



UNIVERSIDAD DE GRANADA

Trabajo de Fin de Grado: Diseño y Desarrollo
de Sistema de Automatismos para Maquetas
de Arquitectura Interactivas

Miguel López Martínez

Estudiante del Grado en Ingeniería Electrónica Industrial

Tutor: Francisco Javier Romero Maldonado

13 de junio de 2025

Diseño y Desarrollo de Sistema de Automatismos para Maquetas de Arquitectura Interactivas

Miguel López Martínez

Palabras clave: ESP32, Home Assistant, automatización, MQTT, sensores, integración domótica, interfaz Lovelace, Automatización, maquetas inteligentes, domótica, sensorización.

Resumen

En esta memoria se presenta el diseño e implementación de un sistema domótico distribuido que emplea microcontroladores **ESP32** como nodos periféricos y la plataforma **Home Assistant** como núcleo de gestión central. El sistema permite el control inteligente de iluminación, monitoreo de variables ambientales, apertura automática de una caja mediante motor DC y piñón-cremallera, así como la visualización de todos los eventos a través de una interfaz personalizable tipo *dashboard*.

La comunicación entre dispositivos se lleva a cabo mediante el protocolo **MQTT**, lo que permite una arquitectura flexible y escalable. Se hace uso de sensores como detectores magnéticos, sensores de luz y movimiento, y actuadores como tiras LED tipo NeoPixel y motores, todos controlados mediante tópicos MQTT definidos en la configuración modular de Home Assistant.

Además, se ha desarrollado una **maqueta interactiva** que materializa el sistema domótico implementado, permitiendo una representación física y tangible de sus funcionalidades. Este tipo de maqueta no solo sirve como demostración visual del sistema, sino que también actúa como herramienta pedagógica para la comprensión de tecnologías IoT. Las maquetas interactivas combinan elementos físicos con componentes electrónicos y digitales, permitiendo la interacción del usuario mediante sensores, botones y visualizaciones dinámicas [1, 2].

El proyecto está enfocado no solo en su funcionalidad técnica, sino también en servir como guía práctica para quienes deseen introducirse en el desarrollo de soluciones IoT domésticas con enfoque modular, local y abierto.

Design and Development of an Automation System for Interactive Architectural Models

Miguel López Martínez

Keywords: ESP32, Home Assistant, automation, MQTT, sensors, smart integration, Lovelace interface, intelligent models, home automation, sensing.

Abstract

This report presents the design and implementation of a distributed home automation system using **ESP32** microcontrollers as peripheral nodes and the **Home Assistant** platform as the central management core. The system enables intelligent lighting control, environmental variable monitoring, automatic box opening via a DC motor and rack-and-pinion mechanism, and event visualization through a customizable *dashboard* interface.

Device communication is handled via the **MQTT** protocol, allowing for a flexible and scalable architecture. The system utilizes sensors such as magnetic switches, light and motion sensors, and actuators including NeoPixel LED strips and motors, all managed through MQTT topics defined in Home Assistant's modular configuration.

Additionally, an **interactive model** was developed to physically represent the implemented home automation system, providing a tangible demonstration of its functionalities. This type of model not only serves as a visual showcase, but also as an educational tool for understanding IoT technologies. Interactive models combine physical elements with electronic and digital components, enabling user interaction through sensors, buttons, and dynamic visual feedback [1, 2].

The project focuses not only on technical functionality, but also aims to serve as a practical guide for those interested in developing modular, local, and open-source IoT solutions for smart homes.

Yo, **Miguel López Martínez**, alumno de la titulación del **Grado en Ingeniería Electrónica Industrial** de la **Facultad de Ciencias** de la **Universidad de Granada**, con DNI **77166405-W**, autorizo la ubicación de la siguiente copia de mi **Trabajo Fin de Grado** en la biblioteca del centro para que pueda ser consultada por las personas que lo deseen.

Fdo.: Miguel López Martínez

Granada, a 12 de junio de 2025

D. Francisco Javier Romero Maldonado, Profesor del **Programa Proyectos Jóvenes Investigadores Doctores** del Departamento de Electrónica y Tecnología de Computadores de la Universidad de Granada.

Informa:

Que el presente trabajo, titulado *Diseño y Desarrollo de Sistema de Automatismos para Maquetas de Arquitectura Interactivas*, ha sido realizado bajo su supervisión por el alumno **Miguel López Martínez**, y autoriza la defensa de dicho trabajo ante el tribunal que corresponda.

Y para que conste, expide y firma el presente informe en Granada a 12 de junio de 2025.

El director del TFG:

Francisco Javier Romero Maldonado

Agradecimientos

Me gustaría agradecer a mi tutor, Francisco Javier Romero Maldonado, por su tutorización y atención durante el desarrollo de este proyecto; sin él, habría sido muy difícil llevarlo a término.

Para finalizar, quiero expresar mi más profundo agradecimiento por el completo apoyo incondicional de mi familia, en especial a mis padres, quienes me han dado la motivación y la fuerza no solo para estudiar, sino para crecer como persona desde que me fui siendo un niño a estudiar fuera, hasta convertirme en la persona que soy hoy por hoy.

Índice

| | |
|---|-----------|
| Índice de Figuras | 15 |
| Índice de Tablas | 16 |
| 1. Introducción | 17 |
| 1.1. Contexto y motivación | 17 |
| 1.2. Planificación temporal del proyecto | 18 |
| 1.3. Estructura del trabajo | 19 |
| 2. Estado del arte | 21 |
| 2.1. Mercado y situación sociocultural | 21 |
| 2.2. Definición y evolución histórica | 23 |
| 2.3. Maquetas interactivas como herramienta tecnológica y educativa | 25 |
| 2.4. Evolución y estado actual de la domótica | 26 |
| 2.5. Normativa aplicable | 26 |
| 2.6. Estudio del Sistema domótico | 27 |
| 2.6.1. Dispositivos | 27 |
| 2.6.2. Arquitectura y funcionamiento | 27 |
| 2.6.3. Interfaces de usuario | 30 |
| 2.6.4. HUBs domóticos o plataformas domóticas | 30 |
| 2.6.5. Tecnologías y protocolos | 31 |
| 3. Implementacion del sistema | 32 |
| 3.1. Integración del sistema domótico sobre maqueta interactiva | 32 |
| 3.2. Tecnologías utilizadas | 33 |
| 3.3. Diagrama de bloques de la arquitectura de red | 34 |
| 3.4. Hardware | 35 |
| 3.5. Instalación de Home Assistant en Raspberry Pi | 41 |
| 3.5.1. Requisitos Previos | 41 |
| 3.5.2. Descarga e Instalación de Home Assistant OS | 42 |
| 3.5.3. Primer Arranque y Acceso Inicial | 42 |
| 3.5.4. Configuración Inicial del Sistema | 42 |
| 3.5.5. Ventajas del Home Assistant OS | 43 |
| 3.5.6. Conclusión | 43 |
| 3.6. Configuración de Red y Acceso Remoto en Home Assistant | 43 |
| 3.6.1. Asignación de IP Fija en la Red Local | 44 |
| 3.6.2. Acceso Remoto Seguro con DuckDNS | 45 |
| 3.6.3. Apertura de Puertos en el Router | 46 |
| 3.6.4. Consideraciones sobre la IP Pública y DIGI Plus | 47 |
| 3.6.5. Resumen | 47 |
| 3.7. Configuración del Broker MQTT Mosquitto en Home Assistant | 47 |
| 3.7.1. Instalación del Complemento Mosquitto | 47 |
| 3.7.2. Creación de Usuario MQTT | 48 |
| 3.7.3. Configuración del Complemento | 48 |
| 3.7.4. Puertos Configurados | 49 |
| 3.7.5. Integración con Home Assistant | 49 |
| 3.7.6. Estructura de configuration.yaml | 49 |

| | |
|---|-----------|
| 3.7.7. Automatización: API Meteorológica para Control Lumínico | 50 |
| 3.7.8. Código de Configuración MQTT | 50 |
| 3.8. Configuración del Código ESP32 | 51 |
| 3.8.1. Función <code>setup()</code> | 51 |
| 3.8.2. Función <code>loop()</code> | 52 |
| 3.8.3. Función <code>medirLux()</code> | 53 |
| 3.8.4. Función <code>detectarMovimiento()</code> | 53 |
| 3.8.5. Control de Caja Motorizada: <code>controlarCaja()</code> , <code>moverCaja()</code> y <code>detenerMotorCaja()</code> | 54 |
| 3.8.6. Función <code>callback()</code> | 55 |
| 3.8.7. Función <code>procesarMensajeJSON()</code> | 56 |
| 3.8.8. Función <code>actualizarLed()</code> | 57 |
| 3.8.9. Función <code>actualizarLed9()</code> | 58 |
| 3.8.10. Función <code>apagarTodasLasLuces()</code> | 58 |
| 3.8.11. Función <code>reconnect()</code> | 58 |
| 3.9. Interfaz de Usuario en Home Assistant (Lovelace) | 59 |
| 3.10. Configuración de Tarjetas de Entidades | 59 |
| 3.10.1. Resumen de Diseño | 60 |
| 3.11. Conexionado ESP32 | 62 |
| 3.11.1. Esquema general de conexiones | 62 |
| 3.11.2. Descripción de la maqueta y distribución de los componentes | 63 |
| 4. Presupuesto | 64 |
| 5. Conclusiones y Trabajo Futuro | 65 |
| 6. Bibliografía | 66 |
| 7. Repositorio del Proyecto | 68 |

Índice de Figuras

| | | |
|-----|---|----|
| 1. | Diagrama de Gantt del proyecto | 18 |
| 2. | Evolución del número de hogares inteligentes en el mundo (2017–2025). Fuente: Statista (2023) | 21 |
| 3. | Porcentaje de población anciana sobre población en edad de trabajar. Fuente: Banco Mundial (2017), visualización por El Orden Mundial | 22 |
| 4. | Evolución del equipamiento TIC en viviendas españolas (2012–2022). Fuente: Instituto Nacional de Estadística | 23 |
| 5. | Ilustración "metropolis del futuro "de la Feria Mundial de Nueva York (1939). Fuente: archivo histórico | 24 |
| 6. | Ejemplo de maqueta interactiva con automatismos domóticos. Fuente: Maquetas.Tech [2] | 26 |
| 7. | Ejemplo de arquitectura domótica centralizada. Fuente: Casadomo | 28 |
| 8. | Ejemplo de arquitectura domótica distribuida. Fuente: Casadomo | 28 |
| 9. | Ejemplo de arquitectura domótica jerárquica o híbrida | 29 |
| 10. | Vista general del sistema domótico implementado en la maqueta interactiva | 33 |
| 11. | Diagrama de bloques de la comunicación entre el ordenador, Home Assistant, ESP32, sensores y actuadores | 34 |
| 12. | Placa Raspberry Pi 3 utilizada como controlador central | 35 |
| 13. | Microcontrolador ESP32 usado como nodo IoT | 36 |
| 14. | Módulo puente H L298N para control de motores | 37 |
| 15. | Sensor de luz TEMT6000 para detección de luminosidad | 38 |
| 16. | Sensor magnético KY-025 para detección de posición | 39 |
| 17. | Motor DC con sistema piñón-cremallera para apertura lineal | 40 |
| 18. | Matriz de LEDs NeoPixel para iluminación RGB direccionable | 41 |
| 19. | Pantalla de configuración inicial de Home Assistant tras el primer arranque | 43 |
| 20. | Configuración de IP estática en Home Assistant | 45 |
| 21. | Configuración del DNS dinámico DuckDNS en el router Zyxel | 46 |
| 22. | Redirección de puertos configurada en el router | 46 |
| 23. | Configuración del complemento MQTT a través de la interfaz gráfica | 48 |
| 24. | Captura de pantalla de la interfaz Lovelace configurada | 60 |
| 25. | Visualización de sensor luz | 61 |
| 26. | Automatizaciones activas como el modo día/noche configuradas mediante la API de Home Assistant | 61 |
| 27. | Esquema de conexión entre el ESP32 y los diferentes sensores y actuadores | 62 |
| 28. | Vista general de la maqueta domotizada desarrollada | 63 |

Índice de Tablas

| | | |
|----|--|----|
| 1. | Comparativa resumida de plataformas domóticas. | 31 |
| 2. | Presupuesto detallado del TFG | 64 |

1. Introducción

En los últimos años, el desarrollo de tecnologías relacionadas con el Internet de las Cosas (IoT) ha revolucionado la manera en que interactuamos con nuestro entorno. La capacidad de conectar dispositivos físicos a redes digitales ha permitido la automatización inteligente de tareas cotidianas, transformando sectores como la industria, la domótica, la salud o la educación.

Dentro de este contexto, las **maquetas interactivas** se presentan como una herramienta valiosa tanto para la visualización de proyectos como para la enseñanza práctica de tecnologías emergentes. Estas maquetas combinan representaciones físicas —como edificios, habitaciones o infraestructuras— con componentes electrónicos, sensores y actuadores, permitiendo simular el funcionamiento de sistemas reales en entornos a escala [1].

Una de las principales ventajas de las maquetas interactivas es su capacidad para ilustrar procesos complejos de forma tangible, convirtiéndolas en recursos altamente eficaces para la formación, el diseño técnico y la divulgación científica. Su aplicación abarca desde presentaciones de proyectos arquitectónicos hasta entornos de prueba para automatización e integración de sistemas inteligentes [2].

Este Trabajo Fin de Grado propone el diseño e implementación de un sistema domótico distribuido, modular y abierto, basado en microcontroladores **ESP32** y gestionado mediante la plataforma **Home Assistant**. La solución incluye funcionalidades de control de iluminación, apertura automatizada de elementos, sensorización ambiental y una interfaz gráfica configurable que facilita su uso y visualización.

El objetivo principal de este proyecto es desarrollar una maqueta interactiva que sirva como demostración tecnológica y como recurso educativo para estudiantes e interesados en el ámbito de la automatización y las tecnologías IoT. Además, se busca documentar y estructurar el desarrollo para que pueda ser replicado y escalado fácilmente en otros entornos.

A continuación, se presenta el contexto en el que se enmarca este trabajo, así como las motivaciones que impulsan su realización.

1.1. Contexto y motivación

En la actualidad, la automatización del entorno doméstico, también conocida como domótica, ha adquirido un protagonismo creciente gracias al auge de las tecnologías del Internet de las Cosas (IoT). Estas tecnologías permiten interconectar dispositivos electrónicos del hogar, facilitando su control remoto y automatizado a través de plataformas digitales.

El concepto de hogar inteligente ha pasado de ser una visión futurista a una realidad cada vez más presente en los hogares modernos. Según datos recientes, el gasto mundial en soluciones de *smart home* ha experimentado un crecimiento sostenido durante la última década [5], al igual que el número total de hogares que incorporan este tipo de tecnologías [6].

Este avance ha sido acompañado por un creciente interés institucional y empresarial en fomentar la domótica como herramienta clave para mejorar la eficiencia energética, la accesibilidad y la seguridad en las viviendas [7]. Asimismo, en el contexto español, las estadísticas reflejan un aumento progresivo en la adopción de tecnologías TIC dentro de los hogares, lo que demuestra una base tecnológica cada vez más preparada para la integración de soluciones domóticas [8].

Paralelamente, las **maquetas interactivas** se han consolidado como un recurso de alto valor didáctico y demostrativo, especialmente en áreas como la arquitectura, la ingeniería y la automatización. Estas maquetas permiten representar en pequeña escala sistemas complejos, integrando sensores, actuadores, microcontroladores y plataformas de visualización para simular entornos reales. Su uso resulta especialmente eficaz en la formación técnica, ya que permite comprender el funcionamiento de sistemas físicos mediante la interacción directa [1, 2].

Además, las maquetas interactivas ofrecen un entorno controlado ideal para experimentar con configuraciones IoT sin los riesgos ni los costes asociados a instalaciones reales. En este sentido, funcionan como laboratorios de pruebas accesibles, que facilitan tanto la fase de diseño como la de validación de soluciones tecnológicas.

La motivación principal de este trabajo surge, por tanto, de la confluencia de dos tendencias clave: el avance sostenido de la domótica como solución real y práctica, y la creciente utilidad de las maquetas interactivas como medio de simulación, aprendizaje y demostración. Desarrollar un sistema domótico funcional y replicable sobre una maqueta interactiva permite no solo validar conceptos técnicos, sino también acercar la tecnología a entornos educativos y divulgativos.

1.2. Planificación temporal del proyecto

El desarrollo del presente Trabajo Fin de Grado ha seguido una planificación secuencial distribuida a lo largo de cinco meses, desde febrero hasta junio. Para organizar las diferentes fases del proyecto, se elaboró un diagrama de Gantt que permitiera visualizar con claridad la estructura temporal de las tareas, su solapamiento y su duración relativa (véase Figura 1).

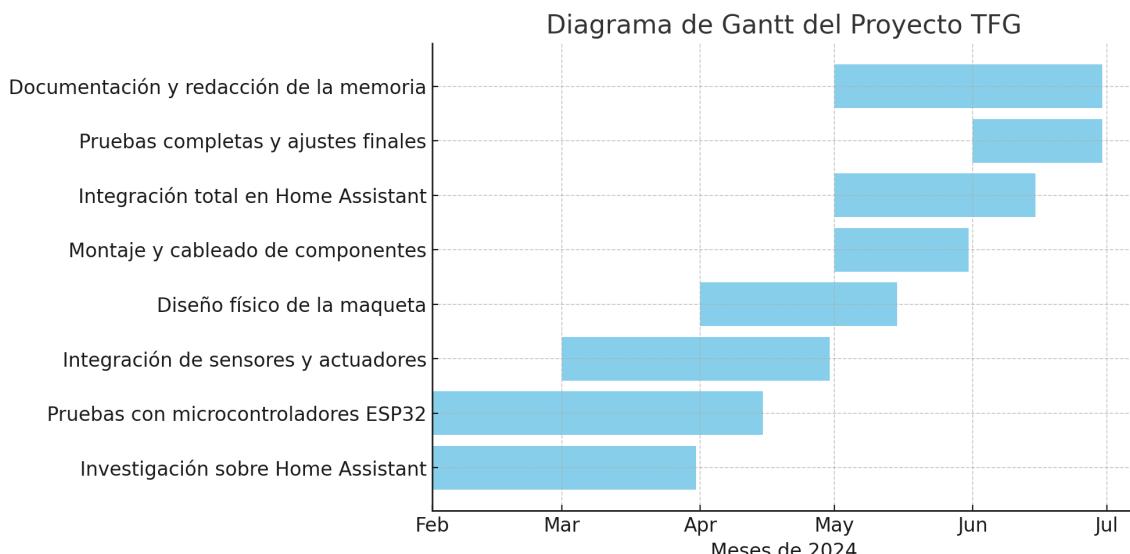


Figura 1: Diagrama de Gantt del proyecto

El proyecto se inició en el mes de febrero con una etapa centrada en la **investigación y análisis de la plataforma Home Assistant**, evaluando su arquitectura, compatibilidad con dispositivos y posibilidades de integración. Esta fase se solapó con las primeras **pruebas prácticas sobre microcontroladores ESP32**, incluyendo su configuración y comunicación con el sistema central mediante el protocolo MQTT.

