Universidad de Cádiz
Escuela Superior de Ingeniería

TRABAJO FIN DE GRADO

GRADO EN INGENIERÍA INFORMÁTICA

IoT Educativo: Diseño y Desarrollo de un Entorno Sensorial para Evaluar y Potenciar el Rendimiento Estudiantil

AUTOR: JAVIER TENA WILSON

Puerto Real, septiembre de 2024

Universidad de Cádiz
Escuela Superior de Ingeniería

TRABAJO FIN DE GRADO

GRADO EN INGENIERÍA INFORMÁTICA

IoT Educativo: Diseño y Desarrollo de un Entorno Sensorial para Evaluar y Potenciar el Rendimiento Estudiantil

DIRECTOR: NESTOR MORA NUÑEZ
CODIRECTOR: ALFONSO GARCÍA DE PRADO
AUTOR: JAVIER TENA WILSON

Puerto Real, septiembre de 2024

Declaración personal de autoría

Javier Tena Wilson con DNI 45383460K, estudiante del Grado en Ingeniería Informática en la Escuela Superior de Ingeniería de la Universidad de Cádiz, como autor de este documento académico titulado 'IoT Educativo: Diseño y Desarrollo de un Entorno Sensorial para Evaluar y Potenciar el Rendimiento Estudiantil' y presentado como Trabajo Final de Grado

DECLARO QUE

Es un trabajo original, que no copio ni utilizo parte de obra alguna sin mencionar de forma clara y precisa su origen tanto en el cuerpo del texto como en su bibliografía y que no empleo datos de terceros sin la debida autorización, de acuerdo con la legislación vigente. Asimismo, declaro que soy plenamente consciente de que no respetar esta obligación podrá implicar la aplicación de sanciones académicas, sin perjuicio de otras actuaciones que pudieran iniciarse. En Puerto Real, a 28 de Agosto de 2024

Fdo: Javier Tena Wilson

Agradecimientos

Este proyecto representa el cierre de una etapa significativa en mi vida, por lo que quiero aprovechar esta sección para expresar mi más sincera gratitud a todas las personas e instituciones que han contribuido a que haya llegado hasta aquí.

En primer lugar, agradezco al personal de administración, mantenimiento y servicios de la Universidad de Cádiz y de la Escuela Superior de Ingeniería. El papel que han desempeñado ha sido fundamental para permitir el desarrollo de mi formación académica.

Agradezco profundamente a todos los profesores que, durante estos últimos cuatro años, han compartido sus conocimientos y experiencias, proporcionando una educación de calidad que ha sido crucial para mi crecimiento personal y profesional.

Un agradecimiento especial a Don Néstor Mora Nuñez y Don Alfonso García de Prado Fontela, quienes me brindaron la oportunidad de trabajar en este proyecto tan interesante. Su orientación y apoyo constante han sido indispensables para el éxito del mismo.

No quiero dejar de mencionar a mis compañeros de titulación, quienes han contribuido a crear un ambiente de aprendizaje positivo y enriquecedor. En particular, agradezco a mis amigos Alejandro Díaz Gómez e Ignacio Gilabert Rodríguez, con quienes he tenido la suerte de compartir esta experiencia al completo, haciendo de mi tiempo en la UCA algo que recordaré con mucho cariño por el resto de mi vida.

Finalmente, un agradecimiento de todo corazón a mi familia por haberme dado tanto amor y apoyo incondicional. Han sido el pilar que me ha ayudado a mantenerme en pie en los momentos más duros y sin ellos no habría podido llegar tan lejos.

Resumen

El presente trabajo de fin de grado tiene como propósito desarrollar un entorno IoT que pueda ser utilizado en aulas educativas por docentes para monitorear el comportamiento de sus alumnos. La motivación principal es ofrecer a los profesores una herramienta adicional para comprobar el grado de interés y atención de sus estudiantes, de manera que puedan adaptar la dinámica y contenidos de sus clases en tiempo real haciendo que el proceso educativo sea más interesante y eficiente para el alumnado.

Para ello, se ha desarrollado utilizando la metodología ADDIE un entorno IoT conformado por múltiples conjuntos de sensores que permitirán medir el movimiento, temperatura, humedad y luz en diferentes puntos de la sala. Cada una de estas agrupaciones de sensores se encuentra conectada a un microcontrolador ESP32-C3 que se encargará de leer y enviar los datos a una placa VisionFive 2 que actúa como middleware, así como alojar otros servicios necesarios para el entorno, como es por ejemplo la página web donde se pueden visualizar las lecturas. Uno de los principales atractivos del proyecto es que tanto los microcontroladores como el middleware utilizan la arquitectura de conjunto de instrucciones libres RISC-V. La adopción de RISC-V en este proyecto refuerza la tendencia creciente hacia el uso de arquitecturas abiertas, fomentando la innovación y reduciendo la dependencia de soluciones cerradas y costosas; además de validar la capacidad de RISC-V para manejar aplicaciones de IoT en tiempo real.

Los resultados obtenidos del desarrollo han sido exitosos, de manera que se concluye que el sistema IoT puede ser utilizado en un aula real para desempeñar su propósito. Además, el cumplimiento de los criterios de calidad impuestos sobre el sistema permite que entre otras cosas sea extensible y escalable, permitiendo el futuro crecimiento del entorno desarrollado.

Palabras clave: IoT, Educación, RISC-V, ESP32, VisionFive2, modelo ADDIE

Índice general

Índice de figuras x

Índice de tablas xi

I Prolegómeno 1

1. Introducción 3
1.1. Motivación . . . 3
1.2. Descripción del sistema actual . . . 3
1.3. Alcance y Objetivos . . . 4
1.4. Glosario de términos . . . 4
1.5. Organización del documento . . . 7

2. Antecedentes 9
2.1. Contexto . . . 9
2.2. Estado de la técnica . . . 9

3. Plan de gestión de proyecto 12
3.1. Metodología de desarrollo . . . 12
3.2. Planificación del proyecto . . . 13
3.3. Organización . . . 15
3.4. Presupuesto . . . 16
3.5. Riesgos . . . 18
3.6. Gestión de versiones . . . 20
3.7. Aseguramiento de calidad . . . 20
3.8. Actualización y mantenimiento del sistema . . . 21

II Desarrollo 24

4. Análisis 26
4.1. Análisis de comportamiento, requisitos técnicos y arquitectura IoT . . . 26
4.2. Análisis de audiencia . . . 29
4.3. Análisis del entorno . . . 30
4.4. Análisis de consideraciones adicionales . . . 30
4.4.1. RISC-V . . . 31
4.4.2. ODS y compromiso social . . . 33

5. Diseño 37
5.1. Elección del hardware . . . 37
5.2. Elección del software . . . 42
5.3. Diseño de la conexión de componentes . . . 46

5.3.1. Conexiones entre sensores y microcontroladores 46
5.3.2. Conexión de microcontroladores y middleware 49
5.4. Diseño de la programación de microcontroladores 50
5.5. Diseño del middleware Node-RED e Interfaz Web 54

6. Desarrollo **57**
6.1. Instalación de Ubuntu en VisionFive 2 57
6.2. Instalación y configuración del software para VisionFive 2 60
6.2.1. Instalación de Node-RED y sus paquetes 60
6.2.2. Mosquitto MQTT 61
6.2.3. Configuración adicional 63
6.3. Red Wi-Fi IoT 64
6.4. Programación de microcontroladores ESP32-C3 66
6.5. Programación Node-RED 69
6.6. Montaje físico de circuitos en caja 71

7. Implementación **74**
7.1. Pruebas piloto 74
7.2. Pruebas en un aula de docencia 75

8. Evaluación **78**
8.1. Evaluación funcional del sistema 78
8.2. Evaluación de la calidad del sistema 79
8.3. Problemas encontrados 81

III Epílogo **85**

9. Conclusiones **87**
9.1. Objetivos alcanzados 87
9.2. Lecciones aprendidas 88
9.3. Trabajo futuro 89

Información sobre Licencia **92**

Bibliografía **94**

IV Anexos **99**

A. Manual del desarrollador **101**
A.1. Introducción 101
A.2. Preparación del entorno de trabajo 101
A.3. Consideraciones generales sobre el desarrollo 103
A.4. Instrucciones para construcción 105

B. Manual de usuario **107**
B.1. Introducción 107
B.2. Instalación 107

B.3. Uso del sistema 108

Índice de figuras

3.1. Fases del modelo ADDIE [6] 12
3.2. Diagrama Gantt de la planificación inicial de las fases del proyecto . . . 15

4.1. Arquitectura del entorno IoT a construir 29
4.2. Interacción del entorno IoT con la audiencia 30

5.1. Sensor DHT22 y sus pines [17] 38
5.2. Sensor GY-302 BH1750 y sus pines [37] 38
5.3. Sensor PIR 501 y sus aspectos configurables y pines [12] 39
5.4. Placa de desarrollo ESP32-C3 utilizada [11] 41
5.5. Pinout de placa de desarrollo ESP32-C3 [14] 47
5.6. Diseño de las conexiones físicas Microcontrolador-Sensores 48
5.7. Temas MQTT y su relación con el middleware y los microcontroladores 50
5.8. Diseño de las partes del programa de los ESP32-C3 51
5.9. Diagrama de flujo del código completo del programa 54

6.1. Posiciones de los switches de VisionFive2 para la instalación del SO [41] 58
6.2. Conexión de los pines de VisionFive 2 con USB UART-TTL [35] 59
6.3. Instalador automático de Node-RED 61
6.4. Archivo de configuración de Mosquitto 62
6.5. Obtener dirección MAC y nombre de la interfaz de red 63
6.6. Configuración de la interfaz de red en VisionFive 2 64
6.7. Lista blanca de dispositivos que pueden usar la red IoT 66
6.8. Configuración de servidor DHCP del extensor de red 66
6.9. Paquete Arduino-IDE con soporte para ESP32-C3 67
6.10. Flujo Node-RED utilizado por el middleware 70
6.11. Interfaz desarrollada para el sistema IoT 71
6.12. Cajas de sensores desarrolladas 72

7.1. Pruebas realizadas en un aula de la ESI 76

8.1. Mensaje de error en sensor de luz BH1750 82
8.2. Desconexiones frecuentes por usar un broker MQTT online 82

9.1. Diagrama resumen del sistema final desarrollado 88

B.1. Manejo de gráficas en la interfaz 109

Índice de tablas

3.1. Planificación temporal inicial del proyecto 14
3.2. Desviación temporal respecto a la planificación inicial 15
3.3. Costes de componentes del sistema 16

Parte I

Prolegómeno

1. Introducción

A continuación, se describe la motivación del presente proyecto y su alcance. También se incluye un glosario de términos y la organización del resto de la presente documentación.

1.1. Motivación

La motivación para el desarrollo de este proyecto surge de la necesidad de mejorar la capacidad de los docentes para monitorizar y adaptar en tiempo real sus estrategias de enseñanza en un entorno educativo que cada vez es más desafiante y avanzado. En un contexto donde las distracciones son frecuentes y las necesidades de los estudiantes varían ampliamente, se hace imprescindible contar con herramientas que permitan un seguimiento más preciso del comportamiento y las condiciones del aula. Esto se hace especialmente relevante cuanto mayor sea el número de discípulos en el aula, puesto a que no es viable que el docente pueda asegurar el interés de cada uno de ellos de manera individual, además de que existen pequeños comportamientos que son dificilmente medibles por parte del profesor, como la cantidad de movimientos que realiza un alumno recientemente, aunque para un computador no supondría ningún problema recoger estos datos invisibles al ojo humano.

Este proyecto se desarrolla dentro del ámbito de la tecnología en un entorno educativo, creando un entorno IoT que ofrece a los docentes datos valiosos sobre el nivel de atención de sus alumnos y las condiciones ambientales del aula, buscando mejorar la calidad de sus lecciones para que esto se refleje directamente en el nivel de aprendizaje de sus discípulos. Además de esto, se utilizará la arquitectura abierta RISC-V en todos los componentes hardware del entorno IoT en los que sea posible, al tratarse de una de las tecnologías más prometedoras de los últimos tiempos, acentuando de esta manera el interés por explorar soluciones abiertas y flexibles que puedan ser adaptadas y escaladas libremente en el futuro.

Por lo tanto, este proyecto no solo trata de aportar una solución práctica a un problema real, sino también contribuir a la investigación y desarrollo de tecnologías alternativas en el ámbito educativo, con la posibilidad de afectar positivamente a la integración de tecnologías emergentes en el proceso de enseñanza y aprendizaje.

1.2. Descripción del sistema actual

El proyecto a desarrollar parte desde cero, no existiendo ningún antecesor que vaya a ser usado como punto de partida, a excepción de la idea principal del proyecto, que se inspira en un antiguo trabajo de investigación realizado por el tutor de este TFG, D. Nestor Mora Nuñez.

Este proyecto investigativo, denominado SCOLE, consistía en un entorno de aprendizaje colaborativo e inteligente que recopilaba datos, los contextualizaba, informaba, tomaba decisiones e interactuaba, optimizando así el proceso de enseñanza-aprendizaje. Aunque SCOLE sirvió como fuente de inspiración, se decidió desarrollar este proyecto de manera completamente autónoma, sin reutilizar ningún aspecto del diseño o implementación de SCOLE. Esto implicó realizar un análisis, diseño, desarrollo, implementación y evaluación del sistema IoT educativo desde cero, permitiendo así ajustar el entorno IoT a las necesidades específicas de este nuevo proyecto.

1.3. Alcance y Objetivos

El objetico fundamental del proyecto consiste en el análisis, diseño, desarrollo, implementación y evaluación de un entorno IoT que será construido en un aula de docencia. Este sistema deberá contar con sensores que recojan datos sobre el comportamiento de los alumnos y las condiciones del aula, de manera que estas lecturas sean tratadas y posteriormente mostradas al profesor de una manera clara y sencilla, permitiéndole tomar la mejor decisión en tiempo real sobre como continuar impartiendo la lección. El proyecto a desarrollar estará limitado a un prototipo funcional de sistema, estableciendo una base sólida y con potencial para que en un futuro pueda ser extendido a algo mucho mayor, haciendo especial énfasis durante su desarrollo en la optimización, eficiencia energética y costos reducidos. Por lo tanto es importante remarcar que no se busca crear un producto comercial.

El segundo objetivo de este proyecto consiste en servir como un caso de estudio sobre la viabilidad de los dispositivos con arquitectura RISC-V para ser utilizados en la creación de entornos IoT. Más específicamente esto se logrará utilizando modelos de dispositivos disponibles en el mercado que estén basados en esta arquitectura, como los microcontroladores y el computador monoplaca elegidos. No entra en el alcance deseado de este proyecto el diseño de un juego de instrucciones basado en arquitectura RISC-V ni la creación de ninguno de los componentes hardware que se van a utilizar, sino que se tratará de aprovechar lo máximo posible todas las soluciones parciales ya existentes y que puedan ser aprovechadas en la creación de los diferentes elementos del sistema.

El último de los objetivos será probar el correcto funcionamiento del entorno IoT en un aula real, a poder ser con alumnos en ella, aunque esto último no es estrictamente necesario. Está fuera de los límites del proyecto realizar un estudio del grado de mejora de la capacidad de aprendizaje de los estudiantes tras instalar este sistema en un aula.

1.4. Glosario de términos

En esta sección de la introducción se recoge una lista ordenada alfabéticamente de aquellos términos relevantes para el proyecto cuyo significado debe ser conocido para entender correctamente la información presentada en este documento:

- **ADDIE**: Modelo pedagógico ampliamente utilizado por muchos diseñadores educativos y programadores de formación para desarrollar programas de educación y formación. Su nombre es un acrónimo que representa los cinco pasos básicos de este modelo para el diseño y desarrollo de la experiencia de aprendizaje, que son los siguientes: Análisis, Diseño, Desarrollo, Implementación y Evaluación. En el modelo ADDIE, cada paso tiene un resultado que impulsa el paso siguiente. Esta secuencia, sin embargo, no requiere una progresión estricta y lineal en ellos. [34]

- **Arquitectura de Juego de Instrucciones (ISA)**: Conjunto de instrucciones que un procesador puede entender y ejecutar. Estas instrucciones definen las operaciones que puede realizar el procesador, como aritméticas, operaciones lógicas y movimiento de datos. Actúa como interfaz entre el hardware y el software, permitiendo a los programadores escribir código que pueda ser ejecutado por el procesador. [21]

- **Calidad del software**: Mide el grado de cumplimiento de los requerimientos formulados por el usuario, de tal forma que si un programa no cumple con alguno de estos requerimientos, es un software de baja calidad. Además del criterio de cumplimiento de los requerimientos existen condiciones implícitas que el software debe cumplir como son eficiencia, seguridad, integridad y consistencia, entre otras. La calidad del producto software se puede interpretar por lo tanto como el grado en que dicho producto satisface los requisitos de sus usuarios aportando de esta manera un valor. [32], [18]

- **Entorno IoT (Internet Of Things)**: Red abierta y completa de objetos inteligentes con capacidad para autoorganizarse, compartir información, datos y recursos, reaccionar y actuar ante situaciones y cambios del entorno. Su arquitectura está convencionalmente formada por una **capa de sensores**, la más baja de la arquitectura IoT encargada de recopilar datos, constando de redes de sensores, sistemas integrados y otras formas de sensores desplegados sobre el terreno; **capa de red**, encargada de transmitir la información recogida por los sensores a la siguiente capa; **capa de gestión de servicio**, que actúa como intermediario bidireccional entre la capa de red y de aplicación, siendo responsable de la gestión de dispositivos y de la información; y **capa de aplicación**, que provee la interfaz de usuario para que este pueda utilizar el sistema IoT. [22], [33]

- **Extensor de Red**: Dispositivo que repite la señal del módem, tomando la señal del módem o router y retransmitiéndola, con el fin de ampliar la cobertura de éste. Esta red extendida puede ser accedida a través de una nueva señal que tiene las mismas características que la del router original, como el nombre de la red (SSID) y la contraseña. [9]

- **Firewall de Red**: Un cortafuegos es un dispositivo de seguridad de red que separa una red interna de confianza de una red externa considerada no fiable, como Internet. Regula el tráfico de red entrante y saliente basándose en reglas de seguridad preestablecidas. Los cortafuegos son fundamentales para proteger las redes de accesos no autorizados, actividades nocivas y amenazas potenciales, y pueden existir en forma de hardware, software, software como servicio (SaaS) o nube pública o privada (virtual). [8]

- **GPIO (General Purpose Input/Output)**: Un puerto GPIO maneja tanto señales digitales entrantes como salientes. Como puerto de entrada, puede utilizarse para comunicar a la CPU las señales ON/OFF recibidas de interruptores, o las lecturas digitales recibidas de sensores. Como puerto de salida, puede utilizarse para controlar operaciones externas basadas en instrucciones de la CPU y resultados de cálculos. Cada pin puede configurarse libremente para funcionar como entrada o salida. [31]
- **I2C (Inter-Integrated Circuit)**: Protocolo de comunicación serial caracterizado por permitir la comunicación entre múltiples dispositivos en un bus de datos utilizando únicamente dos líneas, una para la señal de datos (SDA) y otra para la señal del reloj (SCL). En el bus cada dispositivo tiene una dirección única y la comunicación se realiza mediante un sistema de maestro y esclavo. El dispositivo maestro controla el reloj y envía comandos para leer o escribir datos, mientras que los dispositivos esclavos responden a estas solicitudes. [27]
- **IDE (Integrated Development Environment)**: Un entorno de desarrollo integrado (IDE) es una aplicación de software que ayuda a los programadores a desarrollar código de software de manera eficiente. Aumenta la productividad de los desarrolladores al combinar capacidades como editar, crear, probar y empaquetar software en una aplicación fácil de usar. [2]
- **Interrupción**: señal recibida por el procesador de una computadora, que indica que se debe detener el curso de ejecución actual y pasar a ejecutar código específico para tratar esta situación. Es necesario realizar un cambio de contexto en el que se almacene el estado del programa antes de la señal para poder continuar su ejecución normal una vez finalice la interrupción. [7]
- **MQTT (Message Queuing Telemetry Transport)**: MQTT es un protocolo de mensajería para el Internet de las Cosas (IoT). Está diseñado como un transporte de mensajes basado en la publicación/suscripción extremadamente ligero que resulta ideal para conectar dispositivos remotos con una huella de código pequeña y un ancho de banda de red mínimo. En él, los clientes publican mensajes en temas específicos y se suscriben a temas para recibir mensajes. [24]
- **ODS (Objetivos de Desarrollo Sostenible)**: Agenda aprobada en 2015 por todos los países miembros de la Organización de las Naciones Unidas que está compuesta por 17 Objetivos de Desarrollo Sostenible para que los países y sus sociedades emprendan un camino con el que mejorar la vida de todas las personas. Forma parte de la Agenda 2030, en la cual se establece un plan para alcanzar los Objetivos en 15 años. [28]
- **Punto de Acceso (AP)**: Dispositivo de red que une redes cableadas e inalámbricas, creando una red inalámbrica normalmente a través de Wi-Fi que permtie que dispositivos puedan ser interconectados a una red sin necesidad de cableado. Dependiendo del tipo pueden o no contar con un firewall. [19]
- **RISC-V**: Es una Arquitectura de Juego de Instrucciones de código abierto que ha crecido rápidamente desde su creación. Gracias a sus ventajas, como la ausencia de costes de licencia, la flexibilidad y la escalabilidad, se ha convertido gradualmente en un motor clave de la innovación mundial en el ámbito de los chips.

Aunque ha alcanzado un éxito significativo en los últimos años, su estándar ISA aún está en fase de desarrollo y debe seguir mejorándose para diversos escenarios. [10]

- **U-Boot**: Boot loader para placas embebidas basadas en PowerPC, ARM, MIPS y otros procesadores que puede instalarse en una ROM de arranque. Su función principal es preparar el entorno del sistema y cargar el sistema operativo desde un dispositivo de almacenamiento en el momento del arranque. Es ampliamente utilizado para cargar el kernel en sistemas operativos Linux. [40]

1.5. Organización del documento

Este documento recoge toda la información de interés sobre el trabajo realizado en el proyecto. Se encuentra dividido en cuatro partes bien diferenciadas:

1. **Prolegómeno**: En esta primera parte se encuentran aquellos capítulos que abarcan la introducción al proyecto, sus antecedentes y el plan de gestión que se ha seguido durante su realización. Permite obtener una visión general de en que consistirá el proyecto, los objetivos del mismo y que enfoque se seguirá durante su posterior desarrollo.
2. **Desarrollo**: En esta segunda parte se recoge todo lo relativo a cada una de las fases de la metodología de desarrollo que se ha seguido. Permitirá conocer con exactitud todas las decisiones tomadas a lo largo de la realización del proyecto, todos los pasos seguidos en cada una de las fases y los problemas y desafíos superados.
3. **Epílogo**: En esta tercera parte se agrupan las conclusiones alcanzadas tras la finalización del proyecto, la información sobre las licencias del software utilizado y la bibliografía empleada.
4. **Anexos**: En esta última parte quedarán recogidos los manuales necesarios para el manejo del sistema resultado del desarrollo. Incluirá el manual de desarrollador dirigido a aquellos desarrolladores que desean extender o modificar el sistema, así como el manual de usuario destinado al usuario final en el que se indican las instrucciones de uso del sistema.

Además, se ha entregado junto a este documento en soporte informático los códigos desarrollados para los microcontroladores del sistema, así como el flujo usado por el middleware. Este software desarrollado se encuentra dividido en dos tipos:

- **Resultados finales del proyecto**, compuesto por el programa de los microcontroladores y el flujo del middleware finales.
- **Pruebas**, que contendrá aquellos códigos desarrollados para probar de manera aislada un elemento individual o subsistema del entorno IoT, simplificando así la depuración de errores.

2. Antecedentes

En este capítulo se detalla la situación actual que origina el desarrollo del sistema informático. Asimismo, se describen las alternativas tecnológicas existentes.

2.1. Contexto

La labor de desarrollo realizado en este proyecto busca abordar el problema que tienen muchos docentes de no conocer el comportamiento que tienen sus alumnos durante las lecciones. Pese a que siempre es viable realizar algunas pausas para preguntar a sus alumnos, estas paradas no pueden ser demasiado abundantes ya que causarían grandes retrasos en la programación de la asignatura. Por lo tanto, muchos profesores se ven en la obligación de continuar impartiendo su lección sin llegar a saber si sus discípulos están prestando atención y entienden los contenidos de la misma, o si por el contrario se han distraido o han perdido el interés. Por esta misma razón en este proyecto se desea desarrollar una herramienta que permita al profesor conocer, a través de diversos datos obtenidos que son invisibles al ojo humano, una estimación sobre el grado de participación activa y pasiva que realizan los alumnos en las clases.

En lo que al nivel tecnológico en la mayoría de aulas de centros educativos respecta, este es muy limitado, siendo lo más habitual en aquellas clases más tecnológicamente avanzadas contar únicamente con un ordenador de sobremesa en la mesa del profesor y un proyector para poder compartir con los alumnos la información que el profesor desee desde su computadora.

Por otro lado, desde el punto de vista del proceso de negocio habitual de una clase, la secuencia que se sigue en la impartición de una lección varía mucho de un profesor a otro, aunque la mayoría suele seguir un formato en el que durante la mayor parte del tiempo explican los contenidos que se encuentran en un libro o diapositivas, realizando breves pausas para hacer alguna actividad o ejemplo práctico, normalmente en puntos específicos de la lección o al final de la misma. En estos largos periodos de explicaciones es especialmente habitual que los alumnos pierdan interés por lo que se está explicando, de manera que la herramienta de monitoreo sería extremadamente útil durante estas fragmentos de la lección para que el profesor puediese decidir en tiempo real cuando intercalar de una manera óptima aquellos momentos interactivos con los alumnos; reflejándose esto en un mayor nivel de aprendizaje por parte de estos últimos.

2.2. Estado de la técnica

La mayoría de herramientas utilizadas habitualmente por los profesores para gestionar el grado de atención de sus alumnos suelen ser técnicas tradicionales, como realizar preguntas al azar sobre los contenidos que se han explicado recientemente,

establecer incentivos a aquellos alumnos que muestren interés por la asignatura, realización de exámenes y otros mecanismos de evaluación y observación directa. Pese a que muchas de estas técnicas están muy extendidas y permiten obtener mejores índices de atención, cuentan con una serie de limitaciones intrínsecas, como puede ser que algunas solo se pueden aplicar sobre un grupo reducido de alumnos a la vez o tienen un caracter retroactivo en el que se debe de esperar largos periodos de tiempo (en ocasiones incluso meses) antes de poder medir los resultados.

Por esta misma razón, cada vez son más habituales soluciones tecnológicas que permiten reducir la brecha existente entre profesores y alumnos, como por ejemplo el uso de plataformas digitales como ClassDojo, Edmodo o Google Classroom que permiten a los docentes crear, compartir, ajustar y evaluar materiales de manera online con sus alumnos; o software de monitoreo y control de dispositivos en aquellas aulas donde haya equipamiento informático que pueda ser usado por los alumnos, permitiendo garantizar que nadie haga un uso indebido de los mismos. Sin embargo, a pesar de ser herramientas sumamente potentes, ninguna de ellas logra cumplir el papel que el sistema IoT de este proyecto pretende desempeñar.

La única solución que podría considerarse remotamente cercana sería instalar una o varias cámaras en el techo del aula, que se encargarían de grabar el comportamiento de los alumnos para poder ser analizado posteriormente por el profesor o automáticamente en tiempo real por alguna Inteligencia Artificial entrenada para reconocer patrones de déficit de atención. Sin embargo, esta alternativa no se utiliza en prácticamente ningún centro educativo, ya que es una técnica excesivamente costosa e invasiva, que resultaría incómoda tanto para alumnos como para profesores. Además, plantearía serias preocupaciones éticas y morales entre las familias de los estudiantes.

Por lo tanto, la herramienta que se busca desarrollar en este proyecto es una alternativa innovadora a uno de los problemas más comunes en la enseñanza presencial. Esta herramienta puede combinarse con técnicas tradicionales y materiales educativos online, permitiendo a los docentes llevar un paso más allá su capacidad para transmitir sus conocimientos y pasión por los temas tratados en las lecciones a sus alumnos.

3. Plan de gestión de proyecto

En esta sección se describen todos los aspectos relativos a la gestión del proyecto: metodología, organización, costes, planificación, riesgos y aseguramiento de la calidad.

3.1. Metodología de desarrollo

El modelo ADDIE es un marco ampliamente utilizado para el diseño de programas educativos y de formación. Desarrollado en 1975 en la Universidad de Florida, este modelo ha sido esencial para la creación y evaluación de experiencias de aprendizaje. El acrónimo ADDIE representa las cinco fases principales del proceso: Análisis, Diseño, Desarrollo, Implementación y Evaluación. Se trata de un proceso iterativo y no lineal, tal y como se puede ver en la Figura 3.1.

Figura 3.1: Fases del modelo ADDIE [6]

Debido a la estrecha vinculación de este proyecto con el proceso educativo, se ha decidido dividir el desarrollo del trabajo en las cinco fases principales de este modelo:

- **Análisis**: Se identificará y comprenderá las necesidades que el proyecto deberá satisfacer, las características del público objetivo y del entorno físico y se definirán los elementos y la arquitectura del entorno IoT que darán forma al desarrollo del proyecto.
- **Diseño**: Se planificará de forma detallada los componentes exactos a utilizar, su configuración e interconexiones; además de diseñar a nivel conceptual los diferentes elementos software a programar. Se tomará como base en la toma de decisiones las diferentes necesidades identificadas en la fase de análisis.

- **Desarrollo**: En esta fase se realizará la construcción del sistema IoT en base a los diferentes diseños establecidos en la fase antrior. Se realizará un desarrollo metódico e incremental, comenzando por aquellos elementos más simples y extendiendo los programas y subsistemas generados hasta alcanzar la construcción del sistema final.
- **Implementación**: Se realizarán pruebas sobre el sistema desarrollado en la fase anterior. Estas incluyen pruebas unitarias y de subsistemas, así como pruebas del sistema completo en un entorno de pruebas controlado. Esta fase culminará con la implementación y pruebas del sistema en un aula real de una institución de docencia.
- **Evaluación**: Esta fase es un poco difente a las demás, ya que el Análisis, Diseño, Desarrollo e Implementación seguían un orden lineal, mientras que por el contrario la evaluación es llevada a cabo al mismo tiempo que estas cuatro. En esta fase se evaluará de manera continuada que la funcionalidad y calidad del sistema desarrollado se ajusta a los requisitos del mismo, permitiendo realizar cambios en cualquier momento sobre cualquiera de las otras fases si se considerase necesario. Además, será en esta fase donde se identifiquen y solucionen los problemas que vayan apareciendo durante todo el desarrollo del proyecto.

El uso de esta metodología en el proyecto tiene una serie de ventajas que serán de gran utilidad. La primera de ellas es que seguir esta metodología permite establecer cada fase del proceso de desarrollo de una manera bien estructurada y con objetivos claramente definidos que se ajustan a las necesidades específicas del proyecto a desarrollar. [34]

Otra característica relevante de esta metodología es su simplicidad y la flexibilidad que aporta. Pese a que siga una estructura mayoritariamente lineal, esta no es rígida ya que la fase de evaluación permite identificar cuanto antes problemas en el desarrollo del proyecto, permitiendo realizar cambios sobre las fases anteriores de ser necesario. Este comportamiento iterativo del sistema se ajusta a la perfección con un proyecto dinámico en el que los cambios pueden resultar necesarios en cualquier momento como es el caso de este trabajo.

