UNIVERSIDAD FINIS TERRAE
FACULTAD DE INGENIERÍA
INGENIERÍA CIVIL EN INFORMÁTICA Y TELECOMUNICACIONES

DESARROLLO DE UN SISTEMA DE ENCHUFE INTELIGENTE CON CONTROL DESDE APLICACIÓN MÓVIL PARA LA AUTOMATIZACIÓN Y ASISTENCIA DEL HOGAR

NICOLÁS ALEXANDER GÓMEZ SILVA

Trabajo de título presentado a la Facultad de Ingeniería de la Universidad Finis Terrae, para optar al Título de Ingeniero Civil en Informática y Telecomunicaciones

Profesor Guía: Rodrigo Paredes

Santiago, Chile
2025

Quiero expresar mi más sincero agradecimiento a todas las personas que me acompañaron a lo largo de este proceso, lleno de aprendizajes, desafíos y crecimiento. Agradezco especialmente a mi familia, que, a pesar de la distancia, nunca dejó de brindarme su apoyo incondicional y su amor en cada etapa de este camino.

También quiero dedicar unas palabras a mi compañera de vida, mi novia, quien ha estado presente en todo momento, entregándome su compañía, comprensión y aliento, incluso en los días más difíciles.

Finalmente, mi mayor gratitud es para mi abuelo. Su recuerdo ha sido mi fuerza y su ejemplo, mi guía. Gracias a él he podido avanzar con determinación y sin mirar atrás. Que descanse en paz.

ÍNDICE DE CONTENIDOS

ÍNDICE DE ILUSTRACIONES

ÍNDICE DE TABLAS

Resumen

La presente memoria aborda la baja adopción de tecnologías domóticas en viviendas chilenas, especialmente en contextos donde residen adultos mayores o personas con movilidad reducida. Se identifica la necesidad de una solución accesible, automatizada y fácil de usar, que permita monitorear el entorno y controlar dispositivos eléctricos de forma remota. Como objetivo general, el proyecto diseña e implementa un sistema de enchufe inteligente orientado a la automatización doméstica y asistencia en el hogar.

El sistema utiliza componentes de bajo costo y fácil integración: un microcontrolador Wemos D1 Mini, sensores DHT22 y LDR para monitoreo ambiental, un relé SSR-40DA para el control de corriente y una fuente de alimentación IRM-03-5. El microcontrolador procesa las lecturas de los sensores y publica los datos en tiempo real a través del protocolo MQTT, usando la plataforma Adafruit IO. Se desarrolla también una aplicación móvil en Kotlin que permite visualizar los datos, controlar el enchufe y configurar automatizaciones.

El sistema funciona correctamente en cuanto a recolección de datos, ejecución de automatizaciones y control remoto. La comunicación entre el microcontrolador, la nube y la aplicación se mantiene estable, y las automatizaciones responden adecuadamente a las condiciones definidas.

Palabras clave:

Domótica, automatización, IoT, sensores ambientales, asistencia domiciliaria.

Abstract

This work addresses the low adoption of home automation technologies in Chilean households, particularly in environments where elderly people or individuals with reduced mobility reside. It identifies the need for an accessible, automated, and easy-to-use solution that allows environmental monitoring and remote control of electrical devices. The project designs and implements a smart plug system focused on home automation and domestic assistance.

The system integrates low-cost, easy-to-implement components: a Wemos D1 Mini microcontroller, DHT22 and LDR sensors for environmental monitoring, an SSR-40DA relay for electrical control, and an RM-03-5 power supply. The microcontroller processes the sensor readings and publishes real-time data through the MQTT protocol using the Adafruit IO platform. A mobile application developed in Kotlin allows users to visualize data, control the plug, and configure automation parameters.

The system performs correctly in terms of data acquisition, remote control, and automatic responses. Communication between the microcontroller, cloud service, and mobile application remains stable, and the system reacts properly to configured environmental conditions.

Keywords:

Domotic, home automation, IoT, environmental sensors, home care.

Capítulo 1

1. Introducción

La evolución de la tecnología y la creciente demanda por soluciones que mejoren la calidad de vida en los hogares ha impulsado el desarrollo de sistemas inteligentes capaces de automatizar tareas cotidianas, optimizar el uso de recursos y aumentar la seguridad. En este contexto, la domótica se posiciona como una herramienta clave para transformar viviendas tradicionales en espacios más eficientes, adaptables y confortables.

Si bien la implementación de estas tecnologías ha crecido en diversos países, en Chile su adopción aún se encuentra en una etapa inicial, especialmente en sectores residenciales. Esta situación se ve acentuada por la falta de soluciones accesibles y personalizadas que respondan a las necesidades específicas de la población, como lo es la asistencia a personas mayores o con movilidad reducida.

Ante este panorama, el presente proyecto propone el desarrollo de un enchufe inteligente con capacidad de monitoreo ambiental y control automatizado, orientado tanto a aumentar la comodidad del hogar como al acompañamiento domiciliario. El sistema incluye sensores de temperatura, humedad y luz, y se conecta a una aplicación móvil que permite al usuario visualizar datos en tiempo real, configurar umbrales de acción y recibir alertas según condiciones definidas.

Este primer capítulo presenta el contexto del problema, define la propuesta de solución, los objetivos del proyecto, su alcance, limitaciones y relevancia dentro del área de la automatización doméstica.

1.1. Descripción de la situación actual

En los últimos años, la demanda por soluciones que integran tecnología en el entorno doméstico crece de manera sostenida. El avance de la domótica permite mejorar la eficiencia energética, facilitar el control de dispositivos y aumentar el nivel de seguridad en los hogares. Sin embargo, en países como Chile, la adopción de estas tecnologías aún se mantiene en niveles bajos. Según el *Manual de Aplicación - Construcción Sustentable* del Ministerio de Vivienda y Urbanismo (2019), la implementación de sistemas domóticos en

el país enfrenta desafíos relacionados con la falta de conocimiento técnico por parte de los usuarios y la escasa disponibilidad de soluciones adaptadas a las necesidades locales.

En este contexto, las personas mayores representan un grupo que enfrenta barreras adicionales para la adopción de tecnologías domésticas. De acuerdo con Geriasistencia (2019), la complejidad en el uso de nuevas tecnologías constituye un obstáculo relevante para este grupo etario, sumado a la falta de disposición y al temor de cometer errores irreversibles. Factores como problemas de audición, dificultades visuales y pérdida de destrezas motoras también dificultan su interacción con dispositivos tecnológicos (Addinformática, 2022).

Por otro lado, los sistemas convencionales de control eléctrico no incorporan capacidades de respuesta automática frente a condiciones ambientales, lo que limita su utilidad en escenarios como el sobrecalentamiento de espacios, baja humedad o ausencia de luz natural. La integración de sensores ambientales con dispositivos de control eléctrico, gestionados mediante una aplicación intuitiva, se plantea como una alternativa viable para aumentar la seguridad y el confort en el hogar, especialmente en viviendas habitadas por personas mayores o en situación de dependencia.

Ante esta realidad, se identifica la necesidad de investigar e implementar sistemas de automatización que no sólo sean eficientes, sino también accesibles, personalizables y diseñados desde una perspectiva de asistencia, permitiendo que la tecnología se adapte a los usuarios y no al revés.

1.2. Descripción de la situación problema

A pesar del avance tecnológico y la creciente disponibilidad de soluciones de automatización, muchas viviendas en Chile continúan dependiendo de sistemas tradicionales de control eléctrico que no consideran factores ambientales ni permiten una gestión remota o automatizada. Esta situación limita la implementación de estrategias efectivas orientadas a la eficiencia energética y al confort adaptativo dentro del entorno doméstico.

La brecha tecnológica se profundiza en hogares donde residen personas mayores, quienes enfrentan dificultades para acceder y operar sistemas complejos. La falta de

soluciones accesibles, intuitivas y adaptadas a sus necesidades compromete su autonomía y calidad de vida. Según la CEPAL (2021), el envejecimiento de la población en América Latina representa un desafío urgente para el diseño de entornos habitables, por lo que resulta fundamental incorporar tecnologías que promuevan la independencia y seguridad de las personas mayores.

Adicionalmente, el desconocimiento generalizado respecto a los beneficios de la domótica, junto con su percepción como una tecnología costosa o innecesaria, obstaculiza su adopción masiva en contextos residenciales. La ausencia de plataformas personalizables, de bajo costo y fácil implementación, excluye a una parte significativa de la población que podría beneficiarse del uso de sistemas inteligentes en sus hogares.

En consecuencia, se identifica como problema central la falta de soluciones domóticas accesibles, seguras y adaptadas a usuarios con necesidades especiales o limitaciones tecnológicas. Se plantea la necesidad de desarrollar un sistema que integre monitoreo ambiental, control remoto y automatización, manteniendo la simplicidad y accesibilidad, y que sea viable dentro del contexto técnico y económico de los hogares chilenos.

1.3. Descripción de la solución propuesta

La solución propuesta consiste en el desarrollo de un sistema de automatización doméstica orientado al control inteligente de un enchufe eléctrico. Este sistema incorpora funciones de monitoreo ambiental, control remoto, automatización en función de condiciones físicas del entorno y asistencia para adultos mayores.

El sistema se basa en el microcontrolador Wemos D1 Mini, seleccionado por su tamaño compacto, bajo costo y capacidad de conexión Wi-Fi. Se integran sensores DHT22, que permiten obtener lecturas precisas de temperatura y humedad ambiente, junto con un sensor LDR, encargado de medir los niveles de luminosidad para activar o desactivar dispositivos en función de la luz natural disponible.

La gestión del flujo eléctrico se realiza mediante un relé de estado sólido SSR-40DA, que garantiza una conmutación segura y silenciosa, adecuada para operar dispositivos eléctricos sin generar ruido mecánico.

Adicionalmente, la solución incorpora una aplicación móvil desarrollada en Kotlin, utilizando el framework Jetpack Compose. Esta aplicación permite al usuario controlar manualmente el enchufe, visualizar en tiempo real las variables ambientales y configurar automatizaciones personalizadas.

El sistema implementa una lógica de acceso dual, compuesta por un perfil de usuario común, diseñado para facilitar la operación a personas con baja alfabetización digital, y un perfil de cuidador, que habilita la configuración de umbrales, la activación de modos automáticos y la recepción de alertas. Esta arquitectura busca promover la autonomía del usuario final sin comprometer la supervisión ni la seguridad.

1.4. Objetivo general

- Diseñar un sistema de automatización doméstica basado en un enchufe inteligente, capaz de monitorear variables ambientales como temperatura, humedad y luz, y de controlar el flujo de corriente eléctrica de forma remota o automática, mediante una aplicación móvil con acceso dual. El sistema se orienta tanto al usuario final como al cuidador, promoviendo la seguridad y la asistencia en el entorno del hogar.

1.5. Objetivos Específicos

- Seleccionar los componentes electrónicos adecuados para la construcción del sistema, considerando criterios de bajo consumo, compatibilidad y una integración segura en entornos residenciales.
- Implementar un sistema de monitoreo ambiental, mediante sensores de temperatura, humedad y luminosidad, que permita recolectar información en tiempo real sobre las condiciones del entorno doméstico.
- Diseñar un sistema de automatización programable, capaz de activar o desactivar el enchufe en función de las condiciones ambientales detectadas o de temporizadores definidos por el usuario o el cuidador.
- Desarrollar una aplicación móvil con una interfaz diferenciada para usuarios comunes y cuidadores, que permita visualizar los datos de los sensores, controlar manualmente el enchufe y configurar los parámetros del sistema.

- Integrar comunicación en tiempo real entre el sistema físico y la aplicación móvil, utilizando el protocolo MQTT y una plataforma en la nube (Adafruit IO), con el fin de asegurar la sincronización constante de datos y comandos.
- Evaluar la viabilidad técnica y funcional del sistema mediante la ejecución de pruebas piloto en entornos reales que permitan validar su comportamiento y estabilidad.

1.6. Alcances

El presente proyecto abarca el diseño, construcción y validación de un sistema de automatización doméstica basado en un enchufe inteligente controlado por sensores ambientales y gestionado desde una aplicación móvil personalizada. El desarrollo considera tanto la implementación física del sistema como la programación del microcontrolador, la comunicación en la nube y el diseño de la interfaz de usuario.

Dentro del alcance del proyecto se incluyen los siguientes aspectos:

- La selección y ensamblaje de componentes electrónicos como el Wemos D1 Mini, el sensor DHT22 para temperatura y humedad, el sensor LDR para luminosidad ambiental, un relé de estado sólido SSR-40DA y una fuente de alimentación IRM-03-5, montados en una caja eléctrica con diseño seguro y compacto.
- La programación del microcontrolador en Arduino IDE, incorporando lectura de sensores, lógica de automatización y publicación de datos en tiempo real mediante MQTT a través de Adafruit IO.
- El desarrollo de una aplicación móvil utilizando Kotlin y Jetpack Compose, que permite el acceso a dos perfiles de usuario: uno básico (usuario común) y otro avanzado (cuidador), con capacidad de configurar automatizaciones, umbrales de control, alertas y temporizadores.
- La validación del sistema mediante pruebas funcionales en condiciones reales, que permiten comprobar la precisión de los sensores, la efectividad de las automatizaciones y la estabilidad de la comunicación entre la app y el dispositivo.

Este proyecto se enfoca principalmente en aspectos funcionales, técnicos y de accesibilidad, orientado a demostrar que es posible implementar soluciones de domótica

asequibles y adaptables al contexto chileno, especialmente útiles en hogares donde habitan personas mayores o con movilidad reducida.

1.7. Limitaciones

Si bien el proyecto cumple con los objetivos propuestos y demuestra la viabilidad de implementar un sistema de automatización doméstica funcional y accesible, existen ciertas limitaciones inherentes al alcance, tiempo y recursos disponibles, que se detallan a continuación:

- No se incluye medición directa de consumo energético. Aunque el sistema permite automatizar el encendido y apagado de dispositivos eléctricos, no incorpora sensores de corriente (como el SCT-013), por lo que no se realiza una medición cuantitativa del consumo energético. El foco del proyecto está en el monitoreo ambiental y la respuesta automatizada frente a dichas condiciones.
- No se considera una base de datos personalizada ni backend propio. La comunicación entre el microcontrolador y la aplicación se realiza exclusivamente a través de la plataforma Adafruit IO, lo que limita la flexibilidad en la gestión de datos históricos o personalización de servicios más avanzados.
- La solución está orientada a Android. El desarrollo de la aplicación se centra únicamente en el sistema operativo Android, utilizando Kotlin y Jetpack Compose. No se incluye una versión web ni iOS, aunque su diseño modular permitiría una futura expansión.
- El sistema está diseñado para controlar un solo enchufe. Aunque la arquitectura técnica permite escalar a múltiples dispositivos, el prototipo final está orientado al control de un solo punto de carga, pensado como una solución básica pero funcional para validar la propuesta.
- El sistema requiere conexión a Internet. Tanto la sincronización de datos como el envío de comandos dependen de la disponibilidad de red Wi-Fi. En ausencia de conectividad, el sistema no puede ser gestionado remotamente desde la aplicación móvil.
- La aplicación no integra asistentes virtuales como Alexa o Chromecast debido al tiempo de desarrollo del proyecto y al ser un prototipo, la creación de cuentas y

seguridad de los usuarios se omite, dando relevancia a la integración y programación de los sensores.

Estas limitaciones no afectan la funcionalidad general del sistema, pero representan oportunidades claras de mejora y expansión en futuras iteraciones del proyecto, especialmente en términos de escalabilidad, seguridad de datos y cobertura multiplataforma.

Capítulo 2

2. Marco teórico

El presente capítulo tiene como finalidad establecer los fundamentos conceptuales y técnicos vinculados a los sistemas de automatización en entornos residenciales. Se abordan temáticas clave como la domótica, los sensores ambientales, los protocolos de comunicación, las plataformas de servicios en la nube para el Internet de las Cosas (IoT), y las estrategias de automatización orientadas a mejorar la calidad de vida en el hogar.

Esta revisión teórica permite comprender las tecnologías comúnmente empleadas en la automatización del hogar, sus criterios de selección y los principios que guían su implementación, considerando aspectos como la viabilidad técnica, la escalabilidad y la accesibilidad.

2.1. Domótica e internet de las cosas

La domótica se refiere a la integración de tecnologías que permiten automatizar y controlar diversos sistemas dentro de una vivienda, con el propósito de optimizar la eficiencia energética, aumentar la seguridad y mejorar el confort del usuario. Mediante sensores, actuadores y plataformas de control, es posible gestionar dispositivos eléctricos de forma automática o remota, favoreciendo la personalización del entorno doméstico (Gómez, 2019).

En este contexto, el Internet de las Cosas (IoT, por sus siglas en inglés) constituye el pilar fundamental de la domótica moderna. Se define como una red de dispositivos interconectados que recopilan e intercambian datos a través de internet, permitiendo la toma de decisiones autónomas o remotas. Esta conectividad habilita el desarrollo de hogares inteligentes, donde los dispositivos se comunican entre sí para responder dinámicamente a condiciones del entorno (Bajwa et al., 2021).

2.2. Teoría de Sistemas Abiertos

La teoría de sistemas abiertos plantea que los sistemas mantienen su funcionamiento mediante la interacción constante con su entorno, adaptándose a condiciones cambiantes

para preservar su estabilidad, eficiencia y efectividad. En el contexto de los sistemas domóticos y de automatización del hogar, este enfoque resulta fundamental, ya que dichos sistemas deben percibir información del entorno físico mediante sensores, procesarla y responder de manera adecuada a través de mecanismos de control.

La retroalimentación juega un papel clave dentro de esta teoría, ya que permite ajustar el comportamiento del sistema en función de los cambios detectados en el ambiente o en las preferencias del usuario. Esta capacidad de respuesta dinámica es especialmente valorada en aplicaciones orientadas a la asistencia de personas mayores, donde la adaptabilidad del sistema contribuye directamente a mejorar la autonomía y seguridad de los habitantes (Sajjad & Kastrati, 2021).

2.3. Microcontroladores y Procesadores

Los microcontroladores son dispositivos integrados que combinan un procesador, memoria y periféricos de entrada/salida en un solo chip compacto, permitiendo el control eficiente de sistemas electrónicos. Actúan como el núcleo operativo de dispositivos inteligentes, gestionando múltiples funcionalidades de manera precisa y optimizada, esenciales para sistemas domóticos.

- **Wemos D1 Mini:** Este microcontrolador, basado en el chip ESP8266 de Expressif Systems. Su pequeño tamaño y conectividad Wi-Fi integrada lo convierten en una solución compacta y eficiente para proyectos de IoT. El Wemos D1 Mini, que se aprecia en la Figura 1, es compatible con el entorno de programación Arduino IDE y permite la gestión remota de dispositivos en tiempo real. Además, soporta protocolos de comunicación como MQTT y HTTP, proporcionando una transmisión de datos eficiente y segura. Estas características lo hacen ideal para automatización doméstica, asegurando un equilibrio entre rendimiento, simplicidad y bajo consumo energético (IoTEDU, 2020; Circuito.io, 2023).

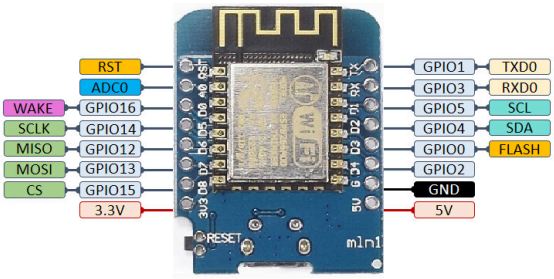


Figura 1: Wemos D1 Mini

2.3.1. Sensores

Los sensores ambientales son componentes fundamentales en los sistemas domóticos, ya que permiten a los microcontroladores recopilar datos del entorno físico para tomar decisiones automatizadas. Su uso posibilita la creación de respuestas inteligentes basadas en variables como temperatura, humedad, o niveles de luz.

- Sensor de temperatura y humedad **DHT22**

El sensor **DHT22**, mostrado en la Figura 2, se utiliza para monitorear la temperatura y la humedad del entorno. Este dispositivo ofrece un amplio rango de operación, midiendo temperaturas entre –40 °C y 80 °C con una precisión de ±0.5 °C, además de registrar niveles de humedad relativa desde 0 % hasta 100 % RH. Su voltaje de operación de 3.3 V a 6 V asegura compatibilidad con plataformas basadas en Arduino y ESP8266. Estas características lo convierten en un componente confiable para automatizaciones basadas en condiciones ambientales, como el ajuste de dispositivos conectados para mejorar el confort del hogar (MCI Electronics, s.f.).

Figura 2: Sensor de temperatura y humedad DHT22

- Sensor de luz **LDR**

El sensor **LDR** (Light Dependent Resistor), ilustrado en la Figura 3, se utiliza para monitorear los niveles de luz ambiental y activar automatizaciones en sistemas de domótica. Este componente varía su resistencia en función de la intensidad lumínica: en condiciones de alta iluminación su resistencia puede descender hasta aproximadamente 1 kΩ, mientras que en oscuridad puede elevarse a valores cercanos a 10 kΩ. Con un voltaje máximo de operación de 150 V, resulta compatible con plataformas como Arduino y ESP8266, siempre que se emplee en configuraciones adecuadas y seguras. Gracias a su ángulo de detección amplio, el LDR permite ajustar automáticamente la iluminación de dispositivos conectados, activándose en condiciones de baja luminosidad y desactivándose cuando hay suficiente luz ambiental. Estas características lo convierten en un componente confiable y económico para el control de iluminación automatizada, mejorando tanto la comodidad del usuario como la personalización del entorno (MCI Electronics, s.f.).

Figura 3: Sensor de luz LDR

2.3.2. Actuadores y Componentes Electrónicos Complementarios

En los sistemas domóticos, la integración de componentes electrónicos es fundamental para permitir la automatización, monitoreo y control eficiente de distintos dispositivos del entorno. Cada componente cumple una función específica dentro del sistema, desde la recolección de datos ambientales hasta la ejecución de acciones automatizadas.

