UNIVERSIDAD AUTÓNOMA DEL ESTADO DE HIDALGO

INSTITUTO DE CIENCIAS BÁSICAS E INGENIERÍA

MAESTRÍA EN INTERNET DE LAS COSAS

PROYECTO TERMINAL

PROTOTIPO DE SISTEMA IOT PARA EL MONITOREO EN TIEMPO REAL DE TEMPERATURA Y HUMEDAD EN EL SITE DE LA UAEH

Para obtener el grado de

Maestro en Internet de las Cosas

PRESENTA

Ing. Víctor Hugo Martínez Cano

Director

Dr. Luis Heriberto García Islas

Codirectora

Mtra. Kristell Daniella Franco Sánchez

Comité tutorial

Dr. Anilú Franco Arcega
Mtra. Kristell Daniella Franco Sánchez
Dr. Luis Heriberto García Islas
Dr. Esteban Rueda Soriano
Mtro. Alberto Suárez Navarrete
Mtro. Melecio Sánchez Ruiz

Mineral de la Reforma, Hgo., a 8 de octubre de 2025

UNIVERSIDAD AUTÓNOMA DEL ESTADO DE HIDALGO

INSTITUTO DE CIENCIAS BÁSICAS E INGENIERÍA

MAESTRÍA EN INTERNET DE LAS COSAS

PROYECTO TERMINAL

PROTOTIPO DE SISTEMA IOT PARA EL MONITOREO EN TIEMPO REAL DE TEMPERATURA Y HUMEDAD EN EL SITE DE LA UAEH

Para obtener el grado de

Maestro en Internet de la Cosas

PRESENTA

Ing. Víctor Hugo Martínez Cano

Director

Dr. Luis Heriberto García Islas

Codirectora

Mtra. Kristell Daniella Franco Sánchez

Comité tutorial

Dr. Anilú Franco Arcega
Mtra. Kristell Daniella Franco Sánchez
Dr. Luis Heriberto García Islas
Dr. Esteban Rueda Soriano
Mtro. Alberto Suárez Navarrete

Mineral de la Reforma, Hgo., a 8 de octubre de 2025

Universidad Autónoma del Estado de Hidalgo
Instituto de Ciencias Básicas e Ingeniería
School of Engineering and Basic Sciences
Área Académica de Computación y Electrónica
Department of Electronics and Computer Science

Mineral de la Reforma, Hgo., a 8 de octubre de 2025

No. De Control: ICBI-AACyE/1978/2025
Asunto: Autorización de impresión

MTRA. OJUKY DEL ROCÍO ISLAS MALDONADO
DIRECTORA DE ADMINISTRACIÓN ESCOLAR DE LA UAEH

El Comité Tutorial del **PROYECTO TERMINAL** del programa educativo de posgrado titulado **"PROTOTIPO DE SISTEMA IOT PARA EL MONITOREO EN TIEMPO REAL DE TEMPERATURA Y HUMEDAD EN EL SITE DE LA UAEH"**, realizado por el sustentante **VÍCTOR HUGO MARTÍNEZ CANO** con número de cuenta **488361** perteneciente al programa de **MAESTRÍA EN INTERNET DE LAS COSAS**, una vez que se ha revisado, analizado y evaluado el documento recepcional de acuerdo a lo estipulado en el Artículo 110 del Reglamento de Estudios de Posgrado, tiene a bien extender la presente:

AUTORIZACIÓN DE IMPRESIÓN

Por lo que la sustentante deberá cumplir los requisitos del Reglamento de Estudios de Posgrado y con lo establecido en el proceso de grado vigente.

Atentamente
"Amor, Orden y Progreso"

El Comité Tutorial

Dra. Anilú Franco Arcega
Miembro del comité

Dr. Esteban Rueda Soriano
Miembro del comité

Dr. Luis Heriberto García Islas
Miembro del comité

Mtra. Kristell Daniella Franco Sánchez
Miembro del comité

Mtro. Alberto Suárez Navarrete
Miembro del comité

Mtro. Melecio Sánchez Ruiz
Miembro del comité

C.c.p. Archivo
MSR

ÁREA ACADÉMICA DE COMPUTACIÓN Y ELECTRÓNICA
Ciudad del Conocimiento, Carretera Pachuca-Tulancingo Km. 4.5 Colonia Carboneras, Mineral de la Reforma, Hidalgo, México. C.P. 42184
Teléfono: 52 (771) 71 720 00 Ext. 40052, 40053
aacye_icbi@uaeh.edu.mx, jesus_ordaz@uaeh.edu.mx

AGRADECIMIENTO

En primer lugar, agradezco a Dios por ser mi fuerza incondicional y por darme la resiliencia necesaria para superar los desafíos. A mi familia, quienes con sus esfuerzos y sacrificios me han brindado las oportunidades para alcanzar esta meta, espero poder recompensar su dedicación con alegrías y logros en este largo camino de la vida. A mis compañeros y amigos, quienes me han enseñado el valor de la amistad, el trabajo en equipo y el amor, elementos fundamentales para alcanzar cualquier objetivo.

Agradezco especialmente al Dr. Luis Heriberto García Islas por su apoyo incondicional y guía en mi proceso de formación profesional, así como al Mtro. Iván Horacio Pérez Tavera, al Mtro. Francisco González Castañeda y al Mtro. Jesús Gabriel Banda, quienes, con sus conocimientos y talento humano, contribuyeron a mi crecimiento académico y personal. A la Universidad Autónoma del Estado de Hidalgo, por abrirme las puertas de esta excelente institución que se convirtió en un segundo hogar, y por brindarme las herramientas necesarias para alcanzar este logro.

Finalmente, agradezco a todas las personas e instituciones que, de manera directa o indirecta, contribuyeron a la realización de este proyecto. Este logro es el resultado de un esfuerzo colectivo, y espero que este trabajo sea un aporte significativo para el campo de la ingeniería y la tecnología.

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

RESUMEN

Este proyecto tiene como objetivo implementar un sistema IoT para el monitoreo en tiempo real de la temperatura y humedad en el site de la Universidad Autónoma del Estado de Hidalgo (UAEH). El sistema utiliza un microcontrolador ESP32 y un sensor DHT11 para capturar datos ambientales, los cuales son transmitidos mediante el protocolo MQTT a un broker y almacenados en una base de datos NoSQL MongoDB. La información es visualizada en una plataforma web, permitiendo el acceso remoto y la supervisión continua de las condiciones ambientales del site. Este sistema busca prevenir fallos en los equipos tecnológicos debido a variaciones en la temperatura y humedad, asegurando la continuidad de los servicios académicos y administrativos. Además, se plantea la posibilidad de expandir el sistema con alertas automáticas y control remoto de las condiciones ambientales.

Palabras clave: Sistema IoT, monitoreo en tiempo real, temperatura, humedad, site, ESP32, DHT11, protocolo MQTT, broker, base de datos NoSQL, MongoDB, plataforma web, acceso remoto, supervisión continua, prevención de fallos.

ABSTRACT

This project aims to implement an IoT system for real-time monitoring of temperature and humidity at the site of the Autonomous University of the State of Hidalgo (UAEH). The system uses an ESP32 microcontroller and a DHT11 sensor to capture environmental data, which is transmitted via the MQTT protocol to a broker and stored in a NoSQL MongoDB database. The information is visualized on a web platform, allowing remote access and continuous monitoring of the site's environmental conditions. This system seeks to prevent failures in technological equipment due to variations in temperature and humidity, ensuring the continuity of academic and administrative services. Additionally, the possibility of expanding the system with automatic alerts and remote control of environmental conditions is proposed.

Keywords: IoT system, real-time monitoring, temperature, humidity, site, ESP32, DHT11, MQTT protocol, broker, NoSQL database, MongoDB, web platform, remote access, continuous monitoring, failure prevention.

PRESENTACIÓN

En la era digital, la disponibilidad y continuidad de los servicios tecnológicos son fundamentales para el funcionamiento de cualquier organización. Desde la educación hasta la industria, la infraestructura tecnológica depende de sistemas interconectados que operan en tiempo real. Sin embargo, detrás de cada servicio en línea, cada base de datos y cada plataforma en la nube, existe un componente crítico que muchas veces pasa desapercibido: los centros de datos. Estos espacios albergan servidores y equipos de telecomunicaciones que constituyen el pilar del ecosistema digital.

Con el avance del Internet de las Cosas (IoT), la capacidad de monitorear y controlar entornos remotos ha evolucionado significativamente. Sensores inteligentes y dispositivos interconectados permiten supervisar en tiempo real variables como temperatura y humedad, factores clave para evitar fallos catastróficos en la infraestructura tecnológica. Implementar soluciones basadas en IoT no solo optimiza la gestión de estos entornos, sino que también fortalece la capacidad de respuesta ante situaciones críticas, asegurando la estabilidad operativa de las organizaciones y la continuidad de sus servicios. Los centros de datos o data centers son espacios especializados donde se alojan equipos críticos de telecomunicaciones, como routers, switches, servidores y cableado estructurado. Estos componentes conforman la columna vertebral de las áreas de sistemas y telecomunicaciones en una organización.

Este proyecto se enfoca en el desarrollo de un prototipo de sistema de monitoreo de temperatura y humedad, implementado específicamente para el site de la Universidad Autónoma del Estado de Hidalgo (UAEH). Este sistema monitorea variables ambientales, como la temperatura y la humedad, que son esenciales para garantizar el óptimo funcionamiento de los equipos tecnológicos y prevenir fallos catastróficos en la infraestructura.

Para lograrlo, el sistema utiliza un ESP32, un microcontrolador de bajo consumo con conectividad Wi-Fi, y el sensor DHT11, encargado de medir la temperatura y la humedad. Los datos recopilados se transmiten mediante el protocolo MQTT a un broker, desde donde se almacenan en una base de datos NoSQL MongoDB, facilitando su procesamiento y visualización en una plataforma web. Esta arquitectura permite un monitoreo remoto eficiente, asegurando el acceso a la información desde cualquier ubicación.

El uso de MQTT y MongoDB ofrece una solución eficiente y escalable para gestionar grandes volúmenes de datos en tiempo real. A su vez, la flexibilidad del sistema permite su implementación en sitios remotos, sin necesidad de cableado adicional o modificaciones físicas considerables, lo cual es clave para asegurar una infraestructura flexible y adaptable.

La implementación de este prototipo es de suma importancia para la UAEH, ya que los sistemas de tecnología de la información en su site juegan un papel crucial en la continuidad de los servicios académicos y administrativos. Un monitoreo ambiental ineficaz puede derivar en sobrecalentamiento de los equipos, daños irreparables y la interrupción de los servicios, afectando tanto a estudiantes como al personal administrativo.

Este proyecto también abre la puerta a futuras extensiones del sistema, como la integración de alertas automáticas y la posibilidad de controlar de manera remota las condiciones ambientales del site, con el fin de asegurar una gestión proactiva de la infraestructura tecnológica.

Dado que la UAEH es una de las principales instituciones educativas en la región, el desarrollo de este tipo de tecnologías no solo optimiza el uso de sus recursos, sino que también fortalece la capacidad de respuesta ante situaciones críticas, minimizando los riesgos operativos y mejorando la continuidad de los servicios.

I. ANTECEDENTES

I.1 SITUACIÓN DE LA UAEH

La Universidad Autónoma del Estado de Hidalgo (UAEH) es una de las instituciones educativas más importantes de la región, con una infraestructura tecnológica que respalda sus funciones académicas y administrativas. Dentro de la universidad, el site es un espacio crítico donde se alojan los servidores y equipos de telecomunicaciones que garantizan el funcionamiento de sus plataformas digitales. Sin embargo, actualmente no se cuenta con un sistema de monitoreo automatizado que permita supervisar de manera continua las condiciones ambientales de este entorno.

Desde hace más de dos décadas, la UAEH ha trabajado en la modernización de su infraestructura tecnológica, implementando la Red Metropolitana de Fibra Óptica y la Red de Microondas WiMax. Esta red, con más de 50 kilómetros de fibra óptica y anchos de banda de hasta 10 Gbps, permite la integración de nuevas tecnologías como el protocolo de Internet IPv6, videoconferencias, telefonía IP y supercómputo (Universidad Autónoma del Estado de Hidalgo [UAEH], 2008). Además, esta infraestructura incluye un sistema de monitoreo y operación alojado en un data center, lo que refuerza la necesidad de optimizar su gestión mediante soluciones IoT.

En este contexto, dos áreas fundamentales dentro de la UAEH pueden desempeñar un papel clave en la implementación de un sistema de monitoreo ambiental basado en IoT:

Dirección de Información y Sistemas (DIyS):

La DIyS es responsable de diseñar y actualizar el sistema de información universitario, estableciendo la infraestructura de tecnologías de información y comunicaciones que se requiere en la institución. Su objetivo principal es coordinar el desarrollo, operación y aprovechamiento de las aplicaciones de los sistemas de información de la universidad, a través de la construcción de aplicaciones y metodologías que permitan potenciar dichos sistemas (UAEH, 2025). Algunas de sus funciones incluyen:

Coordinar las actividades de los sistemas de información en desarrollo y mantenimiento.

Realizar tareas de análisis, modelado y rediseño de procesos.

Promover la mejora continua en materia de sistemas de información universitaria.

A continuación, se muestra el organigrama de la dirección:

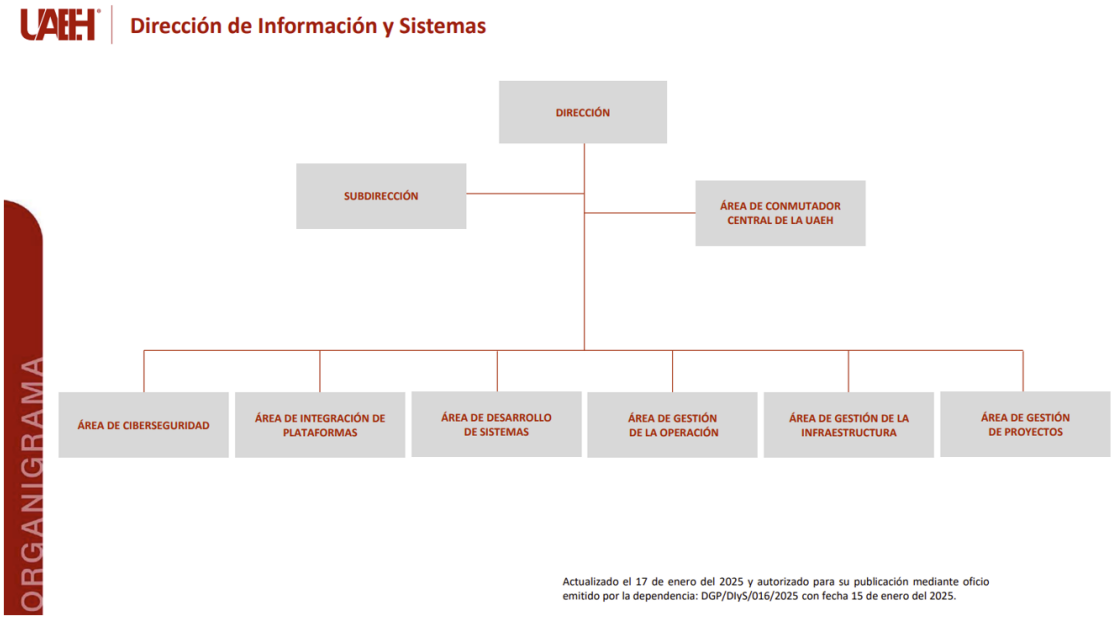

Figura I.1Organigrama de la Dirección de Información y Sistemas

Su experiencia en la gestión de plataformas tecnológicas convierte a la DIyS en un aliado estratégico para la adopción de un sistema de monitoreo ambiental en el site de la UAEH.

Centro de Vinculación Internacional y Desarrollo Educativo (CEVIDE):

El CEVIDE es la unidad encargada de la educación a distancia y la formación continua en la UAEH. Su infraestructura ha sido utilizada para eventos académicos y capacitaciones tecnológicas, como los WorkLabs 2016, donde se promovió la transferencia de conocimiento en nuevas tecnologías (UAEH, 2016). Este centro podría desempeñar un papel fundamental en la difusión y capacitación del personal sobre el uso del sistema de monitoreo ambiental.

