UNIVERSITAT DE BARCELONA

Trabajo de final de grado

GRADO DE INFORMÁTICA

Facultat de Matemàtiques i Informàtica
Universitat de Barcelona

Control ambiental distribuido basado en redes de sensores inalámbricas

Autor: Daniel Yeste López

Director: Dr. Joan Daniel Prades
Realizado en: Departamento de Ingeniería Electrónica y Biomédica

Barcelona, 13 de junio de 2023

Resumen

Con el auge del internet de las cosas, cada vez es más sencilla la interconexión entre dispositivos por tal de poder obtener información y monitorizar de manera no invasiva entornos donde las personas realizan sus actividades cotidianas. Poder orientar el uso de estas tecnologías a algo que tenga un impacto positivo en la vida de las personas está al alcance de todos, y se debería tener conciencia de que el coste y la dificultad para crear estos sistemas es cada vez menor.
En este proyecto, se ha pretendido crear un sistema de control ambiental no invasivo. La creciente preocupación por la polución del aire y su efecto dañino en las personas es cada vez mayor y, aunque como individuos no tenemos la capacidad de solucionar esta problemática, sí podemos tomar pequeñas acciones para mejorar nuestra calidad de vida.
La aportación de este trabajo ha sido la de crear un sistema interconectado entre módulos de sensores, microcontroladores, computadores de placa única y servicios en la nube para monitorizar y obtener los datos de control ambiental que nos permitan tomar medidas en los entornos en los que se ha implementado. Para llevar a cabo esta tarea, se explorarán en las siguientes páginas los componentes de hardware usados, su implementación en software y las técnicas y protocolos para la comunicación entre todos los componentes. Finalmente, se mostrarán los datos obtenidos para ofrecer unas conclusiones y que el lector pueda sacar las suyas propias.

Resum

Amb l'auge de l'Internet de les coses, cada cop és més senzilla la interconnexió entre dispositius per poder obtenir informació i monitoritzar de manera no invasiva entorns on les persones realitzen les seves activitats quotidianes. Poder orientar l'ús d'aquestes tecnologies cap a alguna cosa que tingui un impacte positiu en la vida de les persones és a l'abast de tots, i s'hauria de ser conscient que el cost i la dificultat per crear aquests sistemes és cada cop menor.
En aquest projecte, s'ha creat un sistema de control ambiental no invasiu. La creixent preocupació per la contaminació de l'aire i el seu efecte perjudicial en les persones és cada cop més gran, i tot i que com a individus no tenim la capacitat de solucionar aquesta problemàtica, sí que podem prendre petites accions per millorar la nostra qualitat de vida.
La contribució d'aquest treball ha estat crear un sistema interconnectat entre

mòduls de sensors, microcontroladors, computadors de placa única i serveis al núvol per monitoritzar i obtenir les dades de control ambiental que ens permetin prendre mesures en els entorns on s'ha implementat. Per dur a terme aquesta tasca, a les següents pàgines es mostraràn el hardware utilitzat, la seva implementació en software i les tècniques i protocols per a la comunicació entre tot el conjunt.
Finalment, es mostraran les dades obtingudes per oferir unes conclusions i perquè el lector pugui treure les seves.

Abstract

With the popular and investment rise of Internet of Things, interconnecting devices to obtain information and monitor environments where people carry out their daily activities is becoming increasingly easier. Directing the use of these technologies towards something that has a positive impact on people's lives is within everyone's reach. The cost and difficulty of creating such systems are decreasing and that makes this systems more accessible for the general public.
This project aims to create a non-invasive environmental control system. The growing concern about air pollution and its harmful effects on individuals is increasing, and although we, as individuals, may not have the capacity to solve this issue, we can take small actions to improve our quality of life.
This project is focused on creating an interconnected system involving sensor modules, microcontrollers, single-board computers, and cloud services to monitor and obtain environmental data. To accomplish this task, the following pages will show the hardware components that have been used, their software implementation, and the techniques and protocols for the communication from one hardware piece to each other.

Finally, the obtained data will be presented to provide conclusions and allow the reader to draw their own.

Agradecimientos

Al Dr. J.Daniel y el Dr.Manel López, por proponerme este interesante proyecto y darme la tutorización y las herramientas necesarias para llevarlo a cabo, sin ellos habría sido imposible. A mi familia, que me ha apoyado y ayudado cuando lo he necesitado durante toda mi etapa universitaria.

Índice

1. Introducción

El IoT(Internet of things) o Internet de las Cosas es un término no estandarizado pero cada vez más utilizado en la industria. Este término hace referencia al conjunto de dispositivos físicos, generalmente pequeños y no invasivos para los humanos, que están interconectados con el objetivo de enviar y recibir datos. Como podemos apreciar hoy en día, la tecnología está involucrada de una u otra manera en el mundo en el que vivimos. Para 2030, se espera que el número de dispositivos IoT sea de 24 billones en toda la tierra. Esto significaría que, distribuidos de manera equitativa, cada persona tendrá a su alcance tres dispositivos los cuales obtendrán información, se comunicarán con la red y nos acompañaran durante el día a día. Conociendo estas métricas, no parece inusual que se esté apostando de manera voraz por el desarrollo y creación de más y mejores dispositivos de esta índole. Con la capacidad de ensamblado actual, las grandes tecnológicas cada día son capaces de crear dispositivos más pequeños y sutiles que permiten una portabilidad así como un diseño con una alta sencillez.
Durante mi estancia en la universidad, comencé a trabajar con arquitecturas embebidas y encontré una motivación al ver cómo mis desarrollos podían tener un impacto en el mundo físico, más allá de una interfaz gráfica. Esta experiencia me llevó a dedicarme profesionalmente a este campo, ya que pude apreciar cómo las cosas con las que interactuamos día a día podían programarse y aplicarse en situaciones cotidianas, permitiendo a los usuarios interactuar físicamente con ellas en su entorno diario. Por ello, decidí que debería enfocar este trabajo hacia ese campo, intentando buscar una aplicación real que pudiera ayudar a las personas.
Basándome en esto, mi tutor me propuso iniciar un proyecto de control ambiental orientado a una implementación real en la facultad. Tras el impacto de la pandemia, resurgió una preocupación acerca de como tener buenos hábitos para mantener una buena calidad del aire en los espacios cerrados. Se dio importancia a como pequeños actos podían influir en la calidad de vida de las personas, como una ventilación adecuada en estos espacios o un control de la temperatura para poder trabajar con comodidad y reducir el gasto energético.
Según estudios del Imperial College of London, la contaminación originada en las ciudades está potencialmente vinculada con la hiperactividad y falta de atención. El impacto que tiene la polución del aire de manera cotidiana no es un foco mediático, a pesar de que, realmente, diversos estudios muestran una preocupación constante su influencia en nuestra calidad de vida.
Con estas motivaciones en mente, decidí emplear este proyecto para poder

crear un sistema que me permitiera poder ayudar y facilitar las vidas de los alumnos de la Universitat de Barcelona. Convertir las aulas en lugares de trabajo confortables, así como velar por la salud de los alumnos es prioritario para un paso en la universidad exitoso y enriquecedor.

2. Objetivos

Tras las primeras tomas de contacto con la tutoría, se establecieron los objetivos principales. Para este proyecto, se requería una monitorización no invasiva de las aulas de la universidad. Los datos esenciales a recopilar en esta monitorización eran el dióxido de carbono y los compuestos orgánicos volátiles. Además, en caso de ser posible, se buscaba recopilar otros datos adicionales relacionados con el confort de las personas, para evaluar diferentes condiciones.
Los objetivos de este proyecto eran los siguientes:

- Integrar un sistema que pudiera hacer lecturas en tiempo real de la calidad del aire y otros datos relevantes.
- Medir estos datos por tal de poder cuantificar los valores.
- Almacenar los datos de manera persistente.
- Analizar los datos y obtener conclusiones.

3. Planificación

A pesar de las indicaciones iniciales para dividir la carga de trabajo entre documentación (cuatro semanas) y desarrollo (siete semanas) se decidió mezclar estas dos etapas por tal de trabajar en sprints de dos semanas, según la metodología SCRUM. Para lograrlo, se establecieron objetivos de desarrollo que se desglosaron en las siguientes tareas que ocuparían un sprint cada una:

- Instalación de la Raspberry Pi y creación de una red inalámbrica mediante esta.
- Conexión del ES8266 a la red inalámbrica y comunicación mediante MQTT.
- Montaje de los módulos de los sensores y lectura de estos en el ESP8266.

- Almacenar datos en la nube mediante Raspberry Pi.
- Pruebas sobre los datos recibidos e instalación.

Cada uno de los sprints se pudo completar con éxito, dando espacio para una tarea añadida, la creación de una aplicación móvil con la que poder visualizar los datos de la nube de una forma sencilla.
El resto del tiempo recomendado, se usó para la redacción de la memoria.

4. Ingeniería de concepción

Debido a la naturaleza del proyecto, se propuso dividir el desarrollo en dos apartados; ingeniería de concepción e ingeniería de detalle. En este primer apartado se propone discutir los protocolos de comunicación a utilizar, así como la disposición y elección de los módulos de sensores y los microcontroladores utilizados.

4.1. Hardware

4.1.1. ESP8266

A nivel de hardware, la base del proyecto se concentra en la responsabilidad del ESP8266. Este es un microcontrolador de bajo costo y bajo consumo de energía que se ha vuelto popular en el ámbito de Internet de las Cosas (IoT). Es una solución versátil y eficiente para proyectos que requieren comunicación y extracción de datos.
El ESP8266 destaca por su capacidad de procesamiento y conectividad, lo que lo hace adecuado para nuestro proyecto. Es fácilmente programable y cuenta con conectividad Wi-Fi y Bluetooth integrada permitiéndole conectarse a redes y comunicarse con otros dispositivos.
Como se puede observar en la figura uno su encapsulación facilita la integración con otros componentes, posibilitando crear un sistema que funcione de manera conjunta con otros elementos. Podemos programar el ESP8266 utilizando el entorno de desarrollo Arduino donde diversas bibliotecas disponibles simplifican la comunicación con este dispositivo. Esto hace que la programación sea sencilla y eficaz para los desarrolladores.
La elección del dispositivo, finalmente, se debe a que otros competidores en el mercado carecen de una capacidad mucho mayor y a la par necesaria para

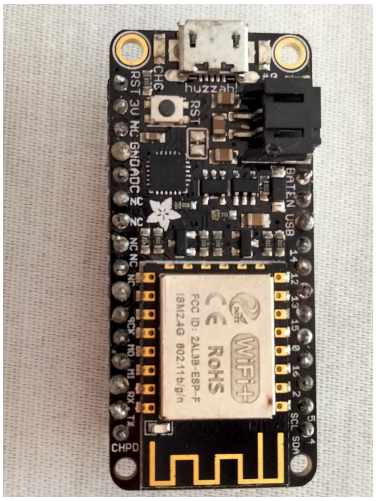


Figura 1: ESP8266

este proyecto. Siendo tanto el tamaño, memoria, su velocidad de procesamiento y conectividad por defecto los que lo hacen idóneo.

Conexión

La disponibilidad de pines se divide en cinco para la alimentación, dos pines seriales, tres pines para comunicación SPI y diez pines GPIO (General PIN Input Output). Concretamente, para este proyecto, únicamente serán necesarios cuatro pines de los disponibles: dos necesarios para la alimentación y otro par para la comunicación I2C. En la figura dos se puede observar la distribución de los pines del microcontrolador.
Para establecer la comunicación con el resto de dispositivos, se podría haber establecido cualquiera de los pines de GPIO pero, por recomendación del fabricante, el trabajo con arduino funciona por defecto mediante los pines SDA y SCL, siendo estos el Serial Data Line y Serial Clock Line, necesarios para la sincronización y el envío de datos. En los microcontroladores, es común el uso de estas conexiones para establecer comunicación mediante I2C.
La Serial Data Line funciona como vía bidireccional para la transmisión de datos entre los dispositivos conectados en un bus I2C, permitiendo tanto la transmisión como la recepción de información. Esta es la única línea de recepción de datos en este protocolo y, al ser full-duplex, es decir, con comunicación bidireccional, es necesario controlar adecuadamente la transmisión por tal de asegurar su funcionamiento y evitar las colisiones. Por otro lado,

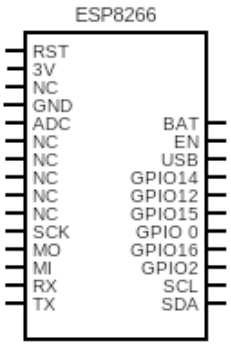


Figura 2: Diagrama de pines del ESP8266

el Serial Clock Line se utiliza para sincronizar la transmisión de datos en el bus. A bajo nivel es crucial que los múltiples dispositivos interconectados tengan acceso a este reloj por tal de sincronizar las operaciones de lectura y escritura.

4.1.2. AmbiMate

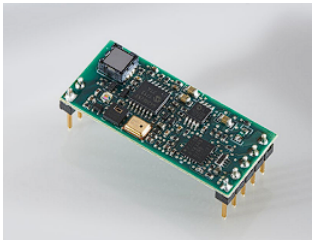


Figura 3: Módulo Ambimate

Una de las tareas de este proyecto era tener control acerca de cuantos más datos ambientales fuera posible para poder obtener información valiosa del sistema. Por ello, se me proporcionó el dispositivo AmbiMate(figura tres), escogiendo esta opción por su polivalencia debido a que integra diferentes sensores capaces de monitorizar la temperatura, humedad relativa, luz y movimiento.
Las especificaciones del producto respecto a sus sensores, así como la distribución en la placa se muestran en la figura cuatro. Se puede observar que hay ciertos componentes no mostrados en la tabla de la figura cuatro. Esto se debe a que existe una variedad de productos que ofrecen un rango más am-

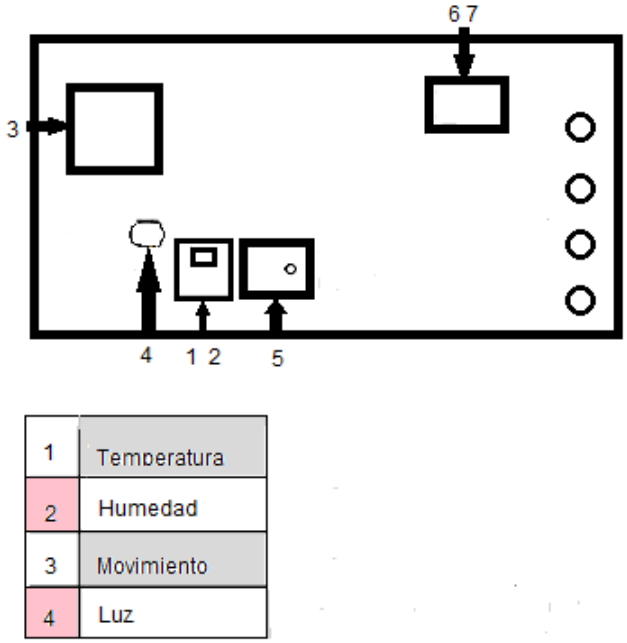


Figura 4: Especificaciones del sensor AmbiMate

plio de sensores integrados, pero, para nuestro caso, únicamente tendremos los nombrados anteriormente. Tras los primeros pasos de aprendizaje con el dispositivo se pudieron hacer pruebas exitosas con todos los sensores disponibles, contrastando los datos con instrumentos analógicos como termómetros de mercurio y digitales como los datos en tiempo real que ofrecía un dispositivo de una famosa compañía de seguridad en el lugar de las pruebas.

Conexión

El AmbiMate ejerce la labor de esclavo en el sistema creado. Para ello, el sensor de ambiente dispone de los pines necesarios para poder ejercer su tarea. Además, contaremos con los pines de alimentación para recibir la tensión y la intensidad necesarias desde nuestra ESP8266. En la figura cinco observamos cual es la distribución de estos pines.

Una vez se haya realizado la conexión con el maestro, se puede proceder a la comunicación mediante I2C. Para iniciar la conexión, deberemos hacer una petición a la dirección *0x2A*, tras esto indicaremos con diferentes valores en

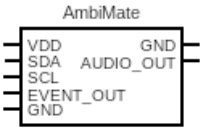


Figura 5: Pines del sensor AmbiMate

hexadecimal los datos que queremos obtener por parte del Ambimate. Estos valores hexadecimales indican una cadena de bits que, dependiendo del valor que transmitamos, realizaremos una petición diferente sobre las lecturas que queremos en ese momento. Los datos recibidos estarán sin procesar de manera que será el programador el que deba adaptarlos para que los valores sean identificables.
Finalmente, los únicos datos que extraeremos de los sensores serán los de temperatura y humedad. Por razones logísticas, se decidió obviar los sensores de luz y movimiento. Debido a la ubicación de estos sensores, los datos podían ser erróneos o confusos para los usuarios finales que consultaran los datos. Así mismo, por pragmatismo, el sensor de sonido parecía no aportar nada necesario ya que, en caso de la contaminación acústica, es sencillamente detectable y combatible en el entorno final.

