UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

Diseño e implementación de una plataforma Web para recolección y visualización de datos en redes de sensores basados en MQTT

TESIS

Que para obtener el título de

Ingeniero en Telecomunicaciones

P R E S E N T A N

Flores Cruz Karina Alejandra

López Ortiz Isaías

DIRECTOR DE TESIS

Dr. José Jaime Camacho Escoto

Ciudad Universitaria, Cd. Mx., 2023

Agradecimientos

Gracias a la Universidad Nacional Autónoma de México, por brindarme una educación dentro de sus aulas, siendo el lugar en donde más he madurado y crecido, donde se me permitió conocer a diferentes personas y formar grandes lazos.

A mis sinodales, por sus contribuciones y el tiempo que dedicaron para revisar nuestro trabajo, a pesar de las actividades que los ocupan.

A mis profesores, por su paciencia y empeño para compartirme su conocimiento. Al Dr. José Jaime, por buscar el mejor desenvolvimiento académico y apoyarnos con el desarrollo de nuestra Tesis; Dr. Oleksandr, por enseñarme con tanta dedicación temas que creí imposibles de comprender y Profesor Raúl Mora, por la pasión e interés que tiene para formar grandes profesionistas, así como introducirlos al mundo laboral.

A mis amigos Juan, Jesica, Erick, Raúl y Humberto, con quienes compartí risas, noches en vela; me escucharon y ayudaron en diferentes aspectos de mi vida.

A mi tía Emma, por brindarme una segunda casa.

A mi padrino Alberto, por estar para mí cuando más le necesite.

A mi hermano, Jorge, por apoyarme con mis estudios, mostrarme el camino, motivarme para dar más de mí y principalmente por su cariño y protección.

A mis padres, Elvira y Raymundo, quienes son mi principal motivo para seguir adelante, les agradezco por haberme forjado como la persona que soy actualmente; brindarme su apoyo incondicional y hacer lo imposible para entenderme y ayudarme. Agradezco cada una de las miles muestras de amor y cuidado que me han regalado, espero algún día retribuir un poco de lo mucho que me han dado.

A mi abuelita, Amparito, quien es recordada con cariño y está presente en los corazones de todas las personas que la conocieron. Le agradezco por todo su tiempo y amor, sé que siempre estará conmigo.

Flores Cruz Karina Alejandra

Quiero aprovechar este capítulo para hacer mención especial de 9 personas.

Mis padres, que cada quien de una forma diferente, estuvieron en todo momento a mi lado y al pendiente de todo lo que necesitara, brindandome de la educación y los valores que tengo hasta este momento, de un cariño infinito e incondicional, y también de todos los recursos necesarios para poder gozar de una gran calidad de vida en todos los aspectos que se puedan considerar.

A mis hermanos, que aunque en el momento de mi educación universitaria ya tenían sus propias responsabilidades laborales y personales, siempre estuvieron al pendiente de mí y apoyándome en todo lo que llegara a necesitar.

A mi novia Karen, quien desde el momento en que comenzamos a hablar, nunca me dejó de lado y siempre sirvió de motor para seguir avanzando y sentirme motivado incluso en los momentos más difíciles y estresantes.

A mis tíos y primos, que de igual forma siempre se preocuparon por mi bienestar e integridad, sobre todo cuando mis padres no estaban, en todo momento y sin tener que pedírselo, se encargaban de que en todo momento me encontrara bien y no me faltara nada.

López Ortiz Isaías

Índice general

1 Introducción

1.1. Planteamiento del problema

Con el inicio de la pandemia de COVID-19, inevitablemente se tuvieron que empezar a tomar medidas de prevención y seguridad para así mermar la propagación del virus. Debido a esto, los espacios conglomerados tuvieron que parar actividades por completo, representando un enorme problema para la sociedad, llegando a un punto tal donde urgía implementar una solución para reabrir dichos espacios.

Una de las medidas tomadas para reabrir los espacios conglomerados fue el control de aforo en lugares cerrados, de manera que la concentración de CO_2 en dichas áreas siempre se mantuviera en un valor adecuado, asegurándonos así que cada persona se encuentra respirando aire que no ha sido respirado por alguien más. Para tener conocimiento certero acerca de la concentración de CO_2 existen diversos dispositivos medidores, así como plataformas web que nos brindan la posibilidad de graficar en tiempo real los datos de cualquier sensor utilizado; no obstante, la gran mayoría de sensores que se encuentran de manera comercial únicamente permiten visualizar el valor medido en tiempo real, lo que nos obliga a estar observando todo el tiempo los datos medidos para poder verificar la concentración de CO_2 que se tiene, es por eso que, para este tipo de plataformas también es esencial poder almacenar los datos medidos para que en cualquier momento estos puedan ser consultados.

Es importante mencionar que, aunque ya existen sistemas medidores de CO_2, estos tienen un precio considerablemente alto tanto para adquirir el sensor como para el servicio de la plataforma. Asimismo, ninguna plataforma web gratuita dedicada a la recolección y visualización de datos proveídos por sensores tiene un amplio espectro de funcionalidades disponibles, pues a pesar de sí poder registrar o eliminar dispositivos y observar los datos en tiempo real, no tienen posibilidades de descargar datos antiguos en diversos formatos, aparte de que ninguna de estas es de código abierto, por lo que todo usuario que las utilice se ve limitado en cuanto a las aplicaciones que podría darle a la plataforma.

1.2. Justificación

El proyecto descrito en esta tesis resulta relevante en varios aspectos. Primero, como ya se mencionó previamente, ante la situación actual de COVID-19 resulta imprescindible tener un control de la concentración de CO_2 en los establecimientos cerrados, así como poder acceder a datos antiguos sin tener la necesidad de estar siempre observando el sensor para conocer los valores medidos.

La disponibilidad de datos antiguos es fundamental para sistemas de control de sensores. Por ejemplo, para los sistemas medidores de CO_2 es indispensable conocer de manera certera el nivel de concentración a lo largo del día, pues al no siempre poder observar el valor medido por el sensor, habrá momentos donde se ignorará si se sobrepasaron los limites recomendados. Se menciona que si la concentración de CO_2 supera las 800 ppm, el 1 % del aire que respiramos, ya ha sido respirado por alguien más.

A pesar de tal urgencia, hasta el momento no se encontró ninguna plataforma gratuita de código abierto que nos permitiera visualizar mediciones antiguas de un sensor, lo que dificulta el adecuado control de la concentración de CO_2 en una habitación. Ya que si en algún momento se sobrepasaron los límites de concentración de CO_2 y no se consultó el sensor, sería imposible alertar de esto a las personas que se encontraban en la habitación, exponiéndolos así a contraer el virus.

En ciertas ocasiones es conveniente tener una plataforma que pueda ser personalizada por el usuario según las prioridades y necesidades que este tenga, es por eso que decidimos hacer a nuestra plataforma de código abierto, de forma que esta tenga un sinfín de posibilidades ante los usuarios que deseen descargarla.

Es importante mencionar que, aunque este proyecto en primera instancia fue enfocado en medidores de CO_2, la plataforma puede funcionar para cualquier sensor que se desee utilizar. De esta forma, se expanden las posibilidades de implementación de la plataforma web desarrollada, por ejemplo, aparte de concentración de CO_2 podría utilizarse para sensores de temperatura, humedad, sonido, proximidad, iluminación, movimiento, entre otros.

Desarrollar una plataforma con las características antes mencionadas brinda un gran beneficio social, pues, por un lado, es útil para todo negocio, oficina o escuela el poder tener un adecuado monitoreo de algún parámetro de interés en el establecimiento. Y por otro lado, el sensor colocado que muestra los datos en tiempo real también brinda mayor confianza a los usuarios que estén dentro del establecimiento, ya que en todo momento ellos tienen la posibilidad de conocer el nivel del parámetro de interés.

1.3. Objetivos

Objetivo general:

Diseñar e implementar una plataforma IoT de código abierto para la recolección y visualización de datos proveídos por sensores, así como comprobar el funcionamiento de esta con un circuito semáforo medidor de CO_2.

Objetivos específicos:

- ➤ Implementar un circuito semáforo medidor de CO_2.
- ➤ Realizar un programa basado en el protocolo MQTT para recibir los datos medidos por el sensor de CO_2 y subirlos a una base de datos.
- ➤ Diseñar una plataforma web que pueda graficar en tiempo real los datos medidos por el sensor colocado; consultar historial de datos antiguos para poder copiarlos, imprimirlos o descargarlos en diversos formatos (CSV, Excel o PDF); y poder agregar, eliminar o restaurar sensores.
- ➤ Subir el código correspondiente a GitHub, de forma que esta sea de código abierto y así cualquier usuario que lo desee pueda acceder a él.
- ➤ Agregar al circuito una batería y una celda solar para que el circuito pueda funcionar de manera más autónoma.

1.4. Contribuciones

Socialmente hablando, el sistema desarrollado es de gran impacto, pues el simple hecho de saber que una habitación se encuentra con un nivel adecuado de CO_2, brinda de una gran seguridad a aquellos que se encuentren dentro de ella. Asimismo, al conocer estos datos y poder realizar las acciones pertinentes para tener una correcta ventilación, se prevé en gran medida la dispersión de los virus que se transmiten a través del aire.

Este proyecto también permite que múltiples miembros de la sociedad resulten beneficiados. El controlar la calidad del aire en un establecimiento puede proporcionar un lugar seguro para sus usuarios. De esta forma, ellos tendrán un mayor sentimiento de tranquilidad al entrar a alguna habitación, beneficiando a dueños de establecimientos y negocios para que tengan un mayor número de clientes, y que paulatinamente se tenga un aumento de aforo en espacios cerrados.

Respecto a los desarrolladores de protocolos, se les brinda una plataforma web de la cual puedan partir para desarrollar algún estudio de interés. Como la plataforma que presenta este proyecto es de código abierto, los desarrolladores tienen la posibilidad de utilizar, modificar y experimentar a su conveniencia la plataforma y sus funcionalidades. Además, les permitirá a los usuarios analizar en mayor profundidad el funcionamiento de MQTT, protocolo utilizado para poder comunicar el sensor MQ135 con la base de datos.

Otro sector de la población que se encuentra beneficiado en buena medida es el de los desarrolladores web. Al tener una plataforma web de código abierto, pueden descargarla e interactuar con ella, teniendo un mayor aprendizaje sobre desarrollo de plataformas enfocadas a la visualización y almacenamiento de datos proveídos por sensores. Así, se podrá compartir el conocimiento e incitar a los desarrolladores web a que diseñen e implementen sus propias plataformas.

2 Sustento teórico

2.1. Microcontroladores

Un microcontrolador es un circuito integrado que contiene los mismos componentes de una computadora, es decir [1]:

- ➤ Unidad de Procesamiento Central (CPU).
- ➤ Memoria volátil para datos temporales (RAM).
- ➤ Memoria no volátil para escritura de programa (ROM, PROM, EEPROM).
- ➤ Puertos de entrada y salida para comunicarse con dispositivos externos.
- ➤ Periféricos (comunicación serial, temporizador, convertidor A/D, entre otros).

Debido a la gran variedad de características que posee el microcontrolador, este resulta en una extensa variedad de aplicaciones en diversos sectores de la industria. En la actualidad existen una gran cantidad de alternativas tanto para uso personal como académico o industrial. Entre los más conocidos podemos mencionar los diferentes modelos de Arduino, Raspberry Pi, ESP32, entre otros.

2.1.1. Arduino

David Cuartielles y Massimo Banzi crearon Arduino como una plataforma de hardware libre, la cual está basada en un entorno de desarrollo integrado (IDE) y un microcontrolador [2]. Debido a sus cualidades, Arduino ha promovido la incursión de principiantes en proyectos electrónicos.

Es importante destacar que el compilador de Arduino es el que permite la programación de alto nivel, pues es quien realiza todas las operaciones lógicas y matemáticas, así como la gestión de recursos para cualquier componente externo que conectemos a la placa. Un punto clave en el hardware de Arduino son los pines analógicos y digitales de entrada y salida que este tiene, pues permiten agregar nuevas funcionalidades sin la necesidad de alterar la anatomía de la placa principal, esto mediante la conexión de dispositivos externos.

Una de las placas Arduino más populares se muestra en la figura 2.1, la cual corresponde a una tarjeta Arduino Uno. [3]Algunos de los componentes a destacar de la tarjeta son los siguientes:

- 14 pines de entrada/salida.
- 6 entradas analógicas.
- Conexión USB y jack de alimentación.
- Botón de reinicio.

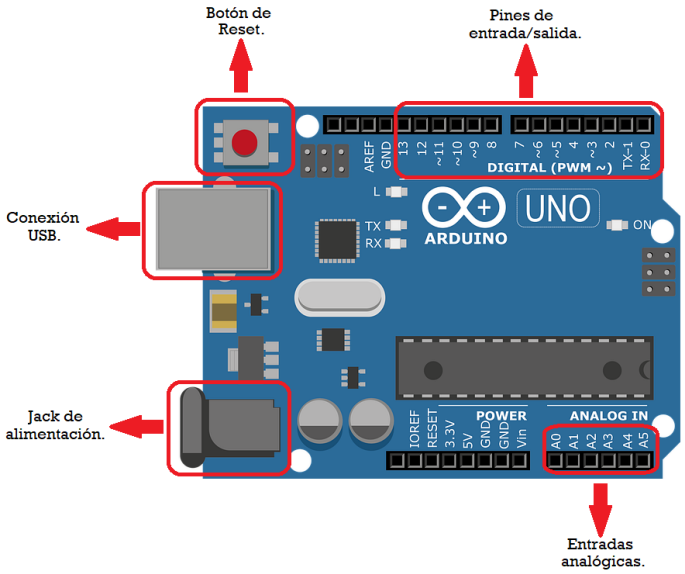

Figura 2.1: Arquitectura de placa Arduino (Recuperado de https://cdn.pixabay.com/photo/2017/03/23/12/32/arduino-2168193_ 480.png el 19 de octubre de 2022).

2.1.2. ESP32

ESP32 es una familia de microcontroladores desarrollada por la empresa Espressif Systems. Esta familia de microcontroladores puede resultar muy superior a la placa Arduino Uno, ya que cuenta con características clave, siendo su principal ventaja la capacidad de comunicación inalámbrica WiFi y Bluetooth. De igual forma, podemos definir al ESP32 como una serie de SoC (System on Chip) y módulos de bajo costo y consumo. [4]

Algunas de las características que diferencian a este equipo de otros microcontroladores son las siguientes, de acuerdo con [5]:

- ***Diseño robusto:*** El ESP32 es capaz de adaptarse a condiciones extremas en diferentes ambientes industriales. Teniendo una temperatura de operación en el rango de -40° C a 120° C.
- ***Muy bajo consumo de energía:*** Al ser diseñado para aplicaciones IoT y dispositivos móviles, el ESP32 cuenta con una combinación de varios softwares patentados que le permiten tener un consumo de energía ultra bajo.

- **Alto grado de integración:** Considerando que el ESP32 incluye interruptores de antena, amplificadores de potencia y recepción de bajo ruido, filtros y módulos de administración de energía, es enorme la variedad de aplicaciones que se pueden tener con esta placa con el mínimo de requisitos extra.

- ***Chip híbrido:*** La ESP32 puede interactuar con otros dispositivos para realizar implementaciones con su funcionalidad de conectividad inalámbrica WiFi y Bluetooth. Esto expande todas las posibilidades que se tienen con este SoC, por ejemplo, para no sobrecargar la capacidad de procesamiento y nivel de batería de una sola placa, se puede hacer que dos dispositivos se comuniquen entre sí mediante WiFi o Bluetooth y haya una partición del procesamiento.

Para visualizar la placa ESP32, se tiene la figura 2.2 donde se pueden observar sus componentes más importantes, los cuales de acuerdo con [6] son:

- 36 pines GPIO (Pines de entrada/salida de propósito general).
- WiFi y Bluetooth integrados.
- 520 Kb de RAM.
- 16 pines ADC (convertidor analógico a digital).
- 2 pines DAC (convertidor digital a analógico).
- Conexión micro USB.

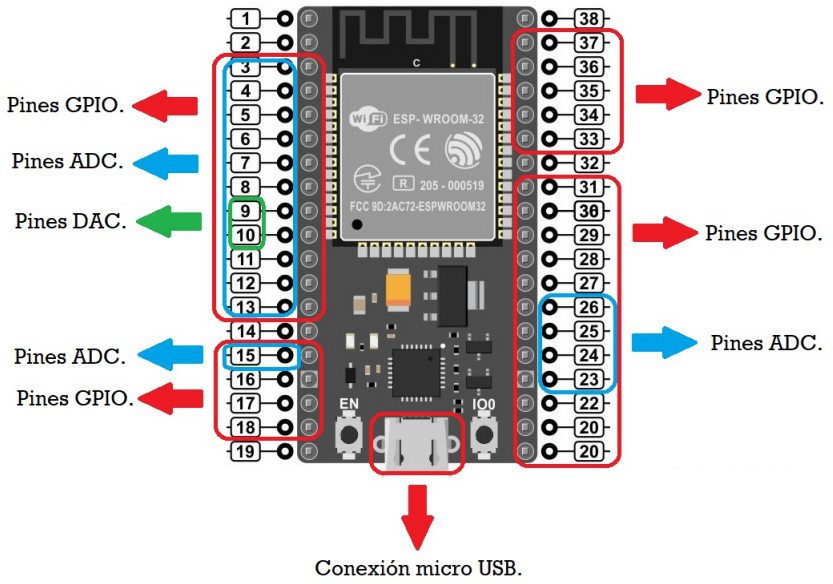

Figura 2.2: Placa ESP32 (Recuperado de https://vasanza.blogspot.com/2021/07/especificaciones-del-modulo-esp32.html el 19 de octubre de 2022).

2.1.3. Centro de carga solar

Un módulo de carga solar está normalmente diseñado para paneles solares de 6[V] a 24 [V], teniendo como función principal recargar una batería de litio de 3.7 [V] a través del panel solar o conexión USB, proveyendo de una salida regulada de 5[V] a 1 [A] [7].

Este módulo se caracteriza por tener un MPPT (Maximum Power Point Tracking) lo que asegura una máxima potencia aprovechada en los paneles solares, así como circuitos de protección. De esta forma se permite una operación de alta eficiencia, estabilidad y seguridad. Este módulo tiene aplicaciones en proyectos de protección, IoT de bajo consumo de energía, y circuitos alimentados por energía solar.

Los principales elementos del centro de carga solar pueden observarse en la figura 2.3, de los cuales podemos destacar:

- ➤ Entrada de alimentación mediante panel solar.
- ➤ Entrada de alimentación mediante USB.
- ➤ Salida USB de 5[V]/1[A].
- ➤ Interfaz de batería recargable.
- ➤ Chips de administración de energía solar y USB.
- ➤ Indicadores LED de la carga de la batería.

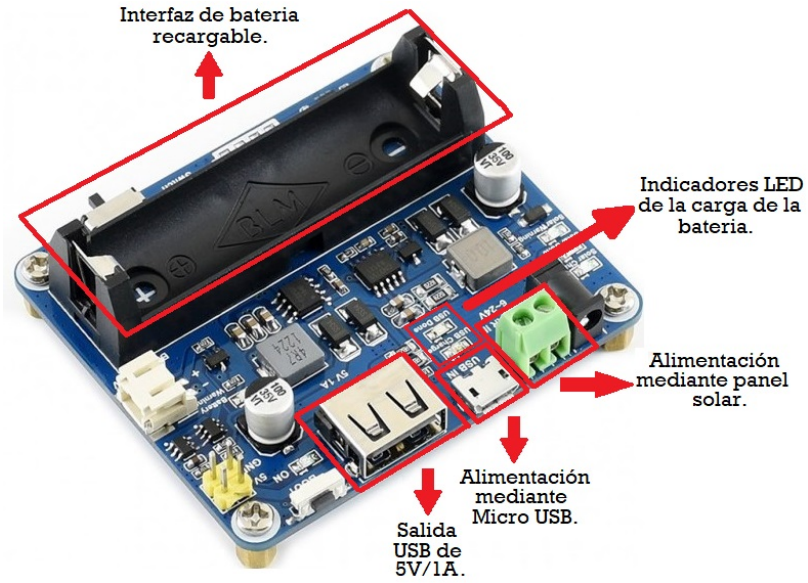

Figura 2.3: Centro de carga solar (Recuperado de https://www.waveshare.com/solar-power-manager.htm el 19 de octubre de 2022).

2.2. MQTT

MQTT [8] (Message Queuing Telemetry Transport) es un estándar de mensajería OASIS (Organization for the Advancement of Structured Information) utilizado en Internet de las Cosas (IoT) y comunicaciones máquina a máquina. Este tiene un diseño de tipo publisher - subscriber ideal para conectar dispositivos de manera remota con huella de código pequeña, que están conectados a redes no fiables y con un mínimo uso de ancho de banda. Hoy en día, este protocolo es aplicado en diversas industrias, tales como la manufactura, telecomunicaciones, entre otros.