La gestión de centros de datos (Data Centers) y sitios de procesamiento de datos ("sites") ha sido un aspecto crítico en la administración de infraestructuras tecnológicas durante décadas. Estos centros albergan servidores, equipos de red y otros sistemas esenciales que soportan la operación continua de servicios digitales en una organización. La importancia de mantener condiciones ambientales óptimas en estos sitios no puede

subestimarse, ya que las variaciones en la temperatura, humedad y calidad del aire pueden causar fallos significativos en el hardware, lo que conlleva a la interrupción de servicios críticos.

A nivel global, se han documentado numerosos incidentes donde la falta de un monitoreo ambiental adecuado ha resultado en sobrecalentamientos, fallos en sistemas de refrigeración, e incluso incendios, que han causado pérdidas económicas sustanciales y daños a la reputación de las organizaciones. Por ejemplo, el caso del apagón de Facebook en 2013, debido a problemas de sobrecalentamiento, subrayó la necesidad de sistemas robustos de monitoreo y control ambiental en los centros de datos (Chernicoff, 2013).

En entornos académicos, como las universidades, la dependencia de servicios digitales ha aumentado exponencialmente en los últimos años, impulsada por la digitalización de procesos administrativos y la adopción de plataformas de enseñanza en línea. Sin embargo, muchos de estos entornos no han actualizado sus infraestructuras para incluir sistemas de monitoreo ambiental automatizados, lo que los deja vulnerables a los riesgos mencionados. Tradicionalmente, estas instituciones han confiado en inspecciones manuales o en sensores no integrados, lo que limita la capacidad para responder rápidamente a las variaciones ambientales y mitigar riesgos potenciales.

El desarrollo de tecnologías como el Internet de las Cosas (IoT) ha permitido la creación de sistemas de monitoreo ambiental más avanzados y accesibles, capaces de proporcionar datos en tiempo real y alertas instantáneas. Estas innovaciones ofrecen la posibilidad de mejorar significativamente la seguridad y eficiencia operativa de los sitios de procesamiento de datos en instituciones educativas, permitiendo una gestión proactiva y basada en datos de los entornos tecnológicos críticos.

A pesar de la disponibilidad de estas tecnologías, muchas universidades aún no las han adoptado plenamente, lo que representa un área de mejora crucial. Implementar un sistema de monitoreo ambiental automatizado y centralizado podría no solo proteger los equipos y sistemas de la universidad, sino también asegurar la continuidad de los servicios digitales vitales para su comunidad educativa (Bonilla & Bonilla, 2024).

II. PLANTEAMIENTO DEL PROBLEMA

El site de la UAEH es un espacio crítico donde se alojan los servidores principales y los equipos de red que garantizan el funcionamiento de servicios académicos y administrativos esenciales para la institución. Sin embargo, este site carece de un sistema de monitoreo ambiental centralizado que permita un control preciso y continuo de variables fundamentales como la temperatura y la humedad en tiempo real y de manera proactiva.

Actualmente, la supervisión de las condiciones ambientales depende de revisiones manuales esporádicas y sensores básicos sin integración a un sistema automatizado, lo que representa un riesgo significativo para la infraestructura tecnológica. La falta de un monitoreo constante impide detectar oportunamente cambios en las condiciones del entorno, lo que podría derivar en fallos catastróficos en los servidores y equipos de comunicación.

El principal riesgo radica en la posible falla del sistema de climatización (minisplits), que, de no ser detectada a tiempo, podría generar un sobrecalentamiento de los equipos, afectando su rendimiento o incluso causando daños irreparables y pérdidas materiales y económicas importantes. Asimismo, niveles inadecuados de humedad pueden acelerar el deterioro de los componentes electrónicos, reduciendo su vida útil y aumentando los costos de mantenimiento y reemplazo.

Esta situación pone en peligro la estabilidad y seguridad de los servicios digitales de la universidad, afectando la operatividad de plataformas académicas, administrativas y de comunicación. En consecuencia, es imprescindible implementar una solución que permita el monitoreo automatizado en tiempo real y la generación de alertas para la toma de decisiones oportunas que garanticen la continuidad operativa de la infraestructura tecnológica.

III. JUSTIFICACIÓN

La implementación de un sistema de monitoreo de temperatura y humedad en el site de la Universidad Autónoma del Estado de Hidalgo es crucial para proteger la infraestructura tecnológica que sustenta tanto los servicios académicos como administrativos. Actualmente, el site depende de sistemas de climatización básicos (minisplits) y de revisiones manuales esporádicas, lo que limita la capacidad para detectar y responder a tiempo ante condiciones ambientales adversas, como cambios bruscos de temperatura o humedad. Esta situación incrementa el riesgo de fallos catastróficos que podrían afectar gravemente la continuidad operativa de los servicios críticos que dependen del correcto funcionamiento de los servidores y equipos de red.

Uno de los aspectos más importantes a considerar son los estándares internacionales para la gestión de las condiciones ambientales en centros de datos, publicados por la ASHRAE (American Society of Heating, Refrigerating and Air-Conditioning Engineers). En su guía ASHRAE TC 9.9, se establecen los rangos recomendados para temperatura y humedad, los cuales son cruciales para evitar problemas como la acumulación de electricidad estática o la formación de condensación, que podrían dañar irreversiblemente los equipos.

Temperatura: El rango recomendado es de 18 °C a 27 °C (64.4 °F a 80.6 °F), siendo el rango óptimo entre 20 °C y 25 °C (68 °F a 77 °F), ideal para balancear la eficiencia energética y la protección de los equipos.

Humedad relativa: El rango recomendado es de 40% a 60%. Humedades por debajo del 40% pueden provocar la acumulación de electricidad estática, mientras que humedades superiores al 60% aumentan el riesgo de condensación en los equipos.

ASHRAE también define clases de operación para los equipos en centros de datos, siendo la Clase A1 la más restrictiva y adecuada para entornos de misión crítica, como el site de la universidad, lo que significa que los equipos deben operar dentro de los rangos más controlados de temperatura y humedad para garantizar su seguridad y rendimiento (SHRAE TC9.9, 2021).

IV. OBJETIVOS

IV.1 Objetivo general

Implementar un sistema IoT de monitoreo ambiental en tiempo real para la supervisión de temperatura, humedad y otros parámetros críticos en el site de la UAEH, con el fin de prevenir fallos en los equipos, garantizar condiciones operativas óptimas y asegurar la continuidad de los servicios académicos y administrativos.

IV.2 Objetivos específicos

1. Analizar y optimizar el funcionamiento del sensor DHT11 para la medición de temperatura y humedad, aplicando metodologías de prueba y calibración con el fin de garantizar la precisión y confiabilidad de los datos recopilados.
2. Diseñar, desarrollar e implementar un sistema de sensores interconectados mediante IoT y MQTT que monitoree en tiempo real las condiciones ambientales del site, asegurando la continuidad operativa de los equipos tecnológicos.
3. Integrar y gestionar los datos recopilados en una plataforma centralizada basada en MongoDB y una interfaz web, utilizando herramientas de análisis para la toma de decisiones en la administración del site.
4. Evaluar y mejorar el desempeño del prototipo en el site de la UAEH mediante pruebas experimentales, identificando oportunidades de optimización en el monitoreo ambiental para su escalabilidad y futura implementación.

IV.3 Hipótesis

La implementación de un prototipo de IoT para el monitoreo en tiempo real de temperatura y humedad en el site de la UAEH mejorará la eficiencia en la detección de variaciones ambientales, reduciendo el riesgo de fallos en los equipos tecnológicos y asegurando la continuidad de los servicios académicos y administrativos.

IV.4 Alcance

El prototipo desarrollado contempla la implementación total para el monitoreo de temperatura y humedad en el site de la UAEH. Se desplegará el hardware y software basado en un ESP32 y un sensor DHT11, con transmisión de datos en tiempo real a una plataforma web. El sistema será instalado y probado en el site, asegurando su operación continua y su integración con la infraestructura existente. Además, se incluirán pruebas de

rendimiento, validación de datos y optimización del sistema para garantizar su fiabilidad y escalabilidad en un entorno de producción.

IV.5 Limitaciones

El dispositivo de monitoreo debe estar conectado de manera permanente a la red eléctrica del site, lo que implica que, en caso de corte de energía, el sistema dejaría de funcionar a menos que se utilicen soluciones como baterías de respaldo o sistemas UPS. Además, el sistema dependerá de la conexión Wi-Fi del site para transmitir los datos en tiempo real, lo que podría verse afectado por fallas, interferencias o inestabilidad en la red. Se requiere una red Wi-Fi segura y confiable para garantizar una transmisión constante de datos, lo que puede ser un desafío si la infraestructura de red es deficiente o está congestionada. El monitoreo se limitará exclusivamente al site.

V. MARCO TEÓRICO

Para llevar a cabo el desarrollo del presente proyecto es necesario tener en cuenta los siguientes conceptos que fundamentan y describen de forma clara y precisa las metodologías a utilizar para la recolección, análisis, e interpretación de la información.

V.1 Internet de las cosas

El Internet de las Cosas (IoT, por sus siglas en inglés) se refiere a la interconexión de dispositivos físicos a través de Internet, permitiendo la recopilación, transmisión y análisis de datos en tiempo real. Estos dispositivos, que pueden ser sensores, electrodomésticos, vehículos o cualquier objeto con capacidad de conectividad, están equipados con software, sensores y otras tecnologías que les permiten comunicarse e interactuar entre sí y con sistemas centralizados. IoT ha revolucionado diversos sectores, desde la industria hasta la salud, al permitir la automatización de procesos, la mejora de la eficiencia operativa y la toma de decisiones basada en datos (Gubbi et al., 2013; Zhang et al., 2023).

En el contexto actual, IoT ha evolucionado para incluir tecnologías avanzadas como la inteligencia artificial (IA) y el aprendizaje automático (ML), lo que permite no solo la recopilación de datos, sino también su análisis predictivo. Esto ha llevado a la creación de sistemas más inteligentes y autónomos, capaces de anticipar fallos y optimizar procesos en tiempo real (Zhang et al., 2023). Además, la integración de IoT con tecnologías emergentes como el 5G y el edge computing ha permitido una mayor escalabilidad y eficiencia en la transmisión de datos, especialmente en entornos con grandes volúmenes de información y baja latencia (Zhang et al., 2021).

Para implementar sistemas IoT, es fundamental comprender los componentes clave que permiten la interacción entre dispositivos y la recopilación de datos. Uno de estos componentes esenciales es el microcontrolador, que actúa como el cerebro de los dispositivos IoT.

V.2 Microcontrolador

Un microcontrolador es un circuito integrado que actúa como una pequeña computadora, optimizado para controlar dispositivos y sistemas específicos en tiempo real (Estudio Electrónica, 2023). A diferencia de las computadoras de propósito general, su diseño prioriza la interacción con el mundo físico a través de sensores y actuadores (IBM, 2023).

Sus componentes principales incluyen una unidad central de proceso (CPU) que ejecuta instrucciones, memoria RAM para datos temporales y memoria ROM para el programa permanente (Universidad Europea, 2023). Además, cuenta con periféricos de entrada y salida que facilitan la comunicación con el entorno externo (Sherlin.xBot.es, 2023).

Características principales:

La programabilidad permite adaptar su funcionalidad a diversas aplicaciones (Estudio Electrónica, 2023). Su tamaño compacto y bajo consumo energético lo hacen ideal para dispositivos portátiles y aplicaciones con baterías limitadas (IBM, 2023). Su bajo costo lo convierte en una opción atractiva para una amplia gama de proyectos (Sherlin.xBot.es, 2023).

Aplicaciones Comunes:

Los microcontroladores se encuentran en una variedad de aplicaciones, incluyendo robótica, automatización industrial, dispositivos electrónicos de consumo y el Internet de las Cosas (IoT) (Universidad Europea, 2023; IBM, 2023; Sherlin.xBot.es, 2023).

Dentro de los microcontroladores más utilizados en proyectos IoT se encuentra el ESP32, que destaca por su capacidad de conectividad y bajo consumo energético.

V.3 ESP32

El ESP32 es un microcontrolador de bajo costo y bajo consumo de energía, desarrollado por Espressif Systems, que integra Wi-Fi y Bluetooth de doble modo en un solo chip (Espressif, 2023). Su versatilidad y características lo convierten en una opción popular para proyectos de Internet de las Cosas (IoT), robótica, automatización del hogar, dispositivos portátiles y más (Microdesys, 2023; Amazon, 2023).

En la siguiente tabla se muestran las características clave:

Tabla V.1 Características ESP32

Característica	Especificación
Tipo	Módulo WiFi + Bluetooth
Modelo	ESP32 38 Pines
Chip USB-Serial	CP2102
Voltaje	3.3V DC (Entradas/Salidas) / 5V DC (Alimentación microUSB)
Consumo de energía	5µA en modo de suspensión
Microcontrolador	Dual Core Tensilica LX6, 240 MHz
Memoria SRAM	520 KB
Memoria Flash	4 MB
Wi-Fi	802.11 BGN HT40 (2.4 GHz hasta 150 Mbit/s)
Seguridad Wi-Fi	WEP, WPA/WPA2 PSK/Enterprise, AES, SHA2, RSA-4096, Criptografía de curva elíptica
Bluetooth	4.2 BR/EDR BLE (Modo de control dual)
GPIOs	30 pines
Interfaces	3x UART, 3x SPI, 2x I2C, 2x I2S, Interfaz de tarjeta SD
Convertidores	ADC de 12 canales, DAC de 2 canales
PWM	Sí
Dimensiones	55 x 28 x 8 mm (sin conectores)
Peso	11 g
Fuente: Elaboración propia a partir de la información consultada	

Figura V.1 Microcontrolador ESP32

Para que un sistema IoT funcione correctamente, es necesario contar con sensores que capturen datos del entorno. Uno de los sensores más comunes en aplicaciones de monitoreo ambiental es el sensor de temperatura y humedad DHT11.

V.4 Sensor de temperatura y humedad DHT11

El DHT11 es un sensor digital de temperatura y humedad relativa de bajo costo y fácil uso. Integra un sensor capacitivo de humedad y un termistor para medir el aire circundante, y muestra los datos mediante una señal digital en el pin de datos (no posee salida analógica).

El protocolo de comunicación entre el sensor y el microcontrolador emplea un único hilo o cable, la distancia máxima recomendable de longitud de cable es de 20m, de preferencia utilizar cable apantallado. Proteger el sensor de la luz directa del sol (radiación UV).

En la siguiente tabla se muestran las características relevantes del sensor:

Tabla V.2 Características sensor DHT11

Especificación	Detalle
Voltaje de Operación	3V – 5V DC
Rango de medición de temperatura	0 a 50 °C
Precisión de medición de temperatura	±2.0 °C
Resolución de temperatura	0.1 °C
Rango de medición de humedad	20% a 90% RH
Precisión de medición de humedad	±5% RH
Resolución de humedad	1% RH
Tiempo de respuesta	1 segundo

Aplicaciones:

- Utilizado en aplicaciones académicas relacionadas al control automático de temperatura.
- Aire acondicionado.
- Monitoreo ambiental en agricultura y más (AV Electronics, 2024b).

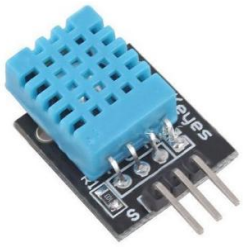

Figura V.2 Sensor DHT11

Una vez que los datos son capturados por el sensor, es necesario transmitirlos de manera eficiente. Para ello, se utiliza el protocolo Message Queuing Telemetry Transport (MQTT), que es ampliamente utilizado en sistemas IoT.

V.5 Message Queuing Telemetry Transport (MQTT)

MQTT es un protocolo de mensajería basado en estándares, o un conjunto de reglas, que se utiliza para la comunicación de un equipo a otro. Los sensores inteligentes, los dispositivos portátiles y otros dispositivos de Internet de las cosas (IoT) generalmente tienen que transmitir y recibir datos a través de una red con recursos restringidos y un ancho de banda limitado. Estos dispositivos IoT utilizan MQTT para la transmisión de datos, ya que resulta fácil de implementar y puede comunicar datos IoT de manera eficiente. MQTT admite la mensajería entre dispositivos a la nube y la nube al dispositivo (¿Qué Es el MQTT? - Explicación del Protocolo MQTT - AWS, s. f.).