Funcionamiento de los sensores

Para saber con exactitud como hacer un programa para comunicarnos con un AmbiMate, se indagó acerca de cuales eran las lecturas de los sensores y su capacidad. Como se ha comentado en el apartado anterior, los únicos sensores que finalmente se usarán para la recogida de datos serán el de humedad y temperatura, así que se dará una breve explicación de sus características.
Por un lado, el sensor de temperatura tiene una resolución de catorce bits para la medición de temperatura. Esta es la cantidad de valores discretos que el dispositivo puede representar en la escala de temperatura. Su rango de operación para las temperaturas comprende entre menos cuarenta grados bajo cero, medidos en Celsius, hasta los ciento veinticinco, asegurándonos que cualquier dato de temperatura podrá ser captado por el sensor en el entorno de pruebas e implementación. También, sabemos con seguridad que su capacidad de respuesta es corta, alrededor de cuarenta milisegundos, asegurando que podremos obtener los datos en tiempo real, desestimando cualquier dato erróneo en función del tiempo.
Por otro lado, el sensor de humedad tiene una resolución de doce bits, pu-

diendo operar en un rango de humedad relativa que va desde un veinte por ciento hasta un ochenta por ciento. Asegura un tiempo de respuesta corto y capacidad de correción de la humedad relativa entre un cero y ochenta grados celsius, asegurando una corrección de los datos en caso de que la medición de la humedad varie por los cambios de temperatura.
Estos datos nos aseguran que podremos trabajar con los sensores de manera adecuada en los entornos destinados y que se cumplirán los requisitos de rápidez a la hora de tomar datos por tal de que se puedan obtener a tiempo real.

4.1.3. CCS811

Dada la naturaleza inicial del proyecto, se debía considerar como tomar los datos de dióxido de carbono(Co2) y de compuestos orgánicos volátiles(COV) ya que son indicadores esenciales para reconocer si el entorno es saludable para las personas que se encuentran en él mediante la medición de la calidad del aire. El CCS811 que se muestra en la figura seis nos permite tomar mediciones de estos dos valores siendo un dispositivo únicamente dedicado a esta tarea.

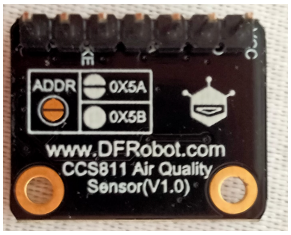


Figura 6: CCS811

Entre algunas de sus características, se encuentran su bajo consumo energético, capacidad de entrar en toma de datos correctos de manera rápida y su pequeño tamaño para poder conectar y ubicar junto con un microcontrolador. Para garantizar un funcionamiento óptimo, es necesario realizar una configuración inicial que permita obtener valores lo más precisos posibles. Este proceso se puede realizar de dos maneras, automática o manual. Si dejamos que el propio dispositivo trabaje sin referencia inicial, este usará un algoritmo interno donde, en cada toma de datos buscará cual es la toma de aire más limpia e ira calibrándose en torno a este. La calibración automática no es recomendable ya que nos puede llevar a obtener resultados poco precisos muy dependientes del entorno en el que se ubique.

La forma más óptima de configuración consiste en establecer una referencia inicial en condiciones adecuadas a las cuales el dispositivo pueda hacer referencia al obtener datos. Para ello, es importante ubicar el sensor CCS811 en un área bien ventilada con corrientes de aire constantes para establecer un punto de referencia de aire limpio. Después de aproximadamente veinte minutos, se obtendrá un valor estable. Este valor es único para cada módulo y debe configurarse individualmente para cada uno con el que se trabaje. De manera redundante, se enviará este valor al CCS811 después de cada lectura para garantizar que las lecturas sigan la referencia establecida.

Conexión

Como otros dispositivos que se comunican mediante el protocolo I2C, se observa en la figura siete los usuales pines de SCL y SDA para las conexiones donde, en nuestro caso, este dispositivo ocupará el rol de esclavo.

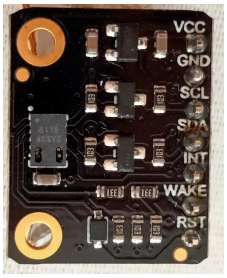


Figura 7: Pines del CCS811

Para el intercambio de datos, se sigue la misma metodología que se mencionó anteriormente, pero en esta ocasión se utilizará una librería proporcionada por el fabricante para facilitar la comunicación. De esta manera, aprovecharemos las funciones provistas por el fabricante para la transmisión, las cuales enmascaran las cadenas de bits enviadas y, al mismo tiempo, permiten obtener los datos de manera legible para el usuario final.

Referencia inicial

La referencia inicial para los sensores fue tomada en Esplugues de Llobregat, una ciudad situada a la periferia de Barcelona. Según los datos del Ministerio para la transición ecológica y el reto demográfico, esta ciudad tiene, anualmente, peores resultados en cuanto a calidad del aire se refiere.

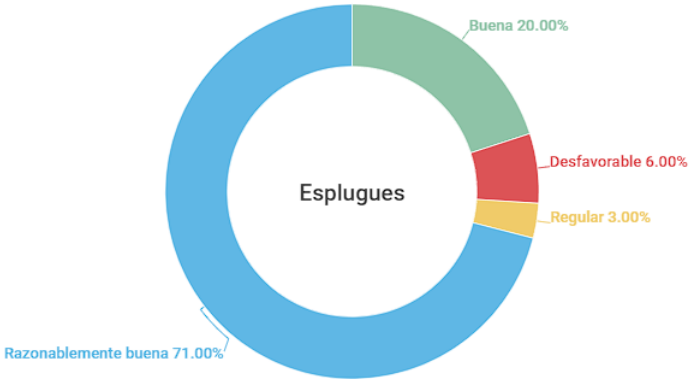


Figura 8: Gráfico que representa la calidad del aire en Esplugues de Llobregat durante un año

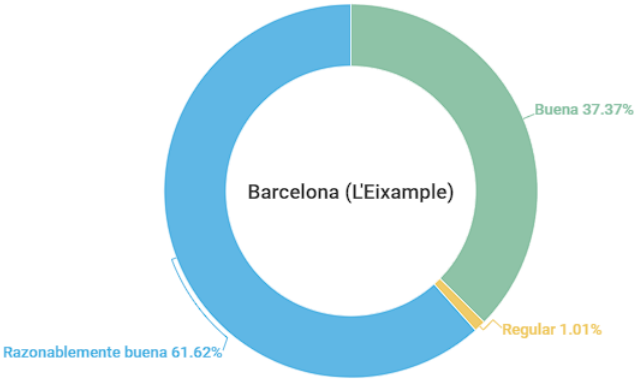


Figura 9: Gráfico que representa la calidad del aire en Barcelona durante un año

Los datos mostrados en las figuras ocho y nueve hacen referencia al periodo que comprende entre el veintitrés de mayo de dos mil veintidós hasta el año siguiente. Dado que se dará una interpretación de los datos a posteriori, se hará una revisión final acerca de si la referencia inicial a afectado a la toma de datos y si ha tenido impacto. Por motivos logísticos, la toma de las referencias base no se pudo realizar en el entorno de implementación, la Universitat de Barcelona.

Interpretación de los datos del CCS811

La medida de Co2 que registra el sensor se transmite como partes por millón (ppm). Esta unidad de medida mide la concentración, en este caso de contaminantes del aire. Aunque el fabricante no específica como el dispositivo lo mide, esta medida se refiere a partes de vapor o gas por cada millón de partes de aire contaminado. Para paliar esta falta de información, se nos provee con tablas para poder entender que rangos de Co2 pueden ser nocivos para las personas.

Dióxido de Carbono (PPM)	Efecto en los Humanos
< 500	Normal
500-1000	Un poco incómodo
1000-2500	Cansado
2500-5000	Insalubre

Por otra parte, tenemos también la medida de concentración de compuestos orgánicos volátiles (COV). Dado que no todos los compuestos orgánicos son nocivos para el humano, este sensor sólo tendrá en cuenta los que sí pueden serlo. De nuevo, el fabricante no proporciona ninguna referencia a cómo se miden estos datos y, por ello, nos proporciona tablas con información sobre los rangos de COV y cómo nos afectan.

Concentración TVOC (PPB)	Efecto en los Humanos
< 50	Normal
50-750	Ansiedad, incomodidad
750-6000	Depresivo, dolores de cabeza
> 6000	Dolor de cabeza y otros problemas nerviosos

Rangos habituales en el entorno de prueba

Por defecto, este dispositivo partirá de la base de que la pureza del aire respecto al dióxido de carbono es de cuatrocientos ppm. Esto se debe a que esta es la marca habitual y registrada en el planeta Tierra en la actualidad. Antes de la era industrial, estos valores se situaban en torno a los trescientos sesenta ppm, pero han aumentado significativamente desde entonces.

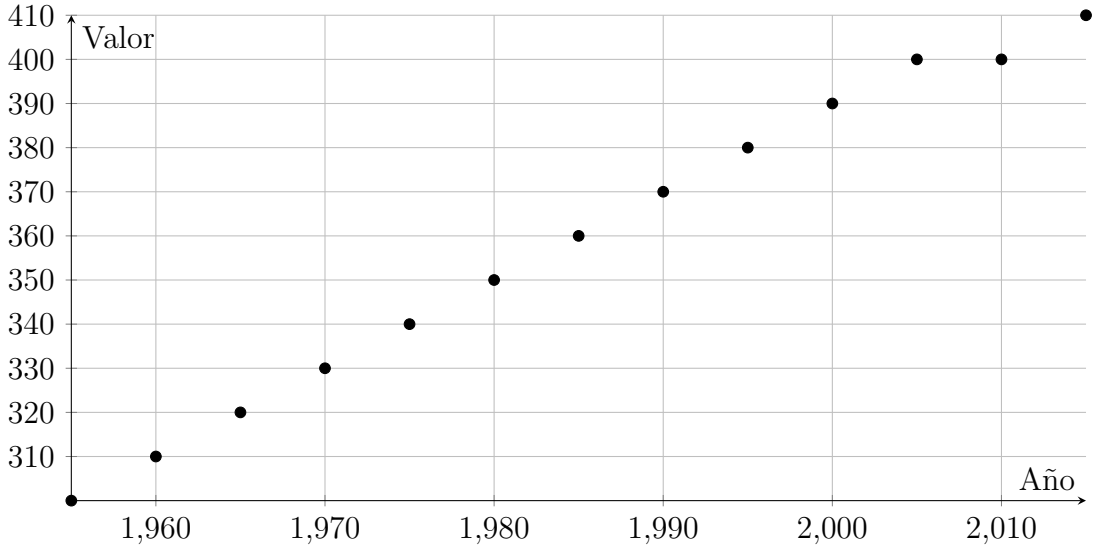


Figura 10: Variación de los niveles de dióxido de carbono a lo largo del tiempo

Teniendo en cuenta esta información mostrada en la figura diez, todos los datos que veremos en los próximos apartados sobre el dióxido de carbono se basan en estos valores actuales, tomando como referencia un lugar con alta ventilación ubicado en Esplugues de Llobregat.

4.1.4. Raspberry Pi

Este es el hardware que nos habilita todo el proceso de comunicación y que nos permite poder realizar tareas con un coste computacional y de memoria más elevadas de lo que nos permitiría nuestro microcontrolador. La Raspberry Pi, mostrada en la figura once, es un computador de placa única bastante popular y ampliamente usado para el desarrollo de soluciones tecnológicas. Concretamente, es un computador ideal para proyectos como automatización del hogar, servidor doméstico, IoT o robótica. Cuenta con HDMI, Ethernet y USB y su sistema operativo es para nuestro caso, Linux, lo que hace que, en resumen, sea un computador con menor capacidad que la media del mercado pero con una portabilidad mayor.
Para que nuestro sistema funcionara, era necesario que las ESP8266 pudieran tener acceso a una red privada y para transmitir la información. La Raspberry Pi tiene capacidad para crear redes inalámbricas propias a partir de una conexión de red, supliendo esta necesidad. A partir de esto, también cumple una función fundamental en la conexión y tratamiento de datos recibidos

Figura 11: Imagen de Raspberry Pi por Jwrodgers, licenciada bajo Creative Commons CC BY-SA 3.0

mediante el protocolo MQTT. Esta Raspberry actuará también como broker en el protocolo, siendo el eje central de la comunicación para finalmente, poder exportar los datos a un servidor en la nube.
En este proyecto es esencial el uso de un computador debido a que se necesita un nexo de unión con los microcontroladores por tal de recibir, tratar, y finalmente subir los datos a la nube. Por ofrecernos esta capacidad y a su vez, una portabilidad que no puede ofrecer una computadora convencional, se escogió para que formase parte del sistema.

4.2. Protocolos de comunicación

4.2.1. MQTT

La idea principal cuando busqué un protocolo de comunicación para que el hardware se comunicara entre sí era encontrar un protocolo escalable, con alta independencia entre dispositivos y ligero. Con estas características, el protocolo más adecuado fue MQTT, un protocolo de comunicación muy popular en la industria de IoT debido a que cumple con los requisitos anteriormente mencionados. MQTT es un protocolo de red ligero basado en la publicación y suscripción por parte de los dispositivos que se comunican. Este servicio debe funcionar sobre un protocolo de transporte que permita transportar datos sin pérdidas, de manera bidireccional y ordenada. Usualmente se usa

el protocolo TCP/IP debido a su robustez y estandarización en los sistemas de comunicación.
Este tipo de comunicación está basada en dos entidades llamadas broker y cliente. Un broker es la entidad central de software en la arquitectura MQTT. Sus tareas, conceptualmente, son similares a las de un broker o corredor de bolsa ya que, tras evaluar a todos los posibles clientes de esta comunicación y comprobar que cumplen con las reglas de comunicación, el broker inicia la transacción demandada. Por tanto, el broker es la unidad central de esta comunicación, lo cual permite que los clientes (dispositivos conectados al broker) tengan escasas responsabilidades en la comunicación y, por tanto, el coste computacional y los recursos destinados a esta comunicación sean mínimos para ellos.
Las funciones del broker, además de orquestar las comunicaciones, es la autentificación de los dispositivos y la verificación de que estos se pueden comunicar de manera segura gracias a la encriptación que se da en la capa de seguridad de transporte. Podríamos observar, entonces, que además de las características nombradas previamente, este protocolo también contempla que las comunicaciones sean seguras mediante la encriptación de estas. Para mantener la cohesión en la comunicación, susodicho broker mantendrá una dirección fija desde la inicialización de este hasta que se finalicen las comunicaciones, ayudando así a poder ubicar mediante una dirección IP al broker de manera sencilla. En ningún momento los clientes conectados podrán ver el tráfico de otros dispositivos ni dependerán de estos, dando una alta escalabilidad a este protocolo.

Metodología de comunicación empleada

La metodología que se ha usado para este proyecto es la de publicación/suscripción. Este patrón de mensajería esta basado en que existirá un agente suscriptor y otro publicador. Estos dos agentes serán dispositivos y, por tanto, clientes del broker, cada uno cumpliendo con un rol en la comunicación. Como su nombre indica, el agente publicador podrá enviar un mensaje al broker y, una vez evaluado, todos los agentes suscriptores serán identificados por este y se les hará llegar el mensaje publicado.
Mediante la utilización de la librería de código abierto Mosquitto, el envío de mensajes se hará mediante tópicos. Esto significará que, al publicar o suscribirse a mensajes, se tendrá que especificar un tópico o grupo al cual pertenecerán los mensajes. Este sistema se denomina servicio de mensajería puro.