- Relé de Estado Sólido **SSR-40 DA**

El relé de estado sólido **SSR-40DA** (ver Figura 4) es un componente electrónico utilizado para controlar el flujo de corriente hacia los dispositivos eléctricos de manera rápida y silenciosa. A diferencia de los relés mecánicos, este modelo no posee partes móviles, lo que elimina el desgaste mecánico y prolonga significativamente su vida útil. Estas características lo convierten en una alternativa confiable para aplicaciones de automatización doméstica en las que se requiere un funcionamiento constante, seguro y de alto rendimiento. Con capacidad de conmutación en tensiones de hasta 250 V AC y una corriente nominal de 40 A, el SSR-40DA asegura una operación robusta en sistemas que demandan conmutaciones frecuentes (MCI Electronics, s.f.).

Figura 4: Relé de estado sólido SSR-40 DA

- Fuente de Alimentación **IRM-03-5**

El módulo de fuente de alimentación **IRM-03-5** (ver Figura 5) convierte la corriente alterna domiciliaria de 220–240 V AC en una salida regulada de 5VDC, necesaria para alimentar de manera segura los componentes electrónicos del sistema, incluidos el microcontrolador, los sensores y el relé de estado sólido. Su diseño compacto y encapsulado permite un montaje eficiente en espacios reducidos, garantizando la estabilidad del sistema y protegiendo los circuitos frente a variaciones eléctricas. Además, incorpora protección contra sobrecargas y certificaciones internacionales de seguridad, lo que la convierte en una opción confiable para aplicaciones de domótica y automatización residencial, donde la fiabilidad y la seguridad eléctrica son fundamentales (MCI Electronics, s.f.).

Figura 5: Fuente de alimentación IRM-03-5

2.4. Protocolo de comunicación - Adafruit IO y Protocolo MQTT

El protocolo MQTT (Message Queuing Telemetry Transport), ampliamente adoptado en aplicaciones de IoT por su eficiencia, bajo consumo de recursos y capacidad para transmitir datos en tiempo real, permite establecer la comunicación entre una aplicación y un sistema domótico.

En el contexto IoT, uno de los protocolos de comunicación más utilizados es MQTT (Message Queuing Telemetry Transport). Este protocolo se caracteriza por su eficiencia, bajo consumo de recursos y capacidad para transmitir datos en tiempo real, lo que lo hace especialmente adecuado para entornos con recursos limitados (Light, 2017).

Plataformas en la nube compatibles con MQTT, como Adafruit IO, permiten la publicación y suscripción de datos mediante el modelo publisher/subscriber. Estas plataformas proporcionan interfaces gráficas accesibles para visualizar variables ambientales o monitorear el estado de dispositivos, y permiten enviar comandos de control sin necesidad de infraestructura de servidor propia. Su compatibilidad con microcontroladores de bajo consumo y su facilidad de integración han facilitado su adopción en sistemas domóticos y educativos (Bajwa et al., 2021).

2.5. Automatización y Control

La automatización en domótica se refiere a la capacidad de los sistemas para ejecutar acciones de forma autónoma en respuesta a variaciones en las condiciones ambientales o a órdenes definidas por el usuario, mediante el uso de sensores y algoritmos preprogramados. Estos sistemas se basan en tecnologías de IoT, que permiten la recolección y transmisión de datos en tiempo real, facilitando una respuesta dinámica y contextual.

La implementación de automatizaciones contribuye significativamente tanto al confort como a la eficiencia energética en entornos residenciales. Al eliminar tareas manuales repetitivas, se mejora la experiencia del usuario, optimizando el uso de los dispositivos conectados. Además, el desarrollo de interfaces accesibles y configuraciones personalizables permite adaptar el funcionamiento de los sistemas domóticos a las necesidades individuales de cada hogar (Nugraha & Fauzi, 2021).

2.5.1. Arduino IDE

Arduino IDE es una herramienta de código abierto ampliamente utilizada para la programación de microcontroladores gracias a su interfaz sencilla y su compatibilidad con una amplia variedad de dispositivos. En este proyecto, Arduino IDE es fundamental para desarrollar la lógica del sistema. Este microcontrolador permite configurar los componentes para la automatización y el control de dispositivos conectados.

La herramienta también facilita la implementación de la comunicación entre el enchufe inteligente y la aplicación móvil mediante protocolos como HTTP y MQTT, asegurando un flujo de datos eficiente y confiable. La incorporación de bibliotecas

específicas en Arduino IDE permite aprovechar al máximo las capacidades de los componentes, desde la lectura de sensores hasta la ejecución de automatizaciones avanzadas. Esto hace posible el desarrollo de un sistema completo y funcional que combina eficiencia energética, automatización y control remoto.

2.5.2. Android Studio

Android Studio es el entorno de desarrollo integrado (IDE) oficial para la creación de aplicaciones móviles en el sistema operativo Android. Está basado en IntelliJ IDEA e incluye un conjunto de herramientas diseñadas para facilitar el diseño, desarrollo, prueba y depuración de aplicaciones móviles. Entre sus características destacadas se encuentran un editor visual para la construcción de interfaces gráficas (usando XML), emuladores de dispositivos virtuales, soporte para múltiples versiones de Android y compatibilidad con bibliotecas externas que permiten extender las funcionalidades de las aplicaciones (Hardy, 2019).

Android Studio admite los lenguajes de programación Java y Kotlin, e incorpora herramientas para el análisis de rendimiento, gestión de dependencias mediante Gradle y pruebas automatizadas. Su enfoque modular y flexible lo convierte en una herramienta robusta tanto para desarrolladores principiantes como avanzados, facilitando la construcción de aplicaciones móviles eficientes, adaptables y con comunicación en tiempo real mediante tecnologías como HTTP, REST o MQTT.

2.5.3. Multímetro

El **multímetro digital** es una herramienta imprescindible en electrónica, ya que permite medir magnitudes eléctricas como voltaje, corriente y resistencia. Su función principal radica en verificar el correcto funcionamiento de circuitos y componentes, facilitando la detección de fallos como cortocircuitos, cables interrumpidos o niveles de voltaje fuera de rango. En proyectos de automatización, resulta especialmente útil durante las fases de diseño, montaje y mantenimiento, pues permite comprobar la continuidad de conexiones, validar las condiciones de alimentación de los dispositivos y confirmar la integridad de señales tanto digitales como analógicas. Estas pruebas son fundamentales para asegurar la integridad eléctrica del sistema (Fluke, 2024).

Capítulo 3

3. Estado del arte

3.1. Domótica e IoT

La domótica ha experimentado un desarrollo significativo en las últimas dos décadas gracias a la convergencia entre dispositivos electrónicos programables, redes inalámbricas e IoT. Esta integración permite automatizar y controlar múltiples funciones del hogar como iluminación, climatización, seguridad y consumo energético desde plataformas remotas, mejorando la eficiencia operativa, la comodidad y la calidad de vida de los usuarios (Bajwa et al., 2021; Mohamed & El Shenawy, 2023).

El IoT se basa en la interconexión de dispositivos a través de internet, permitiendo que sensores, actuadores y microcontroladores compartan información en tiempo real y respondan de forma autónoma a condiciones del entorno. Según Statista (2023), se proyecta que para 2025 existirán más de 75 mil millones de dispositivos IoT conectados en el mundo, siendo el ámbito doméstico uno de los de mayor expansión.

Investigaciones han demostrado que la aplicación de tecnologías IoT en hogares inteligentes contribuye a la sostenibilidad energética, al control remoto de cargas eléctricas, a la detección de anomalías en el entorno y a la asistencia de personas con movilidad reducida (Sajjad & Kastrati, 2021; Fernández et al., 2021). La integración de IoT en sistemas domóticos representa, por tanto, una evolución hacia hogares más inteligentes, seguros e inclusivos.

3.2. Microcontroladores en soluciones IoT

Los microcontroladores son componentes clave en la arquitectura de sistemas IoT. Permiten adquirir datos desde sensores, procesarlos localmente y comunicarlos a plataformas externas. Entre los más utilizados se encuentran el ESP8266 y el ESP32, desarrollados por Espressif Systems. Ambos integran conectividad Wi-Fi, lo que los hace ideales para aplicaciones domóticas sin necesidad de hardware adicional (Hasan et al., 2020; Ristik et al., 2021).

El ESP8266 es valorado por su bajo costo, eficiencia energética y facilidad de programación mediante entornos como Arduino IDE. El ESP32, por su parte, presenta capacidades superiores: doble núcleo de procesamiento, mayor cantidad de pines GPIO, soporte para Bluetooth y cifrado nativo (Maier et al., 2017). Estas diferencias permiten seleccionar el microcontrolador más adecuado en función de los requerimientos del sistema.

Estudios recientes muestran que ambos microcontroladores se han implementado con éxito en proyectos de automatización del hogar, logrando mejoras sustanciales en consumo energético y reducción de costos de implementación (Rokonuzzaman et al., 2021; Hasan et al., 2020). Su compatibilidad con múltiples sensores y protocolos los posiciona como estándares de facto en la industria IoT.

3.3. Automatización doméstica y eficiencia energética

La automatización de dispositivos eléctricos en el hogar ha demostrado ser una estrategia eficaz para reducir el consumo energético, mejorar el confort y facilitar la gestión remota de cargas. Los enchufes inteligentes representan una solución accesible para implementar estas funciones, permitiendo controlar el encendido y apagado de aparatos, monitorear su consumo y programar horarios de funcionamiento (Wang & Kim, 2023; Lin et al., 2020).

La literatura indica que el uso de smart plugs, combinados con sensores ambientales y rutinas automatizadas, puede reducir entre un 20 % y un 30 % del consumo eléctrico en hogares donde se implementan correctamente (Neto et al., 2024; Rokonuzzaman et al., 2021). Además, el análisis de datos recopilados por estos dispositivos permite adaptar su comportamiento a patrones de uso, optimizando aún más la eficiencia.

En escenarios donde se promueve la sostenibilidad, el uso de sistemas de automatización basados en sensores ha demostrado ser eficaz no solo para ahorrar energía, sino también para extender la vida útil de dispositivos eléctricos y reducir la huella de carbono del hogar (Lin et al., 2020).

3.4. Protocolos de Comunicación para IoT

En sistemas IoT, la elección del protocolo de comunicación es crítica para garantizar eficiencia, confiabilidad y escalabilidad. MQTT (Message Queuing Telemetry Transport) ha ganado gran popularidad debido a su arquitectura ligera, operación asincrónica y bajo consumo de ancho de banda. Estas características lo hacen ideal para dispositivos de recursos limitados como sensores o microcontroladores (Light, 2017; Mistry et al., 2020).

MQTT permite la publicación y suscripción de mensajes mediante un broker, lo que posibilita una comunicación bidireccional eficiente entre dispositivos y plataformas móviles o en la nube. Por otro lado, el protocolo HTTP, aunque más pesado, sigue siendo ampliamente utilizado por su compatibilidad universal con servicios web y aplicaciones móviles (Kwon et al., 2022).

Una arquitectura híbrida que combina MQTT para comunicaciones en tiempo real y HTTP para consultas y configuraciones proporciona un equilibrio adecuado entre rendimiento, escalabilidad y facilidad de integración en sistemas de Internet de las Cosas. De esta manera, MQTT asegura una comunicación bidireccional con baja latencia y consumo reducido de ancho de banda, mientras que HTTP facilita procesos de configuración y gestión más seguros gracias a su naturaleza transaccional. Este enfoque mixto resulta especialmente útil en entornos de automatización doméstica, donde se requiere tanto eficiencia en la transmisión de datos como robustez en la administración del sistema (EMQX, 2024).

3.5. Aplicaciones móviles en sistemas domóticos

Las aplicaciones móviles representan la interfaz principal entre el usuario y los sistemas domóticos. Permiten no solo controlar dispositivos desde cualquier lugar, sino también visualizar variables en tiempo real, configurar automatizaciones y recibir alertas sobre condiciones anómalas del entorno (Hardy, 2019; Nugraha & Fauzi, 2021).

El desarrollo de estas aplicaciones se realiza comúnmente en entornos como Android Studio, que proporciona herramientas de diseño visual, emuladores de prueba y

soporte para lenguajes como Java y Kotlin. Estas herramientas permiten crear interfaces accesibles y funcionales, incluso para usuarios con baja alfabetización digital.

Diversos estudios recomiendan que las interfaces móviles para control domótico incluyan funcionalidades como control de energía, programación horaria, gestión multiusuario y visualización gráfica de sensores (Fernández et al., 2021; Luo et al., 2020), lo que mejora la experiencia del usuario y promueve la adopción de tecnologías inteligentes.

3.6. Accesibilidad y asistencia a personas mayores

Un campo emergente dentro de la domótica es su aplicación para el apoyo de personas mayores o con movilidad reducida. Sistemas capaces de detectar condiciones del entorno como temperatura extrema, baja luminosidad o ausencia de movimiento pueden activar alertas o acciones que mejoren la seguridad y autonomía del usuario (Sajjad & Kastrati, 2021; Fernández et al., 2021).

La literatura enfatiza la importancia de desarrollar soluciones de bajo costo, no invasivas y fáciles de usar para este segmento de la población. La implementación de sensores ambientales combinados con control automatizado y monitoreo remoto por cuidadores ha demostrado reducir el estrés, aumentar la independencia y prevenir accidentes domésticos (Luo et al., 2020; Nugraha & Fauzi, 2021).

Además, se ha identificado que la personalización de las interfaces, la posibilidad de tener distintos perfiles de acceso (como usuarios principales y secundarios), y la integración con asistentes virtuales, son factores que incrementan la usabilidad y la aceptación de estos sistemas en adultos mayores.

3.7. Novedad del proyecto y comparativa con otros estudios

A diferencia de otros trabajos que se enfocan principalmente en el monitoreo energético mediante enchufes inteligentes (Lin et al., 2020; Neto et al., 2024), esta propuesta integra múltiples sensores ambientales (temperatura, humedad y luz) para permitir automatizaciones adaptadas a las condiciones del entorno, ampliando las capacidades tradicionales de control básico.

Además, se incorpora una interfaz móvil con sistema de doble cuenta, lo que permite que cuidadores gestionen remotamente las funciones del sistema. Este enfoque de asistencia tecnológica responde a desafíos actuales en inclusión digital, especialmente en adultos mayores, y ha sido poco abordado en estudios previos (Sajjad & Kastrati, 2021; Fernández et al., 2021).

En términos técnicos, se emplea una arquitectura híbrida con MQTT y HTTP, lo que permite combinar eficiencia en la comunicación y compatibilidad con aplicaciones móviles (Mistry et al., 2020; Kwon et al., 2022). Esto garantiza control en tiempo real y configuración flexible, mejorando la experiencia de usuario frente a soluciones más rígidas.

En síntesis, la novedad del proyecto radica en combinar automatización ambiental, asistencia remota, control desde aplicación móvil y bajo costo, dentro de una solución orientada tanto a la eficiencia energética como a la accesibilidad.

Capítulo 4

4. Metodología V-Model

Esta metodología describe el proceso estructurado seguido para el desarrollo del enchufe inteligente, abarcando las fases de diseño, implementación y validación del sistema. Se opta por el modelo **V-Model** debido a que ofrece una estructura clara que vincula cada etapa de desarrollo con su correspondiente fase de pruebas, lo que permite una verificación y validación continua a lo largo del proyecto. A diferencia de metodologías como **ágil**, que priorizan iteraciones rápidas y flexibilidad más adecuadas para desarrollos exclusivamente de software, o **cascada**, que no contempla validaciones intermedias, el V-Model resulta más apropiado para proyectos que combinan hardware y software, donde la detección temprana de errores y el cumplimiento de requisitos funcionales y no funcionales es fundamental.

4.1. Fases del Modelo V-Model Aplicadas al Proyecto

El modelo V-Model vincula cada etapa del desarrollo con una fase de pruebas correspondiente. Esta estructura permite asegurar que todos los componentes y funcionalidades sean verificados de forma continua, alineando el diseño, la implementación y la validación del enchufe inteligente con los objetivos del proyecto.

4.1.1. Definición de Requisitos

Durante esta fase, se identifican los recursos físicos y de software necesarios para el cumplimiento de los objetivos funcionales del sistema. Esta identificación temprana garantiza la disponibilidad de los componentes esenciales desde el inicio del proyecto, minimizando posibles retrasos.

- **Componentes físicos:**
    - **Wemos D1 Mini:** Microcontrolador utilizado como unidad principal de procesamiento y comunicación inalámbrica. Su elección se basa en su tamaño compacto, bajo consumo energético y compatibilidad con redes Wi-Fi, lo que lo convierte en una solución adecuada para proyectos de automatización en espacios reducidos (Hasan et al., 2020; Ristik et al., 2021).

- **Sensor DHT22:** Sensor digital encargado de medir la temperatura y la humedad relativa del entorno. Se selecciona por su amplio rango de medición (−40 °C a 80 °C para temperatura y 0 % a 100 % para humedad), su precisión de ±0.5 °C y su fácil integración con microcontroladores.
- **Sensor LDR:** Componente fotosensible que varía su resistencia según la cantidad de luz que percibe. Su uso se justifica por su bajo costo, facilidad de conexión y utilidad en aplicaciones de control automático de iluminación basada en la luminosidad ambiental.
- **Relé SSR-40DA:** Relé de estado sólido empleado para el control de la alimentación eléctrica de los dispositivos conectados. Se selecciona por su capacidad de operar con cargas de corriente alterna de hasta 40 A, su funcionamiento silencioso y su resistencia mecánica, características esenciales para aplicaciones domóticas seguras y duraderas.
- **Fuente de alimentación HLK-PM01:** Módulo que convierte la tensión alterna de la red eléctrica (220–240 V AC) en corriente continua de 5 V DC, necesaria para alimentar el sistema. Se opta por este modelo debido a su tamaño reducido, eficiencia energética y cumplimiento de normas de seguridad eléctrica.

- **Herramientas de desarrollo:**
  - **Arduino IDE:** Plataforma que se utiliza para programar el microcontrolador Wemos D1 Mini. Se selecciona por su entorno simplificado, su compatibilidad con una amplia gama de placas de desarrollo y la disponibilidad de bibliotecas para sensores, actuadores y comunicación mediante protocolos como MQTT. Además, permite cargar el código directamente al microcontrolador, facilitando el proceso de desarrollo y prueba del sistema (Ristik et al., 2021).
  - **Android Studio:** Entorno de desarrollo que se utiliza para diseñar y programar la aplicación móvil. Se elige por su capacidad para construir interfaces gráficas intuitivas, integrar bibliotecas externas y emular dispositivos Android, lo que facilita la validación de funcionalidades antes de su despliegue final (Hardy, 2019).
- **Protocolo y plataforma de comunicación:**

- **MQTT (Message Queuing Telemetry Transport):** Protocolo de mensajería ligera diseñado específicamente para sistemas IoT. Se selecciona por su eficiencia en el uso del ancho de banda, baja latencia y bajo consumo de energía, lo que lo hace más adecuado que HTTP para entornos con recursos limitados, como microcontroladores. A diferencia de HTTP, que sigue un modelo cliente-servidor, MQTT utiliza un esquema de publicación/suscripción que permite una comunicación más fluida y escalable entre dispositivos (Mistry et al., 2020; Light, 2017).
- **Adafruit IO:** Plataforma en la nube que se utiliza como intermediaria para el envío y recepción de datos entre el sistema físico y la aplicación móvil. Se selecciona por su integración nativa con MQTT, su interfaz visual accesible, y la posibilidad de monitorear y controlar variables del sistema en tiempo real, sin necesidad de implementar un servidor propio.

4.1.2. Diseño del Sistema

En esta etapa se define la arquitectura general del sistema, estableciendo las relaciones entre los componentes físicos y las interfaces digitales. El objetivo principal es organizar la estructura del sistema antes de su implementación, garantizando coherencia entre los elementos de hardware, software y comunicación.

Se consideran los siguientes aspectos:

- **Diagrama eléctrico:** Se elabora un esquema que representa la conexión física entre los sensores (DHT22 y LDR), el microcontrolador (Wemos D1 Mini), el relé SSR-40DA y la fuente de alimentación. Para esta tarea se utiliza el software **KiCad**, herramienta de diseño electrónico de código abierto que permite crear esquemáticos precisos y validados profesionalmente para circuitos electrónicos.
- **Flujo lógico del sistema:** Se diseña el recorrido de los datos desde la captura por sensores, pasando por el procesamiento en el microcontrolador, hasta su transmisión mediante MQTT hacia la nube. Para representar este flujo se emplea **Lucidchart**, plataforma que permite crear diagramas de flujo lógico, arquitectura de sistemas y esquemas de comunicación.

- **Panel de visualización en la nube:** Se planifica la estructura del dashboard en Adafruit IO, plataforma que permite configurar widgets visuales como gráficos, indicadores, botones y controles deslizantes. Esta planificación considera tanto la lectura de variables (temperatura, humedad, luz) como el envío de comandos de encendido o apagado hacia el sistema desde la aplicación móvil.

4.1.3. Desarrollo

En esta fase se implementan los módulos físicos y de software que conforman el sistema completo:

- Programación del microcontrolador Wemos D1 Mini para la lectura de datos desde los sensores DHT22 y LDR, así como el control del relé.
- Configuración de credenciales y temas MQTT para establecer la conexión con la plataforma Adafruit IO.
- Desarrollo de la aplicación móvil en Android Studio, incorporando pantallas de control, visualización de datos en tiempo real y configuración de automatizaciones.
- Integración total del hardware con el software, formando un sistema funcional que permite el control automatizado del enchufe inteligente.

4.1.4. Pruebas Unitarias

Cada componente del sistema es probado de forma individual para verificar su correcto funcionamiento:

Con el fin de asegurar la operatividad individual de cada componente antes de su integración al sistema completo, se realizan **pruebas unitarias** orientadas a validar el correcto funcionamiento de sensores, actuadores, comunicación inalámbrica y la aplicación móvil.

