GRADO EN INGENIERÍA EN TECNOLOGÍAS INDUSTRIALES

TRABAJO FIN DE GRADO

SISTEMA IOT PARA CONTROL DE AULAS, EQUIPO SERVIDOR

Autor: Joaquín Navarrete Sorela

Director: José Daniel Muñoz Frías

Madrid

Declaro, bajo mi responsabilidad, que el Proyecto presentado con el título

SISTEMA IOT PARA CONTROL DE AULAS, EQUIPO SERVIDOR

en la ETS de Ingeniería - ICAI de la Universidad Pontificia Comillas en el

curso académico 2022/23 es de mi autoría, original e inédito y

no ha sido presentado con anterioridad a otros efectos.

El Proyecto no es plagio de otro, ni total ni parcialmente y la información que ha sido

tomada de otros documentos está debidamente referenciada.

Fdo.: Joaquín Navarrete Sorela Fecha: 16/07/2023

Autorizada la entrega del proyecto

EL DIRECTOR DEL PROYECTO

Firmado por MUÑOZ FRIAS JOSE DANIEL - ***7384** el día 30/08/2023 con un certificado emitido por AC FNMT Usuarios

Fdo.: José Daniel Muñoz Frías Fecha: 22/07/2023

GRADO EN INGENIERÍA EN TECNOLOGÍAS DE TELECOMUNICACIÓN

TRABAJO FIN DE GRADO

SISTEMA IOT PARA CONTROL DE AULAS, EQUIPO SERVIDOR

Autor: Joaquín Navarrete Sorela

Director: José Daniel Muñoz Frías

Madrid

SISTEMA IOT PARA CONTROL DE AULAS, EQUIPO SERVIDOR

Autor: Navarrete Sorela, Joaquín.
Director: Muñoz Frías, José Daniel.
Entidad Colaboradora: ICAI – Universidad Pontificia Comillas

RESUMEN DEL PROYECTO

En este trabajo de fin de grado se ha desarrollado un servidor que integra un broker de MQTT, una aplicación web para mejorar la gestión del entorno en las aulas universitarias y se ha programado una cámara basada en el chip ESP32 para detectar el número de personas en un aula y la comunicación entre la aplicación y los sensores de las aulas.

Palabras clave: IOT, Monitorización y control, Entorno educativo

1. Introducción

El entorno académico ha experimentado una transformación significativa en los últimos años, debido a los avances tecnológicos y la creciente demanda de soluciones innovadoras que mejoren la experiencia de aprendizaje. En este contexto, este proyecto se posiciona en la vanguardia de la unión entre tecnología y educación, con el propósito de mejorar la situación en las aulas universitarias.

Las universidades y los centros educativos se enfrentan al desafío constante de brindar un entorno de estudio y trabajo óptimo, que promueva el bienestar de los alumnos y estimule su rendimiento académico. La calidad de la iluminación, la temperatura adecuada y la gestión eficiente de los recursos son elementos esenciales para crear un ambiente propicio para el aprendizaje. Sin embargo, en muchos casos, la monitorización y control se ha llevado a cabo de forma manual o poco eficiente, esto plantea la necesidad de desarrollar soluciones tecnológicas innovadoras que aborden estos desafíos y mejoren la gestión y el control de las aulas.

En respuesta a esta necesidad, nuestro proyecto propone la implementación de un servidor que integra un broker MQTT y una aplicación web. Este proyecto permitirá recopilar datos en tiempo real sobre distintos aspectos del entorno de las aulas como la temperatura, la iluminación o la presencia de personas, además brindará la capacidad de visualizar los datos de las aulas a la vez que se podrá controlar e interactuar con los equipos del aula, como la iluminación y la presencia de personas, a través de una aplicación web accesible tanto desde un dispositivo móvil como de un ordenador.

En un mundo cada vez más conectado y tecnológico, es esencial aprovechar las herramientas disponibles para mejorar la calidad de la educación. A través de este proyecto, buscamos marcar una diferencia tangible en el entorno de las aulas de la universidad, creando un ambiente de estudio más cómodo, adaptativo y conectado que potencie el rendimiento y el bienestar de los involucrados en el proceso educativo.

2. Definición del proyecto

Este proyecto consta de dos partes y por lo tanto de dos trabajos, en este trabajo concreto se va a desarrollar la parte del servidor, las comunicaciones entre el servidor y los distintos dispositivos y el desarrollo de la aplicación web. Para ello lo primero es diseñar y configurar el servidor, configurar el broker de MQTT que permitirá la comunicación eficiente entre los equipos del aula y la aplicación web. Se requiere instalar sensores y actuadores en las aulas de los que se recopilarán datos aunque no se verá en detalle este trabajo debido a que esto se abarca en la otra parte del proyecto.

A continuación se procede a la programación de la aplicación web para visualizar los datos recopilados por los sensores, dicha aplicación estará disponible tanto para dispositivos móviles como ordenadores cuya interfaz estará diseñada para que sea intuitiva para los usuarios.

Además de la aplicación web y del diseño del servidor, se va a programar una cámara que se usará para detectar el número de personas que se encuentran en una clase o simplemente para detectar si hay personas en el aula. La parte del proyecto que concierne a la cámara consiste en dos partes, la primera es realizar los ajustes necesarios para que la cámara pueda hacer fotos correctamente y pueda enviar y recibir información de la que se encarga este proyecto, la otra parte que se realiza en otro proyecto se encarga de analizar el número de personas que hay en el aula con las imágenes que se le proporcionan desde este trabajo.

3. Descripción del sistema

El sistema desarrollado en este proyecto se basa en la integración de un servidor MQTT y una aplicación web para la monitorización de las aulas universitarias. A continuación, se proporciona una descripción detallada de cada componente:

- Servidor MQTT: El servidor MQTT actúa como un intermediario o broker entre los dispositivos IoT y la aplicación web. Proporciona un canal de comunicación eficiente y confiable para el intercambio de mensajes. Este servidor se encarga de recibir los mensajes enviados por los sensores de las aulas y distribuirlos a los clientes conectados, como la aplicación web. Ver Ilustración 1 - Esquema del sistema IoT.
- Aplicación web: La aplicación web es el principal punto de acceso para los usuarios, permitiéndoles monitorear y controlar el estado de las aulas universitarias. A través de la interfaz intuitiva de la aplicación web, los usuarios pueden visualizar en tiempo real información relevante, como la temperatura, la iluminación y la presencia en las aulas. Ver Ilustración 2.

El sistema en su conjunto se basa en la arquitectura cliente-servidor, donde los dispositivos IoT, como los sensores y la cámara, actúan como clientes que envían información al servidor MQTT. A su vez, la aplicación web se conecta al servidor MQTT para recibir y mostrar los datos en una interfaz amigable para el usuario.

La herramienta utilizada para el desarrollo de este proyecto incluye tecnologías y lenguajes como Apache, PHP, MQTT y el ESP32 con su correspondiente entorno de programación (Espressif IDE). Estas herramientas proporcionan los recursos necesarios para implementar

y conectar los diferentes componentes del sistema, permitiendo una comunicación efectiva y una interfaz de usuario intuitiva.

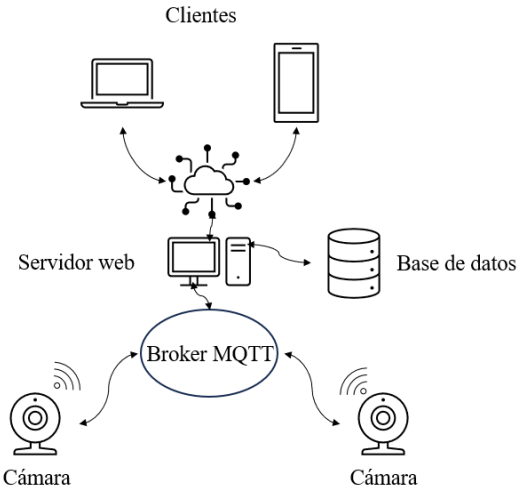


Ilustración 1 - Esquema del sistema IoT

4. Resultados

Los resultados obtenidos en este proyecto han sido altamente satisfactorios y han demostrado la efectividad y utilidad del sistema desarrollado para la monitorización y control de las aulas universitarias. A continuación, se presentan los principales resultados alcanzados:

- Implementación exitosa del sistema: Se logró implementar de manera exitosa el servidor MQTT, la aplicación web y la cámara basada en ESP32. El sistema se encuentra parcialmente funcional y operativo, permitiendo la monitorización en tiempo real de los diferentes parámetros del entorno de las aulas universitarias.
- Visualización de datos en tiempo real: La aplicación web desarrollada proporciona una interfaz amigable y accesible que permite a los usuarios visualizar en tiempo real los datos capturados por los sensores, como temperatura, iluminación y presencia en las aulas. Esta capacidad de visualización en tiempo real brinda a los usuarios una mayor comprensión y conocimiento del estado de las aulas.
- Captura y envío de imágenes: La programación de la cámara basada en ESP32 ha permitido la captura de imágenes y su envío a través del protocolo MQTT. Esto proporciona una perspectiva visual adicional sobre el entorno de las aulas, lo que facilita una monitorización más completa y detallada.
- Mejora de la eficiencia y comodidad en el entorno educativo: Los resultados obtenidos con este proyecto han demostrado una mejora significativa en la eficiencia y comodidad en el entorno de las aulas universitarias. La capacidad de monitorización en tiempo real y el control remoto de los equipos han contribuido a una gestión más eficiente de los recursos y a una mayor comodidad para los estudiantes y profesores.

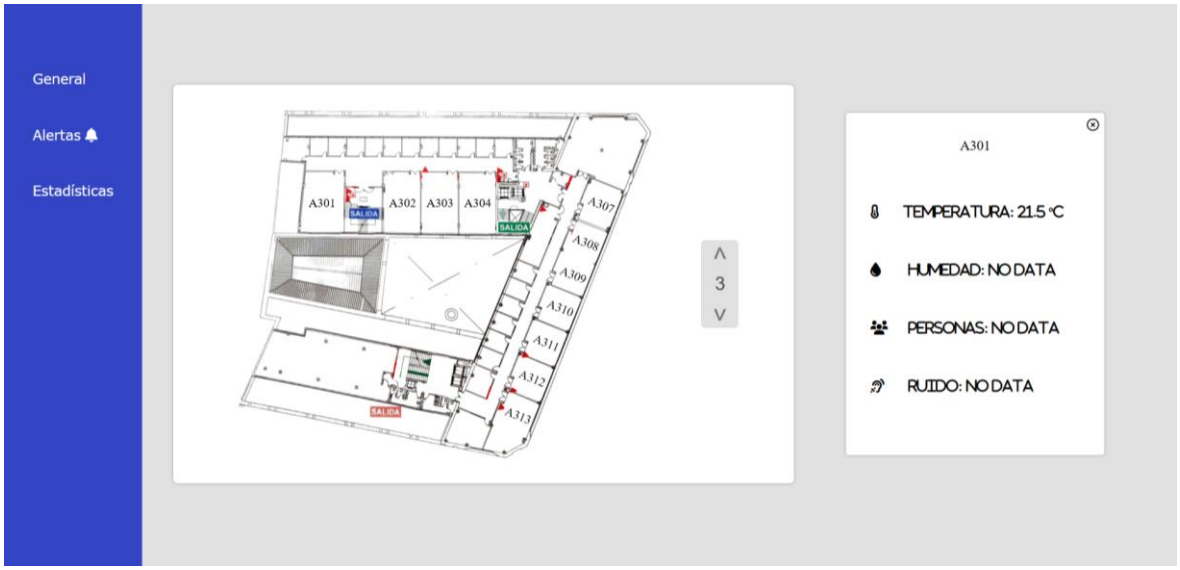


Ilustración 2 – Resultado final de aplicación web

5. Conclusiones

En conclusión, los resultados obtenidos en este proyecto han demostrado la efectividad y utilidad del sistema desarrollado para la monitorización y control de las aulas universitarias. Los logros incluyen la implementación exitosa del sistema, la visualización en tiempo real de datos, la captura y envío de imágenes, y una mejora general en la eficiencia y comodidad del entorno educativo. Estos resultados respaldan la relevancia y aplicabilidad del proyecto en el ámbito educativo.

IOT SYSTEM FOR CLASSROOM CONTROL, SERVER SIDE

Author: Navarrete Sorela, Joaquín.
Supervisor: Muñoz Frías, José Daniel.
Collaborating entity: ICAI – Universidad Pontificia Comillas

ABSTRACT

In this end-of-degree Project, a server has been developed that integrates an MQTT broker, a web application to improve environment management in university classrooms, and a camera based on the ESP32 chip has been programmed to detect the number of people in a classroom and the communication between the web application and the classroom sensors.

Keywords: IOT, Monitoring and control, Educational environment

1. Introduction

The academic environment has undergone a significant transformation in recent years, due to technological advances and the growing demand for innovative solutions that improve the learning experience. In this context, this project is positioned at the forefront of the union between technology and education, with the purpose of improving the situation in university classrooms.

Universities and educational centers face the constant challenge of providing an optimal study and work environment that promotes the well-being of students and stimulates their academic performance. Quality lighting, adequate temperature and efficient resource management are essential elements to create an environment conducive to learning. However, in many cases, monitoring and control has been carried out manually or inefficiently, this raises the need to develop innovative technological solutions that address these challenges and improve classroom management and control.

In response to this need, our project proposes the implementation of a server that integrates an MQTT broker and a web application. This project will allow data to be collected in real time on different aspects of the classroom environment such as temperature, lighting or the presence of people, and will also provide the ability to view classroom data while being able to control and interact with the classrooms. classroom equipment, such as lighting and the presence of people, through a web application accessible from both a mobile device and a computer.

In an increasingly connected and technological world, it is essential to take advantage of the tools available to improve the quality of education. Through this project, we seek to make a tangible difference in the university classroom environment, creating a more comfortable, adaptive and connected study environment that enhances the performance and well-being of those involved in the educational process.

2. Project definition

This project consists of two parts and therefore two works, in this specific work the server part, the communications between the server and the different devices and the development of the web application will be developed. To do this, the first thing is to design and configure the server, configure the MQTT broker that will allow efficient communication between the classroom equipment and the web application. It is required to install sensors and actuators in the classrooms from which data will be collected, although this work will not be seen in detail because this is covered in the other part of the project.

Next, the web application is programmed to view the data collected by the sensors. This application will be available for both mobile devices and computers whose interface will be designed to be intuitive for users.

In addition to the web application and the server design, a camera will be programmed that will be used to detect the number of people in a class or simply to detect if there are people in the classroom. The part of the project that concerns the camera consists of two parts, the first is to make the necessary adjustments so that the camera can take photos correctly and can send and receive information that this project is responsible for, the other part that is done in Another project is responsible for analyzing the number of people in the classroom with the images provided from this work.

3. System description

The system developed in this project is based on the integration of an MQTT server and a web application for monitoring university classrooms. Below is a detailed description of each component:

- MQTT Server: The MQTT server acts as an intermediary or broker between the IoT devices and the web application. Provides an efficient and reliable communication channel for exchanging messages. This server is in charge of receiving the messages sent by the classroom sensors and distributing them to the connected clients, such as the web application. See Ilustración 1 - Esquema del sistema IoT.
- Web application: The web application is the main access point for users, allowing them to monitor and control the status of university classrooms. Through the intuitive interface of the web application, users can view relevant information in real time, such as temperature, lighting and presence in the classrooms. See Ilustración 2.

The entire system is based on client-server architecture, where IoT devices such as sensors and camera act as clients that send information to the MQTT server. In turn, the web application connects to the MQTT server to receive and display the data in a user-friendly interface.

The tool used for the development of this project includes technologies and languages such as Apache, PHP, MQTT and ESP32 with its corresponding programming environment (Espressif IDE). These tools provide the necessary resources to

implement and connect the different components of the system, allowing effective communication and an intuitive user interface.

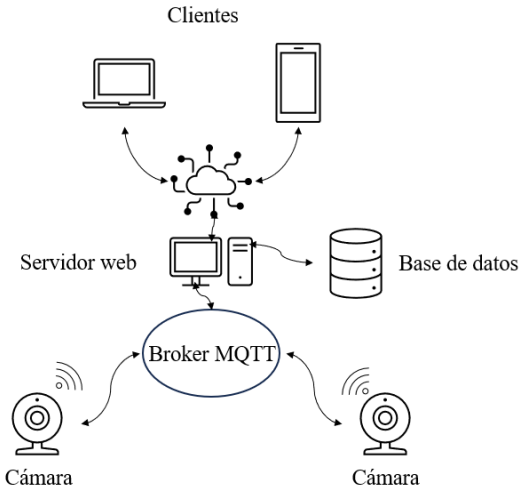


Ilustración 3 – IoT System scheme

4. Resultados

The results obtained in this project have been highly satisfactory and have demonstrated the effectiveness and usefulness of the system developed for the monitoring and control of university classrooms. Below are the main results achieved:

- Successful implementation of the system: The MQTT server, the web application and the camera based on ESP32 were successfully implemented. The system is partially functional and operational, allowing real-time monitoring of the different parameters of the university classroom environment.
- Real-time data visualization: The developed web application provides a friendly and accessible interface that allows users to visualize in real time the data captured by the sensors, such as temperature, lighting and presence in the classrooms. This real-time viewing capability gives users greater understanding and knowledge of classroom status.
- Capturing and sending images: The programming of the ESP32-based camera has allowed the capture of images and their sending through the MQTT protocol. This provides an additional visual perspective on the classroom environment, facilitating more complete and detailed monitoring.
- Improvement of efficiency and comfort in the educational environment: The results obtained with this project have shown a significant improvement in efficiency and comfort in the environment of university classrooms. The real-time monitoring capacity and remote control of equipment have contributed to more efficient management of resources and greater comfort for students and teachers.

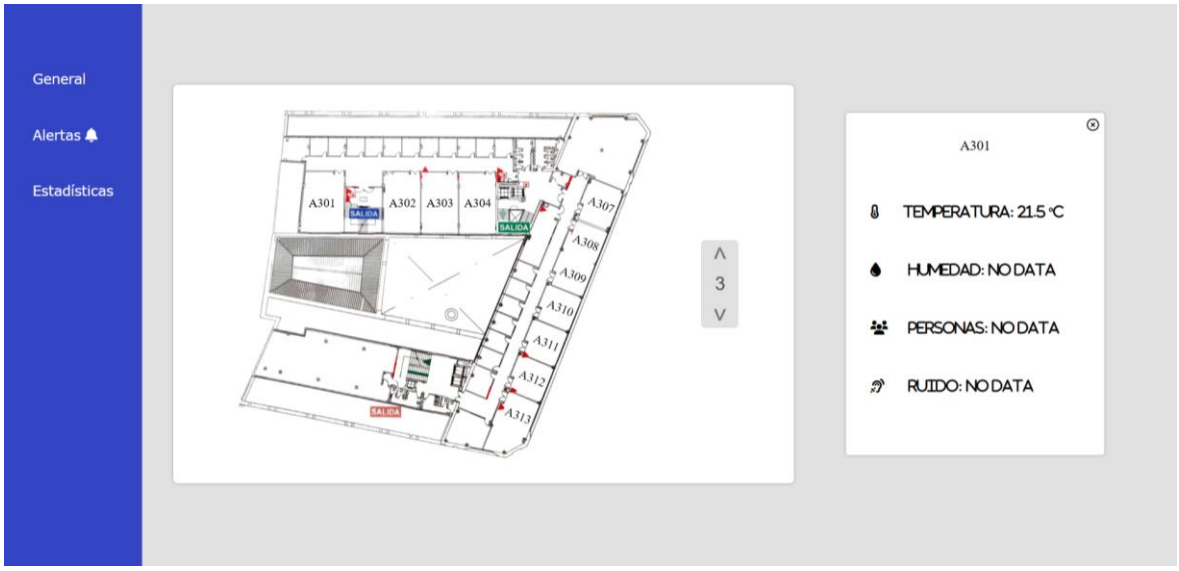


Ilustración 4 – Web application final result

5. Conclusions

In conclusion, the results obtained in this project have demonstrated the effectiveness and usefulness of the system developed for the monitoring and control of university classrooms. Achievements include successful system implementation, real-time data visualization, image capture and delivery, and an overall improvement in the efficiency and comfort of the educational environment. These results support the relevance and applicability of the project in the educational field.

Índice de la memoria

Índice de figuras

Capítulo 1. INTRODUCCIÓN

En el contexto académico actual, el entorno de las aulas universitarias desempeña un papel fundamental en la experiencia educativa de estudiantes y profesores. La calidad del ambiente de estudio, que incluye aspectos como la temperatura, la iluminación y la climatización, puede influir significativamente en el rendimiento y bienestar de los involucrados. Por tanto, surge la necesidad de desarrollar soluciones tecnológicas que permitan monitorear y controlar estos parámetros ambientales, con el objetivo de mejorar la calidad del entorno y brindar una experiencia académica más favorable.

En este Trabajo de Fin de Grado, se presenta un proyecto innovador que busca abordar esta necesidad mediante la implementación de un servidor que integra un broker MQTT, una aplicación web y una cámara para saber el número de personas que hay en un aula. A través de esta solución, se pretende proporcionar a estudiantes y profesores un mayor control sobre su entorno de estudio, permitiéndoles ajustar los parámetros ambientales según sus necesidades individuales. Además, se busca optimizar el uso de recursos y promover prácticas sostenibles en el entorno educativo.

El presente trabajo tiene como objetivo principal analizar, diseñar e implementar esta solución tecnológica, evaluando su eficacia y su impacto en la experiencia académica. Se espera que este proyecto contribuya a mejorar la calidad del entorno en las aulas universitarias, brindando un ambiente de estudio más cómodo y propicio para el aprendizaje.