MQTT fue creado en 1999 por IBM. Originalmente, este protocolo de mensajería tenía el propósito de que dispositivos de monitoreo utilizados en la industria pudieran enviar mediciones a servidores de manera remota. MQTT nació de la idea de transferir información de monitoreo de un gasoducto a un satélite, de manera que se proporcionaran datos de manera fiable y utilizando un ancho de banda mínimo. Esto resultó ser de gran impacto pues, dado que se trataba de situaciones donde resultaba imposible establecer una conexión de línea fija entre el servidor y el dispositivo de medición, era imperativo tener un medio barato y confiable. [9] En 2013, MQTT fue estandarizado como open source por OASIS. Hasta hoy en día, este protocolo sigue siendo gestionado por dicha empresa.

MQTT se ejecuta sobre el protocolo TCP/IP. En su arquitectura existen dos partes principales: el cliente y el broker. El broker es el servidor que se encarga de comunicar a los clientes entre sí, mientras que los clientes se conectan con dicho servidor para recibir información de otro cliente; donde estos pueden ser un editor (sube información a broker), suscriptor (consulta información perteneciente a un editor) o ambos. A pesar de que el broker puede ser instalado en cualquier PC, ya existen servicios de MQTT instalados en la nube, donde los más populares son Mosquitto y HiveMQ.

Este protocolo está controlado por eventos, es decir, que la comunicación no es continua, sino que sucede cada que un cliente tiene algo que publicar o cada que el broker cuenta con nuevos datos para enviar. De manera que el volumen de transmisión de datos se mantenga al mínimo.

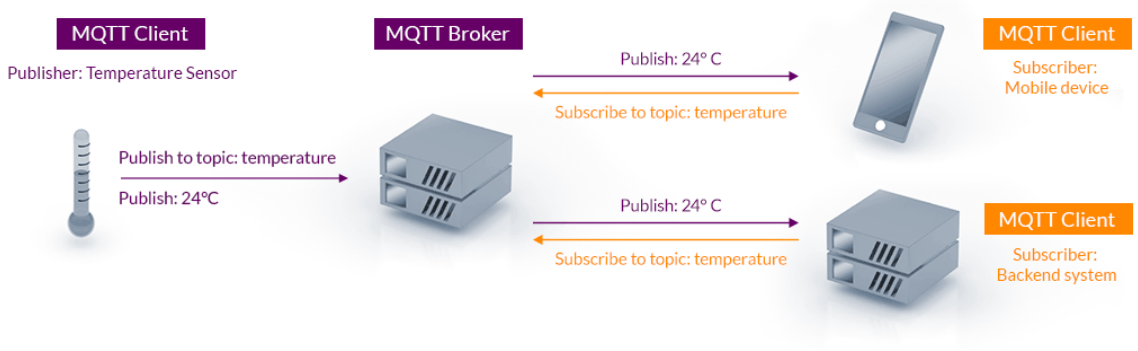

Figura 2.4: Arquitectura de MQTT (Recuperado de https://mqtt.org/assets/img/mqtt-publish-subscribe.png el 19 de octubre de 2022).

En MQTT los mensajes se publican como tópicos. Donde dichos tópicos tienen

estructura jerárquica que utilizan la barra (/) como delimitador, p. ej. Sensores/-CO2/SalonQ308. Permitiendo la categorización de los mensajes que pueden ser enviados o transmitidos, de forma que un mensaje con un tópico en específico solo llegue a los clientes suscritos a él.

La forma de trabajo de los tópicos es que el broker recibe todos los mensajes sin importar el tópico, y reenvía cada uno de ellos únicamente a los clientes suscritos al tópico en específico. Es importante mencionar que, si el broker recibe un mensaje perteneciente a un tópico sin suscriptores, este es descartado, aunque el publisher puede decirle al broker que guarde el mensaje hasta que un cliente se suscriba a él.

En la figura 2.4 se observa un ejemplo de la arquitectura publisher/suscriptor en MQTT, donde un dispositivo móvil y un servidor están suscritos al tópico "temperatura", recibiendo todos los datos proveídos por el cliente, que en este caso es un sensor de temperatura.

Los encabezados de los mensajes en MQTT son de apenas 2 bytes, esto resulta en un ahorro de ancho de banda. Hay 3 niveles de calidad de servicio (QoS), de forma que los desarrolladores de red puedan incrementar la fiabilidad de la conexión.

- QoS 0: También conocido como *Lanzar y olvidar*, se ofrece la menor cantidad de transmisión de datos. Lo que significa que el mensaje se manda al cliente una vez sin ningún tipo de confirmación y asumiendo que una vez enviado la entrega ha sido completada, olvidándolo por completo.

- QoS 1: También conocido como *Entregado al menos una vez*, el broker envía el mensaje una vez y espera respuesta de confirmación del cliente. De manera que, si no se recibe confirmación en un periodo de tiempo, dicho mensaje se reenvía; siendo posible que el cliente reciba varias veces el mismo mensaje si el broker no recibe el mensaje de confirmación a tiempo.

- QoS 2: También conocido como *Entregado exactamente una vez*, el cliente y broker utilizan un enlace de cuatro pasos para que el mensaje solo sea recibido con éxito una única vez.

Las características principales de MQTT se resumen a continuación:

- Ligero y eficiente: Los clientes de MQTT representan una carga de trabajo pequeña, por lo que pueden ser utilizados en microcontroladores con recursos mínimos. Además de que los encabezados de cada mensaje son muy pequeños para optimizar el ancho de banda utilizado.

- Comunicación bidireccional: MQTT permite mensajería de dispositivo a servidor y viceversa, de manera que resulta bastante sencilla la transmisión de mensajes a un grupo de dispositivos.

- Escalabilidad: Tomando en cuenta el crecimiento que hoy en día tienen las aplicaciones con IoT, MQTT resulta idóneo para este tipo de conexiones a millones de dispositivos IoT.

- Mensajería confiable: La confiabilidad en la entrega de mensajes es vital para los casos de uso de IoT. Es por esto por lo que MQTT cuenta con 3 niveles de calidad de servicio dependiendo de la fiabilidad que se requiera.
- Aplicación para redes no confiables: Ya que muchos dispositivos IoT requieren conectarse sobre redes celulares no confiables, MQTT es capaz de realizar sesiones persistentes a estas, de forma que se reduce el tiempo de conexión entre el cliente y el broker.
- Seguridad integrada: MQTT utiliza TLS para cifrar los mensajes y OAuth para autenticar a los clientes.

2.3. Sensor MQ135

El sensor MQ135 es un módulo utilizado para detectar la calidad del aire en un área determinada [10]. Este es un sensor electroquímico, ya que internamente tiene un calentador encargado de aumentar la temperatura del sensor, reaccionando a la presencia de los gases y variando el valor de la resistencia. De igual forma, al comportarse como una resistencia, necesita estar conectado con un resistor de carga R_L para así tener un divisor de voltaje y poder leer el valor detectado. [11]

Debido al calentador que tiene este sensor, es necesario dejarlo cierto tiempo funcionando para que se estabilice y tenga las características que el fabricante muestra en las hojas de datos, dicho tiempo puede ir desde las 6 hasta las 48 horas.

En el mercado, generalmente se suelen encontrar los sensores MQ135 en módulos, de manera que la parte de uso y conexión resulte mucho más sencilla. Así, solo se necesita alimentar el módulo y leer los datos proveídos por el sensor. De igual forma, estos módulos cuentan con una salida digital, la cual, internamente, trabaja con un comparador, y con la ayuda del potenciómetro que viene integrado en el módulo es que se calibra el valor umbral para interpretar la salida como ausencia o presencia de gas.

El MQ135 es idóneo para aplicaciones donde se quiera conocer la calidad del aire, pues con él se puede realizar la detección de gases nocivos en un rango de 10 - 1000 partes por millón (ppm). Este sensor es ampliamente utilizado en sistemas de control de calidad del aire en casas, edificios, oficinas e industrias que trabajan con este tipo de gases. Algunas de sus principales características son:

- Voltaje y corriente de operación de 5[V] y 150[mA], respectivamente.
- Detección de amoniaco, alcohol, óxidos de nitrógeno, benceno, humo y dióxido de carbono.
- Temperatura de operación de -20°C a 70°C.
- Humedad de operación menor a 95 %RH.

En la figura 2.5 se observa el módulo del sensor MQ135, así como sus pines de salida y alimentación.

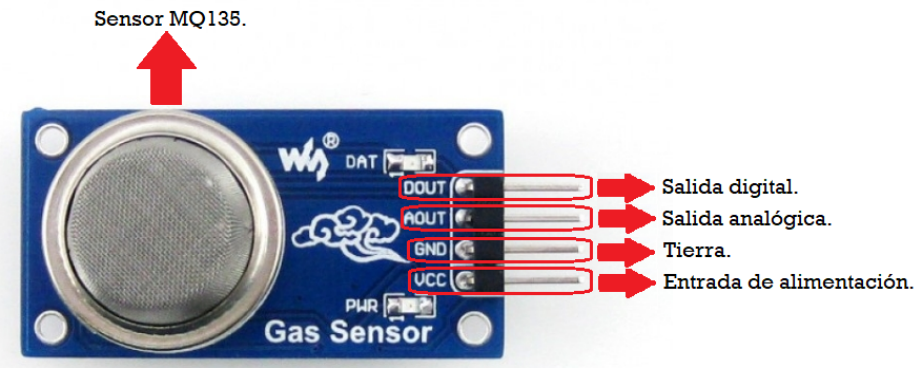

Figura 2.5: Módulo sensor MQ135 (Recuperado de https://www.fasttech.com/product/9642368-waveshare-mq-135-gas-sensor-air-quality-detection el 12 de noviembre de 2022).

2.4. Celdas solares e irradiancia solar

Una **celda solar** es un dispositivo electrónico capaz de capturar la luz solar y convertirla en electricidad. Estas suelen venderse en módulos para formar unidades más grandes, pudiendo incluso estar acopladas a unidades aún mayores para formar paneles solares [12].

El funcionamiento de estas se basa en capturar las partículas de la luz (fotones), de manera que haya un flujo de electrones, lo que se traduce en una corriente eléctrica. Sin embargo, cada celda genera apenas unos cuantos volts de electricidad, por lo que es trabajo del panel solar combinar la energía producida por muchas celdas y tener una cantidad útil de voltaje y corriente. La mayoría de las celdas solares están hechas de silicio, de manera que cuando un rayo de luz incide sobre la celda solar, la energía que este transporta extrae electrones del silicio, siendo obligados a fluir alrededor de un circuito eléctrico y alimentando a los aparatos eléctricos.

En la figura 2.6 se observa un modelo de celda solar comúnmente usado, donde en la vista trasera se muestran los pines de potencial positivo y negativo para la salida de voltaje.

Al realizar un circuito también es importante considerar el suministro autónomo de electricidad. Por ejemplo, un microcontrolador puede recibir electricidad de una celda solar, mientras que esta a su vez puede cargar una batería. De esta forma, la energía utilizada por el microcontrolador en la noche será la de la batería [13].

La **irradiancia solar** es la potencia de la radiación solar por unidad de área, esta se mide en $[\frac{W}{m^2}]$. De igual forma, podemos definir a esta como la magnitud que mide la energía por unidad de superficie de radiación solar incidente en una superficie [14]. Con lo anterior, el desempeño de una celda solar está directamente relacionado con el nivel de irradiancia solar que se tenga, pues a mayor valor de esta, más energía podrá ser aprovechada. Sin embargo, también hay que tomar en cuenta que las celdas solares cuentan con una eficiencia de alrededor del 20 %, por lo que sólo dicha fracción de la energía que reciba será convertida en electricidad [15].

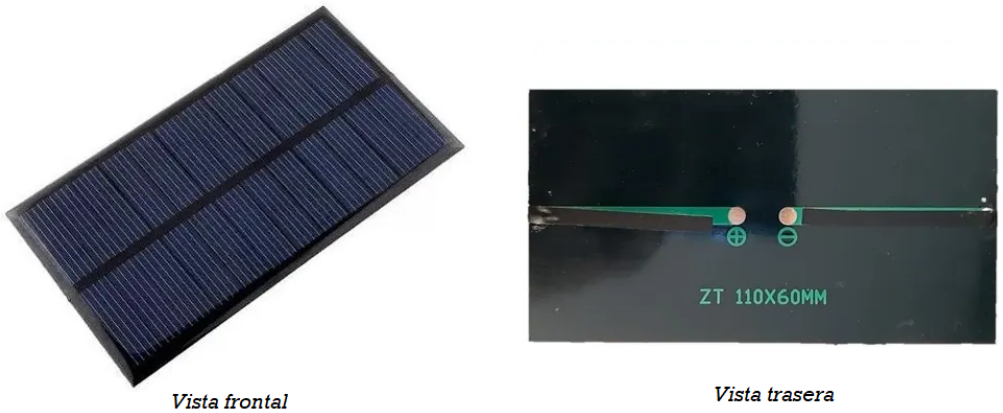

Figura 2.6: Celda solar (Recuperado de https://articulo.mercadolibre.com.mx/MLM-647646775-celda-solar-6v-1w-_JM el 12 de noviembre de 2022).

2.5. Bases de datos

Una base de datos es una recopilación organizada de información que usualmente se almacena electrónicamente en un sistema informático. Los datos comúnmente se estructuran a manera de filas y columnas en una serie de tablas, esto con el propósito de aumentar la eficacia de procesamiento y consulta de datos. Con una base datos es posible acceder, gestionar, modificar, actualizar, controlar y organizar sencillamente grandes volúmenes de datos [16].

Existen diversos tipos de bases de datos, donde cada una se caracteriza por una forma particular de organizar y utilizar los datos. Algunos de los principales tipos son:

- Relacionales.
- De código abierto.
- Almacenes de datos.
- En la nube.
- NoSQL.

SQL es un lenguaje de programación que utilizan las bases de datos relacionales, donde los elementos de este tipo de bases de datos se organizan en un conjunto de tablas con filas y columnas. Este sirve para consultar, manipular y definir los datos, así como proveer de control de acceso a la base de datos.

Un software de bases de datos, también llamado sistema de gestión de bases de datos (DBMS), se utiliza para la creación, edición y gestión de estas, facilitando la interacción con la base de datos. El software también debe de proveer un enfoque de seguridad, realizando copias de seguridad y protegiendo los datos de un posible robo.

Algunos ejemplos de software de bases de datos son MySQL, Microsoft Access, Microsoft SQL Server, FileMaker Pro, Oracle Database y dBASE.

2.5.1. MySQL

MySQL es un sistema de gestión de bases de datos relacionales de código abierto basado en SQL. Este sistema funciona con una estructura cliente-servidor, donde los procesos principales que se llevan a cabo son los siguientes [17]:

- MySQL crea una base de datos para almacenar y manipular datos, definiendo la relación de cada tabla.
- Los clientes realizan solicitudes mediante instrucciones SQL específicas en MySQL.
- El servidor responderá con la información solicitada y esta será enviada a los clientes.

En MySQL, también es posible otorgar distintos privilegios a cada usuario, de manera que cada uno tenga distinta jerarquía y posibilidades al momento de interactuar con la base de datos. Las principales operaciones que pueden realizarse en una base de datos con MySQL son las siguientes [18]:

- ***CREATE:*** Permite al usuario crear una base de datos o una tabla.
- ***SELECT:*** Permite al usuario recuperar datos previamente almacenados.
- ***INSERT:*** Permite al usuario agregar nuevas entradas a las tablas de la base de datos.
- ***UPDATE:*** Permite al usuario modificar entradas existentes.
- ***DELETE:*** Permite al usuario borrar entradas de la tabla.
- ***DROP:*** Permite al usuario eliminar tablas de las bases de datos.

Para realizar la gestión de las bases de datos en MySQL a través de internet, existe la herramienta de software libre *phpMyAdmin*, teniendo soporte para todas las operaciones de MySQL a través de la interfaz de usuario, tales como administrar bases de datos, tablas, columnas, índices, usuarios, permisos, etc [19]. Esta herramienta es de gran utilidad pues provee de una herramienta sumamente sencilla para la creación y administración de bases de datos.

En la figura 2.7 se observa la interfaz de usuario que brinda phpMyAdmin para la gestión de bases de datos, mostrando los datos en la tabla, sus filas y columnas, y todas las tablas que han sido creadas para esa base de datos.

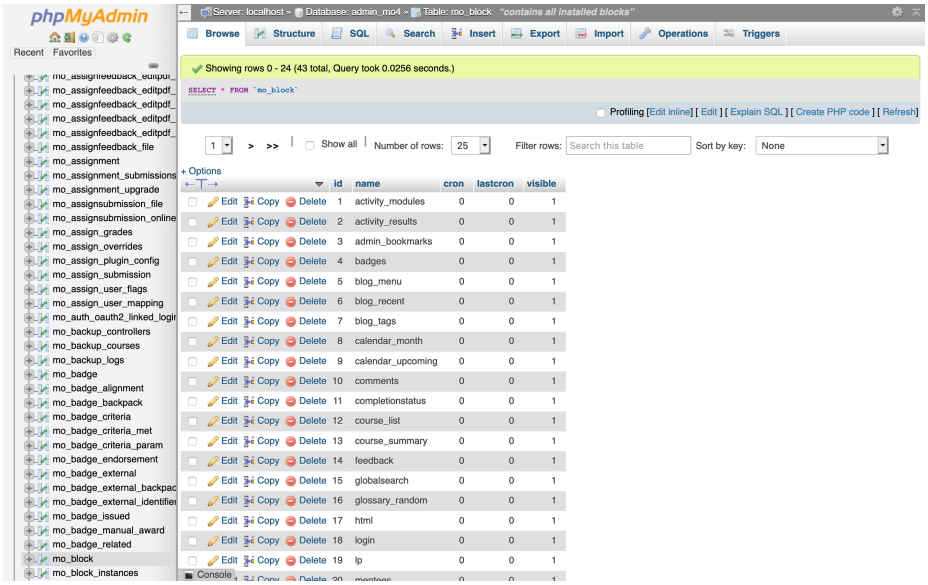

Figura 2.7: Estructura de bases de datos en phpMyAdmin (Recuperado de https://installatron.com/phpmyadmin el 1 de noviembre de 2022).

2.6. Desarrollo web

El desarrollo web es aquella actividad que conlleva la creación y mantenimiento de aplicaciones que se alojan en la red de internet. El desarrollo web implica diferentes tareas a realizar como la codificación, diseño, gestión de contenidos y administración de los mismos. Debido a la complejidad de las tareas, el desarrollo web se rige de la utilización de diferentes herramientas de programación, entre las que se pueden mencionar HTML, CSS, PHP y JavaScript, las cuales, hacen de la aplicación web un sitio dinámico, práctico y funcional [20].

2.6.1. HTML/CSS

HTML (Lenguaje de Marcas de Hipertexto) es el componente básico del desarrollo web, pues define el significado y estructura del contenido web.

HTML utiliza "marcas"para etiquetar texto, imágenes y otro contenido para mostrarlo en un navegador Web. Las marcas HTML incluyen elementos especiales como <head>, <title>, <body>, <header>, <footer>, <article>, <section>, <p>, , , <img>, <aside>, <audio>, <canvas>, <datalist>, <details>, <embed>, <nav>, <output>, <progress>, <video>, <ul>, <ol>, entre otros. Por lo que un elemento HTML se distingue del demás texto en un archivo mediante **etiquetas** [21].

HTML no es el único lenguaje existente para crear documentos de hipertexto, sin embargo, este se ha convertido en el lenguaje estándar para la creación de contenido web [22].

Todos los documentos HTML siguen una estructura básica, donde las etiquetas fundamentales son la cabecera (HEAD) y el cuerpo del documento (BODY). La cabecera puede contener el título, palabras clave, etc. Por otro lado, el cuerpo del documento contiene toda la información que se presentará en el despliegue de la página web. Además de HTML, generalmente se utilizan otras tecnologías para describir la apariencia/presentación de una página web (CSS) o la funcionalidad/comportamiento (JavaScript).

CSS (Hojas de Estilo en Cascada) es un lenguaje de estilos utilizado para la presentación de documentos de hipertexto como HTML o XML. Es decir, CSS [23] no es un lenguaje de programación, tampoco un lenguaje de etiquetas, sino un lenguaje de hojas de estilo [24]. Este permite dar una apariencia mucho más dinámica a las páginas web haciendo uso de poco código.

2.6.2. PHP

PHP [25] (Hypertext Preprocessor) es un lenguaje de programación de código abierto ampliamente utilizado para el desarrollo web, pues este puede ser introducido en HTML. Este es un lenguaje rápido, interpretado, orientado a objetos y multiplataforma, además de que tiene disponible una gran cantidad de librerías de código libre; lo que hace que PHP sea utilizado para desarrollar aplicaciones web complejas. PHP es un lenguaje que corre del lado del servidor, es decir, que los clientes no tienen acceso a ninguna parte de este código.

Una de las grandes virtudes de PHP es que proporciona la posibilidad de utilizarlo para gestionar bases de datos en sistemas como MySQL, por lo que muchas aplicaciones web de consulta de datos están desarrolladas en dueto con PHP y MySQL.

2.6.3. JavaScript

JavaScript es un lenguaje de programación interpretado que corre del lado del cliente por lo que los usuarios podrán ver todo el código que se tenga. JavaScript permite implementar funciones complejas en aplicaciones web, por ejemplo [26]:

- ➤ Mostrar información dinámica.
- ➤ Actualización de contenido.
- ➤ Mapas interactivos.
- ➤ Gráficos 2D/3D.
- ➤ Reproductoras de vídeo.
- ➤ Animación de imágenes.
- ➤ Controlar multimedia.

Utilizar código JavaScript permite personalizar mucho más la experiencia de los usuarios y ejecutar código mucho más poderoso que con PHP. Esto es debido a que cada usuario ejecuta individualmente dicho código y no se saturará al servidor para su ejecución.

2.7. Concentración de CO_2

El CO_2 o **dióxido de carbono** es un gas que puede originarse tanto por fuentes naturales como por actividad humana. Las consecuencias de este son de gran importancia en la calidad del aire, contaminación atmosférica y la emisión de gases de efecto invernadero [27].

La normativa vigente, que provee de notas técnicas de prevención sobre ventilación general en los edificios, recoge la categorización de la calidad del aire en función de la concentración de CO_2. Dicha clasificación es la siguiente:

- **Hasta 350 ppm:** Calidad alta.
- **Entre 350 y 500 ppm:** Calidad buena.
- **Entre 500 y 800 ppm:** Calidad moderada.
- **Entre 800 y 1200 ppm:** Calidad baja.
- **Superior a 1200 ppm:** Calidad mala.

Tomando en cuenta que el CO_2 puede ser producido por cada exhalación en la respiración humana, cuanto más respiramos, más CO_2 se acumula en el aire de un espacio cerrado. De esta manera, la concentración de CO_2 indica la cantidad de aire que ha sido expulsada por alguien más [28].

El nivel de 400 ppm de CO_2 es considerado como el nivel promedio, pues esta concentración es la que se suele encontrar en exteriores; por lo que un aumento a 800 ppm corresponde a un aire del cual el 1 % ya ha sido respirado por alguien más. Lo anterior mencionado puede ser observado de manera más detallada en la figura 2.8, donde se muestra la relación del porcentaje del aire que ya ha sido respirado con la concentración de CO_2 que se tiene en la habitación.

Figura 2.8: Aire ya respirado vs. Concentración de CO_2 (Recuperado de https://elpais.com/especiales/coronavirus-covid-19/como-esquivar-el-coronavirus-en-interiores/ el 12 de noviembre de 2022).

3 Estado del arte

La calidad del aire en espacios cerrados ha sido un tema de inquietud para la comunidad científica, pues se ha reportado que una repetida exposición a contaminantes en ambientes cerrados es una de las potenciales causas de enfermedades crónicas, tales como cáncer de pulmón, enfermedades cardiovasculares e infecciones respiratorias [29]. En la pandemia de COVID-19 se reafirmó la importancia de contar con un aire limpio en un espacio cerrado, es por eso por lo que el énfasis en los aspectos de diseño de los sistemas de monitoreo de calidad del aire, incluyendo el tipo de sensor, microcontrolador a utilizar, arquitectura y conectividad ha crecido a lo largo de los años.

El impacto que tiene la contaminación en espacios cerrados (IAP, por sus siglas en inglés, Indoor Air Pollution) puede llegar a ser hasta 100 veces mayor que aquella presente en exteriores; esto se debe a que los espacios cerrados contribuyen al aumento de contaminantes en mayor medida que un espacio abierto.

Para este tipo de sistemas de monitoreo existen dos potenciales tecnologías que presentan una sólida plataforma de desarrollo: tecnologías de sensores inalámbricos (WSN, por sus siglas en inglés, Wireless Sensor Network) y el internet de las cosas. Existen dos componentes esenciales para estos sistemas, hardware y software. Por un lado, se tiene que la elección del sensor correcto, microcontroladores y gateways es un factor crucial para investigaciones de este tipo. Por otro lado, las tecnologías de comunicación como Wi-Fi, ZigBee, Bluetooth y Ethernet son ampliamente utilizadas para poder contar con actualizaciones en tiempo real de la concentración de contaminación del aire.

Dicho lo anterior, existen diferentes alternativas y aspectos a considerar para medir la calidad del aire en espacios cerrados, así como para visualizar los datos arrojados por los sensores y los sistemas para actuar dependiendo de los datos arrojados.

3.1. Parámetros por medir

Algunos de los sistemas consultados únicamente miden $\boldsymbol{CO_2}$ para monitorear la calidad del aire, tales como [30], [31] y [32]. Sin embargo, otros son capaces de medir una mayor gama de parámetros para monitorear la calidad del aire; como son [33], [34], [35], [36], [37], [38], [39], [40] y [41], que miden la **temperatura y humedad** de la habitación de interés. Asimismo, también resulta valioso medir otros tipos de factores como **compuestos orgánicos volátiles (VOCs)** dimensionados por [33]

y [36]; $\boldsymbol{PM_{10}}$ medido por [33], [42] y [43]; **luminosidad** estudiada por [33] y [40]; **CO** medido por [34], [42] y [44]; **amoniaco ($\boldsymbol{NH_3}$)** mensurado por [38] y [44]; **polvo** cuantificado por [35], [39] y [40]; **alcohol y sulfuro** dimensionado por [38] y [44]. De igual forma, existen sistemas más específicos en cuanto a las variables que estudian, pues encontramos que estos son los únicos que miden los parámetros que se mencionan a continuación: **dicloro ($\boldsymbol{Cl_2}$), ozono ($\boldsymbol{O_3}$) y dióxido de azufre ($\boldsymbol{SO_2}$)** estudiado por [34]; **benceno y humo** medidos por [38]; **presión atmosférica** mensurado por [36]; **oxígeno ($\boldsymbol{O_2}$)**, **óxido de nitrógeno ($\boldsymbol{NO}$), compuestos aromáticos y vapores** medidos por [44]; y finalmente **dióxido de nitrógeno ($\boldsymbol{NO_2}$)** estudiado por [42].

Como podemos observar, existe un sinnúmero de parámetros que pueden ser considerados para determinar la calidad del aire en una habitación, sin embargo, el CO_2 es una variable fundamental para el monitoreo de la Calidad del Aire en Interiores (IAQ) [29].

3.2. Sensores

Para realizar la medición de las variables de interés, se cuenta con una gran variedad de sensores, por ejemplo, los trabajos en [32], [35], [37], [38] y [43] utilizan el **MQ135** para realizar la medición de CO_2. Por otro lado, para estudiar el mismo gas, otros trabajos utilizan diferentes modelos, por ejemplo, [36], [40] y [44] utilizan el **MG-811**, el cual tiene un precio de $1,000.00 MXN aproximadamente, siendo mucho más costoso a comparación de los otros modelos utilizados. [33] Mide con el **SCD30**; [30] realiza el estudio con **MHZ-19**; [34] mide con **INE20-CO2P-NCVSP**; [42] mensura con **MH-Z14**; [39] mide con **S8 0053**; y finalmente, [41] utiliza el modelo **COZIR**.

Enfocándonos ahora en los modelos de sensores utilizados para medir otros parámetros además del CO_2, se observó que [35], [37] y [38] miden temperatura y humedad con **DHT11**. Sin embargo, dicha variable también puede ser medida con los modelos **BME680, BME280, DFRobot BME680, SHT30 y TH2**, los cuales fueron utilizados por [33], [34], [36], [39] y [40], respectivamente.

En cuanto a modelos dedicados a la medición de otros factores, se tiene que [33] utiliza **CCS811** (VOC), **SPS30** (PM_{10}) y **VEML7700** (luminosidad). [34] Monitorea con **4-SO2-20** (SO_2), **4-NO2-20** (NO_2), **OX-A431** (O_3), **4-CO-500** (CO) y **4-Cl2-50** (Cl_2). [35] Realiza su monitoreo con **DSM501** (polvo). [42] Mide con **MICS-4514** (NO_2 y CO) y **GP2Y1010AU** (PM_{10}). [44] Mensura con **MQ-7** (CO), **ME2O-O2** (O_2) y **MQ135** (NO, compuestos aromáticos, alcohol, vapores, NH_3 y sulfuro). [36] Mide con **DFRobot BME680** (VOC y presión). [43] Mesura con **Sharp GP2Y1010AU0F** (PM_{10}). [38] Monitorea con **MQ135** (alcohol, humo, benceno, amoniaco y sulfuro). [39] Mensura con **WZ-S-K** (formaldehído). Y finalmente. [40] mide con **PPD42NS** (polvo) y **TSL2561** (luminosidad).

Realizando ahora una clasificación de los sensores anteriormente mencionados según su principio de funcionamiento, se tiene que **MQ7, MQ135, MG811, CCS811 y WZ-S-K** son del tipo **químico**; es decir, que cuentan de algún elemento dentro de su construcción que reacciona al estar en presencia del parámetro a medirse; algunos

de estos elementos son tubos de cerámica, electrodos, calentadores, etc [45]. Por otro lado, los sensores **SPS30, DSM501, GP2Y1010AU0F, VEML7700 y TSL2561** son del tipo **óptico**; pues cuentan con un fototransistor y/o fotodiodo para poder detectar el parámetro de interés [46]. Asimismo, se encontraron sensores con principio de funcionamiento denominado **NDIR**, los cuales son **MH-Z19, MH-Z14, S8-0053, COZIR y SCD30**; utilizando un sensor de infrarrojo no dispersivo que detecta la cantidad de energía absorbida por el gas deseado, siendo este un principio que también puede denominarse óptico, pero que fue mencionado de manera individual debido a su mecanismo tan particular [47].

Los anteriores principios de funcionamiento son los más comunes entre los sensores mencionados, sin embargo, también hay otros 2 bastante únicos que merecen ser señalados. El sensor **DHT11** es de tipo **resistivo**, pues tal y como lo sugiere su nombre, cuenta con un resistor cuya resistencia varía dependiendo del parámetro a medirse [48]. De igual manera, se tiene que el sensor **MICS-4514** que entra en la clasificación de **MOS**, donde estos miden la cantidad de electrones libres que pasan a través de un material especial, de manera que entre más electrones libres haya, mayor corriente eléctrica se tiene y por lo tanto se detecta mayor cantidad del gas de interés [49].

3.3. Microcontroladores

En cuanto a las unidades de procesamiento se refiere, la más utilizada fue la **ESP8266**, siendo aprovechada por [30], [32] [35], [40] y [44]. Seguida por la **Arduino UNO**, que fue empleada en [37], [38], [40] y [43].

Asimismo, hubo casos donde se utilizaron otros modelos de microcontrolador en vez de los mencionados anteriormente, pues [33] utiliza **HUZZAH32**; [42] aprovecha las capacidades de la **ESP32**; [39] emplea el **STM32F103C8T6**; y [41] maneja el **MSP430F5529**.

Es importante hacer énfasis en que algunos sistemas utilizan más de un microcontrolador, haciendo así que las tareas de procesamiento se realicen de manera fragmentada. [34] Aprovecha las capacidades de **ATMEGA281 y Raspberry Pi**; [44] emplea **ATMega32 y ESP8266**; [43] maneja **Arduino UNO y Raspberry Pi3**; y [40] usa **Arduino UNO y ESP8266**. Donde para todos estos casos, el primer microcontrolador funge el papel de procesamiento y el segundo de conexión a internet. Además, cabe separar del resto el caso de [36], pues a pesar de que también utiliza 2 microcontroladores (**Raspberry Pi 3B+ y Arduino Uno**), las tareas que se les otorgó en este sistema son más complejas en comparación de los demás; pues la Raspberry se encarga de alojar todos los microservicios, poder de procesamiento y conexión a internet, mientras que la Arduino UNO realiza el manejo de procesamiento de los sensores.

3.4. Actuadores

De los artículos consultados, varios de estos cuentan con dispositivos actuadores para poder interactuar con el usuario, de manera que el sistema sea más eficaz e intuitivo al momento de ser utilizado. De lo anterior, cabe mencionar que el uso de actuadores no se encuentra limitado a uno por sistema, sino que pueden usarse varios de manera conjunta, de forma que cada uno cumpla una tarea en específico y la implementación resulte mucho más amigable y asertiva.

El actuador más utilizado fue el uso de **notificaciones y alertas**, empleadas en [30], [39], [41] y [42]; donde tales alertas van desde mensajes SMS, email y hasta notificaciones dentro de una aplicación móvil. Las cuales son enviadas en caso de que se supere cierto valor de referencia del parámetro de interés; donde en el caso de [30] dicho valor puede ser configurado por el usuario.

En segundo lugar, se encuentran empatados el uso de **ventiladores** y **pantallas LCD**, los cuales fueron utilizados por [36], [38] y [43] para el caso de ventiladores, y [36], [41] y [44] para pantallas LCD. Donde los ventiladores tenían un sistema en el que estos tenían una velocidad adaptativa dependiendo de las condiciones que se censan, mientras que la pantalla LCD servía para presentar información en tiempo real.

A pesar de ser un caso similar al anterior donde se tienen ventiladores como actuador, cabe resaltar el uso de aire acondicionado adaptativo realizado por [37], el cual es accionado dependiendo del estado de los parámetros medidos, de manera que se tenga un valor óptimo de calidad de aire, temperatura y humedad.

3.5. Conectividad

Con el fin de realizar la comunicación tanto de forma cableada como inalámbrica de toda la información necesaria para llevar a cabo el funcionamiento del sistema IAQ, se hizo uso de diversos protocolos de comunicaciones, siendo **WiFi** el más recurrente, pues es utilizado por [30], [32] - [36] y [39] - [44]. En segundo lugar, se encuentra **MQTT**, siendo aprovechado por [35], [36], [39] y [43]. Seguido por **I2C** que se emplea en [33] y [34].

De lo anterior, se observa que sólo se hizo mención de aquellos protocolos que fueron utilizados más de una vez, sin embargo, también hubo aquellos casos como el de **ZigBee/Ethernet** que es manejado por [34]; **Bluetooth** que lo emplea [37]; **SMTP** utilizado por [41]; y finalmente **LoRa/RS485** y **GPRS/NB-IoT** que pueden ser aprovechados por [39]. Donde en este caso se tiene la posibilidad de migrar de un protocolo a otro, cosa que también sucede con [34] que puede emplear tanto **Ethernet** como **WiFi**.

3.6. Plataforma IoT

La visualización de datos para su monitoreo resulta muy conveniente en este tipo de sistemas IAQ, donde los artículos consultados utilizan ya sea plataformas que ellos mismos desarrollaron, o bien servicios de monitoreo de datos preexistentes. Empezando con aquellos sistemas con plataforma propia, se tiene a [30] con su aplicación móvil ***iAirCO$_2$***, [36], [37], [39] y [44]. Por otro lado, los artículos que no utilizaron una plataforma de monitoreo propia fueron [33], [42] que manejaron **Blynk App**, y [32], [34] y [40] que aprovecharon **Carbon Insight App**, **Emoncms Web App** y **ThingSpeak**, respectivamente. Donde [43] a pesar de haber declarado usar una plataforma no propia, nunca se especifica el nombre de esta. Asimismo, hay que hacer hincapié en que, de todas las plataformas mencionadas, únicamente el sistema presentado en [42] menciona ser de código abierto.

De lo anterior también es importante recalcar que, para el almacenamiento de los datos proveídos por los sensores colocados, fue necesario el uso de bases de datos. Encontrándose que [30] y [43] utilizan **SQL**, [34] emplea **POSTGRESQL**, [35] y [44] utilizan **MySQL**, y [36] aprovecha **NoSQL MongoDB**.

Finalmente, a manera de sintetizar toda la información mencionada en este capítulo, se presenta la tabla 3.1.

Tabla 3.1: Sistemas de medición de IAQ.

Artículo	Parámetros medidos	Sensor de CO_2	Sensores adicionales	MCU	Actuador	Conectividad	Plataforma IoT
[30]	CO_2	MHZ-19	-	ESP8266	SMS, email o notificación en la app	WiFi	$iAirCO_2$ App (propia)
[32]	CO_2	MQ135	-	ESP8266	-	WiFi	Carbon Insight App (no propia)
[33]	CO_2, temperatura y humedad, VOC, PM_{10} y luminosidad	SCD30	BME680, CCS811, SPS30 y VEML7700	HUZZAH32.	-	I2C y WiFi	Blynk App (no propia)
[34]	CO_2, CO, SO_2, NO_2, O_3, Cl_2, temperatura y humedad	INE20-CO2P-NCVSP	4-SO2-20, 4-NO2-20, OX-A431, 4-CO-500, 4-Cl2-50 y BME280	ATMEGA281 y Raspberry Pi	-	ZigBee, I2C y Ethernet/WiFi	Emoncms Web App(no propia)
[35]	CO_2, polvo, temperatura y humedad	MQ135	DHT11 y DSM501	ESP8266	-	MQTT y WiFi	-
[36]	CO_2, VOC, presión atmosférica, temperatura y humedad	MG-811	DFRobot BME680	Raspberry Pi 3B+ y Arduino Uno R3	Pantalla LCD y ventilador	MQTT y WiFi	App móvil (propia)

Tabla 3.1: Sistemas de medición de IAQ.

Artículo	Parámetros medidos	Sensor de CO_2	Sensores adicionales	MCU	Actuador	Conectividad	Plataforma IoT
[37]	CO_2, temperatura y humedad	MQ135	DHT11	Arduino UNO	Aire acondicionado	Bluetooth	App Móvil (propia)
[38]	CO_2, alcohol, humo, benceno, amoniaco, sulfuro, temperatura y humedad	MQ135	DHT11 y MQ135	Arduino UNO	Ventilador	-	-
[39]	CO_2, formaldehído, polvo, temperatura y humedad	S8 0053	SHT30 y WZ-S-K	STM32F103C8T6	Alertas	LoRa/RS485 o MQTT para GPRS/WiFi/NB-IoT	Móvil y Web App (propia)
[40]	CO_2, polvo, temperatura y humedad	MG-811	TH2, PPD42NS y TSL2561	Arduino UNO y ESP8266	-	WiFi	ThingSpeak (no propia)
[41]	CO_2, temperatura y humedad	COZIR	COZIR	MSP430F5529	SMS y pantalla LCD	SMTP y WiFi	-
[42]	CO_2, CO, PM_{10} y NO_2	MH-Z14	MICS-4514, GP2Y1010AU y DHT22	ESP32	Notificación en la app	WiFi	Blynk App (no propia)
[43]	CO_2 y PM_{10}	MQ135	Sharp GP2Y1010AU0F	Arduino UNO y Raspberry Pi 3	Ventilador	MQTT y WiFi	Web App (no propia)

Tabla 3.1: Sistemas de medición de IAQ.

Artículo	**Parámetros medidos**	**Sensor de CO_2**	**Sensores adicionales**	**MCU**	**Actuador**	**Conectividad**	**Plataforma IoT**
[44]	CO_2, CO, O_2, NO, compuestos aromáticos, vapores, alcohol, sulfuro y NH_3	MG-811	MQ-7, ME2O-O2, MQ135	ATMega32 y ESP8266	Pantalla LCD	WiFi	Web App (propia)

4 Desarrollo

4.1. Diseño de hardware

Como primer paso, era esencial contar con el hardware necesario para poder armar el circuito semáforo medidor de CO_2, de manera que para realizar esto se utilizaron los siguientes componentes:

- ESP32.
- Sensor MQ135.
- 3 LEDs (verde, amarillo y rojo).
- 4 resistores (3 de 220[Ω] y 1 de 1K[Ω]).
- Centro de carga solar.
- Celda solar.

Se aprovecharon las capacidades de procesamiento y conectividad WiFi de la ESP32 para poder realizar el correcto análisis de los datos proveídos por el sensor y tener la capacidad de saber qué LED encender dependiendo de dichos datos. De igual forma, se tiene en el circuito un centro de carga solar junto con su respectiva celda, esto de manera que el modelo no dependa de estar conectado a la corriente todo el tiempo y tenga mayor flexibilidad de uso. El armado del circuito puede observarse en la figura 4.1.

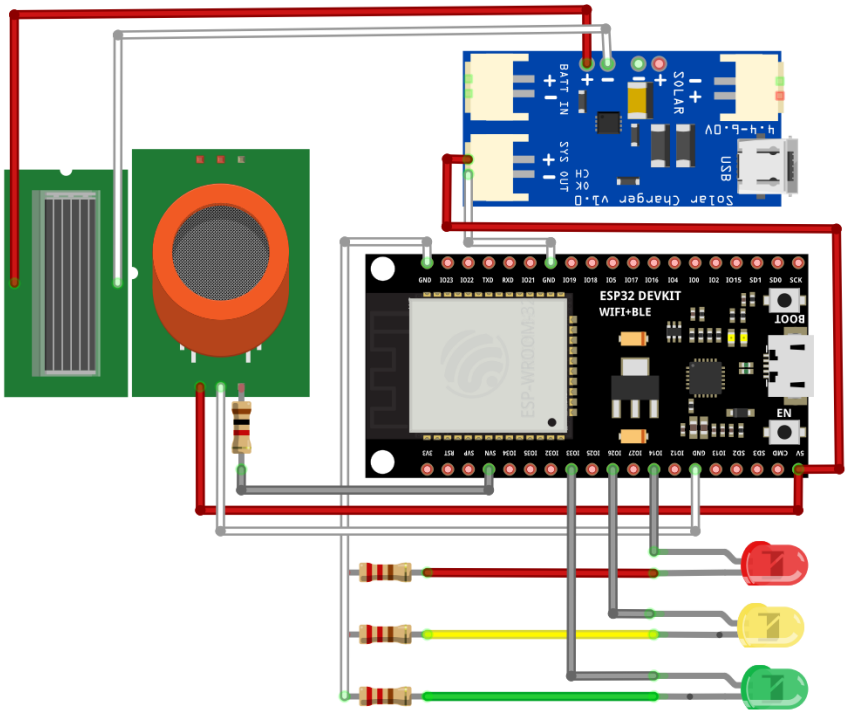

Figura 4.1: Armado de circuito.

Además del armado del circuito, también era necesario contar con una carcasa para poder sacar el modelo a campo y realizar los experimentos que resultaran necesarios sin tener que preocuparse de que los pines se desconectaran o algún componente se moviera de lugar. Por lo que para resolver lo anterior, se generó un modelo 3D en Blender, el cual puede observarse en la figura 4.2, señalando también dónde se coloca cada componente del circuito en este.

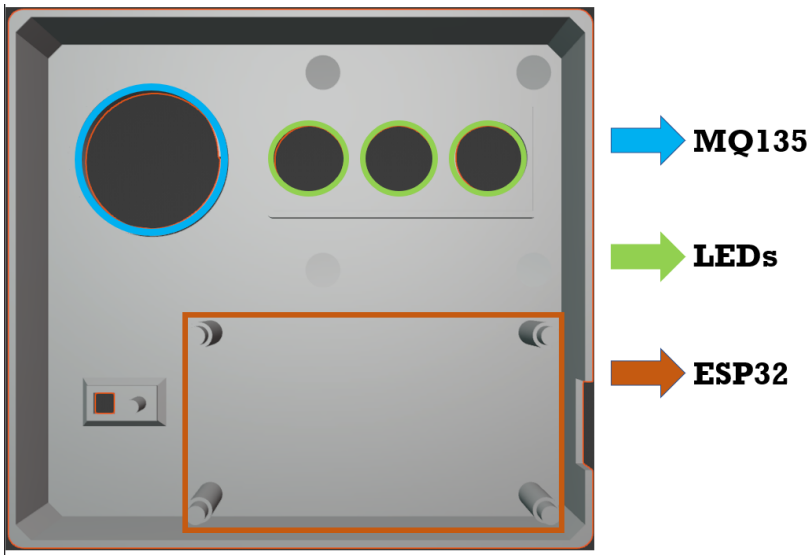

Figura 4.2: Carcasa para circuito (layout).

4.2. Software

4.2.1. AskSensors

Para la parte de codificación y la visualización de los datos, primero se aprovechó de una plataforma IoT de administración de sensores ya existente (AskSensors).

Tal sitio provee al usuario de un código de Arduino bastante intuitivo de utilizar para que todos los datos provenientes del sensor dado de alta sean visualizados, por lo que sólo fue necesario modificar el encendido de los LEDs dependiendo de la concentración de CO_2. En la figura 4.3 se muestra de manera general el funcionamiento del sistema utilizando tal plataforma.

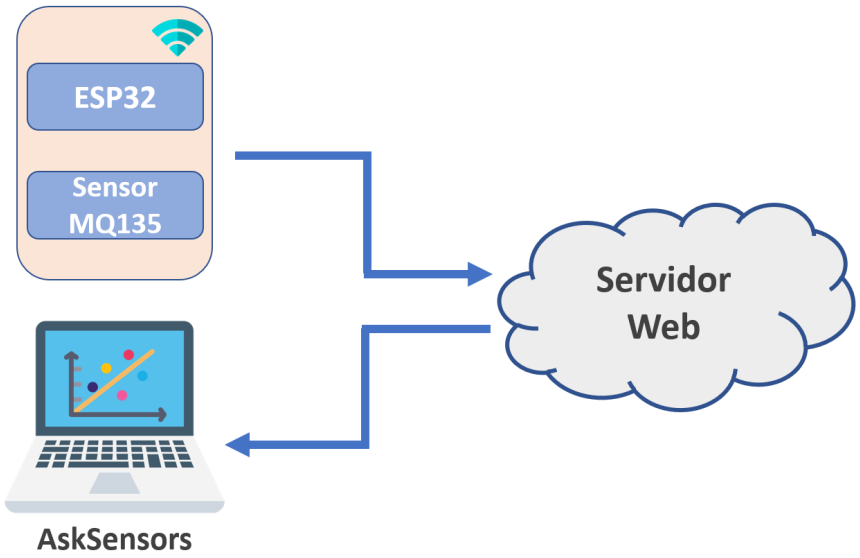

Figura 4.3: Arquitectura de sistema utilizando AskSensors.

Para mostrar la implementación de lo anterior dicho, se ingresó a AskSensors mediante la cuenta que se creó desde un inicio. Sin embargo, se mostraba en la plataforma un anunció diciendo que la prueba gratis había expirado y que para poder seguir utilizando su servicio se debía pagar por la suscripción correspondiente, por lo que se tuvo que crear otra cuenta para poder aprovechar de la prueba gratis de tres meses. Lo anterior puede observarse en la figura 4.4.

Teniendo una cuenta con la prueba gratuita activa, se dio de alta a un nuevo sensor para poder visualizar los datos que este provea, asignándole un identificador y una pequeña descripción. Lo cual puede observarse en la figura 4.5.

Una vez que el sensor fue dado de alta con éxito, la plataforma muestra toda la información correspondiente a este, donde cabe resaltar que la API Key In se utiliza para que el sensor y su respectiva gráfica queden vinculados y puedan comunicarse correctamente. Lo mencionado puede observarse en la figura 4.6.

Habiendo realizado todo lo mencionado anteriormente, ya no faltaba nada para poder observar gráficamente los datos proveídos por el sensor MQ135 utilizado, lo cual se muestra en la figura 4.7.

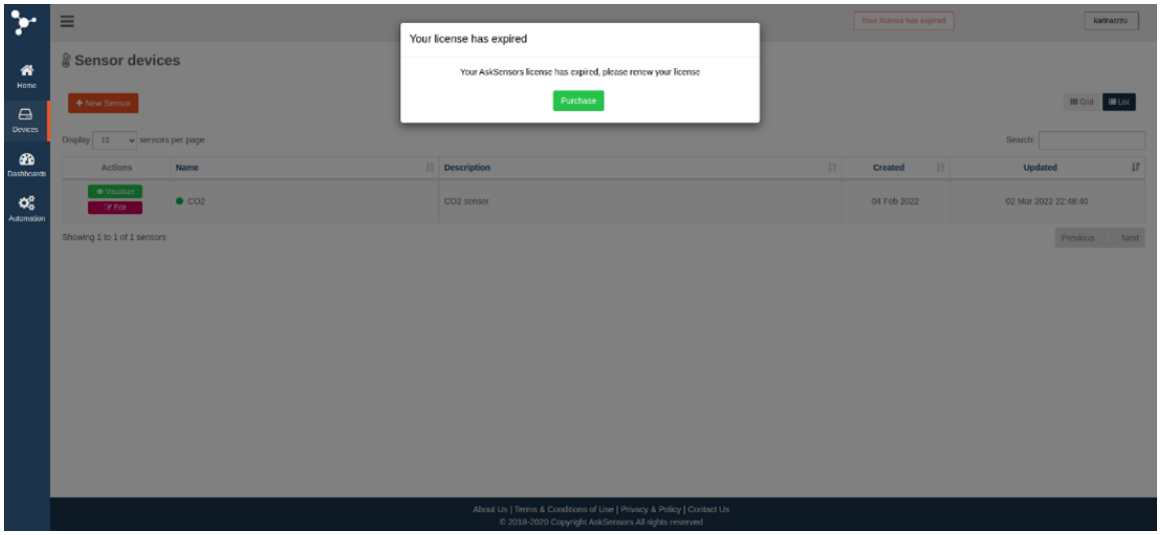

Figura 4.4: Prueba gratuita expirada - AskSensors.

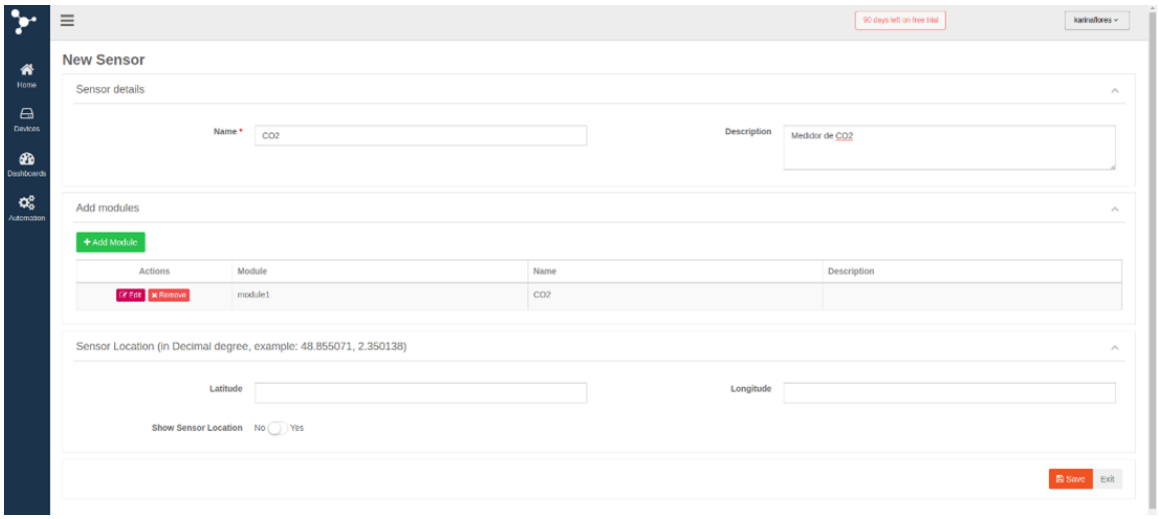

Figura 4.5: Proceso para dar de alta un nuevo sensor - AskSensors.

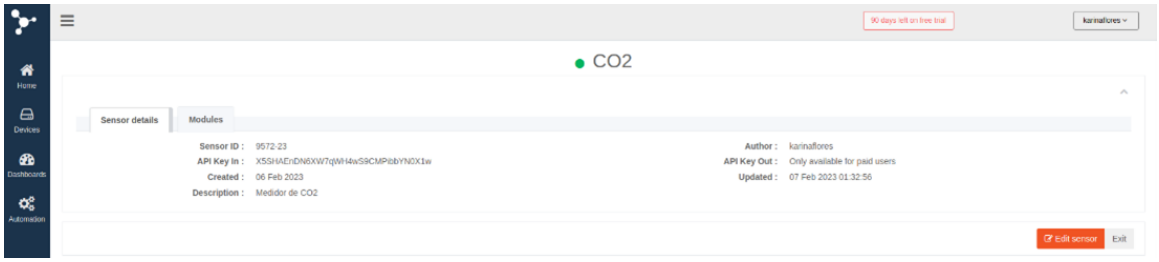

Figura 4.6: Sensor creado con éxito - AskSensors.

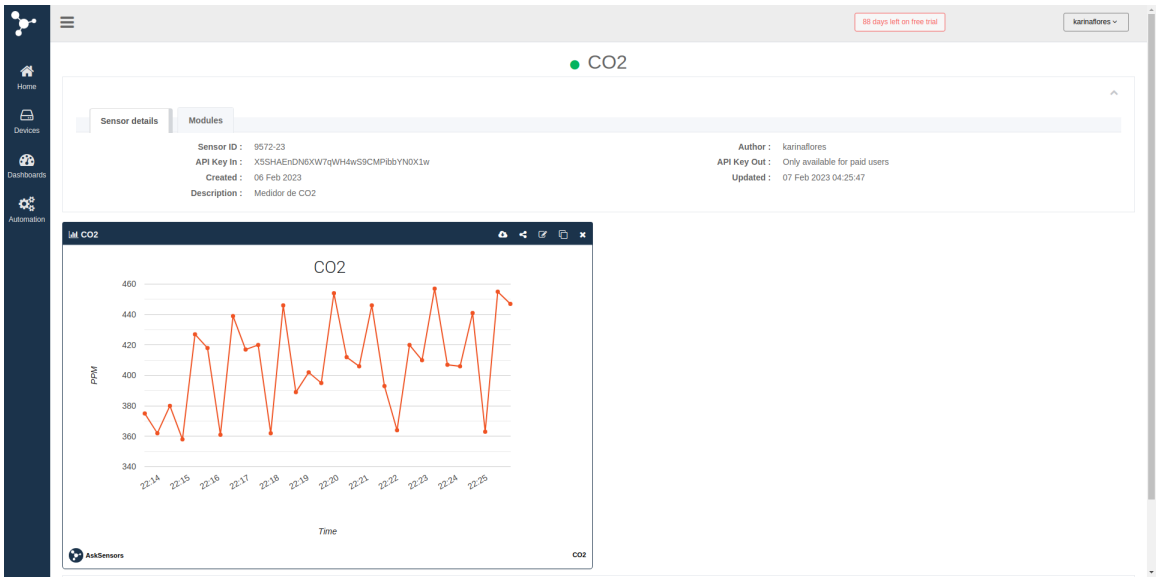

Figura 4.7: Gráfica correspondiente a sensor creado - AskSensors.

4.2.2. Plataforma Web propia

A pesar de que AskSensors resultó bastante amigable como punto de partida, era poco conveniente realizar todo el desarrollo sobre esta, ya que era de paga, no se tenía un completo entendimiento de cómo funcionaba y tampoco se contaban con muchas opciones de personalización sobre el sitio. Por lo que era imperativo desarrollar una plataforma Web de monitoreo de sensores propia para poder tener control total sobre todas aquellas limitantes que se mencionaron.

Implementación de MQTT y MySQL

Antes de comenzar con la codificación correspondiente a la plataforma web, era necesario implementar el protocolo de comunicaciones que esta utilizaría, así como una forma de almacenar los datos que fueran detectados por el sensor.

Como punto de arranque se tomó el código de Arduino con el que ya se contaba, sin embargo, este debía ser modificado para poder implementar el protocolo MQTT. De manera que se creara un nuevo publisher y tópico por cada sensor y sobre este se publicaran los datos para que fueran escuchados por los suscriptores del tópico. Por otro lado, también se codificó un script en Python que se suscribe al tópico creado anteriormente, recibe todos los valores arrojados por el sensor y seguido de esto los sube a una base de datos de MySQL para su oportuno almacenamiento.

Lo anterior mencionado puede ser visualizado de mejor manera en la figura 4.8.

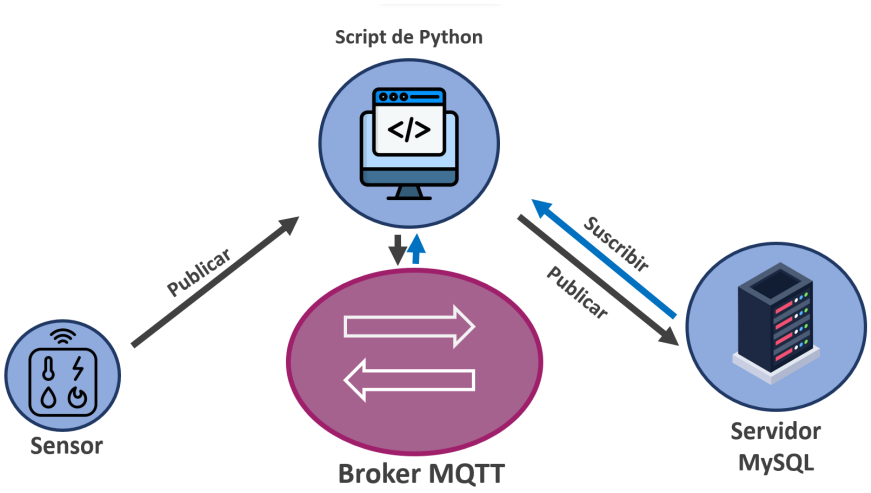

Figura 4.8: Arquitectura de implementación MQTT y MySQL.

Con el propósito de mostrar la implementación del protocolo MQTT, se tiene la figura 4.9, donde se observa el monitor serial de Arduino y el script de Python comunicándose entre sí. Resaltando que en la ejecución de Python puede observarse el tópico al que está suscrito para recibir la información pertinente, el valor de PPM que se leyó por el sensor y la llave correspondiente a QoS.

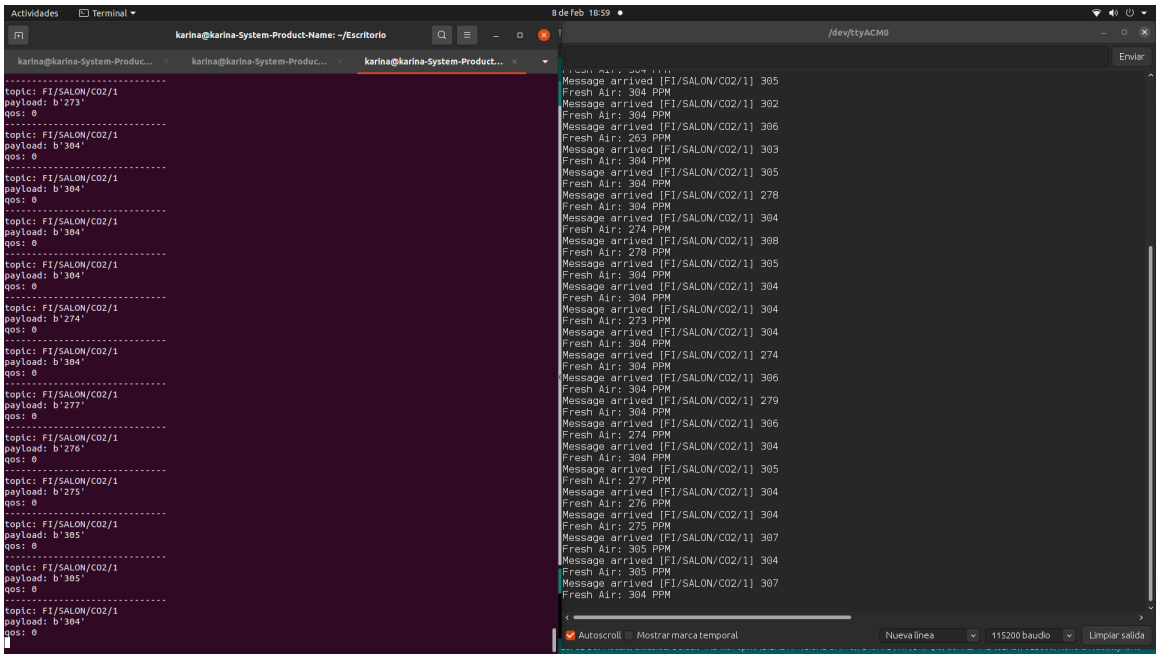

Figura 4.9: Comunicación entre scripts de Python (izquierda) y Arduino (derecha) mediante MQTT.

Mostrando ahora la creación de la base de datos, se tienen las figuras 4.10 y 4.11.

En la primera, se observa la base de datos correspondiente a los sensores dados de alta, donde se tienen las columnas de ID numérico, su nombre y si este se encuentra activo o no. La segunda figura corresponde a todo el registro de valores provenientes de sensores, donde se tienen columnas de ID numérico, el valor leído por el sensor, fecha y hora en que este fue subido a la tabla y el sensor que leyó dicho valor.

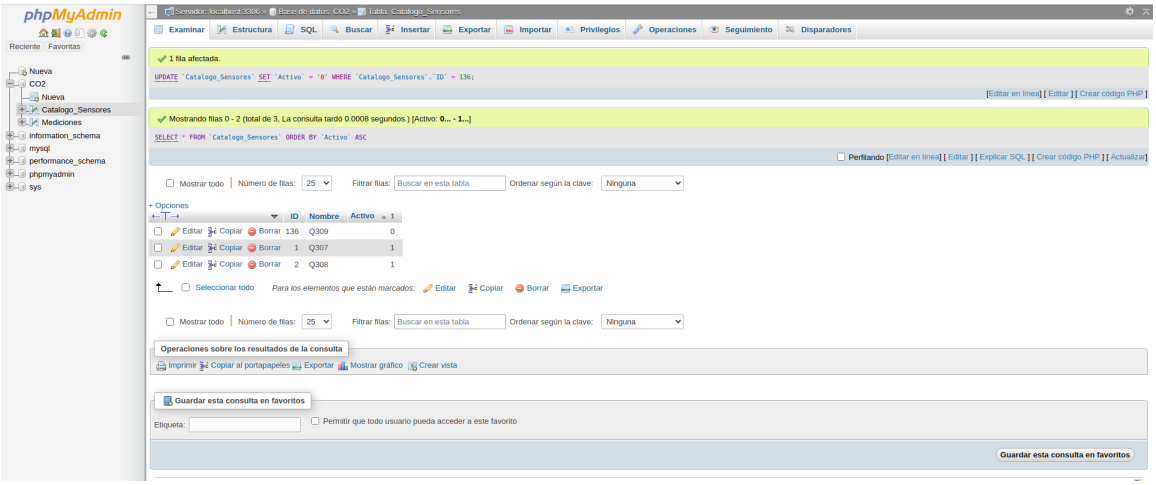

Figura 4.10: Sensores registrados - MySQL.

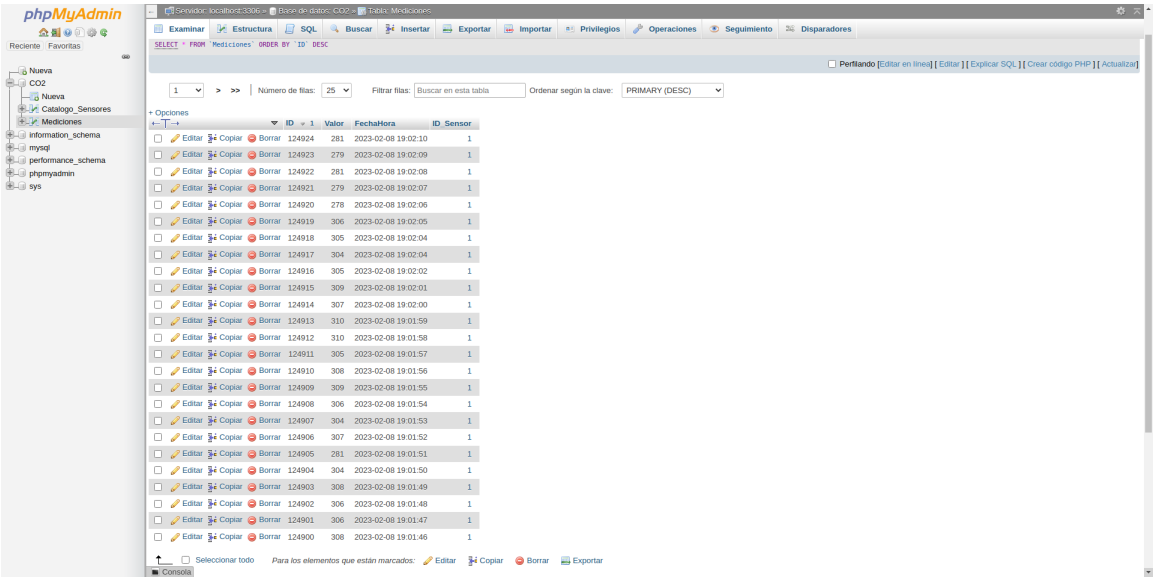

Figura 4.11: Relación de datos provenientes de sensores - MySQL.

Plataforma

Teniendo lista la implementación de MQTT y MySQL para que todos los datos proveídos por el sensor fueran enviados a la base de datos, se tenía un escenario más adecuado para empezar con el desarrollo de la plataforma Web.

El desarrollo de la plataforma web, no se inició con un diseño desde cero, sino que se utilizó la plantilla proporcionada por el sitio adminlte.io. Sitio que justamente tiene una sección de gráficas dentro de su plantilla, por lo que se empezó por remover todo el contenido sobrante para que únicamente se quedaran los elementos necesarios, es decir, un menú lateral para navegar por la plataforma, la pestaña correspondiente a la gráfica y una pestaña adicional que funciona como bienvenida al sitio.

Aunque ya se tenía de una plantilla con los diseños que se necesitaban, esta no contaba con nada para contribuir al funcionamiento de nuestra plataforma, por lo que fue necesario realizar codificación en HTML, JavaScript y PHP para que los puntos de la gráfica correspondieran a los datos medidos por los sensores, haciendo esto mediante una consulta de los últimos 10 datos de la base de datos y actualizándose continuamente para que la visualización sea en tiempo real.

Hasta este momento ya se contaba con una plataforma capaz de mostrar gráficamente y en tiempo real la información proveniente de un sensor, pero aun así era preciso contar con más funciones que solamente esa. Por lo que acudiendo de nuevo al sitio adminlte.io, se agregó a la plataforma la funcionalidad de poder consultar datos antiguos de algún sensor en particular, de forma que el usuario pueda elegir el rango de fechas de los datos a consultar, estos se muestren en una tabla, y que también pueda ser copiada y/o descargada en diversos formatos (CSV, Excel, PDF).

Ahora, con el propósito de cubrir la capacidad de nuestra plataforma de trabajar con múltiples sensores, se volvió a realizar la codificación correspondiente para agregar dicha funcionalidad. De manera que el usuario pueda agregar y/o eliminar el número de sensores que desee.

Con todo lo explicado hasta el momento y complementando la figura 4.8, el funcionamiento general de la plataforma web se muestra en la figura 4.12.

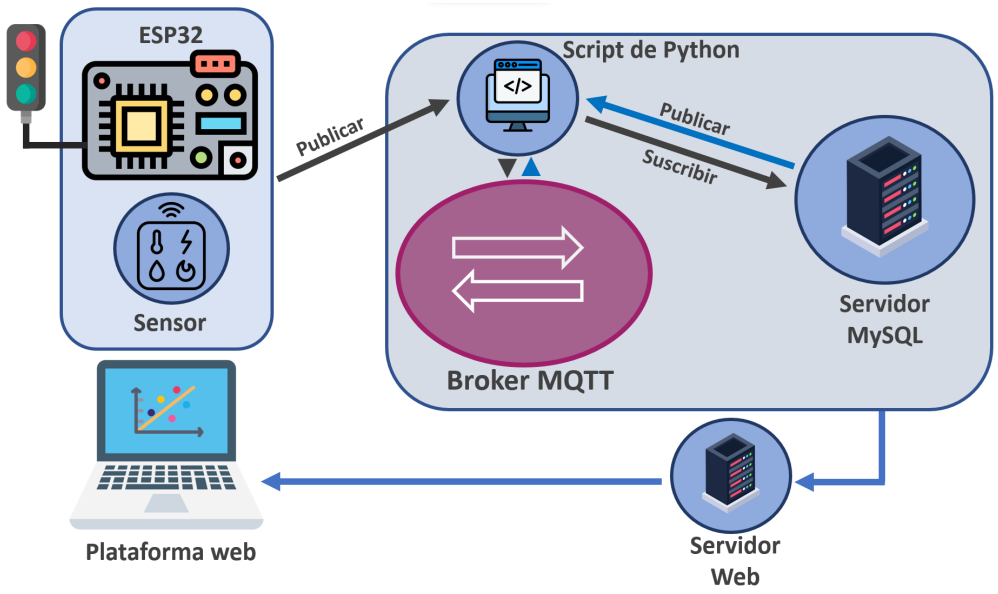

Figura 4.12: Arquitectura de sistema completo.

Con la finalidad de mostrar la pantalla principal de nuestra plataforma Web, se

tiene la figura 4.13. De igual forma, en el menú de la izquierda puede observarse que es posible acceder a las mediciones de los sensores, así como agregar nuevos a la plataforma.

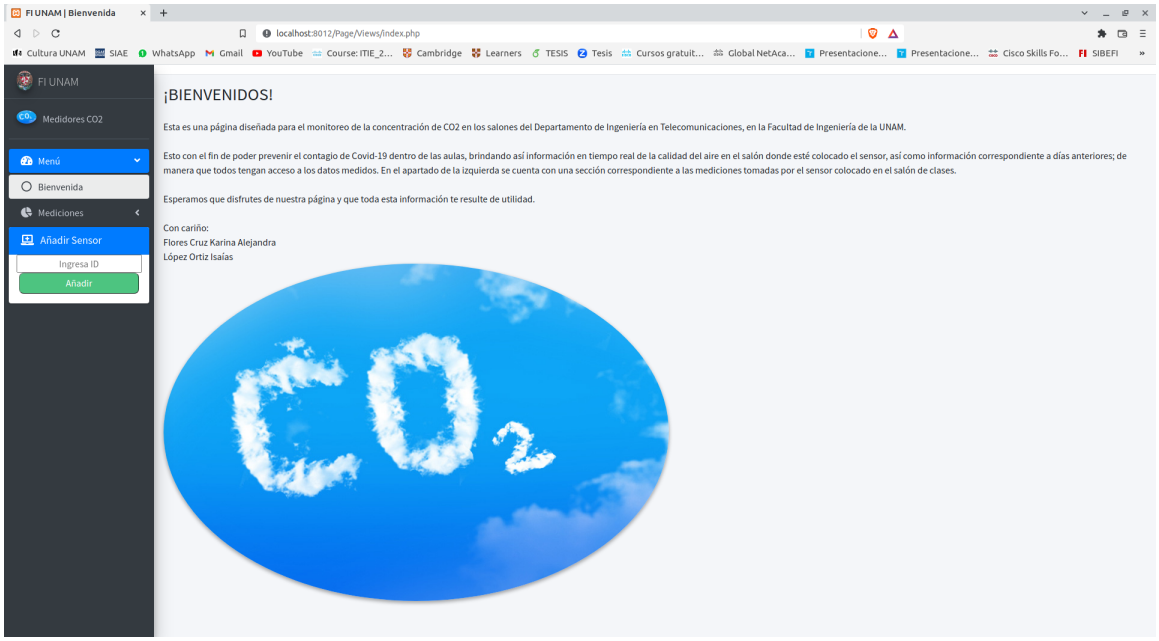

Figura 4.13: Menú principal - Plataforma Web propia.

➤ **Visualización gráfica de datos.**

En la figura 4.14 se muestra la pestaña correspondiente a la gráfica que se tiene en la plataforma Web propia, la cual pertenece al sensor con el nombre *Q307*. De la gráfica, cabe resaltar que en el eje horizontal se tiene la hora a la que fueron tomados los datos, y en el vertical el valor medido.

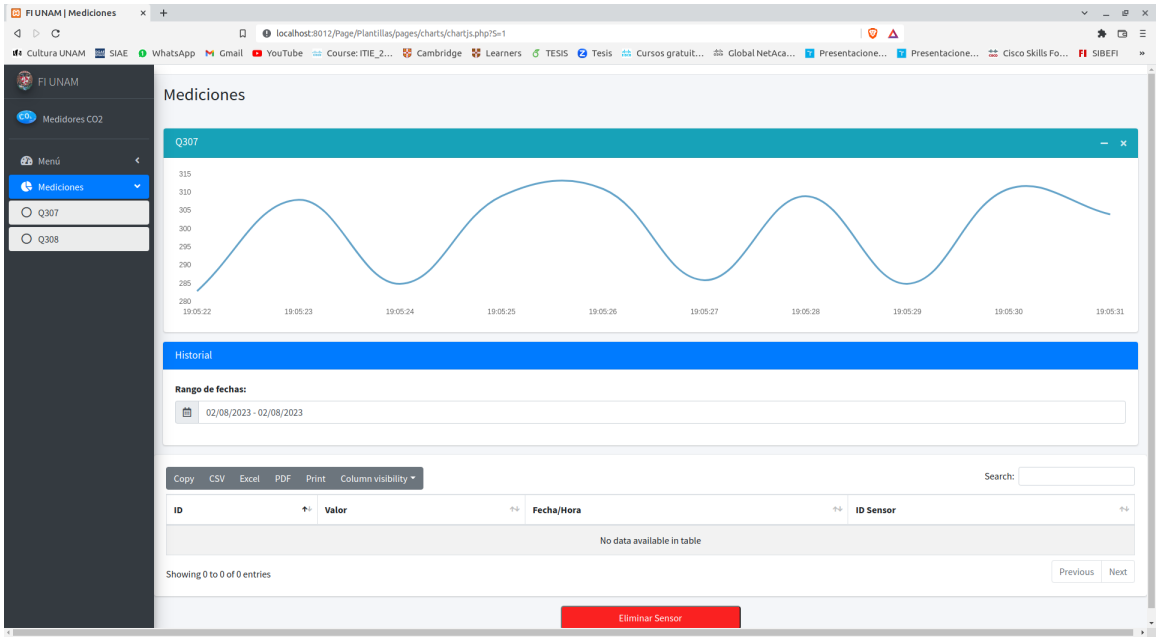

Figura 4.14: Visualización gráfica de datos - Plataforma Web propia.

➤ **Consulta de datos antiguos.**

Como se mencionó anteriormente, la plataforma Web propia no sólo tiene capacidad de mostrar los últimos 10 valores registrados en tiempo real, sino que también es posible realizar consulta de datos antiguos. Para esto, el usuario debe elegir el rango de fechas correspondiente a los datos que se quieran recuperar. Seguido de esto, tales datos se mostrarán en forma de tabla, la cual puede ser copiada, impresa o descargada en diversos formatos (CSV, PDF, Excel). Lo anterior puede observarse en la figura 4.15.

➤ **Agregar nuevo sensor.**

Refiriendo a la capacidad de la plataforma de trabajar con varios sensores a la vez, se tiene la figura 4.16. Donde de lado izquierdo abajo de los sensores dados de alta se tiene una sección de *Añadir Sensor*, donde primero se tiene que teclear el nombre con el que se le desea identificar y después dar click sobre el botón verde.

Hecho lo anterior, en la plataforma Web se muestra una confirmación de que el nuevo sensor ha sido creado con éxito, tal y como se observa en la figura 4.17. Aparte de agregarse el nuevo sensor a la plataforma, también se agrega una nueva fila al

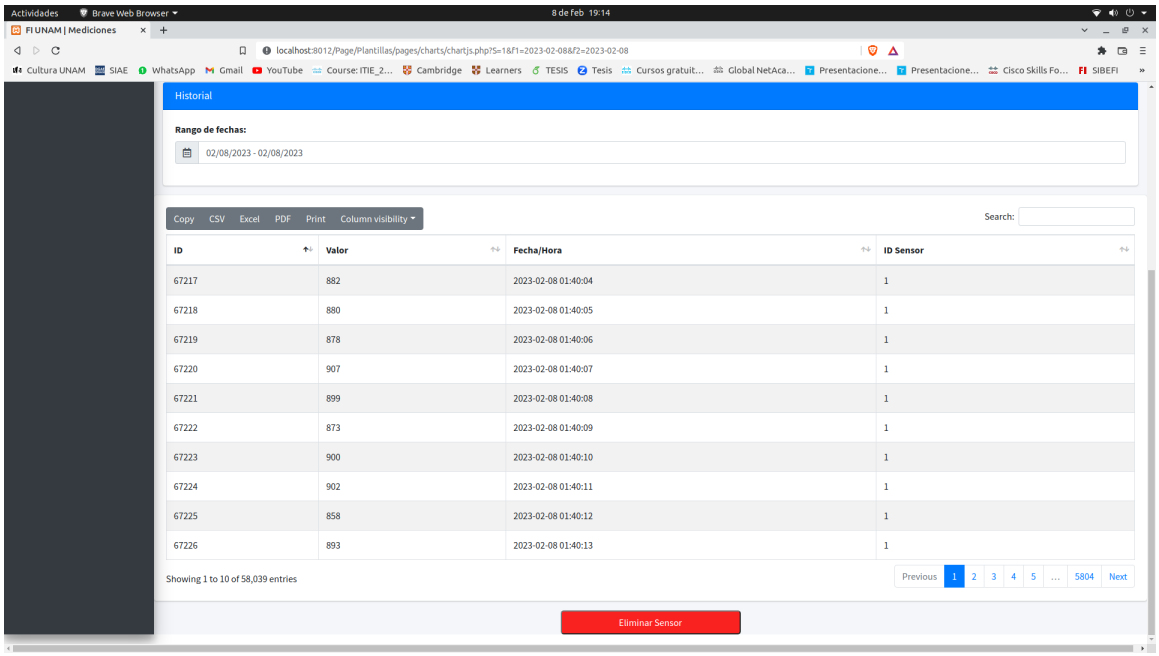

Figura 4.15: Consulta de datos antiguos - Plataforma Web propia.

catálogo de sensores(MySQL) con su respectivo ID, el nombre que tecleó el usuario, y un valor unitario por defecto para la columna de activo.

Para programar esto, se declaró una variable de entorno S que hace referencia a ID_Sensor. De manera que dependiendo del valor de S es que la plataforma sabe diferenciar entre un sensor u otro, siendo así capaz de consultar en la base de datos la información correspondiente a él. En la figura 4.18 puede observarse que, una vez realizados todos los pasos anteriores, ya puede accederse al sensor creado, así como la variable de entorno S ya mencionada.

A manera de evitar confusiones con los usuarios y el funcionamiento de la plataforma, no está permitido que dos sensores tengan un mismo nombre, por lo que, si se crea un nuevo sensor con un nombre ya existente, se mostrará en pantalla lo que se observa en la figura 4.19.

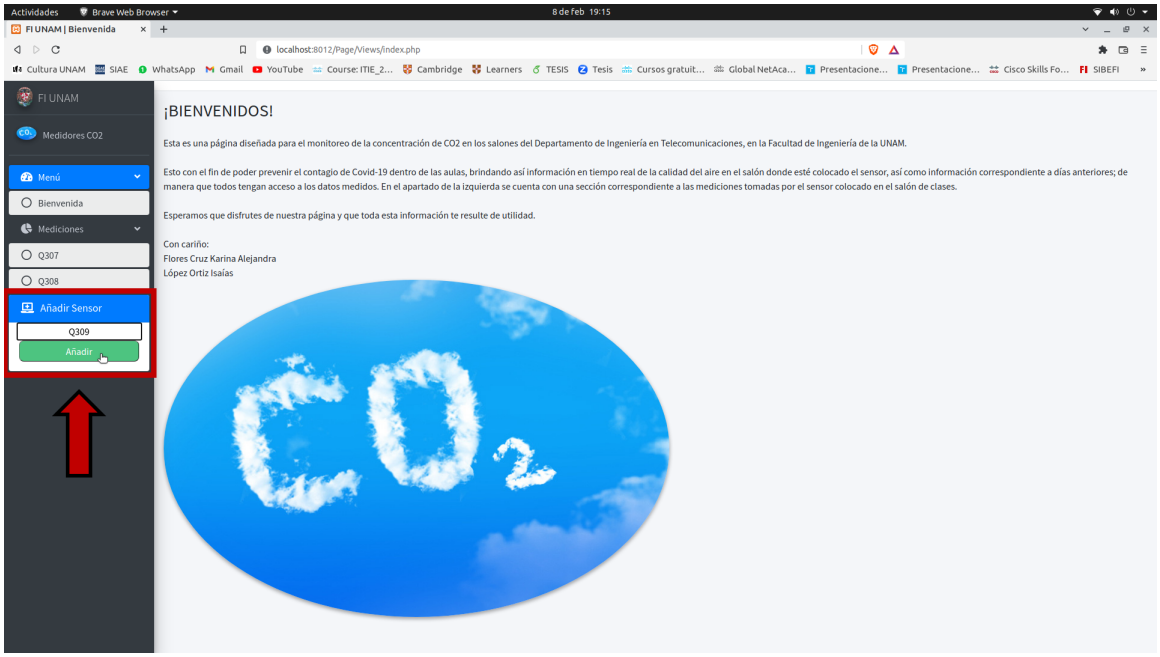

Figura 4.16: Agregar nuevo sensor - Plataforma Web propia.

Figura 4.17: Confirmación de sensor agregado - Plataforma Web propia.

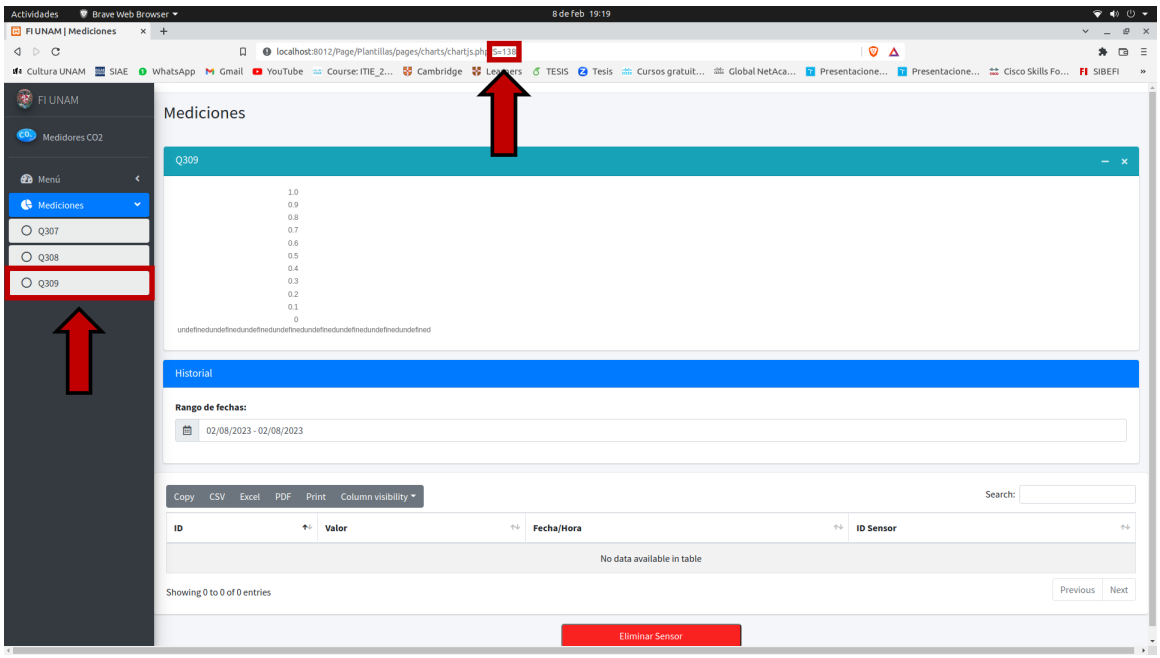

Figura 4.18: Nuevo sensor agregado - Plataforma Web propia.

Figura 4.19: Aviso de sensor ya existente - Plataforma Web propia.

➤ **Eliminar y restaurar sensor.**

Así como se debe tener una opción para crear nuevos sensores, también es importante poder eliminar aquellos que ya no son necesarios, por lo que, en la parte inferior de la pestaña correspondiente a cada sensor, se tiene un botón para realizar dicha acción, de manera que al dar click sobre él se le preguntará al usuario si está seguro de que quiere eliminar el sensor. Esto se muestra en la figuras 4.20 y 4.21.

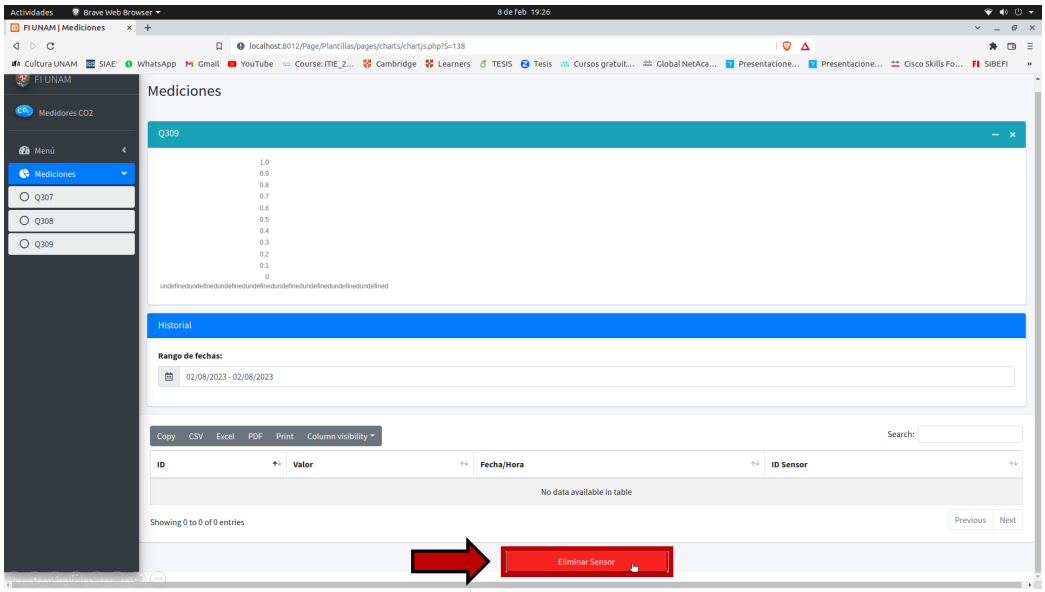

Figura 4.20: Eliminar sensor - Plataforma Web propia.

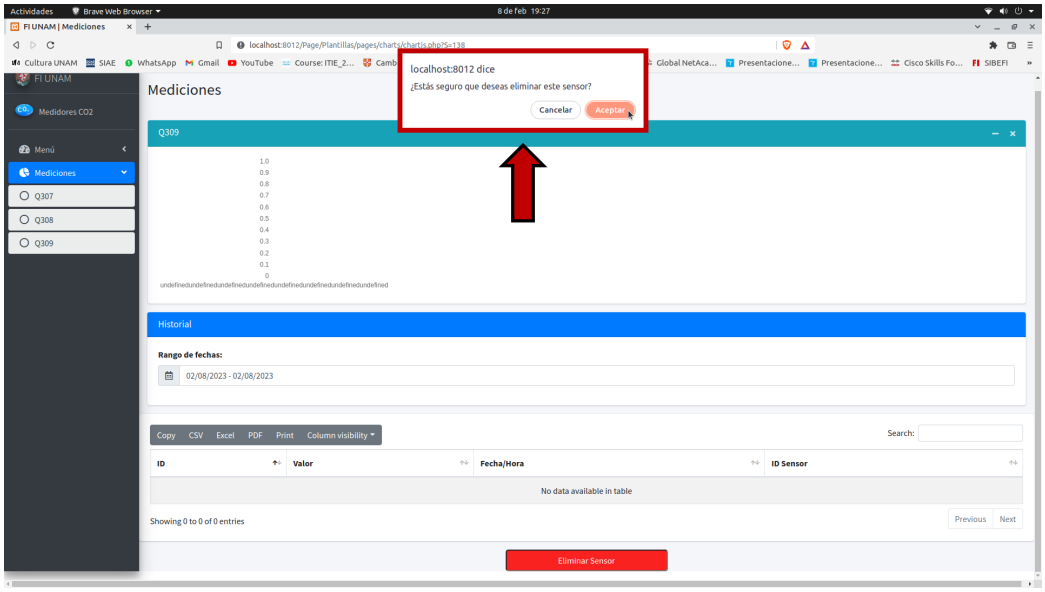

Figura 4.21: Confirmación para eliminar sensor - Plataforma Web propia.

A pesar de que al eliminar un sensor este ya no se mostrará en la plataforma, este no se borra permanentemente, sino que únicamente su columna de activo en MySQL cambia a cero. Por lo que, si se eliminó un sensor y por alguna razón se quiere recuperar toda su información y poder seguir trabajando con él, basta con crearlo de nuevo con el mismo nombre que se le asignó desde un inicio. Al hacer esto, se mostrará en la plataforma un letrero de que tal sensor ha sido restaurado, lo que se muestra en la figura 4.22.

Figura 4.22: Aviso de sensor restaurado - Plataforma Web propia.

5 Experimentación y resultados

Ya teniendo hasta este momento todo lo referente al desarrollo y diseño tanto de hardware como de software, resultaba pertinente realizar experimentos con el sistema creado. De manera que pudiéramos simular diferentes escenarios y así conocer de mejor manera el funcionamiento y comportamiento de la plataforma creada.

5.1. Exp 1: Sensor en ambiente muy contaminado.

Dentro del primer experimento, se tenía como propósito observar el comportamiento del sistema al ser expuesto a un ambiente muy contaminado. Para esto, se provocó un pequeño incendio cerca del sensor MQ135, de forma que todo el humo generado fuera detectado por este y dicha polución súbita en el ambiente se viera reflejada en la plataforma Web.

En cuanto al log de los datos se refiere, se utilizó un intervalo de cinco segundos entre cada lectura tomada por el sensor.

En la figura 5.1, se muestra el comportamiento del sistema en condiciones normales. Donde podemos observar que los valores que estaban siendo medidos por el sensor se encontraban entre 285 y 315 PPM, los cuales, están dentro de un rango característico de un aire limpio.

Seguido de haber observado el comportamiento del sistema en condiciones normales, se realizó la correspondiente exposición del sensor al huno para simular un ambiente contaminado. En la figura 5.2, se observa el precipitado aumento de los valores medidos por el sensor, pasando de 300 a 800 PPM en menos de un minuto, lo cual por supuesto representa un valor peligroso.

Después de exponer el sensor MQ135 al humo generado y observar lo rápido que este hacía que la habitación resultara nada favorable para la salud de quienes se encontraran en esta, se retiró el humo del sensor (sin mitigar el incendio) y se observaron de nuevo los valores mostrados en la plataforma. En la figura 5.3 se muestra que prácticamente de inmediato los valores medidos por el sensor van disminuyendo, yendo de 900 hasta 400 PPM. Donde a pesar de que el valor más bajo (400 PPM) aún no se encuentra en el mismo rango de los valores que se tenían en un inicio, en definitiva, representa una mejora en comparación de los 800 PPM que se tenían.

Por último para este primer experimento, se extinguió por completo el incendio y se ventiló la habitación. En la figura 5.4 se observa que al realizar esto el sensor ya mide valores favorables y mucho más parecidos a los que se tenían antes de provocar

el incendio.

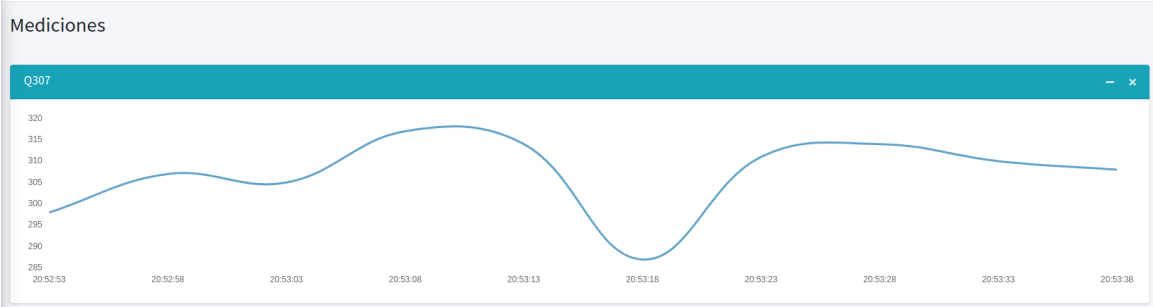

Figura 5.1: Valores tomados en condiciones normales.

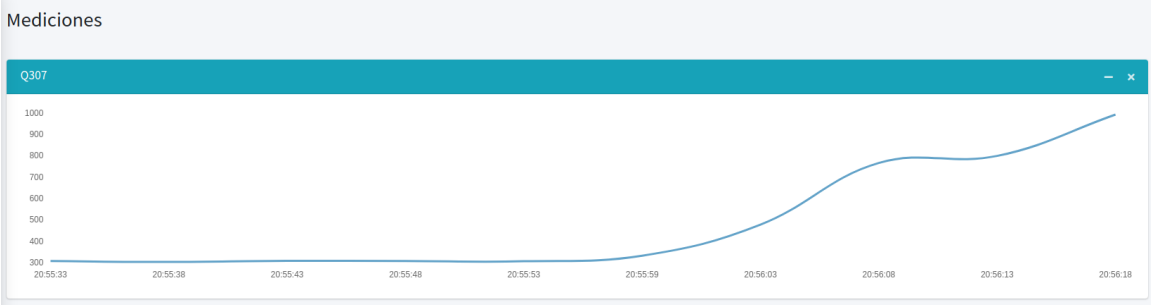

Figura 5.2: Valores tomados en ambiente muy contaminado.

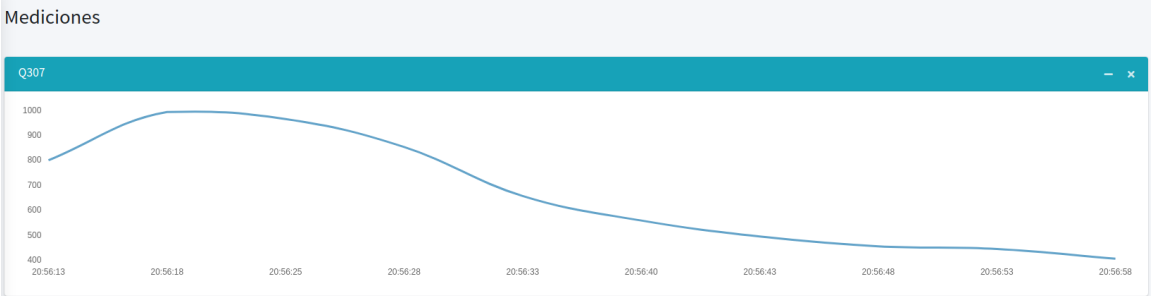

Figura 5.3: Valores tomados al alejar el humo del sensor.

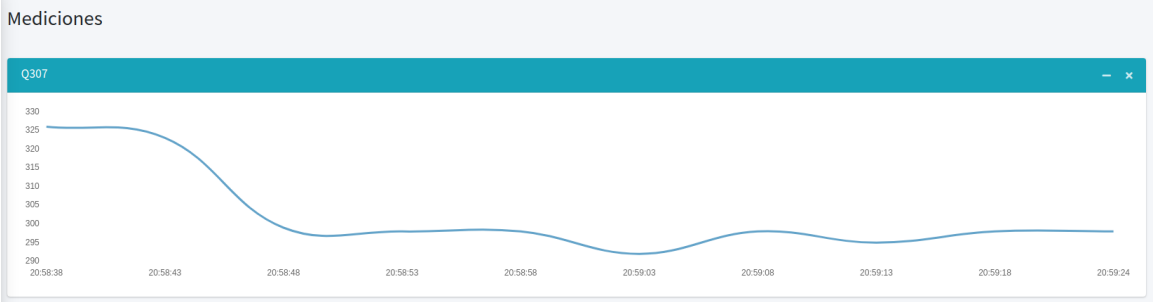

Figura 5.4: Valores tomados al ventilar habitación.

5.2. Exp 2: Diferentes tasas de lectura de datos.

Tomando en cuenta que antes de que la información proveída por los sensores sea visible en la plataforma Web, esta tiene que pasar primero por varios servidores. Para este segundo experimento, se tuvo como objetivo observar el comportamiento y desempeño del sistema a nivel de software cuando este trabaja a diferentes tasas de lectura de datos.

La referencia que se tomó para poder cuantificar el throughput del sistema, fue la comparación de cuántos datos debieron haber sido subidos según la tasa utilizada con los datos que realmente se observaron en la plataforma. Asimismo, para poder asegurar que los resultados obtenidos fueron efectivos para poder llegar a una conclusión legítima sobre el desempeño de nuestro sistema, cada experimento tuvo una duración de aproximadamente 12 horas.

- **Tasa de 0.1 [datos/s].**

Se inició con el log de los datos a las 00:30:03 hrs y se finalizó a las 12:29:56 hrs. Se tuvieron un total de 4,309 mediciones a lo largo de ese tiempo, cuando en teoría tuvieron que haberse subido 4,320, por lo que para esta primera tasa se tuvo un throughput de:

$$\epsilon_1 = \frac{4309}{4320} * 100 = 99.7\,\% \tag{5.1}$$

Para poder saber con exactitud cuántos datos fueron logueados en el lapso que duró el experimento, simplemente se usó la herramienta con la que cuenta nuestra plataforma para consultar datos antiguos, de manera que al realizar la resta de los valores correspondientes al ID del último y primer valores subidos se obtuvo la cantidad de datos registrados por la plataforma, cosa que se visualizaba tal y como se muestra en la tabla 5.1.

Tabla 5.1: Log de primer y último datos con tasa de 0.1 [datos/s].

ID	Valor	FechaHora	ID_Sensor
659545	509	2023-02-20 00:30:03	1
663854	395	2023-02-20 12:29:56	1

- **Tasa de 0.2 [datos/s].**

Para esta tasa de lectura de datos a 0.2 datos/segundo, el experimento inició a las 00:20:00 hrs y finalizó a las 12:20:16 hrs. Registrándose un total de 8,636 mediciones, pero dada la tasa seleccionada debieron haberse subido 8,640 datos, obteniendo el siguiente throughput:

$$\epsilon_2 = \frac{8636}{8640} * 100 = 99.95\,\% \tag{5.2}$$

Así como se hizo anteriormente, para cuantificar los datos subidos a lo largo del experimento, se tomó la diferencia del primer y último datos medidos, apoyándonos de nuevo en la funcionalidad de nuestra plataforma de consultar datos antiguos, visualizándose como se observa en la tabla 5.2.

Tabla 5.2: Log de primer y último datos con tasa de 0.2 [datos/s].

ID	Valor	FechaHora	ID_Sensor
650258	487	2023-02-19 00:20:00	1
658896	272	2023-02-19 12:20:16	1

- **Tasa de 0.5 [datos/s].**

Teniendo un inicio del experimento a las 01:40:01 hrs y finalización a las 13:40:00 hrs, se registraron 21,314 mediciones, cuando en realidad debieron haber sido 21,600, teniendo el siguiente throughput:

$$\epsilon_3 = \frac{21314}{21600} * 100 = 98.67\,\% \tag{5.3}$$

Al utilizar nuestra plataforma para realizar la cuantificación de los datos registrados, esto se mostraba tal y como se observa en la tabla 5.3.

Tabla 5.3: Log de primer y último datos con tasa de 0.5 [datos/s].

ID	Valor	FechaHora	ID_Sensor
174829	415	2023-02-13 01:40:01	1
196143	291	2023-02-13 13:40:00	1

- **Tasa de 1 [dato/s].**

Este experimento con tasa de 1 dato/segundo sin duda podría ser considerado como el más relevante. Ya que este fue el primero en realizarse cronológicamente hablando, por lo que fue aquí donde más problemas se llegaron a presentar, teniendo que ser interrumpidas las ejecuciones del sistema para solucionar los problemas que se tuvieron.

El primer problema que se presentó fue que, a las dos horas de iniciar el log de los datos, la ejecución del programa de Arduino se vio interrumpida debido a que la comunicación MQTT dejaba de funcionar correctamente, mostrándose en el monitor

serial de Arduino el mensaje *Attempting MQTT connection...failed, rc=-2 try again in 5 seconds.* Para solucionar lo anterior, se modificó el programa de Arduino, de forma que, si dicho mensaje citado anteriormente se mostraba 10 veces seguidas, el ESP32 se durmiera por 2 segundos, se volviera a conectar y continuara con la ejecución.

Otro problema que llegó a presentarse era que el ESP32 no siempre se conectaba exitosamente al Access Point, mostrándose en el monitor serial de Arduino que estaba intentando dicha conexión, pero nunca avanzando de ese punto, por lo que, así como se hizo para el primer problema descrito, para solucionar esto se codificó que, si dicho intento de conexión tardaba más de 12 segundos, se reiniciara la ESP32 después de dormirse por dos segundos.

La implementación de lo anterior mencionado se muestra en la figura 5.5, donde se puede observar el monitor serial de Arduino correspondiente a la ejecución realizada para este experimento, reiniciándose un total de tres veces.

Ya habiendo reparado los problemas descritos, ahora sí se pudo realizar el correspondiente experimento, iniciando a las 01:31:05 hrs y finalizando a las 13:31:05 hrs. Teniendo 42,850 datos registrados, cuando debieron haber sido 43,200, teniendo el siguiente desempeño:

$$\epsilon_4 = \frac{42850}{43200} * 100 = 99.1\,\% \tag{5.4}$$

Consultando los datos antiguos mediante la funcionalidad que ofrece nuestra plataforma para contabilizar los datos registrados, esto se mostró tal y como se observa en la tabla 5.4.

Tabla 5.4: Log de primer y último datos con tasa de 1 [dato/s].

ID	Valor	FechaHora	ID_Sensor
131378	361	2023-02-09 01:31:05	1
174232	366	2023-02-09 13:31:05	1

```
Connecting to Skyserver
.....
WiFi connected
IP address:
192.168.100.38
Attempting MQTT connection...connected
Attempting MQTT connection...connected
Attempting MQTT connection...failed, rc=-2 try again in 5 seconds. Restarting.
ets Jun  8 2016 00:22:57

rst:0x5 (DEEPSLEEP_RESET),boot:0x13 (SPI_FAST_FLASH_BOOT)
configsip: 0, SPIWP:0xee
clk_drv:0x00,q_drv:0x00,d_drv:0x00,cs0_drv:0x00,hd_drv:0x00,wp_drv:0x00
mode:DIO, clock div:1
load:0x3fff0018,len:4
load:0x3fff001c,len:1216
ho 0 tail 12 room 4
load:0x40078000,len:10944
load:0x40080400,len:6388
entry 0x400806b4

Connecting to Skyserver
.....
WiFi connected
IP address:
192.168.100.38
Attempting MQTT connection...connected
Attempting MQTT connection...failed, rc=-2 try again in 5 seconds. Restarting.
ets Jun  8 2016 00:22:57

rst:0x5 (DEEPSLEEP_RESET),boot:0x13 (SPI_FAST_FLASH_BOOT)
configsip: 0, SPIWP:0xee
clk_drv:0x00,q_drv:0x00,d_drv:0x00,cs0_drv:0x00,hd_drv:0x00,wp_drv:0x00
mode:DIO, clock div:1
load:0x3fff0018,len:4
load:0x3fff001c,len:1216
ho 0 tail 12 room 4
load:0x40078000,len:10944
load:0x40080400,len:6388
entry 0x400806b4

Connecting to Skyserver
..............
WiFi connected
IP address:
192.168.100.38
Attempting MQTT connection...connected
Attempting MQTT connection...connected
Attempting MQTT connection...connected
Attempting MQTT connection...connected
Attempting MQTT connection...connected
Attempting MQTT connection...connected
Attempting MQTT connection...failed, rc=-2 try again in 5 seconds. Restarting.
```

Figura 5.5: Reseteo de circuito durante experimento.

- **Tasa de 2 [datos/s].**

Para la tasa correspondiente a este experimento, se inició la ejecución a las 01:40:00 hrs y finalizó a las 13:40:00 hrs. Registrándose 85,507 datos, a comparación de los 86,400 que en teoría debieron haberse presentado. Teniendo el siguiente throughput:

$$\epsilon_5 = \frac{85507}{86400} * 100 = 98.9\,\% \tag{5.5}$$

Revisando la plataforma para cuantificar los datos logueados, esto se mostraba como se observa en la tabla 5.5.

Tabla 5.5: Log de primer y último datos con tasa de 2 [datos/s].

ID	Valor	FechaHora	ID_Sensor
197246	432	2023-02-14 01:40:00	1
282753	286	2023-02-14 13:40:00	1

- **Tasa de 5 [datos/s].**

Para esta última tasa de 5 datos/segundo, se inició la ejecución a las 01:00:00 hrs y se finalizó a las 13:00:00. Teniendo 152,555 mediciones realizadas de las 216,000 que debieron haber sido registradas. Por lo que obteniendo el throughput del sistema se tiene:

$$\epsilon_6 = \frac{152555}{216000} * 100 = 70.6\,\% \tag{5.6}$$

Utilzando la plataforma para cuantificar los datos registrados, esto se mostraba como se observa en la tabla 5.6.

Tabla 5.6: Log de primer y último datos con tasa de 5 [datos/s].

ID	Valor	FechaHora	ID_Sensor
494214	496	2023-02-17 01:00:00	1
646769	297	2023-02-17 13:00:00	1

Con todo lo anterior dicho en este experimento, se creyó pertinente realizar la tabla 5.7, la cual relaciona todas las tasas de lectura de datos con las que se experimentó junto con sus recpectivos desempeños y el conteo de los datos recibidos exitosamente. Asimismo, para una mejor visualización de lo anterior, se realizó la gráfica correspondiente a dicha tabla, la cual se observa en la figura 5.6. En ambos puede notarse que cuando la cantidad de paquetes aumenta, el throughput decae debido a que comienzan a ocurrir problemas en el canal.

Tabla 5.7: Resultados obtenidos a diferentes tasas de lectura de datos.

Tasa [datos/s]	Datos enviados	Datos recibidos	Desempeño
0.1	4320	4309	99.7 %
0.2	8640	8636	99.95 %
0.5	21,600	21,314	98.67 %
1	43,200	42,850	99.1 %
2	86,400	85,507	98.9 %
5	216,000	152,555	70.6 %

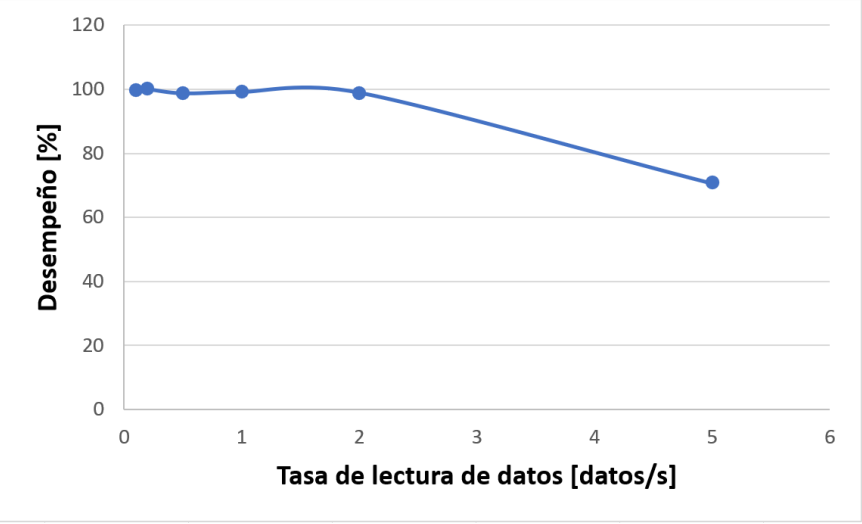

Figura 5.6: Desempeño de sistema a distintas tasas de lectura de datos.

5.3. Exp 3: Máxima tasa de lectura de datos.

Dentro de este tercer experimento, se tuvo como propósito llevar al límite las capacidades de nuestro sistema, de manera que podamos determinar la máxima tasa de lectura de datos de este. Para realizar esto, en el script de Arduino se programó que el lapso entre cada lectura tomada por el sensor MQ135 sea de 1 [ms]. De manera que, en teoría, se suban 1000 mediciones por segundo y con esto analizar el comportamiento de nuestro sistema a dicha tasa.

Para poder tener resultados y conclusiones efectivas en este experimento, no se realizó una sola prueba del sistema con las condiciones presentadas, en total se realizaron cinco. Cabe resaltar que estas no duraron tanto como las descritas en el experimento anterior debido a que se tenía una gran cantidad de muestras en poco tiempo.

5.3.1. Primera prueba

Esta primera prueba tuvo una duración de seis minutos aproximadamente, y de nuevo aprovechando las capacidades de nuestra plataforma de mostrar datos antiguos, se pudo saber cuántas mediciones se tomaron en el lapso mencionado, lo cual se visualizaba como se muestra en la tabla 5.8.

Tabla 5.8: Resultados obtenidos en primera prueba a tasa de 1000 [datos/s].

ID	FechaHora
29274	2023-02-06 15:10:00
29990	2023-02-06 15:16:00

Por lo que obteniendo la resta de los IDs inicial y final de lo realizado en la primera prueba se obtiene lo siguiente:

$$Tasa_1 = \frac{716\ mediciones}{360\ segundos} = 1.98\ [Mediciones/s] \tag{5.7}$$

5.3.2. Segunda prueba

Para esta segunda prueba, se realizó la experimentación por media hora, mostrándose en la plataforma la información correspondiente a la primera y última mediciones tomadas tal y como se muestra en la tabla 5.9.

Tabla 5.9: Resultados obtenidos en segunda prueba a tasa de 1000 [datos/s].

ID	FechaHora
31186	2023-02-06 17:15:00
36908	2023-02-06 17:45:00

De manera que a partir de dicha información se pudo realizar el siguiente cálculo:

$$Tasa_2 = \frac{5722\ mediciones}{1800\ segundos} = 3.17\ [Mediciones/s] \tag{5.8}$$

En este experimento en particular que tuvo una duración mayor que la primera prueba realizada, se presentaron ciertos problemas que valen la pena ser mencionados. Pues en varias ocasiones el sistema se quedaba pasmado y por algunos segundos dejaba de realizar mediciones, cosa que ocurrió siete veces a lo largo de esta prueba. Sumando un total de 17 segundos en los que el sistema no subió nada, representando un 4.7 % del tiempo total de la prueba.

De algo que nos percatamos al haber realizado esta segunda prueba, es que realmente el sistema sí llegó a sufrir de una sobrecarga de los datos que querían subirse. Esto se concluyó debido a que, a pesar de que la ESP32 ya había sido desconectada de la alimentación eléctrica, se seguían subiendo datos a la gráfica correspondiente. Por lo que se llegó a la conclusión que, una vez medidos los valores, estos son insertados en una cola antes de ser subidos a la base de datos y mostrarse en la plataforma, aunque si tal medición de datos es mayor que la tasa a la que pueden ser subidos, se van a presentar estos estancamientos en el sistema. Cosa que, por supuesto debe evitarse a toda costa, pues la hora a la que fueron subidos no corresponde a la que fueron medidos, provocando una discrepancia entre los datos.

Al revisar la base de datos, se observó que lo máximo que se llegó a transmitir en un segundo son cuatro mediciones. Aunque esto no es para nada constante, pues en un segundo se llegaron a realizar tres, dos y hasta una única medición.

5.3.3. Tercera prueba

Para esta tercera prueba, la cual duró una hora, en la plataforma se mostraban tal y como se observa en la tabla 5.10 los datos tomados para realizar el conteo de las mediciones tomadas a lo largo del experimento.

Tabla 5.10: Resultados obtenidos en tercera prueba a tasa de 1000 [datos/s].

ID	FechaHora
38867	2023-02-06 19:00:00
45426	2023-02-06 19:40:00

Por lo que tomando la información que se muestra en dicha tabla se obtiene el siguiente promedio:

$$Tasa_3 = \frac{6559\ mediciones}{1800\ segundos} = 3.6\ [Mediciones/s] \tag{5.9}$$

Así como pasó en la segunda prueba, en esta también ocurrió que incluso cuando ya se había desconectado el ESP32 de la alimentación eléctrica, se seguía observando que en la plataforma había datos aun subiéndose. De esa manera, así como ya había sucedido anteriormente, la tasa de medición de datos era muy superior a la capacidad de carga, por lo que muchos de estos fueron encolados antes de poder ser visualizados.

5.3.4. Cuarta prueba

Al momento de utilizar nuestra plataforma para contar los datos tomados a lo largo del experimento, esto se mostró tal y como se observa en la tabla 5.11. Cabe recalcar que esta cuarta prueba tuvo una duración de una hora.

Tabla 5.11: Resultados obtenidos en cuarta prueba a tasa de 1000 [datos/s].

ID	FechaHora
47982	2023-02-06 20:55:00
55363	2023-02-06 21:25:00

Por lo que de lo anterior se llegó al siguiente promedio:

$$Tasa_4 = \frac{7381\ mediciones}{1800\ segundos} = 4.1\ [Mediciones/s] \tag{5.10}$$

Y así como ya había sucedido anteriormente, en esta cuarta prueba también seguían subiéndose datos después de haber apagado el ESP32.

5.3.5. Quinta prueba

Para esta última prueba, se tuvo una duración mucho mayor que las anteriores, teniendo una duración total de 12 horas de ejecución. Mostrándose en nuestra plataforma tal y como se observa en la tabla 5.12.

Tabla 5.12: Resultados obtenidos en quinta prueba a tasa de 1000 [datos/s].

ID	FechaHora
885001	2023-02-24 01:18:00
1081533	2023-02-24 13:18:00

De lo anterior obteniendo la siguiente tasa promedio:

$$Tasa_5 = \frac{196532\ mediciones}{43200\ segundos} = 4.5\ [Mediciones/s] \tag{5.11}$$

De las cinco anteriores pruebas realizadas, podemos obtener su correspondiente promedio para así concluir cuál es la máxima tasa de transmisión de nuestro sistema:

$$Tasa\ maxima = \frac{1.98 + 3.17 + 3.6 + 4.1 + 4.5}{5} = 3.47\ [Mediciones/s] \tag{5.12}$$

Tomando todo lo mencionado a lo largo de este experimento, se puede observar que este sistema no está diseñado en lo absoluto para una rápida tasa de medición de datos, pues en promedio solo fue capaz de realizar tres en un segundo y siempre teniendo cierta inestabilidad al momento de hacerlo.

Se puede decir con bastante seguridad que se cumplió el objetivo de probar los límites de nuestro sistema, los cuales sinceramente son algo para nada remarcable. Sin embargo, esto no es razón para verlo como un punto negativo, pues nunca se tuvo como objetivo una rápida lectura de datos. Aparte de que es innecesario que esto se haga de esta manera, pues en condiciones normales el sensor toma las mediciones cada 2.5 segundos, medida que es más que suficiente para monitorear la calidad del aire y que el sistema pueda alertar en caso de haber un valor por encima de los 800 PPM. Y en caso de que en realidad quiera hacerse un monitoreo del aire mucho más preciso y detallado, puede sin problema subirse la tasa a una medición por segundo, asegurando de esta forma que el sistema funcionará de manera estable y confiable.

5.4. Exp 4: Varios sensores a un mismo tópico.

Para este cuarto experimento, se tuvo como propósito observar el comportamiento general del sistema cuando se tienen cuatro sensores conectados a un único Access Point a la vez que se suben los datos a una misma base de datos. De igual forma, para poder realizar un mejor análisis de esto, se ejecutaron dos pruebas: una con una tasa de 0.2 [datos/s], y otra a una tasa de 1 [dato/s]; donde cada prueba tuvo una duración de 12 horas para obtener una cantidad suficiente de muestras y asegurar conclusiones efectivas en el experimento.

5.4.1. Primera prueba: 0.2 [datos/s]

Para empezar, a cuatro circuitos se les cargó el correspondiente script de Arduino para que enviaran sus datos al mismo publisher de MQTT, así como la configuración de la tasa de medición ya mencionada. Procurando también que la colocación de los sensores fuera en las esquinas de la habitación para asegurar una máxima cobertura.

Para poder contabilizar cuántos datos fueron subidos con éxito a lo largo de la prueba, se utilizó la herramienta provista por nuestra plataforma para consultar datos antiguos, lo cual se observaba tal y como se muestra en la tabla 5.13.

Tabla 5.13: Resultados obtenidos en primera prueba.

ID	FechaHora
1120858	2023-03-06 01:30:00
1155302	2023-03-06 13:30:00

Tomando los datos mostrados en la tabla 5.13, podemos obtener el número de datos que se perdieron a lo largo de la prueba y así calcular el desempeño que presentó el sistema con las condiciones dadas:

$$\epsilon_1 = \frac{34444}{34560} * 100 = 99.66\,\% \tag{5.13}$$

5.4.2. Segunda prueba: 1 [dato/s]

Para esta segunda prueba, se realizó el mismo procedimiento ya mencionado, cambiando únicamente en el script de Arduino la tasa de lectura de datos, siendo ahora de 1 [dato/s].

Realizando el conteo de los datos subidos con éxito en nuestra plataforma, los resultados obtenidos se observan en la tabla 5.14.

Tabla 5.14: Resultados obtenidos en segunda prueba.

ID	FechaHora
1165342	2023-03-06 21:30:00
1312062	2023-03-07 09:30:00

Ya contando con tal información, era posible realizar el correspondiente cálculo para obtener el desempeño:

$$\epsilon_2 = \frac{146720}{172800} * 100 = 84.9\,\% \tag{5.14}$$

Al realizar este experimento, se pudo notar que sin ningún problema es posible conectar varios sensores a un mismo Access Point para realizar mediciones de una misma variable física, en caso de querer tener mayor cobertura en una misma habitación.

Esto resulta útil en escenarios donde la locación donde se instale el sensor sea un lugar de bajo perfil y con un solo Access Point. Cuestión que podría presentarse en casos donde una familia o negocio pequeño deseen contar con un sistema como este, pues, en ambos casos es usual que solamente se cuente con un Access Point para que todos en el domicilio se conecten a internet. De manera que con este experimento quedó demostrado que realmente es poco lo que se necesita para contar con un sistema como el nuestro, siendo incluso posible contar con varios sensores conectados y no habrá problemas de funcionalidad ni desempeño o confiabilidad.

5.5. Exp 5: Varios sensores a diferentes tópicos.

Similar a lo desarrollado anteriormente, para este quinto experimento también se trabajó con cuatro sensores midiendo al mismo tiempo. La diferencia dentro de este experimento fue que, aunque todos los sensores se conectaban al mismo Access Point, cada uno tenía una gráfica individual para realizar el log de los datos.

A diferencia del experimento anterior, donde se colocaron los cuatro sensores en las esquinas de una habitación, para este se colocó cada uno en diferentes cuartos del domicilio. De nuevo realizando dos pruebas a diferentes tasas de muestreo y cada una teniendo una duración aproximada de 12 horas.

En la figura 5.7 puede observarse el alta de las 4 gráficas correspondientes a las utilizadas para este experimento.

Figura 5.7: Sensores dados de alta.

5.5.1. Primera prueba: 0.2 [datos/s]

Ya realizando el log de los datos con los cuatro sensores colocados, algo que se observó en la base de datos, específicamente en la columna de ID_Sensor, es que los sensores siguieron en todo momento una secuencia ordenada para subir los datos. Lo cual puede observarse en la figura 5.8.

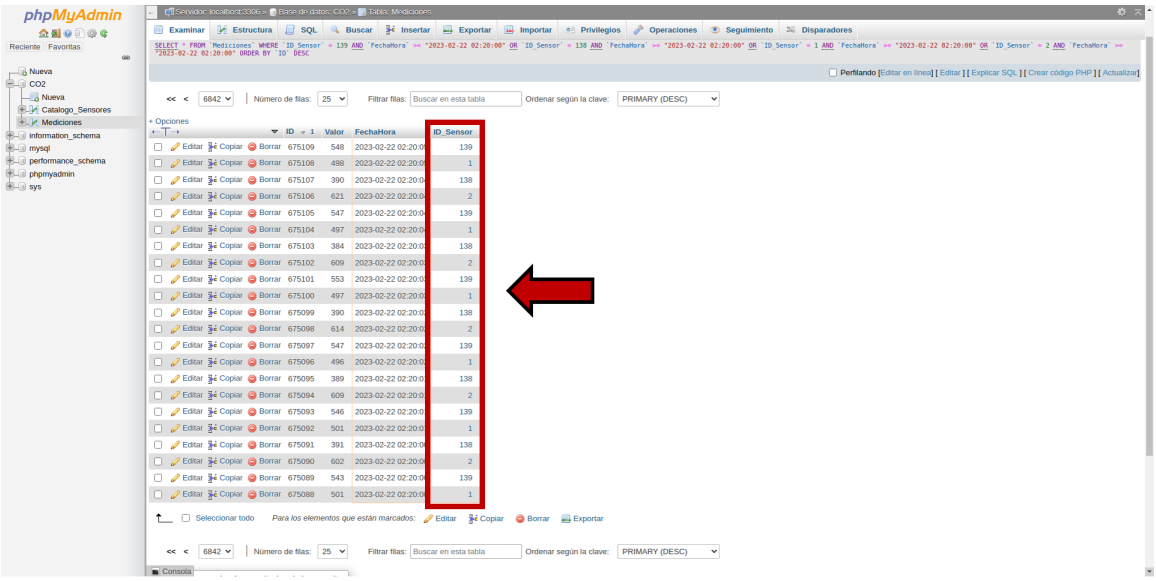

Figura 5.8: Secuencia de carga de datos.

Enfocándonos ahora en el desempeño obtenido para esta primera prueba, así como ya se hizo anteriormente, se utilizó la plataforma web para consultar los datos antiguos necesarios, lo cual se mostraba como en la tabla 5.15.

Tabla 5.15: Resultados obtenidos en primera prueba.

ID	FechaHora
1083856	2023-03-05 12:30:00
1118265	2023-03-06 00:30:00

Por lo que, tomando tal información, puede realizarse el correspondiente cálculo para obtener el desempeño:

$$\epsilon_1 = \frac{34409}{34560} * 100 = 99.5\,\% \tag{5.15}$$

5.5.2. Segunda prueba: 1 [dato/s]

Configurando ahora en los sensores una tasa de lectura de 1 [dato/s], se volvió a realizar el procedimiento correspondiente para observar el comportamiento del sistema con estas nuevas condiciones. Consultando de igual forma los datos antiguos para poder contabilizar aquellos que fueron cargados exitosamente, lo cual se mostraba como se observa en la tabla 5.16.

Tabla 5.16: Resultados obtenidos en segunda prueba.

ID	FechaHora
675088	2023-02-22 02:20:00
832478	2023-02-22 14:19:59

Tomando la información anteriormente mostrada, se realizó el cálculo del desempeño correspondiente al sistema:

$$\epsilon_2 = \frac{157393}{172800} * 100 = 91.08\,\% \tag{5.16}$$

Sirviendo como complemento del experimento anterior, en este pudo notarse que también es posible contar con un solo Access Point, varios sensores y varias gráficas por cada sensor instalado. Por lo que nuestro sistema no solo es útil para tener un mayor rango de medición en una misma habitación, sino también tener todo un sistema de sensores (cada uno con su respectiva gráfica) en un domicilio pequeño con varias habitaciones en las que resulte de interés realizar la medición de la calidad del aire.

Observando esto puede percibirse qué tan versátil y flexible es nuestro sistema, pues con tan solo los sensores necesarios y un único Access Point (el cual puede ser el mismo que se utiliza para conectarse a internet) es posible cubrir un domicilio entero, esto sin sacrificar la funcionalidad ni desempeño. Sin embargo, esta flexibilidad mencionada tampoco es ilimitada, pues algo que también debe tomarse muy en cuenta es que el rango que puede cubrirse con WiFi en realidad no es de muchos metros (tomando en cuenta que el ESP32 trabaja en la banda de 2.4 [GHz] a 17 [dBm]), por lo que, aunque nuestro sistema funciona bien con varios sensores a diferentes gráficas y un solo Access Point, la distancia entre tal Access Point y los sensores puede ser una limitante.

5.6. Exp 6: Diferentes distancias entre AP y sensor.

Hasta este momento, el Access Point y el sensor habían estado muy cercanos el uno del otro, por lo que la conectividad Wi-Fi nunca fue un factor para afectar el desempeño del sistema. Sin embargo, en escenarios reales puede que ya se presente una mayor distancia entre estos componentes, aparte de que también puede haber objetos de por medio que afecten la conectividad. Es por eso que, para este experimento, se realizaron pruebas colocando el AP y sensor a una mayor distancia entre sí, esto con el propósito de observar si se tenía un cambio en el desempeño.

Todas las pruebas siguientes tuvieron una duración de 24 horas y en el sensor se configuró una tasa de 1 [medición/seg].

5.6.1. Primera prueba: 0.5 [m]

Así como ya se ha estado realizando, para contabilizar los datos recibidos se utilizó la funcionalidad de nuestra plataforma de recuperar datos antiguos. Rescatando el primero y último valor de la correspondiente prueba, se tuvo el resultado tal y como se muestra en la tabla 5.17.

Tabla 5.17: Resultados obtenidos en primera prueba.

ID	FechaHora
1363053	2023-04-05 02:00:00
1406118	2023-04-05 13:59:57

Tomando los datos mostrados en la tabla 5.17, podemos obtener el número de datos que se perdieron a lo largo de la prueba y así calcular el desempeño que presentó el sistema con las condiciones dadas:

$$\epsilon_1 = \frac{43068}{43200} * 100 = 99.69\,\% \tag{5.17}$$

5.6.2. Segunda prueba: 5 [m]

Colocando ahora el AP y sensor a una distancia de 5 [m] entre sí, se volvieron a contabilizar los datos recibidos exitosamente, teniendo como resultado lo que se muestra en la tabla 5.18.

Tabla 5.18: Resultados obtenidos en segunda prueba.

ID	FechaHora
1414321	2023-04-10 04:30:00
1457275	2023-04-10 16:30:00

Tomando los datos mostrados en la tabla 5.18, podemos obtener el número de datos que se perdieron a lo largo de la prueba y así calcular el desempeño que presentó el sistema con las condiciones dadas:

$$\epsilon_1 = \frac{42964}{43200} * 100 = 99.45\,\% \tag{5.18}$$

5.6.3. Tercera prueba: 8 [m]

Para esta tercera prueba, el AP y sensor fueron colocados a 8 [m] entre sí. Y al momento de obtener el primero y último datos de la prueba, se pudieron contabilizar y así realizar el correspondiente cálculo para obtener el desempeño, observándose como se muestra en la tabla 5.19.

Tabla 5.19: Resultados obtenidos en tercera prueba.

ID	FechaHora
1461805	2023-04-13 01:00:00
1504703	2023-04-13 13:00:00

Tomando los datos mostrados en la tabla 5.19, podemos obtener el número de datos que se perdieron a lo largo de la prueba y así calcular el desempeño que presentó el sistema con las condiciones dadas:

$$\epsilon_1 = \frac{42898}{43200} * 100 = 99.3\,\% \tag{5.19}$$

5.6.4. Cuarta prueba: 45 [m]

Hasta este momento, el incremento de la distancia entre el AP y sensor había sido muy gradual, y la disminución del desempeño también se presentó en menor medida. Por lo que para este experimento hubo una mayor separación entre los componentes, colocándolos a la mayor distancia posible entre ellos justo antes de perder la conexión, estando ahora a 45 [m] entre sí.

Recuperando los datos subidos para poder contabilizarlos, estos se mostraban como se observa en la tabla 5.20.

Tabla 5.20: Resultados obtenidos en cuarta prueba.

ID	FechaHora
1902044	2023-06-08 20:50:01
1903567	2023-06-08 21:19:58

Tomando los datos mostrados en la tabla 5.20, podemos obtener el número de datos que se perdieron a lo largo de la prueba y así calcular el desempeño que presentó el sistema con las condiciones dadas:

$$\epsilon_1 = \frac{1523}{1800} * 100 = 84.61\,\% \tag{5.20}$$

Algo que se observó en este experimento, es que muchos datos no fueron enviados a la hora en la que realmente se midieron, sino que por momentos la plataforma se quedaba pasmada, los datos medidos en ese momento se iban encolando y se veían reflejados de golpe una vez que el sistema reaccionara de nuevo. Haciendo que varios datos aparecieran como que fueron subidos a una misma hora, cuando en realidad no había sido así.

Finalmente para este experimento, en la figura 5.9 puede observarse la caída del desempeño del sistema a la vez que va aumentando la distancia entre el AP y el circuito.

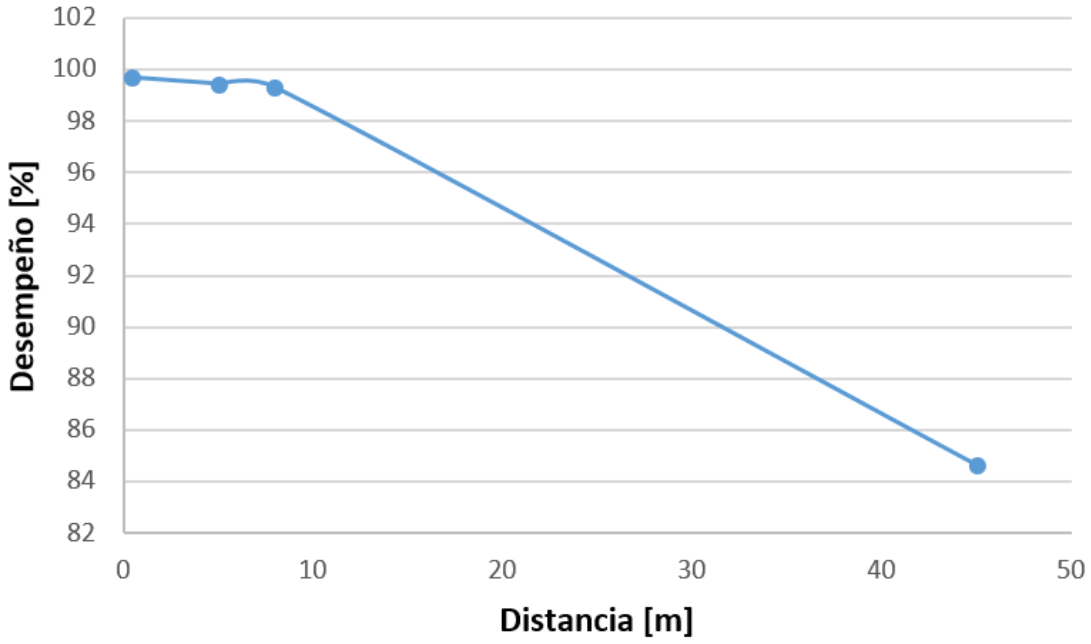

Figura 5.9: Desempeño del sistema a distintas distancias entre AP y circuito.

5.7. Exp 7: Ejecución continua del sistema.

Tomando en cuenta que las pruebas para los experimentos anteriores duraron máximo 24 horas, también se creyó conveniente dejar trabajando nuestro sistema por un mayor lapso, de manera que se le diera un uso más realista y así poder analizar el desempeño de este.

Configurando una tasa de 1 [medición/seg] y con una duración del experimento de 48 horas, se utilizó la plataforma para poder contabilizar los datos subidos con éxito, observándose como se muestra en la tabla 5.21.

Tabla 5.21: Resultados obtenidos en prueba.

ID	FechaHora
1903947	2023-06-08 21:37:00
2076240	2023-06-10 21:36:59

Tomando los datos mostrados en la tabla 5.21, podemos obtener el número de datos que se perdieron a lo largo de la prueba y así calcular el desempeño que presentó el sistema con las condiciones dadas:

$$\epsilon_1 = \frac{172293}{172800} * 100 = 99.7\,\% \tag{5.21}$$

5.8. Exp 8: Variación del ciclo de trabajo del sistema.

Hasta este momento, todos los experimentos fueron realizados conectando el circuito directamente a la corriente eléctrica. Sin embargo, una característica importante de nuestro sistema es la portabilidad, esto para que se tenga la libertad de colocar el sensor en un lugar óptimo para una correcta medición y monitoreo.

Para este octavo experimento, se agregó al circuito una batería recargable y una celda solar. Esto con el propósito de realizar una comparación del desempeño del sistema conectado a la corriente y utilizando la batería (3.7 [V], 8 [A]) y la celda, realizando esto a diferentes valores de ciclo de trabajo. De igual forma, cabe mencionar que este experimento fue realizado en la semana del 23 de julio de 2023, en la cual el clima siempre fue nublado/lluvioso y con temperatura promedio de 20°C.

Para codificar el script donde podamos modificar el ciclo de trabajo al que funciona nuestro sistema, es importante recalcar que se consideró que el sistema tarda aproximadamente 6 segundos en realizar una medición, por lo que para saber por cuánto tiempo debe dormirse el sistema dependiendo del valor de DC, se utilizó la siguiente ecuación:

$$T_{dormido} = \frac{(1 - DC) * 6}{DC} \tag{5.22}$$

5.8.1. Primera prueba: DC de 90 %

Sistema conectado a corriente eléctrica

Al recuperar los datos obtenidos en nuestra plataforma, se mostraba tal y como se observa en la tabla 5.22.

Tabla 5.22: Resultados obtenidos en prueba 1.1.

ID	FechaHora
2152820	2023-07-16 20:00:03
2161428	2023-07-17 07:59:56

Registrándose 8608 en 12 horas, resultando en la siguiente tasa:

$$T_5 = 0.199[Mediciones/seg] \tag{5.23}$$

Sistema conectado a batería y celda solar

Al recuperar los datos registrados en la plataforma, se observaban tal y como se muestran en la tabla 5.23.

Bajo estas condiciones, el sistema trabajó durante 1 hora y 38 minutos, realizando 859 mediciones y resultando en la siguiente tasa:

Tabla 5.23: Resultados obtenidos en prueba 1.2.

ID	FechaHora
2179526	2023-07-31 10:00:00
2180385	2023-07-31 11:38:18

$$T_6 = 0.142857[Mediciones/seg] \tag{5.24}$$

5.8.2. Segunda prueba: DC de 50 %

Sistema conectado a corriente eléctrica

Al consultar nuestra plataforma para recuperar los datos correspondientes a la prueba, se desplegó lo que se muestra en la tabla 5.24.

Tabla 5.24: Resultados obtenidos en prueba 2.1.

ID	FechaHora
2141611	2023-07-11 00:20:02
2145616	2023-07-11 11:59:14

Registrándose 4009 mediciones a lo largo de 12 horas, resultando en la siguiente tasa:

$$T_3 = 0.0928[Mediciones/seg] \tag{5.25}$$

Sistema conectado a batería y celda solar

Tomando el primer y último valores registrados por la plataforma, se recuperó la información mostrada en la tabla 5.25.

Tabla 5.25: Resultados obtenidos en prueba 2.2.

ID	FechaHora
2174264	2023-07-29 10:45:24
2175137	2023-07-29 13:18:56

Bajo estas condiciones, el circuito funcionó durante 2 horas y 34 minutos, realizando un total de 873 mediciones y resultando en la siguiente tasa:

$$T_4 = 0.0909[Mediciones/seg] \tag{5.26}$$

5.8.3. Tercera prueba: DC de 10 %

Sistema conectado a corriente eléctrica

Así como ya se realizó en los experimentos anteriores, para cuantificar las mediciones registradas por el sistema, se utilizó la funcionalidad de nuestra plataforma de mostrar datos antiguos. Recuperando de esto la información de la tabla 5.26 que corresponde a los datos de inicio y fin de la prueba.

Tabla 5.26: Resultados obtenidos en prueba 3.1.

ID	FechaHora
2148511	2023-07-11 22:30:00
2149251	2023-07-12 10:29:05

Registrándose 740 mediciones a lo largo de 12 horas, resultando en la siguiente tasa:

$$T_1 = 0.0171[Mediciones/seg] \tag{5.27}$$

Sistema conectado a batería y celda solar

Tomando el primer y último valores registrados por la plataforma, se recuperó la información mostrada en la tabla 5.27.

Tabla 5.27: Resultados obtenidos en prueba 3.2.

ID	FechaHora
2162744	2023-07-21 21:40:25
2162967	2023-07-22 01:21:07

Con estas condiciones, el circuito solo duró 3 horas y 41 minutos funcionando. Realizando un total de 233 mediciones y resultando con la siguiente tasa:

$$T_2 = 0.01694[Mediciones/seg] \tag{5.28}$$

5.8.4. Cuarta prueba: DC de 1 %

Sistema conectado a corriente eléctrica

Así como ya se realizó en los experimentos anteriores, para cuantificar las mediciones registradas por el sistema, se utilizó la funcionalidad de nuestra plataforma de mostrar datos antiguos. Recuperando de esto la información de la tabla 5.28 que corresponde a los datos de inicio y fin de la prueba.

Tabla 5.28: Resultados obtenidos en prueba 4.1.

ID	FechaHora
2181475	2023-08-27 18:00:07
2181547	2023-08-28 05:57:07

Registrándose 72 mediciones a lo largo de 12 horas, resultando en la siguiente tasa:

$$T_1 = 0.0016666[Mediciones/seg] \tag{5.29}$$

Sistema conectado a batería y celda solar

Tomando el primer y último valores registrados por la plataforma, se recuperó la información mostrada en la tabla 5.29.

Tabla 5.29: Resultados obtenidos en prueba 4.2.

ID	FechaHora
2181338	2023-08-26 08:00:46
2181383	2023-08-26 15:26:09

Con estas condiciones, el circuito duró 7 horas y 27 minutos funcionando. Realizando un total de 45 mediciones y resultando con la siguiente tasa:

$$T_2 = 0.00168350[Mediciones/seg] \tag{5.30}$$

6 Conclusiones

Se cumplieron en su totalidad los objetivos propuestos para la presente tesis. Se desarrolló un circuito medidor de CO_2 junto con su correspondiente plataforma web de monitoreo en tiempo real. Esta plataforma también permite agregar sensores adicionales, así como recuperar datos antiguos que fueron registrados y poder imprimirlos, copiarlos o descargarlos en formatos PDF, CSV y Excel.

Se puso a prueba el sistema en entornos que podrían presentarse en escenarios de uso real. Se realizaron pruebas en un ambiente muy contaminado, a diferentes tasas de lectura de datos, conectando varios sensores a un mismo AP y que carguen datos tanto a una misma como a diferentes tablas de la base de datos; colocando el AP a diferentes distancias del circuito, y finalmente poniendo a funcionar el sistema a diferentes ciclos de trabajo mientras este es alimentado por una batería y una celda solar.

De los experimentos realizados, se pudieron analizar y hacer notar de manera más tangible y realista todas las capacidades del sistema desarrollado. En cuanto al tiempo de respuesta que tiene el sistema a cambios bruscos en el valor de CO_2, se diseñó y armó un circuito que recolecta la concentración de dicho parámetro en la habitación a medir. Además, el sistema puede trabajar a varias tasas de lectura de datos sin perder desempeño (siendo 2 [Mediciones/s] la tasa con mejor relación velocidad - desempeño). Asimismo, si se utiliza más de un sensor para medir una misma habitación, se obtuvo una muy buena respuesta por parte del sistema, pues este siguió funcionando en todo momento y el desempeño nunca se vio comprometido a pesar de estar trabajando con varios circuitos a la vez (siempre se tuvo un desempeño mayor al 90 % teniendo cuatro sensores con una tasa de 1 [Medición/s]). Tomando en cuenta que el sistema trabaja con WiFi, la señal puede ser afectada por muebles, paredes y demás obstáculos que se encuentren en la habitación; sin embargo, al probar el desempeño a diferentes distancias entre el AP y circuito, siempre se tuvo un desempeño mayor al 99 % con una distancia de separación de hasta 8[m].

De toda la información tanto teórica como empírica que se recaudó a lo largo de la redacción de esta tesis, se puede decir con bastante seguridad que con el sistema desarrollado se prometen grandes capacidades y desempeño para el usuario final, aparte de que al ser todo de código abierto y completamente gratuito, se tiene una gama interminable de posibilidades para experimentar y/o modificar el código fuente y circuito para adaptarlo a las necesidades de cada usuario y hasta diseñar e implementar sus propias plataformas con base en la presentada en este trabajo.

Bibliografía

[1] Z. Cucho, R. SÁNCHEZ, and L. Rodríguez, "Microcontroladores," 2007.

[2] C. Millahual, *Descubriendo Arduino*. RedUsers, 2020.

[3] Arduino, "Arduino uno | arduino.cl - compra tu arduino en línea."

[4] J. Guerra, "Esp32 wifi + bluetooth en un solo lugar."

[5] E. Systems, "Esp32 overview | espressif systems."

[6] designthemes, "Instalando el esp32 | tienda y tutoriales arduino."

[7] Waveshare, "Solar power manager, solar power management module, for 6v 24v solar panel."

[8] MQTT, "Mqtt - the standard for iot messaging."

[9] "¿qué es mqtt? definición y detalles."

[10] U. Electronics, "Mq-135 detector de calidad de aire," 2022.

[11] N. MECHATRONICS, "Tutorial sensores de gas mq2, mq3, mq7 y mq135," 2021.

[12] D. ENERGY, "Celdas solares - que es una celda solar? paneles solares | dexen.mx."

[13] revhardware.com and revhardware.com, "Nodemcu: suministro esp8266 con celda solar y batería con energía," 04 2022.

[14] EZOIC, "¿qué es la irradiancia solar? irradiación solar," 02 2019.

[15] "¿qué es la eficiencia energética de los paneles solares?."

[16] O. México, "¿qué es una base de datos?," 2022.

[17] G. B., "¿qué es mysql? explicación detallada para principiantes," 04 2019.

[18] D. A., "Cómo crear un usuario mysql y otorgar privilegios: una guía para principiantes," 07 2020.

[19] phpMyAdmin contributors, "phpmyadmin," 2019.

[20] Tekla, "¿qué es el desarrollo web? [todo lo que necesitas saber]," 03 2022.

[21] M. W. Docs, "Html," 01 2020.

[22] C. Mateu, "Carles mateu software libre desarrollo de u," 03 2004.

[23] MDN, "Css básico - aprende sobre desarrollo web | mdn," 09 2022.

[24] MDN, "Css," 09 2019.

[25] T. P. Group, "Php: ¿qué es php? - manual," 2022.

[26] MDN, "¿qué es javascript? - aprende sobre desarrollo web | mdn," 2022.

[27] S&P, "Efectos del co2 en la contaminación del aire interior y en la salud | s& p," 12 2018.

[28] Petrelis, Virginie, Charlotte, B. Semin, and Francois, "¿por qué la medición de los niveles de co2 puede ayudarnos a luchar contra la covid-19 ?," 01 2022.

[29] J. Saini, M. Dutta, and G. Marques, "Indoor air quality monitoring systems based on internet of things: A systematic review," *International Journal of Environmental Research and Public Health*, vol. 17, p. 4942, 07 2020.

[30] G. Marques, C. R. Ferreira, and R. Pitarma, "Indoor air quality assessment using a co2 monitoring system based on internet of things," *Journal of Medical Systems*, vol. 43, 02 2019.

[31] P. K. Wan, L. Huang, Z. Lai, X. Liu, M. Nowostawski, H. Holtskog, and Y. Liu, "Automated infection risks assessments (aira) for decision-making using a blockchain-based alert system: A case study in a representative building," *Environmental Research*, vol. 216, p. 114663, 01 2023.

[32] F. X. Ming, R. A. A. Habeeb, F. H. B. Md Nasaruddin, and A. B. Gani, "Real-time carbon dioxide monitoring based on iot & cloud technologies," *Proceedings of the 2019 8th International Conference on Software and Computer Applications*, 02 2019.

[33] S. Esfahani, P. Rollins, J. P. Specht, M. Cole, and J. W. Gardner, "Smart city battery operated iot based indoor air quality monitoring system," 12 2020.

[34] M. Benammar, A. Abdaoui, S. Ahmad, F. Touati, and A. Kadri, "A modular iot platform for real-time indoor air quality monitoring," *Sensors*, vol. 18, p. 581, 02 2018.

[35] A. A. Hapsari, A. I. Hajamydeen, D. J. Vresdian, M. Manfaluthy, L. Prameswono, and E. Yusuf, "Real time indoor air quality monitoring system based on iot using mqtt and wireless sensor network," 12 2019.

[36] G. Chiesa, S. Cesari, M. Garcia, M. Issa, and S. Li, "Multisensor iot platform for optimising iaq levels in buildings through a smart ventilation system," *Sustainability*, vol. 11, p. 5777, 10 2019.

[37] E. Alexandrova and A. Ahmadinia, "Real-time intelligent air quality evaluation on a resource-constrained embedded platform," 11 2018.

[38] A. Bushnag, "Air quality and climate control arduino monitoring system using fuzzy logic for indoor environments," 11 2020.

[39] L. Zhao, W. Wu, and S. Li, "Design and implementation of an iot-based indoor air quality detector with multiple communication interfaces," *IEEE Internet of Things Journal*, vol. 6, pp. 9621–9632, 12 2019.

[40] G. Marques and R. Pitarma, "Monitoring health factors in indoor living environments using internet of things," *Advances in Intelligent Systems and Computing*, vol. 570, pp. 785–794, 03 2017.

[41] S. V. Girish, R. Prakash, and A. Balaji Ganesh, "Real-time remote monitoring of indoor air quality using internet of things (iot) and gsm connectivity," *Advances in Intelligent Systems and Computing*, vol. 394, pp. 527–533, 02 2016.

[42] M. Taştan and H. Gökozan, "Real-time monitoring of indoor air quality with internet of things-based e-nose," *Applied Sciences*, vol. 9, p. 3435, 08 2019.

[43] F. Pradityo and N. Surantha, "Indoor air quality monitoring and controlling system based on iot and fuzzy logic," *2019 7th International Conference on Information and Communication Technology (ICoICT)*, 09 2019.

[44] M. Muladi, S. Sendari, and T. Widiyaningtyas, "Real time indoor air quality monitoring using internet of things at university," *2018 2nd Borneo International Conference on Applied Mathematics and Engineering (BICAME)*, 12 2018.

[45] "The mg-811 for metering & recording co2 with the drdaq - pico technology," 12 2016.

[46] A. C. Peixoto and A. F. Silva, "Optical sensor - an overview | sciencedirect topics," 2017.

[47] "Operating principle ndir type gas sensor."

[48] "Dht11 sensor definition, working and applications," 08 2019.

[49] "Operating principle mos type gas sensor."