UNIVERSIDAD POLITÉCNICA SALESIANA

SEDE QUITO

CARRERA DE COMPUTACIÓN

DISEÑO E IMPLEMENTACIÓN DE UN DASHBOARD PARA MONITORIZACIÓN DE DATOS A TRAVÉS DE UN RASPBERRY PI 4

Trabajo de titulación previo a la obtención del título de Ingenieros en Ciencias de la Computación

AUTORES: HÉCTOR JOSÉ UBILLUS BARONA

DAVID ALEXANDER VARGAS DONOSO

TUTOR: MANUEL RAFAEL JAYA DUCHE

Quito – Ecuador

2024

CERTIFICADO DE RESPONSABILIDAD Y AUTORÍA DEL TRABAJO DE TITULACIÓN

Nosotros, Héctor José Ubillus Barona con documento de identificación N.º 0924654395 y David Alexander Vargas Donoso con documento de identificación N.º 1724450158 manifestamos que:

Somos los autores y responsables del presente trabajo; y, autorizamos a que sin fines de lucro la Universidad Politécnica Salesiana pueda usar, difundir, reproducir o publicar de manera total o parcial el presente trabajo de titulación.

Quito, 29 de julio del 2024

Atentamente,

Héctor José Ubillus Barona	David Alexander Vargas Donoso
0924654395	1724450158

CERTIFICADO DE CESIÓN DE DERECHOS DE AUTOR DEL TRABAJO DE TITULACIÓN A LA UNIVERSIDAD POLITÉCNICA SALESIANA

Nosotros, Héctor José Ubillus Barona con documento de identificación N.º 0924654395 y David Alexander Vargas Donoso con documento de identificación N.º 1724450158, expresamos nuestra voluntad y por medio del presente documento cedemos a la Universidad Politécnica Salesiana la titularidad sobre los derechos patrimoniales en virtud de que somos los autores del Proyecto Técnico: "Diseño e implementación de un Dashboard para monitorización de datos a través de un Raspberry Pi 4", el cual ha sido desarrollado para optar por el título de: Ingeniero en Ciencias de la Computación, en la Universidad Politécnica Salesiana, quedando la Universidad facultada para ejercer plenamente los derechos cedidos anteriormente.

En concordancia con lo manifestado, suscribimos este documento en el momento que hacemos la entrega del trabajo final en formato digital a la Biblioteca de la Universidad Politécnica Salesiana.

Quito, 29 de julio del 2024

Atentamente,

Héctor José Ubillus Barona
0924654395

David Alexander Vargas Donoso
1724450158

CERTIFICADO DE DIRECCIÓN DEL TRABAJO DE TITULACIÓN

Yo, Manuel Rafael Jaya Duche con documento de identificación N.º 1710631035, docente de la Universidad Politécnica Salesiana, declaro que bajo mi tutoría fue desarrollado el trabajo de titulación: DISEÑO E IMPLEMENTACIÓN DE UN DASHBOARD PARA MONITORIZACIÓN DE DATOS A TRAVÉS DE UN RASPBERRY PI 4, realizado por Héctor José Ubillus Barona con documento de identificación N.º 0924654395 y David Alexander Vargas Donoso con documento de identificación N.º 1724450158, obteniendo como resultado final el trabajo de titulación bajo la opción Proyecto Técnico que cumple con todos los requisitos determinados por la Universidad Politécnica Salesiana.

Quito, 29 de julio del 2024

Atentamente,

Ing. Manuel Rafael Jaya Duche, MSc.

1710631035

DEDICATORIA

A mis padres y padrastros,

Quienes, con su amor incondicional, sacrificio y sabiduría han sido mi guía y fortaleza en cada paso de este camino. Su apoyo constante y sus enseñanzas han moldeado mi carácter y han inspirado mis sueños. Esta tesis es un reflejo de su esfuerzo y dedicación, y es a ustedes a quienes debo cada logro alcanzado. Gracias por creer en mí y por ser mi fuente inagotable de motivación.

A mis abuelos,

Por la ayuda hacia la familia, su amor, sabiduría y valores han sido una fuente constante de inspiración y fortaleza. Agradezco profundamente sus historias, sus consejos y el cariño incondicional que siempre me han brindado. Este logro es también suyo, pues sus enseñanzas y apoyo han sido fundamentales en mi camino. Con todo mi respeto y amor, les dedico esta tesis.

Héctor Ubillus

DEDICATORIA

A mi madre,

Cuya fortaleza y dedicación son la base de mi éxito. A pesar de las dificultades y desafíos que la vida nos ha presentado, nunca ha dejado de apoyarme y de luchar incansablemente por mi bienestar y asegurar mi futuro. Su amor incondicional y su perseverancia son una fuente constante de inspiración para mí.

A mi abuela,

Por su inquebrantable apoyo y amor. Su presencia constante y su sabiduría han sido fundamentales en cada paso de este camino. Gracias por estar siempre a nuestro lado, brindándonos su cariño y apoyo incondicional.

A ambas, por ser el pilar de mi vida, dedico este trabajo con todo mi amor y gratitud.

David Vargas

AGRADECIMIENTO

El recorrido de una carrera universitaria no fue sencillo, pero he tenido el privilegio de contar con el apoyo incondicional de muchas personas que han hecho posible la culminación de este trabajo. Sus palabras de aliento, paciencia y orientación han sido fundamentales para mantenerme firme en este camino y evitar que me rindiera.

En primer lugar, quiero expresar mi profunda gratitud a mi familia, por su amor inquebrantable y su constante respaldo. Cada uno de ustedes ha sido mi fuente de inspiración y fortaleza, y no habría llegado hasta aquí sin su apoyo.

A mis amigos y compañeros de carrera, gracias por estar a mi lado durante este viaje. Sus ánimos y momentos compartidos hicieron más llevaderos los desafíos académicos.

Finalmente, a todos aquellos que, de una u otra forma, me brindaron su apoyo y aliento, les estoy profundamente agradecido.

Este trabajo no solo representa el resultado de mi esfuerzo, sino también el fruto del apoyo generoso de quienes creyeron en mí. A todos ustedes, mi más sincero agradecimiento.

Héctor Ubillus

AGRADECIMIENTO

Quiero expresar mi más sincero agradecimiento a todas las personas que han contribuido de alguna manera en la realización de este proyecto de titulación. En primer lugar, agradezco a Dios por darme la salud, la fuerza y la perseverancia necesarias para llevar a cabo este trabajo. A mis padres, por su amor incondicional, apoyo constante y por creer siempre en mí. A mi familia en general, por estar siempre a mi lado y ofrecerme su respaldo incondicional en cada etapa de mi vida académica. A mis amigos de la Universidad Politécnica Salesiana, por su apoyo durante estos años de estudio. A los profesores de la Universidad Politécnica Salesiana, por impartirme los conocimientos y habilidades necesarios para el desarrollo de este trabajo y por su compromiso con la excelencia académica. Finalmente, agradezco a todas las personas que de alguna manera contribuyeron a la realización de este proyecto. A todos, ¡muchas gracias!

David Vargas

ÍNDICE DE CONTENIDOS

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

RESUMEN

Este proyecto tiene como objetivo diseñar e implementar un dashboard para el Raspberry Pi 4, inspirado en la plataforma existente Thinger.io. La meta es crear un prototipo de dashboard que incluya los cuatro gráficos más utilizados en Thinger.io, proporcionando funcionalidad para la monitorización de datos tanto analógicos como digitales. Esta solución está específicamente adaptada para los laboratorios de IoT de la Universidad Politécnica Salesiana, ofreciendo una alternativa de código abierto y de bajo costo a las aplicaciones propietarias.

Se decidió emplear la metodología XP para garantizar un desarrollo ágil y adaptable del proyecto. Se comenzó con una investigación detallada de las capacidades del Raspberry Pi 4 para la adquisición y procesamiento de datos de sensores, asegurando una integración efectiva de bibliotecas y APIs . Posteriormente, se diseñó el dashboard, enfocándose en maximizar la compatibilidad con el hardware disponible y en ofrecer representaciones visuales claras y efectivas de los datos mediante la selección cuidadosa de gráficos y widgets. La fase de desarrollo concentro en la implementación básica de la monitorización de datos. Finalmente, se realizaron pruebas para validar la funcionalidad y eficiencia del sistema en un entorno controlado. En conclusión, el proyecto entregará un prototipo funcional de dashboard que servirá como herramienta base para futuros desarrollos en el campo de IoT. Aunque se reconocen las limitaciones actuales, se espera que esta solución no solo beneficie directamente a la comunidad académica de la Universidad Politécnica Salesiana, sino que también fomente la colaboración y la innovación en el ámbito de la tecnología de la información y la comunicación.

Palabras clave: Raspberry Pi 4, Dashboard, IoT, Monitorización de datos, Thinger.io, Open source.

ABSTRACT

This project aims to design and implement a dashboard for the Raspberry Pi 4, inspired by the existing Thinger.io platform. The goal is to create a prototype dashboard that includes the four most used graphs in Thinger.io, providing functionality for monitoring both analog and digital data. This solution is specifically tailored for the Salesian Polytechnic University's IoT labs, offering a low-cost, open source alternative to proprietary applications.

It was decided to employ the XP methodology to ensure an agile and adaptable development of the project. We started with a detailed investigation of the capabilities of the Raspberry Pi 4 for sensor data acquisition and processing, ensuring an effective integration of libraries and APIs. Subsequently, the dashboard was designed, focusing on maximizing compatibility with the available hardware and providing clear and effective visual representations of the data through the careful selection of graphics and widgets. The development phase concentrated on the basic implementation of data monitoring. Finally, tests were performed to validate the functionality and efficiency of the system in a controlled environment.

In conclusion, the project will deliver a functional dashboard prototype that will serve as a base tool for future developments in the IoT field. While acknowledging current limitations, it is expected that this solution will not only directly benefit the academic community of the Salesian Polytechnic University, but will also foster collaboration and innovation in the field of information and communication technology.

Keywords: Raspberry Pi 4, Dashboard, IoT, Data monitoring, Thinger.io, Open source.

CAPÍTULO I

ANTECEDENTES Y GENERALIDADES

Este capítulo presenta una visión general de los antecedentes que motivan este proyecto, abordando el contexto del Internet de las Cosas (IoT) y su creciente importancia en la educación superior. Asimismo, se explorarán las capacidades técnicas de la Raspberry Pi 4 y la necesidad de desarrollar un sistema de dashboard adaptable que optimice la visualización y análisis de datos en entornos académicos.

1. Introducción

El creciente interés en soluciones de Internet de las Cosas (IoT) ha generado una gran demanda de plataformas eficientes para el monitoreo y control de dispositivos (Devis, 2014). La Raspberry Pi 4, gracias a su bajo consumo energético y versatilidad, se ha consolidado como una opción popular para proyectos de IoT. Sin embargo, se ha identificado falta de sistemas diseñados específicamente para el monitoreo de datos tanto analógicos como digitales (Shirgaonkar et al., 2023). A pesar de la existencia de plataformas como Thinger.io (Thinger.Io, 2019), que facilitan la creación de dashboards IoT, aún persiste la falta de soluciones que se integren de manera óptima con la Raspberry Pi 4, especialmente en aplicaciones que requieren la gestión simultánea de datos analógicos y digitales (Alarcón Diaz et al., 2023).

En el contexto de la Universidad Politécnica Salesiana, para la materia de Sistemas Embebidos e IoT, se ha destacado la ausencia de una aplicación para el análisis de datos. Esta situación subraya la razón por la cual se recurre a soluciones como Thinger.io, que, si bien ofrecen ciertas capacidades, no satisfacen completamente las necesidades específicas de proyectos avanzados que involucran la Raspberry Pi 4 (Mirjana Maksimović et al., 2014).

El presente proyecto tiene como objetivo diseñar e implementar un dashboard para el Raspberry Pi 4, inspirado en el existente en thinger.io. Este proyecto consiste en desarrollar un prototipo de dashboard que sea compatible con el Raspberry Pi 4, incorporando los cuatro gráficos más utilizados de thinger.io para monitorizar datos analógicos y digitales. Cabe destacar que, al tratarse de un prototipo, nuestra solución presentará limitaciones en comparación con la aplicación original de thinger.io, careciendo de algunas funcionalidades avanzadas. Sin embargo, una ventaja significativa de nuestro proyecto es que será una solución open source o de software libre, a diferencia de las aplicaciones propietarias. Este dashboard estará específicamente enfocado en la monitorización de datos y su implementación está dirigida a los laboratorios de IoT de la Universidad Politécnica Salesiana. Con este proyecto, se busca proporcionar una herramienta accesible y útil para la visualización y análisis de datos en entornos educativos, fomentando la experimentación y el aprendizaje en el campo de Internet de las Cosas (IoT)

1.2. Problema de estudio

En el contexto de la materia Sistemas Embebidos e Iot perteneciente a la Universidad Politécnica Salesiana, se reconoció la ausencia de aplicativos que permiten el monitoreo y control de dispositivos con Raspberry Pi 4. Esta carencia de dashboards adecuados representa un obstáculo importante para los usuarios que buscan interactuar, analizar y tomar decisiones basadas en los datos recogidos por dispositivos IoT basados en Raspberry Pi 4. Además, en el ámbito académico, especialmente en la Universidad Politécnica Salesiana, se ha observado una falta de proyectos específicos que se enfoquen en la implementación de dashboards para la monitorización eficiente de sensores a través de esta plataforma. Esta problemática limita el avance tecnológico y educativo en el campo de IoT en Ecuador, así como la capacidad de los usuarios para aprovechar al máximo el potencial del Raspberry Pi 4 en proyectos de monitorización de datos. Por lo tanto, existe una necesidad apremiante de desarrollar

dashboards adaptados a las especificaciones del Raspberry Pi 4, que permitan una interacción eficiente y una gestión efectiva de datos en aplicaciones de IoT en el contexto ecuatoriano.

1.2.1. Antecedentes

(Lara, 2019) Su investigación tuvo como objetivo desarrollar un prototipo para la monitorización de parámetros ambientales utilizando tecnologías como Raspberry Pi, Arduino Uno, y un agente multiprotocolo. El proceso y los componentes del sistema fueron adquirir datos ambientales conectados a un Arduino Uno para adquirir parámetros, transmisión de datos capturados por los sensores en el Arduino Uno, transmitidos a una Raspberry Pi, almacenar datos en archivos de texto en la Raspberry Pi, Base de Datos en Otra Raspberry Pi y un agente multiprotocolo que define objetos para consultas, todo este proyecto combinó hardware (Arduino Uno, Raspberry Pi) con software (agente multiprotocolo, aplicación Java) para crear un sistema completo de monitorización y gestión de datos ambientales, destacándose por su eficiencia y capacidad para operar en tiempo real (García & Barragán, 2014). Si, se diseñó y desarrolló una interfaz funcional que permite la comunicación entre el Raspberry Pi y la Tarjeta Mega Arduino Esta interfaz supera las disparidades entre estos dispositivos utilizando un circuito que permite la interconexión a través de los puertos GPIO del Raspberry Pi o mediante conexión serial entre las tarjetas. Los resultados del proyecto demostraron un funcionamiento práctico en un prototipo validando así la viabilidad de la solución propuesta. Se logró crear una interfaz amigable y de fácil comprensión para usuarios generales, permitiéndoles diseñar modelos a su elección de manera efectiva mediante la impresión 3D. (Marcu et al., 2019) ¿Cómo se puede desarrollar e implementar un sistema basado en IoT utilizando Raspberry Pi para el monitoreo del entorno y cuáles son los resultados obtenidos en la deteccion temprana de dichos datos? Se puede desarrollar e implementar un sistema basado en IoT utilizando Raspberry Pi para el monitoreo del entorno mediante la integración de sensores analógicos y digitales. Estos sensores son utilizados para monitorear variables clave, apoyados por un

algoritmo de clasificación de sonidos de fondo. Los resultados del estudio "IoT System for Forest Monitoring" indican que este sistema es capaz de detectar tempranamente cambios de temperatura y fuentes de contaminación. Esta capacidad de detección anticipada se ha demostrado efectiva, confirmando que las soluciones IoT son viables y ofrecen una herramienta valiosa para mejorar la gestión en los entornos. A nivel global, la implementación de soluciones tecnológicas para la monitorización de datos en proyectos de Internet de las Cosas (IoT) está ganando una importancia creciente (Sánchez & Ramoscelli, 2017). Según un informe de Statista (2021), se espera que el número de dispositivos IoT conectados alcance los 30.9 mil millones para 2025, lo que representa un crecimiento exponencial desde los 13.8 mil millones en 2021 (Vailshery, 2023). Esta tendencia subraya la creciente necesidad de soluciones eficientes para la monitorización y gestión de datos. En este escenario, los dashboards para la monitorización de datos juegan un papel crucial, actuando como interfaces que permiten a los usuarios interactuar, analizar y tomar decisiones basadas en los datos recogidos por una multitud de dispositivos IoT.

En Ecuador, el uso del Raspberry Pi 4 en proyectos de Internet de las Cosas (IoT) se está volviendo cada vez más relevante, pero enfrenta el desafío de carecer de dashboards específicamente optimizados para esta plataforma. El Raspberry Pi 4 es una opción preferida debido a su capacidad de procesamiento y versatilidad, lo cual lo hace ideal para una amplia gama de aplicaciones IoT. Sin embargo, la configuración y el monitoreo eficiente de estos sistemas pueden ser complicados debido a la falta de dashboards intuitivos y adaptados a las necesidades específicas de los usuarios y proyectos en Ecuador. (Bosquez & Valencia, 2022)

1.2.2. Importancia

La importancia del aplicativo radica en la necesidad de soluciones tecnológicas para el monitoreo de datos en el contexto del Internet de las Cosas (IoT). Enfocado a la creación de un

sistema de monitoreo de datos analógicos y digitales utilizando la Raspberry Pi 4. La implementación de soluciones de IoT está en auge, con una proyección de 30.9 mil millones de dispositivos conectados para 2025 (Statista, 2021). Esta tendencia resalta la necesidad de plataformas eficientes que permitan la gestión y análisis de los datos generados por estos dispositivos. En Ecuador, y específicamente en la Universidad Politécnica Salesiana, la adopción del Raspberry Pi 4 en proyectos académicos ha sido prominente; sin embargo, existe una notable ausencia de dashboards optimizados para este tipo de plataforma, lo que complica la configuración y monitoreo de sistemas IoT. Este trabajo aporta a las carencias identificadas en el contexto académico al implementar una solución open source, se asegura la transparencia, flexibilidad y la posibilidad de participación de la comunidad en su desarrollo y mejora continua.

1.2.3. Delimitación

Este aplicativo es un prototipo en el que se integra el uso de Raspberry Pi 4 para monitorizar datos en entornos académicos de la Universidad Politécnica Salesiana.

1.3. Justificación

En respuesta a esta problemática, nuestro principal aporte va a consistir en el desarrollo de un dashboard diseñado para ser compatible con el Raspberry Pi 4, el cual incluirá cuatro de los widgets más comunes para una visualización efectiva de datos. Estos widgets son el de texto, gráfico, tacómetro y medidor (gauge). El propósito del dashboard es presentar tanto datos digitales como analógicos, asegurando una interfaz intuitiva y funcional que permita a los usuarios comprender y analizar los datos con facilidad. La inclusión de estos elementos garantizará una representación clara y precisa de la información, facilitando la toma de decisiones y el seguimiento de indicadores clave de rendimiento (Santos, 2024). Este proyecto de investigación adquiere su importancia en el contexto del Internet de las Cosas (IoT) aplicado

a través del uso de Raspberry Pi 4, especialmente dentro de la Universidad Politécnica Salesiana ya que se ha identificado una carencia significativa de dashboards para el monitoreo de datos analógicos y digitales , la implementación de dashboards emerge como un componente clave para facilitar las necesidades específicas del ámbito académico y de investigación (Los desafíos y oportunidades de la adopción del Internet de las cosas (IoT) en Colombia, 2023).

La motivación para emprender este proyecto surgió al observar que plataformas existentes, como Thinger.io, aunque útiles, no cumplían completamente con las necesidades de los usuarios en la Universidad Politécnica Salesiana ya que buscan integrar la Raspberry Pi 4 en aplicaciones de IoT. La relevancia de este proyecto radica en su propuesta de desarrollar una herramienta versátil, eficiente e intuitiva. Diseñada específicamente para el monitoreo de datos en la Raspberry Pi 4, la cual facilitará la comprensión y el análisis de la información, ofreciendo una solución adaptada a las necesidades particulares de la comunidad universitaria y potenciando su capacidad en el campo de la tecnología IoT.

1.4. Objetivos Generales y Específicos.

1.4.1. Objetivo General

Diseñar e implementar un sistema de monitoreo de datos analógicos y digitales para la Raspberry Pi 4 utilizando una interfaz web.

1.4.2. Objetivos Específicos

1. Investigar y analizar las capacidades de Raspberry Pi 4, que involucran la monitorización de datos analógicos y digitales.
2. Diseñar y desarrollar un dashboard incorporando widgets comunes como texto, gráfico, tacómetro y medidor (gauge) para garantizar una visualización efectiva de datos.

3. Configurar y validar la conectividad del Raspberry Pi 4 con una variedad de sensores, asegurando una integración fluida y la capacidad de recibir tanto señales analógicas como digitales sin complicaciones técnicas adicionales.

4. Realizar pruebas del sistema para evaluar su desempeño, incluyendo la precisión en el monitoreo de datos, la estabilidad de la conectividad con los sensores y la usabilidad del dashboard.

5. Presentar resultados que demuestren la eficacia del sistema implementado, destacando su funcionalidad, rendimiento y utilidad en el contexto de la monitorización de datos.

1.5. Alcance

El alcance del proyecto se centra en el diseño e implementación de un sistema de monitoreo para la Raspberry Pi 4, enfocado en el análisis de datos análogos y digitales en un contexto académico. Cada uno de los objetivos del proyecto se especifica para asegurar la consecución de resultados específicos. A continuación, se presenta el alcance detallado de cada objetivo:

- Investigar y analizar las capacidades de Raspberry Pi 4, que involucran la monitorización de datos analógicos y digitales.

Se realizará una investigación detallada sobre cómo la Raspberry Pi 4 puede ser utilizada eficazmente para recopilar y analizar datos análogos y digitales, considerando las capacidades de hardware y software del dispositivo. El proyecto se centra en la maximización de las prestaciones de la Raspberry Pi 4 para la monitorización de datos en entornos académicos, alineándose directamente con el objetivo de investigar y analizar sus capacidades dentro de los laboratorios de la Universidad Politécnica Salesiana.

- Diseñar y desarrollar un dashboard incorporando widgets comunes como texto, gráfico, tacómetro y medidor (gauge) para garantizar una visualización efectiva de datos.

Se desarrollará un diseño e implementación del dashboard se realizarán utilizando Python e interfaces web con la Raspberry Pi 4. La interfaz se centrará en la visualización de datos, incorporando cuatro widgets esenciales: texto, gráfico, tacómetro y medidor (gauge).

- Configurar y validar la conectividad del Raspberry Pi 4 con una variedad de sensores, asegurando una integración fluida y la capacidad de recibir tanto señales analógicas como digitales sin complicaciones técnicas adicionales.

Se realizará la configuración y validación de la Raspberry Pi 4 para asegurar su conectividad con una amplia variedad de sensores. Esto facilitará la recopilación eficiente de datos tanto analógicos como digitales.

- Realizar pruebas del sistema para evaluar su desempeño, incluyendo la precisión en el monitoreo de datos, la estabilidad de la conectividad con los sensores y la usabilidad del dashboard. Presentar resultados y métricas que demuestren la eficacia del sistema y permitan identificar áreas de mejora.
- Se llevarán a cabo pruebas del sistema implementado para evaluar su rendimiento en diferentes aspectos clave, como la precisión en la recopilación y monitoreo de datos, la estabilidad de la conexión con los sensores y la facilidad de uso del dashboard.

Se realizará un análisis rendimiento, así como pruebas para para evaluar su rendimiento en aspectos clave como la precisión en la recopilación y monitoreo de datos, la estabilidad de la conexión con los sensores y la facilidad de uso del dashboard

Con lo ya mencionado, este aplicativo se resalta en el diseño e implementación de un sistema de monitoreo específicamente para Raspberry Pi 4, enfocado en el análisis de datos análogos y digitales para uso académico. Esta delimitación implica una investigación detallada sobre cómo la Raspberry Pi 4 puede ser utilizada eficazmente para recopilar y analizar estos tipos de datos, considerando las capacidades de hardware y software del dispositivo. La implementación se orienta hacia la maximización de las prestaciones de la Raspberry Pi 4 para la monitorización de datos en entornos académicos, alineando directamente con el objetivo de investigar y analizar sus capacidades, dentro de los laboratorios de la Universidad Politécnica Salesiana, aspecto que implica una exploración profunda de las capacidades del sistema para desempeñarse ante ambos tipos de datos asegurando su aplicabilidad en un entorno educativo.

CAPÍTULO II

MARCO TEÓRICO

El presente capítulo establece el fundamento teórico que sustenta el desarrollo del proyecto de monitorización de datos mediante un sistema de dashboard integrado con la Raspberry Pi 4. Se abordarán los conceptos clave del Internet de las Cosas (IoT), la arquitectura de sistemas embebidos, y las tecnologías de procesamiento y visualización de datos.

2.1. Internet de las cosas (IoT)

El Internet de las cosas (IoT) conecta objetos cotidianos al Internet, desde bombillas hasta dispositivos médicos y prendas inteligentes. Estos dispositivos pueden ser interruptores o sensores, facilitando la transferencia de datos sin intervención humana. Un ejemplo es un termostato que ajusta la temperatura según la ubicación del automóvil del usuario. Los sistemas de IoT envían, reciben y analizan datos continuamente, permitiendo acciones como ajustar el termostato según el tráfico o los hábitos de conducción. Las empresas también pueden usar datos de IoT para mejorar servicios como la gestión de la energía (Redhat, 2023).

2.2. Dashboards para Monitoreo de Datos

Los dashboards son herramientas fundamentales en el ámbito del IoT, ya que permiten a los usuarios interactuar, analizar y tomar decisiones basadas en los datos recopilados por dispositivos conectados. En este sentido, el diseño e implementación de dashboards adaptados específicamente para la Raspberry Pi 4 se presenta como una necesidad para satisfacer las demandas de los usuarios en términos de visualización y comprensión de datos (Pires, 2021).

2.3. Integración de Sensores y Monitoreo Continuo

En la implementación de sistemas IoT, la integración de sensores para la captura de datos es crucial. Estos sensores pueden medir una amplia variedad de variables, desde

temperatura y humedad hasta movimiento y calidad del aire. El monitoreo continuo de estos datos a través de la Raspberry Pi 4 ofrece la oportunidad de obtener información en tiempo real sobre el entorno y los dispositivos conectados, permitiendo una toma de decisiones más informada y una respuesta rápida a eventos o situaciones específicas (Ventura, 2023).

2.4. Interfaz Intuitiva y Funcionalidad

La clave para el éxito de un dashboard radica en su capacidad para ofrecer una interfaz intuitiva y funcional que permita a los usuarios comprender y analizar los datos con facilidad. Esto implica la inclusión de widgets comunes como texto, gráficos, tacómetros y medidores (gauge), que aseguran una representación clara y precisa de la información, facilitando la toma de decisiones y el seguimiento de indicadores clave de rendimiento (Adereso Team, 2023).

2.5. Integración Continua

Es una práctica en el desarrollo de software que implica la integración frecuente del código de todos los desarrolladores en un repositorio compartido. Cada integración se verifica con una construcción automatizada y pruebas, lo que permite detectar errores tempranos y reducir problemas de integración (¿Qué es la integración continua?, 2024).

2.6. GitLab

Es una plataforma de repositorios web de código abierto gratuita que permite que varios miembros de un equipo pueden utilizar GitLab para colaborar en el mismo proyecto (GitLab: Saber todo sobre el repositorio Git para DevOps, 2022).

2.7. GitHub

Es una plataforma de repositorios web que permite almacenar, desarrollar, gestionar y controlar los distintos proyectos de software (Muñoz, 2020).

2.8. Middlewares

Es un software que permite a los desarrolladores que sus aplicaciones se comuniquen de forma eficiente mediante los servicios y funcionalidades de la web que ofrece aws (Amazon web services) (¿Qué es el middleware?, 2023).

2.9. Serverless

Serverless es un modelo de computación en la nube que permite ejecutar aplicaciones sin gestionar servidores, asignando recursos automáticamente según la demanda. Facilita el desarrollo y despliegue de aplicaciones, centrándose en el código y no en la infraestructura (What is serverless?, 2024).

2.9.1. Amazon Web Services

Amazon Web Services (AWS) es una plataforma de servicios en la nube ofrecida por Amazon, el cual proporciona una amplia gama de servicios de infraestructura y software bajo demanda, estos proveen almacenamiento, cómputo, bases de datos, análisis de datos, aprendizaje automático, etc (Portal, 2022).

2.9.2. Aws Cognito

Este servicio de Amazon Web Services facilita la autenticación, autorización y gestión de usuarios en aplicaciones web y móviles, además permite a los desarrolladores agregar fácilmente el registro de usuarios, inicio de sesión y el acceso a través de proveedores posee la capacidad de administración de perfiles de usuario y almacenamiento de los datos del usuario de manera segura (Amazon, 2024).

2.9.3. AWS Elastic Beanstalk

Este servicio permite la gestión de aplicaciones que facilita la implementación y escalado de aplicaciones web y servicios desarrollados en diversos lenguajes de programación, como Java, .NET, PHP, Node.js, Python, Ruby, Go, y Docker. Elastic Beanstalk maneja

automáticamente el despliegue, desde la provisión de capacidad, el balanceo de carga, el escalado automático, hasta la supervisión de la salud de la aplicación (Amazon, 2023).

2.10. Recursos Utilizados

Tabla 1

Dispositivos Utilizados

Para este proyecto se utilizó los siguientes componentes:

Cantidad	Componentes
1	Raspberry Pi 4
1	Sensor de temperatura DHT11
1	Pulsadores
1	Sensor Ultrasónico
3	Resistencias de 330Ω
1	Potenciómetro
1	Mcp 3008
1	Protoboard

Nota. Presentación de Componentes electrónicos utilizados en el proyecto monitorización de Dashboard con Rapsberry pi 4. Elaborado por: Los Autores.

2.10.1. Raspberry Pi 4

Este computador de placa única de bajo costo y tamaño reducido permite realizar tareas de Servidor, Proyectos de Hardware e IoT, tareas de Automatización, programación y alojamiento de robótica etc. Posee GPIO de 40 pines para expansión y control de hardware adicional, Wi-Fi 802.11ac, Bluetooth 5.0, Gigabit Ethernet, Puertos USB 3.0 y 2.0 así como 1 puerto Micro HDMI y almacenamiento de Mirco SD (Pantech elearning, 2021).

Figura 1

Raspberry Pi4

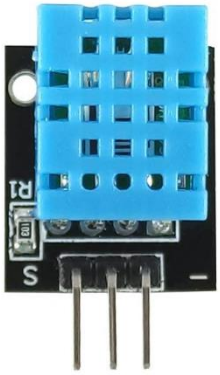
Nota. Componente electrónico Rapsberry Pi. Fuente: (Pantech elearning, 2021).

2.10.2. Sensor de temperatura DHT11

Es un sensor de temperatura del aire, posee tres pines de conexión, GND para la conexión de tierra, DATA, transmisión de datos y VCC, alimentación y una resistencia de 10k ohm (Agarwal, 2019).

Figura 2

Sensor de temperatura DHT11



Nota. Componente electrónico DHT11. Fuente: (Agarwal, 2019).

2.10.3. Pulsadores

Es un componente eléctrico utilizado para cerrar o abrir un circuito temporalmente cuando se presiona (UNIT Electronics, 2024).

Figura 3

Pulsador

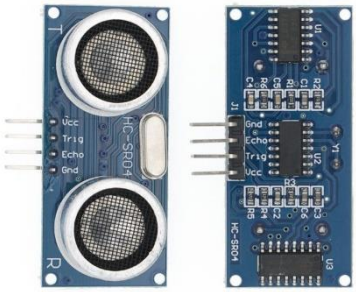
Nota. Componente electrónico Pulsador. *Fuente: (UNIT Electronics, 2024).*

2.10.4. Sensor Ultrasónico

Es un dispositivo que utiliza ondas sonoras de alta frecuencia para medir distancias y detectar objetos (Sme, 2022).

Figura 4

Sensor de distancia Ultrasónico



Nota. Componente electrónico Ultrasónico. *Fuente: (Sme, 2022).*

2.10.5. Resistencias de 330Ω

Es un componente pasivo utilizado en circuitos electrónicos para limitar la corriente, dividir voltajes y proteger componentes sensibles (jotrin, 2023).

Figura 5

Resistencias de 330Ω

Nota. Componente electrónico Resistencias de 330Ω. Fuente: (jotrin, 2023).

2.10.6. Potenciómetro

Es un tipo de resistor variable que permite ajustar manualmente su resistencia para controlar diferentes aspectos de un circuito electrónico (Yida, 2019).

Figura 6

Potenciómetro

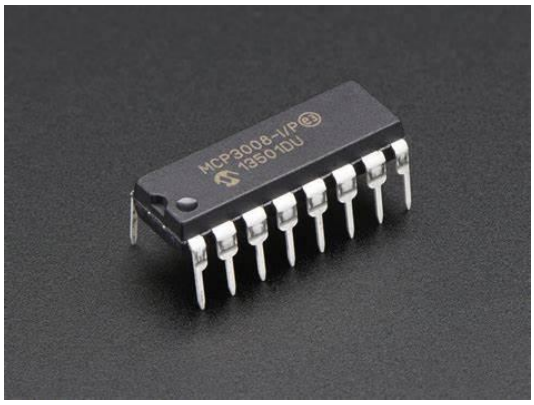
Nota. Componente electrónico Potenciómetro. Fuente: (Yida, 2019).

2.10.7. Mcp 3008

Es un convertidor analógico a digital (ADC) de 10 bits con 8 canales de entrada analógica, utilizado para adquisición de datos, control de motores, robótica, automatización industrial, sensores (Microchip, 2024).

Figura 7

Mcp 3008



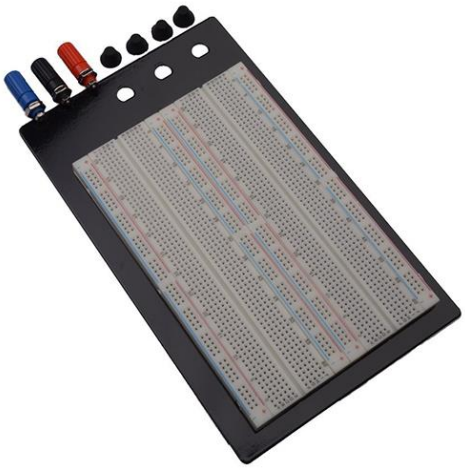
Nota. Componente electrónico Mcp 3008. Fuente: (Microchip, 2024).

2.10.8. Protoboard

Es una placa plástica con agujeros que permiten insertar componentes electrónicos fácilmente gracias a las (PCB) las placas de circuitos impresos para realizar un prototipo de un circuito electrónico (descubrearduino, 2020).

Figura 8

Protoboard



Nota. Componente electrónico Protoboard. Fuente: (descubrearduino, 2020).

2.10.9. Uso de los Componentes Electrónicos

Tabla 2

Uso de cada dispositivo

Componente	Uso	Medición
Raspberry Pi 4	Microordenador utilizado para ejecutar programas y controlar otros componentes electrónicos.	N/A
Sensor de temperatura DHT11	Medición de temperatura y humedad.	Temperatura y Humedad
Pulsadores	Permiten la interacción del usuario con el sistema, generalmente usados para iniciar o detener acciones.	Entrada digital (estado ON/OFF, 1 o 0)

Sensor Ultrasónico	Medición de distancias mediante ondas ultrasónicas.	Distancia (Ultrasónica)
Resistencias de 330Ω	Limitar la corriente en un circuito para proteger componentes electrónicos.	N/A
Potenciómetro	Varía la resistencia en un circuito, permitiendo el ajuste de voltaje y corriente.	Resistencia variable
Mcp 3008	Convertidor analógico a digital (ADC), utilizado para leer señales analógicas con microcontroladores que no tienen entradas analógicas.	Voltaje (ADC)
Protoboard	Plataforma para ensamblar prototipos de circuitos electrónicos de manera temporal y sin soldadura.	N/A

Nota. Presentación del uso dado de los Componentes electrónicos. Elaborado por: Los Autores.

2.11. Librerías

Tabla 3

Librerías Utilizadas

Librerías		
RASPBERRY		
Librería	**Versión**	**Descripción**
gpio adafruit	8.31.0	Permite interactuar con los pines GPIO del Raspberry pi para controlar hardware
requests	2.32.3	Usamos request para realizar peticiones HTTP de manera sencilla
gpiozero	2.0.1	Permite interactuar con los pines GPIO del Raspberry pi para controlar hardware
INTERCOMUNICADOR		
requests	2.32.3	Usamos request para realizar peticiones HTTP de manera sencilla
json	N/A	Es una herramienta estándar para trabajar con datos en formato JSON (JavaScript Object Notation)

cors	3.1.1	Es una extensión de Flask que permite configurar CORS en aplicaciones web construidas con Flask.
FRONT END JQUERY		
chart js	N/A	Es una biblioteca de JavaScript que permite crear gráficos interactivos y animados en aplicaciones web
NODE JS		
cors	2.8.5	Flask que permite configurar CORS en aplicaciones web construidas con Flask.
express	4.17.1	Librería que nos proporciona una amplia variedad de herramientas para el desarrollo Web
Aws-sdk	N/A	Es una biblioteca que nos permitirá interactuar con los servicios de Amazon Web Services desde aplicaciones JavaScript

Cognito-min-js	N/A	Es una biblioteca que facilita la integración de Amazon Cognito en aplicaciones web

Nota. Se indican las librerías utilizas y la versión de cada una de ellas. Elaborado por: Los Autores.

2.12. Factor de conversión

2.12.1. Potenciómetro

Se puede usar la relación $(V_{out}/V_{ref}) \times R$, donde V_{out} es el voltaje de salida del potenciómetro y V_{ref} es el voltaje de referencia.

En un ADC (Convertidor Analógico a Digital), si se tiene una lectura digital D y el ADC tiene n bits, la conversión es: ángulo=$\left(\frac{D}{2^n-1}\right) \times \text{Rango}$

2.12.2. DHT11

La conversión se realiza a nivel del protocolo de comunicación, donde los datos en bruto se convierten directamente a unidades físicas (°C para la temperatura y % para la humedad) mediante fórmulas específicas del sensor.

Generalmente, el DHT11 entrega los datos ya convertidos, por lo que no es necesario un factor de conversión adicional.

2.12.3. Ultrasonido

$$\text{Distancia} = \left(\frac{\textit{Tiempo de ida y vuelta}}{2}\right) \times \text{Velocidad del sonido.}$$

La fórmula comúnmente usada es: $\text{Distancia} = \frac{\text{Tiempo} \times 0.0343}{2}$ (donde 0.0343 cm/µs es la velocidad del sonido en el aire).

2.12.3. Mcp3008

Convierte señales analógicas a digitales con una resolución de 10 bits.

Factor de conversión: Si el valor digital es D y el voltaje de referencia es V_{ref}, la conversión es $V_{in} = \left(\frac{D}{1023}\right) \times V_{ref}$.

2.12.4. Pulsador

No requiere un factor de conversión complejo, pero puede haber una lógica para debouncing (eliminación de rebotes) en el software para interpretar correctamente los estados del pulsador.

CAPÍTULO III

METODOLOGIA

En esta sección se detallará el proceso del Dashboard y la integración de Raspberry Pie 4 para el proyecto, este proceso utiliza la metodología de tipo XP (Programación Externa), el cual la cual favorece un enfoque de desarrollo continuo, permitiendo la mejora constante del proyecto a medida que se cumplen los requisitos, la cual estaría estructurado en las siguientes fases:

- Investigación y Análisis
- Diseño del Sistema
- Desarrollo
- Pruebas

3.1. Fase 1 Investigación y Análisis

En esta fase se analiza la capacidad de Raspberry Pi 4 y sus capacidades de procesamiento para la monitorización de sensores para el desarrollo en backend , los cuales son los siguientes:

3.1.2. Desarrollo del Backend

Programar la lógica para recolectar, procesar y servir los datos de los sensores. Implementar una API REST para la comunicación entre el frontend y el backend.

3.1.3. Desarrollo del Frontend

Crear la interfaz web que consumirá los datos proporcionados por el backend. Utilizar bibliotecas de JavaScript como Chart.js o D3.js para implementar los widgets del dashboard.

3.1.4. Capacidades de los Componentes

Tabla 4

Procesamiento de Componentes

Capacidades de Procesamiento	
CPU	Broadcom BCM2711, Quad-core Cortex-A72 (ARM v8) 64-bit SoC a 1.5GHz.
GPU	VideoCore VI, que soporta salida 4K.
MEMORIA RAM	Disponibles en varias configuraciones de 2GB, 4GB, y 8GB LPDDR4-3200 SDRAM.

Nota. Tabla que indica las capacidades del procesamiento. Elaborado por: Los Autores.

Tabla 5

Conectividad de Componentes

Capacidades de Conectividad	
GPIO (General Purpose Input/Output):	40 pines GPIO que pueden usarse para conectar una variedad de sensores y módulos adicionales. Estos pines permiten la comunicación con dispositivos electrónicos y la implementación de diversas interfaces de hardware.
USB	2 puertos USB 3.0 y 2 puertos USB 2.0 que facilitan la conexión de dispositivos

	de alta velocidad como discos duros externos, cámaras, y otros periféricos.
ETHERNET	Gigabit Ethernet para una conectividad de red rápida y confiable, esencial para la transmisión de datos a través de redes locales o internet.
WI-FI	Conectividad Wi-Fi 802.11ac (2.4GHz y 5GHz) integrada, proporcionando opciones inalámbricas para comunicación de datos.
BLUETHOOTH	Bluetooth 5.0 para la conexión de dispositivos periféricos como teclados, ratones, y otros dispositivos Bluetooth.
HDMI	2 puertos micro-HDMI que soportan resoluciones hasta 4K, útiles para la visualización de datos y desarrollo de interfaces gráficas.

Nota. Tabla que indica las capacidades de conectividad. Elaborado por: Los Autores.

3.1.5. Soporte de Software

La Raspberry Pi 4 es compatible con una amplia variedad de sistemas operativos y entornos de desarrollo, con un fuerte soporte para Python, lo que la hace ideal para el desarrollo de backends:

Sistema Operativo: Raspbian (ahora conocido como Raspberry Pi OS), una distribución basada en Debian optimizada para el hardware de la Raspberry Pi. Otros sistemas operativos como Ubuntu, Windows 10 IoT Core, y varias versiones de Linux también son compatibles.

3.1.6. Análisis de requerimientos de monitoreo

Se define los siguientes tipos de datos analógicos y digitales para la monitorización las cuales son los siguientes sensores:

Tabla 6

Tipos de datos

Tipo de dato	Sensor	Descripción
Analógicos	Potenciómetro	un componente electrónico variable que consta de tres terminales.
	Temperatura	utiliza la variación de resistencia de un material conductor con la temperatura para medir la temperatura.
	Pulsador	Interruptores de botón, generan una señal digital (encendido o apagado) que indica si el botón está presionado o no.

		sensor de temperatura que utiliza la variación de resistencia de un semiconductor con la temperatura para medir la temperatura
	Termistores	
	Presión de galgas extensométricas	Mide la presión de la deformación de una galga
	Células de carga	utiliza la deformación de un elemento elástico para medir la fuerza
	Proximidad	emitiendo un campo electromagnético o un haz de radiación (infrarrojo, ultrasonido, etc.) y detectando cambios en el campo o la señal reflejada
Digitales	Ultrasonido	utiliza ondas de sonido de alta frecuencia para medir la distancia a un objeto.
	Codificadores rotativos	utiliza un disco codificado y un fotodetector para medir la posición angular de un eje rotativo.

	utiliza el efecto Hall
Efecto Hall	para detectar la presencia de un campo magnético

Nota. Se indican los tipos de datos. Elaborado por: Los Autores.

3.1.7. Limitación

La arquitectura está diseñada para recibir datos analógicos y digitales, debido al uso del MCP3008. Este ADC (convertidor analógico a digital) no puede recibir datos en forma de bits directamente, lo que implica que solo puede transmitir datos digitales o discretos. Por lo tanto, si se desean conectar sensores como el DHT11, sería necesario programar su biblioteca directamente en el backend para manejar correctamente sus datos.

3.1.8. Requisitos específicos

3.1.8.1. Requisitos funcionales

Requerimiento Funcional 1	
Nombre	Captura de Datos de Sensores
Descripción	Captura los datos analógicos y digitales provenientes de diversos sensores conectados a la Raspberry Pi 4.

Requerimiento Funcional 2	
Nombre	Sistema de monitoreo
Descripción	Se diseño e implemento un sistema de monitoreo de datos analógicos y digitales tanto en local (simulador de

	datos) así como en línea (mediante AWS) para la Raspberry Pi 4.

Requerimiento Funcional 3	
Nombre	Procesamiento de Datos
Descripción	Se procesan los datos de los sensores de manera eficiente y precisa utilizando el ADC MCP3008 para la conversión de señales analógicas a digitales.

Requerimiento Funcional 4	
Nombre	Visualización de Datos
Descripción	Se diseño una interfaz de usuario (dashboard) que permita visualizar los datos de los sensores en forma de gráficos, medidores, y otros widgets.

Requerimiento Funcional 5	
Nombre	Actualización Automática de Datos
Descripción	Se actualizan los datos automáticamente mostrándolos en la interfaz de usuario a intervalos regulares,

	asegurando que la información presentada sea siempre actual.

Requerimiento Funcional 6	
Nombre	Configuración de Sensores
Descripción	Configuración y calibración de los sensores conectados, incluyendo ajustes como el factor de conversión para sensores analógicos.

Requerimiento Funcional 7	
Nombre	Gráficos dobles
Descripción	Se muestra un gráfico doble que represente simultáneamente la temperatura y la humedad medida por el sensor DHT11.

Requerimiento Funcional 8	
Nombre	Personalización de las Graficas
Descripción	Permite a los usuarios modificar el diseño, los colores y los tipos de gráficos para adaptarlos a sus preferencias.

Requerimiento Funcional 9	
Nombre	Mostar 4 gráficos principales
Descripción	Presentar los cuatro gráficos principales, tacómetro, texto, gráfico y gauge.

Requerimiento Funcional 10	
Nombre	Conectar Sensores que requieran librerías
Descripción	Integrar sensores que necesiten librerías para su funcionamiento, asegurando la compatibilidad y el correcto procesamiento de datos.

Requerimiento Funcional 11	
Nombre	Conectar Sensores básicos estándar
Descripción	Facilitar la conexión de sensores comunes y estándar sin necesidad de configuraciones adicionales.

Requerimiento Funcional 12	
Nombre	Quitar, cambiar y agregar Grafica
Descripción	Ofrecer la capacidad de eliminar, modificar o añadir nuevos gráficos según los requisitos del usuario.

Requerimiento Funcional 13	
Nombre	Factor de Conversión
Descripción	Implementar factores de conversión para ajustar y representar correctamente las unidades de medida de los datos obtenidos de los sensores.

3.1.8.2. Requisitos No funcionales

Requerimiento No Funcional 1	
Nombre	Disponibilidad
Descripción	El sistema debe estar disponible para el acceso y uso de los usuarios finales al menos el 99% del tiempo, excluyendo períodos de mantenimiento programado.

Requerimiento No Funcional 2	
Nombre	Visualización en tiempo real
Descripción	El sistema debe poder mostrar los datos de los sensores de manera inmediata y actualizada según los datos se capturan. El sistema posee un retraso de 3 segundos y para su uso en tiempo real se debe pagar un servicio de AWS.

Requerimiento No Funcional 3	
Nombre	Almacenamiento de los datos obtenidos
Descripción	Los datos almacenados no son accesibles para análisis histórico, reportes y otras formas de procesamiento posterior.

Requerimiento No Funcional 4	
Nombre	Respaldo y recuperación de datos
Descripción	No posee capacidades robustas de respaldo y recuperación de datos en caso de fallas o pérdida de información.

Requerimiento No Funcional 5	
Nombre	Internacionalización y localización
Descripción	Diseñar el sistema para que sea fácilmente adaptable a diferentes idiomas y formatos regionales.

3.2. Fase 2: Diseño del Sistema

3.2.1. Arquitectura del Sistema

Diseñar la arquitectura general del sistema, incluyendo el servidor en la Raspberry Pi, la conexión con los sensores, y la interfaz web para visualización. Seleccionar tecnologías adecuadas para el backend (como Flask o Django) y para el frontend (HTML/CSS/JavaScript).

3.2.2. Diseño del Dashboard

Esbozar el diseño del dashboard, decidiendo qué widgets se incluirán para representar los diferentes tipos de datos, como gráficos para tendencias, tacómetros para velocidades, y medidores para valores instantáneos.

3.2.3. Prototipo de interfaces

Figura 9

Diseño del Login

Pantalla de inicio de sesión con campos para “NOMBRE DE USUARIO” y “CONTRASEÑA”. Dos botones, “INICIAR SESIÓN” y “REGISTRARSE”

Nota. Diseño del Login. Elaborado por: Los autores.

Figura 10

Diseño del Registro

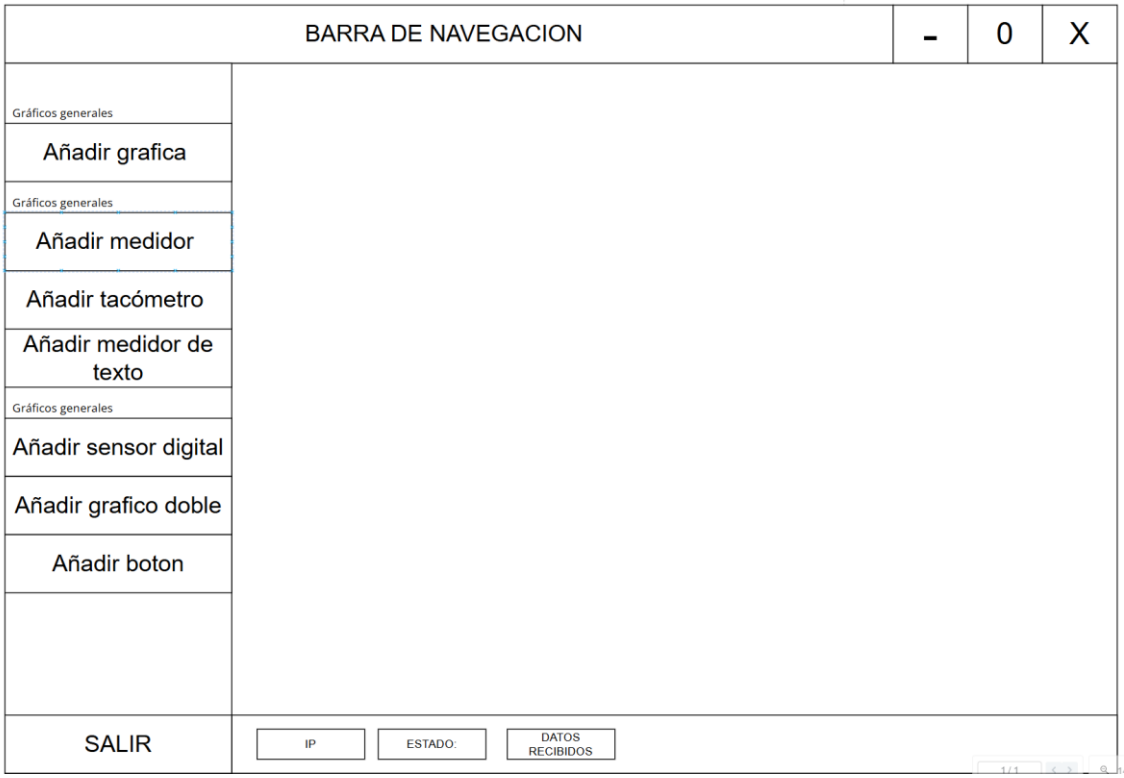
Interfaz gráfica de usuario para una página de registro. Incluye campos para “E-MAIL” y “CONTRASEÑA”, así como botones para “REGISTRARSE”.

Nota. Diseño del Registro. Elaborado por: Los autores.

Figura 11

Diseño del Dashboard

Interfaz gráfica de usuario para agregar elementos como gráficas, medidores de texto, tacómetros y botones a una plataforma digital. Además del botón de “SALIR” para cerrar la interfaz.

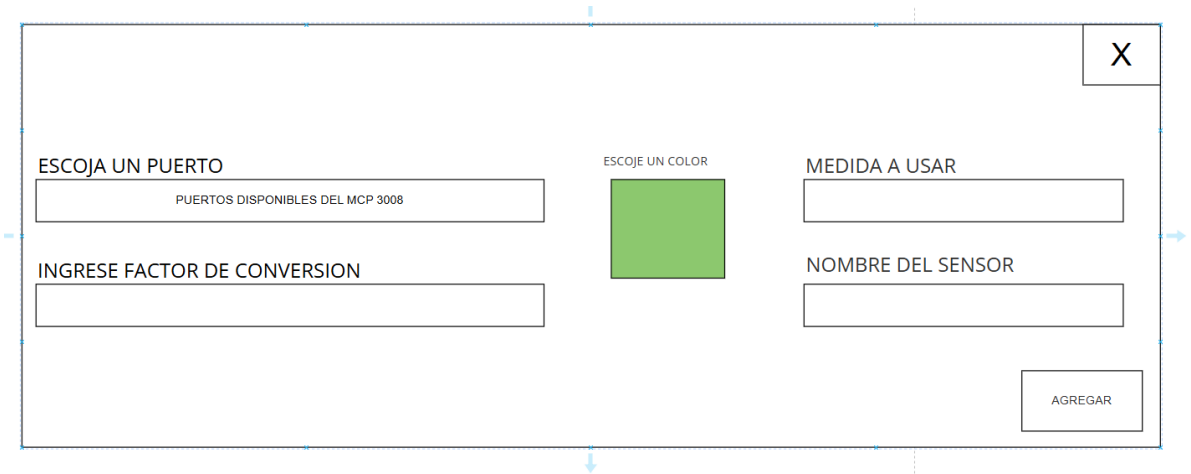


Nota. Diseño del Dashboard. Elaborado por: Los autores.

Figura 12

Ajustes Personalización del Diseño Grafico

Interfaz gráfica de usuario con opciones para seleccionar puertos, ingresar factores de conversión, elegir colores y agregar mediciones de sensores.

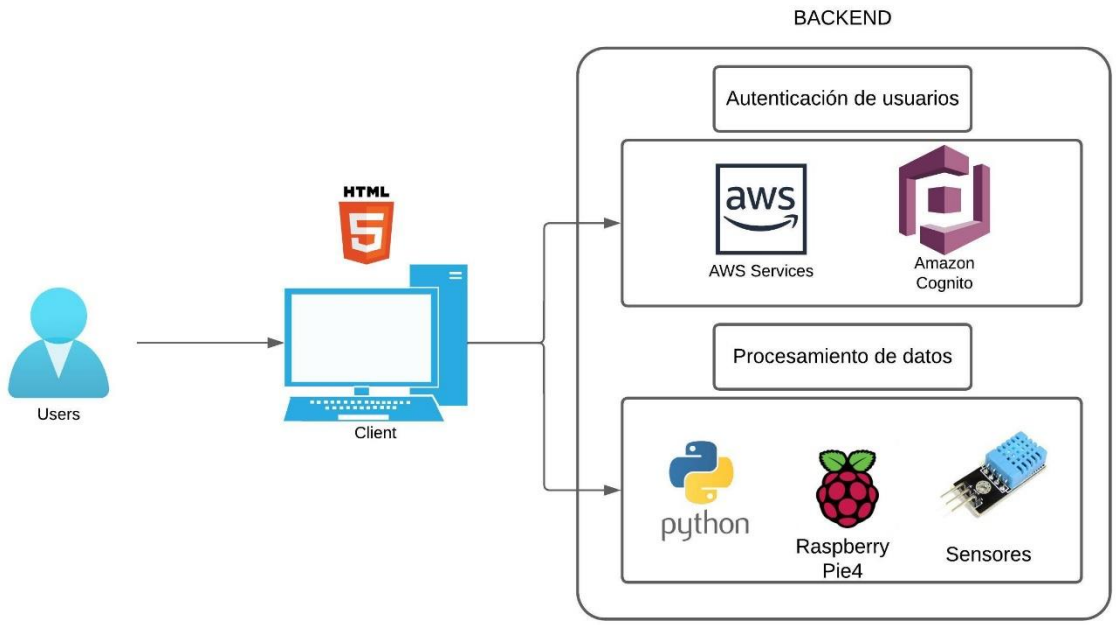


Nota. Diseño de la personalización grafica. Elaborado por: Los autores.

3.2.4. Diseño de Arquitectura del sistema

Figura 13

Diagrama Arquitectura Dashboard



Nota. Diagrama Arquitectura del sistema Dashboard. Elaborado por: Los autores.

En este diseño se muestra la arquitectura del sistema de un Dashboard enfocado en la monitorización de datos de sensores, donde el FrontEnd está realizado con HTML, CSS y JAVASCRIPT para las interacciones con los usuarios en las cuales podrán ingresar y registrarse.

Para el Backend se divide en las siguientes secciones:

Autenticación de usuarios: Utilizando Amazon Cognito, un servicio de AWS, para gestionar la autenticación y autorización de los usuarios. Una de las principales características

de Amazon Cognito es que permite integrar funcionalidades de registro, inicio de sesión y gestión de usuarios de forma segura.

Procesamiento de datos: Implementado con Python, ejecutándose en un Raspberry Pi 4. El Raspberry Pi 4 está conectado a varios sensores que recopilan datos. Los datos de los sensores son procesados y enviados al backend.

3.3. Fase 3: Desarrollo

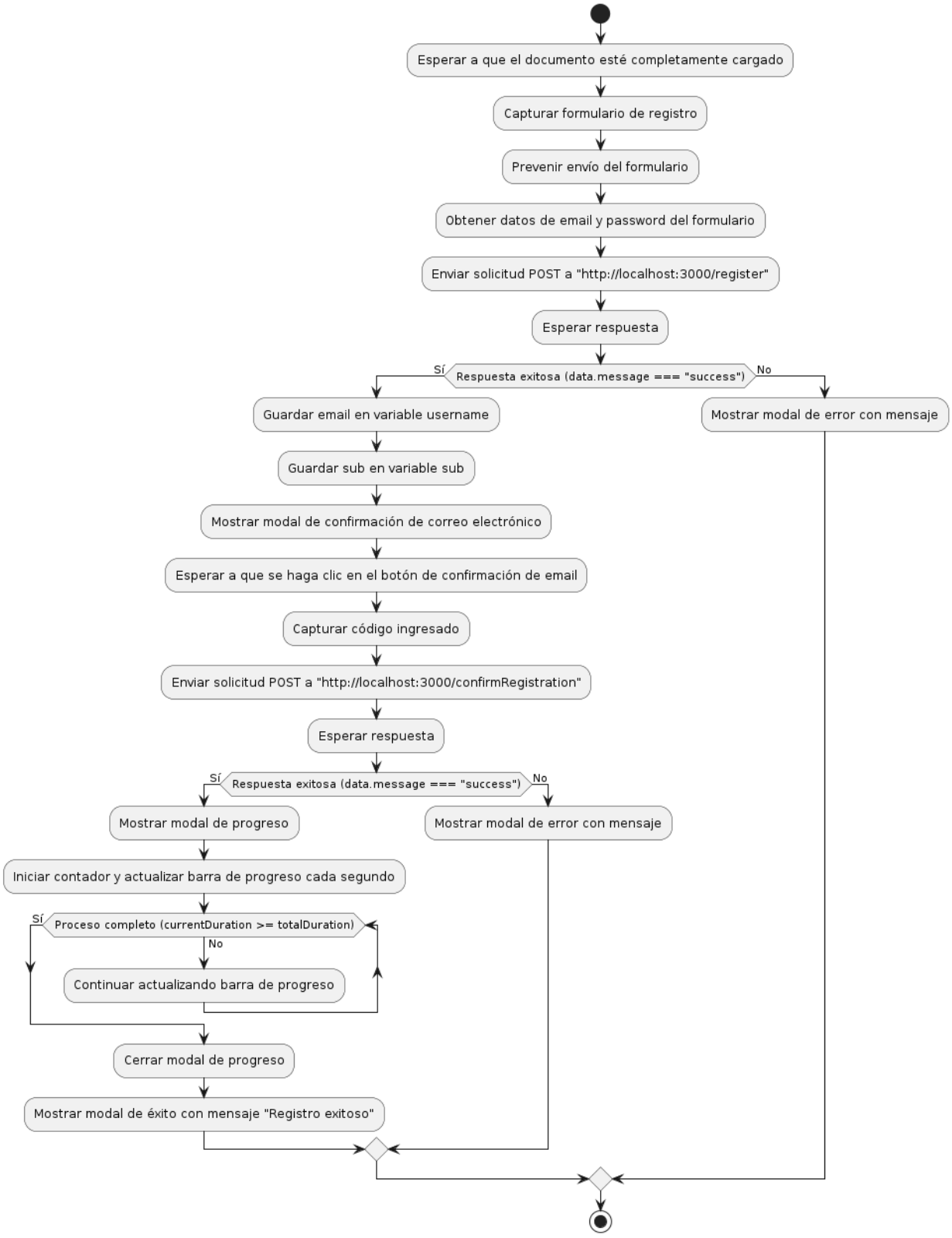
Configuración del Entorno de Desarrollo: Preparar la Raspberry Pi con el sistema operativo adecuado, instalar Python y las bibliotecas necesarias para el desarrollo web.

Desarrollo del Backend: Programar la lógica para recolectar, procesar y servir los datos de los sensores. Implementar una API REST para la comunicación entre el frontend y el backend.

Desarrollo del Frontend: Crear la interfaz web que consumirá los datos proporcionados por el backend. Utilizar bibliotecas de JavaScript como Chart.js o D3.js para implementar los widgets del dashboard.

Figura 14

Diagrama Registro



Nota. Diagrama del registro. Elaborado por: Los autores.

Primero, vamos a manejar las solicitudes POST en la ruta "/login". Aquí esperamos que el usuario envíe datos a través del cuerpo de la solicitud (req.body), incluyendo el correo electrónico y la contraseña. Estos datos se usarán para crear un objeto llamado "authenticationData", que contendrá la información de ambos campos. Posteriormente, se creará otro objeto llamado "userData", necesario para autenticar al usuario.

Después de preparar estos datos, se procederá a crear un usuario en Cognito utilizando el objeto "userData". Esto permitirá interactuar con usuarios de grupos específicos dentro del sistema. Una vez que la autenticación sea exitosa, se ejecutará la función "onSuccess" para confirmar la autenticación en la consola. Además, se generará un identificador llamado "usersub", basado en el token de identificación del usuario.

Finalmente, se prepararán los datos "params" para configurar la aplicación con el nombre y el entorno relacionados con el usuario autenticado. Luego, se utilizará "elasticbeanstalk.describeEnvironments(params, function(err, data) { ... })" para obtener detalles del entorno de Elastic Beanstalk usando los parámetros proporcionados.

En caso de que ocurra un error durante este proceso, se registrará en la consola y se enviará una respuesta de error al usuario. Si los datos se obtienen correctamente (es decir, si "data" está disponible), se verificará la existencia de entornos (data.Environments). Se seleccionará el primer entorno disponible y se verificará que su estado de salud ("Health") sea "Green" (activo y saludable).

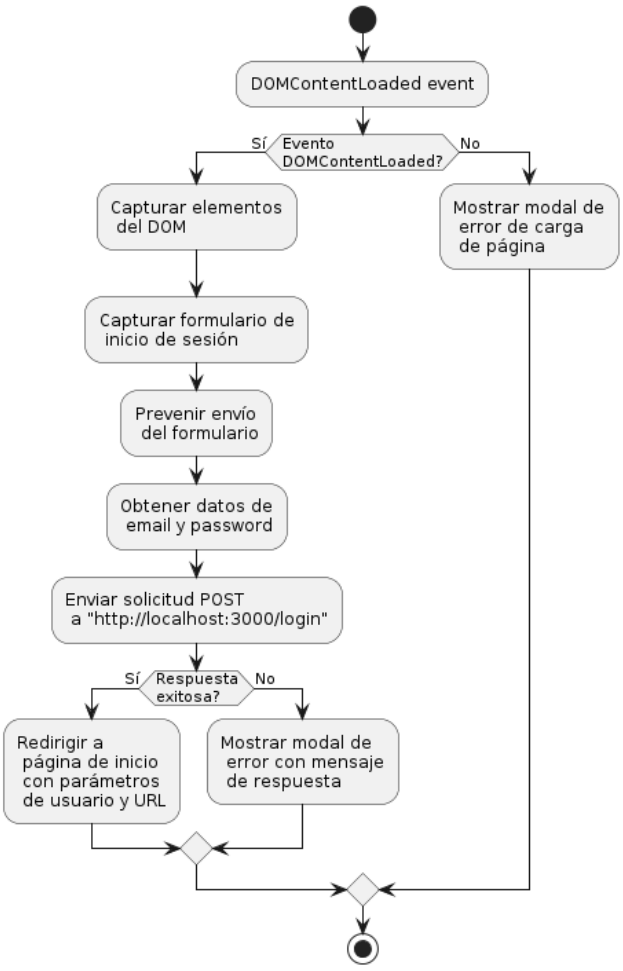
Si el entorno está activo y saludable, se preparará un objeto "datos" con un mensaje de éxito, el correo electrónico del usuario y la URL del punto final del entorno (environment.EndpointURL). Finalmente, se enviará una respuesta JSON al usuario con estado HTTP 200, que incluirá estos datos. Si el entorno no está disponible, se informará al usuario que el entorno aún no está listo.

En caso de que no se encuentren detalles de entorno en la respuesta (data.Environments no existe o está vacío), se devolverá un mensaje de error al cliente con estado HTTP 404 indicando que no se encontraron detalles del entorno.

Para manejar fallos de autenticación, si la autenticación falla (onFailure), se registrará el error en la consola y se enviará una respuesta de error al cliente con estado HTTP 401. Esta respuesta contendrá un mensaje específico del error de autenticación o uno genérico, según corresponda.

Figura 15

Diagrama LOGIN



Nota. Diagrama del Login. Elaborado por: Los autores.

Tenemos la ruta POST "/register", donde vamos a recibir una solicitud POST desde un usuario, generalmente con los datos de email y contraseña desde el "req.body" (el cuerpo de la solicitud). Después de eso, se extraerán los campos "email" y "password" del cuerpo de la solicitud para luego preparar los atributos del usuario. Amazon Cognito requiere que especifiquemos un atributo para registrar al usuario; en este caso, se preparará un atributo para el correo electrónico del usuario.

La variable "attributeList" es una matriz que contendrá los atributos del usuario. Inicializaremos una lista vacía "attributeList" para almacenar los atributos del usuario. Se define un objeto "dataEmail" que contiene el nombre del atributo ("email") y el valor (el email proporcionado por el usuario). Luego, crearemos un objeto llamado "attributeEmail" usando "CognitoUserAttribute". Este objeto representa el atributo de correo electrónico del usuario y se agrega a "attributeList".

Pasamos al registro del usuario en Amazon Cognito, donde "userPool.signUp()" es el método usado para registrar al usuario en el grupo de usuarios de Cognito. Este método toma como argumentos el correo electrónico, la contraseña, "attributeList" (que contiene el atributo del correo electrónico) y parámetros opcionales, como null.

En caso de un error durante el registro (err), se envía una respuesta de error al cliente con estado HTTP 400 y un mensaje explicativo.

Si el registro es exitoso (result contiene información del usuario registrado), "result.user" representa al usuario registrado en Amazon Cognito y "result.userSub" es el identificador único (sub) generado para el usuario registrado. Finalmente, se envía una respuesta al cliente con estado HTTP 200 indicando éxito y devolviendo el identificador único (sub) junto con un mensaje de confirmación.

3.3.1. Clonar Enviroment

Primero, vamos a manejar las solicitudes POST en la ruta "/confirmRegistration". Aquí esperamos que el usuario envíe datos a través del cuerpo de la solicitud (req.body), incluyendo el correo electrónico, el código de confirmación y el identificador único (sub). Estos datos se usarán para crear un objeto userData, que contendrá la información necesaria para confirmar la cuenta del usuario en Cognito.

Después de preparar estos datos, se procederá a crear un usuario en Cognito utilizando el objeto userData. Esto permitirá interactuar con usuarios de grupos específicos dentro del sistema. Se creará un objeto CognitoUser usando userData para representar al usuario en Amazon Cognito.

Luego, se llamará al método confirmRegistration del objeto CognitoUser, pasando el código de confirmación (code). Si hay un error durante la confirmación (err), se enviará una respuesta de error al cliente con estado HTTP 400 y un mensaje explicativo. Si la confirmación es exitosa, se continuará con el proceso.

En caso de éxito, se recorta el identificador único del usuario (sub) a los primeros 20 caracteres y se almacena en userSub. Se registra en la consola que la confirmación fue exitosa.

Posteriormente, se utilizará elasticbeanstalk.describeConfigurationSettings para obtener la configuración del entorno en Elastic Beanstalk, usando los parámetros proporcionados. En caso de que ocurra un error durante este proceso, se registrará en la consola y se notificará al cliente. Si la operación es exitosa y se obtienen los datos correctamente (configData), se procederá a clonar el entorno utilizando elasticbeanstalk.createEnvironment.

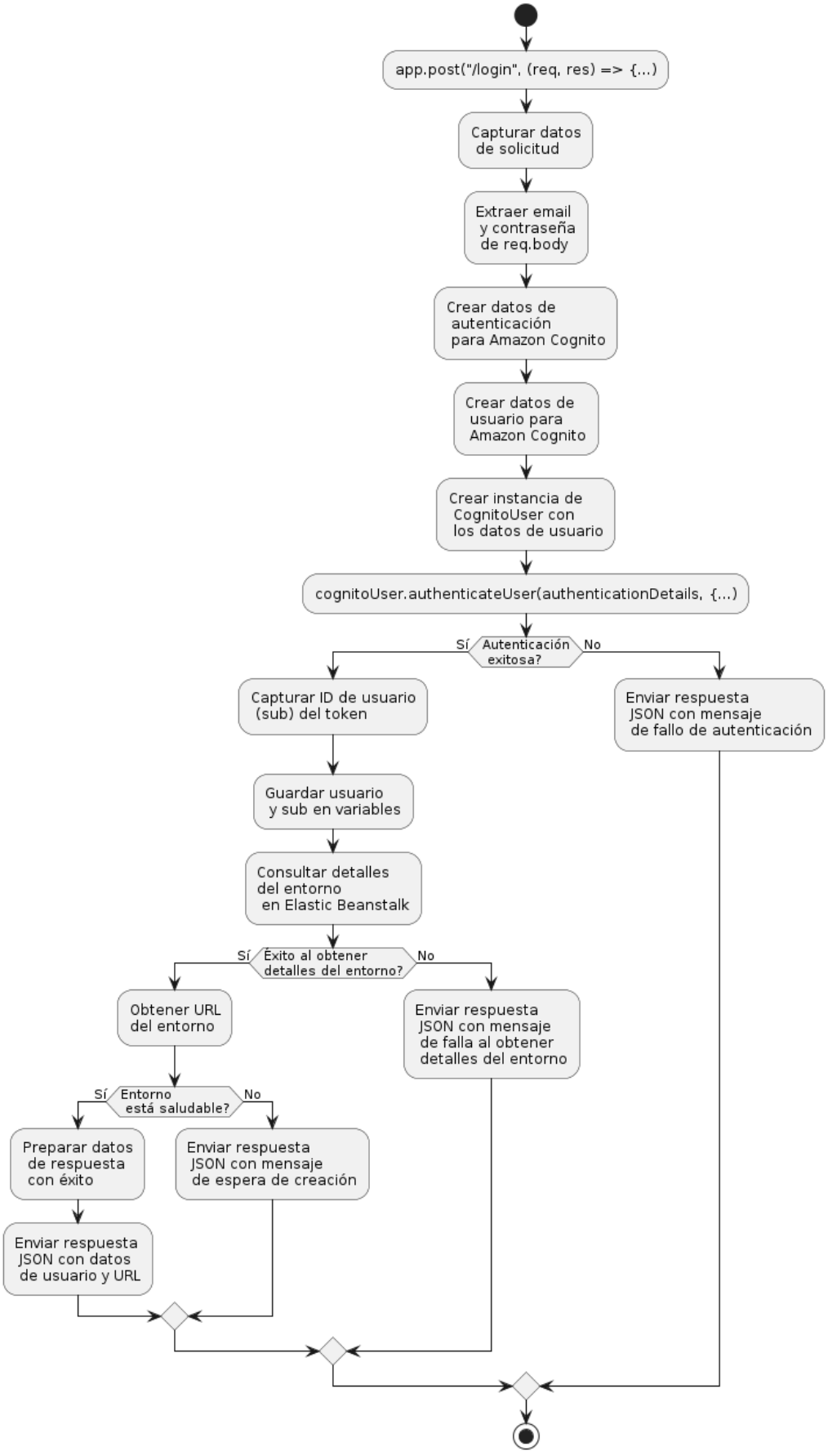
Para clonar el entorno, se llama al método createEnvironment de Elastic Beanstalk con los parámetros necesarios, incluyendo el nombre de la aplicación (ApplicationName), el nombre del nuevo entorno (EnvironmentName), la plantilla de configuración (TemplateName)

y la etiqueta de la versión (VersionLabel). Si hay un error durante la clonación del entorno (err), se registrará en la consola. Si la operación es exitosa, se enviará una respuesta JSON al cliente con estado HTTP 200 y un mensaje de éxito.

Finalmente, se inicia el servidor para escuchar en el puerto especificado y se registra un mensaje en la consola para indicar que el servidor está en funcionamiento.

Figura 16

Diagrama Login del server


app.post("/login", (req, res) => {...)
Capturar datos de solicitud
Extraer email y contraseña de req.body
Crear datos de autenticación para Amazon Cognito
Crear datos de usuario para Amazon Cognito
Crear instancia de CognitoUser con los datos de usuario
cognitoUser.authenticateUser(authenticationDetails, {...)
Autenticación exitosa?
Sí
No
Capturar ID de usuario (sub) del token
Enviar respuesta JSON con mensaje de fallo de autenticación
Guardar usuario y sub en variables
Consultar detalles del entorno en Elastic Beanstalk
Éxito al obtener detalles del entorno?
Sí
No
Obtener URL del entorno
Enviar respuesta JSON con mensaje de falla al obtener detalles del entorno
Entorno está saludable?
Sí
No
Preparar datos de respuesta con éxito
Enviar respuesta JSON con mensaje de espera de creación
Enviar respuesta JSON con datos de usuario y URL

Nota. Diagrama del Login del server. Elaborado por: Los autores.

Primero, el código usa document.addEventListener("DOMContentLoaded", (event) => { ... }); para asegurar que el DOM esté completamente cargado antes de ejecutar el script. Se definen varias variables para manejar elementos del DOM, como progressBar, progressModal, messageModal, modalMessage, closeMessageModalButton, confirmButton, username y sub.

Luego, se agrega un evento submit a un formulario con la clase .sign-up-form. Este evento se encarga de capturar los datos del formulario (correo electrónico y contraseña) y evitar el envío predeterminado del formulario (event.preventDefault()). Los datos se recogen en el objeto data.

Posteriormente, se envía una solicitud fetch a la ruta /register del servidor con el método POST, enviando los datos del formulario en formato JSON. La respuesta del servidor se maneja con .then((response) => response.json()), verificando si el mensaje en los datos de respuesta es "success". Si el registro es exitoso, se almacena el correo electrónico en username y el identificador único en sub, y se muestra el modal de confirmación de correo electrónico llamando a showModal(confirmEmailModal). Si hay un error, se muestra un mensaje de error usando showModalWithMessage.

La función showModal(modal) se utiliza para mostrar un modal, y closeModal(modal) para ocultarlo. showModalWithMessage(message, alertClass) se usa para mostrar un mensaje específico en el modal messageModal con una clase de alerta determinada (alertClass).

Se agrega un evento click al botón de cierre del modal de mensajes (closeMessageModalButton) para cerrar el modal. También se agrega un evento click al botón de confirmación (confirmButton) que maneja la confirmación del registro. Al hacer clic en el botón de confirmación, se recoge el código de confirmación del input correspondiente, junto

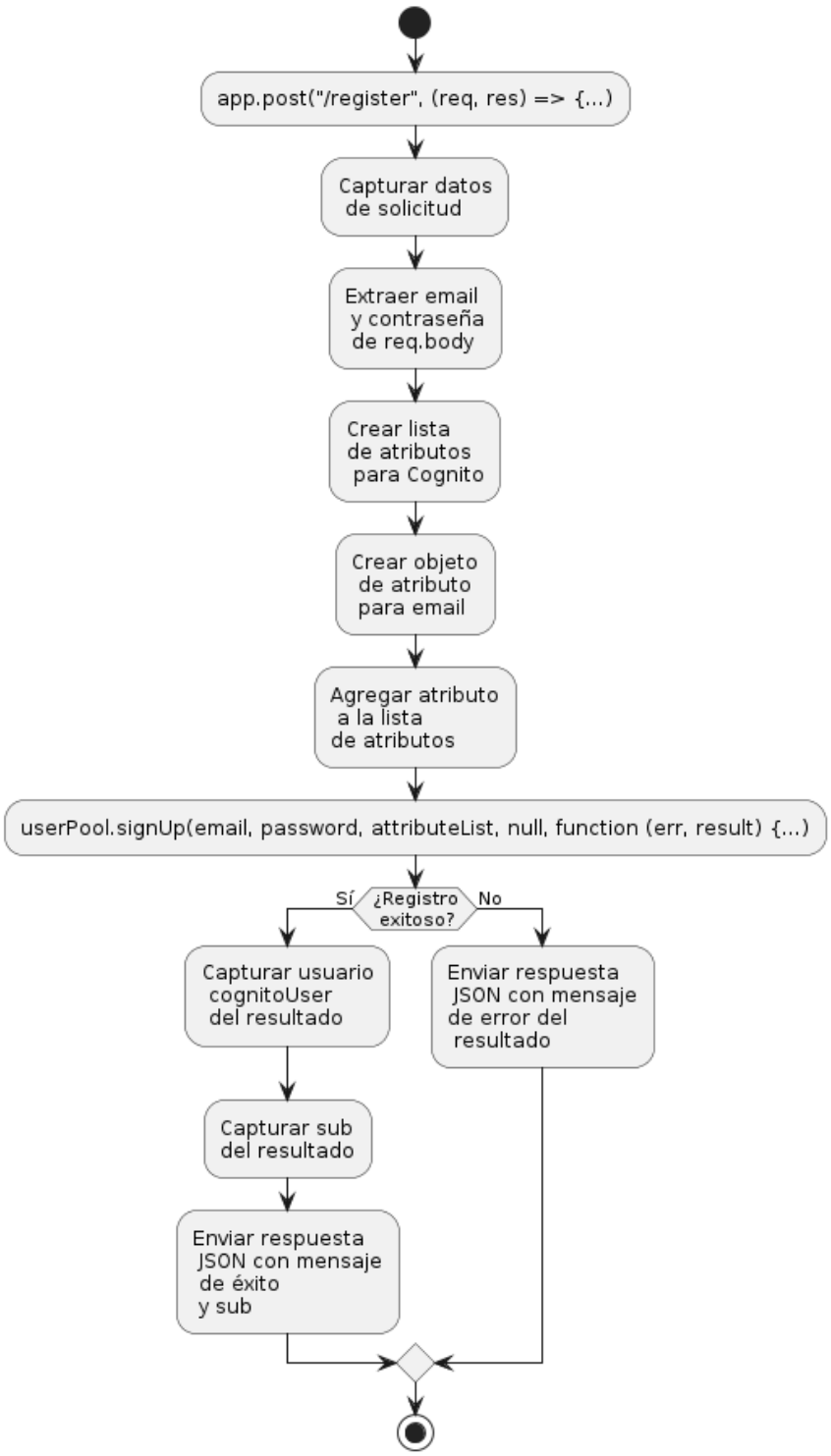
con el correo electrónico y el identificador único, y se envía una solicitud fetch a la ruta /confirmRegistration del servidor con el método POST.

La respuesta de esta solicitud se maneja similarmente, verificando si el mensaje es "success". Si la confirmación es exitosa, se cierra el modal de confirmación, se muestra el modal de progreso (progressModal), y se inicia un proceso de creación simulado con una duración total de 10 segundos, actualizando una barra de progreso cada segundo. Si el proceso de creación simulado finaliza correctamente, se cierra el modal de progreso y se muestra un mensaje de éxito.

Si ocurre un error en cualquier parte del proceso, se captura y se muestra un mensaje de error adecuado.

Figura 17

Diagrama Registro server



Nota. Diagrama Registro del server. Elaborado por: Los autores.

Primero, el código usa document.addEventListener("DOMContentLoaded", (event) => { ... }); para asegurar que el DOM esté completamente cargado antes de ejecutar el script.

Se definen varias variables para manejar elementos del DOM, como errorModal, confirmEmailModal, closeErrorModalButton y confirmButton.

Luego, se agrega un evento submit a un formulario con la clase .sign-in-form. Este evento se encarga de capturar los datos del formulario (correo electrónico y contraseña) y evitar el envío predeterminado del formulario (event.preventDefault()). Los datos se recogen en el objeto data.

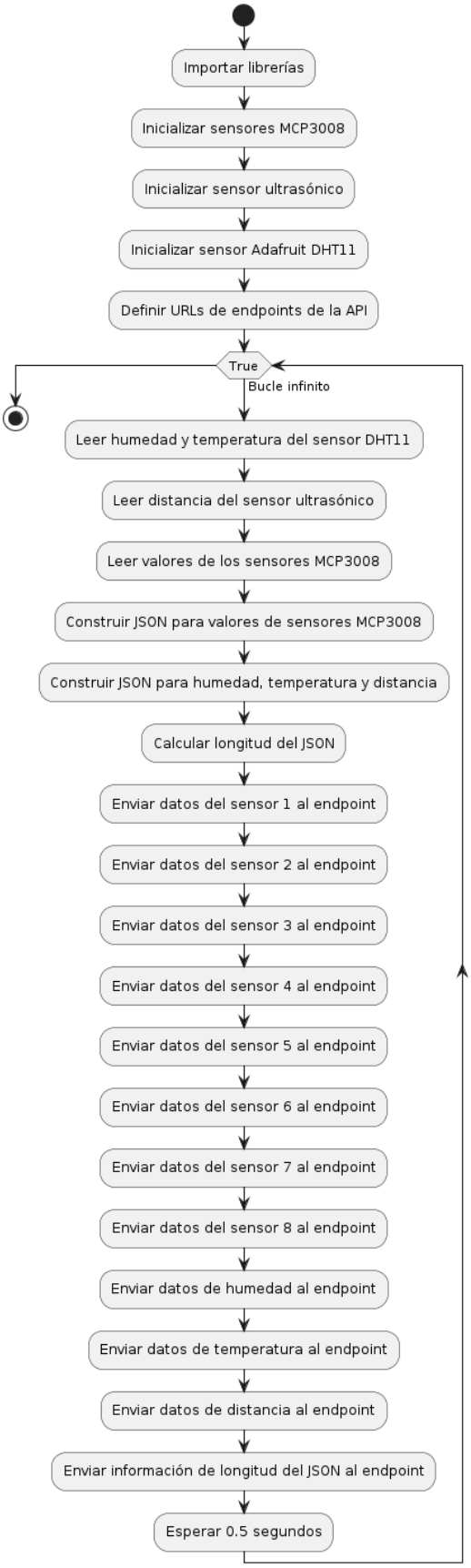
Posteriormente, se envía una solicitud fetch a la ruta /login del servidor con el método POST, enviando los datos del formulario en formato JSON. La respuesta del servidor se maneja con .then((response) => response.json()), verificando si el mensaje en los datos de respuesta es "success". Si el inicio de sesión es exitoso, el usuario es redirigido a una nueva URL que incluye parámetros en la consulta (user y environmentURL) derivados de la respuesta del servidor. Si hay un error, se muestra un mensaje de error usando showModalWithError.

La función showModalWithError(errorMessage) se utiliza para mostrar un mensaje de error en el modal errorModal. Se encuentra el elemento modalMessage y se establece su contenido de texto en el mensaje de error, luego se muestra el modal llamando a showModal(errorModal).

La función showModal(modal) se utiliza para mostrar un modal, y closeModal(modal) para ocultarlo. Un evento click se agrega al botón de cierre del modal de error (closeErrorModalButton) para cerrar el modal cuando el botón es presionado.

Figura 18

Diagrama backend


Importar librerías
Inicializar sensores MCP3008
Inicializar sensor ultrasónico
Inicializar sensor Adafruit DHT11
Definir URLs de endpoints de la API
True
Bucle infinito
Leer humedad y temperatura del sensor DHT11
Leer distancia del sensor ultrasónico
Leer valores de los sensores MCP3008
Construir JSON para valores de sensores MCP3008
Construir JSON para humedad, temperatura y distancia
Calcular longitud del JSON
Enviar datos del sensor 1 al endpoint
Enviar datos del sensor 2 al endpoint
Enviar datos del sensor 3 al endpoint
Enviar datos del sensor 4 al endpoint
Enviar datos del sensor 5 al endpoint
Enviar datos del sensor 6 al endpoint
Enviar datos del sensor 7 al endpoint
Enviar datos del sensor 8 al endpoint
Enviar datos de humedad al endpoint
Enviar datos de temperatura al endpoint
Enviar datos de distancia al endpoint
Enviar información de longitud del JSON al endpoint
Esperar 0.5 segundos

Nota. Diagrama del Backend. Elaborado por: Los autores.

Primero, importamos las librerías necesarias para manejar los sensores y realizar solicitudes HTTP. Utilizamos gpiozero para los sensores PWMLED, MCP3008, y DistanceSensor; requests para enviar datos a través de HTTP; time y sleep para manejar los tiempos de espera; y Adafruit_DHT para el sensor de temperatura y humedad DHT11.

Luego, configuramos las variables para el sensore MCP3008. Esto nos permite leer valores analógicos desde distintos pines numerados del 0 al 7, almacenándolos en las variables pot1 a pot8.

Configuramos el sensor de distancia ultrasónico asignando los pines correspondientes para el eco y el trigger. Utilizamos el pin 27 para el eco y el pin 17 para el trigger, almacenando esta configuración en la variable ultrasonic.

A continuación, configuramos el sensor de temperatura y humedad DHT11, y especificamos el pin que usaremos para leer los datos, que en este caso es el pin 4. Esta configuración se almacena en la variable sensor.

Definimos las URLs a las que enviaremos los datos de los sensores. Estas URLs corresponden a endpoints en un servidor Elastic Beanstalk. Por ejemplo, sensor1, sensor2, y así sucesivamente, hasta sensor11, cada uno representando un sensor diferente. La variable a se usa para enviar información general sobre el tamaño de los datos.

Dentro del bucle while True, leemos los valores de los sensores DHT11 y el sensor ultrasónico. Los valores de humedad y temperatura se obtienen utilizando Adafruit_DHT.read_retry(sensor, pin), y la distancia se mide con ultrasonic.distance.

Luego, construimos un diccionario para cada valor de sensor MCP3008, asignando el valor leído a la clave valor_sensor. Estos diccionarios son sensor_value1 a sensor_value8.

Para los sensores no MCP3008, también construimos diccionarios con el mismo formato: sensor_value9 para la humedad, sensor_value10 para la temperatura, y sensor_value11 para la distancia.

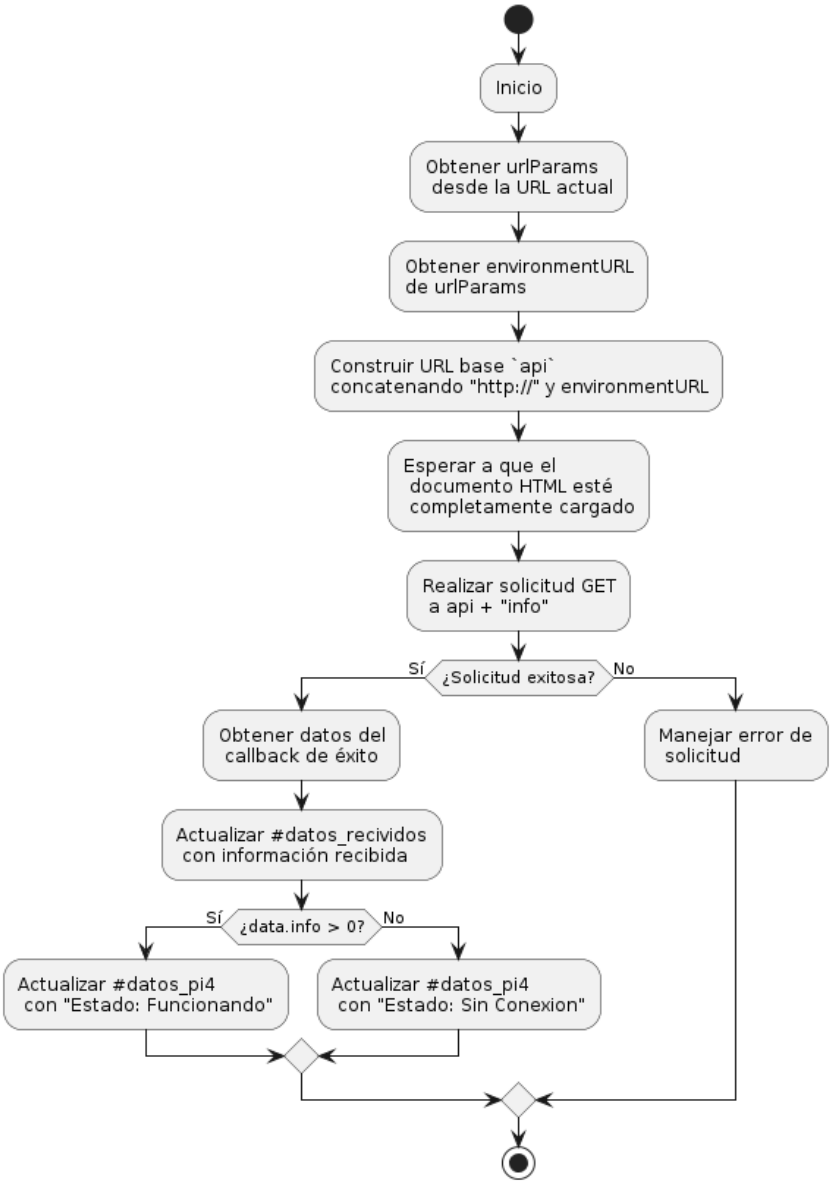
Unimos todos estos diccionarios en una cadena JSON, la convertimos a bytes, y calculamos su longitud. Esta información se almacena en el diccionario info, con la clave info y el valor correspondiente a la longitud de los bytes JSON.

Usamos la librería requests para enviar los datos de cada sensor a sus respectivos endpoints mediante solicitudes POST. Cada sensor MCP3008 tiene su propio endpoint, al igual que los sensores DHT11 y ultrasónico. Finalmente, enviamos la información general info al endpoint a.

Después de enviar los datos, el programa espera 0.5 segundos antes de repetir el proceso, permitiendo que los datos se actualicen continuamente.

Figura 19

Diagrama Obtención De Parámetros



Nota. Diagrama de la obtención de parámetros. Elaborado por: Los autores.

Primero, el script encargado de obtener los parámetros de la url usando “URLSearchParams” para extraer los avalores del “enviromentURL” y “user”. Estor parámetros se usarán para un figurar la url base de la API (“api”) y del usuario (“user”).

Dentro de “$(document).ready(function () { ... });,” de asegura de que el DOM este completamente cargado antes de ejecutar el código, se realizan las siguientes acciones:

1. **Mostrar la IP:**

La ip del entrono se muestra en un elemento con el id “#ip”. Se utlilizara el motodo de “texto” del jQery para insertar el texto “IP” seguido de la url de la api (el url por usuario que se debe colocar en el raspberry)

2. **Actualizar los datos del raspberry pi:**

Se usara “setInterval” para ejecutar una función cada 3000 milisegundos (3 segundos). Esta función envía una solicitud GET a la API para obtener información sobre el estado del dispositivo.

3. **Solicitud GET:**

Utilizando $.get(api + "info", function (data) { ... });, se envía una solicitud a la ruta /info de la API. La respuesta de esta solicitud contiene datos que se utilizan para actualizar el DOM.

4. **Actualizar Elementos del DOM:**

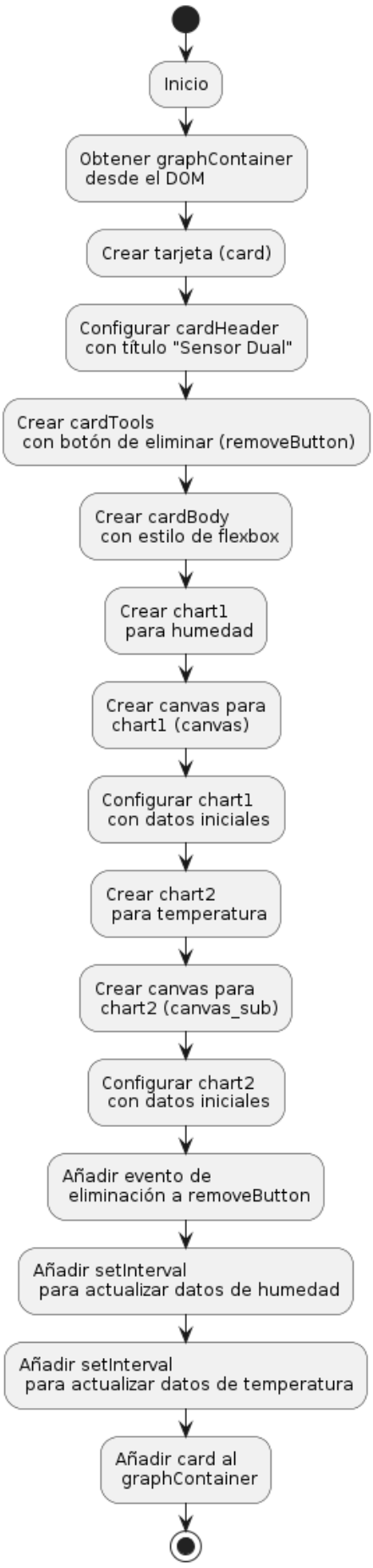
Dentro de la función de callback de $.get, se actualizan dos elementos:

-“#datos_recividos”: Muestra los datos recibidos en bytes.

-“#datos_pi4”: Muestra el estado del dispositivo basado en la información recibida. Si data.info es mayor que 0, se muestra "Funcionando", de lo contrario, se muestra "Sin Conexión".

Figura 20

Diagrama Grafico Doble


Inicio
Obtener graphContainer desde el DOM
Crear tarjeta (card)
Configurar cardHeader con título "Sensor Dual"
Crear cardTools con botón de eliminar (removeButton)
Crear cardBody con estilo de flexbox
Crear chart1 para humedad
Crear canvas para chart1 (canvas)
Configurar chart1 con datos iniciales
Crear chart2 para temperatura
Crear canvas para chart2 (canvas_sub)
Configurar chart2 con datos iniciales
Añadir evento de eliminación a removeButton
Añadir setInterval para actualizar datos de humedad
Añadir setInterval para actualizar datos de temperatura
Añadir card al graphContainer

Nota. Diagrama del Grafico Doble. Elaborado por: Los autores.

Primero, la función addGraph_doble obtiene el elemento con el ID graphContainer del DOM, que es el contenedor principal donde se añadirá la nueva tarjeta con los gráficos. Luego, se crea un nuevo div con la clase card que actúa como una tarjeta contenedora para los gráficos, y se establece su ancho al 100%.

Dentro de esta tarjeta, se crea un encabezado (card-header) que incluye un título (h3) con el texto "Sensor Dual". Además, se añade un botón de eliminación con la clase btn btn-success btn-sm que permite cerrar la tarjeta. Este botón tiene un icono de una 'x' y está configurado para eliminar toda la tarjeta cuando se hace clic en él.

La tarjeta también contiene un cuerpo (card-body) donde se colocarán dos gráficos. Este cuerpo se configura con un estilo flexible para que los gráficos se dispongan en fila y se distribuyan de manera equitativa. Para cada gráfico, se crea un contenedor (chart) que incluye un elemento canvas con dimensiones de 500x300 píxeles. Se asignan IDs únicos a estos elementos canvas para poder identificarlos y manipularlos posteriormente.

Para cada uno de estos elementos canvas, se inicializa un gráfico de tipo línea utilizando Chart.js. El primer gráfico muestra datos de "Humedad" y el segundo gráfico muestra datos de "Temperatura". Ambos gráficos tienen sus ejes y títulos configurados adecuadamente.

La función también incluye la lógica para actualizar los datos de los gráficos cada tres segundos. Utiliza “fetch” para obtener datos de dos endpoints diferentes (sensor9 y sensor10). Los datos obtenidos se añaden a los gráficos correspondientes, actualizando las etiquetas y los valores en tiempo real y asegurando que solo se muestren los 10 puntos de datos más recientes para evitar sobrecargar el gráfico.

Finalmente, la tarjeta completa, con su encabezado y cuerpo, se añade al contenedor principal (graphContainer). Esta configuración asegura que se muestre una tarjeta bien estructurada con dos gráficos que se actualizan dinámicamente con datos de sensores.

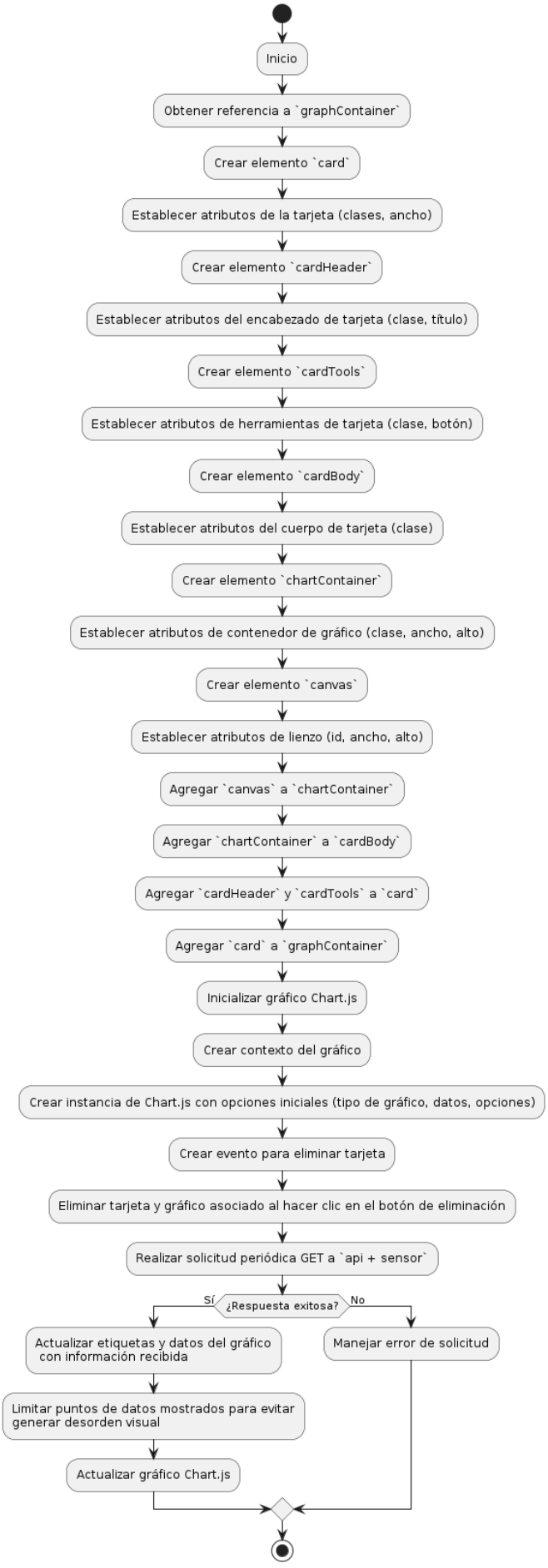
Figura 21

Ejemplo Gráfico Doble

Nota. Funcionamiento Grafico Doble. Elaborado por: Los autores.

Figura 22

Diagrama de Grafico


Inicio
Obtener referencia a `graphContainer`
Crear elemento `card`
Establecer atributos de la tarjeta (clases, ancho)
Crear elemento `cardHeader`
Establecer atributos del encabezado de tarjeta (clase, título)
Crear elemento `cardTools`
Establecer atributos de herramientas de tarjeta (clase, botón)
Crear elemento `cardBody`
Establecer atributos del cuerpo de tarjeta (clase)
Crear elemento `chartContainer`
Establecer atributos de contenedor de gráfico (clase, ancho, alto)
Crear elemento `canvas`
Establecer atributos de lienzo (id, ancho, alto)
Agregar `canvas` a `chartContainer`
Agregar `chartContainer` a `cardBody`
Agregar `cardHeader` y `cardTools` a `card`
Agregar `card` a `graphContainer`
Inicializar gráfico Chart.js
Crear contexto del gráfico
Crear instancia de Chart.js con opciones iniciales (tipo de gráfico, datos, opciones)
Crear evento para eliminar tarjeta
Eliminar tarjeta y gráfico asociado al hacer clic en el botón de eliminación
Realizar solicitud periódica GET a `api + sensor`
¿Respuesta exitosa?
Sí
No
Actualizar etiquetas y datos del gráfico con información recibida
Limitar puntos de datos mostrados para evitar generar desorden visual
Actualizar gráfico Chart.js
Manejar error de solicitud

Nota. Diagrama del Grafico. Elaborado por: Los autores.

La función addGraph se usa para crear un gráfico dentro de un contenedor, identificado por el ID graphContainer. Este componente consiste en una tarjeta (card) que contiene un título, herramientas y un cuerpo donde se aloja el gráfico.

Para comenzar, se obtiene el contenedor principal graphContainer del DOM. Luego, se crea un elemento con la clase card, configurado con un ancho fijo de 500 píxeles para mantener consistencia en el diseño.

Dentro de esta tarjeta, se establece un encabezado (card-header) que incluye un título (<h3>) dinámico (nombre_sensor), proporcionado como argumento a la función. Además, se añade un botón de eliminación con un ícono de 'x', utilizando clases de Bootstrap para su estilo.

El cuerpo de la tarjeta (card-body) contiene un contenedor (chartContainer) destinado al gráfico, ocupando el 100% del ancho y alto disponible dentro de la tarjeta. Dentro de este contenedor, se inserta un elemento <canvas> que actúa como lienzo para dibujar el gráfico. Se genera un ID único para el canvas usando un conteo basado en la cantidad actual de canvas en el contenedor.

El canvas se ajusta dinámicamente para ocupar el espacio disponible en la tarjeta, teniendo en cuenta el espacio ocupado por el encabezado y las herramientas. Esto asegura que el gráfico tenga el tamaño adecuado y se adapte al diseño de la tarjeta.

Se inicializa un gráfico de tipo línea utilizando Chart.js en el canvas creado. Este gráfico tiene configuraciones básicas como el tipo (type: "line"), datos iniciales (vacíos), color de la línea basado en un selector (color_selector), y configuración del eje y con la unidad de medida (medida).

La función también implementa la funcionalidad de eliminación del componente gráfico. Al hacer clic en el botón de eliminación, se elimina completamente la tarjeta del contenedor principal graphContainer, junto con el canvas y el gráfico asociado. Esto se logra destruyendo el gráfico de Chart.js y eliminando el canvas del DOM.

Además, la función establece un intervalo para actualizar dinámicamente los datos del gráfico cada 3 segundos. Utiliza fetch para obtener datos desde un endpoint (api + sensor) pasado como argumento. Los datos recibidos se ajustan según un factor de conversión (conversion) proporcionado, y se actualizan en el gráfico. Se asegura que solo se muestren los últimos 10 puntos de datos para evitar la sobrecarga del gráfico.

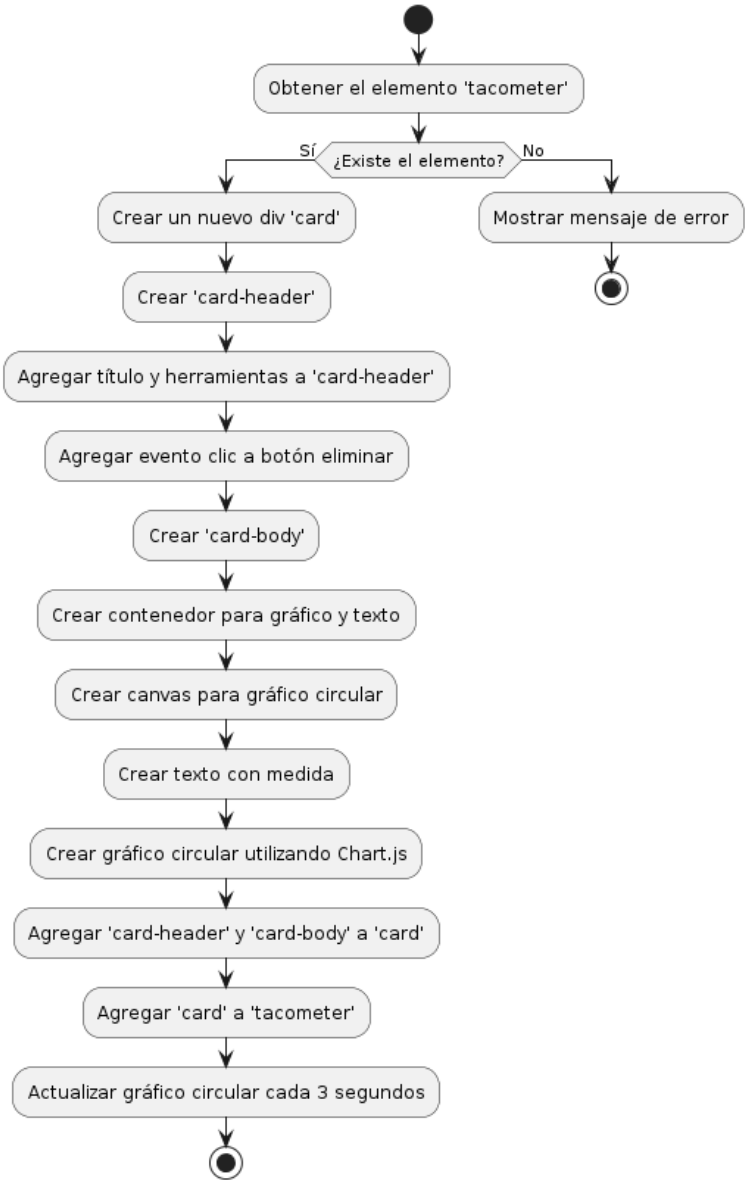
Figura 23

Ejemplo Grafica de Temperatura

Nota. Funcionamiento de la Grafica de temperatura. Elaborado por: Los autores.

Figura 24

Diagrama Tacómetro



Nota. Diagrama del tacómetro. Elaborado por: Los autores.

La función addTacometer se encarga de crear un tacómetro interactivo dentro de un contenedor específico en la página web, identificado por el ID tacometer. Este tacómetro muestra visualmente la lectura de un sensor en forma de un gráfico circular tipo doughnut, acompañado de un texto descriptivo.

Primero, se obtiene el contenedor principal tacometer del DOM donde se añadirá el tacómetro. Luego, se crea un nuevo elemento div con la clase card que servirá como contenedor para el tacómetro.

Dentro de esta tarjeta, se construye un encabezado (card-header) que incluye un título dinámico (nombre_sensor) proporcionado como argumento a la función. Además, se añade un botón de eliminación con un ícono de 'x' utilizando clases de Bootstrap para estilizarlo. Este botón permite eliminar la tarjeta completa cuando se hace clic en él.

El cuerpo de la tarjeta (card-body) contiene un contenedor (container) centrado (text-center) que aloja tanto el gráfico circular como un texto descriptivo de la medida (medida) del sensor.

Dentro del contenedor, se crea un elemento canvas que actúa como lienzo para el gráfico circular. Se genera un ID único para este canvas basado en el número actual de canvas dentro del contenedor tacometer. También se añade un elemento de texto (div) que muestra la medida del sensor concatenada con el texto "[Medida:]".

Se inicializa un gráfico de tipo doughnut utilizando Chart.js en el canvas creado. Este gráfico muestra dos sectores: uno representando la lectura actual del sensor multiplicada por un factor de conversión (conversion), y otro sector complementario que muestra el porcentaje restante hasta el máximo (en este caso, 100%).

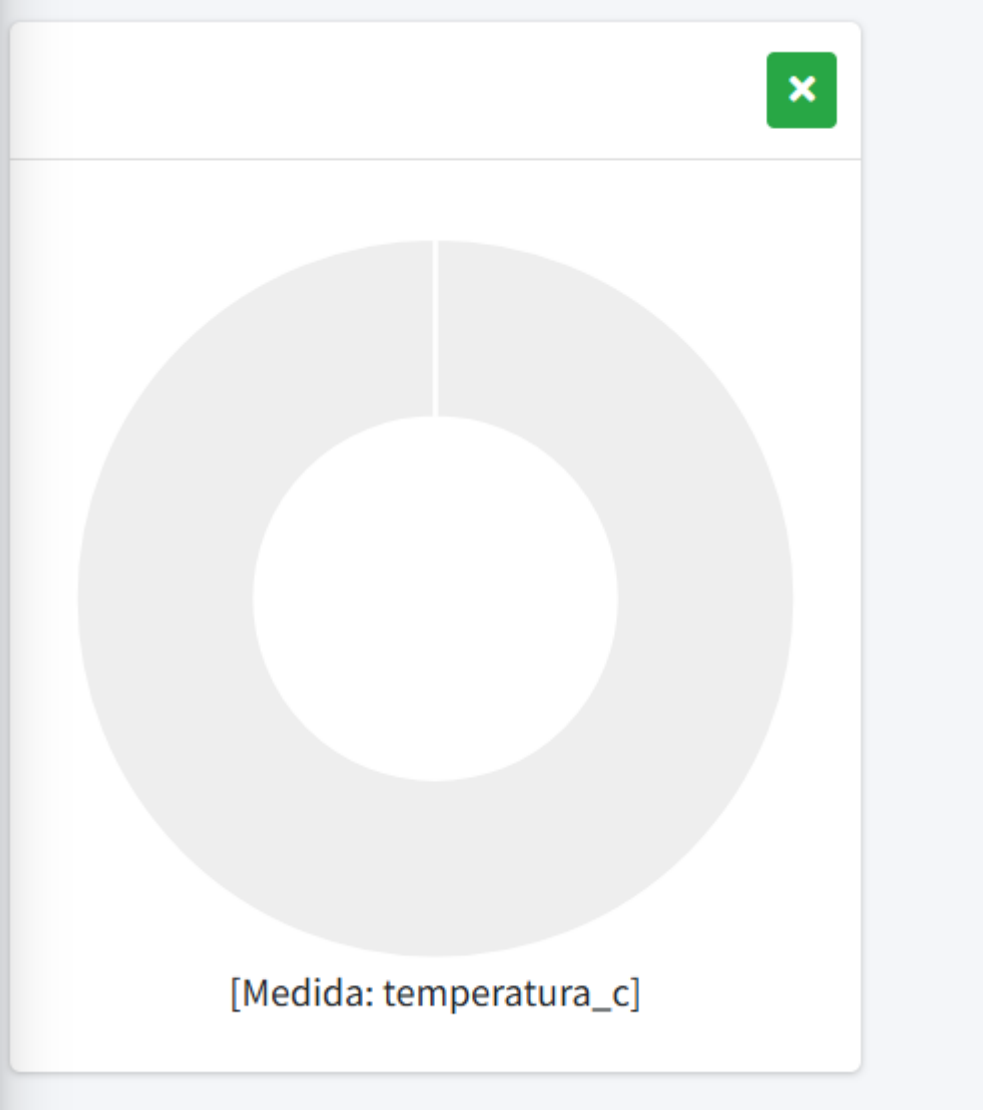
La configuración del gráfico incluye ajustes visuales como el porcentaje de corte del agujero central (cutoutPercentage: 80) para controlar el tamaño del agujero, así como animaciones suaves para escalar y rotar el gráfico.

Además, la función establece un intervalo para actualizar dinámicamente los datos del gráfico circular cada 3 segundos. Utiliza fetch para obtener datos del sensor desde un endpoint (api + sensor) proporcionado como argumento. Los datos recibidos

se ajustan según el factor de conversión y se actualizan en el gráfico para reflejar la lectura más reciente del sensor.

Figura 25

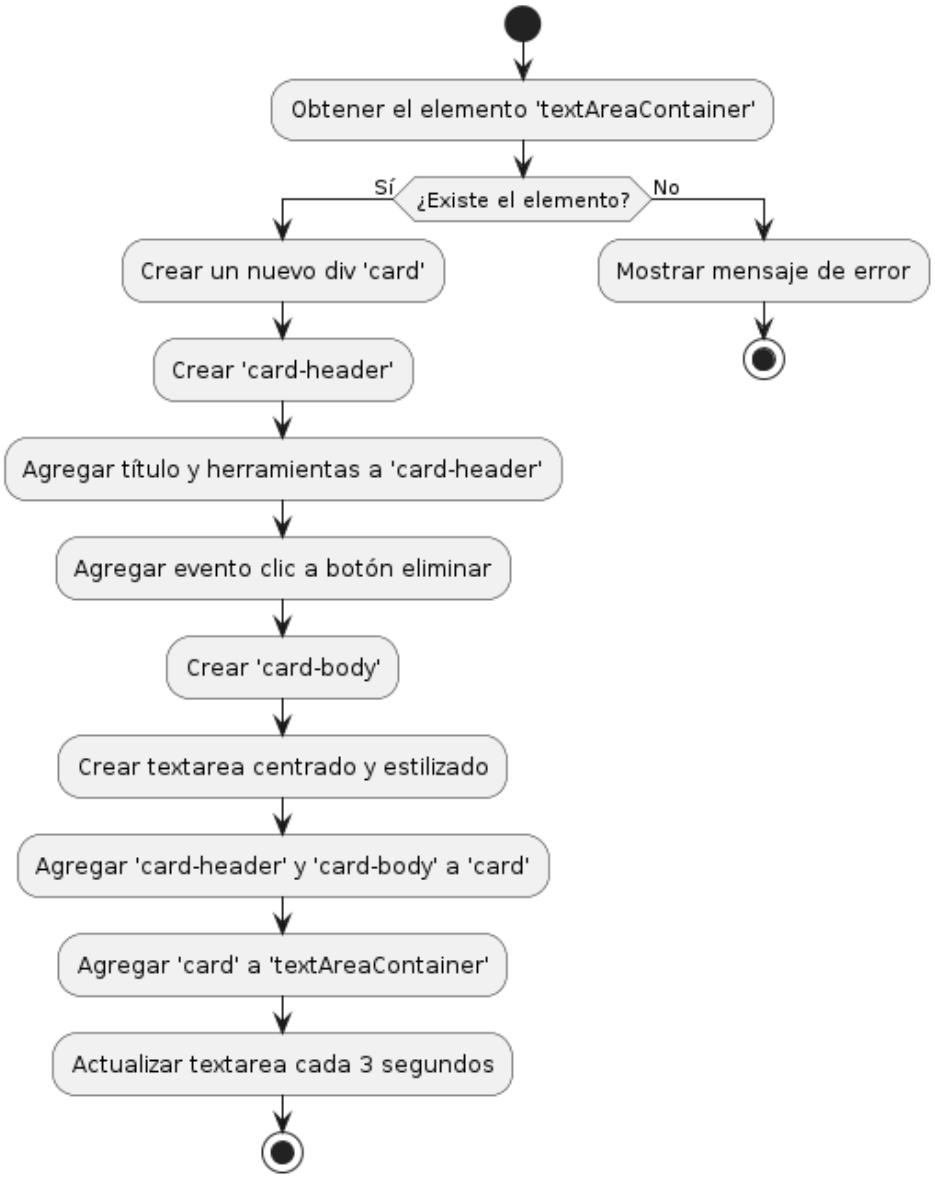
Ejemplo Grafico Tacómetro



Nota. Grafica del tacómetro. Elaborado por: Los autores.

Figura 26

Diagrama Medidor De Texto



Nota. Diagrama del medidor de texto. Elaborado por: Los autores.

La función addTextArea está diseñada para crear un componente de área de texto enriquecido dentro de un contenedor específico en la página web, identificado por el ID textAreaContainer. Este componente muestra datos de un sensor en forma de texto actualizado dinámicamente.

Inicialmente, se obtiene el contenedor principal textAreaContainer del DOM donde se añadirá el área de texto. Se crea un nuevo elemento div con la clase card que actuará como contenedor para el área de texto y otros elementos asociados.

Dentro de esta tarjeta, se construye un encabezado (card-header) que incluye un título dinámico (nombre_sensor) proporcionado como argumento a la función. Además, se añade un botón de eliminación con un ícono de 'x' utilizando clases de Bootstrap para estilizarlo. Este botón permite eliminar la tarjeta completa cuando se hace clic en él.

El cuerpo de la tarjeta (card-body) contiene un elemento textarea que se configura como de solo lectura (readOnly) para evitar que los usuarios modifiquen el contenido. El estilo del textarea se ajusta para centrar el texto, establecer un tamaño de fuente grande y desactivar el redimensionamiento.

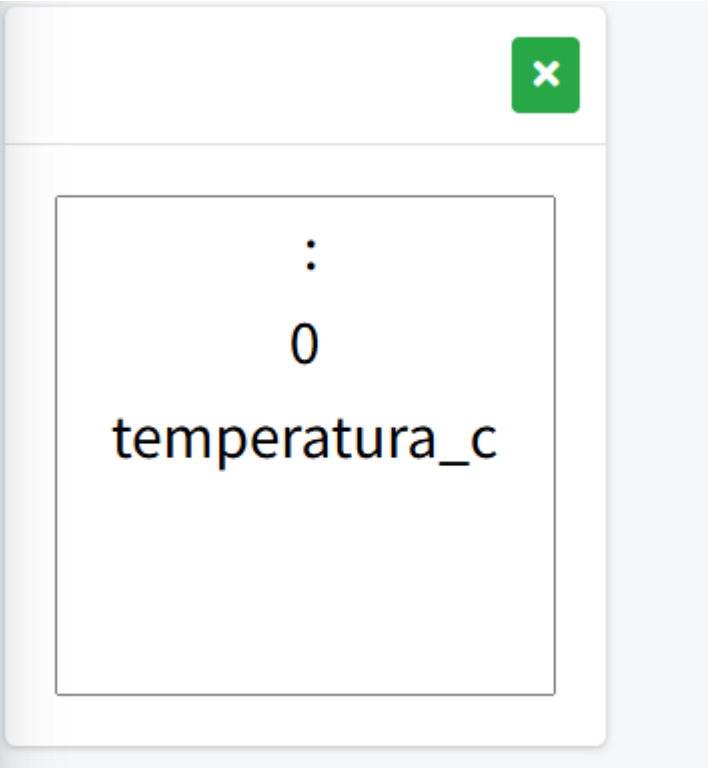
Se añaden eventos al botón de eliminación para manejar la eliminación adecuada de la tarjeta y su contenido asociado del contenedor principal textAreaContainer.

La función utiliza un intervalo que se ejecuta cada 3 segundos para actualizar dinámicamente los datos mostrados en el textarea. Utiliza fetch para obtener datos del sensor desde un endpoint (api + sensor) proporcionado como argumento. Los datos recibidos se ajustan según un factor de conversión y se formatean en un texto legible que incluye el nombre del sensor en mayúsculas, el valor convertido del sensor, y la unidad de medida.

El texto actualizado se asigna al valor del textarea, reemplazando el contenido anterior. Además, se controla el número de líneas mostradas para evitar que el textarea se sobrecargue con información antigua, asegurando que se muestren solo las 10 líneas más recientes.

Figura 27

Ejemplo Grafico Temperatura

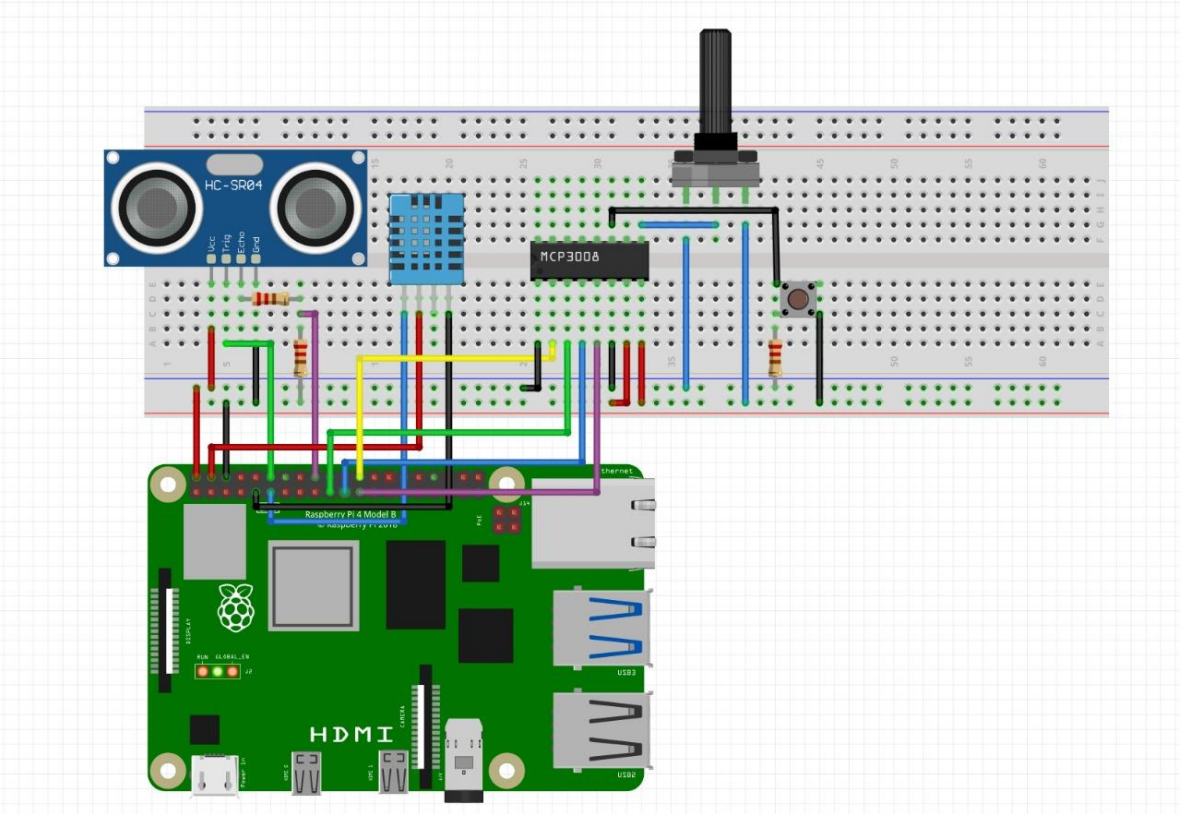


Nota. Grafica de temperatura. Elaborado por: Los autores.

Diseño conexión Raspberry Pi 4 con MCP 3008

Figura 28

Diseño conexión circuito raspberry



Nota. Esquema de conexión de la raspberry pi 4. Elaborado por: Los autores.

3.4. Fase 4: Integración y Pruebas

3.4.1. Conexión con Sensores

Configurar los sensores y validar su funcionamiento con la Raspberry Pi, asegurándose de que el sistema pueda leer tanto señales analógicas (mediante ADC - Convertidor Analógico a Digital, si es necesario) como digitales.

3.4.2. Integración Sistema Completo

Integrar el backend con el frontend y asegurarse de que el sistema funcione de manera cohesiva. Realizar pruebas de carga y rendimiento, además de pruebas de usabilidad y funcionales, ajustando el sistema basado en el feedback recibido.

CAPITULO IV

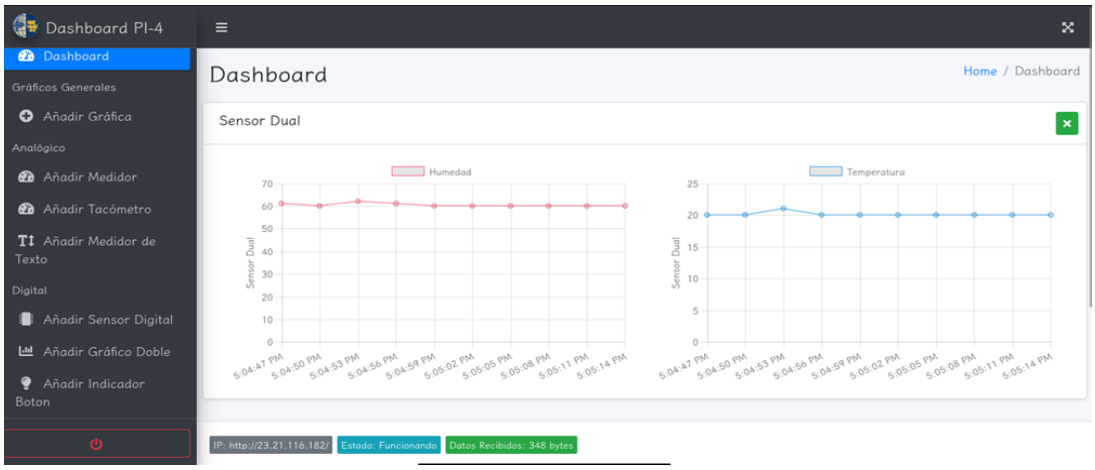
RESULTADOS

En este capítulo se presentarán las pruebas y resultados obtenidos a lo largo del desarrollo y la implementación del sistema de monitorización y el dashboard integrado con la Raspberry Pi 4. Se mostrarán los resultados de cada fase del proyecto, incluyendo la configuración y el rendimiento de la Raspberry Pi con los sensores, la efectividad del backend y la funcionalidad del frontend en la visualización de los datos

4.1. Pruebas y Resultados

Figura 29

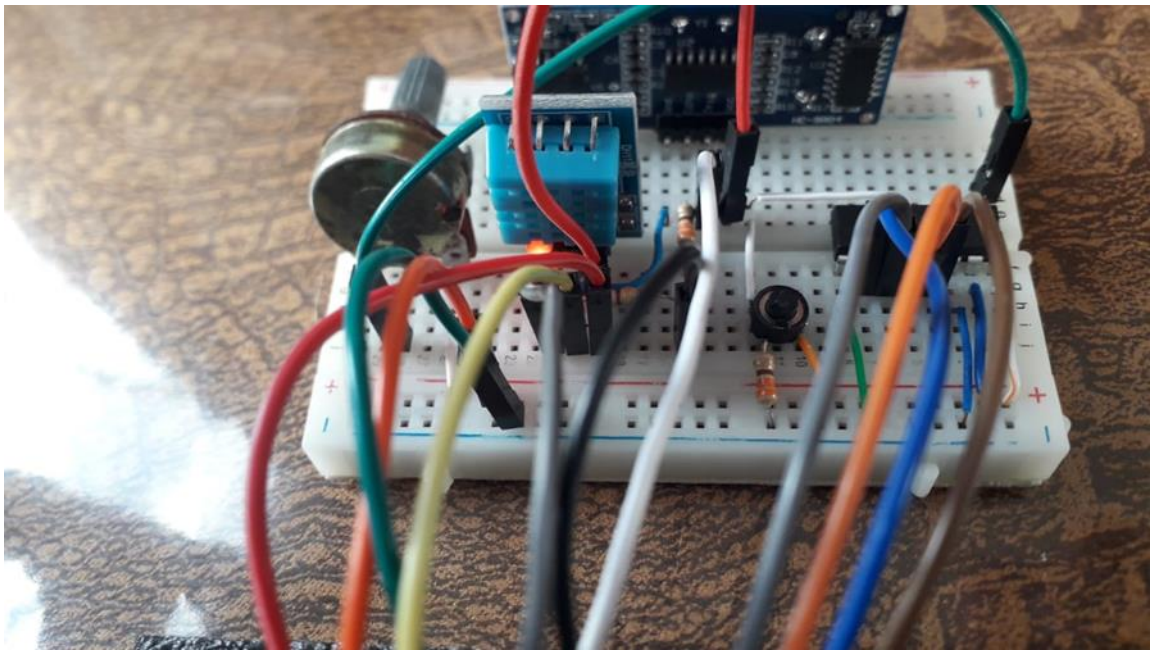
Prueba del sensor DHT 11 en gráficos duales



Nota. Gráficos Duales DHT11. Elaborado por: Los autores.

Figura 30

Conexión sensor DHT11

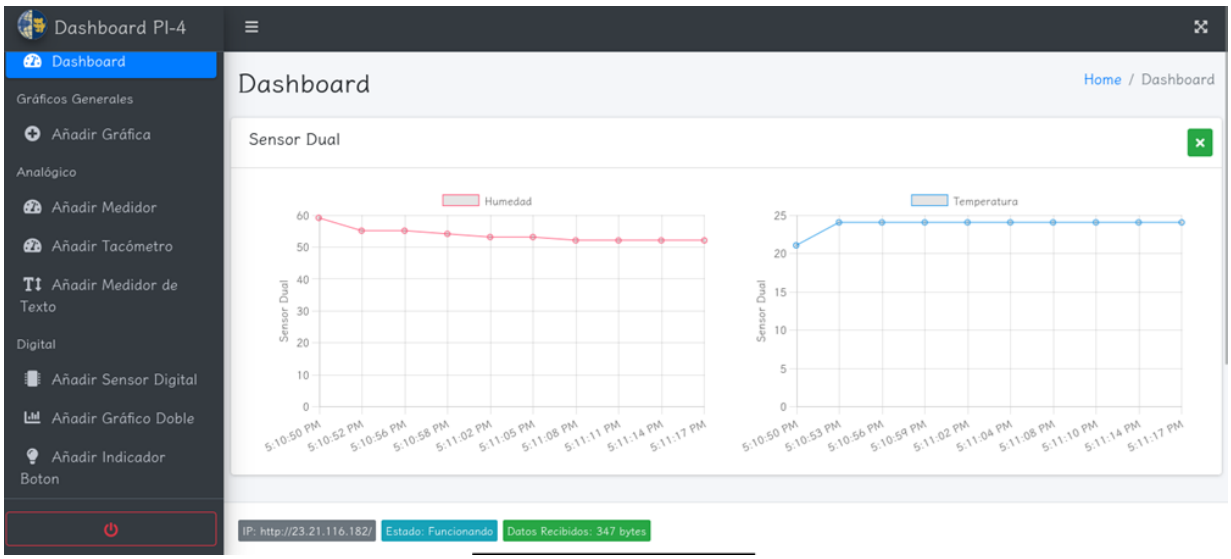


Nota. Sensor DHT 11 conectado al rasperry. Elaborado por: Los autores.

Como se puede observar el sensor envía datos a nuestro Dashboard demostrando el correcto funcionamiento de este en actualización de datos.

Figura 31

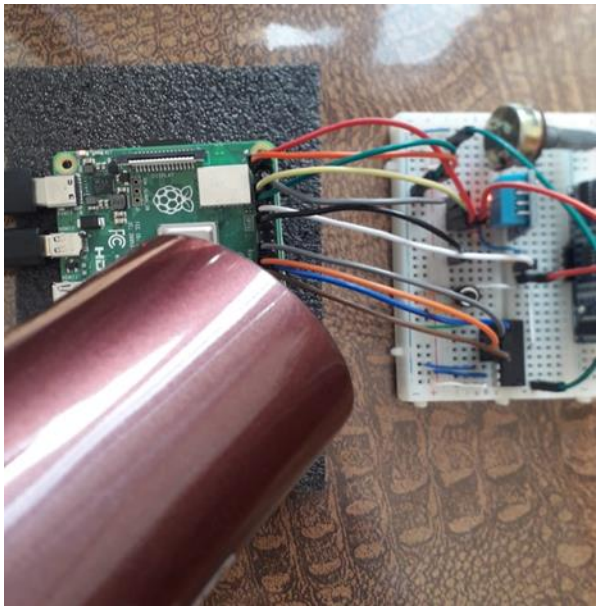
Pruebas de Calor y Humedad



Nota. Resultado al someter calor DHT11. Elaborado por: Los autores.

Figura 32

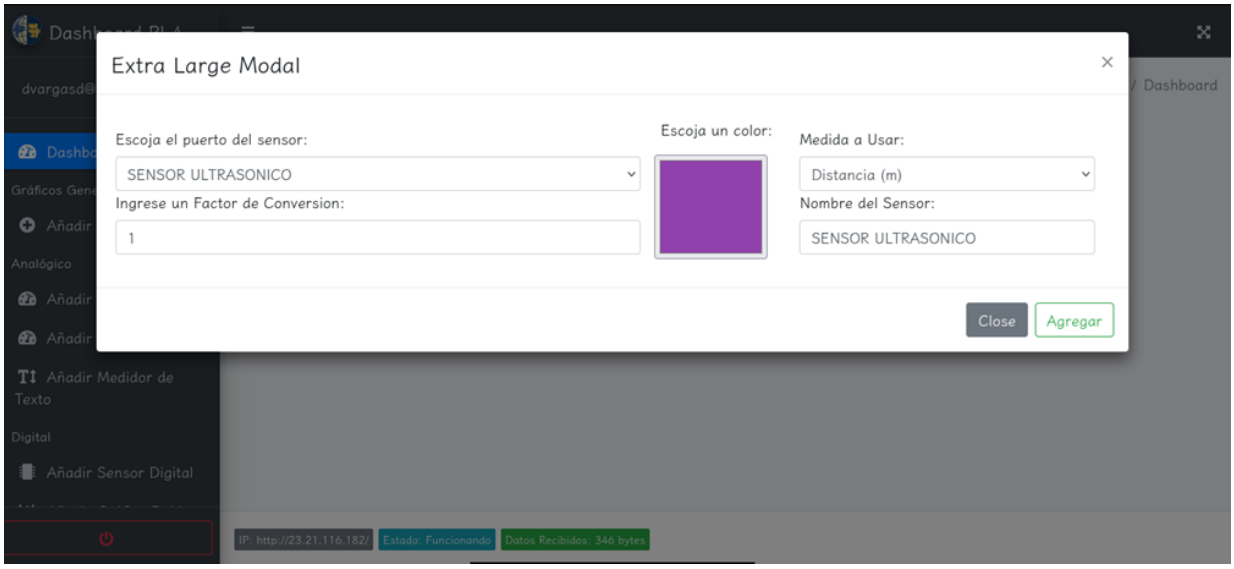
Prueba de sometiendo al calor



Nota. Circuito con prueba de sometimiento de calor usando secador de pelo. Elaborado por: Los autores.

Figura 33

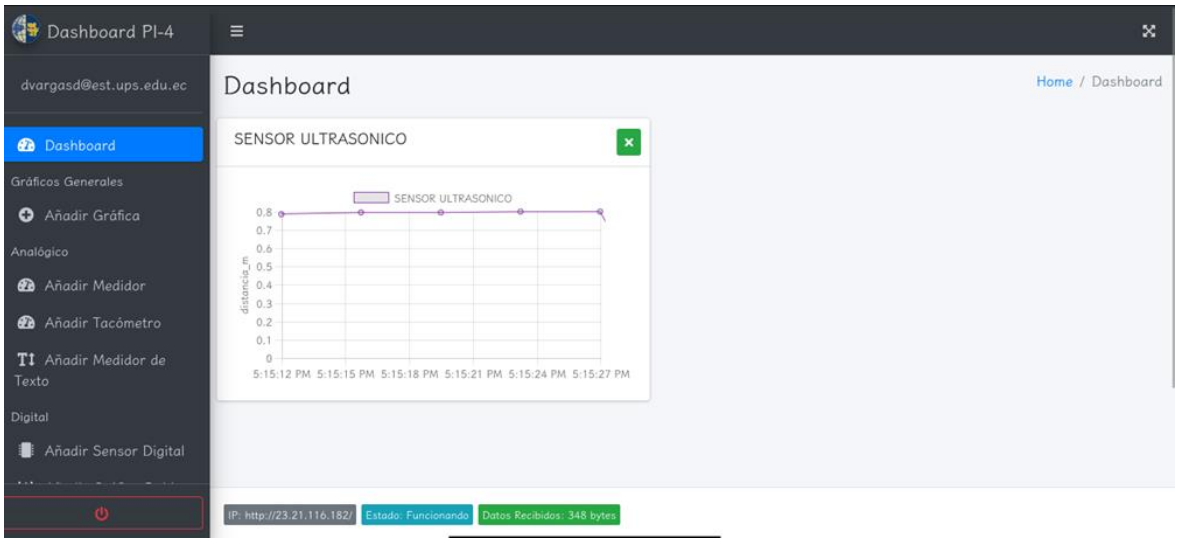
Pruebas Sensor Ultrasónico



Nota. Configuración Dashboard para uso de sensor ultrasónico. Elaborado por: Los autores.

Figura 34

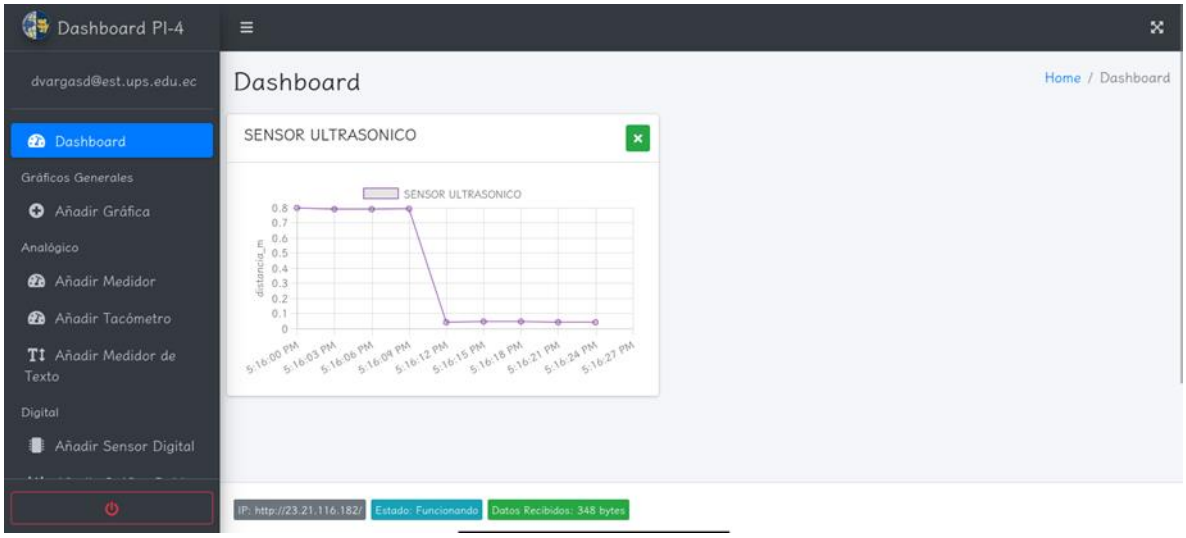
Dashboard Sensor Ultrasónico



Nota. Figura Grafico Añadido para sensor Ultrasónico. Elaborado por: Los autores.

Figura 35

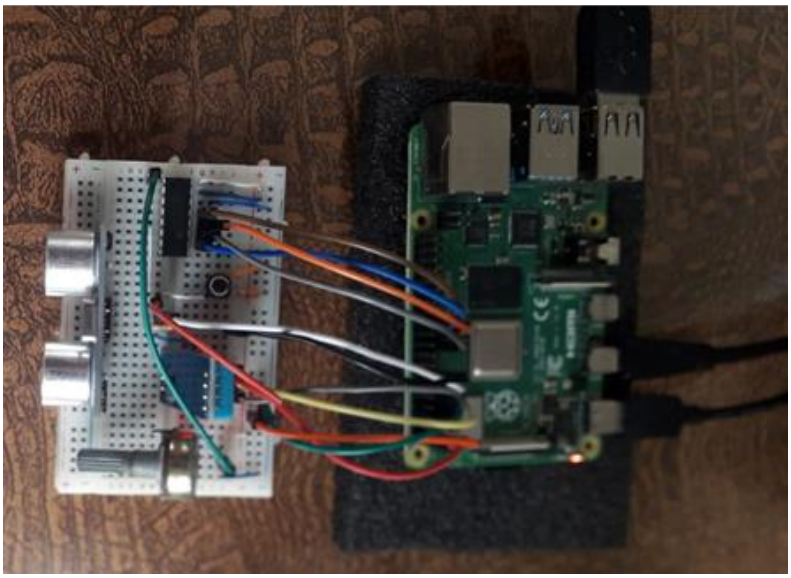
Prueba Dashboard Sensor Ultrasónico



Nota. Figura Prueba Sensor Ultrasónico. Elaborado por: Los autores.

Figura 36

Conexión Sensor Ultrasónico a Raspberry pi 4

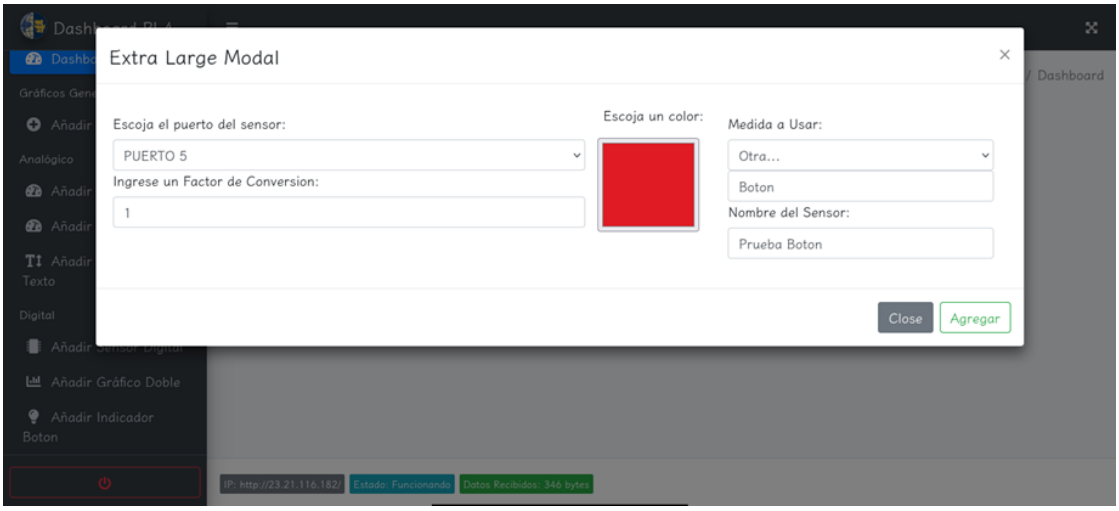


Nota. Figura Sensor ultrasónico con un objeto de prueba. Elaborado por: Los autores.

En este caso la prueba demuestra que el sensor envía datos y al detectar un objeto se calcula la distancia de este frente al sensor

Figura 37

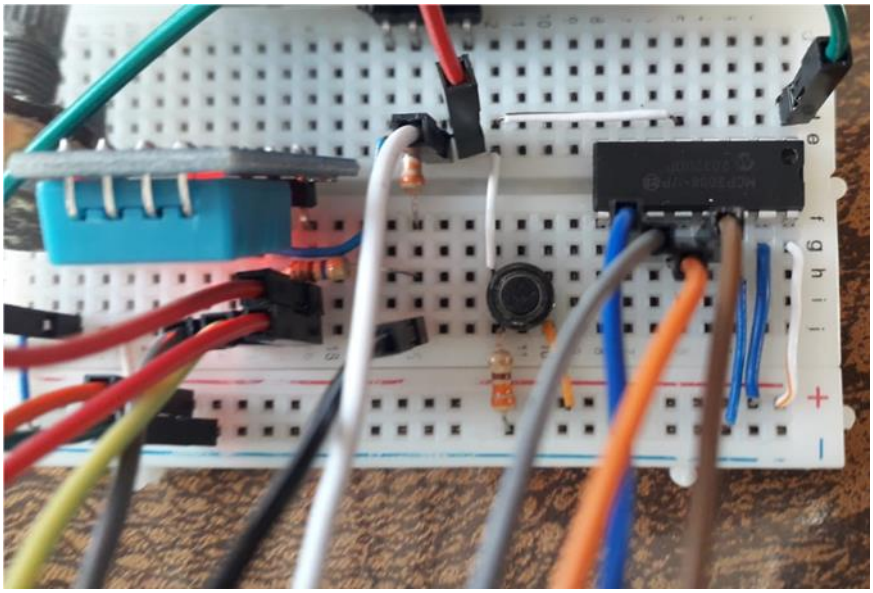
Señales digitales discretas



Nota. Figura Configuración para Grafico Señal Discreta. Elaborado por: Los autores.

Figura 38

Conexión Botón de Pruebas

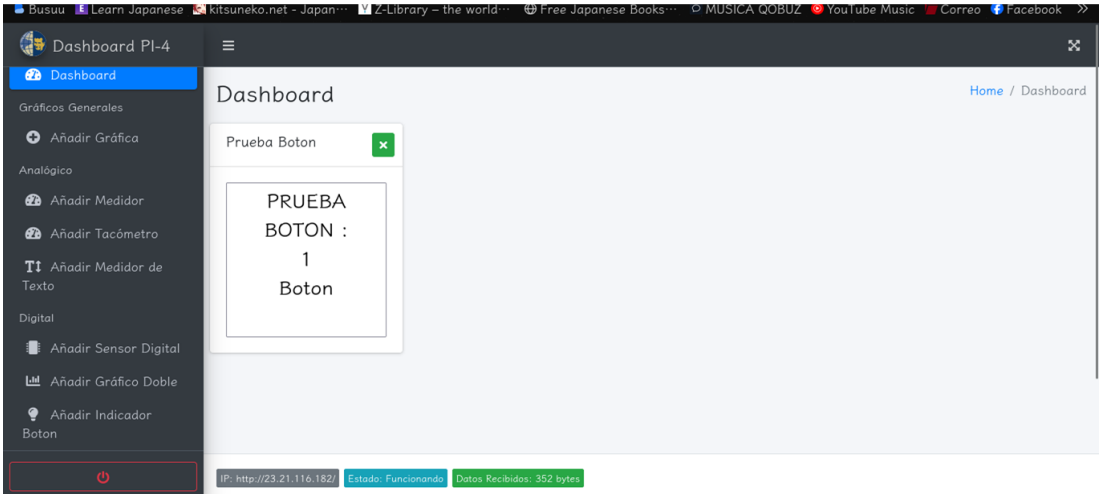


Nota. Figura Conexión botón de prueba. Elaborado por: Los autores.

Se realiza la conexión en el puerto número 5 del MCP3008 para poder enviar la información.

Figura 39

Prueba Dashboard del botón

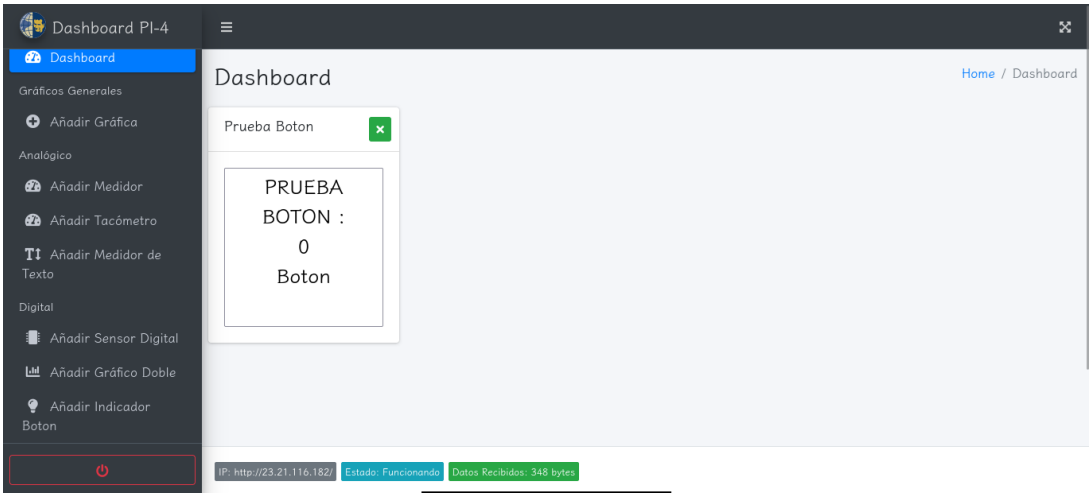


Nota. Prueba Botón Encendido. Elaborado por: Los autores.

Al pulsar nuestro botón el dashboard muestra 1 haciendo entender que se está mandando una señal de encendido

Figura 40

Validación configuración del botón

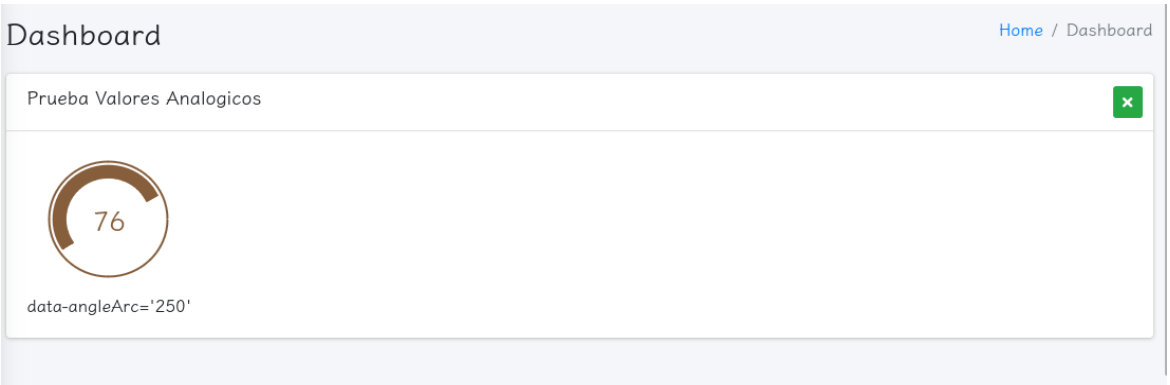


Nota. Figura Prueba Botón Apagado. Elaborado por: Los autores.

Del mismo modo, al contrario, si no se pulsa se nos muestra 0 lógico

Figura 41

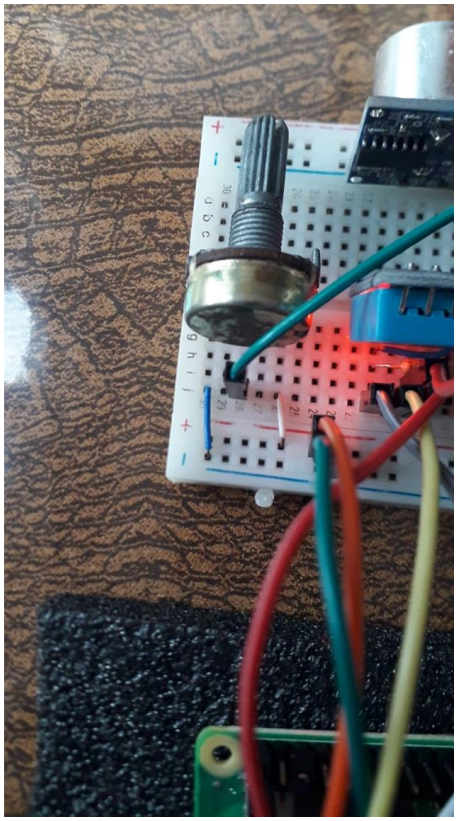
Prueba Sensores Analógicos



Nota. Figura Prueba Medidor Analógico. Elaborado por: Los autores.

Figura 42

Conexión Potenciómetro

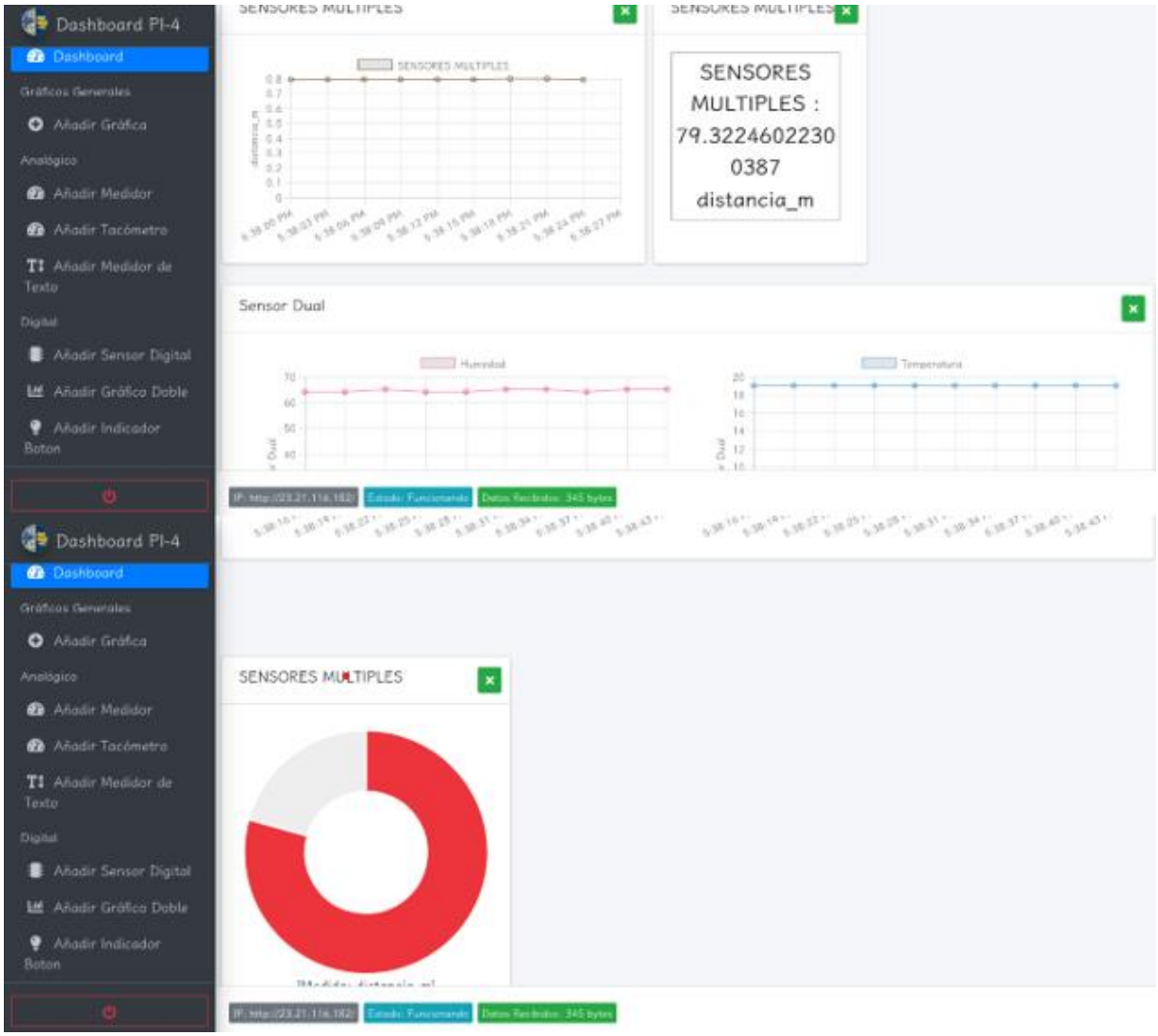


Nota. Figura conexión potenciómetro. Elaborado por: Los autores.

En este caso el cable verde seria la salía de voltaje de nuestro potenciómetro el cual iría conectado al puerto 1 del MCP3008 y mandaría los respectivos datos al raspberry como se ve en la figura.

Figura 43

Prueba Gráficos Múltiples



Nota. Figura funcionamientos gráficos múltiples. Elaborado por: Los autores.

En esta prueba podemos observar que varios gráficos funcionan a la vez con los diferentes sensores, teniendo como limitante la cantidad de request que el navegador pueda manejar.

4.2. Prueba de Conexión a internet con el Raspberry

En el caso de que el raspberry se nos mostrara el cuadro de dialogo Funcionando y la cantidad de Bytes enviados por el raspberry

Figura 44

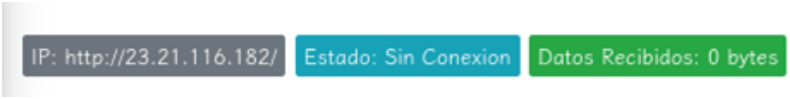
Dashboard Dialogo de Funcionamiento

Nota. Dashboard Dialogo funcionamiento activo. Elaborado por: Los autores.

En el caso que el raspberry pierda la conexión a internet o deje de funcionar automáticamente el dashboard nos informara de ello tras 8 segundos

Figura 45

Dashboard Dialogo de Funcionamiento 2



Nota. Dashboard Dialogo funcionamiento desactivado. Elaborado por: Los autores.

4.3. Pruebas de rendimiento

En esta tabla podemos observar que el promedio de uso del aplicativo en un usuario es 62,7 Kb para envio de datos y salida de datos un 38,7 Kb lo cual nos indica que es muy poco y rentable para el manejo de múltiples usuarios.

Tabla 7

Rendimiento Aplicativo Elastic BeanStalk

	CPU %	Uso de Red (entrada datos)	Usos de Red (salida datos)
HORAS	0,64	62,7 Kb	38,7 Kb
DIAS	0,64	62,7 Kb	38.7 Kb
SEMANAS	2	75,7 Kb	46,4 Kb

Nota. Presentación del rendimiento de conexión de red de los componentes. Elaborado por: Los Autores.

4.3.1. Rendimiento del Aplicativo en el Navegador Web

En este caso el consumo es mínimo en la carga del aplicativo en el navegador ya que fue realizado en su totalidad con javascript y CSS por ende el rendimiento no se ve afectado, podemos ver que 1,5 MB y 1,2 MB de media para JS y CSS.

Figura 46

Aplicativo Web Rendimiento

Nota. Rendimiento del Aplicativo en el navegador Web. Elaborado por: Los autores.

4.4. Latencia del Servidor

Tabla 8

Latencia en el Servidor

UBICACIÓN	BYTES PROMEDIO ENVIADOS POR PETICIÓN	BYTES PROMEDIO RECIBIDOS POR PETICIÓN	LATENCIA
USA Este Virginia	213	594	80 – 100MS

Nota. Presentación de latencia en el servidor. Elaborado por: Los Autores.

CONCLUSIONES

La integración de Amazon Cognito para la gestión de identidades y AWS para el alojamiento y procesamiento de datos ha proporcionado una base escalable que asegura la disponibilidad y seguridad de los datos. Esto ayuda al proyecto al permitir la gestión segura y centralizada de múltiples usuarios, asegurando que el sistema pueda crecer y adaptarse a un mayor número de estudiantes y dispositivos sin perder eficiencia. La arquitectura basada en AWS, capaz de manejar hasta 10,000 solicitudes por segundo, garantiza la protección de información sensible y simplifica la administración de usuarios y dispositivos en un entorno educativo dinámico. Esta infraestructura mejora la confiabilidad y disponibilidad del sistema, asegurando que los datos recolectados estén siempre accesibles y protegidos contra fallos.

El proyecto ha demostrado ser técnica y operativamente viable al integrar eficazmente tecnologías como Elastic Beanstalk, asegurando escalabilidad y sostenibilidad. Esto se evidencia en la capacidad del sistema para manejar hasta 1000 solicitudes de datos por minuto sin degradación de rendimiento y la integración de múltiples sensores. La flexibilidad en la gestión de sensores y la optimización en la adquisición de datos destacan su capacidad de adaptación y expansión de funcionalidades, asegurando su utilidad a largo plazo en entornos académicos y de investigación. Además, el uso de Elastic Beanstalk ha reducido el tiempo de despliegue de nuevas versiones del sistema en un 40%, mejorando significativamente la eficiencia operativa.

Con una base sólida, el proyecto está preparado para evolucionar e incorporar nuevas funcionalidades, como el control remoto de dispositivos y la generación de alertas automáticas. Estas mejoras permitirán a los usuarios interactuar en tiempo real con el sistema, lo que incrementará significativamente su utilidad al permitir respuestas rápidas a cambios en el

entorno monitoreado. Por ejemplo, la capacidad de recibir alertas instantáneas cuando los valores de los sensores superan ciertos umbrales críticos puede mejorar la seguridad en los laboratorios. Además, el control remoto permitirá ajustes inmediatos a los dispositivos conectados, optimizando la operatividad y respuesta ante eventos, y la generación de alertas reducirá los tiempos de respuesta en un 30%.

La viabilidad de utilizar Raspberry Pi ha sido demostrada, superando las limitaciones tradicionales asociadas con Arduino y sensores como el ESP8266. En pruebas realizadas, el sistema basado en Raspberry Pi manejó eficientemente la integración de hasta 20 sensores simultáneamente, con tiempos de respuesta de menos de 200 ms. La implementación exitosa de un sistema escalable y efectivo que integra tecnologías cloud muestra que Raspberry Pi puede proporcionar una experiencia comparable en términos de funcionalidad y flexibilidad. Esto abre nuevas posibilidades para aplicaciones en entornos académicos y de investigación, destacando su potencial para innovar y mejorar en el campo del IoT. Los resultados mostraron una reducción del 25% en los costos operativos en comparación con sistemas basados en Arduino.

A pesar de las limitaciones inherentes a su naturaleza de prototipo, el proyecto ha sido extremadamente valioso. Se ha logrado integrar exitosamente tecnologías clave como Amazon Cognito y AWS, estableciendo una base sólida para la escalabilidad y la gestión eficiente de usuarios. Este sistema puede gestionar actualmente hasta 50 usuarios simultáneos, lo que muestra su capacidad para crecer y adaptarse a mayores demandas. Aunque inicialmente se enfocó en la recepción y visualización de datos en lugar de la interacción bidireccional, el proyecto ha demostrado ser un prototipo funcional efectivo que cumple con las necesidades en entornos académicos. Su naturaleza de código abierto lo hace apto para futuras mejoras y desarrollos por parte de la comunidad académica, facilitando la colaboración y la innovación continua.

RECOMENDACIONES

Se puede incorporar nuevas funcionalidades basadas en el feedback de los usuarios, como alertas automáticas basadas en umbrales predefinidos, informes automáticos generados periódicamente, y capacidades de análisis predictivo utilizando técnicas de machine learning.

Para aplicaciones más grandes, considerar el uso de servicios de cloud computing para manejar grandes volúmenes de datos y proporcionar redundancia y alta disponibilidad. Integrar la Raspberry Pi con servicios en la nube puede aumentar la capacidad de procesamiento y almacenamiento.

Aunque el sistema es escalable en cierta medida, hay limitaciones inherentes a la capacidad de la Raspberry Pi y la arquitectura actual. Para aplicaciones que requieren una alta escalabilidad, como la monitorización de cientos de sensores en múltiples ubicaciones, podría ser necesario migrar a una infraestructura más robusta basada en la nube.

REFERENCIAS

Adereso Team. (2023, octubre 2). *Los beneficios de una interfaz intuitiva*. Adere.so. https://www.adere.so/blog/los-beneficios-de-una-interfaz-intuitiva

Agarwal, T. (2019, agosto 5). *DHT11 Sensor definition, working and applications*. ElProCus - Electronic Projects for Engineering Students. https://www.elprocus.com/a-brief-on-dht11-sensor/

Alarcón Diaz, H. H., Quintana Ortiz, M., Moreno Casachagua, H. R., Soria Cuellar, F. T., & Dantas Cavalcanti, A. C. (2023). Implementación del Raspberry Pi, como computadora alternativa de bajo costo, para el aprendizaje remoto. *Qantu Yachay*, *3*(2), 57–66. https://doi.org/10.54942/qantuyachay.v3i2.59

Amazon. (2023). *AWS Elastic Beanstalk Despliegue y amplíe las aplicaciones web*. Amazon.com. https://aws.amazon.com/es/elasticbeanstalk/

Amazon. (2024). *¿Qué es Amazon Cognito?* Amazon.com. https://docs.aws.amazon.com/es_es/cognito/latest/developerguide/what-is-amazon-cognito.html

Bosquez, C., & Valencia, W. (2022). Telemedicine IoT prototype “doctor pi” for measuring elders vital signs in rural areas of Ecuador. En *Lecture Notes in Electrical Engineering* (pp. 831–840). Springer Nature Singapore.

descubrearduino. (2020, marzo 9). Protoboard ¿Qué es, Cómo funciona y cómo se usa? *Descubrearduino.com*. https://descubrearduino.com/protoboard/

Devis, R. (2014, diciembre 29). Internet de las cosas: situación actual y perspectivas de futuro. *Rankia*. https://www.rankia.com/blog/moviles-tablets-ordenadores/2593688-internet-cosas-situacion-actual-perspectivas-futuro

García, J. E. O., & Barragán, J. N. F. (2014, junio 20). *Diseño de interfaz usando minicomputador Raspberry Pi a ser usada en la impresión en 3 Dimensiones con demostración práctica de funcionamiento en prototipo de impresora Reprap*. Edu.ec. https://www.dspace.espol.edu.ec/bitstream/123456789/29725/1/Resumen%20de%20tesis%20JOrdo%c3%b1ez%20y%20JFiallos%2c%20director%20de%20tesis%20M.Sc.%20Carlos%20Valdivieso%20A.%205%20junio%202014.pdf

GitLab: Saber todo sobre el repositorio Git para DevOps. (2022, diciembre 7). Formación en ciencia de datos | Datascientest.com; DataScientest. https://datascientest.com/es/gitlab-todo-lo-que-hay-que-saber

jotrin. (2023, diciembre 27). *What is a 330 ohm resistor? 330 ohm resistor color code*. Jotrin.com; Jotrin Electronics Limited. https://www.jotrin.com/technology/details/what-is-a-330-ohm-resistor-and-color-code

Lara, A. S. V. (2019, julio). *DESARROLLO DE UN PROTOTIPO PARA LA MONITORIZACIÓN DE PARÁMETROS AMBIENTALES UTILIZANDO UN RASPBERRY PI Y UN AGENTE MULTIPROTOCOLO*. Edu.ec. https://bibdigital.epn.edu.ec/bitstream/15000/20345/1/CD%209816.pdf

Los desafíos y oportunidades de la adopción del Internet de las cosas (IoT) en Colombia. (2023, junio 7). Intelligent Training. https://www.itcolombia.com/iot-en-colombia/

Marcu, A.-E., Suciu, G., Olteanu, E., Miu, D., Drosu, A., & Marcu, I. (2019). IoT System for Forest Monitoring. *2019 42nd International Conference on Telecommunications and Signal Processing (TSP)*, 629–632.

Microchip. (2024). *MCP3008*. Microchip.com. https://www.microchip.com/en-us/product/MCP3008

Mirjana Maksimović, Vladimir Vujović, Davidović, N., & Perišić, B. (2014). *Raspberry Pi as Internet of Things hardware: Performances and Constraints*. Researchgate.net. https://www.researchgate.net/profile/Vladimir-Vujovic/publication/280344140_ELI16_Maksimovic_Vujovic_Davidovic_Milosevic_Perisic/links/55b3368608ae9289a08594aa/ELI16-Maksimovic-Vujovic-Davidovic-Milosevic-Perisic.pdf

Muñoz, A. (2020, septiembre 22). *¿Qué es GITHub y para qué sirve?* Webempresa. https://www.webempresa.com/hosting/que-es-github.html

Muñoz-Sanabria, L. F., Alegría, J. A. H., & Rodriguez, F. J. Á. (2019). XP / architecture (XA): A collaborative learning process for agile methodologies when teams grow. En *Communications in Computer and Information Science* (pp. 244–257). Springer International Publishing.

Pantech elearning. (2021, julio 8). *What are advantages and disadvantages of Raspberry Pi?* Pantech ELearning. https://www.pantechelearning.com/advantages-disadvantages-of-raspberry-pi/

Pardeshi, V., Sagar, S., Murmurwar, S., & Hage, P. (2017). Health monitoring systems using IoT and Raspberry Pi — A review. *2017 International Conference on Innovative Mechanisms for Industry Applications (ICIMIA)*, 134–137.

Patel, D., Maiti, C., & Muthuswamy, S. (2023). Real-time performance monitoring of a CNC milling machine using ROS 2 and AWS IoT towards industry 4.0. *IEEE EUROCON 2023 - 20th International Conference on Smart Technologies*, 776–781.

Pires, R. (2021, octubre 14). *Descubre qué es un dashboard y qué información debe contener*. Rock Content - ES; Rock Content. https://rockcontent.com/es/blog/dashboard/

Portal, T. I. C. (2022, junio 27). *Amazon Web Services*. TIC Portal; European Knowledge Center for Information Technology, SL. https://www.ticportal.es/temas/cloud-computing/amazon-web-services

¿Qué es el middleware? (2023, junio 12). Ibm.com. https://www.ibm.com/es-es/topics/middleware

¿Qué es la integración continua? (2024, mayo 6). Ibm.com. https://www.ibm.com/es-es/topics/continuous-integration

Redhat. (2023, invierno 1). *¿Qué es el Internet de las cosas (IoT) y cómo funciona?* Redhat.com. https://www.redhat.com/es/topics/internet-of-things/what-is-iot

Sánchez, M. A., & Ramoscelli, G. (2017, diciembre 27). *CREACIÓN DE VALOR A PARTIR DEL INTERNET DE LAS COSAS*. Redalyc.org. https://www.redalyc.org/journal/3579/357959311009/html/

Santos, S. (2024, enero 25). *Raspberry Pi: DHT11/DHT22 temperature and humidity data logger (python)*. Random Nerd Tutorials. https://randomnerdtutorials.com/raspberry-pi-temperature-humidity-data-logger/

Shirgaonkar, A. A., Bhide, D. V. V., Mohite, S., Parab, P., & Bangi, K. (2023). Raspberry Pi and IoT based data acquisition and real time, remote data monitoring system. *International journal for research in applied science and engineering technology*, *11*(3), 2231–2236. https://doi.org/10.22214/ijraset.2023.49957

Sme, T. C. (2022, junio 11). Ultrasonic sensor characteristics: A comprehensive guide for DIY projects. *Techie Science*. https://techiescience.com/ultrasonic-sensor-characteristics/

Stefany, N. N. K., & Andrés, C. P. V. (2020). *Diseño y desarrollo de un prototipo de red de sensores IOT utilizando tegnología lorawan para el monitoreo de parámetros*

ambientales en interiores y exteriores. Edu.ec. https://dspace.ups.edu.ec/bitstream/123456789/19439/1/UPS-GT003019.pdf

Thinger.Io. (2019, septiembre 10). Thinger.Io. https://thinger.io/

UNIT Electronics. (2024). *Push Button 2 Pines MicroSwitch*. UNIT Electronics. https://uelectronics.com/producto/push-button-2-pines-microswitch/

Vailshery, L. S. (2023, julio 27). *IoT connected devices worldwide 2019-2030*. Statista. https://www.statista.com/statistics/1183457/iot-connected-devices-worldwide/

Ventura, M. Q. (2023, julio 12). *Iot en Minería: Tecnologías de monitoreo y sensores para mejorar la seguridad y la eficiencia en la operación minera*. Codeauni.com. https://www.codeauni.com/comunidad/blog/108/

Vujović, V., & Maksimović, M. (2015). Raspberry Pi as a Sensor Web node for home automation. *Computers & Electrical Engineering: An International Journal*, *44*, 153–171. https://doi.org/10.1016/j.compeleceng.2015.01.019

What is serverless? (2024). Oracle.com. https://www.oracle.com/cloud/cloud-native/functions/what-is-serverless/

Yida. (2019, octubre 22). *Potentiometer - functions, types and applications*. Latest Open Tech From Seeed. https://www.seeedstudio.com/blog/2019/10/22/potentiometer-functions-types-and-applications/