A través de este trabajo, se busca no sólo abordar un desafío relevante en el ámbito educativo, sino también explorar las posibilidades de aplicación de tecnologías innovadoras, como el Internet de las Cosas (IoT), en la mejora de la experiencia académica. Asimismo, se espera generar conocimiento y recomendaciones que puedan ser de utilidad para futuras investigaciones y desarrollos en este campo.

Este proyecto no solo se limita a mejorar la calidad del entorno de las aulas universitarias, sino que también abarca una serie de implicaciones sociales, económicas y tecnológicas de gran relevancia.

En primer lugar, desde una perspectiva social, el entorno educativo desempeña un papel fundamental en el bienestar y el rendimiento académico de los estudiantes. Al proporcionar un ambiente de estudio más cómodo y adaptado, este proyecto busca contribuir a la formación de profesionales más preparados y motivados. Además, al promover prácticas sostenibles en el entorno educativo, se fomenta una conciencia ambiental entre los estudiantes, cultivando así una generación más comprometida con la preservación del medio ambiente.

En cuanto a la dimensión económica, la implementación de soluciones tecnológicas que optimicen el uso de recursos, como la energía, puede generar beneficios significativos. Al reducir los costes operativos asociados al consumo energético en las aulas universitarias, este proyecto contribuye a la eficiencia económica de las instituciones educativas. Además, al promover prácticas de uso responsable de los recursos, se generan ahorros a largo plazo y se establece una base sólida para la gestión sostenible de los recursos en el ámbito educativo.

En términos tecnológicos, este proyecto se sitúa en la vanguardia de la convergencia entre el Internet de las Cosas (IoT) y la educación. La implementación de un servidor que integra un broker MQTT y una aplicación web representa una aplicación práctica de tecnologías innovadoras en el entorno educativo. Esto no solo mejora la experiencia académica, sino que también impulsa la adopción y el desarrollo de soluciones tecnológicas en otros campos de estudio.

De igual manera, es importante destacar que este proyecto tiene un impacto más allá del ámbito educativo. Al mejorar la experiencia académica y promover prácticas sostenibles, se contribuye a la construcción de una sociedad más educada, equitativa y consciente del medio ambiente. Además, la implementación de soluciones tecnológicas innovadoras en el entorno educativo abre nuevas oportunidades para la investigación y el desarrollo de tecnologías aplicadas a la educación, promoviendo así la innovación y el progreso en este campo.

Capítulo 2. DESCRIPCIÓN DE LAS TECNOLOGÍAS

Durante el desarrollo de este proyecto, se emplearán diversas tecnologías, protocolos y herramientas específicas para su implementación y funcionamiento adecuado. A continuación, se describen algunas de las principales:

2.1 SERVIDOR APACHE

Apache es un popular servidor web de código abierto utilizado para alojar y servir sitios web en Internet. Fue creado en 1995 y ha sido ampliamente adoptado debido a su estabilidad, flexibilidad y seguridad. Apache se destaca por su capacidad para gestionar un gran número de solicitudes simultáneas y por su compatibilidad con múltiples sistemas operativos, incluyendo Linux, Windows y macOS.

Una de las características más destacadas de Apache es su modularidad. El servidor web está compuesto por un núcleo principal llamado Apache HTTP Server, que se encarga de la gestión básica de las solicitudes y respuestas HTTP. A través de módulos, Apache puede ser ampliado y personalizado para adaptarse a las necesidades específicas de cada proyecto.

Apache es compatible con una amplia gama de tecnologías web, incluyendo el lenguaje de programación PHP (es el lenguaje que se va a utilizar en este proyecto), Python, Perl, y muchos más. Además, soporta protocolos como HTTP, HTTPS, FTP, entre otros. Esto permite a los desarrolladores implementar diversas aplicaciones web, sistemas de gestión de contenidos (CMS) y soluciones de comercio electrónico, entre otros tipos de sitios web.

La configuración de Apache se realiza a través de archivos de configuración que especifican cómo se deben manejar las solicitudes entrantes, los directorios y archivos a los que se puede acceder y otras opciones de personalización. Apache también ofrece herramientas de seguridad y autenticación para proteger los sitios web y garantizar la confidencialidad y la integridad de los datos transmitidos.

2.2 LENGUAJE PHP

PHP (acrónimo de Hypertext Preprocessor), es un lenguaje de programación de código abierto ampliamente utilizado en el desarrollo web. Diseñado específicamente para la creación de aplicaciones web dinámicas, PHP se ejecuta en el lado del servidor, lo que significa que el código PHP se procesa en el servidor antes de enviar la respuesta al navegador web del cliente.

Una de las ventajas destacadas de PHP es su facilidad de uso y aprendizaje. Su sintaxis está diseñada para ser clara y legible y basada en el léxico de C, lo que facilita a los desarrolladores escribir y mantener el código. Además, PHP es un lenguaje altamente flexible que permite la fácil integración de fragmentos de código HTML y CSS dentro de los scripts PHP, lo que facilita la creación de páginas web dinámicas y la mezcla de contenido estático y dinámico.

Además, PHP ofrece una sólida integración con diversos sistemas de gestión de bases de datos, como MySQL. Esto permite a los desarrolladores conectarse a bases de datos, ejecutar consultas SQL, recuperar resultados y administrar la conexión a la base de datos de manera eficiente.

Otra característica destacada de PHP es su capacidad para manejar formularios web. Proporciona funciones y utilidades para capturar y validar los datos enviados por los usuarios a través de formularios. Esto facilita el procesamiento de la entrada del usuario y la implementación de lógica de validación para garantizar la integridad de los datos.

PHP también es compatible con la programación orientada a objetos (POO), lo que permite a los desarrolladores organizar su código en clases y objetos reutilizables. Esto promueve una estructura de código modular, mejor mantenibilidad y permite una mayor reutilización de código.

2.3 PROTOCOLO MQTT

MQTT (Message Queuing Telemetry Transport) es un protocolo de mensajería ligero diseñado para la comunicación eficiente y confiable entre dispositivos en redes de sensores o máquinas conectadas. Fue desarrollado por IBM en la década de 1990 y se ha convertido en un estándar ampliamente utilizado en el Internet de las Cosas (IoT) y en aplicaciones de M2M (machine-to-machine).

El objetivo principal de MQTT es proporcionar una forma eficiente de transmitir mensajes en redes con ancho de banda limitado, alta latencia o baja confiabilidad, como redes móviles, satelitales o con conexiones intermitentes. MQTT utiliza un modelo de publicación/suscripción, donde los dispositivos se dividen en dos roles principales: publicadores y suscriptores.

En el modelo de publicación/suscripción, los publicadores son los dispositivos que envían los mensajes y los suscriptores son los dispositivos que reciben los mensajes. Estos dispositivos se comunican a través de un intermediario conocido como broker MQTT, que es responsable de recibir y distribuir los mensajes entre los suscriptores adecuados.

Algunas características clave del protocolo MQTT incluyen:

- Ligereza: MQTT es un protocolo ligero que utiliza un encabezado de mensaje simple y un tamaño reducido de payload, lo que lo hace eficiente en términos de ancho de banda y consumo de energía.
- Calidad de servicio (QoS): MQTT ofrece tres niveles de calidad de servicio para garantizar la entrega de mensajes según las necesidades del usuario. Los niveles de QoS van desde la entrega "al menos una vez" hasta la entrega "exactamente una vez", proporcionando diferentes niveles de confiabilidad.
- Persistencia de sesiones: MQTT permite mantener la persistencia de las sesiones, lo que significa que los suscriptores pueden recibir mensajes que se enviaron mientras no estaban conectados. Esto garantiza que no se pierda ningún mensaje importante.

- Temas (topics): MQTT utiliza el concepto de temas para organizar los mensajes. Los suscriptores pueden suscribirse a un tema específico o a varios temas utilizando patrones de coincidencia, lo que les permite recibir los mensajes que son relevantes para ellos.
- Seguridad: MQTT es compatible con varias opciones de seguridad, como el uso de cifrado TLS/SSL para la comunicación segura y la autenticación basada en certificados o nombres de usuario/contraseña para garantizar la autorización y autenticidad de los dispositivos.

MQTT ha demostrado ser una solución efectiva para la comunicación en entornos donde los recursos son limitados o la conexión es poco confiable. Su diseño ligero, su capacidad de garantizar la entrega de mensajes y su flexibilidad en términos de calidad de servicio hacen que sea una elección popular en aplicaciones de IoT y M2M.

2.4 BROKER MQTT MOSQUITTO

Mosquitto es un broker MQTT de código abierto y ligero desarrollado por la Fundación Eclipse. Actúa como intermediario en la comunicación entre los dispositivos que utilizan el protocolo MQTT en una red. Como broker MQTT, Mosquitto es responsable de recibir los mensajes publicados por los dispositivos y distribuirlos a los suscriptores correspondientes de manera eficiente.

Algunas características destacadas de Mosquitto incluyen:

- Ligereza: Mosquitto está diseñado para ser liviano y eficiente en términos de uso de recursos, lo que lo hace adecuado para implementaciones en dispositivos con recursos limitados, como sistemas embebidos o microcontroladores.

- Soporte para MQTT: Mosquitto implementa el protocolo MQTT en su totalidad, lo que le permite interactuar de manera compatible con otros dispositivos y aplicaciones MQTT en la red.
- Escalabilidad: Mosquitto es capaz de manejar un gran número de conexiones simultáneas y suscripciones, lo que lo convierte en una opción adecuada para implementaciones en escenarios de alto rendimiento.
- Seguridad: Mosquitto ofrece opciones de seguridad para proteger la comunicación MQTT. Es compatible con el uso de TLS/SSL para cifrar la comunicación y asegurarla, lo que proporciona confidencialidad y autenticación de los dispositivos.
- Configuración flexible: Mosquitto permite una configuración flexible para adaptarse a los requisitos específicos de la implementación. Se pueden establecer políticas de acceso y control de seguridad, configurar diferentes niveles de calidad de servicio (QoS) y gestionar la persistencia de los mensajes según sea necesario.

Además de estas características, Mosquitto también proporciona herramientas y utilidades adicionales, como la capacidad de generar registros detallados para facilitar el diagnóstico y la depuración de problemas en la red MQTT.

En resumen, Mosquitto es un broker MQTT de código abierto y ligero que se utiliza como intermediario en la comunicación entre dispositivos que utilizan el protocolo MQTT. Con su diseño ligero, capacidad de escalabilidad y soporte para seguridad, Mosquitto es una opción popular para implementaciones MQTT, permitiendo una comunicación eficiente y confiable en redes de IoT y M2M.

2.5 ESPRESSIF IDE

Espressif IDE es un entorno de desarrollo integrado (IDE) especialmente diseñado para la programación de dispositivos basados en microcontroladores ESP8266 y ESP32 de Espressif Systems. Es una herramienta completa que facilita la escritura, compilación y carga de firmware en estos dispositivos.

Espressif IDE ofrece una interfaz de usuario intuitiva y amigable que permite a los desarrolladores trabajar de manera eficiente en sus proyectos. Proporciona una amplia gama de características y funcionalidades para simplificar el proceso de desarrollo, incluyendo:

- Editor de código: Espressif IDE cuenta con un editor de código integrado que brinda resaltado de sintaxis y otras características útiles, como la indentación automática, completado de código y búsqueda rápida.
- Compilador y herramientas de compilación: El IDE incorpora un compilador y herramientas de compilación específicas para los microcontroladores ESP8266 y ESP32. Esto permite a los desarrolladores compilar su código de manera eficiente y generar el firmware correspondiente.
- Gestión de proyectos: Espressif IDE facilita la gestión de proyectos al proporcionar opciones para crear, abrir y organizar proyectos. Los desarrolladores pueden trabajar en varios proyectos simultáneamente y mantener los archivos y configuraciones relacionados en un lugar organizado.
- Depuración y monitoreo: El IDE ofrece funcionalidades para depurar y monitorear el funcionamiento del firmware en tiempo real. Los desarrolladores pueden establecer puntos de interrupción, rastrear variables y observar el comportamiento del código mientras se ejecuta en el dispositivo.
- Carga de firmware: Espressif IDE permite la carga de firmware directamente en los microcontroladores ESP8266 y ESP32, facilitando la programación de los dispositivos y la prueba de los programas desarrollados.

Además de estas características, Espressif IDE también ofrece documentación detallada, ejemplos de código y una comunidad activa de desarrolladores que comparten conocimientos y experiencias.

En resumen, Espressif IDE es un entorno de desarrollo integrado (IDE) específicamente diseñado para programar dispositivos basados en microcontroladores ESP8266 y ESP32. Proporciona un conjunto completo de herramientas y funcionalidades que facilitan el desarrollo de firmware para estos dispositivos, incluyendo editor de código, compilador, gestión de proyectos, depuración y carga de firmware. Con su interfaz intuitiva y recursos adicionales, Espressif IDE es una herramienta valiosa para los desarrolladores que trabajan en proyectos basados en microcontroladores de Espressif Systems, como la parte de este trabajo relacionada con la cámara.

En resumen, este proyecto se basará en tecnologías como el servidor Apache, el lenguaje PHP y el protocolo MQTT para el desarrollo de la aplicación web y la comunicación con los dispositivos del entorno del aula. Además, se utilizará el Espressif IDE para programar una cámara y habilitar su integración en el sistema. Estas tecnologías y herramientas proporcionarán una base sólida para la implementación exitosa del proyecto y la creación de un entorno educativo mejorado y eficiente.

Capítulo 3. ESTADO DE LA CUESTIÓN

En el ámbito de este proyecto, se han desarrollado diversas soluciones y se han llevado a cabo numerosas investigaciones relacionadas con la monitorización y control de entornos educativos. Estos trabajos existentes brindan un panorama amplio y enriquecedor que nos permite evaluar el estado actual de la tecnología, así como los avances en la investigación y desarrollo de proyectos similares.

En el mercado, es posible encontrar soluciones comerciales que abordan aspectos específicos de la monitorización y control de aulas universitarias. Estas soluciones van desde sistemas de gestión energética que permiten el control eficiente de la iluminación y la climatización, hasta sistemas de seguridad y control de acceso que incluyen cámaras de vigilancia y sistemas de identificación biométrica. Estas soluciones comerciales están diseñadas para mejorar la eficiencia operativa, reducir los costos energéticos y mejorar la seguridad en el entorno educativo.

Además de las soluciones comerciales, la investigación académica ha desempeñado un papel fundamental en el avance de la monitorización y control de aulas universitarias. Numerosos estudios se han enfocado en aspectos como la calidad del aire, la temperatura, la presencia de personas y la calidad de la iluminación en entornos educativos. Estos estudios han explorado diferentes técnicas de sensores, sistemas de adquisición de datos y algoritmos de control para optimizar la eficiencia energética y mejorar la comodidad de los estudiantes y profesores.

En este contexto, es esencial realizar un análisis exhaustivo de los trabajos de investigación existentes para evaluar qué resultados se han obtenido y qué enfoques se han utilizado. Este análisis permitirá identificar las contribuciones que otros proyectos han realizado con relación a los objetivos que se persiguen en este proyecto en particular.

La revisión de los trabajos existentes también ayuda a identificar posibles brechas o limitaciones en las soluciones actuales, lo que a su vez brinda una oportunidad para desarrollar una solución más integral y personalizada que aborde los desafíos específicos del entorno universitario.

En el contexto del estado del arte de este proyecto, **Quodus** es una solución relevante que ha surgido como una opción destacada en el campo de la monitorización y control de aulas universitarias. Se trata de un sistema integral que combina tecnología IoT (Internet de las cosas) y análisis de datos para ofrecer una gestión inteligente y eficiente de los entornos educativos.

Quodus se distingue por su enfoque innovador y su capacidad para recopilar datos en tiempo real sobre diferentes parámetros de las aulas, como temperatura, iluminación, calidad del aire, ocupación y consumo de energía. Utilizando una amplia variedad de sensores conectados a través del protocolo MQTT, Quodus proporciona una visión detallada y actualizada del estado de las aulas, lo que permite a los usuarios tomar decisiones informadas y realizar ajustes necesarios para optimizar el entorno educativo.

Además de la monitorización, Quodus ofrece funcionalidades avanzadas de control y automatización. Permite a los usuarios interactuar con los equipos del aula, como el control de la iluminación y la climatización, a través de una interfaz intuitiva y accesible desde dispositivos móviles u ordenadores. Esto brinda a los usuarios la capacidad de adaptar el entorno según las necesidades de los estudiantes, mejorando su comodidad y experiencia de aprendizaje.

Otra característica destacada de Quodus es su capacidad para analizar los datos recopilados y proporcionar información útil para la toma de decisiones. Mediante técnicas de análisis de datos y aprendizaje automático, Quodus puede identificar patrones, tendencias y sugerir mejoras en el rendimiento energético y la eficiencia de las aulas.

En el contexto del estado del arte, Quodus se posiciona como una solución integral y avanzada que aborda los desafíos de la monitorización y control de aulas universitarias. Su

enfoque basado en IoT, su capacidad para recopilar datos en tiempo real, su interfaz intuitiva y su capacidad de análisis de datos lo convierten en una opción relevante y atractiva para mejorar la eficiencia y la experiencia en el entorno educativo.

Es importante tener en cuenta que la inclusión de Quodus en el estado del arte no implica que sea necesariamente la solución adoptada en este proyecto en particular. Sin embargo, su mención en el contexto del estado del arte destaca su relevancia y contribución al campo, y puede servir como referencia y punto de partida para evaluar las características y funcionalidades que se pueden considerar en el desarrollo de este proyecto. Aparte de Quodus no se ha encontrado ninguna otra solución relevante para este trabajo.

En resumen, la revisión de los trabajos y soluciones existentes en el ámbito de la monitorización y control de aulas universitarias proporciona una base sólida para comprender el estado actual del campo. Este análisis permite contextualizar y justificar la necesidad de desarrollar un proyecto propio, identificar las contribuciones únicas que se pueden realizar y establecer cómo se diferenciará de las soluciones y trabajos previos. Con esta información, se sientan las bases para avanzar en la siguiente etapa del proyecto y desarrollar una solución innovadora y eficaz que aborde las necesidades específicas del entorno universitario.

Capítulo 4. DEFINICIÓN DEL TRABAJO

4.1 JUSTIFICACIÓN

A la vista de todos los trabajos previos realizados y el análisis crítico realizado, se pueden identificar varias razones claras y justificadas para el desarrollo de este proyecto. En primer lugar, a pesar de que existen soluciones comerciales y trabajos de investigación en el ámbito de la monitorización y control de aulas universitarias, se han identificado limitaciones y brechas en las soluciones existentes. Estas limitaciones están relacionadas con la falta de integración de diferentes aspectos del entorno educativo, la complejidad de las soluciones disponibles y la falta de adaptabilidad a los requisitos específicos de cada institución educativa.

Además, se ha observado la necesidad de una solución integral que combine tecnología IoT, análisis de datos y control de equipos en un único sistema. Este proyecto busca abordar una visión más completa que permita no solo monitorizar, sino también interactuar y controlar diferentes parámetros del entorno de las aulas, aunque de momento solo se va a realizar la parte de monitorización ya que para poder controlar distintos parámetros habría que cambiar drásticamente la estructura en la que está el sistema de climatización debido a que se trata de un control analógico.

Otro factor que justifica el desarrollo de este proyecto es la evolución tecnológica y las oportunidades que ofrece la utilización de protocolos como MQTT, el uso de servidores Apache y la programación en lenguaje PHP. Estas tecnologías y herramientas permiten una comunicación eficiente, una gestión de datos robusta y una interfaz de usuario accesible desde diferentes dispositivos, lo que brinda una plataforma sólida para la implementación del proyecto.

Además, se ha identificado la importancia de crear un entorno educativo más eficiente, cómodo y sostenible. El proyecto busca optimizar el consumo de energía, mejorar la

comodidad de los estudiantes y promover una experiencia de aprendizaje más adecuada. Estos aspectos son fundamentales en el contexto actual, donde la eficiencia energética y la calidad del entorno educativo son consideraciones clave.

Este proyecto presenta una oportunidad única y valiosa para aquellos que buscan mejorar la eficiencia y la calidad del entorno educativo en las aulas universitarias. A continuación, destacaremos las razones por las cuales alguien podría querer invertir en este proyecto y adquirirlo:

- Innovación tecnológica: El proyecto combina tecnologías de vanguardia, como el protocolo MQTT, el uso de servidores Apache y la programación en lenguaje PHP, para crear una solución avanzada y eficiente en la monitorización y control de aulas universitarias. Esto implica una ventaja competitiva en el mercado y la oportunidad de liderar el avance en este campo.
- Solución integral: A diferencia de las soluciones existentes, este proyecto ofrece una solución integral que aborda de manera simultánea la monitorización, el control y la interacción con diferentes aspectos del entorno de las aulas universitarias, como la temperatura, la iluminación y los equipos de climatización. Esto proporciona una gestión eficiente y centralizada, optimizando el consumo de energía y mejorando la comodidad de los estudiantes.
- Mejora de la experiencia educativa: El proyecto tiene como objetivo crear un entorno educativo más cómodo y propicio para el aprendizaje. Al monitorear y controlar de manera inteligente los parámetros del entorno, los estudiantes pueden disfrutar de una experiencia educativa más adecuada y centrada en sus necesidades. Esto puede conducir a un mayor compromiso estudiantil, un mejor rendimiento académico y una satisfacción general entre los usuarios.

