrQKADbzbmF0U" //-->
    Your firebase database secret code.
18.
19. #define ON_Board_LED 2
20. #define outputPin 4 //D2
21. #define zerocross 5 //D1
22.
23. //-----SSID
    y CONTRASEÑA del Wi-Fi
24. const char* ssid = "IngElectrica"; //"Redmi K30
    Ultra"; //"MiFibra-E4BB"; //"BIBLIOTECA"; //
25. const char* password = "C@IngEle#U120"; //"pagal
    osdatos"; //"ZiFPczo7"; //"1357986420900"; //
26. //-----
27.

```

```
28.  PZEM004Tv30 pzem(14,12, 0x01); //D5 (TX), D6
    (RX)
29.  //IPAddress ip(192,168,1,1);
30.
31.  //Iluminación
32.  dimmerLamp dimmer(outputPin, zerocross);
33.  int luz_maxima = 1024; //Luz al máximo
34.  int luz_minima = 400; //No hay luz
35.  int nivel_dimmer = 20; //Este valor va entre
    20% yv 80%
36.
37.  int nivel, bucle;
38.
39.
40.  //NTP Server para el tiempo
41.  static const char ntpServerName[] = "us.pool.
    ntp.org";
42.  const int timeZone = 1;
43.  WiFiUDP Udp;
44.  unsigned int localPort = 8888;
45.  time_t getNtpTime();
46.  void sendNTPpacket(IPAddress &address);
47.  const int NTP_PACKET_SIZE = 48;
48.  byte packetBuffer[NTP_PACKET_SIZE];
49.
50.  time_t getNtpTime()
51.  {
52.      IPAddress ntpServerIP;
53.
54.      while (Udp.parsePacket() > 0) ;
55.      WiFi.hostByName(ntpServerName,
        ntpServerIP);
56.      sendNTPpacket(ntpServerIP);
57.      uint32_t beginWait = millis();
58.      while (millis() - beginWait < 1500) {
59.          int size = Udp.parsePacket();
60.          if (size >= NTP_PACKET_SIZE) {
61.              Udp.read(packetBuffer,
                NTP_PACKET_SIZE);
62.              unsigned long secsSince1900;
63.              secsSince1900 = (unsigned long)packetB
                uffer[40] << 24;
64.              secsSince1900 |= (unsigned long)packetB
                uffer[41] << 16;
65.              secsSince1900 |= (unsigned long)packetB
                uffer[42] << 8;
```

```
66.         secsSince1900 |= (unsigned long)packetB
        uffer[43];
67.         return secsSince1900 - 2208988800UL + t
        imeZone * SECS_PER_HOUR;
68.     }
69. }
70.     return 0;
71. }
72.
73. void sendNTPpacket(IPAddress &address)
74. {
75.     memset(packetBuffer, 0, NTP_PACKET_SIZE);
76.     packetBuffer[0] = 0b11100011;
77.     packetBuffer[1] = 0;
78.     packetBuffer[2] = 6;
79.     packetBuffer[3] = 0xEC;
80.     packetBuffer[12] = 49;
81.     packetBuffer[13] = 0x4E;
82.     packetBuffer[14] = 49;
83.     packetBuffer[15] = 52;
84.     Udp.beginPacket(address, 123);
85.     Udp.write(packetBuffer, NTP_PACKET_SIZE);
86.     Udp.endPacket();
87. }
88.
89. void setup() {
90.     Serial.begin(115200);
91.     delay(1000);
92.
93.     WiFi.begin(ssid, password);
94.     Serial.println("");
95.
96.     pinMode(ON_Board_LED, OUTPUT);
97.     digitalWrite(ON_Board_LED, HIGH); //-->
    Enciende el LED hasta que se conecta
98.
99.     //ESPERA A LA CONEXIÓN
100.    Serial.print("Connecting");
101.    while (WiFi.status() != WL_CONNECTED) {
102.        Serial.print(".");
103.        digitalWrite(ON_Board_LED, LOW);
104.        delay(250);
105.        digitalWrite(ON_Board_LED, HIGH);
106.        delay(250);}
107.
108.    //NTP
109.    Udp.begin(localPort);
```

```
110.   setSyncProvider(getNtpTime);
111.
112.
113.   digitalWrite(ON_Board_LED, HIGH); //--> Apaga
      el LED cuando se ha conectado
114.
115.   //Si se ha conectado correctamente, muestra
      la dirección IP
116.   Serial.println("");
117.   Serial.print("Conectado a: ");
118.   Serial.println(ssid);
119.   Serial.print("Dirección IP: ");
120.   Serial.println(WiFi.localIP());
121.   Serial.println();
122.
123.   //--CONFIGURACIÓN DE FIREBASE--
124.   Firebase.begin(FIREBASE_HOST, FIREBASE_AUTH);
125.
126.   //--CONFIGURACION DE DIMMER--
127.   dimmer.begin(NORMAL_MODE,
      ON); //inicializacion del dimmer:
      name.begin(MODE, STATE)
128.
129. }
130.
131. void loop() {
132.
133.   //--INICIALIZO LAS VARIABLES--
134.   float IDLuz = 1;
135.   int ListaPrioridad;
136.   float PotContratada;
137.   float PotTotal;
138.   float HoraEncendido, MinutoEncendido,
      HoraApagado, MinutoApagado;
139.   float Tiempo, TiempoSet1, TiempoSet2,
      TiempoEncendido = 0, TiempoApagado = 0,
      DiaSiCambio = 0, Dia;
140.   //BAÑO
141.   String AutoBath, LBON,
      bit_controlLB = "false";
142.   float NivelBath, PotLB, IDLB, LDRBath,
      newPotLB;
143.   //COCINA
144.   // String ReleCalorCocina, ReleONCocina,
      HornoON, InduccionON, LavadoraON, LavavajillasON,
      MicroondasON, SecadoraON, TeleCocinaON,
      AutoCocina, LCON;
```

```
145.    // String bit_controlLC = "false",
        bit_controlClimaCocina = "false",
        bit_controlHorno = "false", bit_controlInduccion
        = "false", bit_controlLavadora = "false";
146.    // String bit_controlLavavajillas = "false",
        bit_controlMicroondas = "false",
        bit_controlSecadora = "false",
        bit_controlTeleCocina = "false";
147.    // float PotCalentador, PotClimaCocina,
        TemperaturaCocina, UmbralInfCocina,
        UmbralSupCocina;
148.    // float PotHorno, PotInduccion,
        PotLavadora, PotLavavajillas, PotMicroondas,
        PotSecadora, PotTeleCocina, LDRCocina,
        NivelCocina, PotLC;
149.    //PASILLO
150.    // bool ReleCalorPasillo, ReleONPasillo,
        AutoPasillo, LPON;
151.    // float PotClimaPasillo,
        TemperaturaPasillo, UmbralInfPasillo,
        UmbralSupPasillo, LDRPasillo, NivelPasillo,
        PotLP;
152.    //SALÓN
153.    String AutoSalon, LSON,
        bit_controlLS = "false";
154.    float NivelSalon, PotLS, IDLS, LDRSalon,
        newPotLS;
155.    //HABITACIÓN 1
156.    // bool ReleCalorHab1, ReleONHab1, AutoHab1,
        LH1ON, TeleHab1ON;
157.    // float PotClimaHab1, TemperaturaHab1,
        UmbralInfHab1, UmbralSupHab1, LDRHab1, NivelHab1,
        PotLH1, PotTeleHab1;
158.    //HABITACIÓN 2
159.    // bool ReleCalorHab2, ReleONHab2, AutoHab2,
        LH2ON, PCON;
160.    // float PotClimaHab2, TemperaturaHab2,
        UmbralInfHab2, UmbralSupHab2, LDRHab2, NivelHab2,
        PotLH2, PotPC;
161.
162.
163.    //--SE CARGAN LAS VARIABLES BOOLEANAS--
164.    AutoBath = Firebase.getString("Bath/LuzBath/A
        utoBath");
165.    LBON = Firebase.getString("Bath/LuzBath/LBON"
        );
```

```
166. bit_controlLB = Firebase.getString("Bath/LuzBath/bit_controlLB");
167. IDLB = ((Firebase.getString("Bath/LuzBath/IDLB")).substring(1)).toFloat();
168. DiaSiCambio = Firebase.getFloat("Bath/Calentador/DiaSiCambio");
169. PotLB = Firebase.getFloat("Bath/LuzBath/PotLB");
170.
171. /* ReleCalorCocina =
    Firebase.getString("Cocina/ClimaCocina/ReleCalorCocina");
172. ReleONCocina =
    Firebase.getString("Cocina/ClimaCocina/ReleONCocina");
173. HornoON =
    Firebase.getString("Cocina/Horno/HornoON");
174. InduccionON =
    Firebase.getString("Cocina/Induccion/InduccionON");
175. LavadoraON =
    Firebase.getString("Cocina/Lavadora/LavadoraON");
176. MicroondasON =
    Firebase.getString("Cocina/Microondas/MicroondasON");
177. SecadoraON =
    Firebase.getString("Cocina/Secadora/SecadoraON");
178. TeleCocinaON =
    Firebase.getString("Cocina/TeleCocina/TeleCocinaON");
179. AutoCocina =
    Firebase.getString("Cocina/LuzCocina/AutoCocina");
180. LCON =
    Firebase.getString("Cocina/LuzCocina/LCON");
181.
182. ReleCalorPasillo =
    Firebase.getBool("Pasillo/ClimaPasillo/ReleCalorPasillo");
183. ReleONPasillo =
    Firebase.getBool("Pasillo/ClimaPasillo/ReleONPasillo");
184. AutoPasillo =
    Firebase.getBool("Pasillo/LuzPasillo/AutoPasillo");
185. LPON =
    Firebase.getBool("Pasillo/LuzPasillo/LPON");*/
```

