Escuela Politécnica Superior de Linares

Grado en Ingeniería Telemática

UNIVERSIDAD DE JAÉN
Escuela Politécnica Superior de Linares

Trabajo Fin de Grado

SISTEMA DOMÓTICO A ESCALA CON ACCESO REMOTO

Alumno: **Elyssar Myrna Thabet**

Tutor: Prof. D. Ildefonso Ruano Ruano
Depto.: Ingeniería de Telecomunicación

Junio, 2024

Agradecimientos

Quiero expresar mi más sincero agradecimiento a todos los profesores de la Escuela politécnica superior de Linares, que, con su dedicación y conocimientos, han contribuido a mi formación académica a lo largo de estos años.

Agradezco especialmente a mi tutor Prof. D. Ildefonso Ruano Ruano, cuyo apoyo y orientación han sido fundamentales para la realización de este trabajo.

Un agradecimiento muy especial a Pedro Aguilar Aguilar por su constante ayuda en la creación y el montaje del prototipo de este proyecto, y por su motivación.

Finalmente, quiero agradecer profundamente a mi familia, cuyo incondicional apoyo y amor me han permitido llegar hasta aquí: a mi madre, por su incansable dedicación, sus sabios consejos y su amor incondicional que me han dado la fuerza para superar cada obstáculo; a mi padre, por su ejemplo de trabajo duro, su paciencia y su constante ánimo que siempre me ha motivado a dar lo mejor de mí; a mi hermano, por ser mi fuente constante de inspiración y alegría, y por estar siempre a mi lado en los momentos más difíciles. A todos ustedes, mi más sincero agradecimiento y todo mi amor.

Resumen

En un mundo cada vez más interconectado y orientado hacia la eficiencia energética, la automatización residencial se ha erigido como un campo de investigación y desarrollo de vital importancia. La convergencia de tecnologías avanzadas, como la energía solar, sensores, actuadores y sistemas de control remoto, ha permitido la creación de viviendas inteligentes que no solo mejoran la comodidad y la calidad de vida de sus ocupantes, sino que también contribuyen a la preservación del medio ambiente y al cumplimiento de los Objetivos de Desarrollo Sostenible (ODS) establecidos por las Naciones Unidas.

En este contexto, el presente proyecto de Trabajo de Fin de Grado (TFG) se centra en la creación de una maqueta a escala reducida de una vivienda inteligente, que incorpora una amplia gama de sensores y actuadores controlables a través de un sistema de bajo costo con conectividad a Internet, permitiendo así la interconexión y el control de dispositivos, lo que se conoce como Internet de las Cosas (IoT). También se ha desarrollado una aplicación web denominada "**ConnectGuard Residence**" que permite el acceso y control remoto del prototipo, mejorando, además, su seguridad.

Adicionalmente, se han utilizado técnicas de Inteligencia artificial (IA), concretamente se ha usado un modelo de Machine Learning (ML) de reconocimiento facial que funciona como un sistema biométrico para controlar el acceso al interior de la vivienda.

El propósito fundamental de trabajo es mostrar un ejemplo que permite explorar las posibilidades tecnológicas y los beneficios que la automatización residencial a través de IoT ofrece a la vida cotidiana. Desde la generación de energía sostenible mediante la captación de luz solar hasta la implementación de sistemas de control inteligente, este proyecto se sumerge en el apasionante mundo de la domótica, donde la innovación tecnológica se combina con la búsqueda de soluciones que promuevan un entorno más eficiente y sostenible. La maqueta obtenida como fruto de este trabajo trata de ser un ejemplo real que muestre una parte de las posibilidades que ofrecen los avances tecnológicos al incorporar técnicas domóticas, IoT e IA para, en este caso, obtener un hogar inteligente (Smart home) y sostenible.

Esta memoria presenta en detalle la metodología seguida para alcanzar los objetivos propuestos, destacando aspectos de diseño, implementación, pruebas y validación de la maqueta. Además, constituye un ejemplo del avance de la domótica y la eficiencia energética, ofreciendo una perspectiva de futuro en la creación de hogares conectados y más sostenibles.

Palabras clave: Domótica – Internet de las Cosas - Viviendas inteligentes - Automatización residencial - Sensores – Actuadores - Innovación tecnológica - Energía solar - Eficiencia energética – Sostenibilidad

Abstract

In an increasingly interconnected world focused on energy efficiency, residential automation has emerged as a field of vital importance for research and development. The convergence of advanced technologies, such as solar energy, sensors, actuators, and remote-control systems, has enabled the creation of smart homes that not only enhance the comfort and quality of life for their occupants but also contribute to environmental preservation and the achievement of the Sustainable Development Goals (SDGs) set by the United Nations.

In this context, this Final Degree Project (TFG) focuses on creating a small-scale model of a smart home, incorporating a wide range of sensors and actuators controllable through a low-cost system with Internet connectivity, thus allowing the interconnection and control of devices known as the Internet of Things (IoT). Additionally, a web application called "ConnectGuard Residence" has been developed to enable remote access and control of the prototype, enhancing its security.

Furthermore, artificial intelligence (AI) techniques have been utilized, specifically a facial recognition Machine Learning (ML) model that functions as a biometric system to control access to the interior of the home.

The fundamental purpose of this work is to showcase an example that explores the technological possibilities and benefits that residential automation through IoT offers to daily life. From the generation of sustainable energy through solar capture to the implementation of intelligent control systems, this project delves into the exciting world of home automation, where technological innovation combines with the pursuit of solutions that promote a more efficient and sustainable environment. The model resulting from this work aims to be a real example that demonstrates some of the possibilities offered by technological advances when incorporating home automation, IoT, and AI techniques to create a smart and sustainable home.

This report presents in detail the methodology followed to achieve the proposed objectives, highlighting aspects of design, implementation, testing, and validation of the model. Additionally, it serves as an example of the advancement of home automation and energy efficiency, offering a future perspective on the creation of more connected and sustainable homes.

Keywords: Home Automation - Internet of Things - Smart Homes - Residential Automation - Sensors - Actuators - Technological Innovation - Solar Energy - Energy Efficiency - Sustainability.

Índice

Índice de figuras

Índice de tablas

1 Introducción

Este proyecto está basado en el diseño y creación de una Smart Home usando algunos criterios de desarrollo sostenible. Para concretar más, se enfoca en la creación de una maqueta de vivienda inteligente sostenible (Figura 1) que se sustenta en el paradigma de la automatización del hogar (domótica) usando elementos de Internet de las Cosas (IoT) y, aunque en menor medida, de Inteligencia Artificial (IA). En este entorno es donde se incluyen una variedad de dispositivos, sensores y actuadores que colaboran en armonía para mejorar la funcionalidad y eficiencia de un hogar simulado.

La pieza central de este sistema es una **Raspberry Pi**, una computadora de bajo costo y alto rendimiento que se encargará de coordinar y controlar todos los dispositivos y sensores en la maqueta. La elección del lenguaje de programación Python ha sido fundamental para la flexibilidad y facilidad de desarrollo, lo que permitirá una programación precisa y personalizable para cada componente del sistema.

En la actualidad, el diseño de cualquier sistema no puede dejar de lado, el tema de la sostenibilidad, hay que asegurar las necesidades presentes sin hipotecar las del futuro. Por ese motivo, en el diseño de este trabajo se han tenido en cuenta, en la medida de lo posible, la posibilidad de hacer una implementación que ayude a la sostenibilidad. Por ejemplo, además del ahorro que puede suponer el uso de sistemas tecnológicos para el ahorro de energía, se ha incorporado células solares para captar energía en el sistema.

Figura 1 Vistas de la vivienda prototipo

Dentro de esta vivienda inteligente se encuentran los **sensores**, que desempeñan un papel fundamental en la captura de datos. Los sensores de presencia permiten detectar la ocupación de las habitaciones y la presencia de personas en la puerta principal, mientras que los sensores de captación de imágenes (en este caso las cámaras) ofrecen una vista visual del entorno, y permiten el funcionamiento del algoritmo del sistema de reconocimiento facial. Los sensores de pulsación habilitan la interacción con

el sistema y proporcionan información valiosa para las acciones de control, como por ejemplo el **pulsador** que tendrá el rol de timbre de la vivienda.

Por otro lado, los **actuadores** son los encargados de llevar a cabo las acciones físicas. Los sistemas de iluminación se pueden controlar para ajustar la luminosidad según las necesidades, también para activar o desactivar el sistema de seguridad, junto con los altavoces que proporcionan una salida de audio para reproducir los ficheros de audios correspondientes. Los motores permiten la activación de dispositivos móviles, en este caso las puertas de la vivienda. La cerradura electrónica brinda control de acceso seguro. El funcionamiento de todos los elementos se detallará en capítulos posteriores.

Uno de los aspectos más notables de esta vivienda inteligente es su capacidad de ser controlada de forma remota a través de una **aplicación web** desarrollada en el ámbito de este trabajo y llamada **ConnectGuard Residence**. Esto brinda a los usuarios la comodidad de supervisar y ajustar la iluminación, la seguridad, el acceso y otros sistemas, desde cualquier ubicación con acceso a Internet.

Además, para garantizar la seguridad y el control constante, el sistema ha sido configurado para enviar notificaciones en tiempo real a los dispositivos móviles de los usuarios. Este proceso se facilita mediante el empleo de la aplicación Pushbullet [1], que permite recibir notificaciones emergentes y estar al tanto de cualquier evento relevante relacionado con la maqueta de la vivienda.

1.1 Motivación

La creciente demanda de soluciones tecnológicas para mejorar la comodidad, la eficiencia y la seguridad en el hogar ha llevado a un aumento significativo en el interés por la domótica basada en el Internet de las Cosas (IoT). En la actualidad, las personas buscan formas innovadoras de controlar y gestionar sus entornos domésticos de manera más inteligente y sencilla, sin comprometer la seguridad de sus familias y propiedades. En este contexto, la realización de una maqueta de casa domótica basada en IoT se presenta como un proyecto altamente relevante y oportuno.

Esta iniciativa surge de la necesidad de explorar las posibilidades que ofrece la convergencia de la tecnología IoT con la automatización residencial y la seguridad, con el objetivo de diseñar una solución eficiente, accesible y adaptable a las necesidades de las personas. Además, este trabajo tiene como propósito principal contribuir al conocimiento y la comprensión de las ventajas que la domótica puede ofrecer en términos de ahorro energético, seguridad, confort y calidad de vida en el entorno del hogar.

En un momento en el que la sociedad se encuentra cada vez más orientada hacia la conectividad y la eficiencia, la realización de esta maqueta se convierte en un paso fundamental para abordar los desafíos y oportunidades que la IoT brinda al ámbito de

la vivienda inteligente, incorporando de manera destacada el componente de seguridad, que es esencial para la tranquilidad y la protección de las personas y sus propiedades.

Adicionalmente, otro elemento motivador para la realización de este Trabajo Fin de Grado fue la posibilidad de obtener un sistema real, aunque fuera a escala, en el que se pudieran mostrar un uso práctico de las tecnologías relacionadas con la domótica, IoT e incluso IA. Todas estas tecnologías están estrechamente relacionadas con muchos de los estudios impartidos en la Escuela Politécnica Superior de Linares (EPSL). La propia maqueta constituye un sencillo, pero potente escaparate y ejemplo de algunas de las posibilidades tecnológicas que se pueden desarrollar con el estudio de estas titulaciones.

1.2 Estado del Arte

A continuación, se presentarán los conceptos necesarios para comprender el proyecto y se indicará su uso en el mismo.

1.2.1 Domótica

El término "domótica" se deriva de la fusión de las palabras "domus", que significa "casa" en latín, y "automática", indicando que se refiere a la automatización en el ámbito residencial [2].

En el entorno de la domótica, se utilizan diversas tecnologías, como dispositivos electrónicos, sensores, redes de comunicación y sistemas de control, para permitir a los usuarios gestionar y controlar funciones y sistemas de manera automática o remota en su entorno. Esto incluye programar acciones como apagar las luces a una hora específica o ajustar el termostato antes de regresar a casa desde el trabajo. La automatización residencial mejora la comodidad, ahorra en costos de energía y puede aumentar la seguridad mediante dispositivos de IoT como cámaras y sistemas de seguridad [3].

Figura 2: Casa conectada

Fuente: https://www.echeverrimontes.com/blog/ventajas-y-desventajas-de-la-domotica

Se ha llevado a cabo un proyecto de domótica que automatiza tanto una vivienda como su entorno, a escala, utilizando una maqueta diseñada específicamente para este propósito.

1.2.2 Internet of Things

Internet of Things, también conocida como Internet de las Cosas, es la conexión en red de dispositivos interrelacionados mediante Internet. Estos dispositivos se denominan dispositivos IoT, y cada uno de ellos cuenta con sensores, procesadores y

capacidad de comunicación (a través de Wi-Fi, Bluetooth, 4G/5G, u otras tecnologías) que les permite recopilar información y conectarse a la red [4].

La IoT posibilita la comunicación entre estos dispositivos y los sistemas de control centralizados, lo que abre la puerta a una amplia variedad de aplicaciones.

La IoT contribuye a que las personas vivan y trabajen de manera más inteligente. Por ejemplo, los consumidores pueden utilizar dispositivos con IoT incorporada, como automóviles, relojes inteligentes o termostatos, para mejorar su calidad de vida. Por ejemplo, cuando alguien llega a casa, su automóvil podría comunicarse con el garaje para abrir la puerta; el termostato podría ajustarse a una temperatura predefinida; y la iluminación podría adaptarse a una intensidad y color más bajos.

Además de beneficiar a los consumidores, la IoT desempeña un papel crucial en las empresas: Proporciona información en tiempo real sobre el funcionamiento de sistemas, desde el rendimiento de las máquinas hasta la cadena de suministro y la logística.

La IoT continúa evolucionando a un ritmo impresionante; Con más de 7.000 millones de dispositivos IoT conectados en la actualidad, los expertos prevén que esta cifra aumente a 10.000 millones en 2020 y a 22.000 millones en 2025 [5]. Esta tecnología se ha convertido en un elemento fundamental en la transformación digital de diversos sectores, como la atención médica, la agricultura, la manufactura, la logística, la energía y la automatización del hogar. Esta tecnología se destaca como una de las más vitales, ya que cada vez más empresas se dan cuenta del potencial de los dispositivos conectados para mantener su competitividad.

- Funcionamiento de IoT:

  El funcionamiento de la Internet de las cosas (IoT) se basa en la interconexión de dispositivos y objetos a través de Internet para recopilar datos, compartir información y realizar acciones específicas:

  - Los dispositivos IoT recopilan datos relevantes para su función. Por ejemplo, un termostato inteligente puede medir la temperatura ambiente, mientras que un sensor de humedad puede registrar la humedad del suelo en una granja. Estos datos se recopilan de manera constante y pueden variar según la aplicación.
  - Los datos recopilados se transmiten a través de Internet o una red local a servidores o plataformas IoT, donde se almacenan y procesan. Los protocolos de comunicación como MQTT, HTTP, o CoAP se utilizan para garantizar la transferencia segura y eficiente de datos.
  - Los servidores o plataformas IoT procesan y analizan los datos recibidos. Esto puede incluir la identificación de patrones, el cálculo de estadísticas, la detección de eventos o el control de dispositivos basados en reglas predefinidas.
  - Basándose en los datos y el análisis, se pueden tomar diversas acciones. Estas acciones pueden ser automáticas o requerir intervención humana. Por ejemplo, un sistema de riego automático puede activarse cuando los sensores de humedad indican que el suelo está seco, o una alerta puede enviarse a un agricultor si se detecta un nivel de humedad anormal.

- Los usuarios pueden interactuar con los dispositivos IoT a través de aplicaciones móviles, interfaces web o incluso comandos de voz. Esto les permite controlar los dispositivos y recibir notificaciones en tiempo real.
- La seguridad es fundamental en la IoT, ya que implica la transmisión y el almacenamiento de datos sensibles. Se utilizan medidas de seguridad como cifrado, autenticación y control de acceso para proteger la integridad y la privacidad de los datos.

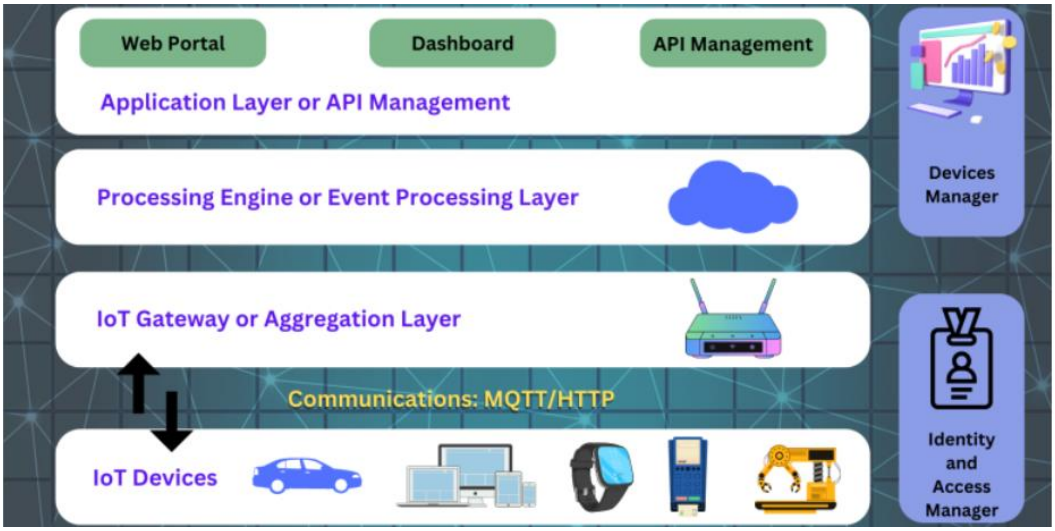


Figura 3: Arquitectura IoT

Fuente*: https://geekflare.com/es/iot-architecture/*

La relación entre IoT y la domótica es que la IoT es una tecnología clave que impulsa la domótica y permite la automatización y el control de dispositivos y sistemas en el hogar de manera más eficiente y avanzada [6].

La IoT proporciona la infraestructura y la capacidad de conexión para que la domótica funcione de manera efectiva. Los dispositivos domésticos inteligentes y sensores en un hogar conectado utilizan la IoT para recopilar datos, comunicarse entre sí y permitir que los usuarios los controlen de forma remota a través de aplicaciones móviles o interfaces en línea.

Por lo tanto, la IoT es una parte fundamental de la domótica, ya que posibilita la automatización y el control inteligente de los sistemas y dispositivos en el entorno residencial [7].

En la maqueta creada para este TFG se han incorporado diversos elementos IoT, lo que permite presentar un ejemplo sencillo pero integral de esta tecnología.

1.2.3 Inteligencia Artificial Aplicada a la Domótica

En este TFG también se integrará una inteligencia artificial (IA), por lo tanto, es fundamental proporcionar una explicación sobre este término.

La inteligencia artificial es una disciplina científica que se enfoca en desarrollar sistemas y programas capaces de realizar tareas que normalmente requieren de la inteligencia humana. Estas tareas incluyen, entre otros, el aprendizaje, la resolución de problemas, el reconocimiento de patrones, el procesamiento del lenguaje natural y la toma de decisiones.

La IA busca emular la capacidad cognitiva humana mediante algoritmos y modelos matemáticos, permitiendo a las máquinas realizar actividades complejas de manera autónoma.

La aplicación de IA en el proyecto se centra en el aprendizaje automático, también conocido como Aprendizaje Automático (ML de Machine Learning, en inglés), una rama de la inteligencia artificial que capacita a los sistemas para identificar patrones mediante datos previos. Estos algoritmos, alimentados con información, habilitan a los computadores para llevar a cabo tareas específicas de manera autónoma, sin la necesidad de programación manual [8].

Se pueden destacar los siguientes enfoques:

- **Aprendizaje Supervisado:** en este caso se proporciona al modelo un conjunto de datos etiquetados que incluye entradas y salidas deseadas. El algoritmo aprende a realizar predicciones o clasificaciones comparando sus salidas con las etiquetas proporcionadas. Este tipo de aprendizaje es más efectivo para problemas de clasificación y regresión, y ha sido el utilizado en este trabajo.

- **Aprendizaje no supervisado:** en el cual el modelo recibe datos sin etiquetas y debe encontrar patrones o estructuras por sí mismo. Se utiliza para descubrir relaciones intrínsecas en los datos, como agrupamiento o reducción de dimensionalidad. Este tipo es más apropiado cuando las etiquetas no están disponibles o son costosas de obtener.

En la IA, en el proceso de entrenamiento del modelo, la selección de un conjunto de datos adecuado es crucial, ya que influye directamente en la efectividad del modelo. Tanto la calidad como la cantidad de datos impactan en la capacidad del modelo para identificar patrones y relaciones, determinando la precisión y los resultados obtenidos.

En este prototipo, se ha implementado un modelo de aprendizaje automático supervisado para la detección biométrica. Este sistema permite abrir o mantener bloqueada una cerradura electrónica que da acceso a la vivienda, basándose en el reconocimiento de los rasgos faciales capturados por una cámara al accionar un pulsador de entrada.

2 Objetivos

El presente Trabajo de Fin de Grado tiene como objetivo principal el diseño y desarrollo de un modelo de vivienda inteligente sostenible que incorpore elementos domóticos que permitan mostrar ejemplos de diversas tecnologías como son la Automatización, Internet de las Cosas (IoT) e Inteligencia Artificial. Ante la imposibilidad de crear un sistema real como el descrito, se ha creado una maqueta a escala reducida de una vivienda, en la cual se incorporan una serie de componentes tecnológicos esenciales con los que se ha logrado trabajar con todas las tecnologías antes indicadas. Estos componentes incluyen sensores y actuadores, que serán controlados mediante un sistema de bajo coste con acceso remoto a través de la aplicación **ConnectGuard Residence**, desarrollada específicamente para este proyecto.

Los objetivos parciales que guiarán este proyecto abarcan la implementación de un sistema de generación de energía a partir de la luz solar y su posterior almacenamiento, la integración de diversos tipos de sensores, como sensores de presencia, cámaras de captación de imágenes, sensores de pulsación, así como la inclusión de actuadores que permitan controlar la iluminación, altavoces, motores, cerraduras electrónicas y pantallas.

Además, se llevará a cabo el desarrollo de un sistema de control remoto accesible a través de plataformas web y móviles, mediante una aplicación denominada “ConnectGuard Residence”, cuyo desarrollo de presentará más adelante, así como la programación del software necesario para el control del sistema. Como opción adicional, se considerará la implementación de un sistema de reconocimiento facial para controlar el acceso a la vivienda

En el contexto de la búsqueda de soluciones innovadoras que incorporen tecnologías avanzadas en la automatización de hogares, contribuyendo al avance de la domótica y la eficiencia energética en la vida cotidiana, este proyecto se compromete de manera activa con los Objetivos de Desarrollo Sostenible (ODS) establecidos por las Naciones Unidas. Al abordar la creación y desarrollo de un modelo de vivienda inteligente con sistemas de energía solar y componentes tecnológicos avanzados, se promueve la consecución de los ODS 7 (Energía asequible y no contaminante), 9 (Industria, innovación e infraestructura) y 11 (Ciudades y comunidades sostenibles). Este enfoque completo destaca el compromiso de este proyecto con la mejora de la calidad de vida y la sostenibilidad, en sintonía con los objetivos globales de desarrollo.

Esta metodología garantiza un enfoque sistemático y eficaz para alcanzar por completo los objetivos del proyecto.

- Metodología seguida para alcanzar estos objetivos:

Para cumplir con los objetivos, se seguirá un enfoque metodológico integral que abarca diversas etapas esenciales.

En primer lugar, se llevará a cabo una revisión y un análisis de las tecnologías existentes relacionadas con la automatización residencial, la energía solar y los sistemas de control remoto. Luego, se procederá a la etapa de diseño, en la cual se conceptualizará la maqueta de la vivienda inteligente, definiendo la disposición de sensores, actuadores y componentes de energía solar.

A continuación, se llevará a cabo la implementación física de la maqueta y la programación de los sistemas de control. La fase de pruebas y validación adquiere un papel crucial para asegurar el funcionamiento óptimo de todos los componentes.

Por último, se evaluará el rendimiento del sistema en términos de eficiencia energética y funcionalidad, y se registrarán los resultados obtenidos.

3 Estudio previo

El propósito de este estudio es seleccionar, de entre las opciones disponibles, los elementos ideales que se emplearán en el diseño del sistema de domótica, considerando los diversos aspectos de los Objetivos de Desarrollo Sostenible (ODS), con el objetivo de fomentar la eficiencia energética, la gestión sostenible de los recursos y la creación de un entorno habitable y respetuoso con el medio ambiente.

3.1 Componentes y funcionamiento del sistema domótico

Un sistema domótico consta de tres elementos principales: sensores, controladores y actuadores:

- Los **sensores** pueden detectar los cambios de presión, luz, temperatura o detección de movimiento. En otras palabras, cuando ocurren cambios o eventos físicos, se generan señales eléctricas que pueden ser procesadas por un controlador. Los sistemas domóticos pueden ajustar esos parámetros (y otros) a sus preferencias.
- Los **controladores** son los dispositivos (ordenadores personales, tabletas o teléfonos inteligentes) que se utilizan para enviar y recibir mensajes sobre el estado de las funciones automatizadas de la casa y ejecutan el programa de control del sistema.
- Los **actuadores** pueden ser luminarias, motores o válvulas motorizadas que realizan cambios sobre el mecanismo real, o la función, de un sistema domótico. Están programados para activarse mediante una orden remota de un controlador, permitiendo convertir señales eléctricas en acciones o variaciones de magnitudes físicas.

Para cumplir con los objetivos descritos en el apartado 2. Objetivos, se ha diseñado un sistema que proporciona las siguientes funcionalidades, cada una de las cuales requiere el uso de varios componentes, además de la creación de una aplicación web llamada "**ConnectGuard Residence**", que es el elemento crucial del sistema permitiendo su control remoto.

- Detección de movimiento en el exterior (sensor PIR), aviso a usuarios (Pushbullet)
- Grabación de imágenes de seguridad con detección (Cámara USB1) y envío de notificación a usuarios (Pushbullet).
- Transmisión en directo (streaming) de imágenes del exterior del prototipo (Cámara USB2)
- Apertura/Cierre de la cerradura electrónica de forma manual remota (ConnectGuard Residence).

- Apertura de cerradura electrónica tras usar el timbre (pulsador), obtener reconocimiento positivo de imagen de usuario captada por cámara exterior (PiCamera) basada en IA. Envío del resultado a usuarios (pushbullet).
- Apertura/Cierre de la puerta de la cochera (Servomotor) de forma manual remota (ConnectGurad Residence)
- Simulación de presencia en interior del prototipo para disuadir posibles intentos de robo consistente en el encendido/apagado de luces (LEDs) y sonido (Altavoz) que es configurable de forma manual y programada.
- Gestión de usuarios del sistema que incluye la incorporación de distintos tipos de usuarios al sistema con posibilidad de actualizar el modelo de aprendizaje automático que realiza el reconocimiento de usuarios, y la eliminación de usuarios (ConnectGuard Residence)
- Control remoto de todos los elementos y funcionalidades del sistema descritas a través de la aplicación web "ConnectGuard Residence" que permite configurar modo de funcionamiento.
- Registro de eventos sucedidos en el sistema (ConnectGuard Residence).
- Iluminación externa sostenible del prototipo (tira de luces LED), basada en una placa solar y una batería 5V.

A continuación, se presenta un esquema genérico del sistema domótico:

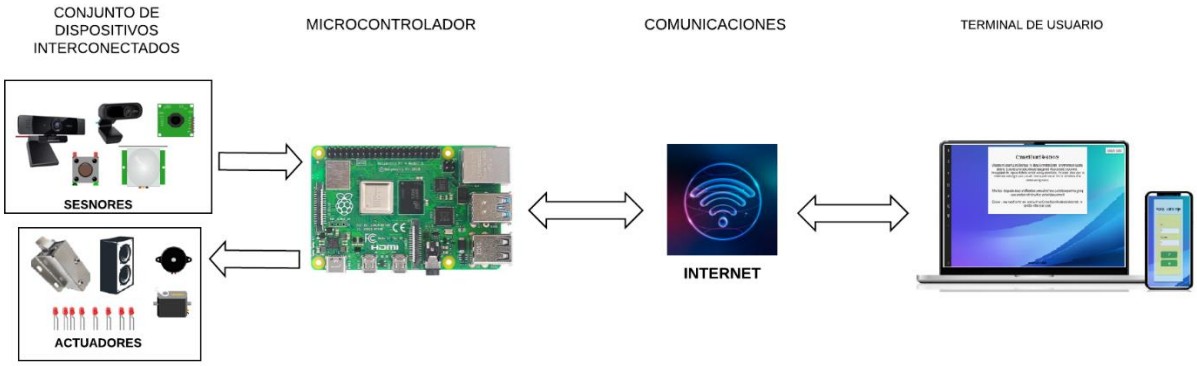


Figura 4 Esquema genérico del sistema domótico

En el siguiente apartado se realizará un análisis detallado y la selección de los componentes necesarios para el sistema. Se evaluarán las distintas opciones disponibles, considerando criterios como eficiencia, durabilidad y compatibilidad con el diseño general del prototipo.

Este estudio permitirá identificar los elementos más adecuados para cumplir con los objetivos establecidos y garantizar el rendimiento óptimo del sistema.

3.2 Microcontrolador

El núcleo de control de este sistema domótico será un Single Board Computer (SBC).

Un SBC es un ordenador completo en la que una placa de circuito único comprende la memoria, la entrada/salida, un microprocesador y todas las demás características necesarias.

Además, este tipo de componentes ofrecen notables ventajas en proyectos tecnológicos. Su tamaño compacto facilita su incorporación en espacios reducidos, mientras que su costo asequible los convierte en una opción viable incluso para presupuestos limitados. Además, su eficiencia en el consumo de energía los hace idóneos para aplicaciones que requieren autonomía o un funcionamiento continuo, lo que los hace una elección versátil y económica en diversas áreas de la tecnología y la electrónica.

A continuación de presentarán los SCB más relevantes que pueden estar empleados en el sistema domótico.

3.2.1 Arduino Uno R3

El Arduino Uno R3 es una placa de microcontrolador muy conocida que utiliza el microcontrolador ATmega328P como su núcleo. Forma parte de la familia Arduino, que consiste en plataformas de hardware y software de código abierto diseñadas para la creación de dispositivos digitales y objetos interactivos. [9]

La placa Arduino Uno R3 incluye varios componentes esenciales para su funcionamiento. Dichos componentes de muestran a continuación. [10]

- Microcontrolador ATmega328P: Este es el cerebro de la placa y es responsable de ejecutar el código que le cargues. Tiene memoria flash para almacenar programas y memoria SRAM para variables temporales.
- Cristal Oscilador: La placa Arduino Uno R3 generalmente utiliza un cristal de 16 MHz para proporcionar un reloj preciso al microcontrolador.
- Puertos Digitales y Analógicos: La placa Arduino Uno R3 tiene una serie de pines digitales y analógicos que se pueden utilizar para conectar sensores, actuadores y otros dispositivos.
- Puerto USB: Se utiliza para cargar programas en la placa y para la comunicación serie con otros dispositivos, como una computadora
- Conector de Alimentación: Puedes alimentar la placa con una fuente de alimentación externa o a través del puerto USB.
- LEDs de Estado: La placa incluye varios LEDs, como el LED de alimentación y LEDs en los pines 13 y 12 que se pueden utilizar para indicar el estado del programa.

- Botón de Reset: Puedes reiniciar la placa presionando este botón.
- Regulador de Voltaje: Regula el voltaje de entrada para proporcionar una alimentación estable al microcontrolador y otros componentes.
- Conectores de Entrada/Salida: Estos conectores permiten conectar sensores, actuadores y otros dispositivos externos a la placa.
- Conector ICSP (In-Circuit Serial Programming): Este conector se utiliza para programar el microcontrolador directamente utilizando un programador externo.
- Componentes Pasivos: La placa también incluye una serie de resistencias, condensadores y otros componentes pasivos que son necesarios para el funcionamiento correcto de la placa.

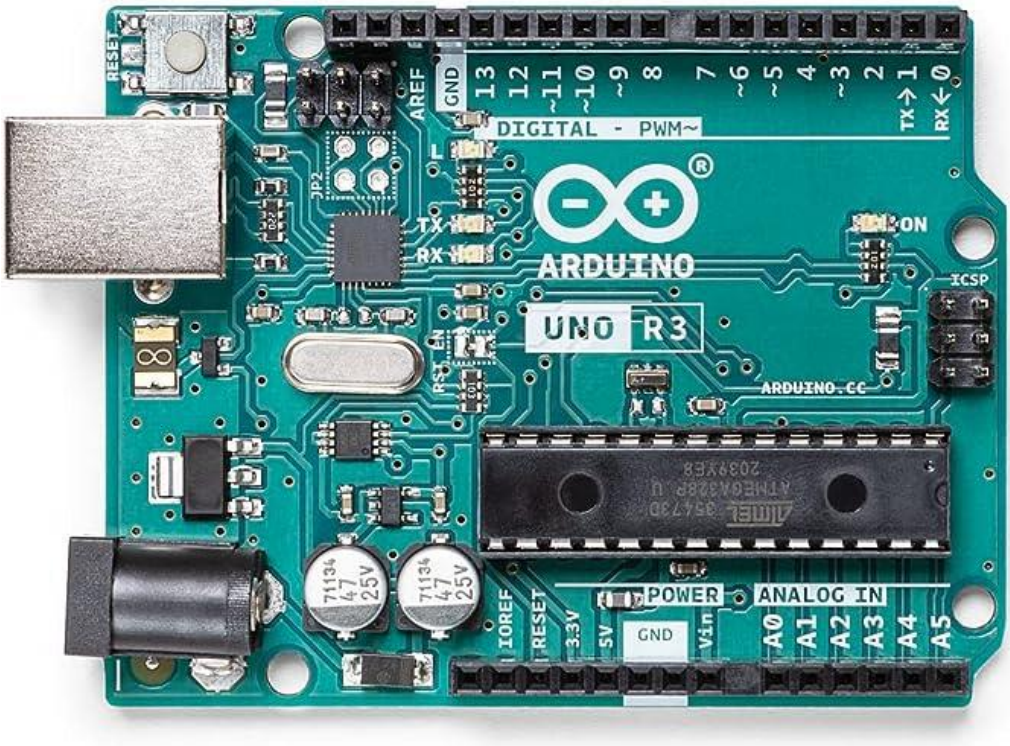


Figura 5: Placa Arduino Uno R3

3.2.2 Raspberry Pi 4B

La Raspberry Pi 4B es una potente y versátil placa de desarrollo que cuenta con una amplia gama de componentes esenciales para su correcto funcionamiento.[11]

Los componentes de este SBC son los siguientes:

- Procesador: La Raspberry Pi 4B utiliza un procesador Broadcom BCM2711 de cuatro núcleos Cortex-A72 a 1.5 GHz, lo que la hace mucho más potente que sus predecesoras.

- Memoria RAM: Se ofrece en opciones de 2 GB, 4 GB y 8 GB de RAM LPDDR4-3200, lo que permite un rendimiento más rápido y una mayor capacidad de ejecución de aplicaciones.
- Puertos USB: Dispone de dos puertos USB 3.0 y dos puertos USB 2.0 para la conexión de periféricos y dispositivos externos.
- Puertos micro HDMI: La Raspberry Pi 4B puede admitir dos monitores a través de dos puertos micro HDMI, lo que permite una configuración de pantalla dual.
- Puerto Ethernet: Ofrece un puerto Ethernet Gigabit para una conexión a Internet más rápida y estable.
- Conectividad inalámbrica: Incorpora Wi-Fi 802.11ac y Bluetooth 5.0 integrados para la conectividad inalámbrica.
- Puerto USB-C: Utiliza un puerto USB-C para la alimentación, proporcionando una fuente de alimentación más estable.
- Pines GPIO: Mantiene la compatibilidad con los pines GPIO de las versiones anteriores, lo que permite la conexión de sensores y otros dispositivos.
- Ranura para tarjeta microSD: Utiliza una tarjeta microSD como almacenamiento principal para el sistema operativo y datos.
- Puerto de cámara y pantalla: Dispone de puertos específicos para conectar cámaras y pantallas a través de cables flexibles.
- Puerto de audio: Incluye una toma de audio de 3.5 mm para la salida de audio.
- Puerto de alimentación: Conexión para la fuente de alimentación micro USB-C.
- LEDs indicadores: La placa incluye LEDs indicadores de estado, como el LED de encendido y el LED de actividad de la tarjeta microSD.
- Sistema operativo: Puede ejecutar una variedad de sistemas operativos, incluyendo Raspberry Pi OS (anteriormente Raspbian), Linux y otros sistemas basados en ARM.

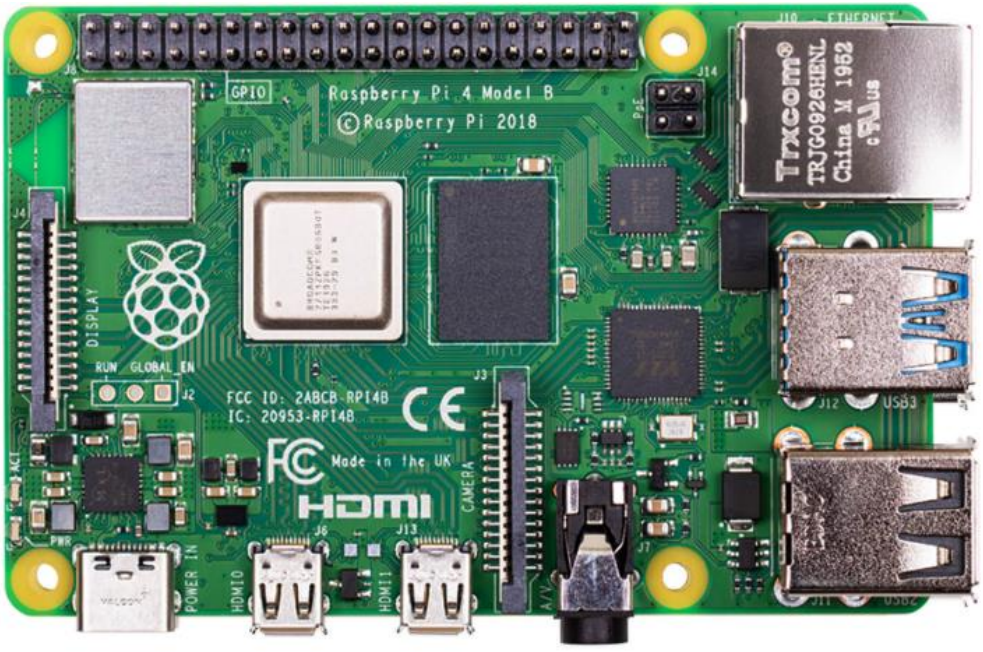


Figura 6: Placa Raspberry Pi 4B

3.2.3 ODROID N2

El ODROID-N2 es un SBC creado por Hardkernel, que es una compañía surcoreana de electrónica de consumo. Fue lanzado como una mejora con respecto al ODROID-N1 y está diseñado para brindar un rendimiento superior en comparación con otras placas de desarrollo similares. [12]

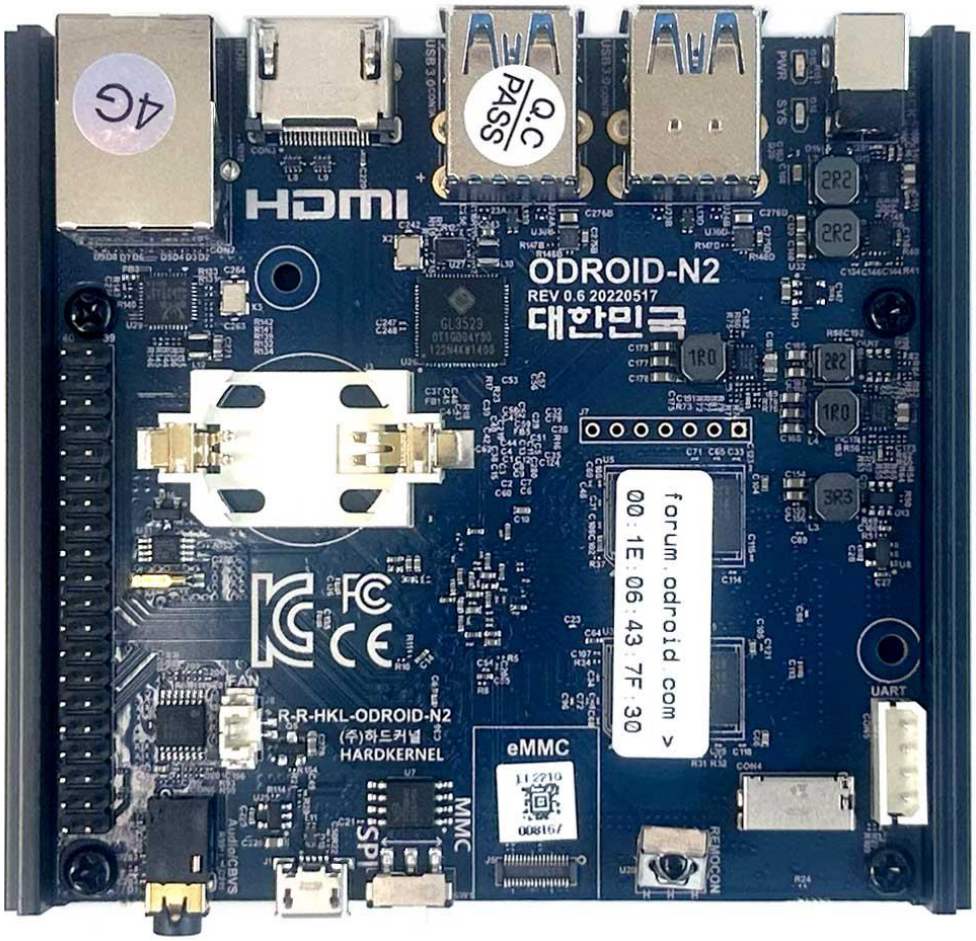


Figura 7: Ordoid N2

Los componentes de la placa Odroid-N2 se presentan a continuación:

1. Procesador:
    - Procesador principal: ARM Cortex-A73 de cuatro núcleos a 1.8 GHz.
    - Procesador secundario: ARM Cortex-A53 de cuatro núcleos a 1.9 GHz.
2. GPU (Unidad de Procesamiento Gráfico):
    - Mali-G52 GPU, que es capaz de manejar gráficos 3D y aplicaciones multimedia.
3. Memoria RAM:

- Ofrece dos opciones de memoria RAM: 2 GB o 4 GB de LPDDR4.