- Eficiencia energética y sostenibilidad: La gestión inteligente de los recursos energéticos es un aspecto crucial en el contexto actual. Este proyecto ofrece la oportunidad de reducir el consumo de energía al optimizar la iluminación y la

climatización de las aulas universitarias, lo que resulta en ahorros significativos a largo plazo. Además, la integración de tecnologías IoT y la capacidad de análisis de datos permiten identificar patrones de consumo y optimizar aún más la eficiencia energética.

- Potencial de escalabilidad: Este proyecto no solo es aplicable a aulas universitarias, sino que también puede extenderse a otros entornos educativos, como escuelas, institutos o centros de formación. Además, las tecnologías utilizadas y la modularidad del sistema permiten una fácil adaptación y expansión a diferentes requisitos y configuraciones.

Este proyecto ofrece una solución innovadora, integral y eficiente para la monitorización y control de aulas universitarias, con el potencial de mejorar la experiencia educativa, ahorrar energía y ofrecer una ventaja competitiva en el mercado. Al invertir en este proyecto, los clientes o emprendedores tendrán la oportunidad de liderar el avance en este campo, contribuir al desarrollo de un entorno educativo más cómodo y sostenible, y obtener beneficios económicos a largo plazo.

4.2 OBJETIVOS

Los objetivos del proyecto son las metas específicas que se persiguen con su realización. A continuación, se presentan los principales objetivos que se pretenden alcanzar:

- Desarrollar un servidor que integre un broker MQTT: El primer objetivo es implementar un servidor que actúe como broker MQTT, proporcionando una plataforma robusta y confiable para la comunicación entre los dispositivos IoT y la aplicación web. Este servidor será el encargado de recibir y distribuir los mensajes de manera eficiente, garantizando una comunicación fluida y segura.
- Crear una aplicación web accesible desde dispositivos móviles y ordenadores: El segundo objetivo es desarrollar una aplicación web intuitiva y de fácil acceso que permita a los usuarios monitorear y controlar el estado de las aulas universitarias. A través de esta aplicación, los usuarios podrán visualizar información en tiempo real

sobre parámetros como la temperatura, la iluminación, la presencia y realizar ajustes en los equipos del aula, como la iluminación y la climatización.

- Programar una cámara basada en el chip ESP32 para capturar imágenes y enviarlas al servidor a través del protocolo MQTT para más tarde procesar esas imágenes y saber el número personas que hay en ese momento. La detección del número de personas que hay no se realiza en este trabajo. Sin tener en cuenta la detección de personas, se desarrollará el código necesario en lenguaje C para capturar imágenes de manera periódica o bajo demanda.
- Integrar sensores para la monitorización del entorno de las aulas: El tercer objetivo es implementar sensores adecuados para la monitorización de diferentes aspectos del entorno de las aulas, como la temperatura, la humedad y la iluminación. Estos sensores recopilarán datos en tiempo real y los enviarán al servidor MQTT, que los distribuirá a la aplicación web para su visualización y análisis. Esta integración de los sensores no se realiza en este proyecto, en este trabajo se recibe directamente la información de los sensores a través de MQTT.
- Garantizar la seguridad y privacidad de los datos: El quinto objetivo es asegurar la seguridad y privacidad de los datos transmitidos y almacenados en el sistema. Se implementarán medidas de seguridad, como la autenticación de usuarios, para proteger la integridad y confidencialidad de la información.

En resumen, los objetivos del proyecto son desarrollar un servidor con un broker MQTT, crear una aplicación web accesible para monitorear el entorno de las aulas universitarias, integrar sensores para la monitorización, permitir la interacción con los equipos del aula y garantizar la seguridad de los datos. Estos objetivos se enfocan en mejorar la eficiencia, la comodidad y la sostenibilidad en el entorno educativo, brindando a los usuarios una herramienta integral y de calidad para la gestión de las aulas.

4.3 METODOLOGÍA

Para lograr los objetivos establecidos en el proyecto, se seguirá un camino estructurado y metodológico que incluye las siguientes etapas:

- Análisis de requisitos: En esta etapa inicial, se llevará a cabo un análisis exhaustivo de los requisitos del proyecto. Se identificarán y definirán de manera precisa los parámetros y características que se deben monitorizar y controlar en el entorno de las aulas universitarias. Además, se establecerán los criterios de rendimiento, las funcionalidades requeridas y las especificaciones técnicas necesarias.
- Diseño del sistema: Con base en los requisitos establecidos, se procederá a diseñar la arquitectura del sistema. Esto implica definir la estructura del servidor MQTT, la configuración de los sensores, la interfaz de la aplicación web y la programación de la cámara. Se realizará una planificación detallada de las etapas de desarrollo, teniendo en cuenta las interdependencias entre los diferentes componentes del proyecto.
- Desarrollo del servidor y la aplicación web: En esta etapa, se implementará el servidor MQTT y se desarrollará la aplicación web. Se utilizarán herramientas y tecnologías como Apache, PHP y MQTT para construir un sistema robusto y escalable. Se garantizará la comunicación adecuada entre el servidor y los dispositivos IoT, así como la correcta visualización de los datos en la interfaz de la aplicación web.
- Integración de los sensores y la cámara: Se procederá a la integración de los sensores necesarios para la monitorización del entorno de las aulas, como los sensores de temperatura, iluminación y presencia. Asimismo, se programará la cámara basada en el ESP32 para la captura de imágenes. Se asegurará la correcta comunicación y sincronización de los datos entre los sensores, la cámara y el servidor MQTT.

- Pruebas y depuración: Una vez que el sistema esté desarrollado, se llevarán a cabo pruebas exhaustivas para verificar su funcionamiento y rendimiento. Se realizarán pruebas de conexión, pruebas de comunicación MQTT, pruebas de respuesta y pruebas de estabilidad. Cualquier problema o error identificado será depurado y corregido.

- Implementación y despliegue: Una vez finalizadas las pruebas y depuraciones, el sistema estará listo para su implementación y despliegue en el entorno de las aulas universitarias. Se realizará la instalación física de los sensores y se configurarán los equipos necesarios. Se garantizará la seguridad y privacidad de los datos mediante medidas adecuadas de autenticación y cifrado.
- Evaluación y optimización: Se realizará una evaluación continua del sistema implementado para identificar posibles mejoras y optimizaciones. Se recopilarán comentarios y retroalimentación de los usuarios finales para realizar ajustes y actualizaciones según sea necesario. Esto asegurará que el proyecto alcance su máximo potencial y cumpla con los objetivos establecidos.

Siguiendo este camino de desarrollo, desde el análisis de requisitos hasta la implementación y evaluación del sistema, se asegurará la consecución exitosa de los objetivos establecidos en el proyecto. Cada etapa se realizará de manera ordenada y metódica, garantizando la calidad y eficiencia del resultado final.

4.4 PLANIFICACIÓN Y ESTIMACIÓN ECONÓMICA

A continuación, se presenta la estructura temporal de las actividades a realizar en el desarrollo del proyecto. Es importante tener en cuenta que los plazos pueden variar dependiendo de las circunstancias específicas del proyecto, la disponibilidad de recursos y otros factores. Esta estimación puede requerir ajustes adicionales según las necesidades o contratiempos que puedan llegar a surgir en el proyecto.

- Análisis de requisitos (3 semanas)
  - Recopilación y documentación de requisitos del proyecto.
  - Definición de funcionalidades y características del sistema.

- Diseño del sistema (4 semanas)
  - Diseño arquitectónico del sistema y estructura de la base de datos.
  - Diseño de la interfaz de usuario y experiencia de usuario.

- Implementación (10 semanas)
  - Desarrollo del servidor MQTT y configuración del broker.
  - Desarrollo de la aplicación web y la interfaz de usuario.
  - Programación de la cámara basada en ESP32 para la captura y envío de imágenes.

- Pruebas y depuración (3 semanas)
  - Realización de pruebas unitarias y de integración.
  - Identificación y solución de errores y fallos.
  - Ajustes y mejoras del sistema basados en los resultados de las pruebas.

- Despliegue (1 semana)
  - Instalación física de los sensores y configuración del entorno.
  - Configuración de la seguridad y privacidad de los datos.

- Mantenimiento y optimización (continuo)
  - Realización de actualizaciones y mejoras según las necesidades y retroalimentación.

Estimación del coste de desarrollo:

La estimación del coste de desarrollo depende principalmente del coste de la cámara y la amortización del equipo utilizado, en este caso un ordenador personal. El precio de la cámara en el momento de su compra fue de 20€, por otro lado para la amortización del equipo se ha tenido en cuenta que la vida útil del ordenador utilizado es de 8 años aproximadamente y teniendo en cuenta el precio del ordenador y el tiempo invertido en el proyecto sale una amortización de 75€. En cuanto a costes debidos a licencias y softwares,

en este proyecto no los hay debido a que los softwares utilizados son de código abierto disponibles de forma gratuita.
Teniendo en cuenta estos costes, el coste total es de 95€ sin tener en cuenta mano de obra debido a que se trata de un proyecto realizado por un estudiante y se ha considerado el tiempo dedicado al trabajo como una inversión en aprendizaje.

Capítulo 5. DESARROLLO DE SERVIDOR APACHE

El uso de un servidor Apache se debe a que para desarrollar una aplicación web se necesita un servidor, se utilizaría el servidor de la universidad para implantar este trabajo pero debido a que hay que desarrollar la aplicación web, es más sencillo instalar un servidor en el ordenador para hacer todas las pruebas necesarias y una vez comprobado que todo funciona correctamente, implantar el trabajo en el servidor de la universidad finalmente.

5.1 INSTALACIÓN DE APACHE

Es importante comentar que la instalación se va a explicar en detalle para Windows, para el resto de sistemas operativos se trata de un proceso similar, si se deseara instalar en otro sistema operativo habría que seguir las instrucciones que se encuentran en la página de descargas de **Apache**. Para comenzar la instalación es necesario acceder a la página web de **Apache**, una vez dentro se hace clic en el apartado *"Files for Microsoft Windows"* en la última versión disponible como se muestra a continuación:

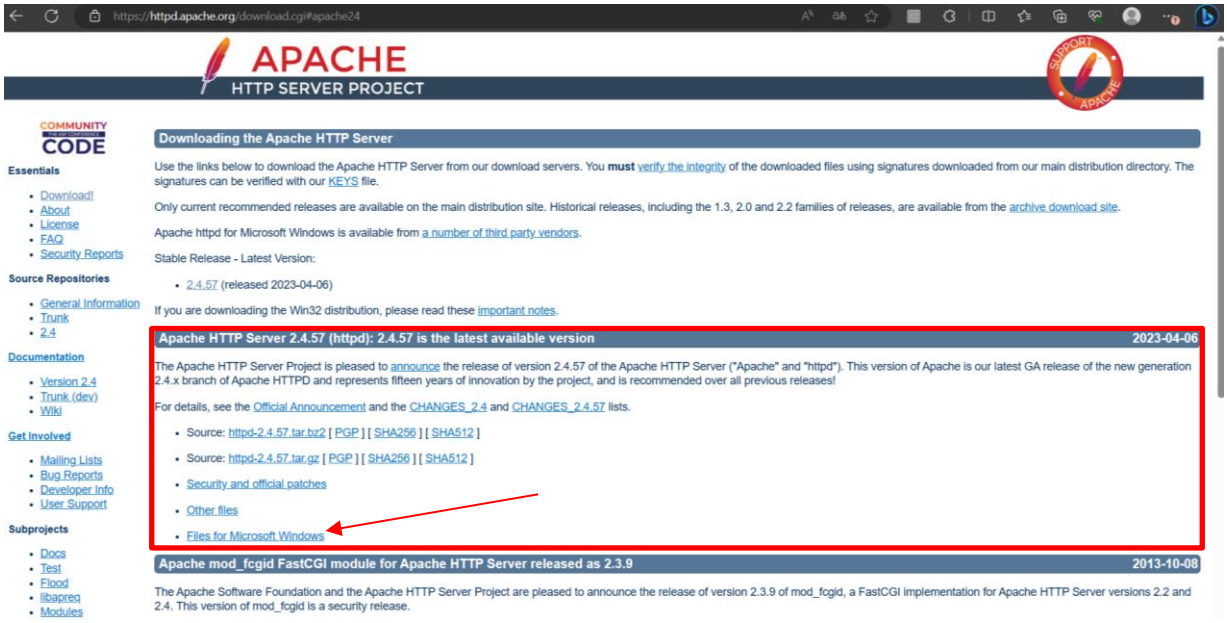


Figura 1: Primer paso instalación Apache

En la siguiente pantalla que aparece hacer clic en la opción *"Apache Lounge"* dentro de la sección *"Downloading Apache for Windows"*. Después de eso hacer clic en la última versión de apache disponible como se muestra a continuación:

Descargar la carpeta comprimida en la ubicación deseada, en el caso de este trabajo se recomienda instalarlo en (C:). A continuación se pasa a configurar ciertos archivos para que funcione correctamente.

5.2 CONFIGURACIÓN DEL SERVIDOR

Una vez instalada la carpeta llamada *"Apache24"*, entrar y abrir el archivo "*httpd.conf*", ubicado en la carpeta *"conf"* dentro de la carpeta *"Apache24"*. En la línea 38 hay que poner la ruta donde se encuentra la carpeta "conf" como se muestra a continuación:

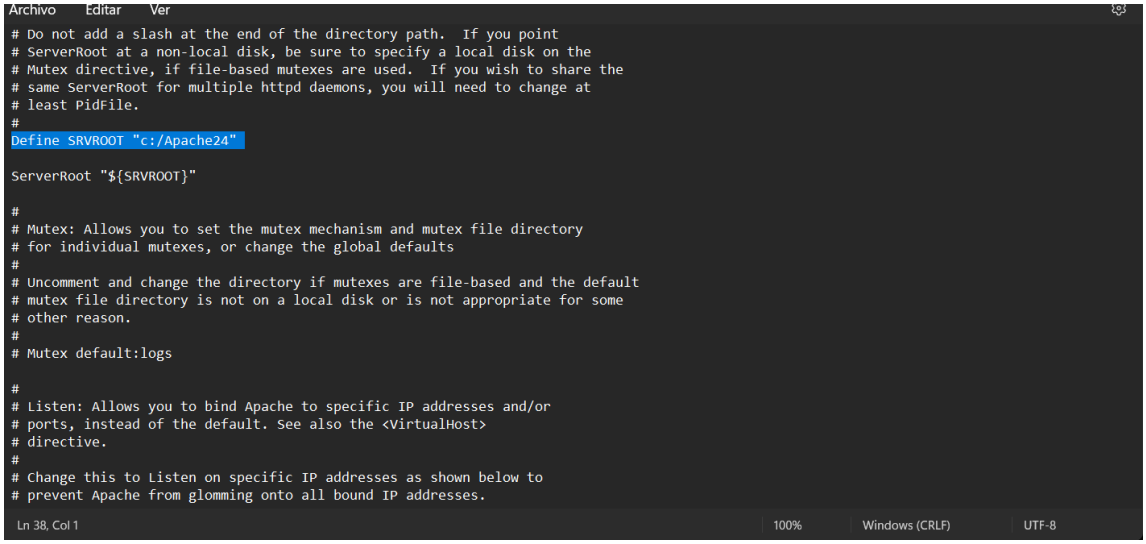
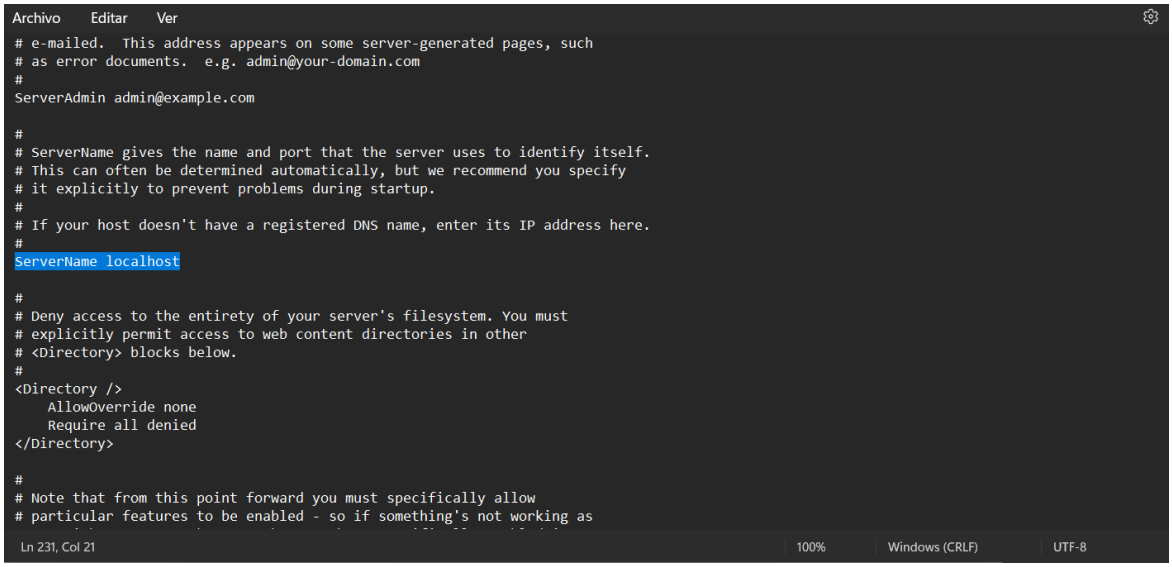


Figura 2: Definir ruta en "httpd.conf"

El siguiente cambio se encuentra en la línea 231, en la que hay que descomentar donde pone ServerName y cambiarle el nombre si se desea. Este cambio no es realmente necesario pero para evitar warnings al arrancar el servidor es recomendable. En la siguiente figura se muestra donde hay que realizar el cambio:



Una vez hechos estos cambios se puede cerrar el archivo *"httpd.conf"*. El último paso para arrancar el servidor es abrir el símbolo de sistema como administrador, a continuación dirigirse al directorio en el que se encuentra la carpeta de "Apache24", luego acceder a la carpeta *"bin"* y escribir el siguiente comando: httpd -k install -n "El nombre que se le quiera dar al servidor", a continuación se muestran visualmente los pasos.

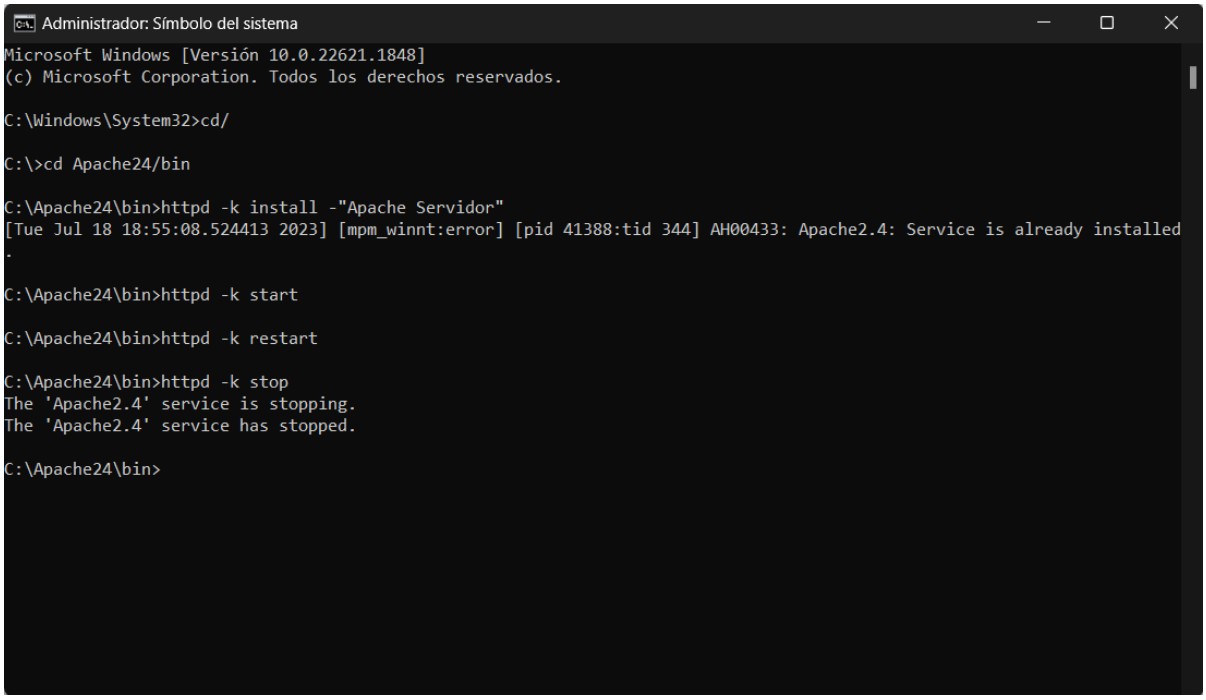


Figura 3: Pasos para activar el servidor en el símbolo del sistema

En la figura superior sale un error debido a que el servidor ya ha sido instalado anteriormente y también se muestran los comandos para arrancar, reiniciar y parar el servidor.

Capítulo 6. LENGUAJE PHP

Para poder realizar la aplicación web es necesario poder programar en el lenguaje PHP y que el servidor sea capaz de procesarlo. Por defecto el servidor Apache que se ha instalado no viene con PHP implementado por lo que hay que instalarlo y posteriormente implementarlo en el servidor para ya poder comenzar a programar la aplicación web.