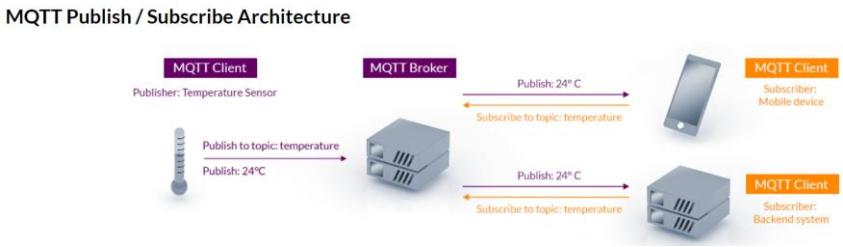

Figura V.3 MQTT publish/subcribe

Para que los dispositivos IoT puedan comunicarse, es esencial contar con una tecnología de conectividad inalámbrica, como el Wireless Fidelity (Wi-Fi).

V.6 Wireless Fidelity (Wi-Fi)

Wireless Fidelity (Wi-Fi o fidelidad inalámbrica), es una tecnología de redes inalámbricas que permite a los dispositivos electrónicos conectarse entre sí de manera fluida a una red mediante frecuencias de radio (*Proofpoint ES*, 2023).

Figura V.4 WI-FI

Además del WiFi, existen otras formas de comunicación inalámbrica que son relevantes en el contexto de IoT, especialmente en aplicaciones donde la distancia o el consumo energético son factores críticos.

V.7 Comunicación Inalámbrica

La comunicación inalámbrica o comunicación a distancia, es aquella capaz de enviar una cantidad de datos de un punto a otro (emisor y receptor) sin la necesidad de un agente o un hardware que conecte ambos puntos físicamente. En general, la tecnología inalámbrica utiliza ondas de radiofrecuencia de baja potencia y una banda específica, de uso libre o privada, para transmitir entre dispositivos (*Funcionamiento*, 2008).

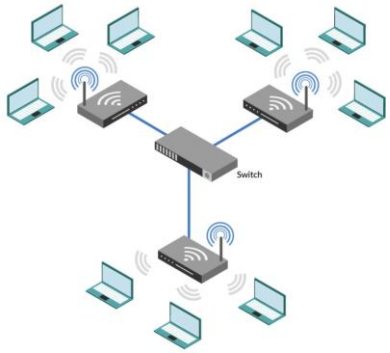

Figura V.5 comunicación inalámbrica

Una vez que los datos son transmitidos, es necesario almacenarlos de manera eficiente. Para ello, se utilizan bases de datos no relacionales, como las bases de datos NoSQL.

V.8 Base de datos no relacional NoSQL

Las bases de datos no relacional están diseñadas para varios modelos de acceso a datos, incluidas las aplicaciones de latencia baja. La base de datos de búsqueda no relacional está diseñada para realizar análisis de datos semiestructurados. Las bases de datos NoSQL son sistemas de almacenamiento de información que no cumplen con el esquema entidad-relación. Mientras que las tradicionales bases de datos relacionales basan su funcionamiento en tablas, joins y transacciones. Las bases de datos NoSQL no imponen una estructura de datos en forma de tablas y relaciones entre ellas, sino que proveen un esquema mucho más flexible. (Martín et al., 2013, p. 166)

Dentro de las bases de datos NoSQL más populares se encuentra MongoDB, que es ampliamente utilizada en proyectos IoT debido a su escalabilidad y facilidad de uso.

V.9 MongoDB

MongoDB es una base de datos NoSQL orientada a documentos, diseñada para almacenar grandes volúmenes de datos de manera flexible. A diferencia de las bases de datos relacionales, MongoDB no utiliza tablas ni filas, sino colecciones y documentos JSON, lo que permite un manejo más dinámico y escalable de los datos (IBM, 2023).

Para garantizar la seguridad en la transmisión y almacenamiento de datos, es común utilizar tokens de autenticación, como los JSON Web Tokens (JWT).

V.10 Json Web Token JWT

JWT es un estándar para representar reclamos entre dos partes de manera segura. Se utiliza principalmente en la autenticación de usuarios en aplicaciones web. Un JWT contiene un encabezado, un cuerpo y una firma, lo que garantiza la integridad de los datos y la autenticación sin la necesidad de almacenar el estado en el servidor (*CICS Transaction Server For Z/OS 6.x*, s. f.).

Además de los aspectos técnicos, es importante comprender los conceptos ambientales que se monitorean en sistemas IoT, como la temperatura y la humedad.

Para garantizar que los sistemas de monitoreo ambiental cumplan con los requisitos necesarios, es fundamental seguir estándares internacionales, como los establecidos por la ASHRAE.

V.11 ASHRAE y los Estándares para Centros de Datos

La American Society of Heating, Refrigerating and Air-Conditioning Engineers (ASHRAE) es una organización que establece estándares internacionales para el control ambiental en infraestructuras tecnológicas, incluyendo data centers y sites de telecomunicaciones. En su documento ASHRAE TC 9.9, se establecen las recomendaciones de temperatura y humedad que deben mantenerse en estos entornos para garantizar la estabilidad operativa y la longevidad de los equipos.

Según ASHRAE (2021), el rango recomendado de temperatura y humedad para los centros de datos clasificados en la Clase A1, que corresponde a equipos de misión crítica, es el siguiente:

- Temperatura: 18°C a 27°C (64.4°F a 80.6°F), siendo el rango óptimo entre 20°C y 25°C (68°F a 77°F).
- Humedad relativa: 40% a 60%.

El mantenimiento de estos parámetros es crucial para evitar problemas como:

- Temperaturas elevadas: Pueden provocar sobrecalentamiento, degradación prematura de los componentes electrónicos y fallos en servidores.
- Temperaturas bajas: Pueden generar condensación y fallos eléctricos.
- Humedad baja (<40%): Aumenta la acumulación de electricidad estática, que puede dañar los equipos sensibles.

Humedad alta (>60%): Favorece la corrosión de componentes y la aparición de condensación, lo que puede causar cortocircuitos. Para desarrollar sistemas IoT, es necesario utilizar herramientas y lenguajes de programación específicos, como el Arduino IDE, HTML, CSS, JavaScript, Node.js, Docker, API y Vite.

V.12 Arduino Integrated Development Environment IDE

El entorno de programación de la placa de Arduino se denomina Integrated Development Environment (IDE) el cual permite llevar a cabo la escritura de las sentencias para el funcionamiento de los elementos físicos de la placa de Arduino. Este software tiene por si solo un conjunto de herramientas que permite, editar el código, compilar y depurar todo a través de una interfaz gráfica, así mismo, nos da la oportunidad de interactuar con el microcontrolador almacenando los programas realizados en su memoria interna para poner en marcha todo el hardware. El aplicativo de IDE es un software libre, ya que está disponible su código fuente mismo que está alojado en GitHub y ofrece instrucciones de compilado Pérez-Tavera (2023).

V.13 Hypertext Markup Language (HTML)

Lenguaje de marcado, utilizado para estructurar y presentar contenido web. Introdujo nuevas etiquetas semánticas como <article>, <section>, y <footer>, lo que mejora la

accesibilidad y SEO de las páginas web. Además, añadió soporte para multimedia con elementos como <audio> y <video>, eliminando la necesidad de plugins externos como Flash (HTML5 - MDN Web Docs Glossary: Definitions Of Web-related Terms | MDN, 2024).

HTML5 (HyperText Markup Language, versión 5) representa la quinta revisión fundamental del lenguaje estándar para estructurar y presentar contenido en la World Wide Web. Desarrollado por el World Wide Web Consortium (W3C) y el Web Hypertext Application Technology Working Group (WHATWG), este estándar fue publicado oficialmente en octubre de 2014, marcando un hito en el desarrollo web (HTML5 - MDN Web Docs Glossary: Definitions Of Web-related Terms | MDN, 2024).

Avances significativos en semántica:

HTML5 introdujo un conjunto revolucionario de elementos semánticos que trascienden la simple presentación para dotar de significado estructural al contenido:

- <article>: Define contenido independiente y autónomo (entradas de blog, artículos de noticias)
- <section>: Delimita agrupaciones temáticas de contenido
- <nav>: Especifica bloques de navegación principal
- <header> y <footer>: Marcan cabeceras y pies de sección/documento
- <main>: Identifica el contenido principal de la página

Estas etiquetas mejoran sustancialmente:

- Accesibilidad: Los lectores de pantalla pueden interpretar mejor la estructura lógica
- SEO: Los motores de búsqueda comprenden con mayor precisión la jerarquía del contenido
- Mantenibilidad: El código se organiza de forma más lógica y comprensible

Impacto en el desarrollo web moderno:

Según el reporte anual de W3Techs (2023), el 94.3% de todos los sitios web utilizan actualmente HTML5, demostrando su adopción como estándar dominante. Su diseño forward-compatible garantiza que el contenido permanecerá accesible incluso con futuras evoluciones del lenguaje.

V.14 Cascading Style Sheets (CSS)

CSS3 representa la culminación de años de evolución en el lenguaje de hojas de estilo, estableciendo un nuevo paradigma en el diseño web moderno. A diferencia de sus predecesores, CSS3 se desarrolló como un conjunto modular de especificaciones que permitieron una implementación gradual y más eficiente por parte de los navegadores (CSS | MDN, 2024). Esta arquitectura modular ha facilitado la adopción progresiva de nuevas características sin romper la compatibilidad con versiones anteriores.

La verdadera revolución de CSS3 radica en su capacidad para crear experiencias visuales ricas y dinámicas directamente en el navegador, sin depender de tecnologías externas. Las transiciones y animaciones, por ejemplo, han liberado a los desarrolladores de la dependencia de JavaScript para efectos visuales básicos, permitiendo crear interfaces más fluidas con un código más limpio y mantenible. Las transformaciones en 2D y 3D, por su parte, han abierto nuevas posibilidades en el diseño de interfaces modernas, desde sutiles efectos de hover hasta complejas composiciones espaciales.

El diseño responsivo experimentó un salto cualitativo con la maduración de las media queries, que permiten adaptar el layout no solo al tamaño de pantalla, sino también a características específicas del dispositivo como la resolución, la orientación o incluso las preferencias del usuario. Esta capacidad ha sido fundamental en una era donde el acceso móvil supera ampliamente al escritorio.

Los nuevos sistemas de layout como Flexbox y Grid han resuelto problemas que perseguían a los desarrolladores durante años, ofreciendo finalmente un control preciso sobre el diseño y la distribución espacial de los elementos. Flexbox simplificó enormemente la creación de layouts flexibles en una dimensión, mientras que Grid introdujo un sistema bidimensional completo y potente.

La inclusión de características tipográficas avanzadas, variables CSS y capacidades de filtrado ha completado un ecosistema que permite crear diseños consistentes y sofisticados con un grado de control sin precedentes. Todo esto mientras se mantiene la filosofía central de CSS: separar claramente la presentación del contenido.

V.15 Javascript

JavaScript es un lenguaje de programación orientado a eventos, esencial para agregar interactividad a las páginas web. Junto con HTML5 y CSS3, forma el núcleo del desarrollo

web moderno. Es un lenguaje versátil que puede ser utilizado tanto en el frontend como en el back-end, sobre todo con la aparición de frameworks y bibliotecas como React y Node.js (JavaScript | MDN, 2023).

Un Lenguaje con Alcance Universal

Lo que distingue a JavaScript es su versatilidad sin precedentes. Originalmente concebido para ejecutarse en el navegador, hoy se ha expandido a prácticamente todos los ámbitos del desarrollo de software:

Frontend Moderno

- Frameworks como React, Angular y Vue.js han redefinido la creación de interfaces de usuario
- Permite construir Single Page Applications (SPAs) con experiencias fluidas similares a aplicaciones nativas
- Facilita el desarrollo de interfaces reactivas mediante el Virtual DOM

Backend y Más Allá

- Node.js permitió la ejecución de JavaScript en el servidor
- Ecosistema de paquetes (npm) más grande del mundo
- Uso creciente en IoT, machine learning (TensorFlow.js) y desarrollo móvil (React Native)

Características Técnicas Clave

JavaScript es un lenguaje:

- Multi-paradigma: Soporta programación orientada a objetos, funcional y basada en eventos
- No bloqueante: Modelo de ejecución asíncrono mediante el event loop
- Dinámico: Tipado débil y flexible que acelera el desarrollo
- Universal: Ejecutable en cualquier navegador sin necesidad de compilación

Impacto en la Web Moderna

Según el informe Stack Overflow 2023, JavaScript lleva 10 años consecutivos como el lenguaje más utilizado. Su evolución continua (ES6+, TypeScript) asegura su relevancia frente a nuevos desafíos como:

- Aplicaciones web progresivas (PWAs)
- Renderizado del lado del servidor (SSR)
- WebAssembly para tareas intensivas

V.16 Node.js

Node.js es un entorno de ejecución que permite ejecutar JavaScript del lado del servidor. Basado en el motor V8 de Google, Node.js es altamente eficiente, especialmente para aplicaciones que requieren manejar múltiples solicitudes de forma concurrente. Su arquitectura no bloqueante y orientada a eventos lo hace ideal para aplicaciones en tiempo real, como chats o transmisiones de datos (Kinsta, 2023).

Motor V8: Compilador JIT (Just-in-Time) que transforma JavaScript a código máquina nativo, logrando performance comparable a lenguajes compilados

- Optimizaciones: Inline caching, hidden classes
- Gestión de memoria: Generational garbage collection

Event Loop: Implementación del patrón Reactor que maneja operaciones asíncronas mediante:

- Fase de polling (epoll/kqueue/IOCP)
- Cola de callbacks (FIFO)
- Thread pool para operaciones blocking (libuv)

Módulo Cluster: Permite escalamiento vertical mediante:

- Fork de procesos hijos (master-worker)
- Balanceo de carga round-robin

V.17 Docker

Docker es una plataforma que permite crear, ejecutar y gestionar contenedores. Los contenedores empaquetan el código de una aplicación junto con sus dependencias, asegurando que la aplicación se ejecute de manera uniforme en cualquier entorno. Docker facilita el despliegue y la escalabilidad de aplicaciones al separar las aplicaciones del sistema operativo (Contenedores de Docker | ¿Qué Es Docker? | AWS, s. f.).

V.18 Application Programming Interface (API)

Una API es un conjunto de reglas y protocolos que permite la comunicación entre diferentes aplicaciones o servicios. Las APIs son fundamentales en el desarrollo de aplicaciones web modernas, ya que permiten la interacción entre el frontend y el backend. Existen diferentes tipos de APIs, como las RESTful APIs, que utilizan métodos HTTP (GET, POST, PUT, DELETE) para la comunicación (Introduction To Web APIs - Learn Web Development | MDN, 2024).

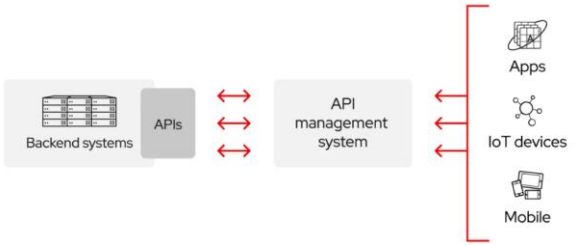

Figura V.6 API

V.19 VITE

Vite es una herramienta de construcción rápida para proyectos de frontend, que aprovecha el soporte nativo de los módulos ES de los navegadores modernos. Se diferencia de otras herramientas como Webpack por su velocidad y simplicidad, ya que inicia un servidor de desarrollo casi al instante sin necesidad de empaquetar el código (Introducción, s. f.).

Además de los aspectos técnicos, es importante comprender los conceptos ambientales que se monitorean en sistemas IoT, como la temperatura y la humedad.

V.20 Temperatura

La temperatura es una magnitud escalar que se define como la cantidad de energía cinética de las partículas de una masa gaseosa, líquida o sólida. Cuanto mayor es la velocidad de las partículas, mayor es la temperatura y viceversa.

Existen distintos tipos de escalas para medir la temperatura. Las más comunes son:

La escala Celsius. También conocida como “escala centígrada”, es la más utilizada junto con la escala Fahrenheit. En esta escala, el punto de congelación del agua equivale a 0 °C (cero grados centígrados) y su punto de ebullición a 100 °C.