3.2. Planificación del proyecto

El proyecto ha sido realizado en el transcurso de algo menos de 6 meses, comenzando a principios de Marzo y finalizando a finales de Agosto. Contar con un tiempo tan limitado para desarrollar este proyecto ha sido uno de los aspectos más característicos del mismo, ya que ha influido en la mayoría de tomas de decisiones a lo largo de todo el trabajo realizado. Por esta razón, ha sido de especial relevancia realizar una adecuada planificación de las tareas que se debían realizar y de la estimación temporal de cada una de ellas, para evitar situaciones en las que se destinase demasiado tiempo en aspectos innecesarios que se salen de los objetivos del proyecto.

Las diferentes tareas en las que se han descompuesto el proyecto se citan a continuación:

1. **Planificación del proyecto**: incluye la definición del alcance y objetivos del proyecto, elección de la metodología y definición de los procedimientos para la gestión del proyecto.
2. **Análisis del sistema**: Primera fase en la metodología ADDIE. Se realizará un análisis de las diferentes necesidades y elementos que molderarán las posteriores fases del desarrollo.
3. **Diseño del sistema**: Segunda fase en la metodlogía ADDIE. Se elegirán los componentes a usar y se diseñará el funcionamiento e interconexión del sistema y sus componentes.
4. **Desarrollo del sistema**: Tercera fase en la metodología ADDIE. Se dividirá en tres partes, la programación de los microcontroladores, configuración de la placa middleware e integración de todos los subsistemas desarrollados. Se requerirá de los componentes hardware para poder constuir el sistema.
5. **Implementación del sistema**: Se realizarán pruebas unitarias, de subsistema y sistema completo en un entorno de pruebas y en una clase real.
6. **Evaluación del sistema**: Quinta fase en la metodología ADDIE. Medirá el cumplimiento de los requisitos funcionales, no funcionalidades y los problemas encontrados en el proyecto. Esta tarea se realizará de manera continuada a medida que avanzan las otras cuatro fases ADDIE.
7. **Documentación**: Consiste en la elaboración del documento final en el que se recogerá toda la información de interés referente al proyecto realizado.

En la Tabla 3.1 se muestra la aproximación inicial del tiempo que se espera dedicar a cada una de las tareas, junto a sus fechas de inicio y fin previstas. Esta planificación se puede ver representada en el diagrama de Gantt de la Figura 3.2, en el que se puede apreciar las fechas de inicio y fin de cada fase respecto al inicio del proyecto.

Tabla 3.1

Planificación temporal inicial del proyecto

Planificación inicial				
Tarea	**Fecha de inicio**	**Fecha de fin**	**Horas estimadas**	**Duración (Días)**
Tarea 1 - Planificación del Proyecto	02/03/2024	10/03/2024	15	9
Tarea 2 - Análisis del Sistema	11/03/2024	24/03/2024	15	14
Tarea 3 - Diseño del Sistema	25/03/2024	14/04/2024	25	21
Tarea 4 - Configuración del Middleware	15/04/2024	28/04/2024	20	14
Tarea 5 - Programación Microcontroladores	29/04/2024	19/05/2024	30	21
Tarea 6 - Interconexión de subsistemas	20/05/2024	02/06/2024	15	14
Tarea 7 - Pruebas Piloto	06/05/2024	09/06/2024	15	35
Tarea 8 - Pruebas en Aula	10/06/2024	14/06/2024	5	5
Tarea 9 - Evaluación del Sistema	18/03/2024	23/06/2024	15	98
Tarea 10 - Redacción de Documentación	24/06/2024	28/07/2024	50	35

En base a estas estimaciones, inicialmente se consideró que el proyecto sería completado en aproximadamente 205 horas, finalizando 149 días después de haber iniciado.

Para evaluar la precisión de esta planificación inicial, en la Tabla 3.2 se muestran los valores reales de tiempo invertido en cada una de las tareas, sus fechas de inicio y fin, y la desviación en número de días entre la fecha de fin real y la planificada para cada una de las tareas.

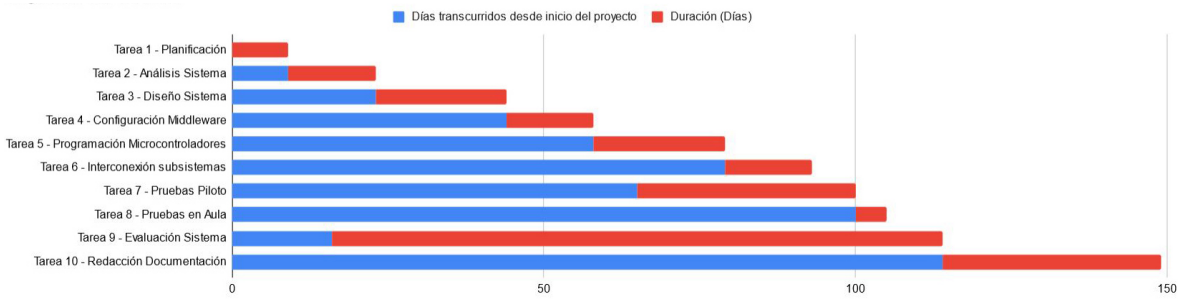


Figura 3.2: Diagrama Gantt de la planificación inicial de las fases del proyecto

Tabla 3.2

Desviación temporal respecto a la planificación inicial

Tiempo real invertido				
Tarea	**Fecha de inicio**	**Fecha de fin**	**Horas trabajadas (Aprox.)**	**Desviación (Días)**
Tarea 1 - Planificación del Proyecto	02/03/2024	10/03/2024	12	0
Tarea 2 - Análisis del Sistema	11/03/2024	24/03/2024	13	0
Tarea 3 - Diseño del Sistema	25/03/2024	14/04/2024	27	0
Tarea 4 - Configuración del Middleware	15/04/2024	28/04/2024	23	0
Tarea 5 - Programación Microcontroladores	29/04/2024	02/06/2024	35	14
Tarea 6 - Interconexión de subsistemas	03/06/2024	07/07/2024	22	35
Tarea 7 - Pruebas Piloto	06/05/2024	14/07/2024	18	35
Tarea 8 - Pruebas en Aula	15/07/2024	16/07/2024	2	32
Tarea 9 - Evaluación del Sistema	18/03/2024	21/07/2024	18	28
Tarea 10 - Redacción de Documentación	29/07/2024	25/08/2024	75	28

Como se puede observar, durante las primeras fases la planificación original se cumplió con gran precisión. Sin embargo, a principios del mes de Mayo la atención que se podía dedicar al proyecto se tuvo que dividir con otras tareas académicas, como es el caso de prácticas empresariales, lo que redujo considerablemente el tiempo disponible para trabajar semanalmente en el mismo. Esto ocasionó retrasos en las fechas de finalización de cada una de las tareas a partir de este mes.

A pesar de esto, la estimación inicial del número de horas dedicadas a cada una de las fases se cumple con relativa precisión. La única fase que supuso un número considerablemente superior de horas trabajadas respecto a las planificadas originalmente fue la redacción de la documentación, que duró 25 horas adicionales. El número de horas totales que se ha trabajado en el proyecto es de 245 aproximadamente y este ha finalizado 176 días después de su inicio.

3.3. Organización

El proyecto ha sido ejecutado por un equipo compuesto por tres personas, quienes han desempeñado los siguientes roles durante la duración del mismo:

- El alumno Javier Tena Wilson ha desempeñado el papel del autor de este proyecto. Ha realizado en su totalidad el análisis, diseño, desarrollo, implementación y evaluación del sistema; la documentación del trabajo realizado y todos los procesos seguidos para la gestión del proyecto. La selección tanto de todo el hardware como el software utilizado en el desarrollo han sido decisiones tomadas bajo su criterio.

- Los docentes académicos D. Nestor Mora Nuñez y D. Alfonso García de Prado Fontela han actuado como tutor y cotutor de este TFG respectivamente. Ambos han aportado a modo de préstamo los diferentes recursos hardware utilizados. Además, durante las reuniones realizadas mensualmente para el seguimiento del proyecto, han ayudado a evaluar como evolucionaba el sistema, además de aportar ideas, opiniones y propuestas para posibles mejoras.

3.4. Presupuesto

A continuación se detalla el presupuesto de costes asociados al proyecto. Nótese que esta es solo una propuesta y, como tal, puede ser sometida a cambios y adecuaciones al ser dependiente del mercado. Los costes del proyecto pueden ser divididos en costes asociados al personal y costes de recursos hardware y software empleados.

El número de horas totales trabajadas en el desarrollo del proyecto ha sido aproximadamente de 170 horas, sin considerar el tiempo dedicado a la documentación del mismo que ha rondado las 75 horas. A la hora de calcular los costes que irían asociados al personal de este trabajo hay que considerar que el proyecto ha abarcado diferentes disciplinas de la Ingeniría Informática, como son la programación de microcontroladores, análisis de requisitos y diseño de sistemas entre otros; siendo el salario habitual de los puestos responsables de cada uno de estos cargos un tanto diferentes; por lo que se ha decidido que se utilizará el sueldo medio de un Ingeniero Informático en España, que es de unos 27.000€ anuales o 13,85€ la hora [38]. Con estos datos, se puede calcular que, si el proyecto hubiese sido realizado por una sola persona bajo contrato, el coste del personal habría supuesto un gasto total de 2354,5€ por el desarrollo del proyecto y de unos 1038,75€ por la documentación del mismo.

Respecto al coste de los componetes hardware y software empleados, se ha desarrollado la Tabla 3.3 en la que se recoge el coste de cada uno de los elementos utilizados (se ha usado el precio más bajo encontrado en páginas de compra fiables), el número de unidades que se han necesitado y el coste total:

Tabla 3.3

Costes de componentes del sistema

Costes componentes			
Producto	**Precio Unitario**	**Unidades**	**Coste Total**
ESP32-C3 Dev Board	7,44€	2	14,88€
Placa VisionFive 2	93,35€	1	93,35€
Extensor Wi-Fi AC750 Mesh	27,00€	1	27,00€
USB Módulo Wi-Fi	5,00€	1	5,00€
Sensor PIR501	3,00€	2	6,00€
Sensor GY-302 BH1750	3,00€	2	6,00€
Sensor DHT22	2,00€	2	4,00€
Cargador + USB	5,00€	3	15,00€
Componentes montaje físico	5,00€	2	10,00€
Software Utilizado	0,00€	-	0,00€

En base a estos datos, se estima que el coste total de los componentes hardware y software del proyecto alcanzan la suma de 182,23€. Por suerte, la mayoría de componentes han sido recibidos como préstamo del departamento al que pertenecen los supervisores del TFG, por lo que el precio real de los elementos utilizados en la creación del sistema es inferior a 20€. Por otro lado, como se puede observar todo el software utilizado en el desarrollo ha sido gratuito, por lo que no ha supuesto ningún coste añadido.

Para finalizar, se realizará un breve visión de como se podría comercializar este sistema, ya que a pesar que no se busca que sea un producto comercial este análisis puede ayudar a evaluar su viabilidad económica. El sistema podría venderse como dos conjuntos de piezas por separado, de manera que se ofrece a los potenciales clientes flexibilidad a la hora de comprar el sistema según sus necesidades:

- **Base del sistema**, formado por una placa VisionFive 2, un extensor de red y un USB Módulo Wi-Fi, todos ellos debidamente configurados. En total, todas estas piezas sumarían un coste de construcción de 125,35€. Se estima que la preparación de este conjunto de piezas requeriría alrededor de 3 horas de trabajo, por lo que supondría un gasto de montaje por parte del personal de 41,55€. La preparación de esta pieza implica la instalación del SO en la placa middleware, la instalación y configuración de su software y la configuración de la red generada por le extensor Wi-Fi. El coste total asociado a la manufacturación de esta pieza es por lo tanto de 166,9€.

  Se trataría de una única compra que establecería la base necesaria para el funcionamiento del sistema. Un precio justo de venta de este paquete sería entre 225 y 300€, lo cual cubriría los costes de adquisición de las piezas y del personal encargado de configurarlo, obteniendo un buen beneficio sin llegar a ser demasiado costoso para el cliente.

- **Caja de sensores**, que contendría un microcontrolador, un sensor de cada tipo y los demás elementos necesarios para su montaje físico. El coste de construcción de este conjunto de piezas sería de 20,44 €. Se estima que el tiempo de montaje de este conjunto sería también de 3 horas, por lo que el coste asociado al personal para su ensamblaje vuelve a ser de 41,55€. El trabajo de montaje incluye configurar el sensor de movimiento, modificar y cargar el programa en los microcontroladores, realizar las conexiones físicas entre el microcontrolador y sus sensores y realizar los agujeros por los que saldrán los sensores en la caja de plástico en la que se va a guardar el circuito. De esta manera, el gasto total que tendría construir esta pieza del sistema es de 61,99€.

  Esta pieza se colocaría en una zona de la clase y permitiría recoger los datos que son enviados a las piezas de la base del sistema, debiendo adquirirse tantas unidades de este conjunto como ubicaciones de la clase se deseen monitorear. Un precio de venta adecuado sería entre 80 y 120€ para hacer frente a los gastos y tener un buen margen de beneficios.

Los cargadores no estarían incluidos en ninguno de estos paquetes, por lo que el cliente podría comprarlos por separado o utilizar los suyos propios.

De esta manera, un potencial cliente necesitaría adquirir inicialmente una unidad

del conjunto 'Base del sistema' y tantas piezas de 'Caja de sensores' como considere necesario para monitorear la clase. Además, en caso de que en un futuro desee aumentar su sistema, podría adquirir más cajas de sensores al mismo precio. Este modelo de negocio brindaría la posibilidad de adoptar el sistema desarrollado en muchos entornos educativos, al estar completamente configurado y ser fácil de instalar, además de ser económicamente asequible para la mayoría de instituciones.

Antes de comercializar el sistema, sin embargo, sería necesario desarrollar una aplicación móvil que permitiese al cliente configurar parámetros de manera sencilla en la placa VisionFive 2, en el extensor de red y en los microcontroladores; como por ejemplo las direcciones IPs.

Es importante considerar que ni en el presupuesto realizado para el proyecto ni en el modelo de negocio propuesto se contemplan los posibles gastos que puedan ir asociados al mantenimiento del sistema. En el caso de este último, sin embargo, se podría adoptar un enfoque similar al modelo de reparación que muchos fabricantes hardware siguen, en el cual cuando un cliente envíe un componente que necesite reparación se le proporcionaría un presupuesto detallado de lo que la empresa le cobraría por arreglárselo según el tipo de fallo detectado, dejándolo en manos del comprador decidir si desea que la pieza sea reparada o no.

3.5. Riesgos

En este apartado se identifican y analizan los riesgos potenciales que podrían afectar el desarrollo del proyecto. Cada riesgo se evalúa en términos de su impacto y probabilidad de ocurrencia, y se proponen planes de contingencia para mitigar los efectos adversos o reducir la probabilidad de que estos riesgos se materialicen.

1. **Fallo de componentes hardware**
    - **Descripción**: Existe la posibilidad de que alguno de los componentes hardware (como sensores, microcontroladores o el middleware) falle durante el desarrollo o implementación.
    - **Impacto**: Medio-Alto. Un fallo en los componentes podría detener completamente el progreso del proyecto, requiriendo tiempo adicional para diagnósticos, reemplazo y posible recalibración del sistema. Sin embargo, el nivel de impacto dependerá en gran medida del tipo de dispositivo que haya fallado.
    - **Probabilidad**: Media-Baja. Aunque los componentes son robustos, siempre existe el riesgo inherente de fallo, debido a posibles problemas en la manufacturación de las piezas.
    - **Plan de Mitigación**: Mantener si es posible componentes de repuesto y realizar pruebas regulares para detectar fallos lo antes posible. En caso de que se detecte que una pieza es defectuosa, reemplazarla y, si es necesario, reajustar la parte afectada del sistema.
2. **Incompatibilidad del middleware basado en RISC-V**

- **Descripción**: La placa VisionFive 2 basado en la arquitectura RISC-V puede presentar problemas de compatibilidad con algún componente software o herramienta seleccionada para el proyecto.
- **Impacto**: Alto. Si se detecta que el middleware no es compatible con ciertas herramientas clave podría ser necesario cambiar la plataforma física que actuará como middleware, lo que implicaría cambios y retrasos significativos en el proyecto.
- **Probabilidad**: Media-Baja. Aunque RISC-V cuente con cada vez más soporte, no está tan avanzado como otras arquitecturas más tradicionales, por lo que siempre existe la posibilidad de que este problema se manifieste.
- **Plan de Mitigación**: Realizar pruebas de compatibilidad lo antes posible. Si se detectan incompatibilidades, considerar migrar a una plataforma más estable o con mejor soporte como Raspberry Pi, y reajustar el proyecto en consecuencia.

3. **Software elegido inadecuado**
   - **Descripción**: Alguno de los softwares seleccionados podría no cumplir con las expectativas o presentar limitaciones que dificulten la implementación del sistema.
   - **Impacto**: Medio. Podría llegar a requerir un cambio en el software seleccionado y, posiblemente, ajustes en el código y la configuración del sistema.
   - **Probabilidad**: Baja. Pese a que la elección de software se hizo basándose en su estabilidad y popularidad, lo que reduce significativamente la probabilidad de ocurrencia de este problema, no se puede descartar completamente la posibilidad de que no cumpla con todos los requisitos del proyecto.
   - **Plan de Mitigación**: Realizar pruebas piloto con cada uno de los programas software seleccionados en entornos controlados antes de la implementación completa del sistema. Si un software no es adecuado, identificar alternativas y realizar una migración de los componentes software necesarios.
4. **Restricciones de red y configuración del firewall**
   - **Descripción**: La red en la que se implementará el sistema IoT puede tener restricciones de firewall o configuraciones de seguridad que impidan la correcta comunicación entre los dispositivos y el servidor MQTT.
   - **Impacto**: Alto. Si el sistema no puede comunicarse a través de la red debido a restricciones, el proyecto no podrá funcionar correctamente.
   - **Probabilidad**: Media. Las políticas de seguridad de red son cada vez más estrictas para prevenir posibles ataques, existiendo una posibilidad real de que los dispositivos del entorno IoT sean bloqueados al tratar de usar la red de la institución.
   - **Plan de Mitigación**: En caso de que este problema llegase a manifestarse se crearía una red independiente de la del centro educativo a la que se puedan

conectar los componentes del sistema utilizando la funcionalidad de Punto de Acceso móvil en un Smartphone.

5. **Problemas de integración del sistema**
   - **Descripción**: Pueden surgir dificultades al integrar todos los componentes del sistema, lo que podría causar retrasos o mal funcionamiento.
   - **Impacto**: Medio. Los problemas de integración pueden requerir tiempo adicional para resolver conflictos entre componentes o reajustar el diseño del sistema.
   - **Probabilidad**: Media-Baja. La integración de múltiples componentes siempre conlleva riesgos inherentes, aunque un correcto análisis y diseño del sistema puede minimizar su posibilidad de ocurrencia.
   - **Plan de Mitigación**: Adoptar un enfoque incremental para la integración del sistema, probando la interoperabilidad de los componentes a medida que se interconectan gradualmente. Esto permitirá identificar y aislar los problemas en etapas tempranas y facilitará la resolución de los mismos.

3.6. Gestión de versiones

Debido al carácter incremental que se ha seguido en el desarrollo del código fuente, no se ha utilizado ningún software para gestionar las versiones ya que no ha sido necesario. En su lugar, durante el desarrollo se fueron creando inicialmente los códigos para controlar cada uno de los componentes individuales del sistema, y una vez se hubieron finalizado, estos se han guardado para ser usados como pruebas unitarias y de subsistemas. El código final desarrollado toma como base los cimientos establecidos en estos códigos individuales, uniéndolos todos ellos en uno solo para obtener el programa final.

En la carpeta de códigos que acompaña a este documento se puede encontrar el código fuente final del proyecto y las versiones anteriores que han sido convertidas en pruebas de los componentes individuales del sistema.

3.7. Aseguramiento de calidad

El correcto funcionamiento del sistema no es el único criterio que se seguirá a la hora de tomar decisiones a lo largo del proyecto. También se buscará que el sistema y software desarrollados se alineen con las características propias de la calidad software, siendo de especial interés para este proyecto los siguientes: [13]

- **Fiablidad**: capacidad del sistema para funcionar de manera continua y sin fallos durante un periodo prolongado.
- **Seguridad**: capacidad del sistema para proteger la integridad y confidencialidad de los datos, y evitar accesos no autorizados.

- **Privacidad**: capacidad del sistema de proteger la información personal de los usuarios, garantizando que no se recolecten, almacenen ni procesen datos que puedan permitir la identificación individual de los usuarios sin su consentimiento explícito.
- **Usabilidad**: mide la facilidad de uso del sistema por parte del usuario final.
- **Modularidad**: división del sistema en componentes independientes que pueden ser desarrollados, mantenidos y reemplazados de manera aislada.
- **Escalabilidad**: capacidad del sistema para manejar un aumento en la carga de trabajo sin comprometer el rendimiento.
- **Extensibilidad**: facilidad con la que el sistema puede adaptarse a nuevos requisitos o funcionalidades.
- **Eficiencia Energética**: mide el consumo de energía del sistema en relación con su rendimiento.
- **Portabilidad**: capacidad del sistema para ser trasladado y ejecutado en diferentes entornos o plataformas sin necesidad de grandes modificaciones.

Todas estas características deseadas en el sistema deberán ser consideradas a lo largo del desarrollo del proyecto, teniendo gran peso en la toma de decisiones durante la fase de análisis y diseño. La evaluación del grado de cumplimiento de la calidad del software descrito en este capítulo se realizará a nivel teórico, valorandose todas las medidas tomadas durante la duración del proyecto para lograr alcanzar estos requisitos no funcionales.

Además, se seguirán buenas prácticas de codificación y desarrollo de sistemas, como llevar a cabo una correcta documentación del código en forma de comentarios claros y explicativos; realizar un desarrollo incremental del sistema; realización de pruebas unitarias de todos los componentes del entorno; y evaluaciones periódicas de los resultados obtenidos para asegurar que se ajusten a los requerimientos del proyecto.

Finalmente, se seguirá el siguiente criterio de aceptación/rechazo de componentes: Un componente se considera aceptable si pasa todas las pruebas unitarias y funcionales sin errores significativos, además de cumplir en gran medida los criterios de calidad definidos. En caso contrario, se considerará no apto y deberá de ser rediseñado.

3.8. Actualización y mantenimiento del sistema

Una vez finalizado y desplegado el sistema final resultado de este proyecto, será necesario realizar cada 6 meses una revisión del mismo para asegurar que su comportamiento siga siendo correcto y que las lecturas realizadas por los sensores se mantengan precisas.

Para comprobar que el entorno IoT siga funcionando correctamente, se realizarán pruebas sobre el sistema completo, de manera que utilizando la interfaz de usuario se comprobará que todos los sensores de cada microcontrolador recogen datos y estos son enviados correctamente al middleware.

En caso de existir algún sensor del que no se reciban datos, se comprobarán que las conexiones físicas entre el microcontrolador y el sensor no estén fallando, tras lo que se utilizará el código de prueba unitaria de dicho sensor para ver si el problema se ha solucionado. Si este aún persiste, el fallo será del propio sensor que ha dejado de funcionar, por lo que deberá ser sustituido por uno nuevo.

Si por el contrario no se reciben datos de ninguno de los sensor de un microcontrolador, entonces se realizará una prueba del subsistema formado por ese microcontrolador y sus sensores. De esta manera se podrá detectar fácilmente si es un problema del funcionamiento del microcontrolador, un fallo de conexión a la red o que todos los sensores han fallado. En caso de que se detecte un problema de funcionamiento del microcontrolador, se sustituirá por uno nuevo.

Finalmente, si ninguno de los microcontroladores son detectados, se comprobará el estado de la red Wi-Fi IoT, de manera que se buscará probar su funcionamiento. Se usará el código que prueba la conexión de un microcontrolador a la red Wi-Fi IoT para comprobar que cada microcontrolador se conecta a la red y ver la dirección IP asignada por el servidor DHCP. En caso de detectar que la configuración de la red hubiese cambiado, se deberá reajustar de manera acorde el código de los microcontroladores y, de ser necesario, la configuración de seguridad de la red del centro educativo.

Una vez comprobado que el sistema completo funciona correctamente, será necesario probar que las lecturas ofrecidas por los sensores de cada microcontrolador son precisas. Para ello se usarán los códigos de pruebas unitarias de los sensores, de manera que se someterán a evaluaciones controladas para observar las lecturas realizadas. En el caso del sensor de movimiento se deberá realizar además una comprobación de la distancia de detección y el retardo entre dos lecturas consecutivas, reajustando los potenciómetros que controlan estos valores en caso de detectarse una descalibración. Si las lecturas ofrecidas por un sensor son erróneas o no son suficientemente precisas, se procederá a realizar una sustitución completa del sensor en cuestión.

Parte II

Desarrollo

4. Análisis

Este capítulo cubre el apartado de análisis en la metodología ADDIE, identificando y analizando los elementos necesarios en el sistema y sus requisitos técnicos, la audiencia objetivo del entorno IoT y el entorno en el que se instalará el sistema. Además, se analizarán aquellas consideraciones adicionales que junto a los requisitos funcionales y de calidad del sistema moldearán las posteriores fases de la metodología de desarrollo.

4.1. Análisis de comportamiento, requisitos técnicos y arquitectura IoT

El entorno IoT a desarrollar debe ser capaz de recoger de manera periódica datos sobre el comportamiento del alumnado y las condiciones del aula durante la impartición de una lección. Estos datos serán procesados por el sistema y presentados al profesor a través de una interfaz, permitiéndole adaptar en tiempo real la dinámica y los contenidos de la clase según las necesidades que detecte. Para cumplir este comportamiento, el entorno IoT debe estar compuesto por los siguientes elementos clave:

1. **Sensores**
   - **Función**: Los sensores serán los encargados de realizar lecturas sobre el comportamiento de los alumnos y las condiciones físicas del aula. Se utilizarán cuatro tipos de sensores:
     - Sensores de Movimiento: Detectan la actividad de los estudiantes dentro del aula cuando estos realizan algún movimiento.
     - Sensores de Temperatura y Humedad: Miden los niveles de temperatura y humedad del aula.
     - Sensores de Luz: Evalúan los niveles de iluminación dentro del aula.
   - **Requisitos técnicos**:
     - Precisión en la lectura de datos.
     - Bajo consumo energético para reducir así la cantidad de energía usada por el sistema IoT.
     - Compatibilidad y conectividad fiable con el microcontrolador

     Resaltar que inicialmente también se barajó la posibilidad de añadir 2 tipos de sensores adicionales. A continuación se describe que función habrían cumplido y por qué finalmente fueron descartados:
     - **Sensor de sonido**: Como su nombre indica, este tipo de sensor se habría encargado de realizar mediciones de los niveles de decibelios del

sonido ambiental en las distintas zonas del aula donde estuviera colocado. Esto permitiría al profesor identificar las áreas en las que los alumnos son más propensos a conversar.

Este tipo de sensor fue descartado debido a que se consideró una medida poco fiable para evaluar el comportamiento de los alumnos, ya que los sonidos ambientales provenientes del exterior del aula también afectarían a las lecturas obtenidas. Una posible solución a este problema habría sido colocar un sensor de sonido en el exterior del aula para intentar cancelar el ruido externo de las lecturas en el postprocesamiento de los datos. Sin embargo, debido a que esta solución se salía del ámbito del proyecto se decidió descartar este sensor, quedando pendiente como parte del trabajo futuro del sistema.

- **Sensor de CO2**: La función de este tipo de sensor habría sido medir la cantidad de dióxido de carbono en el aire. La razón por la que no fue adoptado en el proyecto final se debe a que era poco práctico, puesto a que este tipo de lecturas en un solo aula no revelaría ninguna información de interés. Pese a esto, este tipo de sensor es una idea sumamente interesante en caso de que en un futuro el proyecto se extendiese a múltiples aulas de un centro educativo, puesto a que permitirían saber aquellas zonas donde se genera más cantidad CO2, además de ser una buena herramienta para cuantificar la cantidad de dióixido de carbono que que se genera en las instalaciones de la institución.

2. **Microcontroladores**

   - **Función**: A estos dispositivos serán conectados los sensores y se encargarán de realizar las lecturas y enviar los datos recopilados al middleware. Cada uno de los puestos donde se desea alojar los sensores estará conformado por un sensor de cada tipo y un microcontrolador que los gestionará.
   - **Requisitos técnicos**:
     - Capacidad de procesamiento suficiente para manejar múltiples entradas de sensores simultáneamente.
     - Capacidad de poder conectarse a una red WiFi de manera inalámbrica y estable para transmitir datos al middleware.
     - Compatibilidad con alguno de los IDEs de programación de microcontroladores más populares, al contar estos últimos con bibliotecas creadas por la comunidad que pueden reducir drásticamente el tiempo de desarrollo.
     - Compatibilidad con los sensores utilizados y eficiencia en el consumo energético.
     - Basado en la arquitectura RISC-V.

3. **Middleware**

   - **Función**: Servirá como el núcleo del sistema, alojando todos los servicios

necesarios para el correcto funcionamiento del entorno IoT. Procesará y almacenará los datos enviados por los microcontroladores y los preparará para su visualización. Alojará entre otras cosas la interfaz en la que el profesor podrá ver los datos recopilados y el broker MQTT a través del que se transmitirán los mensajes entre los componentes hardware del sistema.

- **Requisitos técnicos**:
  - Ordenador monoplaca con capacidad de cómputo suficiente para el volumen de datos a manejar.
  - Compatibilidad con sistemas operativos UNIX.
  - Basado en la arquitectura RISC-V

4. **Punto de acceso**

   - **Función**: Establecerá una red inalámbrica a la que se conectarán todos los componentes del entorno IoT, enrutando el tráfico de datos entre ellos.
   - **Requisitos técnicos**:
     - Cobertura de red suficientemente amplia para alcanzar a todos los microcontroladores y middleware dentro del aula.
     - Capacidad de manejo de múltiples conexiones simultáneas.
     - Seguridad en la red generada para proteger los datos transmitidos a través de un SSID y contraseña configurables.

5. **Interfaz web**

   - **Función**: Proporcionará una plataforma visual para que el profesor acceda a la información procesada, presentando los datos de forma clara y sencilla.
   - **Requisitos técnicos**:
     - Diseño interactivo y fácil de usar, además de compatible con diferentes tipos de dispositivos como PC o móvil.
     - Capacidad de mostrar datos en tiempo real e históricos de los mismos a través de gráficas que muestren la evolución de las lecturas con el tiempo.
     - Contar con aspectos configurarables para poder modificar de manera remota el funcionamiento de algunos componentes, como el tiempo de espera entre dos lecturas consecutivas por parte de los microcontroladores.

Todos estos elementos necesarios en el entorno IoT a desarrollar pueden ser agrupados en las cuatro capas propias de la arquitectura de estos tipos de sistemas (ver 'Entorno IoT' en la sección 1.4), como se muestra en la Figura 4.1.