4. Almacenamiento:
    - Ranura para tarjeta microSD: Permite la inserción de una tarjeta microSD para almacenamiento adicional.
    - Ranura para eMMC: También admite almacenamiento eMMC para un rendimiento de almacenamiento más rápido y confiable.
5. Conectividad:
    - Ethernet Gigabit: Un puerto Ethernet Gigabit para la conexión a redes con cable.
    - USB 3.0: Dos puertos USB 3.0 para conectar periféricos de alta velocidad.
    - USB 2.0: Cuatro puertos USB 2.0 adicionales para periféricos más antiguos.
    - HDMI 2.0: Un puerto HDMI 2.0 para salida de video de alta definición.
    - Salida de audio: Un puerto de audio de 3.5 mm para conectar altavoces o auriculares.
6. Puertos de Expansión:
    - GPIO (Entrada/Salida de Propósito General): Varios pines GPIO para la conexión de sensores y otros dispositivos electrónicos personalizados.
7. Conector de Alimentación:
    - Utiliza un conector de alimentación DC de 5V/4A para recibir energía eléctrica.
8. Botones y LEDs:
    - Botón de encendido: Para encender y apagar la placa.
    - Botón de reinicio: Para reiniciar la placa.
    - LEDs indicadores: Pueden incluir LEDs que indican el estado de la alimentación y la actividad del sistema.
9. Sistema Operativo:
    - Compatible con varios sistemas operativos, incluyendo Ubuntu, Android y otros sistemas basados en Linux, que se pueden instalar en la tarjeta de almacenamiento.

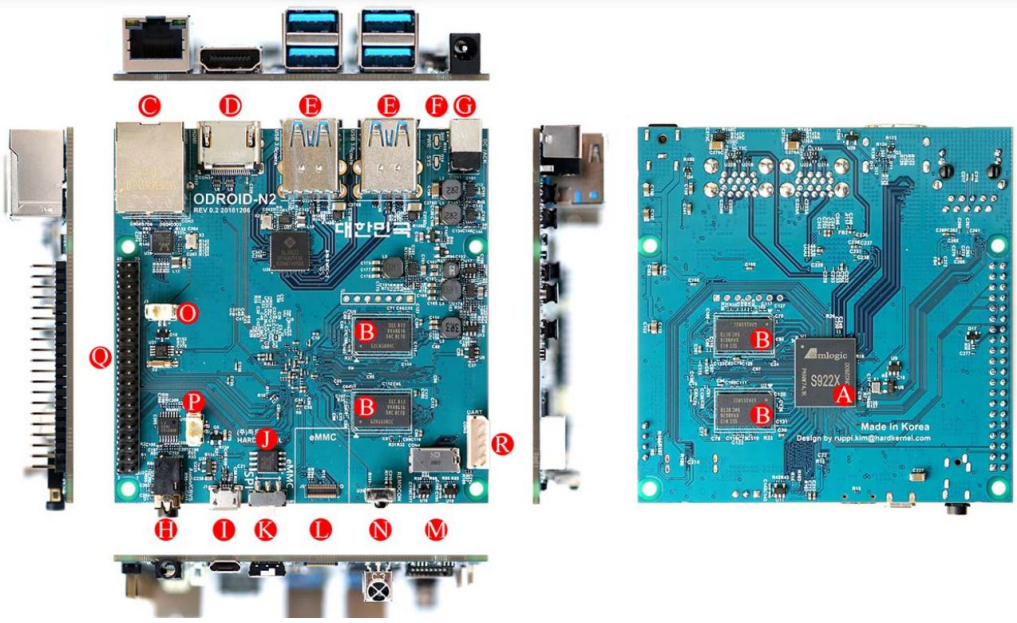


Figura 8: Componentes de placa Ordoid N2

Number	Item	Number	Item
A	S922X CPU	J	1 x SPI Flash 8MiB
B	4 x DDR4 RAM	K	1 x SPI Boot Select Switch
C	1 x RJ45 Ethernet Port (10/100/1000)	L	1 x eMMC Module Socket
D	1 x HDMI 2.0	M	1 x Micro SD Slot
E	4 x USB 3.0	N	1 x IR Receiver
F	2 x System LED Indicators	O	1 x RTC Backup Battery Connector (2-pin)
G	1 x DC Power Jack	P	1 x Active Cooling Fan Connector (2-pin)
H	1 x AV Out (Stereo Audio with Composite video)	Q	40 x GPIO Pins
I	1 x Micro USB2.0 OTG	R	1 x UART for System Console

Figura 9: Descripción de componentes placa Ordoid N2

Fuente: https://www.hardkernel.com/shop/odroid-n2-with-4gbyte-ram/

3.2.4 Elección:

A partir de las explicaciones desarrolladas anteriormente, podemos concluir que el Arduino Uno R3 se enfrenta a limitaciones notables debido a su microcontrolador de 8 bits, lo que restringe su capacidad de procesamiento y le impide ejecutar un sistema operativo completo. Esto lo hace menos adecuado para tareas que requieren una potencia de procesamiento más robusta, como el manejo de múltiples cámaras de seguridad y la transmisión en vivo.

Por otro lado, el Odroid N2, aunque más potente, puede considerarse una opción sobresaliente y costosa si el sistema de casa inteligente no requiere un procesamiento excepcional. Este dispositivo brilla en aplicaciones de alto rendimiento, como servidores, y puede resultar excesivo para un sistema de este tipo.

En contraste, la Raspberry Pi 4 emerge como la elección más idónea en este escenario. Su CPU quad-core y opciones de memoria RAM proporcionan la potencia de procesamiento necesaria para gestionar múltiples cámaras y aplicaciones de video sin problemas. Además, permite la instalación de un sistema operativo completo, como Raspberry Pi OS o una distribución de Linux, lo que simplifica la gestión de cámaras, transmisión en vivo y automatización del hogar. La Raspberry Pi 4 también ofrece una variedad de puertos USB para conectar dispositivos, así como opciones de conectividad como Ethernet Gigabit y Wi-Fi, lo que resulta fundamental para el control de elementos del hogar y la realización de transmisiones en vivo de manera eficaz.

Según los requisitos del proyecto y considerando que el sistema estará compuesto por 2 cámaras conectadas a través de puertos USB, se ha tomado la decisión de utilizar el microcontrolador Raspberry Pi.

3.3 Comunicaciones

Para alcanzar el objetivo del sistema domótico, hay que conectar el microcontrolador, que es la Raspberry Pi, con los distintos sensores y actuadores instalados en la maqueta. Dicha conexión será directa y se efectúa mediante cables eléctricos.

En cuanto al control remoto del sistema domótico, es fundamental contar con un protocolo de transmisión de datos sólido y seguro para garantizar una comunicación efectiva entre la Raspberry Pi, que actúa como el cerebro central del sistema, y la interfaz web a través de la cual los usuarios interactúan con él.

A continuación, se presentarán algunos de los posibles protocolos que pueden ser considerados para esta tarea crucial.

3.3.1 MQTT

MQTT, cuyas siglas significan Message Queuing Telemetry Transport, es un protocolo de comunicación eficiente y liviano diseñado para transmitir mensajes entre dispositivos en una red, generalmente en el ámbito de la Internet de las cosas (IoT). IBM lo desarrolló en la década de 1990 y se ha convertido en un estándar muy utilizado para la comunicación entre dispositivos IoT, aplicaciones móviles y sistemas embebidos.[13], [14]

A continuación, se describen las características principales del protocolo MQTT.

- Eficiencia: MQTT es altamente eficiente en términos de ancho de banda y consumo de energía, lo que lo hace ideal para dispositivos IoT con recursos limitados.

- Comunicación en tiempo real: MQTT es especialmente adecuado para aplicaciones que requieren comunicación en tiempo real, ya que permite la publicación y suscripción de datos de manera instantánea.
- Seguridad: Requiere configuración adicional para habilitar la seguridad (TLS/SSL) y la autenticación, pero es altamente personalizable.
- Escalabilidad: MQTT es escalable y puede manejar una gran cantidad de dispositivos conectados.
- Uso común: Ampliamente utilizado en aplicaciones IoT y en sistemas de mensajería donde se necesita comunicación en tiempo real.

3.3.2 CoAP (Constrained Application Protocol):

CoAP (Constrained Application Protocol) es un protocolo de comunicación diseñado para dispositivos con recursos limitados, como sensores y dispositivos IoT [15].

CoAP es está optimizado para funcionar en redes con restricciones de ancho de banda y energía. Permite la transferencia eficiente de datos y comandos entre dispositivos, utilizando un enfoque de solicitud y respuesta.

Además, CoAP está diseñado para ser ligero y seguro, lo que lo hace adecuado para aplicaciones en entornos donde se requiere una comunicación eficiente y segura entre dispositivos con recursos limitados.

- Eficiencia: CoAP está diseñado específicamente para dispositivos con recursos limitados, lo que lo hace altamente eficiente en términos de ancho de banda y energía.
- Comunicación en tiempo real: Ofrece comunicación en tiempo real adecuada para aplicaciones IoT.
- Seguridad: Requiere extensiones adicionales para habilitar la seguridad, lo que puede hacer que su configuración sea más compleja.
- Escalabilidad: CoAP es escalable y adecuado para aplicaciones con una gran cantidad de dispositivos IoT.
- Uso común: Principalmente utilizado en aplicaciones IoT y sistemas donde los recursos son escasos.

3.3.3 HTTP

HTTP (Hypertext Transfer Protocol) es un protocolo de comunicación empleado en la World Wide Web (WWW) con el propósito de transferir datos, incluyendo documentos de texto, imágenes y otros recursos multimedia, entre un cliente (usualmente un navegador web) y un servidor web [16].

Se fundamenta en el modelo Cliente-Servidor y sigue la estructura de Petición – Respuesta, donde el cliente efectúa solicitudes para acceder a recursos en el servidor, y el servidor responde suministrando dichos recursos.

Las características destacadas de este protocolo son las siguientes:

- Universalidad: HTTP es el protocolo fundamental de la World Wide Web y es ampliamente compatible con todos los navegadores y plataformas.
- Seguridad: Puede ser altamente seguro cuando se utiliza HTTPS, que ofrece encriptación de datos y autenticación.
- Comunicación orientada a recursos: HTTP se basa en solicitudes y respuestas para acceder a recursos identificados por URLs, lo que es adecuado para aplicaciones web estándar.
- Latencia: La comunicación a través de HTTP puede tener latencias más altas en comparación con MQTT y CoAP debido a la naturaleza de solicitud-respuesta.
- Uso común: Ampliamente utilizado en aplicaciones web y en la mayoría de las interacciones en línea.

3.3.4 Elección

Se ha tomado la decisión de emplear HTTP para facilitar la interacción y transmisión de datos en el sistema. Esta elección se fundamenta en la amplia compatibilidad y simplicidad de HTTP, lo que permitirá una comunicación eficiente y efectiva entre los componentes del proyecto.

Además, mediante este protocolo, existe la posibilidad de garantizar la seguridad mediante la implementación de HTTPS para proteger la comunicación entre la Raspberry Pi y la interfaz web.

3.4 Software

En este apartado, se proporcionará una visión general de los lenguajes y métodos utilizados en el desarrollo de este proyecto y de la creación de la página web. A lo largo de los siguientes apartados, se detallarán en profundidad cada uno de los métodos empleados, proporcionando una comprensión exhaustiva de su aplicación en el contexto del proyecto

3.4.1 Python

Python es un lenguaje de programación de alto nivel, interpretado y de propósito general, conocido por su sintaxis clara y legible. Es multiparadigma, soportando programación orientada a objetos, funcional e imperativa. Es ampliamente utilizado en desarrollo web, ciencia de datos, inteligencia artificial, automatización y más, gracias a su extensa biblioteca estándar y marcos populares como Django, Flask, NumPy y

TensorFlow. Su portabilidad y gran comunidad de usuarios lo han convertido en uno de los lenguajes más populares en la industria tecnológica.[17]

En el desarrollo de la página web para el control remoto de la maqueta de la casa domótica, Python se ha utilizado para gestionar la lógica del lado del servidor y las operaciones detrás de escena. Python ha desempeñado un papel crucial en la comunicación con dispositivos domóticos, procesamiento de datos y cálculos específicos necesarios para controlar y monitorear los dispositivos en tiempo real. Además, Python se ha utilizado para mantener la coherencia de los datos y garantizar la seguridad en la transferencia de información.

La elección de Python se justifica debido a su versatilidad y adecuación para proyectos de desarrollo web. Python es conocido por su facilidad de lectura y escritura, lo que acelera el desarrollo y facilita la colaboración entre miembros del equipo. Su amplia gama de bibliotecas y marcos de trabajo, como Flask, ha permitido una implementación efectiva de la lógica del lado del servidor.

Python también se destaca en la manipulación de datos, lo que es esencial para el control de dispositivos domóticos y la presentación de información en tiempo real. La elección de Python en el proyecto proporciona una sólida base para la comunicación con dispositivos, la gestión de datos y el procesamiento necesario para el control de una casa domótica de manera eficiente y segura.

Es esencial destacar el papel crucial desempeñado por varias bibliotecas y módulos que han enriquecido la funcionalidad y eficiencia de nuestro proyecto. A continuación, detallaremos las bibliotecas específicas que hemos empleado para diversas tareas, desde el manejo de la lógica del lado del servidor hasta la comunicación con dispositivos domóticos, el almacenamiento de datos y la mejora de la experiencia del usuario.

Las librerías usadas en Python se describen a continuación:

- Flask: Flask es un micro marco de desarrollo web en Python que has utilizado para construir la parte del servidor de la aplicación web de control remoto de una casa domótica. Es conocido por su simplicidad y flexibilidad, lo que lo hace ideal para aplicaciones web pequeñas y medianas. Se ha aprovechado Flask para definir rutas, vistas y modelos de datos que gestionan la comunicación con los clientes y la lógica de la aplicación. Además, se han utilizado Blueprints para organizar la estructura de la aplicación, permitiendo una separación modular del código y facilitando el mantenimiento y la escalabilidad.
  Flask permite una implementación ágil y eficiente del lado del servidor y proporciona un entorno sólido para construir una API web para controlar dispositivos domóticos.[18]

- OpenCV (Open Source Computer Vision Library): OpenCV es una biblioteca de código abierto ampliamente utilizada para el procesamiento de imágenes y la visión por computadora[19]. Se ha empleado OpenCV para tareas relacionadas

con la visión, como el análisis de imágenes de cámaras de seguridad y el reconocimiento facial. Esta biblioteca se utiliza para enriquecer la funcionalidad del proyecto al permitir el procesamiento de imágenes para la aplicación domótica, lo que contribuye a la seguridad y la toma de decisiones basadas en datos visuales.

- PyMongo: PyMongo es una biblioteca de Python que se utiliza para interactuar con bases de datos MongoDB, que es una base de datos NoSQL ampliamente utilizada[20]. Se ha empleado PyMongo para almacenar y recuperar datos relacionados con la configuración de los usuarios además de los registros de evento. La elección de MongoDB y PyMongo se justifica por su flexibilidad y escalabilidad, lo que permite manejar de manera eficiente una gran cantidad de datos en una aplicación de control domótico.

- Pushbullet: es una biblioteca y un servicio que permite la sincronización de notificaciones y la transferencia de archivos entre dispositivos. Mediante su API y bibliotecas, se puede integrar en las aplicaciones para enviar notificaciones en tiempo real a dispositivos móviles y ordenadores [21]. Se ha utilizado para habilitar notificaciones y alertas en tiempo real, cuando ocurra un evento importante, como una alarma de seguridad o un cambio en el estado de un dispositivo domótico. Esta biblioteca mejora la usabilidad y la capacidad de respuesta de la aplicación, manteniendo a los usuarios informados sobre lo que sucede en su casa en todo momento.

- RPi.GPIO: es una biblioteca de Python utilizada para interactuar con los pines GPIO de una Raspberry Pi[22]. Se ha utilizado RPi.GPIO en el proyecto de control domótico para gestionar la comunicación con dispositivos de hardware como sensores y actuadores. Esta biblioteca permite configurar y controlar los pines GPIO, permitiendo la lectura de entradas digitales (como sensores de movimiento) y la escritura de salidas digitales (como la activación de relés o LEDs). RPi.GPIO facilita la integración de hardware con software en aplicaciones domóticas, proporcionando una interfaz sencilla y eficiente para el control de dispositivos físicos.

- Threading: Threading es un módulo de Python que permite la creación y gestión de hilos (threads) en una aplicación[23]. En el contexto de la aplicación domótica, se ha utilizado Threading para ejecutar tareas concurrentes, como la monitorización de sensores, el procesamiento de datos y la gestión de la comunicación con dispositivos en tiempo real. El uso de hilos permite mejorar la eficiencia y la capacidad de respuesta de la aplicación al ejecutar múltiples operaciones simultáneamente, evitando bloqueos y asegurando un rendimiento fluido.

- Subprocess: Subprocess es un módulo de Python que permite la ejecución de comandos del sistema y la creación de nuevos procesos[24]. En el proyecto de

control domótico, Subprocess se ha empleado para integrar y ejecutar herramientas externas o scripts adicionales que no están directamente accesibles desde el código Python, como es el caso del código que lanza la simulación de presencia. Esta biblioteca proporciona una forma flexible y poderosa de extender la funcionalidad de la aplicación mediante la integración de recursos externos.

En conjunto, estas bibliotecas y herramientas han sido fundamentales para el desarrollo y la funcionalidad del proyecto de control remoto de casa domótica, permitiendo la comunicación eficiente con dispositivos, el almacenamiento de datos, notificaciones en tiempo real y el procesamiento de imágenes para una experiencia de usuario completa y segura.

3.4.2 HTML & CSS

HTML (HyperText Markup Language) es el lenguaje estándar para crear y estructurar páginas web. Utiliza una serie de elementos y etiquetas para definir la estructura y el contenido de una página, como encabezados, párrafos, enlaces, imágenes y formularios. HTML es fundamental para la web, ya que proporciona la base sobre la cual se construyen y renderizan los documentos web en los navegadores.[25]

CSS (Cascading Style Sheets) es el lenguaje utilizado para describir la presentación y el diseño de una página web escrita en HTML. CSS controla la apariencia de los elementos HTML, incluyendo colores, fuentes, márgenes, alineación, y disposición en pantalla. Permite separar la estructura del contenido de su presentación, facilitando el mantenimiento y la actualización del diseño de los sitios web. CSS es esencial para crear páginas web visualmente atractivas y coherentes.[26]

La elección de HTML y CSS se justifica por su versatilidad, amplia adopción y sólido soporte en navegadores web modernos. Estos lenguajes estándar de la web ofrecen una base sólida para el desarrollo de una interfaz de usuario amigable e intuitiva. Además, la separación de la estructura (HTML) y el estilo (CSS) permite una mayor flexibilidad en el diseño y la personalización de la interfaz. Comparados con otras tecnologías de desarrollo de aplicaciones, HTML y CSS son altamente compatibles y accesibles, lo que garantiza una experiencia uniforme para los usuarios, independientemente del dispositivo o plataforma que utilicen.

- HTML: En el desarrollo de la página web para el control remoto de una casa domótica, HTML se ha utilizado para definir la estructura y disposición de los elementos clave de la interfaz de usuario. Se ha creado un menú de navegación que permite a los usuarios acceder a las diferentes áreas de control de dispositivos domóticos, como iluminación, climatización y seguridad. Los botones de control se han implementado como elementos HTML que representan acciones específicas, como encender o apagar luces, ajustar la

temperatura o activar alarmas de seguridad. Los formularios HTML se han utilizado para recopilar información del usuario, como contraseñas o configuraciones personalizadas.

- CSS: Ha sido esencial para la personalización y el diseño visual de la interfaz de usuario. Se han aplicado reglas de estilo para dar formato a los elementos HTML, incluyendo el menú de navegación, los botones y los formularios. Esto ha incluido la definición de colores, fuentes, márgenes, alineación y efectos visuales, con el propósito de crear una experiencia de usuario atractiva y coherente. El menú de navegación ha sido estilizado para ser fácilmente accesible y comprensible, los botones de control han sido diseñados de manera intuitiva y los formularios se han presentado de forma clara y organizada. Ejemplos de código CSS específico se han proporcionado en la memoria para ilustrar cómo se han logrado estos efectos visuales y estilos.

En resumen, HTML y CSS se han empleado para estructurar y diseñar la interfaz de usuario de la página web de control domótico, permitiendo una navegación sencilla, la interacción efectiva con dispositivos domóticos y una experiencia visualmente atractiva para los usuarios.

3.4.3 JavaScript (jquery)

JavaScript se ha utilizado en el desarrollo de la página web para el control remoto de la casa domótica para proporcionar interactividad y funcionalidad en el lado del cliente. Se ha aplicado para gestionar la lógica de la interfaz de usuario, la comunicación con dispositivos domóticos y la actualización en tiempo real de datos. Con JavaScript, se han logrado funciones específicas como la detección de eventos del usuario, la validación de formularios y la manipulación de datos en tiempo real enviando notificaciones al usuario.

La elección de JavaScript se justifica debido a su papel esencial en el desarrollo web. JavaScript es el lenguaje de programación de facto en la programación del lado del cliente y es compatible con la mayoría de los navegadores web modernos. Su versatilidad y amplia adopción lo convierten en una opción sólida para agregar interactividad a la interfaz de usuario y para facilitar la comunicación en tiempo real con dispositivos domóticos. La capacidad de JavaScript para interactuar con el DOM (Document Object Model) de manera eficiente permite la creación de interfaces de usuario dinámicas y responsivas. Además, su amplia comunidad de desarrolladores y el acceso a numerosas bibliotecas y frameworks hacen que JavaScript sea una elección natural para proyectos web complejos como el control de dispositivos domóticos.[27]

Junto con JavaScript, se ha elegido la librería jQuery por su habilidad de simplificar el desarrollo de aplicaciones web interactivas.

jQuery es una biblioteca de JavaScript ampliamente utilizada que ofrece una amplia gama de funciones y complementos que aceleran la programación del lado del cliente. Su sintaxis simplificada y su capacidad para abstraer la complejidad de JavaScript nativo facilitan la creación de interacciones ricas y efectivas en la interfaz de usuario. Además, es compatible con la mayoría de los navegadores web modernos, lo que garantiza una experiencia uniforme para los usuarios en diferentes plataformas[28].

Se ha utilizado para simplificar la manipulación del Document Object Model (DOM), lo que ha permitido una gestión más eficiente de eventos, animaciones y la actualización en tiempo real de dispositivos domóticos. Con jQuery, se han logrado efectos visuales suaves, respuestas interactivas instantáneas y la ejecución de acciones específicas, como el encendido y apagado de dispositivos domóticos con un solo clic.

3.4.4 PushBullet

Para lograr que la Raspberry Pi notifica al usuario de todos los eventos y las alertas de seguridad, se ha usado la aplicación PushBullet.

La aplicación Pushbullet es una solución versátil que ha sido integrada en nuestro proyecto de control domótico basado en Python. Pushbullet es conocida por su capacidad para habilitar la comunicación y notificaciones instantáneas entre dispositivos, lo que mejora significativamente la experiencia de los usuarios en nuestra aplicación domótica.[29]

Pushbullet permite a los usuarios recibir notificaciones en tiempo real en sus dispositivos móviles, tabletas y computadoras. Estas notificaciones pueden variar desde alertas de seguridad, como intrusiones no autorizadas, hasta actualizaciones de estado de los dispositivos domóticos, como el encendido o apagado de luces y otros eventos relevantes.

Además de las notificaciones la aplicación Pushbullet también se integra con otros servicios y aplicaciones, lo que amplía su funcionalidad.

En resumen, esta herramienta desempeña un papel fundamental en el proyecto de control domótico, al proporcionar notificaciones en tiempo real, la capacidad de tomar medidas directas desde dispositivos y facilitar la interacción de usuarios con el sistema, mejorando así la comodidad y la seguridad en general.

3.5 Hardware

A continuación de detallarán los componentes Hardware que formarán parte del sistema.

3.5.1 Sensores de presencia

Los sensores de presencia, también conocidos como sensores de movimiento, son dispositivos que detectan la presencia de personas u objetos en un área determinada y se utilizan comúnmente para la automatización de sistemas de iluminación, sistemas de seguridad y otros sistemas de control. Los tipos más relevantes de sensores de presencia incluyen: Sensores PIR (Infrarrojos Pasivos) y Sensores Ultrasónicos.

- **Sensores Ultrasónicos**: admiten la **detección de objetos sin contacto** y la medición de la distancia de los objetos desde el sensor.
  Emiten ondas ultrasónicas de alta frecuencia (inaudible para los humanos, generalmente en el rango de 20 kHz a 65 kHz) y miden el tiempo que tarda en rebotar en objetos y regresar al sensor.
  Los impulsos se propagan en forma de ondas por el aire y se reflejan tan pronto tocan la superficie.
  Estos sensores son efectivos en la detección de objetos en un rango de distancia predefinido, pero pueden ser afectados por condiciones climáticas adversas, obstáculos que absorben el sonido o superficies absorbentes, por lo tanto, son ideales para áreas donde los sensores PIR no pueden funcionar bien debido a factores ambientales. Son ampliamente utilizados en sistemas de detección de obstáculos para robots, sistemas de estacionamiento de vehículos, sistemas de alarma y otros dispositivos de automatización que requieren medición de distancia[30].

Figura 10: Sensor ultrasónico Fuente: https://www.shellyespana.com/sensor-ultrasonico-hc-sr04/

- **Sensores PIR** (Infrarrojos Pasivos): Un sensor PIR o piroeléctrico funciona comparando la temperatura que desprende un objeto en su campo de visión, con la de su alrededor, de forma que puede detectar con precisión una presencia en un lugar determinado. Estos sensores contienen una lente que divide el área en múltiples segmentos. Cuando un objeto en movimiento atraviesa estos segmentos, la temperatura en esos puntos cambia, generando un patrón de calor en constante cambio.

  El sensor PIR contiene dos detectores que registran estos cambios térmicos. Si los detectores registran un cambio significativo en un corto período, el sensor activa una señal de salida, indicando la detección de movimiento. Son efectivos para detectar la presencia humana y son ampliamente utilizados en sistemas de iluminación automática y seguridad, ya que responden a los cambios de temperatura causados por el movimiento [31].

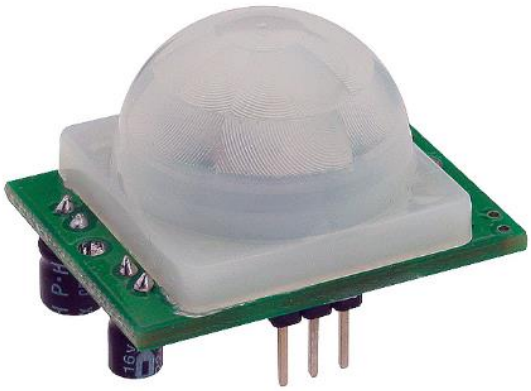


Figura 11 Sensor PIR Fuente: https://diotlabs.daraghbyrne.me/docs/working-with-data/communicating-events/pir

Se ha usado un sensor ultrasónico para el sistema de grabaciones de seguridad, el cual está ubicado en la zona de la entrada principal de la maqueta.

3.5.2 Sensores de captación de imágenes (Cámaras)

El modelo de microcontrolador elegido determina las posibilidades de conexión directa de sensores de captación de imágenes. En el modelo 4B de Raspberry Pi utilizado en este proyecto, existen dos opciones: la primera es mediante el uso de un módulo de cámara Raspberry Pi, y la segunda es a través de los conectores USB. Se han utilizado ambas opciones.

El diseño del sistema requiere tres cámaras: una para el sistema de reconocimiento facial, otra para la transmisión en directo, y una tercera para la grabación

de seguridad. Por estos motivos, se han empleado dos cámaras USB y una cámara Raspberry Pi.

3.5.3 Sensores de Pulsación

Este tipo de sensores de usará para iniciar una acción o función de un dispositivo electrónico: es el pulsador que estará localizado en la puerta principal de la vivienda, y que los visitantes pulsarán para anunciar su presencia.

3.5.4 Actuador: Luz

En el presente sistema domótico, se instalarán luces (LED O BOMBILLAS PEQUEÑAS) en las varias partes de la vivienda, que se podrán encender o apagar remotamente según los comandos del usuario. Es posible también automatizar su enciende/apago en varios intervalos de tiempo para efectuar la simulación de presencia antirrobo: ésta consiste en hacer que desde el exterior parezca que la vivienda está habitada cuando no se está en casa

3.5.5 Actuador: Altavoz

El altavoz se utilizará también para la función de simulación de presencia: Esto se consigue mediante la automatización y control remoto de elementos de una casa domótica, como son las persianas, las luces o la música. Así, se disuade de intentos de ocupación y robo. Adicionalmente se podrá usar para emitir sonidos de alarma ante una detección de intrusión.

3.5.6 Actuador: Cerradura

Una posible solución para controlar la apertura y el cierre de la puerta principal de la vivienda es la cerradura solenoide simple: consiste en un dispositivo electromagnético que se utiliza en situaciones donde se requiere un nivel básico de seguridad y control de acceso [32].

Una cerradura solenoide funciona al aplicar corriente eléctrica a una bobina, y crea un campo magnético: Cuando se energiza la bobina, el centro de metal se mueve en la bobina. Esto hace que el solenoide sea capaz de tirar de un extremo y por la tanto la apertura de la cerradura. Cuando se interrumpe la corriente, un resorte devuelve el perno a la posición bloqueada, asegurando la cerradura.

3.5.7 Actuador: Servo

En este sistema domótico y para poder controlar la apertura y el cierre de la puerta de la cochera, se ha optado a utilizar un servo SG90: es un tipo de motor con un potenciómetro, pequeño y económico, utilizado de manera común en proyectos de robótica.

Gracias a su rango de giro de hasta 180 grados puede controlar la posición de un objeto: esto de logra cuando el potenciómetro recibe una señal de control a través de PWM (Pulse Width Modulation) y entonces gira el motor para alcanzar la posición deseada. La alimentación se efectúa típicamente en el rango de 4.8 a 6 voltios.

3.5.8 Pantalla de control local

Finalmente, el último elemento hardware utilizado en este proyecto es una pantalla táctil HMI de 7 pulgadas. Actúa como sensor y actuador permitiendo interactuar con el sistema, mostrar su estado, controlarlo y configurarlo a través de la interfaz de una aplicación web. Adicionalmente, sirve de apoyo gráfico para el sistema de reconocimiento facial.

4 Materiales y Métodos: Desarrollo del sistema

En esta sección, se detallarán los materiales utilizados y los métodos empleados para el funcionamiento del sistema de la maqueta, proporcionando una comprensión completa de los componentes y procesos involucrados en la implementación del proyecto.

4.1 Materiales

Se han clasificado los elementos en los que se basa IoT en tres categorías: dispositivos finales (sensores o actuadores), procesadores y protocolos de comunicación.

Con esta clasificación en mente, la agrupación de los elementos usados en este TFG es la siguiente:

- Dispositivos finales:
    - Sensores de presencia infrarrojos pasivos para captar movimiento en la puerta principal de la vivienda.

- Un pulsador que actuará de timbre.
- Una cerradura de solenoide controlable, tanto manualmente por el usuario, como automáticamente por la IA de reconocimiento facial, y servirá para poder abrir la puerta principal de la vivienda.
- Módulo de cámara de Raspberry Pi NoIR v1.3, para realizar reconocimiento facial cuando se accione el pulsador, y que servirá para capturar los movimientos sospechosos y notificarlos al usuario.
- Cámara USB para realizar la transmisión en directo desde la vivienda
- Cámara USB para realizar las grabaciones de seguridad.
- Luces led para la iluminación de la vivienda.
- Un altavoz que actuará en tándem con la luz del interior de la casa para la simulación de presencia.
- Una batería para la iluminación nocturna del exterior de la vivienda
- Una placa solar para cargar la batería y detectar la luz ambiental.
- Un interruptor para controlar la iluminación externa de la vivienda.

● Procesadores:

- Raspberry Pi Modelo 4B. La programación del software de control que se ejecuta en este dispositivo se ha realizado utilizando el lenguaje de programación Python y las diversas librerías especificadas en el apartado 3.5.1.

● Comunicaciones:

- El microcontrolador Raspberry Pi centraliza todas las comunicaciones del sistema y se conecta a Internet a través de WiFi usando el protocolo HTTP, a través del cual el sistema de seguridad se conecta a la aplicación web, desde la cual el usuario puede controlarlo.

A continuación, se presenta un esquema general de los componentes electrónicos que conforman el sistema domótico:

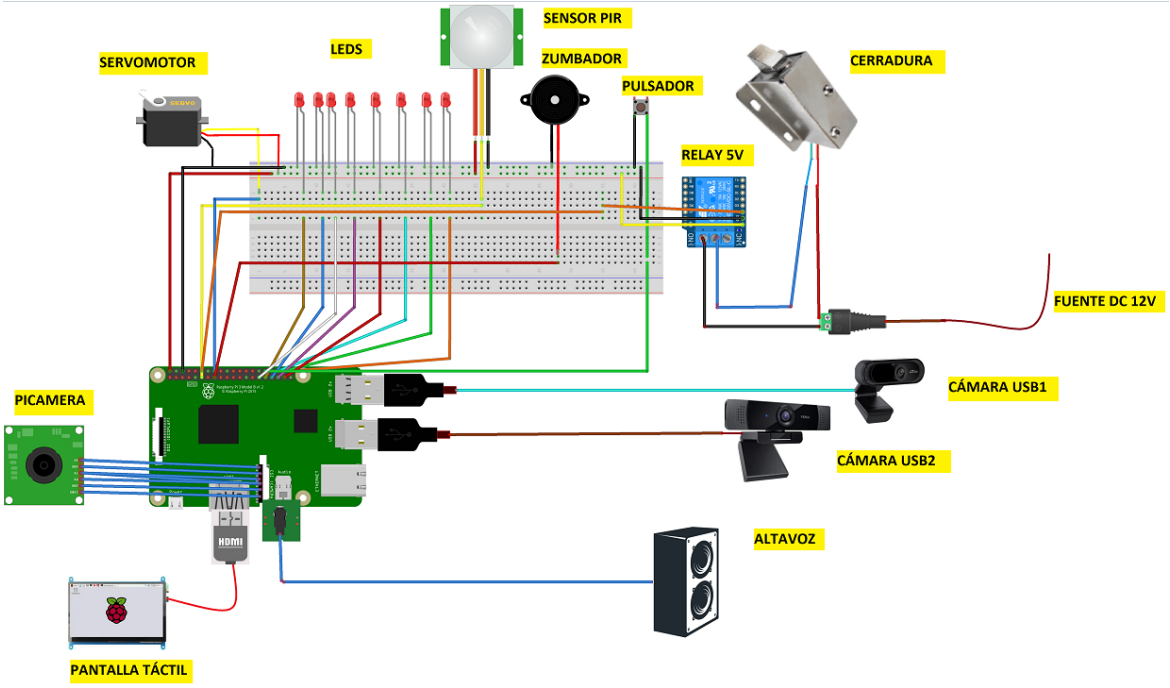


Figura 12 Conexiones de los componentes del sistema domótico

Para la parte de iluminación externa, se usa el circuito del siguiente esquema:

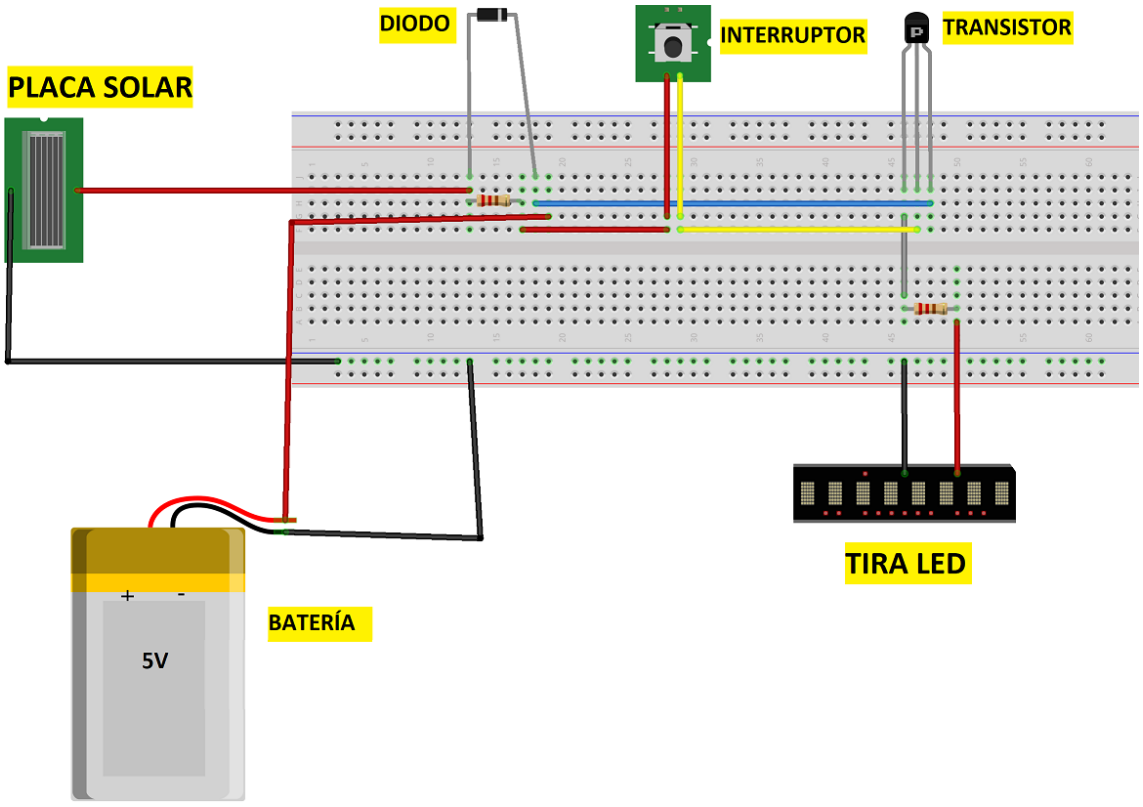


Figura 13 Esquema del circuito de iluminación externa

4.2 Método de funcionamiento del sistema:

El proyecto se centra en un sistema de control de vivienda inteligente que utiliza una Raspberry Pi configurada con las librerías necesarias. El éxito del sistema depende de ambas configuración y automatización adecuadas de la Raspberry Pi.

Una vez configurada y con los componentes hardware conectados, se asegura que el sistema se inicie automáticamente al arrancar la Raspberry Pi, lo que garantiza su disponibilidad constante. El proyecto proporcionará detalles sobre la configuración y el método de inicio automático, permitiendo una gestión eficiente de dispositivos y sistemas de vivienda inteligente, lo que mejora la comodidad y el control para los usuarios.