La escala Fahrenheit. Es la medida utilizada en la mayoría de los países de habla inglesa. En esta escala, el punto de congelación del agua ocurre a los 32 °F (treinta y dos grados Fahrenheit) y su punto de ebullición a los 212 °F.

La escala Kelvin. Es la medida que suele utilizarse en ciencia y establece el “cero absolutos” como punto cero, lo que supone que el objeto no desprende calor alguno y equivale a -273,15 °C (grados centígrados).

La escala Rankine. Es la medida usada comúnmente en Estados Unidos para la medición de temperatura termodinámica y se define al medir los grados Fahrenheit sobre el cero absoluto, por lo que carece de valores negativos o bajo cero.

La temperatura se mide mediante magnitudes termométricas, es decir, diferentes unidades que representan la temperatura a distintas escalas. Para eso se emplea un dispositivo llamado “termómetro” del que existen varios tipos dependiendo del fenómeno que se necesite medir, por ejemplo:

Dilatación y contracción. Existen termómetros para medir los gases (termómetro de gas a presión constante), los líquidos (termómetro de mercurio) y los sólidos (termómetro de columna líquida o bimetálico), que son elementos que se expanden con temperaturas altas o se contraen con temperaturas bajas.

Variación de resistencia eléctrica. Las resistencias eléctricas, es decir, los flujos de electrones que se mueven a través de un material conductor varían según la temperatura que adquieren. Para su medición se emplean termómetros de resistencia eléctrica como los sensores (en base a una resistencia capaz de transformar la variación eléctrica en una variación de temperatura) y los termoeléctricos (que generan fuerza motriz).

Termómetro de radiación térmica. Los fenómenos de radiación emitidos en el sector industrial pueden ser medidos mediante sensores de temperatura como los pirómetros infrarrojos (para medir temperaturas muy bajas de refrigeración) y los pirómetros ópticos (para medir altas temperaturas de hornos y metales de fusión).

Potencial termoeléctrico. La unión de dos metales diferentes que se someten a temperaturas distintas entre sí genera una fuerza electromotriz que se convierte en potencial eléctrico y que se mide en voltios Leskow (2024).

V.21 Humedad

La humedad es la cantidad de vapor de agua que contiene el aire. Siempre hay vapor de agua en el aire y la cantidad varía según diversos factores, por ejemplo, si recién llovió, si se está cerca del mar, si hay o no vegetación en el terreno, la temperatura del aire, entre otros.

Los tipos de humedad pueden ser:

Humedad relativa. Es la capacidad del aire para almacenar agua, que depende del punto de saturación (límite de ese aire para contener agua) y de una determinada temperatura. Su cálculo se define por el cociente entre la cantidad de vapor presente en la atmósfera, dividido la máxima cantidad que podría contener, multiplicado por cien (y el resultado se expresa como porcentaje). Una humedad relativa del 100% indica que ha llegado a su punto límite de saturación y, a partir de ahí, todo excedente de vapor de agua se condensa (se convierte en líquido).

Humedad absoluta. Es la masa de vapor de agua que está presente en determinado volumen de aire, antes de que sea condensada (humedad relativa). Es importante destacar que la temperatura condiciona a la humedad absoluta: las masas de aire caliente poseen mayor capacidad de almacenar vapor de agua que las masas de aire frío. La humedad absoluta se expresa en gramos por metro cúbico.

Humedad normal: El rango de humedad de un laboratorio debe estar entre el 35 - 70 %, considerándose ideal entre el 35 - 55 % (Equipo editorial, Etecé, 2020).

VI. ESTADO DEL ARTE

El Internet de las Cosas (IoT) ha revolucionado el monitoreo ambiental en diversos sectores, proporcionando soluciones eficientes para la supervisión de variables como temperatura y humedad en tiempo real. La evolución de esta tecnología ha permitido la integración de sensores con plataformas digitales, facilitando la toma de decisiones basada en datos y optimizando el mantenimiento de infraestructuras críticas. A medida que la digitalización y la automatización avanzan, los sistemas IoT se han convertido en una pieza clave para mejorar la eficiencia operativa en múltiples industrias (Gubbi et al., 2013).

Aplicaciones de IoT en Diferentes Sectores

A lo largo de los años, el IoT ha demostrado su aplicabilidad en distintas áreas, como la manufactura, la salud, la agricultura y la gestión de ciudades inteligentes. En el ámbito industrial, como menciona Zhang y sus colaboradores en 2023, la adopción de sistemas IoT ha permitido la implementación de redes de monitoreo ambiental que optimizan la eficiencia energética y previenen fallos en maquinaria crítica. Sensores avanzados y dispositivos conectados recopilan datos en tiempo real sobre condiciones ambientales y operativas, lo que permite predecir fallos antes de que ocurran y reducir costos de mantenimiento.

En 2015, Islam y algunos colegas del sector mencionan que los sensores IoT han mejorado el seguimiento de pacientes mediante dispositivos portátiles que transmiten datos biomédicos en tiempo real, asegurando una respuesta rápida ante emergencias (Islam et al., 2015). Dispositivos como marcapasos inteligentes y monitores de glucosa permiten a los profesionales de la salud monitorear a los pacientes de manera remota, optimizando la atención médica y reduciendo hospitalizaciones innecesarias.

En la agricultura, el uso de IoT ha dado paso a la agricultura de precisión, donde sensores monitorean la humedad del suelo, la temperatura y otros factores ambientales para optimizar el uso de recursos como el agua y los fertilizantes (Wolfert et al., 2017). Esta tecnología permite tomar decisiones basadas en datos en tiempo real, mejorando la productividad y sostenibilidad de los cultivos.

En el contexto urbano, las ciudades inteligentes han implementado IoT en sistemas de tráfico, iluminación, recolección de residuos y monitoreo de la calidad del aire. Sensores

conectados a la infraestructura urbana optimizan el consumo energético y mejoran la gestión de servicios públicos, reduciendo costos y mejorando la calidad de vida de los ciudadanos (Zanella et al., 2014).

Tendencias Recientes en IoT para el Monitoreo Ambiental

El monitoreo ambiental mediante IoT ha cobrado una gran importancia en los últimos años debido a la creciente necesidad de garantizar la eficiencia y seguridad de los sistemas tecnológicos en infraestructuras críticas. Estudios recientes han demostrado que el uso de microcontroladores como el ESP32 y sensores como el DHT11 ofrece una solución accesible para la supervisión de temperatura y humedad en centros de datos (Huancayo, 2022).

Otra tendencia en IoT es el uso de computación en el borde (Edge Computing), donde los datos se procesan cerca del sensor, reduciendo la latencia y el tráfico en la red. Esta tecnología es fundamental para sistemas de monitoreo ambiental en tiempo real, ya que permite respuestas rápidas sin necesidad de enviar todos los datos a la nube. Estudios como los de Zhang, Qian y He (2021) han demostrado que el protocolo MQTT es una de las opciones más viables para estos entornos debido a su bajo consumo de ancho de banda y capacidad para operar en redes con conectividad inestable.

Además, la implementación de inteligencia artificial (IA) en IoT ha permitido que los sistemas de monitoreo ambiental no solo recopilen datos, sino que también detecten patrones y generen predicciones sobre posibles fallos en la infraestructura. Por ejemplo, un análisis comparativo realizado por Gharavi y Ghafurian (2021) sobre redes eléctricas inteligentes subraya la necesidad de integrar sensores IoT con modelos de IA para mejorar la gestión de recursos y prevenir fallos en el suministro energético.

Normativas y Estándares para IoT en Centros de Datos

En el ámbito de la gestión de centros de datos, la normatividad establecida por la ASHRAE (American Society of Heating, Refrigerating and Air-Conditioning Engineers) ha definido estándares clave para la regulación de temperatura y humedad en estos entornos. Según ASHRAE TC 9.9 (2021), la temperatura en un data center debe mantenerse entre 18 °C y 27 °C, con un rango óptimo de 20 °C a 25 °C, mientras que la humedad relativa debe estar entre el 40% y el 60%.

Desafíos en la Implementación de IoT en el Monitoreo Ambiental

Uno de los principales desafíos en la implementación de IoT es la seguridad y privacidad de los datos, ya que los dispositivos IoT están expuestos a posibles ciberataques. La adopción de protocolos seguros como TLS y la autenticación mediante JWT han sido estrategias utilizadas para mitigar estos riesgos (CICS Transaction Server for Z/OS 6.x, s.f.).

Otro desafío es la dependencia de la conectividad a Internet, especialmente en entornos donde las redes pueden ser inestables. Para abordar este problema, se han desarrollado soluciones que permiten el almacenamiento temporal de datos en los dispositivos IoT y su transmisión cuando la conexión se restablece (Gubbi et al., 2013).

Además, la escalabilidad de los sistemas IoT es un factor crítico. A medida que aumenta la cantidad de dispositivos conectados, se requiere una infraestructura robusta para gestionar el tráfico de datos y garantizar tiempos de respuesta adecuados. Tecnologías emergentes como 5G y LoRaWAN han sido implementadas para mejorar la cobertura y la eficiencia en la transmisión de datos en redes IoT de gran escala (Zhang et al., 2021).

VII. DESARROLLO

El desarrollo del prototipo se llevó a cabo en varias etapas, comenzando con la configuración del hardware y software, seguido de la implementación del sistema de monitoreo y la validación de los resultados.

VII.1 Descripción del prototipo

El prototipo desarrollado tiene como objetivo principal monitorear la temperatura y humedad dentro del site de la Universidad Autónoma del Estado de Hidalgo, utilizando un sensor DHT11. Este sistema está basado en el microcontrolador ESP32, que se encarga de procesar y enviar los datos recolectados hacia una plataforma centralizada mediante el protocolo MQTT, donde posteriormente son almacenados en una base de datos NoSQL para su visualización y análisis en tiempo real.

VII.2 Componentes del prototipo

El prototipo está compuesto por los siguientes elementos clave:

1. Sensor DHT11: Este sensor digital es utilizado para medir la temperatura y la humedad del ambiente. Sus características principales incluyen:
    - Rango de medición de temperatura: 0°C a 50°C, con una precisión de ±2°C.
    - Rango de medición de humedad: 20% a 90% de humedad relativa, con una precisión de ±5%.
    - Comunicación digital: Utiliza un único pin de datos para transmitir las lecturas al ESP32, lo que simplifica la conexión.
2. ESP32: El ESP32 es un microcontrolador de bajo costo y alto rendimiento, que cuenta con conectividad WiFi y Bluetooth integrados, lo que permite la transmisión inalámbrica de datos. Las principales características que lo hacen ideal para este prototipo incluyen:
    - WiFi integrado: Permite la conexión a redes inalámbricas para la transmisión de datos en tiempo real.
    - Programable con Arduino IDE: Facilita la implementación de código para la adquisición y procesamiento de datos.

- Eficiencia energética: El ESP32 está diseñado para operar con bajo consumo de energía, lo que es ideal para un sistema de monitoreo continuo.

VII.3 Funcionamiento del prototipo

El prototipo sigue un flujo continuo para la captura y transmisión de datos ambientales, el cual se detalla a continuación:

1. Captura de datos con el sensor DHT11: El sensor DHT11 mide la temperatura en grados Celsius (°C) y la humedad en porcentaje de humedad relativa (%HR) del ambiente dentro del site. Las mediciones se toman de manera periódica en intervalos confiables, lo que asegura que el sistema tenga datos actualizados de las condiciones ambientales sin comprometer el rendimiento del sistema ni agotar la batería del ESP32.
2. Procesamiento de datos en el ESP32: Una vez que el sensor DHT11 realiza una medición, los datos se envían al ESP32 a través de su pin de datos. El ESP32, programado con el entorno Arduino IDE, procesa estas lecturas y las prepara para su envío. La programación del ESP32 incluye:
    - Lectura de las señales digitales del DHT11: Convertir las señales provenientes del sensor en valores legibles de temperatura y humedad.
    - Validación de las lecturas: Asegurar que los datos no estén corruptos o fuera de los rangos permitidos.
    - Preparación para el envío: Los valores de temperatura y humedad son empaquetados en un mensaje que será transmitido al broker MQTT en formato JSON.
3. Transmisión de datos mediante MQTT: El ESP32 utiliza su módulo WiFi integrado para conectarse a una red local previamente configurada, especificando los parámetros de red como el SSID y la contraseña. Posteriormente, utiliza el protocolo MQTT para enviar los datos al broker MQTT. Este protocolo fue seleccionado por su eficiencia en la transmisión de datos en redes de ancho de banda limitado y su capacidad de operar en entornos con latencia.

El flujo de comunicación es el siguiente:

- Conexión a la red WiFi: El ESP32 establece una conexión segura con la red WiFi del site.
- Publicación de datos: Los valores de temperatura y humedad son enviados a un topic específico en el broker MQTT.
- Broker MQTT: Actúa como un intermediario, recibiendo los datos del ESP32 y retransmitiéndolos a los suscriptores correspondientes, en este caso, un servidor que almacenará los datos.

4. Almacenamiento en MongoDB: Los datos de temperatura y humedad transmitidos por el ESP32 son recibidos por el servidor, que actúa como un suscriptor del broker MQTT. Este servidor, utilizando un backend desarrollado en Node.js, almacena los datos en una base de datos NoSQL MongoDB. La elección de MongoDB se debe a su flexibilidad en el manejo de datos no estructurados y a su capacidad para gestionar grandes volúmenes de información sin sacrificar el rendimiento.
5. Intervalo de envío de datos: El intervalo de envío de datos ha sido configurado en 30 segundos para balancear la frecuencia de actualizaciones con el consumo energético del sistema. Este intervalo es suficientemente corto para ofrecer un monitoreo en tiempo real, pero lo suficientemente largo para no agotar los recursos del ESP32 y la red WiFi. En un entorno de producción, este intervalo podría ajustarse según las necesidades del site.

VII.4 Validación del prototipo

Para asegurar la fiabilidad del sistema, se realizarán pruebas en las siguientes áreas:

- Pruebas de precisión: Comparación de lecturas del DHT11 con un sensor de mayor precisión (DHT22).
- Pruebas de conectividad: Evaluación de la estabilidad de la transmisión WiFi y el rendimiento del protocolo MQTT.
- Pruebas de carga: Simulación con múltiples sensores para analizar el comportamiento del sistema bajo alta demanda.
- Pruebas de latencia: Medición del tiempo desde la captura del dato hasta su visualización en la interfaz.

VII.5 Visualización de los datos en tiempo real

El sistema también incluye una interfaz web desarrollada con HTML5, CSS3 y JavaScript, que permite visualizar en tiempo real las mediciones de temperatura y humedad capturadas por el ESP32. La interfaz es accesible desde cualquier dispositivo con conexión a Internet, permitiendo que los usuarios supervisen el estado del site desde cualquier lugar. Además de mostrar las lecturas actuales, la plataforma permite consultar datos históricos almacenados en MongoDB, lo que facilita la detección de patrones o anomalías en el ambiente del site.

VII.6 Arquitectura del sistema

La arquitectura se divide en tres capas principales: hardware, software y comunicación, que trabajan de manera conjunta para ofrecer un sistema robusto y fácil de mantener.

VII.7 Configuración del hardware

El hardware del sistema se compone principalmente del microcontrolador ESP32 y el sensor DHT11. El ESP32 fue seleccionado por su capacidad de conectividad WiFi integrada y su bajo consumo energético, lo que lo hace ideal para aplicaciones de monitoreo continuo. El sensor DHT11, por su parte, fue elegido por su bajo costo y facilidad de uso, aunque con limitaciones en la precisión (±2°C para temperatura y ±5% para humedad).

Durante la configuración del hardware, se enfrentaron algunos desafíos técnicos, como la interferencia en la señal del sensor debido a la proximidad de otros dispositivos electrónicos. Para resolver este problema, se implementó un cable apantallado entre el sensor y el ESP32, lo que mejoró la estabilidad de las lecturas.

