DISEÑO DE UN PROTOTIPO IoT DIDÁCTICO PARA EL ANÁLISIS Y VISUALIZACIÓN DE DATOS EN TIEMPO REAL EN ENTORNOS EDUCATIVOS

DESIGN OF A DIDACTIC IoT PROTOTYPE FOR REAL-TIME DATA ANALYSIS AND VISUALIZATION IN EDUCATIONAL ENVIRONMENTS

Javier Silvestre Zavala
Tecnológico Nacional de México / ITS de Irapuato, México
javier.sz1@irapuato.tecnm.mx

Carlos Federico Hernández Farfán
Tecnológico Nacional de México / ITS de Irapuato, México
carlos.hf@irapuato.tecnm.mx

Recepción: 21/noviembre/2025 **Aceptación:** 24/diciembre/2025

Resumen

El IoT es un eje fundamental en áreas como la automatización industrial, las ciudades inteligentes, la salud, la agricultura y más, por lo que debe integrarse en la formación de los ingenieros del país. Este artículo presenta el desarrollo de un prototipo didáctico IoT diseñado para apoyar la enseñanza de sistemas embebidos. La metodología consistió en el desarrollo de un módulo de hardware basado en ESP32 para recolectar datos de sensores y transmitirlos vía MQTT a un módulo de software que incluye un servidor local programado en Node.js. El servidor permite la visualización en tiempo real de los datos mediante WebSockets en una página web y su almacenamiento en una base de datos para su análisis estadístico mediante Python. Las pruebas realizadas demuestran su funcionalidad, estabilidad y precisión. Se espera que su futura aplicación en el aula promueva aprendizajes significativos y una comprensión integral de sistemas IoT.

Palabras Clave: Didáctico, IoT, prototipo, websockets.

Abstract

IoT is a key pillar in areas such as industrial automation, smart cities, healthcare, agriculture, and more; therefore, it must be integrated into the education of engineers

in the country. This article presents the development of a didactic IoT prototype designed to support the teaching of embedded systems. The methodology consisted of developing a hardware module based on the ESP32 to collect sensor data and transmit it via MQTT to a software module that includes a local server programmed in Node.js. The server enables real-time data visualization through WebSockets on a web page and stores the data in a database for statistical analysis using Python. The tests carried out demonstrate its functionality, stability, and accuracy. Its future implementation in the classroom is expected to promote meaningful learning and a comprehensive understanding of IoT systems.

Keywords: *Didactic, IoT, prototype, websockets.*

1. Introducción

Las tecnologías emergentes como Internet de las cosas IoT, MQTT, Node.js y websocktes son fundamentales para la enseñanza de sistemas embebidos en ingeniería porque permiten que los estudiantes desarrollen competencias actuales, aplicables y alineadas con las demandas de la industria. En este sentido, como lo establecen [Montaño, 2023], la innovación en la educación, fomentada por las nuevas tecnologías, se ha transformado en la vía para alcanzar la calidad y eficiencia educativa. La enseñanza de sistemas embebidos sin contemplar su conexión a internet, sensores remotos y servicios en la nube generaría una formación incompleta frente a los desafíos del mercado laboral.

Por otro lado, la integración de un protocolo de comunicación ligero y eficiente como MQTT a los proyectos de IoT, permite enseñar conceptos clave como comunicación cliente-servidor, eficiencia energética y latencia, todos esenciales en sistemas embebidos. La combinación de estos proyectos con Node.JS promueve la integración de hardware y sensores, redes y protocolos de comunicación, programación backend y visualización de datos en tiempo real. Y, si además se incluye un análisis estadístico con herramientas basadas en Python, el escenario está completo para generar proyectos robustos y ajustados a las demandas actuales.

Como lo establecen [Rodríguez, 2002], Python es una excelente alternativa para el análisis y visualización de los datos, así como la computación exploratoria e interactiva, convirtiéndolo en una alternativa sólida para las tareas de manipulación de información. Debido a lo anterior, es de vital importancia que los estudiantes de ingeniería incorporen a su formación estas herramientas que les permitirán desarrollar sistemas conectados, eficientes y escalables, y que, además, son fundamentales en la industria 4.0, la automatización, el análisis de datos en tiempo real y la innovación tecnológica.

En los tecnológicos del país dentro del plan de estudios de la carrera de Ingeniería en Sistemas Computacionales [TecNM, s.f.] existe la asignatura denominada Sistemas Programables que tiene como competencia específica aplicar microcontroladores en el diseño de interfaces hombre-máquina y máquina-máquina de sistemas programables. El temario se enfoca en sensores, actuadores, microcontroladores, buses de comunicación y diseño de interfaces. Sin embargo, es necesario complementarlo con tecnologías emergentes como IoT, MQTT, Node.js y Python para llevar esos contenidos a un entorno real y conectado, donde los sistemas embebidos ya no operan de forma aislada, sino como dispositivos inteligentes en red.

La combinación de estas tecnologías está generando un ecosistema digital cada vez más complejo y dinámico, que plantea tanto oportunidades como desafíos para la ingeniería y la sociedad en general [Endara, 2023]. Existen diversos trabajos en el área de prototipos didácticos que hacen uso de tecnologías emergentes. Por ejemplo, [Villalobos, 2021], desarrollaron un prototipo didáctico de plano inclinado como alternativa de enseñanza centrándose en el análisis de cuerpo rígido con movimiento rectilíneo uniformemente acelerado, utilizando tecnologías IoT basadas en Arduino y ESP32. El sistema detecta el movimiento de un móvil mediante sensores ópticos colocados en diferentes posiciones del plano inclinado, graficando los datos de forma automática utilizando la plataforma de Thingspeak mediante un script de Matlab.

En un trabajo de investigación desarrollado por [Nuñez, 2023], se implementó la enseñanza de la robótica, los sistemas inteligentes, los microcontroladores

embebidos y el internet de las cosas mediante el aprendizaje basado en proyectos. Encontraron que la metodología aplicada favorece los procesos de enseñanza y aprendizaje de la programación de microcontroladores para su uso en sistemas inteligentes.

El diseño y desarrollo de una plataforma de enseñanza a distancia de microcontroladores e internet de las cosas se presenta en un trabajo de investigación de [Pereira,2022]. La plataforma presenta cuatro productos educativos basados en software libre, lo que permite su distribución gratuita en línea y su acceso desde un servidor en la nube. ESP UPDATE posibilita la programación de la tarjeta ESP32 de forma remota a través de OTA. ESPGPIO para ESP32 y RPI GPIO para Raspberry Pi posibilitan el control de electrodomésticos a través de internet. Mientras que IOTUS permite la grabación en línea de ADCES desarrollados con un microcontrolador PIC.

Por su parte, [Mitrovic,2023], presentaron una investigación que ofrece una visión sobre el método de comunicación de WebSocket en sistemas del internet de las cosas, donde el hardware del sistema está basada en el microcontrolador ESP32. En esta investigación se probó la fiabilidad de la transferencia de datos en tiempo real en redes Wi-Fi, comparándola con el método de sondeo largo. El estudio concluye que el método de WebSocket implementado tiene un mejor desempeño que el método de sondeo largo en aplicaciones que demandan una alta frecuencia en el envío de datos.

En un estudio sobre adquisición de datos en tiempo real con ESP32 para IoT, desarrollado por [Maroşan, 2024], se transmitieron los datos de los sensores de forma inalámbrica a través de un broker MQTT de código abierto. Encontraron que la solución propuesta garantiza una alta estabilidad y tiempos de respuesta rápidos, lo que la hace aplicable en diversos ámbitos como la monitorización y el control remoto de hogares inteligentes, la automatización de procesos industriales y el desarrollo de entornos inteligentes.

Un trabajo de investigación que presenta el uso de ESP32 y MQTT para el control de una casa inteligente mediante el reconocimiento de múltiples sensores fue desarrollado por [Makatita, 2024]. El estudio tuvo como propósito evaluar la eficacia

de la aplicación, el diseño del sistema, su desarrollo y las pruebas conducentes. Los resultados obtenidos arrojaron que la implementación de un sistema de hogar inteligente agiliza la gestión de los dispositivos electrónicos de uso cotidiano.

La presente investigación ofrece novedades didácticas y técnicas que pueden aplicarse en el aula, incluyendo una integración práctica del ciclo completo de un sistema IoT. El sistema está dividido en dos módulos principales: el de hardware y el de software. El primero obtiene las lecturas de variables físicas mediante sensores conectados a una tarjeta de desarrollo ESP32. En la tarjeta se programó el manejo de los sensores y la publicación suscripción mediante el protocolo MQTT. Este dispositivo envía y recibe información vía WiFi, conectándose con un broker proporcionado por HiveMQ. Se utilizó MQTT Explorer, cliente genérico de MQTT, y el cliente web de HiveMQ para monitorear y gestionar el tráfico en tiempo real.

Respecto al software, la información generada por los sensores es enviada a un servidor web local utilizando Node.js. Esto permite conectarse mediante WebSockets a una página web local, facilitando la comunicación en tiempo real con el ESP32 sin necesidad de depender únicamente del protocolo MQTT. En la página se muestran las lecturas de los sensores y se grafican los datos recibidos en tiempo real. También es posible visualizar esta información en un dispositivo móvil conectado a la misma red que la tarjeta ESP32 y almacenarla en una base de datos de MongoDB. Utilizando Python, es posible el análisis estadístico de los datos obtenidos de un archivo JSON generado desde la base de datos de MongoDB.

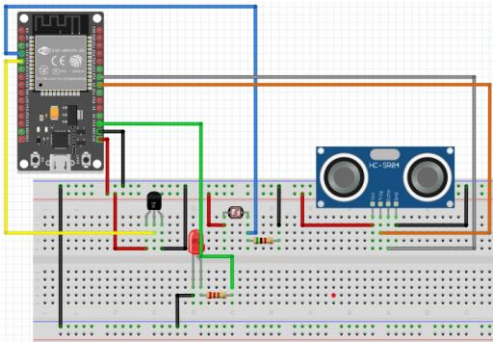
2. Métodos

El enfoque metodológico adoptado fue de tipo proyectivo, que como se establece en [Guía práctica de investigación proyectiva, 2025], se basa en la habilidad de prever y representar posibles escenarios futuros, junto con el empleo de métodos analíticos y prospectivos para entender las dinámicas que pueden afectar el desarrollo de sistemas complejos. En el contexto de esta investigación, la metodología prevé escenarios varios del uso del prototipo, así como el empleo de herramientas analíticas y de prospectiva técnica que permitan entender el desempeño del sistema IoT y su posible aplicación en entornos educativos.

La construcción del prototipo consideró principios de modularidad, escalabilidad y accesibilidad, utilizando componentes de bajo costo y tecnologías de código abierto. El prototipo está conformado por dos módulos interconectados: hardware y software, los cuales se describen a continuación.

Módulo de hardware

Para este módulo se utilizó una tarjeta de desarrollo ESP32 debido a su idoneidad para proyectos IoT por su bajo costo, conectividad integrada que incluye WiFi, bajo consumo de energía y alto rendimiento. Se conectaron tres sensores para la adquisición y análisis de datos, un sensor de distancia HCSR04 conectado a los pines GPIO5 y GPIO18, un sensor de temperatura LM35 conectado al pin de entrada analógica GPIO35 y una fotorresistencia LDR conectada al pin GPIO34, como se puede observar en la Figura 1. Además, se incluyó un led conectado en el GPIO15 para que, además de la suscripción para el envío de datos de los sensores vía MQTT, también se lleve a cabo el proceso de publicación para activar o desactivar actuadores, representados en este caso por el led.



Fuente: elaboración propia

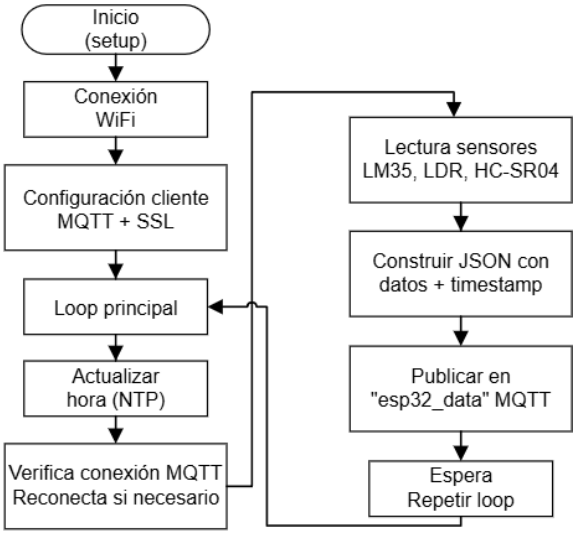
Figura 1 Circuito del prototipo de hardware.

En la tarjeta ESP32 se cargó un código desarrollado en el IDE de Arduino que implementa un sistema IoT seguro que lee los datos de los sensores y los publica en formato JSON en un broker MQTT mediante una conexión SSL/TLS, incluyendo una marca de tiempo NTP. Se establece un enlace a una red WiFi, se sincroniza la hora con un servidor NTP y se conecta a un broker MQTT seguro en el puerto 8883 usando un certificado raíz.

El programa identifica como entradas los sensores:

- Temperatura (analógico LM35), convirtiendo voltaje a grados Celsius.
- LDR, mapeado de $0-4096$ a un rango de 0-100%.
- Ultrasónico HC-SR04, que calcula la distancia midiendo el tiempo del eco.

El programa identifica como salidas un LED, controlado remotamente mediante el tópico MQTT "led_state", y la publicación de los datos de sensores como mensaje JSON en el tópico "esp32_data", con formato de tiempo ISO 8601. En la Figura 2 se muestra el diagrama de flujo del programa cargado en la tarjeta ESP32.



Fuente: elaboración propia

Figura 2 Diagrama de flujo programa ejecutado por ESP32.

Módulo de software

Se implementó un servidor backend usando Node.js, el cual se inicia importando los módulos necesarios: express para crear un servidor web, http para manejar las solicitudes HTTP, socket.io para la comunicación en tiempo real por WebSockets, mqtt para conectarse a un broker MQTT y mongodb para almacenar datos persistentes. El servidor express se configuró para servir archivos estáticos desde una carpeta “public”, permitiendo la interacción del usuario a través de una interfaz web. A su vez, se configuró el enlace con el broker MQTT utilizando credenciales y opciones seguras con el puerto 8883 para el uso de TLS. Después de conectar

exitosamente al bróker MQTT, el sistema se enlaza a una base de datos alojada en la nube mediante MongoDB Atlas, con una variable db que almacena la instancia de conexión una vez establecida.

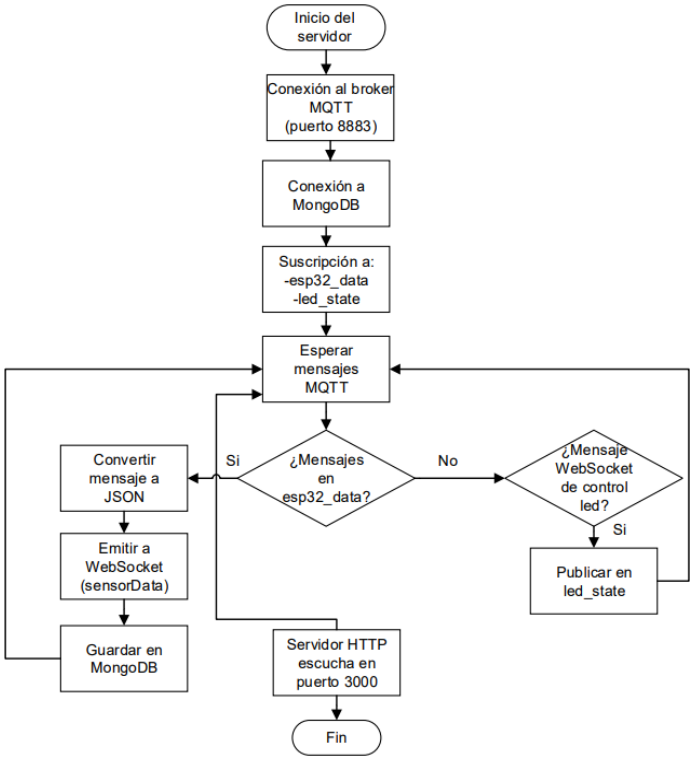
El cliente MQTT se suscribe a los tópicos esp32_data para recibir datos de sensores y led_state para controlar el estado de un LED. Cada vez que se recibe un mensaje en el tópico esp32_data, se convierte de texto a un objeto JSON, se envía a todos los clientes conectados vía WebSocket y se guarda en la base de datos dentro de la colección sensores. Si se detecta algún error en la conexión con el broker, se muestra un mensaje en la consola para facilitar el diagnóstico. Además, el servidor está configurado para manejar conexiones entrantes de clientes WebSocket, de tal manera que, cuando un cliente se conecta, el servidor escucha eventos llamados ledControl, los cuales contienen el estado deseado del LED.

Este estado es enviado al broker MQTT a través del tópico led_state, lo que permite el control remoto de cualquier actuador, aunque en este caso, como ilustración, solo es un LED conectado a la tarjeta ESP32. Finalmente, el servidor HTTP se inicia en el puerto 3000, permitiendo el acceso desde cualquier dirección IP local. También se define una ruta raíz ("/") que responde con un mensaje de prueba para verificar que el servidor está funcionando correctamente. Todo el sistema opera de forma asincrónica y basada en eventos, lo que es ideal para aplicaciones de monitoreo y control en tiempo real. El servidor montado en Node.js representa un componente esencial del prototipo IoT, ya que permite una comunicación eficiente, estable y flexible entre el broker MQTT y la interfaz web del usuario.

Node.js garantiza una respuesta rápida ante la llegada de datos en tiempo real, lo cual es fundamental para mantener la sincronización. En la Figura 3 se muestra el diagrama de flujo del código correspondiente.

Para realizar el análisis estadístico de la información de los sensores, se optó por utilizar Google Colab, aprovechando su gratuidad. Según [Google, s.f.], Colab es una plataforma en línea para ejecutar cuadernos Jupyter sin necesidad de instalación previa, que brinda acceso gratuito a recursos computacionales como GPUs y TPUs. Está diseñada especialmente para facilitar el trabajo en áreas como aprendizaje automático, ciencia de datos y educación. Se creó en esta plataforma

un cuaderno de Python que realiza un análisis temporal de datos provenientes de los sensores. Comienza importando las bibliotecas necesarias como pandas, matplotlib.pyplot y seaborn, estableciendo un entorno de análisis visual y estadístico.

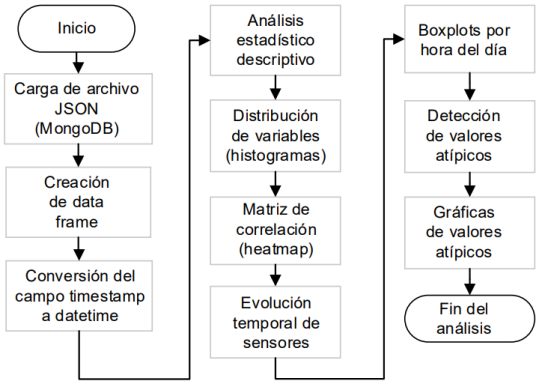


Fuente: elaboración propia

Figura 3 Diagrama de flujo del montaje del servidor local.

Luego, se carga un archivo JSON generado desde la base de datos en MongoDB Atlas con los registros de las variables de temperatura, distancia y luminosidad, cada uno asociado a un $siteId$ y a una marca de tiempo. Esta información se estructura en un DataFrame, lo que facilita su manipulación, limpieza y visualización. Posteriormente, se realiza un preprocesamiento de los datos, convirtiendo las columnas de fecha a un formato datetime, asegurándose que estén ordenados cronológicamente. Enseguida, se exploran mediante estadísticas descriptivas utilizando el método $describe()$ en las columnas numéricas del DataFrame. Para cada $siteId$, se trazan gráficos que muestran tendencias y posibles anomalías, apoyándose en las bibliotecas seaborn y matplotlib.

Con la función $corr()$ en Pandas se calcula la matriz de correlación entre cada par de columnas del DataFrame. Finalmente, el cuaderno emplea técnicas básicas de agrupamiento temporal con funciones como $resample$, que permite generar promedios diarios u horarios de las variables. Esto resulta útil para detectar patrones a largo plazo o fluctuaciones recurrentes. En conjunto, el cuaderno ofrece una herramienta completa para analizar series temporales de sensores IoT, permitiendo comprender su comportamiento y posibles eventos fuera de lo común. En la Figura 4 se observa el diagrama de flujo del cuaderno cargado en Colab.

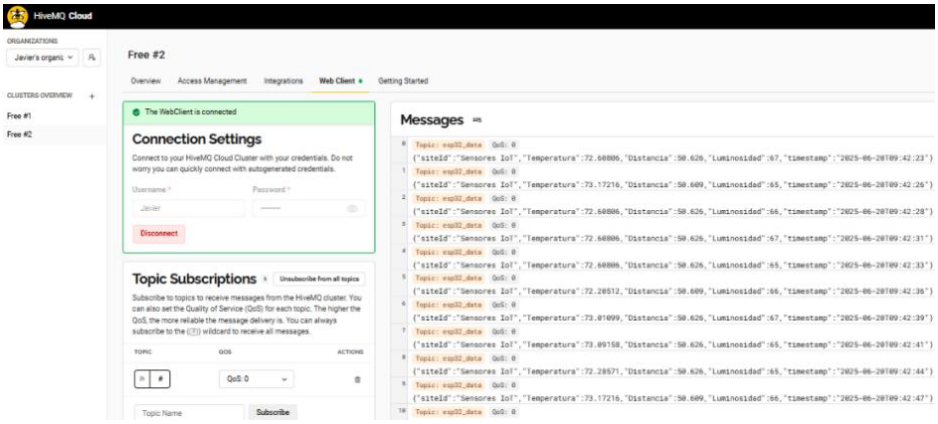


Fuente: elaboración propia

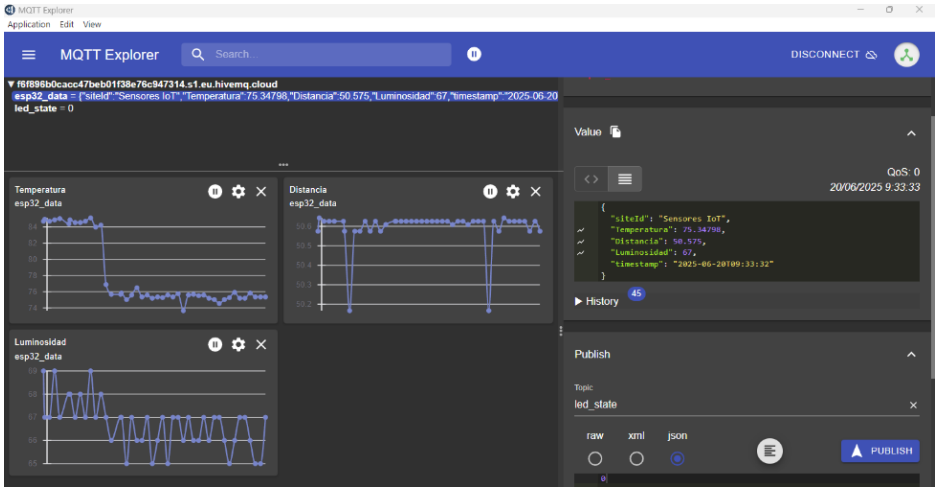
Figura 4 Diagrama de flujo del código Python para análisis estadístico.

3. Resultados

El prototipo desarrollado demostró un desempeño técnico robusto y adecuado para fines didácticos. El funcionamiento de la arquitectura de comunicación se ilustra en la Figura 5. El ESP32 publica lecturas de sensores al broker en la nube de HiveMQ Cloud, que actúa como componente central de mensajería rápida y confiable en sistemas IoT [HiveMQ, s.f.]. En la Figura 5a se observa el cliente web de HiveMQ mostrando en tiempo real los mensajes JSON estructurados con variables de temperatura, distancia, luminosidad y timestamp. La Figura 5b corresponde a MQTT Explorer, cliente suscriptor que grafica automáticamente la evolución de los datos, confirmando su recepción continua y estable. En conjunto, ambas interfaces validan la comunicación en tiempo real sin pérdida de datos y permiten la interacción bidireccional mediante el envío de comandos al dispositivo para controlar el estado del LED integrado.



a) Cliente web de HiveMQ Cloud mostrando mensajes JSON.



b) MQTT Explorer graficando las variables publicadas.

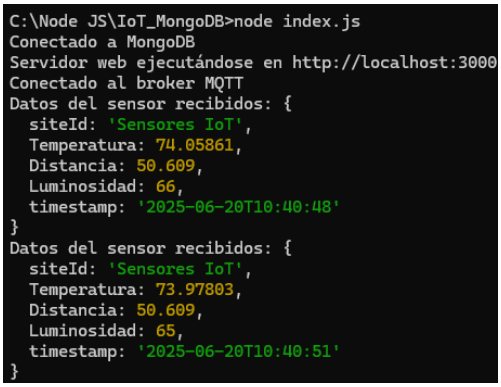
Fuente: elaboración propia

Figura 5 Flujo de datos del sistema.

La Figura 6 muestra el servidor web local en Node.js, que establece conexión tanto con el broker MQTT como con la base de datos MongoDB, recibiendo las lecturas de temperatura, distancia y luminosidad, así como sus respectivas marcas temporales enviadas por el ESP32. Cada mensaje es procesado y almacenado automáticamente, lo que asegura la persistencia de la información para análisis posteriores. Esta arquitectura local permite que el sistema funcione sin depender de la nube, garantizando acceso a los datos desde una red LAN y ofreciendo una página web en tiempo real. Su diseño modular y asincrónico posibilita un flujo continuo y tiempos de respuesta adecuados para la enseñanza.

El correcto registro de múltiples mensajes confirma la estabilidad del sistema, aspecto esencial en la validación técnica. De este modo, el servidor no solo cumple

un rol técnico de intermediario, sino también didáctico, al mostrar a los estudiantes el ciclo completo de un sistema IoT de manera integral.



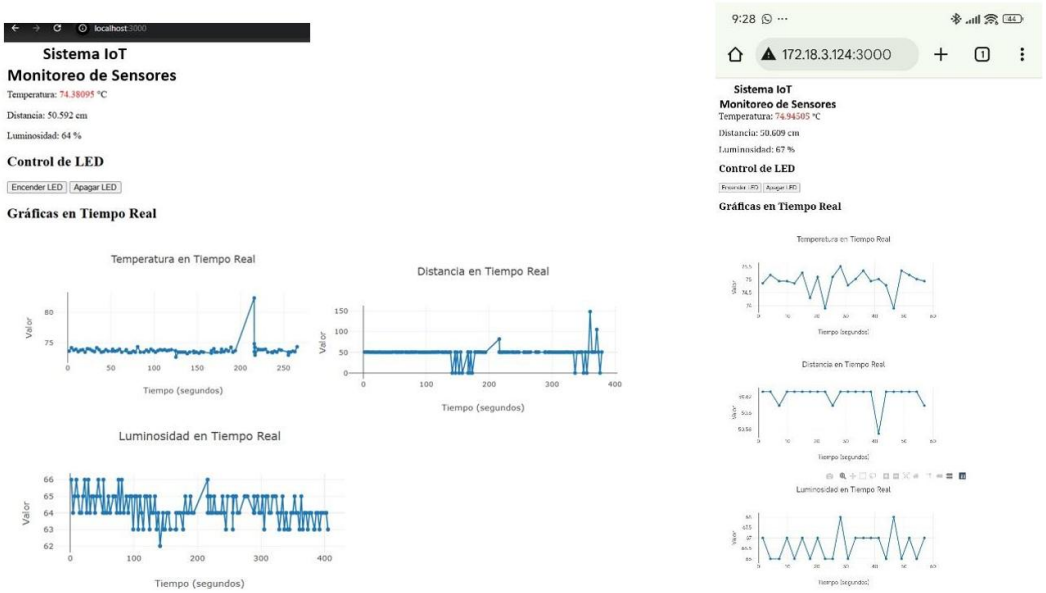
Fuente: elaboración propia

Figura 6 Salida de la consola del servidor Node.js.

La Figura 7 presenta la interfaz web desarrollada en Node.js para la visualización en tiempo real de las lecturas de temperatura, distancia y luminosidad. La página, accesible desde cualquier dispositivo en la red LAN sin depender de internet ni del broker MQTT, incluye un control remoto del LED integrado al ESP32 mediante botones interactivos. En la vista desde una PC que aparece en la Figura 7a se muestran tanto los valores numéricos como gráficas dinámicas generadas con Chart.js, que ilustran la evolución en segundos de las variables. La misma interfaz, desplegada en un dispositivo móvil vía IP local mostrada en la Figura 7b, valida su carácter multiplataforma y su correcta adaptación a distintos tamaños de pantalla. La actualización simultánea de datos y gráficas confirma que el sistema puede ser monitoreado y controlado desde varios puntos de acceso. Esto resulta particularmente útil en entornos educativos, ya que permite a múltiples estudiantes interactuar con la información en tiempo real, analizar el comportamiento de las variables y trabajar de manera colaborativa en la resolución de problemas.

La Figura 8 muestra el procesamiento estadístico realizado en Python con datos exportados desde MongoDB. La temperatura presenta una distribución aproximadamente normal alrededor de los $70\,°C$ con lecturas estables como se observa en la Figura 8a. Mientras que la distancia exhibe un comportamiento multimodal con picos recurrentes producto de movimientos simulados frente al

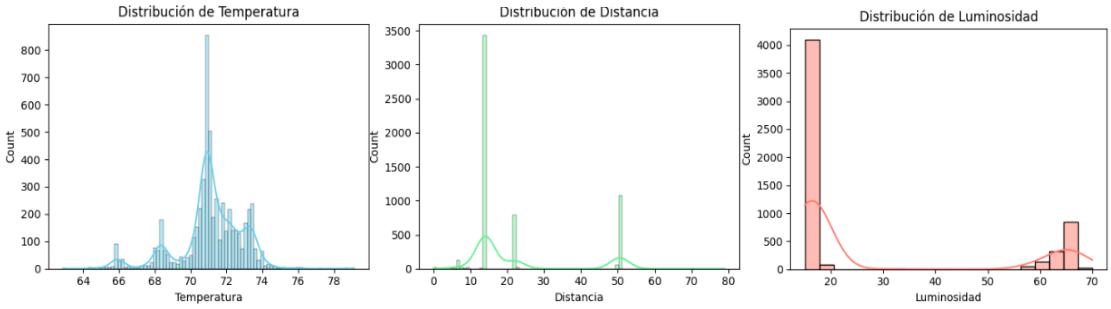
sensor ultrasónico, evidenciando la capacidad del sistema para captar condiciones dinámicas, como se observa en la Figura 8b.



a) Vista desde PC. *b) Vista desde dispositivo móvil.*

Fuente: elaboración propia

Figura 7 Vista de la página web desarrollada en Node.js.



a) Temperatura. *b) Distancia.* *c) lumninosidad.*

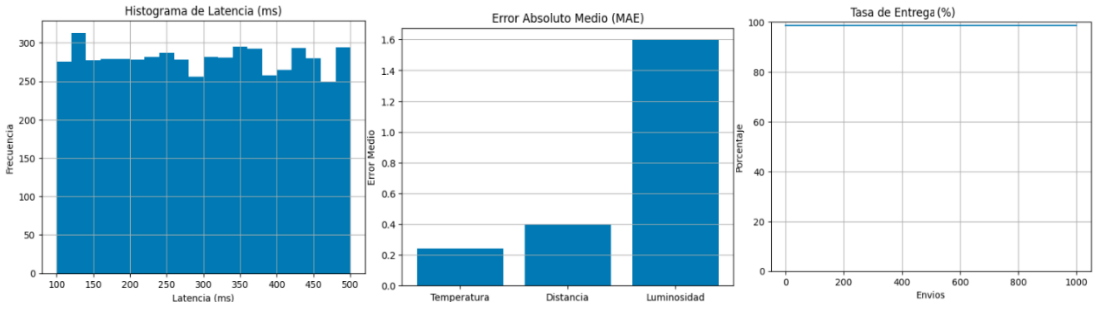
Fuente: elaboración propia

Figura 8 Distribución estadística de las variables medidas por el sistema.

La Figura 8c evidencia que el sistema capta con precisión cambios abruptos, y que la luminosidad presenta una distribución sesgada hacia valores bajos entre 15 y 20%, coherente con un entorno poco iluminado. Esta sensibilidad confirma la fiabilidad del sensor y, junto con el análisis estadístico, aporta información útil para la calibración, la detección de anomalías y el entrenamiento de modelos, lo que consolida al prototipo como una herramienta formativa con un enfoque analítico.

Pruebas funcionales

Se realizaron pruebas funcionales al prototipo enfocadas en la latencia, la exactitud de los datos y la estabilidad del sistema. En la Figura 9 se presentan tres gráficos generados a partir de un cuaderno de Python, utilizando los datos obtenidos por el sistema y los obtenidos mediante instrumentos de medición calibrados. El histograma de latencia observado en la Figura 9a muestra una distribución uniforme de los tiempos de transmisión entre el ESP32 y el servidor local, con una media aproximada de $298\ ms$. Esto confirma que el sistema tiene una respuesta aceptable para aplicaciones educativas en tiempo real, sin retardos perceptibles por el usuario.



a) Latencia. *b) Error absoluto medio.* *c) Tasa de entrega.*

Fuente: elaboración propia

Figura 9 Resultados de validación técnica del prototipo.

La gráfica de error absoluto medio MAE de la Figura 9b evalúa la precisión de los sensores al comparar los datos transmitidos con valores de referencia generados con ruido controlado. Se observan errores medios bajos en temperatura de $0.25\ °C$ y distancia de $0.40\ cm$, mientras que la luminosidad presenta una variabilidad de $1.5\ unidades$, debido a la resolución del sensor. La gráfica mostrada en la Figura 9c representa la tasa de entrega de mensajes después de 1000 envíos, con una pérdida del 2%. El sistema logra una tasa de recepción del 98%, lo que indica una alta estabilidad y confiabilidad del canal de comunicación MQTT en condiciones típicas de red local.

Estos resultados permiten afirmar que el prototipo es técnicamente robusto, con una arquitectura que garantiza transmisión eficiente, precisión aceptable en la medición y baja pérdida de datos, condiciones fundamentales para su implementación posterior como un recurso didáctico confiable.

4. Discusión

La incorporación del IoT en los procesos de formación de ingenieros en Sistemas Computacionales es de vital importancia para la preparación de profesionales capaces de enfrentar los retos de la industria 4.0. Para esto, el diseño, implementación y gestión de soluciones IoT requiere del dominio de competencias técnicas en programación de dispositivos embebidos, redes de comunicación, seguridad informática, desarrollo web y manejo de bases de datos. Esto incluye también habilidades transversales como pensamiento lógico, resolución de problemas y la integración de tecnologías emergentes. Trabajando con prototipos IoT los estudiantes pueden experimentar con sistemas reales, aplicar conocimientos y desarrollar soluciones innovadoras con impacto en su entorno.

Para lograr lo anterior es necesario complementar o actualizar los programas de estudio de las ingenierías tanto en el sector público como en el privado, ya que como menciona [Dizdarević, 2021], ante la vastedad y constantes cambios en las opciones y el contenido de las tecnologías, resulta crucial enfocar los cursos con metas bien definidas. Una opción viable es la utilización de prototipos que integren hardware y software de manera integral, que permitan al estudiante interactuar con cada parte del sistema. Desde la adquisición de datos, su transmisión vía MQTT, la visualización en tiempo real, su almacenamiento estructurado en una base de datos y su análisis estadístico.

A diferencia de las prácticas de laboratorio tradicionales que suelen centrarse en una sola etapa del proceso, el uso de este tipo de prototipos brinda una experiencia más completa, fomentando el pensamiento computacional y la comprensión sistémica. Al respecto, como lo menciona [Martínez, 2024], en ingeniería la implementación práctica del conocimiento es tan crucial como su comprensión teórica. No basta con diseñar soluciones y elaborar diagramas, es esencial modelar y prototipar, pues esto revela fallas y aciertos que el análisis teórico no percibe. Los prototipos ofrecen una experiencia invaluable, mayor certeza y la adquisición de competencias que permiten a los futuros ingenieros enfrentarse exitosamente a los retos del mercado laboral actual. En conjunto, los resultados muestran que la latencia promedio de $298\ ms$, los errores medios bajos de $0.25\ °C$ en temperatura y

$0.40\ cm$ en distancia, así como la tasa de entrega del $98\ \%$, se encuentran dentro de los rangos reportados en otros prototipos IoT educativos y son plenamente adecuados para fines didácticos. Esto confirma que el sistema, además de ser técnicamente funcional, ofrece un nivel de confiabilidad y realismo que lo distingue de propuestas similares basadas únicamente en la validación de hardware.

5. Conclusiones

El desarrollo del prototipo presentado en este trabajo permitió integrar de manera funcional y didáctica los principales componentes de un sistema IoT, incluyendo sensores físicos, un microcontrolador ESP32, comunicación mediante MQTT, un servidor local en Node.js y una interfaz web en tiempo real. Además, se complementó con el almacenamiento de la información en una base de datos para su análisis estadístico.

El prototipo logra simular un entorno tecnológico auténtico, accesible para los estudiantes de ingeniería, en el que pueden observar, analizar y comprender el ciclo completo de adquisición, transmisión, visualización, almacenamiento y análisis de los datos.

La incorporación de herramientas libres y de bajo costo como HiveMQ, MQTT Explorer, MongoDB y Google Colab favorece la sostenibilidad del proyecto en contextos educativos diversos. Las pruebas funcionales realizadas en un entorno controlado confirman la viabilidad técnica del sistema. Los resultados obtenidos de una latencia media de $298\ ms$, una tasa de entrega del $98\ \%$ y errores absolutos medios bajos en las variables sensadas, avalan su potencial para operar de forma estable en escenarios reales. Asimismo, la interfaz web multiplataforma brinda una experiencia de usuario intuitiva y adaptable a distintos dispositivos, lo que facilita su uso en entornos colaborativos, laboratorios o proyectos interdisciplinarios. Estas características posicionan al prototipo no solo como una solución técnica operativa, sino también como una herramienta pedagógica con alto valor formativo.

Como proyección futura se plantea un plan de implementación del prototipo en la asignatura de Sistemas Programables de la carrera de Ingeniería en Sistemas Computacionales. Este plan contempla su integración curricular en los temas de

programación de microcontroladores, puertos y buses de comunicación e interfaces. El diseño de prácticas de laboratorio progresivas que permitan a los estudiantes recorrer el ciclo completo de un sistema IoT, y la evaluación mediante rúbricas del desempeño técnico y competencias transversales. Asimismo, se prevé un piloto en un semestre académico con recolección de evidencias de aprendizaje y retroalimentación, lo cual permitirá validar la efectividad educativa del prototipo y ajustar su diseño con base en la experiencia real de los estudiantes.

6. Referencias y Bibliografía

[1] Dizdarević, J., & Jukan, A. Engineering an IoT-edge-cloud computing system architecture: Lessons learnt from an undergraduate lab course. In 2021 International Conference on Computer Communications and Networks (ICCCN), pp. 1-11. IEEE, 2021.

[2] Endara, J., (2023). Tecnologías emergentes en ingeniería y su impacto en la cultura digital. Ethos Scientific Journal, 1(1), pp. 4-19.

[3] Guía práctica de investigación proyectiva. (n.p.): Ecoe Ediciones, 2025.

[4] Google, (s.f.). Welcome to Colaboratory. https://colab.google/.

[5] HiveMQ, (s.f.). HiveMQ MQTT Broker. https://www.hivemq.com/products/mqtt-broker/.

[6] Makatita, F., & Hakim, N. MQTT protocol-based esp-32 smarthome with multi-sensor recognition. Journal of Electrical, Electronic, Information, and Communication Technology, 6(1), pp. 29-36, 2024.

[7] Maroşan, A., Constantin, G., Gîrjob, C., Chicea, A., Crenganiş, M., & Morarius, F. Real-time data acquisition with ESP32 for IoT applications using open-source MQTT brokers. Proceedings in Manufacturing Systems, 19(2), pp. 61-68, 2024.

[8] Martínez, A. Modelos y prototipos y su importancia en los procesos de enseñanza aprendizaje en ingeniería, EIEI ACOFI, 2024.

[9] Mitrović, N., Đorđević, M., Veljković, S., & Danković, D. Implementation and Testing of Websocket Protocol in ESP32 based IoT systems. Facta Universitatis, Series: Electronics and Energetics, 36(2), pp. 267-284, 2023.

[10] Montaño, E., Cuero, F., & Barrera, D. Innovaciones en la Pedagogía Moderna: Estrategias y Tecnologías Emergentes. Código Científico Revista de Investigación, 4(2), pp. 1041-1068, 2023.

[11] Nuñez, R., Guerrero, A., Villalón, M., Gutiérrez, P, & Sillero, J. Enseñanza de sistemas embebidos en robótica y sistemas inteligentes por medio de prácticas (teaching of embedded systems in robotics and intelligent systems through practices). Pistas Educativas, 44(144), 2023.

[12] Pereira, R., de Souza, C., Patino, D., & Lata, J. Plataforma de enseñanza a distancia de microcontroladores e internet de las cosas. Ingenius. Revista de Ciencia y Tecnología, (28), pp. 53-62, 2022.

[13] Rodríguez, J., & Rodríguez, S. Uso de Python para el análisis de datos aplicado en la investigación. Investigación y Ciencia Aplicada a La Ingeniería, 5(34), pp. 33-40, 2022.

[14] TecNM, (s.f.). Programa de estudio: Sistemas programables. Ingeniería en Sistemas Computacionales. Tecnológico Nacional de México. https://irapuato.tecnm.mx/moferta/sistcomputacionales/pdf/plan_estudios/7%20Sistemas%20Programables.pdf.

[15] Villalobos, E., Cornejo, S., Arellano, S., & Figueroa, A. Prototipo didáctico de plano inclinado con características IoT (inclined plane teaching prototype with IoT characteristics). Pistas Educativas, 43(140), 2021.