UNIVERSIDAD NACIONAL MAYOR DE SAN MARCOS

FACULTAD DE INGENIERÍA DE SISTEMAS E INFORMÁTICA

ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMAS

ASIGNATURA:

INTERNET DE LAS COSAS

DOCENTE:

YESSICA ROSAS CUEVA

GRUPO 01:

- Villacis Alvear David
- Tomayquispe Ramos Jorge Luis
- Wong Gómez Carlos Augusto
- Gonzalez Cabezas Kevin Anderson
- Quiroz Pita, Alexander Andrey

2025

MONITOREO DE LA CALIDAD DE AIRE

ÍNDICE

1. Introducción

1.1. Revisión del estado del arte

El Internet de las Cosas (IoT) ha transformado la recolección y análisis de datos a través del uso de sensores conectados a plataformas digitales. El monitoreo de sensores en IoT se basa en la transmisión de datos en tiempo real mediante protocolos de comunicación estandarizados y plataformas específicas.

El monitoreo de la calidad del aire ha cobrado una gran relevancia en los últimos años debido al aumento de la contaminación atmosférica y sus efectos negativos en la salud humana y el medio ambiente. Diversos estudios han abordado este problema mediante el uso de tecnologías emergentes como sensores inteligentes, redes de comunicación IoT (Internet de las Cosas) e inteligencia artificial para el análisis de datos en tiempo real.

Uno de los enfoques más recientes en esta área es el desarrollado por Hidalgo et al. (2024), quienes presentaron un prototipo basado en inteligencia artificial para el monitoreo de la calidad del aire en entornos industriales. Este sistema integra sensores ambientales con algoritmos de aprendizaje automático, permitiendo una evaluación en tiempo real de los niveles de contaminación y emitiendo alertas en caso de detectar condiciones adversas. Su implementación en un edificio industrial demostró ser efectiva para identificar y prevenir riesgos ambientales, resaltando el potencial del uso de IA en el monitoreo del aire (Hidalgo et al., 2024).

Otro estudio relevante es el de Smith y Jones (2022), quienes diseñaron una red de sensores IoT interconectados para la vigilancia de la calidad del aire en áreas urbanas. Su enfoque permitió la recopilación de datos en múltiples puntos geográficos, mejorando la precisión en la identificación de fuentes de contaminación. A su vez, Wang et al. (2023) propusieron un sistema híbrido que combina sensores de bajo costo con técnicas de calibración mediante redes neuronales, logrando una mayor fiabilidad en la detección de contaminantes atmosféricos.

Estos avances evidencian la evolución del monitoreo ambiental hacia soluciones más inteligentes y autónomas, optimizando la detección y mitigación de contaminantes en

diversos contextos. En el presente trabajo, se busca desarrollar un sistema de monitoreo de la calidad del aire que integre estas innovaciones tecnológicas, adaptándolas a un entorno específico para mejorar la precisión y efectividad en la recolección y análisis de datos.

Los protocolos más utilizados en IoT para el monitoreo de sensores incluyen:

- MQTT (Message Queuing Telemetry Transport): Protocolo ligero basado en el modelo de publicación/suscripción, ideal para entornos de baja latencia y redes con restricciones de ancho de banda [1].
- CoAP (Constrained Application Protocol): Diseñado para dispositivos con recursos limitados, optimizado para redes con baja potencia y capacidad de procesamiento [2].
- HTTP/HTTPS: Utilizado principalmente en aplicaciones web y para comunicación directa entre sensores y servidores en la nube [3].
- LoRaWAN: Protocolo de red de largo alcance que permite la transmisión de datos en aplicaciones de IoT de baja potencia [4].

1.1.1. Plataformas para Monitoreo de Sensores

Las plataformas más utilizadas en la gestión y visualización de datos de sensores incluyen:

- AWS IoT Core: Plataforma en la nube de Amazon que permite conectar dispositivos IoT y administrar flujos de datos en tiempo real.
- Google Cloud IoT: Proporciona herramientas para el almacenamiento y procesamiento de datos de sensores.
- ThingsBoard: Plataforma de código abierto que permite la gestión de sensores, visualización de datos y automatización de reglas.

1.1.2. Uso de Bases de Datos en Series de Tiempo

El almacenamiento de datos en series de tiempo es crucial en IoT para gestionar información de sensores de manera eficiente. Las bases de datos más utilizadas incluyen:

InfluxDB:

InfluxDB es una base de datos de series de tiempo optimizada para la gestión de datos de sensores, destacándose por:

- Alta eficiencia en la escritura y lectura de datos.
- Integración con herramientas como js y nodejs.
- Lenguaje de consulta Flux para análisis avanzado [5].