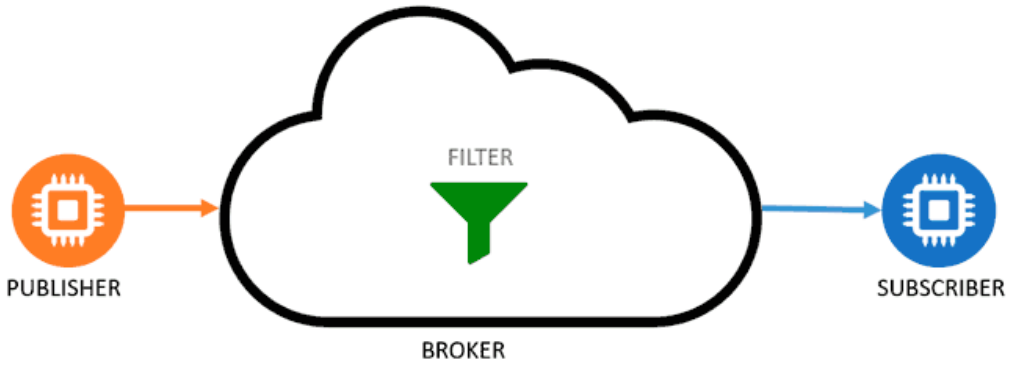


Figura 12: Infograma del servicio de mensajería

Como se puede observar en el infograma de la figura doce, el broker analizará los mensajes y a qué tópico pertenecen para poder distribuirlos entre publicadores y suscriptores. Este servicio establece que, por defecto, los mensajes entregados a un cliente desconectado se perderán. Aún así, se puede implementar la persistencia de datos, por ejemplo, guardando los mensajes no entregados en una base de datos o similares.

Operación de conexión

Por defecto, el broker habilitará el puerto 883 para la conexión. Cuando el dispositivo cliente esté listo este enviará un mensaje CONNECT con la información necesaria a susodicho puerto. En caso de ser autenticada, además del identificador del cliente, este enviará el nombre de usuario y la contraseña para establecer la conexión. Si se conecta mediante el antes mencionado TLS, la conexión se informará por el puerto 8883. El broker responderá a esta petición mediante un CONNACK, conteniendo el resultado de la petición, siendo las más usuales aceptada y denegada. Si la conexión ha sido exitosa, el cliente entonces podrá enviar mensajes de suscripción o publicación mediante mensajes similares. En caso de que quiera publicar un mensaje, se notificará con un PUBLISH que contendrá los datos del mensaje además de, en caso de ser un sistema de mensajería puro, el tópico al cual pertenece este mensaje. Para operaciones de suscripción, el cliente lo notificará mediante un SUBSCRIBE a lo que se responderá con un SUBACK o, en caso de que quiera desucribirse, un UNSUBSCRIBE respondido con un UNSUBACK. Notése en el léxico de los mensajes que todas las respuestas contienen el sufijo ACK. Esto se debe a que en este protocolo se usa este tipo de mensajes ACK, usados ampliamente en comunicaciones entre dispositivos interconectados y que

se usa para responder a los mensajes de petición de conexión.
Además de toda esta información que es intercambiada para los procesos de intercambio de datos o conexión, también se utilizan mensajes para asegurar que la conexión, una vez efectuada, es persistente. De manera periódica los clientes enviarán el mensaje PINGREQ a lo que el broker deberá responder con un PINGRESP en caso de recibirlo de manera exitosa. Finalmente, para la desconexión, el cliente envia un mensaje DISCONNECT.

Estructura del mensaje

Los mensajes del protocolo MQTT se dividen en tres partes. Esta cadena de bytes se dividirá en cabecera fija, cabecera opcional y contenido.

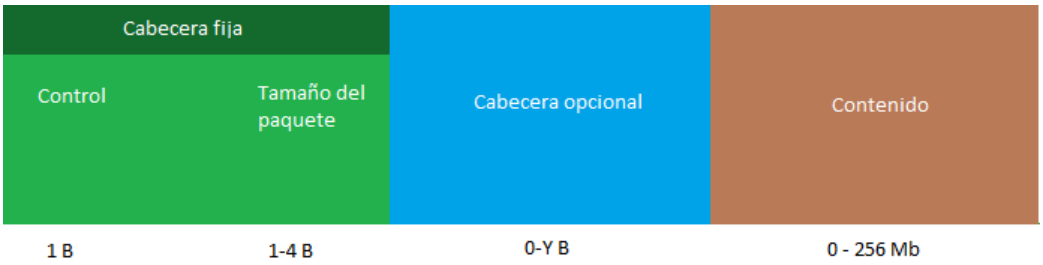


Figura 13: Contenido de un mensaje MQTT

La información de la cabecera fija es obligatoria para que la comunicación sea exitosa. Esta consta de un byte de control y de uno a cuatro bytes para informar de la longitud de mensaje. Para controlar cuantos bytes se usan para la longitud del mensaje, se usarán siete bits para informar el tamaño y el bit de más peso para indicar que viene más información a continuación.

127 (0x7F) (01111111)
16 383 (0xFF, 0x7F) (1111111 01111111)

En este ejemplo, vemos el uso del bit de control en el segundo mensaje para indicar que se transmite más información. Para el byte de control, los cuatro bits más significativos serán usados para identificar el tipo de mensaje que se va a enviar.

Bits de control

Name	Value	Direction of flow	Description
Reserved	0	Forbidden	Reserved
CONNECT	1	Client to Server	Client request to connect to Server
CONNACK	2	Server to Client	Connect acknowledgment
PUBLISH	3	Client to Server or Server to Client	Publish message
PUBACK	4	Client to Server or Server to Client	Publish acknowledgment
PUBREC	5	Client to Server or Server to Client	Publish received (assured delivery part 1)
PUBREL	6	Client to Server or Server to Client	Publish release (assured delivery part 2)
PUBCOMP	7	Client to Server or Server to Client	Publish complete (assured delivery part 3)
SUBSCRIBE	8	Client to Server	Client subscribe request
SUBACK	9	Server to Client	Subscribe acknowledgment
UNSUBSCRIBE	10	Client to Server	Unsubscribe request
UNSUBACK	11	Server to Client	Unsubscribe acknowledgment
PINGREQ	12	Client to Server	PING request
PINGRESP	13	Server to Client	PING response
DISCONNECT	14	Client to Server	Client is disconnecting
Reserved	15	Forbidden	Reserved

Figura 14: OASIS MQTT 3.1.1 Tipos de mensaje

Dependiendo del valor de esos primeros cuatro bits de control el broker podrá identificar qué tipo de mensaje se está enviando. Los siguientes cuatro bits, serán utilizados como flags para dar información adicional sobre los comandos. Excepto los comandos que observamos en la figura quince, estos cuatro bits menos significativos serán codificados como ceros. Aunque a priori parezcan no dar información ya que se enviarán como cero, no debe modificarse esa estructura para que la comunicación funcione.

El contenido más interesante lo encontraremos en el comando de publicar. El bit de DUP se usará para indicar que se debe duplicar el contenido del mensaje enviado y, por otra parte, el bit RETAIN se usará para identificar si se debe guardar el mensaje de manera persistente o no. Para ello, algunos brokers tienen implementada su propia persistencia de datos. Los dos bits de QoS serán codificados a continuación, donde se indicará que método se usará

PUBLISH	Used in MQTT 3.1.1	DUP[1]	QoS[2]	QoS[2]	RETAIN[3]
SUBSCRIBE	Reserved	0	0	1	0
UNSUBSCRIBE	Reserved	0	0	1	0

Figura 15: OASIS MQTT 3.1.1 Tipos de flags de control

y con ello, que nivel de calidad de servicio se quiere dar en la comunicación por tal de transmitir los mensajes de manera exitosa.

QoS

La calidad de servicio (quality of service por sus siglas en inglés) es una medida que se utiliza para cuantificar la eficacia de un servicio que usualmente se da en el campo de las telecomunicaciones. Estas medidas se cuantifican mediante diferentes aspectos como la pérdida de paquetes, retardo, y ratio de bits. Particularmente, son medidas que se toman para tener una visión de como el usuario ve la eficacia de la red. En el caso de MQTT, tenemos tres niveles diferentes de calidad de servicio que serán determinados por los dos bits vistos en la figura quince.

· QoS 0: El mensaje se envía una única vez. En caso de fallo por lo que puede que alguno no se entregue.
· QoS 1: El mensaje se envía hasta que se garantiza la entrega. En caso de fallo, el suscriptor puede recibir algún mensaje duplicados.
· QoS 2: Se garantiza que cada mensaje se entrega al suscriptor, y únicamente una vez.

Al tener dos bits para poder codificar estas opciones, observaremos que realmente podremos tener cuatro opciones. Por tanto, la última opción, es reservada por el sistema.

Cabecera opcional

Algunos paquetes de control usan esta cabecera que se contiene entre la cabecera fija y el contenido. El contenido de la cabecera opcional varía dependiendo del tipo de paquete, donde la mayoría de los paquetes de control incluyen un campo de identificador de paquete de dos bytes. Los paquetes de control PUBLISH (donde QoS > 0), PUBACK, PUBREC, PUBREL, PUBCOMP, SUBSCRIBE, SUBACK, UNSUBSCRIBE, UNSUBACK deben contener un identificador de paquete de dieciséis bits no nulo. El cliente debe

asignar un identificador de paquete actualmente no utilizado cada vez que envía un paquete nuevo de uno de estos tipos. Si el cliente reenvía un paquete en particular, debe usar el mismo identificador de paquete en posteriores comunicaciones. El identificador de paquete estará disponible después de que el cliente haya recibido el ACK correspondiente.

Contenido

Por último, tras formar toda la trama de la manera correspondiente respecto a la comunicación que se quiera tener, se enviará el contenido de la trama, la cual no es obligatoria para todos los tipos de mensaje y sólo sera necesaria para CONNECT, SUBSCRIBE, SUBACK, UNSUBSCRIBE y opcional para PUBLISH.

4.2.2. I2C

Adaptando las herramientas de hardware disponibles al sistema que se quería crear, el único método de comunicación y, a la par, uno de los más extendidos y estandarizados en sistemas electrónicos era el I2C. El I2C(Inter-Integrated-Circuit) es un estándar de comunicación serial síncrona que está caracterizado por ser un bus multi-master, lo que significa que varios dispositivos maestros pueden controlar el bus y comunicarse con varios dispositivos esclavos conectados a ellos.
La terminología de maestro y esclavo se usan para definir los roles de los dispositivos conectados. En este contexto, el maestro es el que tiene el control principal del bus de comunicación, y, por tanto, el encargado de iniciar, coordinar y cerrar las transferencias de datos con los dispositivos esclavos. Pueden coexistir diferentes dispositivos maestros siempre y cuando la transferencia de datos entre ellos y sus esclavos sean coordinados por tal de evitar colisiones en la comunicación. A su vez, en un mismo sistema, pueden convivir múltiples dispositivos esclavos en un sistema, usualmente limitados a un número de ciento veintisiete siempre y cuando se puedan suplir las necesidades de estos dispositivos.
La comunicación con este protocolo se inicia cuando el maestro envía una señal de inicio START y la dirección del dispositivo al cual quiere conectarse. Tras ello, se libera la línea SDA y el dispositivo que posee la dirección indicada responde mediante un ACK y espera. Con la conexión iniciada, el maestro puede enviar o recibir datos, siendo comprobada su veracidad mediante ACK/NACK en cada byte de datos.

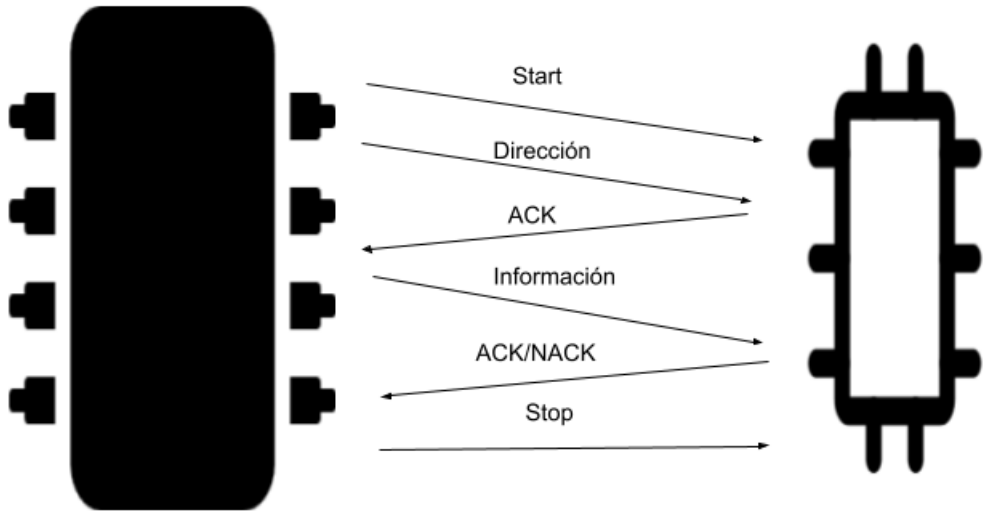


Figura 16: Representación de la comunicación I2C. A la izq. maestro, a la der. el esclavo

En nuestro sistema, un dispositivo maestro ESP8266 conectará mediante I2C con los dispositivos AmbiMate y CCS811, siendo estos últimos los que tomarán datos. Por tanto, como se aprecia en el ejemplo de comunicación de la figura dieciséis, es importante el respetar el funcionamiento del protocolo por tal de que el intercambio de datos entre los tres dispositivos sea exitoso.

4.2.3. Almacenamiento en la nube mediante REST

Una parte de todo el procesamiento de datos era el almacenamiento en la nube. Para ello se contemplaron diversos distribuidores de servidores en la nube como AWS o Azure. Tras realizar una búsqueda de un servicio que se adaptara a las necesidades, se descubrió AdafruitIO. Este es un servicio en la nube que permite la subida de datos, orientándose a aquellos relacionados, sobre todo, con IoT.
Esto se debe a que la subida de datos está limitada a datos de poco tamaño y que, principalmente sus servicios están orientados al almacenamiento de lectura de sensores. Podemos ver que, por ejemplo, tiene servicios de mensajería si se detectan valores inusuales o de alerta, útiles para casos en los que las lecturas de información se vuelvan incongruentes y se necesite revisar el sistema. Además de esto, tiene interfaces gráficas disponibles para compartir la información y verla en tiempo real, disponible para ser conectada a servi-

cios web como Twitter, RSS feeds o servicios meteorológicos.

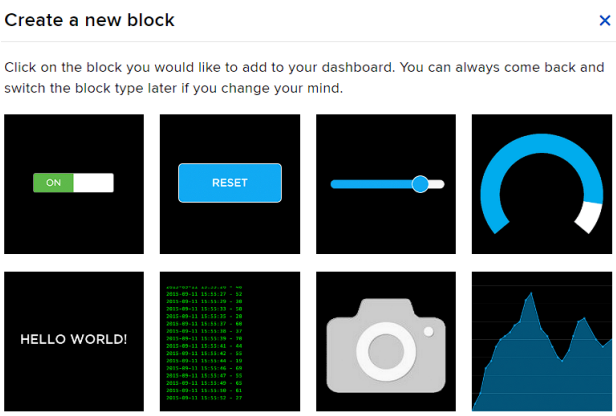


Figura 17: Algunas de las interfaces de AdaFruitIO

Por ejemplo, mediante una interfaz gráfica como las que se muestran en la figura diecisiete, se pueden mostrar los datos de dióxido de carbono y, haciendo estos datos públicos, serán accesibles mediante un enlace generado por esa dashboard. En la figura dieciocho podemos ver un ejemplo de como se podrían mostrar los datos públicos.