- **Sensores ambientales (DHT22 y LDR):**

  Se evalúa la capacidad de los sensores para entregar mediciones estables y coherentes bajo condiciones controladas. En el caso del DHT22, se exponen variaciones de temperatura y humedad para observar el comportamiento del sensor frente a estímulos térmicos. Para el sensor LDR, se simulan cambios de iluminación ambiental, registrando la respuesta ante incrementos y bloqueos de luz. En ambos

casos, las lecturas se monitorean a través del entorno de desarrollo Arduino IDE y luego se verifican en el servidor Adafruit IO y en la aplicación móvil, confirmando un desempeño consistente y acorde al comportamiento esperado para entornos domésticos. El detalle de estas pruebas se encuentra ampliado en la sección correspondiente al desarrollo del sistema (ver sección 5.4.).

- **Relé SSR-40DA:**

  Se verifica su capacidad de conmutación mediante pruebas físicas con una carga resistiva (lámpara), utilizando una rutina de encendido y apagado controlado desde el microcontrolador. Además, se utiliza un multímetro digital en modo de prueba de continuidad para confirmar la activación o corte del paso de corriente según el estado lógico recibido.

- **Conectividad Wi-Fi:**

  Se implementa un código de prueba que monitorea el proceso de conexión del Wemos D1 Mini a la red local. A lo largo de varias sesiones de prueba, el sistema permanece encendido durante varias horas sin registrar desconexiones, validando así la estabilidad de la conexión. Aunque no se mide formalmente el tiempo de reconexión, se considera que el comportamiento es aceptable para un sistema de monitoreo y control remoto.

- **Aplicación móvil:**

  Se revisa la navegación entre pantallas, el encendido y apagado del enchufe y la visualización de los valores enviados por los sensores. Las pruebas se realizan utilizando un único dispositivo Android conectado a dos redes distintas, sin presentar problemas de conectividad.

Estas pruebas permiten comprobar que los módulos electrónicos, el canal de comunicación y la aplicación móvil funcionan de forma independiente conforme a lo esperado, estableciendo una base sólida para su integración en el sistema completo.

4.1.5. Validación del Sistema

Una vez integrados todos los módulos del sistema, se realizan pruebas funcionales en un entorno controlado para validar el comportamiento global frente a las condiciones establecidas en los requerimientos. La validación es ejecutada por el propio desarrollador

del proyecto, quien evalúa el funcionamiento del sistema desde la perspectiva del usuario final, sin la participación directa de terceros.

Las pruebas consideran los siguientes aspectos:

- **Simulación de condiciones ambientales:** Se modifican manualmente las condiciones de luz y temperatura para verificar la reacción del sistema ante los umbrales definidos. Se observa una respuesta correcta del enchufe ante cambios de luminosidad captados por el sensor LDR y de temperatura captados por el DHT22
- **Visualización de datos en la aplicación móvil:** Se valida la correcta recepción y actualización de las variables ambientales en la interfaz de usuario. La sincronización entre la aplicación y la plataforma Adafruit IO se mantiene estable, permitiendo al usuario monitorear en tiempo real el estado de los sensores y el relé.
- **Control del enchufe:** Se evalúa la respuesta del sistema ante comandos enviados desde la aplicación móvil, tanto en modo manual (botón de encendido/apagado) como automático (activación por sensor de luz y temperatura).

Capítulo 5

5. Desarrollo del enchufe

El desarrollo del sistema comienza con la verificación de la compatibilidad entre los distintos componentes seleccionados, incluyendo el microcontrolador Wemos D1 Mini, los sensores DHT22 y LDR, el relé de estado sólido SSR-40DA, la fuente de alimentación y los elementos eléctricos (enchufe macho y hembra).

Componentes del circuito eléctrico

A continuación, se describen los principales componentes utilizados en el armado del circuito eléctrico del enchufe inteligente:

- SSR-40DA: Relé de estado sólido con aislamiento óptico, capaz de controlar cargas de hasta 40 A en corriente alterna (AC). Su diseño elimina partes móviles, ofreciendo mayor durabilidad y funcionamiento silencioso. Este componente permite o interrumpe el paso de corriente hacia el enchufe hembra según las órdenes digitales enviadas por el microcontrolador (HandsOn Technology, s.f.). (Ver Figura 6)
- Socket hembra: Conector donde se enchufan los dispositivos eléctricos que serán controlados por el sistema. (Ver Figura 7)
- IRM-03-5: **IRM-03-5:** Fuente de alimentación encapsulada de tipo conmutada, diseñada para aplicaciones electrónicas que requieren un suministro estable desde la red eléctrica. Convierte una entrada de **85–264V AC** en una salida de **5V DC** con una corriente máxima de **600 mA** (3W de potencia). Este módulo destaca por su tamaño compacto, eficiencia energética, protección contra sobrecarga y sobrevoltaje, lo que lo hace ideal para dispositivos IoT alimentados directamente desde la red eléctrica domiciliaria (MEAN WELL, 2019). (Ver Figura 8)
- Cables de corriente: Conductores que permiten la conexión segura entre la red eléctrica del hogar y el circuito interno del enchufe. (Ver Figura 9)
- Conector macho: Elemento que permite conectar el enchufe inteligente directamente a la red eléctrica del hogar. (Ver Figura 10)

Figura 6: SSR-40DA

Figura 7: Socket hembra

Figura 8: IRM-03-5

Figura 9: cables de corriente

Figura 10: Conector macho

Para lograr la automatización del enchufe inteligente y su correcta interacción con el entorno, se incorporan los siguientes componentes electrónicos:

- **Wemos D1 Mini**: Microcontrolador basado en el chip ESP8266, encargado de ejecutar la lógica del sistema, controlar el relé y establecer comunicación Wi-Fi mediante el protocolo MQTT con la nube (Adafruit IO). Su tamaño compacto, bajo consumo y amplia comunidad de soporte lo convierten en una opción ideal para soluciones de automatización doméstica (Hasan et al., 2020). (Ver Figura 11)
- **Sensor DHT22**: Sensor digital que entrega mediciones de temperatura (rango de −40 °C a 80 °C con una precisión de ±0.5 °C) y humedad relativa (0 % a 100 %, ±2 % de precisión). Su salida digital permite una fácil integración con microcontroladores, siendo adecuado para monitoreo ambiental en contextos residenciales (Components101, s.f.-b). (Ver Figura 12)
- **Sensor LDR**: Fotorresistor cuya resistencia interna disminuye al aumentar la cantidad de luz incidente. Este cambio permite cuantificar los niveles de luminosidad del entorno mediante una lectura analógica. Es utilizado para activar o desactivar el enchufe según las condiciones de iluminación, como la llegada de la noche o el encendido de una lámpara ambiente (Components101, s.f.-a). (Ver Figura 13)

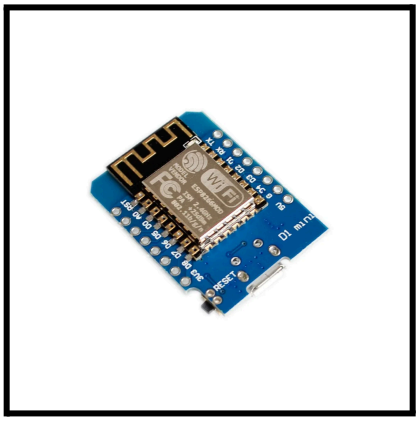


Figura 11: wemos d1 mini

Figura 12: DHT22

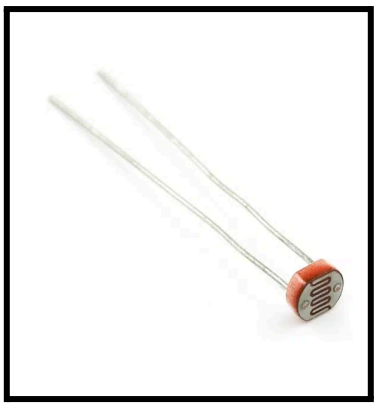


Figura 13: LDR

Componentes para el orden y ensamblaje del sistema

Con el objetivo de mantener una organización estructurada, segura y funcional en el montaje del enchufe inteligente, se utilizan los siguientes elementos complementarios:

- **Caja de corriente IP66**: Estructura de protección que alberga todos los componentes electrónicos del sistema. Su nivel de protección IP66 garantiza aislamiento frente a polvo y salpicaduras, asegurando durabilidad y seguridad en ambientes interiores. Además, facilita una distribución ordenada de los elementos internos. (Ver Figura 14)
- **Jumpers**: Cables flexibles utilizados para realizar las conexiones entre los pines del microcontrolador y los distintos sensores y módulos electrónicos. Su uso permite conexiones limpias y fácilmente modificables durante el proceso de pruebas y ajustes del sistema. (Ver Figura 15)
- **Divisor de cables**: Componente utilizado para organizar los cables de corriente dentro de la caja, evitando enredos y facilitando tanto el mantenimiento como la identificación de cada conexión. Su aplicación mejora la seguridad del sistema y reduce el riesgo de cortocircuitos por contacto accidental. (Ver Figura 16)

Figura 14: Caja de corriente

Figura 15: Jumpers

Figura 16: Divisor de cables

5.1. Preparación de ensamblaje del enchufe

Se diseña un diagrama eléctrico mediante la plataforma **KiCad**, con el objetivo de representar gráficamente las conexiones entre los distintos componentes que conforman el enchufe inteligente. Este esquema, ver Figura 17, permite verificar la coherencia del diseño, anticipar errores de conexión y facilitar el proceso de implementación (KiCad, s.f.).

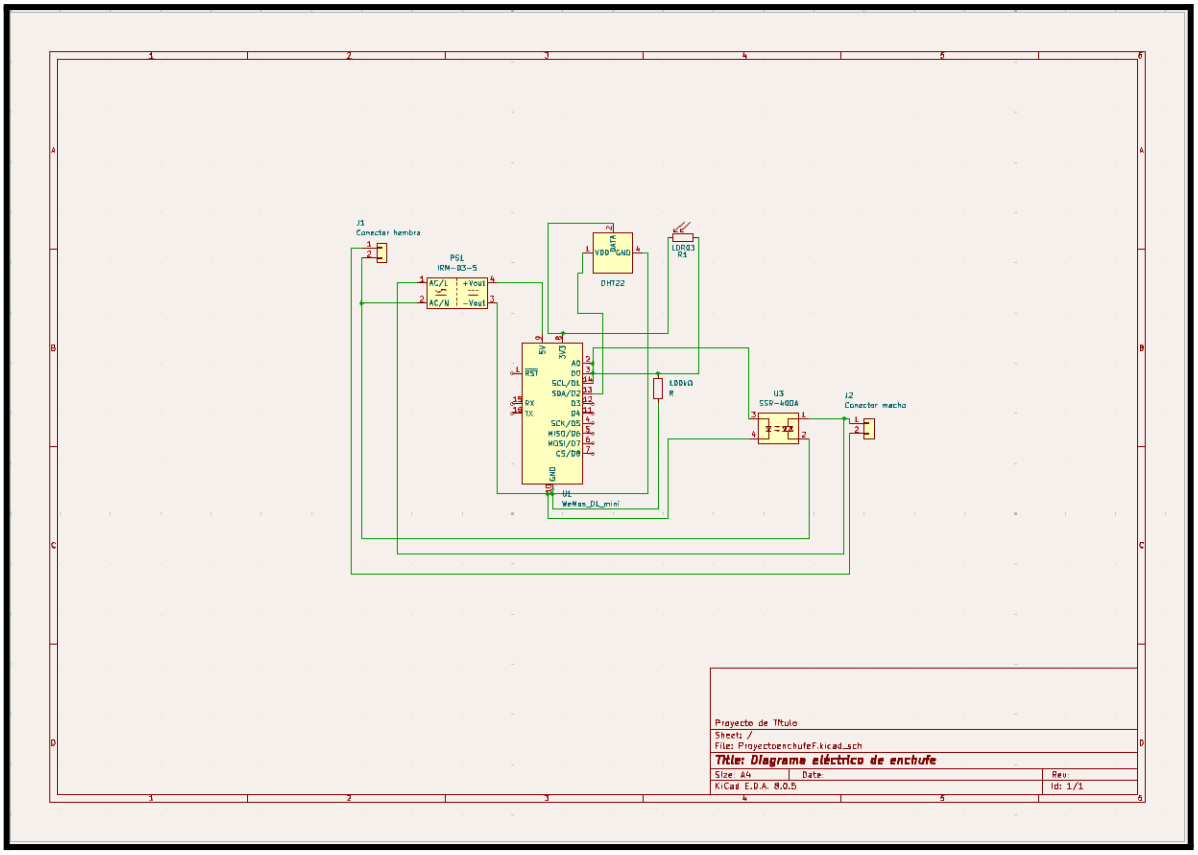


Figura 17: Diagrama eléctrico (elaborado en KiCad).

5.1.1. Componentes principales del circuito

El circuito contempla la conexión del microcontrolador con los sensores ambientales, el relé de estado sólido, la fuente de alimentación IRM-03-5, y los conectores hembra y macho. Estos elementos se interconectan conforme al diseño previamente definido, permitiendo la lectura de datos, su procesamiento y la ejecución de acciones automáticas o manuales a través de la aplicación móvil.

El diseño se planifica bajo una arquitectura compacta y segura, optimizada para ser montada dentro de una caja de protección IP66, con el fin de aislar los componentes expuestos a tensiones peligrosas y garantizar la seguridad del usuario. Asimismo, se considera el uso de conductores aislados, distribución ordenada de cables y separación física entre las zonas de control y potencia.

5.2. Ensamblado eléctrico

El ensamblaje del circuito eléctrico se realiza considerando las características técnicas y de seguridad propias de cada componente involucrado en el sistema. Esta etapa es esencial, ya que establece la conexión funcional entre los elementos que componen el enchufe inteligente, asegurando una operación estable y segura.

Para la conexión de los cables de corriente, se identifican claramente los conductores de fase, neutro y tierra, los cuales se manipulan respetando los estándares de seguridad eléctrica. Se emplean terminales adecuados y aislación térmica para prevenir cortocircuitos y minimizar el riesgo de contacto accidental.

En el caso del enchufe macho (conector a la red domiciliaria de 220V AC) y del socket hembra (salida de conexión para dispositivos eléctricos), se sigue el mismo criterio de instalación, asegurando que cada conductor esté correctamente conectado a su polo correspondiente.

La fuente de alimentación IRM-03-5, ilustrada en la **Figura 19**, convierte la tensión alterna de 220V AC en una tensión continua de 5V DC. Esta salida es utilizada para alimentar el microcontrolador Wemos D1 Mini y los sensores del sistema, garantizando una operación estable de los componentes electrónicos.

Por su parte, el relé SSR-40DA, mostrado en la **Figura 20**, se conecta directamente en la línea de fase. Su función es habilitar o interrumpir el paso de corriente hacia el socket, en función de la señal enviada desde el microcontrolador, permitiendo así el control automatizado o remoto de los dispositivos conectados.

La **Figura 18** muestra el ensamblaje preliminar del circuito eléctrico, en el que los componentes se encuentran conectados entre sí, validando su correcto funcionamiento eléctrico antes de ser montados dentro de la caja de protección. Esta fase permite realizar pruebas de continuidad, identificar posibles errores de conexión y asegurar que todos los elementos respondan correctamente a las señales del microcontrolador.

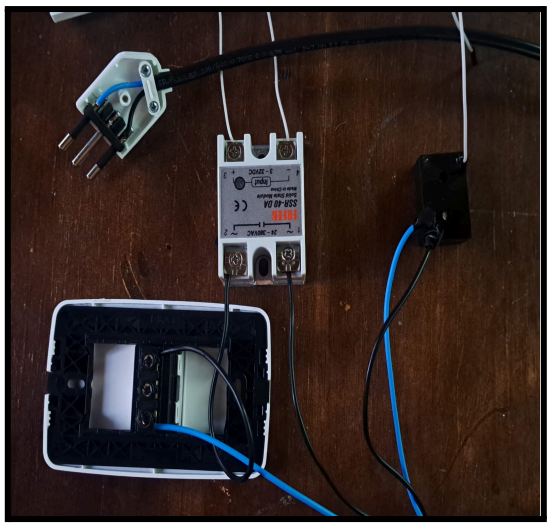


Figura 18: Componentes de circuito eléctrico

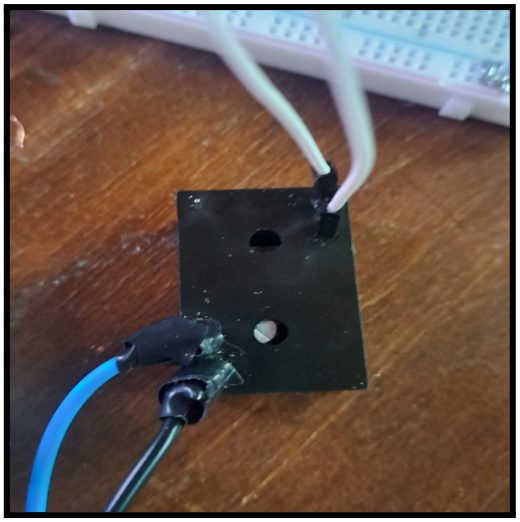


Figura 19: Fuente de alimentación

Figura 20: SSR-40DA

5.3. Preparación y Ensamblaje de Sensores

Se lleva a cabo el ensamblaje y prueba individual de los sensores que forman parte del enchufe inteligente. Esta fase permite validar el funcionamiento de cada componente de forma aislada, identificar posibles defectos de fabricación y asegurar su correcta integración en el circuito final.

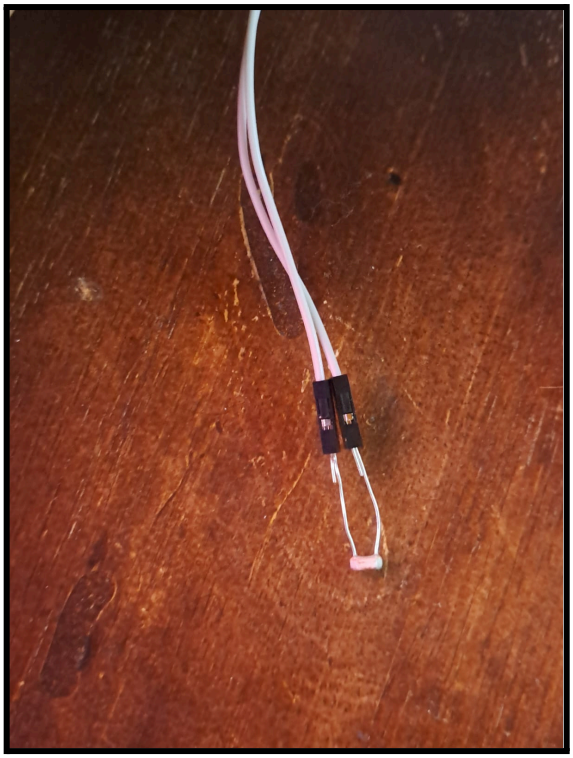


Figura 21: LDR

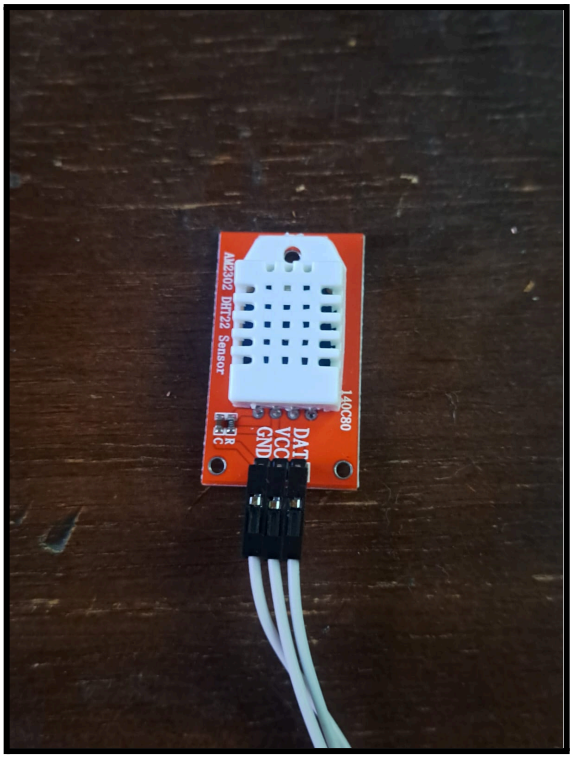


Figura 22: DHT22

Los sensores utilizados en el proyecto son los siguientes:

- **Sensor DHT22:** Sensor digital de temperatura y humedad. Se conecta directamente al microcontrolador a través de una de sus entradas digitales, utilizando una resistencia de 10kΩ entre VCC y DATA para estabilizar la señal. Este sensor permite obtener lecturas precisas que posteriormente se utilizan tanto para la visualización como para el control automático del enchufe. (Ver Figura 22)
- **Sensor LDR (fotoresistencia):** Se conecta en conjunto con una resistencia fija para conformar un divisor de voltaje, permitiendo medir el nivel de luminosidad ambiental. La señal analógica resultante se interpreta en el microcontrolador a través de una entrada ADC, y se emplea como criterio de automatización en función del umbral definido por el usuario. (Ver Figura 21)

Una vez soldados y probados de manera individual, los sensores se integran al circuito principal, manteniendo una disposición que facilite tanto su lectura como su montaje físico dentro de la carcasa protectora.

5.4. Pruebas de Funcionamiento

Antes de integrar los componentes en su configuración final, se realiza una etapa de pruebas individuales con el objetivo de verificar el correcto funcionamiento del circuito eléctrico y los sensores utilizados en el proyecto.

5.4.1. Verificación del Circuito Eléctrico

Para validar las conexiones del circuito eléctrico del enchufe inteligente, se utiliza un multímetro digital como herramienta principal de diagnóstico. Este instrumento permite realizar pruebas de continuidad, con el objetivo de asegurar que no existan interrupciones ni falsos contactos entre los distintos puntos del circuito. Asimismo, se mide la salida de voltaje de la fuente de alimentación IRM-03-5, verificando que entregue de manera estable los 5V DC requeridos por el microcontrolador y los sensores.

Estas mediciones también se aplican a los terminales del relé SSR-40DA y a las líneas de entrada y salida de corriente alterna, con el fin de confirmar la correcta conducción eléctrica y la ausencia de pérdidas. En cada prueba se registra una respuesta estable y dentro de los rangos esperados.