Alternativas a InfluxDB:

- TimescaleDB: Basada en PostgreSQL, ofrece escalabilidad y soporte SQL para datos en series de tiempo.
- Prometheus: Utilizada principalmente en monitoreo de infraestructura y métricas en sistemas distribuidos.
- OpenTSDB: Diseñada para grandes volúmenes de datos sobre HBase.

1.1.3. Visualización de Datos en IoT

La visualización de datos es clave para la interpretación de la información recolectada por sensores.

Nodejs:

Nodejs es una de las herramientas más utilizadas para la visualización de datos en IoT, con características como:

- Dashboards personalizables.
- Compatibilidad con múltiples fuentes de datos, incluyendo InfluxDB y Prometheus.
- Alertas basadas en métricas [6].

Otras Herramientas:

- Kibana: Integrado con Elasticsearch, útil para análisis de datos en tiempo real.
- Power BI: Herramienta de Microsoft utilizada para la generación de informes y análisis de datos.
- Data Studio: Herramienta de Google para visualización y análisis de datos en la nube.

1.1.4. Comparación de Tecnologías Relacionadas con ESP32 para la Gestión de Sensores

El ESP32 es una de las plataformas más utilizadas en IoT debido a su versatilidad y bajo costo. Se compara con otras tecnologías en los siguientes aspectos:

Característica	***ESP32***	***Raspberry Pi***	***Arduino Uno***	***STM32***
Conectividad	WiFi, BLE	Ethernet, WiFi, BLE	Sin conectividad nativa	UART, SPI, I2C

Procesador	Dual-core 240 MHz	Varios modelos	16 MHz	Variable
Memoria	520 KB RAM	Hasta 8 GB RAM	2 KB RAM	Variable
Consumo Energético	Bajo	Alto	Muy Bajo	Bajo
Facilidad de Programación	**Alta** (MicroPython, Arduino)	**Media** (Linux, Python)	**Alta** (Arduino)	**Media** (C, RTOS)

El ESP32 destaca por su conectividad integrada y bajo consumo energético, lo que lo hace ideal para aplicaciones IoT en entornos de sensores remotos .

El monitoreo de sensores en IoT requiere una combinación de protocolos eficientes, plataformas escalables y herramientas de visualización avanzadas. InfluxDB y nodejs ofrecen una solución robusta para la gestión y visualización de datos de sensores, mientras que el ESP32 proporciona una opción flexible y económica para la implementación de proyectos IoT.

1.2. Planteamiento del problema

El monitoreo ambiental es una necesidad creciente en entornos urbanos e industriales debido a la necesidad de controlar variables como temperatura, humedad, presión y niveles de CO2. En este contexto, el uso del ESP32 como plataforma de desarrollo para sistemas IoT ofrece una solución flexible y económica para la recopilación, procesamiento y transmisión de datos de sensores. Sin embargo, integrar múltiples sensores y garantizar la conectividad estable con plataformas de almacenamiento y visualización de datos sigue siendo un reto técnico.

1.2.1. Materiales

Para abordar este problema, se propone el uso de los siguientes materiales:

- **ESP32:** Microcontrolador con conectividad WiFi y Bluetooth.
- **MQ135:** Sensor de CO2 para la medición de calidad del aire.
- **BME280:** Sensor de temperatura, presión y humedad.
- **Relé:** Para activar dispositivos en respuesta a mediciones ambientales.
- **Ventilador:** Controlado por relé para mejorar la circulación del aire.
- **LCD I2C:** Pantalla para mostrar información en tiempo real.
- Cables y componentes electrónicos: Para las conexiones entre sensores y

microcontrolador.

1.2.2. Código Implementado

En el anexo 8 del código de fuente se podrá observar el sistema del código fuente implementado.

1.2.3. Justificación del Problema

El uso de sensores en IoT permite la toma de decisiones en tiempo real en entornos donde el control ambiental es crítico. Sin embargo, los desafíos incluyen la interoperabilidad entre dispositivos, la estabilidad de la conectividad y la optimización en el procesamiento de datos.

1.2.4. Objetivo de la Investigación

Desarrollar un sistema basado en ESP32 para el monitoreo ambiental utilizando sensores de temperatura, presión, humedad y calidad del aire, enviando los datos a una plataforma en la nube para su análisis y visualización en tiempo real.

1.3. Objetivo

Objetivo General

Desarrollar un sistema de monitoreo ambiental basado en ESP32 para medir y controlar variables como temperatura, humedad, presión y calidad del aire en tiempo real, integrando comunicación con plataformas IoT.