- Sensor DHT11: Este sensor digital se encarga de medir la temperatura y la humedad dentro del site. Está conectado directamente al ESP32 mediante uno de sus pines de entrada, utilizando un protocolo de comunicación simple y eficiente que le permite enviar lecturas precisas al microcontrolador. El sensor tiene un rango de medición adecuado para el entorno del centro de datos y es capaz de detectar cambios ambientales significativos que podrían afectar el rendimiento de los equipos dentro del site.
- ESP32: El microcontrolador ESP32 es el cerebro del sistema. Este dispositivo, programado a través del entorno Arduino IDE, procesa las lecturas de temperatura

y humedad, y las transmite a través de su módulo WiFi incorporado. Su capacidad para conectarse a redes inalámbricas lo convierte en una opción ideal para este tipo de sistemas, ya que no requiere cables adicionales para transmitir los datos. Además, el ESP32 puede conectarse a varios sensores en caso de que sea necesario monitorear múltiples puntos dentro del site.

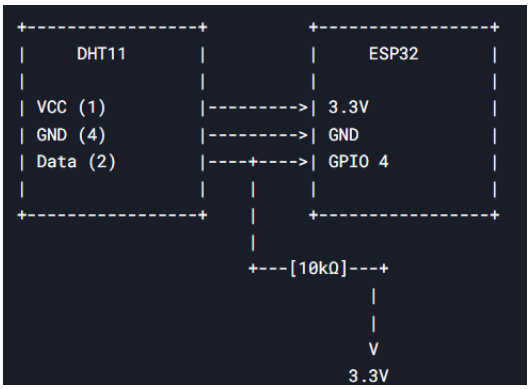

Figura VII.1 Conexión sensor DHT11 a ESP32

En la siguiente imagen se muestra el prototipo para tomar las lecturas:

Figura VII.2 Prototipo sistema de monitoreo de temperatura y humedad

VII.8 Implementación del software

El sistema de monitoreo está impulsado por un software que sigue una estructura organizada y modular, facilitando tanto la lectura de datos como la visualización y almacenamiento de los mismos. Los componentes de software incluyen:

- Código en ESP32 (Arduino IDE): El código programado en el ESP32 tiene dos funciones principales: leer los datos del sensor DHT11 y transmitir esos datos a un servidor central mediante el protocolo MQTT. Este código está diseñado para operar de manera continua, con intervalos de 30 segundos entre cada medición, lo que garantiza una actualización constante de los datos ambientales. Adicionalmente, el código maneja situaciones de error, como la falta de conexión a la red WiFi o la recepción de datos erróneos del sensor, intentando reconectar o leer de nuevo hasta que los problemas se resuelvan.
- Servidor MQTT y broker: El protocolo MQTT se utiliza para transmitir los datos desde el ESP32 hasta el servidor central. MQTT es ideal para este tipo de aplicaciones debido a su eficiencia en redes con ancho de banda limitado. El ESP32 publica los datos de temperatura y humedad en un "topic" del broker MQTT, donde el servidor Node.js suscribe a esos datos y los almacena en una base de datos MongoDB.

El siguiente diagrama muestra el ciclo de funcionamiento desde la captura de los datos hasta su almacenamiento:

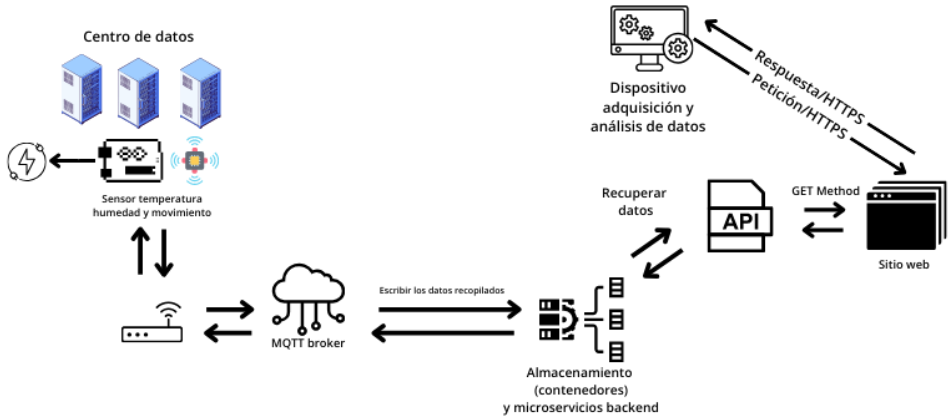

Figura VII.3 Arquitectura de funcionamiento del sistema

VII.9 Comunicación

La comunicación en el sistema de monitoreo sigue una arquitectura cliente-servidor, donde el ESP32 actúa como cliente y el servidor (implementado en Node.js) actúa como receptor de los datos publicados. La arquitectura de comunicación incluye:

- Protocolo MQTT: Este protocolo de mensajería ligera facilita la transmisión de los datos entre el ESP32 y el servidor. MQTT utiliza un modelo de "publish/subscribe" donde el ESP32, como cliente, publica los datos en un broker MQTT, y el servidor central, como suscriptor, recibe esos datos en tiempo real. El protocolo está diseñado para operar en entornos con latencia y garantiza la entrega de los mensajes a pesar de interrupciones temporales en la red.
- Conectividad WiFi: El ESP32 se conecta a la red WiFi del site para transmitir los datos. La red inalámbrica debe estar configurada con acceso a Internet para asegurar que el ESP32 pueda enviar los datos al broker MQTT alojado en un servidor externo. La configuración de red, incluyendo el SSID y la clave de acceso, se almacena en el ESP32, permitiendo que el dispositivo se conecte de manera automática tras reiniciar.
- Servidor Node.js: El servidor, desarrollado en Node.js, recibe los datos que el ESP32 envía a través del broker MQTT y los almacena en una base de datos MongoDB. Además, este servidor expone una API que permite a los clientes, como la interfaz web, consultar los datos almacenados en tiempo real y acceder a los datos históricos.

VII.10 Desarrollo de la interfaz web

El diseño de la interfaz web se desarrolló utilizando HTML5, CSS3, y JavaScript, con un enfoque en el diseño responsive para asegurar que la visualización sea óptima en dispositivos con distintos tamaños de pantalla. La estructura de la página se adapta automáticamente para garantizar que los usuarios puedan visualizar los datos sin importar si acceden desde una computadora de escritorio, una tableta o un teléfono móvil.

Figura VII.4 Página principal del sistema

Figura VII.5 Lista de centros de datos agregados en el sistema

La interfaz permite monitorear la temperatura y humedad en tiempo real, actualizando los valores cada 30 segundos, que es el intervalo de medición configurado en el ESP32. Los

datos se presentan de manera gráfica y numérica para facilitar su interpretación. Se utilizan gráficas dinámicas para mostrar la temperatura y humedad en el site, lo que permite al usuario detectar posibles anomalías de manera visual.

Figura VII.6 Gráficas del sistema

La interfaz permite no solo visualizar los datos en tiempo real, sino también acceder a los datos históricos almacenados en la base de datos MongoDB. Los usuarios pueden seleccionar un rango de fechas y horas para consultar las lecturas pasadas, lo que es útil para identificar patrones o posibles problemas en el site a lo largo del tiempo.

VII.11 Back-End: Procesamiento y almacenamiento de datos

El back-end del sistema de monitoreo ambiental es el núcleo encargado de procesar y almacenar los datos enviados por los sensores conectados al ESP32. Para garantizar la eficiencia y escalabilidad del sistema, se ha implementado una arquitectura basada en Node.js para procesar los datos y MongoDB como base de datos NoSQL, que almacena de manera segura las lecturas de temperatura y humedad.

Estructura general del back-end

El back-end se diseñó como un API RESTful que recibe los datos de los sensores, los procesa, y los almacena en una base de datos. Este enfoque permite que el sistema sea flexible y fácilmente escalable, facilitando la integración de nuevos sensores o futuras ampliaciones del sistema.

El flujo de trabajo del back-end es el siguiente:

1. El ESP32 envía los datos de temperatura y humedad al servidor mediante solicitudes HTTP POST al endpoint correspondiente.
2. El servidor recibe los datos y realiza las validaciones necesarias para asegurarse de que los datos sean correctos y completos.
3. Los datos se almacenan en MongoDB, donde se organizan en documentos que incluyen las lecturas junto con una marca temporal.
4. La API también ofrece endpoints para recuperar los datos almacenados, permitiendo a la interfaz web mostrar la información en tiempo real o consultar datos históricos.

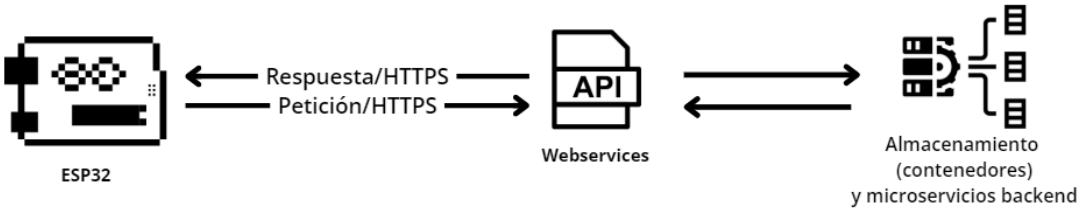

Figura VII.7 Esquema de comunicación endpoints

VII.12 Procesamiento de datos

Cuando el ESP32 envía una nueva lectura de temperatura y humedad, el servidor procesa la solicitud y verifica que los datos recibidos sean válidos antes de almacenarlos. El siguiente código muestra un ejemplo de un endpoint POST que recibe los datos de los sensores.

```
// Endpoint para recibir datos de sensores
app.post('/api/data', (req, res) => {
    const { temperature, humidity } = req.body;

    // Validar datos recibidos
    if (temperature === undefined || humidity === undefined) {
        return res.status(400).send('Datos incompletos');
    }

    const timestamp = new Date();

    // Guardar datos en MongoDB
    db.collection('readings').insertOne({
        temperature,
        humidity,
        timestamp
    }, (err, result) => {
        if (err) return res.status(500).send('Error al guardar los datos');
        res.status(200).send('Datos guardados exitosamente');
    });
});
```

Figura VII.8 Solicitud de tipo POST

Se muestra cómo se recibe una solicitud POST con los datos del sensor y se almacenan en MongoDB. La validación básica garantiza que se reciban tanto la temperatura como la humedad antes de guardar los datos.

VII.13 Almacenamiento de datos MongoDB

El almacenamiento de datos se realiza en MongoDB, una base de datos NoSQL que permite almacenar los datos de manera eficiente en documentos JSON. Cada lectura de los sensores se guarda como un documento que incluye los campos de temperatura, humedad, y un timestamp que registra el momento exacto en que se tomó la medición.

La estructura de los documentos almacenados en MongoDB es la siguiente:

```json
{
    "temperature": 24.5,
    "humidity": 60,
    "timestamp": "2024-10-08T10:30:00.000Z"
}
```

Figura VII.9 Estructura de objeto de datos

Esta estructura permite consultas rápidas y eficientes sobre los datos, tanto para visualización en tiempo real como para el análisis histórico.

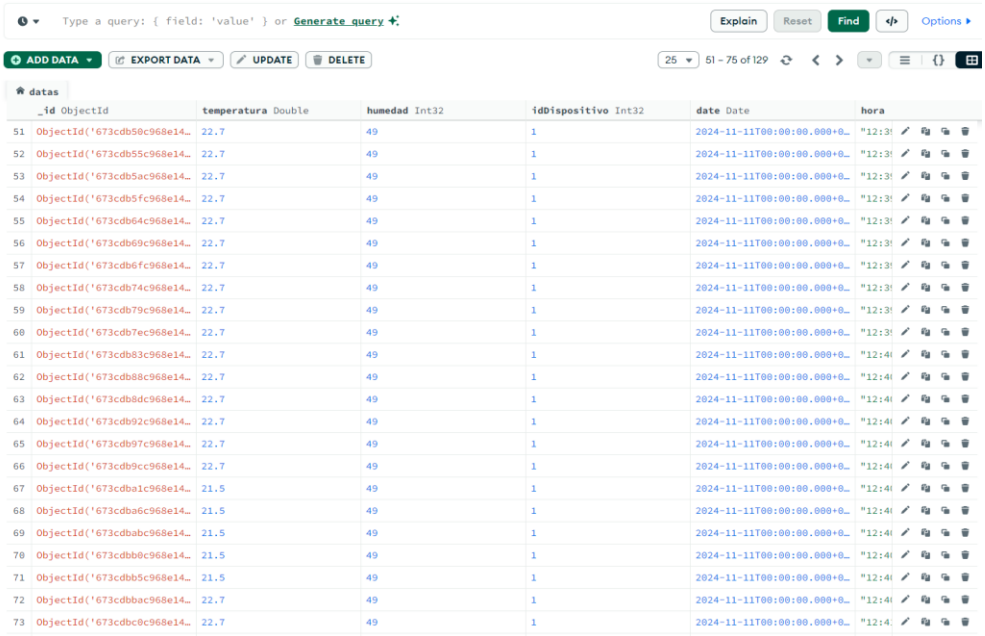

Figura VII.10 Histórico de datos recopilados

VII.14 Endpoints para consultar los datos

El sistema también ofrece endpoints que permiten consultar los datos almacenados en MongoDB. Estos endpoints son utilizados por la interfaz web para mostrar las mediciones en tiempo real y para acceder a los datos históricos. A continuación, se presenta un ejemplo de un endpoint GET que recupera las últimas lecturas de la base de datos.

```javascript
// Endpoint para obtener las últimas lecturas de los sensores
app.get('/api/data/latest', (req, res) => {
    db.collection('readings').find().sort({ timestamp: -1 }).limit(1).toArray((err
        if (err) return res.status(500).send('Error al recuperar los datos');
        res.status(200).json(data[0]);
    });
});

// Endpoint para obtener lecturas históricas entre dos fechas
app.get('/api/data', (req, res) => {
    const { start, end } = req.query;

    db.collection('readings').find({
        timestamp: {
            $gte: new Date(start),
            $lt: new Date(end)
        }
    }).sort({ timestamp: 1 }).toArray((err, data) => {
        if (err) return res.status(500).send('Error al recuperar los datos');
        res.status(200).json(data);
```

Figura VII.11 Solicitud GET para obtención de las lecturas datos históricos

Las últimas lecturas de los sensores para ser mostradas en la interfaz en tiempo real.

Datos históricos dentro de un rango de fechas, que pueden ser utilizados para analizar las tendencias de temperatura y humedad a lo largo del tiempo.

VII.15 Escalabilidad y despliegue para la aplicación en contenedores

Para asegurar que el sistema pueda escalar fácilmente y adaptarse a futuras necesidades, se ha implementado en un entorno de contenedores Docker. Esto permite aislar el entorno de ejecución, facilitando el despliegue tanto en entornos de prueba como de producción sin depender de la infraestructura subyacente Dockerfile para el back-end:

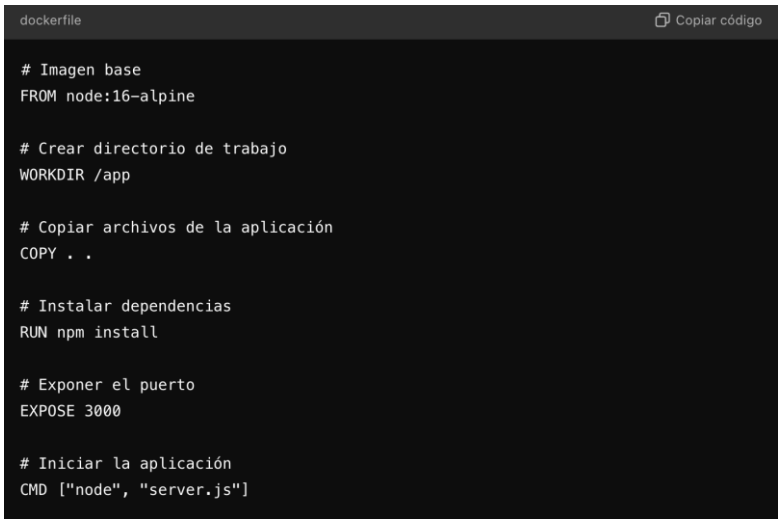

Figura VII.12 Estructura del archivo Docker file

Este contenedor encapsula el servidor Node.js y lo prepara para su despliegue, asegurando que el sistema pueda ejecutarse en cualquier máquina o infraestructura que soporte Docker.