El diseño general del circuito se basa en principios de seguridad eléctrica comunes en sistemas de baja tensión, y todos los componentes utilizados son comercialmente certificados para su uso en aplicaciones domiciliarias. La fuente IRM-03-5 cumple con normativas internacionales como UL60950-1 (Underwriters Laboratories, 2007) y EN60335-1 (International Electrotechnical Commission, 2010), mientras que el relé SSR-40DA incluye aislamiento óptico para proteger el circuito de control frente a la línea de corriente alterna (AC). Además, el sistema es montado dentro de una caja IP66, que ofrece protección contra polvo y salpicaduras, minimizando riesgos de cortocircuitos o descargas accidentales.

El cumplimiento de estas consideraciones contribuye a asegurar que el dispositivo no represente riesgos eléctricos para el usuario ni para su entorno, cumpliendo con los estándares básicos de seguridad recomendados para prototipos funcionales de bajo voltaje en entornos residenciales.

5.4.2. Pruebas de Sensores

En el caso del sensor DHT22, se monitorean las variaciones de temperatura y humedad en un entorno controlado. Las pruebas se realizan dentro de una habitación cerrada, exponiendo el sensor a diferentes fuentes de calor como el cuerpo humano y lámparas. Se observa cómo la lectura de temperatura se incrementa gradualmente, reflejando correctamente los cambios en el entorno. Este comportamiento permite validar tanto la sensibilidad térmica del sensor como la estabilidad de las mediciones entregadas.

Respecto al sensor LDR, se simulan condiciones de iluminación variables utilizando linternas de diferentes potencias y bloqueos parciales de luz directa. Se observa cómo el valor analógico leído varía en función de la luminosidad ambiental, con valores promedio que oscilan entre 150 y 200 lux bajo condiciones normales de iluminación. Estas pruebas permiten confirmar el correcto funcionamiento del divisor de voltaje en el que está conectado y la respuesta eficiente del sensor ante los cambios de luz.

Como parte del proceso de ajuste, se determina un umbral de activación específico para este sensor. Dicha referencia se establece a partir de pruebas realizadas en una habitación de 3,3 m x 2,5 m, evaluando las condiciones reales de uso del enchufe inteligente. A partir de estas mediciones y observaciones personales, se define el umbral mínimo de luminosidad a partir del cual el sistema activa el enchufe de forma automática, adaptándose al entorno operativo.

Ambos sensores presentan un comportamiento estable, predecible y funcional, cumpliendo con los requerimientos para ser integrados al sistema domótico. Las pruebas permiten asegurar que las lecturas entregadas pueden ser utilizadas en tiempo real para tomar decisiones de automatización confiables.

5.5. Ensamblaje Físico del Enchufe

Con el diagrama eléctrico validado y los componentes probados individualmente, se procede al ensamblaje físico completo del enchufe inteligente. Esta etapa contempla la disposición de los elementos dentro de una caja de protección eléctrica, la organización del cableado y la integración definitiva del sistema.

5.5.1. Criterios de ensamblaje

- **Seguridad eléctrica:** Se aíslan cuidadosamente las conexiones de corriente alterna mediante terminales y cinta termo-retráctil, evitando cualquier contacto accidental con las líneas de 220V.
- **Distribución interna:** Los componentes se posicionan de manera que permitan una ventilación adecuada y un fácil acceso para futuras mantenciones. El relé SSR-40DA se fija a una superficie firme para evitar movimientos que puedan comprometer la estabilidad de la conexión.
- **Montaje de sensores:** El sensor DHT22 se ubica en un lateral de la carcasa, permitiendo una exposición directa al ambiente para una lectura precisa. Por su parte, el LDR se posiciona en un punto expuesto a la luz ambiente, evitando obstrucciones que afecten la medición.
- **Conectores de entrada y salida:** El enchufe macho (entrada) y el enchufe hembra (salida controlada) se integran en la carcasa, permitiendo que el sistema sea instalado directamente entre una fuente de energía y cualquier electrodoméstico de uso común.

5.5.2. Proceso de Ensamblaje Físico

El ensamblaje físico del enchufe inteligente se realiza de manera progresiva dentro de una caja de protección eléctrica, incorporando cada componente según la distribución definida en el diseño. A continuación, se detallan las etapas del proceso, junto con su respectiva documentación fotográfica.

- Fase 1: Montaje del microcontrolador y el circuito eléctrico

Se inicia el proceso posicionando el circuito eléctrico y la placa **Wemos D1 Mini** sobre una **protoboard**, ubicada al centro de la caja. Este microcontrolador actúa como el núcleo del sistema, desde donde se gestionan las lecturas de sensores, la ejecución de automatizaciones y el control del relé.

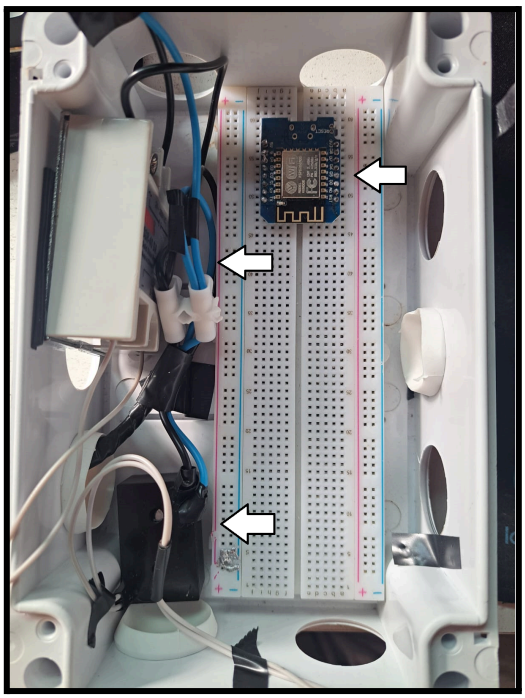


Figura 23: Fase 1 de la conexión

En esta fase de la conexión corresponde al posicionamiento de cada una de las piezas, considerando el espacio a ocupar y la distribución correcta del cableado que corresponde al circuito eléctrico, las flechas indicadas en la Figura 23 corresponden a cada componente que se debe de tener en consideración a la hora de gestionar el espacio de la caja IP66.

- Fase 2: Conexión del relé SSR-40DA

El relé de estado sólido **SSR-40DA** se conecta al Wemos D1 Mini para controlar el flujo de corriente hacia la salida. Este componente permite cortar o restablecer el paso eléctrico sin partes mecánicas, aportando durabilidad y fiabilidad. Viendo la Figura 24 podemos apreciar las flechas que indican las conexiones del relé con la placa Wemos D1 mini y las conexiones de los pines se muestran en la Tabla 1.

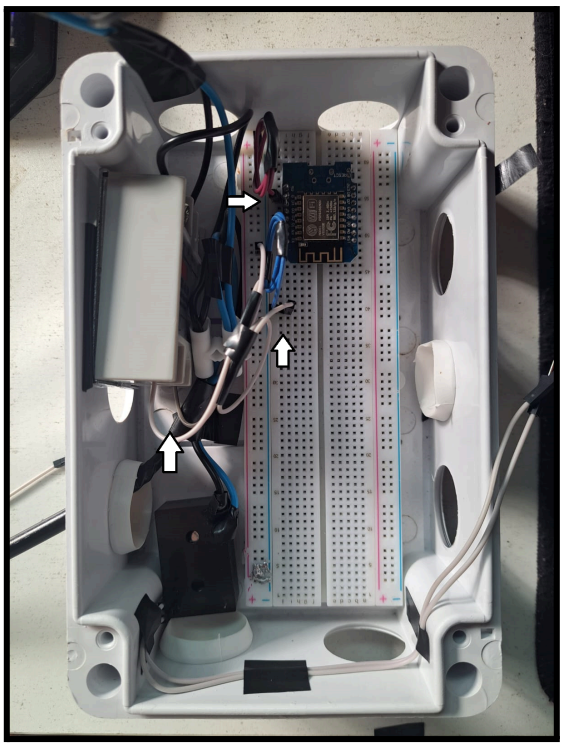


Figura 24: Fase 2 de la conexión

- Tabla de conexiones

SSR-40DA	Wemos d1 mini
SALIDA 3 (+)	D1
SALIDA 4 (-)	GND

Tabla 1: Conexiones eléctricas Fase 2

- Fase 3: Instalación de la fuente de alimentación IRM-03-5

La fuente **IRM-03-5** convierte los 220V AC de entrada en una salida de 5V DC estabilizada. Se fija firmemente dentro de la caja y sus conexiones se fijan con una soldadura de cautín y se aíslan utilizando cinta termo-retráctil para garantizar la seguridad eléctrica. En la Figura 25 se aprecian las flechas que apuntan a las conexiones nuevas que se realizaron entre la wemos D1 mini y la fuente de alimentación y las conexiones de los pines se muestran en la Tabla 2.

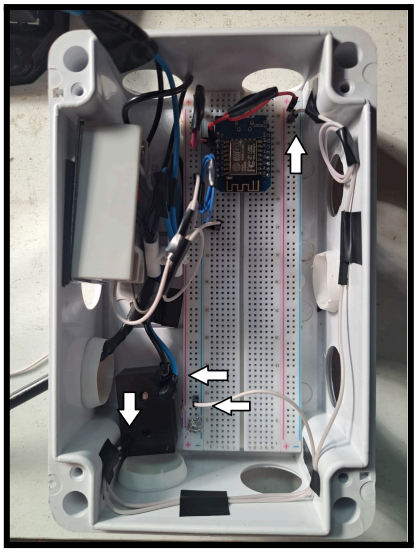


Figura 25: Fase 3 de la conexión

- Tabla de conexiones

IRM-03-5	SSR-40DA	Wemos d1 mini
-	SALIDA 3 (+)	D1
AC (-)	SALIDA 4 (-)	GND
AC (+)		5V

Tabla 2: Conexiones eléctricas Fase 3

- Fase 4: Integración del sensor de luz (LDR)

Se conecta el **sensor LDR** al pin analógico A0 del Wemos mediante un divisor de voltaje, permitiendo medir la luminosidad ambiente. Este sensor habilita el control automático del enchufe en función de la intensidad de luz detectada. La flechas indicadas en la Figura 26 muestra las conexiones nuevas entre el sensor LDR y la Wemos D1 mini y las conexiones de los pines se muestran en la Tabla 3.

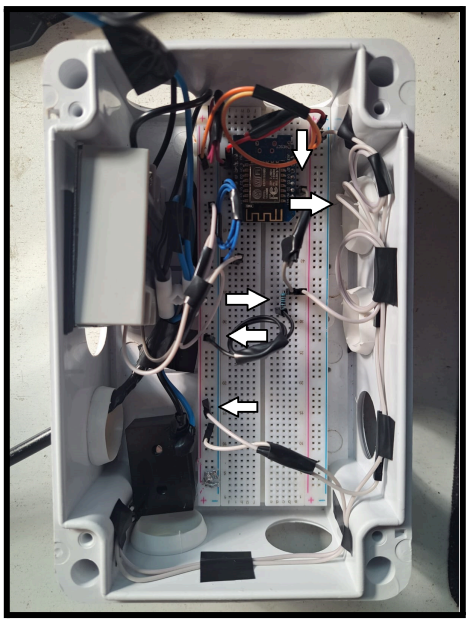


Figura 26: Fase 4 de la conexión

- Tabla conexiones

LDR	IRM-03-5	SSR-40DA	Wemos d1 mini
-	-	SALIDA 3 (+)	D1
-	AC (-)	SALIDA 4 (-)	GND
-	AC (+)	-	5V
SALIDA 1	-	-	A0
SALIDA 2	-	-	3.3V

Tabla 3: Conexiones eléctricas Fase 4

- Fase 5: Conexión del sensor DHT22

Se procede a la integración del sensor DHT22, responsable de medir la temperatura y la humedad relativa del entorno. Este se instala en uno de los laterales de la caja de protección, de forma que quede expuesto al ambiente exterior para garantizar lecturas precisas y representativas. La conexión del sensor se realiza al microcontrolador Wemos D1 Mini, utilizando el pin D2 para la señal de datos (DATA), 3.3V para la alimentación y GND para tierra. La sistematización completa de los componentes utilizados en la

construcción del enchufe inteligente, incluyendo la disposición del sensor DHT22, se muestra en la Figura 27, mientras que todas las conexiones realizadas durante el proceso de ensamblaje se detallan en la Tabla 4.

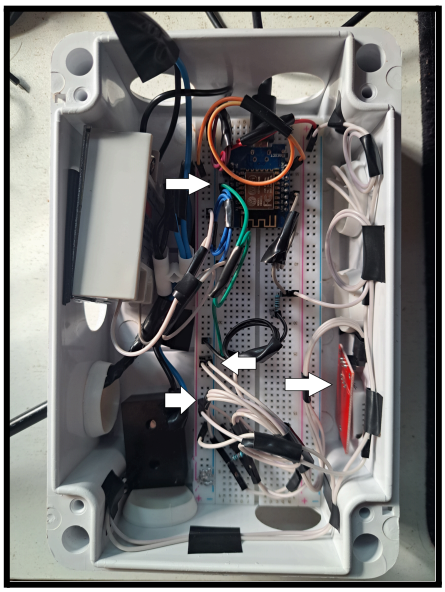


Figura 27: Fase 5 de la conexión

- Tabla de conexiones

DHT22	LDR	IRM-03-5	SSR-40DA	Wemos d1 mini
-	-	-	SALIDA 3 (+)	D1
GND	-	AC (-)	SALIDA 4 (-)	GND
-	-	AC (+)	-	5V
-	SALIDA 1	-	-	A0
VCC	SALIDA 2	-	-	3.3V
DATA	-	-	-	D2

Tabla 4: Conexiones eléctricas Fase 5

- Fase 6: Enchufe final

Se finaliza el montaje con la ubicación correcta del enchufe hembra (salida controlada) y el enchufe macho (entrada a la red eléctrica domiciliaria). Ambos elementos

se fijan en la carcasa mediante **prensacables**, lo que proporciona estabilidad estructural y seguridad en la conexión. El diseño contempla **compatibilidad universal** y una adecuada **protección ambiental** gracias al uso de una **caja estanca con certificación IP65**, la cual garantiza una protección total contra el ingreso de polvo y contra chorros de agua proyectados desde cualquier dirección, permitiendo su utilización en entornos expuestos sin comprometer el funcionamiento interno del sistema. La disposición final de estos elementos puede observarse en la **Figura 28**, donde se muestra el montaje completo del dispositivo en su carcasa.

Figura 28: Fase final del enchufe

Todas las conexiones se aseguran utilizando cables flexibles, conectores de terminal y cinta aislante, garantizando un ensamblaje ordenado, seguro y funcional. Este proceso concluye con una verificación general del circuito para confirmar la correcta alimentación del sistema y el funcionamiento de cada componente.

Este diseño físico final refleja un equilibrio entre funcionalidad, seguridad y portabilidad. El sistema es autónomo, puede trasladarse fácilmente y se encuentra listo para ser utilizado como dispositivo de automatización y monitoreo de sensores en tiempo real.

5.6. Tabla de costos y comparativa con productos del mercado

El análisis de costos presentado en esta sección corresponde exclusivamente al prototipo funcional desarrollado durante este proyecto. Es importante considerar que los

valores indicados no incluyen márgenes de ganancia, costos asociados a garantía, procesos de fabricación a escala, ni gastos en marketing o mano de obra especializada, los cuales sí están contemplados en los precios de productos comerciales.

El costo total del prototipo asciende a $39.420 CLP (aproximadamente 42 USD), como se detalla en la Tabla 5. Este monto contempla todos los componentes físicos necesarios para el funcionamiento del sistema, incluyendo microcontrolador, sensores, relé, fuente de alimentación, carcasa de protección y elementos complementarios de conexión y montaje.

A modo de referencia, se realiza una búsqueda de mercado en mayo de 2025 (ver Tabla 6), identificando que los precios de enchufes inteligentes comercializados en Chile oscilan entre $9.990 y $29.990 CLP (aproximadamente entre 11 y 32 USD). No obstante, estos dispositivos suelen ofrecer funcionalidades limitadas, enfocadas principalmente en el encendido y apagado remoto o la programación horaria mediante aplicaciones genéricas.

El prototipo desarrollado, en cambio, incorpora características avanzadas como:

- Control dual desde una aplicación móvil personalizada (cuenta usuario y cuenta cuidador).
- Automatización basada en condiciones ambientales reales, utilizando sensores de temperatura, humedad y luz.
- Orientación asistencial hacia personas mayores o con movilidad reducida.
- Potencial de escalabilidad y adaptabilidad a nuevos entornos de uso.

Por lo tanto, si bien el costo del prototipo puede resultar superior a ciertas opciones comerciales, se justifica ampliamente por su nivel de personalización, funcionalidad ampliada e impacto social como herramienta de asistencia domiciliaria adaptada a usuarios con necesidades específicas.

Componente	Valor
Wemos D1 Mini	$ 4.990
DHT22	$ 6.290

LDR	$ 1.190
IRM-03-5	$ 4.990
Protoboard	$ 1.000
Jumpers (30 u)	$ 1.000
Cableado de corriente (3 m)	$ 4.990
Caja IP65	$ 8.990
Conector macho	$ 1.990
Conector hembra	$ 2.990
Divisor de cables	$ 1.000
Total	$ 39.420

Tabla 5: Tabla de costos para la construcción del enchufe

Característica / Modelo	Enchufes comerciales (AliExpress, Sodimac, etc.)	Prototipo desarrollado
Precio promedio	$9.990 - $29.990 CLP	$25.000 - $39.420 CLP
Encendido/apagado remoto	Sí	Sí
Automatización por temporizador	Sí (en algunos modelos)	Sí
Automatización por temperatura	No	Sí
Automatización por luz ambiental	No	Sí
Medición de humedad	No	Sí
Aplicación móvil personalizada	No (usan apps genéricas como Smart Life)	Sí (desarrollada en Kotlin)
Diseño orientado a asistencia domiciliaria	No	Sí
Modo cuidador con control remoto completo	No	Sí

Uso de sensores físicos (DHT22, LDR)	No	Sí
Compatibilidad expandible a nuevos sensores	No	Sí

Tabla 6: Tabla comparativa con productos del mercado

5.7. Descripción general del sistema

El sistema desarrollado consiste en un enchufe inteligente diseñado para automatizar el encendido y apagado de dispositivos eléctricos en función de condiciones ambientales y controles remotos definidos por el usuario. Su propósito principal es facilitar la gestión energética y mejorar la autonomía en el hogar, especialmente para personas mayores o con movilidad reducida.

La solución integra sensores de temperatura, humedad y luminosidad (DHT22 y LDR), un microcontrolador Wemos D1 Mini con conectividad Wi-Fi, y un relé de estado sólido que regula el paso de corriente. El sistema puede funcionar de forma manual o automática, según tres modos: por temperatura, por luz ambiental o por temporizador. Solo uno de estos modos puede estar activo a la vez, evitando conflictos de operación.

La comunicación entre el microcontrolador y la aplicación móvil se realiza a través del protocolo MQTT, utilizando la plataforma Adafruit IO como intermediario en la nube. La aplicación, desarrollada en Android Studio, permite visualizar los datos en tiempo real, configurar umbrales y controlar el estado del enchufe desde cualquier ubicación. Cuenta con dos perfiles de acceso: usuario común y cuidador, promoviendo una experiencia adaptada a distintos niveles de alfabetización digital.

Todo el sistema está contenido en una caja de protección eléctrica que garantiza seguridad y durabilidad. Esta arquitectura modular y adaptable permite futuras mejoras, como la integración de nuevos sensores o funcionalidades adicionales.

5.7.1. Validación del sistema completo

- Una vez integrados todos los componentes, se lleva a cabo la validación funcional del sistema completo en un entorno controlado, con el objetivo de confirmar que la

interacción entre sensores, microcontrolador, relé, plataforma en la nube y aplicación móvil se produce de forma coherente, estable y segura.

- Durante la validación, se simulan distintas condiciones ambientales para evaluar la respuesta automática del sistema. Se modifican la temperatura, la luminosidad y los umbrales de tiempo para comprobar que el enchufe se activa o desactiva correctamente según el modo seleccionado (control por luz, temperatura o temporizador). Las pruebas permiten verificar que el sistema reconoce los valores de entrada y ejecuta la acción correspondiente.

- Asimismo, se valida la visualización de datos en la aplicación móvil, comprobando que las lecturas de los sensores se actualizan en tiempo real y coinciden con los valores recibidos desde Adafruit IO. La interfaz permite activar el enchufe de forma remota, y la ejecución de comandos presenta una respuesta inmediata.

- La validación se realiza directamente por el desarrollador del proyecto, utilizando un teléfono móvil con Android, conectado a dos redes distintas: Wi-Fi doméstica y red compartida mediante otro dispositivo. El sistema mantiene la conexión estable en ambos casos y responde sin interrupciones.

- En conjunto, las pruebas de validación permiten confirmar que el sistema funciona como una solución operativa y funcional en condiciones reales, y que es capaz de cumplir con sus objetivos de monitoreo ambiental, automatización y control remoto. Las limitaciones identificadas quedan registradas para su corrección en futuras versiones del sistema.

5.8. Componentes descartados durante el desarrollo

Durante el proceso de desarrollo se utilizaron distintos componentes que, si bien formaron parte de las etapas iniciales de pruebas, fueron posteriormente descartados por motivos de tamaño, eficiencia, compatibilidad o enfoque funcional. A continuación, se presentan los principales elementos retirados del diseño final

- Arduino Uno: Usado en pruebas iniciales, fue reemplazado por el Wemos D1 Mini debido a su falta de conectividad Wi-Fi y mayor tamaño.

- Wemos D1 (formato largo): Se descarta por su tamaño, que dificulta el montaje dentro de la carcasa.
- NodeMCU ESP8266: Sustituido por el Wemos D1 Mini para mantener un diseño más compacto, uniforme y eficiente.
- Sensor de corriente SCT-013: Se decidió no incluirlo para simplificar el sistema, ya que el proyecto se centró en monitoreo ambiental y asistencia.