El sistema presenta las siguientes funcionalidades:

- Control del sistema a través de la aplicación web (Apartado 4.2.1)
- Reconocimiento facial (Apartado 4.2.2)
- Apertura y cierre de la puerta principal de la vivienda (Apartado 4.2.2).
- Control remoto de la cochera (Apartado 4.2.3)
- Simulación de presencia. (Apartado 4.2.4).
- Detección de presencia y notificaciones de seguridad (Apartado 4.2.5).
- Streaming de vídeo en directo del interior de la vivienda (Apartado 4.2.6).
- Control remoto de todos los elementos de la vivienda (Apartado 4.2.7)
- Sistema de iluminación externa (Apartado 4.2.8)
- Envío de notificaciones a usuarios en tiempo real. (Apartado 4.2.9)
- Gestión de usuarios del sistema (Apartado 4.2.10)
- Registro de eventos y acciones en ficheros de texto. (Apartado 4.2.11)

4.2.1 Desarrollo de la aplicación web

La vista principal de la aplicación ConnectGuard Residence es un formulario de inicio de sesión, que se presenta a continuación.

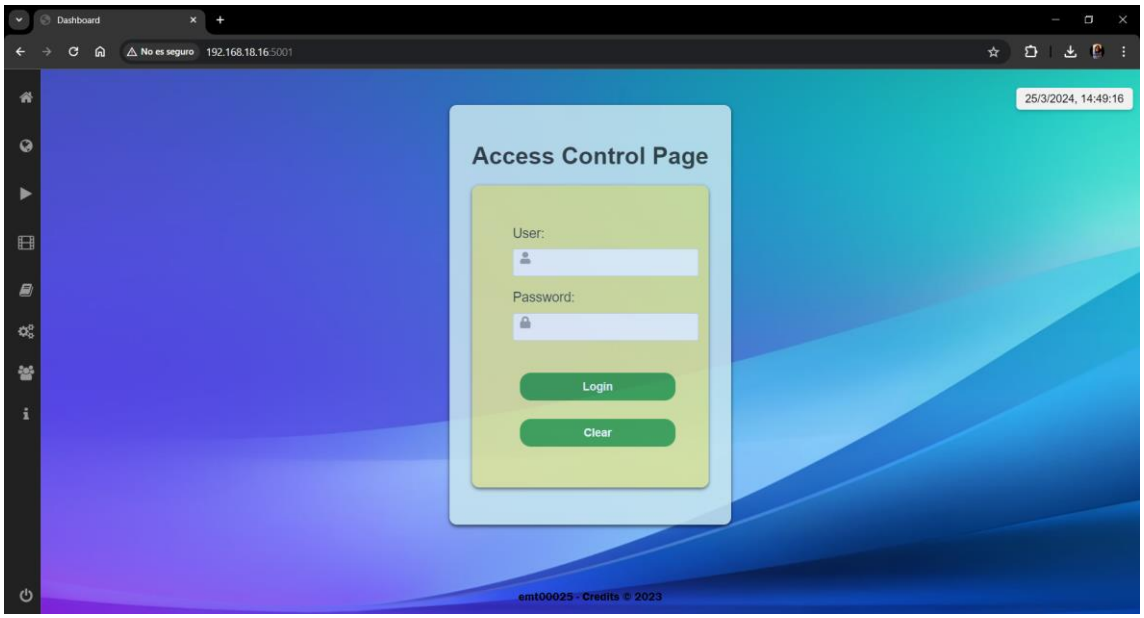


Figura 14 Vista principal de la aplicación web

Todo el desarrollo y la interacción entre cliente y servidor se ha llevado a cabo utilizando el servidor Flask (ver apartado 4.2.1.1) como base. La interfaz web ha sido creada utilizando HTML y CSS para proporcionar una experiencia de usuario intuitiva y atractiva. Para facilitar el acceso a la aplicación durante el desarrollo, se ha utilizado Ngrok (ver apartado 4.2.1.2) para la publicación temporal en línea. Además, se han implementado notificaciones utilizando Pushbullet (ver apartado 4.2.1.4) para mantener a los usuarios informados de manera efectiva.

4.2.1.1 Servidor Flask

Flask es un micro marco de desarrollo web en Python, conocido por su simplicidad y flexibilidad, lo que lo hace ideal para aplicaciones web pequeñas y medianas. Permite definir rutas, vistas y modelos de datos, gestionando la comunicación con los clientes y la lógica de la aplicación. Además, Flask facilita la organización del código mediante el uso de Blueprints, permitiendo una separación modular que mejora el mantenimiento y la escalabilidad. Proporciona un entorno ágil y eficiente para la implementación del lado del servidor y es una herramienta sólida para construir APIs web [33].

La aplicación que permite controlar el sistema de hogar inteligente se basa en un servidor Flask. La configuración de la aplicación se hace de la siguiente manera:

```
app = Flask(__name__, template_folder='templates')
app.config['SECRET_KEY'] = 'mysecretkeyis'
```

Figura 15 Configuración de la aplicación Flask

Lo que permite especificar la ubicación de las plantillas HTML y establecer una llave secreta para la seguridad de la sesión.

Esta aplicación ha sido creada utilizando las tecnologías web fundamentales, como HTML, CSS y JavaScript. A través de esta aplicación, los usuarios podrán acceder y gestionar de manera sencilla todos los dispositivos y sistemas de su hogar, brindando una experiencia de control eficiente y conveniente.

Usando Pushbullet las notificaciones llegan al usuario de tipo administrador en tiempo real, cada vez que se realice una acción en el sistema.

En los siguientes apartados se detallará el código de interacción entre cliente y servidor para cada una de las funcionalidades del sistema.

4.2.1.2 Ngrok

Ngrok es una herramienta que permite exponer un servidor local a Internet mediante la creación de túneles seguros. Se utiliza frecuentemente para el desarrollo y pruebas de aplicaciones web, ya que facilita el acceso remoto a servidores locales sin necesidad de configurar routers o firewalls. Esta herramienta proporciona una URL pública que redirige el tráfico hacia el servidor local, permitiendo a los desarrolladores probar y demostrar aplicaciones en tiempo real [34].

Ngrok se ha empleado con el fin de obtener una URL pública que permite al acceso al sistema de control domótico desde cualquier ubicación. De hecho, una vez se enciende la Raspberry Pi, se ejecuta el script especificado en el apartado I.V Inicio automático para que arranque la aplicación.

El dominio estático que se ha usado para acceder a la aplicación es:

https://measured-moth-intensely.ngrok-free.app

Y es posible acceder mediante el siguiente código QR

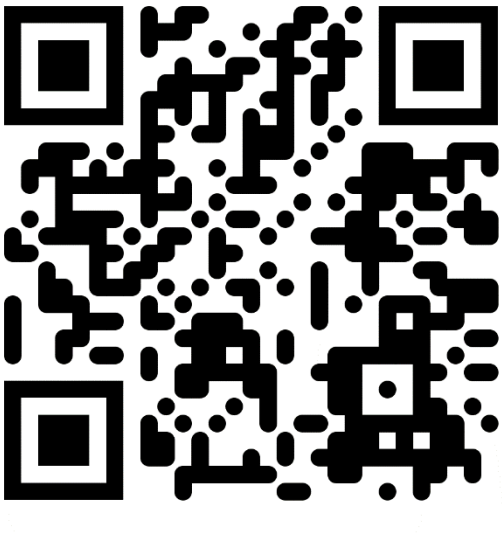


Figura 16 Código QR de acceso a la aplicación web

4.2.1.3 Apache web server

Apache Web Server es uno de los servidores web más populares y ampliamente utilizados en el mundo. Es un software de código abierto que permite a los desarrolladores servir contenido web de manera eficiente y segura. Apache soporta una amplia variedad de módulos que extienden su funcionalidad, incluyendo la autenticación, la administración de permisos, la configuración de sitios virtuales, y más. Es conocido por su robustez, flexibilidad y rendimiento.[35]

Para asegurar la eficacia y accesibilidad de las capturas de seguridad, se optó por implementar el servidor Apache debido a su reputación como una solución confiable y de código abierto. La selección de este servidor no solo garantiza la estabilidad operativa, sino que también ofrece flexibilidad para adaptarse a las necesidades específicas del proyecto.

En el apartado I.II del anexo I- se detallará paso a paso el proceso de instalación, configuración y optimización del servidor Apache, brindando así una referencia completa para su implementación exitosa en el entorno de seguridad del sistema [36].

La configuración de especifica en el anexo de configuración con los pasos y los comandos necesarios.

4.2.1.4 Pushbullet

Pushbullet es una herramienta que facilita la comunicación entre dispositivos al permitir la transferencia instantánea de mensajes, enlaces, archivos y notificaciones. Es

conocida por su simplicidad, ofreciendo integración entre diversos dispositivos, como teléfonos móviles, tabletas y computadoras [1].

Pushbullet es la herramienta empleada en este TFG para permitir que los usuarios de tipo administrador reciban alertas y notificaciones sobre eventos detectados en la vivienda, en tiempo real. La configuración de la cuenta Pushbullet se describe detalladamente en el apartado I.IV del anexo I-, y la funcionalidad de las notificaciones de detalla en el Sistema de envío de notificaciones al usuario

4.2.2 Control de acceso a la casa mediante reconocimiento facial

En el ámbito de este proyecto de desarrollo domótico, se ha implementado una funcionalidad avanzada que converge el uso de un sensor de movimiento por infrarrojos pasivos (PIR), una cámara de vigilancia y una cerradura ubicada en la puerta principal, con el objetivo de fortalecer la seguridad y la interacción en el acceso a la vivienda.

El sistema opera en un escenario donde el sensor PIR detecta movimiento en la proximidad de la puerta principal, la cámara USB de vigilancia se activa capturando una imagen de la persona presente en la entrada, y la guarda en el archivo de grabaciones de seguridad. La imagen capturada se envía al usuario a través de una notificación utilizando el servicio Pushbullet, proporcionando un registro visual de la presencia detectada. Además, esta imagen se archiva en una colección específica denominada "grabaciones" en el sistema domótico, proporcionando un historial de eventos de seguridad para futuras referencias. Esta funcionalidad se describe con detalle en el apartado 4.2.5

En caso de que el timbre esté accionado se activa la cámara, y se desencadena un proceso de reconocimiento facial utilizando un código implementado en Python. Si la identificación facial es exitosa, la cerradura se desbloquea de manera temporal durante un lapso de 5 segundos, permitiendo el acceso al usuario reconocido antes de cerrarse automáticamente. Este enfoque ofrece una capa adicional de seguridad y comodidad, al permitir el acceso autorizado sin necesidad de llaves físicas y al mismo tiempo garantizando un nivel óptimo de seguridad mediante el reconocimiento facial.

Cuando de pulsa el timbre de la casa, se activa el módulo de la cámara, y se ejecuta el código que permite realizar el reconocimiento facial, que se detalla a continuación

El primer paso consiste en realizar la inicialización de las variables como el “dataPath” que es donde se guardan las imágenes de entrenamiento del modelo de reconocimiento facial, el “face_recognizer” que representa el modelo de reconocimiento, y el “cap” que especifica la cámara que captura las caras a reconocer.

A continuación, se define el pin de la cerradura (pin18) y se inicializa la duración máxima de realizar el reconocimiento facial, en este caso 30 segundos.

```
def reconocimiento_facial():
    dataPath = '/home/pi/Desktop/elyssar_code/my_project/rec_facial/test'
    imagePaths = os.listdir(dataPath)
    print('imagePaths=', imagePaths)


    face_recognizer = cv2.face.LBPHFaceRecognizer_create()
    face_recognizer.read('modeloLBPHFace.xml')
    faceClassif = cv2.CascadeClassifier(cv2.data.haarcascades + 'haarcascade_frontalface_default.xml')

    camera = PiCamera()
    camera.resolution = (640, 480)
    camera.framerate = 32
    rawCapture = PiRGBArray(camera, size=(640, 480))

    GPIO.setup(18, GPIO.OUT)  # Salida para controlar la cerradura
    GPIO.output(18, GPIO.HIGH)

    tiempo_limite = 30
    reconocido = False
    inicio_tiempo = time.time()
```

Figura 17 Inicialización de parámetros

En la siguiente parte se explica la lógica del reconocimiento, donde el código implementa un sistema de reconocimiento facial en tiempo real utilizando la PiCamera, que captura continuamente imágenes y las convierte a escala de grises para detectar rostros con un clasificador Haar Cascade.

Una vez detectados, los rostros se extraen, se redimensionan y se someten a un modelo de reconocimiento facial.

Si el rostro es identificado con alta confianza, el sistema cierra la cámara, activa un mecanismo para abrir la cerradura, registra el evento y envía una notificación indicando un reconocimiento exitoso, luego de un retraso cierra la cerradura. En caso de reconocimiento fallido, cierra la cámara, registra el intento fallido y envía una alerta. Tras este aviso, los usuarios del sistema pueden comprobar la identidad de la persona que accionó el pulsador para intentar acceder a la vivienda. Para ello, deben acceder a la web de control remoto para visualizar la imagen de la persona y, en caso de querer permitirle el acceso, podrán activar el sistema de apertura remoto, permitiendo así el acceso. Todo este proceso se detiene si se reconoce a alguien o se supera un tiempo límite, asegurando la limpieza de los recursos de GPIO.

```
for frame in camera.capture_continuous(rawCapture, format="bgr", use_video_port=True):
    image = frame.array
    gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    faces = faceClassif.detectMultiScale(gray, 1.3, 5)

    for (x, y, w, h) in faces:
        rostro = gray[y:y + h, x:x + w]
        rostro = cv2.resize(rostro, (150, 150), interpolation=cv2.INTER_CUBIC)
        result = face_recognizer.predict(rostro)

        cv2.putText(image, '{}'.format(result), (x, y - 5), 1, 1.3, (255, 255, 0), 1, cv2.LINE_AA)

        if result[1] < 100:
            reconocido = True
            camera.close()
            GPIO.output(18, GPIO.LOW) #abrir cerradura
            sleep(5)
            print('Welcome ' + imagePaths[result[0]])
            log("Front door open for {} ".format(imagePaths[result[0]]), level="INFO")
            push = dev.push_note("Alert", "Front door unlocked after successful facial recognition.")
            GPIO.output(18, GPIO.HIGH)
            break
        else:
            reconocido = False
            camera.close()

            log("Someone tried to access from the front door. Unsuccessful facial recognition.", level="INFO")
            push = dev.push_note("Alert", "Unsuccessful facial recognition at front door.")
            break

    cv2.imshow("facial_recognition", image)
    key = cv2.waitKey(1) & 0xFF

    tiempo_actual = time.time()
    tiempo_transcurrido = tiempo_actual - inicio_tiempo

    if key == ord("q") or reconocido or tiempo_transcurrido >= tiempo_limite:
        GPIO.cleanup()
        cv2.destroyAllWindows()
        break
    rawCapture.truncate(0)
```

Figura 18 Lógica de reconocimiento facial

A continuación, se definen los pines del pulsador (pin12) y del zumbador (pin 22), y el código configura un sistema de monitoreo de un botón utilizando GPIO.

```
GPIO.setmode(GPIO.BCM)
BUZZER_PIN = 22
BUTTON_PIN = 12
GPIO.setup(BUTTON_PIN, GPIO.IN, pull_up_down=GPIO.PUD_UP)
GPIO.setup(BUZZER_PIN, GPIO.OUT)
```

Figura 19 Definición de parámetros

Cuando se presiona el botón, el zumbador se activa y se inicia un proceso de reconocimiento facial llamando a la función descrita en la Figura 18 Lógica de reconocimiento facial. Si se detecta y reconoce un rostro con alta confianza, se activa un mecanismo para abrir una cerradura, se registra el evento y se envía una notificación de éxito. Si el reconocimiento falla, se registra el intento y se envía una alerta. La funcionalidad se ejecuta en un hilo separado que asegura que el sistema esté continuamente monitoreando el estado del botón.

```
GPIO.setmode(GPIO.BCM)
BUZZER_PIN = 22
BUTTON_PIN = 12
GPIO.setup(BUTTON_PIN, GPIO.IN, pull_up_down=GPIO.PUD_UP)
GPIO.setup(BUZZER_PIN, GPIO.OUT)

# Crear un bloqueo
lock = threading.Lock()


def monitor_button():
    while True:
            button_state = GPIO.input(BUTTON_PIN)
            if button_state == GPIO.LOW:
                print("bell rung")
                GPIO.output(BUZZER_PIN, GPIO.HIGH)
                time.sleep(1)
                GPIO.output(BUZZER_PIN, GPIO.LOW)  # Apagar el buzzer
                with lock:
                    reconocimiento_facial()  # Llamar a la función de reconocimiento facial
                print("Proceso de reconocimiento facial terminado")


                time.sleep(0.5)
            time.sleep(0.5)
```

Figura 20 Función para simular el timbre

El siguiente flujograma ilustra de manera visual el proceso detallado de la funcionalidad, proporcionando una guía clara y concisa para su comprensión y seguimiento.

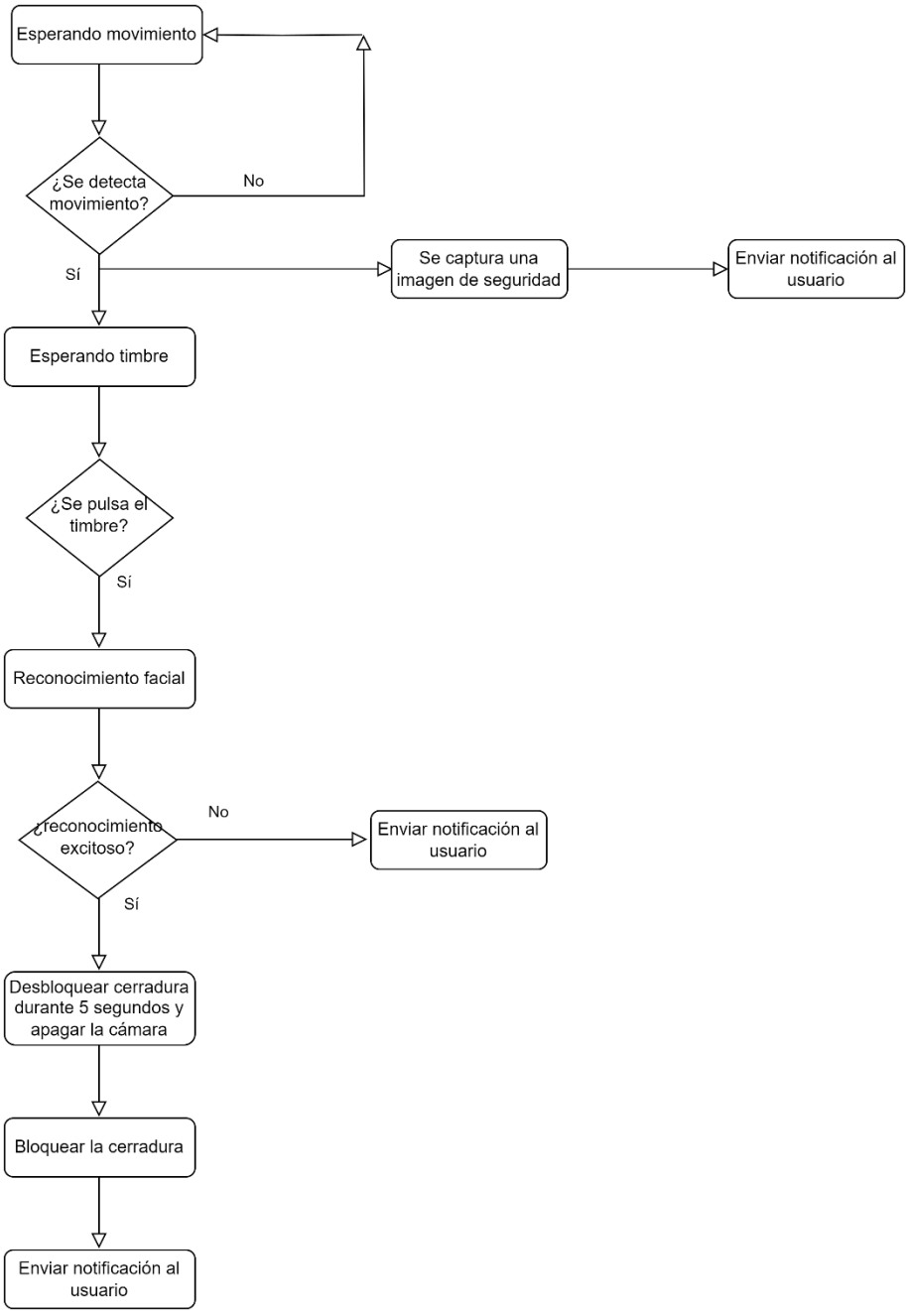


Figura 21 Flujograma del control de acceso a casa

Este sistema integral de seguridad y reconocimiento facial no solo mejora la protección física del hogar, sino que también permite una interacción segura y sin fricciones para los usuarios autorizados, resaltando así la versatilidad y la eficacia de la implementación domótica.

4.2.3 Acceso a la cochera

En el marco de la optimización del acceso vehicular, se ha implementado una cochera automatizada con control remoto para gestionar la entrada y salida de vehículos en el entorno domótico. Una vez situado en frente de la puerta, el usuario podrá abrirla remotamente desde la interfaz de la aplicación web, y cuando esté dentro, podrá cerrarla de la misma manera, pulsando el botón que corresponde a dicha funcionalidad.

El siguiente flujograma representa la funcionalidad anterior:

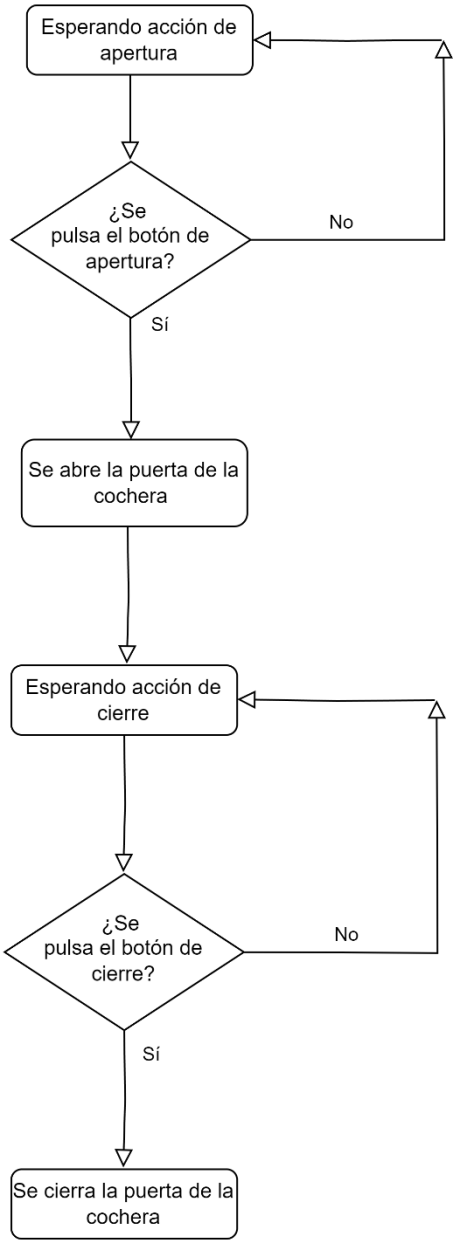


Figura 22 Flujograma del control remoto de la puerta de la cochera

El control de la apertura el cierre de la puerta se realiza mediante un servomotor que se instala en la cochera.

Primero, el pin GPIO 23 se configura como salida y se inicializa el objeto PWM (pwm1) con un ciclo de trabajo inicial de 0.

La función set_angle(pwm, angle) calcula el ciclo de trabajo necesario para posicionar el servomotor en un ángulo específico, utilizando la fórmula estándar para el servomotor.

La función open_door() abre la puerta del garaje moviendo el servomotor a un ángulo de 90 grados. Utiliza un hilo para ejecutar esta operación de manera asincrónica, esperando luego 10 segundos antes de completar la función. Maneja excepciones por interrupción de teclado (KeyboardInterrupt), deteniendo el PWM y limpiando los pines GPIO al finalizar la operación.

La función close_door() cierra la puerta del garaje moviendo el servomotor a un ángulo de 0 grados. Al igual que open_door(), utiliza un hilo para la ejecución asincrónica y maneja excepciones de manera similar, asegurando una salida limpia y ordenada.

```
#define functions to control garage door
GPIO.setup(23, GPIO.OUT)

pwm1 = GPIO.PWM(23, 50)
pwm1.start(0)

def set_angle(pwm, angle):
    duty = angle / 18 + 2
    pwm.ChangeDutyCycle(duty)
    time.sleep(1)
    pwm.ChangeDutyCycle(0)

def open_door():
    try:
        thread1 = threading.Thread(target=set_angle, args=(pwm1, 90))
        thread1.start()
        thread1.join()
        time.sleep(10)
    except KeyboardInterrupt:
        pwm1.stop()
        GPIO.cleanup()

def close_door():
    try:
        thread1 = threading.Thread(target=set_angle, args=(pwm1, 0))
        thread1.start()
        thread1.join()
        time.sleep(1)
    except KeyboardInterrupt:
        pwm1.stop()
        GPIO.cleanup()
```

Figura 23 Funciones para controlar la puerta de la cochera

Dicha puerta será controlada remotamente a través de la aplicación web, se le asigna el pin 99, como se muestra en el código a continuación.

```python
@app.route('/control', methods=['POST'])
def control():
    pin = int(request.form['pin'])
    action = request.form['action']


    if pin == 18:

        if action == 'on':
            GPIO.output(18, GPIO.LOW)  # Abre la cerradura


        elif action == 'off':
            GPIO.output(18, GPIO.HIGH)


    elif pin == 99:
        if action == 'on':
            open_door()
        elif action =='off':
            close_door()


    else:
        # Funcionalidad para los demás pines
        if action == 'on':
            GPIO.output(pin, GPIO.HIGH)
        elif action == 'off':
            GPIO.output(pin, GPIO.LOW)

    return 'OK'
```

Figura 24 Función para controlar elementos de la vivienda

El proceso de control de la puerta y el manejo de la solicitud enviada desde la web viene detallado en el apartado 4.2.7

Por motivos de seguridad, se ha instalado un sensor PIR cerca de la puerta de la cochera, y al detectar la presencia de algún movimiento, se activa la cámara USB de vigilancia, estratégicamente ubicada y se captura una foto de la persona u del objeto presente y se manda una notificación al usuario administrador mediante Pushbullet. El código detallado para esta funcionalidad se presenta en el apartado Imágenes de vigilancia de .

La combinación de sensores, cámaras y sistemas de control motorizado proporciona una solución completa y automatizada para el control de acceso vehicular en el entorno domótico.

4.2.4 Sistema de simulación de presencia

En el marco de este proyecto, se ha desarrollado una funcionalidad fundamental que permite al usuario gestionar un sistema de simulación de presencia a través de la interfaz web del sistema domótico. Esta característica ofrece al usuario el control sobre la activación de elementos clave, como un altavoz y la iluminación de la vivienda, con el propósito de simular la presencia de personas en el hogar durante su ausencia. Al acceder a la página web, el usuario puede personalizar la simulación según sus

preferencias, determinando el volumen de la música deseada, así como la frecuencia con la que se encienden y apagan las luces.

La configuración de esta simulación incluye la posibilidad de especificar el número de veces que se activará el sistema, lo que es el número de ciclos. El intervalo predeterminado entre cada encendido es de 60 segundos, sin embargo, este valor puede ser modificado por el usuario según sus preferencias. Además, se puede definir la duración del ciclo, lo que es el tiempo total en segundos durante el cual cada ciclo estará operativo.

La Figura 25 muestra el flujograma de este sistema de seguridad.

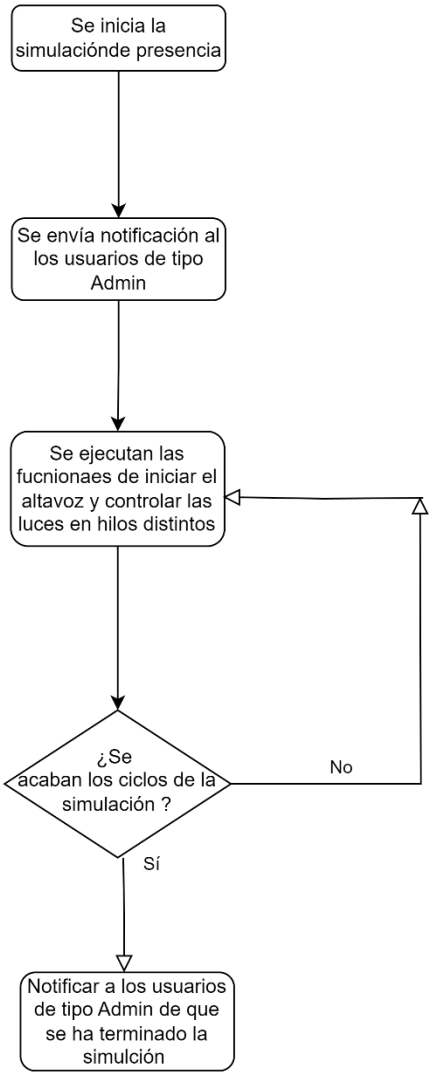


Figura 25 Flujograma e la función de simulación de presencia

Por ejemplo, si el usuario elige una frecuencia “High” y un tiempo de ciclo de 60 segundos e intervalo de ciclo de 10 segundos, el sistema se activará en 20 ocasiones, con las luces activas durante 60 segundos en cada ciclo. Después de cada periodo activo, se apagarán durante 10 segundos antes de reiniciar el siguiente ciclo.

Al inicio de cada simulación, se activará también un archivo de música de duración entre 3 y 4 minutos, independientemente de los parámetros que elige el usuario. Este archivo se elige aleatoriamente del fichero que contiene los distintos ficheros.

El usuario tiene la posibilidad de asignar valores aleatorios a los parámetros de este sistema, y lograría pulsando el botón “Generate Random Values”, como se especifica en el anexo II-Manual de Usuario

En el prototipo del sistema de seguridad, la simulación de presencia se hará con un altavoz y 2 diodos LED (simulando la luz de la habitación principal de la vivienda).

A continuación, se describe el código desarrollado para permitir esta funcionalidad:

```
16
17 def music_play(volume=1.0):
18     if "XDG_RUNTIME_DIR" not in os.environ:
19         os.environ["XDG_RUNTIME_DIR"] = "/tmp/runtime-root"
20     pygame.init()
21     pygame.mixer.init()
22     path = "/home/pi/Desktop/elyssar_code/my_project/simulation/"
23
24     music = [i for i in os.listdir(path) if i.endswith(".mp3")]
25     filename = choice(music)
26     print(filename)
27     pygame.mixer.music.load(path + filename)
28     pygame.mixer.music.set_volume(volume)
29     pygame.mixer.music.play()
30
```

Figura 26 Función de reproducción de música

Se puede ver la línea 20 como se inicializa el módulo Pygame [37], luego se inicializa el módulo mixer que se utiliza para reproducir sonidos y música.

La línea 22 establece la ruta del directorio donde se encuentran los archivos de música. La variable “music” representa una lista de todos los archivos en el directorio especificado y que tienen la extensión ".mp3".

A continuación, se selecciona de manera aleatoria un archivo de música de la lista definida y finalmente se carga el archivo en el reproductor de música de Pygame estableciendo el volumen a un valor predeterminado de 1.

```python
def light_play(frequency, duration, interval ):
    led_pin=21
    GPIO.setup(led_pin, GPIO.OUT)
    cycles = frequency
    for _ in range(cycles):
        GPIO.output(led_pin, GPIO.HIGH)
        sleep(duration)
        GPIO.output(led_pin, GPIO.LOW)
        time.sleep(interval)
```

Figura 27 Función de control de luces

La siguiente función define el método de funcionamiento de las luces del sistema de simulación de presencia, consiste 1 led de color blanco conectado al PIN 21 de la Raspberry pi: la variable “cycles” defenida en la línea 39 es la frecuencia que el usuario puede elegie desde la página web, la variable “duration” define el tiempo de funcionamiento del sistema en cada ciclo, “volumen” es el volumen de reproducción de música en el altavoz y finalmente la variable “Interval” define el tiempo entre dos ciclos consecutivos.

```html
    <label for="frequency">Frequency</label>
    <select id="frequency">
        <option value="5">Low</option>
        <option value="10">Medium</option>
        <option value="20">High</option>
    </select>

     <!-- Information icon for frequency -->
     <div class="tooltip">
      <i class="fas fa-info-circle"></i>
        <span class="tooltiptext"> Number of cycles:
                <br>
                - Low: 5
                <br>
                - Medium: 10
                <br>
                - High: 20
        </span>
      </div>
    <br>


    <label for="duration">Cycle Duration (seconds):</label>
    <input type="number" id="duration" min="10" value="60">
     <!-- Information icon for frequency -->
     <div class="tooltip">
      <i class="fas fa-info-circle"></i>
        <span class="tooltiptext"> The amount of time that a cycle remains activated.
        </span>
      </div>
     <br>


        <label for="intercycletime">Cycle Interval (seconds):</label>
    <input type="number" id="intercycletime" min="10" value="60">
     <!-- Information icon for frequency -->
     <div class="tooltip">
      <i class="fas fa-info-circle"></i>
        <span class="tooltiptext"> The amount of time between consecutive cycles.
        </span>
      </div>
```

Figura 28 Vista de variables modificables por el usuario

Como se puede ver en la Figura 28, la opción “Low” corresponde a un número de ciclos de 5, La opción “Medium” se corresponde a un número de ciclos de 10, y “High” con 20 ciclos.

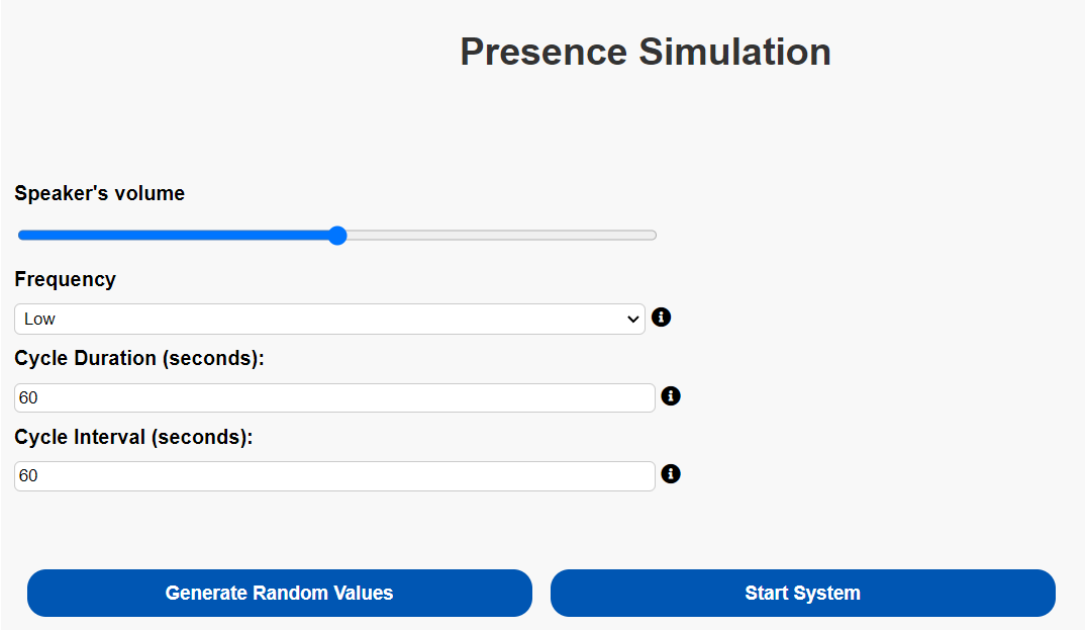


Figura 29 Página web del sistema de simulación de seguridad

Al pulsar el botón "Start System" en la interfaz web, la Raspberry Pi envía las instrucciones necesarias para activar el altavoz y controlar las luces de acuerdo con las preferencias establecidas por el usuario, proporcionando así una simulación de presencia efectiva y personalizable.

```
startButton.addEventListener('click', () => {
    if (!isRunning) {
        startButton.disabled = true;
        randomvalues.disabled = true;
        const volume = parseInt(volumenInput.value);
        const frequency = parseInt(frecuenciaSelect.value);
        const duration = parseInt(tiempoInput.value);
        const interval = parseInt(intercycleInput.value);

        // Datos a enviar al servidor
        const data = {
            volume: volume,
            frequency: frequency,
            duration: duration,
            interval: interval,

        };

        // Enviar solicitud POST al servidor para iniciar la simulación
        fetch('/start_simulation', {
            method: 'POST',
            headers: {
                'Content-Type': 'application/json',
            },
            body: JSON.stringify(data),
        })
        .then(response => {
            if (response.status === 200) {
            //show console log for debugging reasons:
                console.log(`Security simulation started with volume ${volume}, frequency ${frequency}, duration ${duration} seconds, intercycle ${interval} seconds`)
                isRunning = true;
                startButton.disabled = true;
```

Figura 30 Función para enviar datos al servidor

En la Figura 30 se muestra la solicitud enviada desde la página web hasta el servidor para poder iniciar la simulación, el usuario puede especificar los valores del

volumen, frecuencia y tiempo de funcionamiento, y finalmente darle al botón “Start System” para que se envíe la petición.

Cuando se pulsa el botón “Generate Random Values”, los campos se rellenan con valores aleatorios, la función que permite realizar esta acción se presenta en la Figura 31:

```
176     document.getElementById('randomvalues').addEventListener('click', function() {
177     document.getElementById('volume').value = Math.floor(Math.random() * 101);
178     document.getElementById('frequency').selectedIndex = Math.floor(Math.random() * 3);
179     document.getElementById('duration').value = Math.floor(Math.random() * 101) + 50;
180     document.getElementById('intercycletime').value = Math.floor(Math.random() * 101) + 50;
181 });
```

Figura 31 Generar valores aleatorios

En el lado servidor, se reciben los datos de la solicitud Post, y se crean dos hilos para ejecutar ambas funciones descritas en Figura 26 y Figura 27. Además, como se ve en la línea 364, se envía una notificación al administrador del sistema para notificarle de que se ha iniciado el sistema de simulación se seguridad, y finalmente, en la línea 379, se guarda un registro de esta acción en la base de datos de registro, y también en un fichero local, mediante la función “log” descrita en el apartado 4.2.11

```
349 ##security simulation functionality
350 simulation_running = False
351 @app.route('/start_simulation', methods=['POST'])
352 def start_simulation():
353     if request.method == 'POST':
354         # Obtener los datos del cuerpo de la solicitud POST
355         data = request.get_json()
356 
357         # Extrae los datos necesarios de 'data'
358         volume = data['volume']
359         frequency = data['frequency']
360         duration = data['duration']
361         interval= data['interval']
362 
363         #Envia notificación al usuario
364         push = dev.push_note("Alert!", "Security simulation system started!")
365         # Crear los hilos para ejecutar las funciones simultáneamente
366         light_thread = threading.Thread(target=light_play, args=(frequency, duration, interval))
367         music_thread = threading.Thread(target=music_play, args=(volume/100,))
368 
369         # Iniciar ambos hilos
370         light_thread.start()
371         music_thread.start()
372 
373         # Esperar a que ambos hilos terminen
374         light_thread.join()
375         music_thread.join()
376 
377 
378         #log of starting simulation
379         log('Security simulation started: Volume={}, frq={}, t={} seconds, intercycle={} seconds.'.format(volume, frequency, duration,interval) ,level="INFO")
380 
381         return 'Simulation started correctly', 200
```

Figura 32 Función para iniciar la simulación

Este desarrollo amplía las capacidades de la maqueta de casa domótica, ofreciendo una solución innovadora y adaptable a las necesidades de seguridad y comodidad del usuario.

4.2.5 Imágenes de vigilancia de seguridad

En el sistema de seguridad, se ha implementado una función específica que lleva a cabo imágenes de vigilancia exclusivamente al detectar movimiento, haciendo uso de sensores PIR ubicados en el exterior de la residencia.

Esta operación se activa de forma automática: una vez que alguno de los sensores PIR detecta movimiento, se inicia la captura de una imagen y el archivo resultante se transmite al administrador del sistema mediante la aplicación Pushbullet. De manera simultánea, la foto se guarda en el sistema para su almacenamiento seguro.