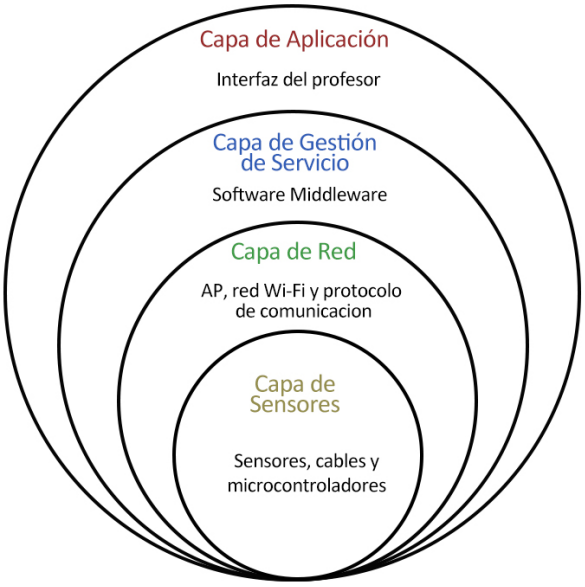


Figura 4.1: Arquitectura del entorno IoT a construir

4.2. Análisis de audiencia

Los usuarios que interactuarán con el sistema IoT pueden ser divididos en dos roles claramente diferenciados:

- El **profesorado**, que utilizará de manera activa el sistema como una herramienta para monitorizar las condiciones del aula y comportamiento de los alumnos gracias a las diferentes lecturas realizadas por los sensores repartidos a lo largo de la sala. Aprovechándose de estos valiosos datos, los docentes podrán hacerse una idea del comportamiento y grado de interés de sus alumnos, de manera que podrán adaptar la dinámica o contenidos de la lección en tiempo real cuando lo consideren necesario.

- El **alumnado**, que interactuarán de manera pasiva con el sistema IoT mientras se imparte la clase. Los valores de las lecturas realizadas por los sensores se verán afectados por el comportamiento de los estudiantes; un ejemplo que ilustra esto se puede producir cuando los alumnos comienzan a relajarse, lo que ocasiona que su respiración se ralentice gradualmente, afectando directamente a los niveles de temperatura y humedad de la sala, además de evidentemente reducirse el número de movimientos detectados.

En resumen, los alumnos serán los encargados de suministrar y afectar a los datos leidos por los sensores, los cuales serán tratados y gestionados por los componentes hardware y software del sistema para finalmente ser mostrados a los profesores, que se encargarán de modificar la lección cuando resulte oportuno, afectando de esta manera el comportamiento de los alumnos y repitiendose así este ciclo hasta que el periodo docente haya finalizado. Estas interacciones entre sistema y profesorado/alumnado queda representado en la Figura 4.2.

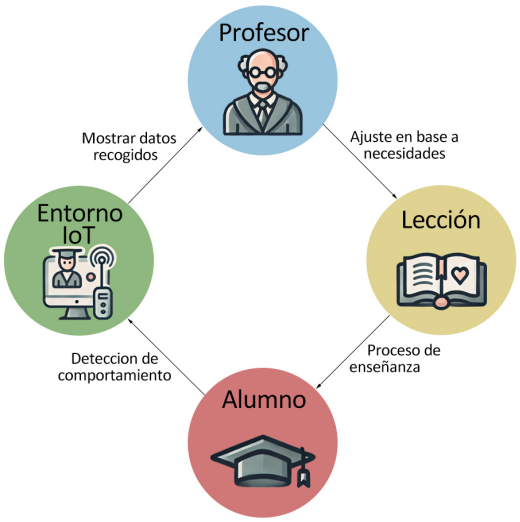


Figura 4.2: Interacción del entorno IoT con la audiencia

4.3. Análisis del entorno

Otro aspecto de especial relevancia a considerar en la fase de análisis son las características de la ubicación donde se instalará el sistema final, en este caso, un aula de una institución educativa. En particular, en este proyecto se realizará la implementación del entorno en un aula de la Escuela Superior de Ingeniería de la Universidad de Cádiz. Por lo tanto, se deben tener en cuenta las siguientes consideraciones sobre la ubicación:

- El clima natural no representará un problema, ya que ni las temperaturas ni los valores de humedad alcanzarán extremos que puedan afectar al correcto funcionamiento de los componentes hardware del entorno IoT
- Las mesas del aula son alargadas y permiten que se sienten múltiples alumnos en cada una. Por ello, el sensor de movimiento que se empleará debe ser capaz de medir distancias de hasta 4 metros, según estimaciones realizadas sobre la longitud de las mesas. Estos sensores se instalarán en el extremo de las mesas que está junto a la pared.
- Debido a la configuración de seguridad de la red interna, probablemente un extensor o punto de acceso que se conecte a la red universitaria no podrá funcionar correctamente debido a prohibiciones impuestas por el firewall de la red. Por esta razón, es necesario considerar alguna alternativa como un dispositivo móvil con la opción de punto de acceso habilitada en caso de que este problema se manifestase.

4.4. Análisis de consideraciones adicionales

Aunque el objetivo principal del proyecto es desarrollar un entorno IoT que permita a los profesores monitorear en tiempo real el comportamiento de sus alumnos y las

condiciones ambientales del aula, cumpliendo un conjunto de criterios que garanticen la calidad del sistema y del software desarrollados, existen una serie de consideraciones e intereses adicionales que afectarán a la toma de decisiones a lo largo del proyecto. Por ello, es crucial realizar un análisis de estas necesidades complementarias que el proyecto deberá satisfacer.

De manera más específica, el proyecto busca ofrecer un caso de estudio sobre la viabilidad de RISC-V como una arquitectura tecnológica para el desarrollo de sistemas IoT, además de alinear los posibles usos de la herramienta con los objetivos establecidos por la Organización de las Naciones Unidas bajo los llamados Objetivos de Desarrollo Sostenible (ODS). A continuación, se analizará cada uno de estos aspectos y cómo pueden alinearse con el proyecto.

4.4.1. RISC-V

RISC-V es una arquitectura de conjunto de instrucciones (ISA) de código abierto y libre de *royalties*. Esta arquitectura ha permitido una nueva era de innovación en procesadores, ya que ofrece modularidad y extensibilidad, lo que facilita la creación de extensiones personalizadas para necesidades específicas. Aunque actualmente es popular en procesadores embebidos, RISC-V todavía tiene que madurar para satisfacer plenamente las demandas de escenarios emergentes como la inteligencia artificial y la computación en la nube. A diferencia de otras arquitecturas de ISA como x86 o ARM, que son controladas por grandes empresas y están protegidas por licencias y patentes, RISC-V se ofrece bajo una licencia abierta, lo que permite a cualquier persona o entidad desarrollar, utilizar y modificar hardware basado en esta arquitectura sin necesidad de pagar licencias a sus creadores. [10]

Uno de los principales atractivos de RISC-V es la gran cantidad de ventajas que ofrece frente a soluciones propietarias. A continuación se detallarán las más importantes:

1. **Código abierto y sin costos de licencias**: A diferencia de las arquitecturas propietarias como x86 y ARM, RISC-V es de código abierto, lo que elimina las altas tarifas de licencias y permite un acceso más amplio y económico a la tecnología de procesadores, favoreciendo a más desarrolladores y empresas innovar sin las barreras de costos asociados a las licencias de arquitecturas propietarias

2. **Flexibilidad y escalabilidad**: RISC-V es altamente modular y escalable, lo que permite a los desarrolladores combinar diferentes extensiones de ISA y crear conjuntos de instrucciones personalizadas para adaptarse a los requerimientos específicos de cada aplicación, permitiendo crear soluciones más eficientes y adaptadas a necesidades concretas, optimizando tanto el rendimiento como el consumo de energía. Esto lo hace adecuado para una amplia variedad de escenarios, desde dispositivos embebidos hasta computación de alto rendimiento.

3. **Innovación global**: Gracias a su naturaleza abierta, RISC-V está fomentando la innovación a nivel global. Una manera en la que esto se ve reflejado es el impulso de iniciativas por parte de órganos internacionales y gobiernos de grandes potencias para promover el desarrollo de esta nueva tecnología. Ejemplos de esto

incluyen iniciativas como la 'European Processor Initiative' y la 'China RISC-V Alliance', que buscan desarrollar chips de alto rendimiento basados en RISC-V para aplicaciones como la supercomputación. Una de las princiaples causas del interés internacional por RISC-V es debida a que ofrece una alternativa a las arquitecturas dominadas por grandes corporaciones, permitiendo a los países y organizaciones reducir su dependencia de tecnologías extranjeras.

4. **Ecosistema en crecimiento**: Al ser una tecnología abierta, RISC-V ha fomentado la colaboración y el desarrollo de una amplia comunidad global. Esto ha llevado a un rápido crecimiento del ecosistema, con la creación de herramientas, software, y hardware que soportan esta arquitectura. Además, su creciente adopción por parte de empresas e instituciones académicas refuerza su sostenibilidad a largo plazo.

Por otro lado y como ya se ha comentado, la flexibilidad que ofrece RISC-V para adaptar la ISA a las necesidades específicas de cada apliación permite que pueda ser utilizado de manera eficiente un una amplia variedad de escenarios. Algunas de sus aplicaciones más habituales en la actualidad son las siguientes:

- **Microprocesadores embebidos**: RISC-V es muy utilizado en aplicaciones embebidas, como sistemas IoT; hogares inteligentes; dispositivos *wearables*, como relojes inteligentes o pulseras de actividad; y *edge computing*, un modelo de computación en el que los datos se procesan cerca de donde se generan, en lugar de en un centro de datos centralizado reduciendo así latencia y velocidad de respuesta en sistemas críticos. Su modularidad permite personalizar los microprocesadores para satisfacer necesidades específicas, como el procesamiento de señales digitales, cifrado/descifrado y tareas emergentes de inteligencia artificial.

- **Procesadores Específicos de Dominio**: Tras alcanzar el límite que impide seguir cumpliendo el principio de la Ley de Moore, que expresaba que aproximadamente cada 2 años se duplican el número de transistores en un microprocesador, se ha desplazado la innovación hacia procesadores específicos de dominio, que optimizan la arquitectura para tareas concretas. RISC-V es ideal para este tipo de procesadores debido a su capacidad de personalización, lo que permite acelerar tareas como el procesamiento gráfico, la computación en la nube, y la aceleración de algoritmos de inteligencia artificial.

- **Computación de alto rendimiento**: En los últimos años, RISC-V ha comenzado a ganar tracción en el ámbito de la computación de alto rendimiento, un sector dominado tradicionalmente por arquitecturas propietarias como x86 y ARM. La adaptabilidad y apertura de RISC-V permiten que se utilice en una variedad de aplicaciones exigentes, desde *smartphones* y portátiles hasta centros de datos y supercomputadoras. Ejemplos de esto son el procesador XuanTie C910 de Alibaba, capaz de ejecutar sistemas operativos tan avanzados como Android 12.0; y Berkeley Out of Order Machine (BOOM), un procesador desarrollado por la Universidad de Berkeley que es un núcleo de alto rendimiento basado en RISC-V.

En este proyecto, se pretende demostrar la viabilidad de RISC-V en un entorno IoT real aprovechando soluciones tecnológicas ya existentes en forma de una placa embebida RISC-V que actuará como middleware del sistema IoT y microcontroladores

basados en RISC-V que se encargarán de leer y controlar los sensores.

Si el proyecto logra cumplir los objetivos funcionales y de calidad establecidos, quedará demostrada la viabilidad de RISC-V como una arquitectura sólida capaz de ser usada como base tecnológica para el desarrollo de sistemas IoT destinados a resolver problemas reales. Además, los entornos desarrollados con esta arquitectura se beneficiarían de todas las ventajas que el uso de esta tecnología implica.

Otro aspecto relevante es que el uso de RISC-V en un proyecto universitario orientado a mejorar la calidad educativa se alinea con los orígenes de esta arquitectura, desarrollada en la Universidad de Berkeley. Esto refuerza la conexión entre el progreso tecnológico y las instituciones de educación.

4.4.2. ODS y compromiso social

Los Objetivos de Desarrollo Sostenible (ODS) constituyen un llamamiento universal a la acción para poner fin a la pobreza, proteger el planeta y mejorar las vidas y las perspectivas de las personas en todo el mundo. En 2015, todos los Estados Miembros de las Naciones Unidas aprobaron 17 Objetivos como parte de la Agenda 2030 para el Desarrollo Sostenible, en la cual se establece un plan para alcanzar los Objetivos en 15 años.

Actualmente, se está progresando en muchos lugares, pero, en general, las medidas encaminadas a lograr los Objetivos todavía no avanzan a la velocidad ni en la escala necesarias. El año 2020 debe marcar el inicio de una década de acción ambiciosa a fin de alcanzar los Objetivos para 2030. [28]

La importancia que tiene esta iniciativa no puede ser subestimada, ya que no solo es apoyada por una gran cantidad de paises a nivel mundial, sino que busca hacer que el mundo sea un lugar mejor y más justo para todos. Por esta razón, en este proyecto se busca alinear los posibles usos del entorno IoT con el cumplimiento del mayor número de objetivos posibles.

De entre todos los objetivos presentados por la ONU, este proyecto se puede alinear principalmente con los siguientes:

- **Objetivo 4: Garantizar una educación inclusiva, equitativa y de calidad y promover oportunidades de aprendizaje durante toda la vida para todos.**

  El proyecto a desarrollar contribuye directamente a este objetivo, al proporcionar herramientas tecnológicas que mejoran la calidad de la educación. La instalación del entorno IoT en las aulas permitiría a los docentes monitorear en tiempo real el comportamiento de los estudiantes, lo que a su vez les ayuda a adaptar sus métodos de enseñanza para satisfacer mejor las necesidades de aprendizaje de cada alumno. Esto se traduciría en una educación más completa y eficiente, ya que el profesor podría abordar más temario y profundizar más en aquellos puntos de especial interés en el itinerario de su asignatura, reflejándose directamente en un nivel de aprendizaje superior para los alumnos que se traducirá en mejores oportunidades en su futuro profesional y en su crecimiento personal.

Además, el sistema IoT permite facilitar la creación de entornos de aprendizaje más inclusivos, donde se pueda prestar especial atención a los estudiantes que necesitan más apoyo, se distraen con facilidad o tienen necesidades especiales. Al permitir una respuesta inmediata a las condiciones del aula, el proyecto promueve una educación más equitativa, donde todos los estudiantes tienen la oportunidad de participar activamente y de aprender en un ambiente optimizado para su éxito.

- **Objetivo 7: Garantizar el acceso a una energía asequible, segura, sostenible y moderna.**

  El proyecto también se alinea con el Objetivo 7 al integrar la eficiencia energética en el diseño y operación del entorno IoT. Al utilizar microcontroladores y sensores que son altamente eficientes en términos de consumo energético, el sistema minimiza la cantidad de energía necesaria para su funcionamiento. Además, la posibilidad de utilizar fuentes de energía renovable, como paneles solares, para alimentar los sensores, microcontroladores y middleware, refuerza el compromiso con el acceso a energía sostenible y moderna.

  Además de esto, el sistema IoT a desarrollar se puede expandir para favorecer el ahorro energético gracias a poder añadir actuadores que respondan a las lecturas de los sensores en base a las condiciones del aula. Por ejemplo, si se detecta que en un aula no hay movimientos y la temperatura y humedad es la misma que la del clima, esto podría ser interpretado como que no hay nadie en el aula, pudiéndose activar actuadores que apagasen las luces y aires acondicionados en la sala en caso de que por accidente se hubiesen dejado encendidos.

  El uso de la arquitectura RISC-V, que está diseñada para ser altamente eficiente, también contribuye a este objetivo, al reducir el consumo de energía en comparación con otras arquitecturas. Este enfoque no solo hace que el sistema sea más respetuoso con el medio ambiente, sino que también ayuda a que pueda ser utilizado en áreas donde el acceso a la energía es limitado, ampliando su alcance y aplicabilidad.

- **Objetivo 9: Construir infraestructuras resilientes, promover la industrialización sostenible y fomentar la innovación**

  El proyecto contribuye a este objetivo al promover la creación de infraestructuras educativas digitales tecnológicamente avanzadas. Al implementar un entorno IoT en las aulas, se están desarrollando nuevas infraestructuras que mejoran la eficiencia y efectividad de la enseñanza. Estos avances no solo mejoran la calidad educativa, sino que también fomentan la innovación en el sector docente.

  La elección de utilizar la arquitectura RISC-V, una tecnología emergente y abierta, refleja un interés por promover la innovación, al fomentar el uso de tecnologías que son accesibles y escalables, oferciendo una opción para que instituciones educativas adoptasen soluciones para mejorar la calidad de enseñanza sin tener que invertir gran cantidad de recursos o depender de tecnologías propietarias. De este modo, este proyecto establece una base sólida para la implementación de tecnologías innovadoras en la educación que puedan adaptarse y evolucionar con el tiempo.

Evidentemente, el actual proyecto no solo se ve alineado con los objetivos anteriormente explicados, sino que de manera indirecta favorece a muchos otros de los desafíos impuestos por la Organización de las Naciones Unidas, debido a la naturaleza interconectada de los objetivos propuestos. Por ejemplo, una educación de calidad en la que se promueva la igualdad de oportunidades entre todos los alumnos puede fomentar a que todos los estudiantes puedan optar a un futuro digno, pudiendo romper de esta manera el ciclo de la pobreza (objetivo 1) y de esta manera reducir en cierta medida las desigualdades existentes entre hombres y mujeres en muchas partes del mundo, fomentando así la igualdad de género (objetivo 5).

Otro ejemplo de este efecto dominó se puede ver reflejado en el ahorro energético, ya que esto también ayuda reducir la cantidad de residuos contaminantes que son generados por las sociedades desarrolladas a la hora de producir la energía debido principalmente al uso de fuentes no renovables como la quema de combustibles fósiles (objetivo 11), gran parte de la cual acaba contaminando el agua de ríos, lagos y océanos, haciendo que grandes fuentes de aguas sean no potables (objetivo 6) y destruyendo en el proceso los ecosistemas marinos (objetivo 14).

En definitiva, ofrecer a la juventud una educación de calidad, fomentando el pensamiento crítico, el crecimiento personal y el interés social, es una de las mejores maneras de mejorar el mundo. La historia ha demostrado con el tiempo que la educación es la base sobre la cual se construyen las sociedades más justas, equitativas y sostenibles. Cuando los jóvenes tienen acceso a una formación que les permite desarrollar habilidades para resolver problemas, comprender su entorno y contribuir de manera positiva a su comunidad, se convierten en los pilares que promueven el cambio, favoreciendo el progreso social y económico. Además, una educación de calidad no solo prepara a las personas para su futuro laboral, sino que también promueve valores de solidaridad, respeto y responsabilidad, fundamentales para la convivencia pacífica y el bienestar global. Invertir en educación es, por tanto, invertir en un futuro mejor para todos, ya que los jóvenes de hoy son quienes construirán el mundo de mañana.

5. Diseño

En este capítulo se aborda la fase de diseño de la metodología ADDIE, en la cual se determinan los modelos de componentes hardware y las herramientas y programas de software que se utilizarán en el desarrollo del proyecto. Además, se define el diseño de las interconexiones entre todos los elementos, del programa que controlará los microcontroladores, del flujo del Middleware y de la interfaz del sistema. Todas las decisiones adoptadas durante esta etapa estarán debidamente justificadas.

5.1. Elección del hardware

En esta sección se detallan los modelos de hardware seleccionados para desempeñar las funciones de los componentes necesarios en el proyecto, previamente identificados en la fase de análisis. Cabe destacar que el principal criterio de selección ha sido la disponibilidad de los productos en el departamento, ya que todas las piezas físicas utilizadas para construir el sistema han sido tomadas a modo de préstamo. De haber sido necesario adquirir cada componente, el costo total del proyecto habría aumentado considerablemente, comprometiendo su viabilidad económica. Aunque esto ha implicado un repertorio de opciones limitado y que los modelos elegidos no son los más avanzados tecnológicamente, las selecciones realizadas cumplen con los requerimientos técnicos del proyecto. A continuación, se detallan los componentes hardware que se utilizarán:

- **Sensor de humedad y temperatura DHT22**: Este sensor es capaz de leer de manera precisa y fiable los valores de humedad y temperatura del aula, ofreciendo los datos obtenidos por un único PIN de salida, facilitando de esta manera la conexión con el microcontrolador.

  Cuenta con rangos de lectura que se ajustan a las necesidades específicas de cualquier aula (temperaturas de entre -40 y 80ºC y valores de humedad relativa mostrados en porcentaje) y un tiempo de respuesta inferior a 5 segundos, de manera que cualquier cambio que se produzca será rápidamente detectado. Además, este sensor no requiere de un tiempo inicial de calibración.

  Respecto a las necesidades energéticas, este sensor tiene un consumo muy bajo, contando con un voltaje de operación de 3.3V y una corriente máxima de 1.5 mA durante la medición (su corriente en reposo es casi despreciable). Para concluir, este sensor cuenta con 4 pines de conexión, de entre los cualos únicamente 3 son necesarios (VCC, GND y DATA), tal y como se puede apreciar en la Figura 5.1. [1]

- **Sensor de luz GY-302 BH1750**: Este sensor de luz digital mide la intensidad de la luz ambiente en lux y proporciona los datos a través de una interfaz I2C.

  El GY-302 BH1750 ofrece una alta precisión, con una resolución de hasta 1 lux,

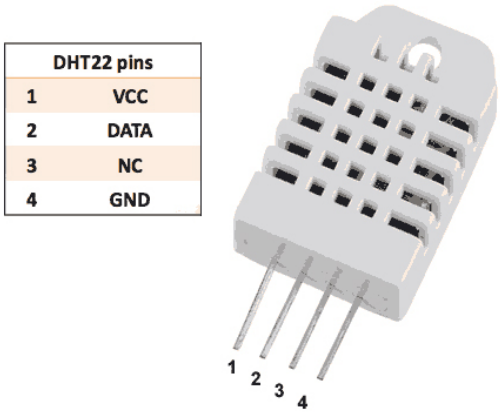


Figura 5.1: Sensor DHT22 y sus pines [17]

y un amplio rango de medición que abarca desde 0 hasta 65.535 lux. Su tiempo de lectura es rápido, con un estándar de solo 120 ms, lo que lo hace ideal para la aplicación prevista. Al igual que el sensor DHT22, el BH1750 no requiere un tiempo de calibración inicial.

Este sensor también es muy eficiente en términos de consumo energético. Opera con un voltaje de 5V y tiene una corriente máxima de 0.12 mA durante la medición. Dispone de un pinout de 5 pines (que requieren soldadura para la conexión): VCC, GND, SCL (para la línea de reloj I2C del microcontrolador), SDA (para la línea de datos I2C del microcontrolador) y ADDR (pin para configurar la dirección I2C; se establece en 0x23 cuando no está conectado o está conectado a GND, y en 0x5C cuando se conecta a 5V)[4]. En este proyecto, se utilizará la dirección 0x23. En la Figura 5.2 se puede observar este sensor y sus pines.

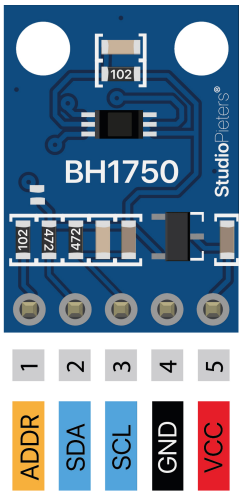


Figura 5.2: Sensor GY-302 BH1750 y sus pines [37]

- **Sensor infrarrojo de movimiento PIR 501**: Este sensor detecta la presencia de movimiento basado en la detección de radiación infrarroja emitida por objetos en movimiento, principalmente seres humanos.

Cuenta con un rango de detección de unos 110º en el eje horizontal, así como una distancia ajustable de entre 3 y 7 metros a través de un potenciómetro en la placa. A diferencia de los otros sensores usados, es altamente recomendado dejar que el PIR 501 tenga un periodo de calibración inicial de 1 minuto, lo cual ayudará a evitar falsos positivos en las lecturas de movimiento.

Una de las características más importantes que se debe resaltar de este sensor es que cuenta con dos modos de funcionamiento diferentes que pueden ser seleccionados ajustando un jumper en la placa:

- **Trigger no repetible (Estado Low)**: cuando se detecta un movimiento, la salida del sensor cambia a HIGH. Una vez transcurre el tiempo establecido por el timer del sensor, este valor cambia automáticamente a LOW. Si durante el periodo de espera se detecta algún otro movimiento, este no se tendrá en cuenta.
- **Trigger repetible (Estado High)**: Similar al anterior, pero en esta ocasión si durante el periodo de espera se detecta un nuevo movimiento, se reiniciará el periodo de espera del timer.

El temporizador incorporado en el sensor se configura mediante un potenciómetro incluido en la placa, permitiendo ajustar el intervalo entre 3 segundos y 5 minutos. Sin embargo, tras realizar múltiples pruebas, se ha observado que el tiempo real de cada temporizador puede variar significativamente debido a diferencias en las condiciones de fabricación. Por lo tanto, se considera que el mínimo periodo de espera realista es de 5 segundos. Además, en ambos modos de funcionamiento, cada vez que el estado cambia de HIGH a LOW, se producirá un intervalo de inactividad de 3 segundos durante el cual el sensor no detectará ningún movimiento.

La salida obtenida de este sensor es un voltaje de 3.3V (estado HIGH) o 0V (estado LOW). Esta salida se mostrará a través de uno de los 3 pines con los que cuenta, siendo los otros dos usados para VCC (de 5V) y GND. En la Figura 5.3 se puede observar los pines y aspectos configurables de este sensor. Además, al igual que los otros sensores el PIR 501 tiene un consumo de corriente realmente bajo. [23]

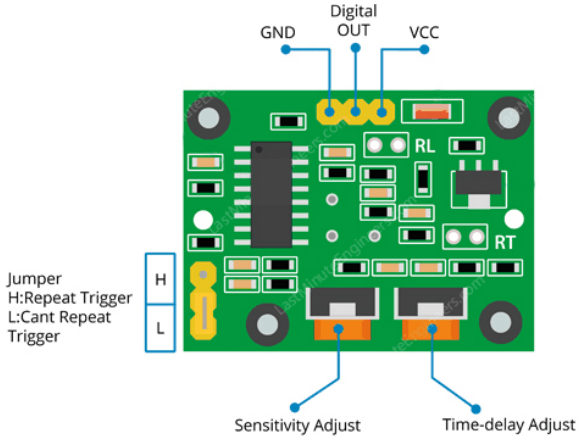


Figura 5.3: Sensor PIR 501 y sus aspectos configurables y pines [12]

Otro aspecto resaltable de este sensor es que tiene incluido un lente de Fresnel, un encapsulado semiesférico hecho de polietileno de alta densidad cuyo objetivo es permitir el paso de la radiación infrarroja en el rango de los 8 y 14 micrones. El lente detecta radiación en un ángulo con apertura de 110° y, adicionalmente, concentra la energía en la superficie de detección del sensor PIR, permitiendo una mayor sensibilidad del dispositivo. [30]

- **Microcontrolador ESP32-C3**: El ESP32-C3 es un microcontrolador basado en la arquitectura RISC-V de 32 bits, desarrollado por Espressif Systems. Está diseñado para ofrecer conectividad inalámbrica 2.4GHz a través de Wi-Fi 802.11b/-g/n y Bluetooth 5.0 LE.

  En cuanto a sus interfaces de entrada/salida, el ESP32-C3 cuenta con 22 pines GPIO que pueden configurarse para diferentes funciones, incluyendo PWM, ADC de 12 bits con hasta 6 canales, UART, e I2C entre otros. Además, incorpora características de seguridad avanzadas como aceleradores de hardware para AES, SHA, RSA, y ECC, junto con un arranque seguro basado en RSA-3072 y cifrado de memoria flash, garantizando la protección de los datos y el código almacenado. Pese a que no todas estas características son necesarias, son un buen añadido que mejora la escalabilidad del proyecto en caso de que en un futuro se desee añadir alguna funcionalidad adicional que requiera de ellas.

  El microcontrolador está diseñado para ser eficiente en términos de consumo de energía, contando además con modos de bajo consumo como la suspensión profunda (menos de 5 µA) y el modo de hibernación (menos de 130 µA) [15]. Esto, junto con su capacidad de operar en un rango de voltaje de 3.0V a 3.6V y su tamaño compacto lo convierte en una solución ideal para controlar los sensores en este proyecto. Más específicamente, el microcontrolador podría ser alimentado usando un cargador de un dispositivo móvil, una batería portátil o incluso usando paneles solares, lo cual reafirma el compromiso de este proyecto por impulsar fuentes de energía no contaminantes.

  Por otro lado, ESP32-C3 también es altamente compatible con plataformas de desarrollo populares como ESP-IDF y Arduino, facilitando su programación e integración en el proyecto. En resumen, este microcontrolador ofrece una gran combinación de funcionalidades y flexibilidad, siendo una opción excelente para el papel que debe desempeñar en el proyecto.

  Para el desarrollo de este proyecto se ha utilizado una de las muchas placas de desarrollo disponibles en el mercado que incorporan este microcontrolador, tal y como se puede observar en la Figura 5.4.

- **Middleware VisionFive 2**: VisionFive 2 es un ordenador monoplaca basado en la arquitectura RISC-V que cuenta con una GPU integrada, diseñada por StarFive Technology. Está orientada hacia aplicaciones que requieren un alto rendimiento y capacidad de expansión, siendo una plataforma robusta para ser usada en sistemas IoT. La placa está equipada con un procesador que incluye cuatro núcleos capaces de operar a una frecuencia de hasta 1.5 GHz.

  En cuanto a la memoria, la VisionFive 2 puede contar con hasta 8 GB de RAM LPDDR4; y para almacenamiento, la placa admite eMMC, MicroSD, y SSD, lo

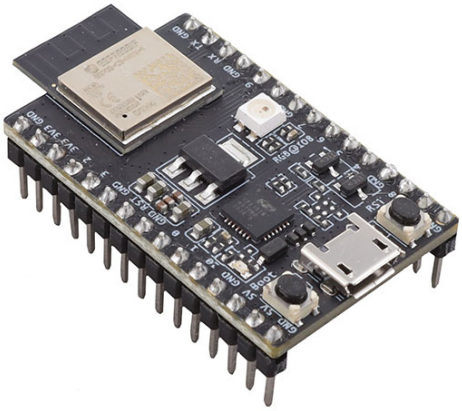


Figura 5.4: Placa de desarrollo ESP32-C3 utilizada [11]

que permite contar con una gran capacidad de almacenamiento. En este proyecto, el Sistema Operativo de la placa ha sido cargado en una tarjeta MicroSD y se ha instalado una memoria SSD para el almacenamiento.

La conectividad es otro punto fuerte de la VisionFive 2, con soporte para Ethernet gigabit; así como con una amplia gama de interfaces de E/S, incluyendo HDMI, USB 3.0, GPIOs y UART [42]. Sin embargo, no tiene soporte para Wi-Fi, por lo que será necesario conectar a uno de sus puertos USB un adaptador Wi-Fi.

En términos de software, la VisionFive 2 es compatible con sistemas operativos basados en Linux, como Debian y Ubuntu, aunque en este último caso el soporte no es nativo.

Todo esto convierte a la placa VisionFive 2 en una opción sumamente completa para desempeñar el papel de middleware en el entorno IoT que se va a desarrollar. Entre las funciones destacadas que desempeñará se encuentran el alojamiento del software encargado de gestionar el entorno IoT, la interfaz para la visualización de los datos recopilados y su papel como broker MQTT, facilitando el envío de mensajes entre el middleware y los microcontroladores.