Durante los meses de marzo y abril se abordó la **integración de sensores y actuadores**, así como el diseño lógico de los flujos de automatización. Paralelamente, se dio inicio al **diseño físico de la maqueta interactiva**, definiendo su distribución, materiales y compartimentación para alojar los distintos elementos electrónicos.

En mayo se ejecutó la **fase de montaje**, que incluyó el ensamblado de la maqueta, el cableado interno, la organización de componentes y su conexión al sistema domótico. También se trabajó en la **integración completa con Home Assistant**, de modo que la maqueta pudiera controlarse de forma centralizada y mostrara su estado en tiempo real mediante la interfaz Lovelace.

Finalmente, el mes de junio estuvo dedicado principalmente a las **pruebas globales del sistema**, ajustes y correcciones funcionales, así como a la **redacción y finalización de la memoria**.

Esta planificación permitió abordar el proyecto de forma estructurada y progresiva, facilitando una gestión eficiente del tiempo y de los recursos disponibles.

1.3. Estructura del trabajo

El presente Trabajo de Fin de Grado se estructura en varios capítulos que reflejan las distintas fases del desarrollo e implementación del sistema domótico sobre maqueta. A continuación, se describe brevemente el contenido de cada uno de ellos:

- **Instalación de Home Assistant en Raspberry Pi:** se detalla el proceso de preparación del entorno base, desde los requisitos iniciales hasta la puesta en marcha del sistema operativo Home Assistant OS.
- **Configuración de red y acceso remoto:** se explican los pasos realizados para garantizar la conectividad local y remota del sistema, incluyendo el uso de DuckDNS, redirección de puertos y solución a las limitaciones de IP pública.
- **Configuración del broker MQTT:** se describe la instalación y ajuste del complemento Mosquitto, así como la integración de la comunicación entre dispositivos usando el protocolo MQTT.
- **Modularización y automatización en YAML:** se presenta la reestructuración del archivo `configuration.yaml` y la implementación de automatismos como el control lumínico mediante información meteorológica.
- **Programación del microcontrolador ESP32:** se explica la lógica embebida implementada en el ESP32, responsable del control de sensores, LEDs y motores mediante mensajes MQTT.
- **Montaje de la maqueta interactiva:** se documenta el proceso constructivo de la maqueta física, desde el diseño de su estructura hasta el ensamblaje de los distintos módulos. Se incluyen detalles sobre el cableado, integración de componentes electrónicos, organización del espacio interno y su adecuación para representar una vivienda inteligente funcional a escala.
- **Interfaz gráfica del usuario:** se expone el diseño de la interfaz Lovelace en Home Assistant, incluyendo la agrupación de entidades y la personalización del entorno visual.

- **Descripción del hardware:** se analiza cada uno de los componentes utilizados, justificando su elección y papel dentro del sistema, desde la Raspberry Pi hasta los sensores y actuadores.
- **Presupuesto y referencias:** se incluye el coste detallado del proyecto y la bibliografía técnica consultada para su desarrollo.

Esta organización permite al lector comprender de forma progresiva el desarrollo del sistema, desde la base tecnológica hasta la integración completa de todos los elementos que conforman la maqueta domótica.

2. Estado del arte

2.1. Mercado y situación sociocultural

El mercado de la domótica ha experimentado un crecimiento exponencial en los últimos años, impulsado por el desarrollo de tecnologías IoT (Internet of Things), el abaratamiento de microcontroladores y el auge de plataformas de código abierto como Home Assistant. Según diversos estudios, la adopción de soluciones de automatización en el hogar responde tanto a motivos de eficiencia energética como a la búsqueda de mayor seguridad, comodidad y accesibilidad.

Desde una perspectiva sociocultural, la digitalización del entorno doméstico refleja un cambio en los hábitos de vida. La sociedad actual demanda entornos inteligentes que se adapten a sus rutinas y que permitan un control remoto, especialmente en un contexto donde el teletrabajo y la conectividad constante han cobrado protagonismo. Además, existe un creciente interés por soluciones sostenibles que reduzcan el consumo energético y que, al mismo tiempo, mejoren la calidad de vida.

En este escenario, proyectos como el desarrollado en este Trabajo de Fin de Grado no solo son técnicamente viables, sino también culturalmente pertinentes. El uso de tecnologías accesibles, como la Raspberry Pi y el ESP32, permite democratizar el acceso a la automatización, ampliando su alcance a sectores educativos, domésticos e incluso profesionales de bajo presupuesto.

Así, el proyecto se enmarca en una tendencia global hacia la creación de espacios inteligentes, eficientes y adaptativos, alineándose con las necesidades actuales de la sociedad y con el rumbo del mercado tecnológico.

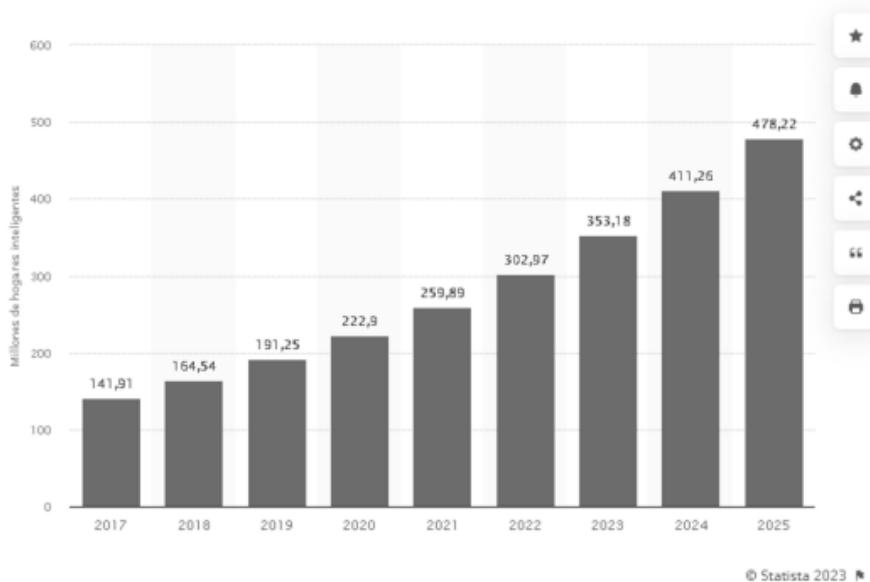


Figura 2: Evolución del número de hogares inteligentes en el mundo (2017–2025). Fuente: Statista (2023).

Como muestra la Figura 1, se prevé que el número de hogares inteligentes pase de 141,91 millones en 2017 a más de 478 millones en 2025. Este crecimiento evidencia una fuerte demanda por sistemas que mejoren la eficiencia, seguridad y confort en el hogar.

En paralelo, la sociedad enfrenta desafíos demográficos importantes, especialmente en países desarrollados. El envejecimiento de la población implica una creciente necesidad de

tecnologías que faciliten la vida diaria a personas mayores, como asistentes automatizados, sensores de actividad o alertas remotas para cuidadores.

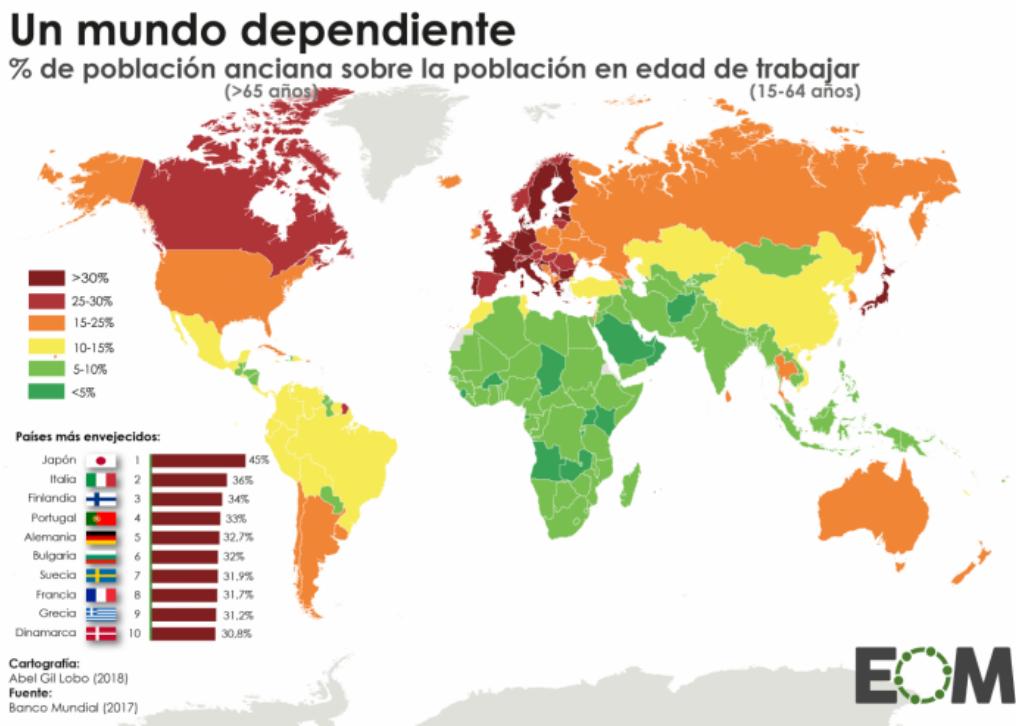


Figura 3: Porcentaje de población anciana sobre población en edad de trabajar. Fuente: Banco Mundial (2017), visualización por El Orden Mundial.

La Figura 2 evidencia que países como Japón, Italia y Alemania tienen ya más de un 30 % de su población en edad avanzada. Esto refuerza la importancia de soluciones domóticas que permitan a estas personas mantener su autonomía y calidad de vida, especialmente en entornos urbanos densamente poblados.

Por otro lado, el contexto tecnológico también es favorable. En España, por ejemplo, el acceso a tecnologías de la información en los hogares ha mejorado significativamente en la última década. El número de viviendas con conexión de banda ancha ha superado el 90 %, y la mayoría dispone de algún tipo de ordenador o dispositivo conectado.

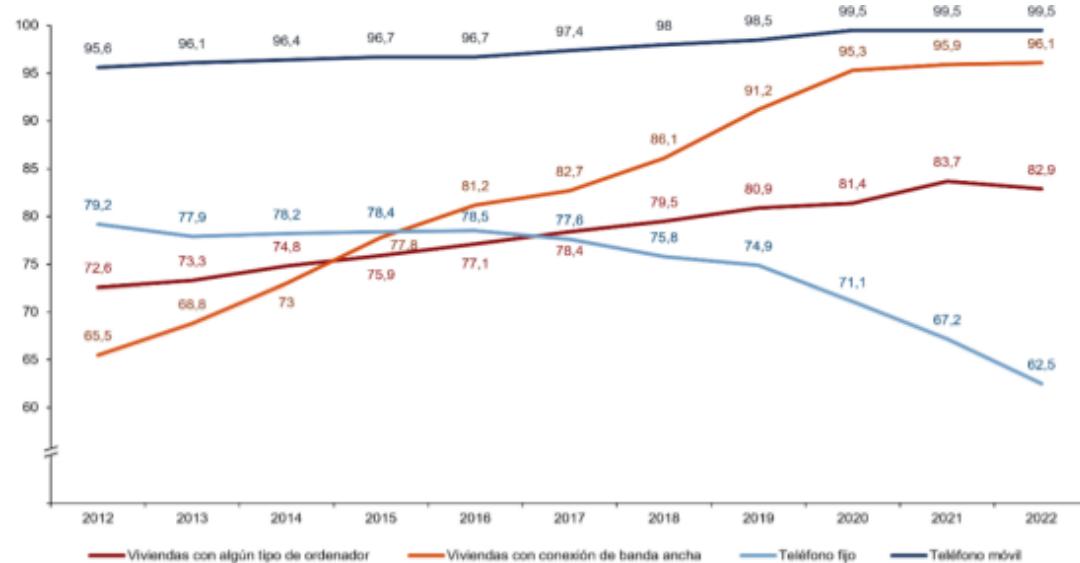


Figura 4: Evolución del equipamiento TIC en viviendas españolas (2012–2022). Fuente: Instituto Nacional de Estadística.

La Figura 3 muestra cómo la infraestructura tecnológica doméstica ha evolucionado favorablemente. Esta base tecnológica facilita la adopción de sistemas de automatización del hogar, haciendo que proyectos como el presente Trabajo de Fin de Grado sean no solo técnicamente viables, sino también relevantes y oportunos desde una perspectiva social y cultural.

En conjunto, el contexto sociocultural y demográfico, sumado al avance en infraestructura digital y la madurez del mercado, constituye un terreno fértil para la expansión de soluciones domóticas accesibles, sostenibles y adaptadas a las nuevas necesidades de la población.

2.2. Definición y evolución histórica

La domótica se define como el conjunto de tecnologías aplicadas al control y la automatización inteligente del hogar. Su objetivo es mejorar la eficiencia energética, la seguridad, el confort y la accesibilidad de las viviendas, mediante la integración de sistemas electrónicos y digitales que interactúan entre sí y con el usuario.

El término proviene del francés *domotique*, una combinación de *domus* (hogar, en latín) y *automatique* (automático), y comenzó a utilizarse a finales del siglo XX con la aparición de los primeros sistemas de automatización residencial.

En sus inicios, la domótica estaba limitada a instalaciones costosas, con sistemas cableados y controladores centralizados que requerían conocimientos técnicos especializados. Estas soluciones eran propias de viviendas de lujo o edificios corporativos.

Con la irrupción de las tecnologías inalámbricas, el desarrollo de sensores inteligentes, y sobre todo la expansión del Internet de las Cosas (IoT), la domótica ha evolucionado hacia modelos más flexibles, asequibles y escalables. La introducción de plataformas de código abierto como Home Assistant, junto con dispositivos de bajo coste como la Raspberry Pi y el ESP32, ha democratizado su acceso y fomentado su expansión tanto en entornos domésticos como educativos.

Hoy en día, la domótica no solo permite automatizar tareas rutinarias, como el encendido de luces o la regulación de la climatización, sino que también facilita la integración de servicios externos (por ejemplo, predicción meteorológica o asistentes virtuales) y el desarrollo de soluciones personalizadas adaptadas al estilo de vida del usuario.

Este proceso de evolución ha sido paralelo al cambio de paradigma en la relación entre el ser humano y la tecnología, donde el hogar se convierte en un entorno interactivo, capaz de aprender, adaptarse y responder de manera autónoma.

Orígenes. Las primeras ideas relacionadas con la automatización se remontan al siglo XX, con inventos como el termostato (1906) y los primeros sistemas de control remoto por cable en la década de 1930. Ya en la Feria Mundial de Nueva York de 1939, se exhibieron conceptos de “casas del futuro”, aunque más orientadas al diseño que a la tecnología funcional.



Figura 5: Ilustración "metropolis del futuro" de la Feria Mundial de Nueva York (1939).
Fuente: archivo histórico.

Década de 1960–70. En los años 60 aparecieron los primeros electrodomésticos programables, como lavadoras o hornos, marcando el inicio de la automatización doméstica a pequeña escala. En 1966, el ingeniero estadounidense Jim Sutherland desarrolló el ‘Echo IV’, un ordenador para el hogar que podía controlar luces, temperatura y almacenar listas de compras, aunque nunca llegó al mercado.

Década de 1980–90. La domótica tomó forma con la estandarización del protocolo X10 (1975), que permitía controlar dispositivos eléctricos a través del cableado eléctrico existente. Durante los años 80 y 90 se comenzaron a ver los primeros sistemas domóticos comerciales, aunque seguían siendo costosos y poco flexibles.

Siglo XXI. A partir del año 2000, con la expansión de Internet y el desarrollo de la conectividad Wi-Fi, surgieron nuevas posibilidades para la automatización del hogar. La

llegada de los smartphones permitió el control remoto de dispositivos, y con la aparición de plataformas como Arduino (2005), Raspberry Pi (2012) y posteriormente Home Assistant, se popularizó el uso de soluciones de código abierto y bajo coste.

Actualidad. Hoy en día, la domótica ha evolucionado hacia modelos descentralizados, modulares y accesibles. Tecnologías como el Internet de las Cosas (IoT), la inteligencia artificial y el aprendizaje automático permiten crear entornos domésticos que aprenden del comportamiento del usuario, optimizan recursos y ofrecen una experiencia personalizada. La domótica ha dejado de ser exclusiva de viviendas de alta gama para integrarse en hogares comunes, con soluciones escalables y adaptables.

2.3. Maquetas interactivas como herramienta tecnológica y educativa

En paralelo al desarrollo de la domótica y el Internet de las Cosas, han surgido herramientas que permiten visualizar e implementar soluciones tecnológicas a pequeña escala. Entre ellas, las **maquetas interactivas** se han consolidado como una alternativa eficaz para simular entornos reales en contextos de aprendizaje, investigación y presentación de proyectos.