Capítulo 6

6. Programación del WEMOS D1 MINI y Servidor IO Adafruit

El presente capítulo describe el código que se implementa en el microcontrolador Wemos D1 Mini, el cual cumple un rol fundamental en la lectura de sensores, el control del relé de estado sólido y la comunicación con la aplicación móvil a través del protocolo MQTT utilizando la plataforma Adafruit IO.

El código se desarrolla en Arduino IDE y se utilizan las librerías adecuadas para el manejo de sensores, conexión Wi-Fi y mensajería MQTT. El programa está organizado en bloques funcionales, los cuales se detallan a continuación.

6.1. Inicialización de bibliotecas y configuración

Durante la etapa de desarrollo, se importan las bibliotecas necesarias para habilitar las funcionalidades centrales del sistema. Estas incluyen el manejo de conexión Wi-Fi, la comunicación con la plataforma **Adafruit IO** mediante el protocolo **MQTT**, y la gestión de los sensores **DHT22** y **LDR**. Estas bibliotecas permiten la lectura de datos ambientales, el control del relé y la sincronización con la aplicación móvil. La **Figura 29** presenta las principales bibliotecas utilizadas.

CodigoWemos.ino

```
1  #include <ESP8266WiFi.h>
2  #include <PubSubClient.h>
3  #include <DHT.h>
4
```

Figura 29: Bibliotecas utilizadas en Arduino IDE

En esta fase, también se definen las **credenciales de red Wi-Fi** y las **claves de acceso a Adafruit IO**, así como los pines del microcontrolador asignados al relé, al sensor DHT22 y al sensor LDR. Estas credenciales permiten la conexión con la red local y el envío de datos a la nube para su posterior visualización y control desde la aplicación móvil.

Cabe destacar que las credenciales utilizadas en esta etapa son **por defecto** y se emplean únicamente con fines de validación funcional del sistema. Su objetivo es verificar la posibilidad de establecer múltiples sesiones y comprobar la sincronización entre el dispositivo físico y la aplicación. Las cuentas fueron creadas directamente desde Android Studio para pruebas básicas.

Si bien no se implementan aún medidas específicas de protección de credenciales, se reconoce que, en una versión futura orientada a producción, será indispensable aplicar buenas prácticas de seguridad, como el almacenamiento externo de credenciales, el uso de variables de entorno o sistemas de autenticación robustos.

6.2. Configuración de red y MQTT

El microcontrolador se conecta a una red Wi-Fi doméstica utilizando las credenciales especificadas en el código. Posteriormente, establece conexión con el broker MQTT provisto por la plataforma **Adafruit IO**, que actúa como intermediario entre el Wemos D1 Mini y la aplicación móvil.

- Para lograr esto, se configuran los siguientes parámetros en el código:
    - Dirección del servidor MQTT: io.adafruit.com
    - Puerto: 1883
    - Usuario y clave de acceso personal (AIO Key) generados desde la cuenta de Adafruit IO.

6.2.1. Configuración de Feeds en Adafruit IO

En la plataforma **Adafruit IO**, se deben crear manualmente los feeds o canales que permiten enviar y recibir datos entre el microcontrolador y la aplicación móvil. Cada feed representa una función específica del sistema, como encendido, sensores o control automático.

Los feeds configurados en este proyecto son los siguientes:

- Interfaz con los feeds en IO Adafruit (ver Figura 30)

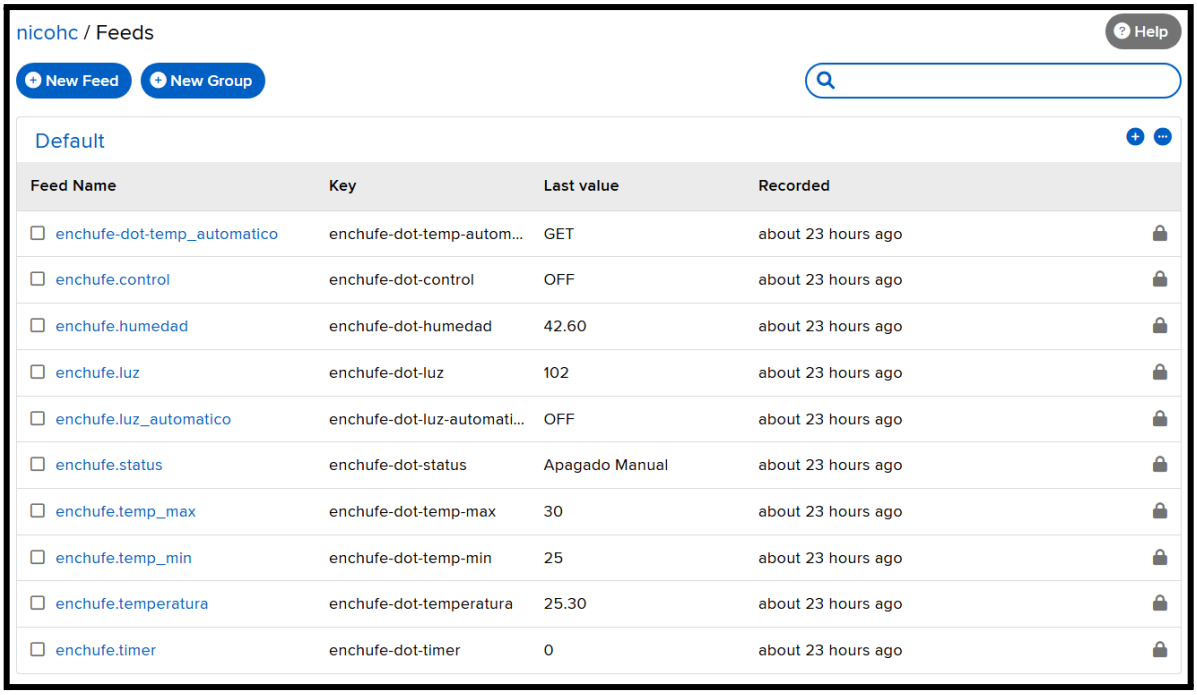


Figura 30: Feeds configurados en la cuenta nicohc dentro de Adafruit IO

- Tabla con los feeds y sus respectivas funciones (ver Tabla 6)

Feed	**Función**
enchufe-dot-temp_automatico	Activa o desactiva el modo automático de temperatura
enchufe.control	Recibe comandos ON/OFF desde la app
enchufe.humedad	Pública la temperatura leída
enchufe.luz	pública el valor del sensor LDR
enchufe.luz_automatico	Activa o desactiva el modo automático de luz
enchufe.status	Pública el estado actual del enchufe
enchufe.temp_max	Define la temperatura máxima de activación
enchufe.temp_min	Define la temperatura mínima de activación
enchufe.temperatura	Pública temperatura leída
enchufe.timer	Define duración del temporizador en minutos

Tabla 7: Tabla de feeds IO Adafruit

6.2.2. Función setEstadoEnchufe()

Esta función encapsula la lógica necesaria para encender o apagar el enchufe de forma segura. Sólo realiza cambios si el estado actual es diferente al solicitado, evitando así operaciones innecesarias.

Además de activar o desactivar el pin del relé, esta función pública el estado actualizado al feed enchufe-dot-status, incluyendo el motivo del cambio, lo que permite que el usuario lo visualice en la app, la función se muestra en la Figura 31.

CodigoWemos.ino

```
49  // ✅ Función para cambiar el estado del enchufe y publicarlo si cambia
50  void setEstadoEnchufe(bool encender, const char* motivo) {
51    if (enchufe_encendido != encender) {
52      enchufe_encendido = encender;
53      digitalWrite(RELAY_PIN, encender ? HIGH : LOW);
54      client.publish(feed_status, motivo, true);
55    }
56  }
```

Figura 31: Feeds configurados en la cuenta nicohc dentro de Adafruit IO

6.2.3. Función mqttCallback(): Recepción de comandos

Esta función actúa como **callback MQTT**, y se ejecuta cada vez que el Wemos recibe un mensaje desde Adafruit IO. Evalúa el tópico y el contenido del mensaje, y actualiza las variables del sistema o ejecuta acciones en consecuencia. (Ver Figura 32)

- Incluye la lógica para:
    - Encendido/apagado manual.
    - Activación/desactivación de control por luz o temperatura.
    - Actualización de umbrales.
    - Inicio del temporizador.

CodigoWemos.ino

```
58  // 📥 Callback de mensajes MQTT
59  void mqttCallback(char* topic, byte* payload, unsigned int length) {
60    String message;
61    for (unsigned int i = 0; i < length; i++) message += (char)payload[i];
62
63    Serial.print("📩 "); Serial.print(topic); Serial.print(": "); Serial.println(message);
64
65    if (String(topic) == feed_control) {
66      if (message == "ON") {
67        timer_activo = false;
68        setEstadoEnchufe(true, "Encendido Manual");
69      } else if (message == "OFF") {
70        timer_activo = false;
71        setEstadoEnchufe(false, "Apagado Manual");
72      }
73    } else if (String(topic) == feed_luz_auto) {
74      control_luz_activado = (message == "ON");
75      if (control_luz_activado) {
76        control_temp_activado = false;
77        timer_activo = false;
78      }
```

Figura 32: Función mqttCallback()

6.2.4. Función setup_wifi(): Conexión Wi-Fi

Función que establece la conexión a la red Wi-Fi utilizando las credenciales configuradas (ver Figura 33). Realiza múltiples intentos hasta lograr la conexión y muestra el estado en el monitor serial para facilitar la depuración.

CodigoWemos.ino

```
101 // Conexión WiFi
102 void setup_wifi() {
103   WiFi.mode(WIFI_STA);
104   WiFi.begin(ssid, password);
105   Serial.print("🔗 Conectando WiFi...");
106   int intentos = 0;
107   while (WiFi.status() != WL_CONNECTED && intentos < 30) {
108     delay(500);
109     Serial.print(".");
110     intentos++;
111   }
112   Serial.println(WiFi.status() == WL_CONNECTED ? "\n✅ Conectado" : "\n❌ No se pudo conectar");
113 }
```

Figura 33: Función setup_wifi()

6.2.5. Función reconnectMQTT(): Reconexión automática

Esta función se reconecta con el broker MQTT si se detecta desconexión (ver Figura 34). En caso de éxito, vuelve a suscribirse a todos los feeds necesarios y publica el

estado actual del enchufe. Esto permite que el sistema recupere la operación automáticamente tras una caída de red o reinicio.

CodigoWemos.ino

```
115  // 📡 Reconexión MQTT
116  void reconnectMQTT() {
117    if (!client.connected()) {
118      Serial.print("🔗 Conectando MQTT...");
119      if (client.connect("WemosD1Mini", mqtt_user, mqtt_pass)) {
120        Serial.println("✅ MQTT Conectado");
121        client.subscribe(feed_control);
122        client.subscribe(feed_luz_auto);
123        client.subscribe(feed_temp_auto);
124        client.subscribe(feed_temp_max);
125        client.subscribe(feed_temp_min);
126        client.subscribe(feed_timer);
127
```

Figura 34: Función reconnectMQTT()

6.2.6. Función loop(): Monitoreo y automatización

La función loop() se ejecuta de forma indefinida y periódica. En ella se realiza el monitoreo constante del entorno, se evalúan las condiciones definidas por el usuario, y se actualiza el estado del enchufe según la lógica del sistema.

Las tareas que cumple este bloque son las siguientes:

6.2.6.1. Reconexión y mantenimiento de MQTT

- Este bloque garantiza que la comunicación MQTT se mantenga activa (ver Figura 35). Si el dispositivo pierde conexión con el broker, la función reconnectMQTT() intenta restablecer. Luego, client.loop() mantiene la sesión viva, procesando cualquier mensaje entrante desde Adafruit IO.

```
148    reconnectMQTT();
149    client.loop();
```

Figura 35: Función cliente.loop()

6.2.6.2. Lectura de sensores

- En este bloque se leen los valores actuales de **temperatura, humedad y luz ambiental**. Si los datos son válidos, se publican en los feeds correspondientes (ver

Figura 36) con el flag retain = true, lo que permite que estos valores estén disponibles aunque el dispositivo o la app se conecten más adelante.

```
155   if (!isnan(temp) && !isnan(hum)) {
156     client.publish(feed_temp, String(temp).c_str(), true);
157     client.publish(feed_humedad, String(hum).c_str(), true);
158     client.publish(feed_luz, String(luz).c_str(), true);
```

Figura 36: Lectura de sensores

6.2.6.3. Lógica de control automático por luz

- Si el modo automático por luz está activado (control_luz_activado = true), el sistema evalúa el valor del sensor LDR (ver Figura 37). Si la luminosidad detectada es inferior al umbral (umbral_luz), el enchufe se enciende. Si supera ese umbral, se apaga. Este comportamiento es útil, por ejemplo, para encender una lámpara automáticamente cuando oscurece.

```
160     if (control_luz_activado) {
161       if (luz < umbral_luz) {
162         setEstadoEnchufe(true, "Encendido por luz");
163       } else {
164         setEstadoEnchufe(false, "Apagado por luz");
165       }
166     }
```

Figura 37: Automatización de sensor de luz

6.2.6.4. Lógica de control automático por temperatura

- Este bloque aplica el mismo principio, pero utilizando como referencia la temperatura ambiental. Si la temperatura medida es igual o superior al umbral máximo, se activa el enchufe (por ejemplo, para encender un ventilador). Si la temperatura desciende al umbral mínimo, el enchufe se apaga (ver Figura 38).

```
168     if (control_temp_activado) {
169       if (temp >= temp_max) {
170         setEstadoEnchufe(true, "Encendido por temperatura");
171       } else if (temp <= temp_min) {
172         setEstadoEnchufe(false, "Apagado por temperatura");
173       }
174     }
175
```

Figura 38: automatización de sensor de temperatura

6.2.6.5. Control por temporizador

- Cuando el temporizador está activo, este bloque calcula cuánto tiempo ha pasado desde que se encendió el enchufe. Si el tiempo transcurrido alcanza el número de minutos definidos por el usuario (timer_minutes), el sistema apaga el enchufe automáticamente y desactiva el temporizador. La lógica mostrada en la Figura 39, permite, por ejemplo, que un dispositivo se mantenga encendido por 15 minutos y luego se apague sin intervención del usuario.

```
176     if (timer_activo && enchufe_encendido) {
177       unsigned long elapsed = (millis() - timer_start) / 60000;
178       if (elapsed >= timer_minutes) {
179         timer_activo = false;
180         setEstadoEnchufe(false, "Apagado por timer");
181       }
182     }
```

Figura 39: Control mediante timer

6.2.6.6. Diagrama de flujo del ciclo principal de funcionamiento

La Figura 40 representa el diagrama de flujo del ciclo principal de ejecución en el sistema del enchufe inteligente, desarrollado en el microcontrolador Wemos D1 Mini. Este ciclo corresponde al contenido de la función loop() en el código fuente y define la lógica continua de lectura de sensores, publicación de datos y decisiones automatizadas según las condiciones establecidas.

- Inicio del ciclo (loop()): El sistema inicia el ciclo principal, en el cual se repiten continuamente las operaciones de control.
- Verificación de conexión MQTT: Se comprueba si el microcontrolador está conectado al servidor MQTT. En caso contrario, se ejecuta la función de reconexión para restablecer la comunicación.
- Comunicación MQTT (cliente.loop()): Si la conexión es exitosa, el sistema mantiene activa la comunicación con el broker de Adafruit IO mediante la función cliente.loop().
- Lectura de sensores: Se realiza la lectura de los datos provenientes del sensor DHT22 (temperatura y humedad) y del sensor LDR (nivel de luz).

- Validación de datos: Si los datos obtenidos no son válidos (por ejemplo, lecturas erráticas o fuera de rango), el sistema reinicia la lectura. En caso de datos válidos, se procede con su publicación.
- Publicación de datos: Los valores medidos son enviados a los feeds correspondientes en la plataforma de Adafruit IO mediante el protocolo MQTT.
- Lógica de automatización:
    - Modo por luz: Si el modo por luz está activado, el sistema evalúa si el valor de luz está por debajo del umbral definido. Si se cumple esta condición, el enchufe se apaga; de lo contrario, se enciende.
    - Modo por temperatura: Si está activado, el sistema verifica si la temperatura es mayor o igual al valor máximo o menor o igual al valor mínimo configurado. En función de ello, el enchufe se enciende o apaga automáticamente.
    - Modo temporizador: En caso de que el temporizador esté activo, se define el tiempo restante y, una vez concluido, el enchufe se apaga automáticamente.

Este flujo asegura que el sistema responda de forma autónoma y eficiente a las condiciones del entorno o a las configuraciones establecidas por el usuario a través de la aplicación móvil, permitiendo un control preciso y flexible del enchufe inteligente.

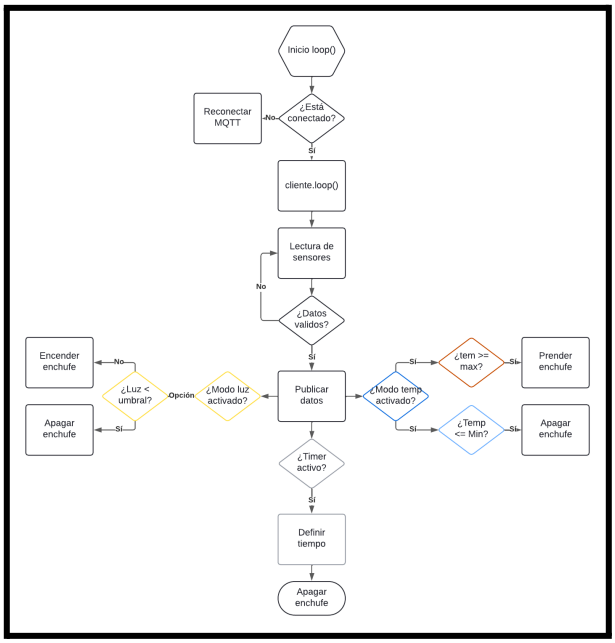


Figura 40: Diagrama de flujos de los sensores

Capítulo 7

7. Desarrollo de la Aplicación Móvil

Este capítulo presenta el proceso de desarrollo de la aplicación móvil que permite al usuario interactuar con el enchufe inteligente. La aplicación fue desarrollada en Kotlin utilizando Jetpack Compose, un framework moderno para construir interfaces de usuario en Android. Además, se implementa comunicación en tiempo real mediante el protocolo MQTT, utilizando la plataforma Adafruit IO como broker.

La aplicación ofrece una experiencia intuitiva y segmentada en dos perfiles de acceso: usuario común y cuidador. Esta diferenciación permite restringir las funcionalidades avanzadas, como la configuración de sensores y automatizaciones, a quienes tengan el conocimiento o la responsabilidad para gestionarlas.

7.1. Arquitectura general de la aplicación

La aplicación se compone de las siguientes vistas:

- **LoginScreen.kt:** Pantalla inicial de autenticación. Permite ingresar como usuario o como cuidador mediante validación local de credenciales.
- **MainActivity.kt:** Punto de entrada de la aplicación. Gestiona el estado de sesión, la navegación entre pantallas y la conexión con el servicio MQTT.
- **EnchufeControlScreen.kt:** Vista principal del usuario común. Permite encender o apagar el enchufe y visualizar en tiempo real los datos de temperatura, humedad, luz, y el estado del sistema.
- **SettingsScreen.kt:** Vista exclusiva del cuidador. Permite configurar umbrales de sensores, activar automatizaciones y temporizador, y gestionar notificaciones.

7.2. Librerías y dependencias utilizadas

La aplicación utiliza las siguientes tecnologías:

- **Jetpack Compose:** Para el diseño de la interfaz gráfica.
- **PubSubClient para MQTT (a través de MQTTManager):** Para conectarse con los *feeds* de Adafruit IO.

- **State management (remember y mutableStateOf):** Para mantener actualizados los valores mostrados en pantalla.
- **Notificaciones locales (NotificationCompat):** Para alertar al usuario en caso de alta temperatura o baja humedad.

7.3. Flujo de navegación

Desde MainActivity.kt, se controla la navegación condicional entre pantallas según el tipo de usuario y las acciones tomadas:

- Al iniciar, el usuario ve la pantalla de login.
- Si inicia sesión como usuario común, se accede a EnchufeControlScreen.
- Si inicia como cuidador, se accede directamente a SettingsScreen.
- Desde cualquier vista se puede volver atrás, cerrar sesión o acceder a la configuración si corresponde.

7.4. Pantalla de Inicio de Sesión (LoginScreen.kt)

La pantalla de inicio de sesión permite seleccionar el tipo de usuario que utilizará la aplicación. Se ofrecen dos opciones: **usuario común**, con acceso a funcionalidades básicas como el encendido y apagado del enchufe y la visualización de sensores; y **cuidador**, con acceso a opciones avanzadas de configuración, automatizaciones, definición de umbrales y recepción de notificaciones. La interfaz está diseñada para ser clara y amigable, incorporando botones grandes, íconos representativos y tipografía legible, lo que resulta especialmente útil para personas mayores o con dificultades visuales. La disposición visual y funcional de esta pantalla se muestra en la Figura 41, correspondiente a la clase LoginScreen.kt.

Figura 41: Pantalla de loginScreen

```
55          Button(
56              onClick = {
57                  if (username == "usuario" && password == "1234") {
58                      onUserLoginSuccess()
59                  } else {
60                      errorMessage = "Credenciales de usuario incorrectas"
61                  }
62              },
63              modifier = Modifier.fillMaxWidth()
64          ) {
65              Text("Entrar como Usuario")
66          }
67
68          Spacer(modifier = Modifier.height(12.dp))
69
70          Button(
71              onClick = {
72                  if (username == "cuidador" && password == "admin") {
73                      onAdminLoginSuccess()
74                  } else {
75                      errorMessage = "Credenciales de cuidador incorrectas"
76                  }
77              },
78              modifier = Modifier.fillMaxWidth()
79          ) {
80              Text("Entrar como Cuidador")
81          }
```

Figura 42: Fragmento de código de loginScreen

7.5. Actividad principal y navegación (MainActivity.kt)

La clase MainActivity.kt, ver Figura 43 define el flujo de navegación condicional de la aplicación, así como la inicialización del gestor MQTT y la suscripción a datos en tiempo real.