VII.16 Pruebas y validación

Para validar el sistema, se realizaron varias pruebas de funcionamiento, incluyendo pruebas de estrés para evaluar la estabilidad del sistema bajo condiciones extremas. Durante estas pruebas, se observó que el sistema es capaz de manejar hasta 1000 lecturas por hora sin pérdida de datos.

Se llevaron a cabo las siguientes pruebas:

- Pruebas de precisión: Se compararon las lecturas del sensor DHT11 con un sensor de mayor precisión (DHT22) y se observó una variación promedio de 1.8°C en temperatura y 4% en humedad.
- Pruebas de conectividad: Se realizaron pruebas de conexión WiFi con diferentes configuraciones de red para evaluar la estabilidad de la transmisión de datos.
- Pruebas de carga: Se simuló el funcionamiento con múltiples sensores transmitiendo datos simultáneamente para evaluar el rendimiento del servidor y la base de datos.

- Pruebas de latencia: Se midieron los tiempos de respuesta desde la captura del dato hasta su visualización en la interfaz web, obteniendo un promedio de 1.2 segundos.

En la siguiente imagen su muestra la lectura de la temperatura y humedad del site:

Figura VII.13 Lectura de temperatura y humedad

VII.17 Seguridad del sistema

La seguridad en sistemas IoT es un aspecto fundamental. Se tomaron las siguientes medidas para proteger los datos transmitidos y almacenados:

- Cifrado de datos: Se implementó TLS para asegurar la comunicación MQTT y HTTPS en la interfaz web.
- Autenticación de usuarios: Se utilizó JWT (JSON Web Tokens) para gestionar el acceso seguro a la API.
- Protección contra ataques: Se implementaron firewalls y listas de control de acceso (ACL) en el servidor para evitar intrusiones no autorizadas.

VIII. RESULTADOS Y DISCUSIÓN

VIII.1 Resultados de las pruebas

Para evaluar el desempeño del sistema de monitoreo de temperatura y humedad, se realizaron pruebas en diferentes condiciones operativas. A continuación, se presentan los principales resultados obtenidos:

Datos de mediciones

Los datos recopilados por el sistema fueron los siguientes:

- Temperatura registrada: 23°C
- Humedad registrada: 50%
- ID del dispositivo: 1
- Fecha y hora de la muestra: 11 de noviembre de 2024, 12:45:47

En la siguiente imagen se muestra el objeto de datos recopilado de las lecturas:

Figura VIII.1 Estructura de objeto de datos temperatura y humedad

Estos valores se encuentran dentro del rango esperado para un site bien climatizado, lo que confirma que el sistema es capaz de capturar datos confiables en tiempo real.

VIII.2 Tiempos de respuesta

Se evaluaron los tiempos de ejecución del sistema mediante herramientas de monitoreo y perfilado en MongoDB y Google PageSpeed Insights:

- Tiempo de consulta en MongoDB: 2 segundos. Se utilizó el MongoDB Profiler, el cual permite registrar todas las operaciones realizadas en la base de datos y medir

su tiempo de ejecución. También se empleó el comando explain("executionStats") para analizar el rendimiento de consultas específicas.

En la siguiente imagen se muestran los comandos de ejecución para el monitoreo de los datos:

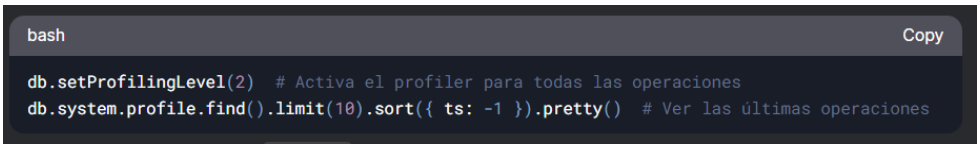

Figura VIII.2 Comandos para la ejecución del monitoreo de datos mongodb

- Tiempo de visualización en la interfaz web: Se utilizaron métricas de Google PageSpeed Insights para evaluar la velocidad de carga:
    - First Contentful Paint (FCP) - 2.7 segundos: Indica el tiempo en el que se renderiza el primer elemento visible en la página. Un menor tiempo de FCP mejora la percepción de velocidad por parte del usuario.
    - Renderizado del mayor elemento con contenido (LCP) - 4.6 segundos: Mide el tiempo en el que el elemento de mayor tamaño dentro de la vista se carga completamente. Es un indicador clave de rendimiento en la experiencia del usuario.
    - Speed Index - 5.1 segundos: Evalúa la rapidez con la que el contenido se visualiza en la pantalla. Un menor índice implica una carga más rápida y fluida.
    - Tiempo de respuesta inicial del servidor - 50 ms: Indica el tiempo que tarda el servidor en responder a la primera solicitud del navegador. Un valor bajo significa una infraestructura optimizada y con buen rendimiento.

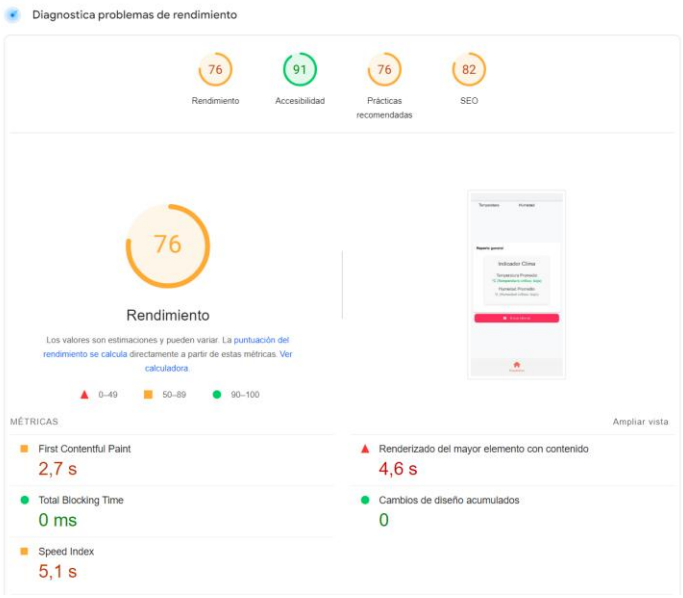

Figura VIII.3 Pruebas de rendimiento del sistema

Estos resultados indican que el sistema tiene un rendimiento aceptable, aunque con margen de optimización en la carga de la interfaz web, especialmente en la reducción de recursos innecesarios en JavaScript y CSS. Se recomienda mejorar la compresión de archivos, minimizar las solicitudes HTTP y optimizar la estrategia de almacenamiento en caché para reducir los tiempos de carga.

VIII.3 Análisis de desempeño

El análisis de desempeño del sistema de monitoreo de temperatura y humedad se centró en evaluar la fiabilidad, estabilidad y eficiencia del sistema en diferentes condiciones operativas. A continuación, se desglosan los aspectos clave:

1. Fiabilidad del monitoreo:
    - Consistencia de los datos: Los datos capturados por el sensor DHT11 mostraron una consistencia razonable dentro de los rangos esperados para un site climatizado. Aunque el sensor tiene una precisión limitada (±2°C en temperatura y ±5% en humedad), las mediciones se mantuvieron dentro de los parámetros aceptables para un entorno no crítico.
    - Validación de datos: Durante las pruebas, se compararon las lecturas del DHT11 con un sensor de mayor precisión (DHT22), observándose una variación promedio de 1.8°C en temperatura y 4% en humedad. Esto

confirma que, aunque el DHT11 no es el sensor más preciso, es adecuado para aplicaciones básicas de monitoreo ambiental.

2. Estabilidad de la transmisión:
    - Conectividad WiFi: El sistema demostró una estabilidad aceptable en la transmisión de datos a través de la red WiFi. Sin embargo, se identificó que la estabilidad de la red es un factor crítico. En caso de interrupciones en la conectividad, el sistema deja de transmitir datos, lo que podría ser un problema en entornos con redes inestables.
    - Protocolo MQTT: El uso del protocolo MQTT permitió una transmisión eficiente de datos, incluso en redes con ancho de banda limitado. El tiempo de latencia promedio desde la captura del dato hasta su visualización en la interfaz web fue de 1.2 segundos, lo que es aceptable para un sistema de monitoreo en tiempo real.
3. Carga y eficiencia del servidor:
    - Tiempos de consulta en MongoDB: Las consultas a la base de datos MongoDB mostraron un tiempo de respuesta promedio de 2 segundos, lo que es adecuado para la mayoría de las aplicaciones. Sin embargo, en entornos con grandes volúmenes de datos, este tiempo podría aumentar, lo que sugiere la necesidad de optimizar las consultas o escalar la infraestructura de la base de datos.
    - Rendimiento de la interfaz web: Las métricas de Google PageSpeed Insights revelaron que la interfaz web tiene un tiempo de carga inicial (First Contentful Paint) de 2.7 segundos y un tiempo de renderizado del mayor elemento con contenido (LCP) de 4.6 segundos. Aunque estos valores son aceptables, se identificaron oportunidades de mejora, como la reducción del uso de archivos CSS y JavaScript innecesarios, y la implementación de almacenamiento en caché para recursos estáticos.
4. Escalabilidad:
    - Capacidad de manejo de datos: El sistema demostró ser capaz de manejar hasta 1000 lecturas por hora sin pérdida de datos, lo que es suficiente para un entorno con un solo sensor. Sin embargo, en caso de expandir el sistema

para monitorear múltiples puntos dentro del site, se requeriría una arquitectura más robusta, como la implementación de un sistema de colas (por ejemplo, RabbitMQ) para gestionar la transmisión de datos desde múltiples dispositivos.

VIII.4 Impacto del sistema y prototipo

El sistema de monitoreo de temperatura y humedad ha tenido un impacto significativo en la gestión ambiental del site de la Universidad Autónoma del Estado de Hidalgo. A continuación, se detallan los principales beneficios y mejoras observadas:

1. Mejora en la eficiencia del monitoreo:
    - Reducción de revisiones manuales: Antes de la implementación del sistema, el monitoreo ambiental dependía de revisiones manuales esporádicas, lo que limitaba la capacidad de detectar cambios bruscos en las condiciones ambientales. Con el sistema IoT, el monitoreo se realiza de manera continua y en tiempo real, lo que permite una supervisión más eficiente y proactiva.
    - Automatización de procesos: La automatización del monitoreo ha liberado recursos humanos que antes se dedicaban a tareas manuales, permitiendo que el personal se enfoque en actividades más estratégicas.
2. Reducción del riesgo de fallos:
    - Detección temprana de anomalías: El sistema permite detectar cambios bruscos en la temperatura y humedad, lo que facilita la identificación temprana de posibles problemas en el sistema de climatización. Esto reduce el riesgo de fallos catastróficos que podrían afectar la operatividad de los equipos críticos.
    - Alertas en tiempo real: Aunque el sistema actual no incluye notificaciones automáticas (como SMS o correo electrónico), la visualización en tiempo real de los datos permite a los responsables del site tomar acciones inmediatas en caso de detectar condiciones fuera de los rangos óptimos.
3. Optimización del uso de la infraestructura:
    - Toma de decisiones informadas: Con los datos históricos almacenados en MongoDB, los administradores del site pueden analizar tendencias y

patrones en las condiciones ambientales, lo que facilita la toma de decisiones informadas sobre el mantenimiento preventivo y la optimización del sistema de climatización.

- Ahorro energético: Al mantener las condiciones ambientales dentro de los rangos óptimos, el sistema contribuye a reducir el consumo energético del site, lo que se traduce en ahorros económicos y una menor huella de carbono.

4. Impacto en la continuidad de los servicios:
    - Protección de equipos críticos: Al garantizar que la temperatura y humedad se mantengan dentro de los rangos recomendados por la ASHRAE, el sistema protege los servidores y equipos de red de posibles daños causados por sobrecalentamiento o humedad excesiva. Esto asegura la continuidad de los servicios académicos y administrativos que dependen de la infraestructura tecnológica del site.
    - Mejora en la confiabilidad de la red: Al minimizar los riesgos de fallos en los equipos, el sistema contribuye a mejorar la confiabilidad y estabilidad de la red universitaria, lo que beneficia tanto a estudiantes como al personal administrativo.
5. Posibilidades de expansión:
    - Integración de nuevos sensores: La arquitectura del sistema permite la fácil integración de nuevos sensores, como sensores de presión atmosférica o calidad del aire, lo que ampliaría las capacidades de monitoreo del site.
    - Implementación de alertas avanzadas: En futuras iteraciones, se podría implementar un sistema de notificaciones automáticas (SMS, correo electrónico) para alertar a los responsables en caso de condiciones críticas, lo que mejoraría aún más la capacidad de respuesta del sistema.

VIII.5 Hallazgos durante el desarrollo

Durante el desarrollo del prototipo, se identificaron varios hallazgos importantes:

1. Precisión del sensor DHT11: Aunque el sensor DHT11 es adecuado para aplicaciones básicas, se observó que su precisión es limitada en comparación con sensores más avanzados como el DHT22 o el BME280. Esto podría afectar la fiabilidad del sistema en entornos donde se requiera una mayor precisión.
2. Dependencia de la red WiFi: El sistema depende completamente de la red WiFi para la transmisión de datos. Durante las pruebas, se observó que la estabilidad de la red es crucial para el funcionamiento del sistema. En caso de interrupciones en la red, el sistema deja de transmitir datos, lo que podría ser un problema en entornos con redes inestables.
3. Escalabilidad del sistema: El sistema actual está diseñado para monitorear un solo punto dentro del site. Sin embargo, se identificó que el uso de múltiples sensores y ESP32 podría requerir una arquitectura más robusta, como la implementación de un sistema de colas (por ejemplo, RabbitMQ) para gestionar la transmisión de datos desde múltiples dispositivos.
4. Consumo energético: Aunque el ESP32 tiene un consumo bajo, el monitoreo continuo podría reducir su eficiencia en entornos sin suministro constante de energía. Se recomienda evaluar la integración de baterías recargables o sistemas de energía alternativos.

VIII.6 Comparación con soluciones comerciales

Comparando este sistema con soluciones comerciales como NetBotz 200 o Sensaphone, se pueden destacar las siguientes diferencias:

1. Costo-Beneficio:

- Prototipo: El costo del prototipo es significativamente menor en comparación con los sistemas comerciales, que pueden superar los $500 USD. Esta ventaja económica lo convierte en una opción atractiva para usuarios con presupuestos limitados.
- Soluciones Comerciales: Ofrecen un mayor nivel de sofisticación y funcionalidades, pero su precio refleja la inversión en desarrollo, hardware y soporte técnico.

2. Personalización y Escalabilidad:

- Prototipo: La arquitectura del sistema facilita la integración de nuevos sensores y funcionalidades según las necesidades del usuario. Esto permite adaptar el sistema a diferentes escenarios y aplicaciones.
- Soluciones Comerciales: Por lo general, ofrecen un conjunto predefinido de sensores y funcionalidades, lo que puede limitar la flexibilidad para la personalización.

3. Precisión:

- Prototipo: El sensor DHT11, utilizado en el prototipo, cumple con su función en entornos no críticos. Sin embargo, para aplicaciones que requieren una mayor precisión, sensores más avanzados como el BME280 o SHT31 podrían ser una mejor alternativa.
- Soluciones Comerciales: Emplean sensores de mayor precisión para obtener datos más confiables en entornos que requieren mediciones exactas.

4. Capacidad de Alerta:

- Prototipo: El sistema desarrollado aún no incluye funcionalidades de alerta como notificaciones SMS o correo electrónico. Esta funcionalidad podría implementarse en futuras versiones para mejorar la reactividad ante eventos críticos.
- Soluciones Comerciales: Ofrecen notificaciones SMS y correo electrónico, permitiendo una respuesta rápida ante eventos como cambios de temperatura, humedad o intrusiones.

IX. CONCLUSIONES Y RECOMENDACIONES

El desarrollo e implementación del sistema IoT para el monitoreo en tiempo real de temperatura y humedad en el site de la Universidad Autónoma del Estado de Hidalgo (UAEH) ha demostrado ser una solución viable y efectiva para la gestión ambiental de infraestructuras críticas. A través de la integración de tecnologías como el microcontrolador ESP32, el sensor DHT11, el protocolo MQTT y la base de datos NoSQL MongoDB, se ha logrado crear un sistema que permite la captura, transmisión, almacenamiento y visualización de datos ambientales en tiempo real. Este sistema no solo optimiza el monitoreo de las condiciones ambientales, sino que también contribuye a la prevención de fallos en los equipos tecnológicos, asegurando la continuidad de los servicios académicos y administrativos.