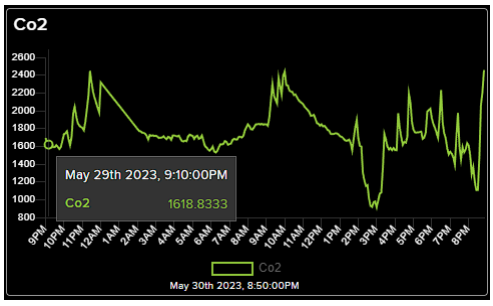
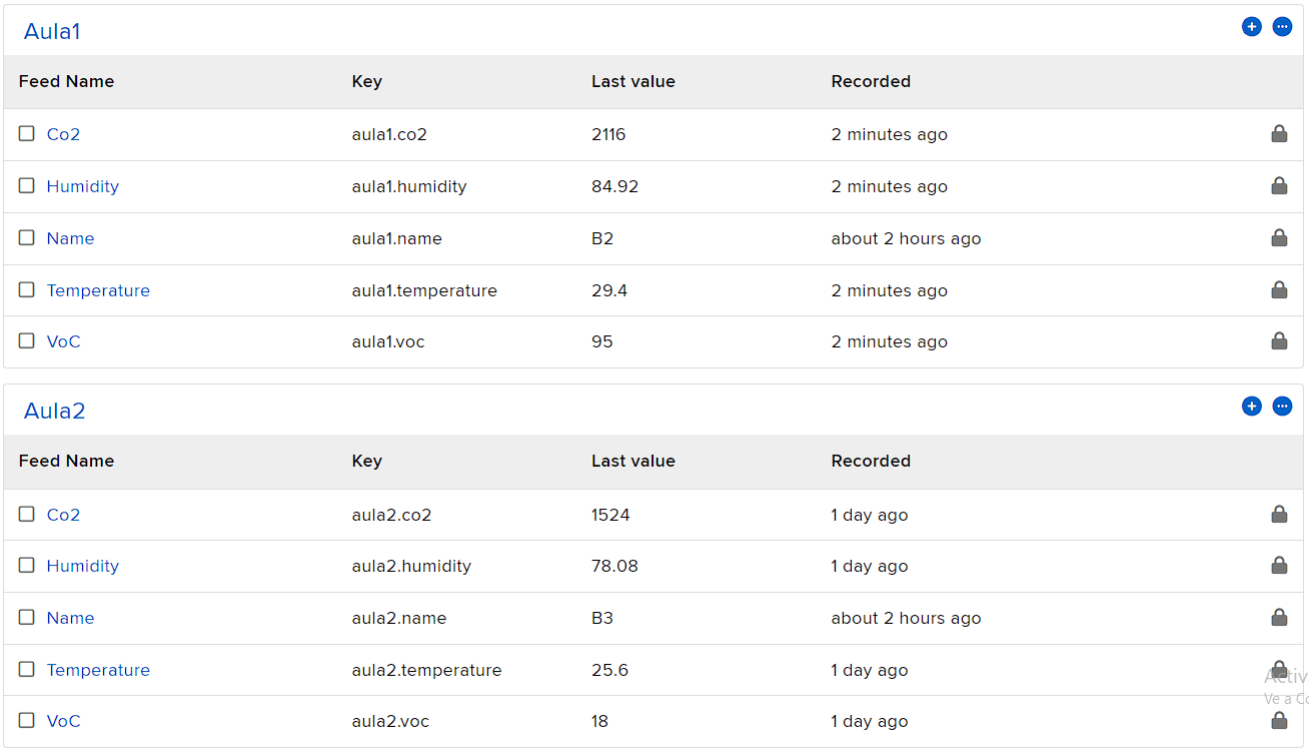


Figura 18: Grupos asignados a Adafruit

Los datos en *AdafruitIo* se distribuyen en *feeds*. Además, se pueden crear grupos para organizarlos de manera más eficiente. Se decidió organizar los datos en distribuciones por aula de tal manera que se pudiera adquirir los datos con llamadas más sencillas a nivel de código y para poder identificarlos rápidamente.

Observamos en la figura diecinueve las llaves de cada feed, necesarios para



Aula1

Feed Name	Key	Last value	Recorded
Co2	aula1.co2	2116	2 minutes ago
Humidity	aula1.humidity	84.92	2 minutes ago
Name	aula1.name	B2	about 2 hours ago
Temperature	aula1.temperature	29.4	2 minutes ago
VoC	aula1.voc	95	2 minutes ago

Aula2

Feed Name	Key	Last value	Recorded
Co2	aula2.co2	1524	1 day ago
Humidity	aula2.humidity	78.08	1 day ago
Name	aula2.name	B3	about 2 hours ago
Temperature	aula2.temperature	25.6	1 day ago
VoC	aula2.voc	18	1 day ago

Figura 19: Visualización gráfica de dióxido de carbono a lo largo de un día

poder enviar una petición REST y guardar los datos. Tenemos también los nombres de estos y sus últimos valores, donde indica en que momento se ha realizado la última llamada al servicio. Dentro de cada feed, se pueden filtrar los datos para obtener una visualización de como han evolucionado con el tiempo.

Peticiones REST

Este servicio cloud ofrece a su disposición diversas librerías para diferentes lenguajes de programación o entornos como Arduino, Python, CircuitPython y Ruby. Aunque las herramientas que nos proporcionan son adecuadas, se prefirió usar peticiones mediante curl, una herramienta que se usa primordialmente para realizar solicitudes HTTP. Principalmente, se tomó esta decisión debido a su versatilidad y a que diferentes servicios no eran funcionales para algunas peticiones. Las peticiones más usadas para este servicio son las de POST para actualizar los datos que llegan de los sensores y las de GET para extraerlos. Dentro de sus variantes, tenemos diversas opciones para extraer

los datos, por ejemplo, extraer el último dato de un feed o obtener todos los datos en una franja temporal.

4.3. Arquitectura del sistema

La jerarquía del sistema está organizada de manera que los datos obtenidos se transmitan en los diferentes dispositivos hasta llegar a la nube, donde serán almacenados. De manera periódica, la ESP8266 pedirá los datos tanto al AmbiMate como al CCS811. Estos, tras un corto período de recogida de datos, enviarán mediante I2C los datos que ahora estarán en posesión del microcontrolador. Una vez recibidos, este publicará un mensaje mediante MQTT a la que la Raspberry estará suscrito y recibirá. Una vez recibido el mensaje, está enviará una petición REST a los servidores de AdafruitIo almacenando finalmente los datos.

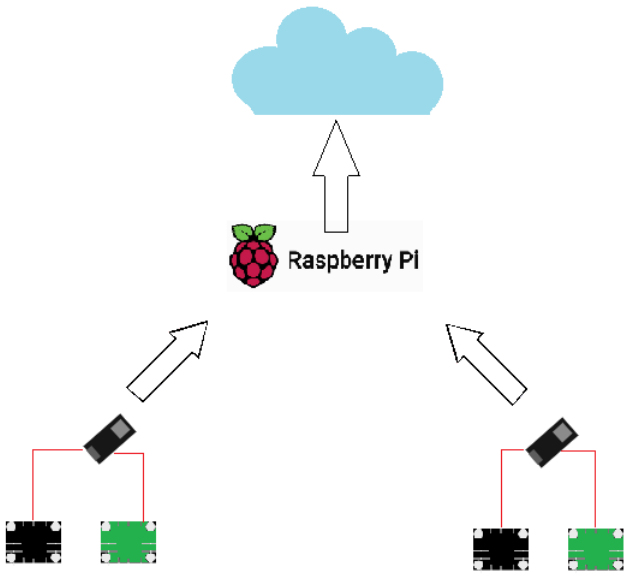


Figura 20: Infograma sobre la jerarquía del sistema

En la figura veinte se muestra como el proceso transcurre desde abajo hacia arriba, finalizando una vez los datos han sido alojados en la nube. Este sistema, como tal, puede sostener, con las limitaciones de este, a un número mayor de ESP8266 que estén enviando datos. El sistema, probado con hasta

cuatro ESP8266 con sus correspondientes módulos de sensores comunicando, se cree que podría abarcar más, todo dependiendo de la periodicidad con la que se quieren obtener los datos y la robustez de este. Por ejemplo, si se quieren obtener datos cada segundo y a la par enviar a la nube, el sistema podría no soportarlo o ralentizarse ya que existen limitaciones en la velocidad de la red a la hora de subir los datos.

4.3.1. Conexión de la ESP8266 con los sensores Ambimate y CCS811

La conexión de los sensores Ambimate y CCS811 se realizará utilizando los pines SCL (Serial Clock) y SDA (Serial Data). La ESP8266 actuará como la fuente de alimentación principal para todo el sistema. Esto significa que la ESP8266 será alimentada mediante una batería, proporcionando la potencia y la corriente necesarias para el correcto funcionamiento de los sensores y de la propia ESP8266.
Durante el montaje del circuito, hay que tener en cuenta que todos los elementos se comuniquen entre si de manera correcta. A la hora de montarlo, las recomendación es de hacerlo en una protoboard o similares. El soldado de todos los pines es necesario para poder darle robustez al sistema a la vez que se evitan errores en la comunicación debido a la mala gestión de los elementos físicos del circuito.
A la par que se monta el circuito, es recomendable usar un voltímetro para comprobar que la tensión es la adecuada para cada elemento. En el Anexo A se puede encontrar un diagrama del circuito para poder visualizar cual ha sido la conexión final entre los componentes. En el anexo B se puede encontrar la conexión final del circuito en físico.

5. Ingeniería de detalle

En esta sección se pretende abarcar toda la implementación que se ha hecho a nivel de software sobre las herramientas disponibles.

5.1. Raspberry Pi como router

Para permitir la comunicación entre las ESP8266 y la Raspberry Pi, así como la transmisión de datos a través de MQTT, fue necesario configurar la Raspberry Pi como un router para proporcionar una red Wi-Fi a la que las ESP8266 pudieran conectarse. Para lograr esto, se utilizó la tarjeta de red

de la Raspberry Pi para el computador en un punto de acceso inalámbrico.
En este proceso, se empleó el programa hostapd. Diseñado específicamente para sistemas operativos Linux, ofrece funcionalidades de autenticación y encriptación necesarias para crear una red inalámbrica segura. Hostapd permite configurar los parámetros de red, como el nombre de la red (SSID), el modo de seguridad y la contraseña.
Además, se utilizó dnsmasq, un software de código abierto, que desempeña un papel fundamental en este escenario. Dnsmasq cumple dos funciones principales: DHCP (Protocolo de Configuración Dinámica de Host) y DNS (Sistema de Nombres de Dominio).
La función DHCP de dnsmasq permite asignar direcciones IP automáticamente a todos los dispositivos que se conectan al punto de acceso. Esto significa que cada dispositivo que se conecte recibirá una dirección IP única dentro de la red, lo que facilita su identificación y comunicación con otros dispositivos.
Por otro lado, la función DNS de dnsmasq resuelve las consultas de nombres de dominio realizadas por los dispositivos conectados a la red. Esto permite que los dispositivos puedan traducir los nombres de dominio en direcciones IP y acceder a los recursos de Internet correspondientes.
En el contexto de una red doméstica, los enrutadores comerciales incluyen estas funcionalidades de manera predeterminada. Al utilizar dnsmasq en este escenario, se imita ese comportamiento y se crea una red estable y sin errores, que permite la conexión de múltiples dispositivos y garantiza que cada uno de ellos reciba una dirección IP dentro del rango asignado.
Mediante hostapd, se configuró una red con seguridad WPA-2. Para mayor seguridad, se podría haber implementado que únicamente se permitiera conectar a los dispositivos que tuvieran las MAC de las ESP8266, pero se consideró que con una red con contraseña ya era suficiente.
Por pragmatismo, se necesitaba que la red se creara sin necesidad de acceder a la interfaz gráfica o ejecutar ningún comando de ejecución. En los sistemas operativos Linux, existe un archivo de configuración llamado rc.local, que se encuentra en la carpeta /etc/ del sistema. Es un archivo de script que se ejecuta durante el arranque del sistema, antes de que se inicie el entorno de usuario, siendo útil para ejecutar comandos o scripts en el arranque y así automatizar el sistema. Como administrador del sistema, se puede modificar este archivo para poder ejecutar comandos de bash que a su vez nos permiten ejecutar scripts de código sin necesidad de acceder a la Raspberry.
Cada vez que se encienda la Raspberry, se ejecutará el script para crear la red Wi-Fi. Inicialmente, se asegurará de finalizar los procesos que puedan crear conflictos y desbloquear cualquier bloqueo de radiofrecuencia. Tras esto, se eliminaran las reglas existentes en las tablas de IP, estableciendo nuevas para

permitir el enrutamiento de paquetes mediante la interfaz de Ethernet y la interfaz inalámbrica. Se establecen las opciones para permitir el enrutamiento IP y se configuran las máscaras de subred en la interfaz inalámbrica. Finalmente, se elimina la ruta predefinida de la interfaz inalámbrica y se crea un archivo de configuración para dnsmasq que especifica la interfaz inalámbrica, el rango DHCP, el servidor DNS y otras opciones relacionadas.
Tras reiniciar ambos servicios de dnsmasq y hostapd, la Raspberry Pi ya estará proveyendo de una red inalámbrica a la que se pueda acceder.

5.2. Programación del ESP8266

La programación de este microcontrolador se ha realizado en lenguaje C++ usando los IDE (entorno de desarrollo integrado) Sloeber y Arduino. Mientras que Arduino se ha utilizado para pruebas específicas en pequeñas funcionalidades como pruebas en los sensores o la conexión WiFi, la implementación que engloba todo el programa se ha hecho en Sloeber. Sloeber es una herramienta de código abierto basada en Eclipse. Esta ofrece una interfaz más avanzada y a su vez facilita la escritura y depuración de código.
Se han utilizado diversas librerías para facilitar la programación del microcontrolador. Entre ellas, tenemos las propias desarrolladas por Arduino para tareas de conexión a red, conexión MQTT, y comunicación I2C. Dado que el microcontrolador tiene una capacidad de almacenaje bastante pobre en comparación a los tamaños medios que poseen las herramientas de hardware, la programación ha sido meticulosa en cuanto al almacenamiento de datos se refiere. Ha sido importante el control de las variables por tal de no alojar datos innecesarios y trabajar únicamente con los imprescindibles para la correcta ejecución en el programa. Para ese caso, se han evitado el uso de bibliotecas estándar de Arduino como String y se ha optado por el tipo de dato primitivo char*. Esta lógica se ha aplicado con diferentes tipos de datos primitivo por tal de ser lo más conciso en cuanto a tamaño de memoria se necesita para que todo el programa funcione. Trabajando con micro-controladores, es fundamental esta aproximación.
En la figura veintiuno se puede ver un diagrama de flujo que muestra la lógica inicial del programa.

Para inicializar los módulos de los sensores, se les piden los datos por tal de apreciar que las lecturas son correctas. Un sensor con un cableado que no hace buena conexión o que está estropeado devolverá cero en todas sus lecturas. Si se identifica este patrón, se procede a dar el mensaje de error. Las lecturas de los sensores se hacen de manera individual ya que así se debe proceder

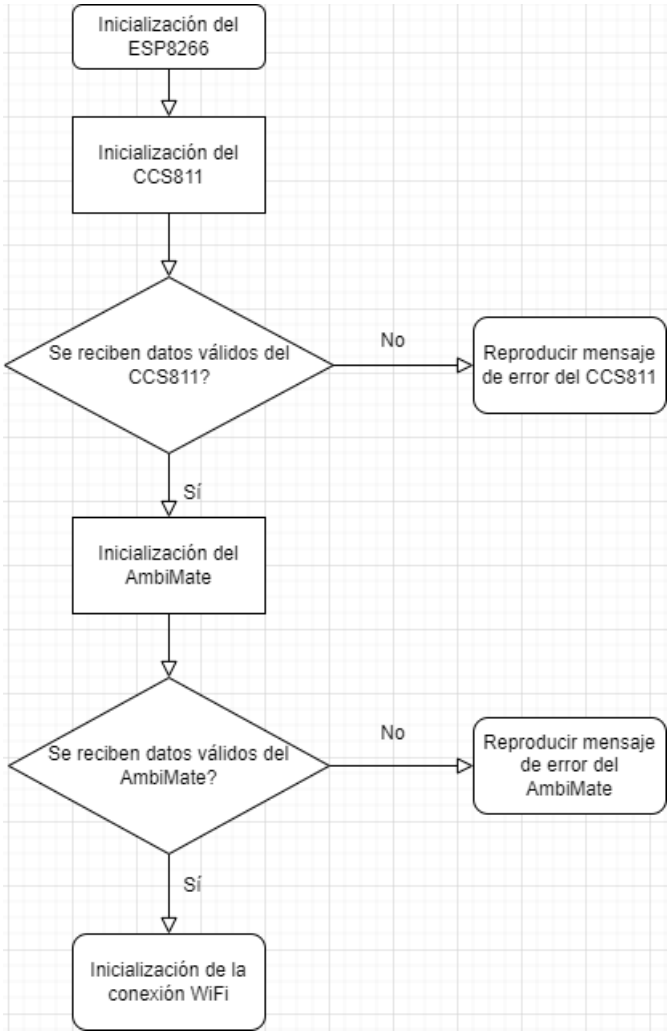


Figura 21: Diagrama de flujo previo a la conexión a la red inalámbrica

para trabajar con el protocolo de comunicación I2C. Primero se establecerá comunicación con el CCS811 y, una vez recibidos los datos y cerrada la conexión, se puede abrir de nuevo la conexión con AmbiMate para repetir el mismo procedimiento. Es de importancia que se cierre la comunicación para evitar las colisiones en el bus.
El mensaje de error está inspirado en las placas bases. Cuando estas identifican que un componente fundamental para el arranque no funciona, emiten un pitido con una cadencia diferente dependiendo del error. Como la ESP8266 no tiene acceso a un altavoz, se decidió usar el LED integrado para poder notificar al usuario de un mal funcionamiento. Los mensajes de error son los siguientes:

Dispositivo	Señal de error LED
CCS811	Iluminación de un segundo, dos iluminaciones de medio segundo
AmbiMate	Tres iluminaciones de medio segundo

Estos mensajes de error se diseñaron para que el usuario pudiera reemplazar las piezas o conexión en mal estado, identificadas de una manera sencilla. Una vez se ha comprobado que cada pieza funciona de manera individual, se puede proceder a las conexiones de red y la transmisión de mensajes. El diagrama de flujo tras la comprobación de los sensores se aprecia en la figura veintidós.