6.1 INSTALACIÓN PHP

Para instalar PHP en Windows hay que descargar la última versión de PHP en su **página oficial** en formato zip, a continuación extraer el contenido en una carpeta de elección propia, en este caso "C:\php", después de eso renombrar el archivo "php.ini-development" a "php.ini" como se muestra a continuación:

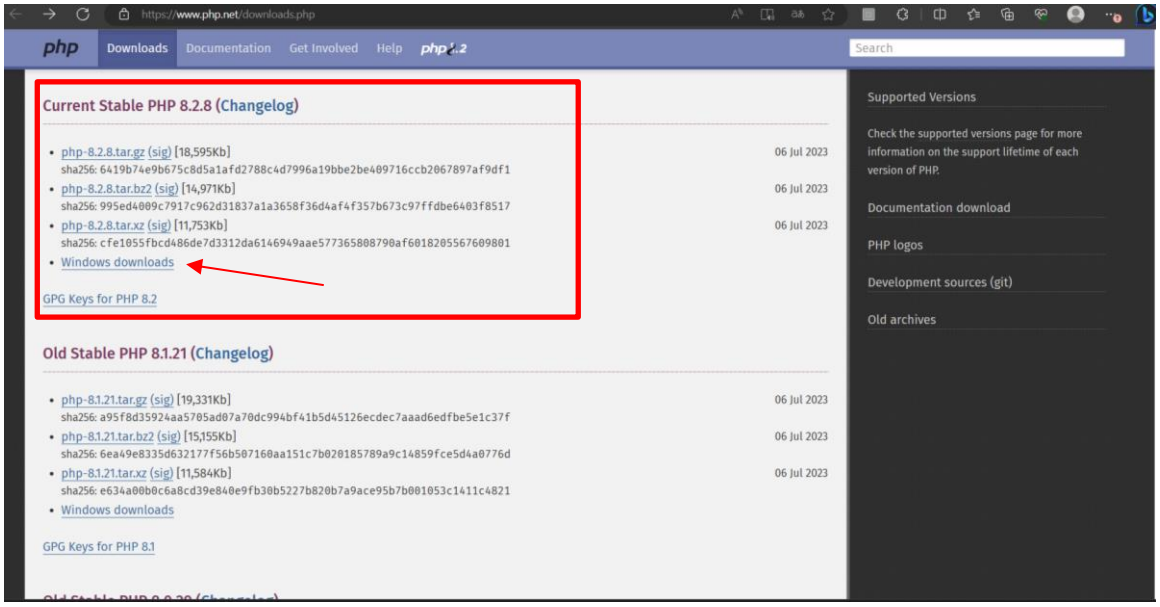
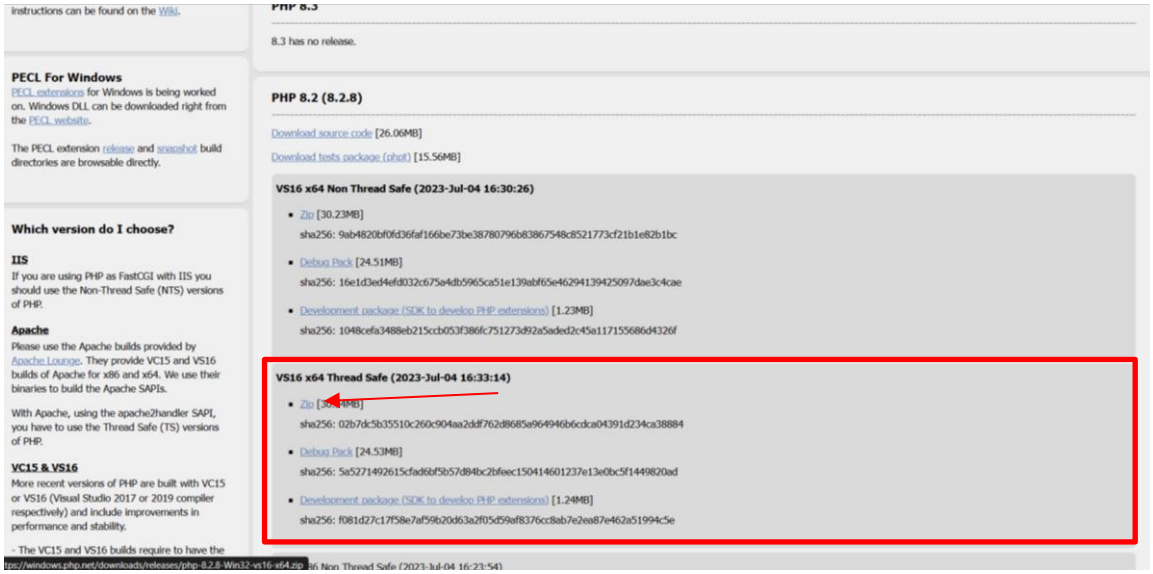


Figura 4: Como descargar la última versión de php paso 1



6.2 CONFIGURACIÓN PHP

Una vez se ha renombrado el archivo a “php.ini” hay que realizar unos cambios dentro del archivo, el primero es en la línea 380 y se trata de donde pone Off reemplazarlo por On como se muestra a continuación, esto se hace para que si ocurre un fallo en la página web no se filtre información sensible al usuario.

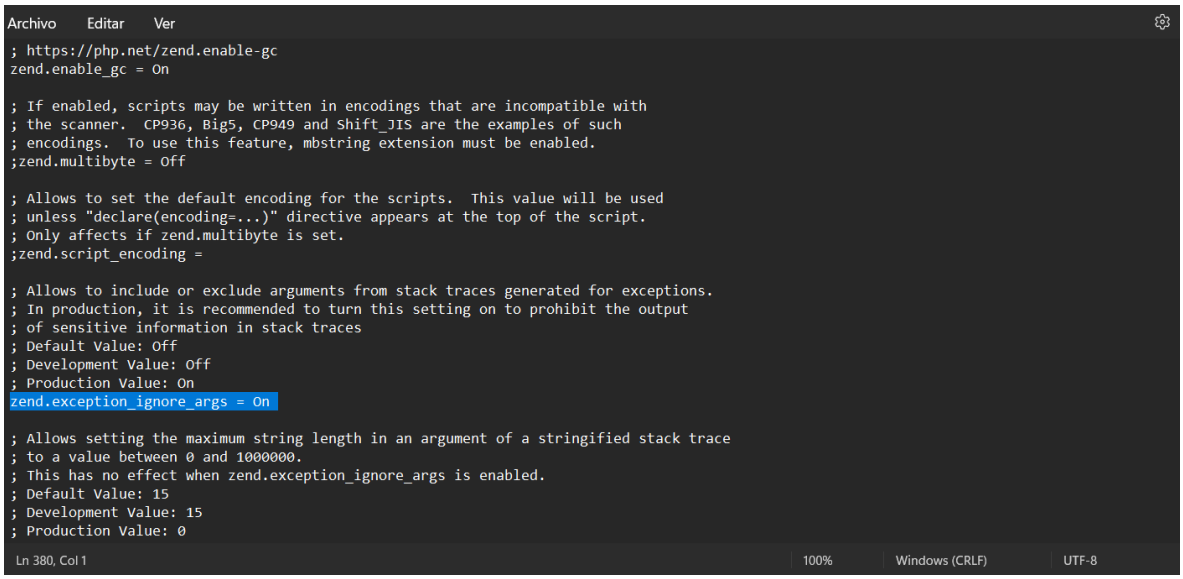


Figura 5: Primer cambio en archivo "php.ini"

Justo debajo en la línea 387 cambiar el valor de 15 a 0. En la línea 489 modificar el valor de la variable error_reporting que en origen era: "E_ALL" a "E_ALL & ~E_DEPRECATED & ~E_STRICT" como aparece en la línea 487. Esto se hace para que se muestren los errores de código mientras se está desarrollando la aplicación en la propia web.

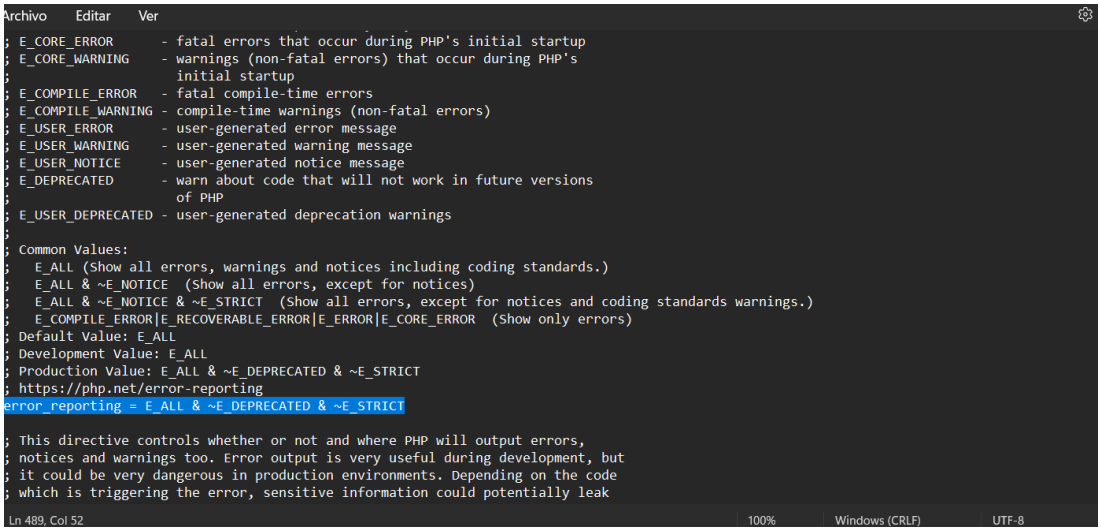


Figura 6: Configuración archivo "php.ini" paso 2

En la línea 515 modificar el valor a Off para evitar que se filtren detalles de configuración cuando ocurra algún error al arrancar.

Los últimos cambios que hay que realizar están relacionados con las extensiones que se quieran habilitar y se encuentran en las líneas 923, 933, 935 y 939 que únicamente consiste en borrar el punto y coma que hay al principio de cada línea como se muestra a continuación:

Figura 7: Configuracion "php.ini" cambios en extensiones

El siguiente paso después de cerrar el archivo php.ini habiendo modificado lo anterior sería agregar la ruta de la carpeta PHP a la variable de entorno “PATH” del sistema para que se puedan ejecutar comandos PHP desde cualquier ubicación de la línea de comandos, para ello se abre el explorador de archivos y se hace clic derecho en “Este equipo” y luego hacer clic en propiedades como se muestra a continuación:

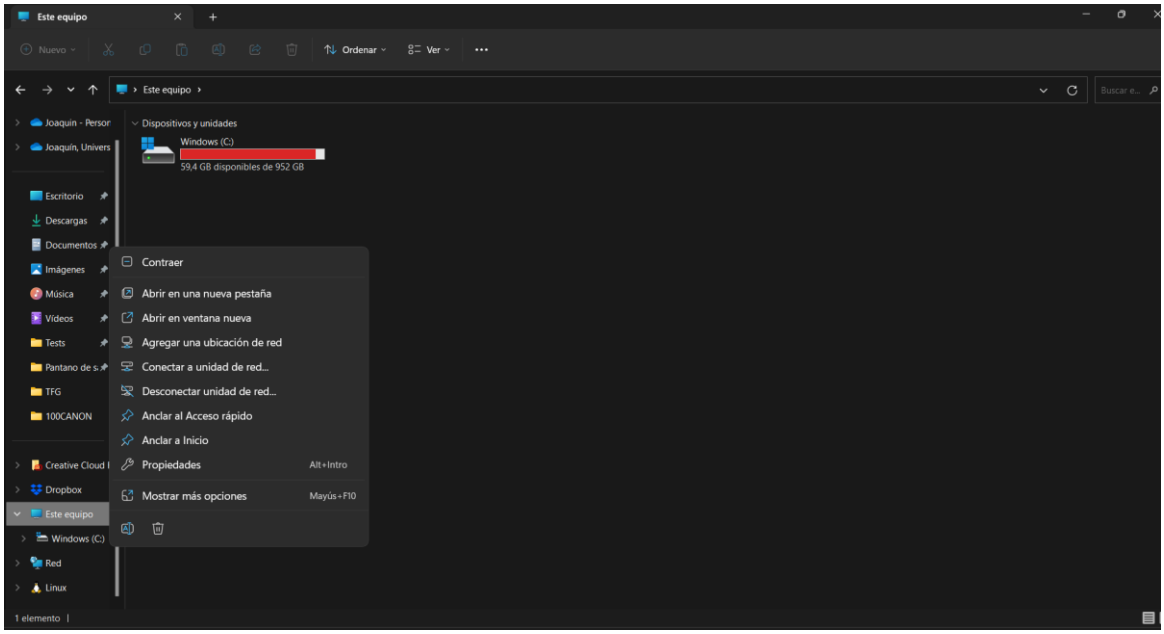


Figura 8: Configuración php a variable de entorno path, paso 1

Después de hacer clic en propiedades se abre la ventana de configuración, hacer clic en "configuración avanzada del sistema" y seguidamente se hace clic en "variables de entorno":

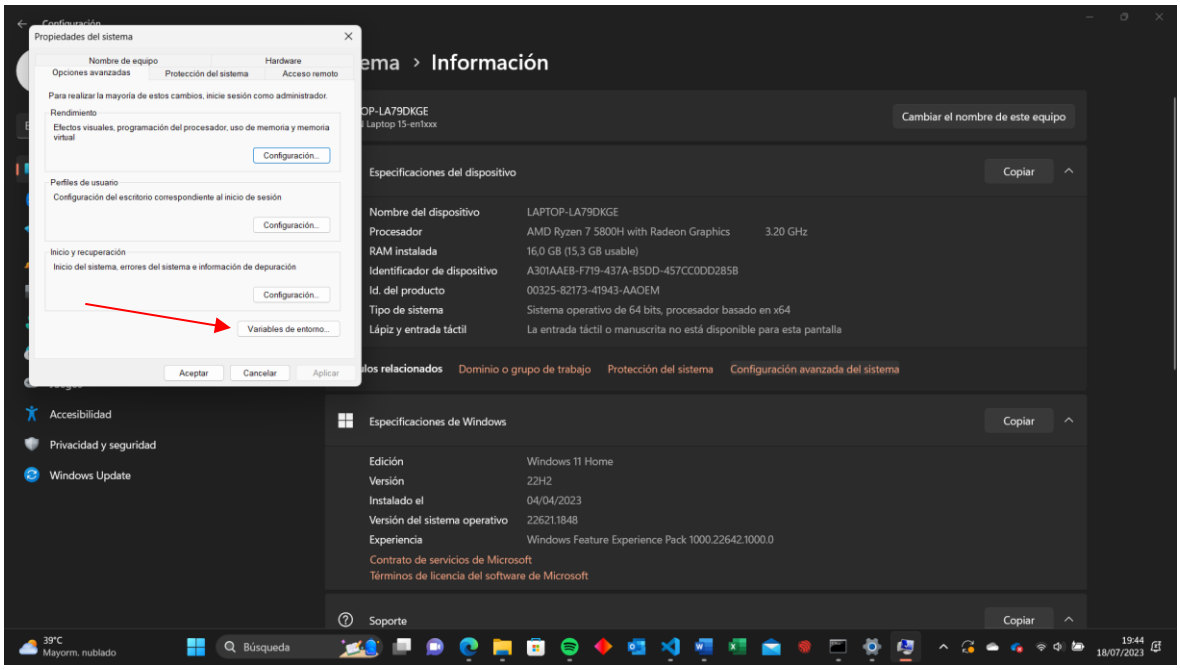
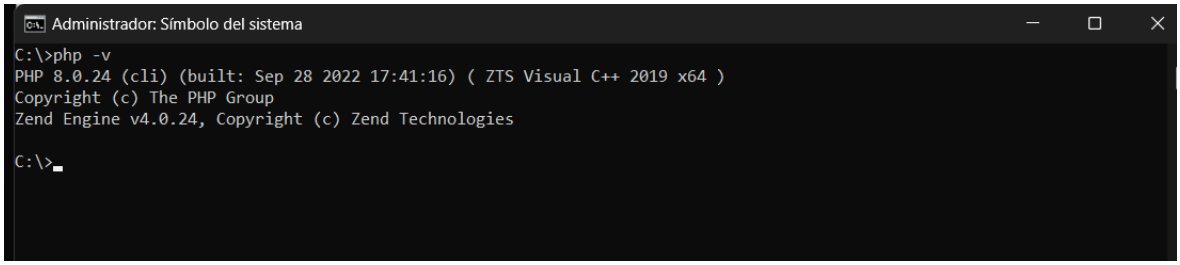


Figura 9: Configuración php a variable de entorno path, paso 2

A continuación en la ventana de variables de entorno, se hace clic en la variable "path" y a continuación clic en "editar", se abre una nueva ventana en la que hay que hacer clic en "nuevo" y escribir la ruta a la carpeta donde se encuentran los archivos descargados anteriormente de PHP, en este caso "C:\php".

Para comprobar que se ha instalado correctamente basta con abrir el símbolo de sistema y escribir php -v como se muestra a continuación verificando que no hay errores.



6.3 IMPLEMENTACIÓN PHP EN APACHE

El último paso para configurar PHP está relacionado con el servidor Apache y es que hay que realizar un pequeño cambio en el archivo "httpd.conf" en la carpeta "conf" de la carpeta "Apache24". Estos cambios se realizan en la línea 187, justo después de todos los LoadModule que aparecen, hay que añadir unas líneas al archivo que son:

```
LoadModule php_module "c:/php/php8apache2_4.dll"
AddType Application/x-httpd-php .php
PHPIniDir "c:/php" #en esta línea poner la ruta a la carpeta donde se ha
#descargado PHP
```

Esto añade el módulo de php al servidor ya que sin esto no va a funcionar correctamente la aplicación web.

Una vez realizados estos cambios hay que reiniciar el servidor para que se efectúen los cambios y poder programar la aplicación web.

Capítulo 7. DESARROLLO DE APLICACIÓN WEB

Para desarrollar la aplicación web se ha utilizado el software gratuito VisualStudio Code, la estructura de los documentos de esta aplicación es la siguiente:

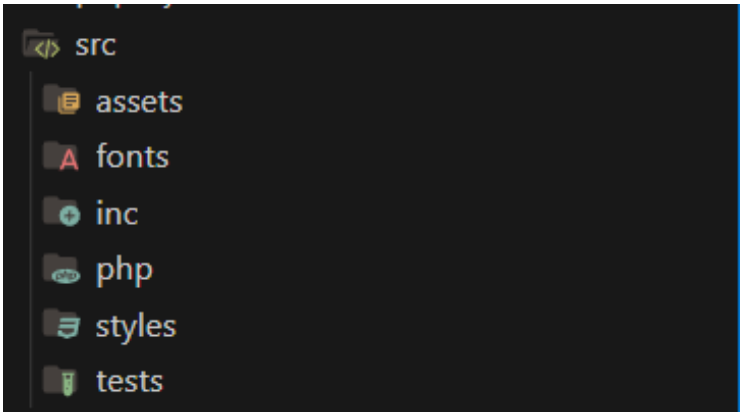


Figura 10: Estructura de aplicación web

En la carpeta de "*assets*" se encuentran las imágenes de los planos de las plantas 3, 4 y 5 del edificio de ICAI ya que estas son las que se han tenido en cuenta para el trabajo aunque en un futuro se pueden agregar más. La carpeta "*fonts*" contiene los tipos de letra que se van a utilizar en la aplicación, "*inc*" contiene el "header" de la aplicación que es la barra de navegación que se encuentra en la parte izquierda de la aplicación. En la carpeta "*php*" se encuentran todas las funciones que se han creado para esta aplicación para comunicarse con una base de datos en la que se almacenan los datos ambientales de las aulas para poder mostrarlas por pantalla si se requiere y funciones propias necesarias para el correcto funcionamiento de la aplicación. La carpeta "*styles*" contiene todos los archivos .css para que tenga buena apariencia la aplicación y por último la carpeta "*tests*" es la carpeta que contiene lo que es el código HTML y PHP que componen la aplicación.

7.1 *ESTRUCTURA VISUAL DE LA APLICACIÓN*

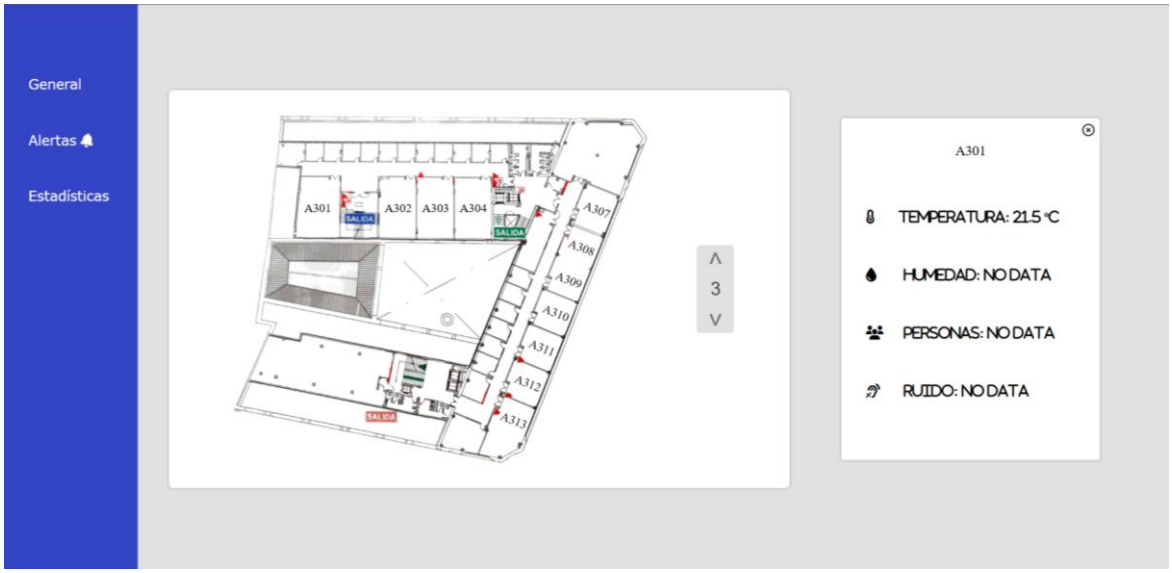


Figura 11: Estructura visual de la aplicación web

Como se puede apreciar en la figura superior, la página consiste en un "header" que es la franja azul en la que hay un menú para elegir entre general, alertas y estadísticas. En la pestaña de general se muestra el plano de las plantas de ICAI en el que se puede subir o bajar de planta utilizando las flechas entre las plantas 3 y 5, dentro de cada planta se puede seleccionar un aula para visualizar los datos de temperatura, humedad, personas y ruido aunque debido a que este proyecto se realiza en paralelo con otro que es el que se encarga de obtener estos valores, no se existen datos que se puedan mostrar así que en ese caso se muestra "NO DATA".