A continuación, se presenta el código que permite lograr esta funcionalidad.

```
# Configurar pin del sensor PIR
GPIO.setmode(GPIO.BCM)
pir_sensor1_pin = 17

GPIO.setup(pir_sensor1_pin, GPIO.IN, pull_up_down=GPIO.PUD_DOWN)


#camera USB 2: index 2

def detect_motion(pir_sensor1_pin):
    if GPIO.input(pir_sensor1_pin):
        return True
    else:
        return False


def pir_notification(sensor1_pin):
    if detect_motion(sensor1_pin):
      print ("motion detected")
      # Activa la camara USB y captura una imagen
      capture_image()

      # Guarda la imagen en el archivo de grabaciones de seguridad y ennvia la imagen al usuario a tra de Pushbullet
      image_filename = save_image()

      time.sleep(120)

    return 'Notification processed successfully'
```

Figura 33 Funcionamiento del sistema de vigilancia de seguridad

Para poder grabar la imagen, se he implementado la siguiente función capture_image():

```
def capture_image():
    # Activa la camara USB y captura una imagen: Comando para capturar imagen
    os.system('v4l2-ctl --device=/dev/video2 --set-fmt-video=width=1920,height=1080,pixelformat=JPEG --stream-mmap --stream-to=./image.jpg --stream-count=1')
```

Figura 34 Función para grabar imagen de seguridad

La siguiente etapa sería guardar la imagen grabada, usando la función “save_image()”:

```
def save_image():
    # Genera un nombre de archivo basado en la fecha y hora actual
    current_datetime = datetime.datetime.now().strftime('%Y-%m-%d_%H-%M-%S')
    image_filename = f'alert_{current_datetime}.jpg'
    image_directory = f'/var/www/html/grabaciones/{image_filename}'

    # Mueve la imagen capturada a la carpeta de grabaciones con el nombre generado
    os.rename('image.jpg', image_directory)
    push = dev.push_note("Motion detected","Motion detected in front door! Access the webpage to visualize the saved image.")

    return image_filename
```

Figura 35 Función para guardar imagen de seguridad

A la imagen grabada se le asigna un nombre formado por la fecha y hora del momento de su captura, y se guarda en el directorio especificado en la variable “image_directory”.

Todas las imágenes guardadas se pueden ver en la página web, accediendo a la opción “Recordings”, tal y como se describe en el Anexo II Manual de Usuario.

Una vez guardada la imagen, se le notifica al administrador de que se ha detectado un movimiento en la entrada de la vivienda, enviándole una notificación con Pushbullet.

El texto del mensaje de la notificación sería:

“Motion detected in front door! Access the webpage to visualize the saved image.”

y su asunto es:

“Motion detected”.

Adicionalmente, se ofrece al administrador la posibilidad de acceder a la página web del sistema para visualizar las capturas almacenadas, con opciones adicionales para descargarla o eliminarla según sea necesario, como se explica el Anexo II.I Manual de usuario de administradores.

A continuación, se presenta le código que permite la visualización de las imágenes grabadas.

```python
# Ruta para mostrar la página de recordings
@app.route('/recordings.html' ,methods=['GET'])
def recording_page():
    get_user_type()
    if not session.get('logged_in'):
        return render_template('login.html')

    folder_path = '/var/www/html/grabaciones/'
    image_names = os.listdir(folder_path)

    return render_template('recordings.html', image_names=image_names)
```

Figura 36 Función para devolver la lista de las imágenes de seguridad

Para descargar o borrar cualquiera de las imágenes de grabaciones, se proporcionan las siguientes funciones de la Figura 37:

```python
@app.route('/download/<filename>', methods=['GET'])
def download(filename):
    # Devuelve el archivo solicitado
    folder_path = '/var/www/html/grabaciones/'
    return send_from_directory(folder_path, filename, as_attachment=True)

@app.route('/delete/<filename>', methods=['POST'])
def delete_image(filename):
    # Eliminar el archivo
    folder_path = '/var/www/html/grabaciones/'

    file_path = os.path.join(folder_path, filename)
    os.remove(file_path)
    image_names = os.listdir(folder_path)
    return render_template('recordings.html', image_names=image_names)
```

Figura 37 Funciones para visulaizar y borrar grabaciones de seguridad

La función **download(filename)** responde a solicitudes GET en la ruta '/download/<filename>'. Su objetivo es devolver un archivo específico como una descarga adjunta al navegador del usuario. Se especifica la carpeta donde se encuentra el archivo (folder_path) y se utiliza send_from_directory de Flask para enviar el archivo al cliente como una descarga.

La función **delete_image(filename)** responde a solicitudes POST en la ruta '/delete/<filename>'. Esta función se encarga de eliminar un archivo específico del servidor. Primero, construye la ruta completa del archivo a eliminar (file_path) utilizando la carpeta base (folder_path) y el nombre del archivo recibido como parámetro. Luego, elimina el archivo usando os.remove(file_path). Posteriormente, actualiza la lista de nombres de imágenes en la carpeta (image_names) y renderiza una plantilla HTML ('recordings.html') con la lista actualizada de nombres de imágenes para mostrar al usuario después de eliminar el

Esta acción de queda registrada también en el archivo de logs, como se especifica en el apartado 4.2.11

El diagrama de flujo de la Figura 38 proporcionará una comprensión más clara de esta funcionalidad.

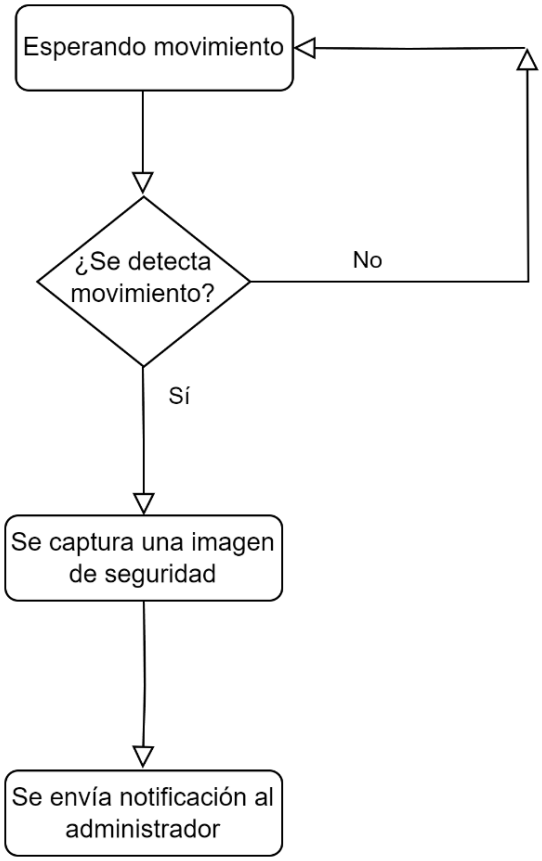


Figura 38: Flujograma de Grabación de Imágenes de Seguridad por Detección de Movimiento

4.2.6 Sistema de Transmisión en Vivo desde el Interior del Hogar

En el marco de la ampliación de las capacidades de seguridad y vigilancia del sistema domótico, se ha incorporado una funcionalidad de transmisión en directo desde la cámara de seguridad ubicada en el interior de la residencia.

El usuario tiene acceso a esta característica a través de la página web, cuya descripción detallada se proporcionará en el apartado 4.2.7 en la documentación. Dentro de la sección designada como "Live Streaming", el usuario se encuentra con dos opciones: "Start" y "Stop". Al seleccionar "Start", se inicia la transmisión en tiempo real desde la cámara de seguridad, permitiendo al usuario visualizar la transmisión en su dispositivo de acceso. Posteriormente, al optar por "Stop", la transmisión se interrumpe de inmediato.

La Figura 39 ilustra el diagrama de flujo de la funcionalidad de transmisión en tiempo real y ayuda a entenderla mejor:

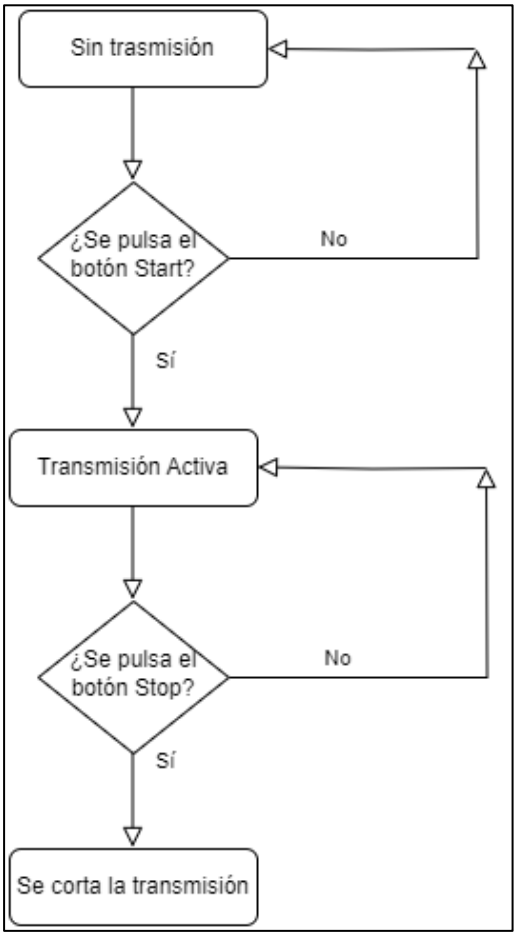


Figura 39 Flujograma de la transmisión en vivo

A continuación, se muestra la vista de la página de transmisión en directo:

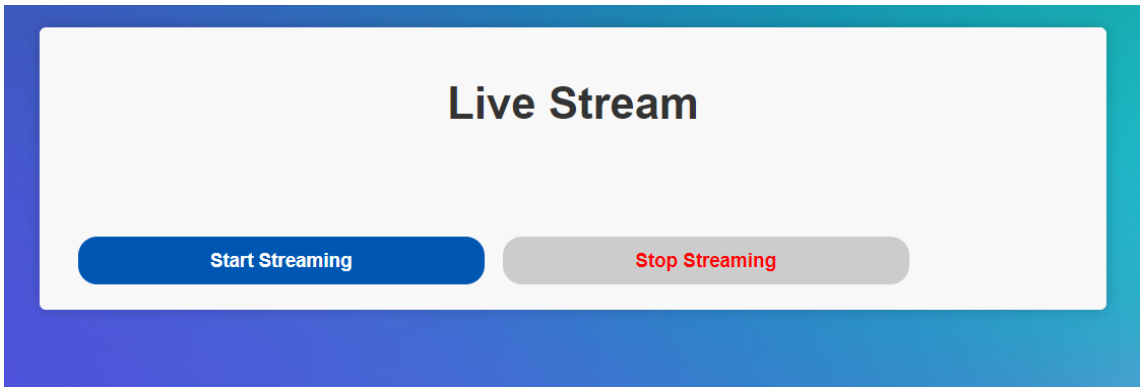


Figura 40 Vista de la página de transmisión en directo

La Figura 41 y Figura 42 ilustran los botones en el código HTML de la página de transmisión en directo y el script de las funciones que permiten enviar las solicitudes de empezar y detener la transmisión al servidor.

```
35      <div id="controls">
36          <button id="start-btn">Start Streaming</button>
37          <button id="stop-btn" disabled>Stop Streaming</button>
38      </div>
```

Figura 41 Botones de control de la transmisión

```
41      <script>
42          $(document).ready(function () {
43              var videoInterval;
44
45              function startStreaming() {
46                  $.ajax({
47                      url: "/video_feed",
48                      success: function (data) {
49                          console.log(data);
50                      }
51                  });
52              }
53
54              $("#start-btn").click(function () {
55                  $(this).prop("disabled", true);
56                  $("#stop-btn").prop("disabled", false);
57                  startStreaming();
58                  $("#video-container").empty();
59                  var video = $('<img>');
60                  video.attr('src', '/video_feed');
61                  $("#video-container").append(video);
62              });
```

Figura 42 Función de solicitar la transmisión

Cuando se hace clic en el botón “Start”, se deshabilita ese botón, se habilita el botón “Stop”, se llama a la función “startStreaming”, se vacía el contenedor de video, se

crea un elemento “img” y se le asigna la fuente del video “/video_feed", que será la ruta de iniciar la transmisión, y finalmente se agrega este elemento al contenedor de video.

```
64              function stopVideo() {
65                  clearInterval(videoInterval);
66                  var blankImageSrc = "";
67                  $("#video-container").empty().append($("<img>").attr("src", blankImageSrc));
68              }
69
70              $("#stop-btn").click(function () {
71                  $(this).prop("disabled", true);
72                  $("#start-btn").prop("disabled", false);
73                  stopVideo();
74                  $.ajax({
75                      url: "/stop_video_feed",
76                      success: function (data) {
77                          console.log(data);
78                      }
79                  });
80              });
81          });
82      </script>
```

Figura 43 Función de solicitar la detención de transmisión

La función “stopVideo” definida en la línea 64 detiene cualquier intervalo de video que se esté reproduciendo, limpia el contenedor de video y muestra una imagen en blanco.

Cuando se hace clic en el botón “Stop”, se deshabilita ese botón, se habilita el botón “Start”, se llama a la función “stopVideo”, y se realiza una solicitud AJAX a “/stop_video_feed”, que será la ruta de detener la transmisión. Los datos recibidos se muestran en la consola del navegador para permitir la depuración del código.

La Figura 44 muestra las rutas que se solicitan con los botones de control.

```
306 #start and stop streaming
307 @app.route("/video_feed")
308 def video_feed():
309     start_camera()
310     return Response(generate(), mimetype="multipart/x-mixed-replace; boundary=frame")
311
312
313 @app.route("/stop_video_feed")
314 def stop_video_feed():
315     stop_camera()
316     return "Video feed stopped."
317
```

Figura 44 Rutas de manejo de las solicitudes en el servidor

Cuando se pulsa “Start”, se solicita la ruta "/video_feed", se llama a la función video_feed() y se inicia la transmisión, y cuando se pulsa “Stop”, se solicita la ruta “/stop_video_feed”, que llama a la función “stop_camera()” con el fin de detener la transmisión. Finalmente se devuelve un mensaje indicando que el video se ha detenido correctamente, para la correcta depuración de código.

Para manejar las solicitudes en la parte del servidor, se han creado las funciones presentadas en la Figura 45.

```
124 #stream functionality
125 camera = None
126 video_feed_active = False
127 #camera USB 1
128 USB_camera_index=1
129
130 def start_camera():
131     global camera, video_feed_active
132     if camera is None:
133         camera = cv2.VideoCapture(USB_camera_index)
134         log("Live streaming from Main Camera started", level="INFO")
135         video_feed_active = True
136
137 def stop_camera():
138     global camera, video_feed_active
139     if camera is not None:
140         camera.release()
141         camera = None
142         log("Live streaming from Main Camera stopped", level="INFO")
143         video_feed_active = False
144
145
146 def generate():
147     while video_feed_active:
148         ret, frame = camera.read()
149         if ret:
150                 gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
151                 (flag, encodedImage) = cv2.imencode(".jpg", frame)
152                 if not flag:
153                     continue
154                 yield(b'--frame\r\n' b'Content-Type: image/jpeg\r\n\r\n' +
155                     bytearray(encodedImage) + b'\r\n')
```

Figura 45 Funciones para manejar las solicitudes de transmisión

- **start_camera():** Esta función inicia la transmisión en vivo desde la cámara principal. Primero, la línea 125 verifica si la cámara está disponible y si es así, la inicializa usando el índice de la cámara USB especificado, y que en este caso se trata de la cámara con índice 1. Luego, la línea 135 establece la variable video_feed_active en True para indicar que la transmisión de video está activa.

- **stop_camera():** Esta función detiene la transmisión en vivo desde la cámara principal. Primero, en la línea 139 se verifica si la cámara está inicializada, y si es así, la libera y la establece en "None". También establece "video_feed_active" en "False" para indicar que la transmisión de video está detenida.

- **generate():** Esta función genera imágenes de video mientras la transmisión de video está activa. Utiliza un bucle while que continúa mientras la transmisión está activa. Dentro del bucle, lee cada cuadro de la cámara convierte el cuadro a escala de grises, lo codifica como una imagen JPEG y lo envía como un objeto bytearray para ser transmitido. Este proceso se repite continuamente mientras la transmisión de video está activa.

Ambas acciones de inicio y detención de la transmisión quedan registradas en un registro específico que se almacena en la base de datos de registros del sistema, esta acción se hace mediante la función de "log". Esta funcionalidad no solo proporciona un acceso conveniente a la transmisión en tiempo real, sino que también asegura un registro detallado de las interacciones del usuario con la cámara de seguridad, mejorando así la monitorización y la gestión remota de la seguridad residencial.

4.2.7 Control de elementos de la vivienda

Con el objetivo de proporcionar una experiencia de control y monitoreo integral, se ha incorporado la funcionalidad de control de elementos domóticos a través de la página web, específicamente bajo la sección denominada "Control Page". Esta sección brinda a los usuarios la capacidad de interactuar con diversos elementos de la vivienda, incluyendo luces, la puerta principal y la puerta de la cochera, mediante botones "ON" y "OFF" para luces, y "ABRIR" y "CERRAR" para puertas. Al seleccionar la opción deseada, el usuario puede realizar la acción correspondiente, ya sea encender o apagar las luces, o abrir y cerrar las puertas.

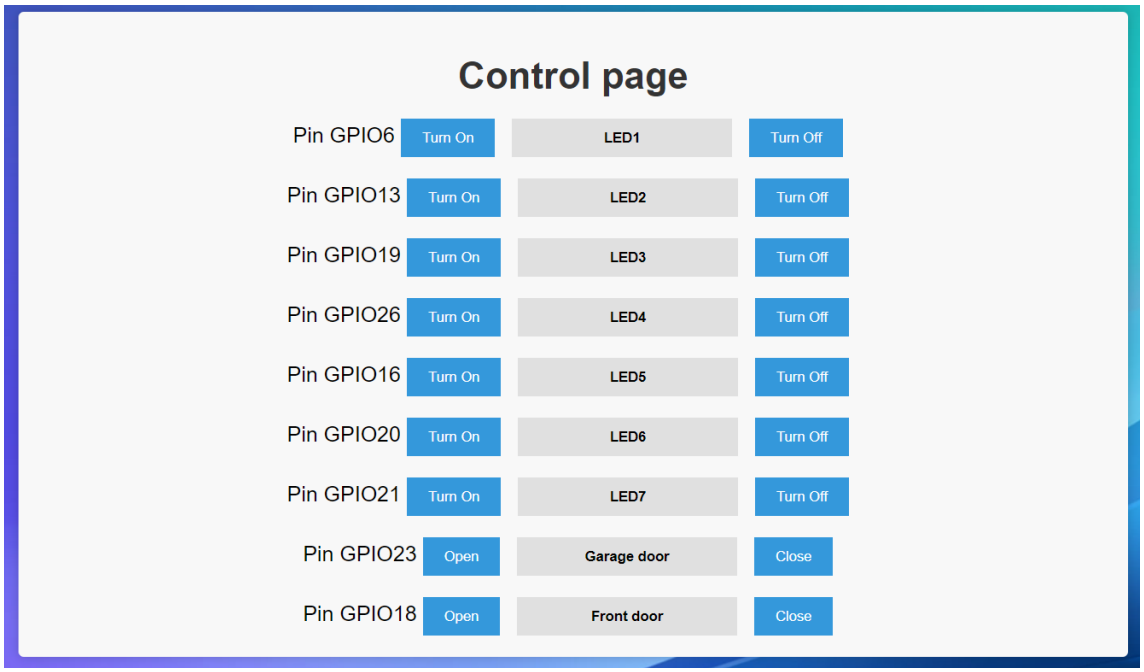


Figura 46: Vista de la página de control remoto de elementos de la vivienda

Desde la vista anterior de la página web, se pueden lanzar varias acciones, por medio de los botones que envían solicitudes POST al servidor.

El código que permite la interacción en tiempo real con los elementos está presentado en la Figura 47:

```
<script>
  $(document).ready(function() {

      function sendRequest(pin, action) {
          $.post('/control', { pin: pin, action: action }, function(data) {
              // Manejar la respuesta del servidor
              console.log(data);
          });
      }

      $('.on-button').click(function() {
          var pin = $(this).data('pin');
          sendRequest(pin, 'on');
      });

      $('.off-button').click(function() {
          var pin = $(this).data('pin');
          sendRequest(pin, 'off');
      });
  });

</script>
```

Figura 47 Código para enviar solictudes de control al servidor

Este script en JavaScript, utilizando jQuery, permite controlar todos los dispositivos de la vivienda desde la interfaz web. Se define una función sendRequest que envía solicitudes POST al servidor Flask con los datos del pin y la acción ('on' o 'off'). Se configuran manejadores de eventos para botones con las clases.on-button y .off-button, que envían la solicitud correspondiente cuando se hace clic en ellos. Al recibir la respuesta del servidor, esta se registra en la consola para oder depurar el código.

En el lado servidor, primero se definen los pines de cada elemento.

```
#monitoring functionality
GPIO.setmode(GPIO.BCM)
# define the Pins being used for this functionality
pins = [6,3,19,26,16,20,21]


for pin in pins:
    GPIO.setup(pin, GPIO.OUT)
    GPIO.output(pin, GPIO.LOW)

GPIO.setup(18, GPIO.OUT)
GPIO.output(18, GPIO.HIGH)
```

Figura 48 Definición de los pines de los elementos

Luego se crea una función para manejar solicitudes POST en la ruta '/control'. Su propósito es controlar dispositivos ejecutando acciones basadas en los datos recibidos del formulario POST. La función extrae el número de pin y la acción requerida del formulario, luego realiza operaciones específicas dependiendo del número de pin y la acción recibidos. Esto permite gestionar diversas funciones: activar o desactivar pines GPIO para controlar dispositivos físicos o ejecutar funciones definidas en el sistema, como es el caso del control de la puerta de cochera, proporcionando una interfaz para la automatización y el control remoto dentro de la aplicación web.

```
@app.route('/control', methods=['POST'])
def control():
    pin = int(request.form['pin'])
    action = request.form['action']


    if pin == 18:

        if action == 'on':
            GPIO.output(18, GPIO.LOW)  # Abre la cerradura


        elif action == 'off':
            GPIO.output(18, GPIO.HIGH)


    elif pin == 99:
        if action == 'on':
            open_door()
        elif action =='off':
            close_door()


    else:
        # Funcionalidad para los demás pines
        if action == 'on':
            GPIO.output(pin, GPIO.HIGH)
        elif action == 'off':
            GPIO.output(pin, GPIO.LOW)

    return 'OK'
```

Figura 49 Función para manejar solicitudes de control de elementos

4.2.8 Sistema de iluminación externa

El sistema diseñado para el funcionamiento de los LEDs situados en la zona externa de la vivienda combina de manera eficiente la energía solar, una batería, un transistor y una resistencia que sirve para proteger las luces LEDs.

La placa solar desempeña un papel dual: primero, convierte la energía solar en electricidad para cargar la batería durante el día, asegurando un suministro constante de energía; y segundo, actúa como un sensor de luz, detectando la disminución de la

intensidad lumínica al anochecer para activar el encendido automático de los LEDs. Cuando la placa solar detecta que la luz solar ha disminuido por debajo de cierto umbral, envía una señal al transistor, el cual actúa como un interruptor que permite el flujo de corriente desde la batería hacia los LEDs. Para garantizar la protección de los LEDs y mantener un flujo de corriente seguro, se incluye una resistencia en el circuito. Esta resistencia limita la cantidad de corriente que puede pasar a través de los LEDs, evitando daños por sobre corriente y prolongando la vida útil de los componentes.

La Figura 13 representa el esquema del circuito diseñado para llevar a cabo las funciones descritas anteriormente.

En conjunto, este sistema automatizado aprovecha la energía solar para proporcionar iluminación eficiente y autónoma, encendiéndose automáticamente al anochecer y recargando la batería durante el día para un funcionamiento continuo y sostenible.

4.2.9 Sistema de envío de notificaciones al usuario

En la estrategia de notificaciones en tiempo real implementada en el sistema domótico, se ha optado por utilizar Pushbullet como la herramienta central para informar a los usuarios sobre diversas acciones y eventos relevantes. Pushbullet se presenta como una solución efectiva y versátil para enviar notificaciones instantáneas a dispositivos móviles y navegadores web, permitiendo una comunicación rápida y directa con los usuarios.

La elección de Pushbullet se fundamenta en su facilidad de integración con sistemas de desarrollo y su amplia compatibilidad con una variedad de plataformas, lo que lo convierte en una opción accesible y eficiente. Al utilizar esta herramienta, se habilita la capacidad de enviar mensajes informativos en tiempo real a los usuarios, proporcionando una experiencia interactiva y manteniéndolos actualizados sobre las acciones realizadas en su entorno domótico.

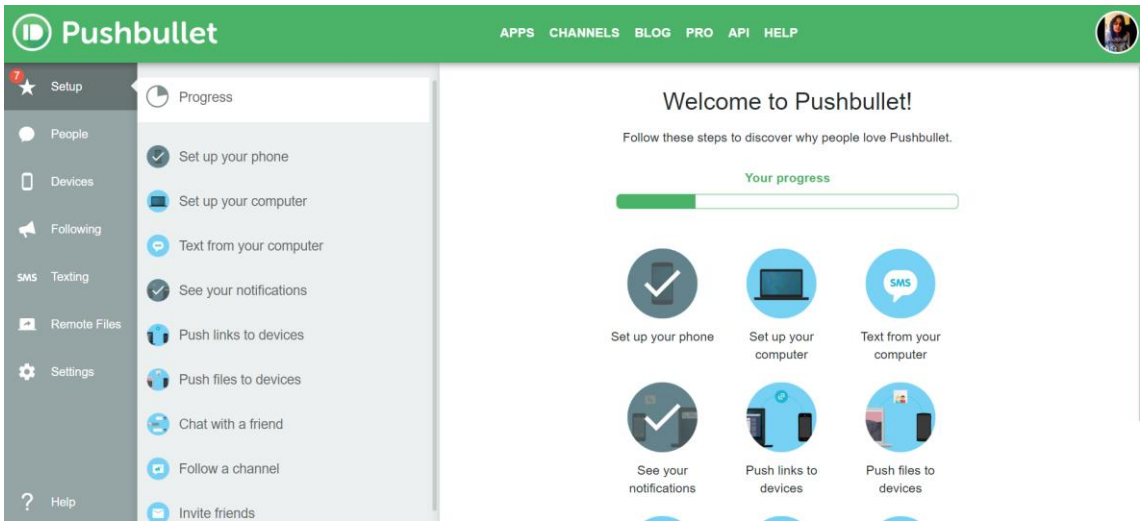


Figura 50 Página principal de Pushbullet

En el contexto del sistema, las notificaciones Pushbullet se utilizan para informar al usuario sobre eventos específicos, como el inicio y detención de la transmisión en directo, cambios en el estado de luces y puertas, así como cualquier otra acción relevante que requiera la atención del usuario. Estas notificaciones están diseñadas para ser claras y concisas, brindando información esencial sobre la acción realizada y facilitando una comprensión inmediata de la situación.

```
# Configurate the connexion to the database
client = MongoClient('mongodb+srv://emt00025:3TXTOQd1gdLJl6FW@smarthome.fk7eyc2.mongodb.net/')
db = client['users']
collection = db['users']
logs_collection = db['logs']

users = collection.find()
for user in users:
    pushbullet_key =  user.get('pushbullet_key')
    pushbullet_id =  user.get('pushbullet_id')
    # pb initialized to None
    pb = None

    if pushbullet_key:
        pb = Pushbullet(pushbullet_key)
    if pushbullet_id and pb:
        dev = pb.get_device(pushbullet_id)
```

Figura 51 Configuración de Notificaciones Pushbullet para Admins

La capacidad de Pushbullet para enviar notificaciones a través de múltiples dispositivos asegura que los usuarios estén informados en tiempo real, independientemente de la plataforma que estén utilizando en ese momento. Además, la integración con la aplicación web y la infraestructura del sistema domótico permite una comunicación fluida y coherente.

Como se ilustra en la Figura 51, el código presentado facilita el envío de notificaciones a los usuarios registrados en Pushbullet. Durante la iteración a través de los documentos de la colección de usuarios en la base de datos “users”, se extraen las claves API y los IDs de dispositivos de Pushbullet de cada usuario. Luego, se establece una conexión con Pushbullet para cada usuario que disponga tanto de una clave como de un ID válidos. Una vez que la conexión está establecida, se encuentra el dispositivo asociado a cada

usuario y, posteriormente, se puede enviar notificaciones a través de Pushbullet a esos dispositivos.

push = dev.push_note("<title>", "<notification text>")

La configuración de la clave API como el ID de dispositivo se detalla en el apartado I.IV del Anexo I-

La Figura 52 muestra un ejemplo de la notificación enviada al usuario cuando se inicia la simulación de presencia en la vivienda.

Figura 52 Ejemplo de notificación en tiempo real

Es importante señalar que únicamente los usuarios de tipo "admin" poseen estos atributos en la base de datos; el resto de usuarios no cuentan con “pushbullet_key” ni “pushbullet_id”.

En resumen, Pushbullet se ha seleccionado como la herramienta de notificación preferida debido a su versatilidad, facilidad de uso y capacidad para proporcionar información instantánea en tiempo real. La integración de Pushbullet en el sistema domótico fortalece la comunicación con los usuarios, garantizando que estén al tanto de las acciones y eventos cruciales que ocurren en su entorno residencial de manera rápida y efectiva.

4.2.10 Base de datos MongoDB

En la estructura fundamental del proyecto de domótica, se ha optado por emplear MongoDB[38] como sistema de gestión de base de datos. MongoDB, un sistema de base de datos NoSQL, se ha seleccionado por su flexibilidad, escalabilidad y capacidad para manejar grandes volúmenes de datos de manera eficiente.

MongoDB desempeña un papel crucial en el control de acceso y la gestión de usuarios en la aplicación web. La información de inicio de sesión y los perfiles de usuario se almacenan de manera segura en colecciones específicas, permitiendo una autenticación rápida y segura durante el proceso de inicio de sesión. Además, MongoDB facilita la creación y eliminación eficiente de usuarios, contribuyendo así a una gestión integral de las identidades en el sistema.

Para ello, se instala la biblioteca PyMongo[20] y se importa el módulo MongoClient [39].

Figura 53 Importar MongoClient

La table de la entidad usuario contine los atributos presentados en la Figura 54:

Usuario	
Nombre	*String*
Contraseña	*String*
Nombre de usuario	*String*
Tipo de usuario	*String*
Id Pushbullet	*String*
Clave Pushbulllet	*String*
Tiempo de sesión	*Int*

Figura 54 Table de la entidad Usuario

Un ejemplo de un usuario de tipo administrador registrado al sistema se presenta a continuación:

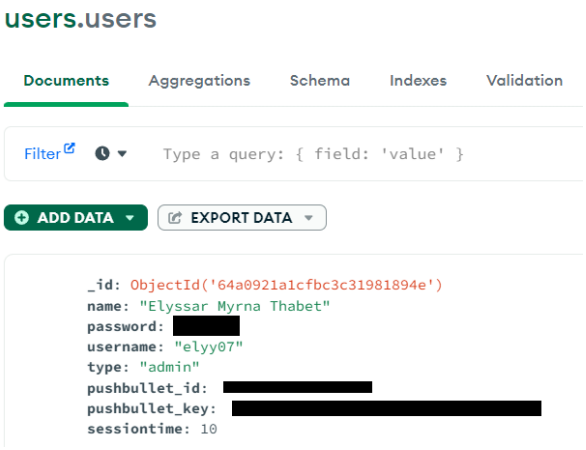


Figura 55 Ejemplo de un usuario de la base de datos

Otra tabla de la base de datos es la que contiene los registros de todos los eventos del sistema.

Registro	
Fecha	String
Tipo	String
Mensaje	String

Figura 56 Atributos de la table de registros

Un ejemplo de algunos registros se presenta a continuación:

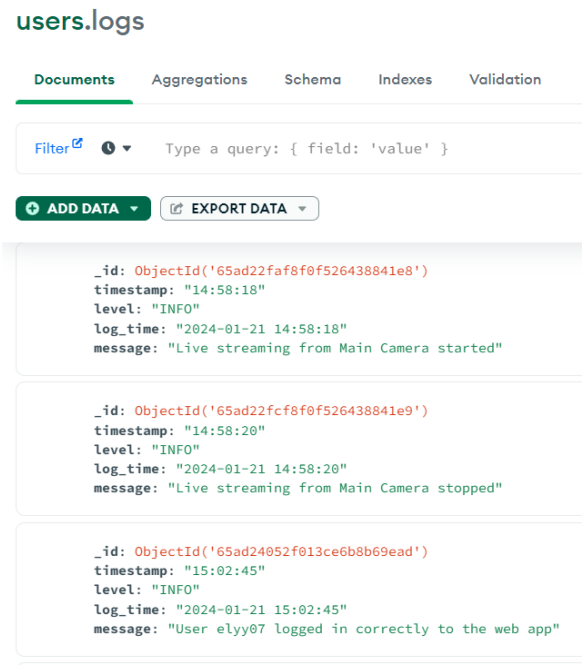


Figura 57 Ejemplo de registros del sistema

La configuración de la conexión a una base de datos MongoDB, alojada en un servicio en la nube, se realiza mediante el código mostrado en la Figura 58. En este

código, se instancia el cliente utilizando la URI especificada y se define la base de datos del proyecto, denominada "users".

Posteriormente, se establecen las colecciones que representan las tablas a utilizar: la colección "users" representa la tabla de usuarios, mientras que la colección "logs" representa la tabla de registros.

```
# Configurate the connexion to the database
client = MongoClient('mongodb+srv://emt00025:3TXTOQd1gdLJl6FW@smarthome.fk7eyc2.mongodb.net/')
db = client['users']
collection = db['users']
logs_collection = db['logs']
```

Figura 58 Configuración de la base de datos

A continuación, se explicarán los métodos de acceso al sistema y el proceso de registro de usuarios.

Desde la página web, el usuario debe validar su acceso ingresando sus credenciales en el formulario de inicio de sesión. La transmisión de los datos se realiza a través de una solicitud POST, como se ilustra en la Figura 59.

```
<script>
    $(document).ready(function() {
        // sending form info to the server
        $('form').submit(function(e) {
            e.preventDefault();

            var username = $('#username').val();
            var password = $('#password').val();

            // verify login info:
            $.post('/verificar-usuario', { username: username, password: password }, function(data) {
                $('#login-result').text(data);

                // Login done correctly: redirect to dashboard page
                if (data === 'Access allowed') {
                    window.parent.frames['content'].location.href = '/dashboard.html';
                }
            });
        });

    });
</script>
```

Figura 59 Función de la solicitud de inicio de sesión

El procesamiento de esta solicitud en el servidor se lleva a cabo en la ruta "/verificar-usuario", que devuelve "Acceso permitido" si el inicio de sesión es exitoso,

redirigiendo al usuario a la página principal de la aplicación; de lo contrario, devuelve "Acceso denegado". El desarrollo de esta función se muestra en la Figura 60:

```
276 # process login form
277 @app.route('/verificar-usuario', methods=['POST'])
278 def verificar_usuario():
279     username = request.form.get('username')
280     password = request.form.get('password')
281 
282     result = collection.find_one({'username': username, 'password': password})
283     if result:
284         log("User {} logged in correctly to the web app".format(username), level="INFO")
285         session['logged_in'] = True
286         user_type = result.get('type')
287         session['type'] = user_type
288         get_user_type()
289         timesession = int(result['sessiontime'])
290 
291 #you have to login again after session time is expired, for security reasons
292         app.config['PERMANENT_SESSION_LIFETIME'] = timedelta(minutes=timesession)
293 
294         return 'Access allowed'
295 
296     else:
297         session['logged_in'] = False
298         return 'Access refused'
```

Figura 60 Función para verificar el inicio de sesión

Para Facilitar la compresión de la funcionalidad de inicio de sesión, se proporciona el flujograma siguiente:

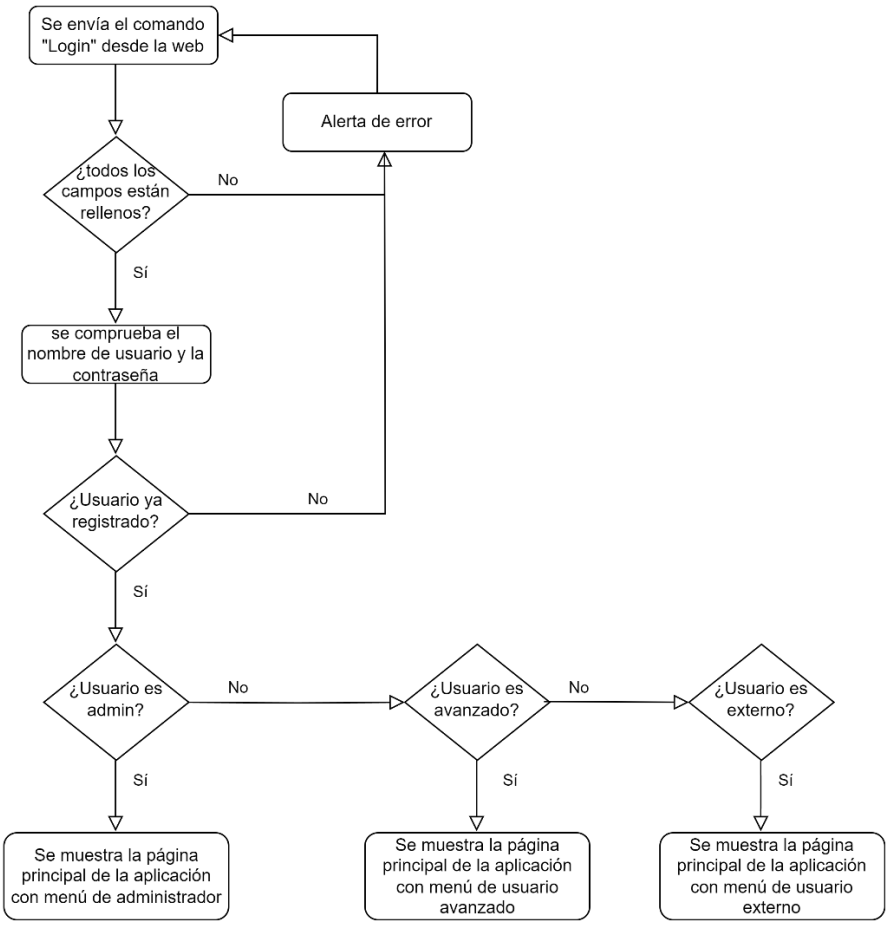


Figura 61 Flujograma del inicio de sesión

Una vez que se ha iniciado sesión correctamente, se registra el tiempo de la sesión del usuario, tras el cual será necesario iniciar sesión nuevamente, por motivos de seguridad.

Asimismo, se verifica el tipo de usuario, ya sea administrador, usuario avanzado o usuario externo, y se le presentan las opciones de menú predefinidas correspondientes. Esta comprobación se hace leyendo el campo "type" del usuario conectado, como se puede ver en la línea 297 de la Figura 62

```
293 #get the type of the connected user
294 @app.route('/get_user_type')
295 def get_user_type():
296     if 'type' in session:
297         return session['type']
298     else:
299     # if not logged in, deault user is guest
300         return 'guest'
```

Figura 62 Función para devolver el tipo de usuario

En la página de Menú, se hace la comprobación del tipo de usuario recibiendo la respuesta del servidor, y el código implementado en la Figura 63 permite mostrar las opciones según el tipo de usuario:

```
124 <script>
125     $(document).ready(function() {
126         // request to get the user type:
127         $.ajax({
128             url: '/get_user_type',
129             type: 'GET',
130             success: function(userType) {
131                 // show the user type: helps debugging the code
132                 console.log('User type:', userType);
133                 // hide all menu options
134                 $(".menu-link").hide();
135                 // Show specific options for every user:
136                 if (userType === 'admin') {
137                     // show all options for admin
138                     $(".menu-link").show();
139 
140                 } else if (userType === 'advanced') {
141                     // show options for advanced user
142                    $(".menu-link").show();
143                    $(".usermanagement-option, .security-option, .recordings-option").hide();
144 
145                 } else if (userType === 'external') {
146                     //  // show options for external user
147                     $(".menu-link").show();
148                     $(".usermanagement-option, .streaming-option, .security-option, .recordings-option , .monitoring-option").hide();
149                 }
150             },
151             error: function() {
152                 // for debugging
153                 console.log('Error obtaining user type.');
154             }
155         });
156     });
157 </script>
```

Figura 63 Función para mostrar el menú de opciones

El usuario administrador cuenta con la capacidad de agregar y eliminar usuarios de la base de datos.