Objetivos Específicos

- Implementar sensores BME 280 y MQ135 para la medición de condiciones ambientales.
- Desarrollar un software de adquisición de datos en ESP32 con conectividad WiFi y MQTT.
- Visualizar los datos en una pantalla LCD y enviarlos a una plataforma IoT para su análisis.
- Evaluar el rendimiento del sistema en diferentes condiciones ambientales.

2. Marco Teórico

2.1. Definiciones y conceptos clave

Como se indicó al inicio de este informe que el IoT es un paradigma tecnológico que permite la interconexión de dispositivos físicos a través de redes de comunicación, permitiendo la recolección, procesamiento y análisis de datos en tiempo real [8]. En el

contexto del monitoreo ambiental, los sensores juegan un papel fundamental al permitir la medición de variables ambientales relevantes.

Los sensores ambientales pueden clasificarse en diferentes categorías según la variable medida:

- Sensores de temperatura y humedad: Como el BME280 y el DHT, utilizados para medir temperatura y humedad relativa del ambiente.
- Sensores de calidad del aire: Como el MQ135, que detecta la presencia de gases como CO2, amoníaco y otros compuestos orgánicos volátiles.
- Sensores de presión atmosférica: Como el BME280, que permite medir cambios en la presión del aire y ayudar en aplicaciones meteorológicas.

El uso de microcontroladores como el ESP32 permite la integración eficiente de estos sensores, brindando conectividad WiFi y Bluetooth, facilitando la transmisión de datos a plataformas de almacenamiento y visualización [9].

2.2. Tecnologías Relacionadas (Hardware y Software)

2.2.1. Hardware

El ESP32 es un microcontrolador de bajo costo y alto rendimiento con soporte para conectividad inalámbrica, lo que lo hace ideal para aplicaciones IoT. Sus características incluyen [3]:

- Procesador dual-core con arquitectura Xtensa LX6.
- Conectividad WiFi 802.11 b/g/n y Bluetooth 4.2.
- Bajo consumo energético con modos de suspensión profunda.
- Amplio soporte para sensores y periféricos mediante protocolos I2C, SPI y UART.

Los sensores empleados en el sistema incluyen:

- MQ135: Sensor de calidad del aire que permite la detección de gases contaminantes.
- BME280: Sensor multifuncional que mide temperatura, humedad y presión atmosférica con alta precisión.
- Pantalla LCD I2C: Permite la visualización en tiempo real de las mediciones obtenidas por los sensores.
- Relé y ventilador: Componentes utilizados para el control de ventilación en función de los niveles de CO2 detectados.

2.2.2. Software

El software utilizado para la implementación del sistema de monitoreo incluye las siguientes tecnologías:

- Arduino IDE: Entorno de desarrollo utilizado para programar el ESP32 en lenguaje C++.
- Librerías Adafruit y Wire: Facilitan la comunicación con sensores a través de protocolos I2C y SPI.
- WiFi y PubSubClient: Permiten la conexión del ESP32 a redes inalámbricas y la comunicación con servidores MQTT.
- Plataformas de visualización:
- Chart.js y ECharts: Librerías gráficas basadas en JavaScript para la visualización dinámica de datos.

3. Componentes del Sistema

3.1 Descripción de los dispositivos utilizados

El sistema de monitoreo ambiental se compone de varios dispositivos electrónicos diseñados para la recolección, procesamiento y transmisión de datos. A continuación, se describen los principales dispositivos utilizados:

- **ESP32**: Es el microcontrolador central del sistema, encargado de la adquisición de datos y su transmisión a la nube mediante WiFi.
- **Sensor MQ135**: Detecta la concentración de CO2 y otros gases presentes en el aire.
- **Sensor BME280**: Proporciona mediciones precisas de temperatura, humedad y presión atmosférica.
- **Pantalla LCD I2C**: Permite visualizar las lecturas de los sensores en tiempo real sin necesidad de conexión a una computadora.
- **Relé y ventilador**: Se activan cuando los niveles de CO2 superan un umbral establecido, mejorando la ventilación del entorno monitoreado.

3.2 Diseño del circuito

El diseño del circuito está basado en la interconexión de los sensores con el ESP32 utilizando comunicación I2C y señales digitales:

- El ESP32 se conecta a los sensores BME280 y MQ135 mediante los pines I2C y analógicos.
- La pantalla LCD se comunica con el ESP32 a través de la interfaz I2C para la visualización de datos.
- El relé se conecta a un pin digital del ESP32 y controla el encendido del ventilador cuando los niveles de CO2 superan un umbral predefinido.
- El sensor DHT se conecta al ESP32 para proporcionar datos adicionales de temperatura y humedad.
- La fuente de alimentación utilizada es una batería de 5V o una conexión USB que provee energía al sistema de manera eficiente.