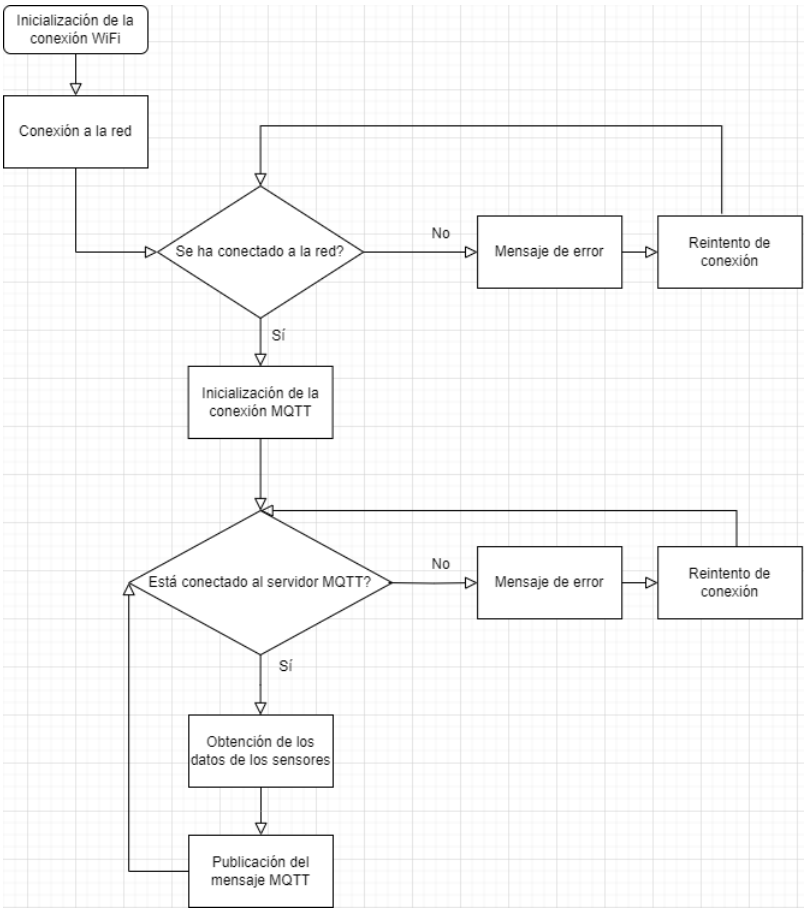


Figura 22: Diagrama de flujo de la lógica de transmisión

En la figura veintidós vemos la lógica principal de la transmisión de datos. Se esperan dos bucles infinitos con sus respectivos mensajes de error. Dado que sin ambas conexiones funcionales es imposible que el dispositivo pueda hacer sus tareas de transmisión, no se puede avanzar en la lógica del programa. En este caso habría que identificar cuál es el error y comprobar si el dispositivo Raspberry Pi funciona como puerta de enlace.
El retraso para la lectura de los sensores es de un minuto. Debido a la propia naturaleza de los datos y a limitaciones con el módulo CCS811, se concretó que era suficiente la lectura de datos a cada minuto, siendo muy difícil que los valores puedan variar drásticamente en un período de tiempo tan corto en circunstancias normales.

5.3. Raspberry como broker en MQTT

La sección de MQTT hablaba sobre la importancia del broker en este protocolo de comunicación. El broker actúa como intermediario en las comunicaciones, responsable de recibir, almacenar y reenviar los mensajes publicados. Para que el sistema funcione de manera autónoma, se decidió usar la propia RasberryPi como broker. Para ello, se utilizó mosquitto, una implementación de broker en código abierto que se encarga de recibir almacenar y reenviar los mensajes entre publicadores y suscriptores.
Esto permite el no tener dependencias externas ya que indicando la propia IP de la RapberryPi al iniciar la comunicación MQTT, los microcontroladores ya se puede comunicar con esta. Existen diversas formas de implementar este protocolo. Por ejemplo, se podría haber usado un servidor externo este actuara como broker en la comunicación.
Dado que la carga de datos en la Raspberry Pi no iba a ser muy elevada, se decidió, con el objetivo de abaratar costes y dar autonomía al sistema, usar la Raspberry Pi como broker.

5.4. Tratamiento de datos con la Raspberry Pi

Usando un programa desarrollado en Python, la Raspberry Pi se encarga mediante la librería de Paho MQTT de subscribirse a los mensajes del tópico en el que los sensores están enviando sus mensajes. Una vez se hayan recibido estos datos, el programa se encargará de subirlos a la nube mediante llamadas curl. Al no ser un computador tan potente, la Raspberry Pi tiene problemas de latencia y de inestabilidad en la conexión. Como esto causaba muchas pérdidas de datos en el momento de la subida de estos a la nube,

se decidió usar la biblioteca TinyDB para tratarlo. En la figura veintitrés se puede observar la lógica que sigue el programa para almacenar los datos en la nube.

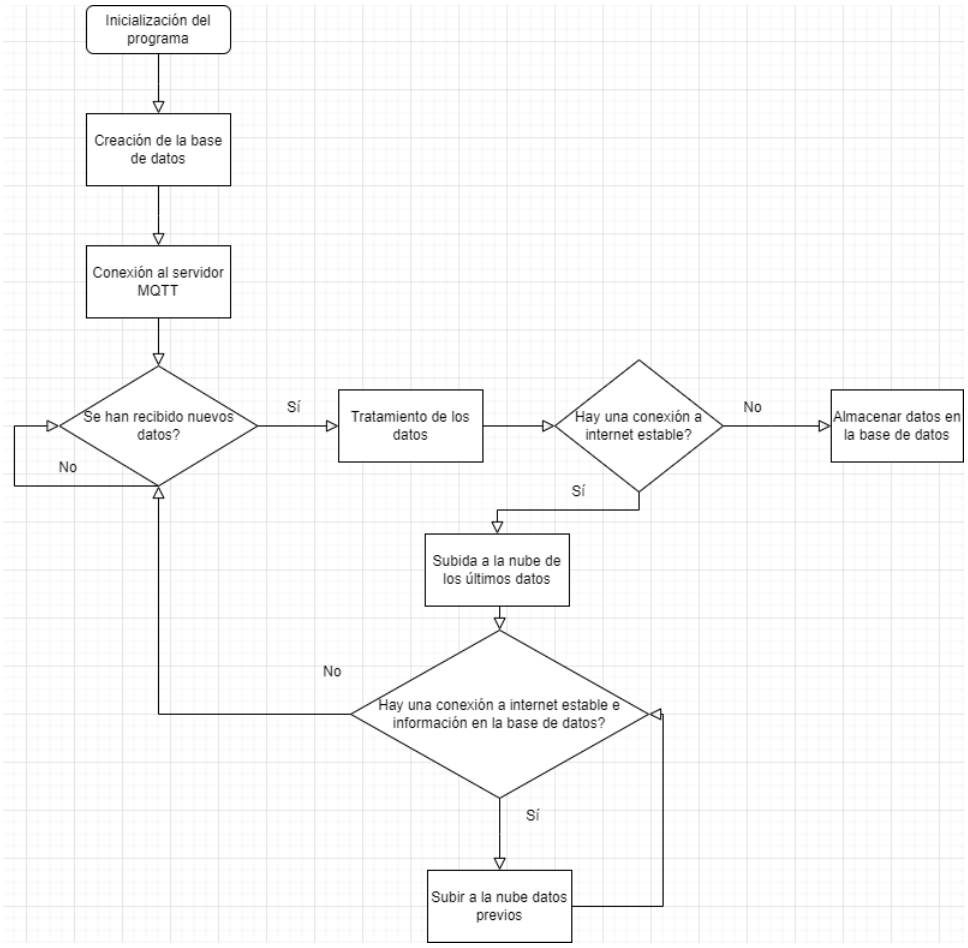


Figura 23: Diagrama de flujo del tratamiento de datos y almacenamiento en la nube

El programa se mantendrá en un bucle constante a la espera de nuevos mensajes. Gracias a el almacenamiento de los datos que se pueden perder debido a una conexión inestable, se reduce drásticamente la posibilidad de pérdida de datos. Por otro lado, el servicio en la nube AdafruitIO permite añadir una marca de tiempo, de manera que los datos quedarán organizados en los datos finales a pesar de que hayan podido almacenarse en instantes de tiempo diferentes.
Para darle autonomía al sistema, tal y como se ha hecho para la creación

de la red Wi-Fi, también se ejecutará este programa tras el arranque de la Raspberry Pi. Esto, nos permitirá, que una vez tenga alimentación y esté conectada a una red, el sistema ejecutará el programa sin necesidad de hacer nada.

5.5. Desarrollo de aplicación móvil

Para ofrecer al usuario final una interfaz sencilla y accesible, se propuso el desarrollo de una aplicación móvil que pudiera monitorizar los datos del dispositivo. Otros métodos de acceso que se han mencionado como peticiones REST o acceso a URL podrían ser más complejos para un público común, así que se optó por simplificarlo en una interfaz que obtuviera los datos más recientes de manera legible e intuitiva.
Considerando que uno de los posibles usuarios podría ser un conserje en un lugar público con servicios de mantenimiento, se decidió preguntar a dos hombres que realizaban estas tareas por tal de poder crear una aplicación a su gusto. Este criterio se aplicó porque a día de hoy, aún existen muchísimas empresas que no tienen en cuenta al usuario final ni desarrollan teniendo a las personas como el eje central del producto. Aunque la cantidad de información recopilada pueda estar sesgada debido al número reducido de entrevistados, el núcleo del diseño concordaba con las ideas que se querían trasladar a la aplicación final. Las premisas que se extrajeron de esas entrevistas eran dos: que la aplicación fuera sencilla e intuitiva. Con esto en mente, se diseñó una interfaz que diera la información de manera directa e intuitiva en la navegación, sin servicios adicionales y con las opciones de navegación que fueran únicamente las necesarias para obtener la información.
Con toda esta información, se diseñó la aplicación con la que se obtuvieran datos de forma eficiente y amigable para el usuario. En el anexo C se pueden encontrar imágenes de como se ve finalmente la interfaz gráfica.

6. Viabilidad técnica

El sistema diseñado finalmente, fue instalado en la Universitat de Barcelona. Se considera un sistema robusto debido a que al ser instalado, no necesitó ningún ajuste intermedio por tal de hacerlo funcionar en otro entorno.
De la misma manera, el sistema, una vez instalado correctamente, no ha necesitado ningún tipo de supervisión en ningún momento para que siguiera funcionando. En esta sección, se pretende abalar estos hechos con los datos recogidos y, al mismo tiempo, destacar posibles limitaciones del sistema.

6.1. Limitaciones técnicas conocidas

Tras la implementación y las pruebas del sistema, se observaron limitaciones técnicas que podían impedir un buen funcionamiento. Como la conexión de la Raspberry Pi tiende a ser inestable, es habitual que se produzca una pérdida de datos en la subida a la nube si se pretenden almacenar muchos datos al mismo tiempo. Estos casos se podrían dar de dos maneras; enviando datos en un espacio de tiempo muy corto de manera constante desde un pequeño número de ESP8266 o con un número de ESP8266 muy grande enviando datos.
En el primer caso se pudo probar enviando datos a cada segundo, con cuatro ESP8266. Aunque la pérdida de datos no era significativa dado que se hicieron mejoras en el software como la implementación de una base de datos, sí que la cantidad de datos que se perdían tendía a ser mayor. También, hay que comprender que la transacción de datos entre los módulos de los sensores y ESP8266 no es instantánea, así que siempre debe haber un tiempo de espera respetando las limitaciones de los hardware por tal de un buen funcionamiento. Por ello, se comprobó que el envío de información de los sensores funcionaba bien a partir de un periodo de diez segundos aproximadamente para esta casuística.
Por otro lado, tras la instalación del sistema en la Universitat de Barcelona, se observó como la estructura del edificio podía limitar que los dispositivos estuvieran interconectados entre ellos.

En la figura veinticuatro se observa el punto de instalación en el aula B2 de la universidad. A pesar de que el aula más cercana era la B3, fue imposible conectar un dispositivo ya que la conexión Wi-Fi era muy débil. En cambio, podemos ver como el rango de conexión sí permitía la instalación en el aula B1, a priori, más lejana. Debido a la propia estructura del edificio, las paredes que separan las aulas B2 y B3 impiden que haya una conexión adecuada y, por tanto, imposibilita la implementación del sistema en estas dos aulas con la Raspberry Pi conectada en el aula B2.
Para paliar estas deficiencias del sistema, es conveniente que previamente a la instalación se compruebe como va a estar estructurado el sistema y que posibles limitaciones se pueden tener con la red inalámbrica. Como se puede observar, es importante la estructura del entorno ya que la tarjeta de red de la Raspberry Pi no tiene una alta capacidad de emisión.

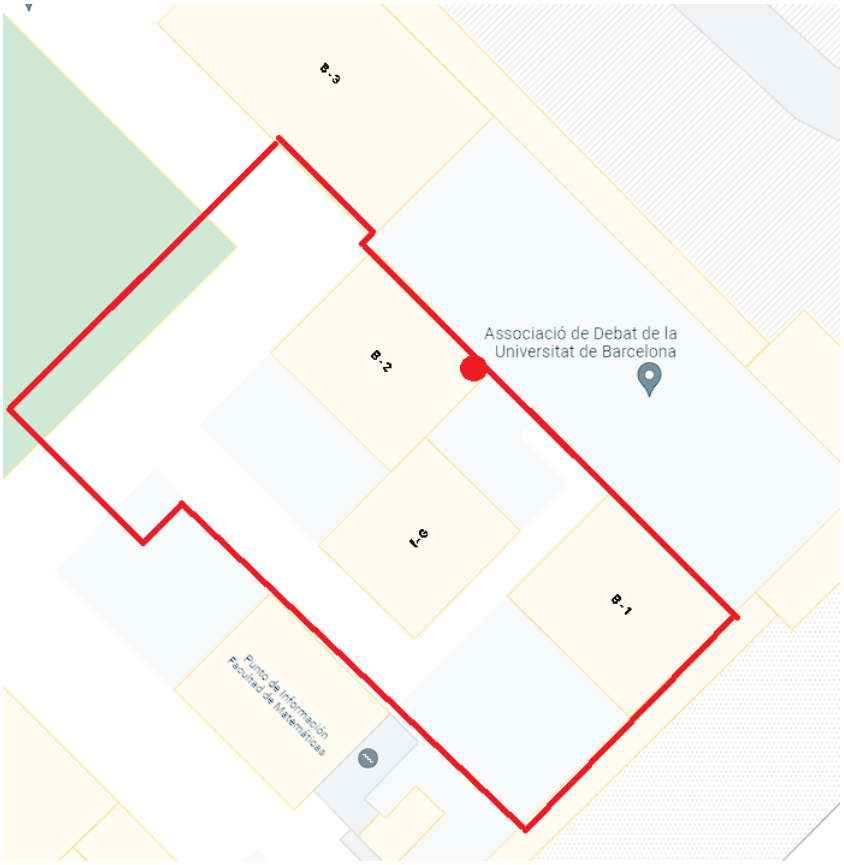


Figura 24: Plano de la Universitat de Barcelona, con indicaciones de la instalación

6.2. Análisis de los datos obtenidos

La importancia de los datos para este proyecto es notoria. En esta sección se va a proceder a analizar datos obtenidos en diferentes entornos por tal de contrastar estos. También se buscarán similitudes y patrones clave en cuanto a lo que muestran los datos. Los entornos de obtención de los datos han sido en el entorno de prácticas, un piso ubicado en Esplugues de Llobregat, y la Universitat de Barcelona.
Debido a que el volumen de datos no es lo suficientemente grande, aplicar técnicas de predicción podrían llevarnos a resultados inexactos y, por tanto, se realizarán cálculos como el coeficiente de Pearson para buscar concordancias en los datos. Este calculo parece óptimo para evaluar la correlación que existe entre COV y CO2 y entre la humedad y la temperatura.
A continuación, se subdividirá el análisis de los datos dependiendo del entorno

que se hayan tomado para, posteriormente, buscar relación entre ellos.

6.2.1. Calidad del aire en un espacio abierto

En la sección 4.1.3 se encuentra detallado el espacio de pruebas con mejor calidad del aire. Concretar, que debido a la naturaleza de los sensores y a sus requisitos, que el nivel de dióxido de carbono sea bajo en la lectura del sensor, no indica que este aire sea el más limpio, si no que será el más limpio que ha podido identificar el sensor y en el que se basa para sus mediciones.

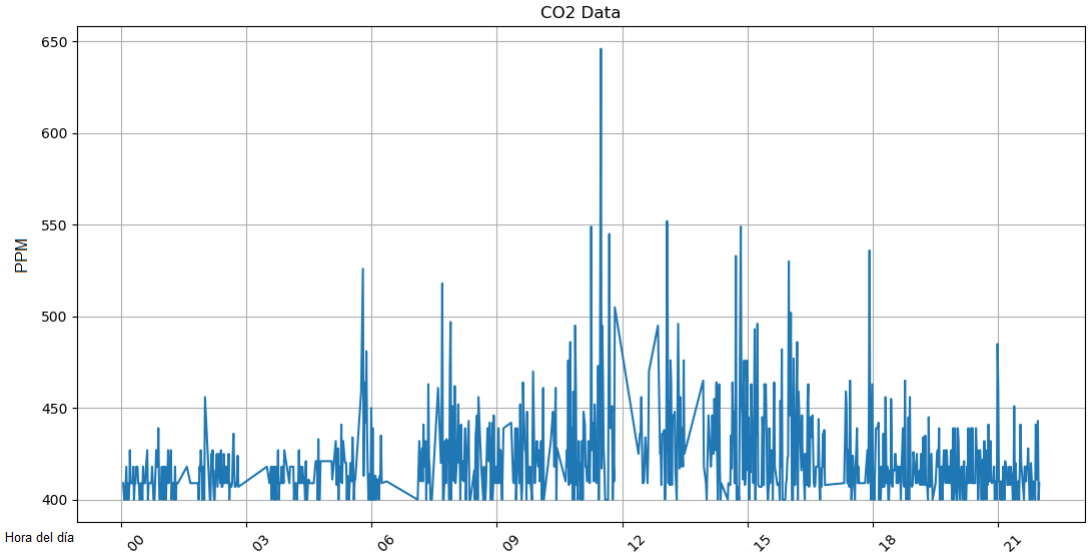


Figura 25: Nivel de dióxido de carbono en un ambiente con ventilación

En la figura veinticinco podemos ver la fluctuación de los niveles de dióxido de carbono, medido en ppm, respecto a la referencia inicial del sensor.
Con estos datos, ya podemos observar como los niveles de dióxido de carbono aumentan durante el día, donde también se ve la mayor varianza. Observamos su coincidencia con el horario laboral diurno del territorio de Barcelona, donde la mayor parte de la población está más activa según datos del Instituto nacional de seguridad e higiene en el trabajo . Los valores de CO2 son bastante estables a lo largo del día con picos de no más de dos cientas partículas por millón.

Como se puede apreciar, los datos de la figura veintiséis donde se ven los

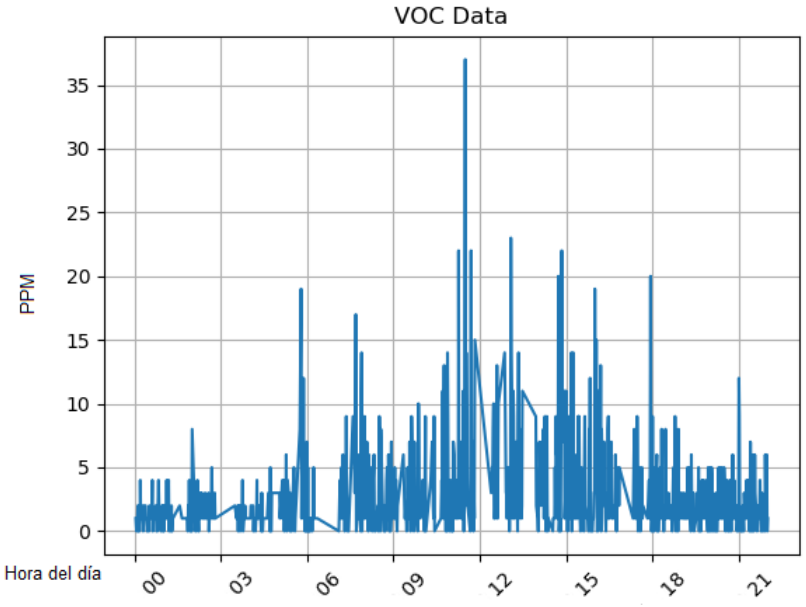


Figura 26: Nivel de compuestos orgánicos volátiles en un ambiente con ventilación

datos de COV guardan cierta relación en cuanto al CO2 se refiere. Ambos tienen picos en horas similares, caso que observaremos a continuación. En ambos casos la calidad del aire no es excelente pero se ciñe a los datos que normalmente se dan en la ciudad. La figura veintisiete muestra la concordancia que hay entre COV y CO2. Estos datos fueron tomados entre el veintisiete y el treinta y uno de mayo de dos mil veintitrés. Se puede observar que los picos más altos de CO2 se estiman junto con los de VOC. Según el coeficiente de correlación de Pearson:

$$r = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}}$$

El índice de correlación es de 0.75, indicando una fuerte dependencia entre COV Y CO2.

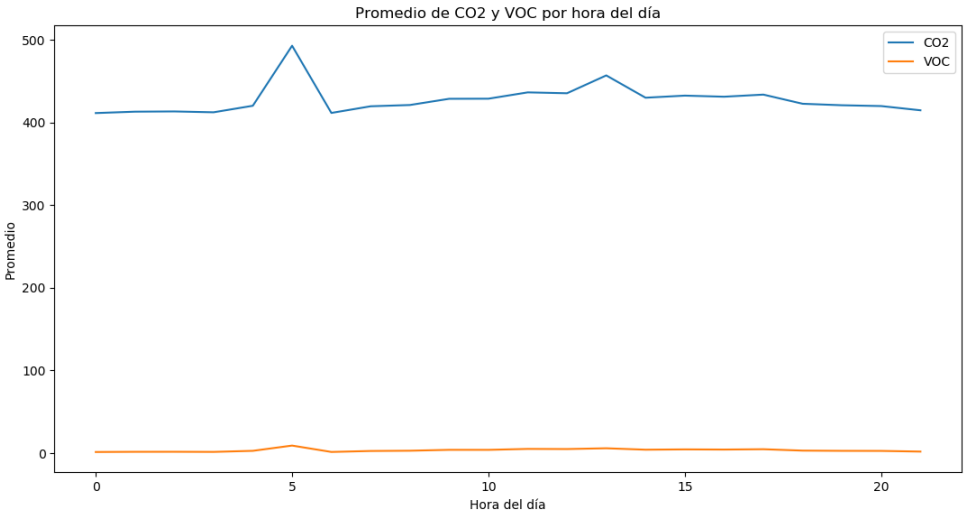


Figura 27: Comparación de la media diaria entre CO2 y COV

6.2.2. Calidad del aire en un espacio cerrado

Compararemos los datos de la anterior sección con datos tomados en una habitación cerrada cercana. Estos datos se han extraído durante la misma franja temporal para poder equipararlos y evaluarlos de mejor manera. En la figura veintiocho observamos la fluctuación de dióxido de carbono en una habitación cerrada.

Se puede observar que la variación en los valores es mucho mayor que en un ambiente con ventilación. Además, se pueden determinar en qué horas del día ha habido mayor corriente de nuevo aire mejorando la calidad del aire. Los valores que se aprecian en este gráfico ya indican que las personas ubicadas en ese ambiente pueden ser perjudicadas por la calidad del aire que respiran. En la varianza de estos datos, también se debe contemplar que el lugar donde se han tomado las muestras es transitado por personas fumadoras, haciendo que los niveles de dióxido de carbono también suban y bajen dependiendo de este componente. Procedemos, también, a ver los resultados de COV en la figura veintinueve.

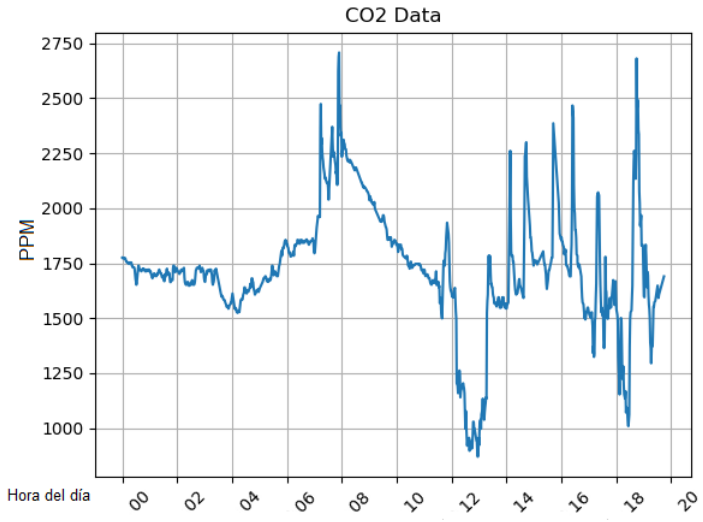


Figura 28: Nivel de dióxido de carbono en una habitación cerrada

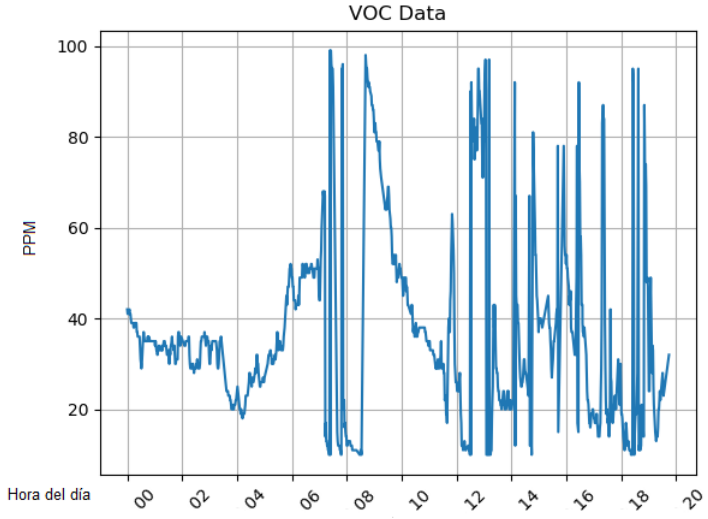


Figura 29: Nivel de compuestos orgánicos volátiles en una habitación cerrada

Estos datos contrastan con los de dióxido carbono en la medida en que ambos son muy variables durante el día. Se puede apreciar como hay puntos en los que los niveles son insalubres, pero con un margen de mejora muy elevado.
Todo esto nos indica que a la vez que la variabilidad en los datos es mucha, también indica que también tenemos mayor control. En los momentos que se percibió una baja calidad del aire, con la correcta ventilación se pudo mejorar este estado. Por tanto, con una higiene correcta en cada uno de las habitaciones, se puede mejorar la calidad del aire media.

6.2.3. Calidad del aire en las aulas

La finalidad de este proyecto era poder obtener datos de las aulas de la facultad central de la Universitat de Barcelona. Finalmente, se ubicaron tres sistemas con módulos de sensores en la facultad de matemáticas. Por problemas técnicos se desestimaron los datos del despacho EB5, ya que un mal funcionamiento del módulo CCS811 impidió que se pudieran mostrar datos de calidad. Los datos más interesantes los encontramos en el aula B2, donde debido a su tamaño y mayor fluctuación de personas se considera que los datos tienen más estima para este caso. En la figura treinta se observan los datos de dióxido de carbono tomados el día ocho de junio de dos mil veintitrés.

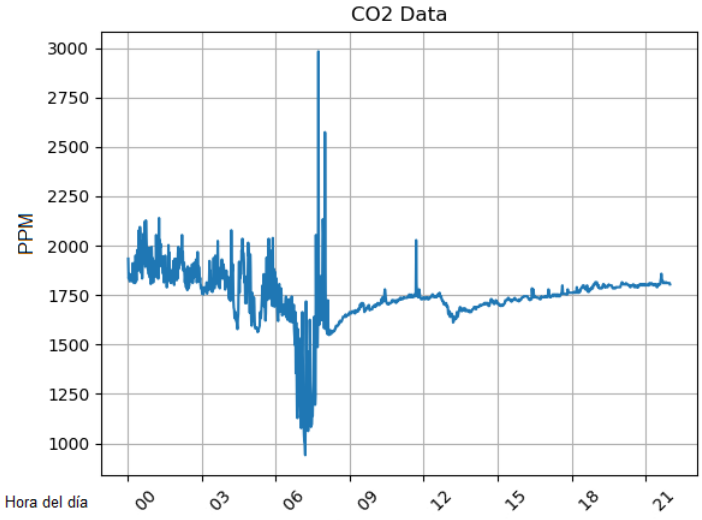


Figura 30: Dióxido carbono en el aula

En la gráfica de la figura treinta se observa una mayor constancia en los datos obtenidos. Esto podría ser debido, entre otras cosas, a que la capacidad de ventilación en las aulas de la facultad es mayor. Además, las actividades que se realizan en el aula, en este caso exámenes, son actividades sedentarias y por tanto no disparan los niveles del dióxido de carbono. Aún así, se ve una variabilidad muy grande al inicio de la jornada laboral, donde el aula inicia su ventilación después de estar cerrada durante la noche y, el resto del día, donde usualmente el aula se cierra. En general, los niveles de dióxido de carbono que se han tomado son nocivos para los alumnos y se debería mejorar este aspecto de cara al futuro.

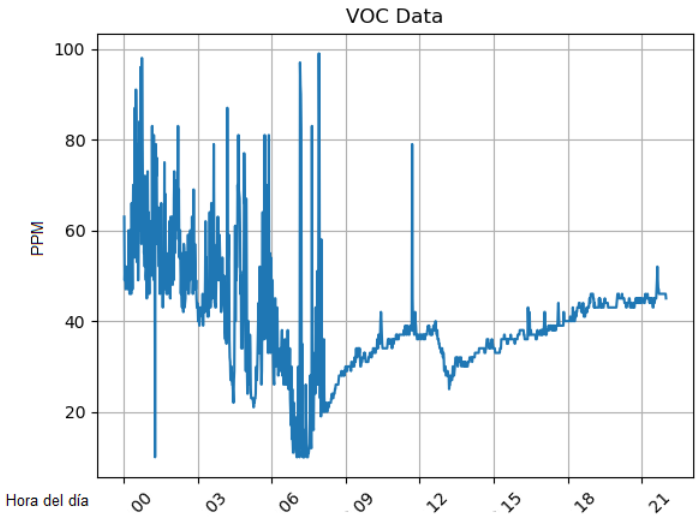


Figura 31: Nivel de compuestos orgánicos volátiles en el aula

La gráfica de la figura treinta y uno muestra los niveles de COV que, aunque, aumenta o disminuye a la par que el dióxido de carbono en sus mayores picos, se puede apreciar como durante la noche este ha aumentado considerablemente. Los niveles de COV también son nocivos para el alumnado en general, mostrando niveles elevados durante todas las etapas del día.
A partir de esta información, podemos concluir que, a pesar de poder conseguir niveles moderados de dióxido de carbono en el aula, se contempla una estabilidad en un rango nocivo para los usuarios de esta. Con una ventilación adecuada, se cree que se podría conseguir niveles más bajos y mantenerlos estables durante todo el día y que, debido a la propia actividad que se suele desarrollar en el aula, es difícil que haya momentos en los que la calidad del aire empeore rápidamente.

6.2.4. Relación entre COV y CO2

En las secciones anteriores se ha podido apreciar como existe una fuerte relación entre el COV y el CO2 en los datos mostrados. Para evitar confusiones en las conclusiones que se puedan crear a partir de este fenómeno, se propone explicar de manera resumida posibles razones por las cuales pueda existir esta correlación.
El COV y el CO2 son dos gases diferentes con propiedades y efectos distintos. Por un lado, los COV son compuestos químicos que contienen carbono y se evaporan fácilmente a temperatura ambiente. Por otro lado, el CO2 es un gas compuesto por un átomo de carbono y dos átomos de oxígeno. Una vez diferenciados, podemos proceder a extraer hipótesis de porque ambos tienen una correlación tan marcada.
Una de las razones en consideración podrían ser que ambos tienen fuentes comunes. La combustión puede generar tanto COV como CO2, siendo transportados por el aire de manera equivalente. Otra de las razones podría ser que las actividades humanas, en muchos momentos como por ejemplo la limpieza con uso de productos químicos, el uso de calefacción o cocinas de gas provoca que ambos gases se encuentren en el aire.
Debido a los lugares de obtención de datos, las conclusiones que se pueden sacar sobre esta correlación provienen de una mezcla de ambas teorías aquí expuestas. Diversas fuentes de polución acaban provocando que no sólo se lance dióxido de carbono si no otros compuestos nocivos, de manera que estos acaban viajando por el aire de manera igual y, por tanto, acabe existiendo una correlación entre ellos debido a que las corrientes de aire en las que se contienen estos gases hace que llegue este aire poluto a los sensores.

6.2.5. Conclusiones sobre los datos extraídos

En los anteriores apartados, hemos podido ver como, a pesar de que los datos se tomaron en diferentes lugares, son bastante similares entre ellos. La proximidad de ambos lugares, a pesar de la expuesta calidad del aire en las dos ciudades, no parece haber afectado de manera destacable en la toma de las muestras.
A pesar de que en ningún momento, en los valores más bajos, se puede apreciar una calidad del aire apta, si que se ha podido ver como con una ventilación adecuada puede paliar esta carencia. Los datos tanto de CO2 como COV en los lugares donde se han extraído los datos, indicarían que, las personas que se ubiquen en estos lugares durante un tiempo prolongado podrían sufrir signos de cansancio o ansiedad únicamente por la calidad del

aire.
Por otro lado, y como dato añadido, también se quería hacer una consideración a la temperatura en las aulas durante los exámenes. En la figura treinta y dos podemos ver como las temperaturas no son las adecuadas si se quiere que los alumnos puedan hacer sus exámenes de una manera óptima.
Según el Real Decreto 39/1997, las temperaturas en el aula se deberían

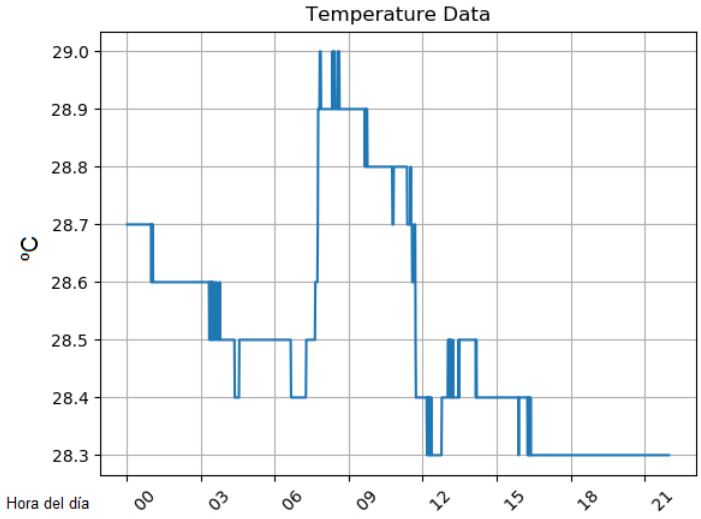


Figura 32: Temperatura en el aula

comprender entre diecisiete y veintisiete grados. Aunque la diferencia entre la temperatura del aula y la que aplica la ley no es notoria, este sistema fue creado para, precisamente, facilitar que en casos como estos, se pueda adaptar de manera sencilla las condiciones del entorno.
Se concluye pues, con estos datos, que a pesar de que la calidad del aire por si mismo tiende a ser nociva en los lugares en los que se han tomado los datos, se pueden tomar medidas y acciones para, con esta información, mejorar la calidad de vida de los usuarios de estos.

7. Viabilidad económica

En el contexto de este proyecto, es esencial analizar y comprender los costes económicos asociados a su implementación. Considerar los gastos es fundamental ya que supone poder realizar una reflexión acerca de si este producto podría ser lanzado al mercado con éxito. En esta sección, se exa-

minarán los costes económicos implicados en la realización de este trabajo en el ámbito académico y el que podría conllevar en el ámbito empresarial. Por ello, se mostrarán los costes que se han asimilado durante el desarrollo del proyecto como trabajo de final de grado y también se mostrará que coste tendrían si se optimizara y se contara con lo necesario para el ámbito empresarial.
Los precios han sido estimados durante la estación primaveral del año dos mil veintitrés. Tanto los precios de los componentes como los de los productos que tienen similitud con el sistema del proyecto están actualizados en mayo de ese mismo año.

7.1. Análisis de costes y presupuesto para el desarrollo

En esta sección se verán los costes que se han asumido durante el desarrollo del trabajo de final de grado por cada circuito que contiene el microcontrolador y los sensores. Se han estimado el precio de los pines y del cableado que se ha utilizado, dividiendo el valor total de los pines por el número de pines usados, haciendo lo mismo con el cableado, estimado en euro por metro. El precio por circuito se estima en el siguiente cuadro.

Componente	**Descripción**	**Coste por unidad (€)**
Adafruit Feather HUZZAH with ESP8266	Módulo ESP8266	14.95
AmbiMate Sensor Module	Sensor AmbiMate	39.90
CCS811	Sensor CCS811	15.51
Protoboard	Protoboard	3.33
Cableado (Gasto aproximado por sistema)	Cableado	0.87
Pines	Pines	0.24
Total		74.80

En el caso de este proyecto, se han utilizado hasta cuatro sistemas como estos, pero finalmente se han implementado tres. También se debería añadir el gasto de la Raspberry Pi y la suscripción al servicio en la nube de AdafruitIo, de manera que se pudiera trabajar con el máximo de prestaciones.

En el cuadro uno se pueden apreciar los costes finales que ha tenido el proyecto para que pudiera funcionar inicialmente.

Componente	Unidades	Coste por unidad (€)	Coste total (€)
Adafruit Feather HUZZAH	3	14.95	44.85
AmbiMate Sensor Module	3	39.90	119.70
CCS811	3	15.51	46.53
Protoboard	3	3.33	9.99
Pack de pines	3	0.24	0.72
Raspberry Pi 4	1	62.90	62.90
AdafruitIo	1	10.00	10.00
Batería de Litio	3	9.99	29.97
Coste Total			324.66

Cuadro 1: Costes totales del proyecto en euros.

7.2. Análisis de costes y presupuesto para el mundo empresarial

En esta sección, se pretende simular el precio para el desarrollo e implementación en el sector. La explicación se dará desde un punto de vista en el cual el propio alumno que hace este proyecto es el que trabaja por cuenta propia, sin mano de obra añadida. Se empezará por los costes de alquiler para un lugar para el desarrollo. Teniendo en cuenta que no es necesario mucho espacio para la creación y la programación de todo el sistema, podemos optar por un lugar pequeño. Ya que no es necesario que el lugar de trabajo se encuentre en una capital de provincia, podemos optar por el alquiler de un local en la periferia desde donde podamos visitar rápidamente a clientes para la instalación pero se minimicen los costes del alquiler. Un local de estas características podría suponer un coste que comprende el rango de entre ciento cincuenta y dos cientos euros, según el portal de inmuebles idealista. Sumados a los costes de electricidad estimados y a la aclimatación inicial, el coste mensual se expresa en el cuadro dos.

Concepto	Coste (€)
Alquiler local	175
Coste de luz	70
Aclimatación inicial	200
Total	445

Cuadro 2: Costes adicionales del proyecto en euros.

Podemos obviar el coste de la soldadura de los elementos. El precio del estaño es muy barato y la electricidad que consume el soldador ya podría venir incluido con los costes anteriores. Una vez que los programas ya se han creado, tanto la aplicación móvil como los necesarios para que funcione el sistema de control ambiental, no demanda mucho tiempo la implementación de un nuevo sistema. Se pueden cargar los programas correspondientes, copiando la tarjeta de memoria de la Raspberry Pi y cargando el programa en cada ESP8266. Futuras iteraciones en estos dos programas se podrían considerar como costes extra. El montaje, total de un sistema, se debería estimar en función del número de salas y, por tanto de microcontroladores que se necesitan instalar.
La pretensión será prestar un producto para pequeña y mediana empresa, donde se puede amortizar el precio de la Raspberry Pi si se instala en un lugar donde se necesita un control en muchas salas próximas, de manera que una unidad de estas puede suplir varios ESP8266 para que se puedan abaratar costes. En un hogar, por ejemplo, no sería interesante si sólo se quiere monitorizar una sala.
Para ejemplificar y poder calcular costes, se va a exponer el caso en el que, dada una pequeña empresa, se quieren cubrir seis salas para medir la calidad del aire. También, suponemos el escenario ideal donde todo el sistema puede tener alimentación directamente del sistema eléctrico y no es necesario el uso de baterías y, por tanto, no hemos de aumentar costes por posibles recambios.

Componente	**Unidades**	**Coste por unidad (€)**	**Coste total (€)**
Adafruit Feather HUZZAH	6	14.95	89.70
AmbiMate Sensor Module	6	39.90	239.40
CCS811	6	15.51	93.06
Protoboard	6	3.33	19.98
Pack de pines	3	0.24	0.72
Raspberry Pi 4	1	62.90	62.90
Mano de obra	1	60.00	60.00
Coste Total			565.76

Cuadro 3: Costes totales del proyecto ejemplificado.

En el cuadro tres podemos ver como por un precio bastante competitivo se podrían tener monitorizadas todas las salas mediante una sencilla aplicación móvil, comprobando la calidad del aire en tiempo real así como la tempera-

tura y humedad.
Además de los costes de montaje y producción, también se debería tener en cuenta el presupuesto invertido en captación de clientes. Al ser un trabajo por cuenta propia, se debería estimar como mano de obra del propio trabajador el tiempo destinado a la captación. Dado que estos datos son muy susceptibles al trabajo de marketing que se haga, no se pueden asegurar cifras en este ámbito. Se optará por considerar el CAC(Coste de adquisición de cliente) como un coste añadido pero no cuantificable en este caso.

7.3. Comparativa del precio en el mercado

Usualmente, los productos del mercado que son similares a este, tienen una pantalla integrada para poder monitorizar los datos. En este trabajo de final de grado, se obvió esa opción ya que aumentaría los costes y realizaría la misma función que hace la aplicación móvil. Al tener este público objetivo empresarial, se considera más práctico el poder acceder a todos los datos de todas las salas desde una misma aplicación.
Partiendo de esta premisa, se pueden encontrar rápidamente por internet tres productos de funcionalidad similar, mostrados en el cuadro cuatro.

Producto	**Precio (€)**
Temtop Medidor de CO2	199,99
Airthings View Radon 2989	199,00
Atmotube Monitor profesional	119,00

Cuadro 4: Productos similares

Destacar que, ofrecen una aplicación móvil para monitorizar los datos el Arthings View Radon y el Atmotube Monitor profesional. Si tomáramos el caso anterior, dada una implementación en múltiples salas, el coste se dispararía para ambos productos, como se muestra en el cuadro cinco.

Producto	**Unidades**	**Precio por unidad (€)**	**Precio total (€)**
Atmotube	6	119,00	714,00
Temtop Medidor de CO2	6	159,99	959,94

Cuadro 5: Productos y precios.

Sin tener en cuenta la mano de obra, vemos como el precio es superior en ambos casos. Además, no son dispositivos preparados para ser monitorizados en conjunto, si no de manera individual. Esto indica que dentro de un mercado en concreto, se podría ser competitivo y ofrecer un control de la calidad del aire de una calidad media sin disparar los costes.

7.4. Comparativa de prestaciones

La comparativa de prestaciones será con el producto AtmoTube Monitor profesional. Estos dispositivos están hechos para ser instalados de manera individual en el ámbito doméstico, en contraposición al producto expuesto en esta memoria que está hecho para el control de diferentes salas de manera simultánea. A nivel de los datos que se recopilan, se presupone, de manera teórica, que este competidor dispone de sensores de mayor precisión debido a su precio, pero es algo discutible y supuesto. Este cuenta con la transmisión de datos vía Bluetooth, sin necesidad de tener un servidor como la Raspberry Pi actuando como tal, añadiendo más simplicidad. También consta con autonomía energética, contrariamente al supuesto sistema expuesto.
A nivel visual, el producto comparado tiene un diseño más funcional y estético, hecho para ser portable y agradable para el cliente. Tomando estos datos, las prestaciones, si no es importante el diseño estético, son similares al nivel de un único sistema.
La ventaja de prestaciones que puede ofrecer el producto con ESP8266 es que da una mayor versatilidad a la hora de cubrir espacios mayores y necesitar un control de sala. Además, da autonomía al usuario para las reparaciones ya que la substitución de piezas no requiere de ningún conocimiento técnico extra una vez se conocen los detalles de como se establecen las conexiones del circuito. En caso de que un sensor individual deje de funcionar, el propio usuario podría adquirir, o pedir a la empresa una pieza nueva con la que cubrir la pieza invalidada. Como un extra, se podrían vender licencias de código con las cuales el cliente obtuviera el código con el que se ha programado el sistema para que este, en caso de que quiera ampliar o modificarlo, con los suficientes conocimientos que se le proporcionarían, pudiera hacerlo. Las otras propuestas están orientadas para un uso personal y doméstico, mientras que para un uso en un lugar de tamaño medio donde se necesite monitorizar de forma constante y sencilla, esta propuesta es adecuada.
En conclusión la diferencia en la prestación de servicios hace que el producto creado en cuestión sea ideal para el nicho concreto que se ha explorado.

8. Conclusiones

Respecto a la idea inicial de este proyecto, se puede ver como se han ido cumpliendo cada uno de los objetivos propuestos. Las limitaciones de presupuesto no han impedido que se haya podido crear este sistema que finalmente ha cumplido con los requisitos que se propusieron al principio. A pesar de que en el mercado existen sensores mucho más precisos, se puede observar como en las diferentes tomas de datos que, aunque no tengamos las mejores herramientas, se han conseguido mediciones que reflejan la realidad. La recolección de datos y el funcionamiento del sistema en general funcionó en el entorno de implementación sin necesidad de ajustar nada de este, lo cual muestra que es un sistema robusto y preparado para poder ser instalado en cualquier entorno que cumpla los requisitos mínimos; una fuente de alimentación y una roseta para la conexión de red por cable.
No obstante, debido a limitaciones técnicas y burocráticas, no se pudo extraer una gran cantidad de datos como se hubiera querido al principio. La vida útil de las baterías o la necesidad de habilitar el entorno para ubicar los sensores durante un largo periodo de tiempo impidió poder habilitar la recogida de datos en la Universitat de Barcelona durante más tiempo. Por ello, se considera fructífero el resultado final en el entorno de implementación pero se podría estudiar como habilitar estos sensores de manera permanente.
Por otro lado, se podría haber mejorado la presentación del dispositivo, por ejemplo, mejorando su diseño para poder ser ubicado fácilmente en las paredes, mediante una carcasa diseñada y visualmente atractiva para los usuarios. También, en lugar de usar una protoboard para montar todo el circuito, se podrían haber soldado los componentes en otro tipo de placas por tal de darle más estabilidad y robustez a todos los componentes físicos.
Desde el inicio este trabajo supuso un gran reto. Crear sin ningún tipo de información previa este tipo de sistema supone un esfuerzo tanto de investigación como de desarrollo. Las directrices eran claras, pero buscar información sobre todos los componentes y realizar un software que se adapte a tus necesidades era dar vida a una idea. El crear este tipo de sistemas requiere, a diferencia del software, de un trabajo manual y de muchas pruebas de campo. El estar trabajando con estos dispositivos y tener que observar su comportamiento bajo diferentes estímulos me pareció muy enriquecedor.
En definitiva, este trabajo me sirvió para darme cuenta de que aún queda mucho por hacer y descubrir, que siempre puede existir una necesidad para muchos usuarios y, que como ingenieros, se debe tener la capacidad para resolver estas.

9. Trabajo futuro

Considero que este trabajo podría ser el propulsor para un sistema mayor y más complejo. Con una primera iteración que asegura la robustez del sistema a pequeña escala y una obtención de datos fiables, se podría ampliar este proyecto por tal de implementar un sistema de control ambiental en muchas facultades de la Universitat de Barcelona.
Extender este proyecto en el futuro podría suponer cosas como un mayor rango de cobertura que abarcara más dispositivos, el uso de machine learning para hacer predicciones a tiempo real o poder cuantificar el número de alumnos y automatizar tareas como la distribución de las aulas en exámenes. En resumen, hacer la vida de los alumnos y profesores más sencilla mediante una monitorización de las actividades. Por supuesto, también considero que el sistema se debería adaptar a las personas y, en ningún caso, crear dilemas morales o éticos que puedan crear la sensación de control que puede tener un sistema como este. Nuestra relación con la tecnología debe suponer una convivencia consensuada, no un control desmedido.

Referencias

[1] Andrew Banks and Rahul Gupta, *MQTT Version 3.1.1*, 2014, http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/mqtt-v3.1.1.html

[2] AirDfRobot, *CCS811 Module Specifications*, https://wiki.dfrobot.com/Gravity:%20CCS811%20Air%20Quality%20Sensor%20SKU:%20SEN0318

[3] TE Connectivity, *AmbiMate Sensor Module Specifications*, 2018, https://www.ttieurope.com/content/dam/tti-europe/manufacturers/te-connectivity/resources/ENG_SS_114-133092_E.pdf

[4] Ministerio para la transición ecológica y el reto demográfico, *Visor de calidad del aire*, https://www.miteco.gob.es/es/calidad-y-evaluacion-ambiental/temas/atmosfera-y-calidad-del-aire/calidad-del-aire/visor

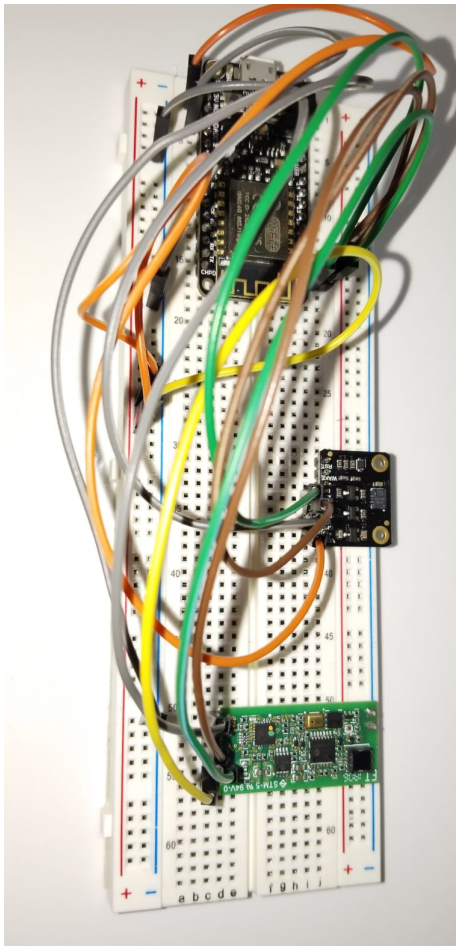
[5] Nicola Jones (Yale School of Environment), *How the World Passed a Carbon Threshold and Why It Matters*, 2017, https://e360.yale.edu/features/how-the-world-passed-a-carbon-threshold-400ppm-and-why-it-matters

[6] Ministerio de empleo y seguridad social, *Jornada y horarios de trabajo*, https://www.insst.es/documents/94886/96082/Jornada+y+horarios+de+trabajo.pdf/ad9dd0c7-80c7-4cff-9b42-afeffd7b934f?t=1560049185620

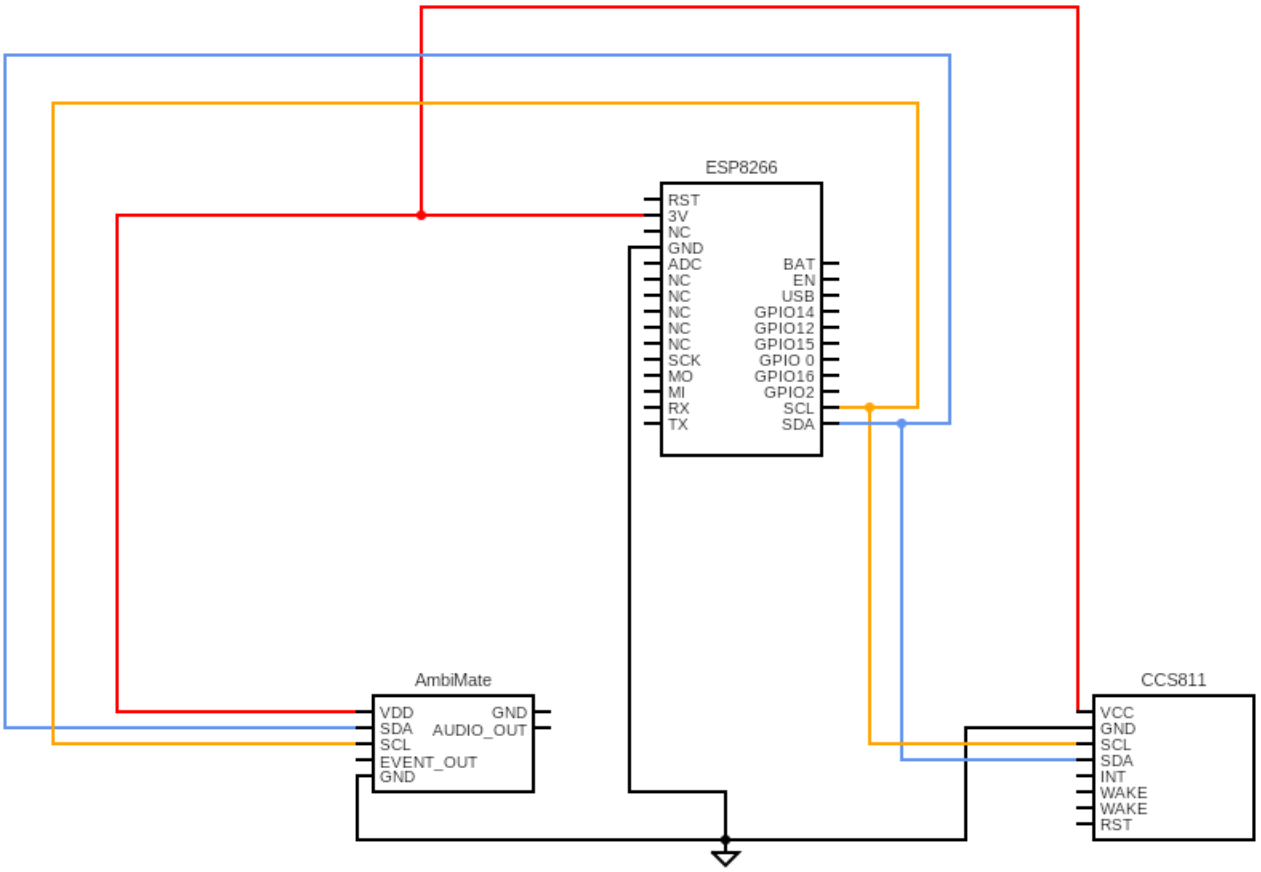
[7] Elaine Fuertes, *Traffic-related air pollution and hyperactivity/inattention, dyslexia and dyscalculia in 2 adolescents of the German GINIplus and LISAplus birth cohorts*, 2013, https://spiral.imperial.ac.uk/bitstream/10044/1/59997/10/ENVINT-D-16-01095_R1.pdf

Anexos

Anexo A

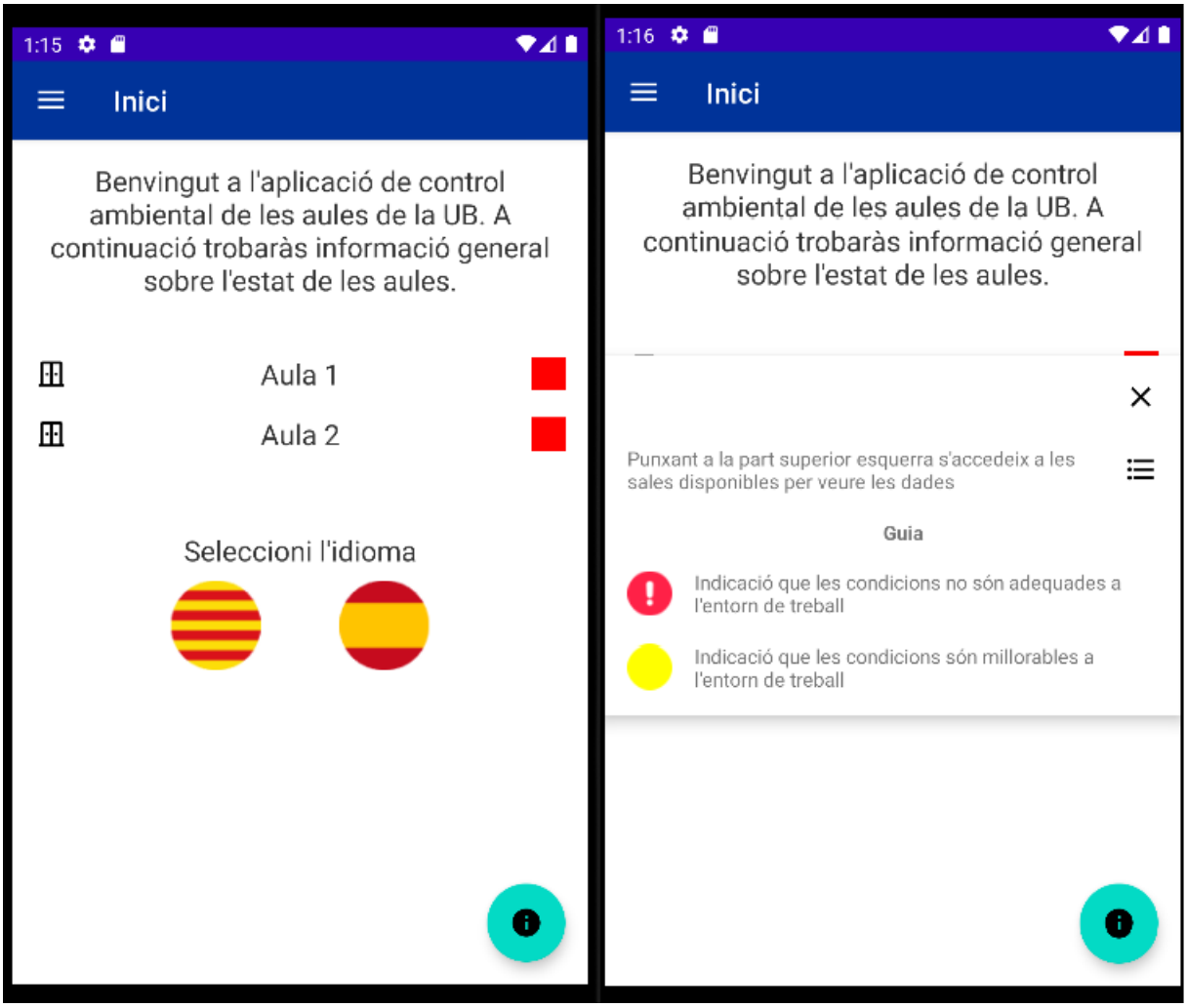


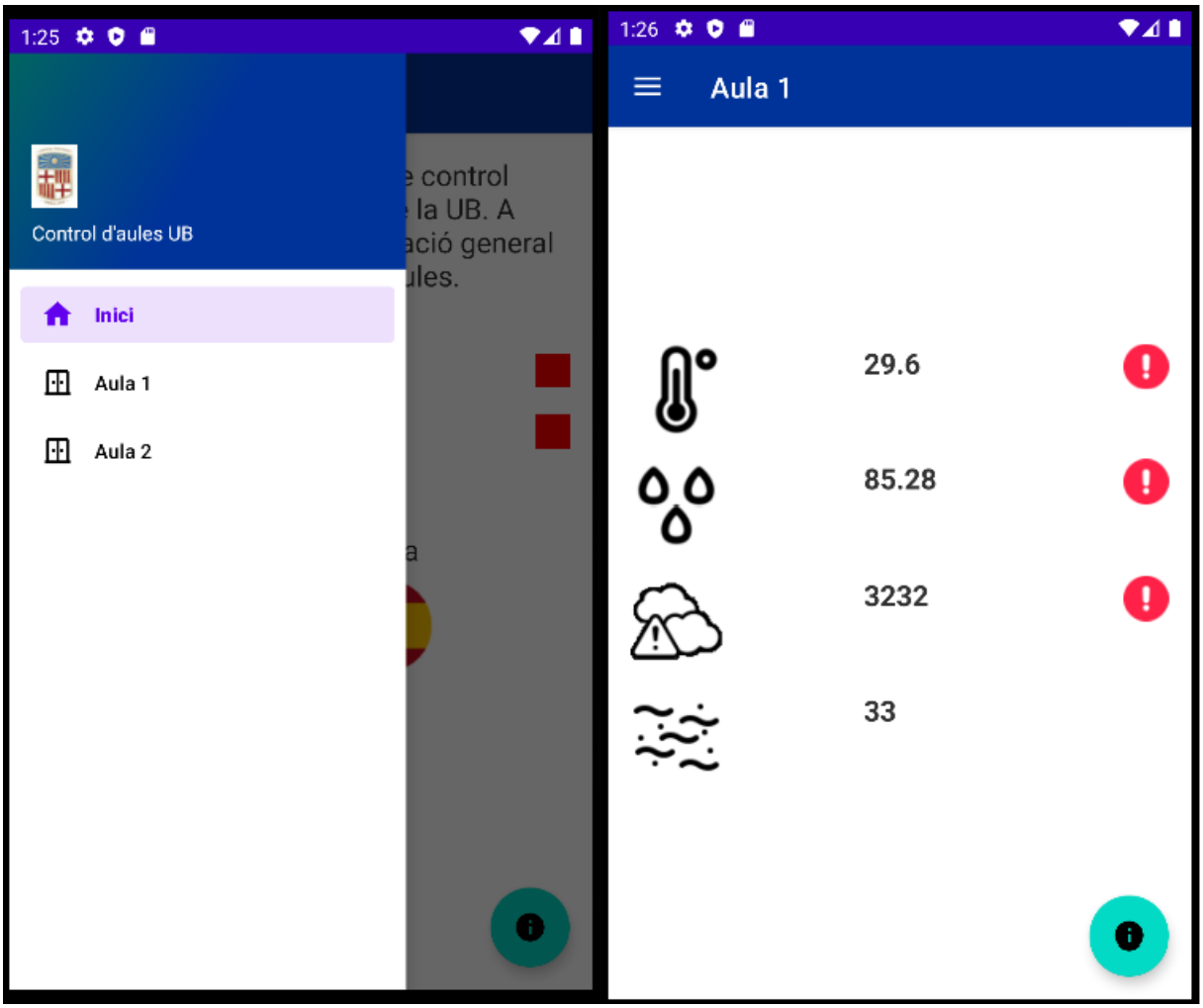
Anexo B


ESP8266
RST
3V
NC
GND
ADC
NC
NC
NC
NC
NC
SCK
MO
MI
RX
TX
BAT
EN
USB
GPIO14
GPIO12
GPIO15
GPIO 0
GPIO16
GPIO2
SCL
SDA
AmbiMate
VDD
SDA
SCL
EVENT_OUT
GND
GND
AUDIO_OUT
CCS811
VCC
GND
SCL
SDA
INT
WAKE
WAKE
RST

Anexo C


1:24
Inici de sessió
Correu electrònic
Contrasenya
INICIAR SESSIÓ


1:15
Inici
Benvingut a l'aplicació de control ambiental de les aules de la UB. A continuació trobaràs informació general sobre l'estat de les aules.
Aula 1
Aula 2
Seleccioni l'idioma
1:16
Inici
Benvingut a l'aplicació de control ambiental de les aules de la UB. A continuació trobaràs informació general sobre l'estat de les aules.
Punxant a la part superior esquerra s'accedeix a les sales disponibles per veure les dades
Guia
Indicació que les condicions no són adequades a l'entorn de treball
Indicació que les condicions són millorables a l'entorn de treball


1:25
Control d'aules UB
Inici
Aula 1
Aula 2
1:26
Aula 1
29.6
85.28
3232
33