En la página de gestión de usuarios, se muestra un formulario para agregar un nuevo usuario, así como una lista que incluye el nombre y el tipo de los usuarios ya registrados.

Con el fin de obtener la lista de usuarios, se ha desarrollado la clase representada en la Figura 64, la cual modela un usuario mediante la inicialización de sus atributos, y los convierte en un diccionario que resulta útil para almacenar y manipular los datos.

```
415 #MANAGE USERS
416 class User:
417     def __init__(self, name, password, username, type, pushbullet_key, pushbullet_id, sessiontime):
418         self.name = name
419         self.password = password
420         self.username = username
421         self.type= type
422         self.pushbullet_key=pushbullet_key
423         self.pushbullet_id=pushbullet_id
424         self.sessiontime=sessiontime
425 
426 
427     def toDBCollection(self):
428         return{
429             'name': self.name,
430             'password': self.password,
431             'username': self.username,
432             'type': self.type,
433             'pushbullet_key': self.pushbullet_key,
434             'pushbullet_id': self.pushbullet_id,
435             'sessiontime': self.sessiontime
436         }
```

Figura 64 Clase usuario

El código se las líneas 429 y 430 de la Figura 65 permite comprobar que la sesión sigue activa, esta funcionalidad se detallará en el apartado 4.2.1. Para recuperar todos los usuarios almacenados en la base de datos se utiliza el método find() de la colección de usuarios, definido en la línea 434.

Estos usuarios se asignan a la variable “usersReceived”, luego, tanto la lista de usuarios recuperados como una variable de resultado de operación se envían a la plantilla 'usermanagement.html' para su representación en la interfaz de usuario.

```
426 # get users
427 @app.route('/usermanagement.html' ,methods=['GET'])
428 def home():
429     if not session.get('logged_in'):
430         return render_template('login.html')
431     operation_result = None
432     users = db['users']
433     # get all users from the DB
434     usersReceived = users.find()
435     #  Pass the list of users to the template
436     return render_template('usermanagement.html', users=usersReceived,operation_result=operation_result)
```

Figura 65 Función para mostrar usuarios

Para eliminar a un usuario, se ha incorporado un botón "Eliminar" asociado a cada usuario en la página web, el cual envía una solicitud de tipo Delete al servidor, como se muestra en la Figura 66.

Figura 66 Formulario para borrar usuarios

```
<script>
    function confirmDelete(username) {
        if (confirm("⚠ This action cannot be undone. Are you sure you want to delete this user? ⚠")) {
            // If the user confirms, redirect to the delete route
            window.location.href = "/delete/" + username;
        }
    }
</script>
```

Figura 67 Solicitud para borrar usuario

El servidor gestiona esta solicitud en la ruta variable "/delete/string:username", lo que permite eliminar al usuario cuyo nombre está especificado en la ruta.

```
# Method Delete
@app.route('/delete/<string:username>')
def delete(username):
    users = db['users']
    users.delete_one({'username': username})
    usersReceived = users.find()
    operation_result = True

    return render_template('usermanagement.html',users=usersReceived, operation_result=operation_result)
```

Figura 68 Función para borrar usuario

Para crear un nuevo usuario, se proporciona el formulario presentado en la Figura 69:

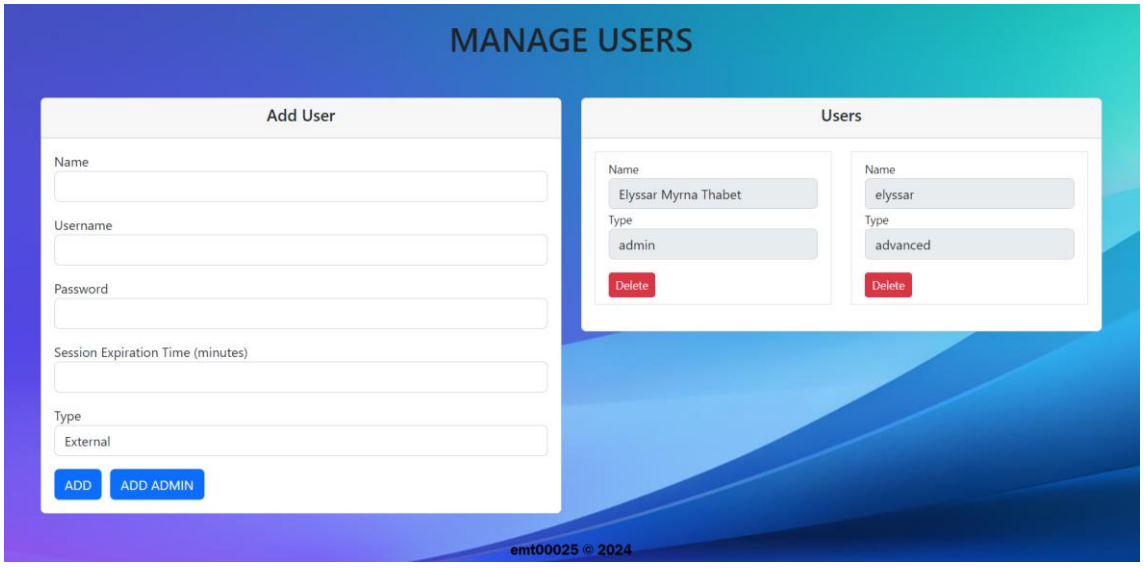


Figura 69 Formulario para crear nuevo usuario

En el lado del servidor, se reciben los datos enviados a través del formulario. Se verifica si el nombre de usuario ya existe en la base de datos. En caso afirmativo, se emite una alerta y el usuario no se registra correctamente.

Si los datos del usuario son correctos y el nombre de usuario no está previamente utilizado, se procede a crear el usuario de manera exitosa. Finalmente, se registra esta acción en los registros de la aplicación para llevar correctamente el seguimiento del sistema. Además, se muestra una alerta indicando que la operación se ha realizado correctamente.

A continuación, se presenta la función que permite crear un nuevo usuario. Esta función corresponde a los usuarios de tipo externo y avanzado.

```python
# Method for creating new user
@app.route('/users', methods=['POST'])
def addUser():
    users = db['users']
    name = request.form['name']
    password = request.form['password']
    username = request.form['username']
    sessiontime= request.form ['sessiontime']
    user_type = request.form['type']

    # Performs a query to verify whether the user already exists
    existing_user = users.find_one({'username': username})

    if existing_user:
        # User already exists, displays an error alert
        usersReceived = users.find()
        operation_result = False
        return render_template('usermanagement.html', users=usersReceived, operation_result=operation_result)

    pushbullet_key = None
    pushbullet_id = None

    if name and username and password and user_type and sessiontime:
        user = User(name,password,username , user_type, pushbullet_key,pushbullet_id, sessiontime)
        users.insert_one(user.toDBCollection())
        operation_result = True
        usersReceived = users.find()
        log("User {} added successfully".format(username), level="INFO")
        return render_template('usermanagement.html', users=usersReceived, operation_result=operation_result)


    else:
        usersReceived = users.find()
        operation_result = False
        return render_template('usermanagement.html', users=usersReceived, operation_result=operation_result)
```

Figura 70 Función para crear nuevo usuario

En el caso de que el usuario sea de tipo Administrador del sistema, aparecerán nuevos campos en el formulario de registro, dichos campos son “clave de pushbullet” y el “id del dispositivo de pushbullet”.

Una vez todos los campos están completados correctamente, se pulsará el botón “Perform Facial Recognition” para que se registra el rostro del usuario y se vuelva a entrenar el modelo.

```python
@app.route('/perform_facial_recognition', methods=['POST'])
def perform_facial_recognition():

    users = db['users']
    name = request.json.get('name')
    password = request.json.get('password')
    username = request.json.get('username')
    user_type = request.json.get('type')
    sessiontime = request.json.get('sessiontime')
    pushbullet_key = request.json.get('pushbullet_key')
    pushbullet_id = request.json.get('pushbullet_id')


    # check if the user already exists
    existing_user = users.find_one({'username': username})

    if existing_user:
        # existing user: error alert

       return jsonify({'success': False, 'message': 'El usuario ya existe'})

    if not pushbullet_id and not pushbullet_key:
        # error if pushbullet id wasn't introduced
       return jsonify({'success': False, 'message': 'missing Pushbullet info'})

    if name and username and password and user_type and pushbullet_key and pushbullet_id:
        subprocess.run(['python3', '/home/pi/Desktop/elyssar_code/my_project/rec_facial/capturar.py', username])
        subprocess.run(['python3', '/home/pi/Desktop/elyssar_code/my_project/rec_facial/entreno.py'])
        user = User(name, password,username , user_type, pushbullet_key, pushbullet_id, sessiontime)
        users.insert_one(user.toDBCollection())
        log("Admin {} added correctly".format(username), level="INFO")

        return jsonify({'success': True, 'message': 'Facial recognition and successful user registration'})
```

Figura 71 Función para reconocimiento facial del usuario Admin

Al presionar el botón "Perform Facial Recognition", se ejecutará la función de captura de rostros y, posteriormente, se realizará el entrenamiento del modelo.

Para poder capturar imágenes, el código configura un sistema de captura de imágenes faciales para un usuario específico. Obtiene el nombre de usuario desde los argumentos que se envían en el formulario de registro, define una ruta base donde se almacenarán las imágenes, creando una subcarpeta específica para el usuario si no existe. Se inicia la captura de video desde la cámara y carga el clasificador Haar Cascade para la detección de rostros. Además, inicializa un contador para llevar el control del número de imágenes capturadas.

```python
username = sys.argv[1]
dataPath = '/home/pi/Desktop/elyssar_code/my_project/rec_facial/test'
personPath = os.path.join(dataPath, username)

if not os.path.exists(personPath):
    print('Carpeta creada: ',personPath)
    os.makedirs(personPath)

cap = cv2.VideoCapture(0)

faceClassif = cv2.CascadeClassifier(cv2.data.haarcascades+'haarcascade_frontalface_default.xml')
count = 0
```

Figura 72 Definición de parámetros para capturar imágenes

El código de la Figura 73 captura imágenes faciales en tiempo real desde la PiCamera. En un bucle continuo, lee cada fotograma de la cámara, lo redimensiona, lo convierte a escala de grises y copia el fotograma original. Utiliza un clasificador Haar Cascade para detectar rostros en el fotograma en escala de grises.

Para cada rostro detectado, dibuja un rectángulo alrededor del rostro en el fotograma original, extrae la región del rostro, la redimensiona y guarda la imagen del rostro en una

carpeta específica. Muestra el fotograma procesado en una ventana y finaliza la captura cuando se detectan 20 rostros. Finalmente, libera la cámara.

```
while True:

    ret, frame = cap.read()
    if ret == False:
        break
    frame =  imutils.resize(frame, width=640)
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    auxFrame = frame.copy()

    faces = faceClassif.detectMultiScale(gray,1.3,5)

    for (x,y,w,h) in faces:
        cv2.rectangle(frame, (x,y),(x+w,y+h),(0,255,0),2)
        rostro = auxFrame[y:y+h,x:x+w]
        rostro = cv2.resize(rostro,(150,150),interpolation=cv2.INTER_CUBIC)
        cv2.imwrite(personPath + '/rostro_{}.jpg'.format(count),rostro)
        count = count + 1
    cv2.imshow('frame',frame)

    k =  cv2.waitKey(1)
    if k == 27 or count >= 20:
        break

cap.release()
cv2.destroyAllWindows()
```

Figura 73 Lógica para capturar imágenes

Luego, el código de la Figura 74 se encarga de preparar y entrenar un modelo de reconocimiento facial utilizando imágenes almacenadas en los directorios específicos.

Primero, define la ruta base donde están almacenadas las imágenes y lista las carpetas de personas en esa ruta, imprimiéndolas en pantalla por motivos de depuración de código. Luego, inicializa listas para almacenar las etiquetas y los datos faciales, así como un contador de etiquetas. Para cada carpeta de persona, lee las imágenes de rostros, agregando las etiquetas correspondientes y almacenando los datos de las imágenes en escala de grises. Después, crea un reconocedor facial LBPH y entrena el modelo con los datos recopilados. Finalmente, guarda el modelo entrenado en un archivo XML y notifica al usuario que el modelo ha sido almacenado.

```python
dataPath = '/home/pi/Desktop/elyssar_code/my_project/rec_facial/test'
peopleList = os.listdir(dataPath)
print('Lista de personas: ', peopleList)

labels = []
facesData = []
label = 0

for nameDir in peopleList:
    personPath = dataPath + '/' + nameDir
    print('Leyendo las imágenes')

    for fileName in os.listdir(personPath):
        print('Rostros: ', nameDir + '/' + fileName)
        labels.append(label)
        facesData.append(cv2.imread(personPath+'/'+fileName,0))
    label = label + 1

face_recognizer = cv2.face.LBPHFaceRecognizer_create()

# Entrenando el reconocedor de rostros
print("Entrenando...")
face_recognizer.train(facesData, np.array(labels))

face_recognizer.write('modeloLBPHFace.xml')
print("Modelo almacenado...")
```

Figura 74 Código de entrenamiento del modelo

Para llevar a cabo esta acción, es necesario registrar al nuevo administrador desde la casa, porque que la identificación se realiza a través de la cámara ubicada en la puerta principal. Esta misma cámara facilita el control de acceso mediante reconocimiento facial, activado al presionar el timbre de la vivienda, como se detalla en el apartado 4.2.1

```
142         function toggleAdminFields() {
143     var adminFields = document.getElementById("admin-fields");
144     var nameField = document.getElementById("name");
145     var usernameField = document.getElementById("username");
146     var passwordField = document.getElementById("password");
147     var sessionField = document.getElementById("sessiontime");
148     var typeField = document.getElementById("type");
149 
150 
151     if (adminFields.style.display === "none") {
152 
153 
154         // Verificar que los campos no estén vacíos y que el tipo sea Admin antes de deshabilitarlos
155       if (nameField.value.trim() &&
156           usernameField.value.trim() &&
157           passwordField.value.trim() &&
158           sessionField.value.trim() &&
159           typeField.value.trim() === "admin") {
160               adminFields.style.display = "block";
161               document.getElementById("addButton").disabled = true;
162               document.getElementById("addAdminButton").disabled = true;
163               nameField.disabled = true;
164               usernameField.disabled = true;
165               passwordField.disabled = true;
166               sessionField.disabled = true;
167               typeField.disabled = true;
168       } else {
169           // Aquí puedes manejar el caso en que uno o más campos están vacíos o el tipo no es Admin
170           alert("Please complete all fields and select 'Admin' in the 'Type' field.");
171       }
172     } else {
173         adminFields.style.display = "none";
174     }
```

Figura 75 Comprobar datos de nuevo usuario de tipo administrador

4.2.11 Registro

Para registrar cada acción realizada en el sistema y proporcionar una trazabilidad completa, MongoDB aloja una colección dedicada a almacenar logs de actividades. Cada vez que un usuario realiza una acción, ya sea el encendido de luces, la apertura de puertas o cualquier otra operación, se genera un registro que captura los detalles cruciales, como el usuario responsable, la acción específica y la marca de tiempo correspondiente. Estos logs no solo son esenciales para la auditoría y la identificación de problemas, sino que también se utilizan para satisfacer las solicitudes de usuarios que desean revisar su historial de actividades.

Para garantizar una trazabilidad completa de las acciones realizadas en el sistema domótico, se ha implementado un sólido sistema de registros que documenta cada interacción y evento. Estos registros no solo proporcionan una huella detallada de las actividades realizadas, sino que también facilitan la identificación de posibles problemas y mejoras en el rendimiento del sistema.

Cada vez que se realiza una acción, ya sea el encendido de luces, el control de puertas, o cualquier otra funcionalidad del sistema, se genera un registro que captura información crucial. Estos registros son esenciales para el monitoreo del sistema y permiten una revisión histórica de las actividades y estarán almacenados en un fichero

local. Por motivo de seguridad, estarán almacenados también en una base de datos, accesibles desde la página web.

A continuación, se presenta una selección de logs de acciones representativas para ilustrar la estructura y contenido de los registros generados:

Si el timbre fue tocado:

- [FECHA HORA] Front door bell activated

Si la persona tiene acceso:

- [FECHA HORA] Front door open for <user_name>.

Si la persona no tiene acceso:

- [FECHA HORA] Someone tried to access from the front door. Unsuccessful facial recognition.

Si se abre la puerta de la cochera:

- [FECHA HORA] Garage door open

Si se cierra la puerta de la cochera:

- [FECHA HORA] Garage door closed

Cuando un usuario inicia sesión:

- [FECHA HORA] User <user_name> logged in correctly to the web app

Cuando se añade un nuevo usuario:

- [FECHA HORA] User *<user_name>* added correctly

Cuando se añade un nuevo administrador:

- [FECHA HORA] Admin *<user_name>* added correctly

Cuando se borra un usuario:

- [FECHA HORA] User *<user_name>* deleted correctly

Cuando se inicia una simulación de presencia:

- [FECHA HORA] Security simulation started : volume = *<volume>*, frq *<frequency>* , t *<duration>* seconds, intercycle <Interval> seconds.

Cuando se inicia la transmisión en directo:

- [FECHA HORA] Live streaming from Main Camera started

Cuando se detiene la transmisión en directo:

- [FECHA HORA] Live streaming from Main Camera stopped

Cuando se detecta movimiento en la puerta principal de la vivienda o en la puerta de la cochera:

- [FECHA HORA] Motion detected at front door. Image captured.

Cuando se borran los registros del almacenamiento local:

- [FECHA HORA] Local storage cleared.

Antes de profundizar en el funcionamiento detallado del sistema, es crucial entender el proceso de registro de eventos y mensajes. Esto se evidencia en la función que se muestra en la Figura 76, la cual se encarga de almacenar registros en una ubicación específica del microcontrolador.

```
58 #functionality to create logs
59 def log(logtxt,level):
60     log_path = "/home/pi/Desktop/elyssar_code/my_project/webapp"
61     y = datetime.datetime.now().strftime("%Y")
62     ypath = os.path.join(log_path, y)
63     m = datetime.datetime.now().strftime("%m")
64     mpath = os.path.join(ypath, m)
65     d = datetime.datetime.now().strftime("%d")
66     dpath = os.path.join(mpath, f"{d}.txt")
67 
68     if not os.path.exists(ypath):
69         os.makedirs(ypath)
70 
71     if not os.path.exists(mpath):
72         os.makedirs(mpath)
73 
74     timestamp = datetime.datetime.now().strftime('%Y-%m-%d %H:%M:%S')
75     log_message = "[{}] LOGGER:: {}:: {}:: {}".format(timestamp, level, time.strftime('%Y-%m-%d %H:%M:%S'), logtxt)
76 
77 
78         # Create file my_logs.txt if it doesn't exist
79     if not os.path.exists('my_logs.txt'):
80         with open('my_logs.txt', 'w') as file:
81             file.write(log_message + "\n")
82 
83     with open('my_logs.txt', 'a') as log_file:
84         log_file.write(log_message + "\n")
85 
86     with open(dpath, "a") as log_file:
87         log_file.write(log_message + "\n")
88 
89 
90 
91     # Save logs in the database
92     timestamp = datetime.datetime.now().strftime("%H:%M:%S")
93     log_entry = {
94         "timestamp": timestamp,
95         "level": level,
96         "log_time": time.strftime('%Y-%m-%d %H:%M:%S'),
97         "message": logtxt
98     }
99     logs_collection.insert_one(log_entry)
100
```

Figura 76 Función para almacenar registros

Los archivos de registro se organizan en una estructura de carpetas según el año, mes y día en que se registran los mensajes, en la dirección especificada en la línea 60. Además, los mensajes de registro también se escriben en un archivo principal llamado my_logs.txt en la carpeta del proyecto.

La linea 99 de la Figura 76 es la función que permite registrar los logs en la base datos, cuya configuración se describe en la Figura 58 .

Con el fin de poder leer los registros desde el fichero my_logs.txt y mostrarlos en la página web, se ha creado un botón “Update” en la interfaz de usuario, una vez pulsado este botón se envia la solitud descrita en la al servidor.

```
<script>
    function actualizarLogs() {
        // empty the logs list
        const logList = document.getElementById('log-list');
        logList.innerHTML = "";

        // Request the updated logs from the server
        $.get('/get-logs', function(data) {
            data.forEach(log => {
                const logItem = document.createElement('li');
                logItem.textContent = log;
                logList.appendChild(logItem);
            });
        });
    }
</script>
```

Figura 77 Solicitud para leer registros

Dicha solicitud se maneja en el servidor según lo especificado en la siguiente ilustración y devuelve la lista de los registros, donde el primer elemento de la lista es el más reciente.

```
#read the saved logs in local storage
@app.route('/get-logs')
def get_logs():
    with open('my_logs.txt', 'r') as log_file:
        logs = log_file.readlines()
        logs.reverse()
    return jsonify(logs)

```

Figura 78 Función para devolver registros

Existe también la opción de borrar dichos registros, para lograrlo se ha creado el botón “Clear logs” que, una vez pulsado, se envía una solicitud definida en la Figura 79 al servidor para borrar la información guardada en el fichero my_logs.txt.

```
<script>

    function borrarLogs() {
    const confirmDelete = window.confirm("⚠ This action cannot be undone. Are you sure you want to delete all logs from local storage? ⚠");

    if (confirmDelete) {
        $.get('/clear-logs', function(response) {
            if (response === "Logs cleared successfully") {
                const logList = document.getElementById('log-list');
                logList.innerHTML = "";
            }
        });
    }
}

</script>
```

Figura 79 Solicitud para borrar registros

El lado servidor maneja esta solicitud según muestra la Figura 80 y una vez confirmada la acción por parte del usuario, se procederá a eliminar el contenido del fichero en cuestión.

```python
#delete logs from local storage
@app.route('/clear-logs')
def clear_logs():
    with open('my_logs.txt', 'w') as log_file:
        log_file.truncate(0)
    return "Logs cleared successfully"
```

Figura 80 Función para borrar registros del almacenamiento local

Se utiliza el cliente MongoClient de la biblioteca PyMongo para conectarse a la base de datos como se muestra en el apartado 4.2.10

```python
# Configurate the connexion to the database
client = MongoClient('mongodb+srv://emt00025:3TXTOQd1gdLJl6FW@smarthome.fk7eyc2.mongodb.net/')
db = client['users']
collection = db['users']
logs_collection = db['logs']
```

Figura 81 Configuración de la base de datos

La línea 37 muestra la colección de los registros, donde la función descrita en la línea 99 de la Figura 76 inserta todos los registros generados.

Para leer los registros de la base de datos, se envía una solicitud cuando se pulsa el botón “All logs” desde la página web, lo que lleva a enviar la solicitud de la Figura 82 al servidor.

```javascript
<script>
    function databaseLogs(database) {
    const logList = document.getElementById('log-list');
    logList.innerHTML = "";
    // Request the updated logs from the server for the selected database
    $.get(`/get-logs-db?database=${database}`, function(data) {
        data.forEach(log => {
            const logItem = document.createElement('li');
            logItem.textContent = log;
            logList.appendChild(logItem);
        });
    });
}
</script>
```

Figura 82 Solicitud de registros de la base de datos

El servidor maneja la solicitud y devuelve la lista de registros de la base de datos al usuario, ordenados cronológicamente.

```
103 # get logs from the database
104 @app.route('/get-logs-db')
105 def get_logsdb():
106     logs = logs_collection.find()
107     logs_text = [log_entry['message'] for log_entry in logs]
108     logs_text.reverse()
109     return jsonify(logs_text)
```

Figura 83 Función de manejar registros de la base de datos

4.2.12 Creación de la maqueta

En el desarrollo de la maqueta de la casa domótica prototipo, se llevó a cabo una detallada construcción que comprende dos plantas. La planta baja, conocida como planta 0, incluye una entrada, un salón-comedor, una cocina y un garaje, todos rodeados por un jardín que añade un toque de naturaleza al diseño.

La primera planta, o planta 1, comprende un baño, dos habitaciones y una terraza equipada con una piscina, proporcionando un espacio de relajación y confort.

Para la fabricación de esta maqueta, se utilizaron materiales sostenibles que no causan daño al medio ambiente, incluyendo elementos impresos en 3D y madera, los cuales fueron fundamentales para la creación de detalles precisos como las ventanas y la escalera interna. Además, el tejado de la casa está equipado con una placa solar, subrayando el compromiso con la sostenibilidad y el uso de energías renovables. Complementando esta tecnología, se instaló una pantalla que permite el control integral de todas las funciones domóticas de la vivienda, facilitando la gestión de los diferentes sistemas de manera eficiente y centralizada. Esta maqueta no solo representa una estructura arquitectónica moderna, sino también un ejemplo de la integración de tecnologías avanzadas en el hogar y el uso responsable de materiales.

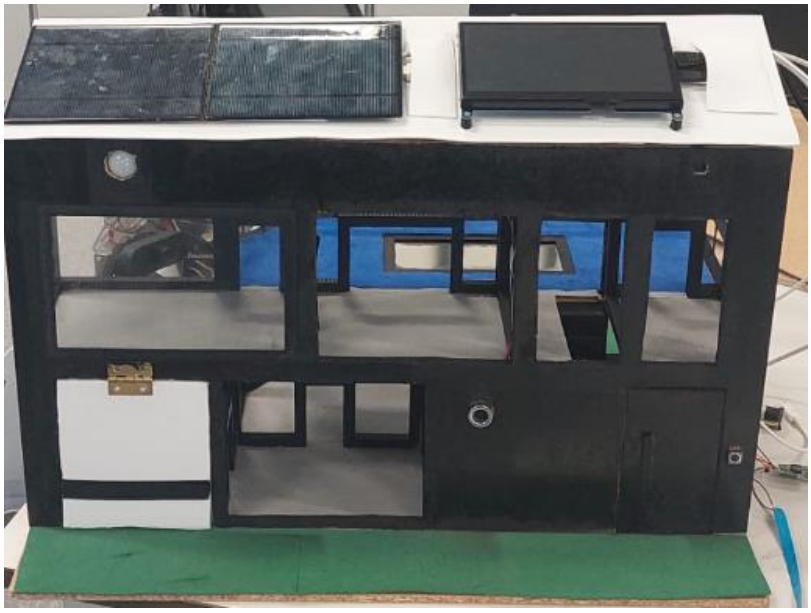


Figura 84 Vista principal de la vivienda

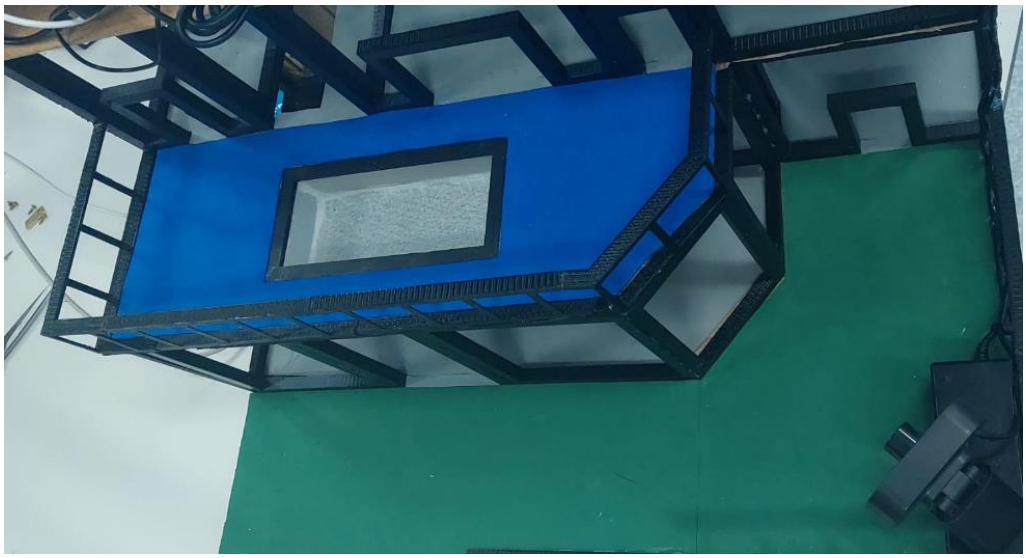


Figura 85 Vista de la terraza de la vivienda

4.3 Presupuesto

En esta parte se presenta en presupuesto para la realización de la maqueta del sistema domótico de este proyecto.

4.3.1 Materiales

El coste total de materiales utilizados tanto para el desarrollo del diseño como la implementación del prototipo se muestran en la Tabla 1

Ud.	Concepto	P.Unitario	Subtotal
1	Cables DuPont Macho - Macho (20 cm / 60 unidades)	1,50 €	1,50 €
1	Cables DuPont Hembra - Macho (20 cm / 60 unidades)	1,50 €	1,50 €
1	Cables DuPont Macho - Hembra (20 cm / 40 unidades)	1,50 €	1,50 €
1	Módulo de sesnor PIR HC-SR501	1,95 €	1,95 €
1	Módulo 4 relés 5V	4,25 €	4,25 €
1	Memoria Miscro SD 32 GB	8,90 €	8,90 €
1	Raspberry pi 4 Model B (4 GB RAM)	82,50 €	82,50 €
1	Fuente de alimentación Raspberry Pi 4, USB-C, 5,1V 3A	8,20 €	8,20 €
1	DC 12V Cerradura magnética puerta eléctrica bloqueo de solenoide	14,95 €	14,95 €
2	Cámara web PC HD 1080p	18,15 €	36,30 €
1	Trasnformador 240V a 12 V - 5A	15,75 €	15,75 €
8	LED	0,30 €	2,40 €
1	Cable Micro HDMI a HDMI	7,85 €	7,85 €
1	Teclado y ratón inalámbricos	17,90 €	17,90 €
1	Pulsador	0,35 €	0,35 €
1	Interruptor	1,15 €	1,15 €
1	Placa de prototipo 16x5 cm (830 puntos)	3,50 €	3,50 €
1	Cámara Raspberry Pi NoIR v1,3 (5MP 1080p)	22,50 €	22,50 €
1	Cable Ribbon 15cm	2,50 €	2,50 €
1	Tira de luces LED	18,00 €	18,00 €
1	Pantalla 7 pulgadas	42,00 €	42,00 €
1	Tablero agromerado	5,00 €	5,00 €
		Total (sin I.V.A)	300,45 €

Tabla 1 Presupuesto de materiales

4.3.2 Mano de obra

El coste total de mano de obra utilizado tanto para el desarrollo del diseño como la implementación del prototipo se muestran a continuación en la Tabla 2

Horas	Concepto	Precio/Hora	Subtotal
6	Especificación de los requisitos	55,00 €	330,00 €
18	Diseño	55,00 €	990,00 €
36	Implementación	55,00 €	1.980,00 €
8	Pruebas	55,00 €	440,00 €
		Total (sin I.V.A)	3.740,00 €

Tabla 2 Presupuesto de mano de obra

4.3.3 Resumen

El resumen del presupuesto total se muestra en la Tabla 3.

Concepto	Importe
Materiales	300,45 €
Mano de obra	3.740,00 €
Total sin I.V.A	4.040,45 €
I.V.A (21%)	848,49 €
Total	4.040,45 €

Tabla 3 Presupuesto total

Asciende el importe total a **CUATRO MIL CUARENTA EUROS CON CUARENTA Y CINCO CÉNTIMOS.**

5 Resultados y Discusión

El proyecto de desarrollo del sistema domótico a escala con acceso remoto ha culminado con éxito en la creación de un prototipo funcional de un sistema de seguridad con elementos domóticos, junto una aplicación web para la gestión y el control remotos del sistema.

Tras evaluar y seleccionar cuidadosamente las tecnologías y componentes adecuados, se ha logrado implementar cada elemento del sistema de manera que funcione de manera simultánea y sin conflictos.

La construcción de la maqueta proporciona una representación física de cómo se instalaría el sistema en una vivienda real, facilitando las pruebas y demostraciones. Reconociendo que este prototipo es solo el punto de partida, se vislumbra la necesidad de añadir más actuadores y sensores en una implementación real, así como una revisión completa del esquema electrónico para integrar los diversos elementos domóticos disponibles en el mercado.

En definitiva, el proyecto ha sentado las bases para un sistema de seguridad domótico versátil y adaptable a las necesidades futuras del usuario.

5.1 Objetivos ODS

En el marco del proyecto, se ha prestado especial atención a los Objetivos de Desarrollo Sostenible (ODS) de las Naciones Unidas [40].

Estos objetivos han servido como guía para orientar las decisiones de diseño y desarrollo, con el objetivo de contribuir al cumplimiento de metas globales en áreas clave como energía limpia y asequible (ODS 7), Industria, innovación e infraestructura (ODS 9), ciudades y comunidades sostenibles (ODS 11) y acción por el clima (ODS 13). La utilización de la luz solar como fuente de energía renovable ejemplifica el compromiso del proyecto con la promoción de prácticas sostenibles y la reducción de la huella ambiental [41].

Además, se ha considerado cómo el sistema de seguridad domótico puede fomentar la eficiencia energética y el uso responsable de los recursos, contribuyendo así a un futuro más sostenible [42].

6 Conclusiones y propuestas de mejora

En base a los hallazgos obtenidos, es evidente que las tecnologías IoT desempeñan un papel fundamental en la ampliación de la flexibilidad y versatilidad de los sistemas domóticos, facilitando su implementación en entornos prácticos.

La elección de Python como lenguaje de programación para el prototipo del sistema de seguridad resalta la potencia y accesibilidad inherentes a este lenguaje. La amplia gama de bibliotecas disponibles, como OpenCV para la visión artificial y scikit-learn para el aprendizaje automático, ilustra la capacidad de Python para integrar herramientas avanzadas de forma sencilla. Esta facilidad de implementación se ve respaldada por comunidades activas y comprometidas, lo que agiliza la búsqueda de recursos e información pertinente, contribuyendo así a la eficacia y eficiencia del desarrollo tecnológico en este ámbito.

Tras evaluar detenidamente el sistema de seguridad domótico desarrollado se han identificado áreas de oportunidad que podrían mejorar significativamente su rendimiento y funcionalidad. En esta introducción, se presentarán las principales propuestas de mejora, destacando su importancia y el impacto que podrían tener en la experiencia del usuario y en la eficacia del sistema de seguridad domótico.

- Gestión de sesiones de usuarios: Se sugiere mejorar el sistema para evitar que el inicio de sesión de un usuario cierre la sesión de otro, permitiendo así la simultaneidad de múltiples usuarios conectados.
- Reconocimiento desde varias cámaras: Se propone ampliar la capacidad de reconocimiento del sistema mediante la integración de reconocimiento facial desde varias cámaras, lo que mejoraría la precisión y la cobertura del sistema de seguridad.
- Cierre automático de la cerradura: Para aumentar la seguridad, se plantea implementar una función que cierre automáticamente la cerradura después de un tiempo predeterminado desde que se abrió mediante la web.
- Obtención del estado inicial de los elementos de la vivienda: Se considera importante obtener el estado inicial de los elementos al acceder a la página de "Control", lo que permitiría al usuario conocer la situación actual de manera inmediata y precisa.

Al implementar estas mejoras, se busca no solo incrementar la satisfacción del usuario, sino también fortalecer la integridad y la capacidad del sistema para adaptarse a diversas situaciones y entornos.

7 Referencias

[1] «Pushbullet - Your devices working better together». Accedido: 20 de abril de 2023. [En línea]. Disponible en: https://www.pushbullet.com/

[2] R. Solé, «Qué es la domótica y qué debemos tener en cuenta de esto», HardwarEsfera. Accedido: 20 de abril de 2023. [En línea]. Disponible en: https://hardwaresfera.com/articulos/que-es-domotica/

[3] zelec, «Domótica y seguridad contra robos y ocupaciones ilegales de viviendas», Zelec. Accedido: 15 de abril de 2023. [En línea]. Disponible en: https://www.zelec.es/domotica-y-seguridad-contra-robos-y-ocupaciones-ilegales-de-viviendas/

[4] «What is IoT (Internet of Things) and How Does it Work? | Definition from TechTarget», IoT Agenda. Accedido: 14 de mayo de 2023. [En línea]. Disponible en: https://www.techtarget.com/iotagenda/definition/Internet-of-Things-IoT

[5] «Internet de las Cosas (IoT): Qué es, para qué sirve y cómo funciona». Accedido: 13 de abril de 2023. [En línea]. Disponible en: https://www.cursosaula21.com/internet-de-las-cosas-iot-que-es-y-como-funciona/

[6] Redaccion, «Domótica e IoT para hacer que tu casa sea inteligente - Zemsania», Zemsania Global Group. Accedido: 10 de mayo de 2023. [En línea]. Disponible en: https://zemsaniaglobalgroup.com/domotica-e-iot-hogar/

[7] «Hogar inteligente-IOT», Domótica Sistemas. Accedido: 12 de junio de 2023. [En línea]. Disponible en: https://domoticasistemas.com/tienda/323-objetos-conectados-iot/

[8] Marketing, «▷ Tipos de aprendizaje en la Inteligencia Artificial», EDS Robotics. Accedido: 27 de abril de 2023. [En línea]. Disponible en: https://www.edsrobotics.com/blog/tipos-aprendizaje-inteligencia-artificial/

[9] «A000066-datasheet.pdf». Accedido: 7 de mayo de 2023. [En línea]. Disponible en: https://docs.arduino.cc/resources/datasheets/A000066-datasheet.pdf

[10]? «Arduino UNO R3 Características, Especificaciones», Proyecto Arduino. Accedido: 4 de junio de 2023. [En línea]. Disponible en: https://proyectoarduino.com/arduino-uno-r3/

[11] R. P. Ltd, «Buy a Raspberry Pi 4 Model B», Raspberry Pi. Accedido: 17 de junio de 2023. [En línea]. Disponible en: https://www.raspberrypi.com/products/raspberry-pi-4-model-b/

[12] «odroid-n2:odroid-n2 [ODROID Wiki]». Accedido: 17 de junio de 2023. [En línea]. Disponible en: https://wiki.odroid.com/odroid-n2/odroid-n2

[13] «¿Qué es el MQTT? - Explicación del protocolo MQTT - AWS», Amazon Web Services, Inc. Accedido: 17 de junio de 2024. [En línea]. Disponible en: https://aws.amazon.com/es/what-is/mqtt/

[14] L. Llamas, «¿Qué es MQTT? Su importancia como protocolo IoT», Luis Llamas. Accedido: 17 de junio de 2024. [En línea]. Disponible en: https://www.luisllamas.es/que-es-mqtt-su-importancia-como-protocolo-iot/

[15] «Goto IoT | Introducción a CoAP». Accedido: 25 de junio de 2023. [En línea]. Disponible en: https://www.gotoiot.com/pages/articles/coap_intro/index.html

[16] «¿Qué es el HTTP?», IONOS Digital Guide. Accedido: 17 de mayo de 2023. [En línea]. Disponible en: https://www.ionos.es/digitalguide/hosting/cuestiones-tecnicas/protocolo-http/

[17] «What is Python? Executive Summary», Python.org. Accedido: 20 de junio de 2024. [En línea]. Disponible en: https://www.python.org/doc/essays/blurb/

[18] «Flask: A simple framework for building complex web applications.»