Estas maquetas permiten la representación física de espacios, edificios o viviendas, integrando componentes electrónicos tales como sensores de presencia, iluminación LED, motores, pulsadores y microcontroladores. Combinadas con plataformas de automatización como Home Assistant, permiten emular un sistema domótico real en formato reducido, facilitando la interacción, la experimentación segura y la comprensión de los flujos de información entre dispositivos.

Según IAmAnufacturing [1], las maquetas interactivas pueden clasificarse en distintos tipos según su nivel de integración tecnológica: desde maquetas que solo representan estructuras físicas, hasta aquellas con retroalimentación en tiempo real, conectividad inalámbrica y visualización de eventos vía dashboards. Además de su uso en arquitectura, también se emplean en la formación técnica, ferias de divulgación y prototipado funcional de soluciones domóticas.

Por ejemplo, en [2] se destacan casos reales de maquetas funcionales utilizadas para representar sistemas de control lumínico, apertura automática de puertas, detección de presencia y simulación de eventos ambientales. Estas soluciones aportan no solo valor técnico, sino también un componente visual y tangible que favorece la comunicación de ideas complejas a públicos no técnicos.

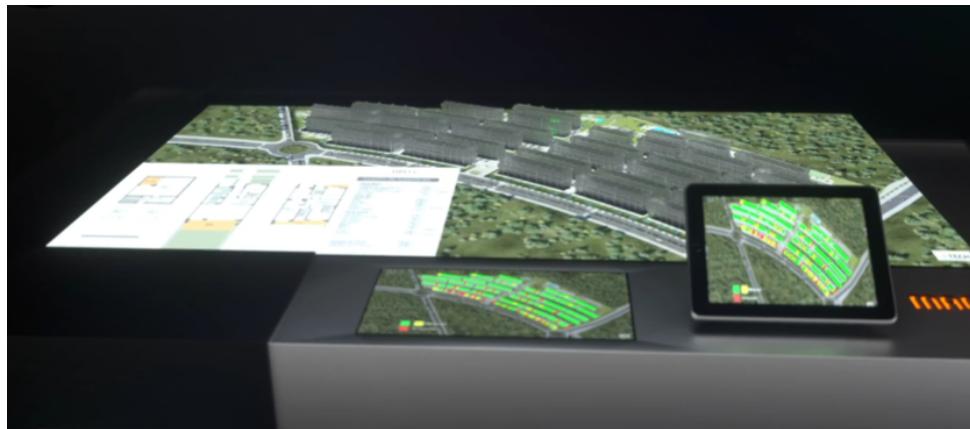


Figura 6: Ejemplo de maqueta interactiva con automatismos domóticos. Fuente: Maquetas.Tech [2]

Su aplicación en el contexto educativo y de proyectos de fin de grado es especialmente valiosa, ya que permite al alumno enfrentarse a retos reales de integración tecnológica, programación embebida, sensorización y diseño físico, consolidando conocimientos de forma transversal y práctica.

2.4. Evolución y estado actual de la domótica

Como se ha planteado en estudios anteriores, la evolución de la domótica depende tanto de la disponibilidad de tecnologías eficientes como de la aceptación y uso por parte del usuario final. Diversos autores coinciden en que el progreso en este campo no solo ha estado impulsado por la innovación técnica, sino también por factores socioculturales y económicos [13–15].

Las primeras iniciativas se centraban principalmente en la automatización de tareas domésticas básicas, mientras que en la actualidad se integran conceptos de eficiencia energética, accesibilidad, seguridad, y conectividad IoT. Este cambio de paradigma ha llevado a una redefinición del concepto de “hogar inteligente”, evolucionando hacia un entorno personalizado, adaptativo y conectado.

2.5. Normativa aplicable

El diseño y desarrollo de la solución domótica propuesta se ha realizado teniendo en cuenta la normativa vigente en materia de eficiencia energética, automatización de edificios y digitalización del entorno construido.

En el contexto europeo, se ha seguido la **Directiva 2018/844/UE**, que establece medidas destinadas a fomentar la eficiencia energética y la incorporación de sistemas automatizados de control en edificios tanto residenciales como terciarios. Esta directiva promueve la instalación de tecnologías inteligentes como parte fundamental en la transición energética y la reducción del consumo [11].

A nivel nacional, el proyecto se ha alineado con el **Real Decreto 42/2022, de 18 de enero**, que adapta la legislación española a los objetivos de la directiva europea. Este real decreto establece requisitos técnicos para las instalaciones en edificios, incluyendo la posibilidad de incorporar infraestructuras de domótica, gestión energética, y sistemas de automatización que mejoren el rendimiento y sostenibilidad del inmueble [12].

El cumplimiento de estas normativas garantiza que el sistema desarrollado no solo sea funcional, sino también coherente con las políticas actuales de eficiencia energética, sostenibilidad y digitalización de infraestructuras.

2.6. Estudio del Sistema domótico

2.6.1. Dispositivos

Un sistema domótico se compone de diversos dispositivos electrónicos que interactúan entre sí para ejecutar funciones de automatización, monitorización y control en el entorno doméstico. Estos dispositivos se pueden clasificar según su función dentro del sistema:

- **Sensores:** Son los encargados de recoger información del entorno. Existen sensores de presencia o movimiento (PIR), de luminosidad (como el TEMT6000), de temperatura, humedad, apertura de puertas (magnéticos tipo KY-025), entre otros. Su función es detectar cambios en el ambiente y enviarlos al sistema de control.
- **Actuadores:** Dispositivos que ejecutan una acción a partir de una orden del sistema, como encender una luz, abrir una persiana o activar un motor. Entre los más comunes están los relés, servomotores, motores DC y LEDs RGB como los NeoPixel, que permiten una iluminación personalizada.
- **Controladores:** Son el cerebro del sistema. Se encargan de procesar la información proveniente de los sensores y enviar órdenes a los actuadores. En sistemas domóticos modernos, estos controladores pueden ser microcontroladores como el ESP32 o minicomputadoras como la Raspberry Pi, que además permiten conectividad inalámbrica y control remoto.
- **Interfaz de usuario:** Es el medio por el cual el usuario puede interactuar con el sistema. Puede ser física (paneles táctiles, interruptores inteligentes) o digital, como aplicaciones móviles, asistentes virtuales o dashboards web como Lovelace en Home Assistant.
- **Pasarelas de comunicación (gateways):** Actúan como puentes entre distintos protocolos de comunicación, permitiendo que dispositivos heterogéneos (Zigbee, Wi-Fi, Bluetooth, etc.) se integren en un mismo sistema. En Home Assistant, esto puede incluir integraciones mediante MQTT o complementos como Zigbee2MQTT.
- **Red de comunicaciones:** Es fundamental para la interacción entre los elementos del sistema. Puede ser cableada (Ethernet), inalámbrica (Wi-Fi, Zigbee, Z-Wave) o híbrida. La elección del medio depende de la cobertura deseada, el coste y la compatibilidad de los dispositivos.

Estos componentes trabajan de manera coordinada para convertir una vivienda convencional en un entorno inteligente y eficiente. La elección y configuración de los dispositivos es clave para garantizar la funcionalidad, fiabilidad y escalabilidad del sistema.

2.6.2. Arquitectura y funcionamiento

La arquitectura de un sistema domótico define cómo se estructuran sus componentes y cómo se comunican entre ellos. Existen diferentes modelos de arquitectura, que pueden clasificarse en función de su nivel de centralización, el tipo de red utilizada y la forma de interacción entre los dispositivos.

1. Arquitectura centralizada. Todos los dispositivos están conectados a un controlador central, que recibe las señales de los sensores y envía las órdenes a los actuadores. Este modelo es sencillo de implementar y adecuado para instalaciones pequeñas, pero presenta limitaciones en cuanto a escalabilidad y tolerancia a fallos: si el controlador falla, todo el sistema queda inoperativo.

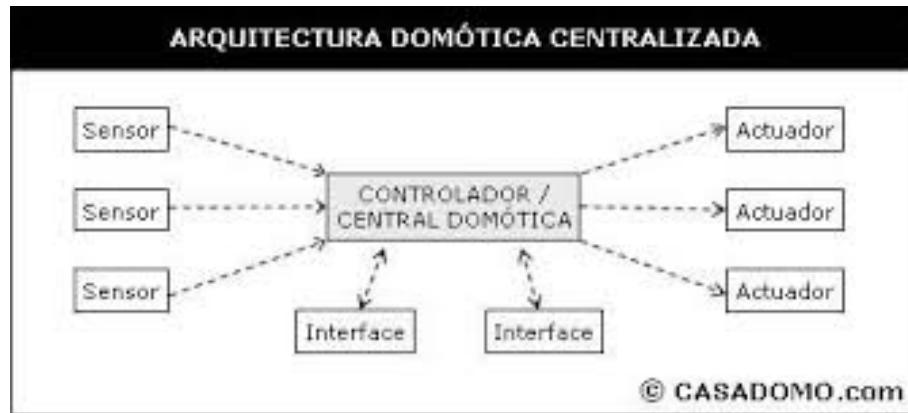


Figura 7: Ejemplo de arquitectura domótica centralizada. Fuente: Casadomo.

2. Arquitectura distribuida. En este modelo, cada subsistema puede operar de forma autónoma y comunicarse con otros nodos sin depender de un único controlador. Ofrece mayor robustez y flexibilidad, siendo ideal para sistemas complejos o viviendas con múltiples zonas independientes. Protocolos como MQTT permiten implementar este tipo de arquitectura con facilidad, especialmente en plataformas como Home Assistant.

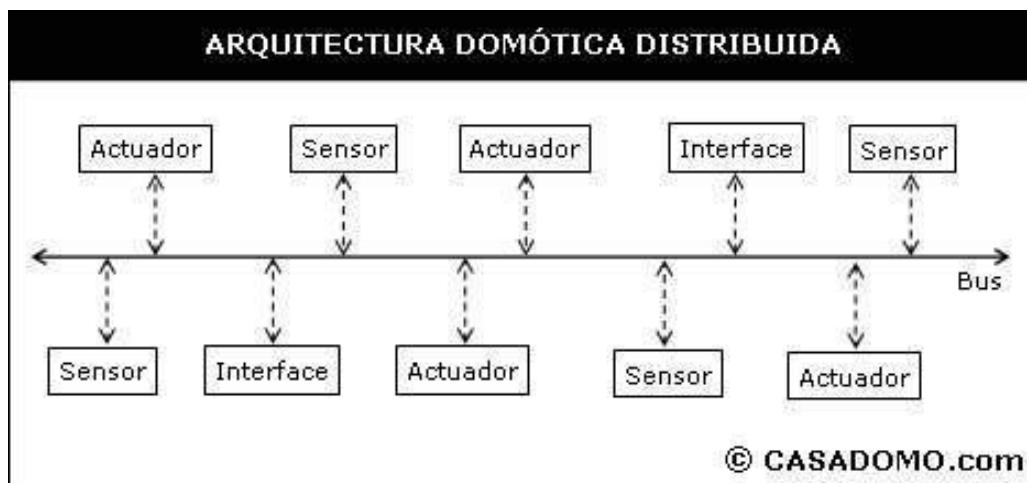


Figura 8: Ejemplo de arquitectura domótica distribuida. Fuente: Casadomo.

3. Arquitectura jerárquica o híbrida. Combina los modelos anteriores. Se utilizan nodos locales para tareas específicas (por ejemplo, control de iluminación en una habitación), que se comunican con un sistema de gestión central que coordina el conjunto. Es habitual en proyectos que integran sensores IoT, nodos ESP32 y una plataforma de control como Raspberry Pi.

ARQUITECTURA DOMÓTICA JERÁRQUICA O HÍBRIDA

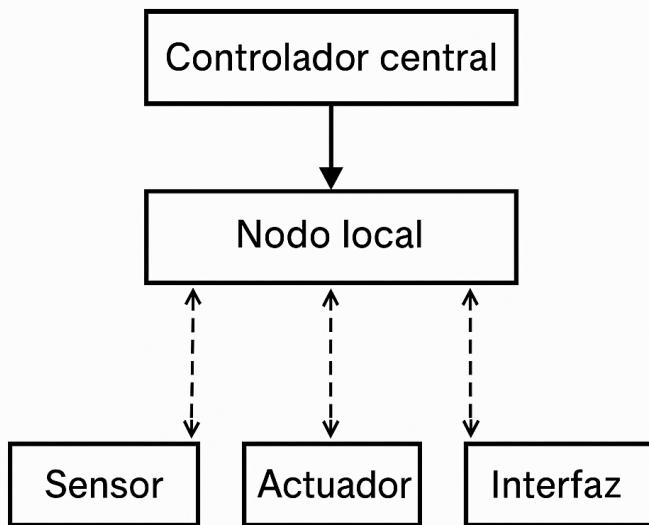


Figura 9: Ejemplo de arquitectura domótica jerárquica o híbrida.

Funcionamiento general. El ciclo básico de funcionamiento en un sistema domótico es el siguiente:

1. Un sensor detecta una condición del entorno (por ejemplo, presencia o baja luminosidad).
2. El dato es transmitido a través de la red al controlador (ESP32, Home Assistant, etc.).
3. El controlador analiza la información y toma una decisión basada en reglas o automatizaciones predefinidas.
4. Se envía una orden al actuador correspondiente (encender luz, abrir una persiana, etc.).
5. El sistema puede informar al usuario mediante una interfaz gráfica o notificación.

Este flujo se puede realizar de forma manual (el usuario acciona un interruptor digital), semiautomática (por programación horaria), o completamente automatizada (condiciones lógicas y sensores).

Ejemplo aplicado al proyecto. En el presente Trabajo de Fin de Grado, se ha adoptado una arquitectura distribuida basada en comunicación MQTT. El ESP32 actúa como nodo de sensores y actuadores, mientras que Home Assistant, instalado en una Raspberry Pi 3, centraliza la lógica de automatización y ofrece la interfaz gráfica al usuario.

2.6.3. Interfaces de usuario

La interfaz de usuario es el componente del sistema domótico que permite la comunicación entre el usuario y los dispositivos de control. Su función principal es ofrecer un medio accesible, intuitivo y eficiente para gestionar, supervisar y configurar el entorno automatizado del hogar.

Existen diferentes tipos de interfaces, que pueden clasificarse según su forma de interacción y su nivel de complejidad:

- **Interfaces físicas:** Incluyen interruptores inteligentes, pantallas táctiles empotradas o mandos a distancia. Son útiles para el control local inmediato de luces, persianas o climatización, y suelen estar instaladas en lugares estratégicos del hogar.
- **Interfaces gráficas:** Son paneles de control visual accesibles desde dispositivos como ordenadores, tablets o smartphones. En este proyecto se utiliza Lovelace, la interfaz predeterminada de Home Assistant, que permite visualizar sensores, activar automatismos, y agrupar dispositivos por categorías o ubicaciones.
- **Aplicaciones móviles:** Las apps de domótica permiten el control remoto del sistema desde cualquier lugar, siempre que exista conexión a Internet. Home Assistant, por ejemplo, dispone de una aplicación oficial para Android e iOS, con funcionalidades avanzadas como geolocalización y notificaciones push.
- **Asistentes virtuales:** Plataformas como Google Assistant, Amazon Alexa o Apple Siri permiten controlar dispositivos mediante comandos de voz. Su integración con Home Assistant ofrece una experiencia de uso natural y manos libres, ideal para accesibilidad o comodidad en situaciones cotidianas.
- **Notificaciones y automatizaciones inteligentes:** Además de controlar manualmente el sistema, las interfaces pueden configurarse para enviar alertas o actuar de forma autónoma ante determinados eventos (por ejemplo, enviar un aviso al detectar movimiento o baja temperatura).

Importancia del diseño. Una buena interfaz debe ser clara, personalizable y adaptarse al perfil del usuario. En entornos residenciales, la simplicidad y la estética son factores clave, mientras que en entornos técnicos o experimentales puede priorizarse la funcionalidad y el acceso a métricas detalladas.

Aplicación al proyecto. En este Trabajo de Fin de Grado se ha utilizado la interfaz Lovelace de Home Assistant para agrupar entidades (luces, sensores, interruptores, etc.) de manera lógica y funcional. El diseño se ha mantenido minimalista y claro, permitiendo una supervisión sencilla del estado del sistema y un control ágil desde distintos dispositivos.

2.6.4. HUBs domóticos o plataformas domóticas

Un HUB domótico, o plataforma de control, es el núcleo del sistema de automatización. Su función principal es centralizar la gestión de sensores, actuadores y dispositivos inteligentes, facilitando su comunicación, integración y control desde una única interfaz.

Existen numerosas plataformas en el mercado, tanto de código abierto como comerciales, que varían en términos de filosofía, compatibilidad, facilidad de uso y posibilidades de personalización.

A continuación, se presenta una tabla comparativa entre algunas de las plataformas más relevantes:

| Criterio | Home Assistant | OpenHAB | SmartThings | Google Home |
|----------------------|----------------|------------|-------------|-------------|
| Licencia | Abierta | Abierta | Cerrada | Cerrada |
| Compatibilidad | Muy alta | Alta | Alta | Alta |
| Facilidad de uso | Media | Media-Baja | Alta | Muy alta |
| Acceso remoto | Sí (gratis) | Sí | Sí | Sí |
| Requiere suscripción | No | No | Opcional | No |
| Personalización | Muy alta | Alta | Media | Baja |

Cuadro 1: Comparativa resumida de plataformas domóticas.

Como se observa, las plataformas de código abierto como Home Assistant y OpenHAB destacan por su alta capacidad de personalización y compatibilidad, aunque pueden requerir más conocimientos técnicos para su configuración. En cambio, soluciones comerciales como Google Home, Alexa o Apple HomeKit ofrecen una experiencia más amigable pero menos flexible, y suelen depender del ecosistema del fabricante.

Elección en el proyecto. En este Trabajo de Fin de Grado se ha optado por **Home Assistant** debido a su carácter abierto, su comunidad activa, su integración con protocolos como MQTT y su gran compatibilidad con hardware de bajo coste como la Raspberry Pi y el ESP32.