- Se utilizan variables isUserLoggedIn e isAdminLoggedIn para controlar el estado de sesión.
- Se mantiene la conexión activa con Adafruit IO mediante mqttManager.
- Se gestionan notificaciones locales si se superan ciertos umbrales de temperatura o humedad.
- Según el tipo de usuario, se carga la vista correspondiente

```
76                  // Navegación
77                  when {
78                      isAdminLoggedIn -> {
79                          SettingsScreen(
80                              onBack = { isAdminLoggedIn = false },
81                              mqttManager = mqttManager,
82                              controlLuz = controlLuz,
83                              controlTemp = controlTemp,
84                              alertasActivadas = alertasActivadas
85                          )
86                      }
87
88                      isUserLoggedIn -> {
89                          if (isInSettings) {
90                              SettingsScreen(
91                                  onBack = { isInSettings = false },
92                                  mqttManager = mqttManager,
93                                  controlLuz = controlLuz,
94                                  controlTemp = controlTemp,
95                                  alertasActivadas = alertasActivadas
96                              )
```

Figura 43: Fragmento de código de MainActivity

También se definen los canales de notificación para emitir alertas locales cuando la temperatura supere los 35°C o la humedad caiga por debajo del 30%.

7.6. Pantalla de control del enchufe (EnchufeControlScreen.kt)

Esta pantalla, que se puede ver en la Figura 44 y su código en la Figura 45, corresponde a la vista principal del usuario común, permitiendo la interacción básica con el sistema de forma simple, visual y funcional. Está diseñada para facilitar el acceso a las acciones esenciales sin exponer configuraciones avanzadas.

Funcionalidades principales:

- Visualización del estado del enchufe (encendido/apagado).
- Visualización de datos de sensores:
  - 🌡️ Temperatura
  - 💧 Humedad
  - 💡 Nivel de luz
- Visualización del tiempo restante del temporizador.
- Activación/desactivación del enchufe mediante botones dedicados.
- Visualización del estado de los modos automáticos (luz, temperatura, alertas).
- Acceso a la configuración y cierre de sesión.

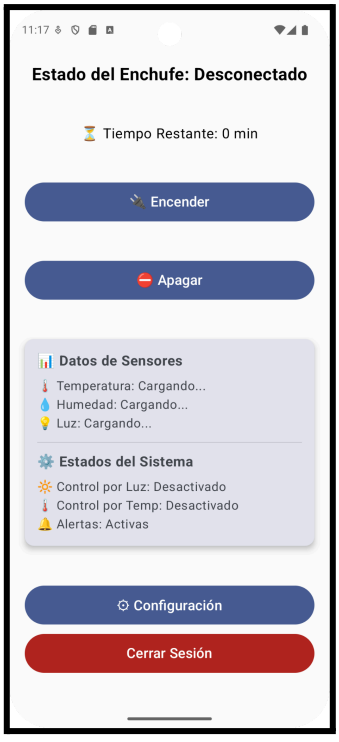


Figura 44: Pantalla de EnchufeControlScreen

```
Button(
    onClick = {
        isSending = true
        mqttManager.sendMessage("ON", "nicohc/feeds/enchufe-dot-control")
    },
    modifier = Modifier
        .fillMaxWidth()
        .height(50.dp)
) {
    Text("▶ Encender", fontSize = 18.sp)
}

Button(
    onClick = {
        isSending = true
        mqttManager.sendMessage("OFF", "nicohc/feeds/enchufe-dot-control")
    },
    modifier = Modifier
        .fillMaxWidth()
        .height(50.dp)
) {
    Text("⛔ Apagar", fontSize = 18.sp)
}
```

Figura 45: Fragmento código de EnchufeControlScreen

Cada acción de encendido o apagado se refleja en tiempo real a través de MQTT y es registrada por el microcontrolador. Además, se incluye un pequeño retraso visual de "Enviando..." que mejora la experiencia del usuario y da retroalimentación del proceso.

7.7. Pantalla de configuración avanzada (SettingsScreen.kt)

Esta pantalla, que se ve en la Figura 46 y su código en la Figura 47, es exclusiva para el **perfil cuidador**, quien tiene acceso completo a los parámetros que rigen el comportamiento automático del sistema. Está pensada para una gestión sencilla pero completa de la automatización.

Funcionalidades disponibles:

- Activar o desactivar control automático por luz (modo noche/día).
    - Ajuste del umbral de luminosidad mediante Slider.
- Activar o desactivar control automático por temperatura.
    - Configuración de temperatura máxima y mínima.
- Activar o desactivar temporizador.

- Selección de duración entre 1 y 60 minutos.
- Desactiva automáticamente los otros modos para evitar conflictos.

- Activar o desactivar alertas locales (notificaciones).
- Guardar los parámetros configurados, los cuales se envían a Adafruit IO a través del mqttManager.

Figura 46: Pantalla de SettingsScreen

```
116         Button(
117             onClick = {
118                 mqttManager.sendMessage(tempMax.toString(), "nicohc/feeds/enchufe-dot-temp_max")
119                 mqttManager.sendMessage(tempMin.toString(), "nicohc/feeds/enchufe-dot-temp_min")
120                 mqttManager.sendMessage(umbralLuz.toString(), "nicohc/feeds/enchufe-dot-luz_umbral")
121 
122                 if (useTimer) {
123                     mqttManager.sendMessage(timer.toString(), "nicohc/feeds/enchufe-dot-timer")
124                 } else {
125                     mqttManager.sendMessage("0", "nicohc/feeds/enchufe-dot-timer")
126                 }
127             },
128             modifier = Modifier.fillMaxWidth()
129         ) {
130             Text("Guardar Configuración", fontSize = 18.sp)
131         }
```

Figura 47: Fragmento de código de SettingsScreen

La configuración realizada en esta vista es persistente mientras se mantenga conectada la sesión MQTT, y permite adaptar el comportamiento del enchufe a las condiciones y necesidades del entorno.

7.8. Diagrama General de la Aplicación Móvil

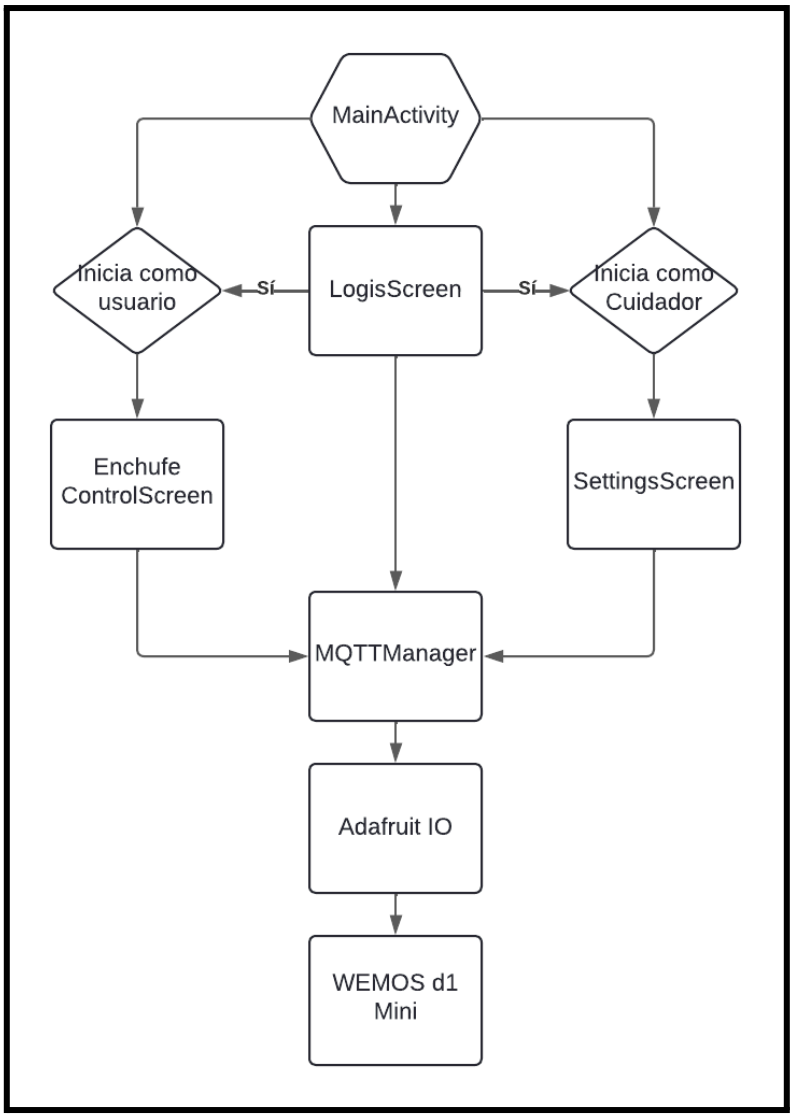


Figura 48: Diagrama de flujos de la aplicación

El diagrama de la Figura 48 ilustra la estructura jerárquica de la aplicación. Desde MainActivity, se gestiona la navegación entre las vistas principales, según el tipo de usuario autenticado. Además, se mantiene la conexión con la plataforma Adafruit IO mediante el gestor MQTT, lo que permite enviar y recibir datos en tiempo real entre el dispositivo físico y la app.

Capítulo 8

8. Resolución de objetivos, conclusión y discusión

En este capítulo se presenta el análisis final del proyecto, el cual contempla la resolución de cada uno de los objetivos específicos definidos al inicio del trabajo, explicando cómo se implementan y qué resultados se obtienen en la práctica. A partir de estos resultados, se desarrolla una discusión crítica que permite reflexionar sobre el rendimiento del sistema, las decisiones de diseño adoptadas, sus limitaciones y los aprendizajes obtenidos durante su construcción.

Este análisis permite evaluar el grado de cumplimiento de los objetivos y examinar el impacto del sistema en un entorno real, considerando especialmente su aplicación en la automatización del hogar y la asistencia a personas mayores. Finalmente, se establecen conclusiones generales que sintetizan los principales aportes del proyecto, junto con recomendaciones para futuras mejoras o desarrollos derivados.

8.1. Resolución de objetivos

A continuación, se presenta la resolución de cada uno de los objetivos específicos definidos para el desarrollo del sistema de enchufe inteligente, detallando cómo se implementa cada uno de ellos y los resultados que se obtienen.

8.1.1. Determinar los componentes clave para la construcción del sistema

Para la construcción del prototipo de enchufe inteligente, se seleccionan componentes que cumplan con criterios de compatibilidad eléctrica, eficiencia energética, facilidad de integración, bajo costo y disponibilidad en el mercado local. La elección de cada uno se realiza tras una etapa previa de análisis y validación, considerando distintas alternativas antes de definir la arquitectura final del sistema.

Durante este proceso, se descartan ciertos componentes que inicialmente fueron considerados, como el **sensor SCT-013**, orientado a la medición de corriente. Aunque este sensor ofrecía la posibilidad de cuantificar el consumo energético en tiempo real, se opta

por retirarlo del diseño debido a la complejidad adicional en el procesamiento de señales analógicas, el uso de un amplificador operacional adicional (como el LM358) y la baja precisión en condiciones de baja carga, lo cual no resultaba crítico para los objetivos de asistencia y automatización definidos en el proyecto.

En su lugar, se priorizan sensores ambientales como el **DHT22** y el **LDR**, que permiten implementar automatizaciones más simples y eficientes, centradas en la comodidad del usuario y la adaptabilidad al entorno. Asimismo, se opta por el uso del **relé de estado sólido SSR-40DA**, descartando soluciones con relés mecánicos tradicionales, debido a su mayor durabilidad, conmutación silenciosa y menor riesgo de fallas por desgaste físico.

8.1.2. Implementar un sistema de monitoreo ambiental

Con el fin de capturar condiciones del entorno en tiempo real, se implementa un sistema de monitoreo ambiental mediante sensores de temperatura, humedad y luminosidad. Esta capacidad permite no solo registrar datos relevantes para el confort y la seguridad del usuario, sino también generar automatizaciones basadas en condiciones ambientales.

Para validar su funcionamiento, se realizan pruebas en un entorno cerrado, donde se expone el sensor de temperatura a fuentes de calor controladas, observando respuestas proporcionales a través del monitor serial del entorno de desarrollo. De forma similar, el sensor de luz es evaluado mediante pruebas con linternas de distinta intensidad y mediante el bloqueo total de la luz, constatando variaciones lógicas en los valores medidos.

8.1.3. Diseñar un sistema de automatización y control remoto

Con el objetivo de facilitar la gestión del enchufe sin intervención directa constante, se diseña un sistema de automatización configurable desde la aplicación móvil. Este sistema contempla tres modos operativos: automatización por nivel de luz, por temperatura y por temporizador, todos gestionados desde una interfaz de usuario intuitiva. Para prevenir conflictos lógicos, se establece que al activar un modo, los otros dos se desactivan automáticamente.

Adicionalmente, se habilita un modo de control manual y se incorporan alertas locales mediante mensajes visuales en la app, activadas ante condiciones críticas como temperaturas elevadas o niveles bajos de humedad.

La lógica del sistema es probada en un entorno controlado, verificando que las condiciones ambientales captadas por los sensores activan o desactivan el enchufe según los umbrales definidos. Si bien no se realizan pruebas formales de usabilidad con usuarios finales ni se recopilan experiencias de personas mayores, el sistema demuestra un funcionamiento autónomo acorde a los parámetros establecidos, constituyendo una base técnica funcional sobre la cual podrían realizarse futuras evaluaciones con grupos objetivo específicos.

8.1.4. Desarrollar una aplicación móvil intuitiva

Con el fin de facilitar el control remoto del sistema, se desarrolla una aplicación móvil en Android Studio utilizando Kotlin y la librería Jetpack Compose. La interfaz permite al usuario visualizar los valores captados por los sensores ambientales, encender o apagar el enchufe, configurar umbrales de automatización y gestionar alertas. Se incluye además un sistema de doble cuenta, diferenciando entre un perfil de usuario común y otro de cuidador, con acceso a funciones avanzadas.

Durante el desarrollo, la aplicación es probada de forma iterativa en un dispositivo Android, validando que cada pantalla funcione correctamente, que los valores se actualicen en tiempo real y que las funciones básicas de control respondan sin errores. Si bien no se realizan pruebas de usabilidad con usuarios finales o personas mayores, la arquitectura de la aplicación se diseña considerando principios de simplicidad, navegación directa y claridad visual, en base a recomendaciones de accesibilidad para dispositivos móviles.

A futuro, se proyecta complementar esta solución con evaluaciones formales de experiencia de usuario, orientadas a validar su eficacia y facilidad de uso en el grupo objetivo.

8.2. Discusión

El desarrollo del enchufe inteligente permite analizar una serie de aspectos técnicos y funcionales que influyen directamente en la efectividad del sistema. A lo largo del proceso, se observa que la elección de componentes simples pero funcionales, como el Wemos D1 Mini y los sensores DHT22 y LDR, resulta adecuada para los objetivos del proyecto. Estos elementos permiten construir un sistema compacto, de bajo costo y fácil de ensamblar, sin sacrificar funcionalidades esenciales como el monitoreo ambiental y el control remoto.

Uno de los principales aciertos del proyecto es la implementación de una arquitectura de control clara y segmentada. La decisión de ofrecer un doble acceso, diferenciando entre usuario común y cuidador, resulta clave para abordar el enfoque de asistencia al adulto mayor. Este diseño permite adaptar el uso del sistema según las capacidades o necesidades del usuario, promoviendo tanto la autonomía como el acompañamiento.

Durante las pruebas funcionales, el sistema demuestra ser estable y confiable, con una buena capacidad de respuesta frente a los comandos enviados desde la aplicación. Sin embargo, se identifican algunas áreas de mejora, como la necesidad de sincronizar de forma más persistente el estado del sistema entre el microcontrolador y la interfaz móvil, especialmente después de reinicios o pérdida de conexión.

En términos de automatización, los tres modos implementados (por luz, temperatura y temporizador) cubren situaciones comunes dentro de un entorno doméstico. La lógica de exclusión entre modos evita conflictos y garantiza un comportamiento predecible. Aun así, se reconoce que existe espacio para expandir las condiciones de control, incorporando sensores adicionales o aprendizaje del comportamiento de los usuarios.

Por otro lado, la conexión mediante MQTT y el uso de Adafruit IO ofrece una solución rápida y funcional para el prototipado y despliegue del sistema. No obstante, se observa que para una versión a mayor escala o en producción, sería conveniente evaluar plataformas más robustas o incluso desarrollar un backend propio para garantizar mayor independencia y seguridad.

Finalmente, se concluye que el sistema cumple su propósito no sólo como proyecto técnico, sino también como una solución aplicable en contextos reales. La combinación entre automatización, monitoreo y facilidad de uso lo convierten en una herramienta efectiva para hogares que buscan mayor control energético y asistencia personalizada.

8.3. Conclusiones

A partir del desarrollo del presente proyecto, se concluye que es posible implementar un sistema de automatización y monitoreo ambiental funcional, de bajo costo y orientado a la asistencia doméstica utilizando tecnologías accesibles como el Wemos D1 Mini, sensores analógicos y una aplicación móvil personalizada.

La arquitectura propuesta permite controlar el estado de un enchufe inteligente de manera remota o automática, en función de variables ambientales tales como la temperatura, la humedad y la luminosidad. Esta capacidad se complementa con un sistema de alertas locales y la posibilidad de configurar el comportamiento del sistema desde una interfaz amigable, incluso por parte de usuarios con escaso dominio tecnológico.

Uno de los principales aportes del proyecto es la incorporación de una doble vista de usuario, que distingue entre un perfil de uso básico y uno avanzado (cuidador). Esta diferenciación permite adaptar la experiencia según el tipo de usuario, lo que refuerza el enfoque de asistencia al adulto mayor sin limitar el control total del sistema.

Asimismo, la implementación de la comunicación mediante el protocolo MQTT con la plataforma Adafruit IO demuestra ser efectiva para el monitoreo en tiempo real y la transmisión de comandos entre la aplicación móvil y el dispositivo físico. La lógica de funcionamiento incorporada en el microcontrolador permite gestionar correctamente las automatizaciones sin generar conflictos entre los distintos modos de operación.

Las pruebas funcionales realizadas confirman que el sistema se comporta de manera estable frente a distintos escenarios, respondiendo de forma oportuna y confiable. Además, el diseño modular del software y hardware permite escalar el sistema o adaptarlo a otros dispositivos del hogar inteligente con relativa facilidad.

En conjunto, el proyecto valida la viabilidad técnica y funcional de una solución domótica orientada al ahorro energético, la automatización y la asistencia domiciliaria. El resultado es un sistema replicable, educativo y útil, tanto para usuarios finales como para futuras líneas de desarrollo en el área de hogares inteligentes.

8.4. Recomendaciones

A partir del análisis y evaluación del sistema implementado, se proponen las siguientes recomendaciones para futuras mejoras, escalabilidad del proyecto o implementación en contextos reales:

- **Ampliar las funcionalidades de automatización:**
  Se sugiere incorporar nuevos sensores (por ejemplo, de gas, movimiento o presencia) para ampliar las condiciones de activación del enchufe y adaptarse a distintos contextos de uso.
- **Mejorar la persistencia del estado del sistema:**
  Se recomienda reforzar la sincronización entre el microcontrolador y la aplicación móvil para mantener actualizado el estado del enchufe y las configuraciones, incluso después de interrupciones o reinicios.
- **Desarrollar una versión multiplataforma de la aplicación:**
  La creación de versiones para otros sistemas operativos como iOS o incluso una versión web permitiría ampliar el acceso al sistema a una mayor cantidad de usuarios.
- **Integración con asistentes virtuales:**
  Se considera beneficioso integrar el sistema con plataformas como **Google Assistant** o **Amazon Alexa**, lo que permitiría controlar el enchufe mediante comandos de voz.
- **Aumentar la seguridad del sistema:**
  Para una eventual implementación en hogares reales, se sugiere reforzar la seguridad de las credenciales, cifrar las comunicaciones y evitar el uso de claves fijas en el código fuente.
- **Agregar guía de uso para usuarios mayores:**
  Se recomienda incorporar dentro de la aplicación tutoriales interactivos o guías

simples que orienten al usuario paso a paso en el uso de las funciones básicas, priorizando a personas con poca experiencia tecnológica.

- **Escalar a control de múltiples enchufes:**
  Adaptar el sistema para controlar más de un enchufe desde la misma aplicación aumentaría su utilidad práctica y permitiría gestionar múltiples dispositivos del hogar desde un solo punto.

9. Glosario

Adafruit IO: Plataforma en la nube que permite la visualización y control de dispositivos IoT mediante el protocolo MQTT.

Automatización: Proceso mediante el cual un sistema realiza acciones de manera autónoma, sin intervención directa del usuario.

Ciclo de trabajo (duty cycle): Proporción de tiempo durante el cual una señal o sistema está activo, común en el control de relés o actuadores.

Domótica: Conjunto de tecnologías orientadas a la automatización y control inteligente de una vivienda.

Enchufe inteligente: Dispositivo que permite gestionar remotamente el encendido o apagado de cargas eléctricas, generalmente mediante sensores o aplicaciones móviles.

IoT (Internet of Things): Conjunto de dispositivos conectados a Internet que pueden enviar y recibir datos para interactuar con su entorno.

MQTT (Message Queuing Telemetry Transport): Protocolo de mensajería ligero y eficiente, usado comúnmente en sistemas IoT para comunicación entre dispositivos y servidores.

Microcontrolador: Circuito integrado programable que puede ejecutar instrucciones para interactuar con sensores, actuadores y otros dispositivos electrónicos.

Sensor DHT22: Sensor digital que permite medir temperatura y humedad con buena precisión y estabilidad.

Sensor LDR: Componente que varía su resistencia según la intensidad de luz, permitiendo medir la luminosidad del entorno.

Relé SSR (Solid State Relay): Relé de estado sólido que permite conmutar corriente eléctrica sin partes móviles, ideal para automatización silenciosa y duradera.