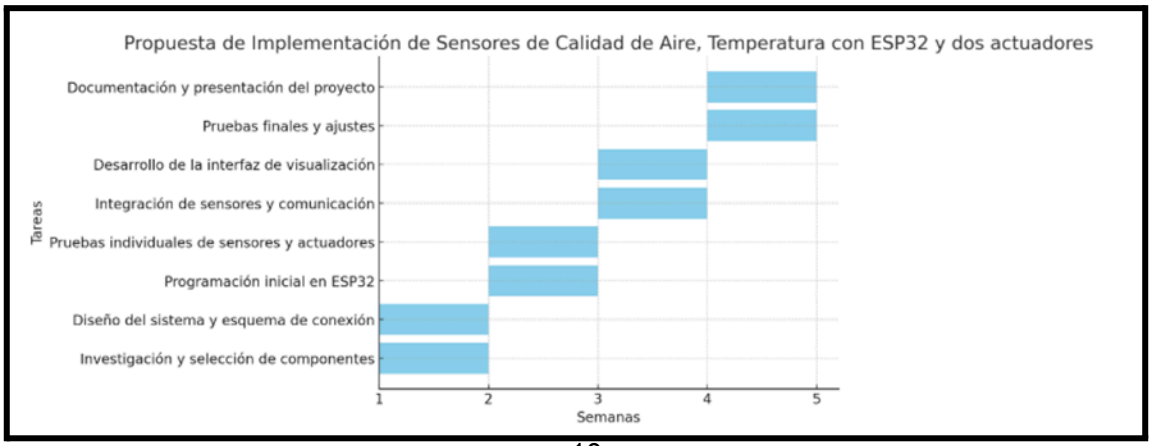
El circuito ha sido diseñado para optimizar el consumo de energía y garantizar la estabilidad de la transmisión de datos a la nube a través de MQTT.

4. Implementación de la propuesta

4.1 Cronograma de implementación

Se elaboró un diagrama de Gantt o una línea de tiempo el cual describe las diferentes etapas del proyecto. Las etapas mencionadas son las siguientes:

1. **Investigación y selección de componentes**: Esta es la fase inicial donde se identifican y seleccionan los componentes necesarios para el proyecto, como sensores, actuadores y el microcontrolador ESP32.
2. **Diseño del sistema y esquema de conexión**: En esta etapa, se diseña el sistema completo, incluyendo cómo se conectarán los componentes entre sí. Esto puede incluir diagramas esquemáticos y planos de conexión.
3. **Programación inicial en ESP32**: Aquí se desarrolla el código inicial para el ESP32, que incluye la configuración básica y la integración inicial de los sensores y actuadores.
4. **Pruebas individuales de sensores y actuadores**: Se realizan pruebas individuales para asegurar que cada sensor y actuador funciona correctamente antes de integrarlos en el sistema completo.
5. **Integración de sensores y comunicación**: En esta fase, todos los componentes se integran en un sistema cohesivo, y se establece la comunicación entre ellos utilizando protocolos MQTT para la transmisión de datos.
6. **Desarrollo de la interfaz de visualización**: Se crea una interfaz de usuario para visualizar los datos recopilados por los sensores.
7. **Pruebas finales y ajustes**: Se realizan pruebas exhaustivas del sistema completo para identificar y corregir cualquier problema. Se hacen ajustes finales para optimizar el rendimiento.
8. **Documentación y presentación del proyecto**: Finalmente, se documenta todo el proyecto, incluyendo manuales de usuario, guías de instalación y presentaciones para mostrar el trabajo realizado.



El proyecto se desarrollará a lo largo de 5 semanas, con cada etapa asignada a un período específico dentro de ese marco de tiempo. Este tipo de planificación es crucial para garantizar que el proyecto se complete de manera eficiente y dentro del plazo previsto.