En las pestañas de alertas y estadísticas no aparece nada debido a que se necesitan datos para poder mostrar algún tipo de estadística o alerta. Lo único que se ha implementado en el menú de alertas es que cuando exista un error de algún tipo en el aula aparece una campana al lado de las alertas para llamar la atención y hacer saber que hay un fallo.

7.2 HEADER DE LA APLICACIÓN

Debido a que la franja azul del menú va a estar en todo momento presente en la aplicación, se ha decidido crear un archivo "header.php" que componga esa parte de la página utilizando el siguiente código:

```
<?php
require_once("../php/functions/functions.php");
require_once("../php/functions/db_connection.php");
?>
<!DOCTYPE html>
<script src="https://kit.fontawesome.com/a921d0c18e.js"
crossorigin="anonymous"></script>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet" href="../styles/menu_style.css">
    <link rel="stylesheet" href="../styles/content_style.css">
    <link rel="stylesheet" href="../styles/aulas_style.css">
    <link rel="stylesheet" href="../styles/info_style.css">

    <title>IOT</title>
</head>
<body>
    <div class="wrapper">
        <nav class="nav-container">
            <div class="menu">
                <form method="get">
                <ul>
                    <li><a href = "/src/tests/index.php" class="general"
name="general">General</a></li>
                    <br>
                    <li name="alertas"><a href="/src/tests/alertas.php"
class="alertas" name="alertas">Alertas</li>
                    <br>
                    <li><a href="/src/tests/estadisticas.php"
class="estadisticas" name="estadisticas">Estadísticas</a></li>
                </ul>
                </form>
            </div>
        </nav>
```

Este código hace que sea reutilizable para cualquier menú que se pueda llegar a seleccionar.

7.3 PESTAÑA GENERAL

La pestaña general es a la que se accede por defecto a la página, el documento en el que está definida esta parte se llama "index.php" y es la que se encarga de generar la parte de los planos de ICAI y sus respectivas aulas dependiendo de la planta en la que se encuentre, esto se realiza dinámicamente gracias a php, desde este archivo también se crea la sección de información que muestra los datos de las aulas. El código de este archivo es el siguiente:

```
<?php
session_start();
require_once("../php/functions/functions.php");
require_once("../php/functions/db_connection.php");
require('../inc/header.php');

CheckFloorUp($MAX_FLOOR);
CheckFloorDown($MIN_FLOOR);
?>
<script src="https://kit.fontawesome.com/a921d0c18e.js"
crossorigin="anonymous"></script>
<div class="main-content">
    <div class="planos">
        <div class=<?=$_SESSION['planta']?>>
            <img  id=<?=$_SESSION['planta']?>
src="../assets/<?=$_SESSION['planta']?>icai.jpg">
            <div class="aulas">
                <form method="get">
                <?php generate_floor($_SESSION['planta_actual']);?>

                </form>
            </div>
        </div>
        <div class="selector">
            <form method="post">
                <input type="submit" value=">" class="upbutton<?php
if($_SESSION["planta_actual"]==$MAX_FLOOR)echo($MAX_FLOOR);?>" name="add"/>
                <input type="text" readonly class="planta" name="planta_actual"
value="<?=$_SESSION["planta_actual"]?>" size="0"/>
                <input type="submit" value="<" class="downbutton<?php
if($_SESSION["planta_actual"]==$MIN_FLOOR)echo($MIN_FLOOR);?>" name="sub"/>
            </form>
        </div>
    </div>
    <?php
    if(isset($_GET['aula'])){
        $aula = $_GET['aula'];
        require_once("info.php");
    }?>
</div>
```

A continuación se va a explicar parte por parte de que se encarga cada línea de código.

```
session_start();
require_once("../php/functions/functions.php");
require_once("../php/functions/db_connection.php");
require('../inc/header.php');
```

En esta parte se crea una sesión nueva al entrar a la página por primera vez con el fin de que recordar donde se quedó la última vez que se abrió, las líneas en las que pone "require" o "require_once" sirven para importar un archivo de forma parecida a cuando se utiliza "include" en C para cargar librerías.

```
CheckFloorUp($MAX_FLOOR);
CheckFloorDown($MIN_FLOOR);
```

Estas son unas funciones que comprueban si se encuentra en el último piso o en el primero para no dejar al usuario hacer clic en el botón para subir o bajar.

```
<script src="https://kit.fontawesome.com/a921d0c18e.js"
crossorigin="anonymous"></script>
<div class="main-content">
    <div class="planos">
        <div class=<?=$_SESSION['planta']?>>
            <img  id=<?=$_SESSION['planta']?>
src="../assets/<?=$_SESSION['planta']?>icai.jpg">
            <div class="aulas">
                <form method="get">
                <?php generate_floor($_SESSION['planta_actual']);?>

                </form>
            </div>
        </div>
```

Esta parte de aquí se encarga de generar el plano de ICAI dependiendo de la planta en la que se encuentre haciendo uso de una función llamada "generate_floor", función de php que genera código html para poder hacer clic en las distintas aulas, el código de esta función es el siguiente:

```
function generate_floor($floor){
    $valid_array = [1,2,3,4,7,8,9,10,11,12,13];
    for($i=1;$i<14;$i++){
        if($i<10){
          $aula = "A".$floor."0".$i;
        }else{
            $aula = "A".$floor.$i;
        }
        if(in_array($i,$valid_array)){
```

```
            echo("<div class='aula$i $aula aula' name='$aula'><a
href='?aula=$aula'>$aula</a></div>");
        }
    }
}
```

Esta función únicamente toma como entrada el numero de la planta en la que se encuentra y da como salida código html para generar las aulas y poder seleccionarlas.

A continuación se muestra el código que se utiliza para crear los botones para subir o bajar de planta en indicar el piso actual:

```
        <div class="selector">
            <form method="post">
                <input type="submit" value=">" class="upbutton<?php
if($_SESSION["planta_actual"]==$MAX_FLOOR)echo($MAX_FLOOR);?>" name="add"/>
                <input type="text" readonly class="planta" name="planta_actual"
value="<?=$_SESSION["planta_actual"]?>" size="0"/>
                <input type="submit" value="<" class="downbutton<?php
if($_SESSION["planta_actual"]==$MIN_FLOOR)echo($MIN_FLOOR);?>" name="sub"/>
            </form>
        </div>
    </div>
```

Hay que tener en cuenta que todo este código HTML y php está acompañado de .css para que se vea bien visualmente ya que sin eso no se vería correctamente la página.

```
    <?php
    if(isset($_GET['aula'])){
        $aula = $_GET['aula'];
        require_once("info.php");
    }?>
```

Estas últimas líneas de código sirven para generar la información una vez se ha seleccionado un aula para visualizarla ya que importa el contenido de “info.php” para poder mostrarlo.

7.3.1 “INFO.PHP”

Este archivo se ha mencionado anteriormente y se encarga de generar la información de un aula y mostrarla, para ello se ha creado el siguiente código:

```
<?php
require_once("../php/functions/db_connection.php");
require_once("../php/functions/functions.php");

$DbCon = new MyDbConn($aula);
?>
<script></script>
```

```
<div class='info-container'>
    <div class="exit-info"><a href="/src/tests/index.php"><i class="fa-regular
fa-circle-xmark"></i></a></div>
    <div class='data-list'>
        <div class="encabezado">
        <?php
        echo($DbCon->GetAula());
        ?>
        </div>
        <div class="display-data">

            <i class="fa-solid fa-temperature-
half"><?=WriteSpace(6)?>Temperatura: <?=$DbCon->GetTemp()?></i>

            <i class="fa-solid fa-droplet"><?=WriteSpace(7)?>Humedad: <?=$DbCon-
>GetHumidity()?></i>

            <i class="fa-solid fa-users"><?=WriteSpace(5)?>Personas: <?=$DbCon-
>GetPersonas()?></i>

            <i class="fa-solid fa-ear-listen"><?=WriteSpace(6)?>Ruido: <?=$DbCon-
>GetRuido()?></i>
        </div>
    </div>
</div>
<?php
$DbCon->CloseCon();
```

En primer lugar se crea una conexión con la base de datos en la que se encuentra la información de las aulas, luego se utilizan funciones creadas para obtener la información de temperatura, humedad, personas y ruido de dicha base de datos y mostrarla. Se ha utilizado una función que se llama WriteSpace cuya única función es escribir espacios para compensar el espacio ocupado por los distintos iconos, a continuación se muestra una imagen de la parte de información para que se entienda mejor que es necesario para que esté todo alineado.

Figura 12: Pestaña de información del menú general

Todas las hojas de estilos se van a incluir en los anexos debido a la extensión de estas y de la complejidad para explicar cada uno de sus elementos al igual que los planos utilizados y las funciones de php.

7.4 BASE DE DATOS

Para la base de datos se ha utilizado "phpMyAdmin" debido a la facilidad que aporta para conectarse a ella desde lenguaje php, la información necesaria para mostrarse por pantalla se colocado en una única tabla que contiene la información de todas las aulas de todas las plantas como se muestra a continuación:

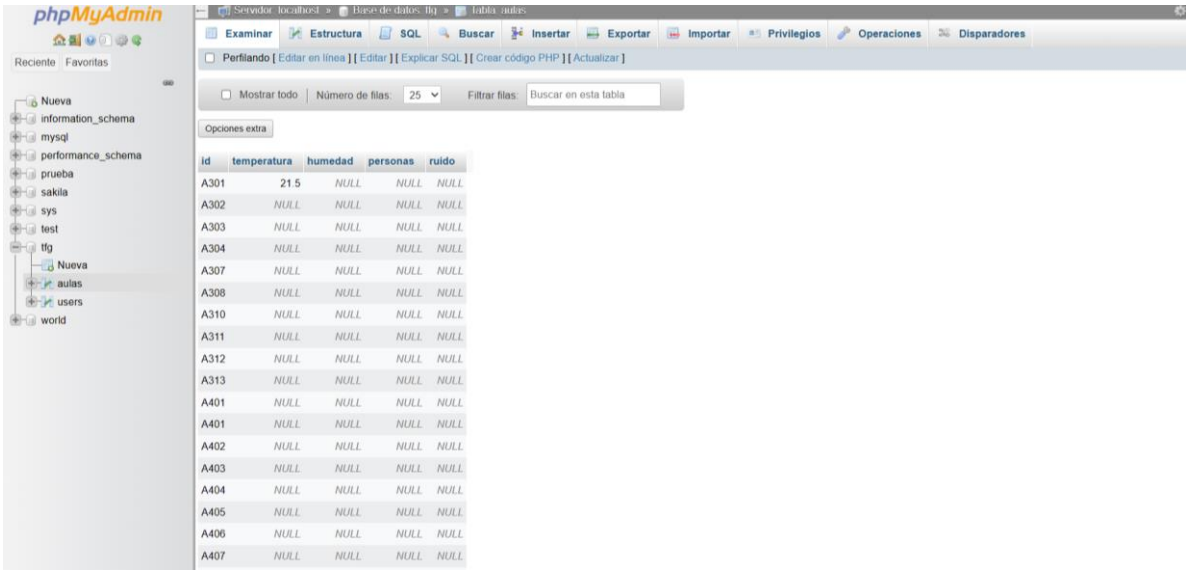


Figura 13: Base de datos con la información de las aulas

Como se puede observar la gran mayoría está vacío, esto se debe a que como este trabajo depende de otro que proporciona la información, todavía no existe esa información y por lo tanto no se puede mostrar.

Capítulo 8. DESARROLLO DE CÁMARA

La cámara que se ha decidido desarrollar es una cámara M5Stack Timer F, basada en el chip ESP32, se ha elegido esta cámara debido a que tiene conexión a Wifi para poder enviar y recibir información. Dicha cámara se va a conectar a un servidor de MQTT para poder enviar las imágenes que captura.

8.1 INSTALACIÓN Y CONFIGURACIÓN MQTT

Para el desarrollo de la cámara es importante haber configurado previamente el servidor y broker de MQTT (Message Queuing Telemetry Transport) ya que es imprescindible para el funcionamiento de ésta.

Se ha decidido instalar un broker de MQTT llamado "mosquitto" para Windows, para ello hay dirigirse a su **página oficial** y descargarse la última versión siguiendo los siguientes pasos:

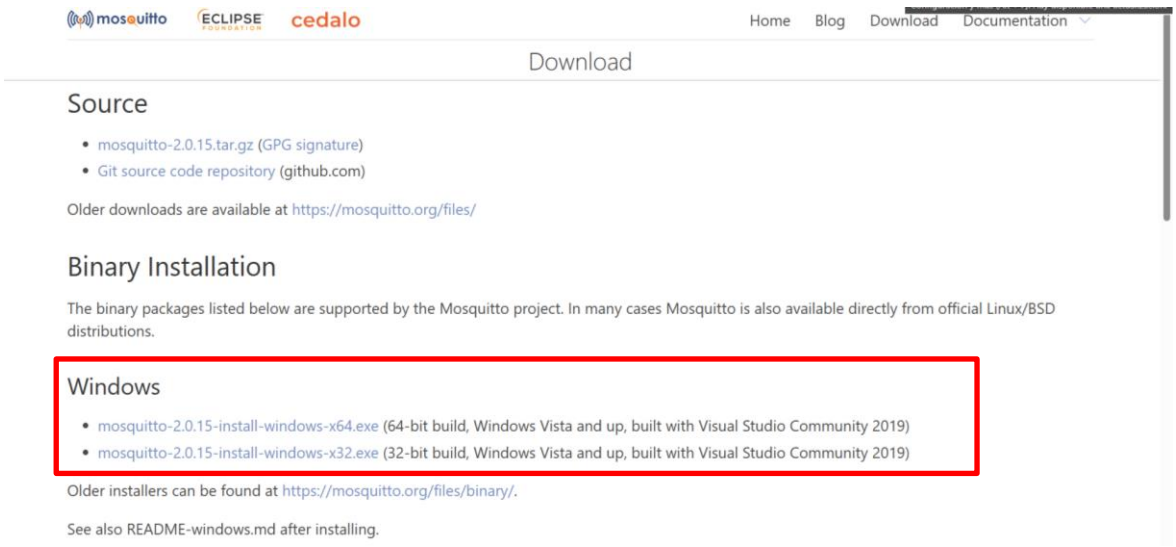


Figura 14: Instalación mosquitto, paso 1

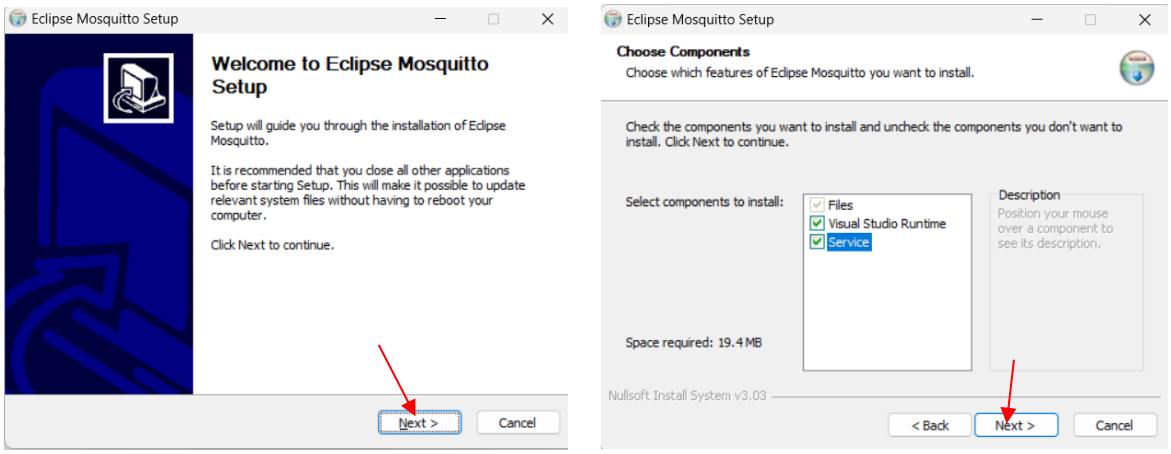


Figura 15: Instalación mosquitto, paso 2

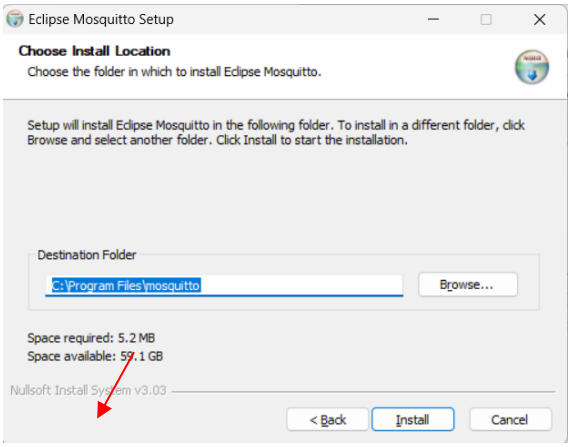


Figura 16: Instalación mosquitto, paso 3

Una vez realizados estos pasos se accede a la carpeta de "mosquitto" donde se haya guardado al instalarlo, y se crea un nuevo archivo llamado "mqtt.conf" para configurar el broker de MQTT, para ello solo hay que escribir lo siguiente en el archivo:

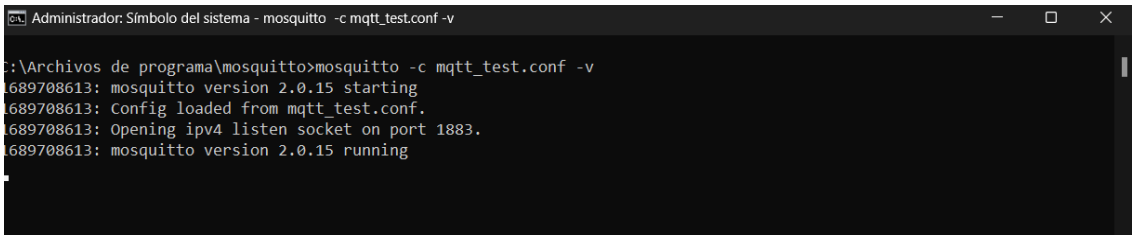
```
listener 1883 0.0.0.0
allow_anonymous true
```

Esto sirve para permitir el acceso a dispositivos que se encuentran en la red local, una vez realizados los cambios hay que comprobar que funciona correctamente.

Para ello hay que abrir el símbolo del sistema y acceder a la carpeta de "mosquitto", una vez dentro hay que arrancar el broker lo primero y para ello se utiliza la siguiente línea de código:

```
mosquitto -c mqtt.conf -v
```

Una vez se ejecuta se obtiene lo siguiente:



Para enviar un mensaje hay que abrir otro símbolo del sistema y escribir los siguiente también dentro de la carpeta de "mosquitto":

```
mosquitto_pub -h localhost -m "El mensaje que sea" -q 1
```

Donde la q sirve para indicar el QoS que se desee. El QoS (Quality of Service) en MQTT es un nivel de calidad que determina el grado de fiabilidad y garantía de entrega de los mensajes enviados entre el cliente y el broker MQTT. Hay tres niveles de QoS disponibles en MQTT:

- QoS 0 (At Most Once): En este nivel, se garantiza que el mensaje se entregue una vez sin ninguna confirmación ni reintentos. No hay garantía de entrega, ya que el mensaje se envía sin confirmación y no se almacena en el broker.
- QoS 1 (At Least Once): En este nivel, se garantiza que el mensaje se entregue al menos una vez. El cliente envía el mensaje al broker y espera una confirmación de entrega (PUBACK). Si no recibe la confirmación, el cliente volverá a enviar el mensaje hasta que reciba la confirmación.

- QoS 2 (Exactly Once): En este nivel, se garantiza que el mensaje se entregue exactamente una vez. El cliente y el broker realizan un proceso de confirmación más complejo, lo que implica un intercambio de mensajes adicionales para garantizar que el mensaje se entregue solo una vez, incluso en caso de pérdida de conexión o fallas intermedias.

El nivel de QoS elegido debe basarse en los requisitos del proyecto y la importancia de la entrega de mensajes sin pérdida. Cuanto mayor sea el nivel de QoS, mayor será la fiabilidad de la entrega, pero también se incrementará la sobrecarga de comunicación debido a los intercambios de mensajes adicionales.

Para recibir mensajes se utiliza otro comando que se queda "escuchando continuamente a un topic en concreto" que es el siguiente:

```
mosquitto_sub -h localhost -t "El topic que sea"
```

8.2 DESARROLLO DE CAMARA EN ESPRESSIF IDE

Espressif IDE es un entorno de desarrollo integrado (IDE, por sus siglas en inglés) creado por Espressif Systems, una empresa especializada en el desarrollo de microcontroladores y soluciones IoT. Es un software que proporciona un conjunto de herramientas y recursos para facilitar la programación de dispositivos basados en los microcontroladores ESP32 y ESP8266, que son ampliamente utilizados en aplicaciones de Internet de las cosas (IoT).

8.2.1 INSTALACIÓN ESPRESSIF IDE

La instalación de este programa es muy sencilla, lo primero es ir a la **página de descargas oficial** y hacer clic en el "Universal Online Installer", aceptar todo y proceder a la instalación de la aplicación.

Una vez completada la instalación se debería abrir una ventana como la siguiente:

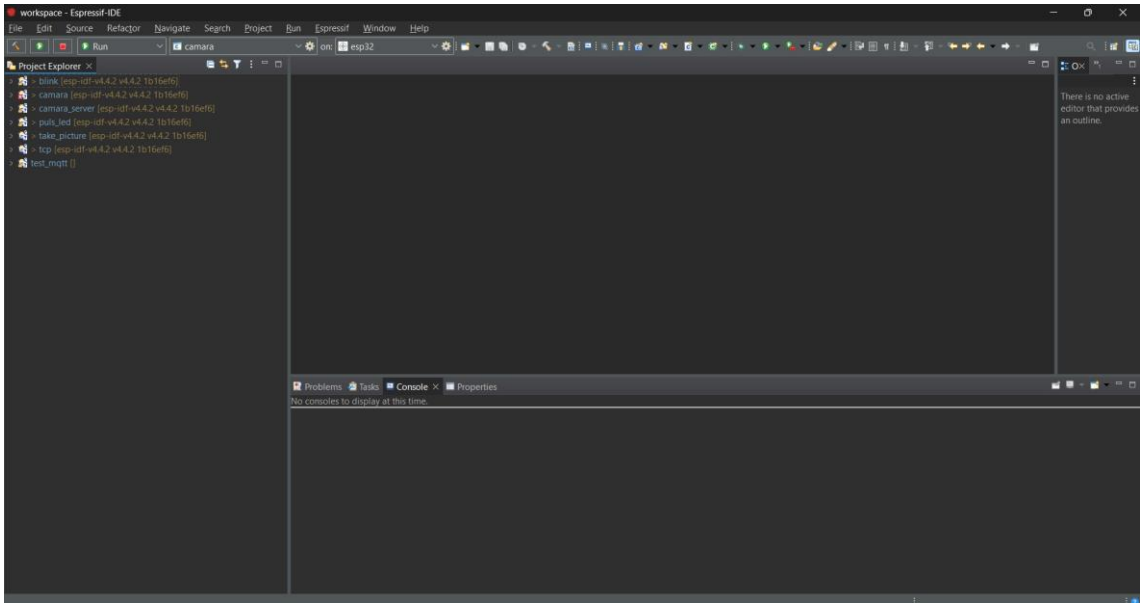


Figura 17: Aplicación Espressif IDE

8.2.2 ESTRUCTURA DEL PROYECTO

EL proyecto de Espressif consta de una carpeta llamada "main" con un archivo "main.c" y dos subcarpetas más. Dichas subcarpetas son "drivers" en la cual se incluyen todos los archivos con extensión .c y "headers" en la cual se incluyen los que tienen extensión .h como se muestra a continuación:

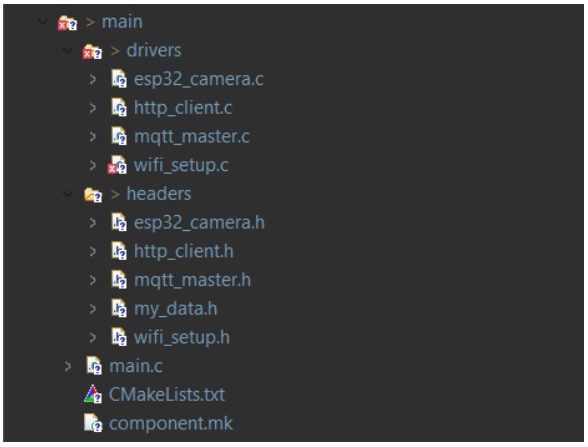


Figura 18: Estructura proyecto Espressif

8.2.2.1 Archivo main.c

El archivo main.c es el principal, el que controla como se va a comportar la cámara, el código es muy sencillo ya que lo más complicado se ha dejado en los drivers para dejar el main más limpio y es el siguiente:

```
#include <stdio.h>

#include "headers/esp32_camera.h"
#include "headers/wifi_setup.h"
#include "headers/mqtt_master.h"
#define MIN_TIME 60000


void app_main(void)
{

	if(ESP_OK != init_camera()){
		return;
		}
	wifi_setup();
	initialize_mqtt();
	cam_setup_loop(); // WAITS UNTIL CAM SETUP IS FINISHED, TO FINISH THIS
PROCESS, SEND VIA MQTT ANY MESSAGE TO TOPIC "/casa/CAMSETUP/STOP"
	printf("CAM SETUP FINISHED");
	while (1){
		vTaskDelay(MIN_TIME/portTICK_PERIOD_MS);
		take_send_picture();
	}
```

Este programa incluye los archivos que aparecen en los #include, los cuales se van a añadir al anexo ya que son muy extensos, el archivo "esp32_camera.h" contiene las funciones necesarias para inicializar la cámara, configurarla según se quiera y capturar imágenes. El archivo "wifi_setup.h" contiene las funciones que se encargan de inicializar y configurar la red wifi dependiendo de unos parámetros definidos en un archivo llamado "my_data.h". Por último, el archivo "mqtt_master.h" es el que contiene todas las funciones relacionadas con MQTT, tanto la inicialización como de gestionar los mensajes que se envían y se reciben.

8.3 COMUNICACIÓN SERVIDOR-CÁMARA

La comunicación entre éstos se ha realizado a través de un script de python utilizando la librería "paho mqtt". Se ha decidido utilizar python debido a dos motivos, la necesidad de enviar datos a la cámara para poder hacer la configuración inicial, esto significa que cuando se coloca la cámara por primera vez en un aula no todas tienen los mismos niveles de iluminación ni posición relativa al aula por lo que se ha diseñado un script para enviar los ajustes que se quieran realizar a la cámara en cuanto a brillo, contraste, saturación y demás hasta conseguir el resultado deseado y desde el propio script de python enviar un mensaje para que la cámara salga de ese estado de configuración y pase a su estado normal de funcionamiento.

También se ha utilizado python para crear otro script que esté continuamente corriendo que se trata de un script que "escucha" a todos los mensajes que envía la cámara y decide qué hacer con ellos, ya que en ocasiones los mensajes recibidos son imágenes y otras tantas únicamente son confirmaciones de la recepción de mensajes.

8.3.1 SCRIPT DE CONFIGURACIÓN DE LA CÁMARA EN PYTHON

El funcionamiento del script ya se ha mencionado pero a continuación se muestra el código:

```
import paho.mqtt.publish as publish
import time

STOP = "/casa/CAMSETUP/STOP"
BRIGHTNESS = "/casa/CAMSETUP/BRIGHTNESS"
CONTRAST = "/casa/CAMSETUP/CONTRAST"
SATURATION = "/casa/CAMSETUP/SATURATION"
EXPOSURE_CTRL = "/casa/CAMSETUP/EXPOSURE_CONTROL"
WHITE_BALANCE = "/casa/CAMSETUP/WHITE_BALANCE"
WHITE_BAL_GAIN = "/casa/CAMSETUP/WHITE_BAL_GAIN"
WB_MODE = "/casa/CAMSETUP/WHITE_BALANCE_MODE"
LENS_CORRECTION = "/casa/CAMSETUP/LENS_CORRECTION"
VERTICAL_FLIP = "/casa/CAMSETUP/VERTICAL_FLIP"
GAINCEILING = "/casa/CAMSETUP/GAINCEILING"
DENOISE = "/casa/CAMSETUP/DENOISE"
SHARPNESS = "/casa/CAMSETUP/SHARPNESS"
SPECIAL_EFFECT = "/casa/CAMSETUP/SPECIAL_EFFECT"
TAKE_PICTURE = "/casa/CAMSETUP/TAKE_PICTURE"
```

```
CASA_BROKER_IP = "192.168.160.121"
UNI_BROKER_IP = "172.24.149.157"
UNI_MOVIL_BROKER_IP = "192.168.43.40"
BROKER = UNI_MOVIL_BROKER_IP

def stop_setup():
    publish.single(STOP, 0, qos=1, hostname=BROKER)

def send_setup(msg_dict: list):

    publish.multiple(msg_dict, hostname=BROKER)

def take_pic():
    publish.single(TAKE_PICTURE, 1, qos=1, hostname=BROKER)

def main():
    msg_dict = [
        {"topic": BRIGHTNESS, "payload": 0, "qos": 1, "retain": 0},
        {"topic": CONTRAST, "payload": 0, "qos": 1, "retain": 0},
        {"topic": SATURATION, "payload": 0, "qos": 1, "retain": 0},
        {"topic": EXPOSURE_CTRL, "payload": 1, "qos": 1, "retain": 0},
        {"topic": WHITE_BALANCE, "payload": 1, "qos": 1, "retain": 0},
        {"topic": WHITE_BAL_GAIN, "payload": 0, "qos": 1, "retain": 0},
        {"topic": WB_MODE, "payload": 3, "qos": 1, "retain": 0},
        {"topic": LENS_CORRECTION, "payload": 1, "qos": 1, "retain": 0},
        {"topic": VERTICAL_FLIP, "payload": 1, "qos": 1, "retain": 0},
        {"topic": GAINCEILING, "payload": 0, "qos": 1, "retain": 0},
        {"topic": DENOISE, "payload": 0, "qos": 1, "retain": 0},
        {"topic": SHARPNESS, "payload": 0, "qos": 1, "retain": 0},
        {"topic": SPECIAL_EFFECT, "payload": 0, "qos": 1, "retain": 0},
    ]
    # send_setup(msg_dict)
    time.sleep(5)
    take_pic()
    #stop_setup()

if __name__ == "__main__":
    main()
```

Como se puede observar, son muchos los ajustes que se le pueden realizar a la cámara, el funciomiento real de este script es que hay que modificar manualmente el diccionario "msg_dict" que contiene todos los ajustes y el nivel al que se quiere ajustar por lo general van del -5 al 5 en el payload.

Cuando se ejecuta el script se envían dichos mensajes y se dan 5 segundos para que la cámara realice los cambios que ha recibido y manda la orden para hacer una foto o detiene la configuración de la cámara si se comenta "take_pic()" y descomenta "stop_setup()".

8.3.2 SCRIPT EN PYTHON EN SERVIDOR

Este script de python está creado con la intención de que esté trabajando en segundo plano en el servidor de la universidad y su función es la de interceptar los mensajes que se envían a través de MQTT y decidir que hacer con esos mensajes debido a que se envían imágenes a través de protocolo pero también llegan mensajes de confirmaciones de cada envío y por lo tanto hay que filtrar lo que se quiere guardar y lo que no, en este caso únicamente guarda las imágenes para su posterior uso. A continuación se muestra el código:

```
import paho.mqtt.client as mqtt

TOPIC = "/casa/#"
num = 0
CASA_BROKER_IP = "[redacted]"
UNI_BROKER_IP = "[redacted]"
UNI_MOVIL_BROKER_IP = "[redacted]"
BROKER = UNI_MOVIL_BROKER_IP

def on_connect_callback(client: mqtt.Client, userdata, flags, rc,
properties=None):
    print("Connected with result code " + str(rc))
    client.subscribe(TOPIC)

def on_message_callback(client, userdata, message):
    global num

    if message.topic == "/casa/CAM":
        num += 1
        FILENAME = f"picture{num}.jpg"
        file = open(FILENAME, "wb")
        file.write(message.payload)
        file.close()
    else:
        print(f"{message.topic}:{str(message.payload)}")

def main():

    client = mqtt.Client(client_id="Pagina web")
    client.on_connect = on_connect_callback
    client.on_message = on_message_callback

    client.connect(BROKER, 1883, 60)

    # Blocking call that processes network traffic, dispatches callbacks and
    # handles reconnecting.
    # Other loop*() functions are available that give a threaded interface and a
    # manual interface.
    client.loop_forever(retry_first_connection=True)
```

Capítulo 9. ANÁLISIS Y RESULTADOS

9.1 ANÁLISIS DE PROBLEMAS Y SOLUCIÓN

A lo largo de todo el proyecto han ido surgiendo imprevistos que con tiempo se han solucionado, sobre todo en lo que concierne a la cámara debido a que se trata de bastante código y había que entender cómo compilaba el programa Espressif IDE. A continuación se van a comentar por encima los mayores problemas que han surgido:

- El código no compilaba correctamente en el proyecto ya que no es un entorno de trabajo con el que estábamos familiarizados. Después de muchas pruebas y mucho research en foros de "esp" y personas que habían tenido errores similares se consiguió que el proyecto compilara sin errores ya que había que incluir el repositorio de github a la carpeta de "Espressif/frameworks/esp-idf-v4.4.2/components" para que al hacer "build" en el proyecto ese repositorio de github con todos los drivers y librerías necesarias estuvieran accesibles.

- Después de que todo compilara bien surgió otro error, que era en el momento de volcar el código a la placa esp32, este era el resultado en la consola:

```
-      esptool.py v3.3.2-dev
-      Serial port COM3
-      Connecting.......
-      Chip is ESP32-D0WDQ6-V3 (revision 3)
-      Features: WiFi, BT, Dual Core, 240MHz, VRef calibration in efuse,
Coding Scheme None
-      Crystal is 40MHz
-      MAC: 78:21:84:9b:dc:ec
-      Uploading stub...
-      Running stub...
-      Stub running...
-      Configuring flash size...
-      Traceback (most recent call last):
-        File "C:/Espressif/frameworks/esp-idf-
v4.4.2/components/esptool_py/esptool/esptool.py", line 5399, in <module>
-          _main()
-        File "C:/Espressif/frameworks/esp-idf-
v4.4.2/components/esptool_py/esptool/esptool.py", line 5392, in _main
```

```
-          main()
-        File "C:/Espressif/frameworks/esp-idf-
v4.4.2/components/esptool_py/esptool/esptool.py", line 4814, in main
-          esp.flash_set_parameters(flash_size_bytes(args.flash_size))
-        File "C:/Espressif/frameworks/esp-idf-
v4.4.2/components/esptool_py/esptool/esptool.py", line 1056, in
flash_set_parameters
-          self.check_command("set SPI params",
ESP32ROM.ESP_SPI_SET_PARAMS,
-        File "C:/Espressif/frameworks/esp-idf-
v4.4.2/components/esptool_py/esptool/esptool.py", line 495, in
check_command
-          val, data = self.command(op, data, chk, timeout=timeout)
-        File "C:/Espressif/frameworks/esp-idf-
v4.4.2/components/esptool_py/esptool/esptool.py", line 468, in command
-          p = self.read()
-        File "C:/Espressif/frameworks/esp-idf-
v4.4.2/components/esptool_py/esptool/esptool.py", line 413, in read
-          return next(self._slip_reader)
-      StopIteration
-      CMake Error at run_serial_tool.cmake:56 (message):
-        Python
-        C:/Espressif/frameworks/esp-idf-
v4.4.2/components/esptool_py/esptool/esptool.py
-        --chip esp32 failed
```

El error es el texto marcado en rojo y realmente no es muy intuitivo ver de dónde viene el error al configurar el "flash size". Antes de esto se han tenido que hacer unos ajustes en el archivo "sdkconfig" del proyecto, el cual cuando se ejecuta se abre un menú para ajustar las opciones del proyecto. Las opciones que se cambiaron fueron en el apartado de "Serial flasher config":

- "flash SPI speed" de 40MHz a 80MHz.
- "Flash size" de 2MB a 4MB que es lo que soporta la cámara.

Dentro del apartado de "Component config" en "ESP32-specific":

- Enable de la opción "Support for external, SPI-connected RAM"
- "Set RAM clock speed" de 40MHz a 80MHz

Una vez guardados estos cambios ya se puede explicar el error marcado en rojo. Como se ha dicho antes el propio error no es muy indicativo de lo que está pasando realmente. Tras

mucho research en foros de personas que les había ocurrido errores similares se llegó a la conclusión de que el problema era el que el baud-rate que viene por defecto(460800) en el proyecto era muy alto para el puerto y se intentó cambiarlo en el mismo menú del archivo "sdkconfig" en el que previamente se modificaron unos ajustes ya que hay una opción para cambiar el baud-rate pero realmente ese no era el que había que modificar ya que cuando se volvía a "flashear" el proyecto a la placa, seguía que el baud-rate era el mismo que antes(460800).

Buscando se encontró un archivo en el que por defecto estaba ese valor de 460800, para solucionarlo se bajó a 115200 y tras realizar ese cambio se solucionó el problema de volcar el código a la placa y ya funcionaba correctamente. El archivo se llama "serial_ext.py" y se encuentra en "Espressif/frameworks/esp-idf-v4.4.2/tools/idf_py_actions/" en la línea 188.

Después de arreglar este error surgió otro a la hora de hacer la foto ya que la frecuencia del XCLK era de 20MHz y es demasiado alta si se quieren hacer fotos con una calidad un poco más alta y un framesize no muy pequeño, por lo que bajándolo a 10MHz se solucionó el problema. A partir de ese momento ya se podían obtener imágenes y modificar los ajustes de la cámara ya que la calidad de foto depende mucho de la iluminación y de los ajustes.

Otro aspecto destacable que ha presentado dificultad ha sido la creación de la aplicación web ya que nunca haber practicado nada con php ni HTML, se convirtió en un desafío aunque no imposible ni mucho menos, con tiempo y práctica todo se soluciona.

9.2 RESULTADOS

El resultado más visible ha sido la aplicación web debido a que en cualquier momento se puede acceder a ella no como en el caso del servidor MQTT ni la cámara. A continuación se muestra los resultados de la página web pasando por distintas partes de la página:

Figura 19: Resultado de la aplicación web, apartado general, planta 3

Figura 20: Resultado de la aplicación web, apartado general, planta 3 con información de A302

El resultado de esta aplicación ha resultado bastante satisfactorio ya que ha quedado una interfaz muy intuitiva, funcional y preparada para futuras actualizaciones como las estadísticas o las alarmas.

Por otro lado la cámara basada en el ESP32 funcionó perfectamente como se esperaba pero la idea de dicha cámara era de colocarla en el techo del aula y al ser lente gran angular se

suponía que iba a poder capturar todo el aula o casi toda, en la realidad no capturaba suficiente parte del aula desde ese lugar, aunque como última opción antes de adquirir otra cámara con lente con un ángulo más abierto se propuso poner la cámara en la pared del fondo para así capturar mayor parte del aula como solución alternativa.

Capítulo 10. CONCLUSIONES Y TRABAJOS FUTUROS

Comentar las conclusiones del proyecto, destacando lo que se ha hecho, dejando claros qué objetivos se han cubierto y cuáles son las aportaciones hechas.

Para concluir, el proyecto ha logrado desarrollar un sistema integral y eficiente para la monitorización de las aulas universitarias, cumpliendo con los objetivos establecidos. Se ha implementado con éxito un servidor MQTT y una aplicación web accesible desde dispositivos móviles y ordenadores. Además, se ha integrado la programación de una cámara basada en el ESP32 para capturar imágenes y enviarlas a través del protocolo MQTT aunque no haya sido el resultado esperado.

En términos de los objetivos establecidos, se ha logrado:

1. Desarrollar un servidor MQTT: Se ha implementado un servidor MQTT robusto y confiable que actúa como broker, asegurando la comunicación eficiente entre los dispositivos IoT y la aplicación web.

2. Crear una aplicación web accesible: Se ha desarrollado una interfaz web intuitiva que permite a los usuarios monitorear y controlar el estado de las aulas universitarias. Los usuarios pueden visualizar en tiempo real información sobre la temperatura, iluminación, presencia y ruido.

3. Integrar cámara: Se ha programado una cámara basada en el ESP32 para capturar imágenes y enviarlas a través del protocolo MQTT, brindando una perspectiva visual adicional.

Las principales aportaciones de este proyecto son:

- Solución integral y eficiente: Se ha creado una solución integral que aborda la monitorización de las aulas universitarias, optimizando la eficiencia energética, mejorando la comodidad de los estudiantes y brindando una gestión centralizada.
- Tecnologías avanzadas: Se han utilizado tecnologías de vanguardia, como el protocolo MQTT, el servidor Apache, el lenguaje PHP y la programación en ESP32, lo que demuestra la aplicación de tecnologías modernas en el ámbito educativo.
- Mejora de la experiencia educativa: El proyecto contribuye a mejorar la experiencia educativa al proporcionar un entorno más cómodo y adaptado a las necesidades de los estudiantes, permitiendo un mayor compromiso y un mejor rendimiento académico.

Como futura continuación de este proyecto, el siguiente paso más lógico sería continuar la aplicación web utilizando estadísticas y alarmas como se dejado indicado en este proyecto. Otra continuación muy interesante podría ser la digitalización y centralización de los sistemas de control ambiental de las aulas, ya que actualmente son analógicos y con la aplicación web, servidor y broker MQTT ya existentes gracias a este proyecto sería la solucion perfecta para la gestión y supervisión centralizada.

Capítulo 11. BIBLIOGRAFÍA

[1] W3 School "PHP course" https://www.w3schools.com/php/

[2] Espressif "Espressif Forum" https://www.esp32.com/

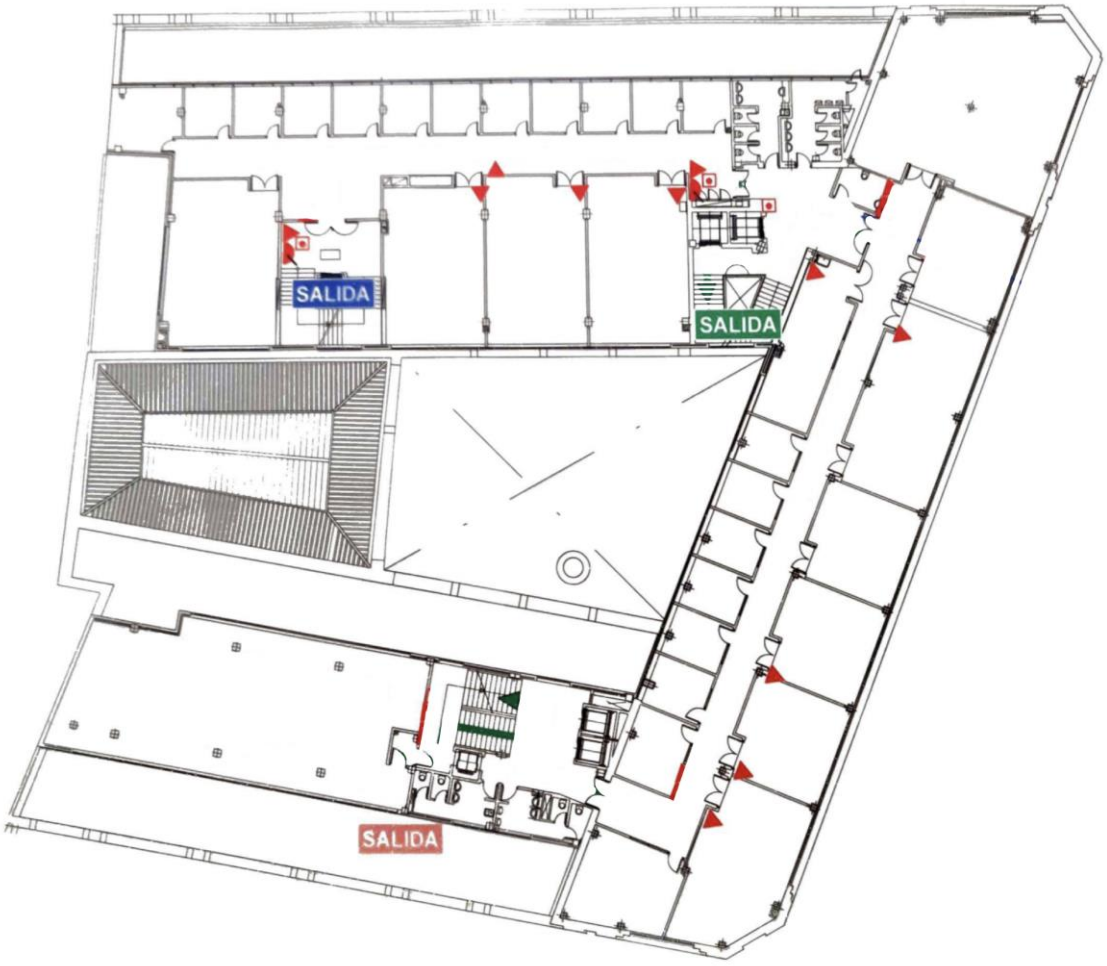
[3] Espressif "ESP-IDF Programming Guide" https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/

[4] PierreF, Roger.Light "paho-mqtt" https://pypi.org/project/paho-mqtt/

[5] Quodus Solution https://quodus.ai/

ANEXO I

Planos de las plantas de ICAI:



ANEXO II

Archivos relacionados con la cámara en Espressif

esp32_camera.c:

```
// support IDF 5.x
#ifndef portTICK_RATE_MS
#define portTICK_RATE_MS portTICK_PERIOD_MS
#endif

#include "esp_camera.h"
#include "../headers/esp32_camera.h"
#include "../headers/mqtt_master.h"
#include "../headers/my_data.h"

static const char *TAG = "take_picture";
static int setup_loop_status = 0;

//static cam_param_t params;

//static cam_param_t cam_options[4] = {
//            {
//                    .brightness = 0,
//                    .contrast = 0,
//                    .saturation = 0,
//                    .enable_exposure_ctrl = false,
//                    .enable_white_balance = false,
//                    .enable_white_balance_gain = false,
//                    .wb_mode = AUTO,
//                    .enable_lens_correction = false,
//                    .enable_vertical_flip = true,
//                    .gainceiling = GAINCEILING_2X,
//                    .denoise = 0,
//                    .sharpness = 0,
//                    .special_effect = NORMAL,
//            },
//            {
//                    .brightness = 1,
//                    .contrast = 0,
//                    .saturation = 0,
//                    .enable_exposure_ctrl = false,
//                    .enable_white_balance = false,
//                    .enable_white_balance_gain = false,
//                    .wb_mode = AUTO,
//                    .enable_lens_correction = false,
//                    .enable_vertical_flip = true,
```

```
//                      .gainceiling = GAINCEILING_2X,
//                      .denoise = 0,
//                      .sharpness = 0,
//                      .special_effect = NORMAL,
//              },
//              {
//                      .brightness = 2,
//                      .contrast = 0,
//                      .saturation = 0,
//                      .enable_exposure_ctrl = false,
//                      .enable_white_balance = false,
//                      .enable_white_balance_gain = false,
//                      .wb_mode = AUTO,
//                      .enable_lens_correction = false,
//                      .enable_vertical_flip = true,
//                      .gainceiling = GAINCEILING_2X,
//                      .denoise = 0,
//                      .sharpness = 0,
//                      .special_effect = NORMAL,
//
//              },
//              {
//                      .brightness = 3,
//                      .contrast = 0,
//                      .saturation = 0,
//                      .enable_exposure_ctrl = false,
//                      .enable_white_balance = false,
//                      .enable_white_balance_gain = false,
//                      .wb_mode = AUTO,
//                      .enable_lens_correction = false,
//                      .enable_vertical_flip = true,
//                      .gainceiling = GAINCEILING_2X,
//                      .denoise = 0,
//                      .sharpness = 0,
//                      .special_effect = NORMAL,
//              }
//
//};


static camera_config_t camera_config = {
    .pin_pwdn = CAM_PIN_PWDN,
    .pin_reset = CAM_PIN_RESET,
    .pin_xclk = CAM_PIN_XCLK,
    .pin_sccb_sda = CAM_PIN_SIOD,
    .pin_sccb_scl = CAM_PIN_SIOC,

    .pin_d7 = CAM_PIN_D7,
    .pin_d6 = CAM_PIN_D6,
    .pin_d5 = CAM_PIN_D5,
    .pin_d4 = CAM_PIN_D4,
    .pin_d3 = CAM_PIN_D3,
    .pin_d2 = CAM_PIN_D2,
```

```
    .pin_d1 = CAM_PIN_D1,
    .pin_d0 = CAM_PIN_D0,
    .pin_vsync = CAM_PIN_VSYNC,
    .pin_href = CAM_PIN_HREF,
    .pin_pclk = CAM_PIN_PCLK,

    //XCLK 20MHz or 10MHz for OV2640 double FPS (Experimental)
    .xclk_freq_hz = 10000000,
    .ledc_timer = LEDC_TIMER_0,
    .ledc_channel = LEDC_CHANNEL_0,

    .pixel_format = PIXFORMAT_JPEG, //YUV422,GRAYSCALE,RGB565,JPEG
    .frame_size = FRAMESIZE_240X240,    //QQVGA-UXGA, For ESP32, do not use sizes
above QVGA when not JPEG. The performance of the ESP32-S series has improved a
lot, but JPEG mode always gives better frame rates.

    .jpeg_quality = 2, //0-63, for OV series camera sensors, lower number means
higher quality
    .fb_count = 1,       //When jpeg mode is used, if fb_count more than one, the
driver will work in continuous mode.
       .fb_location = CAMERA_FB_IN_PSRAM,
    .grab_mode = CAMERA_GRAB_WHEN_EMPTY,
};

void cam_setup_loop(void){
       while (1){
              if(setup_loop_status){
                     return;
              }
       }
}

void set_setup_loop_stop(void){
       setup_loop_status = 1;
}
esp_err_t init_camera(void)
{
    //initialize the camera
    esp_err_t err = esp_camera_init(&camera_config);
    if (err != ESP_OK)
    {
        ESP_LOGE(TAG, "Camera Init Failed");
        return err;
    }
    return ESP_OK;
}

void adjust_camera_parameters(cam_param_t * parameters){

       sensor_t *sensor = esp_camera_sensor_get();

       sensor->set_brightness(sensor,parameters->brightness);//from -3 to 3
       sensor->set_contrast(sensor,parameters->contrast);//from -3 to 3
```

```
        sensor->set_saturation(sensor,parameters->saturation);//from -4 to 4
        sensor->set_exposure_ctrl(sensor,parameters->enable_exposure_ctrl);//
true(1) or false(0)
        sensor->set_whitebal(sensor,parameters->enable_white_balance);// true(1)
or false(0)
        sensor->set_awb_gain(sensor,parameters->enable_white_balance_gain);//
true(1) or false(0)
        sensor->set_wb_mode(sensor,parameters->wb_mode);//from 0 to 4,
(white_balance_mode_t MACROS)
        sensor->set_lenc(sensor,parameters->enable_lens_correction);// true(1) or
false(0)
        sensor->set_vflip(sensor,parameters->enable_vertical_flip);// true(1) or
false(0)
        sensor->set_gainceiling(sensor,parameters->gainceiling);// from 0 to
6,(gainceiling_t)
        sensor->set_denoise(sensor,parameters->denoise);// from 0 to 8
        sensor->set_sharpness(sensor,parameters->sharpness); //from -3 to 3
        sensor->set_special_effect(sensor,parameters->special_effect); //from 0 to
6, (special_effects_t MACROS)
}
esp_err_t take_send_picture(void){
        ESP_LOGI(TAG, "Taking picture...");
        camera_fb_t *pic = esp_camera_fb_get();

        // use pic->buf to access the image
        ESP_LOGI(TAG, "Picture taken! Its size was: %zu bytes", pic->len);
        //send_message(CAM_TOPIC,"Se ha sacado la foto",10,1);
        esp_camera_fb_return(pic);
        vTaskDelay(1000/portTICK_PERIOD_MS);
        send_message(CAM_TOPIC,(char *)pic->buf,pic->len,1);

        return ESP_OK;
}

esp_err_t setup_brightness(int msg){
        sensor_t *sensor = esp_camera_sensor_get();
        sensor->set_brightness(sensor,msg);
        ESP_LOGI(TAG,"Brightness set to %i",msg);
        //take_send_picture();
        return ESP_OK;
}

esp_err_t setup_contrast(int msg){
        sensor_t *sensor = esp_camera_sensor_get();
        sensor->set_contrast(sensor,msg);
        ESP_LOGI(TAG,"Contrast set to %i",msg);
        //take_send_picture();
        return ESP_OK;
}

esp_err_t setup_saturation(int msg){
        sensor_t *sensor = esp_camera_sensor_get();
        sensor->set_saturation(sensor,msg);
```

```
        ESP_LOGI(TAG,"Saturation set to %i",msg);
        //take_send_picture();
        return ESP_OK;
}

esp_err_t setup_exposure_ctrl(int msg){
        sensor_t *sensor = esp_camera_sensor_get();
        sensor->set_exposure_ctrl(sensor,msg);
        ESP_LOGI(TAG,"Exposure control set to %i",msg);
        //take_send_picture();
        return ESP_OK;
}

esp_err_t setup_whitebal(int msg){
        sensor_t *sensor = esp_camera_sensor_get();
        sensor->set_whitebal(sensor,msg);
        ESP_LOGI(TAG,"White balance set to %i",msg);
        //take_send_picture();
        return ESP_OK;
}

esp_err_t setup_white_bal_gain(int msg){
        sensor_t *sensor = esp_camera_sensor_get();
        sensor->set_awb_gain(sensor,msg);
        ESP_LOGI(TAG,"White balance gain set to %i",msg);
        //take_send_picture();
        return ESP_OK;
}

esp_err_t setup_wb_mode(int msg){
        sensor_t *sensor = esp_camera_sensor_get();
        sensor->set_wb_mode(sensor,msg);
        ESP_LOGI(TAG,"White balance mode set to %i",msg);
        //take_send_picture();
        return ESP_OK;
}

esp_err_t setup_lens_correction(int msg){
        sensor_t *sensor = esp_camera_sensor_get();
        sensor->set_lenc(sensor,msg);
        ESP_LOGI(TAG,"Lens correction set to %i",msg);
        //take_send_picture();
        return ESP_OK;
}

esp_err_t setup_vflip(int msg){
        sensor_t *sensor = esp_camera_sensor_get();
        sensor->set_vflip(sensor,msg);
        ESP_LOGI(TAG,"Vertical flip set to %i",msg);
        //take_send_picture();
        return ESP_OK;
}
```

```
esp_err_t setup_gainceiling(int msg){
	sensor_t *sensor = esp_camera_sensor_get();
	sensor->set_gainceiling(sensor,msg);
	ESP_LOGI(TAG,"Gainceling set to %i",msg);
	//take_send_picture();
	return ESP_OK;
}

esp_err_t setup_denoise(int msg){
	sensor_t *sensor = esp_camera_sensor_get();
	sensor->set_denoise(sensor,msg);
	ESP_LOGI(TAG,"Denoise set to %i",msg);
	//take_send_picture();
	return ESP_OK;
}

esp_err_t setup_sharpness(int msg){
	sensor_t *sensor = esp_camera_sensor_get();
	sensor->set_sharpness(sensor,msg);
	ESP_LOGI(TAG,"Sharpness set to %i",msg);
	//take_send_picture();
	return ESP_OK;
}

esp_err_t setup_special_effect(int msg){
	sensor_t *sensor = esp_camera_sensor_get();
	sensor->set_special_effect(sensor,msg);
	ESP_LOGI(TAG,"Special effect set to %i",msg);
	//take_send_picture();
	return ESP_OK;
}
```

mqtt_master.c:

```
/*
 * mqtt_master.c
 *
 *  Created on: 10 dic. 2022
 *      Author: navar
 */
#include "../headers/mqtt_master.h"
#include "../headers/esp32_camera.h"
#define TAG "MQTT"

char STOP[] = "/casa/CAMSETUP/STOP";
char BRIGHTNESS[] = "/casa/CAMSETUP/BRIGHTNESS";
char CONTRAST[] = "/casa/CAMSETUP/CONTRAST\0";
char SATURATION[] = "/casa/CAMSETUP/SATURATION\0";
```

```
char EXPOSURE_CTRL[] = "/casa/CAMSETUP/EXPOSURE_CONTROL\0";
char WHITE_BALANCE[] = "/casa/CAMSETUP/WHITE_BALANCE\0";
char WHITE_BAL_GAIN[] = "/casa/CAMSETUP/WHITE_BAL_GAIN\0";
char WB_MODE[] = "/casa/CAMSETUP/WHITE_BALANCE_MODE\0";
char LENS_CORRECTION[] = "/casa/CAMSETUP/LENS_CORRECTION\0";
char VERTICAL_FLIP[] = "/casa/CAMSETUP/VERTICAL_FLIP\0";
char GAINCEILING[] = "/casa/CAMSETUP/GAINCEILING\0";
char DENOISE[] = "/casa/CAMSETUP/DENOISE\0";
char SHARPNESS[] = "/casa/CAMSETUP/SHARPNESS\0";
char SPECIAL_EFFECT[] = "/casa/CAMSETUP/SPECIAL_EFFECT\0";
char TAKE_PICTURE[] = "/casa/CAMSETUP/TAKE_PICTURE";

esp_mqtt_client_handle_t client;

static int char_to_int(char* msg){
       int sign = 1;
       int num = 0;
       char p;

       if ('-' == *msg){
              sign = -1;
              msg++;
       }
       p = *msg - '0';
       num =(int)p;

       return num*sign;
}

void initialize_mqtt(void){
       const esp_mqtt_client_config_t mqtt_cfg = {
           .uri = BROKER_URL,
              .reconnect_timeout_ms = 5,
              .keepalive = 20
       };
       client = esp_mqtt_client_init(&mqtt_cfg);
       esp_mqtt_client_register_event(client, ESP_EVENT_ANY_ID,
mqtt_event_handler, client);
       esp_mqtt_client_start(client);
}
void log_error_if_nonzero(const char *message, int error_code)
{
    if (error_code != 0) {
        ESP_LOGE(TAG, "Last error %s: 0x%x", message, error_code);
    }
}

void mqtt_event_handler(void* event_handler_arg,esp_event_base_t event_base,
int32_t event_id, void* event_data){
       int msg_id;
       esp_mqtt_event_handle_t event = event_data;
       esp_mqtt_client_handle_t client = event->client;
```

```
       ESP_LOGI(TAG, "Event dispatched from event loop base=%s, event_id=%d",
event_base, event_id);
       switch (event->event_id){
       case MQTT_EVENT_CONNECTED:
              ESP_LOGI(TAG,"MQTT CONNECTED");
              msg_id = esp_mqtt_client_publish(client,"/casa/test","ESP32 is up
and running",0,1,0);
              ESP_LOGI(TAG,"Publish sent successfully, msg id:%d",msg_id);

              msg_id = esp_mqtt_client_subscribe(client,"/casa/test",1);
              ESP_LOGI(TAG,"suscribed successfully, msg id:%d",msg_id);
              msg_id = esp_mqtt_client_subscribe(client,"/casa/CAMSETUP/#",1);
              break;
       case MQTT_EVENT_DISCONNECTED:
              ESP_LOGI(TAG, "MQTT DISCONNECTED");
              break;
       case MQTT_EVENT_SUBSCRIBED:
              ESP_LOGI(TAG, "MQTT EVENT SUBSCRIBED, msg_id=%d", event->msg_id);
              msg_id = esp_mqtt_client_publish(client, "/topic/qos0", "data", 0,
0, 0);
              ESP_LOGI(TAG, "sent publish successful, msg_id=%d", msg_id);
              break;
       case MQTT_EVENT_UNSUBSCRIBED:
              ESP_LOGI(TAG, "MQTT EVENT UNSUBSCRIBED, msg_id=%d", event->msg_id);
              break;
       case MQTT_EVENT_PUBLISHED:
              ESP_LOGI(TAG, "MQTT EVENT PUBLISHED, msg_id=%d", event->msg_id);
              break;
       case MQTT_EVENT_DATA:
              ESP_LOGI(TAG, "MQTT EVENT DATA");
              printf("TOPIC=%.*s\r\n", event->topic_len, event->topic);
              printf("DATA=%.*s\r\n", event->data_len, event->data);
              // AQUI VA UNA FUNCION SI QUEREMOS HACER ALGO CON LOS DATOS
RECIBIDOS
              get_message_handler(event);
              break;
       case MQTT_EVENT_ERROR:
              ESP_LOGI(TAG, "MQTT_EVENT_ERROR");
              if (event->error_handle->error_type ==
MQTT_ERROR_TYPE_TCP_TRANSPORT) {
                     log_error_if_nonzero("reported from esp-tls", event-
>error_handle->esp_tls_last_esp_err);
                     log_error_if_nonzero("reported from tls stack", event-
>error_handle->esp_tls_stack_err);
                     log_error_if_nonzero("captured as transport's socket errno",
event->error_handle->esp_transport_sock_errno);
                     ESP_LOGI(TAG, "Last errno string (%s)", strerror(event-
>error_handle->esp_transport_sock_errno));
              }
              break;
       default:
           ESP_LOGI(TAG, "Other event id:%d", event->event_id);
           break;
```

```
        }
}

esp_err_t get_message_handler(esp_mqtt_event_handle_t event){
        char *topic = event->topic;
        char  *msg = event->data;
        int message;
        message = char_to_int(msg);

        ESP_LOGI(TAG,"topic es: %s",topic);
        if (strcmp(topic,STOP) == 0){
                set_setup_loop_stop();
        }else if (strcmp(topic,BRIGHTNESS) == 0){
                setup_brightness(message);
        }else if (strcmp(topic,CONTRAST) == 0){
                setup_contrast(message);
        }else if (strcmp(topic,SATURATION) == 0){
                setup_saturation(message);
        }else if (strcmp(topic,EXPOSURE_CTRL) == 0){
                setup_exposure_ctrl(message);
        }else if (strcmp(topic,WHITE_BALANCE) == 0){
                setup_whitebal(message);
        }else if (strcmp(topic,WHITE_BAL_GAIN) == 0){
                setup_white_bal_gain(message);
        }else if (strcmp(topic,WB_MODE) == 0){
                setup_wb_mode(message);
        }else if (strcmp(topic,LENS_CORRECTION) == 0){
                setup_lens_correction(message);
        }else if (strcmp(topic,VERTICAL_FLIP) == 0){
                setup_vflip(message);
        }else if (strcmp(topic,GAINCEILING) == 0){
                setup_gainceiling(message);
        }else if (strcmp(topic,DENOISE) == 0){
                setup_denoise(message);
        }else if (strcmp(topic,SHARPNESS) == 0){
                setup_sharpness(message);
        }else if (strcmp(topic,SPECIAL_EFFECT) == 0){
                setup_special_effect(message);
        }else if (strcmp(topic,TAKE_PICTURE) == 0){
                take_send_picture();
        }
        return ESP_OK;
}

void send_message(const char *topic, const char* data, int len, int qos){
        int msg_id;
        msg_id = esp_mqtt_client_publish(client,topic,data,len,qos,0);
        ESP_LOGI(TAG,"Publish sent successfully, msg id:%d",msg_id);


}
```

wifi_setup.c:

```
/*
 * wifi_setup.c
 *
 *  Created on: 9 dic. 2022
 *      Author: navar
 */
#include "../headers/wifi_setup.h"

#define TAG "WIFI"
#define WIFI_SUCCESS 1 << 0
#define WIFI_FAILURE 1 << 1
#define MAX_FAILURES 10


static EventGroupHandle_t s_wifi_event_group;
esp_mqtt_client_handle_t client;
static int s_retry_num = 0;

// Event handler for wifi events
static void wifi_event_handler(void *arg,esp_event_base_t event_base, int32_t
event_id, void* event_data){
	if (event_base == WIFI_EVENT && event_id == WIFI_EVENT_STA_START){
		ESP_LOGI(TAG,"Connecting to AP...");
		esp_wifi_connect();
	}else if(event_base == WIFI_EVENT && event_id ==
WIFI_EVENT_STA_DISCONNECTED){
		if(s_retry_num<MAX_FAILURES){
			ESP_LOGI(TAG,"Reconnecting to AP...");
			esp_wifi_connect();
			s_retry_num++;
		}else{
			xEventGroupSetBits(s_wifi_event_group,WIFI_FAILURE);
		}

	}
}

//event handler for ip events
static void ip_event_handler(void* arg, esp_event_base_t event_base, int32_t
event_id, void* event_data)
{
	if (event_base == IP_EVENT && event_id == IP_EVENT_STA_GOT_IP)
	{
	  ip_event_got_ip_t* event = (ip_event_got_ip_t*) event_data;
	  ESP_LOGI(TAG, "STA IP: " IPSTR, IP2STR(&event->ip_info.ip));
	  s_retry_num = 0;
	  xEventGroupSetBits(s_wifi_event_group, WIFI_SUCCESS);
    }

}
```

```
esp_err_t initialize_wifi(void){
	int status = WIFI_FAILURE;
	s_wifi_event_group = xEventGroupCreate();
	esp_netif_t *sta_netif = esp_netif_create_default_wifi_sta();
	assert(sta_netif);

	wifi_init_config_t cfg = WIFI_INIT_CONFIG_DEFAULT();

	esp_event_handler_instance_t wifi_event_handler_instance;
	esp_event_handler_instance_t ip_event_handler_instance;

	ESP_ERROR_CHECK(esp_wifi_init(&cfg));

	ESP_ERROR_CHECK(esp_event_handler_instance_register(WIFI_EVENT,ESP_EVENT_A
NY_ID,&wifi_event_handler,NULL,&wifi_event_handler_instance));
	ESP_ERROR_CHECK(esp_event_handler_instance_register(IP_EVENT,IP_EVENT_STA_
GOT_IP,&ip_event_handler,NULL,&ip_event_handler_instance));


	wifi_config_t wifi_config = {
		.sta = {
				.ssid = WIFI_SSID,
				.threshold.authmode = WIFI_MODE,
#if CASA||UNI_MOVIL
				.password = WIFI_PASWD,
#endif
				.pmf_cfg = {
					.required = false,
				},
		},
	};
	ESP_ERROR_CHECK(esp_wifi_set_mode(WIFI_MODE_STA));
	ESP_ERROR_CHECK(esp_wifi_set_config(WIFI_IF_STA,&wifi_config));

#if UNIVERSIDAD == 1
	ESP_ERROR_CHECK(esp_wifi_sta_wpa2_ent_set_identity((uint8_t *)WIFI_ID,
strlen(WIFI_ID)));
	ESP_ERROR_CHECK(esp_wifi_sta_wpa2_ent_set_username((uint8_t *)WIFI_USER,
strlen(WIFI_USER)));
	ESP_ERROR_CHECK(esp_wifi_sta_wpa2_ent_set_password((uint8_t *)WIFI_PASWD,
strlen(WIFI_PASWD)));
	ESP_ERROR_CHECK(esp_wifi_sta_wpa2_ent_enable());
#endif
	ESP_ERROR_CHECK(esp_wifi_start());
    ESP_LOGI(TAG, "STA initialization complete");

    // TIME TO WAIT
    EventBits_t bits = xEventGroupWaitBits(
		s_wifi_event_group,
			WIFI_SUCCESS|WIFI_FAILURE,
			pdFALSE,
			pdFALSE,
```

```
                    portMAX_DELAY
                    );
    /* xEventGroupWaitBits() returns the bits before the call returned, hence we
can test which event actually
         * happened. */
       if (bits & WIFI_SUCCESS) {
              ESP_LOGI(TAG, "Connected to ap");
              status = 0;
       } else if (bits & WIFI_FAILURE) {
              ESP_LOGI(TAG, "Failed to connect to ap");
              status = -1;
       } else {
              ESP_LOGE(TAG, "UNEXPECTED EVENT");
              status = -1;
       }
       /* The event will not be processed after unregister */
       ESP_ERROR_CHECK(esp_event_handler_instance_unregister(IP_EVENT,
IP_EVENT_STA_GOT_IP, ip_event_handler_instance));
       ESP_ERROR_CHECK(esp_event_handler_instance_unregister(WIFI_EVENT,
ESP_EVENT_ANY_ID, wifi_event_handler_instance));
       vEventGroupDelete(s_wifi_event_group);

       return status;

}

esp_err_t wifi_setup(void)
{
       ESP_LOGI(TAG,"Arrancando...");
       ESP_LOGI(TAG,"Free memory: %d bytes",esp_get_free_heap_size());

       esp_log_level_set("*", ESP_LOG_INFO);
       esp_log_level_set("MQTT_CLIENT", ESP_LOG_VERBOSE);
       esp_log_level_set("MQTT_EXAMPLE", ESP_LOG_VERBOSE);
       esp_log_level_set("TRANSPORT_BASE", ESP_LOG_VERBOSE);
       esp_log_level_set("esp-tls", ESP_LOG_VERBOSE);
       esp_log_level_set("TRANSPORT", ESP_LOG_VERBOSE);
       esp_log_level_set("OUTBOX", ESP_LOG_VERBOSE);

       ESP_ERROR_CHECK(nvs_flash_init());
       ESP_ERROR_CHECK(esp_netif_init());
       ESP_ERROR_CHECK(esp_event_loop_create_default());

       ESP_ERROR_CHECK(initialize_wifi());

       return ESP_OK;

}
```

ANEXO III

Funciones de PHP para la aplicación web:

```
<?php

class MyDbConn{
    protected $conn;
    protected $aula;
    private $table = "aulas";
    private $dbhost = "localhost";
    private $dbuser = "root";
    private $dbpass = "Aqcf48%6*@8k";
    private $db = "tfg";

    function __construct($aula)
    {
        $this->conn = new mysqli($this->dbhost, $this->dbuser, $this-
>dbpass,$this->db) or die("Connect failed: %s\n". $this->conn -> error);
        $this->aula = $aula;
    }

    public function GetTemp(){
        $result = $this->GetDataFromDB("temperatura")['temperatura'];
        if($result!=""){
            return $result." °C";
        }
        echo($result);
         return "NO DATA";
    }
    public function GetHumidity(){
        $result = $this->GetDataFromDB("humedad")['humedad'];
        if($result!=""){
            return $result."%";
        }
         return "NO DATA";
    }
    public function GetPersonas(){
        $result = $this->GetDataFromDB("personas")['personas'];
        if($result!=""){
            return $result;
        }
         return "NO DATA";
    }
    public function GetRuido(){
        $result = $this->GetDataFromDB("ruido")['ruido'];
        if($result!=""){
            return $result."dB";
```

```
        }else return "NO DATA";
    }
    public function CloseCon(){
        $this->conn->close();
    }
    public function GetAula(){
        return $this->aula;
    }
    public function SetAula($aula){
        $this->aula = $aula;
    }
    private function GetDataFromDB($data){
        $query = "SELECT $data FROM $this->table WHERE id='$this->aula'";
        $result = $this->conn->query($query);
        if($result->num_rows>0){
            return $result->fetch_assoc();
        }else return "NO DATA";
    }

}
```

```
<?php
$MAX_FLOOR = 4;
$MIN_FLOOR = 2;

function generate_floor($floor){
    $valid_array = [1,2,3,4,7,8,9,10,11,12,13];
    for($i=1;$i<14;$i++){
        if($i<10){
          $aula = "A".$floor."0".$i;
        }else{
            $aula = "A".$floor.$i;
        }
        if(in_array($i,$valid_array)){
            echo("<div class='aula$i $aula aula' name='$aula'><a
href='?aula=$aula'>$aula</a></div>");
        }
    }
}
function WriteSpace($num){
    $spaces = "";
    for($i=0;$i<=$num;$i++){
        $spaces .=" ";
    }
    return $spaces;
}

function SetDefaultValues(){
    // Si es la primera vez que se entra a la página, se accede a la tercera
planta por defecto
if(!isset($_SESSION['planta_actual'])){
```

```
    $_SESSION['planta_actual'] = 3;
    $_SESSION['planta'] = "planta".$_SESSION["planta_actual"];
}
if(!isset($_SESSION['alertas'])){
    SetAlert(1);
}
}

function CheckFloorUp($MAX_FLOOR){
    if (isset($_POST['add'])){
        $_SESSION["planta_actual"] = $_POST['planta_actual'];
        if($_SESSION["planta_actual"]<$MAX_FLOOR){
            $_SESSION["planta_actual"]++;
            $_SESSION['planta'] = "planta".$_SESSION["planta_actual"];
        }

    }
}

function CheckFloorDown($MIN_FLOOR){
    if (isset($_POST['sub'])){
        $_SESSION["planta_actual"] = $_POST['planta_actual'];
        if($_SESSION["planta_actual"]>$MIN_FLOOR){
            $_SESSION["planta_actual"]--;
            $_SESSION['planta'] = "planta".$_SESSION["planta_actual"];
        }

    }
}
function AlertManager(){
    if(isset($_GET['alertas'])){
        $_SESSION['alertas'] = 0;
    }
}
function CheckAlerts(){
    if ($_SESSION['alertas'] == 0){
        return 0;
    }elseif ($_SESSION['alertas'] == 1){
        return 1;
    }
}

function SetAlert($level){
    $_SESSION['alertas'] = $level;
}
```

ANEXO IV

Archivos de estilo .css.

aulas_style.css:

```
:root{
    --superior:19.5%;
}
.aula1{
    top:var(--superior);
    left:15%;
    clip-path:polygon(15% 0, 100% 0, 100% 100%, 0% 100%);

}
.aula2{
    top:var(--superior);
    left:34.5%;
}
.aula3{
    top:var(--superior);
    left:43.1%;
}
.aula4{
    top:var(--superior);
    left:51.8%;
}
.aula7{
    top:23%;
    left:80.6%;
    transform: rotate(22deg);
}
.aula8{
    top:34%;
    left: 77%;
    transform: rotate(22deg);
}
.aula9{
    top:43%;
    left: 74%;
    transform: rotate(22deg);
}
.aula10{
    top:52.2%;
    left:71%;
    transform: rotate(22deg);
}
.aula11{
```

```
    top:62%;
    left:67.8%;
    transform: rotate(22deg);
}
.aula12{
    top:71.8%;
    left:64.6%;
    transform: rotate(22deg);
}
.aula13{
    top:81.6%;
    left:61.5%;
    transform: rotate(22deg);
}
```

content_style.css:

```
@font-face {
    font-family: "noto sans";
    src: url(../fonts/NotoSans/NotoSans-Regular.ttf);
}
*{
    text-decoration: none;
}
:root{
    --duration:1s;
    --fuente:"'Courier New', Courier, monospace";
}
.main-content{
    width: 90%;
    height: 100%;
    background-color: rgb(225, 225, 225);
    display: flex;
    align-items: center;
    /* font-family: "noto sans"; */
    font-family: var(--fuente);
    }

.planos{
    width: 60%;
    height:70%;
    /* margin-top: 3%; */
    margin-left:3%;
    background-color: white;
    border-radius: 6px;
    display: flex;
    align-items: center;
    box-shadow: 0px 0px 4px 0px rgb(163, 162, 162);

}
```

```
.planta2{
    margin-left:10%;
    width:70%;
    height: 90%;
    position: relative;
}
#planta2{
    width:100%;
    height: 100%;
    rotate:1deg;
}
.planta3{
    margin-left:10%;
    width:70%;
    height: 90%;
    position: relative;
}
#planta3{
    width:100%;
    height: 100%;
    rotate:1deg;
}
.planta4{
    margin-left:10%;
    width:70%;
    height: 90%;
    position: relative;
}
#planta4{
    width:100%;
    height: 100%;
    rotate:1deg;
}

.aula a{
    color:black;
}
.aula1{
    width: 10%;
    height: 16%;
    background-color: white;
    display: flex;
    justify-content: center;
    align-items: center;
    position: absolute;
    /* cursor:pointer; */
}

.aula2, .aula3, .aula4{
    width: 8%;
    height: 16%;
    background-color: white;
    display: flex;
```

```
    justify-content: center;
    align-items: center;
    position: absolute;
    cursor:pointer;
}
.aula7, .aula13{
    width: 7.2%;
    height: 10%;
    background-color: white;
    display: flex;
    justify-content: center;
    align-items: center;
    position: absolute;
    cursor: pointer;
}
.aula8{
    width: 7.5%;
    height: 9%;
    background-color: white;
    display: flex;
    justify-content: center;
    align-items: center;
    position: absolute;
    cursor: pointer;
}

.aula9{
    width: 7.5%;
    height: 9%;
    background-color: white;
    display: flex;
    justify-content: center;
    align-items: center;
    position: absolute;
    cursor: pointer;
}

.aula10, .aula11, .aula12{
    width: 7.5%;
    height: 9%;
    background-color: white;
    display: flex;
    justify-content: center;
    align-items: center;
    position: absolute;
    cursor: pointer;
}

@keyframes aula1{
    100%{width:12%;height:19%;margin-left:-1%;margin-top:-1.5%;box-shadow: 0 0 0
2px grey;}
}
```

```
.aula1:hover{
    animation-name: aula1;
    animation-duration:var(--duration);
    width:12%;
    height:19%;
    margin-left:-1%;
    margin-top:-1.5%;
    box-shadow: 0 0 0 2px grey;
    border-radius:2%;
    z-index:1;
    clip-path:polygon(15% -10%, 110% -10%, 110% 110%, 0% 110%);
    font-size: large;

}
@keyframes aula234 {
    100%{width:10%;height:19%;margin-left:-1%;margin-top:-1%;box-shadow: 0 0 0
2px grey;}
}
.aula2:hover, .aula3:hover, .aula4:hover{
    animation-name: aula234;
    animation-duration:var(--duration);
    width:10%;
    height:19%;
    margin-left:-1%;
    margin-top:-1%;
    box-shadow: 0 0 0 2px grey;
    border-radius:2%;
    z-index:1;
    font-size: large;

}
@keyframes aula7 {
    100%{width:10%;height:13%;margin-left:-1.5%;margin-top:-1%;border-radius:
2%;box-shadow: 0 0 0 2px grey;}
}
.aula7:hover{
    animation-name:aula7;
    animation-duration:var(--duration);
    width:10%;
    height: 13%;
    margin-left: -1.5%;
    margin-top:-1%;
    box-shadow: 0 0 0 2px grey;
    border-radius: 2%;
    z-index: 1;
    font-size: large;

}
@keyframes aula8 {
    100%{width:10%;height:10%;margin-left:-1.5%;margin-top:-1%;border-radius:
2%;box-shadow: 0 0 0 2px grey;}
}
.aula8:hover{
```

```
    animation-name:aula8;
    animation-duration:var(--duration);
    width:10%;
    height: 10%;
    margin-left: -1.5%;
    margin-top:-1%;
    box-shadow: 0 0 0 2px grey;
    border-radius: 2%;
    font-size: large;
}

.aula9:hover{
    animation-name:aula8;
    animation-duration:var(--duration);
    width:10%;
    height: 10%;
    margin-left: -1.5%;
    margin-top:-1%;
    box-shadow: 0 0 0 2px grey;
    border-radius: 2%;
    font-size: large;
}

@keyframes aula10 {
    100%{width:10%;height:12%;margin-left:-1.5%;margin-top:-1%;border-radius:
2%;box-shadow: 0 0 0 2px grey;}
}
.aula10:hover, .aula11:hover, .aula12:hover{
    animation-name:aula10;
    animation-duration:var(--duration);
    width:10%;
    height: 12%;
    margin-left: -1.5%;
    margin-top:-1%;
    box-shadow: 0 0 0 2px grey;
    border-radius: 2%;
    z-index: 1;
    font-size: large;
}
@keyframes aula13 {
    100%{width:10%;height:19%;margin-left:-3%;margin-top:-1%;border-radius:
2%;box-shadow: 0 0 0 2px grey;}
}
.aula13:hover{
    animation-name:aula13;
    animation-duration:var(--duration);
    width:10%;
    height: 19%;
    margin-left: -3%;
    margin-top:-1%;
    box-shadow: 0 0 0 2px grey;
    border-radius: 2%;
    z-index: 1;
```

```
    font-size: large;

}

.selector{
    width:6%;
    height: 22%;
    background-color: rgb(225, 225, 225);
    display: flex;
    justify-content: center;
    position: relative;
    left: 5%;
    border-radius: 5px;
}

.up{
    width: 80%;
    height: 20%;
    position: relative;
    display: flex;
    justify-content: center;
    align-items: center;
    cursor: pointer;
    top:10%;

}
.down{
    width: 80%;
    height: 20%;
    position: relative;
    display: flex;
    justify-content: center;
    align-items: center;
    cursor: pointer;
    top:70%;
}
.upbutton{
    width: 50%;
    height: 30%;
    cursor: pointer;
    position: absolute;
    background-color: rgb(225, 225, 225);
    color:rgb(117, 117, 117);
    border:none;
    right: 25%;
    transform: rotate(-90deg);
    font-size:xx-large;
    font-weight: 300;
}

.upbutton4{
    width: 50%;
    height: 30%;
```

```
    cursor:not-allowed;
    position: absolute;
    background-color: rgb(225, 225, 225);
    color:rgb(117, 117, 117);
    border:none;
    right: 25%;
    transform: rotate(-90deg);
    font-size:xx-large;
    font-weight: 300;

}
.planta{
    width: 50%;
    height: 30%;
    position: absolute;
    display: flex;
    justify-content: center;
    align-items: center;
    font-size: 150%;
    background-color: rgb(225, 225, 225);
    color: rgb(70, 70, 73);
    border: none;
    top:35%;
    right: 12%;

}
.downbutton{
    width: 50%;
    height: 30%;
    cursor: pointer;
    position: absolute;
    top:70%;
    background-color: rgb(225, 225, 225);
    color:rgb(117, 117, 117);
    border:none;
    right:25%;
    transform: rotate(-90deg);
    font-size: xx-large;
}
.downbutton2{
    width: 50%;
    height: 30%;
    cursor: not-allowed;
    position: absolute;
    top:70%;
    background-color: rgb(225, 225, 225);
    color:rgb(117, 117, 117);
    border:none;
    right:25%;
    transform: rotate(-90deg);
    font-size: xx-large;
}
.fa-arrow-up{
```

```
    color:rgb(117, 117, 117);
    font-size: 140%;
    position: absolute;
    top:10%;

}
.fa-arrow-down{
    color:rgb(117, 117, 117);
    font-size:140%;
    position: relative;
    top:70%;
}
.upbutton:hover{
    color:rgb(53, 53, 53);
}
.downbutton:hover{
    color:rgb(53, 53, 53);
}
```

info_style.css:

```
:root{
    --fuente:"'Courier New', Courier, monospace";
    --duration:1s;
}
.info-container{
    width: 25%;
    height: 60%;
    /* background-color: rgb(195, 71, 71); */
    background-color: white;
    display: flex;
    margin-left: 5%;
    font-family: "noto sans";
    border-radius: 1%;
    box-shadow: 0px 0px 4px 0px rgb(163, 162, 162);
}
.exit-info{
    position: relative;
    width: 5%;
    height: 5%;
    display: flex;
    align-items: center;
    justify-content: center;
    top:1%;
    left: 93%;
    color: black;
}
.exit-info i{
    color: black;
}
```

```
.exit-info i:hover{
    animation-name: exit;
    animation-duration: var(--duration);
    color:rgb(69, 69, 69);
    font-size: x-large;
}

.data-list{
    width: 90%;
    height: 90%;
    background-color: white;
    display: list-item;
    align-self: center;
    justify-self: center;
    border-radius: 0;
}
.encabezado{
    width: 100%;
    height: 10%;
    display: flex;
    justify-content: center;
    align-items: center;
    font-size: large;
    font-family: var(--fuente);
}

.display-data{
    width: 100%;
    height: 90%;
    display: flex;
    flex-direction: column;
    align-items:flex-start;
    justify-content: space-evenly;
    overflow-wrap: break-word;
    padding-left: 5%;
    font-size: large;
}
@media (width<500px){
    .display-data{
        font-size: small;
    }
}
```

menu_style.css:

```
*{
    margin: 0;
    padding: 0;
    box-sizing: border-box;
    list-style: none;
```

```
    text-decoration: none;
}

.wrapper{
    display: flex;
    position: absolute;
    width: 100%;
    height: 100%;
}

.nav-container{
    display: flex;
    width: 12%;
    height: 100%;
    padding-top: 1%;
    background-color: rgb(195, 71, 71);
    background-color: rgb(51, 68, 197);
}
.menu {
    width: 100%;
    height: 40%;
    font-family:Verdana, Geneva, Tahoma, sans-serif;
    display: flex;
    justify-content: center;
    align-items: center;
}
.menu li{
    position: relative;
    margin-top: 30%;
}

.menu a{
    cursor: pointer;
    font-size: large;
    font-weight:normal;
    color: white;
}
.menu a:hover{
    color:antiquewhite;
}
```