```
186.
187.   AutoSalon = Firebase.getString("Salon/LuzSalon/AutoSalon");
188.   LSON = Firebase.getString("Salon/LuzSalon/LSON");
189.   bit_controlLS = Firebase.getString("Salon/LuzSalon/bit_controlLS");
190.   IDLS = ((Firebase.getString("Salon/LuzSalon/IDLS")).substring(1)).toFloat();
191.   PotLS = Firebase.getFloat("Salon/LuzSalon/PotLS");
192.
193.
194. /*   ReleCalorHab1 =
        Firebase.getBool("Hab1/ClimaHab1/ReleCalorHab1");
195.   ReleONHab1 =
        Firebase.getBool("Hab1/ClimaHab1/ReleONHab1");
196.   AutoHab1 =
        Firebase.getBool("Hab1/LuzHab1/AutoHab1");
197.   LH1ON =
        Firebase.getBool("Hab1/LuzHab1/LH1ON");
198.   TeleHab1ON =
        Firebase.getBool("Hab1/TeleHab1/TeleHab1ON");
199.
200.   ReleCalorHab2 =
        Firebase.getBool("Hab2/ClimaHab2/ReleCalorHab2");
201.   ReleONHab2 =
        Firebase.getBool("Hab2/ClimaHab2/ReleONHab2");
202.   AutoHab2 =
        Firebase.getBool("Hab2/LuzHab2/AutoHab2");
203.   LH2ON =
        Firebase.getBool("Hab2/LuzHab2/LH2ON");
204.   PCON = Firebase.getBool("Hab2/PC/PCON");*/
205.
206.   PotContratada = ((Firebase.getString("PotContratada")).substring(1)).toFloat();
207.   PotTotal = Firebase.getFloat("PotTotal");
208.   ListaPrioridad = ((Firebase.getString("ListaPrioridad")).substring(1)).toInt();
209.   HoraApagado = (Firebase.getString("HoraApagado")).toFloat();
210.   MinutoApagado = (Firebase.getString("MinutoApagado")).toFloat();
211.   HoraEncendido = (Firebase.getString("HoraEncendido")).toFloat();
212.   MinutoEncendido = (Firebase.getString("MinutoEncendido")).toFloat();
```

```
213.
214.   int sexta_posicion = ListaPrioridad % 10;
215.   int quinta_posicion = sexta_posicion % 10;
216.   int cuarta_posicion = quinta_posicion % 10;
217.   int tercera_posicion = cuarta_posicion % 10;
218.   int segunda_posicion = tercera_posicion % 10;
219.   int primera_posicion = segunda_posicion % 10;
220.
221.   if ((PotContratada*0.60) > PotTotal){
222.       //--BAÑO--
223.       if (IDLuz == IDLB){
224.           if (AutoBath == "true"){
225.               LDRBath = Firebase.getFloat("Bath/LuzBath/LDRBath");
226.               nivel = constrain(LDRBath, 100, 1200);
227.               // (LDRSalon, luz_minima, luz_maxima)
228.               Restringimos el valor del nivel de luz
229.               nivel_dimmer = map(nivel, 1200, 100, 20, 80); // Al revés porque, mientras mayor nivel
230.               de LDR, menos luz necesitamos
231.               if (nivel <= 1000) {
232.                   dimmer.setPower(nivel_dimmer); //
233.                   name.setPower(0%-100%) o 20%-80% en nuestro caso
234.                   //NivelBath = nivel_dimmer*100/255;
235.                   NivelBath = map(nivel_dimmer, 20, 80, 0, 100); //Para mostrarlo en la interfaz
236.                   Firebase.setFloat("Bath/LuzBath/NivelBath", NivelBath);
237.                   LBON = "true";
238.                   Firebase.setString("Bath/LuzBath/LBON", LBON);
239.                   //digitalWrite(ReleLuz, HIGH);
240.                   if (bit_controlLB == "false"){
241.                       bit_controlLB = "true";
242.                       Firebase.setString("Bath/LuzBath/bit_controlLB", bit_controlLB);
243.                       delay(3000);
244.                       PotLB = pzem.power();
245.                       Firebase.setFloat("Bath/LuzBath/PotLB", PotLB);
246.                       PotTotal = PotTotal + PotLB;
247.                       Firebase.setFloat("PotTotal", PotTotal);
248.                   }
249.                   newPotLB = pzem.power();
250.                   if (abs(newPotLB - PotLB) > 1) {
251.                       PotTotal = PotTotal - PotLB;
```

```
248.         PotLB = newPotLB;
249.         Firebase.setFloat("Bath/LuzBath//PotLB", PotLB);
250.         PotTotal = PotTotal + PotLB;
251.         Firebase.setFloat("PotTotal",
    PotTotal);
252.     }
253.
254.     Serial.println(pzem.voltage());
255.     Serial.println(pzem.power());
256.     Serial.print("nivel_dimmer: ");
257.     Serial.println(nivel_dimmer);
258.     Serial.print("Nivel Bath: ");
259.     Serial.println(NivelBath);
260.
261.     } else if (nivel > 1000) {
262.         dimmer.setPower(20);
263.         LBON = "false";
264.         Firebase.setString("Bath/LuzBath//LBON", LBON);
265.         //digitalWrite(ReleLuz, LOW);
266.         PotTotal = PotTotal - PotLB;
267.         Firebase.setFloat("PotTotal",
    PotTotal);
268.         PotLB = 0;
269.
270.         Serial.print("PotTotal: ");
271.         Serial.println(PotTotal);
272.         Serial.print("PotLB: ");
273.         Serial.println(PotLB);
274.
275.         Firebase.setFloat("Bath/LuzBath//PotLB", PotLB);
276.         bit_controlLB = "false";
277.         Firebase.setString("Bath/LuzBath//bit_controlLB", bit_controlLB);
278.     }
279.     } else {
280.         if (LBON == "true") {
281.             NivelBath = (Firebase.getString("Bath/LuzBath/NivelBath")).toFloat();
282.             nivel_dimmer = map(NivelBath, 0, 100,
    20, 80);
283.             dimmer.setPower(nivel_dimmer);
284.             //analogWrite(valor_luz,
    nivel_dimmer);
285.             //digitalWrite(ReleLuz, HIGH);
```

```

286.         if (bit_controlLB == "false") { //Par
           a no sumar sin parar la potencia
287.             bit_controlLB = "true";
288.             Firebase.setString("Bath/LuzBath/bit_
               t_controlLB", bit_controlLB);
289.             delay(3000);
290.             PotLB = pzem.power();
291.             Firebase.setFloat("Bath/LuzBath/Pot
               LB", PotLB);
292.             PotTotal = PotTotal + PotLB;
293.             Firebase.setFloat("PotTotal",
               PotTotal);
294.         }
295.         newPotLB = pzem.power();
296.         if (abs(newPotLB - PotLB) > 1) {
297.             PotTotal = PotTotal - PotLB;
298.             PotLB = newPotLB;
299.             Firebase.setFloat("Bath/LuzBath//Po
               tLB", PotLB);
300.             PotTotal = PotTotal + PotLB;
301.             Firebase.setFloat("PotTotal",
               PotTotal);
302.         }
303.     } else {
304.         dimmer.setPower(20);
305.         //digitalWrite(ReleLuz, LOW);
306.         Serial.print("PotLB: ");
307.         Serial.println(PotLB);
308.         PotTotal = PotTotal - PotLB;
309.         Firebase.setFloat("PotTotal",
               PotTotal);
310.         PotLB = 0;
311.         Firebase.setFloat("Bath/LuzBath/PotLB
               ", PotLB);
312.         bit_controlLB = "false";
313.         Firebase.setString("Bath/LuzBath/bit_
               controlLB", bit_controlLB);
314.     }
315. }
316. }
317. //-----
318.
319. //--SALÓN--
320.     if (IDLuz == IDLS){
321.         LDRSalon = Firebase.getFloat("Salon/LuzSa
               lon/LDRSalon");
322.         if (AutoSalon == "true"){

```

```
323.         nivel = constrain(LDRSalon, 100, 1200);  
           // (LDRSalon, luz_minima, luz_maxima)  
           Restringimos el valor del nivel de luz aunque, en  
           nuestro caso, no hace falta  
324.         nivel_dimmer = map(nivel, 1200, 100, 20  
           , 80); // Al revés porque, mientras mayor nivel  
           de LDR, menos luz necesitamos  
325.         if (nivel <= 1000) {  
326.             dimmer.setPower(nivel_dimmer);  
327.             //analogWrite(valor_luz,  
           nivel_dimmer);  
328.             //NivelSalon = nivel_dimmer*100/255;  
329.             NivelSalon = map(nivel_dimmer, 20, 80  
           , 0, 100);  
330.             Firebase.setFloat("Salon/LuzSalon/Niv  
           elSalon", NivelSalon);  
331.             LSON = "true";  
332.             Firebase.setString("Salon/LuzSalon/LS  
           ON", LSON);  
333.             //digitalWrite(ReleLuz, HIGH);  
334.             if (bit_controlLS == "false"){  
335.                 bit_controlLS = "true";  
336.                 Firebase.setString("Salon/LuzSalon/  
           bit_controlLS", bit_controlLS);  
337.                 delay(3000);  
338.                 PotLS = pzem.power();  
339.                 Firebase.setFloat("Salon/LuzSalon/P  
           otLS", PotLS);  
340.                 PotTotal = PotTotal + PotLS;  
341.                 Firebase.setFloat("PotTotal",  
           PotTotal);  
342.             }  
343.             newPotLS = pzem.power();  
344.             if (abs(newPotLS - PotLS) > 1) {  
345.                 PotTotal = PotTotal - PotLS;  
346.                 PotLS = newPotLS;  
347.                 Firebase.setFloat("Salon/LuzSalon//  
           PotLS", PotLS);  
348.                 PotTotal = PotTotal + PotLS;  
349.                 Firebase.setFloat("PotTotal",  
           PotTotal);  
350.             }  
351.         } else if (nivel > 1000) {  
352.             dimmer.setPower(20);  
353.             LSON = "false";  
354.             Firebase.setString("Salon/LuzSalon/LS  
           ON", LSON);
```

```
355.         //digitalWrite(ReleLuz, LOW);
356.         PotTotal = PotTotal - PotLS;
357.         Firebase.setFloat("PotTotal",
    PotTotal);
358.         PotLS = 0;
359.         Firebase.setFloat("Salon/LuzSalon/Pot
    LS", PotLS);
360.         bit_controlLS = "false";
361.         Firebase.setString("Salon/LuzSalon/bi
    t_controlLS", bit_controlLS);
362.     }
363.     } else {
364.         if (LSON == "true"){
365.             NivelSalon = (Firebase.getString("Sal
    on/LuzSalon/NivelSalon")).toFloat();
366.             nivel_dimmer = map(NivelBath, 0, 100,
    20, 80);
367.             dimmer.setPower(nivel_dimmer);
368.             //analogWrite(valor_luz,
    nivel_dimmer);
369.             //digitalWrite(ReleLuz, HIGH);
370.             if (bit_controlLS == "false") {
371.                 bit_controlLS = "true";
372.                 Firebase.setString("Salon/LuzSalon/
    bit_controlLS", bit_controlLS);
373.                 delay(3000);
374.                 PotLS = (pzem.power());
375.                 Firebase.setFloat("Salon/LuzSalon/P
    otLS", PotLS);
376.                 PotTotal = PotTotal + PotLS;
377.                 Firebase.setFloat("PotTotal",
    PotTotal);
378.             }
379.             newPotLS = pzem.power();
380.             if (abs(newPotLS - PotLS) > 1) {
381.                 PotTotal = PotTotal - PotLS;
382.                 PotLS = newPotLS;
383.                 Firebase.setFloat("Salon/LuzSalon//
    PotLS", PotLS);
384.                 PotTotal = PotTotal + PotLS;
385.                 Firebase.setFloat("PotTotal",
    PotTotal);
386.             }
387.         } else {
388.             dimmer.setPower(20);
389.             //digitalWrite(ReleLuz, LOW);
390.             PotTotal = PotTotal - PotLS;
```