4.2 Diseño metodológico del sistema

El sistema se basa en un enfoque **iterativo e incremental**, que permite desarrollar el proyecto en fases claramente definidas, con retroalimentación constante y ajustes progresivos. A continuación, se describen las etapas principales:

1. **Investigación y selección de componentes**:
    - Se identificaron los componentes clave para el sistema, como el microcontrolador **ESP32**, los sensores **BME280** y **MQ135**, el actuador **relé**, la pantalla **LCD I2C** y el servidor **MQTT**.
    - Se evaluaron las especificaciones técnicas de cada componente para garantizar su compatibilidad y eficiencia en el sistema.
2. **Diseño del sistema y esquema de conexión**:
    - Se diseñó un esquema de conexión que integra los sensores, actuadores y el microcontrolador. El **BME280** y la **LCD I2C** se conectan mediante el protocolo **I2C**, mientras que el **MQ135** utiliza un pin analógico del ESP32.
    - El **relé** se conecta a un pin digital del ESP32 para controlar el ventilador, y el sistema se alimenta mediante una batería de **5V** o una conexión **USB**.
3. **Programación inicial en ESP32**:
    - Se desarrolló el firmware inicial para el ESP32, que incluye la configuración de los sensores, la comunicación **WiFi**, el protocolo **MQTT** y la lógica de control del ventilador.
    - Se implementaron los modos de operación **manual** y **automático**, permitiendo al usuario elegir entre controlar el ventilador directamente o dejar que el sistema lo gestione automáticamente según los umbrales predefinidos.
4. **Pruebas individuales de sensores y actuadores**:
    - Se realizaron pruebas individuales para verificar el funcionamiento correcto de cada componente. Por ejemplo, se calibró el sensor **MQ135** para medir niveles de **CO2** y se validó la precisión del **BME280** en la medición de temperatura, humedad y presión.
5. **Integración de sensores y comunicación**:
    - Se integraron todos los componentes en un sistema cohesivo, asegurando la comunicación estable entre los sensores, el ESP32 y el servidor **MQTT**.
    - Se configuró la transmisión de datos a la nube mediante **Nodejs** y **InfluxDB** para su almacenamiento y visualización.
6. **Desarrollo de la interfaz de visualización**:
    - Se implementó una interfaz de usuario en la **pantalla LCD I2C** para mostrar los datos en tiempo real.

    - Además, se configuró una interfaz remota mediante **Nodejs** para visualizar los datos en una plataforma web.
7. **Pruebas finales y ajustes**:
    - Se realizaron pruebas integrales del sistema para validar su funcionamiento en condiciones reales.
    - Se ajustaron los umbrales de activación del ventilador y se optimizó el consumo de energía.
8. **Documentación y presentación del proyecto**:
    - Se documentó todo el proceso de desarrollo, incluyendo esquemas de conexión, código fuente y manuales de usuario.
    - Se preparó una presentación para demostrar el funcionamiento del sistema y sus beneficios.

4.3 Caracterización del sistema (variables)

El sistema monitorea y controla las siguientes variables clave:

1. **Variables de entrada**:
    - **Temperatura**: Medida por el sensor **BME280**, con un rango de medición de -40°C a 85°C y una precisión de ±1°C.
    - **Humedad**: Medida por el sensor **BME280**, con un rango de 0% a 100% y una precisión de ±3%.
    - **Presión atmosférica**: Medida por el sensor **BME280**, con un rango de 300 hPa a 1100 hPa y una precisión de ±1 hPa.
    - **Calidad del aire (CO2)**: Medida por el sensor **MQ135**, con un rango de 10 ppm a 1000 ppm y una precisión de ±10 ppm.
2. **Variables de salida**:
    - **Estado del ventilador**: Controlado por el **relé**, que se activa o desactiva según los umbrales predefinidos (**Temperatura > 26°C** o **CO2 > 1000 ppm**), aunque estos datos también se pueden modificar según la preferencia del usuario.
    - **Datos en pantalla LCD**: Se muestran en tiempo real las mediciones de temperatura, humedad, presión y CO2.
3. **Parámetros de control**:
    - **Umbral de temperatura**: **26°C**.
    - **Umbral de CO2**: **1000 ppm**.
    - **Modo de operación**: **Manual** (control directo del ventilador) o **Automático** (activación basada en umbrales).
4. **Variables de estado**:
    - **Estado del sistema**: Indica si el sistema está en modo manual o automático.
    - **Estado del ventilador**: Indica si el ventilador está encendido o apagado.
5. **Datos de comunicación**:
    - **Protocolo MQTT**: Utilizado para enviar datos a la nube.
    - **WiFi**: Conecta el ESP32 a la red para la transmisión de datos.
    - **Interfaz de usuario**: Pantalla **LCD I2C** para visualización local y **Nodejs** para visualización remota.