Entre las principales conclusiones del proyecto se encuentran:

1. Funcionalidad del sistema: El sistema desarrollado cumple con los objetivos planteados, permitiendo el monitoreo continuo de la temperatura y humedad en el site de la UAEH. La integración de tecnologías IoT ha demostrado ser eficiente para la transmisión de datos en tiempo real, lo que facilita la toma de decisiones proactivas en caso de detectar condiciones ambientales adversas.

2. Precisión y limitaciones del sensor DHT11: Aunque el sensor DHT11 es adecuado para aplicaciones básicas de monitoreo, se identificó que su precisión es limitada en comparación con sensores más avanzados como el DHT22 o el BME280. Esto sugiere que, en entornos donde se requiera una mayor precisión, sería recomendable considerar la integración de sensores más precisos.

3. Dependencia de la red WiFi: El sistema depende completamente de la red WiFi para la transmisión de datos. Durante las pruebas, se observó que la estabilidad de la red es un factor crítico para el funcionamiento del sistema. En caso de interrupciones en la conectividad, el sistema deja de transmitir datos, lo que podría ser un problema en entornos con redes inestables. Por lo tanto, se recomienda explorar soluciones de respaldo, como la integración de baterías o sistemas de energía ininterrumpida (UPS).

4. Escalabilidad y optimización: El sistema actual está diseñado para monitorear un solo punto dentro del site. Sin embargo, se identificó que la integración de múltiples

sensores y dispositivos podría requerir una arquitectura más robusta, como la implementación de un sistema de colas (por ejemplo, RabbitMQ) para gestionar la transmisión de datos desde múltiples dispositivos. Además, se recomienda optimizar el consumo energético del sistema, especialmente en entornos donde el suministro de energía no sea constante.

5. Impacto en la gestión del site: La implementación del sistema ha permitido una mejora significativa en la eficiencia del monitoreo ambiental, reduciendo la dependencia de revisiones manuales y permitiendo una supervisión continua y en tiempo real. Esto ha contribuido a la detección temprana de anomalías y a la reducción del riesgo de fallos en los equipos críticos.

IX.1 Recomendaciones

Con base en los hallazgos y conclusiones del proyecto, se proponen las siguientes recomendaciones para futuras iteraciones y mejoras del sistema:

1. Integración de sensores más avanzados: Para mejorar la precisión de las mediciones, se recomienda evaluar la integración de sensores más avanzados, como el DHT22 o el BME280, que ofrecen una mayor precisión en la medición de temperatura, humedad y presión atmosférica. Esto sería especialmente útil en entornos donde se requiera un monitoreo más preciso y confiable.

2. Implementación de sistemas de respaldo: Dada la dependencia del sistema de la red WiFi, se recomienda explorar soluciones de respaldo, como la integración de baterías recargables o sistemas de energía ininterrumpida (UPS), para garantizar la continuidad del monitoreo en caso de cortes de energía o fallos en la red.

3. Optimización del consumo energético: Para mejorar la eficiencia energética del sistema, se sugiere explorar la integración de fuentes de energía alternativas, como paneles solares, o la implementación de estrategias de bajo consumo en el ESP32, como el uso de modos de suspensión o hibernación.

4. Expansión del sistema: El sistema actual está diseñado para monitorear un solo punto dentro del site. Sin embargo, se recomienda explorar la expansión del sistema para incluir múltiples sensores y dispositivos, lo que permitiría un monitoreo más completo de las condiciones ambientales en diferentes áreas del site. Para ello, se

podría implementar un sistema de colas (por ejemplo, RabbitMQ) para gestionar la transmisión de datos desde múltiples dispositivos.

5. Implementación de alertas avanzadas: Aunque el sistema actual permite la visualización de datos en tiempo real, se recomienda implementar un sistema de notificaciones automáticas, como alertas por SMS o correo electrónico, para informar a los responsables del site en caso de detectar condiciones críticas. Esto mejoraría la capacidad de respuesta del sistema y permitiría una gestión más proactiva de las condiciones ambientales.

6. Optimización de la interfaz web: Se identificaron oportunidades de mejora en la velocidad de carga de la interfaz web, especialmente en la reducción del uso de archivos CSS y JavaScript innecesarios. Se recomienda implementar estrategias de optimización, como la compresión de archivos, la minimización de solicitudes HTTP y el uso de almacenamiento en caché para recursos estáticos.

7. Integración con inteligencia artificial: En futuras iteraciones, se podría explorar la integración de algoritmos de aprendizaje automático para predecir fallos en el site basados en los datos históricos recopilados. Esto permitiría una gestión más proactiva y preventiva de la infraestructura tecnológica.

IX.2 Proyecciones futuras

El sistema de monitoreo desarrollado tiene un gran potencial para ser expandido y mejorado en futuras iteraciones. Algunas de las proyecciones futuras incluyen:

1. Integración de nuevos sensores: Además de la temperatura y humedad, se podría integrar sensores para monitorear otros parámetros ambientales, como la presión atmosférica, la calidad del aire o los niveles de CO2. Esto permitiría un monitoreo más completo de las condiciones ambientales en el site.

2. Implementación de sistemas de control automático: En el futuro, se podría explorar la integración de sistemas de control automático que permitan ajustar las condiciones ambientales en el site de manera remota. Por ejemplo, se podría implementar un sistema que active o desactive los sistemas de climatización en función de las lecturas de los sensores.

3. Expansión a otros sitios: El sistema desarrollado podría ser implementado en otros sitios de la universidad o en otras instituciones educativas que requieran un

monitoreo ambiental similar. La arquitectura modular del sistema facilita su adaptación a diferentes entornos y necesidades.

4. Integración con sistemas de gestión de infraestructuras (DCIM): En el futuro, se podría explorar la integración del sistema con sistemas de gestión de infraestructuras de centros de datos (DCIM), lo que permitiría una gestión más integral de los recursos tecnológicos y ambientales en el site.

IX.3 Reflexión final

El desarrollo de este sistema de monitoreo IoT ha demostrado que es posible implementar soluciones tecnológicas de bajo costo para mejorar la gestión de infraestructuras críticas en entornos académicos. Aunque el sistema actual tiene algunas limitaciones, como la precisión del sensor y la dependencia de la red WiFi, ha demostrado ser una herramienta valiosa para la prevención de fallos y la optimización del uso de los recursos tecnológicos en el site de la UAEH.

Este proyecto no solo ha contribuido a la mejora de la infraestructura tecnológica de la universidad, sino que también ha abierto la puerta a futuras investigaciones y desarrollos en el campo del IoT y la gestión ambiental de centros de datos. Se espera que este sistema sirva como base para futuras implementaciones y mejoras, contribuyendo así a la creación de entornos más seguros y eficientes para el desarrollo de actividades académicas y administrativas.

X. Referencias

Bonilla, L., & Bonilla, L. (2024, 27 junio). Riesgos de un Data Center en 2024: principales problemas. Data Center Market. https://www.datacentermarket.es/dcm-xl/riesgos-de-un-data-center-principales-problemas/

Chernicoff, D. (2013, 12 junio). When it's raining inside your data center, you have a problem. ZDNET. https://www.zdnet.com/article/when-its-raining-inside-your-datacenter-you-have-a-problem/

SHRAE TC9.9 Data Center Power Equipment Thermal Guidelines and Best Practices. (n.d.). Retrieved April 30, 2022, from https://www.ashrae.org/File%20Library/Technical%20Resources/Bookstore/ASHRAE_TC0909_Power_White_Paper_22_June_2016_REVISED.pdf

Qué es la Internet de las cosas (IoT) y qué son los dispositivos de IoT (2019, 12 marzo). /. https://latam.kaspersky.com/resource-center/definitions/what-is-iot AV Electronics. (2024b, octubre 7).

DHT11 Temperatura y humedad – AV Electronics. https://avelectronics.cc/producto/dht11-temperatura-y-humedad/

ESP32 38 Pines ESP WROOM 32. (s/f). UNIT Electronics. Recuperado el 8 de octubre de 2024, de https://uelectronics.com/producto/esp32-38-pines-esp-wroom32/?srsltid=AfmBOop3NWvtlCqJSeb9pb4YXUTfblNw0gvApb05ajxVntefflVdFaIA Pérez-Tavera, I. H. (2023, 5 enero).

Arduino IDE. https://repository.uaeh.edu.mx/revistas/index.php/prepa4/article/view/10474

HTML5 - MDN Web Docs Glossary: Definitions of Web-related terms | MDN. (2024, 25 julio). MDN Web Docs. https://developer.mozilla.org/en-US/docs/Glossary/HTML5

CSS | MDN. (2024, 6 junio). MDN Web Docs. https://developer.mozilla.org/es/docs/Web/CSS

JavaScript | MDN. (2023, 24 julio). MDN Web Docs. https://developer.mozilla.org/es/docs/Web/JavaScript

Qué es Node.js y por qué deberías usarlo. Kinsta®. https://kinsta.com/es/base-de-conocimiento/que-es-node-js/

Contenedores de Docker | ¿Qué es Docker? | AWS. (s. f.). Amazon Web Services, Inc. https://aws.amazon.com/es/docker/

Introduction to web APIs - Learn web development | MDN. (2024, 5 agosto). MDN Web Docs. https://developer.mozilla.org/en-US/docs/Learn/JavaScript/Client-side_web_APIs/Introduction

Introducción. (s. f.). Vitejs. https://es.vitejs.dev/guide/

MQTT - Explicación del protocolo MQTT - AWS. (s. f.). Amazon Web Services, Inc. https://aws.amazon.com/es/what-is/mqtt/

IBM. (2023, 12 mayo). ¿Qué es MongoDB? | IBM. https://www.ibm.com/mx-es/topics/mongodb

CICS Transaction Server for z/OS 6.x. (s. f.). https://www.ibm.com/docs/es/cics-ts/6.x?topic=cics-json-web-token-jw

Qué es el wifi - Tipos de conexiones wifi y seguridad | Proofpoint ES. (2023, 24 diciembre). Proofpoint. https://www.proofpoint.com/es/threat-reference/wifi#:~:text=Wifi%2C%20que%20es%20una%20contracci%C3%B3n,red%20mediante%20frecuencias%20de%20radio.

Leskow, E. C. (2024, 2 septiembre). Temperatura - Concepto, escalas, medición, tipos y ejemplos. Concepto. https://concepto.de/temperatura/

Equipo editorial, Etecé. (2020, 29 agosto). Humedad - Concepto, tipos, medición, clima y nubes. Concepto. https://concepto.de/humedad/

Martín, A., Chávez, S. B., Rodríguez, N. R., Valenzuela, A., y Murazzo, M. A. (2013). Bases de datos NoSQL en Cloud Computing (p. 166).

Gharavi, H., & Ghafurian, R. (2011). Smart grid: The electric energy system of the future. Proceedings of the IEEE, 99(6), 917-921.

Islam, S. M. R., Kwak, D., Kabir, M. H., Hossain, M., & Kwak, K. S. (2015). The Internet of Things for health care: A comprehensive survey. IEEE Access, 3, 678-708.

Montalvo, R. M. (2020). Cyber risk from IoT technologies in the supply chain. Computers in Industry, 123, 103335.

Wolfert, S., Ge, L., Verdouw, C., & Bogaardt, M. J. (2017). Big data in smart farming–A review. Agricultural Systems, 153, 69-80.

Zanella, A., Bui, N., Castellani, A., Vangelista, L., & Zorzi, M. (2014). Internet of Things for smart cities. IEEE Internet of Things Journal, 1(1), 22-32.

Funcionamiento. (2008, 21 septiembre). TECNOLOGÌA DE COMUNICACIÒN INALÀMBRICA. https://arodriguezr.wordpress.com/como-funciona-lo-inalambrico/

Zhang, Y., Qian, C., & He, Z. (2021). Wireless energy transfer technologies for IoT devices. IEEE Internet of Things Journal, 8(5), 3297-3308.

Gubbi, J., Buyya, R., Marusic, S., & Palaniswami, M. (2013). Internet of Things (IoT): A vision, architectural elements, and future directions. Future Generation Computer Systems, 29(7), 1645-1660.

Huancayo, (2022). Diseño de módulo de control basado en plataforma IoT para monitoreo de motor de inducción de baja potencia. Repositorio Continental. https://repositorio.continental.edu.pe/handle/20.500.12394/12585

Escobar, J., & Sánchez, L. (2022). Implementación de un prototipo de un sistema de monitoreo ambiental de temperatura, humedad, presión y calidad del aire basado en ESP32 y página web. Escuela Politécnica Nacional. https://bibdigital.epn.edu.ec/handle/15000/25599

Zhang, Y., Shan, K., Li, X., Li, H., & Wang, S. (2023). Research and technologies for next-generation high-temperature data centers: State-of-the-art and future perspectives. Renewable and Sustainable Energy Reviews, 171, Article 112991. https://doi.org/10.1016/j.rser.2022.112991

CICS Transaction Server for Z/OS 6.x. (s.f.). Security mechanisms in IoT environments. IBM.

Gharavi, H., & Ghafurian, R. (2021). Smart Grid: The Electric Energy System of the Future. IEEE Xplore.

Zhang, Y., Shan, K., Li, X., Li, H., & Wang, S. (2023). Advancements in IoT-Based Industrial Monitoring Systems.

Estudio Electrónica. (2023). *¿Qué es un microcontrolador?* Estudio Electrónica. https://www.estudioelectronica.com/que-es-un-microcontrolador/

IBM. (2023). ¿Qué es un microcontrolador? IBM. https://www.ibm.com/mxes/think/topics/microcontroller

Universidad Europea. (2023). Microcontrolador: ¿Qué es y para qué sirve? Universidad Europea. https://universidadeuropea.com/blog/que-esmicrocontrolador/

Sherlin.xBot.es. (2023). 1. ¿Qué es un microcontrolador? Sherlin.xBot.es. https://sherlin.xbot.es/microcontroladores/introduccion-a-los microcontroladores/que-es-un-microcontrolador

Amazon. (2023). Placa de desarrollo ESP32 Procesador de microcontrolador de doble .... Amazon.com.mx. https://www.amazon.com.mx/desarrollo-Procesador microcontrolador-Bluetooth-integrado/dp/B07RY9MVCV

Espressif. (2023). *ESP32 Series*. Espressif Systems. https://www.espressif.com/en/products/socs/esp32

Microdesys. (2023). *Microcontrolador Esp32*. Microdesys. https://microdesys.es/docs/microcontrolador-esp32/

Programar Fácil. (2023). *ESP32 Wifi + Bluetooth en un solo lugar*. Programar Fácil. https://programarfacil.com/esp8266/esp32/

Universidad Autónoma del Estado de Hidalgo. (2008). Pone en marcha la UAEH Red Metropolitana de Fibra Óptica y la Red de Microondas WiMax. Garceta UAEH, 61. https://www.uaeh.edu.mx/garceta/n61/nota2.htm

Universidad Autónoma del Estado de Hidalgo. (2016). Programa WorkLabs 2016. https://www.uaeh.edu.mx/dlcyt/documentos/programa_worklabs_2016.pdf

Universidad Autónoma del Estado de Hidalgo. (2025). Manual de organización de la Dirección de Información y Sistemas. https://www.uaeh.edu.mx/planeacion/sistemas/images/documentosAdministrativos/2025/mo-diys-v6.pdf

Chen, J., Yang, A., & Chen, S. (2021). IEEE Internet of Things Journal, 8(4), 2634-2652. https://doi.org/10.1109/JIOT.2020.3026400

Farooq, M. S., Riaz, S., Abid, A., Abid, K., & Naeem, M. A. (2023). Computers and Electronics in Agriculture, 204, 107541. https://doi.org/10.1016/j.compag.2022.107541

Hasan, M., Islam, M. M., Zarif, M. I. I., & Hashem, M. M. A. (2023). IEEE Access, 11, 1833-1862. https://doi.org/10.1109/ACCESS.2022.3233315

Rahman, M. A., Hossain, M. S., Alrajeh, N. A., & Gupta, B. B. (2022). IEEE Transactions on Industrial Informatics, 18(5), 3511-3521. https://doi.org/10.1109/TII.2021.3105720