```

391.         Firebase.setFloat("PotTotal",
    PotTotal);
392.         PotLS = 0;
393.         Firebase.setFloat("Salon/LuzSalon/Pot
    LS", PotLS);
394.         bit_controlLS = "false";
395.         Firebase.setString("Salon/LuzSalon/bi
    t_controlLS", bit_controlLS);
396.     }
397. }
398. }
399. //-----
400. }
401. //
402. // CONTROLANDO POTENCIA SUPERIOR AL 60%
403. //
404.     else if ((PotContratada*0.60) <= PotTotal) {
405.         for (int i = 6; i >= 1; i--){ //No voy a
            hacer la programación de las demás habitaciones,
            solo de las dos que tengo en el prototipo
406.             if (i == 6){
407.                 if (sexta_posicion == 2){
408.                     if (IDLuz == IDLB){
409.                         if (LBON == "true") {
410.                             dimmer.setPower(20);
411.                             //digitalWrite(ReleLuz, LOW);
412.                             LBON = "false";
413.                             Firebase.setString("Bath/LuzBath/
                                LBON", LBON);
414.                             PotTotal = PotTotal - PotLB;
415.                             Firebase.setFloat("PotTotal",
                                PotTotal);
416.                             PotLB = 0;
417.                             Firebase.setFloat("Bath/LuzBath/P
                                otLB", PotLB);
418.                             bit_controlLB = "false";
419.                             Firebase.setString("Bath/LuzBath/
                                bit_controlLB", bit_controlLB);
420.                         }
421.                     }
422.                 }
423.                 if (sexta_posicion == 1){
424.                     if (IDLuz == IDLS){
425.                         if (LSON == "true") {
426.                             dimmer.setPower(20);
427.                             //digitalWrite(ReleLuz, LOW);
428.                             LSON = "false";

```

```
429.         Firebase.setString("Salon/LuzSalon/LSON", LSON);
430.         PotTotal = PotTotal - PotLS;
431.         Firebase.setFloat("PotTotal", PotTotal);
432.         PotLS = 0;
433.         Firebase.setFloat("Salon/LuzSalon/PotLS", PotLS);
434.         bit_controlLS = "false";
435.         Firebase.setString("Salon/LuzSalon/bit_controlLS", bit_controlLS);
436.     }
437. }
438. }
439.     delay(3000);
440. }
441.     if (i == 5){
442.         if (quinta_posicion == 2){
443.             if (IDLuz == IDLB){
444.                 if (LBON == "true") {
445.                     dimmer.setPower(20);
446.                     //digitalWrite(ReleLuz, LOW);
447.                     LBON = "false";
448.                     Firebase.setString("Bath/LuzBath/LBON", LBON);
449.                     PotTotal = PotTotal - PotLB;
450.                     Firebase.setFloat("PotTotal", PotTotal);
451.                     PotLB = 0;
452.                     Firebase.setFloat("Bath/LuzBath/PotLB", PotLB);
453.                     bit_controlLB = "false";
454.                     Firebase.setString("Bath/LuzBath/bit_controlLB", bit_controlLB);
455.                 }
456.             }
457.         }
458.         if (quinta_posicion == 1){
459.             if (IDLuz == IDLS){
460.                 if (LSON == "true") {
461.                     dimmer.setPower(20);
462.                     //digitalWrite(ReleLuz, LOW);
463.                     LSON = "false";
464.                     Firebase.setString("Salon/LuzSalon/LSON", LSON);
465.                     PotTotal = PotTotal - PotLS;
```

```

466.          Firebase.setFloat("PotTotal",
PotTotal);
467.          PotLS = 0;
468.          Firebase.setFloat("Salon/LuzSalon
/PotLS", PotLS);
469.          bit_controlLS = "false";
470.          Firebase.setString("Salon/LuzSalon
n/bit_controlLS", bit_controlLS);
471.      }
472.  }
473.  }
474.  delay(3000);
475.  }
476.  if (i == 4){
477.      if (cuarta_posicion == 2){
478.          if (IDLuz == IDLB){
479.              if (LBON == "true") {
480.                  dimmer.setPower(20);
481.                  //digitalWrite(ReleLuz, LOW);
482.                  LBON = "false";
483.                  Firebase.setString("Bath/LuzBath/
LBON", LBON);
484.                  PotTotal = PotTotal - PotLB;
485.                  Firebase.setFloat("PotTotal",
PotTotal);
486.                  PotLB = 0;
487.                  Firebase.setFloat("Bath/LuzBath/P
otLB", PotLB);
488.                  bit_controlLB = "false";
489.                  Firebase.setString("Bath/LuzBath/
bit_controlLB", bit_controlLB);
490.              }
491.          }
492.      }
493.      if (cuarta_posicion == 1){
494.          if (IDLuz == IDLS){
495.              if (LSON == "true") {
496.                  dimmer.setPower(20);
497.                  //digitalWrite(ReleLuz, LOW);
498.                  LSON = "false";
499.                  Firebase.setString("Salon/LuzSalon
n/LSON", LSON);
500.                  PotTotal = PotTotal - PotLS;
501.                  Firebase.setFloat("PotTotal",
PotTotal);
502.                  PotLS = 0;

```

```
503.         Firebase.setFloat("Salon/LuzSalon
/PotLS", PotLS);
504.         bit_controlLS = "false";
505.         Firebase.setString("Salon/LuzSalon
n/bit_controlLS", bit_controlLS);
506.     }
507. }
508. }
509.     delay(3000);
510. }
511.     if (i == 3){
512.         if (tercera_posicion == 2){
513.             if (IDLuz == IDLB){
514.                 if (LBON == "true") {
515.                     dimmer.setPower(20);
516.                     //digitalWrite(ReleLuz, LOW);
517.                     LBON = "false";
518.                     Firebase.setString("Bath/LuzBath/
LBON", LBON);
519.                     PotTotal = PotTotal - PotLB;
520.                     Firebase.setFloat("PotTotal",
PotTotal);
521.                     PotLB = 0;
522.                     Firebase.setFloat("Bath/LuzBath/P
otLB", PotLB);
523.                     bit_controlLB = "false";
524.                     Firebase.setString("Bath/LuzBath/
bit_controlLB", bit_controlLB);
525.                 }
526.             }
527.         }
528.         if (tercera_posicion == 1){
529.             if (IDLuz == IDLS){
530.                 if (LSON == "true") {
531.                     dimmer.setPower(20);
532.                     //digitalWrite(ReleLuz, LOW);
533.                     LSON = "false";
534.                     Firebase.setString("Salon/LuzSalon
n/LSON", LSON);
535.                     PotTotal = PotTotal - PotLS;
536.                     Firebase.setFloat("PotTotal",
PotTotal);
537.                     PotLS = 0;
538.                     Firebase.setFloat("Salon/LuzSalon
/PotLS", PotLS);
539.                     bit_controlLS = "false";
```

```
540.         Firebase.setString("Salon/LuzSalon/bit_controlLS", bit_controlLS);
541.     }
542. }
543. }
544.     delay(3000);
545. }
546.     if (i == 2){
547.         if (segunda_posicion == 2){
548.             if (IDLuz == IDLB){
549.                 if (LBON == "true") {
550.                     dimmer.setPower(20);
551.                     //digitalWrite(ReleLuz, LOW);
552.                     LBON = "false";
553.                     Firebase.setString("Bath/LuzBath/LBON", LBON);
554.                     PotTotal = PotTotal - PotLB;
555.                     Firebase.setFloat("PotTotal", PotTotal);
556.                     PotLB = 0;
557.                     Firebase.setFloat("Bath/LuzBath/PotLB", PotLB);
558.                     bit_controlLB = "false";
559.                     Firebase.setString("Bath/LuzBath/bit_controlLB", bit_controlLB);
560.                 }
561.             }
562.         }
563.         if (segunda_posicion == 1){
564.             if (IDLuz == IDLS){
565.                 if (LSON == "true") {
566.                     dimmer.setPower(20);
567.                     //digitalWrite(ReleLuz, LOW);
568.                     LSON = "false";
569.                     Firebase.setString("Salon/LuzSalon/LSON", LSON);
570.                     PotTotal = PotTotal - PotLS;
571.                     Firebase.setFloat("PotTotal", PotTotal);
572.                     PotLS = 0;
573.                     Firebase.setFloat("Salon/LuzSalon/PotLS", PotLS);
574.                     bit_controlLS = "false";
575.                     Firebase.setString("Salon/LuzSalon/bit_controlLS", bit_controlLS);
576.                 }
577.             }
```

```

578.         }
579.         delay(3000);
580.     }
581.     if ((PotContratada*0.60) > PotTotal) {
582.         break;
583.     }
584. }
585. }
586.
587.
588. // Conditions for handling errors.
589. if (Firebase.failed()) {
590.     Serial.print("Setting /Value failed :");
591.     Serial.println(Firebase.error());
592.     delay(100);
593.     return;
594. }
595.
596. Serial.println("Setting successful");
597. Serial.println();
598. //-----*/
599.
600. }
601.

```

- **Código Arduino IDE – Sensor**

```

1. #include <ESP8266WiFi.h>
2. #include <FirebaseArduino.h>
3. #include <ArduinoJson.h>
4. #include <ESP8266HTTPClient.h>
5. #include <PZEM004Tv30.h>
6. #include <WEMOS_SHT3X.h>
7. #include <Time.h>
8. #include <TimeSib.h>
9. #include "dimmable_light.h"
10. #include <WiFiUdp.h>
11. #include <BH1750.h>
12. #include <Wire.h>
13. // #include <SoftwareSerial.h>
14.
15. #define FIREBASE_HOST "control-consumo-electrico-default-
    rtdb.firebaseio.com" //--> URL address of your Firebase Realtime
    Database.
16. #define FIREBASE_AUTH
    "ATu7ew99iTEgZLQIjeA76JgoVjCsrQKADbzbmF0U" //--> Your firebase
    database secret code.
17.
18. #define ON_Board_LED 2
19.

```

```

20. //-----SSID y CONTRASEÑA
    del Wi-Fi
21.   const char* ssid = "IngElectrica"; //"Redmi K30
    Ultra"; //"MiFibra-E4BB"; //"BIBLIOTECA"; //
22.   const char* password = "C@IngEle#U120"; //"pagalosdatos"; //"Zi
    FPczo7"; //"1357986420900"; //
23. //-----
24.
25.
26.   BH1750 LDR;
27.
28.   void setup() {
29.       Serial.begin(115200);
30.       delay(500);
31.
32.       WiFi.begin(ssid, password);
33.       Serial.println("");
34.
35.       pinMode(ON_Board_LED, OUTPUT);
36.       digitalWrite(ON_Board_LED, HIGH); //--> Enciende el LED
    hasta que se conecta
37.
38.       //ESPERA A LA CONEXIÓN
39.       Serial.print("Connecting");
40.       while (WiFi.status() != WL_CONNECTED) {
41.           Serial.print(".");
42.           digitalWrite(ON_Board_LED, LOW);
43.           delay(250);
44.           digitalWrite(ON_Board_LED, HIGH);
45.           delay(250);}
46.
47.
48.       digitalWrite(ON_Board_LED, HIGH); //--> Apaga el LED cuando
    se ha conectado
49.
50.       //Si se ha conectado correctamente, muestra la dirección IP
51.       Serial.println("");
52.       Serial.print("Conectado a: ");
53.       Serial.println(ssid);
54.       Serial.print("Dirección IP: ");
55.       Serial.println(WiFi.localIP());
56.       Serial.println();
57.
58.       //--CONFIGURACIÓN DE FIREBASE--
59.       Firebase.begin(FIREBASE_HOST, FIREBASE_AUTH);
60.
61.       //--CONFIGURACION DE SENSOR DE LUMINOSIDAD--
62.       Wire.begin(); //D2 y D1 (SDA, SCL)
63.       LDR.begin();
64.   }
65.
66.   void loop() {
67.
68.       //--INICIALIZO LAS VARIABLES--
69.       float IDLuz = 1; // 1 = Salón , 2 = Baño
70.       float LDRBath, IDLB;
71.       float LDRSalon, IDLS;
72.
73.       //--SE CARGAN LAS VARIABLES BOOLEANAS--
74.       IDLB = ((Firebase.getString("Bath/LuzBath/IDLB")).substring
    (1)).toFloat();

```

```

75.
76.     IDLS = ((Firebase.getString("Salon/LuzSalon/IDLS")).substri
ng(1)).toFloat();
77.
78.
79.     if (IDLuz == IDLB){
80.         //--BAÑO--
81.         LDRBath = LDR.readLightLevel();
82.         LDRBath = LDRBath * 1.25;
83.         Firebase.setFloat("Bath/LuzBath/LDRBath", LDRBath);
84.     }
85.     if (IDLuz == IDLS){
86.         //--SALÓN--
87.         LDRSalon = LDR.readLightLevel();
88.         LDRSalon = LDRSalon * 1.25;
89.         Firebase.setFloat("Salon/LuzSalon/LDRSalon",
LDRSalon);
90.     }
91.
92.     // Conditions for handling errors.
93.     if (Firebase.failed()) {
94.         Serial.print("Setting /Value failed :");
95.         Serial.println(Firebase.error());
96.         delay(500);
97.         return;
98.     }
99.
100.    Serial.println("Setting successful");
101.    Serial.println();
102.    //-----*/
103.
104. }

```

- **Código Arduino IDE – Climatización**

```

1. #include <ESP8266WiFi.h>
2. #include <FirebaseArduino.h>
3. #include <ArduinoJson.h>
4. #include <ESP8266HTTPClient.h>
5. #include <PZEM004Tv30.h>
6. #include <WEMOS_SHT3X.h>
7. #include <Time.h>
8. #include <TimeSib.h>
9. #include "dimmable_light.h"
10.    #include <WiFiUdp.h>
11.    // #include <SoftwareSerial.h>
12.
13.    #define FIREBASE_HOST "control-consumo-electrico-default-
rtbd.firebaseio.com" //--> URL address of your Firebase Realtime
Database.
14.    #define FIREBASE_AUTH
"ATu7ew99iTEgZLQIjeA76JgoVjCsrQKADbzbmF0U" //--> Your firebase
database secret code.
15.
16.    #define ON_Board_LED 2
17.
18.    //-----SSID y CONTRASEÑA
del Wi-Fi

```

```

19.  const char* ssid = "IngElectrica"; //"Redmi K30
    Ultra"; //"MiFibra-E4BB"; //"BIBLIOTECA"; //
20.  const char* password = "C@IngEle#U120"; //"pagalosdatos"; //"Zi
    FPczo7"; //"1357986420900"; //
21.  //-----
22.
23.  PZEM004Tv30 pzem(14,12, 0x01); //D5 (TX), D6 (RX)
24.  //IPAddress ip(192,168,1,1);
25.
26.  int nivel, bucle;
27.
28.  //Climatizacion
29.  SHT3X sht30(0x45); //D1 y D2
30.  const int ReleON = D3;
31.  const int ReleCalor = D4;
32.
33.
34.  //NTP Server para el tiempo
35.  static const char ntpServerName[] = "us.pool.ntp.org";
36.  const int timeZone = 1;
37.  WiFiUDP Udp;
38.  unsigned int localPort = 8888;
39.  time_t getNtpTime();
40.  void sendNTPpacket(IPAddress &address);
41.  const int NTP_PACKET_SIZE = 48;
42.  byte packetBuffer[NTP_PACKET_SIZE];
43.
44.  time_t getNtpTime()
45.  {
46.      IPAddress ntpServerIP;
47.
48.      while (Udp.parsePacket() > 0) ;
49.      WiFi.hostByName(ntpServerName, ntpServerIP);
50.      sendNTPpacket(ntpServerIP);
51.      uint32_t beginWait = millis();
52.      while (millis() - beginWait < 1500) {
53.          int size = Udp.parsePacket();
54.          if (size >= NTP_PACKET_SIZE) {
55.              Udp.read(packetBuffer, NTP_PACKET_SIZE);
56.              unsigned long secsSince1900;
57.              secsSince1900 = (unsigned long)packetBuffer[40] << 2
4;
58.              secsSince1900 |= (unsigned long)packetBuffer[41] << 1
6;
59.              secsSince1900 |= (unsigned long)packetBuffer[42] << 8
;
60.              secsSince1900 |= (unsigned long)packetBuffer[43];
61.              return secsSince1900 - 2208988800UL + timeZone * SECS
_PER_HOUR;
62.          }
63.      }
64.      return 0;
65.  }
66.
67.  void sendNTPpacket(IPAddress &address)
68.  {
69.      memset(packetBuffer, 0, NTP_PACKET_SIZE);
70.      packetBuffer[0] = 0b11100011;
71.      packetBuffer[1] = 0;
72.      packetBuffer[2] = 6;
73.      packetBuffer[3] = 0xEC;

```

```
74.     packetBuffer[12] = 49;
75.     packetBuffer[13] = 0x4E;
76.     packetBuffer[14] = 49;
77.     packetBuffer[15] = 52;
78.     Udp.beginPacket(address, 123);
79.     Udp.write(packetBuffer, NTP_PACKET_SIZE);
80.     Udp.endPacket();
81. }
82.
83. void setup() {
84.     Serial.begin(115200);
85.     delay(500);
86.
87.     WiFi.begin(ssid, password);
88.     Serial.println("");
89.
90.     pinMode(ON_Board_LED, OUTPUT);
91.     digitalWrite(ON_Board_LED, HIGH); //--> Enciende el LED
    hasta que se conecta
92.
93.     //ESPERA A LA CONEXIÓN
94.     Serial.print("Connecting");
95.     while (WiFi.status() != WL_CONNECTED) {
96.         Serial.print(".");
97.         digitalWrite(ON_Board_LED, LOW);
98.         delay(250);
99.         digitalWrite(ON_Board_LED, HIGH);
100.        delay(250);}
101.
102.    //NTP
103.    Udp.begin(localPort);
104.    setSyncProvider(getNtpTime);
105.
106.
107.    digitalWrite(ON_Board_LED, HIGH); //--> Apaga el LED cuando
    se ha conectado
108.
109.    //Si se ha conectado correctamente, muestra la dirección IP
110.    Serial.println("");
111.    Serial.print("Conectado a: ");
112.    Serial.println(ssid);
113.    Serial.print("Dirección IP: ");
114.    Serial.println(WiFi.localIP());
115.    Serial.println();
116.
117.    //--CONFIGURACIÓN DE FIREBASE--
118.    Firebase.begin(FIREBASE_HOST, FIREBASE_AUTH);
119.
120.    //--CONFIGURACIÓN DE PINES--
121.    pinMode(ReleON, OUTPUT);
122.    pinMode(ReleCalor, OUTPUT);
123.    digitalWrite(ReleON, LOW);
124.    digitalWrite(ReleCalor, LOW);
125.
126. }
127.
128. void loop() {
129.
130.    //--INICIALIZO LAS VARIABLES--
131.    int ListaPrioridad;
132.    float PotContratada;
```

```

133.     float PotTotal;
134.     float HoraEncendido, MinutoEncendido, HoraApagado,
        MinutoApagado;
135.     float Tiempo, TiempoSet1, TiempoSet2,
        TiempoEncendido = 0, TiempoApagado = 0;
136.
137.     //COCINA
138.     // String ReleCalorCocina, ReleONCocina, HornoON,
        InduccionON, LavadoraON, LavavajillasON, MicroondasON,
        SecadoraON, TeleCocinaON, AutoCocina, LCON;
139.     // String bit_controlLC = "false", bit_controlClimaCocina
        = "false", bit_controlHorno = "false", bit_controlInduccion =
        "false", bit_controlLavadora = "false";
140.     // String bit_controlLavavajillas = "false",
        bit_controlMicroondas = "false", bit_controlSecadora = "false",
        bit_controlTeleCocina = "false";
141.     // float PotCalentador, PotClimaCocina, TemperaturaCocina,
        UmbralInfCocina, UmbralSupCocina;
142.     // float PotHorno, PotInduccion, PotLavadora,
        PotLavavajillas, PotMicroondas, PotSecadora, PotTeleCocina,
        LDRCocina, NivelCocina, PotLC;
143.     //PASILLO
144.     // bool ReleCalorPasillo, ReleONPasillo, AutoPasillo,
        LPON;
145.     // float PotClimaPasillo, TemperaturaPasillo,
        UmbralInfPasillo, UmbralSupPasillo, LDRPasillo, NivelPasillo,
        PotLP;
146.     //SALÓN
147.     String ReleCalorSalon, ReleONSalon, AutoSalon,
        bit_controlClimaSalon = "false";
148.     float PotClimaSalon, TemperaturaSalon, UmbralInfSalon,
        UmbralSupSalon, newPotClimaSalon;
149.     //HABITACIÓN 1
150.     // bool ReleCalorHab1, ReleONHab1, AutoHab1, LH1ON,
        TeleHab1ON;
151.     // float PotClimaHab1, TemperaturaHab1, UmbralInfHab1,
        UmbralSupHab1, LDRHab1, NivelHab1, PotLH1, PotTeleHab1;
152.     //HABITACIÓN 2
153.     // bool ReleCalorHab2, ReleONHab2, AutoHab2, LH2ON, PCON;
154.     // float PotClimaHab2, TemperaturaHab2, UmbralInfHab2,
        UmbralSupHab2, LDRHab2, NivelHab2, PotLH2, PotPC;
155.
156.
157.     //--SE CARGAN LAS VARIABLES BOOLEANAS--
158.
159.     /* ReleCalorCocina =
        Firebase.getString("Cocina/ClimaCocina/ReleCalorCocina");
160.     ReleONCocina =
        Firebase.getString("Cocina/ClimaCocina/ReleONCocina");
161.     HornoON = Firebase.getString("Cocina/Horno/HornoON");
162.     InduccionON =
        Firebase.getString("Cocina/Induccion/InduccionON");
163.     LavadoraON =
        Firebase.getString("Cocina/Lavadora/LavadoraON");
164.     MicroondasON =
        Firebase.getString("Cocina/Microondas/MicroondasON");
165.     SecadoraON =
        Firebase.getString("Cocina/Secadora/SecadoraON");
166.     TeleCocinaON =
        Firebase.getString("Cocina/TeleCocina/TeleCocinaON");

```

```

167.     AutoCocina =
        Firebase.getString("Cocina/LuzCocina/AutoCocina");
168.     LCON = Firebase.getString("Cocina/LuzCocina/LCON");
169.
170.     ReleCalorPasillo =
        Firebase.getBool("Pasillo/ClimaPasillo/ReleCalorPasillo");
171.     ReleONPasillo =
        Firebase.getBool("Pasillo/ClimaPasillo/ReleONPasillo");
172.     AutoPasillo =
        Firebase.getBool("Pasillo/LuzPasillo/AutoPasillo");
173.     LPON = Firebase.getBool("Pasillo/LuzPasillo/LPON");*/
174.
175.     ReleCalorSalon = Firebase.getString("Salon/ClimaSalon/ReleC
        alorSalon");
176.     ReleONSalon = Firebase.getString("Salon/ClimaSalon/ReleONSA
        lon");
177.     AutoSalon = Firebase.getString("Salon/LuzSalon/AutoSalon");
178.     bit_controlClimaSalon = Firebase.getString("Salon/ClimaSalo
        n/bit_controlClimaSalon");
179.     PotClimaSalon = Firebase.getFloat("Salon/ClimaSalon/PotClim
        aSalon");
180.
181.     /* ReleCalorHab1 =
        Firebase.getBool("Hab1/ClimaHab1/ReleCalorHab1");
182.     ReleONHab1 = Firebase.getBool("Hab1/ClimaHab1/ReleONHab1");
183.     AutoHab1 = Firebase.getBool("Hab1/LuzHab1/AutoHab1");
184.     LH1ON = Firebase.getBool("Hab1/LuzHab1/LH1ON");
185.     TeleHab1ON = Firebase.getBool("Hab1/TeleHab1/TeleHab1ON");
186.
187.     ReleCalorHab2 =
        Firebase.getBool("Hab2/ClimaHab2/ReleCalorHab2");
188.     ReleONHab2 = Firebase.getBool("Hab2/ClimaHab2/ReleONHab2");
189.     AutoHab2 = Firebase.getBool("Hab2/LuzHab2/AutoHab2");
190.     LH2ON = Firebase.getBool("Hab2/LuzHab2/LH2ON");
191.     PCON = Firebase.getBool("Hab2/PC/PCON");*/
192.
193.     PotContratada = ((Firebase.getString("PotContratada")).subs
        tring(1)).toFloat();
194.     PotTotal = Firebase.getFloat("PotTotal");
195.     ListaPrioridad = ((Firebase.getString("ListaPrioridad")).su
        bstring(1)).toInt();
196.     HoraApagado = (Firebase.getString("HoraApagado")).toFloat()
        ;
197.     MinutoApagado = (Firebase.getString("MinutoApagado")).toFlo
        at();
198.     HoraEncendido = (Firebase.getString("HoraEncendido")).toFlo
        at();
199.     MinutoEncendido = (Firebase.getString("MinutoEncendido")).t
        oFloat();
200.
201.     int sexta_posicion = ListaPrioridad % 10;
202.     int quinta_posicion = sexta_posicion % 10;
203.     int cuarta_posicion = quinta_posicion % 10;
204.     int tercera_posicion = cuarta_posicion % 10;
205.     int segunda_posicion = tercera_posicion % 10;
206.     int primera_posicion = segunda_posicion % 10;
207.
208.     if ((PotContratada*0.60) > PotTotal){
209.         if(sht30.get()==0){
210.             TemperaturaSalon=(sht30.cTemp); //Temperatura en
                Celsius

```

```

211.         Firebase.setFloat("Salon/ClimaSalon/TemperaturaSalon",
    TemperaturaSalon);
212.     }
213.     Tiempo = hour()*100 + minute();
214.     TiempoSet1 = HoraApagado*100 + MinutoApagado;
215.     TiempoSet2 = HoraEncendido*100 + MinutoEncendido;
216.     if (((TiempoSet2 - TiempoSet1) >= 0) && (AutoSalon == "true")){ //Los tiempos puestos están en el mismo día (no está el
    de apagado a las 23h y el de encendido a las 8h por ejemplo)
217.         if ((Tiempo >= TiempoSet1) && (Tiempo < TiempoSet2)){
218.             AutoSalon = "false";
219.             ReleONSalon = "false";
220.             Firebase.setString("Salon/ClimaSalon/ReleONSalon",
    ReleONSalon);
221.         }
222.     } else {
223.         if (((Tiempo >= TiempoSet1) && (Tiempo <= 2359)) || ((
    Tiempo >= 0) && (Tiempo < TiempoSet2))) && (AutoSalon == "true")
    ){
224.             AutoSalon = "false";
225.             ReleONSalon = "false";
226.             Firebase.setString("Salon/ClimaSalon/ReleONSalon",
    ReleONSalon);
227.         }
228.     }
229.
230.     if (AutoSalon == "true"){
231.         UmbralInfSalon = ((Firebase.getString("Salon/ClimaSalon
    /UmbralInfSalon")).substring(1)).toFloat();
232.         UmbralSupSalon = ((Firebase.getString("Salon/ClimaSalon
    /UmbralSupSalon")).substring(1)).toFloat();
233.         Serial.println(UmbralInfSalon);
234.         Serial.println(UmbralSupSalon);
235.         Serial.println(TemperaturaSalon);
236.         if ((TemperaturaSalon <= UmbralInfSalon) || (Temperatur
    aSalon >= UmbralSupSalon)){
237.             ReleONSalon = "true";
238.             digitalWrite(ReleON, HIGH);
239.             Firebase.setString("Salon/ClimaSalon/ReleONSalon",
    ReleONSalon);
240.             if (TemperaturaSalon <= UmbralInfSalon){
241.                 ReleCalorSalon = "true";
242.                 digitalWrite(ReleCalor, HIGH);
243.                 Firebase.setString("Salon/ClimaSalon/ReleCalorSalon
    ", ReleCalorSalon);
244.             } else {
245.                 ReleCalorSalon = "false";
246.                 digitalWrite(ReleCalor, LOW);
247.                 Firebase.setString("Salon/ClimaSalon/ReleCalorSalon
    ", ReleCalorSalon);
248.             }
249.             if (bit_controlClimaSalon == "false"){
250.                 bit_controlClimaSalon = "true";
251.                 Firebase.setString("Salon/ClimaSalon/bit_controlCli
    maSalon", bit_controlClimaSalon);
252.                 delay(3000);
253.                 PotClimaSalon = pzem.power();
254.                 Firebase.setFloat("Salon/ClimaSalon/PotClimaSalon",
    PotClimaSalon);
255.                 PotTotal = PotTotal + PotClimaSalon;
256.                 Firebase.setFloat("PotTotal", PotTotal);

```

```

257.         }
258.         newPotClimaSalon = pzem.power();
259.         if (abs(newPotClimaSalon - PotClimaSalon) > 1) {
260.             PotTotal = PotTotal - PotClimaSalon;
261.             PotClimaSalon = newPotClimaSalon;
262.             Firebase.setFloat("Salon/ClimaSalon/PotClimaSalon",
PotClimaSalon);
263.             PotTotal = PotTotal + PotClimaSalon;
264.             Firebase.setFloat("PotTotal", PotTotal);
265.         }
266.     } else {
267.         ReleONSalon = "false";
268.         digitalWrite(ReleON, LOW);
269.         Firebase.setString("Salon/ClimaSalon/ReleONSalon",
ReleONSalon);
270.         PotTotal = PotTotal - PotClimaSalon;
271.         Firebase.setFloat("PotTotal", PotTotal);
272.         PotClimaSalon = 0;
273.         Firebase.setFloat("Salon/ClimaSalon/PotClimaSalon",
PotClimaSalon);
274.         bit_controlClimaSalon = "false";
275.         Firebase.setString("Salon/ClimaSalon/bit_controlClima
Salon", bit_controlClimaSalon);
276.     }
277. } else {
278.     if (ReleONSalon == "true") {
279.         digitalWrite(ReleON, HIGH);
280.         if (ReleCalorSalon == "true") {
281.             digitalWrite(ReleCalor, HIGH);
282.         } else {
283.             digitalWrite(ReleCalor, LOW);
284.         }
285.         if (bit_controlClimaSalon == "false") {
286.             bit_controlClimaSalon = "true";
287.             Firebase.setString("Salon/ClimaSalon/bit_controlCli
maSalon", bit_controlClimaSalon);
288.             delay(3000);
289.             PotClimaSalon = pzem.power();
290.             Firebase.setFloat("Salon/ClimaSalon/PotClimaSalon",
PotClimaSalon);
291.             PotTotal = PotTotal + PotClimaSalon;
292.             Firebase.setFloat("PotTotal", PotTotal);
293.         }
294.         newPotClimaSalon = pzem.power();
295.         if (abs(newPotClimaSalon - PotClimaSalon) > 1) {
296.             PotTotal = PotTotal - PotClimaSalon;
297.             PotClimaSalon = newPotClimaSalon;
298.             Firebase.setFloat("Salon/ClimaSalon/PotClimaSalon",
PotClimaSalon);
299.             PotTotal = PotTotal + PotClimaSalon;
300.             Firebase.setFloat("PotTotal", PotTotal);
301.         }
302.     } else {
303.         digitalWrite(ReleON, LOW);
304.         PotTotal = PotTotal - PotClimaSalon;
305.         Firebase.setFloat("PotTotal", PotTotal);
306.         PotClimaSalon = 0;
307.         Firebase.setFloat("Salon/ClimaSalon/PotClimaSalon",
PotClimaSalon);
308.         bit_controlClimaSalon = "false";

```

```

309.         Firebase.setString("Salon/ClimaSalon/bit_controlClima
Salon", bit_controlClimaSalon);
310.     }
311. }
312. }
313. //
314. // CONTROLANDO POTENCIA SUPERIOR AL 60%
315. //
316.     else if ((PotContratada*0.60) <= PotTotal) {
317.
318.         for (int i = 6; i >= 1; i--){ //No voy a hacer la
programación de las demás habitaciones, solo de las dos que
tengo en el prototipo
319.             if (i == 6){
320.                 if (sexta_posicion == 1){
321.                     if (ReleONSalon == "true") {
322.                         digitalWrite(ReleON, LOW);
323.                         Firebase.setString("Salon/ClimaSalon/ReleONSalon"
, ReleONSalon);
324.                         PotTotal = PotTotal - PotClimaSalon;
325.                         Firebase.setFloat("PotTotal", PotTotal);
326.                         PotClimaSalon = 0;
327.                         Firebase.setFloat("Salon/ClimaSalon/PotClimaSalon
", PotClimaSalon);
328.                         bit_controlClimaSalon = "false";
329.                         Firebase.setString("Salon/ClimaSalon/bit_controlC
limaSalon", bit_controlClimaSalon);
330.                     }
331.                 }
332.                 delay(3000);
333.             }
334.             if (i == 5){
335.                 if (quinta_posicion == 1){
336.                     if (ReleONSalon == "true") {
337.                         digitalWrite(ReleON, LOW);
338.                         Firebase.setString("Salon/ClimaSalon/ReleONSalon"
, ReleONSalon);
339.                         PotTotal = PotTotal - PotClimaSalon;
340.                         Firebase.setFloat("PotTotal", PotTotal);
341.                         PotClimaSalon = 0;
342.                         Firebase.setFloat("Salon/ClimaSalon/PotClimaSalon
", PotClimaSalon);
343.                         bit_controlClimaSalon = "false";
344.                         Firebase.setString("Salon/ClimaSalon/bit_controlC
limaSalon", bit_controlClimaSalon);
345.                     }
346.                 }
347.                 delay(3000);
348.             }
349.             if (i == 4){
350.                 if (cuarta_posicion == 1){
351.                     if (ReleONSalon == "true") {
352.                         digitalWrite(ReleON, LOW);
353.                         Firebase.setString("Salon/ClimaSalon/ReleONSalon"
, ReleONSalon);
354.                         PotTotal = PotTotal - PotClimaSalon;
355.                         Firebase.setFloat("PotTotal", PotTotal);
356.                         PotClimaSalon = 0;
357.                         Firebase.setFloat("Salon/ClimaSalon/PotClimaSalon
", PotClimaSalon);
358.                         bit_controlClimaSalon = "false";

```

```

359.         Firebase.setString("Salon/ClimaSalon/bit_controlC
    limaSalon", bit_controlClimaSalon);
360.     }
361. }
362.     delay(3000);
363. }
364.     if (i == 3){
365.         if (tercera_posicion == 1){
366.             if (ReleONSalon == "true") {
367.                 digitalWrite(ReleON, LOW);
368.                 Firebase.setString("Salon/ClimaSalon/ReleONSalon"
    , ReleONSalon);
369.                 PotTotal = PotTotal - PotClimaSalon;
370.                 Firebase.setFloat("PotTotal", PotTotal);
371.                 PotClimaSalon = 0;
372.                 Firebase.setFloat("Salon/ClimaSalon/PotClimaSalon
    ", PotClimaSalon);
373.                 bit_controlClimaSalon = "false";
374.                 Firebase.setString("Salon/ClimaSalon/bit_controlC
    limaSalon", bit_controlClimaSalon);
375.             }
376.         }
377.         delay(3000);
378.     }
379.     if (i == 2){
380.         if (segunda_posicion == 1){
381.             if (ReleONSalon == "true") {
382.                 digitalWrite(ReleON, LOW);
383.                 Firebase.setString("Salon/ClimaSalon/ReleONSalon"
    , ReleONSalon);
384.                 PotTotal = PotTotal - PotClimaSalon;
385.                 Firebase.setFloat("PotTotal", PotTotal);
386.                 PotClimaSalon = 0;
387.                 Firebase.setFloat("Salon/ClimaSalon/PotClimaSalon
    ", PotClimaSalon);
388.                 bit_controlClimaSalon = "false";
389.                 Firebase.setString("Salon/ClimaSalon/bit_controlC
    limaSalon", bit_controlClimaSalon);
390.             }
391.         }
392.         delay(3000);
393.     }
394.     if ((PotContratada*0.60) > PotTotal) {
395.         break;
396.     }
397. }
398. }
399.
400.
401. //-----
402.
403. // Conditions for handling errors.
404. if (Firebase.failed()) {
405.     Serial.print("Setting /Value failed :");
406.     Serial.println(Firebase.error());
407.     delay(500);
408.     return;
409. }
410.
411. Serial.println("Setting successful");
412. Serial.println();

```

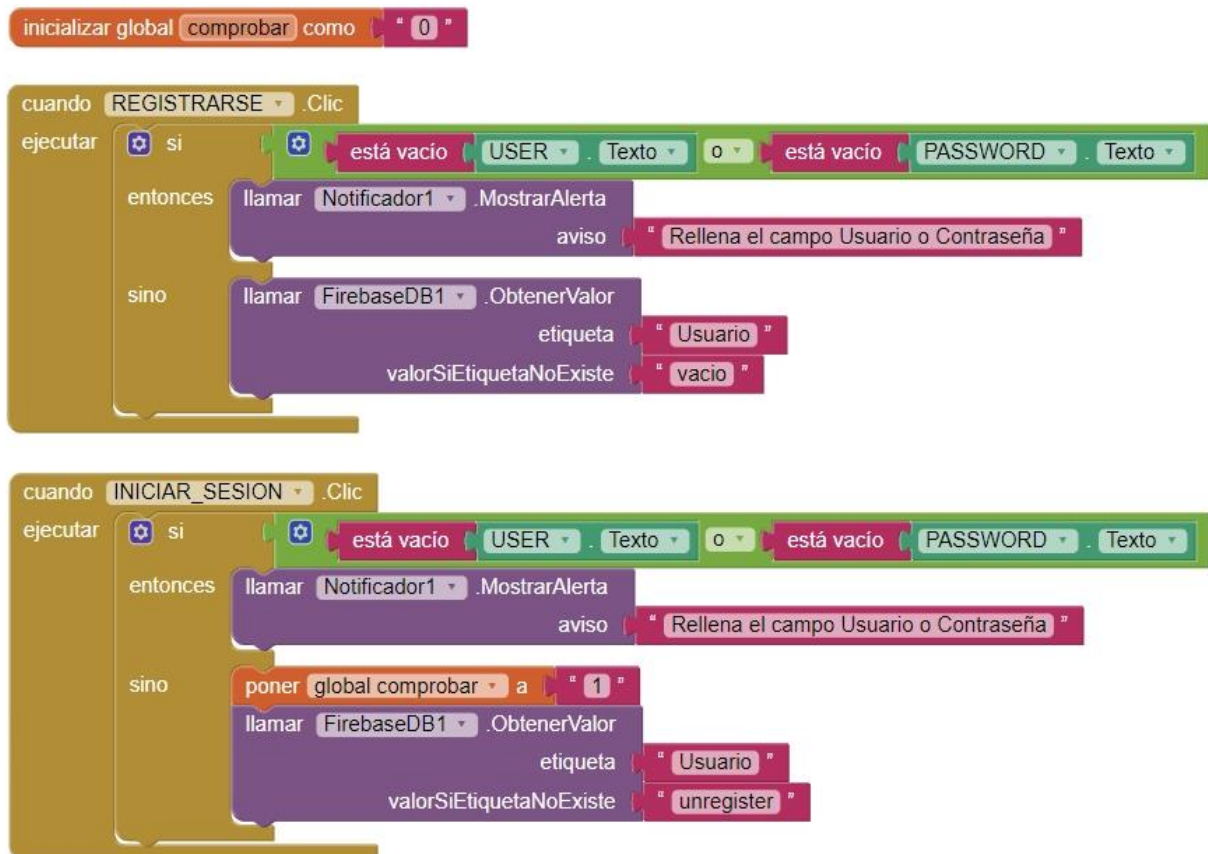
```

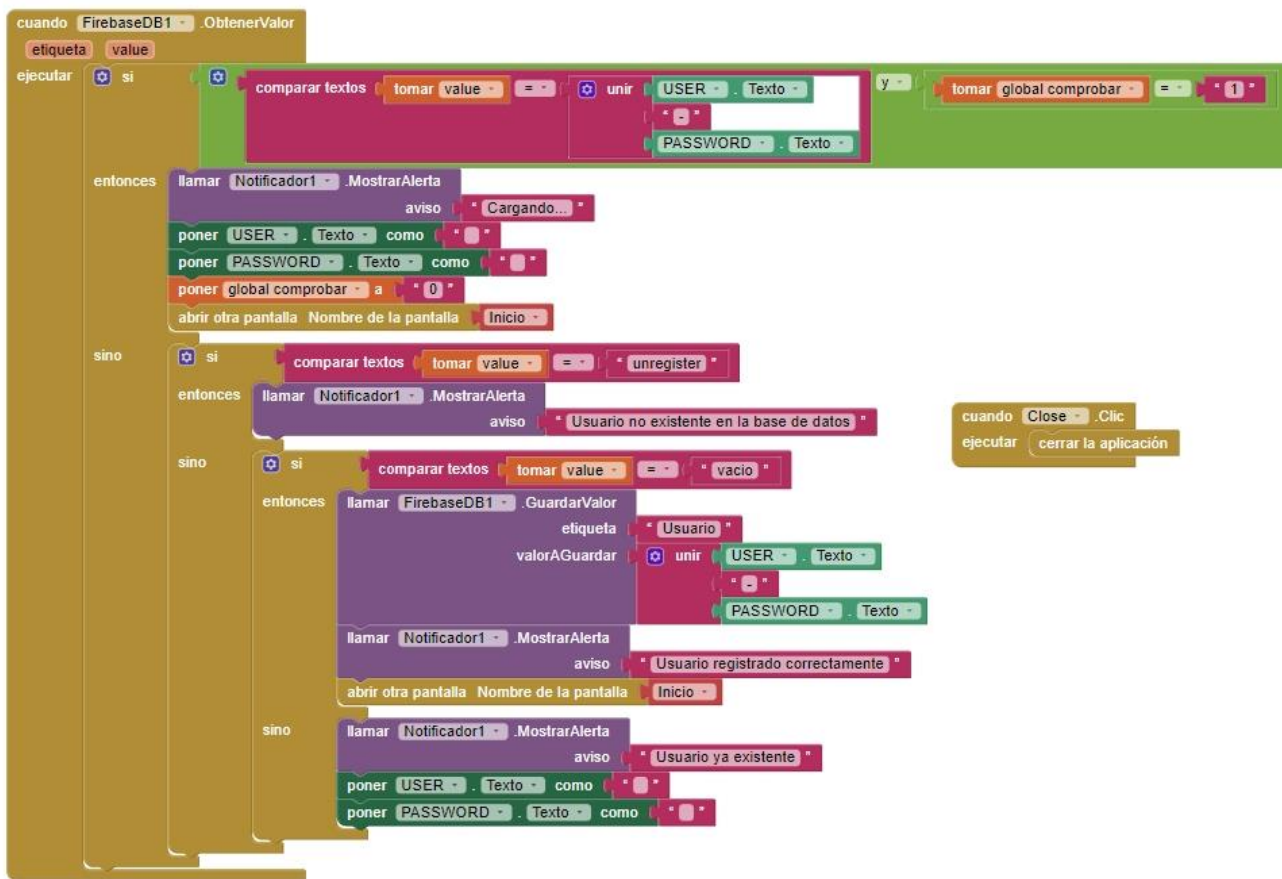
413. //-----*/
414. }
415.

```

- **Código de bloques de la app (MIT App Inventor)**

Pantalla de Login



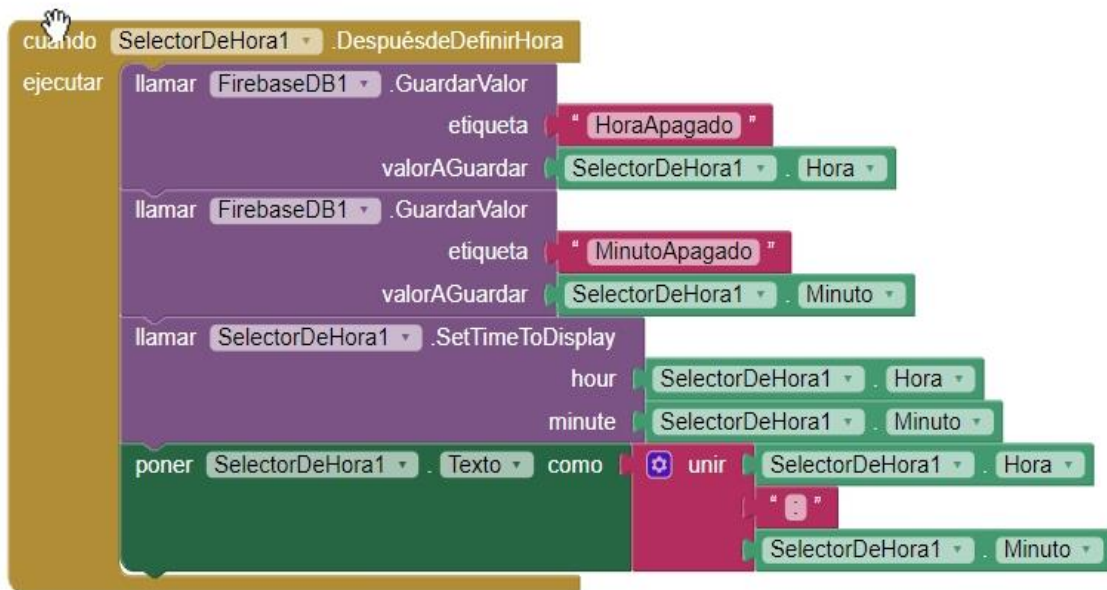


Pantalla de Inicio



Pantalla de Ajustes Smart Home

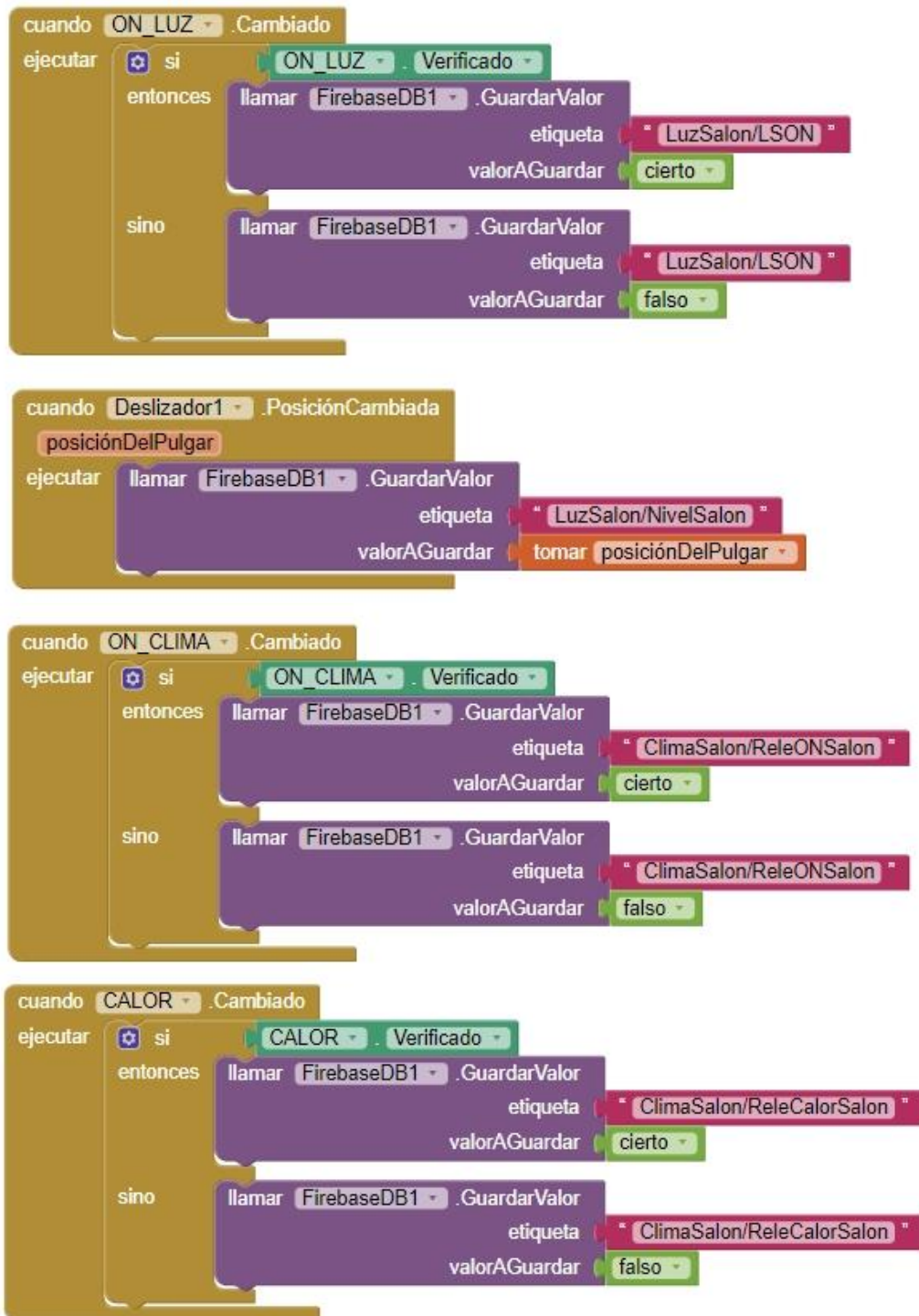


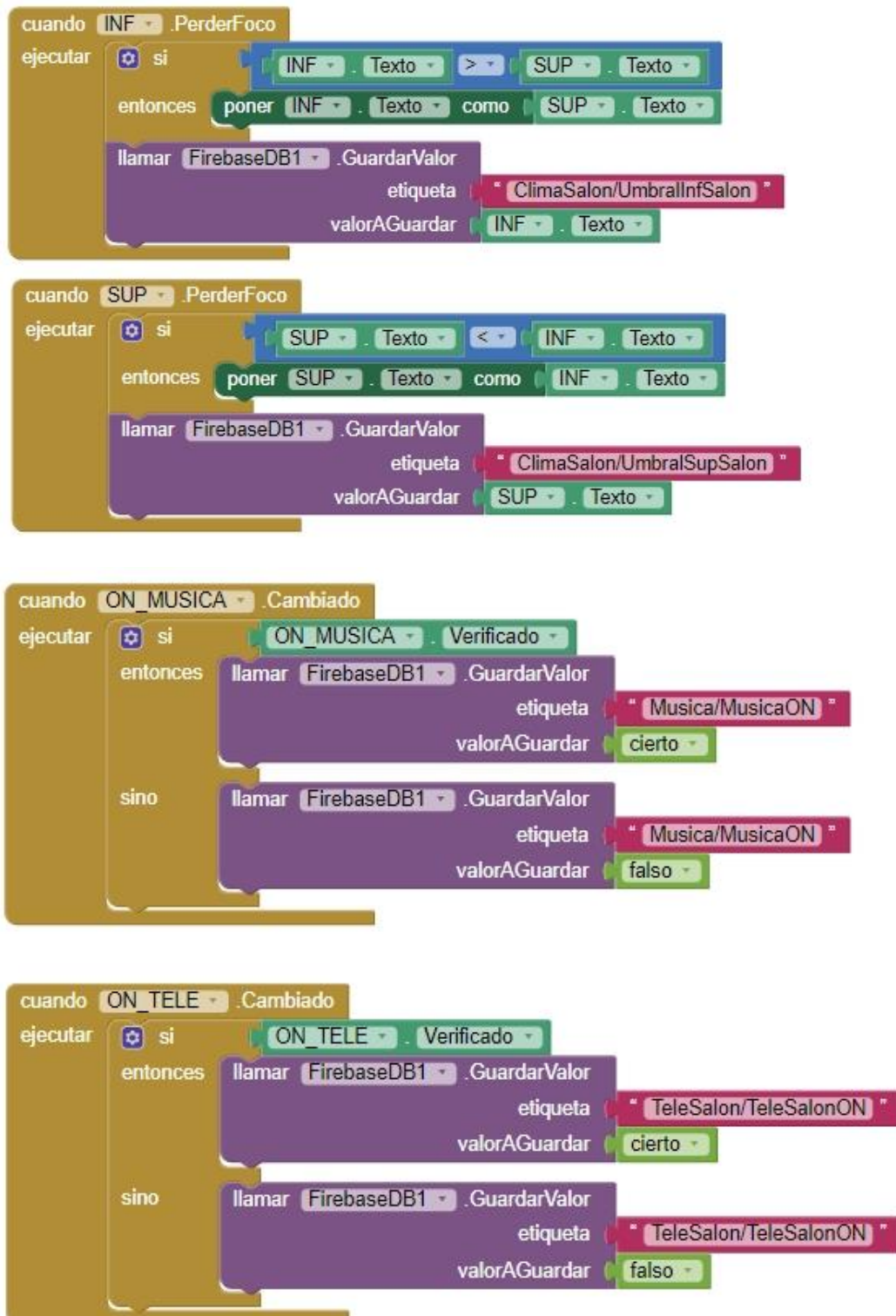


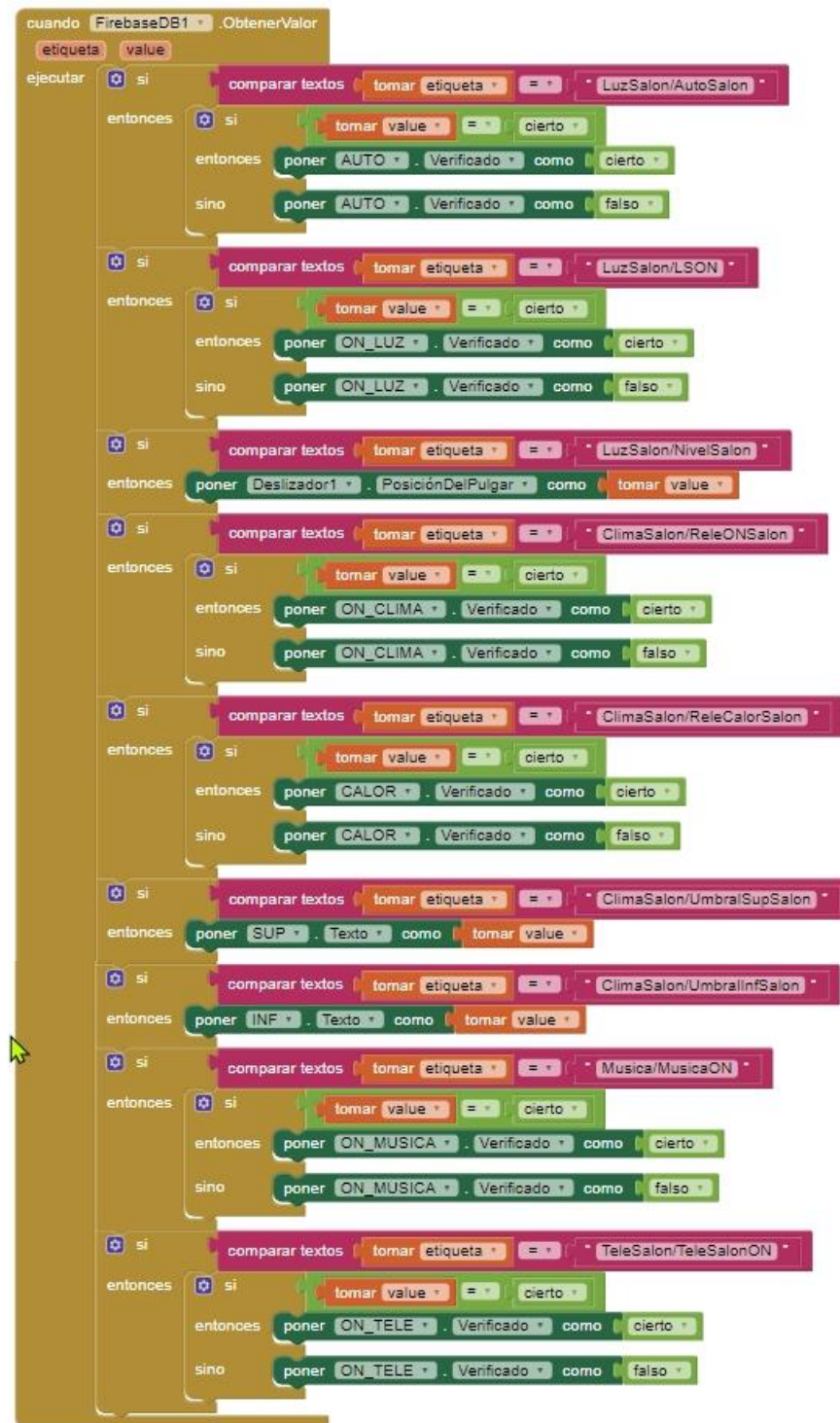


Pantalla de Salón





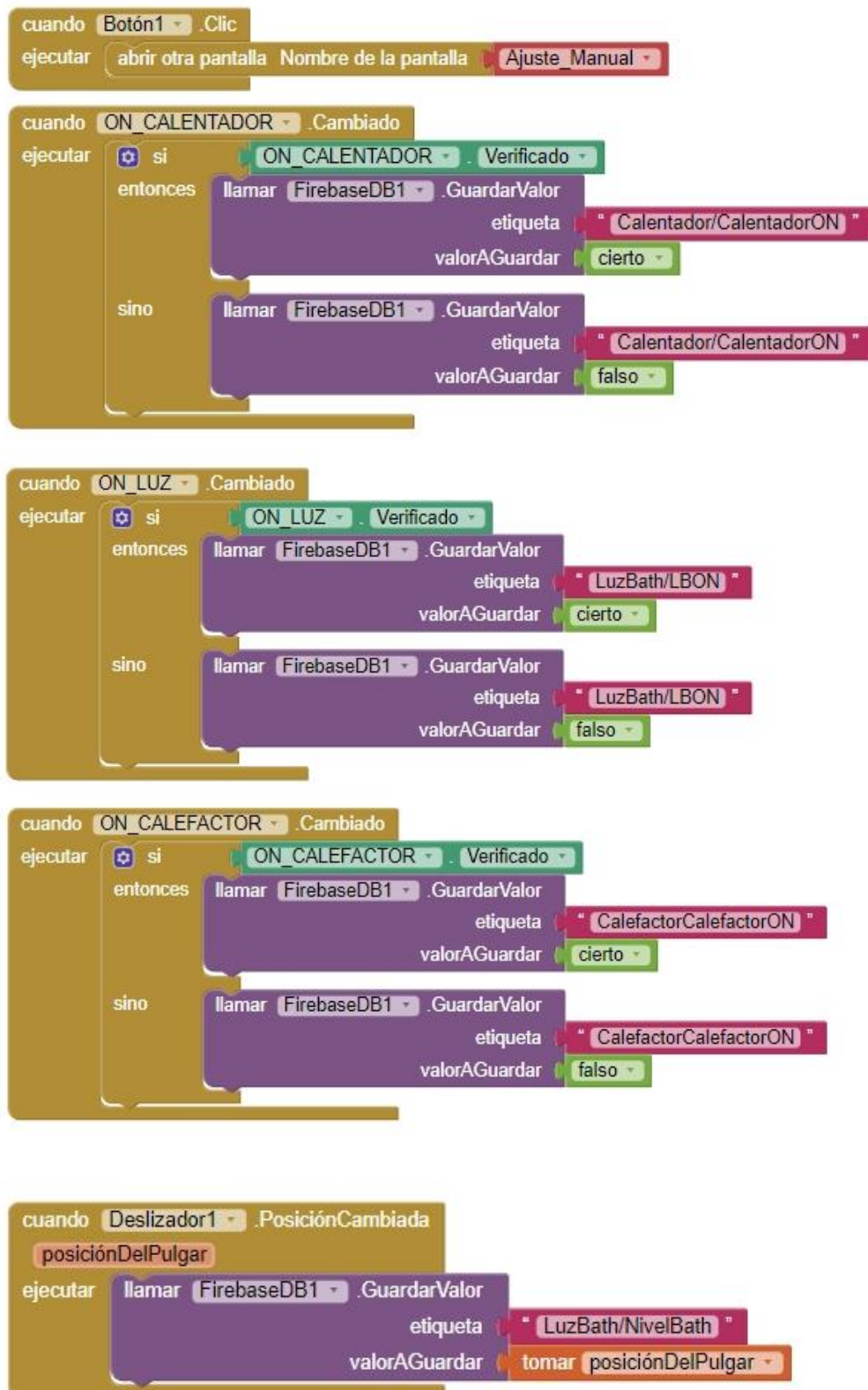


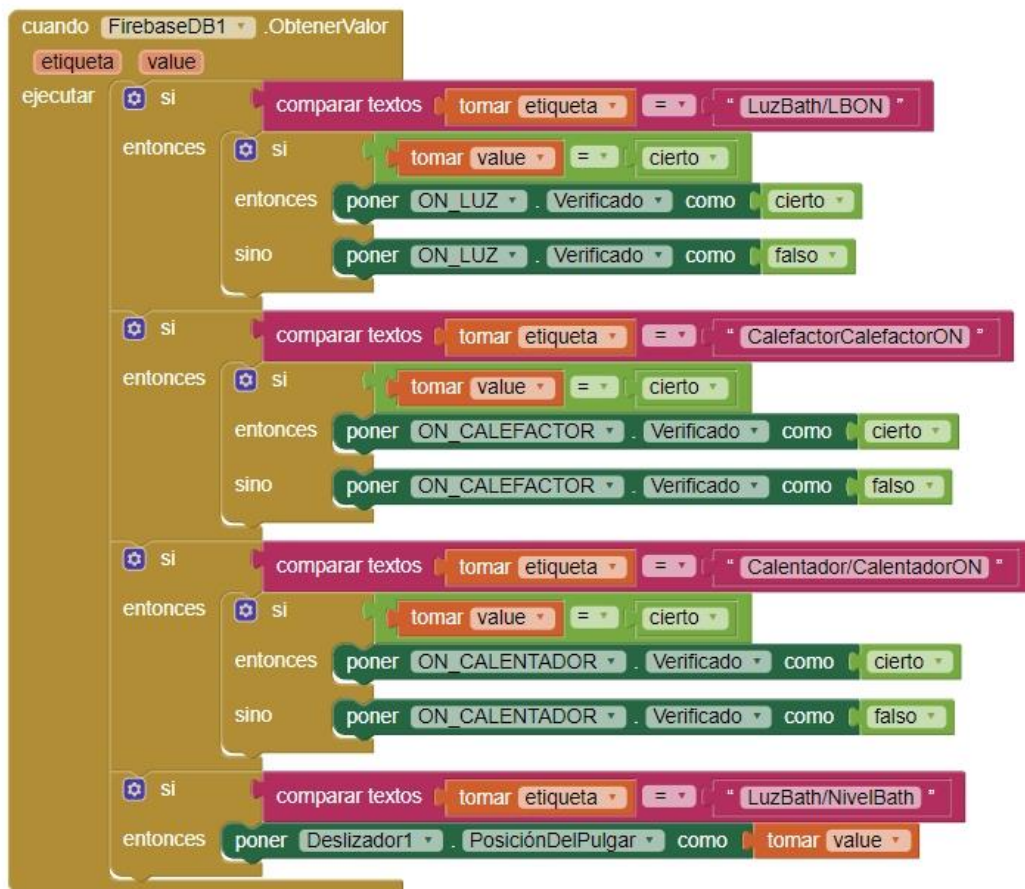




Pantalla de Cuarto de baño







Pantalla de ConsumoNow