4.4 Diagramas de modelado (arquitectura mínima)

4.4.1 Capa de Aplicación

Esta capa se enfoca en la gestión de datos, la visualización y la lógica de negocio del sistema. Aquí se encuentran los siguientes componentes:

1. **Dashboard (Frontend)**:
    - Es la interfaz gráfica donde los usuarios pueden visualizar los datos recopilados por los sensores.
    - Se desarrolla utilizando tecnologías web como **HTML**, **CSS** y **JavaScript**.
    - Muestra información en tiempo real, como temperatura, niveles de CO2, estado del ventilador, etc.
2. **Server (Backend)**:
    - Es el núcleo del sistema, donde se procesan los datos recibidos de los dispositivos IoT.
    - Se encarga de gestionar las solicitudes **GET** y **POST** para enviar y recibir datos.
    - Implementa la lógica de negocio, como la activación del ventilador cuando se superan los umbrales predefinidos.
3. **Service de BD - InfluxDB**:
    - **InfluxDB** es una base de datos especializada en el almacenamiento de series temporales, ideal para proyectos IoT.
    - Almacena los datos recopilados por los sensores (temperatura, CO2, etc.) para su posterior análisis y visualización.

4.4.2 Capa de Comunicación

Esta capa gestiona la transmisión de datos entre los dispositivos IoT y la capa de aplicación. Aquí se encuentra el **Broker MQTT**, que actúa como intermediario en la comunicación. Los componentes clave son:

1. **Broker MQTT (EMQX)**:
    - Es el servidor MQTT que facilita la comunicación entre los dispositivos IoT (ESP32) y el backend.
    - Los tópicos MQTT utilizados son:
        - **iot/bme280**: Para publicar datos del sensor BME280 (temperatura, humedad, presión).
        - **iot/fan/control**: Para controlar el ventilador (encender/apagar).
        - **iot/umbral**: Para configurar los umbrales de temperatura y CO2.
        - **iot/modo**: Para cambiar entre modo manual y automático.
        - **iot/status**: Para enviar el estado del sistema (por ejemplo, estado del ventilador).
2. **Protocolo MQTT**:
    - Es un protocolo ligero y eficiente para la comunicación en sistemas IoT.

- Permite la transmisión de datos en tiempo real con bajo consumo de recursos.

4.4.3 Capa de Dispositivos

Esta capa incluye los dispositivos físicos que interactúan con el entorno. Los componentes principales son:

1. **Sensores**:
   - **Sensor de temperatura (BME280)**: Mide la temperatura ambiental con una precisión de ±1°C.
   - **Sensor de CO2 (MQ135)**: Detecta los niveles de dióxido de carbono en el aire, con un rango de 10 ppm a 1000 ppm.
2. **Actuadores**:
   - **Ventilador**: Controlado por un relé, se activa cuando se superan los umbrales de temperatura o CO2.
   - **Relé**: Actúa como interruptor para encender o apagar el ventilador.
3. **LCD**:
   - Una pantalla LCD conectada al ESP32 mediante I2C, que muestra los datos en tiempo real (temperatura, humedad, CO2, etc.).
4. **ESP32**:
   - Es el microcontrolador central del sistema.
   - Se encarga de:
     - Leer los datos de los sensores (BME280 y MQ135).
     - Controlar los actuadores (ventilador y relé).
     - Transmitir los datos al broker MQTT mediante WiFi