2.6.5. Tecnologías y protocolos

Los sistemas domóticos modernos integran una amplia variedad de tecnologías y protocolos de comunicación para garantizar la interoperabilidad, estabilidad y escalabilidad del sistema. La elección de los protocolos depende de factores como el tipo de dispositivos, el alcance deseado, la seguridad y la infraestructura de red disponible.

A continuación, se describen las principales tecnologías y protocolos utilizados en domótica:

- **Wi-Fi:** Muy extendido por su presencia en casi todos los hogares. Permite una alta tasa de transferencia y fácil integración con routers domésticos. Sin embargo, su consumo energético es elevado, lo que lo hace menos adecuado para sensores alimentados por batería.
- **Bluetooth y Bluetooth Low Energy (BLE):** Utilizado en dispositivos de corto alcance como cerraduras inteligentes, balizas o sensores portátiles. BLE destaca por su bajo consumo energético, ideal para soluciones móviles o de bajo mantenimiento.
- **Zigbee:** Protocolo inalámbrico de bajo consumo y malla (mesh), diseñado específicamente para IoT y domótica. Es muy fiable para sensores y actuadores distribuidos por la casa. Requiere un coordinador o gateway para su integración con plataformas como Home Assistant.
- **Z-Wave:** Similar a Zigbee, con arquitectura de malla y bajo consumo. Opera en bandas de frecuencia distintas al Wi-Fi, reduciendo interferencias. Aunque más estable, su ecosistema suele ser más costoso.

- **MQTT (Message Queuing Telemetry Transport)**: Protocolo ligero basado en el modelo publicación/suscripción. Ideal para sistemas distribuidos como el del presente proyecto, permite una comunicación eficiente entre microcontroladores (ESP32) y la plataforma central (Home Assistant).
- **HTTP/REST y WebSockets**: Utilizados para integraciones con servicios web, APIs o dashboards. Permiten el control remoto, la visualización de datos y la ejecución de automatizaciones desde navegadores o apps.
- **ESP-NOW (ESP32)**: Protocolo exclusivo de los microcontroladores ESP32, permite comunicación directa entre dispositivos sin necesidad de router. Útil para sistemas independientes o descentralizados.
- **KNX**: Estándar europeo para automatización de edificios, muy robusto y usado en instalaciones profesionales. Su implementación suele ser cableada y requiere hardware especializado.

Implementación en el proyecto. En este Trabajo de Fin de Grado, se ha utilizado principalmente comunicación **Wi-Fi** para conectar el microcontrolador ESP32 con la red local, y el protocolo **MQTT** para la transmisión de datos entre sensores, actuadores y Home Assistant. Esta combinación ofrece una solución flexible, escalable y de bajo coste.

3. Implementacion del sistema

3.1. Integración del sistema domótico sobre maqueta interactiva

La implementación práctica del sistema domótico se ha llevado a cabo sobre una maqueta interactiva diseñada y construida específicamente para este Trabajo Fin de Grado. Esta maqueta actúa como entorno físico de pruebas y demostración, permitiendo validar todas las funcionalidades del sistema en un espacio controlado y visualmente representativo.

El sistema se compone de múltiples nodos periféricos basados en microcontroladores **ESP32**, cada uno encargado de gestionar sensores y actuadores ubicados en diferentes zonas de la maqueta. Estos nodos se comunican con un servidor central basado en **Home Assistant**, instalado en una **Raspberry Pi 3**, mediante el protocolo de mensajería ligera **MQTT**. Este esquema permite una arquitectura distribuida, escalable y modular.

Entre los componentes integrados se incluyen:

- **Sensores de movimiento PIR**, para detección de presencia en diferentes habitaciones.
- **Sensores magnéticos**, utilizados para la supervisión de apertura y cierre de puertas o compartimentos.
- **LDRs (fotodiódos)**, que permiten detectar los niveles de luminosidad ambiental y activar automatismos como la iluminación nocturna.
- **Tiras LED NeoPixel**, empleadas para la simulación de iluminación ambiental dinámica y controlada por software.

- **Motores DC con sistema piñón-cremallera**, utilizados para la apertura automática de una caja o compartimento dentro de la maqueta.

Cada uno de estos elementos ha sido programado en los microcontroladores ESP32 utilizando firmware personalizado, que gestiona tanto la recolección de datos como el envío y recepción de mensajes MQTT. La lógica de automatización se define en el archivo `configuration.yaml` de Home Assistant, empleando una estructura modular y basada en disparadores y condiciones.

Además, se ha desarrollado una **interfaz gráfica mediante Lovelace**, que permite al usuario monitorizar y controlar el sistema en tiempo real. Esta interfaz incluye tarjetas visuales para luces, sensores y actuadores, ofreciendo una experiencia interactiva y comprensible tanto para usuarios técnicos como no técnicos.

La maqueta no solo representa un entorno físico, sino que también actúa como herramienta educativa y de validación. La integración de todas estas tecnologías en un modelo funcional permite visualizar de forma tangible cómo interactúan los distintos componentes de un sistema domótico, facilitando la comprensión y demostración del proyecto.

Esta implementación refleja fielmente la tendencia actual en domótica: soluciones descentralizadas, accesibles, basadas en estándares abiertos y orientadas al aprendizaje y la replicabilidad.

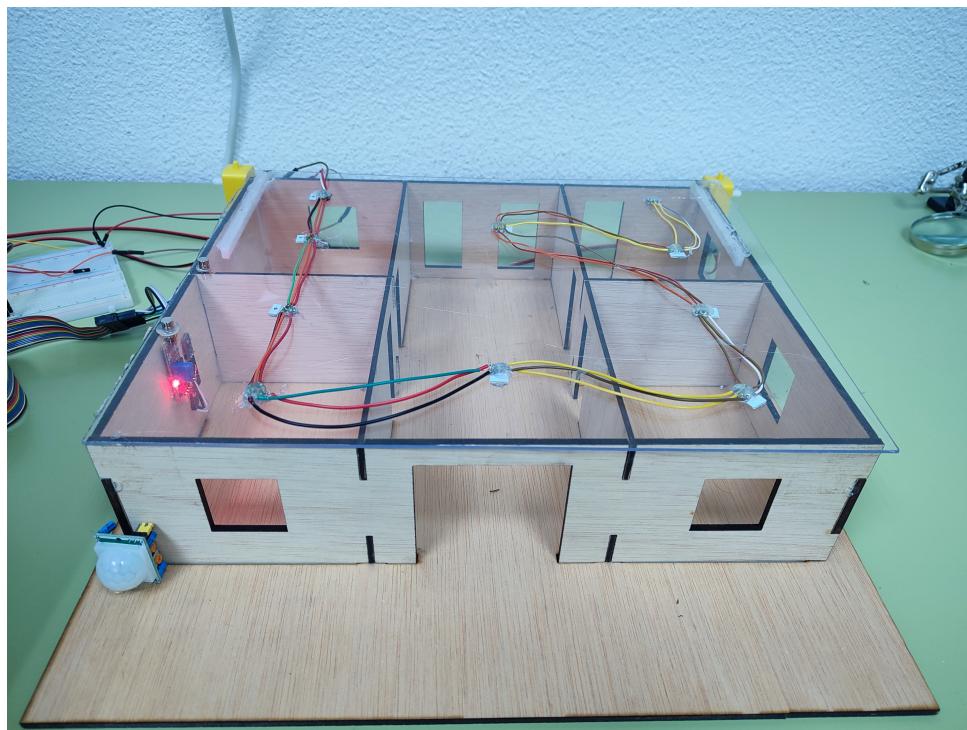


Figura 10: Vista general del sistema domótico implementado en la maqueta interactiva

3.2. Tecnologías utilizadas

Para el desarrollo de la solución domótica implementada en este proyecto, se han empleado tecnologías ampliamente adoptadas en entornos de automatización e Internet de las Cosas (IoT).

En lo que respecta a la comunicación entre el microcontrolador ESP32 y los servicios domóticos como Home Assistant, se ha utilizado el protocolo MQTT. Este protocolo

de mensajería ligera es especialmente adecuado para dispositivos embebidos y redes con limitaciones de ancho de banda, debido a su bajo consumo de recursos y simplicidad en la implementación [9].

Además, para facilitar la interacción entre diferentes plataformas y servicios en la red, se han empleado APIs. Estas interfaces permiten una comunicación estructurada, segura y eficiente entre dispositivos, servidores y usuarios, facilitando la integración del sistema en entornos heterogéneos [10].

El uso conjunto de estas tecnologías proporciona una arquitectura robusta y escalable, adaptada a las necesidades del entorno domótico, permitiendo el control remoto, la monitorización en tiempo real y la automatización de las tareas del hogar.

3.3. Diagrama de bloques de la arquitectura de red

El siguiente diagrama representa la arquitectura de red y la interacción entre los distintos elementos del sistema domótico, desde la interfaz del usuario hasta la ejecución física de acciones en la maqueta.

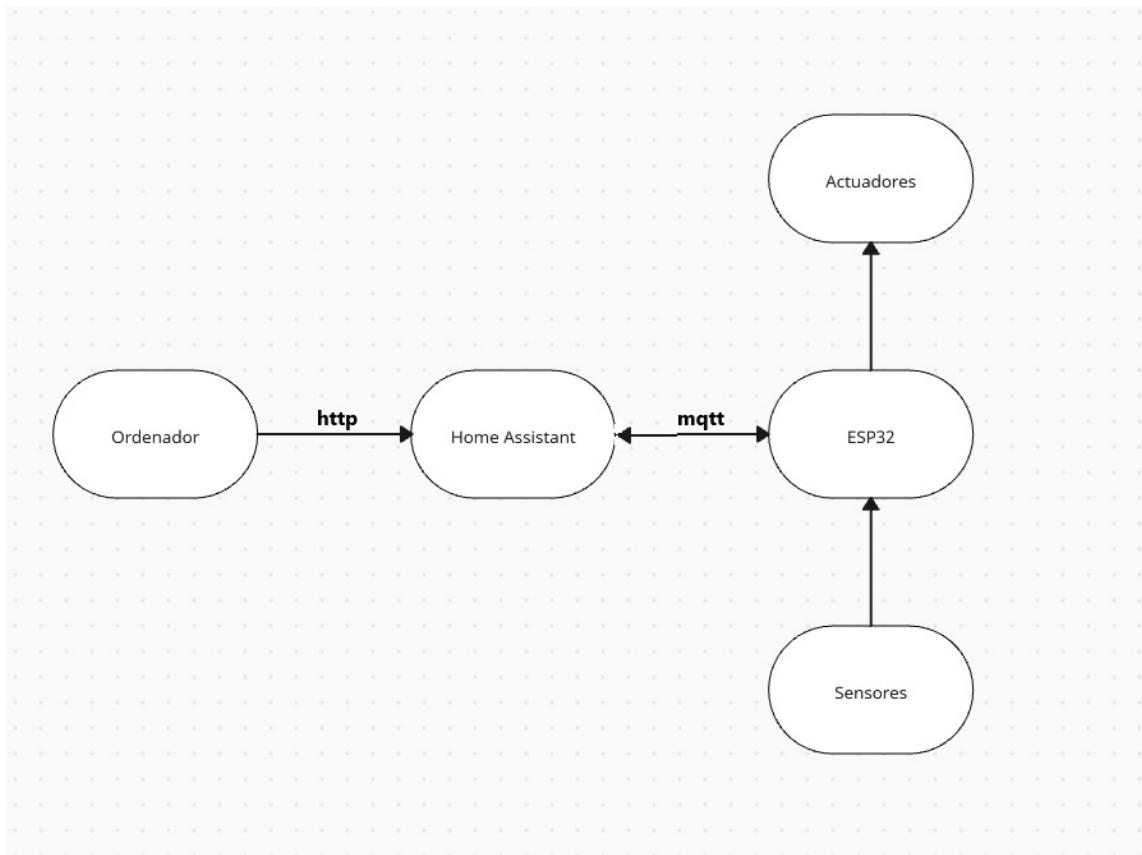


Figura 11: Diagrama de bloques de la comunicación entre el ordenador, Home Assistant, ESP32, sensores y actuadores

Como se observa en la Figura 11, el **ordenador del usuario** accede a la interfaz Lovelace de Home Assistant mediante la red local Wi-Fi. Esta red conecta también a la **Raspberry Pi 3**, que actúa como servidor domótico, y al **microcontrolador ESP32**, que opera como nodo periférico.

El ESP32 intercambia mensajes con Home Assistant mediante el protocolo MQTT, permitiendo una comunicación **bidireccional**: por un lado, envía valores capturados por

los sensores (como presencia, luz o contacto magnético), y por otro, recibe instrucciones para accionar dispositivos como tiras LED o motores DC.

Esta estructura permite una integración modular, escalable y eficiente de todos los elementos, tanto físicos como lógicos, dentro de la maqueta domótica.

3.4. Hardware

Raspberry Pi 3

Para este proyecto de domótica, se ha elegido la **Raspberry Pi 3** como unidad central de procesamiento. Esta decisión responde a varios criterios clave [29]:

- **Costo reducido:** Comparada con otros miniordenadores o microcontroladores más avanzados, la Raspberry Pi 3 ofrece un equilibrio ideal entre funcionalidad y precio, situándose por debajo de los 60 euros incluyendo fuente de alimentación y carcasa.
- **Recursos:** Con su procesador quad-core ARM Cortex-A53 y 1 GB de RAM, es más que capaz de ejecutar servicios de domótica como Home Assistant, además de manejar múltiples sensores y actuadores en paralelo sin problemas de rendimiento.
- **Compatibilidad y comunidad:** La Raspberry Pi 3 cuenta con un amplio soporte de software y una comunidad activa. Esto facilita la instalación de sistemas operativos optimizados (como Raspberry Pi OS o distribuciones específicas para domótica como Home Assistant OS), así como la resolución de problemas.
- **Conectividad integrada:** Dispone de conectividad Wi-Fi y Bluetooth incorporadas, eliminando la necesidad de módulos externos y simplificando la comunicación con sensores, nodos IoT y dispositivos móviles.



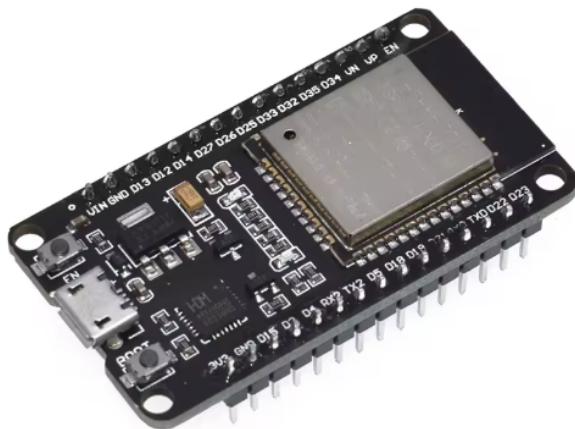
Figura 12: Placa Raspberry Pi 3 utilizada como controlador central.

En resumen, la Raspberry Pi 3 se presenta como una solución económica, accesible y suficientemente potente para cubrir los requerimientos del presente sistema de automatización del hogar.

ESP32

El microcontrolador **ESP32** ha sido elegido como componente clave para tareas distribuidas dentro del sistema domótico, especialmente como nodo periférico encargado de la lectura de sensores y el control de dispositivos remotos. Esta decisión se basa en sus múltiples ventajas [28].

- **Capacidad de cómputo destacable:** El ESP32 cuenta con un procesador dual-core a 240 MHz y 520 KB de SRAM, lo cual le permite ejecutar múltiples tareas en paralelo, desde la adquisición de datos hasta su procesamiento y transmisión. Esto lo convierte en una plataforma potente para microcontrolador, muy superior a alternativas como el ESP8266.
- **Conectividad a Internet:** Gracias a su módulo Wi-Fi integrado, el ESP32 puede conectarse directamente a una red local o a Internet, permitiendo la comunicación constante con la Raspberry Pi o servidores en la nube. Esta característica lo hace ideal para aplicaciones de Internet de las Cosas (IoT), donde la comunicación remota es esencial.
- **Soporte para protocolos IoT:** El ESP32 es compatible con protocolos como MQTT, ampliamente utilizado en domótica por su eficiencia y estructura basada en *publicación/suscripción*. Esto le permite suscribirse a múltiples tópicos y reaccionar en tiempo real ante mensajes recibidos, facilitando una arquitectura descentralizada y reactiva.
- **Bajo coste y consumo:** Por un precio inferior a 4 euros y con modos de ahorro energético avanzados, el ESP32 permite desplegar múltiples nodos inalámbricos sin comprometer ni el presupuesto ni la eficiencia energética del sistema.
- **Flexibilidad en conexiones:** Dispone de numerosos pines para entradas y salidas digitales, analógicas, PWM, y buses de comunicación como I2C, SPI o UART, lo que permite conectarle fácilmente sensores, actuadores y otros periféricos.



Módulo puente H L298N

En el diseño del sistema se ha seleccionado el módulo **puente H L298N** para el control de motores de corriente continua. Aunque existen alternativas más simples y económicas como el **L293D**, el **L298N** representa una solución más robusta y adecuada para aplicaciones que demandan una mayor potencia, incluso por encima de la que se prevé utilizar en este proyecto [33].

- **Control eficiente mediante PWM:** El L298N permite el control tanto de la velocidad como del sentido de giro de los motores mediante señales PWM, lo que ofrece un control fino y eficiente con una implementación sencilla desde microcontroladores como el ESP32 o la Raspberry Pi.
- **Versatilidad y flexibilidad:** A diferencia de un controlador de motor paso a paso —opción inicialmente considerada por su capacidad para controlar con precisión el número de vueltas y la posición— el uso del L298N con motores DC proporciona mayor flexibilidad, menor coste y menor complejidad en el control.
- **Robustez:** El L298N está diseñado para manejar cargas de mayor corriente, lo que garantiza mayor durabilidad y menor riesgo de fallos ante picos de consumo. Esto lo hace adecuado tanto para motores pequeños como para otros más potentes si se decidiera escalar el sistema en el futuro.
- **Alternativa de control de posición:** Aunque los motores DC no permiten un control de posición directo como los paso a paso, este inconveniente se compensa mediante el uso de dos sensores magnéticos que permiten determinar la posición de forma efectiva, fiable y económica.



Figura 14: Módulo puente H L298N para control de motores.

Por todo lo anterior, el módulo L298N ofrece un equilibrio entre coste, potencia y control, convirtiéndose en la opción más adecuada para el sistema motorizado del proyecto.

Sensor de luz TEMT6000

En una primera etapa del diseño del sistema domótico, se incorporó el **sensor de luz TEMT6000** como una solución sencilla y económica para automatizar el encendido y apagado de las luces en función de la luminosidad ambiental [32].

- **Motivación inicial:** El objetivo era que el sistema fuese capaz de detectar condiciones de baja iluminación (por ejemplo, al anochecer) y activar las luces de forma autónoma, sin necesidad de intervención del usuario ni programación horaria fija.
- **Funcionamiento:** El TEMT6000 es un fototransistor que genera una señal analógica proporcional a la luz incidente, lo que permite determinar con precisión los niveles de luminosidad del entorno.
- **Limitación en el entorno de pruebas:** Aunque el planteamiento inicial era prometedor, se encontró una limitación significativa: la maqueta del sistema está situada permanentemente en una habitación interior, sin exposición directa a la luz natural. Esto redujo drásticamente la utilidad del sensor, que no reflejaba de forma realista las condiciones lumínicas exteriores.
- **Solución alternativa:** Para superar esta limitación, se optó por sustituir parcialmente la función del TEMT6000 mediante el uso de una **API meteorológica**, capaz de consultar información sobre la hora de salida y puesta del sol en tiempo real, y adaptar el comportamiento del sistema de iluminación en función de estos datos.

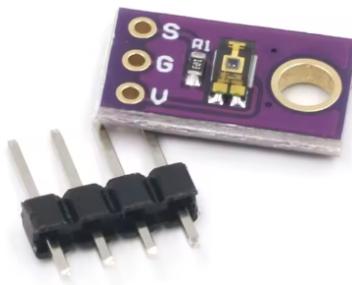


Figura 15: Sensor de luz TEMT6000 para detección de luminosidad.

Aunque el TEMT6000 no se utilizó de forma activa en la versión final, su incorporación inicial demostró una aproximación válida y su uso podría retomarse en instalaciones reales con acceso a luz exterior.

Sensor magnético KY-025

Para conocer el estado de apertura o cierre de una caja dentro del sistema domótico, se ha optado por emplear el **sensor magnético KY-025** en lugar de otros mecanismos más tradicionales como botones de contacto o finales de carrera [31].

- **Necesidad del sistema:** Era fundamental incorporar un mecanismo que permitiera detectar si la caja estaba abierta o cerrada, con el fin de activar determinadas funciones (como alarmas, registro de eventos o desactivación del sistema) en función de su estado.
- **Solución inicial:** En un primer planteamiento, se consideró el uso de pulsadores o interruptores mecánicos que cambiaron de estado al contacto con la tapa de la caja. Sin embargo, este enfoque presentaba problemas de fiabilidad y desgaste por fricción, así como una mayor complejidad en el montaje físico.

- **Ventajas del KY-025:** El sensor magnético KY-025, que funciona en conjunto con un imán, permite detectar sin contacto físico la presencia o ausencia del imán. Esto simplifica notablemente la instalación, elimina desgaste mecánico y mejora la fiabilidad del sistema a largo plazo.
- **Funcionamiento en el proyecto:** Al colocar un imán en la tapa de la caja y el sensor KY-025 en la base, se puede determinar si la caja está cerrada (el sensor detecta el campo magnético) o abierta (el imán se aleja). Este sistema no requiere presión, alineación precisa ni mantenimiento, y es más tolerante a pequeños desplazamientos o vibraciones.
- **Relación con otros elementos:** Además, este tipo de sensor se ha integrado con éxito en el sistema de detección de posición del motor, ya que permite definir estados lógicos (posición 1 o posición 2) sin necesidad de un motor paso a paso, lo que reduce costes y complejidad.

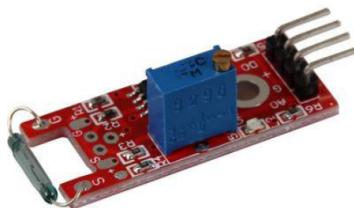


Figura 16: Sensor magnético KY-025 para detección de posición.

Gracias a estas ventajas, el KY-025 se ha consolidado como una solución eficaz, económica y robusta para la detección de estados físicos en el proyecto.

Elección del motor DC con piñón-cremallera

Para el mecanismo de apertura y cierre de una tapa dentro del sistema, se ha optado por un conjunto sencillo formado por un **motor DC acoplado a un sistema de piñón y cremallera**, ambos fabricados en plástico y adquiridos como un pack económico. Esta elección responde a criterios de simplicidad, funcionalidad y facilidad de integración.

- **Objetivo funcional:** El sistema únicamente necesita realizar un movimiento de apertura y cierre, sin necesidad de un control preciso del ángulo, velocidad variable compleja o esfuerzos mecánicos elevados. Por tanto, se descartaron soluciones más complejas por ser innecesarias.

- **Alternativas evaluadas:**

- *Servomotores:* Aunque permiten un control preciso del ángulo, requieren un mayor control electrónico y suelen ser más costosos.
- *Motores paso a paso:* Fueron considerados inicialmente por su capacidad de controlar con precisión el número de pasos y, por tanto, la posición. Sin embargo, resultan innecesarios para una tarea binaria como abrir o cerrar, y requieren una electrónica de control más compleja.

- *Sistemas con bisagras automatizadas o actuadores lineales:* También se valoraron, pero se descartaron por su coste, volumen y la necesidad de piezas adicionales.
- **Ventajas del sistema piñón-cremallera:**
- Su mecánica simple permite convertir el movimiento rotativo del motor en uno lineal con facilidad.
 - No requiere el uso de piezas impresas en 3D, lo cual es una gran ventaja práctica. La impresión 3D de elementos pequeños puede presentar problemas de precisión, ajuste o resistencia mecánica, lo cual se evita completamente con esta solución comercial.
 - El montaje es directo y funcional, reduciendo el tiempo de desarrollo y minimizando los puntos de fallo mecánico.
- **Complemento con sensores:** El sistema se integra perfectamente con los sensores magnéticos KY-025, que permiten detectar la posición final de la tapa (abierta o cerrada), sin necesidad de retroalimentación por parte del motor.

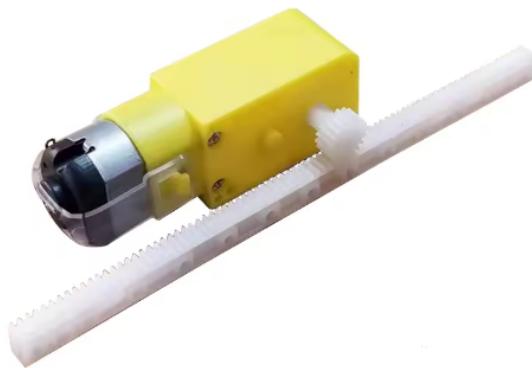


Figura 17: Motor DC con sistema piñón-cremallera para apertura lineal.

Esta solución representa un equilibrio ideal entre simplicidad mecánica, fiabilidad y coste, cumpliendo con creces los requerimientos funcionales del proyecto.

LEDs NeoPixel

Para este proyecto se ha optado por utilizar LEDs del tipo *NeoPixel* (basados en el chip WS2812), los cuales ofrecen ventajas significativas en comparación con LEDs tradicionales o matrices de control más complejas [30].

Uno de los principales motivos para su elección es que cada LED es direccionable individualmente y todos pueden ser controlados en cadena a través de un único pin digital del microcontrolador. Esto simplifica considerablemente el cableado, reduciendo el número de pines necesarios y dejando libres otros recursos del microcontrolador para otras tareas.

Además, los NeoPixel permiten definir el color y el brillo de cada LED mediante simples estructuras de datos y bucles iterativos, lo que facilita la implementación del código de control. Su compatibilidad con bibliotecas como `Adafruit_NeoPixel` en plataformas como ESP32 hace que su uso sea accesible y estable.

Otro aspecto clave es su escalabilidad: no existe una limitación práctica estricta sobre la cantidad de LEDs que se pueden utilizar, más allá del consumo eléctrico y la memoria del microcontrolador. Esto significa que el sistema puede crecer fácilmente en número de LEDs sin requerir rediseño del hardware o de la arquitectura de control.

Por estas razones —facilidad de conexión, control eficiente mediante código, y escalabilidad— los NeoPixel han resultado una opción ideal para el sistema de iluminación RGB implementado en la maqueta domótica del proyecto.

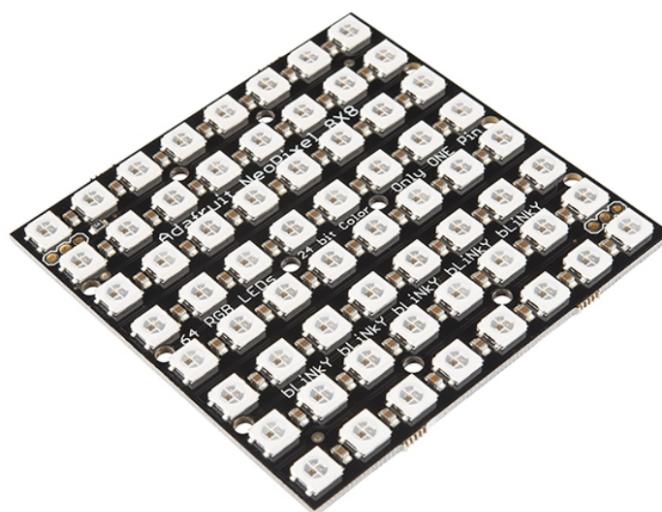


Figura 18: Matriz de LEDs NeoPixel para iluminación RGB direccional.

3.5. Instalación de Home Assistant en Raspberry Pi

Para la implementación del centro de control domótico, se utilizó una **Raspberry Pi 3** como dispositivo principal por su bajo consumo, buen rendimiento y compatibilidad con Home Assistant. La instalación se realizó siguiendo las instrucciones oficiales del proyecto en <https://www.home-assistant.io/installation/raspberrypi>.

3.5.1. Requisitos Previos

Antes de iniciar el proceso, se necesitaban los siguientes elementos:

- Raspberry Pi 3
 - Fuente de alimentación oficial
 - Tarjeta microSD (32 GB, clase 10)
 - Conexión a Internet (preferiblemente por cable Ethernet)
 - Ordenador con lector de tarjetas SD
 - Software **Raspberry Pi Imager**

3.5.2. Descarga e Instalación de Home Assistant OS

Home Assistant se ejecuta sobre su propio sistema operativo llamado **Home Assistant OS**, especialmente optimizado para dispositivos como la Raspberry Pi.

1. Se accedió a la web oficial: <https://www.home-assistant.io/installation/raspberrypi>
2. Se descargó la imagen correspondiente a la Raspberry Pi utilizada, en formato .img.
3. Con el programa **Raspberry Pi Imager** se grabó la imagen en la tarjeta microSD.

3.5.3. Primer Arranque y Acceso Inicial

1. Se insertó la tarjeta microSD en la Raspberry Pi.
2. Se conectó la Raspberry Pi a la red local mediante cable Ethernet.
3. Se encendió el dispositivo y se esperaron aproximadamente 20 minutos para que se completara la instalación inicial.
4. Desde un navegador web, se accedió a Home Assistant a través de la dirección:

`http://homeassistant.local:8123`

En caso de fallo con la dirección local, también se podía usar la IP local asignada al dispositivo (por ejemplo, `http://192.168.1.140:8123`).

3.5.4. Configuración Inicial del Sistema

Una vez iniciado Home Assistant:

- Se creó un usuario administrador.
- Se estableció la ubicación geográfica del dispositivo.
- Se configuraron los sistemas de detección automática de dispositivos y servicios.

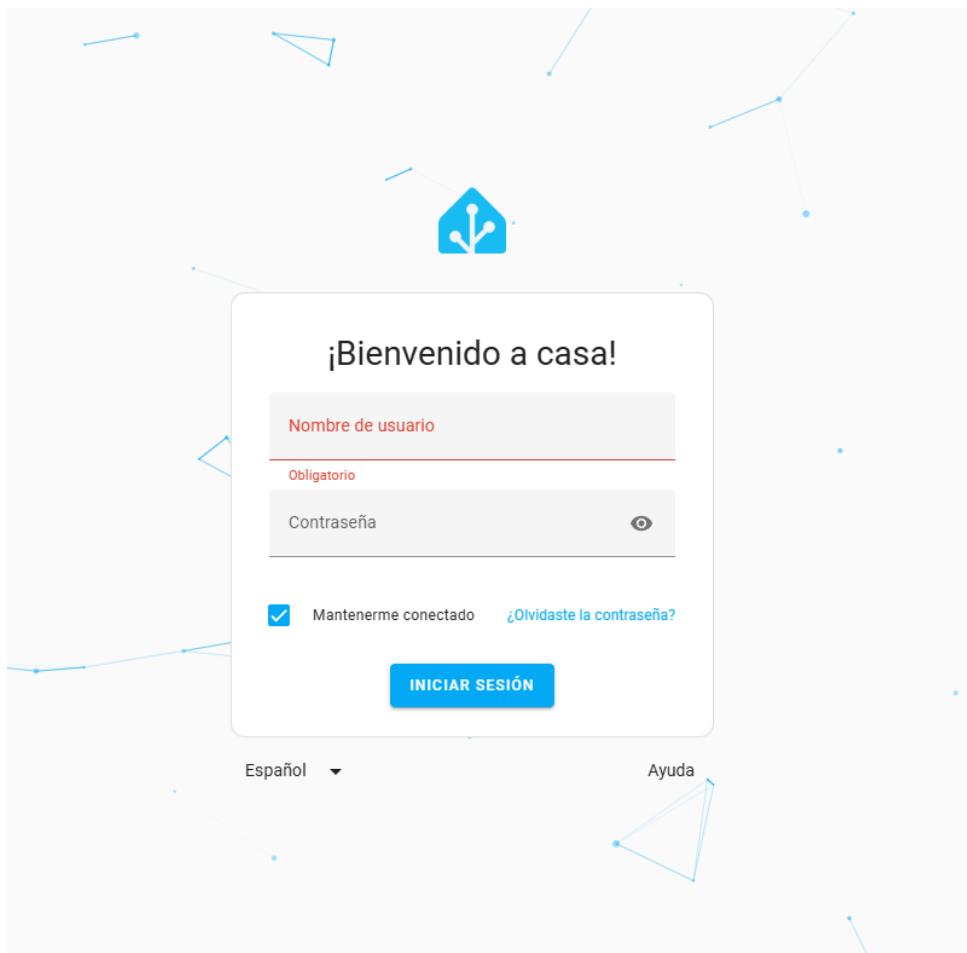


Figura 19: Pantalla de configuración inicial de Home Assistant tras el primer arranque.

3.5.5. Ventajas del Home Assistant OS

Se eligió instalar la versión completa de Home Assistant OS (en lugar de Home Assistant Core sobre Raspbian), ya que incluye:

- Sistema operativo ligero y optimizado
- Actualizaciones automáticas y seguras
- Acceso a **Supervisor**, que permite gestionar complementos (Add-ons)
- Mayor compatibilidad con integraciones como MQTT, DuckDNS, entre otros

3.5.6. Conclusión

La instalación de Home Assistant en la Raspberry Pi permitió transformar este pequeño ordenador en un potente centro de control domótico. Su fiabilidad, consumo reducido y versatilidad lo hacen ideal para proyectos de automatización del hogar.

3.6. Configuración de Red y Acceso Remoto en Home Assistant

Una correcta configuración de red es esencial para garantizar el acceso seguro, estable y remoto al sistema domótico basado en Home Assistant. Esta sección detalla cómo se

estableció una IP fija local, se configuró un dominio dinámico con DuckDNS, se abrieron puertos en el router y se resolvieron limitaciones con la IP pública mediante el proveedor de servicios DIGI.

3.6.1. Asignación de IP Fija en la Red Local

Para evitar problemas de conectividad con dispositivos y servicios dependientes de una dirección IP constante, se configuró Home Assistant con una IP estática:

- **Dirección IP:** 192.168.1.140
- **Máscara de red:** 255.255.255.0
- **Puerta de enlace:** 192.168.1.1
- **DNS primario:** 100.90.1.1
- **DNS secundario:** 100.100.1.1

Configura las interfaces de red

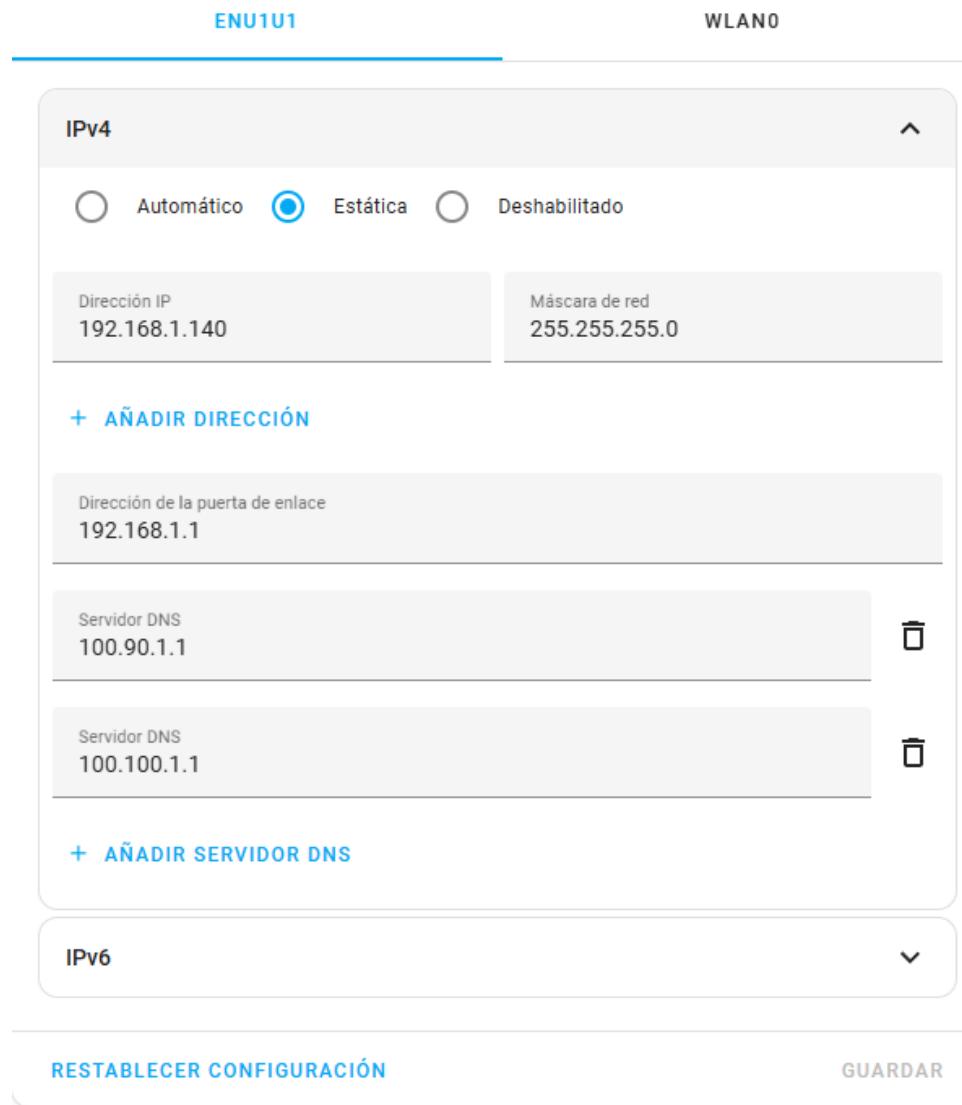


Figura 20: Configuración de IP estática en Home Assistant.

3.6.2. Acceso Remoto Seguro con DuckDNS

Para permitir el acceso externo a Home Assistant, se utilizó el complemento oficial **DuckDNS**, que permite vincular una IP dinámica a un subdominio personalizado. El procedimiento fue el siguiente:

1. Registro en <https://duckdns.org> con una cuenta de Google.
2. Creación del subdominio: tfg-miguel-lopez.duckdns.org.
3. Instalación del complemento DuckDNS desde Ajustes → Complementos → DuckDNS.
4. Configuración del archivo del complemento:

```

1 lets_encrypt:
2   accept_terms: true
3   algo: secp384r1
4   certfile: fullchain.pem
5   keyfile: privkey.pem
6   token: *****
7 domains:
8   - tfg-miguel-lopez.duckdns.org

```

5. Adición de certificados SSL en configuration.yaml:

```

1 http:
2   ssl_certificate: /ssl/fullchain.pem
3   ssl_key: /ssl/privkey.pem

```

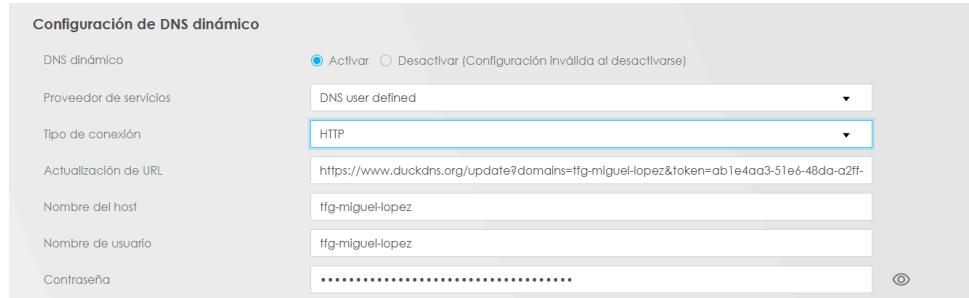


Figura 21: Configuración del DNS dinámico DuckDNS en el router Zyxel

3.6.3. Apertura de Puertos en el Router

Para habilitar el acceso remoto, se abrió el puerto 8123 (HTTP) desde el NAT del router. Aunque se probaron configuraciones con HTTPS, el protocolo HTTP ofreció mayor estabilidad en este caso práctico.

- **Puerto externo:** 8123
- **Puerto interno:** 8123 (por defecto en Home Assistant)
- **Protocolo:** TCP



Figura 22: Redirección de puertos configurada en el router



ugr

Universidad
de Granada

3.6.4. Consideraciones sobre la IP Pública y DIGI Plus

Una de las principales dificultades fue la ausencia de una IP pública dedicada. Las compañías de bajo coste, como DIGI, suelen asignar IPs compartidas (CG-NAT), lo que impide la apertura de puertos de forma directa. Esta limitación se resolvió contratando el servicio **DIGI Plus**, que proporciona una IP pública real, indispensable para:

- Abrir puertos para acceso remoto
- Hacer funcionar DuckDNS correctamente
- Evitar restricciones de conectividad en el entorno de red

Sin una IP pública, múltiples usuarios de una misma zona comparten la misma IP externa, lo que impide la llegada directa de conexiones entrantes desde Internet a Home Assistant.

3.6.5. Resumen

La configuración de red incluyó los siguientes pasos clave:

1. Asignación de IP fija local para estabilidad en la red.
2. Configuración de acceso remoto mediante DuckDNS y certificados SSL.
3. Apertura de puertos en el router para redirigir tráfico externo a Home Assistant.
4. Contratación de IP pública dedicada con DIGI Plus para garantizar conectividad total.

Esta infraestructura de red permite acceder de forma remota, segura y confiable al sistema domótico desde cualquier parte del mundo.

3.7. Configuración del Broker MQTT Mosquitto en Home Assistant

Para habilitar la comunicación entre los dispositivos del sistema domótico mediante el protocolo MQTT, se utilizó el complemento oficial **Mosquitto Broker** en Home Assistant [19]. A continuación, se detalla el procedimiento de instalación y configuración realizado.

3.7.1. Instalación del Complemento Mosquitto

La instalación se llevó a cabo desde la interfaz de Home Assistant siguiendo los siguientes pasos:

1. Navegar a Configuración → Complementos → Tienda de complementos.
2. Buscar el complemento *Mosquitto broker* y hacer clic en él.
3. Presionar el botón INSTALAR.

3.7.2. Creación de Usuario MQTT

Para permitir la autenticación de clientes, se creó un usuario MQTT desde Home Assistant:

- **Nombre de usuario:** miguelmqtt
- **Contraseña:** miguelmqtt

Este usuario fue creado a través del menú Configuración → Personas → Usuarios y no directamente desde la configuración del complemento.

3.7.3. Configuración del Complemento

La configuración YAML utilizada en el complemento fue la siguiente:

```

1 logins:
2   - username: miguelmqtt
3     password: miguelmqtt
4 customize:
5   active: false
6   folder: mosquitto
7 certfile: fullchain.pem
8 keyfile: privkey.pem
9 require_certificate: false

```

Cabe destacar que esta configuración también puede realizarse mediante la interfaz gráfica de Home Assistant, como se muestra en la siguiente imagen:

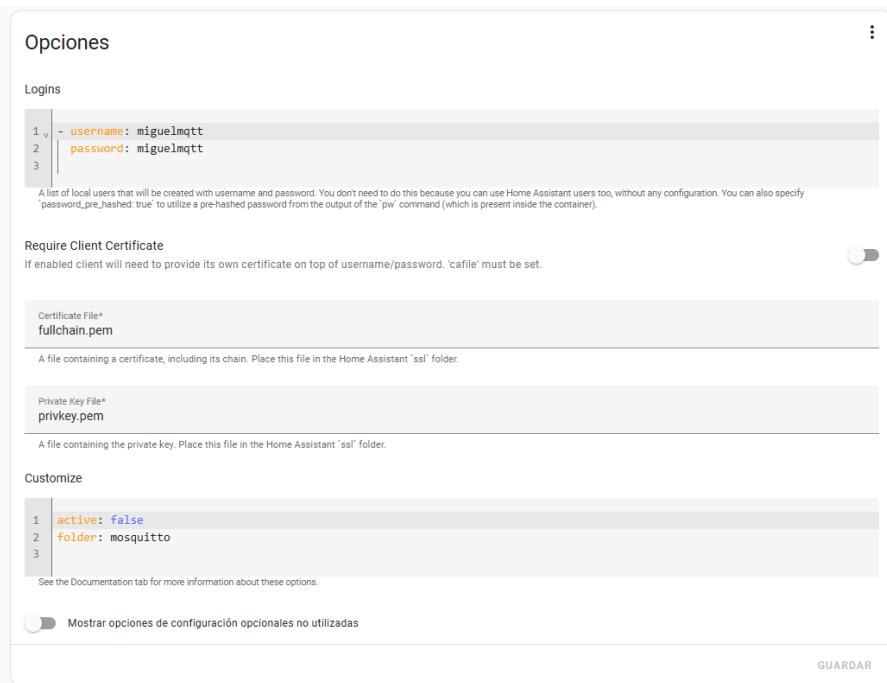


Figura 23: Configuración del complemento MQTT a través de la interfaz gráfica.

3.7.4. Puertos Configurados

Se configuraron los siguientes puertos en el complemento, los cuales fueron expuestos en el host para permitir distintas formas de conexión:

- **1883/tcp** - MQTT estándar sin cifrado
- **1884/tcp** - MQTT sobre WebSocket sin cifrado
- **8883/tcp** - MQTT cifrado con SSL
- **8884/tcp** - MQTT sobre WebSocket cifrado con SSL

3.7.5. Integración con Home Assistant

Una vez iniciado el complemento, se detectó automáticamente la integración MQTT mediante:

- Configuración → Dispositivos y servicios → Integraciones

Se habilitó la detección automática y se completó la integración con un solo clic, permitiendo que Home Assistant pudiera tanto publicar como suscribirse a tópicos MQTT.

Notas Adicionales

- No se permite el acceso anónimo, por lo que es obligatorio el uso de credenciales.
- No se activaron las listas de control de acceso (ACL), aunque es posible configurarlas posteriormente si se desea restringir el acceso por usuario o tópico.

Durante el desarrollo del sistema domótico, se decidió realizar una reestructuración completa del archivo `configuration.yaml` para mejorar la organización, legibilidad y mantenimiento del código. En lugar de mantener toda la configuración centralizada, se optó por dividirla en módulos temáticos mediante directivas `!include`. Esta práctica no solo facilita la depuración de errores, sino que también reduce la probabilidad de fallos de compilación ante configuraciones extensas o desordenadas [17].

Además, se definieron scripts reutilizables en archivos independientes, permitiendo encapsular acciones complejas que pueden ser llamadas desde diferentes automatizaciones [21].

3.7.6. Estructura de configuration.yaml

```

1 # Carga por defecto de las integraciones. No eliminar.
2 default_config:
3
4 # Temas personalizados
5 frontend:
6   themes: !include_dir_merge_named themes
7
8 # Inclusión de módulos externos

```



```

9  automation: !include automations.yaml
10 script: !include scripts.yaml
11 scene: !include scenes.yaml
12 mqtt: !include mqtt.yaml

```

3.7.7. Automatización: API Meteorológica para Control Lumínico

Se integró una automatización basada en la entidad `sun.sun` de Home Assistant, que utiliza datos astronómicos (amanecer, anochecer) para determinar si es de día o de noche. Esta automatización publica dicha información mediante MQTT cada 5 minutos, permitiendo a un ESP32 controlar las luces de la maqueta [20].

```

1
2
3 - alias: "Enviar estado de día o noche por MQTT cada 5 minutos"
4   trigger:
5     - platform: state
6       entity_id: sun.sun
7     - platform: time_pattern
8       minutes: "/5"
9   action:
10    - service: mqtt.publish
11      data:
12        topic: "casa/luz/dia_noche"
13        payload: >
14          {% if is_state('sun.sun', 'above_horizon') %}
15            "es de dia"
16          {% elif is_state('sun.sun', 'below_horizon') %}
17            "es de noche"
18          {% else %}
19            "desconocido"
20          {% endif %}
21        retain: true

```

3.7.8. Código de Configuración MQTT

La integración con Home Assistant se realiza mediante el archivo `mqtt.yaml`, donde se definen distintas entidades vinculadas a los tópicos MQTT publicados o suscritos por el ESP32 [18]. A continuación, se describe de forma orientativa cómo se estructura esta configuración:

- **Luces RGB (light):** Cada LED se define como una entidad `light` con soporte para color RGB, brillo, y control mediante JSON. Se utiliza el mismo tópico tanto para el estado como para el comando. Todos los LEDs siguen el mismo patrón, cambiando únicamente el índice [24]:

```

light:
  - name: "LED 1"
    command_topic: "home/led_1/set"
    schema: json

```



```
brightness: true
supported_color_modes: ["rgb"]
```

- **Sensores (sensor)**: Se definen entidades de solo lectura para datos como iluminación ambiental o distancia. Se usan plantillas para dar formato a los valores recibidos [22].
- **Sensores binarios (binary_sensor)**: Representan estados booleanos como presencia (movimiento), luz ambiental baja, o proximidad detectada. Se definen con los tópicos correspondientes y las cargas esperadas para encendido/apagado [22].
- **Controles numéricos (number)**: Permiten ajustar parámetros como la velocidad de motores. Se definen con valores mínimos, máximos y pasos, y se controlan mediante tópicos `set` [18].
- **Interruptores (switch)**: Usados para activar o desactivar funciones binarias como abrir/cerrar la caja o encender/apagar una luz nocturna. Las cargas definen claramente la acción a ejecutar [23].

```
switch:
  - name: "Caja - Abrir"
    command_topic: "home/caja/estado/set"
    payload_on: "abrir"
    payload_off: "cerrar"
```

3.8. Configuración del Código ESP32

Para empezar la primera parte del código se incluyen las librerías necesarias para conexión WiFi, comunicación MQTT, control de LEDs RGB, manejo de mensajes JSON y PWM para motores.

Se definen las credenciales WiFi, la configuración del cliente MQTT y los tópicos utilizados para controlar LEDs, motores, sensores (luz, movimiento, distancia) y actuadores como una caja automatizada o luz nocturna.

Se configuran los pines de entrada y salida para los diferentes sensores y actuadores como motores o LEDs. También se establecen variables globales que gestionan el estado del sistema, como la iluminación ambiental, el movimiento o la inactividad.

Se declara una enumeración que representa los distintos estados de una caja motorizada, y se definen funciones para el manejo de mensajes MQTT y procesamiento de datos en formato JSON.

3.8.1. Función `setup()`

La función `setup()` inicializa el sistema al encender el dispositivo. Se ha optado por una estructura modular, separando cada bloque funcional en métodos independientes (`configurarPines()`, `conectarWiFi()`, `configurarMQTT()`, `configurarMCPWM()`), lo que mejora significativamente la legibilidad, mantenibilidad y escalabilidad del código.

Esta organización permite localizar y modificar fácilmente cada sección del proceso de configuración sin afectar el resto del sistema, cumpliendo buenas prácticas de programación embebida.

```

1 void setup() {
2     Serial.begin(115200);
3     pixels.begin();
4     configurarPines();
5     conectarWiFi();
6     configurarMQTT();
7     configurarMCPWM();
8     Serial.println(" Configuración completada");
9     ultimoMovimiento = millis();
10 }

```

3.8.2. Función loop()

La función `loop()` es el núcleo del sistema embebido. Se ejecuta de forma continua y gestiona tanto la conectividad MQTT como las tareas periódicas relacionadas con sensores, actuadores y lógica de automatización.

```

1 void loop() {
2     if (!client.connected()) reconnect();
3     client.loop();
4     controlarCaja();
5
6     static unsigned long lastTime = 0;
7     if (millis() - lastTime >= 5000) {
8         lastTime = millis();
9         medirLux();
10        detectarMovimiento();
11
12        if (millis() - ultimoMovimiento >= tiempoInactividad &&
13            !lucesApagadasPorInactividad) {
14            apagarTodasLasLuces();
15            lucesApagadasPorInactividad = true;
16        }
17
18        if (controlLed9Activo) actualizarLed9();
19    }
20 }

```

La función `loop()` constituye el núcleo operativo del sistema embebido. En lugar de agrupar toda la lógica en un único bloque monolítico, se han modularizado las tareas principales, facilitando así su comprensión, depuración y posible ampliación futura.

En primer lugar, se verifica y mantiene activa la conexión con el broker MQTT, lo que asegura la comunicación continua con el sistema remoto. A continuación, se llama a la función `controlarCaja()`, que gestiona el comportamiento de una caja motorizada en función de entradas externas (mensajes o sensores).

El sistema emplea una temporización no bloqueante, basada en la comparación de `millis()`, para ejecutar tareas periódicas (cada 5 segundos) sin interrumpir el flujo del programa. Dentro de esta sección se incluyen la medición de luz ambiental y la detección

de movimiento, que son las principales fuentes de información para la lógica de automatización.

Una vez detectado un periodo prolongado de inactividad (sin movimiento durante un tiempo configurado), el sistema activa un modo de ahorro energético, apagando las luces y marcando su estado para evitar repeticiones innecesarias.

Por último, si está habilitado el modo automático del LED 9 (por ejemplo, control basado en condiciones de día/noche), se actualiza su estado visual de forma independiente, reforzando así el carácter modular y adaptativo del sistema.

Conclusión: La función `loop()` coordina las operaciones reactivas y periódicas del sistema: mantenimiento de conexión, control de actuadores, gestión de sensores y ejecución de automatismos energéticamente eficientes. Cada responsabilidad se delega en funciones separadas, lo que mejora la legibilidad, facilita el mantenimiento y permite la evolución del sistema sin comprometer su estabilidad.

3.8.3. Función `medirLux()`

La función `medirLux()` permite medir la intensidad de luz ambiental utilizando un sensor analógico de luz (como un fototransistor o una fotorresistencia), convirtiendo el valor leído a una estimación en lux y enviándola al servidor MQTT. Esta información puede ser utilizada por otros nodos del sistema para ajustar iluminación, activar alertas o tomar decisiones contextuales.

```

1 void medirLux() {
2     valorSensor = analogRead(sensorPin);
3     voltaje = (valorSensor / 4095.0) * 3.3;
4     lux = (600.0 * voltaje) / 2.3;
5     char luxMessage[8];
6     dtostrf(lux, 1, 2, luxMessage);
7     client.publish(luxTopic, luxMessage);
8     Serial.print("Lux: ");
9     Serial.println(luxMessage);
10 }
```

3.8.4. Función `detectarMovimiento()`

Esta función tiene por objeto decodificar el valor del sensor de movimiento y, en caso de detección, reiniciar el contador de inactividad que apaga automáticamente las luces tras un período sin movimiento. Además, publica el estado del sensor al broker MQTT y brinda retroalimentación por consola.

```

1 void detectarMovimiento() {
2     int sensorValue = digitalRead(SENSOR_PIN);
3
4     // Si se detecta movimiento, actualiza el tiempo
5     if (sensorValue == HIGH) {
6         ultimoMovimiento = millis();
7         lucesApagadasPorInactividad = false;
8     }
9 }
```

```

9
10 // Publicar al broker MQTT como lo hacías
11 char motionState[2];
12 sprintf(motionState, "%d", sensorValue);
13 client.publish(motionTopic, motionState);
14
15 Serial.print("Movimiento: ");
16 Serial.println(sensorValue ? "Detectado" : "No detectado");
17 }

```

3.8.5. Control de Caja Motorizada: controlarCaja(), moverCaja() y detenerMotorCaja()

Este conjunto de funciones controla el comportamiento de una caja motorizada utilizando dos sensores digitales que indican su posición. Con esta información, el sistema es capaz de determinar si la caja está completamente cerrada, abierta, en movimiento o en un estado inválido debido a una lectura inconsistente de los sensores.

La señal `abrirCaja`, enviada desde Home Assistant a través de MQTT, indica si el sistema debe proceder a abrir o cerrar la caja. Según esta señal y el estado actual de la caja, se ejecutan las acciones correspondientes.

Para mejorar la claridad del código y evitar duplicación de lógica, se han modularizado las acciones relacionadas con el movimiento de los motores. En particular, se han definido las funciones `moverCaja(int velocidad)` y `detenerMotorCaja()` para encapsular el comportamiento de inicio y parada del motor, respectivamente. Esto permite abstraer la lógica de control físico del motor y mantener el código principal enfocado en la lógica de alto nivel (decisiones basadas en estados).

Lógica de estados:

- **CAJA_CERRADA:** ambos sensores activos (HIGH).
- **CAJA_ABIERTA:** solo el sensor 2 activo.
- **EN_TRANSICION:** ambos sensores inactivos (LOW), la caja se está moviendo.
- **ESTADO_INVALIDO:** cualquier combinación no contemplada.

Lógica de acción del motor:

- Si la caja está cerrada y debe abrirse, se activa el motor en dirección de apertura.
- Si está abierta y debe cerrarse, se activa en sentido contrario.
- Si está en movimiento (transición), el motor actúa conforme a la orden `abrirCaja`.
- Si el estado es inválido, se detiene el motor por seguridad.

3.8.6. Función callback()

La función `callback()` es invocada automáticamente cada vez que el cliente MQTT recibe un mensaje en uno de los tópicos suscritos. Su objetivo es interpretar el mensaje, identificar el tópico, y ejecutar la acción correspondiente sobre luces, motores o la caja motorizada.

```

1 void callback(char* topic, byte* payload, unsigned int length) {
2     String message;
3     for (int i = 0; i < length; i++) {
4         message += (char)payload[i];
5     }
6
7     Serial.print(" MQTT [");
8     Serial.print(topic);
9     Serial.print("]: ");
10    Serial.println(message);

```

Lectura del mensaje:

- Se convierte el arreglo de bytes `payload` en un `String` legible (`message`).
 - Se imprime el contenido y el tópico desde el que fue recibido.
-

```

1 if (String(topic) == topic_luz_nocturna) {
2     if (message == "encender") {
3         controlLed9Activo = true;
4         actualizarLed9();
5         Serial.println(" Luz nocturna activada (azul o rojo según hora)");
6     } else if (message == "apagar") {
7         controlLed9Activo = false;
8         pixels.setPixelColor(8, pixels.Color(0, 0, 0));
9         pixels.show();
10        Serial.println(" Luz nocturna desactivada");
11    }
12    return;
13 }

```

Control de luz nocturna:

- Si el mensaje es `encender`, se activa el modo automático del LED 9.
 - Si el mensaje es `apagar`, el LED 9 se apaga manualmente.
-

```

1 if (String(topic) == topic_motor2) {
2     procesarMotor(message, gpioDir3, gpioDir4,
3     MCPWM_UNIT_1, MCPWM_TIMER_1, MCPWM_OPR_A);
4     return;
5 }

```



Control del segundo motor:

- El mensaje se pasa a la función `procesarMotor()`, que controla el motor conectado al conjunto de pines de la caja o un eje secundario.

```

1  if (String(topic) == topic_motor) {
2      velocidadCaja = constrain(message.toInt(), 0, 100);
3      Serial.print(" Velocidad caja: ");
4      Serial.println(velocidadCaja);
5
6      if (velocidadCaja == 0) {
7          Serial.println(" Velocidad 0 - Deteniendo motor");
8          detenerMotorCaja();
9      }
10     return;
11 }
```

Ajuste de velocidad del motor principal (caja):

- Se actualiza la variable `velocidadCaja` según el mensaje recibido.
- Si se recibe velocidad 0, el motor se detiene inmediatamente como medida de seguridad.

```

1  if (String(topic) == topic_estado_caja) {
2      abrirCaja = (message == "abrir");
3      Serial.print(" Acción recibida: ");
4      Serial.println(abrirCaja ? "ABRIR" : "CERRAR");
5      return;
6 }
```

Control de estado de la caja:

- Cambia el valor de `abrirCaja` según el mensaje recibido ("abrir" o "cerrar").
- Esto afectará la lógica de movimiento evaluada en la función `controlarCaja()`.

3.8.7. Función procesarMensajeJSON()

La función `procesarMensajeJSON()` es responsable de interpretar un mensaje JSON recibido a través de MQTT para controlar dinámicamente la iluminación de un LED. Extrae del mensaje los valores RGB deseados y los aplica al LED correspondiente, determinado por el tópico del mensaje. Además de modificar el estado visual, esta función también reinicia el temporizador de inactividad del sistema.

Este doble propósito es clave en entornos domóticos: por un lado, permite controlar la iluminación desde plataformas como Home Assistant mediante comandos estructurados en JSON; por otro, reconoce que la intervención manual implica actividad del usuario, por

lo que reinicia el temporizador que evita el apagado automático por inactividad. De este modo, se integra de forma armoniosa el control automático con la interacción humana, priorizando la experiencia del usuario y asegurando una respuesta adecuada del sistema ante comandos remotos.

```

1 void procesarMensajeJSON(char* topic, StaticJsonDocument<256>& doc) {
2     String state = doc["state"];
3
4
5     int r = doc["color"]["r"];
6     int g = doc["color"]["g"];
7     int b = doc["color"]["b"];
8
9     if (r < 0 || r > 255 || g < 0 || g > 255 || b < 0 || b > 255) return;
10
11    actualizarLed(topic, r, g, b);
12
13    // Reset por actividad manual desde Home Assistant
14    ultimoMovimiento = millis();
15    lucesApagadasPorInactividad = false;
16
17    Serial.println(" Actividad manual detectada: reiniciando temporizador");
18 }
```

3.8.8. Función actualizarLed()

La función `actualizarLed()` permite controlar individualmente cada LED de una tira NeoPixel en función del tópico MQTT recibido y de los valores RGB especificados. Su objetivo principal es establecer el color deseado en el LED correspondiente, lo cual permite implementar un control remoto y dinámico de la iluminación desde una plataforma de automatización, como Home Assistant.

La lógica de la función consiste en recorrer una lista de tópicos previamente definidos, comparando cada uno con el tópico recibido. Si se detecta una coincidencia, se actualiza el color del LED asociado a esa posición con los valores RGB proporcionados. Tras aplicar el color, se llama al método `show()` para reflejar físicamente el cambio, y se finaliza la función inmediatamente para optimizar la ejecución.

Esta función actúa como puente entre los mensajes MQTT (enviados como parte de comandos remotos) y la respuesta visual del sistema, siendo esencial para ofrecer un control granular e inteligente de la iluminación en tiempo real.

```

1 void actualizarLed(const char* topic, int r, int g, int b) {
2     for (int i = 0; i < 9; i++) {
3         if (String(topic) == ledControlTopic[i]) {
4             pixels.setPixelColor(i, pixels.Color(r, g, b));
5             pixels.show();
6             return;
7         }
8     }
9 }
```



3.8.9. Función actualizarLed9()

La función `actualizarLed9()` gestiona la iluminación del noveno LED de una tira NeoPixel en función del estado ambiental almacenado en la variable `estadoDiaNoche`. Su finalidad es proporcionar una indicación visual del entorno, utilizando colores simbólicos: azul para representar la noche y rojo para el día.

El funcionamiento se basa en una condición que evalúa si el valor de la variable indica un estado nocturno. En tal caso, el LED se configura en azul; en caso contrario, se establece en rojo. Finalmente, el método `show()` aplica el cambio de color a la tira de LEDs.

Esta función forma parte de una lógica de automatización que facilita la adaptación del entorno según el contexto horario, lo que puede integrarse con sensores físicos o servicios externos como APIs meteorológicas. De este modo, se contribuye tanto a la funcionalidad estética como a la información contextual dentro del sistema domótico.

```

1 void actualizarLed9() {
2     if (estadoDiaNoche == "es de noche") {
3         pixels.setPixelColor(8, pixels.Color(0, 0, 255));
4     } else {
5         pixels.setPixelColor(8, pixels.Color(255, 0, 0));
6     }
7     pixels.show();
8 }
```

3.8.10. Función apagarTodasLasLuces()

La función `apagarTodasLasLuces()` se encarga de desactivar los 9 LEDs RGB del sistema, estableciendo su color en negro (`RGB(0, 0, 0)`), lo que equivale a apagarlos visualmente. Esta operación se realiza recorriendo todos los LEDs mediante una estructura de repetición, donde se envía el comando de apagado individualmente a cada uno. Al finalizar el proceso, se imprime un mensaje en el monitor serial para confirmar la ejecución.

Este mecanismo se utiliza típicamente como respuesta a situaciones de inactividad prolongada, actuando como medida de ahorro energético y reforzando el comportamiento automatizado del sistema domótico. Su implementación permite reducir el consumo eléctrico y prolongar la vida útil de los componentes LED cuando no se requiere iluminación activa.

```

1 void apagarTodasLasLuces() {
2     for (int i = 0; i < 9; i++) {
3         actualizarLed(ledControlTopic[i], 0, 0, 0); // Apaga cada LED
4     }
5     Serial.println(" Todas las luces apagadas por inactividad");
6 }
```

3.8.11. Función reconnect()

Esta función asegura que el cliente MQTT permanezca conectado. Si la conexión se pierde, intenta reconectarse al servidor de manera indefinida hasta lograrlo. Es fundamental para mantener la comunicación en sistemas IoT confiables.

```

1 void reconnect() {
2     while (!client.connected()) {
3         Serial.print("Conectando al servidor MQTT... ");
4         if (client.connect("ESP32_Client", mqtt_username, mqtt_password)) {
5             Serial.println(" Conectado");
6         } else {
7             Serial.print(" Fallo, rc=");
8             Serial.print(client.state());
9             Serial.println(" Intentando de nuevo en 5 segundos");
10            delay(5000);
11        }
12    }
13 }
```

La función `reconnect()` implementa un bucle que se ejecuta continuamente mientras el cliente MQTT no esté conectado. Dentro del bucle, el ESP32 intenta establecer una conexión con el broker MQTT utilizando un identificador y las credenciales configuradas. Si la conexión es exitosa, se informa por el monitor serial y se sale del bucle. En caso contrario, se muestra el código de error correspondiente y se espera cinco segundos antes de intentar de nuevo. Este mecanismo asegura que, ante una pérdida de conexión por fallos de red, reinicios del broker o cortes de energía, el dispositivo se reconecte automáticamente de forma indefinida, proporcionando robustez y resiliencia al sistema de comunicación.

3.9. Interfaz de Usuario en Home Assistant (Lovelace)

Para la visualización y control del sistema domótico se utilizó la interfaz gráfica **Lovelace**, incluida por defecto en Home Assistant [25]. En una etapa inicial se exploraron opciones adicionales desde el repositorio *HACS* (Home Assistant Community Store) para mejorar el frontend. Sin embargo, se optó por conservar la interfaz predeterminada de Home Assistant, debido a su diseño minimalista, intuitivo y suficientemente funcional para las necesidades del proyecto.

Una de las grandes ventajas de Lovelace es que detecta automáticamente las nuevas entidades definidas en los archivos de configuración YAML, siempre que el sistema se reinicie correctamente. Esto simplifica el proceso de incorporación de dispositivos y automatizaciones a la interfaz, especialmente aquellas entidades generadas mediante integración MQTT [18].

3.10. Configuración de Tarjetas de Entidades

Durante el desarrollo del sistema domótico, se decidió realizar una reestructuración completa del archivo `configuration.yaml` para mejorar la organización, legibilidad y mantenimiento del código. En lugar de mantener toda la configuración centralizada, se optó por dividirla en módulos temáticos mediante directivas `!include`. Esta práctica no solo facilita la depuración de errores, sino que también reduce la probabilidad de fallos de compilación ante configuraciones extensas o desordenadas [17].

Para facilitar este trabajo, se utilizó el complemento **File Editor**, disponible como add-on oficial en Home Assistant, que permite editar archivos YAML directamente desde la interfaz web del sistema [26]. En paralelo, se habilitó el complemento **Terminal & SSH**, el cual permitió acceder al sistema de archivos completo mediante una consola

remota integrada, útil para tareas de depuración y operaciones avanzadas como la revisión de logs o reinicio manual de servicios en caso de fallo [27].

Además, para estructurar la información y facilitar el control de los distintos dispositivos del sistema desde la interfaz gráfica, se emplearon tarjetas del tipo **entities**. Todas siguen el siguiente formato general:

```

1 type: entities
2 entities:
3   - entidad_1
4   - entidad_2
5   - ...

```

A continuación, se detallan las entidades agrupadas según su funcionalidad:

- **Luces RGB (LEDs)** light_led_1 a light_led_9
- **Control de la Caja Electrónica** switch.caja_abrir, number.velocidad_caja
- **Velocidad del Segundo Motor** number.velocidad_motor
- **Sensores Binarios** binary_sensor_movimiento, binary_sensor_luz_ambiente
- **Sensor Analógico** sensor.luz_ambiente
- **Automatización Día/Noche vía MQTT** estado_dia_o_noche_mqtt

3.10.1. Resumen de Diseño

La configuración se centró en una interfaz simple, funcional y clara, priorizando la agrupación lógica de entidades. Se evitó el uso de tarjetas personalizadas innecesarias para mantener la estabilidad y simplicidad del sistema. Esto también asegura que el sistema sea fácilmente comprensible por otros usuarios o técnicos que necesiten intervenir en el futuro.

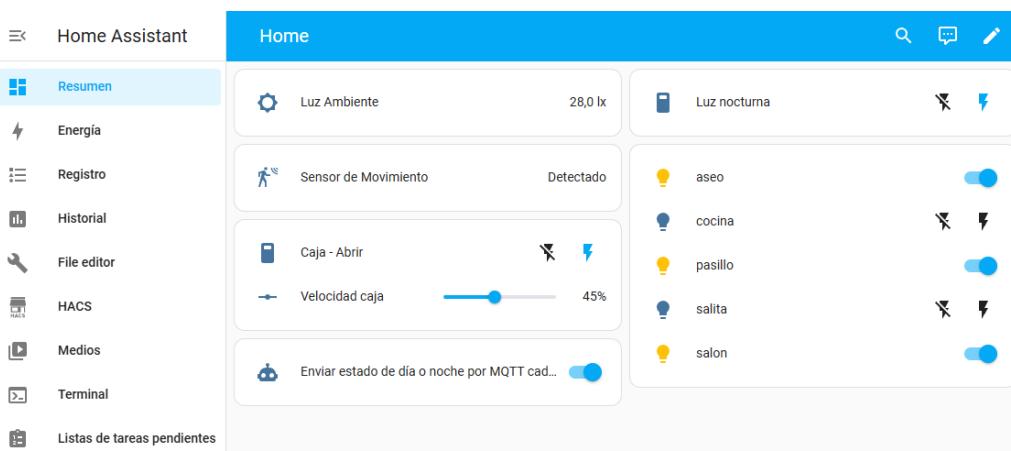


Figura 24: Captura de pantalla de la interfaz Lovelace configurada

En la interfaz se pueden observar en tiempo real los niveles de distintos sensores integrados en la maqueta, como el sensor de luz ambiental. Además, se han implementado automatizaciones que permiten, por ejemplo, activar distintos modos según el momento

del día (modo día/noche), los cuales se gestionan mediante llamadas a la API de Home Assistant.

A continuación, se muestran dos capturas de la interfaz Lovelace donde se aprecian estos elementos:



Figura 25: Visualización de sensor luz.

| 13 de junio de 2025 | |
|---------------------|--|
| | Enviar estado de día o noche por MQTT cada 5 minutos disparado por el estado de Sun 12:36:05 - Hace 4 minutos - Trazas |
| | Enviar estado de día o noche por MQTT cada 5 minutos disparado por patrón de tiempo 12:35:01 - Hace 5 minutos - Trazas |
| | Enviar estado de día o noche por MQTT cada 5 minutos disparado por el estado de Sun 12:32:05 - Hace 8 minutos - Trazas |
| | Enviar estado de día o noche por MQTT cada 5 minutos disparado por patrón de tiempo 12:30:00 - Hace 10 minutos - Trazas |
| | Enviar estado de día o noche por MQTT cada 5 minutos disparado por el estado de Sun 12:28:05 - Hace 12 minutos - Trazas |
| | Enviar estado de día o noche por MQTT cada 5 minutos disparado por patrón de tiempo 12:25:00 - Hace 15 minutos |
| | Enviar estado de día o noche por MQTT cada 5 minutos disparado por el estado de Sun 12:24:05 - Hace 16 minutos |
| | Enviar estado de día o noche por MQTT cada 5 minutos disparado por el estado de Sun 12:20:05 - Hace 20 minutos |
| | Enviar estado de día o noche por MQTT cada 5 minutos disparado por patrón de tiempo 12:20:00 - Hace 20 minutos |
| | Enviar estado de día o noche por MQTT cada 5 minutos disparado por el estado de Sun 12:16:04 - Hace 24 minutos |

Figura 26: Automatizaciones activas como el modo día/noche configuradas mediante la API de Home Assistant.

3.11. Conexionado ESP32

3.11.1. Esquema general de conexiones

El sistema está basado en un microcontrolador ESP32 al que se conectan diversos sensores y actuadores. La Figura 28 muestra un esquema simplificado del conexionado físico del prototipo. Con el fin de facilitar la comprensión visual, no se han incluido las conexiones de alimentación en el esquema. Sin embargo, es importante destacar que todos los sensores y actuadores están alimentados mediante una fuente externa de 5 V, y todas las tierras (GND) están conectadas entre sí, incluyendo la del ESP32 y la fuente externa, garantizando así una referencia común.

- **LEDs Neopixel:** Conectados al pin GPIO 2 del ESP32.
- **Sensor PIR:** Su señal de salida está conectada al pin GPIO 35.
- **Sensor de luz:** El pin de señal se conecta al GPIO 34.
- **Sensor magnético 1:** Conectado al pin GPIO 12.
- **Sensor magnético 2:** Conectado al pin GPIO 13.
- **Puente H (control del motor):**
 - PWM: Conectado al pin GPIO 18.
 - DIR1: Conectado al pin GPIO 5.
 - DIR2: Conectado al pin GPIO 17.
- **Motor DC:** Conectado directamente a las salidas del puente H.

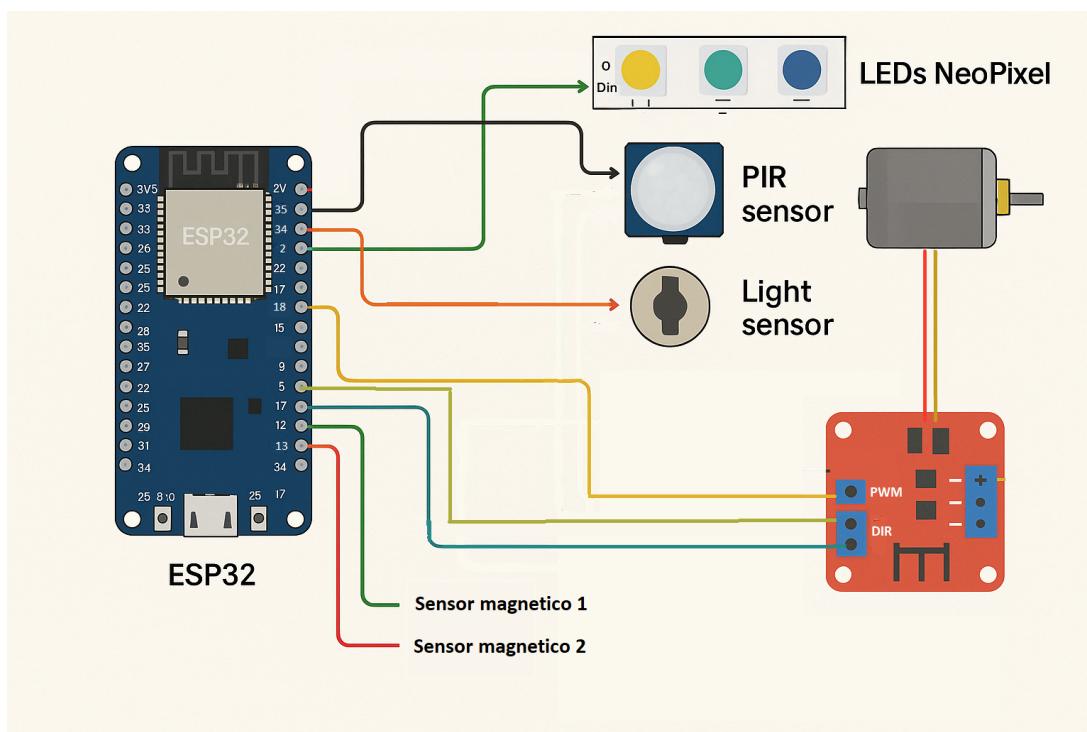


Figura 27: Esquema de conexión entre el ESP32 y los diferentes sensores y actuadores.

3.11.2. Descripción de la maqueta y distribución de los componentes

La maqueta desarrollada para este proyecto tiene como objetivo principal representar físicamente una vivienda domotizada a escala. Está compuesta por varias estancias claramente delimitadas, cada una equipada con elementos electrónicos que simulan las funcionalidades de una instalación domótica real.

En la parte superior de la estructura se encuentran únicamente los módulos de iluminación LED. El microcontrolador ESP32 no se encuentra integrado en la maqueta, sino que se conecta externamente mediante un cable, lo que facilita su programación, mantenimiento y sustitución.

Para garantizar un control preciso del estado del cajón móvil que contiene parte del sistema de control, se han incorporado dos sensores magnéticos de final de carrera que permiten detectar si está completamente abierto o cerrado. Además, se ha añadido un sensor de luz ambiental que permite medir el nivel de iluminación general del entorno, proporcionando información útil para la automatización adaptativa.

El sistema de iluminación de cada estancia está compuesto por tiras de LEDs Neopixel, controladas de forma individual por el ESP32.

Este tipo de instalación busca representar los principios de la automatización del hogar moderno, como los planteados en otros estudios similares [3, 4]. En dichos trabajos se explora el potencial del Internet de las Cosas para mejorar la eficiencia, accesibilidad y gestión energética dentro del entorno doméstico.

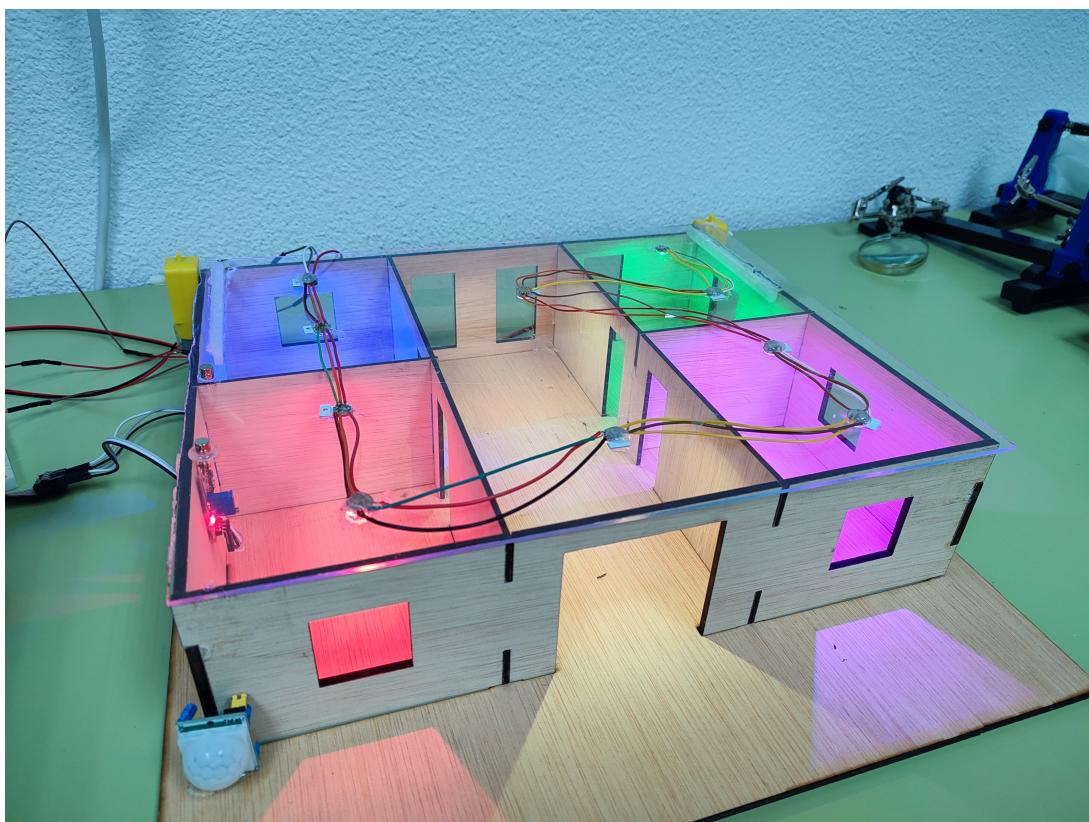


Figura 28: Vista general de la maqueta domotizada desarrollada.

4. Presupuesto

A continuación se detalla el presupuesto estimado para el desarrollo del Trabajo Fin de Grado (TFG), incluyendo los componentes electrónicos necesarios junto con su precio y enlace de compra:

| Componente | Precio (€) | Enlace |
|--------------------------------|---------------|--|
| Raspberry Pi 3 + fuente + caja | 54.72 | https://es.aliexpress.com/item/1005007655721767.html |
| ESP32 | 3.79 | https://es.aliexpress.com/item/1005006347872239.html |
| Tarjeta SD 32GB Clase 10 | 7.94 | https://www.pccomponentes.com/ |
| Puente H L298N | 2.96 | https://es.aliexpress.com/item/1005008756464145.html |
| Detector de luz TEMT6000 | 0.52 | https://es.aliexpress.com/item/1005001565357867.html |
| Imanes de neodimio | 1.65 | https://es.aliexpress.com/item/1005008955502063.html |
| 2x Detector imán KY-025 | 1.02 | https://es.aliexpress.com/item/1005008685008619.html |
| Fuente alimentación 5V | 3.49 | https://es.aliexpress.com/item/1005007805634378.html |
| Motor DC piñón-cremallera | 3.29 | https://es.aliexpress.com/item/1005007676181805.html |
| Matriz NeoPixel 8x8 (64 LEDs) | 47.85 | https://ultra-lab.net/producto/neopixel-neomatrix-8x8-64-led-rgb/ (0.75€/LED) |
| Total | 126.33 | |

Cuadro 2: Presupuesto detallado del TFG

5. Conclusiones y Trabajo Futuro

Conclusiones

El desarrollo del sistema de automatización para maquetas arquitectónicas ha demostrado ser una solución eficaz para ilustrar conceptos de domótica en entornos educativos y de exposición. La utilización de microcontroladores ESP32, junto con la integración de sensores y actuadores, ha permitido implementar un entorno controlado, interactivo y adaptable. La plataforma Home Assistant ha facilitado la gestión centralizada, aportando flexibilidad, escalabilidad y una interfaz intuitiva para el usuario final.

Este proyecto ha cumplido con los objetivos propuestos: diseñar un sistema funcional, económico y replicable que pueda adaptarse a distintas configuraciones arquitectónicas. Además, ha servido como base para validar la viabilidad del uso de tecnologías de automatización en modelos físicos a escala.

Trabajo Futuro

A partir del trabajo realizado, se abren múltiples líneas de mejora y expansión:

- **Optimización del consumo energético:** mediante modos de bajo consumo en los ESP32 y una gestión más eficiente del encendido de componentes.
- **Incorporación de nuevas tecnologías de comunicación:** como Zigbee o Z-Wave para mejorar la robustez de la red de sensores y actuadores.
- **Desarrollo de funcionalidades avanzadas:** incluyendo algoritmos de inteligencia artificial para detección de patrones de uso o mantenimiento predictivo.
- **Mejoras en la interfaz gráfica:** para permitir una interacción más rica y visualmente atractiva mediante dashboards personalizables.
- **Aplicación en otros entornos:** como viviendas reales, instalaciones industriales a pequeña escala, o demostradores tecnológicos en ferias.

Este trabajo sienta las bases para futuras investigaciones en el campo de la automatización inteligente aplicada a entornos físicos reducidos.

6. Bibliografía

Referencias

- [1] IAmAnufacturing. (2024). *Tipos de maquetas interactivas*. Recuperado de <https://www.iamanufacturing.com/blog/tipos-de-maquetas-interactivas/> [Accedido el 12 de junio de 2025].
- [2] Maquetas.Tech. (2024). *Maquetas Interactivas*. Recuperado de <https://www.maquetas.tech/maquetas-interactivas/> [Accedido el 12 de junio de 2025].
- [3] Escobar, G. (2018). *Sistema de monitoreo energético y control domótico basado en tecnología “Internet de las Cosas”*. Revista Investigación & Desarrollo de la Universidad Privada Boliviana.
- [4] Anglés Millán, S. (2014). *Metodología y criterios para evaluar la influencia de la domótica y su preinstalación en los edificios*. Tesis doctoral. ETSAM, Madrid.
- [5] Fernández, R. (2021). *Smart home: gasto mundial en hogares inteligentes 2015–2025*. Statista. <https://es.statista.com/estadisticas/1118032/gasto-global-en-hogares-inteligentes>
- [6] Fernández, R. (2021). *Número de smart homes en el mundo 2017–2025*. Statista. <https://es.statista.com/estadisticas/573159/evolucion-del-numero-de-hogares-inteligentes-a-nivelmundial>
- [7] CEDOM. (2013). *¿Qué es la Domótica?*. <http://www.cedom.es>
- [8] Instituto Nacional de Estadística (INE). (2022). *Encuesta TIC en los Hogares*. <https://www.ine.es/prensa/prensa.htm>
- [9] HiveMQ. (s.f.). *MQTT Protocol*. <https://www.hivemq.com/mqtt/mqtt-protocol/>
- [10] Fernández, Y. (s.f.). *API: qué es y para qué sirve*. <https://www.xataka.com/basics/api-que-sirve>
- [11] Unión Europea. (2018). *Directiva 2018/844 EU*. <https://www.boe.es/doue/2018/156/L00075-00091.pdf>
- [12] Gobierno de España. (2022). *R.D 42/2022, de 18 de enero*. <https://www.boe.es/eli/es/rd/2022/01/18/42/con>
- [13] Recuero, A. (1999). *Estado actual y perspectivas de la domótica*. Informes de la Construcción, 50(459), 9–21.
- [14] Junstrand, S., Passaret, X., & Vázquez, D. (2005). *Domótica y hogar digital*. Parainfo.
- [15] Junstrand, S., Passaret, X., & Vázquez Álvarez, D. (2004). *Domótica y hogar digital*. Ed. Paraninfo.
- [16] Home Assistant. *Documentación oficial de la plataforma*. <https://www.home-assistant.io/docs/>

- [17] Home Assistant. *Configuración del archivo configuration.yaml.* <https://www.home-assistant.io/docs/configuration/basic/>
- [18] Home Assistant. *Integración MQTT.* <https://www.home-assistant.io/integrations/mqtt/>
- [19] Home Assistant. *Broker MQTT Mosquitto (Add-on).* <https://www.home-assistant.io/addons/mosquitto/>
- [20] Home Assistant. *Automatizaciones.* <https://www.home-assistant.io/docs/automation/>
- [21] Home Assistant. *Scripts.* <https://www.home-assistant.io/docs/scripts/>
- [22] Home Assistant. *Sensores personalizados MQTT.* <https://www.home-assistant.io/integrations/sensor.mqtt/>
- [23] Home Assistant. *Actuadores y switches MQTT.* <https://www.home-assistant.io/integrations/switch.mqtt/>
- [24] Home Assistant. *Iluminación RGB con MQTT.* <https://www.home-assistant.io/integrations/light.mqtt/>
- [25] Home Assistant. *Configuración de dashboards (Lovelace UI).* <https://www.home-assistant.io/lovelace/>
- [26] Home Assistant. *Add-on File Editor para editar archivos YAML.* https://github.com/home-assistant/addons/tree/master/file_editor
- [27] Home Assistant. *Terminal y SSH Add-on.* <https://www.home-assistant.io/addons/ssh/>
- [28] Espressif. *ESP32-WROOM-32 Datasheet.* https://www.espressif.com/sites/default/files/documentation/esp32-wroom-32_datasheet_en.pdf
- [29] Raspberry Pi Foundation. *Raspberry Pi 3 B+ Datasheet.* <https://datasheets.raspberrypi.com/rpi3/raspberry-pi-3-b-plus-product-brief.pdf>
- [30] Adafruit. *WS2812B Datasheet.* <https://cdn-shop.adafruit.com/datasheets/WS2812B.pdf>
- [31] Joy-IT. *Sensor Magnético KY-025.* <https://datasheet4u.com/datasheet/Joy-IT/KY-025-1402036>
- [32] Vishay. *Sensor de Luz TEMT6000.* <https://www.vishay.com/docs/81579/temt6000.pdf>
- [33] STMicroelectronics. *L298N Puente H Datasheet.* <https://www.st.com/resource/en/datasheet/1298.pdf>

7. Repositorio del Proyecto

Todo el código fuente desarrollado para este Trabajo de Fin de Grado se encuentra disponible en el siguiente repositorio público de GitHub:

<https://github.com/miguellopezmartinez1999/Centro-Control-Dom-tico>

El repositorio incluye:

- Código del ESP32 para sensores, actuadores y comunicación MQTT.
- Configuración modular del sistema Home Assistant.
- Scripts de automatización , entidades personalizadas y archivos YAML.
- Backup con todos los datos del Home Assitant.
- Video demostración apertura y cierre de la caja.