[19] «OpenCV: Introduction». Accedido: 20 de julio de 2023. [En línea]. Disponible en: https://docs.opencv.org/4.x/d1/dfb/intro.html

[20] «PyMongo 4.7.3 documentation». Accedido: 17 de julio de 2023. [En línea]. Disponible en: https://pymongo.readthedocs.io/en/stable/

[21] «Pushbullet API». Accedido: 17 de junio de 2024. [En línea]. Disponible en: https://docs.pushbullet.com/

[22] «RPi.GPIO: A module to control Raspberry Pi GPIO channels». Accedido: 20 de diciembre de 2023. [POSIX :: Linux]. Disponible en: http://sourceforge.net/projects/raspberry-gpio-python/

[23] «threading — Paralelismo basado en hilos — documentación de Python - 3.8.19». Accedido: 20 de enero de 2024. [En línea]. Disponible en: https://docs.python.org/es/3.8/library/threading.html

[24] «subprocess — Subprocess management», Python documentation. Accedido: 20 de enero de 2024. [En línea]. Disponible en: https://docs.python.org/3/library/subprocess.html

[25] «HTML: Lenguaje de etiquetas de hipertexto | MDN». Accedido: 6 de julio de 2023. [En línea]. Disponible en: https://developer.mozilla.org/es/docs/Web/HTML

[26] «CSS: Cascading Style Sheets | MDN». Accedido: 6 de julio de 2023. [En línea]. Disponible en: https://developer.mozilla.org/en-US/docs/Web/CSS

[27] «JavaScript | MDN». Accedido: 6 de junio de 2023. [En línea]. Disponible en: https://developer.mozilla.org/en-US/docs/Web/JavaScript

[28] O. F.- openjsf.org, «jQuery». Accedido: 15 de julio de 2023. [En línea]. Disponible en: https://jquery.com/

[29] «pushbullet.py: A simple python client for pushbullet.com». Accedido: 17 de julio de 2023. [En línea]. Disponible en: https://github.com/randomchars/pushbullet.py

[30] «Funcionamiento y tecnología de un sensor de ultrasonidos», wenglor sensoric group. Accedido: 11 de junio de 2023. [En línea]. Disponible en:

https://www.wenglor.com/es/Principio-de-funcionamiento-y-tecnologia-de-un-sensor-de-ultrasonidos/s/Funktionsprinzip+und+Technologie+eines+Ultraschall-Sensors

[31] galeondev, «Qué es un sensor de movimiento PIR - Microsegur Blog Seguridad», Microsegur. Accedido: 03 de abril de 2023. [En línea]. Disponible en: https://microsegur.com/que-es-un-sensor-de-movimiento-pir/

[32] «Cerradura Eléctrica 12v Chapa Puerta Solenoide», UNIT Electronics. Accedido: 17 de junio de 2024. [En línea]. Disponible en: https://uelectronics.com/producto/cerradura-electrica-solenoide/

[33] «Welcome to Flask — Flask Documentation (3.0.x)». Accedido: 2 de junio de 2023. [En línea]. Disponible en: https://flask.palletsprojects.com/en/3.0.x/

[34] «ngrok | Unified Application Delivery Platform for Developers». Accedido: 17 de junio de 2024. [En línea]. Disponible en: https://ngrok.com/

[35] «¿Qué Es Apache Web Server? Una Mirada Básica a lo que Es y Cómo Funciona», Kinsta®. Accedido: 20 de junio de 2024. [En línea]. Disponible en: https://kinsta.com/es/base-de-conocimiento/que-es-apache/

[36] L. Llamas, «Cómo montar un servidor web Apache en Raspberry Pi», Luis Llamas. Accedido: 17 de junio de 2024. [En línea]. Disponible en: https://www.luisllamas.es/como-montar-un-servidor-web-apache-en-raspberry-pi/

[37] «pygame: Python Game Development». Accedido: 24 de junio de 2024. [MacOS, Microsoft :: Windows, POSIX, Unix]. Disponible en: https://www.pygame.org

[38] «MongoDB Documentation». Accedido: 17 de junio de 2024. [En línea]. Disponible en: https://www.mongodb.com/docs/

[39] «mongo_client – Tools for connecting to MongoDB - PyMongo 4.7.3 documentation». Accedido: 21 de junio de 2024. [En línea]. Disponible en: https://pymongo.readthedocs.io/en/stable/api/pymongo/mongo_client.html

[40] M. J. Gamez, «Objetivos y metas de desarrollo sostenible», Desarrollo Sostenible. Accedido: 21 de junio de 2024. [En línea]. Disponible en: https://www.un.org/sustainabledevelopment/es/objetivos-de-desarrollo-sostenible/

[41] M. Moran, «Energía», Desarrollo Sostenible. Accedido: 21 de junio de 2024. [En línea]. Disponible en: https://www.un.org/sustainabledevelopment/es/energy/

[42] M. Moran, «Ciudades», Desarrollo Sostenible. Accedido: 21 de junio de 2024. [En línea]. Disponible en: https://www.un.org/sustainabledevelopment/es/cities/

[43] R. P. Ltd, «Raspberry Pi OS», Raspberry Pi. Accedido: 20 de junio de 2024. [En línea]. Disponible en: https://www.raspberrypi.com/software/

8 ANEXOS

8.1 ANEXO I: MANUAL DE INSTALACIÓN

I- Preparación de la Raspberry Pi:

En la sección siguiente, se desglosa el Anexo I: Manual de Instalación, ofreciendo instrucciones detalladas para llevar a cabo la instalación del sistema de manera óptima y eficaz.

I.I. Configuración inicial:

Para implementar el proyecto, se empleó una Raspberry Pi 4B con el sistema operativo Raspbian 10 (Buster). Para configurar este sistema en la Raspberry Pi, se requiere utilizar una herramienta para formatear e instalar el sistema operativo en una tarjeta micro SD, tal como el Raspberry Pi Imager[43].

Figura 86 Raspberry Pi imager

Una vez que se haya instalado correctamente, se debe insertar la tarjeta microSD en la ranura correspondiente de la Raspberry Pi. Luego, se conecta la Raspberry Pi a la red eléctrica utilizando un adaptador de corriente compatible. A continuación, se deben conectar un teclado, un ratón y un monitor a la Raspberry Pi para poder utilizarla. Una vez que todo esté conectado correctamente, se puede encender la Raspberry Pi y esperar a que el sistema se inicie. Con estos pasos, estará listo para comenzar a utilizar la Raspberry Pi.

Para comenzar, se actualizan los paquetes instalados en la Raspberry ejecutando el comando en la terminal:

- ***sudo apt update && sudo apt upgrade -y***

El siguiente paso consiste en habilitar SSH para poder conectarse a la Raspberry-Pi con más comodidad. Además, se habilita la funcionalidad “Camera” para poder conectar los módulos de cámaras.

Finalmente se procede a instalar Python3 para permitir la ejecución de la aplicación del sistema domótico, la cual está desarrollada en Python.

La instalación se realiza mediante el comando:

- ***sudo apt install python3***

I.II. Instalación de dependencias:

Para llevar a cabo el proyecto, se necesitan varias librerías y dependencias, principalmente la librería OpenCV, la biblioteca libre de visión artificial originalmente desarrollada por Intel.

El primer paso consiste en instalar las dependencias usando el comando:

- ***sudo apt install build-essential cmake pkg-config libjpeg-dev libtiff5-dev libjasper-dev libpng-dev libavcodec-dev libavformat-dev libswscale-dev libv4l-dev libxvidcore-dev libx264-dev libfontconfig1-dev libcairo2-dev libgdk-pixbuf2.0-dev libpango1.0-dev libgtk2.0-dev libgtk-3-dev libatlas-base-dev gfortran libhdf5-dev libhdf5-103 libhdf5-dev libhdf5-serial-dev libqtgui4 libqtwebkit4 libqt4-test python3-pyqt5 python3-dev***

  Una vez instalada, pasamos a instalar Pip, que es un sistema de gestión de paquetes utilizado para instalar y administrar paquetes de software escritos en Python, mediante el comando:

- ***sudo apt update***
- ***sudo apt install python3-pip***

Además, es necesario instalar la API de PiCamera para el módulo de cámara que se va a conectar:

- ***pip install picamera***

El siguiente paso es instalar OpenCv con el comando:

- ***pip install opencv-python***

Con este proceso, OpenCV queda instalado en el entorno virtual y se encuentra listo para su uso. Además, se puede verificar la versión instalada mediante el siguiente comando:

```
pi@raspberrypi:~ $ python -c "import cv2; print(cv2.__version__)"
4.1.1
pi@raspberrypi:~ $
```

Figura 87 Verificación de OpenCV

Ahora es necesario instalar las dependencias adicionales que son requeridas para el funcionamiento óptimo del sistema. Se usará Pip, el gestor de paquetes de Python:

- ***pip install numpy imutils flask pyngrok RPi.GPIO gpiozero pygame pymongo pymongo[srv]***

También se procederá a instalar el servidor Apache, que será utilizado para alojar y servir contenido web. Este paso se realiza con el comando:

- ***sudo apt update***
- ***sudo apt install apache2***

Es posible verificar el estado de Apache mediante el siguiente comando:

- ***sudo systemctl status apache2***

I.III. Configuración de Ngrok

Una etapa crucial en el proceso de instalación es la configuración de ngrok, la cual facilita la publicación de la aplicación y, por consecuencia, el acceso al sistema domótico desde cualquier ubicación.

Para conseguirlo, es necesario registrarse en la página web de ngrok :

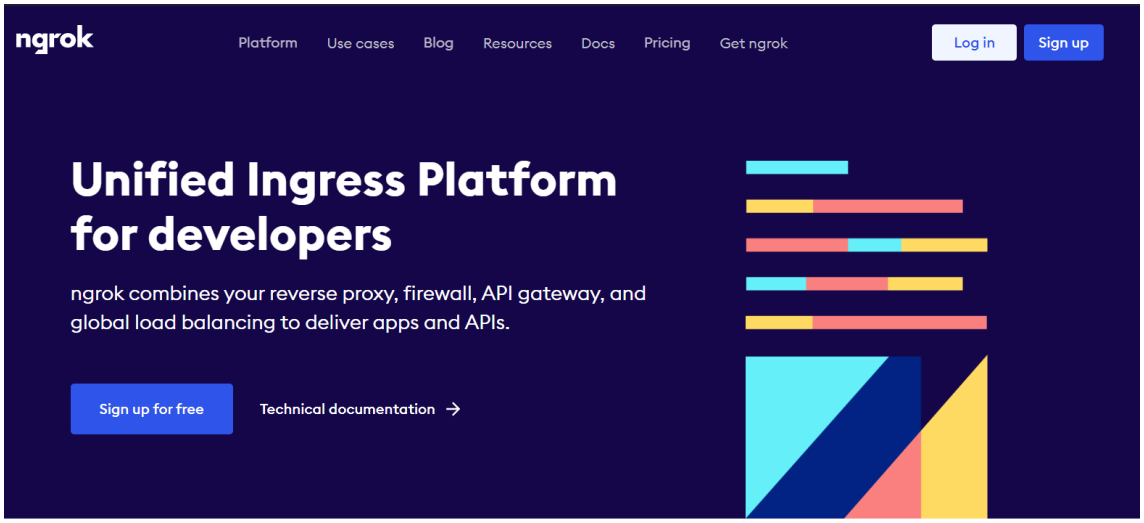


Figura 88 Página principal de ngrok

Una vez registrado, el usuario procederá a crear un Authtoken que es necesario para que ngrok genere nuevos túneles y exponga el servidor local del sistema domótico al público. Para ello, se accede al panel de control de ngrok en la página web y se selecciona "Your Authtoken" en el menú de la izquierda.

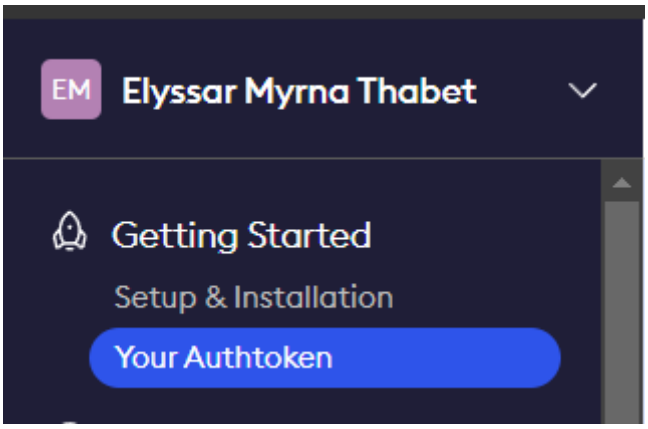


Figura 89 Authtoken de ngrok

I.IV. Configuración de Pushbullet

Otra configuración importante es la de Pushbullet, que posibilita a los usuarios recibir notificaciones del sistema en tiempo real.

En la web de Pushbullet, hay que conectarse a la cuenta personal de correo electrónico, como se muestra en la siguiente figura.

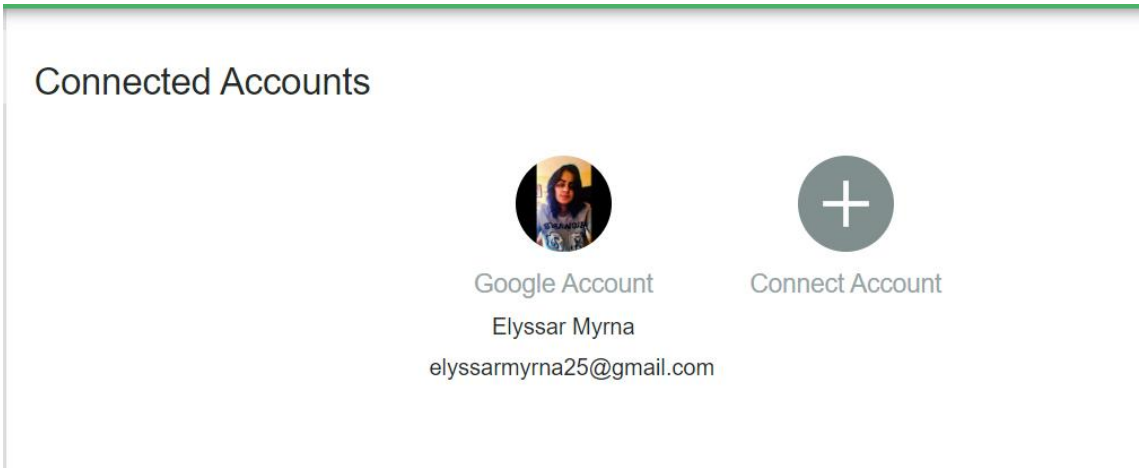


Figura 90 Configurar cuenta de usuario a Pushbullet

Una vez conectada la cuenta, el siguiente paso es instalar la aplicación Pushbullet en el dispositivo móvil de usuario e iniciar sesión en la cuenta.

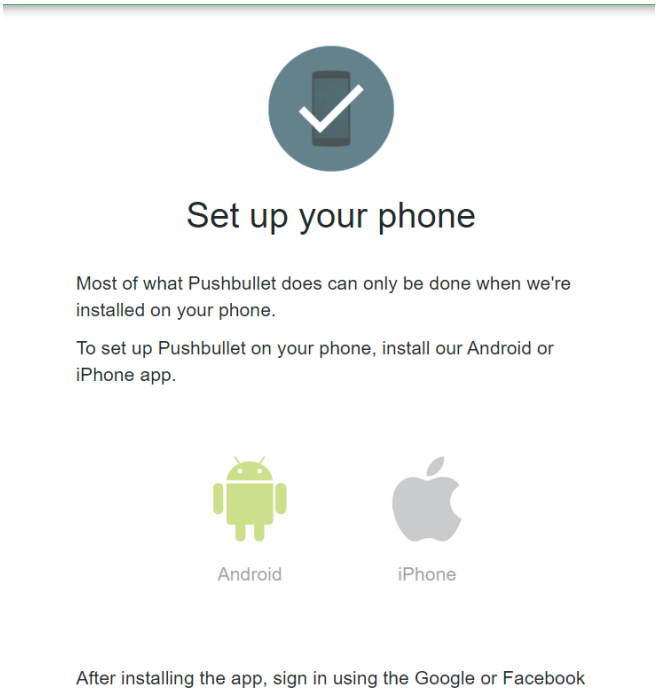


Figura 91 Configurar dispositivo móvil con Pushbullet

Finalmente, se crea un token que permite la comunicación con la aplicación del sistema domótico.

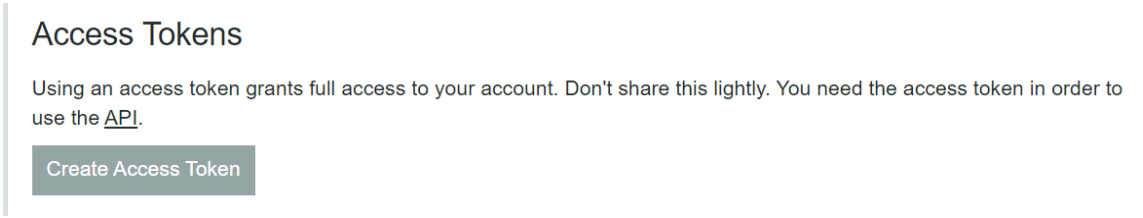


Figura 92 Creación de API token en Pushbullet

El último paso es añadir el identificador del dispositivo móvil, para poder usar la funcionalidad de las notificaciones en tiempo real.

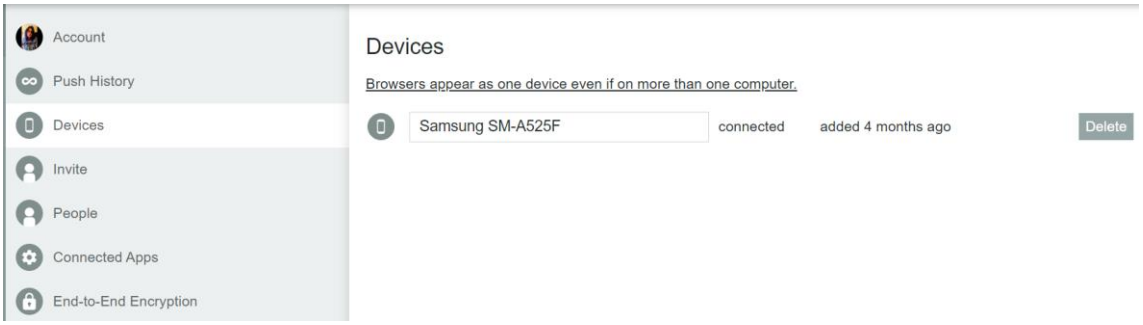


Figura 93 Configurar dispositivos de Pushbullet

Se necesitarán ambos parámetros, tanto el ID del dispositivo móvil como el token de API, para poder recibir las notificaciones.

I.V. Inicio automático

Para lograr que la aplicación del sistema domótico se ejecute al iniciar la Raspberry Pi, se ha desarrollado un script llamado “start_app.sh” para tal propósito.

```
#!/bin/bash
# Ir al directorio de la aplicacion
cd /home/pi/Desktop/elyssar_code/my_project/webapp/
# Ejecutar la aplicacion
PYTHONPATH=/home/pi/.local/lib/python3.7/site-packages /usr/bin/python3 /home/pi/Desktop/elyssar_code/my_project/webapp/myappcontrol.py &
# Esperar un momento
sleep 10

# Ejecutar ngrok
ngrok http 5002 --domain=measured-moth-intensely.ngrok-free.app &
```

También, para que arranque en la pantalla local, se ha creado la siguiente parte del script para tener en cuenta el caso de cuando la aplicación se ejecuta en local

```
if is_monitor_connected; then

  xrandr --output HDMI-0 --auto
    # Ejecutar Chromium en la pantalla correcta
  /usr/bin/chromium-browser --noerrdialogs --disable-infobars --kiosk http://localhost:5002 --display=:0
else
  echo "No se detecto un monitor."
fi
```

Ahora, para que se ejecute en anterior script de manera automática cuando arranca la Raspberry pi, se modifica el fichero autostart como se muestra a continuación:

```
  GNU nano 3.2

@lxpanel --profile LXDE-pi
@pcmanfm --desktop --profile LXDE-pi
@xscreensaver -no-splash
@/home/pi/Desktop/elyssar_code/my_project/webapp/start_app.sh
```

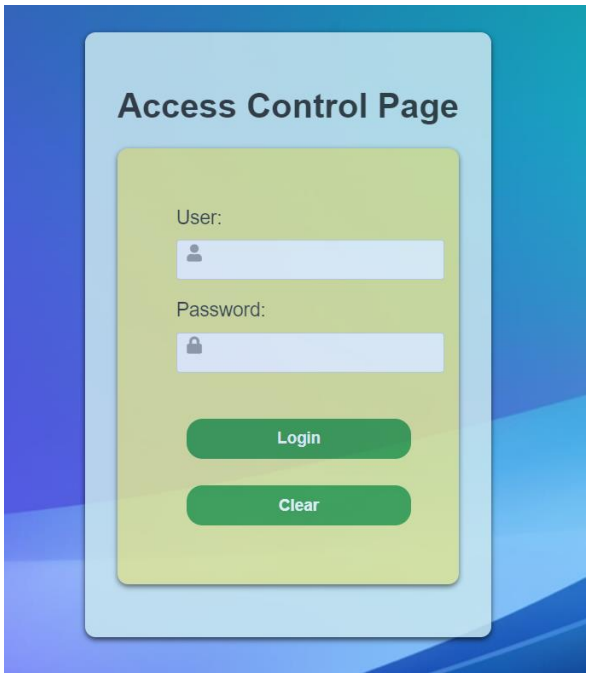
8.2 ANEXO II: MANUAL DE USUARIO

II- Manual de Usuario

En este anexo se incluyen los manuales de usuario correspondientes.

II.I. Para administradores

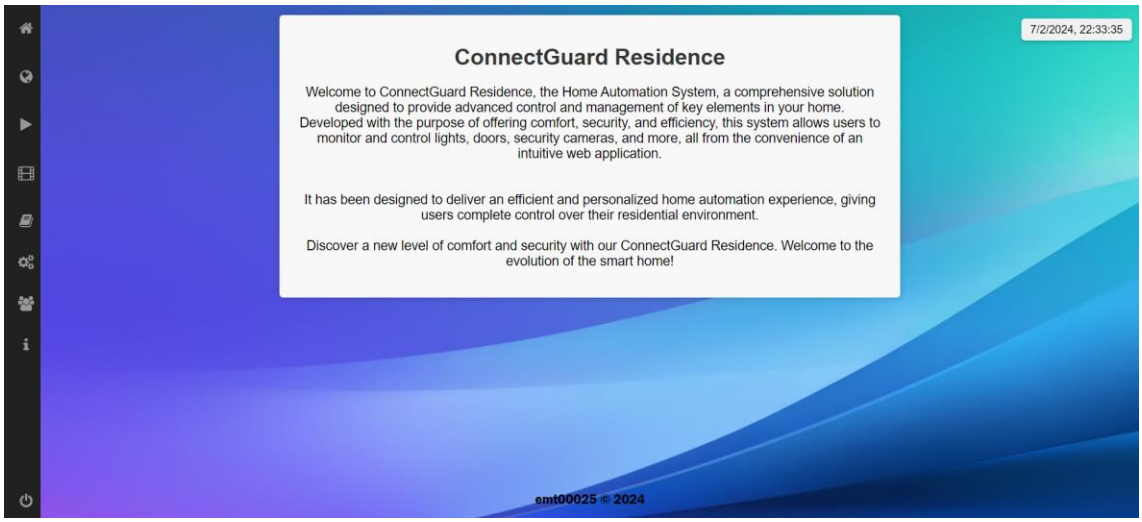
Una vez que se accede a la página web, se solicitarán las credenciales.



Dependiendo del rol asignado, si es como administrador, se tendrá acceso a todas las funciones y elementos del menú para gestionar y controlar el sistema domótico.

A continuación, se presenta la vista de la Página Web para Administradores:

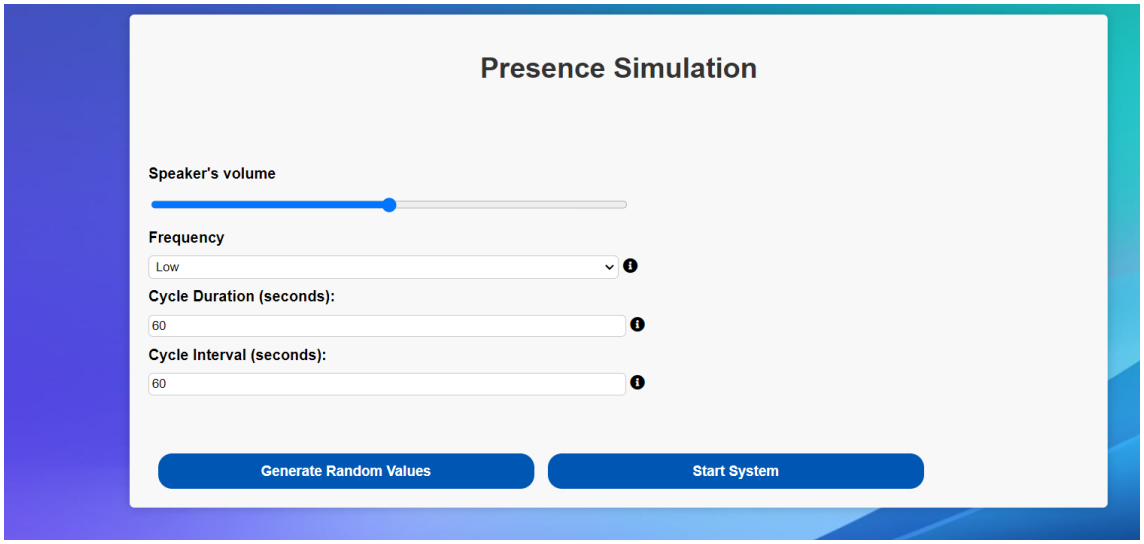
- Dashboard: presenta una breve introducción al sistema domótico



Esta vista no permite realizar ninguna acción.

➢ Menú de Opciones - Administrador:

- Simulación de presencia:

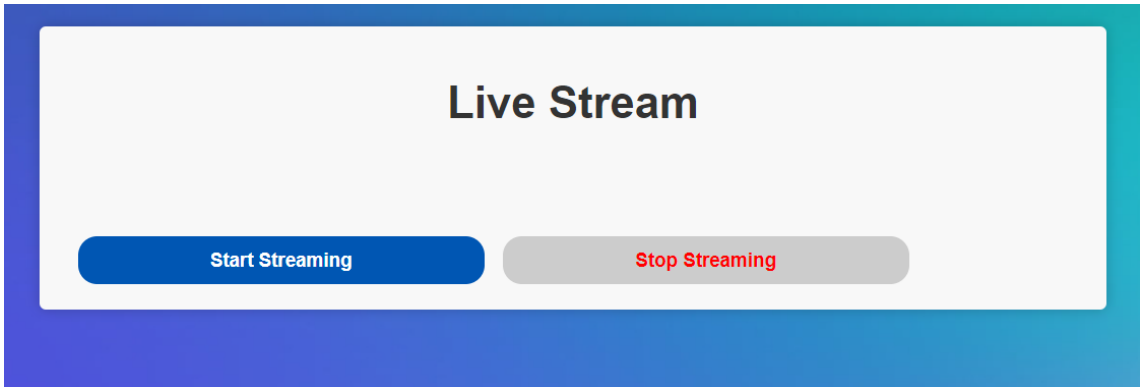


Descripción: permite realizar una nueva simulación de presencia.
Acciones disponibles:

- Elegir valores de los distintos parámetros: Volumen del altavoz, frecuencia de funcionamiento, duración del ciclo y tiempo entre ciclos, luego pulsar el botón “Start System”.

- Pulsar el botón “Generate Random Values” para que se generen valores aleatorios de los distintos parámetros y finalmente pulsar el botón “Start System” para que empiece la simulación.

- Transmisión en Directo:

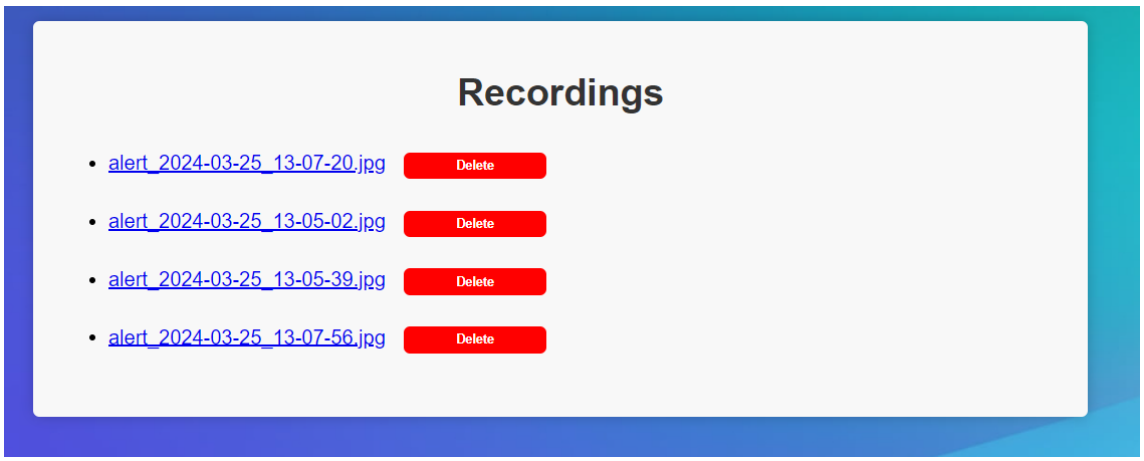


Descripción: Facilita la visualización en tiempo real de la cámara de seguridad.
Acciones Disponibles:

- Pulsar el botón “Start Streaming” para que empieza la transmisión en directo desde la cámara de seguridad.

- Pulsar el botón “Stop Streaming” para detener la transmisión en directo.

- Visualizar Grabaciones:

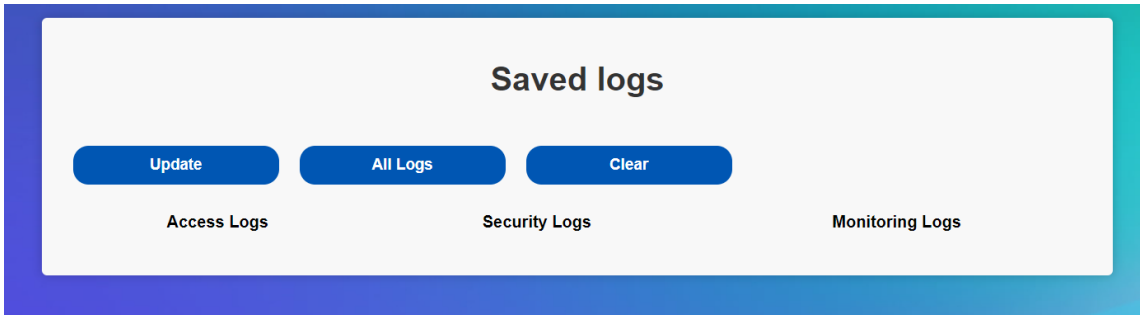


Descripción: Permite al administrador ver grabaciones almacenadas.
Acciones Disponibles:

- Descargar la grabación deseada pulsando sobre el nombre correspondiente.

- Borrar una grabación pulsando el botón “Delete”

- Logs y Registros:



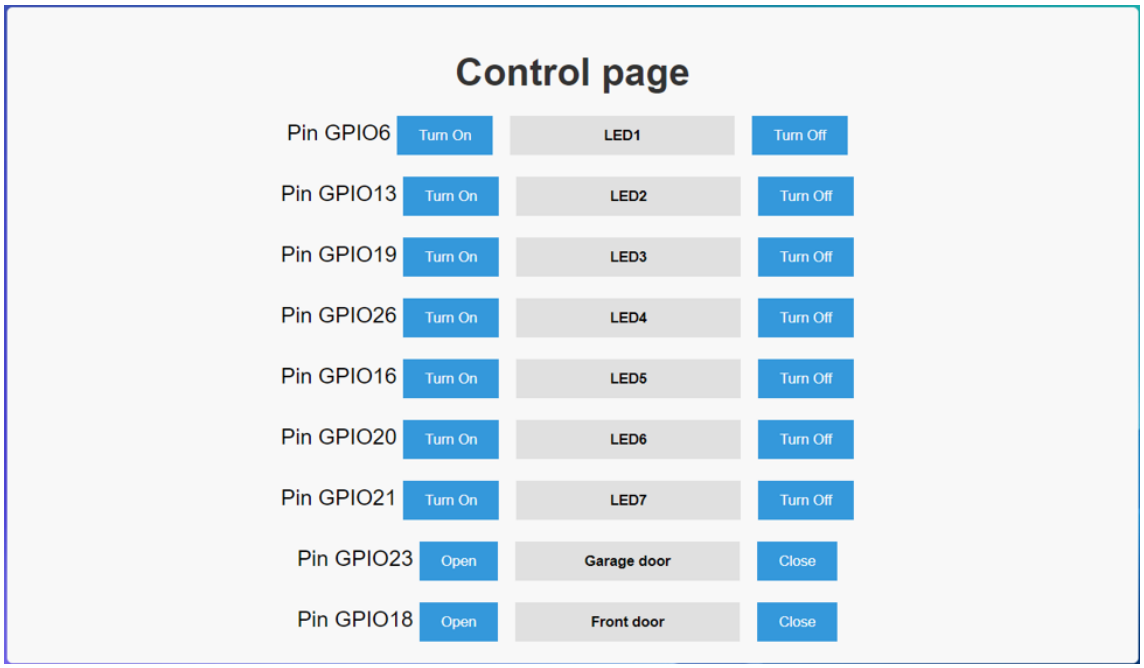
Descripción: Ofrece acceso a los registros del sistema.
Acciones Disponibles:

- Pulsar el botón “Update” para cargar y visualizar los registros almacenados en el entono local.

- Pulsar el botón “All Logs” para cargar y visualizar los registros almacenados en la base de datos de registros.

- Pulsar el botón “Clear” para borrar los registros almacenados en el entorno local.

- Control de Elementos:

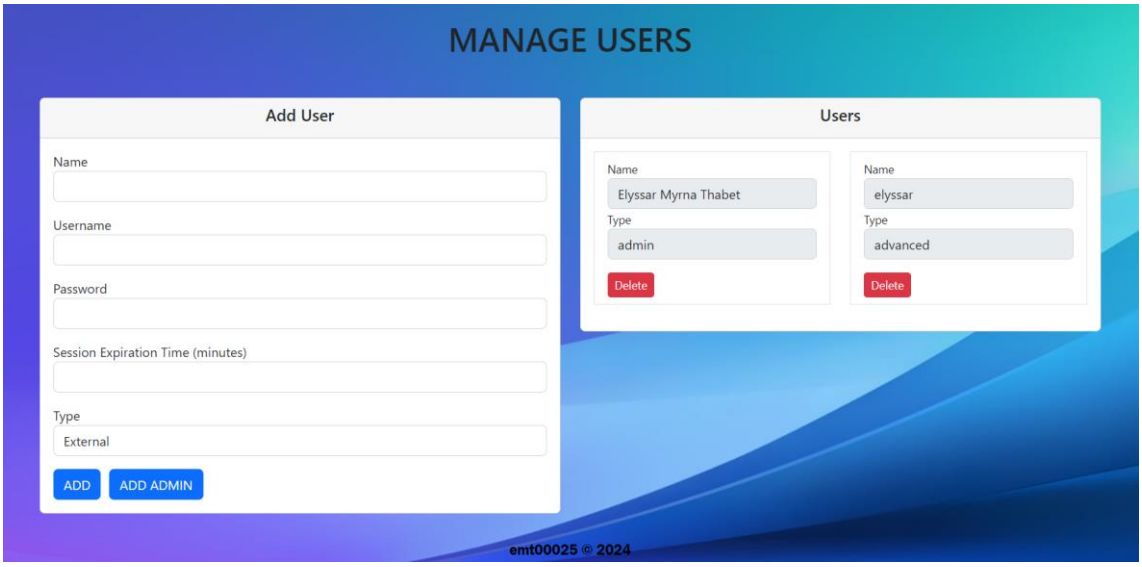


Permite al administrador controlar luces y puertas.
Acciones Disponibles:

- Pulsa el botón “Turn On” correspondiente al pin del elemento que se desea Encender en caso de las luces, o el que se desea Abrir en caso de la cerradura y la puerta de la cochera.

- Pulsar el botón “Turn Off” correspondiente al pin del elemento que se desea Apagar en caso de las luces, o el que se desea Cerrar en caso de la cerradura y la puerta de la cochera.

- Control de Usuarios:



Descripción: Permite la gestión de usuarios en el sistema.

Acciones Disponibles:

- Añadir un usuario: rellenar el formulario y pulsar el botón “ADD” en caso de querer añadir un usuario externo u avanzado.

- Añadir un admin: rellenar el formulario y pulsar el botón “ADD ADMIN”, rellenar los campos reservados para el tipo Admin, y pulsar el botón “Perform Facial Recognition”

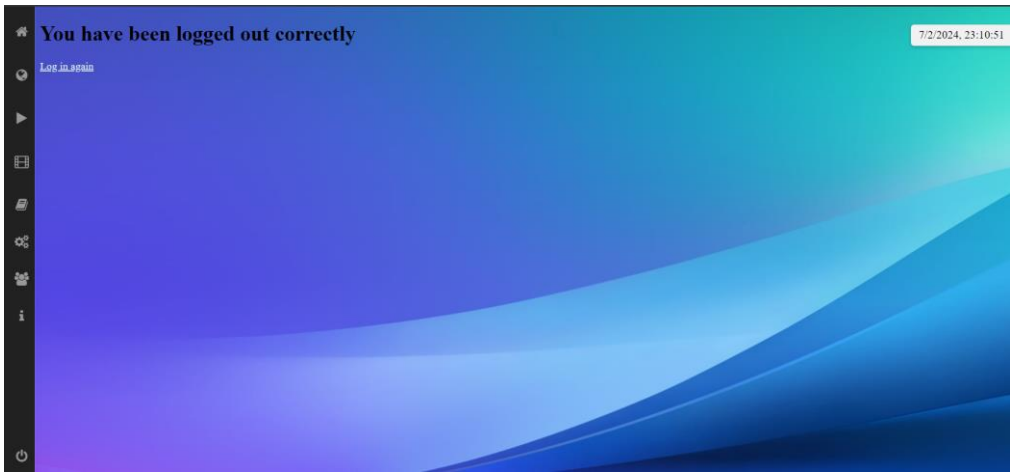
NOTA: para poder añadir un admin correctamente, el reconocimiento facial se tiene que realizar mediante la cámara localizada en la puerta principal de la vivienda.

- Borrar usuario pulsando el botón “DELETE” correspondiente al usuario en cuestión.

- Información del proyecto:

Presenta un texto descriptivo general acerca de la aplicación, la razón detrás de su desarrollo y su utilidad.
Esta vista no permite realizar ninguna acción.

- Cerrar sesión:

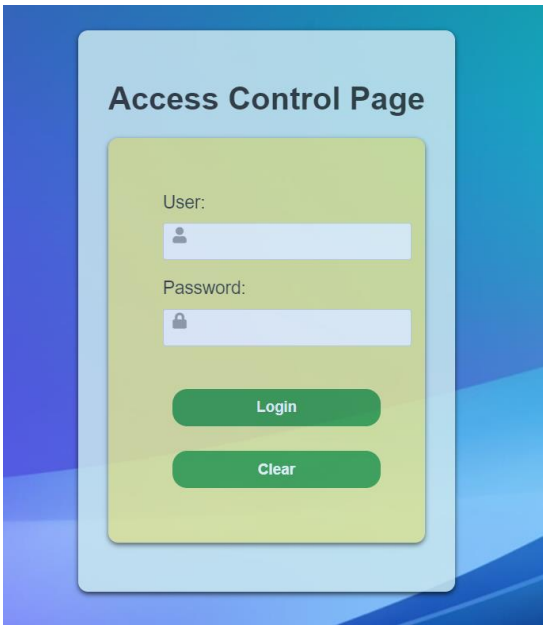


Descripción: permite cerrar la sesión actual de usuario.
Acciones disponibles:

- Pulsar la opción “Logout” para cerrar sesión.
- Pulsar “Login again” para redirigir al formulario inicial de Login de nuevo.

II.II. Para usuarios avanzados

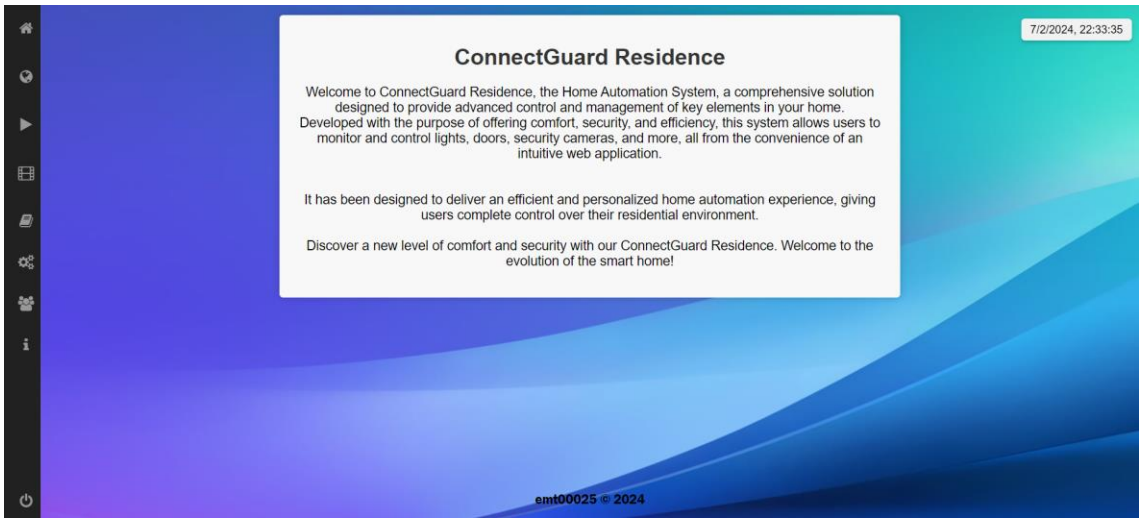
Una vez que se accede a la página web, se solicitarán las credenciales.