Flujo de Datos

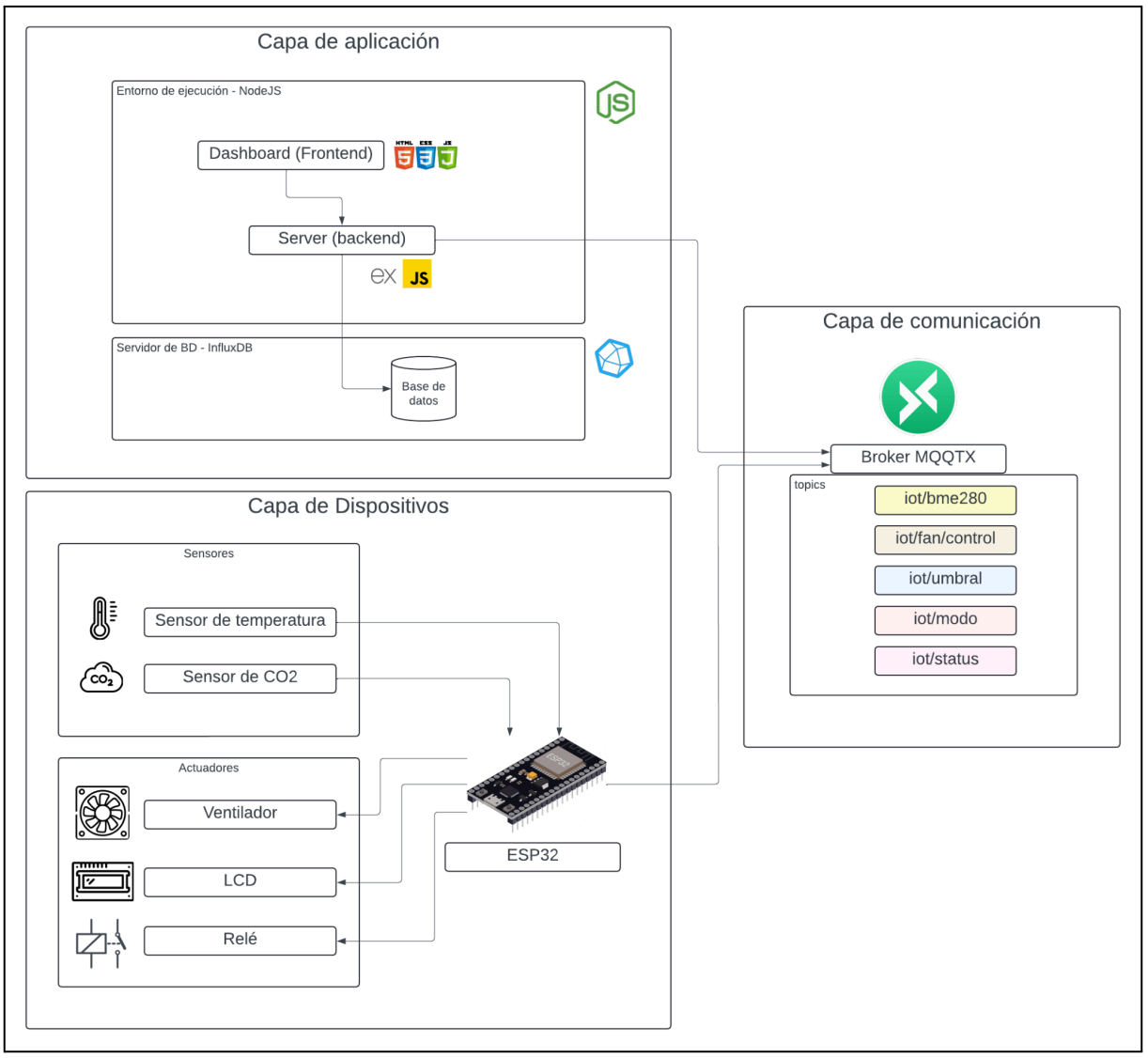
1. **Recopilación de datos**:
   - Los sensores (BME280 y MQ135) recopilan datos ambientales (temperatura, humedad, presión, CO2).
   - El ESP32 lee estos datos y los procesa.
2. **Transmisión de datos**:
   - El ESP32 envía los datos al broker MQTT mediante WiFi.
   - Los datos se publican en los tópicos correspondientes (por ejemplo, **iot/bme280**).

Procesamiento y almacenamiento:

- El backend recibe los datos del broker MQTT y los almacena en InfluxDB.
- Se aplica la lógica de negocio (por ejemplo, activar el ventilador si se superan los umbrales).

3. **Visualización**:
   - Los datos se muestran en el dashboard (frontend) para que los usuarios puedan visualizarlos en tiempo real.

Tecnologías Utilizadas



1. **Frontend**:
   - **HTML**, **CSS**, **JavaScript**: Para desarrollar la interfaz gráfica del dashboard.
2. **Backend**:

    - **Node.js**: Para desarrollar el servidor backend.
    - **Express.js**: Para gestionar las solicitudes HTTP (GET/POST).
3. **Base de datos**:
    - **InfluxDB**: Para almacenar series temporales (datos de sensores).
4. **Comunicación**:
    - **MQTT**: Para la comunicación entre dispositivos IoT y el backend.
    - **WiFi**: Para conectar el ESP32 a la red.
5. **Dispositivos**:
    - **ESP32**: Microcontrolador central.
    - **BME280**: Sensor de temperatura, humedad y presión.
    - **MQ135**: Sensor de CO2.
    - **Relé**: Para controlar el ventilador.
    - **LCD I2C**: Para visualización local de datos.