- **Extensor Wi-Fi TP-LINK AC750**: El AC750 Mesh Wi-Fi Range Extender RE200, fabricado por TP-Link, es un dispositivo diseñado para mejorar y extender la cobertura de una red Wi-Fi existente. Este extensor de red es compatible con el estándar 802.11ac, lo que le permite operar en bandas de 2.4 GHz y 5 GHz. El uso de la función de 2.4 GHz en el proyecto será obligatorio, puesto a que los microcontroladores ESP32-C3 no admiten un ancho de banda de 5 GHz.

  Además, el RE200 es fácil de configurar gracias a su botón WPS (Wi-Fi Protected Setup), que permite una conexión rápida y sencilla con el enrutador [39]. También es compatible con la aplicación TP-Link Tether, que facilita la gestión del dispositivo desde un teléfono móvil, permitiendo configurar y monitorizar el estado del extensor de red de manera sencilla. La gran cantidad de posibilidades de configuración con las que cuenta será utilizado en el proyecto para añadir una capa de seguridad adicional, como puede ser una función de lista blanca a través de la que solo los dispositivos especificados podrán hacer uso del extensor.

A pesar de no tratarse de un Punto de Acceso en si mismo, este extensor podrá realizar la misma función casi sin ningún problema. Uno de los pocos inconvenientes que surgen debido al uso de un extensor de red es que a la hora de configurar el SSID y contraseña de la red, este último siempre tomará el mismo valor que la contraseña que tenga el router al que se haya conectado el extensor. A excepción de esta pequeña carencia, este extensor de red podrá generar sin problemas una red inalámbrica estable a la que los microcontroladores y middleware se conectarán para comunicarse usando MQTT.

Es importante considerar la necesidad de una alternativa en caso de que el firewall de la red interna de la institución bloquee las conexiones de los dispositivos al extensor Wi-Fi. En tal situación, se utilizará un punto de acceso móvil en un dispositivo Android como solución alternativa, ya que este es capaz de soportar hasta 8 conexiones de dispositivos simultáneamente.

5.2. Elección del software

Con los componentes hardware que se usarán en el proyecto seleccionados, es crucial elegir las herramientas de software más adecuadas que permitirán implementar y gestionar de manera eficiente el proyecto IoT. En esta sección, se detallarán las decisiones tomadas respecto a las plataformas, entornos de desarrollo, bibliotecas, y protocolos que se utilizarán para construir el sistema. Se evaluarán las opciones disponibles y se explicarán las razones detrás de cada elección, asegurando que el software seleccionado no solo cumpla con los requisitos técnicos, sino que también facilite un desarrollo fluido del proyecto.

- **Sistema Operativo**: En la selección del sistema operativo para la placa VisionFive 2 se consideraron dos candidatos principales: Debian y Ubuntu Server.

  La ventaja principal de Debian radica en que la documentación oficial de StarFive describe detalladamente el proceso de instalación en la placa, lo que garantiza un soporte nativo. Pese a esto, se optó por utilizar una imagen preparada de Ubuntu Server. Esta decisión se basa en que Ubuntu Server es una versión ligera del popular sistema operativo Ubuntu, diseñada específicamente para funcionar sin entorno gráfico, operando exclusivamente desde la terminal. Esto reduce el consumo de recursos del sistema y mejora la eficiencia energética, lo cual es ideal dado que la placa tan solo ejecutará una serie de servicios sin necesidad de una interfaz gráfica. Además, al utilizar únicamente la terminal, toda la configuración puede realizarse cómodamente a través de un cliente SSH o serial, como PuTTy, desde un ordenador de sobremesa o portátil.

  Otro factor decisivo en la elección de este sistema operativo fue la familiaridad y preferencia personal del desarrollador del proyecto. Aunque ambos sistemas operativos son muy similares, en la mayoría de los proyectos previos que involucraron Linux utilizó Ubuntu. Esto proporciona un conocimiento previo que acelerará el desarrollo y facilitará la resolución de cualquier problema que pueda surgir.

  Además, existe una amplia variedad de guías detalladas que explican el proceso

de instalación de Ubuntu Server en la placa VisionFive 2. También se dispone de una imagen preinstalada de Ubuntu Server, diseñada específicamente para VisionFive 2, que se puede descargar desde la página oficial de Ubuntu. Por lo tanto, aunque este sistema operativo no cuenta con soporte nativo por parte de los fabricantes, la comunidad de Ubuntu ha simplificado considerablemente el proceso de instalación.

- **Node-RED**: El software seleccionado para implementar la lógica del entorno IoT es Node-RED, una herramienta de desarrollo basada en flujos que facilita la conexión de dispositivos, APIs y servicios online de manera sencilla e intuitiva, lo que la convierte en una opción excelente para proyectos IoT.

  Su tiempo de ejecución ligero se basa en Node.js, aprovechando al máximo su modelo no bloqueante basado en eventos. Esto lo hace ideal para ejecutarse en hardware de bajo coste, como las placas Raspberry Pi o VisionFive 2.

  Uno de sus principales atractivos es su interfaz gráfica que permite a los usuarios arrastrar y soltar bloques (nodos) que representan diferentes funcionalidades, y luego conectarlos mediante 'hilos' para crear flujos de datos complejos con poco o ningún código, permitiendo crear aplicaciones de forma sencilla. Otra ventaja significativa es la gran cantidad de opciones de integración con servicios y protocolos como MQTT, HTTP y WebSockets, lo que permite adaptarse a las necesidades específicas del proyecto. [26]

  Finalmente, y probablemente la característica más relevante que llevó a elegir Node-RED sobre otras alternativas como Microsoft Azure IoT Hub o Google Cloud IoT Core, es la capacidad de extender sus funcionalidades base mediante paquetes descargables. Entre ellos, destaca el paquete **Dashboard**, que permitirá desarrollar de manera sencilla una página web para mostrar en ella los datos que recibe Node-RED, además de permitir al profesor gestionar diversos parámetros de los dispositivos IoT, siendo todo esto gestionado y automatizado por Node-RED. Además, Node-RED Dashboard es compatible con diversos tipos de dispositivos, como navegadores de ordenadores y dispositivos móviles, favoreciendo de esta manera la portabilidad del sistema.

- **Broker MQTT Eclipse Mosquitto**: Para realizar el envío y recepción de mensajes entre los microcontroladores y middleware, se usará MQTT, un protocolo de mensajería ligero utilizado ampliamente en aplicaciones de IoT debido a su eficiencia, bajo consumo energético y capacidad para funcionar en redes con bajo ancho de banda, alta latencia o poca fiabilidad. Esto último es posible ya que MQTT ofrece diferentes niveles de QoS, o calidad de servicio, permitiendo elegir el nivel de garantía de entrega de mensajes. A continuación se detallan las características de los tres niveles disponibles: [16]

  - **QoS 0 (at most once)**: el mensaje se envía una vez y no se garantiza que llegue al destino. No hay confirmación de recepción, por lo que si ocurre un problema durante la transmisión, el mensaje puede perderse. Es el más rápido y se usa cuando la pérdida de la información no es un problema crítico.
  - **QoS 1 (at least once)**: se garantiza que el mensaje se entregue al menos

una vez, pero podría entregarse varias veces. El emisor espera una confirmación del receptor y si no recibe la confirmación, reenvía el mensaje hasta que la recibe. Esto asegura que el mensaje llega, aunque podría resultar en duplicados.

- **QoS 2 (exactly once)**: garantiza que el mensaje se entregue exactamente una vez sin duplicados. Para lograr esto, se usa un proceso de confirmación en dos pasos entre el emisor y el receptor. Es ideal para situaciones donde la duplicación o la pérdida de mensajes no son aceptables, aunque es el más lento de todos.

De entre todos los niveles, en el proyecto se deberá de contar con garantía de recepción de mensaje (puesto a que una pérdida de un mensaje supondría en un largo periodo en el que no se reciben información de los sensores). Por esta razón, el nivel QoS 0 no podrá ser utilizado. Por otro lado, la recepción de mensajes duplicados no es ningún problema, por lo que la mejor opción es el nivel QoS 1 al ser más eficiente que el nivel 2.

Eclipse Mosquitto es un broker MQTT de código abierto que implementa el protocolo MQTT. Actúa como intermediario, gestionando la comunicación entre los clientes de MQTT. Los clientes pueden publicar mensajes en temas específicos, y otros clientes pueden suscribirse a esos temas para recibir actualizaciones en tiempo real.

Algunas de las razones por las que Mosquitto es idóneo para este proyecto es su compatibilidad con varios lenguajes de programación y plataformas IoT, entre ellos Node-RED, además de ser ligero, eficiente y escalable, permitiendo atender a un gran número de clientes MQTT sin que esto impacte en gran medida al rendimiento del sistema en el que se ejecuta. También permite establecer una autenticación de clientes MQTT para impedir la conexión de dispositivos no autorizados.

Mosquitto MQTT se ejecutará sobre la placa VisionFive 2 y actuará como intermediario entre los microcontroladores y el software middleware. Otra alternativa barajada para gestionar este intercambio de información fue usar un broker MQTT online, como por ejemplo broker.emqx.io, pero tras realizar unas pruebas para evaluar su viabilidad se llegó a la conclusión de que era demasiado inestable como para considerarlo una buena opción.

- **Arduino IDE**: es una de las opciones más populares y ampliamente utilizadas para la programación de microcontroladores. Se trata de un conjunto de herramientas software que permiten a los programadores desarrollar y cargar todo el código necesario para hacer que los microntroladores funcionen tal y como se desee. Pese a que originalmente fue diseñado para las placas Arduino, el IDE es compatible con una amplia gama de microcontroladores, incluyendo el ESP32-C3 utilizado en este proyecto, permitiendo por lo tanto aprovechar todas sus herramientas y bibliotecas.

  Utiliza un lenguaje de programación simplificado basado en C/C++; contando con una interfaz sencilla y una gran comunidad de desarrolladores que han contribuido en forma de bibliotecas y extensiones que se pueden importar al proyecto

para añadir determinadas funcionalidades sin necesidad de programarlas desde cero [3]. En este proyecto, se han descargado y utilizado las siguientes bibliotecas y contribuciones de la comunidad: [5]

- **Soporte para placas ESP32-C3**: Para poder hacer uso de todas las herramientas de Arduino-IDE en la programación de los microcontroladores ESP32-C3 será necesario descargar el soporte oficial de las placas ESP32. Esto se puede hacer especificando en las preferencias de Arduino-IDE la URL desde la que se puede descargar el paquete que da soporte a estas placas, de manera que una vez instalado se puede seleccionar una placa ESP32-C3, permitiéndose así poder programar los microcontroladores de este proyecto en Arduino-IDE.

  Este soporte incluye, entre otras cosas, funcionalidades para controlar las interrupciones síncronas a través de la configuración del timer interno de la placa (a través de la **biblioteca esp32-hal-timer**); e interrupciones asíncronas cuando se detecte un cambio de nivel lógico en un pin concreto configurable por el programador.

- **WiFi (version 1.2.7), de Arduino**: Esta biblioteca permite a los microcontroladores con capacidad Wi-Fi, como el ESP32-C3, conectarse a redes Wi-Fi. Proporciona funciones para gestionar la conexión a redes inalámbricas, configurar la red, obtener la dirección IP, y realizar comunicaciones a través de protocolos como TCP/IP y UDP. Esta biblioteca es necesaria para establecer la comunicación de bajo nivel que será usada para el envío/recepción de mensajes MQTT entre microcontroladores y Middleware.

- **PubSubClient (version 2.8), de Nick O'Leary**: Esta biblioteca implementa el protocolo MQTT en el microcontrolador. La biblioteca permite que el microcontrolador actúe como cliente MQTT, conectándose a un broker MQTT, suscribiéndose a temas y publicando mensajes. Esta implementación soporta QoS de nivel 0 o 1, de manera que se utilizará este último para garantizar la entrega/recepción de los mensajes entre microcontroladores y middleware.

- **DHT sensor library (version 1.4.6), de Adafruit**: Esta biblioteca permite leer datos de sensores de temperatura y humedad de la serie DHT, como el DHT22 que se usa en este proyecto. Proporciona funciones para inicializar el sensor, leer la temperatura y humedad, y manejar la comunicación con el sensor; simplificando considerablemente el proceso de lectura de este tipo de sensor. Para su correcto funcionamiento, requiere de la biblioteca **Adafruit Unified Sensor**.

- **BH1750 (version 1.3.0), de Christopher Laws**: Esta biblioteca facilita la lectura de datos del sensor de luz BH1750. La biblioteca permite configurar el sensor y leer la luminosidad en el entorno, proporcionando valores precisos que pueden ser utilizados para ajustar la iluminación o realizar análisis ambientales. Debido a que este sensor utiliza una interfaz I2C, es necesario incluir en el proyecto la librería Wire, a través de la cual se podrá inicializar una comunicación I2C en los pines GPIO del ESP32-C3 donde se

conectarán los pines SDA y SCL del BH1750.

Otras alternativas viables que podrían haber sido usadas en lugar de Arduino IDE son PlatformIO, un IDE que soporta múltiples plataformas y frameworks, tratándose de una herramienta avanzada que se puede integrar con Visual Studio Code ofreciendo un entorno más potente con funcionalidades como depuración avanzada y la gestión de múltiples proyectos pero mucho más complejo que Arduino IDE; y Espressif IDE, el entorno de desarrollo oficial para la programación de microcontroladores ESP32, proporcionando acceso a las APIs y herramientas más avanzadas para aprovechar todas las características del hardware, aunque es significativamente más complejo que la opción elegida y no tiene una comunidad tan grande.

PlatformIO fue descartado ya que no se le podía sacar provecho en el desarrollo del proyecto a ninguna de las funcionalidades avanzadas que ofrecía, además de tener una curva de aprendizaje mucho más pronunciada que habría retrasado el tiempo de desarrollo. Por otro lado, se decidió no usar Espressif IDE ya que todas las características que se querían usar en el microcontrolador (como Wi-Fi, timers e interrupciones) estaban disponibles en Arduino-IDE usando las bibliotecas y extensiones anteriormente explicadas.

5.3. Diseño de la conexión de componentes

Una vez identificados los componentes hardware y software que conformarán la arquitectura del sistema, el siguiente paso será diseñar la manera en la que se interconectarán todos ellos; ya que se requiere de un trabajo colaborativo para que todos los elementos puedan desempeñar su función y el sistema IoT en su conjunto logre alcanzar sus objetivos. Para esto, se realizará una división de las conexiones necesarias organizándolas en dos tipos: conexiones entre sensores y microcontroladores, y entre microcontroladores y middleware.

5.3.1. Conexiones entre sensores y microcontroladores

Como se ha explicado con anterioridad, los sensores son los elementos encargados de realizar las lecturas sobre las condiciones del aula, ya sean movimiento, temperatura, humedad o luz. Por otro lado, los microcontroladores son los responsables de controlar y leer los datos obtenidos por los sensores, transformarlos en caso de que sea necesario y enviárselos al middleware.

Para poder controlar y leer los sensores, será necesario realizar conexiones físicas entre los pines GPIO del microcontrolador y los pines de los sensores a través de cables. Cada tipo sensor se deberá de conectar de una manera diferente al microcontrolador, aunque en todos ellos es necesario conectar sus pines VCC y GND a los del microcontrolador para recibir alimentación eléctrica, y sus pines de datos a aquellos designados en el ESP32-C3 para realizar las lescturas.

Por otro lado, la placa de desarrollo ESP32-C3 que se va a utilizar cuenta con un gran número de pines que podrán ser utilizados para la interconexión, tal y como se puede apreciar en la Figura 5.5 .De entre todos estos podemos distinguir pines GND y alimentación de 3.3V y de 5V a los que se conectarán los sensores, además de pines GPIO a través de los cuales se podrán recibir los datos.

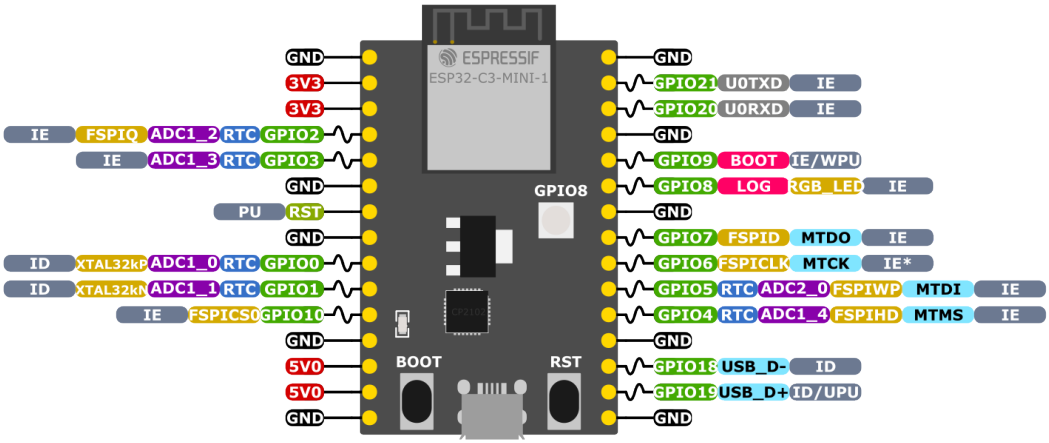


Figura 5.5: Pinout de placa de desarrollo ESP32-C3 [14]

Con esto en mente, ya se puede pasar a estudiar individualmente como se conectará cada uno de los sensores al microcontrolador:

- **Sensor de humedad y temperatura DHT22**: Este sensor requiere de un voltaje de 3.3V para funcionar, por lo que su pin VCC se conectará a uno de los pines de alimentación del microcontrolador que corresponda con este voltaje; mientras que su pin GND se conectará a uno de los pines de tierra del ESP32-C3. El pin DATA por el que se enviarán los datos se puede conectar a cualquiera de los pines GPIO del microcontrolador, en el caso de este proyecto se ha decidido usar el pin GPIO2.

- **Sensor infrarrojo de movimiento PIR 501**: En el caso de este sensor, se requiere de un voltaje de 5V para asegurar su correcto funcionamiento (aunque pruebas realizadas revelan que usar un voltaje de 3.3V no causa ningún problema), por lo que de igual manera que en el sensor anterior su pin VCC se conectará a uno de los pines de alimentación del microcontrolador que corresponda a un voltaje de 5V y su pin GND se conectará a uno de los pines de tierra del ESP32-C3. El pin DATA por el que se envia un HIGH o LOW lógico en base a si se ha detectado movimiento se puede conectar de igual manera a cualquier pin GPIO, usándose en este proyecto el GPIO3.

  Además, el sensor se configurará a través de sus potenciómetros a una distancia de 4 metros y un tiempo de retardo lo menor posible (entre 3 y 5 segundos), estableciendo además el modo de funcionamiento a 'Trigger no repetible'.

- **Sensor de luz GY-302 BH1750**: Este sensor, al igual que el anterior, requiere de 5V de alimentación, realizándose las mismas conexiones para VCC y GND. Sin embargo, y a diferencia de los demás, en este sensor será necesario establecer una conexión I2C para los datos. El ESP32-C3 cuenta con un sólo canal I2C

y cualquier pin GPIO puede establecerse como SDA o SCL. En este proyecto, se configurará el GPIO18 como SDA y el GPIO19 como SCL. No es necesario conectar el pin ADDR (que permite configurar la dirección I2C del dispositivo esclavo), ya que un estado no conectado equivale a la dirección 0x23, que es la que se usará en el proyecto.

Con el objetivo de ordenar las conexiones, hacer que estas sean más fiables y menos propensas a fallar por daños en los cables y para facilitar la posterior encapsulación del circuito diseñado en una pequeña caja que lo proteja, se usará una *breadboard* para conectar el microcontrolador, los cables y los sensores. En la Figura 5.6 se pueden observar todas las interconexiones físicas descritas en este apartado.

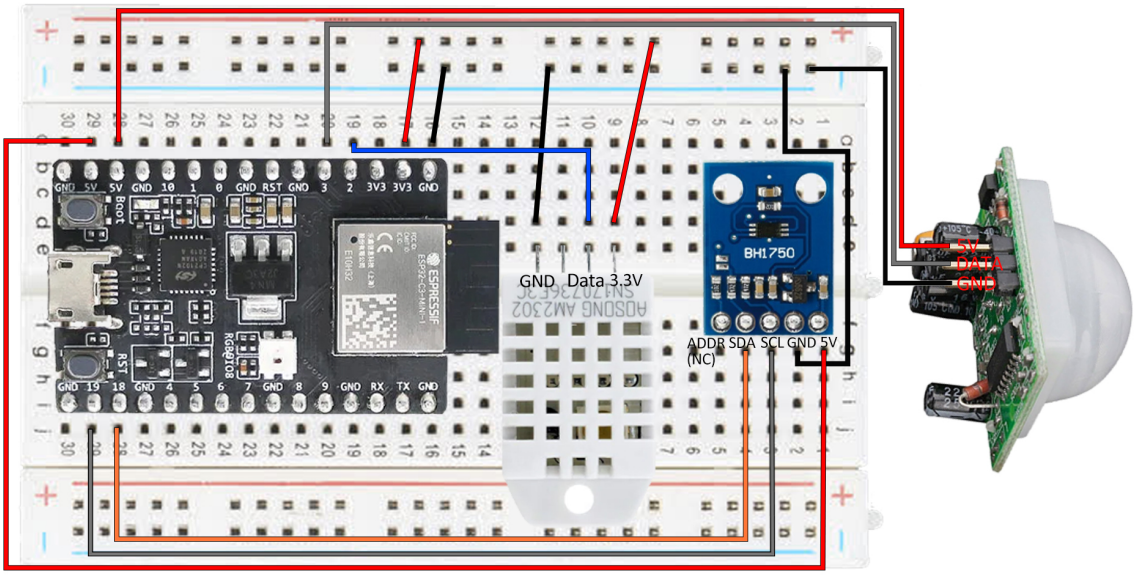


Figura 5.6: Diseño de las conexiones físicas Microcontrolador-Sensores

Para el desarrollo de este proyecto se utilizará un total de 2 microcontroladores, y cada microcontrolador estará conectado con un sensor de cada tipo. Cada uno de estos dos conjuntos de componentes se introducirá en una pequeña caja a la que se le harán orificios por los que sacar los sensores y para poder conectar el microcontrolador a la corriente eléctrica. El motivo de encapsular los componentes en una caja es proteger el circuito creado de daños accidentales, así como añadir una capa de transparencia para camuflar los elementos del entorno IoT.

Además, esta caja contará con superficies de velcro en sus laterales, que serán usados para poder pegar y ajustar en ellos unas palas de cartón-piedra que se usarán para reducir el ángulo de visión del sensor de movimiento, evitando de esta manera que lea movimientos en zonas no deseadas del aula. Originalmente se consideró usar imanes para conectar estas palas, pero esta idea fue rápidamente descartada ya que los campos magnéticos podían afectar negativamente a los microcontroladores y sensores, haciendo que todo el sistema fallase en el proceso.

5.3.2. Conexión de microcontroladores y middleware

Una vez esté unificado cada microcontrolador con sus sensores, funcionando como un único circuito, es hora de realizar la conexión entre los microcontroladores, que leerán y ajustarán los datos procedentes de los sensores, y el middleware IoT, que recibirá todos estos datos periódicamente para mostrárselos al profesor en tiempo real.

Para la transferencia de estos datos no es una opción viable usar un medio físico, puesto a que los sensores se pueden encontrar localizados en cualquier lugar del aula, de manera que usar cables limitaría ampliamente la instalación del sistema. Por esta razón, se usará como canal de comunicación una red Wi-Fi creada por el extensor de red. El uso de esta red Wi-Fi, en lugar de la red del centro educativo, ofrece varias ventajas como permitir configurar un servidor DHCP para asignar direcciones IP dinámicas a los microcontroladores, establecer un control de acceso mediante autenticación basada en SSID y contraseña, y aislar el sistema IoT del tráfico existente en la red del centro. Esto proporciona una capa adicional de protección contra ciertos tipos de ataques, como el Sniffing y el Spoofing.

Una vez identificado el medio de comunicación, es necesario establecer el protocolo que se utilizará para la transferencia de datos. Como se explicó anteriormente, se empleará MQTT, ya que no solo facilita una comunicación sencilla y controlada a través de mensajes organizados por temas específicos a los que los microcontroladores y el middleware pueden suscribirse, sino que también garantiza la entrega de los datos. Si se transfirieran los datos directamente por Wi-Fi utilizando protocolos como UDP o TCP, surgirían numerosos problemas y dificultades que no están presentes con MQTT.

El encargado de mediar estos mensajes será el servicio MQTT Mosquitto que se levantará automáticamente en la placa VisionFive 2 tras arrancar el sistema. Tanto el servicio Node-RED de la placa middleware como cada uno de los microcontroladores actuarán como clientes MQTT, y como tales, deberán de autenticarse usando las credenciales MQTT configuradas en el broker, siendo un nombre de cliente y constraseña. A continuación se explica los temas que se usarán en el proyecto y como se relacionan con cada uno de los componentes:

- **Topics de datos**: el middleware Node-RED estará suscrito a todos los temas de este tipo, a través de los cuales recibirá los datos que los microcontroladores han leido de sus sensores.

  Este tipo de tema sigue la siguiente estructura: `datos/x/y`, donde 'x' identifica el número del microcontrolador del que procede el mensaje e 'y' el tipo de lectura realizada, siendo los posibles valores de este último 'humedad', 'temperatura', 'luz' o 'movimiento'. La razón por la que se realiza la distinción sobre la procedencia de los datos es para permitir que el origen de cada lectura sea mostrada al profesor a través de la interfaz.

  Cada ciertos intervalos de tiempo (la duración de cada intervalo dependerá de la configuración actual del microcontrolador) los microcontroladores ESP32-C3 enviarán los datos al middleware usando el tema correspondiente.

- **Topics de configuración**: los mensajes con este tipo de temas serán enviados por el middleware cuando el profesor realice un cambio en la configuración de los

microcontroladores desde la pestaña de configuración en la interfaz web. Estos topics tienen la estructura `config/z`, donde 'z' se corresponde con el tipo de cambio a realizar.

En el proyecto final tan solo existe un tema de este tipo: `config/timer`, que sirve para modificar el intervalo de tiempo que los microcontroladores deben esperar antes de enviar el siguiente grupo de mensajes de datos. Pese a esto, el número de temas de configuración puede ser fácilmente ampliado en un futuro en caso de que se detecten nuevas necesidades.

En la Figura 5.7 quedan ilustrados los diferentes temas utilizados por los microcontroladores y middleware y como se relacionan a cada uno de estos.

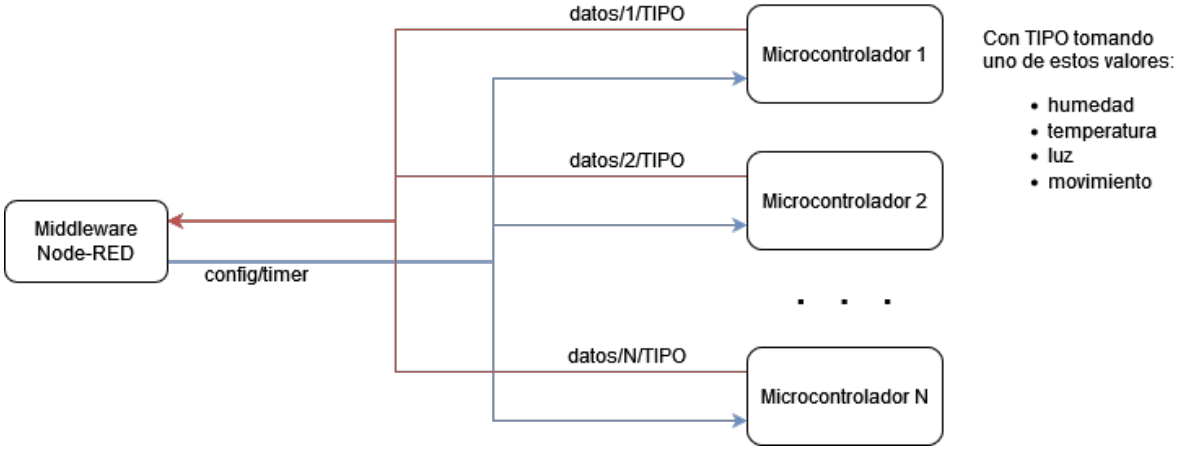
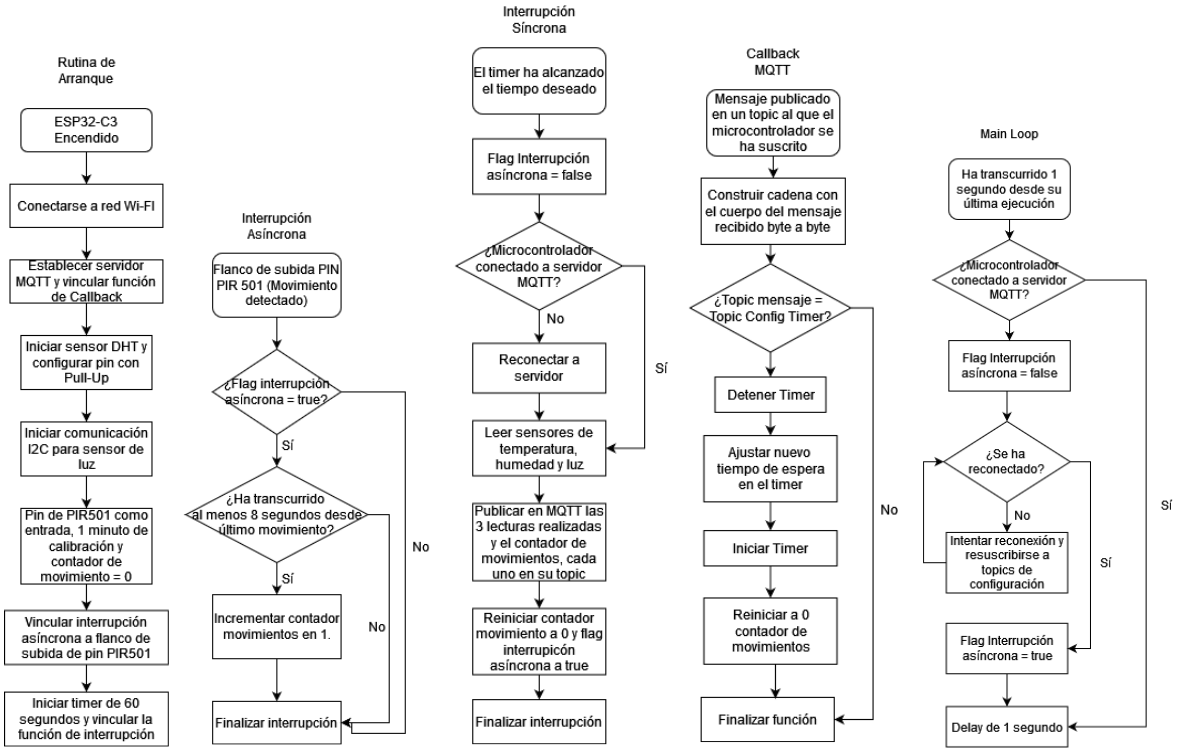


Figura 5.7: Temas MQTT y su relación con el middleware y los microcontroladores

5.4. Diseño de la programación de microcontroladores

El funcionamiento general que deberá seguir el programa de los microcontroladores ESP32-C3 es relativamente sencillo y puede ser dividido en las partes descritas a continuación. Sin embargo, antes de continuar es importante considerar que en todas ellas se deberá de incluir información de depuración en la fase de desarrollo para facilitar las pruebas que se realizarán posteriormente. En la Figura 5.8 se puede observar el comportamiento de cada una de las partes del programa.

- **Arranque**: Cuando el microcontrolador es encendido, el programa comenzará estableciendo la conexión con la red Wi-Fi y el servidor MQTT, de manera que se asegura que el dispositivo pueda comunicarse continuamente y de manera eficiente con el middleware. Una vez conectado, se configuran los sensores y pines para la captura de datos ambientales y de movimiento, permitiendo la posterior lectura periódica de temperatura, humedad, nivel de luz y detección de movimiento en el aula. Por necesidades del propio sensor PIR 501, se esperará un minuto a que el sensor de movimiento quede calibrado antes de continuar.



(a) Setup (b) Int. Asíncrona (c) Int. Síncrona (d) Callback (e) Loop Principal

Figura 5.8: Diseño de las partes del programa de los ESP32-C3

Tras esto, se iniciará un temporizador de 60 segundos que se encargará de ejecutar la interrupción síncrona a través de la que se enviarán peródicamente los datos al middleware, se configurará la interrupción asíncrona necesaria para el sensor de movimiento, y se vinculará la conexión MQTT a una función de callback que se llamará automáticamente cuando el microcontrolador reciba un mensaje MQTT.

En la Figura 5.8a se puede observar el diagrama de flujo con el proceso de la configuración inicial del microcontrolador.

- **Interrupción asíncrona**: El sensor de movimiento PIR 501 no cuenta con ninguna funcionalidad que permita almacenar el número de movimientos detectados en un cierto intervalo de tiempo, por lo que esta cuantificación se deberá programar en el microcontrolador.

  Para ello, se usará un contador auxiliar de movimientos, inicializado a 0 al inicio del programa. Este contador será actualizado en una función que será configurada como una interrupción asíncrona, siendo invocada cada vez que se detecte un flanco de subida en el pin donde se conecta la salida de datos del sensor de movimiento (GPIO3). Además de este comportamiento básico, existen una serie de restricciones que deberemos imponer a la interrupción asíncrona para que las lecturas de los movimientos se realicen en los momentos adecuados.

  Como se mencionó anteriormente, el sensor de movimiento ofrece varios modos de operación, de los cuales se utilizará el modo de "Trigger no repetible". Este modo asegura que, una vez detectado un movimiento, el sensor se mantenga en estado

HIGH hasta que finalice el periodo de espera, sin volver a LOW antes de tiempo. Este funcionamiento se complementará con un retardo de 5 segundos, lo que permite que entre dos flancos ascendentes (de LOW a HIGH) transcurra un total de 8 segundos (5 segundos de retardo más 3 segundos de espera obligatoria entre lecturas consecutivas). Esta pausa garantiza que no se produzcan falsas lecturas debido a la recalibración automática del sensor PIR 501 después de detectar un movimiento, así como por posibles interferencias electromagnéticas captadas por el cable. Por esta razón, se debe implementar un mecanismo que evite incrementar el contador de movimientos a menos que hayan pasado al menos 8 segundos desde la última actualización.

Además, se establecerá una bandera de interrupción asíncrona que se habilitará o deshabilitará dentro del cuerpo de la interrupción síncrona. Esta bandera tiene el propósito de evitar cambios en el sensor de movimiento mientras se está ejecutando la interrupción síncrona. Esto es especialmente relevante en el caso poco frecuente en que, tras la invocación de la interrupción síncrona por parte del temporizador, se detecte un movimiento que dispare la interrupción asíncrona antes de que la interrupción síncrona haya finalizado. En tal situación, el microcontrolador detendría la ejecución de la interrupción síncrona para atender la interrupción asíncrona, y luego reanudaría la interrupción síncrona. De esta forma, se asegura que el valor del contador de movimientos no se modifique hasta que la interrupción síncrona haya concluido completamente.

En la Figura 5.8b se puede observar el diagrama de flujo con el comportamiento que tendrá la interrupción asíncrona.

- **Interrupción síncrona**: El microcontrolador debe suministrar los datos leídos de los sensores al middleware cada ciertos periodos de tiempo. Estos intervalos de tiempo son gestionados por el timer configurable del ESP32-C3. Cuando el timer alcanza el tiempo deseado se ejecutará una interrupción síncrona, que deshabilitará la bandera de la interrupción asíncrona, impidiendo así que se puedan detectar nuevos movimientos, y comprobará que el microcontrolador aún esté conectado al servidor MQTT. En caso de ser así, realizará una lectura del sensor de temperatura, humedad y luz, además de una lectura del contador auxiliar que cuenta el número de movimientos y que es incrementado por la interrupción asíncrona.

  Tras ésto, enviará al middleware los datos a través de MQTT usando los temas que corresponda. Una vez finalizado, se reiniciará el contador auxiliar de movimientos a 0 y se reestablecerá la bandera de interrupciones asíncronas para que se puedan volver a contar los movimientos detectados.

  En la Figura 5.8c se puede observar el diagrama de flujo con el comportamiento que tendrá la interrupción síncrona.

- **Función de callback MQTT**: cuando es publicado por el middleware un mensaje MQTT con un tema al que el microcontrolador se encuentra suscrito, el sistema llama automáticamente a la función designada como de *callback*. En ella, se deberá leer el tema del mensaje y su cuerpo, este último debiendo hacerse carácter a carácter. Tras esto, se podrá utilizar el tema del mensaje para decidir

que se debe hacer con el mensaje recibido.

En este proyecto, los microcontroladores solo estarán suscritos al tema de configuración del timer, por lo que al recibir un mensaje con esta temática el temporizador interno se detendrá, se establecerá como perido de espera el nuevo valor recibido en el cuerpo del mensaje y se reiniciará. Además, se deberá reestablecer a 0 el valor del contador auxiliar de movimientos, ya que el timer ha vuelto a comenzar a contar desde el principio.

En la Figura 5.8d se puede observar el diagrama de flujo con el comportamiento que tendrá la función de callback.

- **Loop principal**: Al igual que en la mayoría de los códigos de microcontroladores, se contará con una función que se ejecutará en un bucle de manera indefinida. En esta función solo se verificará que el dispositivo siga conectado a la red Wi-Fi y al servidor MQTT, lo que resulta esencial, ya que una desconexión implicaría que el microcontrolador no recibiría los mensajes de configuración con los temas MQTT a los que está suscrito, y, por lo tanto, la función de callback no se llamaría. En caso de necesitar realizar una reconexión, se deshabilitará la bandera de interrupción asíncrona para evitar la ejecución de su código durante el proceso de reconexión. Luego, el dispositivo se conectará y suscribirá nuevamente a sus temas de configuración, y, una vez completado el proceso, se volverá a habilitar la bandera de la interrupción asíncrona.

  Además, al finalizar cada iteración del bucle principal, se incorporará un retardo de 1 segundo. Este retardo evita el desperdicio de ciclos de reloj innecesarios, lo que contribuye a lograr un ahorro energético.

  En la Figura 5.8e se puede observar el diagrama de flujo con el comportamiento que tendrá el bucle principal.

Con esto ya están diseñadas todas las partes que conforman el programa a desarrollar para los microcontroladores. Destacar que, a excepción del Loop principal, todo el código funciona a partir de interrupciones o llamadas automáticas a funciones, por lo que el consumo del microcontrolador será mínimo, solo teniendo que ejecutar instrucciones cuando se produzca un evento de movimiento, se reciba un mensaje MQTT apropiado o el reloj ejecute la interrupción síncrona. Esto hace que el programa desarrollado para este proyecto sea energéticamente eficiente. En la Figura 5.9 se puede apreciar como estas partes del programa interactúan entre sí y cuando son ejecutadas.

Una de las decisiones más interesantes que se ha tenido que tomar en el diseño del programa de los microcontroladores ha sido el mecanismo para notificarles cuándo deben enviar los datos. Se consideraron dos opciones principales:

- **Notificación a través de interrupción síncrona** ejecutada por el temporizador del ESP32-C3, que es la opción elegida y que se ha descrito anteriormente. Este enfoque presenta el inconveniente de que, inicialmente, los temporizadores de los diferentes microcontroladores no están sincronizados, ya que se inician automáticamente en el momento exacto en que cada uno se enciende.
- **Notificación por parte del middleware** a los microcontroladores. En esta alternativa, el middleware Node-RED habría notificado simultáneamente a todos

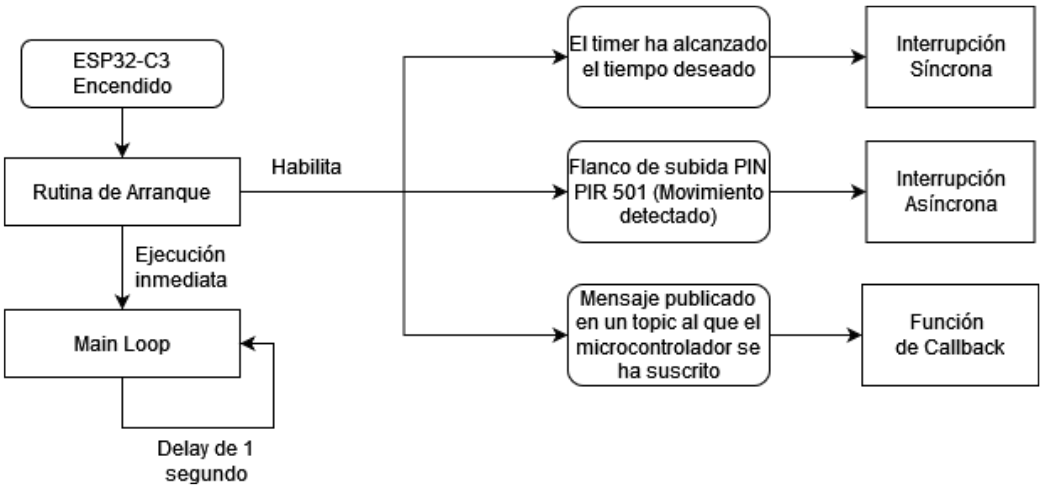


Figura 5.9: Diagrama de flujo del código completo del programa

los microcontroladores cuando finalizara el periodo de espera, indicándoles que debían enviar las lecturas de datos. Aunque este método evita el problema de la desincronización inicial, genera un mayor volumen de mensajes MQTT, ya que el middleware debe enviar un aviso antes de recibir los datos.

Finalmente, se optó por la primera opción debido a que proporciona una clara separación de responsabilidades entre el middleware y los microcontroladores. Esto significa que los ESP32-C3 pueden operar de manera autónoma y solo necesitan comunicarse con el middleware cuando es necesario enviar datos, lo que mejora la fiabilidad y escalabilidad del sistema. Además, el problema de sincronización entre microcontroladores se puede resolver fácilmente enviando una directiva de configuración del temporizador desde el middleware, lo que reiniciará todos los temporizadores y los sincronizará para comenzar a contar desde cero simultáneamente.

5.5. Diseño del middleware Node-RED e Interfaz Web

El middleware será construido con Node-RED, una plataforma flexible y escalable para gestionar los datos procedentes de los sensores conectados a los microcontroladores. Utilizando los nodos MQTT In y Out, el middleware se suscribe a los temas en los que los microcontroladores publican las lecturas de los sensores y también puede publicar mensajes de configuración dirigidos a los microcontroladores cuando sea necesario.

Los datos recibidos a través del nodo MQTT In son filtrados según su tema, lo que permite identificar tanto el tipo de dato como el microcontrolador de origen. Posteriormente, estos datos se envían a los nodos responsables de construir las gráficas en la interfaz web, donde se indica claramente el microcontrolador del que provienen los datos.

Para diseñar la interfaz de visualización de datos, se utilizarán los nodos del paquete Dashboard en el flujo de Node-RED que se desarrollará. Se crearán gráficas en

tiempo real para mostrar la evolución de las variables monitorizadas, con una gráfica específica para cada magnitud a medir. Cada gráfica representará datos del mismo tipo provenientes de diferentes microcontroladores, y Node-RED diferenciará estos datos utilizando líneas de colores distintos, gracias a la asignación de temas diferentes a cada mensaje dentro del flujo.

Estas gráficas son interactivas y se generan y actualizan en tiempo real, lo que permite a los profesores observar patrones y tendencias cruciales para la toma de decisiones durante las clases.

La interfaz web contará con dos pestañas diferentes: una para mostrar las gráficas con las lecturas de datos y otra para el menú de configuración. En el menú de configuración se ofrecerán las siguientes opciones:

- **Modificar el intervalo de tiempo entre recepciones de datos**: Se proporcionará un formulario que solo acepta números enteros, donde el profesor deberá establecer el intervalo en minutos. El valor debe ser mayor o igual a uno. Una vez introducido, esta información se enviará a todos los microcontroladores mediante MQTT Out con el tema config/timer. Esta acción también sincronizará los relojes de todos los microcontroladores, de modo que a partir de entonces, todos enviarán sus datos al middleware simultáneamente.
- **Reiniciar el contenido de todas las gráficas**: Se implementará un botón que, al ser pulsado, generará un array vacío que se enviará a todos los nodos de las gráficas, dejando las gráficas completamente vacías.
- **Simulación de control de puertas y ventanas**: Se incluirán dos botones tipo switch para que el profesor indique cuándo se han abierto las puertas o ventanas del aula. Esta funcionalidad se añade porque algunas magnitudes del aula, como la temperatura y la humedad, están directamente relacionadas con el estado de las ventanas y las puertas. Dado que no se dispone de sensores para detectar la apertura o cierre de estas, se opta por permitir que el profesor controle manualmente esta información. Estos datos se reflejarán en una quinta gráfica, que mostrará dos líneas: una para la puerta y otra para la ventana, con valores de 0 (cerrada) o 1 (abierta).

El uso de Dashboard facilita enormemente la creación de la interfaz web. Una vez que el diagrama de flujo esté configurado como se ha descrito, cada nodo Dashboard se ajustará para seleccionar la pestaña donde se deberá mostrar, así como se configurarán parámetros estéticos, como el color de las líneas y el tiempo durante el cual los datos permanecen visibles en las gráficas (en este proyecto, los datos desaparecerán de las gráficas después de una hora). Finalmente, la web ajustará automáticamente proporciones y tamaños en función del tipo de navegador utilizado, diferenciando entre ordenadores y dispositivos móviles.

6. Desarrollo

En este capítulo se aborda la fase de desarrollo de la metodología ADDIE. Se describirá de manera detallada todos los pasos necesarios para instalar y configurar el software requerido para preparar el entorno de desarrollo. Además, se realizará una breve descripción de los códigos de los microcontroladores, del flujo Node-RED y de la interfaz del usuario obtenidos como resultados de esta fase; junto a la manera en la que se han encapsulado los circuitos de los sensores en cajas para su protección.

6.1. Instalación de Ubuntu en VisionFive 2

El primer paso que se decidió tomar en el desarrollo del entorno IoT fue instalar el sistema operativo en la placa VisionFive 2. La razón para empezar por aquí es simple, las placas con arquitecturas RISC-V son relativamente modernas y en consecuencia no existe demasiada información al respecto. Por esto mismo, instalar el SO es el aspecto más importante a priorizar, ya que presentaba el mayor obstáculo en el proyecto. Además, en caso de no haber conseguido completar la instalación, se habría necesitado sustituir la elección del SO o incluso la placa middleware, por lo que cuanto antes se lograse completar menos riesgos de fracasar tendría el proyecto.

Después de un largo periodo de búsqueda de información al respecto se encontró una guía oficial creada por la comunidad de Ubuntu que explicaba el proceso de instalación de Ubuntu Server para las placas VisionFive 2. A continuación se detallarán los diferentes pasos que se necesitaron tomar:

1. **Comprobar el número de revisión**: La guía explica el proceso de instalación para aquellas placas VisionFive 2 que tienen el número de revisión v1.3B. El primer paso por lo tanto fue garantizar que la placa utilizada tuviese ese número, estando este valor de revisión impreso en la superficie de la misma.

2. **Seleccionar el modo de arranque por SD-Card**: La placa VisionFive 2 puede arrancar el firmware desde SPI flash, tarjeta SD, eMMC o UART, selecciconando la fuente de arranque mediante interruptores DIP. En este caso, como el sistema operativo se cargará en una tarjeta SD, los switches deberán colocarse en las posiciones RGPIO = 0 y RGPI1 = 1, tal y como se muestra en la Figura 6.1.

3. **Descargar y flashear Ubuntu Server en la tarjeta SD**: en la página oficial de descargas de Ubuntu existe una sección para dispositivos con arquitectura RISC-V. Entre todos los dispositivos soportados se encuentra la placa VisionFive 2, por lo que se descargará la imagen preinstalada de Ubuntu Server desde aquí. Tras esto, se debe flashear la tarjeta SD con el sistema operativo que se acaba de descargar, por ejemplo, desde un sistema Ubuntu se podría utilizar el comando:

```
xzcat ubuntu-23.10-preinstalled-server-riscv64+visionfive2.img.xz
    | sudo dd bs=1M conv=fsync of=/dev/sda1
```

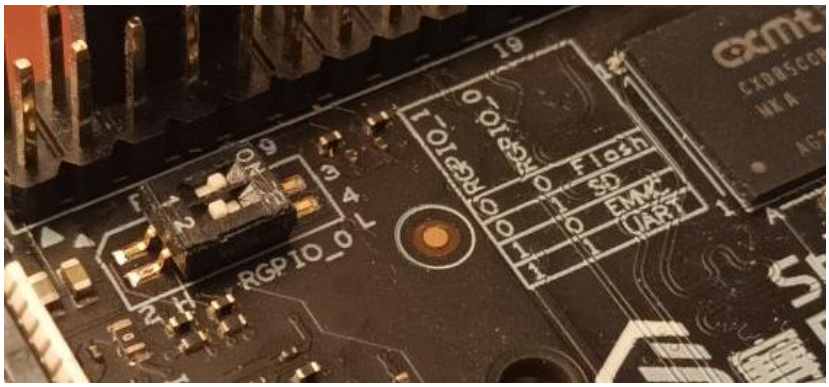


Figura 6.1: Posiciones de los switches de VisionFive2 para la instalación del SO [41]

Donde 23.10 se corresponderá con la versión instalada del sistema operativo y /dev/sda1 es el el nombre del dispositivo SD.

Una vez haya finalizado, se puede introducir la SD-card en la ranura correspondiente de la placa VisionFive 2.

4. **Preparar una conexión serial para la placa**: Pese a que la placa VisionFive 2 cuenta con puertos HDMI para ser conectada a un monitor, resulta mucho más cómodo trabajar con ella desde un ordenador usando un cliente que admita comunicaciones serial, usándose en este proyecto PuTTY.

   Para permitir la comunicación serial se necesitará un adaptador USB UART-TTL. Se deberá conectar los pines del adaptador USB con los pines GPIO de la placa VisionFive 2 como se muestra en la Figura 6.2 y tras esto se conectará el USB al puerto deseado en el ordenador.

   Tras esto, habrá que configurar los parámetros de la comunicación serial en PuTTY, más específicamente seleccionar la comunicación por Serial, una velocidad de 115200 bps y el puerto COM en el que se conectará el adaptador USB. Además, es una buena práctica asegurarse que la configuración del puerto USB está correctamente configurada en Windows, ya que en muchas ocasiones tiene una velocidad de 9600 bps por defecto. Para ello, se buscará el puerto COM utilizado en el 'Device Manager', y se accederá a 'Propiedades'>'Configuración del Puerto' y se comprobará que el número de bits por segundo es el adecuado.

5. **U-Boot y primer arranque**: Un aspecto que ha cobrado gran protagonismo en la instalación de Ubuntu ha sido U-boot, un software de arranque de código abierto utilizado en sistemas embebidos para inicializar y cargar el sistema operativo en la memoria durante el proceso de arranque. La versión de U-Boot proporcionada por el fabricante del hardware no cumple con los requisitos o estándares definidos por la especificación EBBR (Embedded Base Boot Requirements). Debido a esta incompatibilidad, no es posible arrancar directamente el sistema operativo Ubuntu en ese dispositivo sin realizar cambios manuales en la configuración o en el código de U-Boot.

   Por suerte, la imagen preinstalada cuenta con una versión de U-Boot compatible,

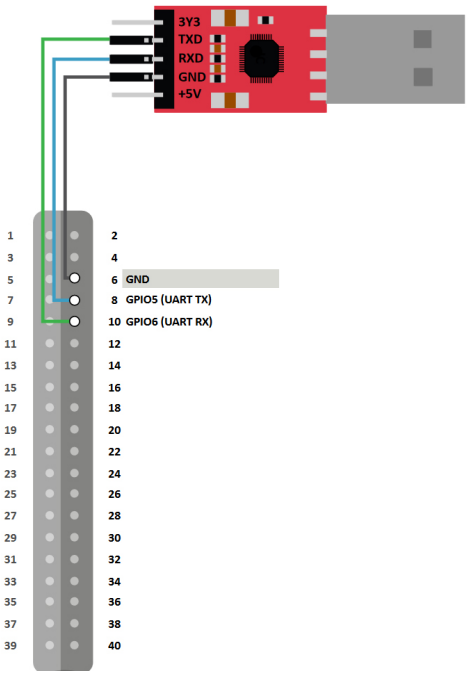


Figura 6.2: Conexión de los pines de VisionFive 2 con USB UART-TTL [35]

por lo que el proceso de actualización es muy simple. Al arrancar el sistema por primera vez se deberá pulsar 'Enter' cuando aparezca el mensaje "Hit any key to stop autoboot:", lo que permitirá acceder a la consola de U-Boot, donde se forzará al sistema a restaurar todas las variables de entorno a sus valores predeterminados de fábrica (que serán las que incorpora la imagen Ubuntu Server) usando los siguientes comandos:

```
env default -f -a
env save
```

Tras esto habrá que reiniciar la placa, la cual se iniciará con normalidad como cualquier sistema Ubuntu. Tras esperar a que el sistema inicialice por primera vez se solicitará un inicio de sesión, teniendo que utilizar el usuario y contraseña 'ubuntu'. Por último, el sistema pedirá que se cambie la contraseña, usándose en este proyecto las siguientes credenciales de inicio de sesión:

- **Usuario**: ubuntu
- **Contraseña**: StarFive2

Por último, se debe de actualizar la lista de paquetes y los paquetes desactualizados del sistema usando los siguientes comandos y una vez finalizado se deberá reiniciar el sistema:

```
sudo apt-get update
sudo apt-get upgrade
```

Con esto la instalación del sistema operativo en la placa middleware está finali-

zada. Por desgracia, usar Ubuntu Server en la placa VisionFive 2 introduce una serie de limitaciones, como no poder aprovechar la GPU que el sistema tiene integrada [41]. Por suerte ninguna de ellas es un inconveniente para la aplicación que se le va a dar en este proyecto.

Cabe destacar que la guía seguida también ofrece la opción de usar la imagen de instalación de Ubuntu en lugar de la imagen preinstalada. Originalmente se intentó usar esta opción, aunque por desgracia no fue posible, ya que requería realizar una actualización de U-Boot manual que, pese a que en un primer momento pareció funcionar, a la hora de arrancar el sistema operativo este fallaba. Por lo tanto, se decidió usar la opción que se ha descrito anteriormente.

6.2. Instalación y configuración del software para VisionFive 2

Una vez instalado el sistema operativo, este se comportará como cualquier sistema Ubuntu. Por lo tanto, el siguiente paso que se tomó en la fase de desarrollo fue instalar todo el software que la placa VisionFive 2 necesita para poder desarrollar su función como middleware satisfactoriamente; además de configurar algunos aspectos del sistema necesarios para su correcto desempeño.

6.2.1. Instalación de Node-RED y sus paquetes

El elemento software principal que necesita la placa es Node-RED, que permitirá que desempeñe su papel como middleware. Por esta misma razón, es esencial que Node-RED se inicie automáticamente como un servicio al arrancar el sistema, garantizando que siempre que la placa esté encendida Node-RED estará disponible.

Para instalar Node-RED y configurarlo como servicio existen dos alternativas posibles, instalación y configuración manual o usando un script para sistemas embebidos suministrado en la página oficial de Node-RED. Inicialmente se intentó usar la primera opción, ya que el script de instalación y configuración automática había sido diseñado para placas Raspberry Pi, por lo que era posible que esto ocasionase imprevistos.

Pese a que se logró instalar manualmente y que este se ejecutase como servicio, tras reiniciar la placa este no se lanzaba automáticamente, por lo que se decidió desinstalarlo y probar suerte con el script automático. Este script entre otras cosas elimina cualquier instalación anterior de Node-RED, establece directivas en el sistema que ayudan a facilitar su configuración e implementa algunas medidas de seguridad para proteger los flujos Node-RED desplegados de posibles ataques procedentes de una red externa. El comando para ejecutar el script en cuestión es el siguiente, que tras ser introducido inicia el instalador automáticamente como se puede ver en la Figura 6.3.

```
bash <(curl -sL https://raw.githubusercontent.com/node-red/linux-installers
    /master/deb/update-nodejs-and-nodered)
```

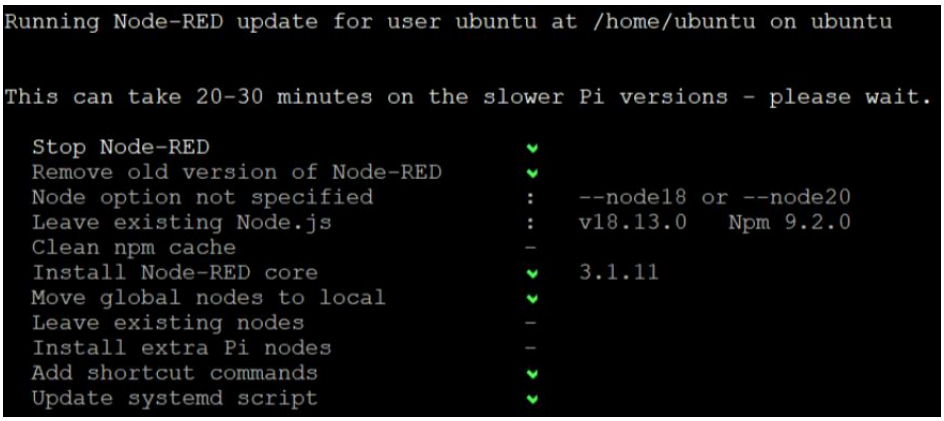


Figura 6.3: Instalador automático de Node-RED

Una vez haya finalizado la instalación, se puede usar el siguiente comando que es añadido por el instalador para habilitar Node-RED como un servicio que se ejecuta al arrancar el sistema:

```
sudo systemctl enable nodered.service
```

Con esto, ya está preparado Node-RED para ser utilizado [25]. Se puede acceder al editor de Node-RED de la placa desde el navegador de cualquier dispositivo siempre que esté conectado a la misma red que la VisionFive 2, introduciendo en la barra de búsqueda la siguiente página, que se correspondo con el socket de la placa asociado al servicio Node-RED:

```
http:\\Direción-IP-VisionFive:1880
```

Por último, solo es necesario instalar el paquete Dashboard de Node-RED para tener disponibles los nodos y herramientas necesarios para levantar una interfaz web desde Node-RED. La manera más sencilla de hacerlo es usando el gestor de paquetes NPM. Los siguientes comandos muestran como instalar NPM en caso de que aún no lo esté y el paquete Dashboard:

```
sudo apt install -y nodejs npm
sudo npm install -g node-red-dashboard
```

6.2.2. Mosquitto MQTT

Otra de las funciones esenciales que deberá cumplir la placa VisionFive 2 será el papel de broker MQTT para permitir la comunicación entre Node-RED y los micro-controladores ESP32-C3. Para ello, el primer paso será descargar Mosquitto MQTT usando el siguiente comando:

```
sudo apt-get install mosquitto mosquitto-clients
```

Una vez finalice la instalación, se añadirán unas credenciales de autenticación para los clientes MQTT, que pese a no ser estrictamente necesario, es una buena medida para incrementar la seguridad. Para poder conectarse al servidor MQTT, es necesario conectarse en primer lugar a la red Wi-Fi del entorno IoT, la cual ya cuenta con un

usuario y contraseña. Sin embargo, si un atacante lograse conectarse a la red necesitaría obtener además las credenciales de autenticación MQTT para poder suscribirse a temas y publicar mensajes fraudulentos. Para crear estas credenciales, se ha usado el siguiente comando, que creará un usuario con el nombre especificado y tras esto pedirá que se introduzca su contraseña:

```
sudo mosquitto_passwd -c /etc/mosquitto/passwd NOMBRE_USUARIO
```

Las credenciales de usuario MQTT creadas y utilizadas en este proyecto son las que se indican a continuación. Nótese que para que Mosquitto MQTT pueda atender a todos los componentes del entorno IoT es necesario que en Node-RED se introduzcan estos parámetros en la configuración de autenticación del cliente MQTT; y que en el programa de los ESP32-C3 se realice la conexión al broker MQTT pasándo este usuario y contraseña en la función de conexión.

- **Usuario MQTT**: `IoT_MQTT_user`
- **Contraseña**: `IoT_MQTT_pass`

A continuación, se deberá modificar el archivo de configuración de Mosquitto. En él, se deberá de impedir conexiones de usuarios anónimos, vincular el archivo donde se ha guardado las credenciales del usuario MQTT y hacer que MQTT escuche por todas las interfaces de red (lo cual permitirá conexiones tanto desde localhost y como desde dispositivos remotos) a través del puerto 1883. Para ello, se usará el siguiente comando para editar el fichero de configuración y se añadirá la información que aparece en la Figura 6.4.

```
sudo nano /etc/mosquitto/mosquitto.conf
```

```
  GNU nano 7.2              /etc/mosquitto/mosquitto.conf *
# Place your local configuration in /etc/mosquitto/conf.d/
#
# A full description of the configuration file is at
# /usr/share/doc/mosquitto/examples/mosquitto.conf.example

#pid_file /run/mosquitto/mosquitto.pid

persistence true
persistence_location /var/lib/mosquitto/

log_dest file /var/log/mosquitto/mosquitto.log

include_dir /etc/mosquitto/conf.d

allow_anonymous false
password_file /etc/mosquitto/passwd

bind_address 0.0.0.0
port 1883
```

Figura 6.4: Archivo de configuración de Mosquitto

Con esto solo queda reiniciar el servicio Mosquitto para que se apliquen los cambios [36]. Se puede comprobar que todo esté funcionando correctamente con los siguientes comandos:

- `sudo systemctl status mosquitto`, para comprobar que el servicio esté correctamente levantado

- `sudo ss -tuln | grep 1883`, para comprobar que Mosquitto escucha por todas las interfaces de red usando el puerto 1883.

6.2.3. Configuración adicional

Ya están instalados todos los elementos esenciales para que la placa StarFive 2 cumpla su propósito en el sistema IoT. Sin embargo, aún quedan una serie de elementos pendientes de ser configurados para asegurar que tanto el middleware Node-RED como el broker MQTT funcionen de una manera óptima.

El primero de estos elementos consiste en configurar la interfaz de red para poder utilizar un adaptador USB Wi-Fi. Como ya se explicó anteriormente, la placa VisionFive 2 solo soporta conexiones de red a través de un adaptador Ethernet, lo que no es una opción válida, ya que se desea utilizar una red Wi-Fi creada por el extensor de red. Por lo tanto, se usará un adaptador USB Wi-Fi genérico que sea compatible con Ubuntu en uno de los puertos USB de la placa VisionFive 2. Se puede comprobar que el dispositivo es reconocido por Ubuntu con el comando `lsusb`.

Tras esto, habrá que modificar la configuración de esta interfaz de red, estableciendo una dirección IP estática dentro del rango de direcciones IP admitidas por el extensor de red. Para poder configurar correctamente esto, es necesario identificar previamente el nombre de la interfaz en cuestión, que se puede obtener con el comando mostrado en la Figura 6.5.

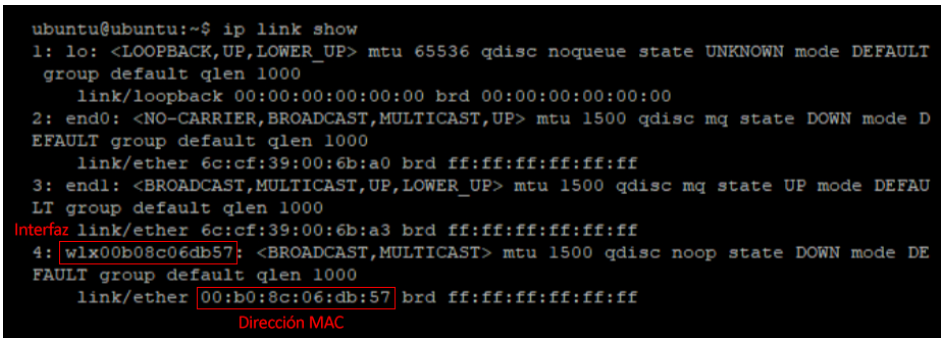


Figura 6.5: Obtener dirección MAC y nombre de la interfaz de red

Utilizando el comando `sudo nano /etc/netplan/01-netcfp.yaml` se accederá con el editor de texto al fichero donde se almacenan las configuraciones de las interfaces de red, en el que habrá que añadirla configuración de red descrita anteriormente, como se puede ver en la Figura 6.6.

Como medida alternativa en caso de que el extensor de red no funcione por restricciones del firewall de la red, se podría cambiar esta configuración a una dirección IP suministrada dinámicamente por un Punto de Acceso móvil habilitado en un Smartphone con sistema Android.

El último aspecto relevante a configurar es la hora del sistema. En la interfaz web que Node-RED construirá la información se mostrará en gráficas que relacionará la magnitud medida con el tiempo. Node-RED tomará esta información de la hora

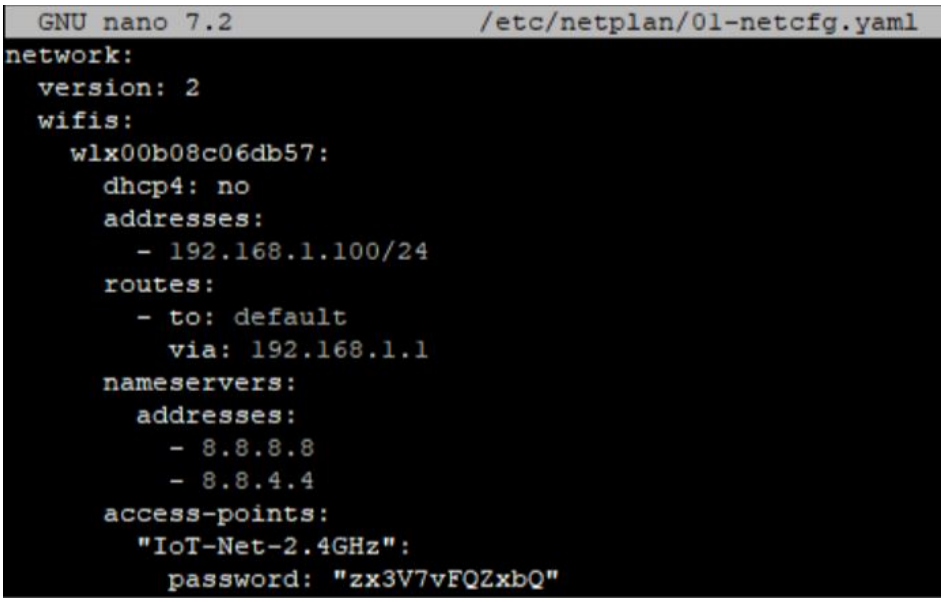


Figura 6.6: Configuración de la interfaz de red en VisionFive 2

interna del sistema, por lo que si no está debidamente configurada el tiempo mostrado en estas gráficas será incorrecto. Para evitar este problema se utilizar el protocolo NTP que automáticamente tomará la hora local de un servidor NTP fiable. En principio, el protocolo NTP debería de estar habilitado y configurado por defecto, aunque se puede usar el comando `timedatectl` para comprobar toda la información relativa a la fecha y hora del sistema.

El único parámetro que es necesario configurar es la zona horaria, que se deberá de ajustar a la hora de Madrid con el siguiente comando:

```
sudo timedatectl set-timezone Europe/Madrid
```

Nótese que cuando el sistema es apagado, al ser enciendido posteriormente mantendrá la misma hora y fecha, por lo que será responsabilidad del protocolo NTP actualizar hora. En caso de que la placa no tenga acceso a Internet, no se podrá actualizar el reloj y por lo tanto la hora mostrada en las gráficas de Node-RED no se corresponderán a la real.

Para concluir, se deberá tomar la dirección MAC del adaptador Wi-Fi USB para su posterior uso cuando se configure el extensor de red. Para ello, se puede usar nuevamente el comando de la Figura 6.5.

6.3. Red Wi-Fi IoT

El próximo paso en el desarrollo IoT será establecer la red Wi-Fi que será usada para la comunicación entre los microcontroladores y middleware, además de permitir a la placa VisionFive 2 poder conectarse a Internet. Como ya se explicó en la fase de diseño, para establecer esta red Wi-Fi se usará el extensor Wi-Fi TP-LINK AC750.

Para configurar este dispositivo, el primer paso será descargar la aplicación Tether

de TP-Link en un dispositivo móvil, la cual permitirá realizar la configuración del extensor de manera sencilla. Tras esto, se deberá crear una cuenta TPLink-ID y usar esas credenciales para iniciar sesión en la aplicación.

A continuación, se deberá conectar el extensor Wi-Fi a la corriente eléctrica en una zona cercana al router de la red del centro educativo. Una vez se encienda, a través de la aplicación Tether se pulsará la opción de buscar nuevos dispositivos, seleccionando la opción de RE Estándar. Al poco tiempo se detectará automáticamente el extensor Wi-Fi, tras lo que se pedirá que se cree una contraseña para acceder al menú de confiuración del extensor. En el proyecto se ha usado la contraseña '`IoTAP-UCA-Pass`'.

El siguiente paso será seleccionará la red que será extendida por el dispositivo, necesitando introducir su contaseña para que el extensor pueda conectarse. Esta será la misma contraseña que la red generada utilizará y no podrá ser modificada, siendo esta la principal desventaja de haber usado un extensor Wi-Fi en lugar de un punto de acceso.

Una vez conectado, el extensor de red creará dos redes, una de 2.4GHz y otra 5GHz. Puesto a que los microcontroladores ESP32-C3 no soportan Wi-Fi 5GHz, se deberá de usar la primera opción. Desde la aplicación se podrá modificar el SSID de la red generada, usándose a lo largo de todo el proyecto el nombre de red '`IoT-Net-2.4GHz`'.

Con los aspectos básicos configurados no será necesario seguir usando la aplicación. En su lugar y con el extensor de red aún conectado a la red del centro, se introducirá en la barra de búsqueda de un navegador web la dirección IP que fue asignada al extensor Wi-Fi, accediendo al menú de configuración de este último que cuenta con opciones configurables adicionales. Se deberá de usar la contraseña de configuración `IoTAP-UCA-Pass` para acceder a las opciones de configuración, como ya se indicó anteriormente.

En primer lugar se accederá a Configuración > Control de Acceso > Lista Blanca, donde se puede establecer las direcciones MAC de aquellos dispositivos que podrán usar la red extendida. Cualquier dispositivo con una MAC que no figure en esta lista no podrá usar la red IoT. De esta manera, pese a que el extensor de red no ofrezca ocultación de dispositivos al no genera una red Wi-Fi privada, sino tan solo extiende a otra, permitirá aislar los componentes IoT de los otros dispositivos de la red del centro educativo. Dentro de esta *WhiteList* se introducirá la dirección MAC del adaptador Wi-Fi del middleware anteriormente obtenida, así como las direcciones MAC de los dos microcontroladores, tal y como se puede ver en la Figura 6.7.

Por último se deberá configurar el servicio DHCP del extensor Wi-Fi para evitar que se le asigne la misma IP a varios dispositivos. Más específicamente, debido a que se usará una dirección IP estática en la placa VisionFive 2, esta deberá ser excluida de la pool de direcciones IPs asignables por el protocolo DHCP. Además de esto, para evitar problemas es recomendable asegurar que ningún otro dispositivo en la red educativa tenga asignada alguna de las direcciones IPs dentro de la pool. En la Figura 6.8 se puede ver una de las configuraciones DHCP utilizadas durante el desarrollo, en el que la placa VisionFive 2 tenía asignada de manera estática la dirección .100, siendo por lo tanto excluida del rango de IPs asignables.

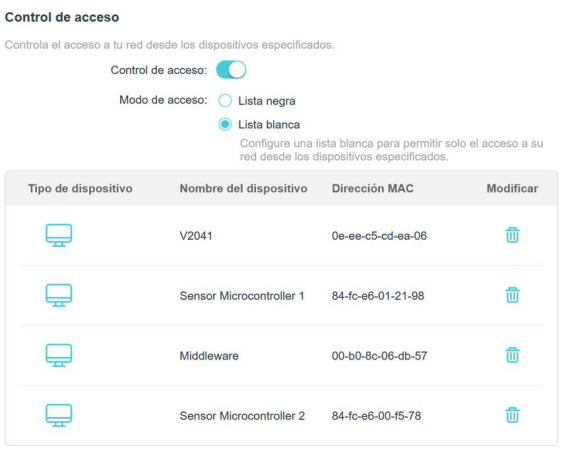


Figura 6.7: Lista blanca de dispositivos que pueden usar la red IoT

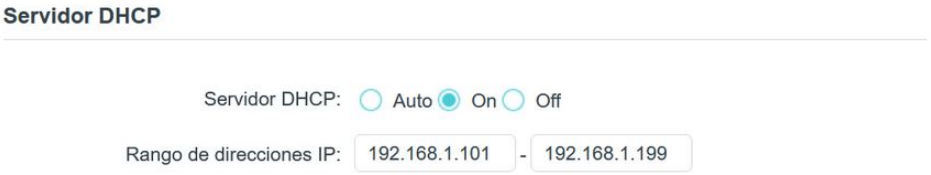


Figura 6.8: Configuración de servidor DHCP del extensor de red

6.4. Programación de microcontroladores ESP32-C3

Con todos los demás dispositivos ya configurados, el siguiente paso que se tomó en el desarrollo del entorno IoT fue programar los microcontroladores, siendo el primer paso instalar y configurar el entorno de desarrollo.

Como ya se comentó anteriormente, se ha usado Arduino IDE junto con un paquete creado por Espressif que añade soporte para todas las placas de desarrollo de la familia ESP32. Para instalar este paquete, se tiene que acceder en Arduino IDE a la opción Preferencias > URL y añadir en el recuadro designado la siguiente URL:

```
https://raw.githubusercontent.com/espressif/arduino-esp32
    /gh-pages/package_esp32_dev_index.json
```

Tras esto, se accede a Tools > Board > Board Manager, y en la barra de búsqueda se introduce 'ESP32' y se instala el paquete de Espressif, que descargará todo lo necesario para ofrecer soporte a todas las placas ESP32, incluida la ESP32-C3. En la Figura 6.9 se puede ver el paquete en cuestión que se debe instalar.

Una vez termine de descargarse, se seleccionará la placa en Tools > Board > esp32 y se seleccionará la opción ESP32-C3 Dev Module y se indicará también el puerto COM al que se conectará la placa. Con esto ya se puede programar el microcontrolador usando Arduino IDE. [29]

Antes de proceder con el desarrollo de los programas hubo que instalar también los paquetes que serán utilizados, indicados previamente en la fase de diseño. Esto se podrá hacer desde el menú 'Library Manager' localizado en la barra izquierda de

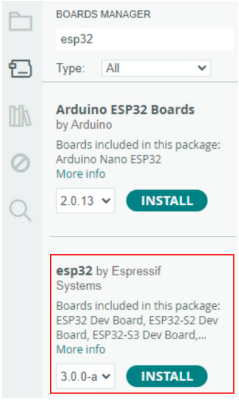


Figura 6.9: Paquete Arduino-IDE con soporte para ESP32-C3

herramientas, teniendo que introducir el nombre del paquete en la barra de búsqueda. Con esto se mostrarán todos los paquetes que han coincidido con el nombre introducido, usando como guía el nombre del autor para instalar el paquete correcto.

Con esto, ya está preparado todo el entorno de programación. Cabe resaltar que en todos los códigos desarrollados se usará la herramienta Serial Monitor de Arduino IDE para mostrar los mensajes de depuración dejados en el código que ayudarán a identificar posibles fallos y mejoras. Además, esta herramienta Serial cuenta con la opción de mostrar la hora, minutos y segundos en la que se escribió un mensaje, lo cual por ejemplo servirá para comprobar si los eventos síncronos ocurren con la frecuencia deseada. Para usar comunicación Serial en los microcontroladores, se han usado entre otras las siguientes funciones que están integradas de manera nativa en Arduino IDE:

- `Serial.begin(9600)` para iniciar la comunicación serial con una tasa de transferencia de 9600bps.
- `Serial.print("Mensaje")` para transmitir por serial la cadena de caracteres "Mensaje".

Antes de comenzar a programar todos los elementos, se realizaron todas las conexiones físicas necesarias entre los microcontroladores y sensores según lo indicado en la fase de diseño, ya que es un requisito indispensable para poder leer los datos procedentes de los sensores.

La programación de los microcontroladores se ha realizado de forma incremental, de manera que inicialmente se probaron y programaron elementos individuales, como los sensores, y posteriormente una vez todos estos elementos funcionaban de manera individual, se unieron los códigos en uno solo para formar el programa final de los microcontroladores. Este enfoque permite aislar los problemas, de manera que no será necesario encontrar en el programa completo que pieza es la que está fallando. Los diferentes códigos individuales desarrollados son los siguientes:

- **Prueba de Funcionamiento de placa**: El primer código desarrollado buscó probar si la instalación del paquete para los microcontroladores ESP32 había sido exitosa. Para ello, en este pequeño programa se comprobó que un pequeño LED integrado en la placa de desarrollo ESP32-C3 parpadease intermitentemente. Además, este código también sirve para comprobar que cada una de las placas

no tengan un defecto de fábrica que no permitan reconocer la conexión USB con el ordenador desde el que se subirá el programa.

- **Prueba Serial**: Con el objetivo de probar el correcto funcionamiento de la herramienta Serial de Arduino IDE, se programó este sencillo código que inicia la comunicación Serial y escribe un mensaje de prueba de manera periódica.
- **Prueba de Wi-Fi**: El siguiente programa realizado fue uno en el que se usaría el paquete Wi-Fi instalado para conectarse a la red Wi-Fi local. En este código se espera a estar conectado a la red Wi-Fi y tras esto se muestra la IP dinámica asignada por el servidor DHCP.
- **Obtener MAC**: Permite obtener la dirección MAC de la interfaz de red del ESP32-C3. Este programa es necesario para poder añadir los microcontroladores a la *WhiteList* del extensor de red.
- **Sensor de Humedad-Temperatura**: Este código inicializa y lee el sensor de temperatura-humedad DHT22 utilizando las funciones descargadas en el paquete específico de este sensor.
- **Sensor de Luz**: Este programa inicializa la comunicación I2C necesaria para comunicarse con el sensor de luz en los pines especificados, tras lo cual se puede realizar lecturas en el sensor usando su biblioteca específica.
- **Sensor de movimiento**: Para este sensor se desarrollaron dos códigos. Uno de ellos simplemente comprueba si el sensor ha detectado movimiento y muestra un mensaje acorde. El segundo código es un tanto más complejo, ya que en él se programa el comportamiento para cuantificar los movimientos estableciendo de manera sencilla un delay de 8 segundos entre dos movimientos consecutivos.

Una vez terminados todos estos códigos individuales, se unieron en un programa que recrea el comportamiento que debe tener los microcontroladores pero sin llegar a enviar los datos por MQTT, optando en su lugar por escribirlos por la terminal serial. De esta manera se puede comprobar si un fallo de funcionamiento en los microcontroladores es debido a la comunicación con el middleware o es culpa del propio programa. A este código se le añadió además la interrupción síncrona, en la que se muestran los datos leidos por los sensores, y asíncrona, en la que se incrementa el contador auxiliar de movimientos en intervalos de al menos 8 segundos. [20]

Finalmente, al código anterior se le añadió comunicación con el middleware usando MQTT y la función de callback para recibir los mensajes de configuración, obteniendo de esta manera el código completo.

Cabe resaltar que el mismo código se puede cargar en tantos microcontroladores como se desee. Sin embargo, es necesario realizar algunos cambios en ciertas variables para asegurar que la comunicación MQTT sea semánticamente correcta:

- **Temas usado para transmitir los datos**: Cada microcontrolador cuenta con 4 temas propios que debe usar para transmitir cada uno de los 4 tipos de lecturas que realiza. Por lo tanto, cuando se vaya a cargar el programa en cada microcontrolador, habrá que asegurarse que el número que aparece en cada uno de los temas de datos se corresponda con el número del microcontrolador. Por ejemplo,

el microcontrolador 1 tendría los siguientes temas de datos:

```
const char* TopicHumedad = "datos/1/humedad";
const char* TopicTemperatura = "datos/1/temperatura";
const char* TopicMovimiento = "datos/1/movimiento";
const char* TopicLuz = "datos/1/luz";
```

- **Nombre de cliente MQTT**: Mosquitto no admite que existan dos clientes MQTT con el mismo nombre, por lo que cada microcontrolador deberá conectarse con su propio nombre de cliente MQTT. Para simplificar los nombres, se usará un nombre genérico en todos los microcontroladores seguido del número del microcontrolador en cuestión. Continuando con el ejemplo anterior, el sensor 1 deberá tener el siguiente valor:

```
const char* mqtt_clientID = "ESP32C3_Client_1";
```

Por último, es necesario comprobar que la dirección IP de la placa Middleware, así como el SSID y contraseña de la red Wi-Fi, tienen los valores correctos, ya que en caso contrario no se podría realizar la comunicación. Estos valores están localizados en las siguientes constantes:

```
const char* ssid = "IoT-Net-2.4GHz";
const char* password = "zx3V7vFQZxbQ";
const char* mqtt_server = "192.168.1.100";
```

Junto a esta documentación se puede encontrar tanto el programa final de los microcontroladores como todos los códigos y pruebas parciales anteriormente descritas, todos ellos debidamente comentados para facilitar la comprensión de los mismos.

6.5. Programación Node-RED

El único aspecto que falta para completar el desarrollo del entorno IoT es la creación del flujo Node-RED que actuará como middleware y como la interfaz gráfica al mismo tiempo. Para acceder al editor de Node-RED se puede usar un navegador web en un dispositivo que esté en la misma red que la placa VisionFive 2 accediendo a la siguiente página:

```
http:\\Dirección_IP_VisionFive2:1880
```

Utilizando este editor se ha creado el flujo final del middleware, el cual se puede ver en la Figura 6.10.

Usando los nodos MQTT In se consigue que el flujo reciba los mensajes de datos procedentes de los microcontroladores, usando el tema configurado en cada nodo para separar los diferentes tipos de datos y enviarlos a sus gráficas correspondientes; mientras que usando un nodo MQTT Out se publicarán mensajes con el tema de configuración del temporizador al que todos los microcontroladores están suscritos, permitiendo así

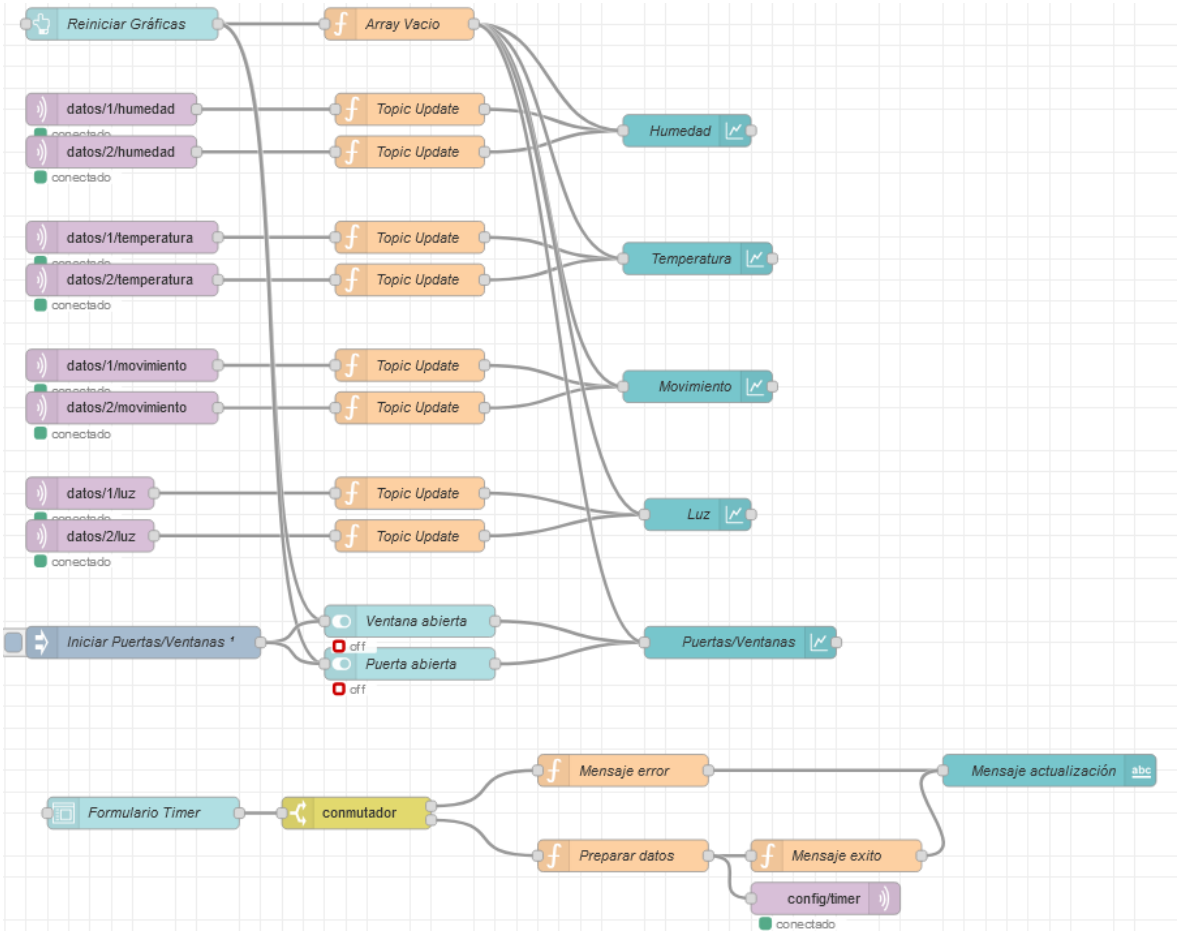
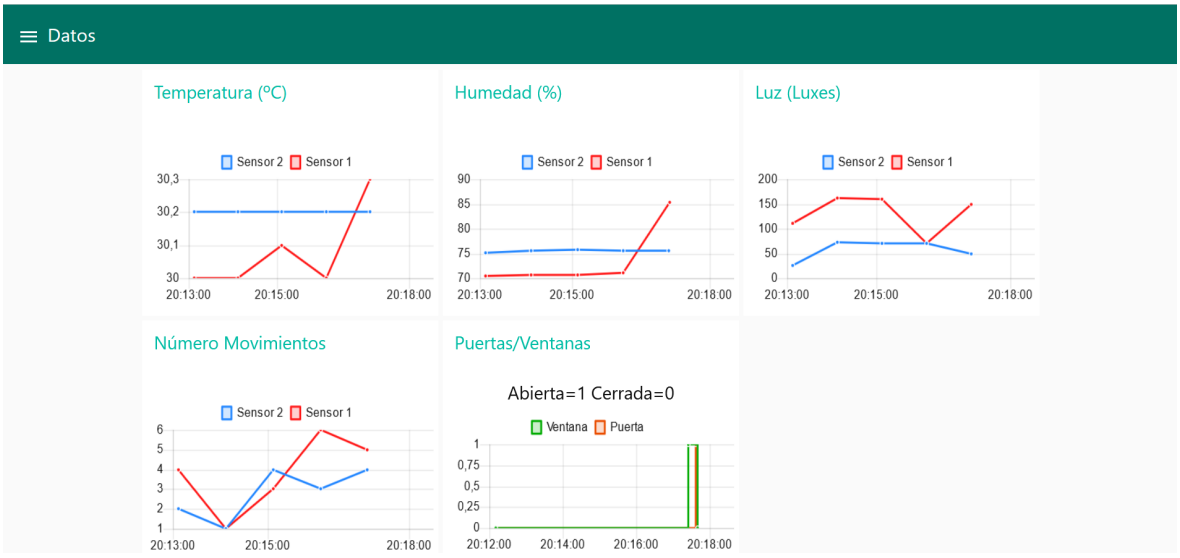


Figura 6.10: Flujo Node-RED utilizado por el middleware

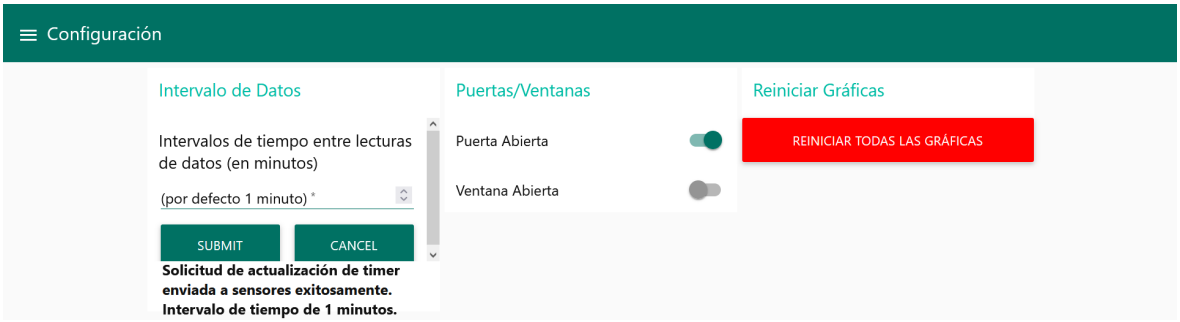
que el profesor cambie los intervalos de tiempo en los que se transmiten los datos desde los microcontroladores hasta el middleware.

Además, ha sido necesario establecer la información relativa al broker MQTT para que Node-RED pueda actuar como cliente. Para ello, se ha establecido la dirección IP en la que el broker se encuentra como 'localhost' en lugar de usar la dirección IP que tiene la placa VisionFive 2 dentro de la red. Esto permite que, en caso de que la dirección IP de la placa VisionFive 2 cambie, no sea necesario modificar dicha dirección en el software middleware, puesto a que se usa la dirección de loopback. Gracias a esto, el flujo Node-RED no cuenta con ningún elemento que deba ser configurado en caso de que se produzca un cambio de la red. También se configurará el nombre de cliente MQTT y contraseña para poder conectarse al broker Mosquitto.

El otro aspecto relevante desarrollado en Node-RED es la interfaz gráfica, utilizando el paquete Dashboard. Se han usado nodos incluidos en dicho paquete para crear las dos pestañas de la interfaz según lo establecido en la fase de diseño. Además, Dashboard cuenta con una gran cantidad de opciones configurables para personalizar muchos aspectos visuales de la interfaz del usuario, como paleta de colores e iconos, lo que ha permitido hacerla bastante agradable visualmente. No se entrará en más detalles sobre las configuraciones gráficas ya que no tiene demasiada relevancia, sino que en su lugar se mostrará en las Figuras 6.11 el resultado final.



(a) Ventana de datos



(b) Venta de configuración

Figura 6.11: Interfaz desarrollada para el sistema IoT

6.6. Montaje físico de circuitos en caja

Con el entorno IoT ya creado en su totalidad y antes de proceder a realizar pruebas e implementar el sistema en un aula de la Escuela Superior de Ingeniería, se decidió introducir los microcontroladores, cables y sensores en pequeñas cajas de plástico que los protejan de golpes y que ayuden a camuflar los componentes del sistema IoT.

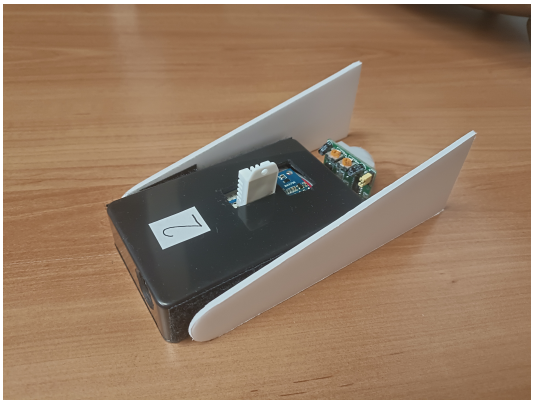
Para ello, se introdujo cada *breadboard* con los circuitos dentro de una de las cajas, y se cortaron múltiples partes de la misma. Se crearon agujeros rectangulares en la zona superior del contenedor con la forma del sensor de luz y de temperatura-humedad. Estos orificios sirven para que estos dos sensores puedan sobresalir de la caja, de manera que puedan leer correctamente las condiciones del aula. Puesto a que la cara superior de la caja era considerablemente más alta que la breadboard usada, fue necesario colocar una pequeña plataforma bajo el circuito para elevar los sensores lo suficiente como para que sobresaliesen por completo.

En la parte trasera se hizo un agujero por el que se pudiese conectar un USB a la placa para poder suministrarle alimentación, y en la parte frontal se hizo un pequeño espacio por el que poder conectar los cables a los pines del sensor de movimiento. Tras

terminar de agujerear los contenedores, se pegó a cada uno de sus laterales una tira de velcro de bucles, que servirán como sujeción para las palas que limitarán el campo de visión del sensor de movimiento.

Las palas fueron diseñadas en una hoja de cartón-piedra, siendo dibujadas con una forma que permita tapar un campo de visión adecuado, contando con suficiente longitud y anchura para ello. Tras esto, fueron recortadas y a cada una de ellas se le pegó una pequeña sección de velcro de enganche. Finalmente, cada una de las dos cajas fue numerada para evitar confusiones a la hora de situarlas en el aula. En las Figuras 6.12 se puede ver como han quedado las dos cajas de sensores una vez finalizada la fase de desarrollo.

(a) Caja de sensores 1 vista desde delante



(b) Caja de sensores 2 vista desde atrás

Figura 6.12: Cajas de sensores desarrolladas

7. Implementación

En este capítulo se aborda la fase de implementación de la metodología ADDIE. En ella, se recogen todos los tipos de pruebas realizadas sobre los componentes individuales, los subsistemas y el sistema al completo, tanto en un entorno de pruebas controlado como en un entorno educativo real, todo ello destinado a demostrar que el sistema podría ser implementado y adoptado en un aula real de docencia real para desempeñar su propósito como herramienta de monitoreo.

7.1. Pruebas piloto

En primer lugar y antes de probar el sistema IoT en un aula real, se realizaron una serie de pruebas en un entorno doméstico controlado. De esta manera, se podría detectar de manera previa errores en el sistema, localizar potenciales problemas y limitaciones y establecer una serie de consideraciones sobre el entorno desarrollado.

Las primeras pruebas de este tipo que se realizaron ocurrieron de manera simultánea a la fase de desarrollo. Más específicamente, se realizaron pruebas unitarias para comprobar que todos los sensores funcionaban individualmente a través de los códigos de prueba desarrollados y documentados en la fase de desarrollo. De esta manera, se logró entender a nivel práctico el funcionamiento de cada uno de los 3 sensores, siendo especialmente relevante el del sensor de movimiento PIR 501, ya que permitió identificar los valores exactos de retardo y distancia que ofrecía las lecturas más precisas.

Tras ésto, se probó que el subsistema formado por los tres sensores, el microcontrolador y el circuito que los conecta funcionase correctamente. Esta prueba consistió en un programa con el mismo comportamiento que debía tener el código final del microcontrolador, pero en lugar de enviar los datos leídos al middleware estos se mostraban a través del monitor serial en el ordenador. De esta manera se pudieron eliminar aquellos errores relacionados con el cableado y el comportamiento de los microcontroladores y sensores antes de construir el sistema final, reduciendo en gran medida el tiempo de depuración al haber aislado estos errores del resto del sistema.

Las siguientes y últimas pruebas de este tipo realizadas se llevaron a cabo una vez concluidos todos los pasos descritos en la fase de desarrollo. El objetivo de estas pruebas es evaluar el funcinamiento del sistema IoT en su conjunto en un entorno en el que todas las condiciones estén completamente controladas. Para ello, se conectó el extensor Wi-Fi a la red local, se situaron cada una de las dos cajas con los microcontroladores y sensores en diferentes habitaciones y se pusieron a personas delante de cada uno de ellos para actuar como sujetos de pruebas. Algunas de las pruebas realizadas incluyeron respirar cerca del sensor de temperatura-humedad para observar incrementos en sus lecturas, andar o quedarse quito frente al sensor de movimiento o iluminar/tapar el sensor de luz para comprobar que sus lecturas cambiaban adecuadamente. También

se envió diferentes mensajes de configuración para ajustar los temporizadores de los microcontroladores a 2, 4 y 10 minutos; para comprobar si dichos cambios se aplicaban correctamente y si ambos microcontroladores enviaban las lecturas de sus sensores a la vez.

Todas las lecturas realizadas por los sensores bajo estas condiciones de prueba fueron satisfactiorias, pudiéndose ver reflejadas en las gráficas de la interfaz de Node-RED, por lo que de esta manera se comprobó que el comportamiento del sistema completo era correcto. Además, estas pruebas permitieron ver algunos aspectos que pueden resultar de interés a la hora de poner en marcha el sistema en un aula real, como puede ser que los microcontroladores son capaces de manetenerse conectados al broker MQTT incluso cuando están bastante alejados del extensor de red o que las cajas donde están los sensores y microcontroladores son especialmente delicadas, puesto a que un movimiento brusco puede resultar en que algún cable se desconecte, lo que supondría en que al menos uno de los 3 sensores dejase de funcionar.

7.2. Pruebas en un aula de docencia

Una vez se terminaron las pruebas en el entorno controlado, se concertó un día para realizar las pruebas en un aula de seminarios vacía en el pasillo F de la Escuela Superior de Ingeniería, la cual cumple las características identificadas en la sección del Análisis del Entorno. En la Figura 7.1a se puede ver la distribución del aula en cuestión.

Las dos cajas de sensores se conectaron en los extremos adyacentes a la pared de dos de las mesas de alumnos (Figura 7.1b), mientras que el middleware se conectó a una toma de corriente en la mesa del profesor, junto a un portátil en el que se abrió la página con la interfaz web (Figura 7.1c). Se explicó el funcionamiento del sistema y la forma en la que un profesor debería usar el sistema al tutor del TFG, D. Nestor Mora Nuñez, quién pudo configurar los microcontroladores a su gusto y entendió con facilidad la información mostrada en las gráficas.

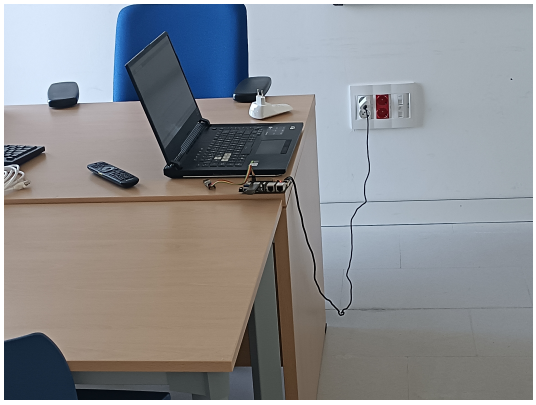
Sin embargo, realizar las pruebas en un entorno real mostró la presencia de un problema que se había identificado en etapas tempranas del proyecto, las restricciones de seguridad de la red. La Escuela Superior de Ingeniería cuenta con una red con una configuración de seguridad excelente. Debido a que el uso de dispositivos de red no autorizados, como switches, hubs o extensores de red, pueden suponer un serio problema de seguridad, ya que si no cuentan con una adecuada configuración pueden ser instrumentos utilizados por usuarios malintencionados para realizar ataques, es una práctica habitual que los firewalls de red bloqueen todo el tráfico procedente de este tipo de dispositivos no autorizados. Este es el caso de lo ocurrido en las pruebas realizadas, ya que a pesar de haber configurado correctamente el extensor de red para que utilizase la red pública de la ESI, los dispositivos que se conectaban a la red IoT eran bloquedos.

Por esta misma razón, se utilizó para la realización de estas pruebas un punto de acceso móvil en un SmartPhone con Android, ya que este sistema operativo en sus últimas versiones admite poder conectar a dicho AP hasta 8 dispositivos, siendo

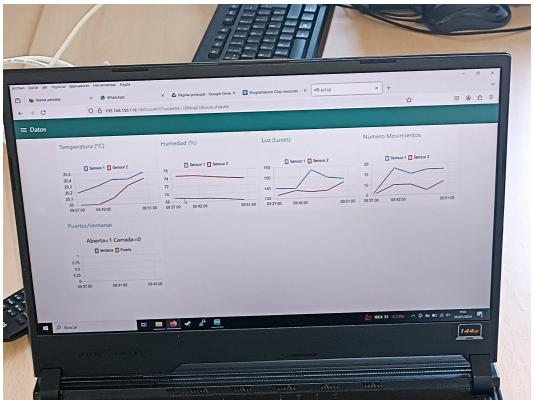
adecuado para los pocos elementos del entorno IoT que se estaban probando. Pese a que esta alternativa no es la más elegante ni la más escalable, para el ámbito de este proyecto es más que suficiente, ya que excede las capacidades del proyecto poder cambiar la configuración del firewall de toda la red de la ESI para permitir el tráfico procedente del extensor de red del entorno IoT, siendo por lo tanto una opción inviable. A excepción de esto, las pruebas en el aula fueron un rotundo éxito, no surgiendo ningún problema imprevisto en la realización de las mismas. En la Figura 7.1d se pueden ver las gráficas finales obtenidas tras haber finalizado las pruebas.

(a) Aula de pruebas

(b) Localización de los sensores



(c) Localización de la placa middleware



(d) Gráficas con resultados de las pruebas

Figura 7.1: Pruebas realizadas en un aula de la ESI

8. Evaluación

En este capítulo se aborda la fase de evaluación en la metodología ADDIE. En ella se evaluarán las diferentes medidas que se han tomado en las anteriores fases, enfocándolas hacia el cumplimiento del funcionamiento del sistema y de los criterios de calidad del mismo. También se abordarán de manera resumida aquellos problemas más relevantes que han aparecido a lo largo del proyecto.

8.1. Evaluación funcional del sistema

El entorno IoT desarrollado ha logrado alcanzar los diferentes requerimientos técnicos que debía satisfacer, tanto cada uno de los elementos a nivel individual, como todos ellos en su conjunto actuándo como un único sistema; además de cumplir satisfactoriamente las diferentes necesidades identificadas en la fase de análisis y decisiones tomadas durante el diseño del mismo. A continuación se hace una breve evaluación de los diferentes elementos del sistema que permiten afirmar el éxito en el desarrollo del proyecto:

- Los **sensores** han sido sido analizados, su configuración diseñada y su utilización programados cuidadosamente, de manera que han logrado superar las diferentes pruebas de funcionamiento realizados sobre cada uno de ellos.
- Los **microcontroladores** son capaces de leer correctamente los sensores, así como enviar dichas lecturas al middleware Node-RED y recibir mensajes de configuración del mismo. Además, su timer interno les permiten controlar la periodicidad de las lecturas de estos datos y su módulo Wi-Fi integrado permite realizar su transmisión de manera inalámbrica.
- El **extensor de red** logra cumplir su función como punto de acceso en un entorno de red completamente controlado, ya que en caso de ser necesario se puede realizar una reconfiguración de la red local. Sin embargo, en un entorno de una red educativa real, las diferentes restricciones de seguridad impuestas por el administrador de la red impide en algunas ocasiones el funcionamiento del extensor; debiendo usarse en estos casos una alternativa más limitada como el punto de acceso de un Smartphone.
- La **placa VisionFive 2** se ha podido adaptar a la perfección a las necesidades del proyecto, actuando tanto de middleware como de broker MQTT. El proceso de instalación de un sistema operativo que no tiene soporte de fábrica es sumamente sencillo gracias a los esfuerzos de la comunidad Ubuntu en el caso de este proyecto. Una vez instalado el sistema operativo, esta placa se comporta como un sistema Ubuntu convencional, lo cual ha demostrado la viabilidad de los computadores con arquitectura RISC-V para su uso en entornos IoT.
- Las **conexiones físicas** entre los microcontroladores y sensores son conceptual-

mente sencillas, aunque deben realizarse con sumo cuidado y delicadeza ya que un fallo de conexión o contacto en uno de los cables supone inevitablemente un fallo de funcionamiento en uno de los sensores. Este tipo de fallos son muy dificiles de arreglar, ya que a simple vista el cableado parece correcto, por lo que se puede pensar erroneamente que el causante es un error software.

- La **comunicación entre middleware y microcontroladores** es sencilla y fiable gracias al uso de MQTT, el cual logra a través de la suscripción a temas que cada elemento reciba únicamente los mensajes con alguno de los topics que necesita utilizar, reduciendo de esta manera la carga de la red.

8.2. Evaluación de la calidad del sistema

El sistema desarrollado satisface los diferentes criterios de calidad establecidos en la planificación del proyecto. A continuación se describe como se ha cumplido cada uno de ellos:

- **Fiabilidad**: El sistema desarrollado es capaz de funcionar de manera continuada y sin fallos. En las diferentes pruebas realizadas se ha demostrado el correcto funcionamiento de los diferentes componentes tanto a nivel individual como en su conjunto. El procesamiento de los datos y las comunicaciones entre los componentes del entorno se realiza completamente en la red local, sin necesidad de usar servicios de terceros en la nube que actuaría como un punto único de fallo en caso de que se produjese una caida de estos, de manera que todos los componentes del sistema IoT se encuentran en la red local bajo el control del administrador del sistema. Además, el uso de MQTT permite tener una comunicación inalámbrica fiable, al garantizar la recepción de los mensajes gracias al uso de QoS de nivel 1.
- **Seguridad**: El sistema desarrollado utiliza una red local protegida por un nombre usuario y una contraseña para evitar que usuarios no deseados puedan conectarse a la red usada por los microcontroladores y middleware, teniendo además configurada una lista blanca de dispositivos permitidos, por lo que cualquier dispositivo con una MAC distinta a la de los microcontroladores o VisionFive no podrá usar la red extendida. Además, el broker MQTT ha sido configurado para atender únicamente a aquellos clientes que puedan identificarse con unas credenciales concretas, por lo que será muy dificil que un usuario mal intencionado pueda suscribirse a temas y publicar falsos mensajes que sean recibidos por los componentes del sistema.
- **Privacidad**: La privacidad cobra gran importancia en este proyecto debido a que en el sistema se recolectan y transmiten datos de un grupo humano. Por esta razón, es importante que el sistema no recolecte, almacene ni procese datos que puedan permitir la identificación individual de los usuarios sin su consentimiento explícito. El entorno IoT desarrollado cumple con estos criterios de privacidad, puesto que todos los datos recolectados son tomados de las condiciones ambientales del aula, de forma que a pesar que estas lecturas sean afectadas por el comportamiento de los alumnos, el grupo del que se toman los datos, no se puede

averiguar de qué usuario procede cada lectura. Además de esto y por razones éticas, se deberá de informar tanto a discípulos como a docentes sobre el sistema que se encuentra instalado en el aula, asegurándose que entiendan como funciona para que puedan consentir a que los datos del aula sean tomados.

- **Usabilidad**: El sistema IoT desarrollado es sumamente fácil de utilizar. Tan solo es necesario conectar las cajas de sensores y la placa VisionFive 2 a una fuente de corriente eléctrica para que el profesor pueda acceder a una interfaz milimalista y fácil de entender que le permitirá ver los datos leídos y realizar algunas configuraciones sobre los sensores. Además, el sistema trata de ser lo más transparente y menos intrusivo posible para no distraer a los alumnos que estén dando clase. A esto hay que añadir que el profesor no necesitará instalar ninguna aplicación para usar el sistema, tan solo necesitará conectarse a la interfaz utilizando un navegador web.
- **Modularidad**: El sistema está comformado por un conjunto de elementos, desempeñando cada uno un papel concreto dentro de todo el conjunto. Debido a esto el sistema cuenta con una modularidad intrínseca, lo cual permite aislar los errores que se produzcan, facilitar el mantenimiento individual de cada uno de los elementos y simplificar el proceso de incorporar nuevas funcionalidades al poder configurar y probar cada uno de los componentes del sistema unitariamente.
- **Escalabilidad**: Pese a que el proyecto solo ha sido desarrollado para dos grupos de sensores, el número total de estos se puede aumentar fácilmente. El mismo código desarrollado puede ser programado en tantos microcontroladores como se desee, aunque será necesario modificar los valores de algunas constantes. Además, MQTT Mosquitto es ampliamente conocido por poder manejar hasta decenas de miles de clientes conectados simultaneamente gracias a su bajo consumo de recursos, por lo que la comunicación no se convertirá en un cuello de botella. El único aspecto que es un poco limitante para la escalabilidad del sistema es el flujo Node-RED desarrollado, que deberá de ser modificado ligeramente para poder recibir datos de más microcontroladores.
- **Extensibilidad**: Existen dos enfoques para añadir nuevas funcionalidades al sistema desarrollado. La primera de ellas será en los microcontroladores, siendo el cambio más probable la adición de nuevos tipos de sensores. La otra alternativa sería añadir nuevas opciones de configuración a la interfaz del profesor. Ambos tipos de adiciones son fáciles de añadir y requerirían realizar ligeros cambios en secciones muy concretas del código de los microcontroladores y en la interfaz para reflejar los cambios realizados.
- **Eficiencia energética**: El sistema cuenta con un consumo energético muy reducido gracias a que todos los componentes han sido optimizados para utilizar la menor cantidad de recursos posibles. Los sensores utilizados cuentan con un consumo de energía muy bajo, lo cual se ve respaldado por un diseño del software de los microcontroladores que tan solo consume ciclos de reloj en los momentos en los que se produce un evento como son la recepción de un mensaje o una interrupción. Además, la placa VisionFive 2 utiliza un sistema operativo basado en terminal que ha sido configurado para tan solo utilizar aquellos servicios esenciales necesarios para que el entorno pueda funcionar.

- **Portabilidad**: El sistema desarrollado es relativamente sencillo de portear a cualquier otro modelo de dispositivos en caso de ser necesario. La placa VisionFive 2 se comporta como un sistema Ubuntu convencional, de manera que todo el software y la configuración utilizados puede instalarse en una placa diferente sin ningún problema. Por el lado de los microcontroladores, el programa desarrollado en Arduino IDE utiliza bibliotecas que son compatibles con una extensa cantidad de microcontroladores, de manera que el único cambio que se debería realizar si se decidiese utilizar otro tipo de microcontrolador es sobre el temporizador que controla las interrupciones síncronas, ya que es el único paquete utilizado específico de la familia ESP32. Además, la interfaz del usuario es independiente del tipo de máquina o sistema operativo desde el que se desea visualizar, al ser accedido desde el navegador web.

8.3. Problemas encontrados

A lo largo de las otras cuatro fases de la metodología seguida en este proyecto han ido apareciendo una serie de problemas que no habían sido considerados inicialmente, lo cual ha ocasionado que se hayan tenido que realizar algunos cambios que han podido incluso llegar a afectar a fases anteriores. Pese a que muchos de ellos ya han sido tratados en sus capítulos respectivos, a continuación se ha realizado una recopilación de aquellos que han resultado ser más relevantes o interesantes, cuales han sido sus causas y que consecuencias han tenido en el proyecto:

- **Instalación de U-Boot manual**: Originalmente se pretendió usar una imagen con instalador en lugar de una preinstalada cuando se iba a meter el sistema operativo en la placa VisionFive 2. Para ello, se descargaron dos fichero desarrollados por la comunidad de Ubuntu que debía ser aplicado en el menú de configuración de U-Boot accesible al arrancar la placa. Para lograr esto era necesario subir el archivo a la VisionFive 2, utilizándose para ello un servidor TFTP configurado en un computador personal que compartiría estos dos archivos. Para poder descargarlos, era neceario conectar la placa a la red, necesitándose para ello una conexión al router via cable Ethernet. Una vez descargaron, se aplicaron una serie de comandos concretos para actualizar U-Boot. Sin embargo y pese a que a primera vista pareció funcionar, a la hora de instalar el sistema operativo aparecía un error relacionado a un problema de kernel en el arranque. Por esta misma razón, se usó la imagen preinstalada que contaba con una versión de U-Boot válida directamente integrada.
- **Falsos positivos en el sensor de movimiento**: El sensor PIR 501 es un sensor con un funcionamiento muy específico, lo que dificultó considerablemente diseñar una manera de darle uso en la medición de los movimientos en el aula, ocasionando que se necesitase adaptar múltiples veces el comportamiento del programa de los microcontroladores. Originalmente el sensor solamente era leido cuando se producía un flanco de subida, ya que esto significaba que se había detectado movimiento. Sin embargo, las lecturas con falsos positivos eran habituales, ya que muchas veces un solo movimiento era reconocido múltiples veces. Tras investigar al respecto se descubrió que eran habituales las interfecerencias en este tipo de

sensores, por lo que se restringió el tiempo entre dos lecturas consecutivas para hacer frente a este problema que habitualmente era causado por un 'debounce' en la señal digital dada por el sensor.

- **Soldadura en los pines del sensor de luz**: El sensor GY-302 BH1750 comprado traía la placa del sensor y los pines por separados, siendo necesario soldar ambos para poder conectarlo a una placa *breadboard*. Una vez unidas ambas piezas, se realizó la prueba unitaria para comprobar que este sensor funcionase correctamente, pero en lugar de la lectura esperada se obtuvo un mensaje de error (Figura 8.1). Tras indagar en la posible causa, se encontró un comentario en un foro que explicaba que ese problema era causado porque uno de los pines no estaba bien unido, por lo que se decidió reaplicar la soldadura. Esta solución hizo que el problema desapareciese por completo.

```
[BH1750] ERROR: undefined error
Sensor de luz BH1750 inicializado
[BH1750] Device is not configured!
Nivel de luz: -2.00 lux
```

Figura 8.1: Mensaje de error en sensor de luz BH1750

- **Uso de broker MQTT Online**: Inicialmente se pretendía usar un broker MQTT online como MQEX. Sin embargo, durante la realización de las pruebas del sistema en el entorno doméstico se pudo comprobar que las desconexiones de los microcontroladores eran demasiado habituales, ya que la conexión con el servidor MQTT era muy inestable (Figura 8.2). Por esto mismo, se decidió iniciar el servicio Mosquitto MQTT en la placa VisionFive 2, pasando a actuar esta como broker MQTT.

```
12:56:51.353 -> Intentando conexión MQTT...Conectado
12:57:01.959 -> Intentando conexión MQTT...Conectado
12:57:12.474 -> Intentando conexión MQTT...falló, rc=2 Intentar de nuevo en 5 segundos
12:57:17.928 -> Intentando conexión MQTT...Conectado
12:57:28.449 -> Intentando conexión MQTT...Conectado
12:57:38.998 -> Intentando conexión MQTT...Conectado
```

Figura 8.2: Desconexiones frecuentes por usar un broker MQTT online

- **Hora incorrecta**: Este problema apareció en dos ocasiones diferentes durante la realización de las pruebas del sistema mientras se utilizaba la interfaz desarrollada. En la primera ocasión, el tiempo que aparecía en las gráficas con la hora en la que se habían recibido los datos era incorrecta, estando un par de horas retrasadas debido a que la zona horaria no era la correcta. La segunda instancia de este problema surgió cuando, al arrancar la placa tras múltiples días de haber estando realizando pruebas, aparecía la fecha y hora del día anterior; siendo el causante en esta ocasión que la placa no se había podido conectar a Internet por problemas con la red doméstica y en consecuencia no pudo obtener de un servidor NTP la hora actual.
- **Desconexiones intermitentes de los microcontroladores**: Durante las pruebas del sistema realizadas en el ambiente controlado se encontró una instancia

de un error muy peculiar en la que los microcontroladores se conectaban y desconectaban intermitentemente del broker MQTT. Tras investigar que es lo que podía fallar en el código, se encontró que accidentalmente se había puesto en ambos microcontroladores el mismo ID de cliente MQTT cuando se realizaba la conexión al servidor MQTT. Mosquitto no admite múltiples clientes con el mismo identficiador, de manera que cuando uno de los microcontroladores se conectaba, el otro era desconectado por el propio servidor, por lo que ambos microcontroladores entraban en un ciclo infinito de desconexiones y reconexiones. La solución a este problema fue establecer un ID propio a cada microcontrolador que forma el sistema.

- **Firewall de la Escuela Superior de Ingeniería**: Pese a que en un entorno controlado no había ningún problema para usar el extensor de red, la configuración de seguridad de la red pública de la ESI impedía el acceso de los dispositivos IoT a la red. Por esta misma razón se tuvo que recurrir al uso de un punto de acceso en un teléfono Android. Este problema no tiene una solución real, ya que a no ser que se reciba permiso expreso de la institución en cuestión y se gestione el proceso con el administrador de la red se carecen de los medios para hacerle frente.

Tal y como se puede observar, la mayoría de problemas han aparecido en la fase de desarrollo o de pruebas, siendo problemas aislados que no han cambiado en gran medida la visión global del proyecto. Esto refuerza la importancia de realizar un análisis y diseño exhaustivo, puesto a que cualquier error aparecerá en una fase muy posterior del proyecto, obligando a reevaluar las fases anteriores.

Parte III

Epílogo

9. Conclusiones

En este último capítulo se describen los objetivos del proyecto alcanzados, se detallan las lecciones aprendidas tras el desarrollo del presente proyecto y se identifican las posibles oportunidades de mejora sobre el sistema desarrollado.

9.1. Objetivos alcanzados

El objetivo fundamental del proyecto, que consistía en el análisis, diseño, desarrollo, implementación y evaluación de un entorno IoT para monitorear el comportamiento de los alumnos en un aula de docencia, se ha logrado cumplir e incluso superar con creces las expectativas iniciales. El sistema IoT desarrollado no solo logra satisfacer los requisitos funcionales inicialmente propuestos, sino que además cuenta con funcionalidades de configuración que permitirán al profesor ajustar algunos aspectos del sistema a su gusto. Además, el resultado del proyecto cumple con todas las características propias de la calidad del software deseadas, lo que facilitará la realización de futuros proyectos para expandirlo. En la Figura 9.1 se muestra un diagrama que funciona a modo de resumen del sistema final desarrollado, permitiendo entender a simple vista y de forma simplificada todos los componentes utilizados y como interactúan entre sí.

Respecto al estudio de la viabilidad y uso de la arquitectura RISC-V en entornos IoT, se ha logrado demostrar que no existe una diferencia real frente al uso con otras arquitecturas propietarias como x86 o ARM. Las diversas comunidades de sistemas operativos y de los IDEs de programación de los microcontroladores han acogido los dispositivos RISC-V, creando paquetes e imágenes listas para ser utilizadas de una manera rápida y sencilla, además de guiar en todos los pasos que se deben seguir a través de guías bien redactadas. Una vez todo el software base está preparado, los dispositivos con esta arquitectura se comportan como cualquier otro, habiendo presentado las ventajas propias de los dispositivos RISC-V como pueden ser los costos de adquisición reducidos y el trabajo colaborativo de aquellos voluntarios que ayudan a que la arquitectura siga creciendo.

Finalmente, se pudieron realizar pruebas sobre el funcionamiento del sistema en un aula real, demostrando que, pese a que aparecieron ciertas restricciones que se salían del control del proyecto, obligando a realizar ciertos cambios para poder ser rodeadas, la base establecida por el sistema IoT es robusta y fiable. Sin embargo y pese a lo que originalmente se buscaba, no se ha podido realizar ninguna prueba real en un aula con alumnos, sino que se ha tenido que simular el comportamiento de los alumnos en una clase vacía. La razón de esto ha sido una serie de retrasos en la planificación inicial del proyecto que ocasionó que el prototipo funcional del sistema ya probado finalizase a principios del mes de julio, habiendo concluido por aquel entonces las clases para alumnos de todos los cursos y titulaciones.

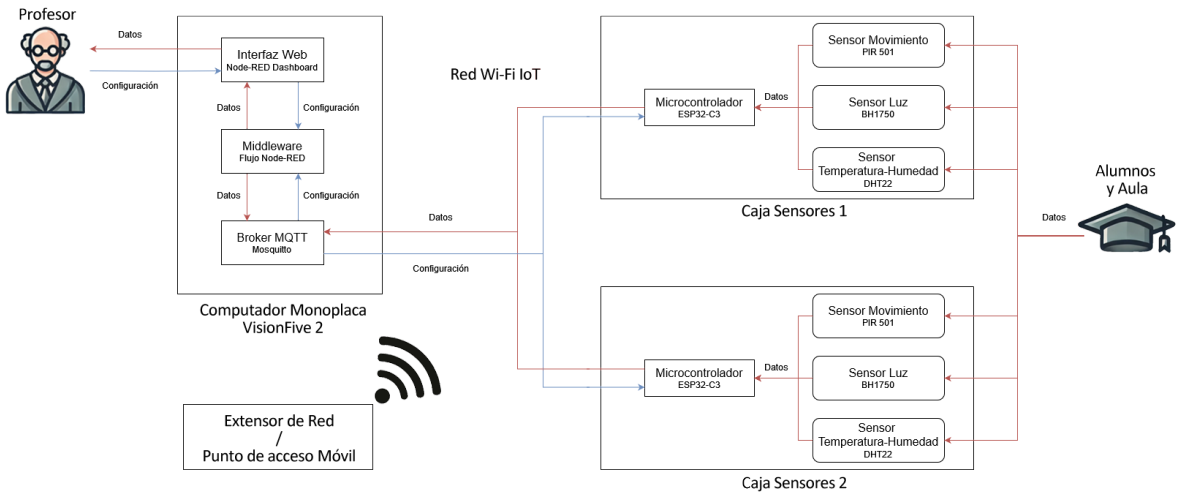


Figura 9.1: Diagrama resumen del sistema final desarrollado

9.2. Lecciones aprendidas

A lo largo del desarrollo del proyecto se han ido adquirido una serie de buenas prácticas. Una de las más importantes ha sido la gestión del tiempo y de las prioridades, ya que haber realizado una planificación y gestión efectiva del tiempo ha sido determinante para el éxito del proyecto. El uso de una metodología de desarrollo basada en el modelo ADDIE ha facilitado la división del proyecto en fases claras y a establecer límites a los mismas, lo que ayudó a mantener el enfoque. Además, haber priorizado el cumplimiento de tareas críticas en primer lugar ayudó a asegurar que se abordaran los mayores desafíos del proyecto con suficiente antelación.

Otro aspecto que ha cobrado una gran importancia ha sido elaborar una documentación de manera continuada en forma de borradores con los conceptos clave, medidas y decisiones tomadas y problemas solucionados. Esto no solo ha ayudado en gran medida a la elaboración de este documento sino que además servía como elemento motivador al poder ver el progreso que iba teniendo el proyecto.

Finalmente, la identificación de posibles puntos de fallos en el sistema de manera adelantada a que estos se manifestasen ha ayudado a reducir en gran medida el tiempo invertido en la depuración del sistema. Además, la realización de diferentes pruebas para construir el sistema de manera inicialmente individual con la visión de posteriormente conectar todos los elementos desarrollados ha ayudado a aislar aquellos problemas inicialmente no previstos. Todo esto ha resultado en un desarrollo del proyecto muy controlado y poco accidentado.

Desde un punto de vista tecnológico, se han adquirido conocimientos sobre la arquitectura RISC-V, que era completamente desconocida para el autor antes de comenzar el proyecto. RISC-V ha demostrado ser una alternativa con un gran potencial frente a las arquitecturas que dominan el mercado gracias a ser *open* y todas las características que esto conlleva.

La última lección aprendida es la importancia de la elección de componentes hardware y software en un proyecto. Una correcta selección no solo puede ayudar

significativamente en posteriores fases de desarrollo e implementación del sistema, sino que además pueden contribuir a cumplir los requerimientos de calidad impuestos al proyecto, como por ejemplo ha sido la manera en la que el uso de componentes ligeros y de bajo consumo ha ayudado a que el sistema sea más energéticamente eficiente.

En definitiva, el proyecto realizado no solo ha sido una experiencia que ha permitido adquirir conocimientos desde el punto de vista técnológico, de gestión de proyectos y de buenas prácticas; sino que además el trabajo desarrollado ha supuesto una experiencia sumamente interesante y gratificante, generando en el autor un gran interés por continuar desarrollando proyectos de esta índole en un futuro.

9.3. Trabajo futuro

Pese a que el proyecto ha sido finalizado alcanzando todos los objetivos impuestos inicialmente, se han detectado durante su desarrollo una serie de mejoras que podrían ser realizadas en el sistema para acentuar la capacidad de monitorización del mismo:

- **Añadir nuevos sensores al sistema**: Una de las mejoras más sencillas de implementar serían añadir nuevos tipos de sensores a las cajas de sensores. Estos permitirían aportar nueva información valiosa, como por ejemplo la cantidad de sonido o de CO2 que hay en cada zona del aula, permitiendo al profesor tomar decisiones más informadas.
- **Persistencia**: Pese a que el principal objetivo del proyecto es permitir el monitoreo en tiempo real del comportamiento de los alumnos, una mejora que se podría realizar es añadir algún mecanismo para almacenar en la placa VisionFive 2 los datos recogidos. De esta manera, el profesor podría comprobar las lecturas de los sensores una vez más cuando la clase hubiese finalizado, pudiendo realizar cambios a la estructura de la siguiente lección de manera anticipada.
- **Aplicación móvil para configurar los sensores**: Un cambio que mejoraría bastante la calidad del sistema desarrollado sería añadir la posibilidad de configurar aquellos parámetros de los microcontroladores que están almacenados en variables dentro del código a través de una aplicación que utilizase Bluetooth. De esta manera no sería necesario recompilar el código y subirlo a todos los microcontroladores manualmente cuando algún parámetro como la dirección IP de la placa middleware cambiase.
- **Lecturas más precisas**: En el estado actual del proyecto, los microcontroladores realizan en cada intervalo de tiempo una única lectura en tres de los cuatro sensores, siendo realizadas justo antes de enviar los datos al middleware para que sean mostrados al profesor. Pese a que es funcional, el hecho de que puedan transcurrir largos periodos de tiempo sin lecturas significa que se puede perder información relevante, como picos de temperatura, humedad o luz. Por esta misma razón, una mejora de las funcionalidades del entorno sería realizar lecturas con mayor frecuencia (por ejemplo cada 10 segundos), de manera que se almacenase el valor máximo y mínimo que cada sensor haya detectado. Una vez transcurriese el periodo de espera que marcase la frecuencia con la que se envían los datos

al middleware, estos valores extremos llegarían al profesor, mostrándole de esta manera información más relevante.

- **Entorno IoT formado por múltiples aulas**: Pese a que el sistema creado ha sido enfocado para ser usado en una sola aula, es posible construir un entorno IoT que se extienda a lo largo de todas las clases de un centro educativo. Para ello, se podría replicar el sistema construido en todas las aulas y configurar los dispositivos middleware para que, además de mostrar los datos en la interfaz del profesor, estos también sean enviados a un servidor central donde se recogerá y almacenará las lecturas de todas las clases. Incluso sería posible que múltiples aulas que están cerca entre sí pudiesen compartir un mismo middleware, lo cual reducirían en gran medida el coste total del entorno, al ser la placa VisionFive 2 el elemento más caro del sistema con diferencia.

  Esta red IoT sería un mecanismo excelente para establecer un sistema de control centralizado donde se pueda observar las condiciones de todas las aulas, permitiendo tomar decisiones en base a las mismas. Por ejemplo, un aula en la que los sensores de luz detectasen que las lámparas estan encendidas pero en la que no se detectan movimientos significaría que accidentalmente se quedaron las luces encendidas cuando la última clase terminó.

Finalmente, pese a que el sistema IoT por limitaciones causadas por el calendario académico no se ha podido probar en un aula con alumnos reales, a fecha de la redacción de este documento se está intentando concertar un día para realizar pruebas prácticas del sistema en una lección real en el aula de Tecnología del instituto IES Mar de Cádiz localizado en el municipio de El Puerto de Santa María. La realización de estas pruebas además cuenta con un doble propósito, evaluar el grado de utilidad que el sistema desarrollado supone para un profesor en un ambiente real, y utilizarse para realizar una jornada de divulgación sobre la Ingeniería Informática dirigida a los alumnos, tratando de fomentar el interés del alumnado hacia esta disciplina que cuenta con un futuro tan prometedor.

Información sobre Licencia

Información sobre Licencia

Todo el software utilizado en el proyecto es de código abierto y puede ser instalado gratuitamente desde las páginas oficiales o a través de los repositorios de Ubuntu. A continuación se presentan las licencias específicas para las principales herramientas y librerías utilizadas:

Arduino IDE: La licencia de Arduino IDE es una licencia GNU General Public License v2.0. Esta licencia permite a los usuarios copiar, modificar y distribuir el software bajo los términos de la GPL.

- Enlace a la licencia: https://www.gnu.org/licenses/old-licenses/gpl-2.0.html

Node-RED: Node-RED se distribuye bajo la licencia Apache License 2.0, que permite a los usuarios usar, modificar y redistribuir el software bajo ciertas condiciones.

- Enlace a la licencia: https://www.apache.org/licenses/LICENSE-2.0

Mosquitto MQTT: El protocolo MQTT y las librerías asociadas están disponibles bajo la licencia Eclipse Public License 1.0, que permite su uso, modificación y distribución bajo ciertas condiciones.

- Enlace a la licencia: https://www.eclipse.org/org/documents/epl-v10.html

Ubuntu: Ubuntu es un sistema operativo de código abierto basado en la licencia GNU General Public License, que permite su libre uso, distribución y modificación.

- Enlace a la licencia: https://www.gnu.org/licenses/gpl-3.0.html

Bibliografía

[1] Adafruit Industries. Digital humidity and temperature sensor am2302. Technical report, Adafruit Industries, 2024. URL https://cdn-shop.adafruit.com/datasheets/Digital+humidity+and+temperature+sensor+AM2302.pdf. Datasheet.

[2] Amazon Web Services. What is an integrated development environment (ide)?, 2024. URL https://aws.amazon.com/es/what-is/ide/. Definición IDE.

[3] Arduino. Arduino software, 2024. URL https://www.arduino.cc/en/software. Página oficial.

[4] Art of Circuits. Gy-302 bh1750 16-bit serial output light lux sensor module, 2024. URL https://artofcircuits.com/product/gy-302-bh1750-16-bit-serial-output-light-lux-sensor-module. Especificaciones técnicas.

[5] Autor del paquete. Documentación paquetes arduino ide, 2024. URL https://www.arduino.cc/reference/en/libraries/. Documentación obtenida de la página de Arduino IDE y GitHub.

[6] CEOL. ¿ya conoces el modelo addie para diseñar tu formación?, 2021. URL https://www.ceolevel.com/ya-conoces-el-modelo-addie-para-disenar-tu-formacion. Imagen.

[7] Mirian Cifredo Chacón. Diseño de computadores empotrados, 2023. Apuntes de la asignatura, definición interrupción.

[8] Cisco Systems, Inc. What is a firewall?, 2024. URL https://www.cisco.com/site/us/en/learn/topics/security/what-is-a-firewall.html. Definición Firewall.

[9] Computer Weekly. ¿qué es un extensor de red y para qué sirve?, 2020. URL https://www.computerweekly.com/es/opinion/Que-es-un-extensor-de-red-y-para-que-sirve. Definición y funcionamiento de extensor red Wi-Fi.

[10] Enfang Cui, Tianzheng Li, and Qian Wei. Risc-v instruction set architecture extensions: A survey. *IEEE Access*, 11:24696–24711, 2023. ISSN 2169-3536. doi: 10.1109/ACCESS.2023.3246491. URL https://ieeexplore.ieee.org/document/10167491. Definición, características y aplicaciones de RISC-V.

[11] Digikey. Esp32-c3-devkitm-1, 2024. URL https://www.digikey.es/en/products/detail/espressif-systems/ESP32-C3-DEVKITM-1/13684315. Imagen.

[12] Electronics-Lab. Pir sensor pinout with jumper setting sensitivity time adjustment bis0001 ic labeling diagram, 2024. URL https://www.electronics-lab.com/project/pir-motion-sensor-led-ceiling-light-arduino-compatible/pir-sensor-pinout-with-jumper-setting-sensitivity-time-adjustment-biss0001-ic-labeling-diagram/. Imagen.

[13] ESI. Ingeniería del software, 2022. Apuntes de la asignatura sobre criterios de calidad.

[14] Espressif Systems. Esp32-c3 devkitm-1, 2024. URL https://docs.espressif.com/projects/esp-idf/en/v5.2/esp32c3/hw-reference/esp32c3/user-guide-devkitm-1.html. Imagen.

[15] Espressif Systems. Esp32-c3 datasheet. Technical report, Espressif Systems, 2024. URL https://www.espressif.com/sites/default/files/documentation/esp32-c3_datasheet_en.pdf. Datasheet.

[16] HiveMQ. Mqtt essentials: Part 6 - mqtt quality of service levels, 2024. URL https://www.hivemq.com/blog/mqtt-essentials-part-6-mqtt-quality-of-service-levels/. Web sobre los niveles de calidad de servicio de MQTT.

[17] Instructables. How to use dht-22 sensor: Arduino tutorial, 2015. URL https://www.instructables.com/How-to-use-DHT-22-sensor-Arduino-Tutorial/. Imagen.

[18] ISO. Iso 25010: Software quality model, 2024. URL https://iso25000.com/index.php/normas-iso-25000/iso-25010. Definición de calidad del software según este estándar.

[19] Juniper Networks. ¿qué es un punto de acceso en redes?, 2024. URL https://www.juniper.net/mx/es/research-topics/what-is-an-access-point-in-networking.html. Definicion AP.

[20] Last Minute Engineers. Configuring and handling esp32 gpio interrupts in arduino ide, 2024. URL https://lastminuteengineers.com/handling-esp32-gpio-interrupts-tutorial/. Información sobre como programar las interrupciones en el ESP32-C3 con Arduino IDE.

[21] Lenovo. Instruction set architecture (isa), 2024. URL https://www.lenovo.com/us/en/glossary/instruction-set-architecture/. Definición ISA.

[22] Somayya Madakam, R. Ramaswamy, and Siddharth Tripathi. Internet of things (iot): A literature review. *Journal of Computer and Communications*, 3(5):164–173, 2015. doi: 10.4236/jcc.2015.35021. URL https://www.scirp.org/journal/paperinformation?paperid=56616. Definición IoT.

[23] MPJA. Hc-sr501 pir motion detector. Technical report, MPJA, 2024. URL https://www.mpja.com/download/31227sc.pdf. Datasheet.

[24] MQTT.org. Mqtt: A lightweight messaging protocol, 2024. URL https://mqtt.org/. Definición y funcionamiento de MQTT.

[25] Node-RED. Node-red on raspberry pi, 2024. URL https://nodered.org/docs/getting-started/raspberrypi. Guía instalación Node-RED.

[26] Node-RED. Node-red, 2024. URL https://nodered.org/. Web oficial.

[27] NXP Semiconductors. Um10204: I2c-bus specification and user manual. Technical report, NXP Semiconductors, 2021. URL https://www.nxp.com/docs/en/user-guide/UM10204.pdf. Definición I2C y funcionamiento básico.

[28] ONU. Agenda de desarrollo sostenible, 2024. URL https://www.un.org/sustainabledevelopment/es/development-agenda/. Definición de los Objetivos de Desarrollo Sostenible.

[29] Profetolocka. Programando el esp32-c3 con arduino ide, 2021. URL https://www.profetolocka.com.ar/2021/06/02/programando-el-esp32-c3-con-arduino-ide/. Guía.

[30] Punto Flotante. *Manual del Usuario: Sensor de Movimiento PIR HC-SR501*. Punto Flotante, 2017. URL https://puntoflotante.net/MANUAL-DEL-USUARIO-SENSOR-DE-MOVIMIENTO-PIR-HC-SR501.pdf.

[31] Renesas Electronics Corporation. Mcu programming and peripherals: Gpio, 2024. URL https://www.renesas.com/us/en/support/engineer-school/mcu-programming-peripherals-01-gpio. Definición GPIO.

[32] Salud Electrónica. Calidad del software, 2021. URL https://saludelectronica.com/calidad-del-software/. Definición de calidad del software.

[33] Soumyalatha and Shruti G. Hegde. Study of iot: Understanding iot architecture, applications, issues and challenges. In *Proceedings of the 1st International Conference on Innovations in Computing & Networking (ICICN16)*, Bangalore, India, 2016. International Journal of Advanced Networking & Applications (IJANA). URL https://www.researchgate.net/profile/Soumyalatha-Naveen/publication/330501274_Study_of_IoT_Understanding_IoT_Architecture_Applications_Issues_and_Challenges/links/5c434fea458515a4c731d4bb/Study-of-IoT-Understanding-IoT-Architecture-Applications-Issues-and-Challenges.pdf. Arquitectura IoT.

[34] Adamantia G. Spatioti, Ioannis Kazanidis, and Jenny Pange. A comparative study of the addie instructional design model in distance education. *Information*, 13(9): 402, 2022. doi: 10.3390/info13090402. URL https://www.mdpi.com/2078-2489/13/9/402. Información sobre qué es ADDIE y sus fases.

[35] StarFive. Visionfive 2 quick start guide, 2024. URL https://www.scs.stanford.edu/~zyedidia/docs/visionfive/visionfive2_qsg.pdf. Imagen.

[36] Steve's Internet Guide. Mosquitto username and password authentication - configuration and testing, 2022. URL http://www.steves-internet-guide.com/mqtt-username-password-example/. Guía sobre como crear las credenciales de autenticación en Mosquitto.

[37] Studio Pieters. Bh1750 light sensor module, 2024. URL https://www.studiopieters.nl/bh1750-light-sensor-module/. Imagen.

[38] Talent.com. Salario medio para ingeniero informático en españa, 2024, 2024. URL https://es.talent.com/salary?job=ingeniero+informtico. Salario medio de Ingeniero Informático en España.

[39] TP-Link. Tp-link re200 - range extender specifications, 2024. URL https://www.tp-link.com/us/home-networking/range-extender/re200/#specifications. Web de especificaciones.

[40] U-Boot Contributors. Repositorio u-boot en github, 2024. URL https://github.com/u-boot/u-boot. Definición U-Boot.

[41] Ubuntu Wiki. Guía instalación s.o. visionfive 2, 2024. URL https://wiki.ubuntu.com/RISC-V/StarFive%20VisionFive%202.

[42] VisionFive. Visionfive 2 datasheet. Technical report, VisionFive, 2024. URL https://doc-en.rvspace.org/VisionFive2/PDF/VisionFive2_Datasheet.pdf. Datasheet.

Parte IV

Anexos

A. Manual del desarrollador

A continuación se recogen las instrucciones necesarias para evolucionar el sistema. Este manual está dirigido a los desarrolladores que pretenden extender o modificar el entorno desarrollado, con el fin de incorporar nuevas funcionalidades o modificar las ya existentes.

A.1. Introducción

Este sistema ha sido desarrollado con el objetivo de crear un entorno IoT que permita la monitorización de movimientos y condiciones ambientales en tiempo real mediante el uso de sensores, utilizádose además microcontroladores y un middleware basados en RISC-V. Los principales objetivos junto al correcto funcionamiento del sistema en su totalidad incluyen garantizar la fiabilidad y seguridad de las comunicaciones, asegurar la modularidad y escalabilidad del sistema, y ofrecer una interfaz web intuitiva y fácil de usar.

El software abarca la configuración de microcontroladores ESP32-C3, el desarrollo de un middleware en una placa VisionFive 2 utilizando Node-RED, y la implementación de un sistema de comunicación basado en MQTT. El alcance del software abarca desde la recolección de datos de los sensores hasta la presentación de los mismos en una interfaz web.

A.2. Preparación del entorno de trabajo

Para poder desarrollar y evolucionar el software es necesario preparar un entorno de trabajo adecuado que cumpla con los siguientes requisitos:

- **Hardware**:
  - Placa VisionFive 2 o un dispositivo similar compatible con Ubuntu. En su defecto podría usarse un computador convencional o servidor para que actuase como middleware.
  - Tarjeta Micro-SD para almacenar el sistema operativo.
  - Adaptador USB Wi-Fi si la placa elegida no cuenta con un módulo Wi-Fi.
  - Microcontroladores de la familia ESP32.
  - Sensores PIR 501, GY-302 BH1750 y DHT22.
  - Punto de Acceso, extensor de red o router.
  - Cables para poder interconectar los sensores y microcontroladores.

- Un cable USB tipo C para la placa VisionFive 2 y un cable USB micro para cada microcontrolador que se vaya a usar.

- **Software**:
  - Sistema Operativo: imagen preinstalada de Ubuntu Server compatible con la placa a usar.
  - Node-RED con el paquete Dashboard: Para el desarrollo del middleware y la interfaz web.
  - MQTT Broker: Mosquitto u otro broker MQTT similar.
  - Arduino IDE: Para la programación de los microcontroladores ESP32. Deberá de instalarse el paquete que da soporte al microcontrolador que se ha decidido usar, así como los paquetes específicos del proyecto indicados en la sección 5.2 'Elección del software' necesarios para poder compilar los códigos de los microcontroladores.
  - PuTTY u otro software similar que permita realizar conexiones serial.
  - App o software de configuración del extensor Wi-Fi o punto de acceso elegido.

Para preparar el sistema será necesario seguir los siguientes pasos:

1. **Instalar el Sistema Operativo en la placa**: Si se utiliza una placa VisionFive 2 se pueden seguir los pasos explicados en la sección 6.1 'Instalación de Ubuntu en VisionFive 2' de la documentación de este proyecto.
2. **Instalación del software para el middleware**: Será necesario configurar el sistema para instalar correctamente Node-RED, Dashboard, Mosquitto MQTT y realizar otras configuraciones adicionales. En la sección 6.2 'Instalación y configuración del software para VisionFive 2' se realiza una guía de como dejar la placa middleware completamente preparada, incluso si se ha decidido no utilizar como plataforma la VisionFive 2.
3. **Instalar el software de programación de microcontroladores**: Es necesario instalar Arduino IDE, los paquetes requeridos y seleccionar el tipo de placa que se va a programar, especificando el puerto COM al que se va a conectar el microcontrolador a la hora de subir el programa. Los pasos para hacer esto se encuentran en la sección 6.4 'Programación Microcontroladores ESP32-C3'.
4. **Configura el extensor o punto de acceso**: Si se va a utilizar un extensor o punto de acceso para generar una red que solamente puede ser utilizada por los dispositivos del sistema IoT, habrá que configurarla. La mayoría de dispositivos cuentan con software o una aplicación móvil que facilita este proceso. Si se utiliza un extensor Wi-Fi de la marca TP-Link probablemente se pueda seguir los pasos de la sección 6.3 'Red Wi-Fi IoT'.
5. **Configuración y conexiones físicas de los sensores**: Configurar el sensor PIR501 y realizar las interconexiones entre cada microcontrolador y sus sensores según lo indicado en la subsección 5.3.1 'Conexiones entre sensores y microcontroladores'.

6. **Descargar los códigos**: En la carpeta de códigos que acompaña a este documento se encuentran los códigos desarrollados, incluyendo el código final de los microcontroladores, pruebas para probar elementos individuales del sistema y el flujo Node-RED con interfaz.
7. **Importar el flujo final del middleware Node-RED**, localizado en el archivo JSON en la carpeta de códigos.

A.3. Consideraciones generales sobre el desarrollo

El sistema presentado en este proyecto puede ser recreado con tantos microcontroladores como se desee. Sin embargo, para lograr que funcione correctamente será necesario asegurarse que el ID del cliente MQTT de cada microcontrolador sea diferente y que todos ellos tienen correctamente configurada el SSID, password y dirección IP del middleware en base a la red Wi-Fi que se ha decidido usar, así como los tópics MQTT por el que enviarán los datos. En la zona final de la sección 6.4 'Programación Microcontroladores ESP32-C3' se explican todos estos valores que deben ser ajustados. Además, en base al número de microcontroladores totales que se use deberán de añadirse más nodos al flujo de Node-RED para poder recibir los datos a través de los topics de cada microcontrolador.

En caso de que se desee extender o modificar las funcionalidades del sistema base, se realizan las siguientes recomendaciones a modo de guía en base a los cambios que son más probables que se realicen:

- Si se añade alguna **opción adicional al menú de configuración de la interfaz** será necesario crear un nuevo tema por el que se enviará el mensaje de configuración en Node-RED. Además, todos los microcontroladores deberán suscribirse a dicho tema, lo cual se puede lograr añadiendo en la función reconnect() del código la siguiente llamada, siendo NuevoTopicConfig una variable global o constante en la que se almacena el nombre del tema:

  ```
  client.subscribe(NuevoTopicConfig);
  ```

  Tras esto, el comportamiento que deberá tener el microcontrolador al recibir un mensaje de este tipo deberá de ser incluido en la función callback que maneja los mensajes MQTT entrantes. Para ello, se deberá añadir al final de esta función una claúsura condicional if que manejará los mensajes recibidos con este tema:

  ```
  void callback(char* topic, byte* payload, unsigned int length) {
      // Recepción del mensaje
      // Manejar los mensajes con el nuevo topic
      if (String(topic) == NuevoTopicConfig) {
         int data = message.toInt(); // Convertir la cadena recibida a entero
         Serial.print("Datos recibidos: ");
         Serial.println(data);
         //Configuración del nuevo comportamiento del microcontrolador
      }
  }
  ```

Es importante considerar que los mensajes se reciben byte a byte, por lo que estos son almacenados en una cadena de caracteres. En caso de que el cuerpo del mensaje recibido sea un parámetro numérico este deberá ser convertido al tipo adecuado mediante una conversión explícita.

- Si se desea añadir **nuevos tipos de sensores a los microcontroladores** se recomienda realizar un diseño incremental. Es buena idea comenzar creando un programa de pruebas que se pueda usar para comprobar que el sensor en cuestión funciona y ayude a hacerse una idea de como se puede configurar y leer; además de recurrir a la documentación del mismo para saber como conectarlo físicamente al microcontrolador. Se recomienda buscar si ya existe un paquete en Arduino IDE para manejar el sensor elegido, puesto a que esto simplificará en gran medida el proceso de desarrollo.

  Una vez ya esté funcionando, se puede pasar a incorporar el control del nuevo sensor al código final del microcontrolador. En la función setup() se incluirá la configuración de inicio para los pines del microcontrolador a los que esté conectado el nuevo sensor, así como cualquier función de configuración incluida por el paquete instalado.

  Tras esto se podrá pasar a introducir las lecturas sobre los datos. En la mayoría de los casos una sola lectura será suficiente, como es el caso del sensor de humedad-temperatura y de luz de este proyecto, por lo que se podrá incluir en la interrupción síncrona del código, justo antes de enviar los datos por MQTT al middleware. En caso de necesitar usar un comportamiento más específico para el sensor, como ha sido el caso del sensor de movimiento en este proyecto, se recomienda realizar un buen diseño de su comportamiento para asegurar que no se produzcan fallos en su posterior programación.

  Para concluir, será necesario añadir un nuevo tema de datos por el que se enviará las lecturas realizadas por el nuevo sensor al middleware, realizando una llamada a la función de publicación en el cuerpo de la interrupción síncrona. En último lugar, será necesario añadir los nodos necesarios en Node-RED para poder recibir los datos enviados por este nuevo tema y una gráfica adicional en la interfaz para que se puedan mostrar las lecturas realizadas.

- El código de los microcontroladores es compatible con todos los dispositivos de la famila ESP32 (tan solo es necesario elegir en Arduino IDE una placa de desarrollo del modelo que se está utilizando). Si se desea **portear el programa** para que funcione con otra familia de microcontroladores será necesario asegurar que esta satisface todos los requisitos técnicos del mismo (módulo Wi-Fi, timer, interfaz I2C...), además de que en el programa se deberá modificar la configuración y uso del timer para las interrupciones síncronas, ya que se usa una librería específicamente creada para la familia ESP32.

En caso de que se desee extender o modificar el sistema de cualquier otra manera se recomienda realizar una lectura de las secciones 5.3.2 'Conexión de microcontroladores y middleware', 5.4 'Diseño de la programación de microcontroladores' y 5.5 'Diseño del middleware Node-RED e Interfaz Web' para comprender el comportamiento de

los diferentes componentes del sistema y como se comunican entre sí; la sección 6.4 'Programación de microcontroladores ESP32-C3' y 6.5 'Programación Node-RED' en la que se explica el código desarrollado y las pruebas unitarias programadas, así como el flujo Node-RED y la interfaz creados; y el código final de los microcontroladores y flujo, que están debidamente comentados y dan una visión más específica de donde se realiza cada acción y por qué.

Además, se recomienda el uso de mensajes de depuración a través de la terminal serial incluida en Arduino IDE, ya que facilitará la comprobación del correcto funcionamiento del código y la depuración de fallos. Otra buena práctica es el uso de constantes o variables globales definidas al principio del código, donde se podrá almacenar valores y cadenas que contendrán información como direcciones IPs o topics usados en el código, lo que permitirá poder modificarlas en caso de ser necesario de una manera sencilla y controlada.

A.4. Instrucciones para construcción

A continuación se detalla la secuencia de pasos necesarios para construir y depurar el sistema:

1. Asegurar que todo el entorno de desarrollo está correctamente instalado y configurado, comprobando que se han cumplido todos los pasos descritos en la sección A.2 'Preparación del entorno de trabajo'.
2. Lanzar el flujo Node-RED en el middleware.
3. Compilar y cargar el código desarrollado en Arduino IDE en cada uno de los microcontroladores, asegurándose de que las constantes globales tienen los valores correctos.
4. Comprobar que la red Wi-Fi usada por el entorno IoT está levantada y es accesible.
5. Conectar los microcontroladores y la placa VisionFive 2 a la corriente eléctrica. Esto se puede hacer utilizando cargadores de dispositivos móviles convencionales. Gracias a la configuración introducida en la placa VisionFive 2 todos los servicios necesarios, incluidos el broker MQTT y flujo de Node-RED, se iniciarán automáticamente tras cargarse el sistema.
6. Visualizar los resultados usando la interfaz web. Accediendo a la página donde se aloja la interfaz web se pueden ver las gráficas que muestran los datos leidos por los sensores y se puede probar las opciones de configuración implementadas.

Para depurar el sistema se podrá usar tanto la Interfaz Web como los mensajes de depuración enviados por los microcontroladores a través del monitor Serial de Arduino IDE. Estos últimos permitirán identificar la zona exacta del código donde el comportamiento no es el esperado. En caso de detectar algún error, es una buena práctica utilizar los códigos de pruebas unitarios desarrollados para el elemento que ocasiona fallo, permitiendo de esta manera identificar si es un problema del sensor, microcontrolador o comunicación de los datos al middleware.

B. Manual de usuario

Las instrucciones de uso del sistema se detallan a continuación. Este manual se dirige al usuario final del sistema objetivo de este proyecto.

B.1. Introducción

El sistema desarrollado es una solución IoT diseñada para la monitorización del comportamiento de los alumnos y condiciones ambientales de un aula de educación en tiempo real. Este sistema ha sido desarrollado con las siguientes características en mente de cara a su uso por parte del usuario final:

- Facilitar la recolección y visualización de datos usando sensores conectados a microcontroladores.
- Proporcionar una interfaz web sencilla e intuitiva para visualizar y configurar la recolección de estos datos.
- Transparencia del entorno IoT subyacente, tanto para el profesor como para los alumnos que están siendo monitoreados.

Este sistema está pensado para su uso en entornos educativos por parte de un profesor, actuando como una herramienta que les permita monitorear en tiempo real el comportamiento de sus alumnos (cantidad de movimiento realizado) y las condiciones ambientales del aula educativa (luz, temperatura y humedad), con la idea de mejorar la dinámica y efectividad del proceso de enseñanza. Este sistema ayudará al docente a identificar a aquellos alumnos que no están prestando atención, de manera que podrá modificar durante la propia lección los contenidos o enfoque de la misma para tratar de volver a captar su interés.

El alcance del sistema desarrollado incluye la configuración de sensores, la recolección de datos en tiempo real y la visualización de estos a través de una interfaz con panel de control accesible desde cualquier navegador web.

B.2. Instalación

Esta guía considera que el sistema ya ha sido debidamente configurado e implementado por parte de algún miembro cualificado del personal. En caso de no ser así, por favor seguir los pasos establecidos en el Manual del Desarrollador para saber como preparar y lanzar todos los componentes del sistema.

A continuación se detallan los diferentes pasos que se deben seguir para iniciar el sistema:

- Situar los microntroladores en las zonas que se desean monitorizar. Se pueden colocar las palas de cartón-piedra en los velcros laterales de las cajas de sensores para reducir el ángulo de visión del sensor de movimiento.
- Encender el punto de acceso o extensor de red Wi-Fi que se va a utilizar si aún no se ha hecho.
- Conectar a la corriente eléctrica utilizando cargadores de teléfono móvil tanto la placa VisionFive 2 como los microcontroladores. Es necesario esperar un par de minutos para que los sensores se calibren y la placa arranque todos los servicios automáticamente
- Acceder a la página web en la que se aloja la Interfaz del sistema. Esta puede ser accedida desde un navegador web de cualquier dispositivo, siempre y cuando dicho dispositivo esté conectado a la misma red que el sistema IoT. La dirección de la página web es la siguiente:

  `http://DIRECCION_VISIONFIVE:1880/ui`

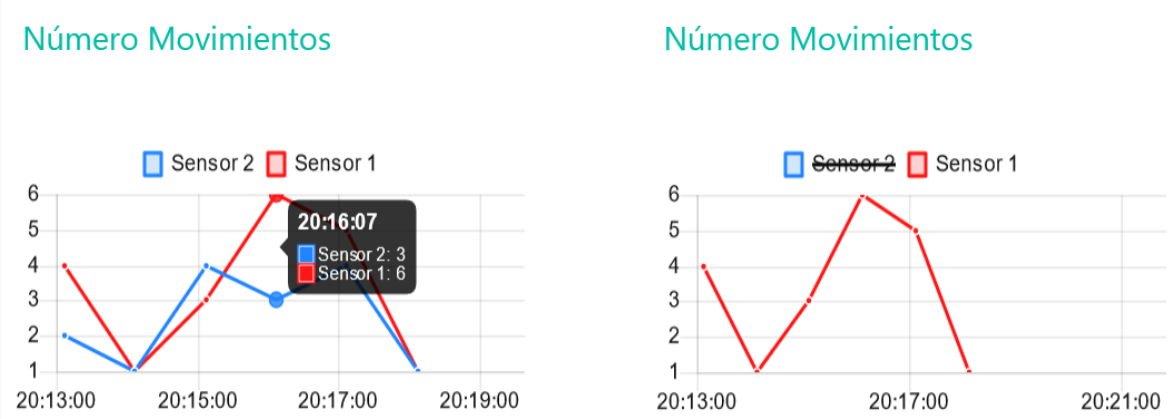
  Donde `'DIRECCION_VISIONFIVE'` se sustituirá por la dirección IP de la placa VisionFive 2 en la red IoT. Si no conoce el valor de dicha dirección, consúltelo con el miembro del personal que configuró el sistema o acceda a la placa VisionFive 2 e introduzca el comando '`ip a`' para obtener la dirección IP de la interfaz de red que está usando la placa:

B.3. Uso del sistema

El profesor únicamente interactuará con el sistema a través de la interfaz. Esta estará compuesta por dos ventanas diferentes:

- **Ventana principal**, en la que se mostrarán las gráficas con todos los datos recopilados. Habrá una gráfica para cada uno de las tipos de magnitudes leídas por los sensores y en cada una de ellas se mostrará una línea por cada microcontrolador que ha enviado datos, mostrándose de esta manera la evolución que cada parámetro ha tenido en el transcurso de la última hora. Se podrá ver el valor exacto de cada lectura posicionando el cursor del ratón sobre los puntos que conforman las líneas (Figura B.1a), así como ocultar las lecturas de aquellos sensores que no se quieren ver pulsando en el nombre que aparece justo arriba de la gráfica en cuestión (Figura B.1b).

  Además de las cuatro gráficas para las lecturas de movimiento, temperatura, humedad y luz se incluye una quinta gráfica, en la que se mostrarán las aperturas/cierres de puertas y ventanas, puesto a que son un factor a tener en cuenta a la hora de evaluar algunos parámetros como la temperatura o la humedad.
- **Ventana de configuración**, en la que se podrá vaciar los datos de todas las gráficas, marcar cuando una puerta/ventana del aula ha sido abierta/cerrada y establecer el tiempo de espera entre dos lecturas consecutivas de los sensores. En este último, el valor mínimo y que usa por defecto el sistema es de 1 minuto,



(a) Mostrar información sobre las lecturas

(b) Ocultar sensores

Figura B.1: Manejo de gráficas en la interfaz

aunque se recomienda usar valores de entre 2 y 5 minutos para obtener mejores resultados.

Los alumnos interactuarán de manera pasiva con el sistema, por lo que no necesitarán hacer nada más que prestar atención a la lección que se está impartiendo. Es recomendable no tocar ni mover los sensores mientras se están usando, para evitar desconectar por accidente algún cable o descalibrar el sensor de movimiento.