Dependiendo del rol asignado, si es avanzado, se tendrá acceso a ciertas funciones y elementos del menú para gestionar y controlar el sistema domótico.

A continuación, se presenta la vista de la Página Web para usuarios avanzados:

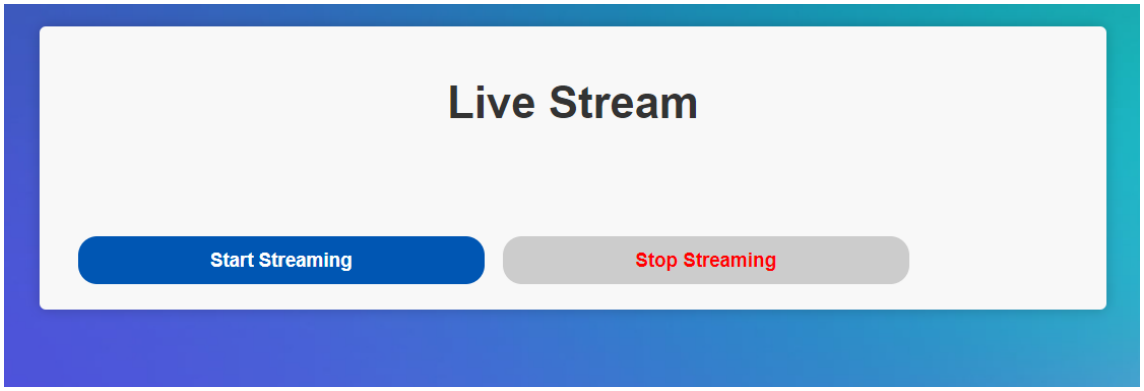
- Dashboard: presenta una breve introducción al sistema domótico



Esta vista no permite realizar ninguna acción.

➢ Menú de Opciones – Usuario Avanzado:

- Transmisión en Directo:

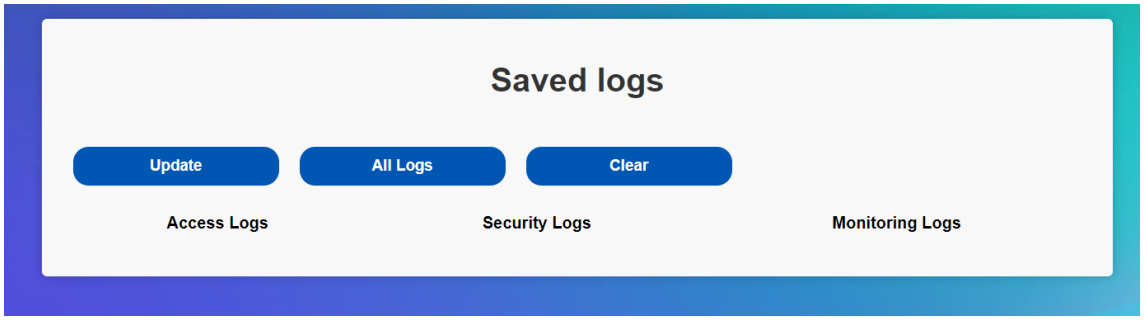


Descripción: Facilita la visualización en tiempo real de la cámara de seguridad.
Acciones Disponibles:

- Pulsar el botón “Start Streaming” para que empieza la transmisión en directo desde la cámara de seguridad.

- Pulsar el botón “Stop Streaming” para detener la transmisión en directo.

- Logs y Registros:



Descripción: Ofrece acceso a los registros del sistema.
Acciones Disponibles:

- Pulsar el botón “Update” para cargar y visualizar los registros almacenados en el entorno local.

- Pulsar el botón “All Logs” para cargar y visualizar los registros almacenados en la base de datos de registros.

- Pulsar el botón “Clear” para borrar los registros almacenados en el entorno local.

- Control de Elementos

Permite al administrador controlar luces y puertas.
Acciones Disponibles:

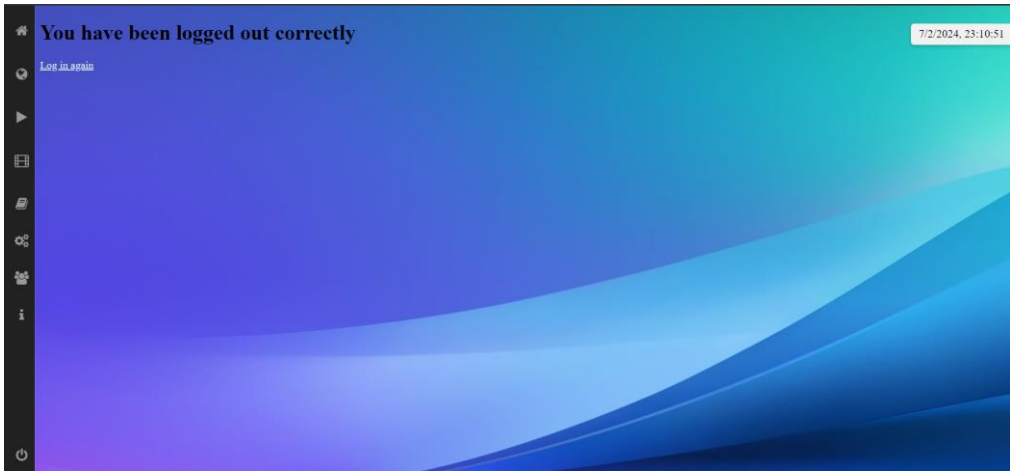
- Pulsa el botón “Turn On” correspondiente al pin del elemento que se desea Encender en caso de las luces, o el que se desea Abrir en caso de la cerradura y la puerta de la cochera.

- Pulsar el botón “Turn Off” correspondiente a al pin del elemento que se desea Apagar en caso de las luces, o el que se desea Cerrar en caso de la cerradura y la puerta de la cochera.

- Información del proyecto:

Presenta un texto descriptivo general acerca de la aplicación, la razón detrás de su desarrollo y su utilidad.
Esta vista no permite realizar ninguna acción.

- Cerrar sesión:

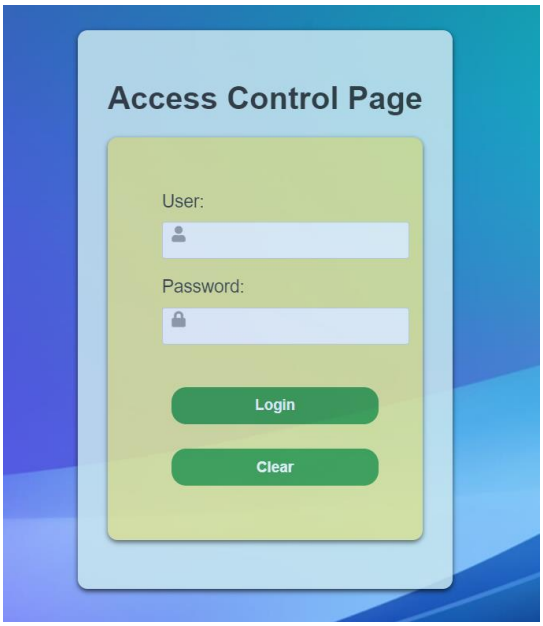


Descripción: permite cerrar la sesión actual de usuario.
Acciones disponibles:

- Pulsar la opción “Logout” para cerrar sesión.
- Pulsar “Login again” para redirigir al formulario inicial de Login de nuevo.

II.III. Para usuarios externos

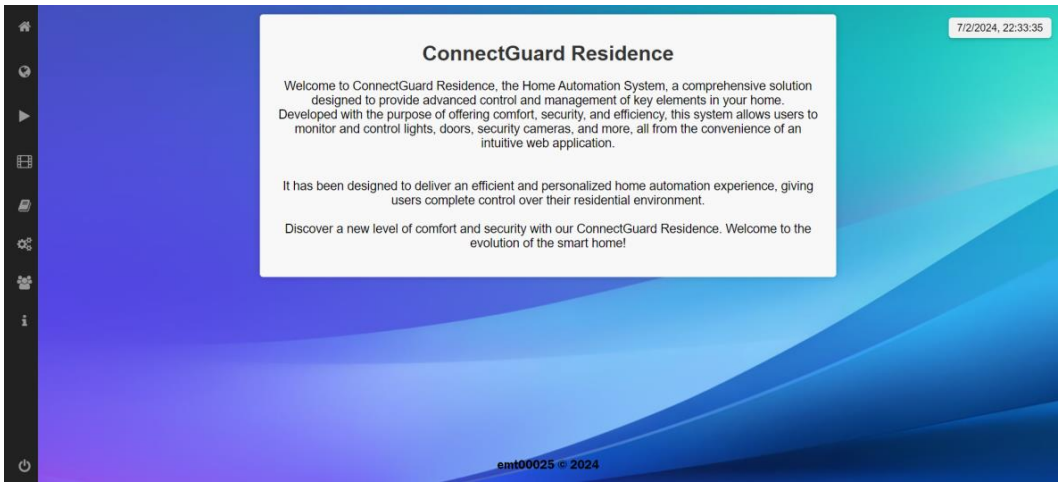
Una vez que se accede a la página web, se solicitarán las credenciales.



Dependiendo del rol asignado, si es externo, se tendrá acceso limitado a los elementos del menú para poder visualizar algunos parámetros del sistema domótico.

A continuación, se presenta la vista de la Página Web para usuarios externos:

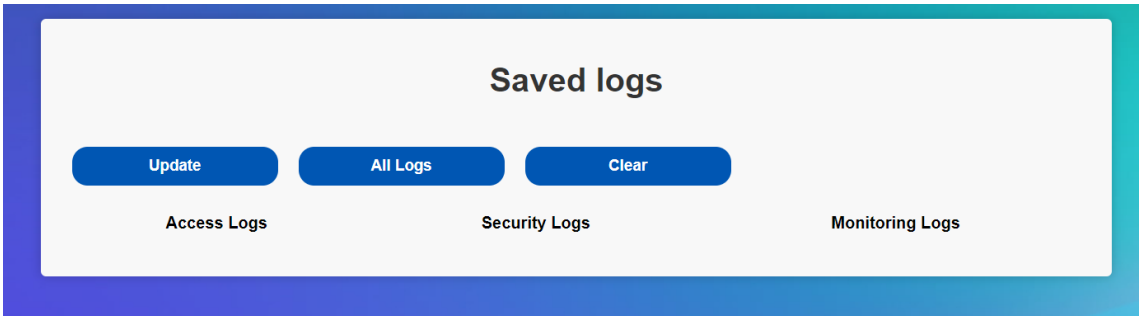
- Dashboard: presenta una breve introducción al sistema domótico



Esta vista no permite realizar ninguna acción.

➢ Menú de Opciones – Usuario Externo:

- Logs y Registros:



Descripción: Ofrece acceso a los registros del sistema.
Acciones Disponibles:

- Pulsar el botón “Update” para cargar y visualizar los registros almacenados en el entono local.

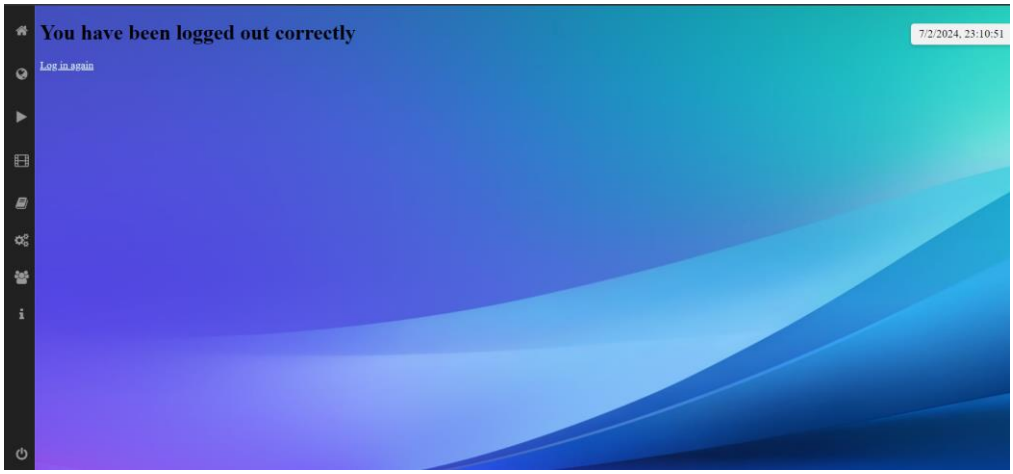
- Pulsar el botón “All Logs” para cargar y visualizar los registros almacenados en la base de datos de registros.

- Pulsar el botón “Clear” para borrar los registros almacenados en el entorno local.

- Información del proyecto:

Presenta un texto descriptivo general acerca de la aplicación, la razón detrás de su desarrollo y su utilidad.
Esta vista no permite realizar ninguna acción.

- Cerrar sesión:



Descripción: permite cerrar la sesión actual de usuario.
Acciones disponibles:

- Pulsar la opción “Logout” para cerrar sesión.

- Pulsar “Login again” para redirigir al formulario inicial de Login de nuevo.

8.3 ANEXO III: MANUAL DE MANTENIMIENTO

III- Manual de mantenimiento

El mantenimiento de la maqueta de la casa domótica es esencial para asegurar su funcionamiento óptimo y prolongar su vida útil. Es importante tener en cuenta que solo los usuarios con permisos de administrador pueden llevar a cabo estas tareas.

Los componentes se han colocado estratégicamente en la maqueta para que, en caso de que se rompa o se queme algún componente, sea fácil de reemplazar.

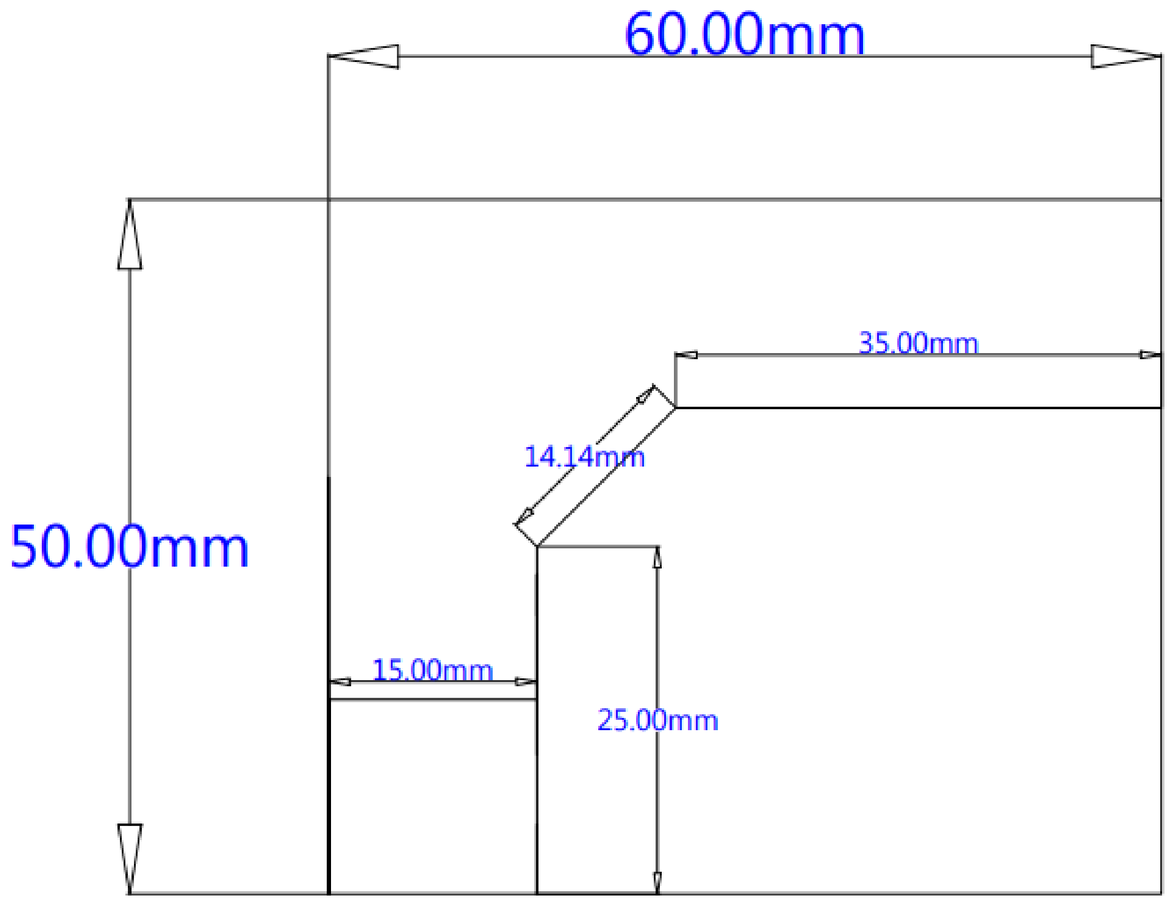
Además, se puede ver en el prototipo que todos los cables están rotulados claramente para identificar a qué elemento corresponde cada uno, facilitando así las tareas de reparación y actualización.

Por otra parte, algunas configuraciones y ajustes se pueden realizar a través de la web, Para cambiar el nombre de los pines de la página de control, se accede al código dentro de la carpeta “templates”, llamada “monitring.html”, y se modifica el atributo “value” de los elementos de tipo “input” utilizando el código que se encuentra en las líneas: 44, 54, 64, 73, 82, 91, 101, 110, 119 y 128

lo que permite una gestión más flexible y eficiente de los diferentes componentes domóticos. Este enfoque asegura que cualquier intervención sea rápida y precisa, minimizando el tiempo de inactividad del sistema.

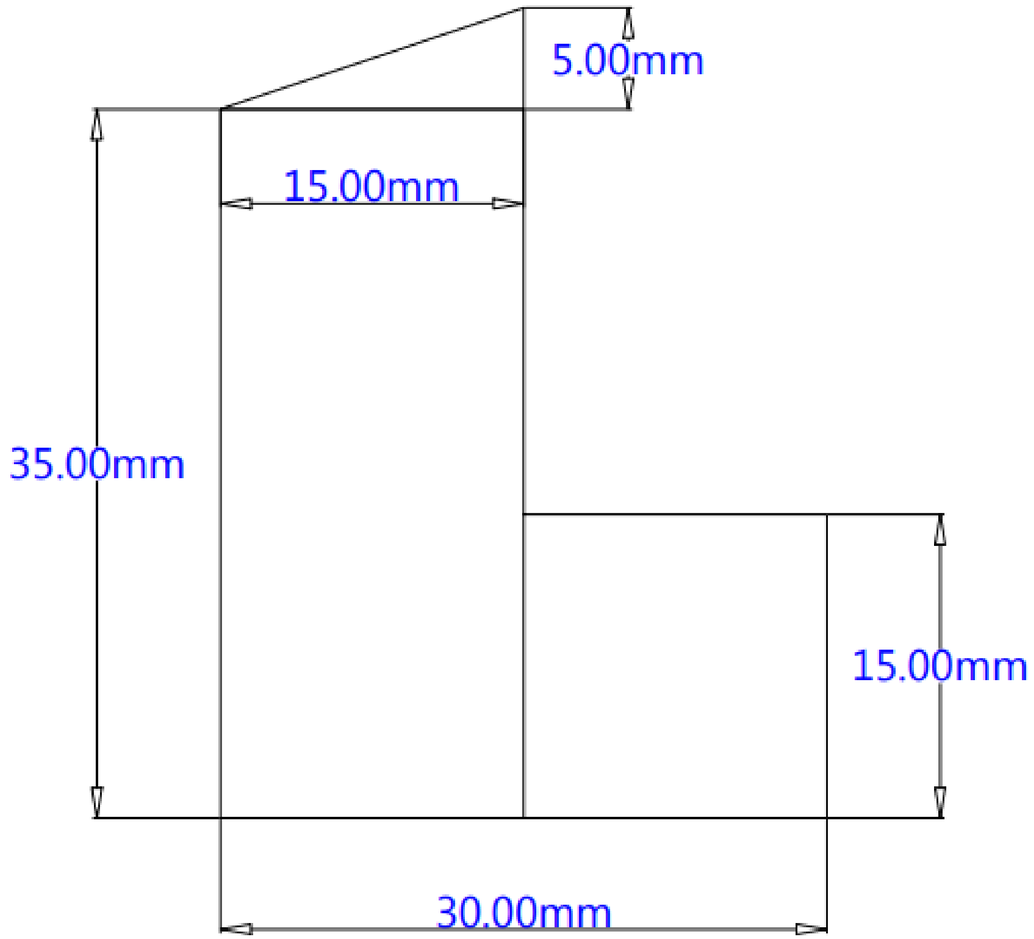
En relación a la base de datos, para borrar los registros, se puede acceder directamente a la tabla “logs” se elimina la información guardada. Eso permite evitar el colapso de la base de datos debido a la gran cantidad de información almacenada.

8.4 ANEXO IV: PLANOS



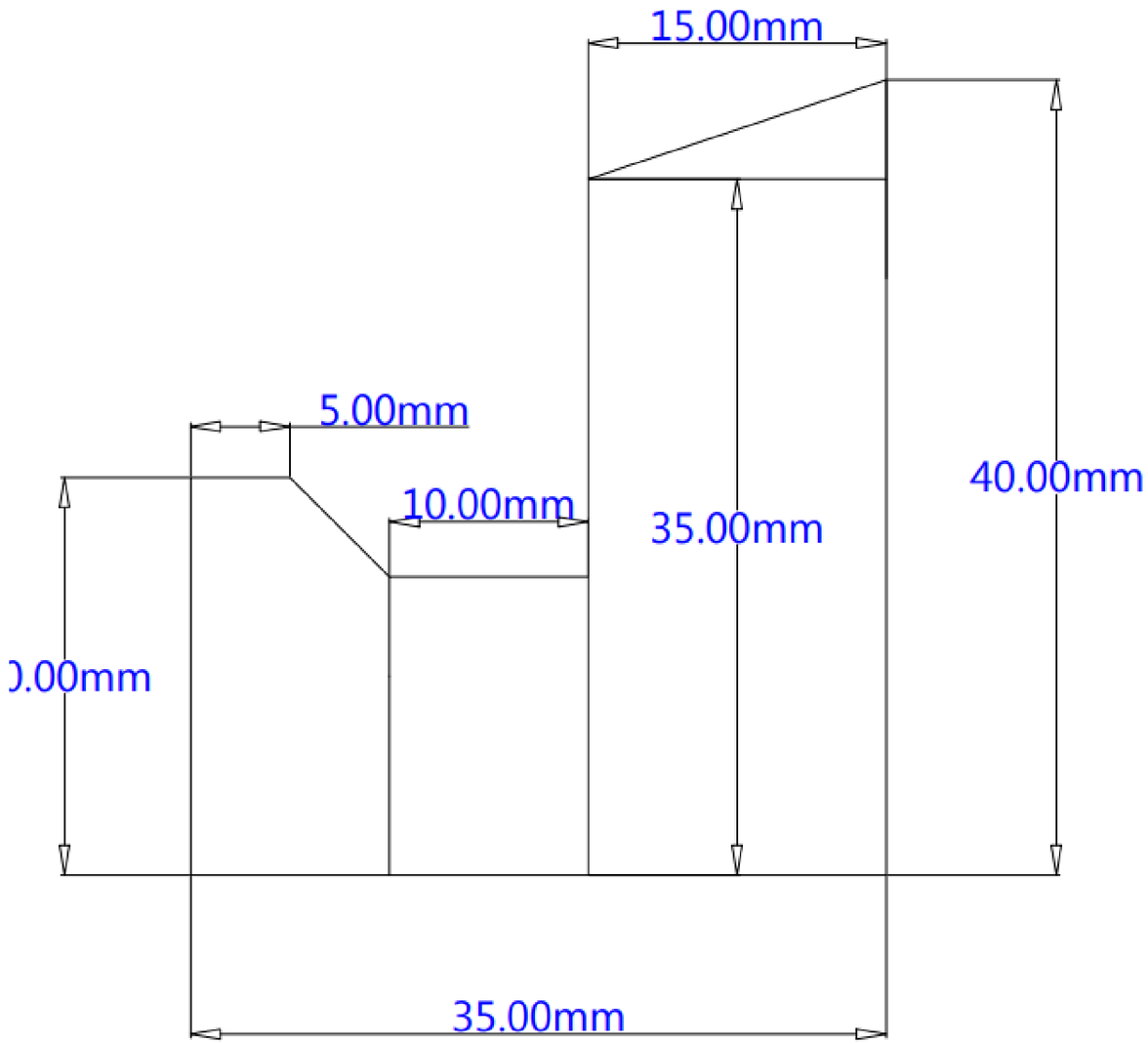
	FECHA	NOMBRE	FIRMA	ESCUELA POLITÉCNICA SUPERIOR LINARES
DIBUJADO	21/06/2024	Elyssar Myrna Thabet		
COMPROBADO				

ESCALA: 10:1	Sistema domótico a escala con acceso remoto Vista desde arriba	Nº PLANO 1 de 19
		SUSTITUYE A:
		SUSTITUIDO POR:



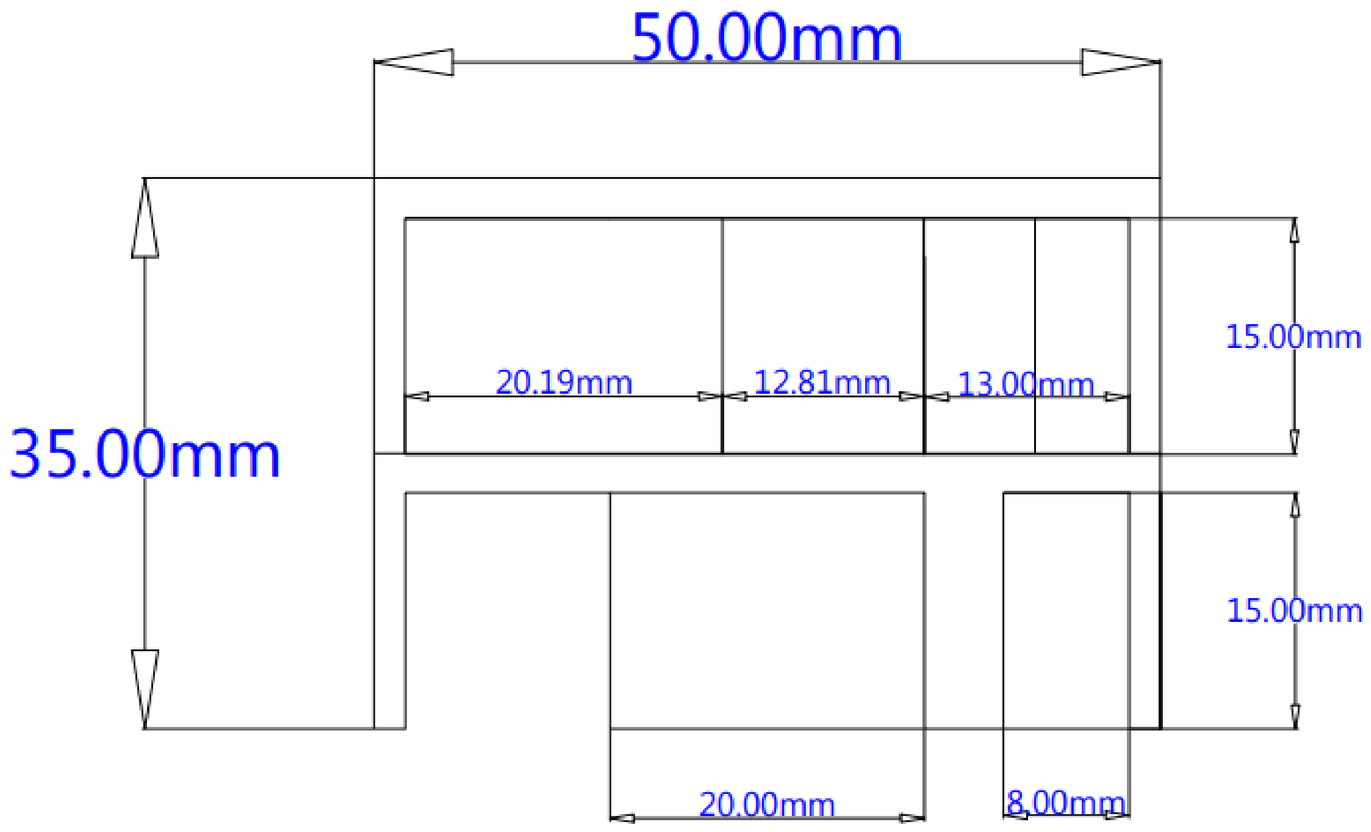
	FECHA	NOMBRE	FIRMA	ESCUELA POLITÉCNICA SUPERIOR LINARES
DIBUJADO	21/06/2024	Elyssar Myrna Thabet		
COMPROBADO				

ESCALA: 10:1	Sistema domótico a escala con acceso remoto Vista desde lado derecho	Nº PLANO 2 de 19
		SUSTITUYE A:
		SUSTITUIDO POR:



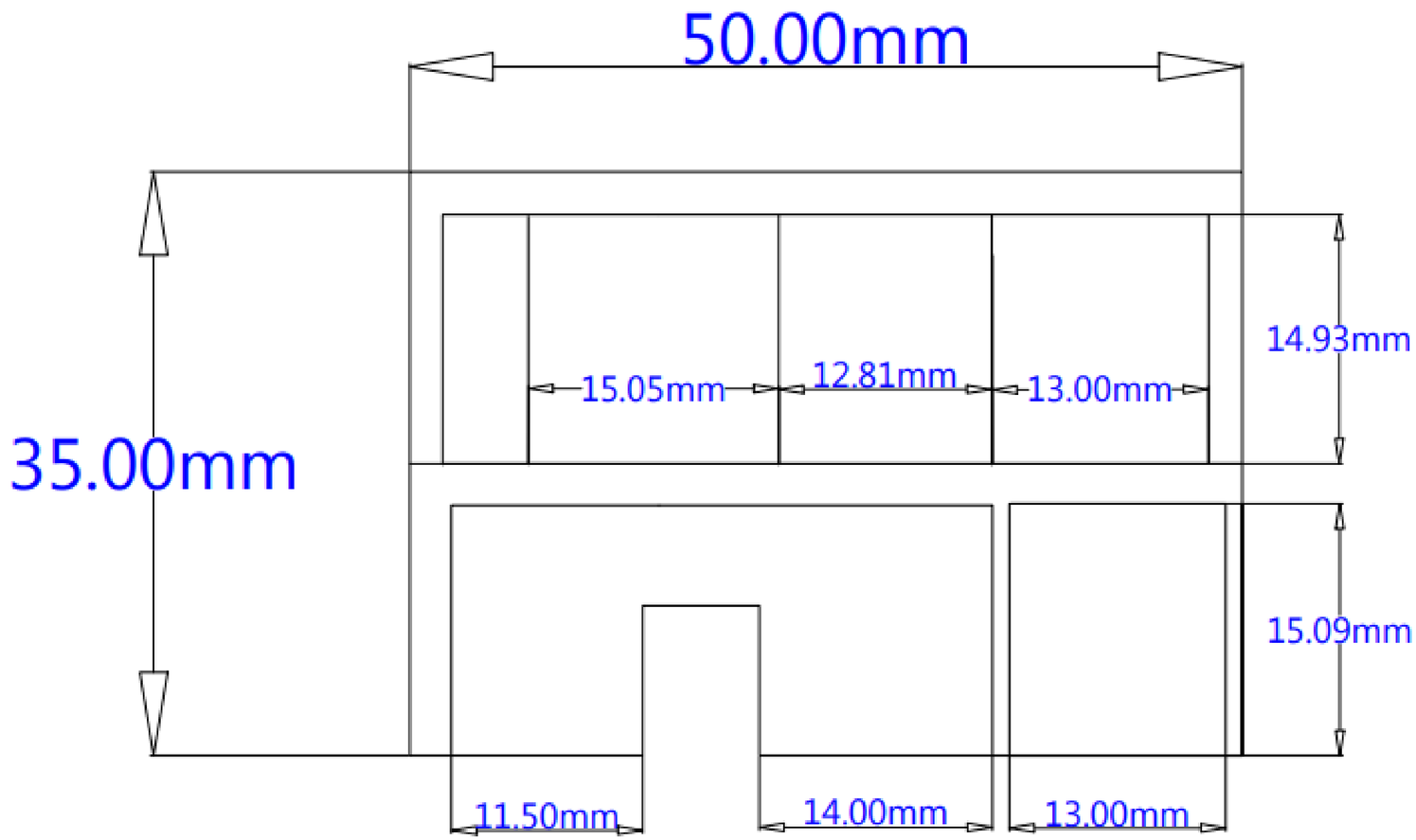
	FECHA	NOMBRE	FIRMA	ESCUELA POLITÉCNICA SUPERIOR LINARES
DIBUJADO	21/06/2024	Elyssar Myrna Thabet		
COMPROBADO				

ESCALA: 10:1	Sistema domótico a escala con acceso remoto Vista desde lado izquierdo	Nº PLANO 3 de 19
		SUSTITUYE A:
		SUSTITUIDO POR:



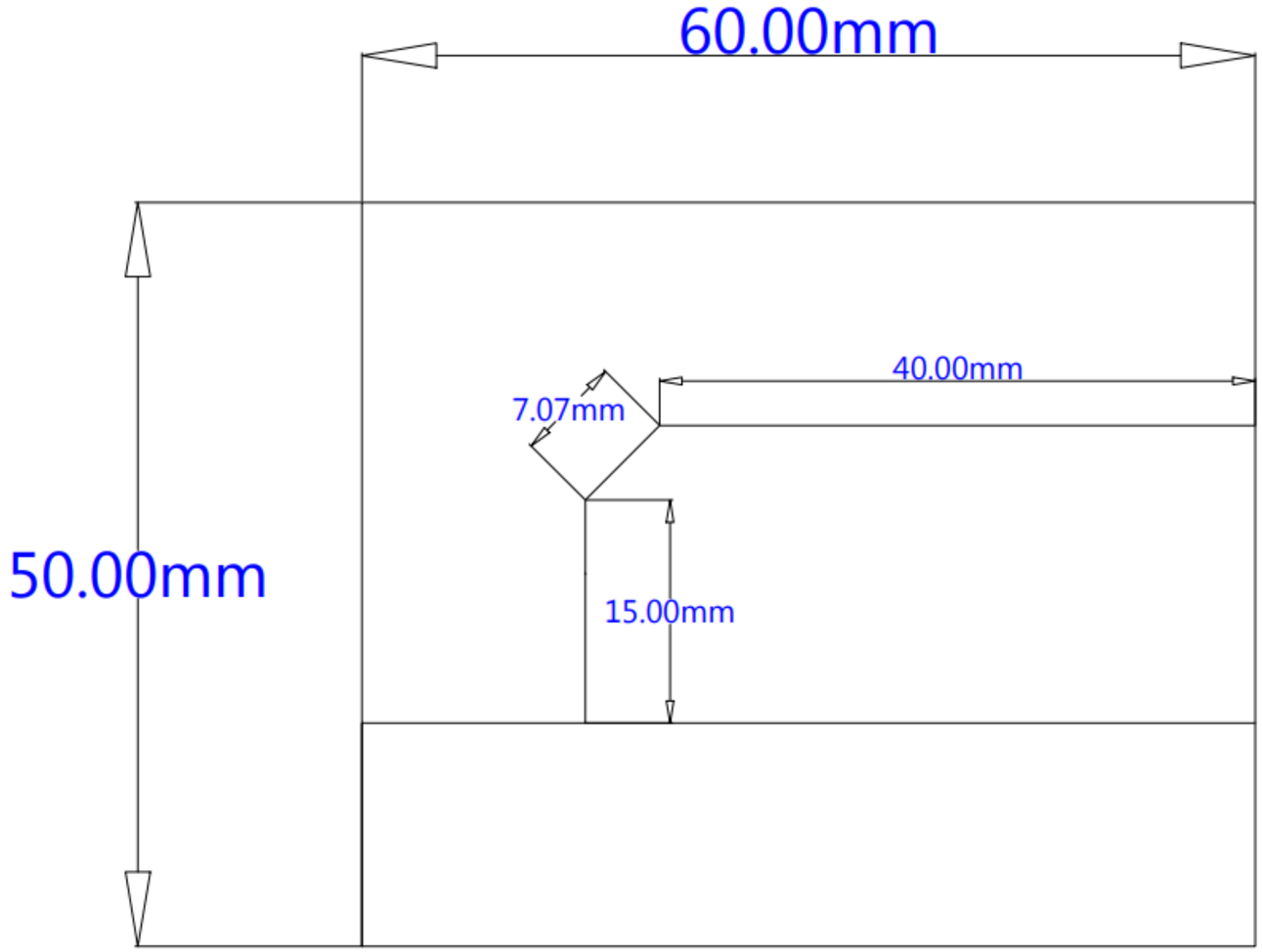
	FECHA	NOMBRE	FIRMA	ESCUELA POLITÉCNICA SUPERIOR LINARES
DIBUJADO	21/06/2024	Elyssar Myrna Thabet		
COMPROBADO				

ESCALA: 10:1	Sistema domótico a escala con acceso remoto Vista frontal	Nº PLANO 4 de 19
		SUSTITUYE A:
		SUSTITUIDO POR:



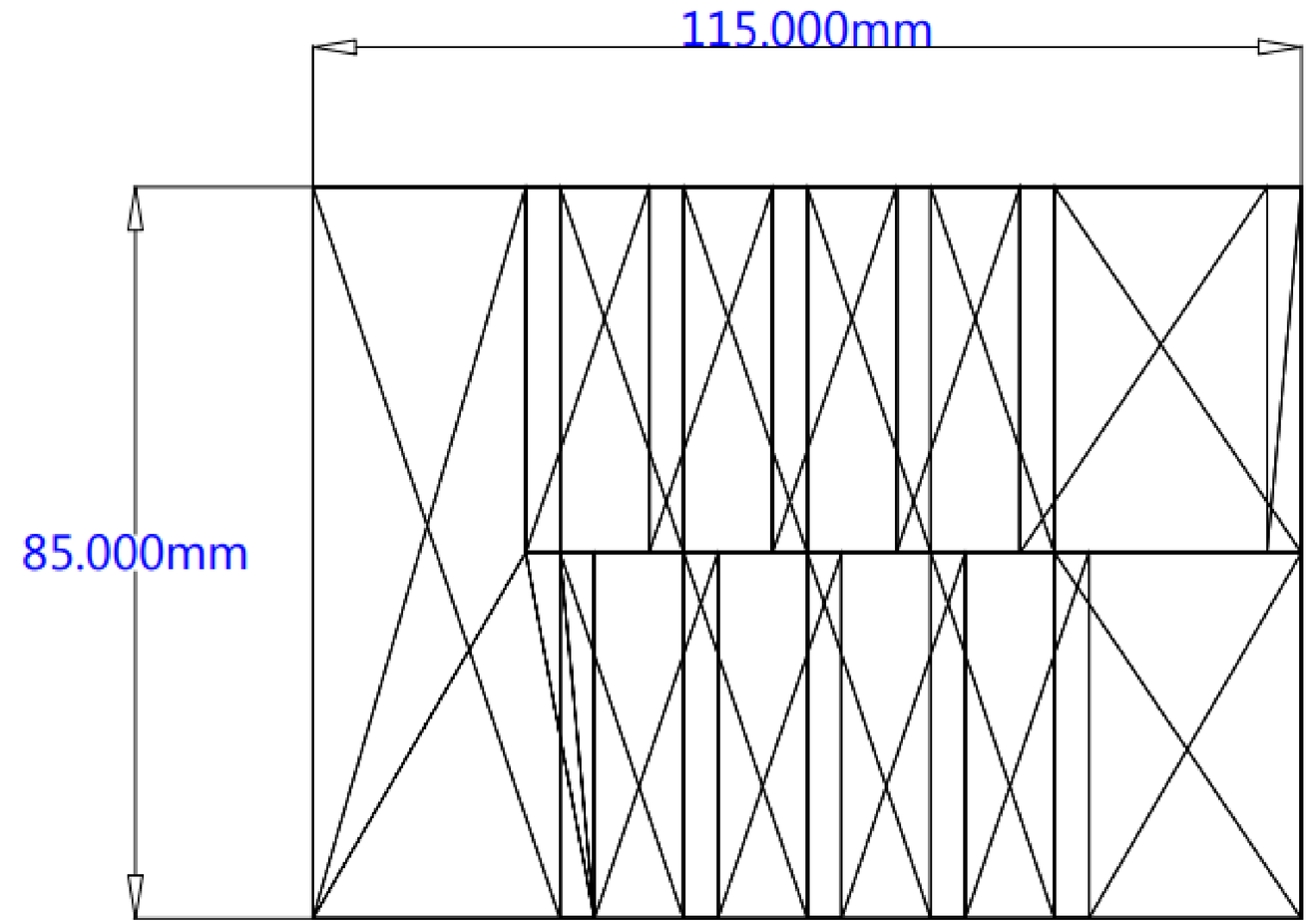
	FECHA	NOMBRE	FIRMA	ESCUELA POLITÉCNICA SUPERIOR LINARES
DIBUJADO	21/06/2024	Elyssar Myrna Thabet		
COMPROBADO				