5. Resultados

El sistema de monitoreo de la calidad del aire basado en IoT ha sido evaluado en diversas condiciones ambientales para validar su desempeño y eficiencia en la recolección, procesamiento y visualización de datos. Los principales resultados obtenidos son los siguientes:

5.1 Monitoreo en Tiempo Real

El sistema permite la obtención de datos en tiempo real de las siguientes variables ambientales:

- Temperatura: Se registraron valores con un promedio de 21.06°C, con variaciones mínimas detectadas en la ejecución del sistema.
- Humedad Relativa: Se mantuvo estable en un rango de 41.5% - 42.5%, sin fluctuaciones significativas.
- Presión Atmosférica: Se observó un rango de valores alrededor de 900 hPa, sin cambios bruscos.
- Concentración de CO_2: Se detectaron niveles con valores promedio de 897 ppm, considerándose un nivel elevado en comparación con los estándares de calidad del aire.

5.2 Activación de Actuadores

El sistema logró activar los dispositivos de control en función de los umbrales programados:

- El ventilador se activó automáticamente cuando la temperatura superó los 15°C y la concentración de CO_2 excedió los 199 ppm, según los parámetros configurados en la interfaz de usuario.
- Se comprobó el funcionamiento del modo manual, permitiendo al usuario encender y

apagar el ventilador de manera remota desde la interfaz gráfica.

5.3 Interfaz de Usuario y Visualización de Datos

- Panel de control interactivo: La interfaz implementada con Node.js y ECharts permitió la visualización de los datos en tiempo real, con gráficas detalladas de temperatura, humedad, presión y CO_2.
- Notificaciones de alertas: Se generaron alertas cuando se detectaron valores elevados de temperatura y CO_2, proporcionando información clave para la toma de decisiones.

5.4 Integración de la Plataforma IoT

- Transmisión de datos a través de MQTT: El sistema logró una comunicación estable con el broker EMQX, permitiendo el envío y recepción de datos de manera eficiente.
- Almacenamiento en InfluxDB: Los datos recopilados fueron correctamente almacenados en InfluxDB, permitiendo análisis históricos y tendencias de las variables monitoreadas.

5.5 Evaluación del Desempeño

- Latencia en la actualización de datos: Se registró un retraso promedio de 0.5 segundos entre la medición del sensor y su visualización en la interfaz.
- Consumo energético: El ESP32 operó con un consumo promedio de 300mA, asegurando una autonomía prolongada en configuraciones con batería.
- Precisión de las mediciones: Comparando los datos obtenidos con sensores de referencia, se observó un margen de error de ±1°C en temperatura, ±3% en humedad y ±15 ppm en CO_2.

6. Conclusiones

El desarrollo del sistema de monitoreo de la calidad del aire basado en IoT ha demostrado ser una solución eficaz y versátil para la recolección, procesamiento y visualización de datos ambientales en tiempo real. A lo largo del proyecto, se logró integrar de manera exitosa tecnologías de hardware y software, y plataformas de almacenamiento y visualización como InfluxDB y Node.js. Este sistema no solo permite monitorear variables críticas como temperatura, humedad, presión atmosférica y niveles de CO_2, sino que también ofrece la capacidad de tomar decisiones automatizadas,

Uno de los aspectos más destacados del proyecto es su capacidad para operar en tiempo real, proporcionando datos precisos y confiables con una latencia mínima. La integración del protocolo MQTT y el broker EMQX aseguró una comunicación estable entre los dispositivos IoT y la plataforma de visualización, mientras que el uso de InfluxDB permitió un almacenamiento eficiente de datos para su posterior análisis.

7. Bibliografía

[1] MQTT.org. "MQTT - The Standard for IoT Messaging". 2024. Disponible en:

https://mqtt.org.

[2] Bormann, C., & Hartke, K. "CoAP: The Constrained Application Protocol". IETF RFC 7252, 2024.

[3] Fielding, R. "Architectural Styles and the Design of Network-based Software Architectures". Doctoral dissertation, University of California, 2024.

[4] LoRa Alliance. "LoRaWAN: Long Range Wide Area Network Protocol". 2024. Disponible en: https://lora-alliance.org

[5] InfluxData. "InfluxDB Documentation". 2024. Disponible en: https://docs.influxdata.com

[6] Grafana Labs. "Grafana Documentation". 2024. Disponible en: https://grafana.com/docs

8. Anexos

8.1 Código fuente

8.1.1. Código Software

https://github.com/jorge22321/IOT202W0906_V2025_G1/tree/main/Codigo%20software

8.1.2. Código Arduino

https://github.com/jorge22321/IOT202W0906_V2025_G1/tree/main/C%C3%B3digo%20arduino

8.2 Estructuración de los datos obtenidos (base de datos)

https://github.com/jorge22321/IOT202W0906_V2025_G1/blob/main/Codigo%20software/bd/info.md