Wemos D1 Mini: Placa de desarrollo basada en el chip ESP8266, con conectividad Wi-Fi y ampliamente usada en proyectos IoT de bajo costo.

10. Lista de abreviaturas

AC: Corriente Alterna/Alternating current

DC: Corriente Continua/Direct Current

DHT: Digital Humidity and Temperature

GPIO: General Purpose Input/Output

IoT: Internet of Things

LDR: Light Dependent Resistor

MQTT: Message Queuing Telemetry Transport

SSR: Solid State Relay

UI: User Interface

VCC: Voltaje común del circuito (+)/ Common circuit voltage (+)

GND: Ground (–)/ tierra del circuito (–)

IDE: Integrated Development Environment

IP65: Grado de protección contra polvo y agua/Degree of protection against dust and water

UX: User Experience

11. Bibliografía

Addinformática. (2022). *Brecha digital: desafíos del acceso a Internet para las personas mayores.* https://addinformatica.com/noticias/brecha-digital-desafios-del-acceso-a-internet-para-las-personas-mayores/

Bajwa, H. A., Imran, A. S., Khan, Z. A., Sajjad, M., & Kastrati, Z. (2021). Towards smart home automation using IoT-enabled edge-computing paradigm. *Sensors, 21*(14), 4932. https://doi.org/10.3390/s21144932

Chua, Q. S., & Yi, S. Y. (2021). IoT Smart Distribution Box (SDB) for Homestay Energy Management System by Using Arduino and ESP32. *IEEE Xplore.* https://ieeexplore.ieee.org/document/9490420

Circuito.io. (2023). *Guías y diagramas de conexión de sensores para Arduino.* https://www.circuito.io

Comisión Económica para América Latina y el Caribe (CEPAL). (2021). *Envejecimiento y derechos de las personas mayores.* https://www.cepal.org/es/publicaciones/46886-envejecimiento-derechos-personas-mayores-decada-seguimiento-la-carta-san-jose

Components101. (s.f.-a). *LDR Sensor Datasheet.* https://components101.com/sites/default/files/component_datasheet/LDR%20Datasheet.pdf

Components101. (s.f.-b). *DHT22 Sensor Datasheet.* https://components101.com/sites/default/files/component_datasheet/DHT22-Datasheet.pdf

Domótica M&C. (2022). *La pyme que busca cambiar los paradigmas de las casas inteligentes en Chile.* https://constructor.lacuarta.com/noticias/yo-construyo/domotica-mc-la-pyme

Duangphasuk, S., & Thammarat, C. (2020). Review of Internet of Things (IoT): Security Issue and Solution. https://doi.org/10.1109/ECTI-CON49241.2020

EMQX. (2024, junio 27). *MQTT vs HTTP: Ultimate IoT Protocol Comparison Guide*. EMQX. https://www.emqx.com/en/blog/mqtt-vs-http-protocols-in-iot-iiot

Fernández, J., García, E., & Rodríguez, A. (2021). Smart homes for aging populations: A review. *Journal of Ambient Intelligence and Smart Environments, 13*(1), 45–64.

Fluke. (2024). *How to test for continuity with a multimeter*. Fluke Corporation https://www.fluke.com/en-us/learn/blog/digital-multimeters/how-to-test-for-continuity

Generadoras de Chile. (2023). *Estadísticas Clave 2023: Sistema Eléctrico Nacional.* https://generadoras.cl/documentos/reportes-anuales/estadisticas-clave-2023-sistema-electrico-nacional

Geriasistencia. (2019). *Personas mayores y nuevas tecnologías: beneficios y dificultades.* https://www.geriasistencia.com/personas-mayores-y-nuevas-tecnologias-beneficios-y-dificultades/

Gómez Díaz, L. (2019). *Gestión de los recursos domiciliarios desde la utilización de software y hardware libre. Mutis*, 9(1), 40–49. https://doi.org/10.21789/22561498.1461 revistas.utadeo.edu.co+1

Gupta, V., & Sinha, A. (2020). Security and Privacy in IoT-enabled Smart Home: A Systematic Review. *Sensors, 20*(20), 5867. https://doi.org/10.3390/s20205867

HandsOn Technology. (s.f.). *SSR-40DA Solid State Relay Datasheet.* https://handsontec.com/dataspecs/discrete/SSR-40DA.pdf

Hardy, B. (2019). *Android Studio Development Essentials – Android 10 Edition.* Payload Media.

Hasan, M. M., Mishu, M. H., & Islam, M. M. (2020). Comparison of ESP8266 and ESP32 microcontrollers for IoT applications. *International Journal of Advanced Computer Science and Applications, 11*(12), 131–135. https://thesai.org/Downloads/Volume11No12/Paper_16-Comparison_of_ESP8266_and_ESP32.pdf

International Electrotechnical Commission (IEC). (2010). *IEC 60335-1: Household and similar electrical appliances – Safety – Part 1: General requirements* (5th ed.). IEC.

IoTEDU. (2020). *¿Qué es la domótica y cómo funciona?* https://iot-edu.org/blog/que-es-la-domotica

KiCad. (s.f.). *Official website of KiCad EDA*. https://www.kicad.org/

Kwon, H., Kim, S., & Lee, J. (2022). MQTT and HTTP-based communication analysis in smart homes. *International Journal of Smart Home, 16*(2), 43–54.

Light, A. (2017). MQTT Essentials – A Lightweight IoT Protocol. *HiveMQ Blog.* https://www.hivemq.com/blog/mqtt-essentials/

Light, R. (2017). Mosquitto: Server and client implementation of the MQTT protocol. *Journal of Open Source Software, 2*(13), 265. https://doi.org/10.21105/joss.00265

Lin, Y., Li, Y., & Zhao, H. (2020). Energy-efficient smart plugs for domestic automation: A case study. *Energy and Buildings, 224,* 110264. https://doi.org/10.1016/j.enbuild.2020.110264

Luo, Z., Zhang, W., & Liu, Y. (2020). Remote assistance technologies for elderly in smart homes. *Procedia Computer Science, 176,* 3122–3130. https://doi.org/10.1016/j.procs.2020.09.214

Maier, A., Felser, M., & Zolotas, A. (2017). Industrial Communication Protocols and Performance: ESP8266 vs. ESP32. *IEEE Transactions on Industrial Electronics, 64*(3), 2077–2084.

MCI Electronics. (s.f.). *Fuente de poder 5V 0.6A 3W Mean Well IRM-03-5: AC-DC encapsulada, certificación de seguridad internacional*. https://mcielectronics.cl/shop/product/fuente-de-poder-5v-0-6a-3w-mean-well-irm-03-5/

MCI Electronics. (s.f.). *Sensor de humedad y temperatura DHT22.* https://mcielectronics.cl/shop/product/sensor-de-humedad-y-temperatura-dht22/

MCI Electronics. (s.f.). *Sensor de luz LDR.* https://mcielectronics.cl/shop/product/sensor-de-luz-ldr-10499/

MCI Electronics. (s.f.). *Relé SSR-40A 250 V AC: Relé de estado sólido para automatización y control de cargas*. https://mcielectronics.cl/shop/product/rele-ssr-40a-250v/

MEAN WELL. (2019). *IRM-03-5: 3W AC-DC encapsulated power supply module*. Mean Well Enterprises Co., Ltd. https://www.meanwell.com/productPdf.aspx?i=403

Mehndiratta, P., Gupta, A., & Garg, D. (2021). Development and performance evaluation of DHT22 sensor-based IoT system for real-time monitoring of temperature and humidity. *Journal of Sensors, 2021*. https://doi.org/10.1155/2021/5564342

Ministerio de Energía. (2018). *Estudio de consumos energéticos del sector residencial en Chile 2018.* https://www.energia.gob.cl/sites/default/files/documentos/informe_final_caracterizacion_residencial_2018.pdf

Ministerio de Vivienda y Urbanismo (MINVU). (2019). *Manual de Aplicación - Construcción Sustentable.* https://csustentable.minvu.gob.cl/wp-content/uploads/2020/01/MANUAL-CVS-012020.pdf

Mistry, I., Tanwar, S., & Kumar, N. (2020). A systematic review of smart homes based on the internet of things. *Telecommunication Systems.* https://doi.org/10.1007/s11235-020-00713-y

Mistry, R., Tanwar, S., & Kumar, N. (2020). Security of MQTT Protocol in Smart Homes. *Computer Communications, 152,* 180–195. https://doi.org/10.1016/j.comcom.2020.01.004

Mistry, R., Tanwar, S., & Kumar, N. (2020). A systematic survey on the MQTT protocol: Present and future perspectives. *Computer Communications, 146,* 125–138. https://doi.org/10.1016/j.comcom.2019.09.010

Mohamed, K., & El Shenawy, A. (2023). A smart IoT-based home automation system for controlling and monitoring home appliances. *International Review of Automatic Control, 16*(5). https://doi.org/10.15866/ireaco.v16i5.23872

Neto, J., Duarte, L., & Rodrigues, J. (2024). Monitoring energy savings with IoT smart plugs. *Journal of Green Technologies, 9*(1), 22–30.

Nugraha, R. A., & Fauzi, A. (2021). Smart Home System for Elderly People Based on Internet of Things (IoT). *Journal of Physics: Conference Series, 1869*(1), 012081. https://doi.org/10.1088/1742-6596/1869/1/012081

Nugraha, R. P., & Fauzi, A. (2021). Development of Energy-Efficient Smart Plug Based on IoT to Improve Energy Consumption Behavior. *IEEE Transactions on Consumer Electronics, 67*(3), 182–189. https://doi.org/10.1109/TCE.2021.3082845

Ristik, M., Begovic, A., & Soldo, B. (2021). ESP8266 and ESP32 in automation: Comparative study. *Electronics, 10*(17), 2102. https://doi.org/10.3390/electronics10172102

Rokonuzzaman, M., Mishu, M. K., & Islam, M. R. (2021). Design and Implementation of an IoT-Enabled Smart Plug Socket for Home Energy Management. *IEEE Smart Grid Conference.* https://ieeexplore.ieee.org/document/9490420

Rokonuzzaman, M., Mishu, R. I., & Islam, M. S. (2021). IoT-Based Energy Management in Smart Homes: A Case Study. *International Journal of Computer Applications, 183*(25), 22–28. https://doi.org/10.5120/ijca2021921761

Sajjad, A., & Kastrati, S. (2021). A review of smart homes for elderly safety and health care. *Sensors, 21*(5), 1613. https://doi.org/10.3390/s21051613

Sajjad, M., & Kastrati, Z. (2021). Towards smart home automation using IoT-enabled edge-computing paradigm. *Sensors, 21*(14), 4932. https://doi.org/10.3390/s21144932

Statista. (2023). *Number of connected devices worldwide 2015–2025.* https://www.statista.com/statistics/471264/iot-number-of-connected-devices-worldwide/

Underwriters Laboratories (UL). (2007). *UL 60950-1: Information Technology Equipment – Safety – Part 1: General Requirements* (2nd ed.). Underwriters Laboratories Inc.

Venkatraman, S., Overmars, A., & Thong, M. (2021). Smart Home Automation—Use Cases of a Secure and Integrated Voice-Control System. *Systems, 9*(4), 77. https://www.mdpi.com/2079-8954/9/4/77

Wang, J., & Kim, H. S. (2023). Visualizing the Landscape of Home IoT Research: A Bibliometric Analysis Using VOSviewer. *Sensors, 23*(6), 3086. https://doi.org/10.3390/s23063086

Anexos.

SettingsScreen.kt

```
import androidx.compose.foundation.layout.*
import androidx.compose.material3.*
import androidx.compose.runtime.*
import androidx.compose.ui.Modifier
import androidx.compose.ui.text.font.FontWeight
import androidx.compose.ui.unit.dp
import androidx.compose.ui.unit.sp
import com.example.enchufe2.mqtt.MQTTManager
```

```
@Composable
fun SettingsScreen(
    onBack: () -> Unit,
    mqttManager: MQTTManager,
    controlLuz: MutableState<Boolean>,
    controlTemp: MutableState<Boolean>,
    alertasActivadas: MutableState<Boolean> // ✅ Agregado
) {
    var umbralLuz by remember { mutableStateOf(500) }
    var tempMax by remember { mutableStateOf(30) }
    var tempMin by remember { mutableStateOf(25) }
    var timer by remember { mutableStateOf(10) }

    LaunchedEffect(Unit) {
        if (mqttManager.isConnected()) {
            mqttManager.sendMessage("GET",
"nicohc/feeds/enchufe-dot-luz_automatico")
            mqttManager.sendMessage("GET",
"nicohc/feeds/enchufe-dot-temp_automatico")
        }
    }

    Column(
        modifier = Modifier
            .fillMaxSize()
            .padding(16.dp),
        verticalArrangement = Arrangement.Center
    ) {
        Text("⚙ Configuración de Sensores", fontSize = 24.sp,
fontWeight = FontWeight.Bold)
        Spacer(modifier = Modifier.height(20.dp))
```

```
        // 🔔 Notificaciones
        Text("🔔 Notificaciones", fontSize = 18.sp, fontWeight
= FontWeight.Bold)
        Switch(
            checked = alertasActivadas.value,
            onCheckedChange = { alertasActivadas.value = it }
        )

        Spacer(modifier = Modifier.height(20.dp))

        // 💡 Control de luz
        Text("💡 Control por Luz", fontSize = 18.sp,
fontWeight = FontWeight.Bold)
        Switch(
            checked = controlLuz.value,
            onCheckedChange = {
                controlLuz.value = it
                mqttManager.sendMessage(if (it) "ON" else
"OFF", "nicohc/feeds/enchufe-dot-luz_automatico")
            }
        )
        Spacer(modifier = Modifier.height(10.dp))
        Slider(value = umbralLuz.toFloat(), onValueChange = {
umbralLuz = it.toInt() }, valueRange = 100f..1000f, steps =
9)
        Text("Umbral de Luz: $umbralLuz lx")

        Spacer(modifier = Modifier.height(20.dp))

        // 🌡️ Control de temperatura
        Text("🌡️ Control por Temperatura", fontSize = 18.sp,
fontWeight = FontWeight.Bold)
        Switch(
```

```
            checked = controlTemp.value,
            onCheckedChange = {
                controlTemp.value = it
                mqttManager.sendMessage(if (it) "ON" else
"OFF", "nicohc/feeds/enchufe-dot-temp_automatico")
            }
        )
        Spacer(modifier = Modifier.height(10.dp))

        Text("Temperatura Máxima: $tempMax°C")
        Slider(value = tempMax.toFloat(), onValueChange = {
tempMax = it.toInt() }, valueRange = 20f..40f)

        Spacer(modifier = Modifier.height(10.dp))

        Text("Temperatura Mínima: $tempMin°C")
        Slider(value = tempMin.toFloat(), onValueChange = {
tempMin = it.toInt() }, valueRange = 15f..30f)

        Spacer(modifier = Modifier.height(20.dp))

        // ⏳ Control por Temporizador
        var useTimer by remember { mutableStateOf(false) }

        Text("⏳ Control por Temporizador", fontSize = 18.sp,
fontWeight = FontWeight.Bold)

        Switch(
            checked = useTimer,
            onCheckedChange = {
                useTimer = it
                if (it) {
                    controlLuz.value = false
```

```
                    controlTemp.value = false
                    mqttManager.sendMessage("OFF",
"nicohc/feeds/enchufe-dot-luz_automatico")
                    mqttManager.sendMessage("OFF",
"nicohc/feeds/enchufe-dot-temp_automatico")
                } else {
                    mqttManager.sendMessage("0",
"nicohc/feeds/enchufe-dot-timer")
                }
            }
        )

        Spacer(modifier = Modifier.height(10.dp))

        Slider(
            value = timer.toFloat(),
            onValueChange = { timer = it.toInt() },
            valueRange = 1f..60f,
            enabled = useTimer
        )
        Text("$timer min", modifier = Modifier.padding(start =
8.dp))

        Button(
            onClick = {
                mqttManager.sendMessage(tempMax.toString(),
"nicohc/feeds/enchufe-dot-temp_max")
                mqttManager.sendMessage(tempMin.toString(),
"nicohc/feeds/enchufe-dot-temp_min")
                mqttManager.sendMessage(umbralLuz.toString(),
"nicohc/feeds/enchufe-dot-luz_umbral")

                if (useTimer) {
```

```
                    mqttManager.sendMessage(timer.toString(),
"nicohc/feeds/enchufe-dot-timer")
                } else {
                    mqttManager.sendMessage("0",
"nicohc/feeds/enchufe-dot-timer")
                }
            },
            modifier = Modifier.fillMaxWidth()
        ) {
            Text("Guardar Configuración", fontSize = 18.sp)
        }

        Spacer(modifier = Modifier.height(20.dp))

        Button(onClick = { onBack() }, modifier =
Modifier.fillMaxWidth()) {
            Text("Volver", fontSize = 18.sp)
        }
    }
}
```

MQTTManager

```
package com.example.enchufe2.mqtt

import android.util.Log
import kotlinx.coroutines.*
import org.eclipse.paho.client.mqttv3.*
import androidx.compose.runtime.MutableState

class MQTTManager {
    private val brokerUrl = "tcp://io.adafruit.com:1883"
    private val username = "nicohc"
    private val password = "aio_KOQz9077t0pnepBgwcsibIrWSALK"
```

```
    private val topicControl =
"nicohc/feeds/enchufe-dot-control"
    private val topicStatus =
"nicohc/feeds/enchufe-dot-status"
    private val topicTemp =
"nicohc/feeds/enchufe-dot-temperatura"
    private val topicHumedad =
"nicohc/feeds/enchufe-dot-humedad"
    private val topicLuz = "nicohc/feeds/enchufe-dot-luz"
    private val topicTimer = "nicohc/feeds/enchufe-dot-timer"
    private val topicLuzAuto =
"nicohc/feeds/enchufe-dot-luz_automatico"
    private val topicTempAuto =
"nicohc/feeds/enchufe-dot-temp_automatico"

    private lateinit var mqttClient: MqttClient

    fun connect(
        estado: MutableState<String>,
        temperatura: MutableState<String>,
        humedad: MutableState<String>,
        luz: MutableState<String>,
        timer: MutableState<String>,
        controlLuz: MutableState<Boolean>,
        controlTemp: MutableState<Boolean>
    ) {
        try {
            // Si ya está conectado, cerramos antes de volver
a crear
            if (this::mqttClient.isInitialized &&
mqttClient.isConnected) {
                mqttClient.disconnect()
```

```
                Log.d("MQTT", "🔌 Cliente desconectado antes
de reconectar")
            }

            val uniqueClientId =
"AndroidClient-${System.currentTimeMillis()}"
            mqttClient = MqttClient(brokerUrl, uniqueClientId,
null)

            val options = MqttConnectOptions().apply {
                userName = username
                password =
this@MQTTManager.password.toCharArray()
                isAutomaticReconnect = true
                isCleanSession = true
            }

            mqttClient.connect(options)
            Log.d("MQTT", "✅ Conectado a MQTT")

            subscribeToTopics(estado, temperatura, humedad,
luz, timer, controlLuz, controlTemp)

        } catch (e: Exception) {
            e.printStackTrace()
            Log.e("MQTT", "❌ Error al conectar MQTT:
${e.message}")
        }
    }

    private fun subscribeToTopics(
        estado: MutableState<String>,
        temperatura: MutableState<String>,
```

```
        humedad: MutableState<String>,
        luz: MutableState<String>,
        timer: MutableState<String>,
        controlLuz: MutableState<Boolean>,
        controlTemp: MutableState<Boolean>
    ) {
        mqttClient.subscribe(topicStatus) { _, message ->
            estado.value = message.toString()
            Log.d("MQTT", "Estado del enchufe:
${estado.value}")
        }

        mqttClient.subscribe(topicTemp) { _, message ->
            temperatura.value = message.toString() + "°C"
            Log.d("MQTT", "📡 Temperatura recibida:
${temperatura.value}")
        }

        mqttClient.subscribe(topicHumedad) { _, message ->
            humedad.value = message.toString() + "%"
            Log.d("MQTT", "📡 Humedad recibida:
${humedad.value}")
        }

        mqttClient.subscribe(topicLuz) { _, message ->
            luz.value = message.toString() + " lx"
            Log.d("MQTT", "📡 Luz recibida: ${luz.value}")
        }

        mqttClient.subscribe(topicTimer) { _, message ->
            timer.value = "$message min"
            Log.d("MQTT", "📡 Temporizador recibido:
${timer.value}")
```

```
        }

        mqttClient.subscribe(topicLuzAuto) { _, message ->
            controlLuz.value = (message.toString() == "ON")
            Log.d("MQTT", "💡 Luz automática:
${controlLuz.value}")
        }

        mqttClient.subscribe(topicTempAuto) { _, message ->
            controlTemp.value = (message.toString() == "ON")
            Log.d("MQTT", "🌡️ Temperatura automática:
${controlTemp.value}")
        }
    }

    fun sendMessage(message: String, topic: String) {
        CoroutineScope(Dispatchers.IO).launch {
            try {
                if (!mqttClient.isConnected) {
                    mqttClient.reconnect()
                    delay(2000)
                }
                if (mqttClient.isConnected) {
                    val mqttMessage =
MqttMessage(message.toByteArray()).apply { qos = 1 }
                    mqttClient.publish(topic, mqttMessage)
                    Log.d("MQTT", "📤 Mensaje enviado a
$topic: $message")
                } else {
                    Log.e("MQTT", "❌ No se pudo enviar el
mensaje, MQTT desconectado.")
                }
            } catch (e: Exception) {
```

```
                e.printStackTrace()
                Log.e("MQTT", "❌ Error al enviar MQTT:
${e.message}")
            }
        }
    }

    fun disconnect() {
        try {
            if (this::mqttClient.isInitialized &&
mqttClient.isConnected) {
                mqttClient.disconnect()
                Log.d("MQTT", "🔌 Cliente desconectado
correctamente")
            }
        } catch (e: Exception) {
            Log.e("MQTT", "❌ Error al desconectar:
${e.message}")
        }
    }

    fun isConnected(): Boolean {
        return this::mqttClient.isInitialized &&
mqttClient.isConnected
    }
}
```

EnchufeControlScreen.kt

```
package com.example.enchufe2.screens

import androidx.compose.foundation.layout.*
import androidx.compose.material3.*
import androidx.compose.runtime.*
```

```
import androidx.compose.ui.Alignment
import androidx.compose.ui.Modifier
import androidx.compose.ui.text.font.FontWeight
import androidx.compose.ui.unit.dp
import androidx.compose.ui.unit.sp
import com.example.enchufe2.mqtt.MQTTManager
import kotlinx.coroutines.delay

@Composable
fun EnchufeControlScreen(
    onLogout: () -> Unit,
    onSettingsClick: () -> Unit,
    mqttManager: MQTTManager,
    estado: MutableState<String>,
    temperatura: MutableState<String>,
    humedad: MutableState<String>,
    luz: MutableState<String>,
    timer: MutableState<String>,
    controlLuz: MutableState<Boolean>,
    controlTemp: MutableState<Boolean>,
    alertasActivadas: MutableState<Boolean>
) {
    var isSending by remember { mutableStateOf(false) }

    val displayEstado = if (isSending) "Enviando..." else
estado.value

    Column(
        modifier = Modifier
            .fillMaxSize()
            .padding(20.dp),
        verticalArrangement = Arrangement.SpaceEvenly,
        horizontalAlignment = Alignment.CenterHorizontally
```

```
    ) {
        Text("Estado del Enchufe: $displayEstado", fontSize =
22.sp, fontWeight = FontWeight.Bold)
        Text("⏳ Tiempo Restante: ${timer.value}", fontSize =
18.sp)

        Button(
            onClick = {
                isSending = true
                mqttManager.sendMessage("ON",
"nicohc/feeds/enchufe-dot-control")
            },
            modifier = Modifier
                .fillMaxWidth()
                .height(50.dp)
        ) {
            Text("🔌 Encender", fontSize = 18.sp)
        }

        Button(
            onClick = {
                isSending = true
                mqttManager.sendMessage("OFF",
"nicohc/feeds/enchufe-dot-control")
            },
            modifier = Modifier
                .fillMaxWidth()
                .height(50.dp)
        ) {
            Text("⛔ Apagar", fontSize = 18.sp)
        }

        Card(
```

```
            modifier = Modifier.fillMaxWidth(),
            elevation =
CardDefaults.cardElevation(defaultElevation = 4.dp)
        ) {
            Column(modifier = Modifier.padding(16.dp)) {
                Text("📊 Datos de Sensores", fontSize = 18.sp,
fontWeight = FontWeight.Bold)
                Spacer(modifier = Modifier.height(8.dp))
                Text("🌡️ Temperatura: ${temperatura.value}",
fontSize = 16.sp)
                Text("💧 Humedad: ${humedad.value}", fontSize
= 16.sp)
                Text("💡 Luz: ${luz.value}", fontSize = 16.sp)

                HorizontalDivider(modifier =
Modifier.padding(vertical = 12.dp))

                Text("⚙️ Estados del Sistema", fontSize =
18.sp, fontWeight = FontWeight.Bold)
                Spacer(modifier = Modifier.height(8.dp))
                Text("🔆 Control por Luz: ${if
(controlLuz.value) "Activado" else "Desactivado"}", fontSize
= 16.sp)
                Text("🌡️ Control por Temp: ${if
(controlTemp.value) "Activado" else "Desactivado"}", fontSize
= 16.sp)
                Text("🔔 Alertas: ${if
(alertasActivadas.value) "Activas" else "Inactivas"}",
fontSize = 16.sp)
            }
        }

        Column {
```

```
            Button(
                onClick = onSettingsClick,
                modifier = Modifier
                    .fillMaxWidth()
                    .height(50.dp)
            ) {
                Text("⚙ Configuración", fontSize = 18.sp)
            }

            Spacer(modifier = Modifier.height(12.dp))

            Button(
                onClick = onLogout,
                colors =
ButtonDefaults.buttonColors(containerColor =
MaterialTheme.colorScheme.error),
                modifier = Modifier
                    .fillMaxWidth()
                    .height(50.dp)
            ) {
                Text("Cerrar Sesión", fontSize = 18.sp, color
= MaterialTheme.colorScheme.onError)
            }
        }
    }

    LaunchedEffect(isSending, estado.value) {
        if (isSending) {
            delay(1000)
            isSending = false
        }
    }
}
```

MainActivity.kt

```
package com.example.enchufe2

import android.app.NotificationChannel
import android.app.NotificationManager
import android.content.Context
import android.os.Build
import android.os.Bundle
import androidx.activity.ComponentActivity
import androidx.activity.compose.setContent
import androidx.activity.enableEdgeToEdge
import androidx.compose.runtime.*
import androidx.core.app.NotificationCompat
import androidx.core.app.NotificationManagerCompat
import com.example.enchufe2.mqtt.MQTTManager
import com.example.enchufe2.screens.*
import com.example.enchufe2.ui.theme.Enchufe2Theme

class MainActivity : ComponentActivity() {

    private val ALERT_CHANNEL_ID = "sensor_alert_channel"
    private val ALERT_NOTIFICATION_ID = 1001

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        enableEdgeToEdge()

        // ✅ Solicitar permiso de notificaciones (Android
13+)
        if (Build.VERSION.SDK_INT >=
Build.VERSION_CODES.TIRAMISU) {
```

```
requestPermissions(arrayOf(android.Manifest.permission.POST_NOTIFICATIONS), 100)
        }

        createNotificationChannel()

        setContent {
            Enchufe2Theme {
                val mqttManager = remember { MQTTManager() }

                var isUserLoggedIn by remember { mutableStateOf(false) }
                var isAdminLoggedIn by remember { mutableStateOf(false) }
                var isInSettings by remember { mutableStateOf(false) }

                val estado = remember { mutableStateOf("Desconectado") }
                val temperatura = remember { mutableStateOf("Cargando...") }
                val humedad = remember { mutableStateOf("Cargando...") }
                val luz = remember { mutableStateOf("Cargando...") }
                val timer = remember { mutableStateOf("0 min") }
                val controlLuz = remember { mutableStateOf(false) }
                val controlTemp = remember { mutableStateOf(false) }
```

```
                val alertasActivadas = remember {
mutableStateOf(true) }

                // 🔌 Conexión MQTT
                LaunchedEffect(Unit) {
                    if (!mqttManager.isConnected()) {
                        mqttManager.connect(
                            estado, temperatura, humedad, luz,
timer, controlLuz, controlTemp
                        )
                    }
                }

                // 🔔 Verificar alertas
                LaunchedEffect(temperatura.value,
humedad.value, alertasActivadas.value) {
                    if (!alertasActivadas.value)
return@LaunchedEffect

                    val tempValue =
temperatura.value.replace("°C", "").toFloatOrNull()
                    val humedadValue =
humedad.value.replace("%", "").toFloatOrNull()

                    if (tempValue != null && tempValue > 35f)
{
                        showNotification("⚠️ Alerta de
Temperatura", "La temperatura es muy alta: $tempValue°C")
                    }

                    if (humedadValue != null && humedadValue <
30f) {
```

```
                        showNotification("⚠️ Alerta de
Humedad", "La humedad está muy baja: $humedadValue%")
                    }
                }

                // Navegación
                when {
                    isAdminLoggedIn -> {
                        SettingsScreen(
                            onBack = { isAdminLoggedIn = false
},
                            mqttManager = mqttManager,
                            controlLuz = controlLuz,
                            controlTemp = controlTemp,
                            alertasActivadas =
alertasActivadas
                        )
                    }

                    isUserLoggedIn -> {
                        if (isInSettings) {
                            SettingsScreen(
                                onBack = { isInSettings =
false },
                                mqttManager = mqttManager,
                                controlLuz = controlLuz,
                                controlTemp = controlTemp,
                                alertasActivadas =
alertasActivadas
                            )
                        } else {
                            EnchufeControlScreen(
```

```
                                onLogout = { isUserLoggedIn =
false },
                                onSettingsClick = {
isInSettings = true },
                                mqttManager = mqttManager,
                                estado = estado,
                                temperatura = temperatura,
                                humedad = humedad,
                                luz = luz,
                                timer = timer,
                                controlLuz = controlLuz,
                                controlTemp = controlTemp,
                                alertasActivadas =
alertasActivadas
                            )
                        }
                    }

                    else -> {
                        LoginScreen(
                            onUserLoginSuccess = {
isUserLoggedIn = true },
                            onAdminLoginSuccess = {
isAdminLoggedIn = true }
                        )
                    }
                }
            }
        }
    }

    // 🔔 Crear canal de notificación
    private fun createNotificationChannel() {
```

```
        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
            val name = "Sensor Alerts"
            val descriptionText = "Canal para alertas de
temperatura y humedad"
            val importance =
NotificationManager.IMPORTANCE_HIGH
            val channel =
NotificationChannel(ALERT_CHANNEL_ID, name, importance).apply
{
                description = descriptionText
            }
            val notificationManager: NotificationManager =
                getSystemService(Context.NOTIFICATION_SERVICE)
as NotificationManager

notificationManager.createNotificationChannel(channel)
        }
    }

    // 🚨 Mostrar notificación
    private fun showNotification(title: String, message:
String) {
        // Verificar permiso de notificación (Android 13+)
        if (Build.VERSION.SDK_INT >=
Build.VERSION_CODES.TIRAMISU) {
            val permissionCheck =
checkSelfPermission(android.Manifest.permission.POST_NOTIFICA
TIONS)
            if (permissionCheck !=
android.content.pm.PackageManager.PERMISSION_GRANTED) {
                return // Salir sin mostrar la notificación si
no se tiene permiso
            }
```

```
        }

        val builder = NotificationCompat.Builder(this,
ALERT_CHANNEL_ID)
            .setSmallIcon(R.drawable.ic_launcher_foreground)
            .setContentTitle(title)
            .setContentText(message)
            .setPriority(NotificationCompat.PRIORITY_HIGH)

        with(NotificationManagerCompat.from(this)) {
            notify(ALERT_NOTIFICATION_ID, builder.build())
        }
    }

}
```

LoginScreen.kt

```
package com.example.enchufe2.screens

import androidx.compose.foundation.layout.*
import androidx.compose.material3.*
import androidx.compose.runtime.*
import androidx.compose.ui.Modifier
import androidx.compose.ui.text.input.*
import androidx.compose.ui.unit.dp
import androidx.compose.ui.unit.sp

@Composable
fun LoginScreen(
    onUserLoginSuccess: () -> Unit,
    onAdminLoginSuccess: () -> Unit
) {
    var username by remember { mutableStateOf("") }
```

```
    var password by remember { mutableStateOf("") }
    var errorMessage by remember { mutableStateOf("") }

    Column(
        modifier = Modifier
            .fillMaxSize()
            .padding(24.dp),
        verticalArrangement = Arrangement.Center
    ) {
        Text("🔐 Iniciar Sesión", fontSize = 26.sp)

        Spacer(modifier = Modifier.height(24.dp))

        OutlinedTextField(
            value = username,
            onValueChange = {
                username = it
                errorMessage = ""
            },
            label = { Text("Usuario") },
            modifier = Modifier.fillMaxWidth()
        )

        Spacer(modifier = Modifier.height(16.dp))

        OutlinedTextField(
            value = password,
            onValueChange = {
                password = it
                errorMessage = ""
            },
            label = { Text("Contraseña") },
```

```
            visualTransformation =
PasswordVisualTransformation(),
            modifier = Modifier.fillMaxWidth()
        )

        Spacer(modifier = Modifier.height(24.dp))

        Button(
            onClick = {
                if (username == "usuario" && password ==
"1234") {
                    onUserLoginSuccess()
                } else {
                    errorMessage = "Credenciales de usuario
incorrectas"
                }
            },
            modifier = Modifier.fillMaxWidth()
        ) {
            Text("Entrar como Usuario")
        }

        Spacer(modifier = Modifier.height(12.dp))

        Button(
            onClick = {
                if (username == "cuidador" && password ==
"admin") {
                    onAdminLoginSuccess()
                } else {
                    errorMessage = "Credenciales de cuidador
incorrectas"
                }
```

```
            },
            modifier = Modifier.fillMaxWidth()
        ) {
            Text("Entrar como Cuidador")
        }

        if (errorMessage.isNotEmpty()) {
            Spacer(modifier = Modifier.height(16.dp))
            Text(
                text = errorMessage,
                color = MaterialTheme.colorScheme.error,
                fontSize = 14.sp
            )
        }
    }
}
```

Codigo arduino IDE:

```
#include <ESP8266WiFi.h>
#include <PubSubClient.h>
#include <DHT.h>

//Pines
#define RELAY_PIN D1
#define LDR_PIN A0
#define DHT_PIN D2
#define DHTTYPE DHT22

DHT dht(DHT_PIN, DHTTYPE);

//  WiFi
const char* ssid = "HC";
const char* password = "Carmiso123";
```

```
//  MQTT
const char* mqtt_server = "io.adafruit.com";
const int mqtt_port = 1883;
const char* mqtt_user = "nicohc";
const char* mqtt_pass = "aio_KOQz9077t0pnepBgwcsibIrWSALK";

WiFiClient espClient;
PubSubClient client(espClient);

// Feeds
const char* feed_control = "nicohc/feeds/enchufe-dot-control";
const char* feed_status = "nicohc/feeds/enchufe-dot-status";
const char* feed_temp = "nicohc/feeds/enchufe-dot-temperatura";
const char* feed_humedad = "nicohc/feeds/enchufe-dot-humedad";
const char* feed_luz = "nicohc/feeds/enchufe-dot-luz";
const char* feed_luz_auto = "nicohc/feeds/enchufe-dot-luz_automatico";
const char* feed_temp_auto =
"nicohc/feeds/enchufe-dot-temp_automatico";
const char* feed_temp_max = "nicohc/feeds/enchufe-dot-temp_max";
const char* feed_temp_min = "nicohc/feeds/enchufe-dot-temp_min";
const char* feed_timer = "nicohc/feeds/enchufe-dot-timer";

//  Variables
bool enchufe_encendido = false;
bool control_luz_activado = false;
bool control_temp_activado = false;
bool timer_activo = false;
int umbral_luz = 500;
float temp_max = 30.0;
float temp_min = 25.0;
int timer_minutes = 0;
unsigned long timer_start = 0;

//  Función para cambiar el estado del enchufe y publicarlo si cambia
void setEstadoEnchufe(bool encender, const char* motivo) {
  if (enchufe_encendido != encender) {
    enchufe_encendido = encender;
    digitalWrite(RELAY_PIN, encender ? HIGH : LOW);
    client.publish(feed_status, motivo, true);
  }
}
```

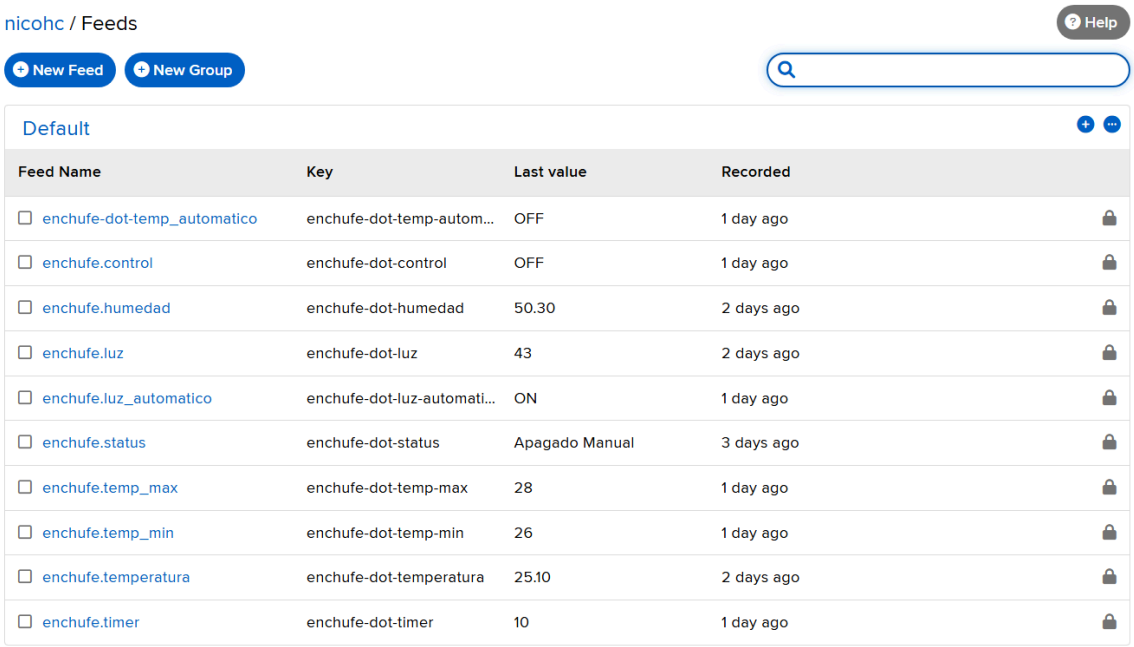
```
//  Callback de mensajes MQTT
void mqttCallback(char* topic, byte* payload, unsigned int length) {
  String message;
  for (unsigned int i = 0; i < length; i++) message +=
(char)payload[i];

  Serial.print("📩 "); Serial.print(topic); Serial.print(": ");
Serial.println(message);

  if (String(topic) == feed_control) {
    if (message == "ON") {
      timer_activo = false;
      setEstadoEnchufe(true, "Encendido Manual");
    } else if (message == "OFF") {
      timer_activo = false;
      setEstadoEnchufe(false, "Apagado Manual");
    }
  } else if (String(topic) == feed_luz_auto) {
    control_luz_activado = (message == "ON");
    if (control_luz_activado) {
      control_temp_activado = false;
      timer_activo = false;
    }
  } else if (String(topic) == feed_temp_auto) {
    control_temp_activado = (message == "ON");
    if (control_temp_activado) {
      control_luz_activado = false;
      timer_activo = false;
    }
  } else if (String(topic) == feed_temp_max) {
    temp_max = message.toFloat();
  } else if (String(topic) == feed_temp_min) {
    temp_min = message.toFloat();
  } else if (String(topic) == feed_timer) {
    timer_minutes = message.toInt();
    if (timer_minutes > 0) {
      control_luz_activado = false;
      control_temp_activado = false;
      timer_activo = true;
      timer_start = millis();
```

```
      setEstadoEnchufe(true, "Encendido por temporizador");
    }
  }
}

//  Conexión WiFi
void setup_wifi() {
  WiFi.mode(WIFI_STA);
  WiFi.begin(ssid, password);
  Serial.print("🔗 Conectando WiFi...");
  int intentos = 0;
  while (WiFi.status() != WL_CONNECTED && intentos < 30) {
    delay(500);
    Serial.print(".");
    intentos++;
  }
  Serial.println(WiFi.status() == WL_CONNECTED ? "\n✅ Conectado" :
"\n❌ No se pudo conectar");
}

// 📡 Reconexión MQTT
void reconnectMQTT() {
  if (!client.connected()) {
    Serial.print("🔗 Conectando MQTT...");
    if (client.connect("WemosD1Mini", mqtt_user, mqtt_pass)) {
      Serial.println("✅ MQTT Conectado");
      client.subscribe(feed_control);
      client.subscribe(feed_luz_auto);
      client.subscribe(feed_temp_auto);
      client.subscribe(feed_temp_max);
      client.subscribe(feed_temp_min);
      client.subscribe(feed_timer);

      // Publicar el estado actual
      client.publish(feed_status, enchufe_encendido ? "Encendido
Manual" : "Apagado Manual", true);
    } else {
      Serial.print("❌ MQTT Error: ");
      Serial.println(client.state());
    }
  }
```

```
}

void setup() {
  Serial.begin(115200);
  dht.begin();
  pinMode(RELAY_PIN, OUTPUT);
  digitalWrite(RELAY_PIN, LOW);
  setup_wifi();
  client.setServer(mqtt_server, mqtt_port);
  client.setCallback(mqttCallback);
}

void loop() {
  reconnectMQTT();
  client.loop();

  float temp = dht.readTemperature();
  float hum = dht.readHumidity();
  int luz = analogRead(LDR_PIN);

  if (!isnan(temp) && !isnan(hum)) {
    client.publish(feed_temp, String(temp).c_str(), true);
    client.publish(feed_humedad, String(hum).c_str(), true);
    client.publish(feed_luz, String(luz).c_str(), true);

    if (control_luz_activado) {
      if (luz < umbral_luz) {
        setEstadoEnchufe(true, "Encendido por luz");
      } else {
        setEstadoEnchufe(false, "Apagado por luz");
      }
    }

    if (control_temp_activado) {
      if (temp >= temp_max) {
        setEstadoEnchufe(true, "Encendido por temperatura");
      } else if (temp <= temp_min) {
        setEstadoEnchufe(false, "Apagado por temperatura");
      }
    }
```

```
    if (timer_activo && enchufe_encendido) {
      unsigned long elapsed = (millis() - timer_start) / 60000;
      if (elapsed >= timer_minutes) {
        timer_activo = false;
        setEstadoEnchufe(false, "Apagado por timer");
      }
    }
  } else {
    Serial.println("⚠ Error leyendo sensores");
  }

  delay(10000);
}
```

Feeds registrados en el servidor:



Feed Name	Key	Last value	Recorded
enchufe-dot-temp_automatico	enchufe-dot-temp-autom...	OFF	1 day ago
enchufe.control	enchufe-dot-control	OFF	1 day ago
enchufe.humedad	enchufe-dot-humedad	50.30	2 days ago
enchufe.luz	enchufe-dot-luz	43	2 days ago
enchufe.luz_automatico	enchufe-dot-luz-automati...	ON	1 day ago
enchufe.status	enchufe-dot-status	Apagado Manual	3 days ago
enchufe.temp_max	enchufe-dot-temp-max	28	1 day ago
enchufe.temp_min	enchufe-dot-temp-min	26	1 day ago
enchufe.temperatura	enchufe-dot-temperatura	25.10	2 days ago
enchufe.timer	enchufe-dot-timer	10	1 day ago