Rayes, A., & Salam, S. (2022). Internet of Things from Hype to Reality (3rd ed.). Springer. https://doi.org/10.1007/978-3-030-90158-5

Xu, X., Lu, Y., Vogel-Heuser, B., & Wang, L. (2023). *IEEE/CAA Journal of

Automatica Sinica, 10*(1), 1-17. https://doi.org/10.1109/JAS.2022.106061

XI. Anexos

XI.1 Código del microcontrolador ESP32

```
#include <ESP8266WiFi.h>
#include <ESP8266HTTPClient.h>
#include <DHT.h>
// --- Configuración WiFi ---
const char* ssid = "TU_SSID";       // Nombre de tu red WiFi
const char* password = "TU_PASSWORD"; // Contraseña de tu WiFi
// --- Configuración del DHT11 ---
#define DHTPIN D4       // Pin de datos del DHT11 (ajusta según tu conexión)
#define DHTTYPE DHT11
DHT dht(DHTPIN, DHTTYPE);
// --- Configuración de la URL destino ---
String serverName = "http://TU-SERVIDOR.com/api/datos";  // tu endpoint
void setup() {
  Serial.begin(115200);
  dht.begin();
  // Conexión WiFi
  WiFi.begin(ssid, password);
  Serial.print("Conectando a WiFi...");
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("\nConectado a WiFi!");
}
void loop() {
  if (WiFi.status() == WL_CONNECTED) {
    float humedad = dht.readHumidity();
    float temperatura = dht.readTemperature();
```

```
    if (isnan(humedad) || isnan(temperatura)) {
      Serial.println("Error al leer el sensor DHT11!");
      return;
    }
    // Crear el JSON o query para enviar
    String postData = "temperatura=" + String(temperatura) + "&humedad=" + String(humedad);
    // --- Enviar datos por HTTP POST ---
    HTTPClient http;
    http.begin(serverName);
    http.addHeader("Content-Type", "application/x-www-form-urlencoded");
    int httpResponseCode = http.POST(postData);
    if (httpResponseCode > 0) {
      String respuesta = http.getString();
      Serial.println("Respuesta del servidor: " + respuesta);
    } else {
      Serial.println("Error en la petición: " + String(httpResponseCode));
    }
    http.end();
  } else {
    Serial.println("WiFi desconectado!");
  }
  delay(10000); // cada 10 segundos
}
```

XI.2 Código del contendor para la API y la aplicación

```
const bodyParser = require('body-parser');
const express = require('express');
const mongoose = require('mongoose');
const cors = require('cors');
const dayjs = require('dayjs');
```

```
const axios = require('axios');
const moment = require('moment-timezone');
const nodemailer = require('nodemailer');
//configuracion a mongo
const app = express()
app.use(cors());
app.use(bodyParser.json());
// Conexión a MongoDB
mongoose.connect(uri)
  .then(() => console.log('Conectado a MongoDB'))
  .catch(err => console.error('Error al conectar a MongoDB:', err));
// Definición del esquema y modelo
const fecha = dayjs();
const SensorSchema = new mongoose.Schema({
    temperatura: Number,
    humedad: Number,
    idDispositivo: Number,
    date: Date,
    hora: String
});
const SensorData = mongoose.model('data', SensorSchema); // Nombre de la colección
const UsuarioSchema = new mongoose.Schema({
    nombre: String,
    apellidoPaterno: String,
    apellidoMaterno: String,
    correo: String,
    contrasena: String,
    date: Date,
    hora: String
});
const UsuarioData = mongoose.model('usuarios', UsuarioSchema); // Nombre de la colección
```

```
const SensorInventarioSchema = new mongoose.Schema({
    nombre_dispositivo: String,
    numero_serie: String,
    versionDispositivo: String,
    idDispositivo: Number,
    date: Date,
    hora: String
});
const SensorInventarioData = mongoose.model('inventario', SensorInventarioSchema); // Nombre de la colección


const usuarioDispositivoSchema = new mongoose.Schema({
    idUsuario: String,
    numero_serie: String,
    nombre_dispositiv_ubicacion: String,
    nombre_dispositivo: String,
    idDispositivo: Number,
    date: Date,
    hora: String
});
const UsuarioDispositivoData = mongoose.model('usuarioDispositivo', usuarioDispositivoSchema); // Nombre de la colección
app.post('/api/data', async (req, res) => {
    try {
        const moment = require('moment-timezone');
        // Obtener la fecha y hora en la zona horaria específica
        const fechaConZonaHoraria = moment.tz(new Date(), 'America/Mexico_City');
        // Extraer las horas, minutos y segundos
        const horas = fechaConZonaHoraria.format('HH');
        const minutos = fechaConZonaHoraria.format('mm');
        const segundos = fechaConZonaHoraria.format('ss');
        // Formatear la hora como "HH:mm:ss"
```

```
        const horaFormateada = `${horas}:${minutos}:${segundos}`;


        const { temperatura, humedad, idDispositivo } = req.body;
        const dataDataCenter = new SensorData({ temperatura, humedad, idDispositivo, date: fecha.format("YYYY-MM-DD"), hora: horaFormateada });
        await dataDataCenter.save();
        res.status(201).json({ message: 'Datos guardados correctamente',response:201});
    } catch (error) {
    }
});
app.post('/api/usuarios', async (req, res) => {
    try {
        const { nombre, apellidoPaterno, apellidoMaterno, correo, contrasena } = req.body;
        const dataUsuario = new UsuarioData({ nombre, apellidoPaterno, apellidoMaterno, correo, contrasena, date: fecha.format("YYYY-MM-DD"), hora: fecha.format("HH:mm:ss") });
        await dataUsuario.save();
        res.status(201).json({ message: 'Usuario guardado correctamente',response:201});
    } catch (error) {
    }
});
app.post('/api/usuarioConsularDatos', async (req, res) => {
    try {
        const { idUsuario } = req.body;
        const user = await UsuarioData.findById(idUsuario);
        res.status(200).json({ user });
    } catch (error) {
        res.status(500).json({ error: 'Error al procesar la solicitud', message: 'error', status: 500 });
        console.log(error);
    }
});
app.post('/api/usuarioActualizarDatos', async (req, res) => {
```

```
    try {
        const {idUsuario ,nombre, apellidoPaterno, apellidoMaterno, correo} = req.body;
        const user = await UsuarioData.findById(idUsuario);
        user.nombre = nombre;
        user.correo = correo;
        user.apellidoPaterno = apellidoPaterno;
        user.apellidoMaterno = apellidoMaterno;
        user.save();
        res.status(200).json({ message: 'Datos actualizados correctamente', status: 200 });
    } catch (error) {
        res.status(500).json({ error: 'Error al procesar la solicitud', message: 'error', status: 500
});
    }
});
app.post('/api/inventario', async (req, res) => {
    try {


        const moment = require('moment-timezone');
        // Obtener la fecha y hora en la zona horaria específica
        const fechaConZonaHoraria = moment.tz(new Date(), 'America/Mexico_City');
        // Extraer las horas, minutos y segundos
        const horas = fechaConZonaHoraria.format('HH');
        const minutos = fechaConZonaHoraria.format('mm');
        const segundos = fechaConZonaHoraria.format('ss');
        // Formatear la hora como "HH:mm:ss"
        const horaFormateada = `${horas}:${minutos}:${segundos}`;
        const { nombre_dispositivo, numero_serie, versionDispositivo, idDispositivo } =
req.body;
        const sensorInventarioData = new SensorInventarioData({ nombre_dispositivo,
numero_serie, versionDispositivo, idDispositivo, date: fecha.format("YYYY-MM-DD"), hora:
horaFormateada });
        await sensorInventarioData.save();
        res.status(201).json({ message: 'Datos guardados correctamente' });
```

```
    } catch (error) {
        res.status(500).json({ error: 'Error al guardar los datos' });
        console.log(error);
    }
});
// Ruta para el login
app.post('/api/login', async (req, res) => {
    try {
        const { correo, contrasena } = req.body;
        const user = await UsuarioData.findOne({ correo, contrasena });


        if (user) {
            res.status(200).json({ message: 'success', nombre: user.nombre, idUsuario: user._id });
        } else {
            res.status(401).json({ error: 'Usuario o contraseña incorrectos', message: 'error', status: 401 });
        }
    } catch (error) {
        res.status(500).json({ error: 'Error al procesar la solicitud' });
        console.log(error);
    }
});
app.post('/api/usuarioDispositivo', async (req, res) => {
    try {
        const moment = require('moment-timezone');
        // Obtener la fecha y hora en la zona horaria específica
        const fechaConZonaHoraria = moment.tz(new Date(), 'America/Mexico_City');
        // Extraer las horas, minutos y segundos
        const horas = fechaConZonaHoraria.format('HH');
        const minutos = fechaConZonaHoraria.format('mm');
        const segundos = fechaConZonaHoraria.format('ss');
```

```
        // Formatear la hora como "HH:mm:ss"
        const horaFormateada = `${horas}:${minutos}:${segundos}`;
        const { idUsuario, numero_serie,nombre_dispositiv_ubicacion } = req.body;
        const dispositivo = await SensorInventarioData.findOne({ numero_serie });
        let nombre_dispositivo = dispositivo.nombre_dispositivo;
        let idDispositivo = dispositivo.idDispositivo;
        const usuarioDispositivo = new UsuarioDispositivoData({ idUsuario, numero_serie,nombre_dispositiv_ubicacion, nombre_dispositivo,idDispositivo, date: fecha.format("YYYY-MM-DD"), hora: horaFormateada });
        await usuarioDispositivo.save();
        res.status(201).json({ message: 'Datos guardados correctamente', status: 201 });
    } catch (error) {
        res.status(500).json({ error: 'Error al guardar los datos', message: 'error', status: 500 });
        console.log(error);
    }
});
app.post('/api/usuarioDispositivoData', async (req, res) => {
    try {
        const { idUsuario } = req.body;
        const dispositivos = await UsuarioDispositivoData.find({ idUsuario });
        res.status(200).json({ dispositivos });
    } catch (error) {
        res.status(500).json({ error: 'Error al procesar la solicitud', message: 'error', status: 500 });
        console.log(error);
    }
});
app.post('/api/sensorData', async (req, res) => {
    try {
        const { idDispositivo } = req.body;
        //const sensorData = await SensorData.find({ idDispositivo });
        //const sensorData = await SensorData.findOne({ idDispositivo }).sort({ _id: -1 });
```

```
        let numero_serie = idDispositivo;
        const dispositivo = await SensorInventarioData.findOne({ numero_serie });
        let dataSensorDashboard = {
            sensorData:await SensorData.findOne({ idDispositivo }).sort({ _id: -1 }),
            dispositivo:dispositivo
        }
        res.status(200).json({ dataSensorDashboard });
    } catch (error) {
        res.status(500).json({ error: 'Error al procesar la solicitud', message: 'error', status: 500
});
        console.log(error);
    }
});
app.post('/api/condicionDataCenter', async (req, res) => {
    try {
        const { idDispositivo } = req.body;
        const sensorData = await SensorData.findOne({ idDispositivo }).sort({ _id: -1 })
        // Determinar si la temperatura y la humedad son adecuadas
        let indicador = 'En condiciones óptimas';
        if (sensorData.temperatura < 18 || sensorData.temperatura > 27) {
            indicador = 'Temperatura fuera del rango óptimo';
        } else if (sensorData.humedad < 40 || sensorData.humedad > 60) {
            indicador = 'Humedad fuera del rango óptimo';
        }
        // Incluir el indicador en la respuesta JSON
        res.status(200).json({ sensorData, indicador });
        //res.status(200).json({ sensorData });
    } catch (error) {
        res.status(500).json({ error: 'Error al procesar la solicitud', message: 'error', status: 500
});
        console.log(error);
    }
```

```
});
app.post('/api/consultarDatosMes', async (req, res) => {
    try {
        const { idDispositivo} = req.body;
        const datosPorMes = await SensorData.aggregate([
            { $match: { idDispositivo } }, // Filtrar por idDispositivo
            {
                $group: {
                    _id: {
                        year: { $year: "$date" },
                        month: { $month: "$date" }
                    },
                    totalDatos: { $sum: 1 },
                    sumaTemperatura: { $sum: "$temperatura" },
                    sumaHumedad: { $sum: "$humedad" }
                }
            },
            {
                $addFields: {
                    monthName: {
                        $arrayElemAt: [
                            [
                                "", "January", "February", "March", "April", "May", "June",
                                "July", "August", "September", "October", "November", "December"
                            ],
                            "$_id.month"
                        ]
                    }
                }
            },
            {
```

```
            $project: {
                _id: 0,
                year: "$_id.year",
                month: "$monthName",
                promedioTemperatura: { $cond: [{ $eq: ["$totalDatos", 0] }, 0, { $divide: ["$sumaTemperatura", "$totalDatos"] }] },
                promedioHumedad: { $cond: [{ $eq: ["$totalDatos", 0] }, 0, { $divide: ["$sumaHumedad", "$totalDatos"] }] }
            }
        },
        { $sort: { year: 1, "_id.month": 1 } } // Ordenar por año y mes
    ]);
    res.status(200).json({ datosPorMes });
  } catch (error) {
    res.status(500).json({ error: 'Error al procesar la solicitud', message: 'error', status: 500 });
    console.log(error);
  }
});



app.post('/api/email', async (req, res) => {
  try {
    const { correo, asunto, mensaje } = req.body;
    // Configura el transporte
    let transporter = nodemailer.createTransport({
      host: 'smtp.gmail.com', // Servidor SMTP de Gmail
      port: 465,              // Puerto para SSL
      secure: true,
      auth: {
        user: '@uaeh.edu.mx', // tu correo
        pass: ''        // tu contraseña de correo
      }
```

```
        });
        // Define los detalles del correo
        let mailOptions = {
            from: 'victor_martinez@uaeh.edu.mx',       // Remitente
            to: correo,   // Destinatario
            subject: asunto,     // Asunto
            //text: mensaje, // Contenido en texto
            html: '<p>'+mensaje+'</p>' // Opcional: contenido HTML
        };
        // Envía el correo
        transporter.sendMail(mailOptions, (error, info) => {
            if (error) {
            return console.log('Error al enviar el correo:', error);
            }
            console.log('Correo enviado:', info.response);
        });
        res.status(200).json({ message: 'Correo enviado correctamente', status: 200 });
    } catch (error) {
        res.status(500).json({ error: 'Error al enviar el correo', message: 'error', status: 500 });
        console.log(error);
    }
})
// Iniciar el servidor
const PORT = process.env.PORT || 5003;
app.listen(PORT, () => console.log(`Servidor corriendo en http://localhost:${PORT}`));
```

GLOSARIO

IoT: Internet de las Cosas (Internet of Things).

UAEH: Universidad Autónoma del Estado de Hidalgo.

MQTT: Message Queuing Telemetry Transport (Protocolo de mensajería para IoT).

NoSQL: Not Only SQL (Bases de datos no relacionales).

ESP32: Microcontrolador con conectividad WiFi y Bluetooth.

DHT11: Sensor digital de temperatura y humedad.

API: Interfaz de Programación de Aplicaciones (Application Programming Interface).

HTML5: Lenguaje de marcado para la estructuración de contenido web.

CSS3: Hojas de estilo en cascada para el diseño web.

JavaScript: Lenguaje de programación para interactividad en páginas web.

Node.js: Entorno de ejecución de JavaScript en el servidor.

MongoDB: Base de datos NoSQL orientada a documentos.

JWT: JSON Web Token (Token de autenticación basado en JSON).

ASHRAE: American Society of Heating, Refrigerating and Air-Conditioning Engineers.

DCIM: Data Center Infrastructure Management (Gestión de infraestructura de centros de datos).

WiFi: Wireless Fidelity (Tecnología de red inalámbrica).

HTTP: Protocolo de Transferencia de Hipertexto (Hypertext Transfer Protocol).

CoAP: Constrained Application Protocol (Protocolo de aplicación para dispositivos limitados).

Zigbee: Protocolo de comunicación inalámbrica para IoT.

LoRaWAN: Long Range Wide Area Network (Red de área amplia de largo alcance).

5G: Quinta generación de tecnología de redes móviles.

UPS: Sistema de alimentación ininterrumpida (Uninterruptible Power Supply)