ESCALA: 10:1	Sistema domótico a escala con acceso remoto Vista desde atrás	Nº PLANO 5 de 19
		SUSTITUYE A:
		SUSTITUIDO POR:



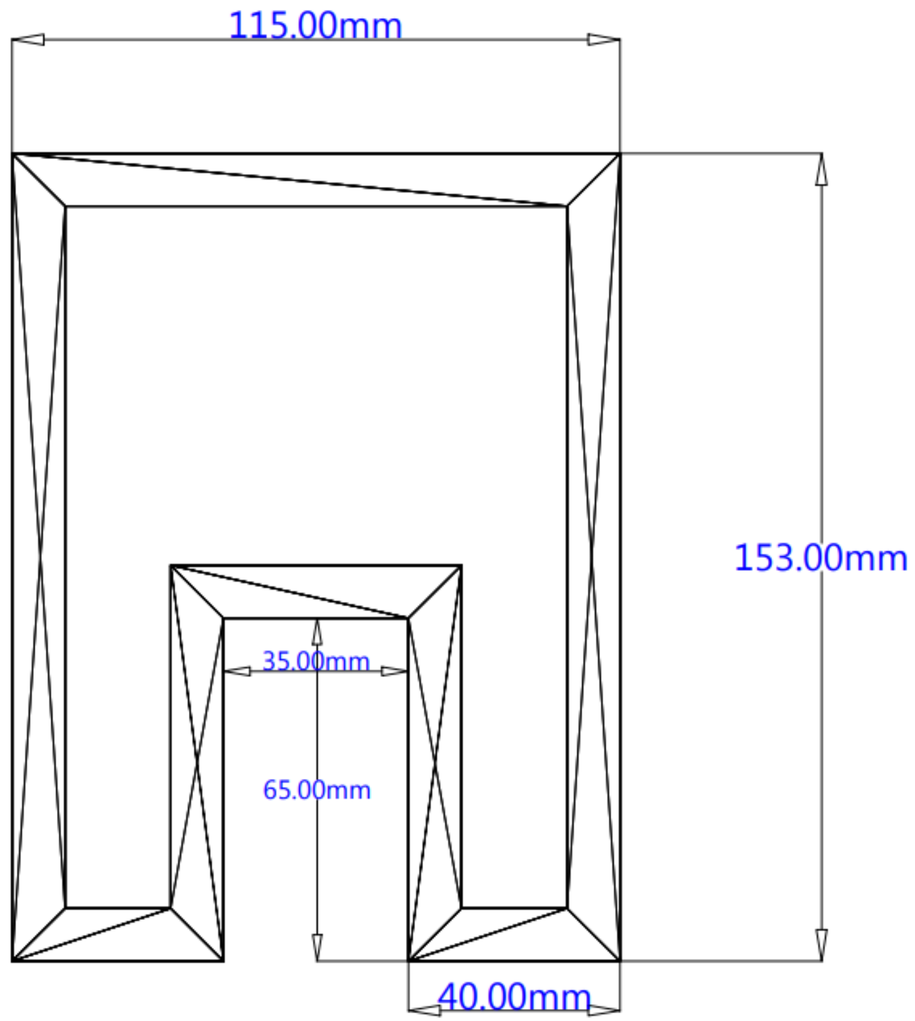
	FECHA	NOMBRE	FIRMA	ESCUELA POLITÉCNICA SUPERIOR LINARES
DIBUJADO	21/06/2024	Elyssar Myrna Thabet		
COMPROBADO				

ESCALA: 10:1	Sistema domótico a escala con acceso remoto Suelo	Nº PLANO 6 de 19
		SUSTITUYE A:
		SUSTITUIDO POR:



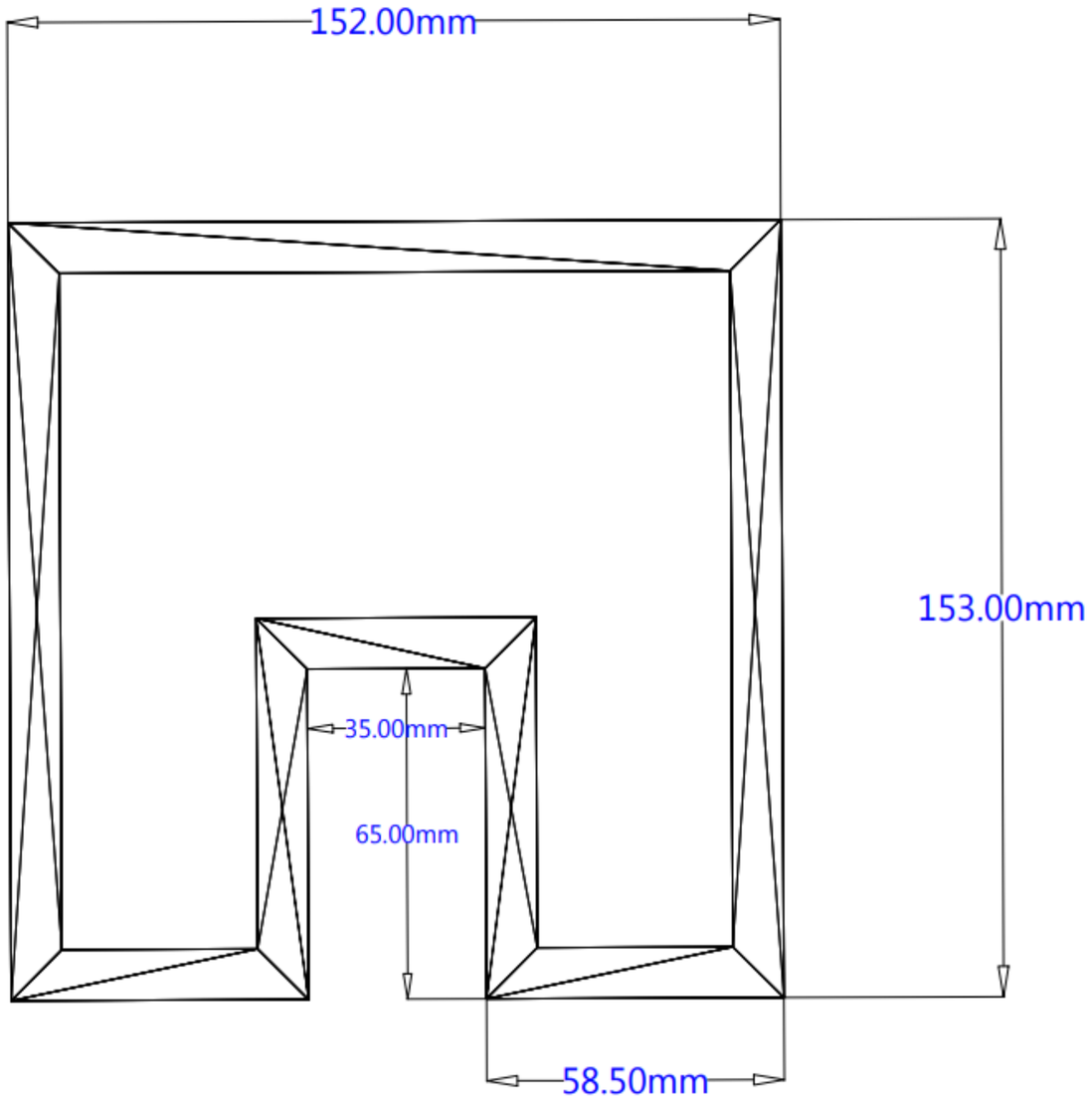
	FECHA	NOMBRE	FIRMA	ESCUELA POLITÉCNICA SUPERIOR LINARES
DIBUJADO	21/06/2024	Elyssar Myrna Thabet		
COMPROBADO				

ESCALA: 10:1	Sistema domótico a escala con acceso remoto Escalera	Nº PLANO 7 de 19
		SUSTITUYE A:
		SUSTITUIDO POR:



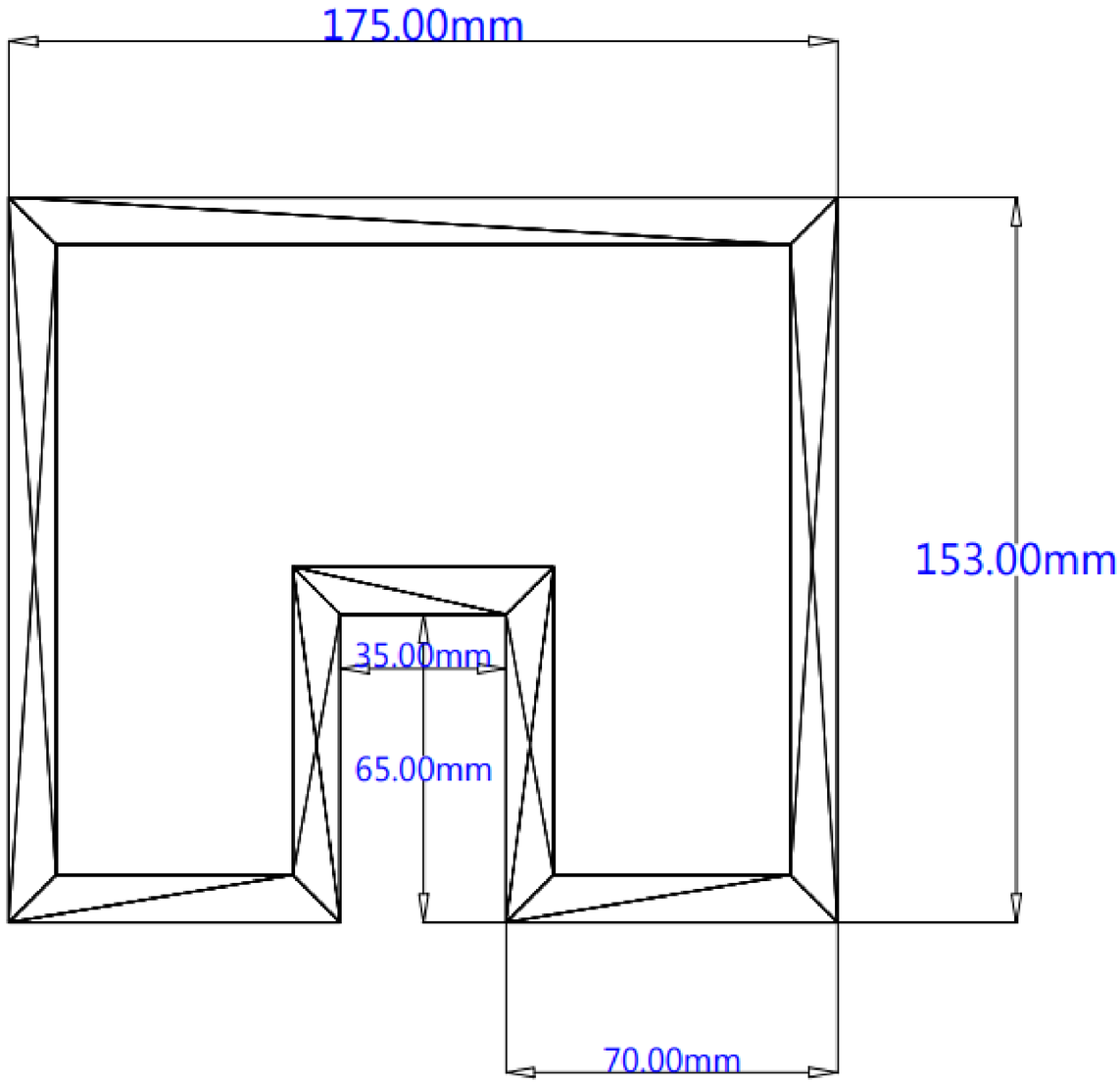
	FECHA	NOMBRE	FIRMA	ESCUELA POLITÉCNICA SUPERIOR LINARES
DIBUJADO	21/06/2024	Elyssar Myrna Thabet		
COMPROBADO				

ESCALA: 10:1	Sistema domótico a escala con acceso remoto Marco 1 Puerta	Nº PLANO 9 de 19 SUSTITUYE A: SUSTITUIDO POR:



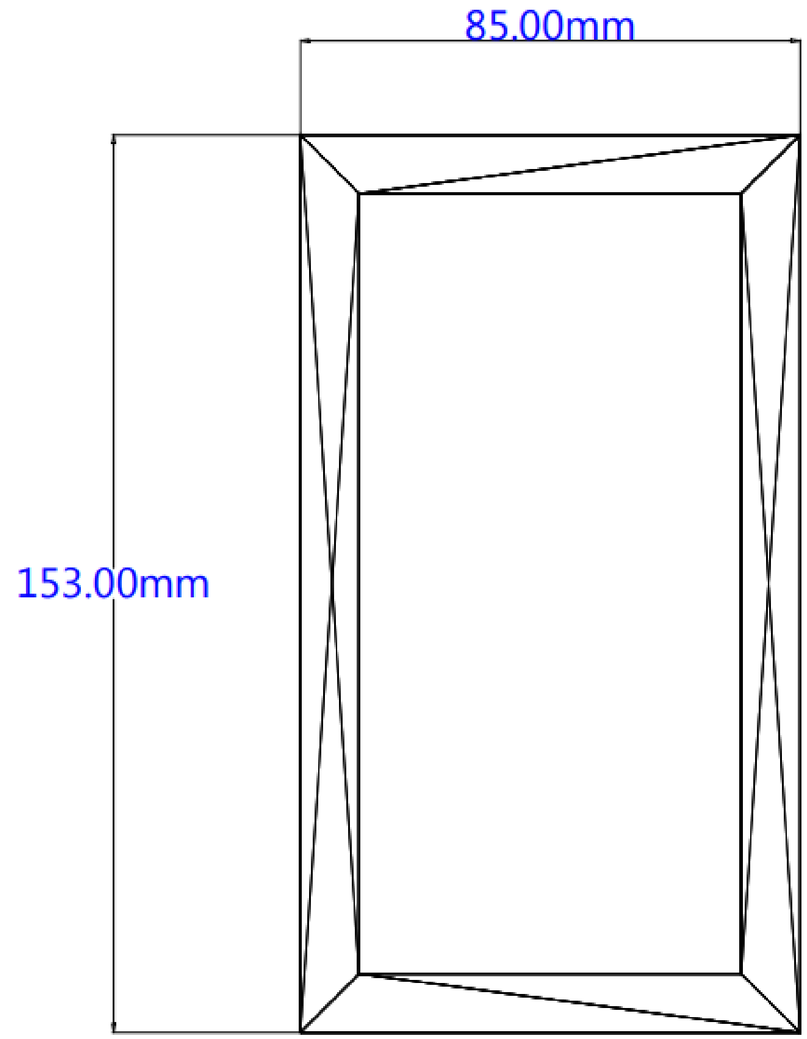
	FECHA	NOMBRE	FIRMA	ESCUELA POLITÉCNICA SUPERIOR LINARES
DIBUJADO	21/06/2024	Elyssar Myrna Thabet		
COMPROBADO				

ESCALA: 10:1	Sistema domótico a escala con acceso remoto Marco 2 Puerta	Nº PLANO 10 de 19
		SUSTITUYE A:
		SUSTITUIDO POR:



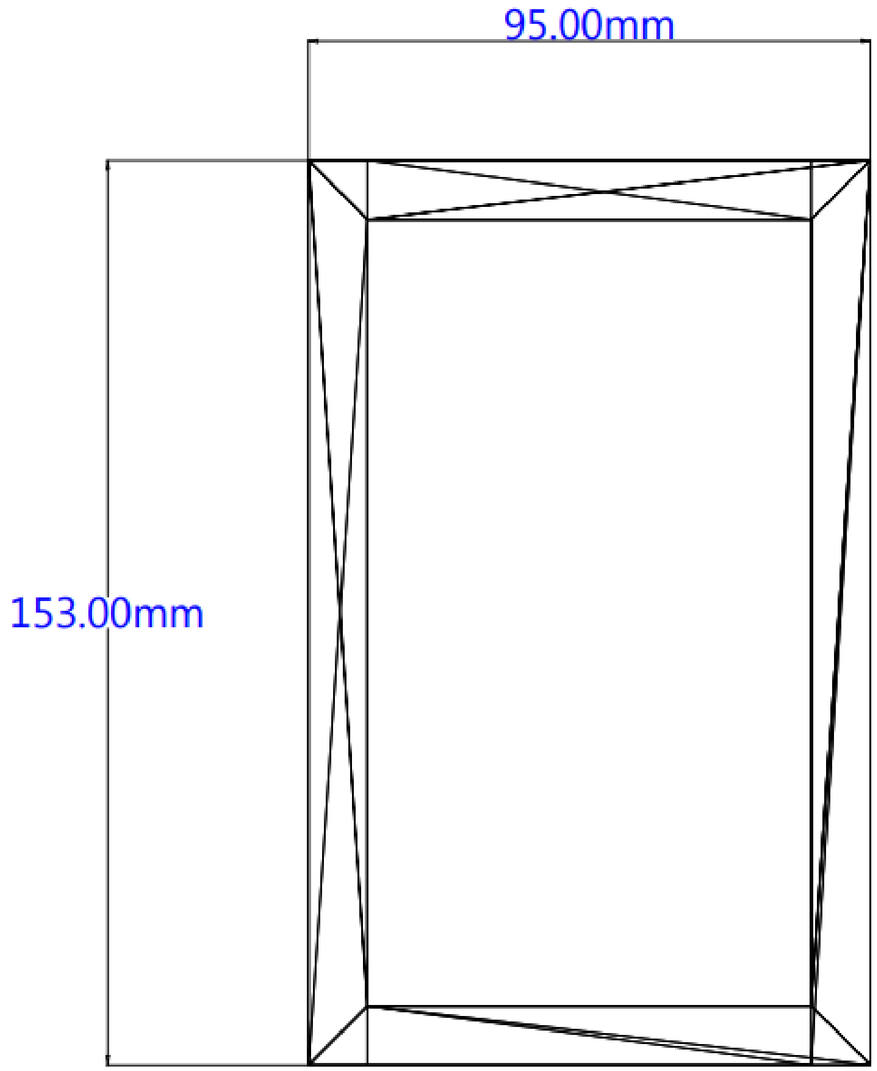
	FECHA	NOMBRE	FIRMA	ESCUELA POLITÉCNICA SUPERIOR LINARES
DIBUJADO	21/06/2024	Elyssar Myrna Thabet		
COMPROBADO				

ESCALA: 10:1	Sistema domótico a escala con acceso remoto Marco 3 Puerta	Nº PLANO 11 de 19
		SUSTITUYE A:
		SUSTITUIDO POR:



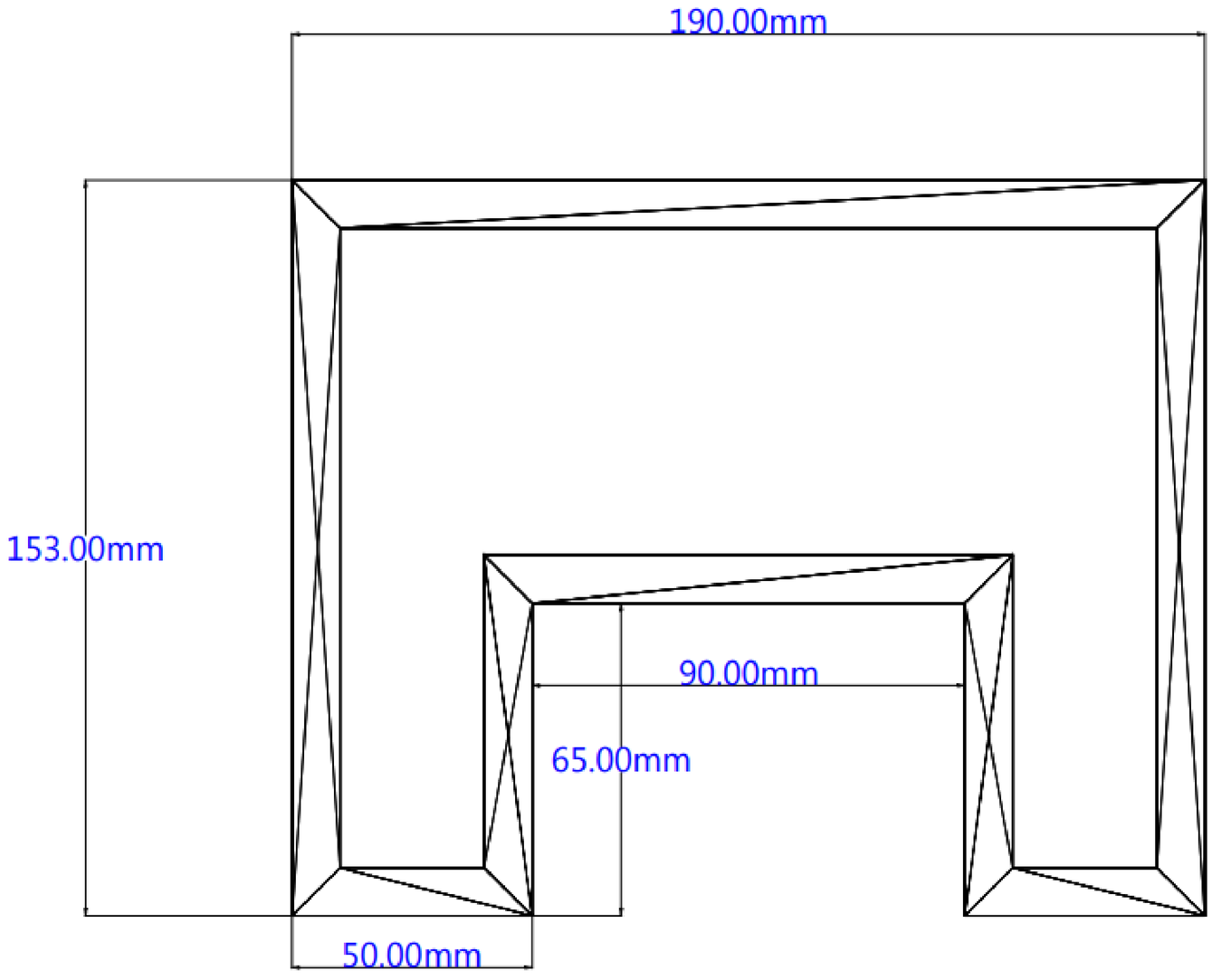
	FECHA	NOMBRE	FIRMA	ESCUELA POLITÉCNICA SUPERIOR LINARES
DIBUJADO	21/06/2024	Elyssar Myrna Thabet		
COMPROBADO				

ESCALA: 10:1	Sistema domótico a escala con acceso remoto Marco Esquina 1	Nº PLANO 12 de 19
		SUSTITUYE A:
		SUSTITUIDO POR:



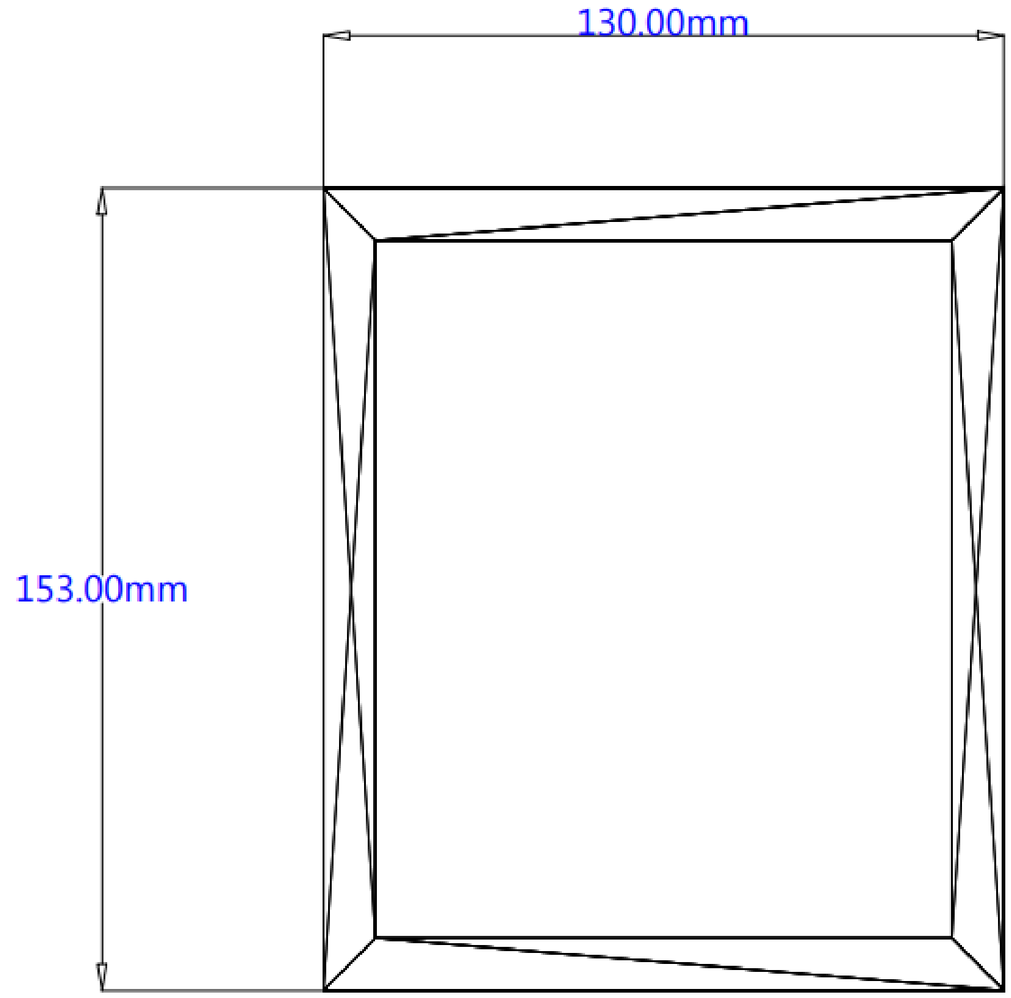
	FECHA	NOMBRE	FIRMA	ESCUELA POLITÉCNICA SUPERIOR LINARES
DIBUJADO	21/06/2024	Elyssar Myrna Thabet		
COMPROBADO				

ESCALA: 10:1	Sistema domótico a escala con acceso remoto Marco Esquina 2	Nº PLANO 13 de 19
		SUSTITUYE A:
		SUSTITUIDO POR:



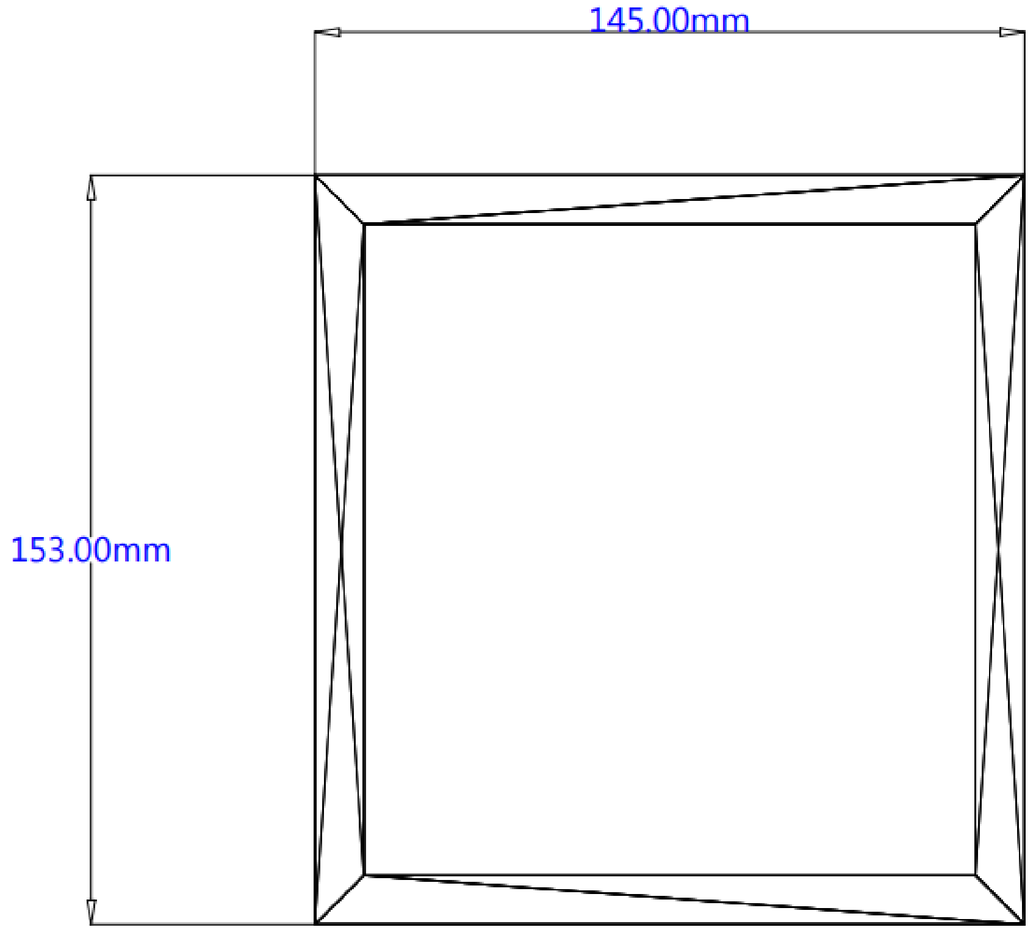
	FECHA	NOMBRE	FIRMA	ESCUELA POLITÉCNICA SUPERIOR LINARES
DIBUJADO	21/06/2024	Elyssar Myrna Thabet		
COMPROBADO				

ESCALA: 10:1	Sistema domótico a escala con acceso remoto Marco Puerta planta 1	Nº PLANO 14 de 19
		SUSTITUYE A:
		SUSTITUIDO POR:



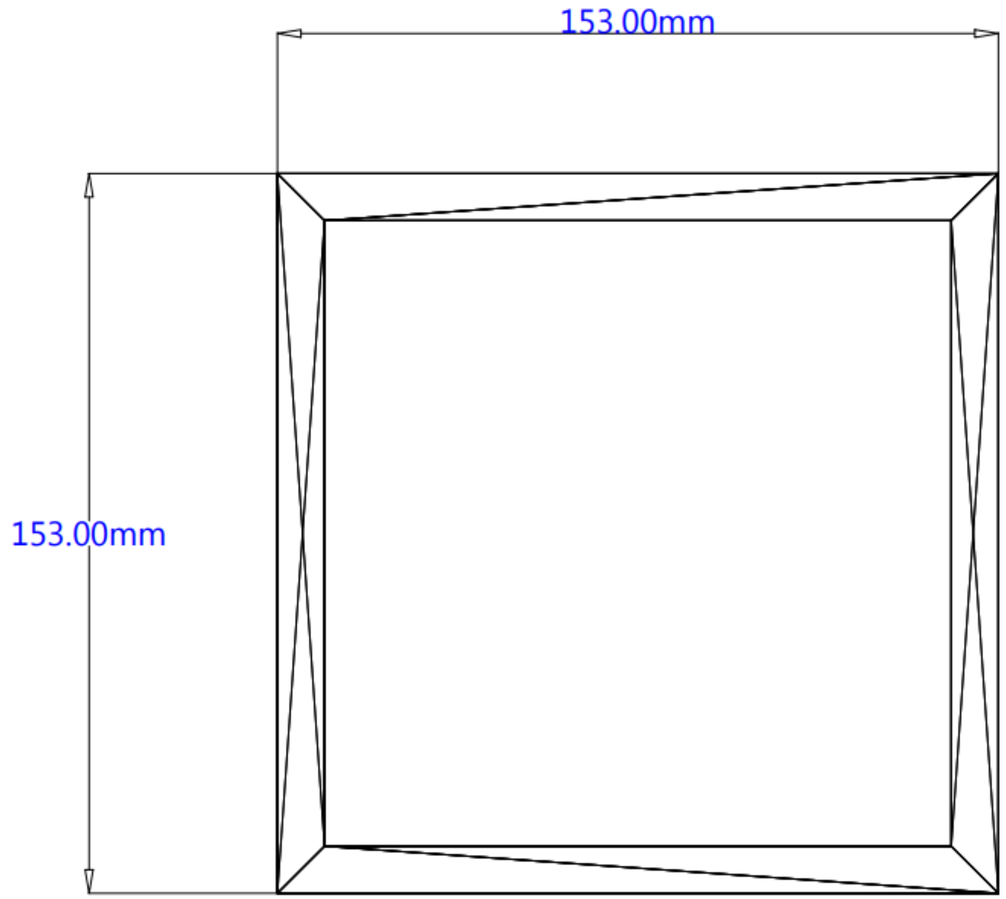
	FECHA	NOMBRE	FIRMA	ESCUELA POLITÉCNICA SUPERIOR LINARES
DIBUJADO	21/06/2024	Elyssar Myrna Thabet		
COMPROBADO				

ESCALA: 10:1	Sistema domótico a escala con acceso remoto Marco salón 1	Nº PLANO 15 de 19
		SUSTITUYE A:
		SUSTITUIDO POR:



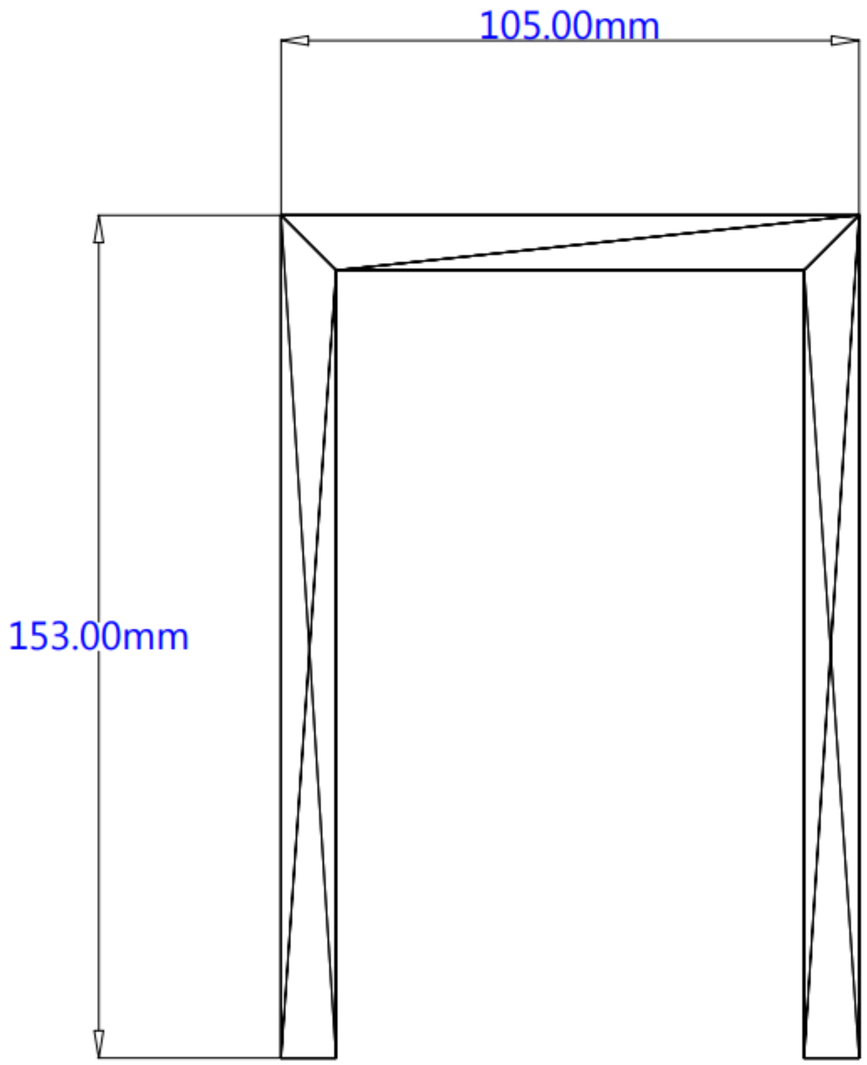
	FECHA	NOMBRE	FIRMA	ESCUELA POLITÉCNICA SUPERIOR LINARES
DIBUJADO	21/06/2024	Elyssar Myrna Thabet		
COMPROBADO				

ESCALA: 10:1	Sistema domótico a escala con acceso remoto Marco salón 2	Nº PLANO 16 de 19
		SUSTITUYE A:
		SUSTITUIDO POR:



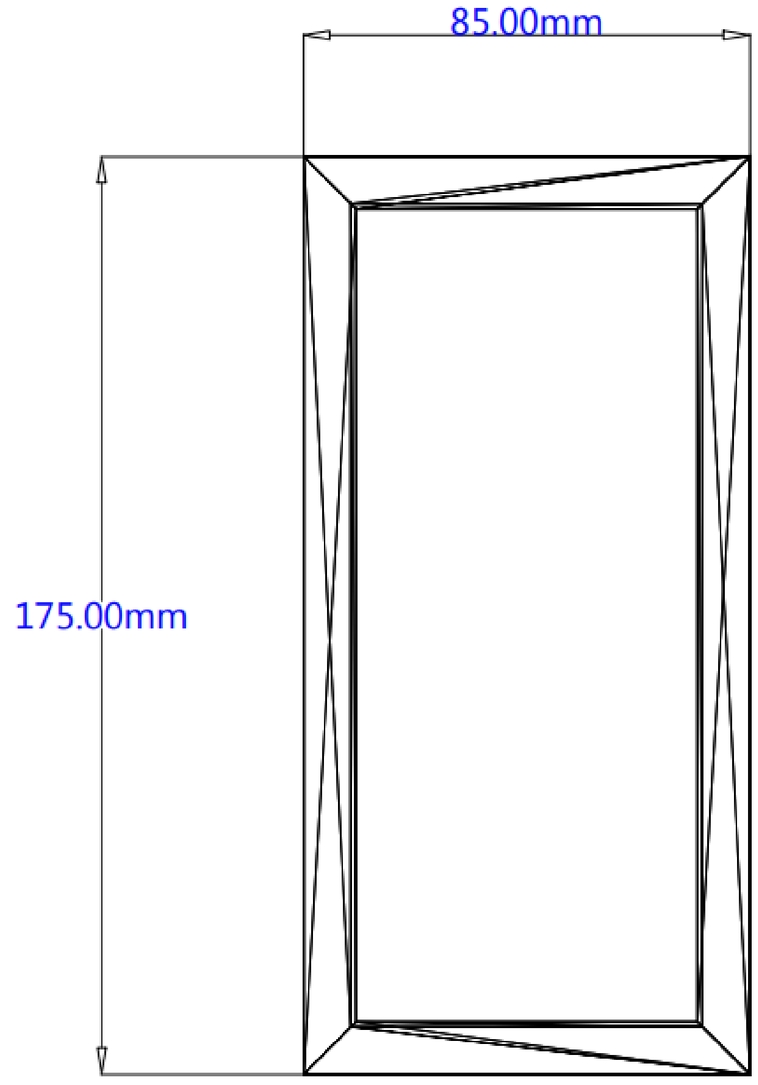
	FECHA	NOMBRE	FIRMA	ESCUELA POLITÉCNICA SUPERIOR LINARES
DIBUJADO	21/06/2024	Elyssar Myrna Thabet		
COMPROBADO				

ESCALA: 10:1	Sistema domótico a escala con acceso remoto Marco salón 3	Nº PLANO 17 de 19
		SUSTITUYE A:
		SUSTITUIDO POR:



	FECHA	NOMBRE	FIRMA	ESCUELA POLITÉCNICA SUPERIOR LINARES
DIBUJADO	21/06/2024	Elyssar Myrna Thabet		
COMPROBADO				

ESCALA: 10:1	Sistema domótico a escala con acceso remoto Marco puerta salón	Nº PLANO 18 de 19
		SUSTITUYE A:
		SUSTITUIDO POR:



	FECHA	NOMBRE	FIRMA	ESCUELA POLITÉCNICA SUPERIOR LINARES
DIBUJADO	21/06/2024	Elyssar Myrna Thabet		
COMPROBADO				

ESCALA: 10:1	Sistema domótico a escala con acceso remoto Piscina	Nº PLANO 19 de 19
		SUSTITUYE A:
		SUSTITUIDO POR: