DESARROLLO DE UN SISTEMA DE OPERACIÓN Y GESTIÓN PARA LA FAMILIA DE DISPOSITIVOS IOT DE TABLESA S.A., ENFOCADOS EN CONSUMO ENERGÉTICO

Carlos Alberto Gaviria Gomez

UNIVERSIDAD DEL VALLE
FACULTAD DE INGENIERÍA
ESCUELA DE INGENIERÍA ELÉCTRICA Y ELECTRÓNICA
SANTIAGO DE CALI
2024

DESARROLLO DE UN SISTEMA DE OPERACIÓN Y GESTIÓN PARA LA FAMILIA DE DISPOSITIVOS IOT DE TABLESA S.A., ENFOCADOS EN CONSUMO ENERGÉTICO

Carlos Alberto Gaviria Gomez

Director:
Eval Bladimir Bacca Cortes Ph.D

Codirector:
Alvaro Duvan Acosta Aux M.Sc

UNIVERSIDAD DEL VALLE
FACULTAD DE INGENIERÍA
ESCUELA DE INGENIERÍA ELÉCTRICA Y ELECTRÓNICA
SANTIAGO DE CALI
2024

Nota de aceptación

Firma del director del trabajo

Firma del codirector del trabajo

Firma del evaluador

Firma del evaluador

AGRADECIMIENTOS

A Tablesa S.A., por la oportunidad y la confianza depositada en mi trabajo y conocimientos. Su apoyo fue fundamental para la realización de este proyecto.

A Protección S.A., mi actual empleador, por permitirme desarrollar este proyecto paralelamente con mis responsabilidades laborales. Su comprensión y flexibilidad fueron esenciales para el éxito de este trabajo.

A la Universidad del Valle, especialmente a las personas que me guiaron en este camino académico:

Al Ph.D. Eval Bladimir Bacca Cortes, mi director de proyecto, quien siempre nos motivó a dar lo mejor de nosotros. Sin duda alguna, el mejor profesor de la carrera. Sus enseñanzas han sido la piedra angular de este proyecto y de mi vida laboral.

Al M.Sc. Duvan Acosta Aux, mi codirector, un excelente amigo y profesional. Sus consejos y guía fueron clave para recorrer este camino académico y adquirir los conocimientos necesarios para realizar esta tesis.

Ha sido un honor contar con ambos como mentores en mi carrera como estudiante y profesional.

A mi novia, Helen Caicedo, la persona que más admiro y que afortunadamente tengo a mi lado. Me acompañó durante todo el proceso y me ayudó a cerrar con broche de oro este proyecto. Su habilidad para estructurar y ordenar mi desorden fue clave para el cumplimiento de este trabajo.

A mis amigos:

A Christian Tilano, arquitecto y compañero de la infancia. Gracias a sus palabras de motivación y compañía logramos llegar hasta el final.

A Andrés Gómez y Luis Miguel Lara, ingenieros electrónicos y camaradas en este competitivo mundo. Hemos construido un ambiente de competencia sana y confianza que me mantiene motivado y siempre queriendo llegar más lejos. Sus palabras de apoyo, amenazas y chistes sobre obtener mi título de

ingeniero fueron claves para sentir la necesidad de culminar esta etapa y seguir adelante.

A Juan David García Libreros, mi mejor amigo. Iniciamos la universidad juntos y, tras su partida, fue muy difícil continuar solo y mucho más culminar esta etapa. Me tomó tiempo sanar, sentirme merecedor y poder llegar a este punto en el que puedo graduarme en su honor. Gracias por todo el apoyo emocional que necesité para dar por terminada esta etapa. Este logro también va por él.

A mi familia:

A mi hermana Linda Simonds, inicialmente tenía la tarea de ser un ejemplo para ella, pero resultó ser todo lo contrario. Es una de las personas que más admiro y me llena de motivación para llegar muy lejos.

A mi madre, quien siempre ha creído en mí, incluso en momentos en los que yo mismo no lo hacía. Sus palabras en los momentos más importantes y sus enseñanzas fueron el combustible que me mantuvo firme para la culminación de este proyecto.

RESUMEN

El presente trabajo de grado desarrolla un sistema de operación y gestión para la familia de dispositivos IoT de Tablesa S.A., orientados al consumo energético. La introducción aborda la importancia del Internet of Things (IoT) en la Industria 4.0 y los desafíos que enfrenta Tablesa con su creciente red de dispositivos, incluyendo la autenticación segura, actualizaciones de firmware remotas, almacenamiento estructurado de datos y la gestión intuitiva a través de una interfaz de usuario. Los objetivos se centraron en establecer las necesidades de Tablesa, diseñar e implementar un sistema de operación y gestión que facilite la autenticación, actualización y administración de dispositivos IoT, y definir un protocolo de pruebas para evaluar la solución propuesta. Para el desarrollo iterativo y adaptativo del proyecto se empleó la metodología ágil SCRUM. El sistema se diseñó utilizando una arquitectura de microservicios y tecnologías Serverless en Amazon Web Services (AWS), integrando herramientas como AWS Amplify, Cognito y DynamoDB. Los resultados demostraron que el sistema es escalable, seguro y eficiente, logrando mejoras concretas como una reducción del 82.02% en los tiempos de puesta en marcha de dispositivos, comparado con el sistema anterior. Este avance permite una preparación y despliegue más rápido y eficiente de los dispositivos IoT, reduciendo significativamente el tiempo y esfuerzo requeridos. También se resalta la importancia de la autenticación segura y la actualización de firmware Over-The-Air (OTA), así como la necesidad de un almacenamiento adecuado de datos para futuros análisis. Las conclusiones indican que el sistema desarrollado proporciona una solución escalable, facilitando la administración de la red de dispositivos IoT de Tablesa, y las recomendaciones sugieren continuar con la optimización del sistema y explorar nuevas funcionalidades que mejoren aún más la eficiencia operativa y la capacidad de respuesta de la empresa.

Palabras clave: IoT, IIoT, OTA, Infraestructura en la nube, Arquitectura Serverless

ABSTRACT

This thesis develops an operation and management system for the IoT device family of Tablesa S.A., focused on energy consumption. The introduction addresses the importance of the Internet of Things (IoT) in Industry 4.0 and the challenges faced by Tablesa with its growing network of devices, including secure authentication, remote firmware updates, structured data storage, and intuitive management through a user interface. The objectives focused on establishing Tablesa's needs, designing and implementing an operation and management system that facilitates the authentication, updating, and administration of IoT devices, and defining a testing protocol to evaluate the proposed solution. The agile SCRUM methodology was employed for the iterative and adaptive development of the project. The system was designed using a microservices architecture and serverless technologies on Amazon Web Services (AWS), integrating tools such as AWS Amplify, Cognito, and DynamoDB. The results demonstrated that the system is scalable, secure, and efficient, achieving concrete improvements such as a 82.02% reduction in device setup times compared to the previous system. This advancement allows for faster and more efficient preparation and deployment of IoT devices, significantly reducing the time and effort required. The importance of secure authentication and Over-The-Air (OTA) firmware updates is also highlighted, as well as the need for adequate data storage for future analysis. The conclusions indicate that the developed system provides a scalable solution, facilitating the management of Tablesa's IoT device network, and the recommendations suggest continuing the system's optimization and exploring new functionalities to further improve the company's operational efficiency and responsiveness.

Keywords: IoT, IIoT, OTA, Cloud Infrastructure, Serverless Architecture

TABLA DE CONTENIDO

1. INTRODUCCIÓN..... 14
1.2 PLANTEAMIENTO DEL PROBLEMA..... 16
1.3 OBJETIVOS..... 18
1.3.1 Objetivo general..... 18
1.3.2 Objetivos específicos..... 18
1.4 SOLUCIÓN PLANTEADA..... 18
2. ANTECEDENTES Y MARCO TEÓRICO..... 20
2.1 INTRODUCCIÓN..... 21
2.2 ANTECEDENTES..... 21
2.3 MARCO TEÓRICO..... 25
2.3.1 Diseño de software..... 26
2.3.2 IoT orientado a la industria..... 29
2.3.3 Ciclo de vida de los dispositivos IoT..... 30
2.3.4 Tecnologías y protocolos de comunicación..... 31
2.3.5 Metodología Ágil en el desarrollo de software SCRUM..... 35
2.4 OBSERVACIONES FINALES..... 36
3. DESARROLLO DEL SISTEMA DE OPERACIÓN Y GESTIÓN PARA LA FAMILIA DE DISPOSITIVOS IOT DE TABLESA S.A., ENFOCADOS EN CONSUMO ENERGÉTICO..... 38
3.1 INTRODUCCIÓN..... 39
3.2 DISEÑO DEL SISTEMA DE OPERACIÓN Y GESTIÓN PARA LA FAMILIA DE DISPOSITIVOS IOT DE TABLESA S.A..... 39
3.2.1 Requerimientos funcionales..... 39
3.2.2 Requerimientos no funcionales..... 41
3.2.3 Historias de usuario..... 41
3.2.4 Diagrama conceptual..... 48
3.2.5 Modelo de datos estructurado..... 49
3.3 IMPLEMENTACIÓN DEL SISTEMA DE OPERACIÓN Y GESTIÓN PARA LA FAMILIA DE DISPOSITIVOS IOT DE TABLESA S.A..... 51
3.3.1 Infraestructura del sistema..... 52
3.3.2 Interfaz Web de Visualización de Dispositivos IoT..... 59
3.3.3 Firmware..... 72
3.4 CONCLUSIONES..... 82

4. PRUEBAS Y RESULTADOS..... 84
4.1 INTRODUCCIÓN.....85
4.2 PRUEBAS DE INTEGRACIÓN..... 85
4.2.1 Puesta en marcha de dispositivos..... 85
4.2.2 Registro de dispositivos..... 89
4.2.3 Autenticación de dispositivos y actualización de firmware..... 92
4.2.4 Almacenamiento de datos adquiridos..... 94
4.2.5 Configuración remota de parámetros..... 97
4.3 CONCLUSIONES.....100
5. CONCLUSIONES Y TRABAJOS FUTUROS.....102
5.1 CONCLUSIONES.....103
5.1.1 Limitaciones..... 104
5.1.2 Desafíos.....104
5.2 TRABAJOS FUTUROS..... 105
6. REFERENCIAS..... 107
7. ANEXOS.....110

ÍNDICE DE FIGURAS

Figura 1.1 Infografía que ilustra los actores clave de la Industria 4.0 y sus objetivos..... 15
Figura 1.2 Diagrama de bloques de la solución planteada..... 19
Figura 2.1 Diagrama de arquitectura de referencia para microservicios RESTful en AWS..... 26
Figura 2.2 Arquitectura de referencia de monitoreo de condición de activos 28
Figura 2.3 Ciclo de vida útil de los dispositivos IoT..... 30
Figura 2.4 Diagrama de arquitectura de un lago de datos para energías renovables..... 34
Figura 2.5 Flujo de Scrum en un sprint..... 36
Figura 3.1 Diagrama conceptual del sistema de operación y gestión para dispositivos IoT de Tablesa S.A..... 49
Figura 3.2 Modelo de datos estructurado del sistema..... 50
Figura 3.3 Arquitectura general del sistema..... 52
Figura 3.4 Infraestructura del sistema..... 53
Figura 3.5 Estructura de archivo CSV para el registro de dispositivos..... 55
Figura 3.6 Sistema de registro de dispositivos IoT..... 55
Figura 3.7 Módulo de verificación de registro de dispositivos..... 56
Figura 3.8 Módulo de almacenamiento de datos adquiridos..... 57
Figura 3.9 Módulo de Actualización de Firmware..... 58
Figura 3.10 Infraestructura del aplicativo web..... 60
Figura 3.11 Pantalla de Inicio de sesión..... 61
Figura 3.12 Componente de información del dispositivo..... 62
Figura 3.13 Estados de conexión del dispositivo..... 63
Figura 3.14 Componente de barra de búsqueda..... 64
Figura 3.15 Resultado de la barra de búsqueda..... 65
Figura 3.16 Componente de Filtro de búsqueda..... 65
Figura 3.17 Pantalla principal - Listado de dispositivos..... 66
Figura 3.18 Pantalla de registro de dispositivos..... 68
Figura 3.19 Estados de solicitud de registro de dispositivos..... 69
Figura 3.20 Menú desplegable de ajustes de un dispositivo..... 69
Figura 3.21 Pantalla de configuración de parámetros..... 71
Figura 3.22 Modularización de firmware base y sus dependencias..... 72

Figura 3.23 Particiones de memoria del ESP32 con 16MB de almacenamiento para Arduino IDE.....75
Figura 3.24 Ubicación de archivo de configuración de tarjetas de Arduino IDE para ESP32..... 76
Figura 3.25 Configuración de partición de 16MB para la tarjeta ESP32 Dev Module de Arduino IDE.....77
Figura 3.26 FlashTools para formateo y carga inicial de firmware en múltiples dispositivos.....78
Figura 3.27 Configuración de red WiFi mediante Punto de Acceso generado por el dispositivo IoT..... 79
Figura 3.28 Distribución de tareas en núcleos del ESP32 consumo de recursos y periféricos.....81
Figura 3.29 Modularización de firmware completo y sus dependencias..... 81
Figura 4.3 Selección de chip y modo de trabajo en FlashTools..... 86
Figura 4.4 Distribución de memoria y archivos compilados para ESP32.....87
Figura 4.5 Estado inicial de panel de conexión de puerto serial 1..... 87
Figura 4.6 Estado de sincronización de conexión de panel 1..... 87
Figura 4.7 Estado de carga de firmware base de panel 1..... 87
Figura 4.8 Estado de finalización de carga de firmware de panel 1..... 88
Figura 4.9 Pasos 1 y 2 de conexión a punto de acceso del dispositivo IoT.... 88
Figura 4.10 Archivo de excel test.csv..... 90
Figura 4.11 Exportar como CSV desde Microsoft Excel..... 90
Figura 4.12 Botón de registro de dispositivos.....90
Figura 4.13 Pasos 1 y 2 del formulario de registro de dispositivos.....91
Figura 4.14 Paso 3. Carga de archivo .csv..... 91
Figura 4.15 Opciones para cargar el archivo seleccionado..... 91
Figura 4.16 Mensaje de registro completado exitosamente..... 91
Figura 4.17 Verificación de registro usando la interfaz web..... 92
Figura 4.18 Dispositivo en estado “Registrado”.....93
Figura 4.19 Proceso de configuración WiFi en el punto de acceso del dispositivo IoT..... 94
Figura 4.20 Dispositivo en estado “Conectado”..... 94
Figura 4.21 Muestra de datos adquiridos el día 28 de mayo del 2024 en el minuto 23:51..... 96
Figura 4.22 Muestra de registros de estados de conexión el día 28 de mayo del 2024.....97
Figura 4.23 Botón de ajustes del dispositivo..... 98

Figura 4.24 Sincronización de interfaz con dispositivo para configuración de parámetros..... 99
Figura 4.25 Edición de parámetro “Etiqueta” en dispositivo IoT.....99
Figura 4.26 Visualización de actualización de parámetro “Etiqueta” en un dispositivo.....100
Figura 7.1 Primer lote de dispositivos preparados para la etapa de carga de firmware.....112
Figura 7.2 Proceso de carga de firmware a 97 dispositivos mediante un puerto serial usando el programa FlashTools.....112

ÍNDICE DE TABLAS

Tabla 2.1 Comparación de Antecedentes en el Desarrollo de Sistemas IoT.... 21
Tabla 3.1 Sprint 01 - Febrero 2 a 15 - Entorno de desarrollo..... 42
Tabla 3.2 Sprint 02 - Febrero 16 a 29 - Firmware Base..... 42
Tabla 3.3 Sprint 03 - Marzo 1 a 14 - Firmware Completo..... 43
Tabla 3.4 Sprint 04 - Marzo 15 a 28 - Firmware Completo..... 44
Tabla 3.5 Sprint 05 - Marzo 29 a Abril 11 - Infraestructura del sistema..... 45
Tabla 3.6 Sprint 06 - Abril 12 a 25 - Infraestructura del sistema..... 45
Tabla 3.7 Sprint 07 - Abril 26 a Mayo 9 - Interfaz de administración..... 46
Tabla 3.8 Sprint 08 - Mayo 10 a 23 - Pruebas..... 47
Tabla 4.1 Prueba de puesta en marcha de 97 dispositivos IoT..... 84
Tabla 4.2 Prueba de Registro de dispositivos IoT a partir de una lista de IDs 88
Tabla 4.3 Prueba de Autenticación de dispositivos..... 91
Tabla 4.4 Prueba de Almacenamiento de datos adquiridos..... 94
Tabla 4.5 Prueba de Configuración remota de parámetros..... 96
Tabla 7.1 Ficha técnica OneMeter V1.0..... 110
Tabla 7.2 Tiempos en segundos de puesta en marcha de 47 dispositivos.... 112

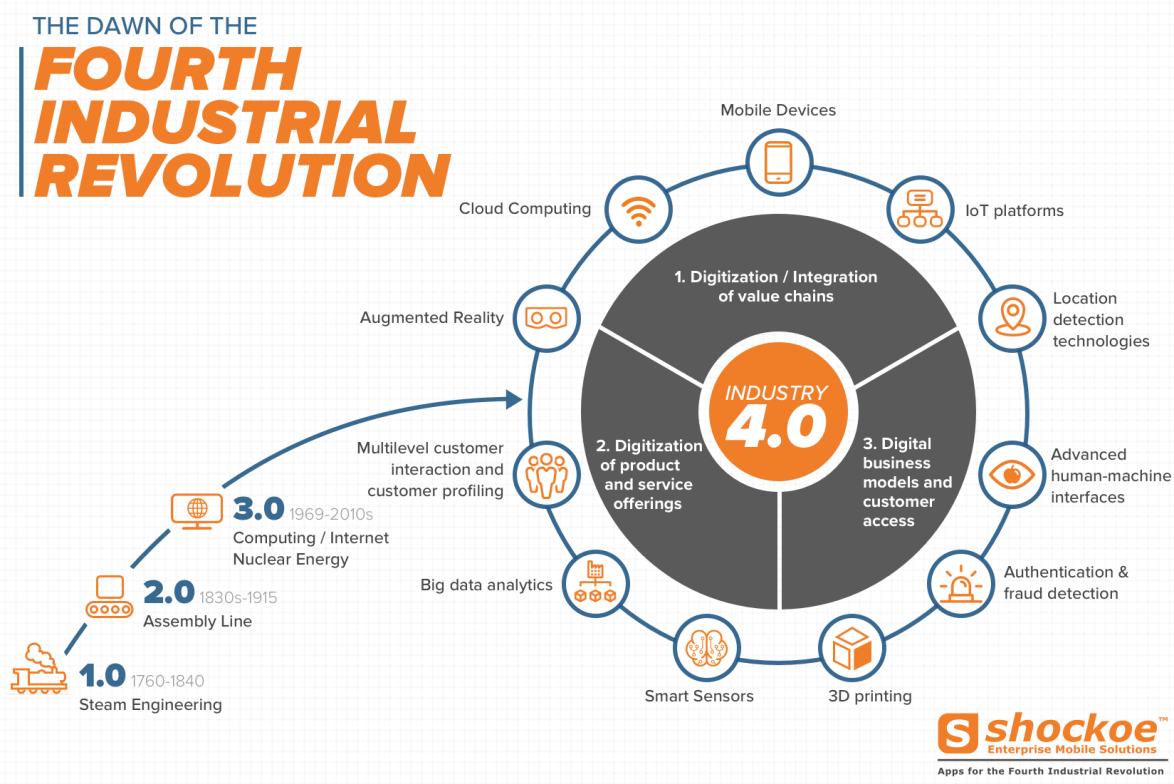
1.
INTRODUCCIÓN

> *"Solo podemos ver un poco del futuro, pero lo suficiente para darnos cuenta de que hay mucho por hacer."*
>
> *- Alan Turing*

En el contexto contemporáneo de la Industria 4.0, el Internet de las Cosas (IoT) se ha consolidado como un pilar fundamental para la digitalización de procesos industriales. La Figura 1.1 ilustra las principales tecnologías presentes en la actualidad que integra la industria 4.0 para mejorar la eficiencia operativa en diversos sectores. Este avance tecnológico promete solucionar problemas críticos como los periodos de inactividad no programados, al tiempo que ofrece herramientas para el mantenimiento prescriptivo en ambientes de planta. [1]

Figura 1.1

Infografía que ilustra los actores clave de la Industria 4.0 y sus objetivos



Nota. La cuarta revolución industrial (Moore, 2019) https://www.researchgate.net/figure/The-Fourth-Industrial-Revolution-Moore-2019_fig2_359250374 [accessed 23 May, 2024]

En esta línea, emergen empresas como Tablesa S.A. que se dedican a proporcionar sistemas integrados de dispositivos IoT con un enfoque específico en la gestión del consumo energético. Sin embargo, la expansión en la implementación de estos dispositivos plantea desafíos considerables en la operación y gestión de una red IoT a gran escala. Con aproximadamente 200 dispositivos en funcionamiento y una creciente demanda, Tablesa enfrenta la necesidad de responder ante la administración de cantidades de

dispositivos cada vez más grandes y resolver entre dos a tres peticiones diarias de mantenimiento.

Mi principal motivación con el desarrollo de este proyecto radica en la oportunidad de aplicar mis conocimientos de ingeniería electrónica para construir una solución tangible que aborde desafíos reales dentro de la industria colombiana. Este proyecto me permitió poner a prueba los conocimientos adquiridos durante mi carrera y también representó un escenario ideal para verificar su efectividad en la resolución de problemas prácticos y su contribución a la transformación de los procesos industriales. Aspirar a ser ingeniero electrónico implica utilizar esos conocimientos para influir positivamente en otras áreas profesionales, creando soluciones que mejoren la calidad de vida y fomenten el desarrollo económico.

En el transcurso de este documento compartiré con ustedes el proceso por el cual se desarrolló el sistema de operación y gestión para los dispositivos IoT que Tablesa implementa en entornos industriales. Este sistema se enfocó en aspectos cruciales como la autenticación segura de dispositivos, actualizaciones de firmware over-the-air (OTA) [2], almacenamiento estructurado de los datos adquiridos y la visualización gráfica del estado operativo de cada dispositivo. Estos aspectos permitieron una mejora en tiempos de comercialización, respuesta ante fallas y costos laborales.

Para el desarrollo de este proyecto se empleó el marco ágil para la gestión de proyectos, SCRUM. [3] Esto nos permitió incorporar de manera ágil y efectiva las mejores prácticas del ciclo de vida del desarrollo de software y conceptos como la Integración Continua y Despliegue Continuo (CI/CD). [4] Este enfoque permitió un desarrollo iterativo y adaptativo, fomentando la colaboración interdisciplinaria y permitiendo ajustes rápidos ante cualquier desafío emergente garantizando un desarrollo más fluido y un despliegue eficiente del firmware en los dispositivos IoT.

1.2 PLANTEAMIENTO DEL PROBLEMA

En la era de la transformación digital, la adopción masiva de dispositivos IoT ha brindado oportunidades sin precedentes para mejorar la eficiencia y la conectividad en diversos sectores empresariales. Sin embargo, a medida que las organizaciones implementan flotas de dispositivos IoT en sus operaciones, se enfrentan a desafíos significativos en la programación, gestión y análisis de estos dispositivos. [2] Tablesa, una empresa colombiana emergente en el campo del IoT, con cerca de 200 dispositivos en campo, enfrenta desafíos cruciales en las etapas de operación y gestión del ciclo de

vida de sus dispositivos, especialmente al atender pedidos que superan esta cantidad. Actualmente, la preparación de un lote de dispositivos se traduce en una carga laboral intensiva ya que implican procesos manuales como la carga de firmware, carga de cifrados, configuración, pruebas de comunicación y validación de equipos. Estos procesos requieren contratar personal altamente capacitado para realizar los pasos necesarios por cada dispositivo. Un lote de 20 dispositivos equivale a un día de trabajo por persona, lo cual se escala drásticamente cuando se contempla la preparación de 500, 1000 o más. Esta ineficiencia no sólo conlleva repercusiones económicas significativas en términos de costos laborales, sino que también impacta en el tiempo de comercialización y en la capacidad de respuesta de la empresa. Como solución a esta problemática, el mercado ofrece diversas alternativas:

Empresas que ofrecen servicios integrales de IoT: Como Link Labs [5], que provee una solución completa y centralizada para la gestión de dispositivos IoT. Sin embargo, estas soluciones generalmente están condicionadas al uso de dispositivos fabricados por la misma empresa, limitando la flexibilidad en la selección de hardware.

Soluciones parciales específicas para etapas del ciclo de vida de los dispositivos IoT: Tal como Monitev [6], que se centra en el monitoreo y gestión de dispositivos. Aunque eficiente en su enfoque, carece de integración con etapas de procesamiento independientes, restringiendo el flujo de datos y su posterior análisis.

Herramientas de desarrollo especializadas en IoT: Amazon IoT Core [7] destaca en esta categoría al ofrecer una variedad de servicios que abarcan todas las etapas del ciclo de vida del dispositivo IoT, desde la autenticación hasta la visualización de datos. También ofrece una flexibilidad y control total de los datos, permitiendo una personalización profunda de la solución. Sin embargo, esta solución requiere de una implementación más compleja y personal especializado.

En este contexto y bajo los requerimientos específicos de Tablesa, surge la necesidad de desarrollar un sistema que permita la autenticación de dispositivos, actualizaciones de firmware, administración intuitiva para personal no técnico y un almacenamiento adecuado de datos para futuros análisis, todas ellas comprendidas en las etapas de operación y gestión del ciclo de vida de los dispositivos IoT. [8] La ingeniería electrónica, con su capacidad para fusionar hardware y software, se posiciona como la llave para articular soluciones innovadoras a estos retos.

Por lo tanto, la pregunta que guía esta investigación es: ¿Cómo desarrollar un sistema de operación y gestión para la familia de dispositivos IoT de Tablesa S.A., enfocados en consumo energético?

1.3 OBJETIVOS

Para el desarrollo de este proyecto se definieron los siguientes objetivos:

1.3.1 Objetivo general

Desarrollar un sistema de operación y gestión para la familia de dispositivos IoT de Tablesa S.A. enfocados en consumo energético.

1.3.2 Objetivos específicos

1. Establecer las necesidades actuales de Tablesa en relación con la autenticación, administración y actualización de sus dispositivos IoT, identificando áreas críticas y oportunidades de mejora con el fin de definir los requerimientos funcionales del sistema de operación y gestión.
2. Diseñar un sistema de operación y gestión que facilite la autenticación, actualización y administración segura de los dispositivos IoT, además de garantizar el almacenamiento estructurado de los datos generados.
3. Implementar el sistema de operación y gestión que facilite la autenticación, actualización y administración segura de los dispositivos IoT.
4. Definir un protocolo de pruebas de la solución propuesta evaluando su funcionalidad, y compatibilidad considerando los requerimientos inicialmente planteados.

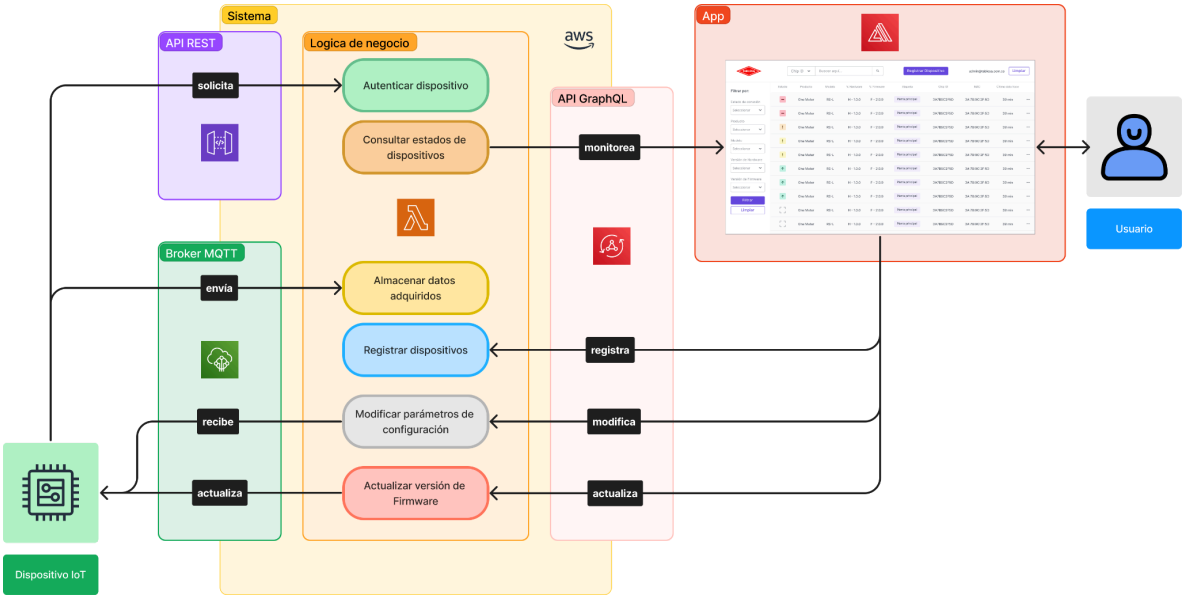
1.4 SOLUCIÓN PLANTEADA

La Figura 1.2 muestra un diagrama de bloques que ilustra la arquitectura y los componentes principales de la solución planteada. El sistema de operación y gestión para dispositivos IoT de Tablesa S.A. abarca tres componentes principales: una infraestructura en la nube que se encarga de la mensajería, autenticación y almacenamiento de datos; un firmware personalizado para la adquisición, transmisión y procesamiento de datos, con capacidades de actualización remota de firmware Over-The-Air (OTA), y una interfaz de usuario diseñada para facilitar la interacción con el sistema, que permita el

registro de nuevos dispositivos, la visualización del estado de los dispositivos y la gestión de su configuración de forma remota.

Figura 1.2

Diagrama de bloques de la solución planteada



2.

ANTECEDENTES Y MARCO TEÓRICO

"Para entender una ciencia, es necesario conocer su historia."

- Auguste Comte

2.1 INTRODUCCIÓN

El desarrollo del IoT avanza de forma creciente y constante. En sus inicios fue un concepto que carecía de estandarización; sin embargo, en el transcurso de los años se han propuesto múltiples arquitecturas de redes IoT [9], algunas con características de seguridad, otras haciendo énfasis en el despliegue multiplataforma o en propiedades de escalabilidad. No fue hasta finales del año 2018 que la Organización Internacional de Normalización (ISO) publicó su primer estándar para IoT, el cual fue revisado por última vez en el año 2021. [10] Gracias a la constante evolución e incursión en nuevas áreas de implementación del IoT, se torna necesario una revisión de estudios exploratorios que sinteticen los trabajos realizados. Esta revisión proporciona una perspectiva más amplia sobre las soluciones existentes y las estrategias que se han empleado para abordar problemas similares en el campo del IoT. Este capítulo muestra algunos de los trabajos que contribuyeron al desarrollo de este proyecto.

2.2 ANTECEDENTES

A continuación, se presenta una tabla comparativa de los antecedentes revisados. La Tabla 2.1 incluye las siguientes columnas: “Referencia” (Ref) o Número de referencia del estudio; “Objetivo” o propósito principal del estudio;
“Tecnologías” que muestra las herramientas, tecnologías o plataformas utilizadas en el estudio; “Aplicación” que indica el contexto en el cual se aplica el estudio; “Resultados” que describe los principales hallazgos del estudio; y por último “Relevancia” que describe cómo se relaciona o aplica el estudio al proyecto actual de Tablesa S.A.

Tabla 2.1
Comparación de Antecedentes en el Desarrollo de Sistemas IoT

Ref	Objetivo	Tecnologías	Aplicación	Resultados	Relevancia
[1]	Analizar enfoques de mantenimiento predictivo en la industria energética	Sensores IoT, Big Data	Industrial IoT	Técnicas de mantenimiento predictivo que mejoran la eficiencia operativa	Sistema de almacenamiento de datos adquiridos
[1], [11]	Arquitectura de actualización de firmware IoT	Actualización de firmware, Seguridad IoT	Seguridad IoT	Proceso seguro de actualización de firmware	Sistema de actualización de firmware

[2]	Diseñar actualizaciones OTA para dispositivos IoT	Servicios web RESTful en la nube	Gestión de dispositivos IoT	Sistema de actualización de firmware mediante OTA seguro y eficiente	Sistema de actualización de firmware
[12]	Mecanismo de actualización OTA para ESP8266	Microcontroladores ESP8266, WiFi	Gestión de dispositivos IoT	Actualización de firmware para dispositivos embebidos	Sistema de actualización de firmware
[13]	Revisar dimensiones de privacidad en IIoT	Protocolos de seguridad IoT	Seguridad IoT	Importancia de la privacidad y seguridad en IIoT	Mejores prácticas de seguridad para el sistema IoT de Tablesa
[13], [14], [15]	Autenticación basada en IoT en la nube	Autenticación IoT, Cloud Computing	Seguridad IoT	Autenticación en dispositivos IoT	Módulo de autenticación de dispositivos IoT
[16]	Analizar técnicas de Big Data en IoT	Arquitectura Big Data	Data Analytics	Importancia de analítica de datos en IoT	Diseño de sistema IoT pensado en la integración de análitica de datos
[17]	Plataforma de monitoreo de alumbrado público	Aplicación web, Sensores IoT	Gestión de dispositivos IoT	Monitoreo y control eficiente de alumbrado	Interfaz Web de monitoreo de dispositivos IoT
[18]	Sistema de ajuste automático de iluminación	IA, Sensores IoT	Gestión de dispositivos IoT	Optimización de eficiencia energética con IA	Módulos de transmisión, almacenamiento y visualización de datos
[19]	Monitoreo remoto de pacientes con IoT	IoT, Nube, Interfaz web	Gestión de dispositivos IoT	Gestión eficiente de grandes volúmenes de datos IoT	Módulos de transmisión, almacenamiento y visualización de datos

Como se observa en la Tabla 2.1, se ha realizado una comparación de diversos estudios relevantes en el ámbito del desarrollo de sistemas IoT. Cada estudio presenta diferentes objetivos, tecnologías utilizadas, aplicaciones, resultados clave y su relevancia específica para el proyecto de Tablesa S.A. A continuación, se ofrecerá una descripción breve de cada una de las referencias listadas, resaltando sus principales contribuciones y cómo sus hallazgos pueden ser aplicados o adaptados al desarrollo del sistema de operación y gestión para la familia de dispositivos IoT de Tablesa S.A.

Primero, en [1], los autores proporcionan un análisis exhaustivo de la evolución de los enfoques de mantenimiento en la industria de la energía. El documento aborda tanto las técnicas tradicionales como las basadas en análisis de Tecnologías de la Información (TI) para la detección de fallos y el mantenimiento predictivo. El artículo también presenta conceptos y soluciones que utilizan tecnologías emergentes relacionadas con la Industria 4.0, como el IoT, el análisis prescriptivo y Big Data.

Segundo, en [11], los autores abordan la necesidad en seguridad y actualización de firmware que surge en el contexto de la creciente interconexión de dispositivos IoT. La arquitectura propuesta permite capacidades importantes como la actualización de configuraciones, corrección de vulnerabilidades y la adición de nuevas funcionalidades. La adopción de tales mecanismos es esencial en el diseño y mantenimiento de dispositivos IoT, y representa un hito importante en la evolución de la tecnología IoT.

Tercero, en [2], los autores abordan el diseño y la gestión de actualizaciones de firmware Over-the-Air (OTA) para dispositivos IoT mediante servicios web RESTful basados en la nube. La investigación analiza la arquitectura y el mecanismo de las actualizaciones OTA, destacando sus ventajas y desventajas. Presenta un sistema de actualización y gestión de firmware, permitiendo la construcción, carga y liberación del firmware a los dispositivos.

Cuarto, en [12] los autores proponen un sistema para desarrollar y publicar actualizaciones de firmware seguras para dispositivos embebidos basados en microcontroladores ESP8266 a través de la misma conexión WiFi utilizada durante la operación normal y criptográficamente firmadas para prevenir la inyección de software malicioso. Este trabajo proporciona una base sólida para el desarrollo de mecanismos de actualización seguros y eficientes para dispositivos IoT.

Quinto, en [13], los autores proporcionan una visión integral de las dimensiones de privacidad en el Industrial Internet of Things (IIoT). La investigación aborda la privacidad industrial en los ecosistemas IIoT, explorando técnicas contemporáneas y mejores prácticas para garantizar la privacidad y seguridad de los datos. Concluye ofreciendo una visión holística y recomendaciones para establecer un ecosistema IIoT seguro, enfatizando la importancia de comprender y aplicar estas dimensiones de privacidad para fortalecer las defensas contra amenazas y proteger datos sensibles.

Sexto, en [14], los autores proponen una nueva versión del esquema de autenticación basado en IoT en circunstancias de computación en la nube por Zhou et al. [15], la nueva versión resuelve problemas de seguridad tales como: autenticación mutua y acuerdo de clave de sesión. Este artículo permite observar cómo solucionar un problema de autenticación en una red IoT.

Séptimo, en [16], los autores proponen una arquitectura para el análisis de grandes datos de IoT, destacando la necesidad de técnicas eficientes para manejar el rápido crecimiento de los datos en entornos industriales. Además, se exploran las oportunidades y desafíos en la integración de IoT y big data. Este trabajo subraya la importancia de la analítica de grandes datos en la era de IoT y ofrece una visión detallada de cómo estas tecnologías pueden ser aplicadas en contextos industriales para mejorar la toma de decisiones y la eficiencia operativa.

Octavo, en [17], los autores desarrollan una plataforma de monitoreo para la gestión de alumbrado público inteligente en el campus de Thammasat University, Rangsit Campus, Tailandia. La investigación se enfoca en la implementación de una aplicación web que proporciona al personal del campus una interfaz para monitorear, controlar y recopilar datos de dispositivos y sensores en el área. Esta solución permite la implementación de etapas de análisis de datos posteriores.

Noveno, en [18], los autores expanden el trabajo de [17] implementando un sistema de ajuste automático de iluminación en la calle. La investigación se centra en la optimización de la eficiencia energética mediante el uso de inteligencia artificial (IA) para predecir los valores de luminancia y ajustar automáticamente la iluminación de la calle. La implementación de este sistema permitió la adquisición periódica de datos de iluminación en intervalos de 10 minutos, lo cual es adecuado para el monitoreo continuo de las condiciones de luminancia. Basado en estos datos, los ajustes automáticos en la iluminación se realizaron cada 20 minutos, lo que

demuestra una aplicación práctica del sistema de gestión de grandes cantidades de datos IoT con capacidades de integrar herramientas de aprendizaje automático.

Décimo, en [19], los autores presentan un modelo basado en la nube para el monitoreo remoto de pacientes, enfocándose en los desafíos asociados con la gestión de grandes volúmenes de datos generados por dispositivos IoT. El sistema propuesto consta de tres partes principales: la recolección de datos, la red (funciones de procesamiento y almacenamiento) y la aplicación (interfaz para trabajadores de la salud y cuidadores). Esta investigación es relevante para el proyecto en cuestión, ya que aborda directamente problemáticas encontradas en el ciclo de vida de los datos desde su adquisición hasta su visualización.

La investigación y el desarrollo en el ámbito del IoT han experimentado un crecimiento significativo y complejo en los últimos años, abarcando diversas áreas como la seguridad, la gestión de actualizaciones de firmware, la privacidad industrial y la integración con tecnologías emergentes como Big Data e inteligencia artificial. A través de una revisión exhaustiva de los trabajos mencionados, se pueden observar distintas aproximaciones y metodologías dirigidas hacia desarrollos a medida, que reflejan la naturaleza adaptable y específica de los sistemas IoT. En el caso particular de Tablesa, una empresa que ya cuenta con una infraestructura de hardware y firmware en etapas avanzadas, la solución más coherente y sostenible parece ser la adopción de un Desarrollo de Software a Medida. Tal enfoque permitiría una integración y adaptación con los componentes y sistemas ya existentes, garantizando así una alineación más adecuada con las necesidades y objetivos de la empresa.

2.3 MARCO TEÓRICO

En esta sección se recopila la base conceptual y metodológica necesaria para entender y desarrollar un sistema de operación y gestión para dispositivos IoT en el contexto de la Industria 4.0. Esta sección examina los principios fundamentales del diseño de software, la integración de tecnologías IoT en entornos industriales, el ciclo de vida de los dispositivos IoT, y las tecnologías y protocolos de comunicación que facilitan su implementación. Al explorar estos conceptos, se establece un entendimiento claro de los retos y oportunidades presentes en la gestión de dispositivos IoT, así como las mejores prácticas y soluciones tecnológicas que pueden ser aplicadas en etapas operativas.

2.3.1 Diseño de software

El diseño de software es un proceso crítico en el desarrollo de sistemas, que consiste en definir la arquitectura, componentes, interfaces y otros elementos necesarios para cumplir con los requerimientos del proyecto. Este proceso involucra la creación de modelos y estructuras que guían la construcción del software, asegurando que sea funcional, escalable y mantenible. El diseño de software abarca desde la macroestructura de la aplicación (arquitectura) hasta los detalles específicos de los componentes individuales, permitiendo que los desarrolladores traduzcan los requisitos funcionales en una solución técnica efectiva. [20] En este contexto, es recomendable basarse en estructuras consolidadas como lo son las 'Arquitecturas de Referencia'. Un ejemplo de arquitectura de referencia recomendada podemos verla en la Figura 2.1 que ilustra el escenario de implementación de un microservicio RESTful en Amazon Web Services (AWS) para poder realizar consultas a una base de datos desde una interfaz web.

Figura 2.1
Diagrama de arquitectura de referencia para microservicios RESTful en AWS

https://docs.aws.amazon.com/images/wellarchitected/latest/serverless-applications-lens/images/reference-architecture-for-restful-microservices.png

Esta arquitectura para microservicios en la nube, está diseñada principalmente para utilizar servicios gestionados en la mayor medida posible, reduciendo el trabajo pesado asociado con la construcción desde cero, gestión de la seguridad, mantenimiento de servidores y la escalabilidad del sistema. Estas arquitecturas proporcionan un marco estandarizado que facilita la implementación de soluciones robustas y eficientes, asegurando coherencia y mejores prácticas en el diseño del sistema. [21]

2.3.1.1 Arquitectura de referencia

El término arquitectura de referencia define el conocimiento, los procesos, los métodos y las tecnologías sobre cómo diseñar un sistema en un dominio de aplicación determinado. Por lo tanto, una arquitectura de referencia guía y

apoya a los diseñadores, permitiendo una homogeneidad en la comunicación entre las diferentes partes que componen el sistema buscado. [22], [23] Las primeras arquitecturas IoT se adaptaron de la arquitectura subyacente de los sistemas de almacenamiento de datos, ya que tienen los componentes necesarios que conforman una arquitectura básica de los sistemas IoT. En la arquitectura del almacén de datos, algunos nodos generan datos (como un ERP - Enterprise Resource Planning) y una interfaz que captura, adapta e integra los datos generados. También hay un lugar de concentración y almacenamiento de datos, una interfaz que permite consultar los datos y, finalmente, los nodos que consumen los datos. [24]

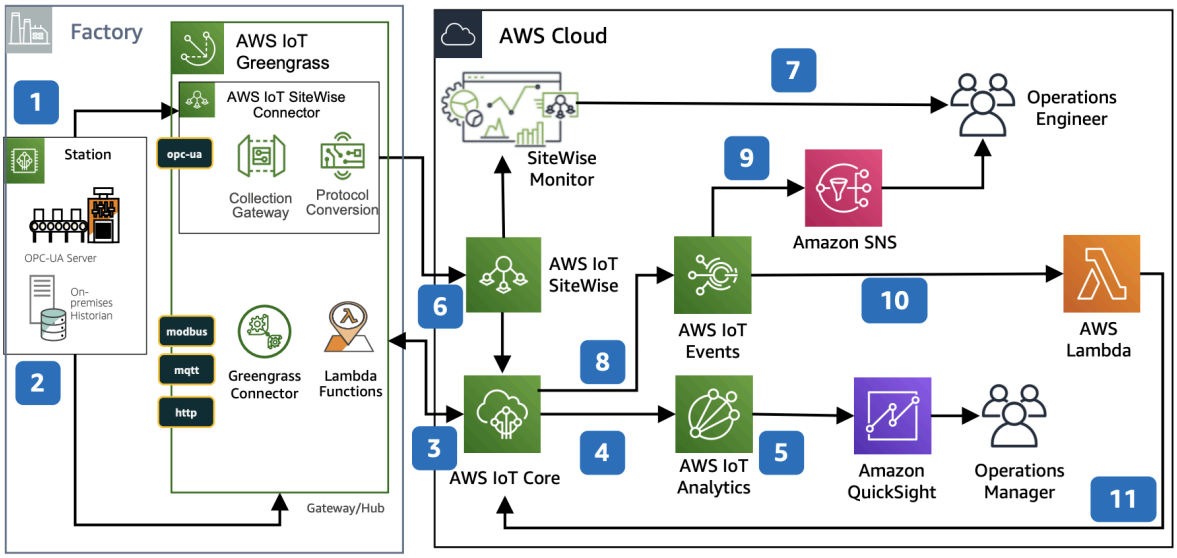
Podemos dividir la arquitectura base de IoT en hardware, software y arquitecturas de procesos. La arquitectura de hardware es compatible con el entorno informático distribuido, como peer-to-peer (P2P), EPCglobal y autónomo. La arquitectura de software proporciona un modelo mediante el cual todas las funcionalidades de las diferentes capas del sistema se entregan como servicios, como la Arquitectura Orientada a Servicios (SOA). La arquitectura del proceso define un modelo de sistema con procesos (por ejemplo, flujo de trabajo). [25] Además, la arquitectura base de IoT se puede dividir en cuatro capas: (i) detección; (ii) red; (iii) servicios; y (iv) interfaz. La capa de detección abarca la adquisición (sensores) y la actuación (actuadores), que se ocupa principalmente del hardware. La capa de red es responsable de soportar la comunicación (por ejemplo, protocolos y control de flujo). La capa de servicios está destinada a controlar y gestionar las funcionalidades utilizadas y proporcionadas. Finalmente, la capa de interfaz es responsable de administrar los métodos de interacción con los usuarios o con otros sistemas.

2.3.1.2 Arquitectura Serverless de microservicios

Para el desarrollo del sistema, se adoptó una arquitectura serverless de microservicios, inspirada en las prácticas recomendadas del dominio IoT y alineada con las modernas tendencias de arquitectura en la nube. Específicamente, el diseño se basó en una arquitectura de referencia proporcionada por AWS, que se encuentra detallada en la Figura 2.2.

Figura 2.2

Arquitectura de referencia de monitoreo de condición de activos



Nota. Solución para monitorear la salud de los equipos de fábrica, detectar condiciones de falla y responder a eventos usando AWS IoT SiteWise, AWS IoT Greengrass, AWS IoT Core y AWS IoT Events. https://d1.awsstatic.com/architecture-diagrams/ArchitectureDiagrams/aws-industrial-asset-condition-monitoring-ra.pdf

Esta arquitectura de referencia destaca por su integración y manejo de activos industriales a través de la nube, ofreciendo un marco estructural para la monitorización de condiciones y la gestión automatizada de dispositivos IoT. La elección de esta arquitectura de referencia permitió implementar un sistema orientado a eventos con una base sólida en la recopilación y el análisis de datos, y la interacción segura y continua entre los componentes del sistema.

Características de la Arquitectura Serverless de microservicios:

- **Desacoplamiento:** Cada función serverless actúa como un microservicio, siendo independiente en términos de desarrollo y operación. Esto facilita la gestión de cambios y la actualización de sistemas sin interrupciones significativas.
- **Autogestión:** La infraestructura subyacente y la escalabilidad son gestionadas por el proveedor de servicios en la nube, lo que reduce la carga operativa y permite a los desarrolladores centrarse en la lógica de negocio.

- **Modelo de pago por uso:** Como en cualquier entorno serverless, pagas solo por lo que usas, lo que mejora la eficiencia de costos, especialmente en aplicaciones con uso variable o impredecible.

Ventajas de Combinar Microservicios con Serverless:

- **Agilidad y Escalabilidad:** La combinación permite una escalabilidad y flexibilidad excepcionales, con servicios que pueden escalar automáticamente en respuesta a la demanda sin requerir intervención manual.
- **Mantenimiento Reducido:** Al minimizar las necesidades de gestión de infraestructura, los equipos pueden dedicar más recursos a innovar y mejorar la funcionalidad del producto.
- **Resiliencia y Aislamiento de Fallas:** El fallo en una función serverless afecta menos al sistema global, similar a cómo los microservicios pueden fallar de manera aislada sin impactar en toda la aplicación.

Consideraciones de Implementación:

- **Diseño de Base de Datos:** Cada función puede operar dentro de su propio contexto limitado, gestionando sus propios datos o interactuando con subconjuntos de la base de datos para mantener cierto nivel de independencia.
- **Gestión de Estado:** La gestión del estado puede ser un desafío, ya que los microservicios y las funciones serverless son preferiblemente stateless. Aquí, servicios como S3 o bases de datos gestionadas pueden almacenar estados externos.

2.3.2 IoT orientado a la industria

El Industrial Internet of Things (IIoT) se centra en la aplicación de tecnologías IoT en entornos industriales, conectando maquinaria, sensores y dispositivos para recopilar y analizar datos en tiempo real. Esta integración permite mejorar la eficiencia operativa, reducir costos y aumentar la productividad. El IIoT es un componente clave de la Industria 4.0, que representa la cuarta revolución industrial, caracterizada por la digitalización y la automatización avanzada de los procesos industriales. [1]

Uno de los principales impactos del IIoT en la Industria 4.0 es su capacidad para transformar los enfoques de mantenimiento. Tradicionalmente, el mantenimiento se realizaba de manera reactiva, es decir, después de que ocurría una falla. Sin embargo, con la implementación del IIoT, es posible

adoptar enfoques de mantenimiento predictivo y prescriptivo. Estos enfoques utilizan datos en tiempo real y algoritmos avanzados para prever fallas antes de que ocurran y sugerir acciones preventivas, mejorando significativamente la disponibilidad y confiabilidad de la maquinaria industrial. [13]

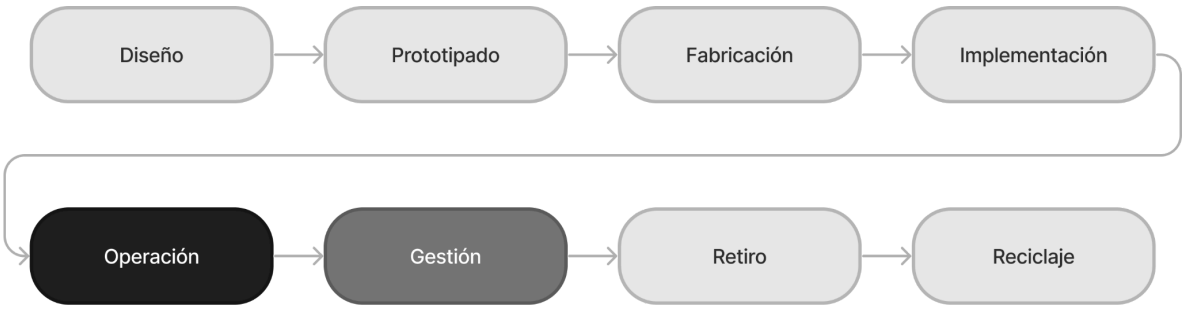
El IIoT también juega un papel crucial en la detección rápida de fallas, lo que es esencial para minimizar los tiempos de inactividad no planificados. Al integrar sensores inteligentes y sistemas de monitoreo continuo, las empresas pueden detectar anomalías en el comportamiento de la maquinaria casi instantáneamente. Esta capacidad de respuesta rápida reduce el tiempo y los costos asociados con el mantenimiento correctivo, además de optimizar la producción y mejorar la calidad del producto final. [1]

Además de los beneficios operativos, el IIoT también plantea consideraciones importantes en términos de seguridad y privacidad de los datos. La gran cantidad de datos generados por los dispositivos conectados debe ser protegida contra accesos no autorizados y brechas de seguridad. La implementación de robustas medidas de seguridad y prácticas de gestión de datos es crucial para garantizar la integridad y confidencialidad de la información industrial. [13]

2.3.3 Ciclo de vida de los dispositivos IoT

El ciclo de vida de los dispositivos IoT abarca todas las etapas desde su concepción hasta su retiro y reciclaje. Estas etapas incluyen diseño, prototipado, fabricación, implementación, operación, gestión, retiro y reciclaje. La Figura 2.3 muestra la secuencia de las etapas donde cada una de estas es crucial para asegurar que los dispositivos IoT funcionen de manera eficiente y segura durante su vida útil. [8]

Figura 2.3
Ciclo de vida útil de los dispositivos IoT



Estas etapas se describen a continuación:

- **Diseño y Prototipado:** En las primeras etapas, se definen los requisitos funcionales y no funcionales del dispositivo, seguidos por la creación de prototipos para validar el diseño y realizar pruebas iniciales. Estas fases son fundamentales para garantizar que el dispositivo cumpla con los estándares de calidad y rendimiento esperados.
- **Fabricación:** Una vez que el prototipo ha sido validado, el dispositivo entra en la fase de fabricación, donde se producen en masa siguiendo los diseños y especificaciones aprobados. Esta etapa requiere un control de calidad riguroso para asegurar la consistencia y fiabilidad del dispositivo.
- **Implementación:** Esta fase involucra la instalación y configuración de los dispositivos en su entorno operativo. Es crucial que se realice de manera efectiva para minimizar problemas futuros en las etapas de operación y gestión.
- **Operación:** Durante la operación, los dispositivos IoT recolectan y transmiten datos en tiempo real. Aquí es donde se enfrentan desafíos significativos, especialmente cuando se manejan grandes volúmenes de dispositivos y datos. La administración eficaz de los dispositivos incluye asegurar su conectividad continua, la actualización de firmware y la autenticación segura.
- **Gestión:** La gestión implica supervisar el rendimiento y la salud de los dispositivos, identificando y resolviendo problemas rápidamente. Los retos en esta fase incluyen la administración de grandes volúmenes de datos, la identificación de errores y la realización de mantenimiento predictivo y preventivo. La capacidad para gestionar estos aspectos de manera eficiente es crítica para maximizar la vida útil de los dispositivos y asegurar su rendimiento óptimo.
- **Retiro y Reciclaje:** Finalmente, cuando los dispositivos alcanzan el final de su vida útil, deben ser retirados y reciclados de manera segura para minimizar el impacto ambiental y recuperar materiales valiosos.

2.3.4 Tecnologías y protocolos de comunicación

En el IoT, los dispositivos y las redes se integran para formar una red informática. Tales dispositivos imponen varias restricciones gracias a las limitaciones en procesamiento, memoria y consumo de energía. Los protocolos tradicionales de las redes de computadoras, como el Protocolo de Control de Transmisión (TCP), no manejan bien estas limitaciones, dada la sobrecarga impuesta por las capas de procesos en esos protocolos. Por lo

tanto, Internet Engineering Task Force (IETF) ha definido y desarrollado paquetes de aplicaciones para dispositivos con limitaciones de recursos. Ejemplos de protocolos desarrollados por IETF son el Protocolo de aplicación restringida (CoAP) y el Protocolo de enrutamiento para redes de baja potencia y Lolssy (RPL). [26] Otros protocolos también se utilizan en el desarrollo de soluciones para IoT, especialmente cuando el foco está en la comunicación M2M. Estos protocolos incluyen el transporte de telemetría de mensajes en cola (MQTT), el protocolo avanzado de mensajes en cola (AMQP), el protocolo extensible de mensajes y presencia (XMPP) y el servicio de distribución de datos (DDS), así como los servicios de transferencia de estado representativo (REST ó RESTful) y Arquitectura Unificada OPC (OPCUA). [27]

En esta sección se han mencionado varias tecnologías y protocolos de comunicación que podrían ser utilizados en el contexto del IoT. Sin embargo, dado el enfoque particular del proyecto y las limitaciones en términos de ancho de banda, capacidad de procesamiento y escalabilidad, se ha seleccionado MQTT como la opción más adecuada. A continuación, se expone en detalle la justificación de esta selección, así como las ventajas de MQTT frente a otros protocolos, en función de los requerimientos específicos del proyecto.

2.3.4.1 MQTT

Es un protocolo de mensajería asíncrono basado en publicación y suscripción desarrollado por IBM para la comunicación M2M. Este protocolo es confiable y ofrece tres niveles de QoS: (i) disparar y olvidar, que requiere reconocimiento para cada mensaje; (ii) al menos una vez, lo que requiere el reconocimiento de un mensaje que se envía al menos una vez; y (iii) entregado exactamente una vez, asegurando que un mensaje se entregue solo una vez. MQTT se ejecuta sobre el protocolo TCP y la seguridad se realiza utilizando los servicios de protocolo. [28]

MQTT se usa ampliamente en aplicaciones IoT por varias razones en comparación con otros protocolos, especialmente en relación a los requerimientos específicos del proyecto de Tablesa:

- **Eficiencia en el uso del ancho de banda:** MQTT está diseñado para ser extremadamente ligero, lo que lo hace ideal para dispositivos con recursos limitados. Su estructura de mensajes minimiza la sobrecarga de datos, permitiendo una comunicación eficiente y conservando el ancho de banda. En comparación, AMQP, aunque robusto, tiene una

mayor sobrecarga debido a su complejidad, lo que lo hace menos adecuado para dispositivos con restricciones de recursos y redes con ancho de banda limitado.

- **Frecuencia de transmisión:** MQTT es capaz de manejar frecuencias de transmisión superiores a 0.5 segundos de manera efectiva, lo cual es crucial para el monitoreo de dispositivos IoT de Tablesa. Por otro lado, CoAP, diseñado para comunicaciones más esporádicas y de baja frecuencia, puede no ser tan eficiente en escenarios donde se requiere transmisión de datos en intervalos cortos y constantes.
- **Gestión de grandes cantidades de dispositivos:** El modelo de publicación/suscripción de MQTT facilita la escalabilidad, permitiendo la gestión eficiente de grandes cantidades de dispositivos. Esto es esencial para Tablesa, que maneja cientos de dispositivos y anticipa un crecimiento continuo. En contraste, XMPP, aunque útil para la comunicación en tiempo real, puede volverse complicado y menos eficiente a medida que aumenta el número de dispositivos debido a su arquitectura más pesada y a la necesidad de manejo de presencia.
- **Detección de eventos de conexión/desconexión:** MQTT soporta la detección de eventos de conexión y desconexión de dispositivos a través de su característica de Last Will and Testament (LWT). LWT permite que un mensaje específico sea enviado automáticamente si un dispositivo se desconecta inesperadamente, proporcionando una notificación inmediata de fallas de conexión. Esta capacidad es vital para el monitoreo y mantenimiento continuo de una red IoT, permitiendo una respuesta rápida a cualquier interrupción en la red. En comparación, RESTful, aunque flexible y ampliamente utilizado, no ofrece una solución integrada para la detección de eventos de conexión/desconexión, requiriendo mecanismos adicionales para lograr la misma funcionalidad.
- **Flexibilidad en la Calidad de Servicio (QoS):** Los tres niveles de QoS que ofrece MQTT permiten ajustar el nivel de garantía de entrega de mensajes según los requerimientos de la aplicación, proporcionando un equilibrio entre fiabilidad y eficiencia. En contraste, OPC UA, aunque potente y adecuado para aplicaciones industriales, puede ser excesivo en términos de complejidad y recursos para escenarios donde una solución ligera y rápida es más beneficiosa.

En resumen, MQTT se destaca como el protocolo más adecuado en el contexto de los requerimientos específicos del proyecto debido a su eficiencia en el uso del ancho de banda, su capacidad para manejar frecuencias de transmisión altas, su escalabilidad para gestionar grandes volúmenes de dispositivos, su soporte para la detección de eventos de

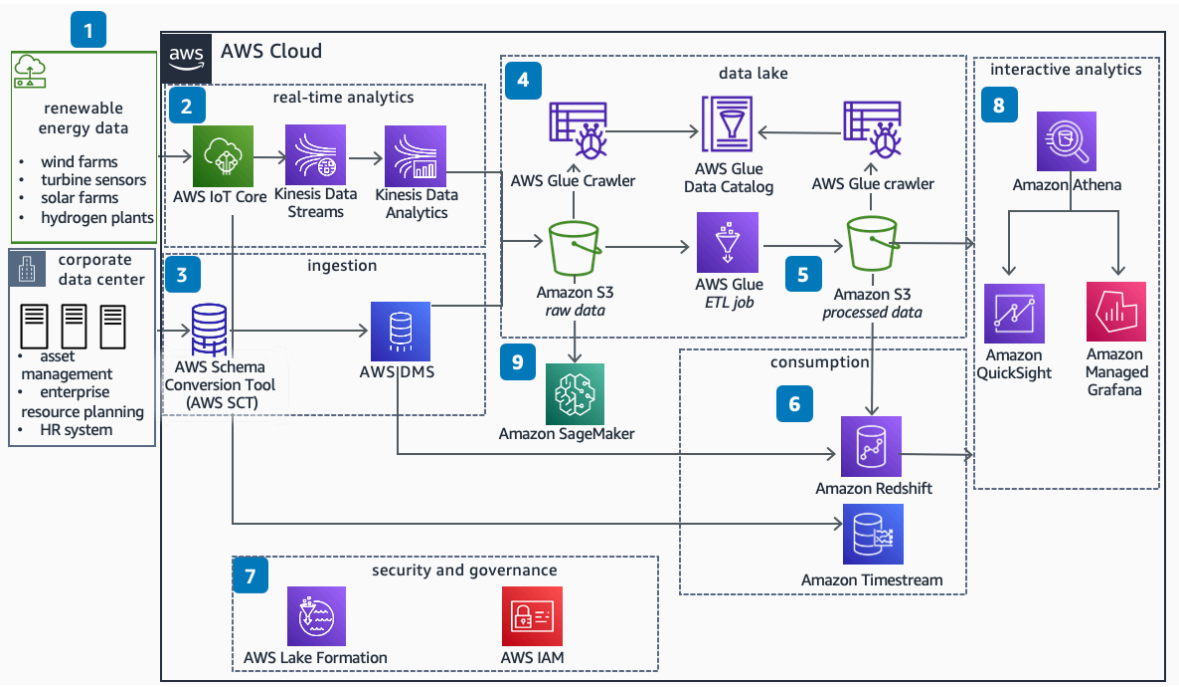
conexión/desconexión a través de LWT, y su flexibilidad en la calidad de servicio.

2.3.4.2 Plataforma IoT - AWS IoT

Amazon IoT o AWS IoT [29] es una plataforma basada en la nube que permite la interacción entre los dispositivos conectados, los servicios de AWS y otros dispositivos. Los dispositivos IoT pueden intercambiar mensajes con la plataforma AWS IoT utilizando los protocolos de transporte de telemetría de mensajes (MQTT), HTTP o WebSockets a través del SDK de dispositivos AWS IoT. La plataforma Amazon IoT proporciona autenticación mutua, intercambio seguro de datos, control de autorización basado en certificados X.509, tokens y registro. Además, AWS IoT tiene un motor de reglas que permite crear aplicaciones de IoT que recopilan, procesan, analizan y actúan sobre los datos generados por los dispositivos de IoT. [30] En la Figura 2.4 podemos ver una implementación de AWS IoT y toda su familia de servicios para crear un lago de datos de energía renovable, visualizarlos y hacer predicciones con modelos de aprendizaje automático.

Figura 2.4

Diagrama de arquitectura de un lago de datos para energías renovables



Nota. Diagrama de arquitectura de referencia que muestra cómo puede utilizar los servicios de AWS para visualizar datos y hacer predicciones con aprendizaje automático.

https://docs.aws.amazon.com/images/architecture-diagrams/latest/data-lake-architecture-for-renewable-energy/images/data-lake-architecture-for-renewable-energy.png

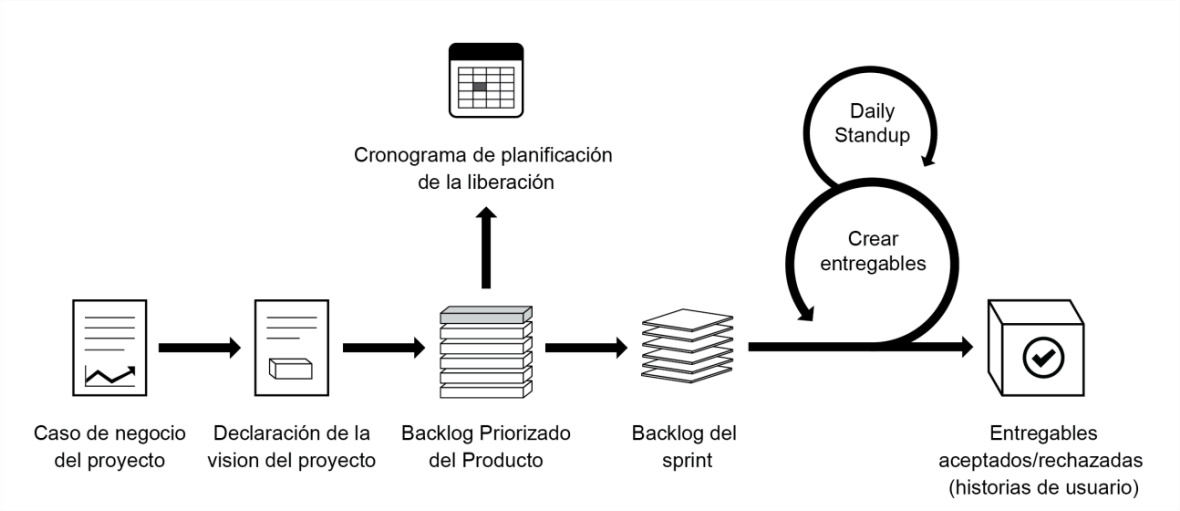
2.3.4.3 Tecnologías de la comunicación

La tecnología de la comunicación es uno de los pilares que permiten la existencia de IoT, y la conectividad es uno de los principales componentes del paradigma de IoT. Estas tecnologías permiten la comunicación entre dispositivos (M2M - Machine-to-Machine) y dispositivos con Internet. La comunicación inalámbrica permite la integración de varios sensores, actuadores y sistemas en IoT. La tecnología Wi-Fi se divide en convencional (802.11b / g / n), WiFi de baja potencia (802.11 ah) o HaLow, Redes celulares (3G, 4G y evolución a largo plazo - LTE) y Red de área amplia de baja potencia (LPWAN). La tecnología convencional tiene un alcance interior de 70 metros interior y un alcance exterior de 230 metros. Utiliza una topología en estrella y tiene una velocidad de bits de 1 Mbps a 2,4 o 5 GHz, con un rendimiento que varía de 2 a 50 Mbps. No se recomienda para sistemas de ultra baja potencia, pero se emplea ampliamente en ciudades inteligentes y aplicaciones M2M. La tecnología HaLow tiene una velocidad de bits más baja (de 150 a 400 Kbps a 900 MHz), con un rendimiento de hasta 100 Kbps. Sin embargo, consume menos energía y sufre menos interferencia operando a 900 MHz. También utiliza la topología en estrella, con un rango entre 700 y 1000 metros, y se emplea principalmente en ciudades inteligentes y M2M.

2.3.5 Metodología Ágil en el desarrollo de software SCRUM

El marco de trabajo SCRUM [3] representa una transformación radical en la metodología de desarrollo de proyectos, alejándose de los enfoques prescriptivos y top-down que han prevalecido en el pasado. SCRUM se inspira en sistemas evolutivos, adaptativos y autocorrectivos, y ha sido fundamental en la creación de software y productos en la industria tecnológica. A diferencia del método Waterfall, donde un proyecto se completa en etapas distintas y se mueve paso a paso hacia su liberación, SCRUM promueve la organización de proyectos en pequeños equipos, estableciendo "sprints" de una semana a un mes para promover la agilidad, la colaboración, la adaptabilidad y la entrega continua de valor en el desarrollo de proyectos. Además, incorpora el concepto de Producto Mínimo Viable para obtener retroalimentación inmediata de los consumidores. [4] La figura 2.5 proporciona una visión general del flujo de un proyecto Scrum.

Figura 2.5
Flujo de Scrum en un sprint



Nota. En un sprint el Equipo SCRUM *trabaja en la creación de entregables que potencialmente generen valor en el producto. [4]*

SCRUM ha demostrado ser una herramienta poderosa no solo en la tecnología sino también en la transformación de cómo las empresas trabajan, crean, planifican y piensan, ofreciendo resultados en menos tiempo y con mayor calidad.

2.4 OBSERVACIONES FINALES

El análisis de los antecedentes y el marco teórico del desarrollo de IoT revela la evolución y complejidad creciente de esta tecnología. A lo largo de los años, el IoT ha pasado de ser un concepto sin estandarización a un campo con múltiples arquitecturas y estándares bien definidos, como los promulgados por la ISO en 2018 y revisados en 2021. Esta evolución ha facilitado la adopción de IoT en diversos sectores industriales, permitiendo la implementación de soluciones más seguras, escalables y eficientes.

Los estudios revisados proporcionan una comprensión profunda de las diferentes aproximaciones y metodologías empleadas en el diseño y gestión de sistemas IoT. Por ejemplo, el uso de mantenimiento predictivo y prescriptivo, apoyado por tecnologías emergentes como el análisis de datos y el Big Data, ha demostrado ser fundamental para mejorar la eficiencia operativa y la confiabilidad en la industria energética y otros sectores. La implementación de actualizaciones de firmware Over-the-Air (OTA) y la gestión de la seguridad son aspectos críticos que han sido abordados de

manera efectiva en la literatura, destacando la importancia de mantener los dispositivos IoT actualizados y protegidos contra vulnerabilidades.

El diseño de software en el contexto de IoT se beneficia enormemente de las arquitecturas de referencia, que proporcionan un marco estandarizado y probado para el desarrollo de soluciones robustas. La adopción de una arquitectura serverless de microservicios, inspirada en las mejores prácticas del dominio IoT y apoyada por plataformas en la nube, permite una escalabilidad y agilidad excepcionales, facilitando el desarrollo y despliegue de componentes de manera independiente y eficiente.

El Industrial Internet of Things (IIoT) juega un papel crucial en la transformación de la Industria 4.0, permitiendo la automatización avanzada y la mejora de la productividad a través de la recopilación y análisis de datos en tiempo real. Los beneficios operativos del IIoT, como la detección rápida de fallas y el mantenimiento predictivo, se ven complementados por la necesidad de medidas de seguridad robustas para proteger la gran cantidad de datos generados.

El ciclo de vida de los dispositivos IoT, que abarca desde el diseño hasta el retiro y reciclaje, presenta desafíos significativos en las etapas de operación y gestión, especialmente cuando se manejan grandes volúmenes de dispositivos y datos. La administración eficaz de estos dispositivos, la identificación rápida de errores y la implementación de mantenimiento predictivo son esenciales para maximizar la vida útil y el rendimiento de los dispositivos IoT.

Finalmente, las tecnologías y protocolos de comunicación juegan un papel fundamental en la conectividad y funcionalidad de los dispositivos IoT. La evolución de estos protocolos, como MQTT y CoAP, permite una comunicación eficiente y segura entre dispositivos, soportando la amplia gama de aplicaciones y servicios que definen el ecosistema IoT actual.

En conclusión, el desarrollo de IoT ha avanzado significativamente, proporcionando soluciones innovadoras y eficientes para diversos retos industriales. Sin embargo, la gestión de la seguridad, la actualización de dispositivos y la administración de grandes volúmenes de datos siguen siendo áreas cruciales que requieren atención continua para asegurar el éxito y la sostenibilidad de las implementaciones IoT en el futuro.

3.

DESARROLLO DEL SISTEMA DE OPERACIÓN Y GESTIÓN PARA LA FAMILIA DE DISPOSITIVOS IOT DE TABLESA S.A., ENFOCADOS EN CONSUMO ENERGÉTICO

"La tecnología nos permite hacer grandes cosas, pero de nada sirve si no la usamos para mejorar el mundo."

- Elon Musk

3.1 INTRODUCCIÓN

En este capítulo, se describe el proceso de desarrollo del sistema de operación y gestión para los dispositivos IoT de Tablesa S.A., enfocados en el consumo energético. El sistema propuesto se ha desarrollado para abordar las necesidades presentes en las etapas de operación y gestión del ciclo de vida de los dispositivos IoT de Tablesa S.A. Estas etapas incluyen la autenticación, actualización de firmware, monitoreo remoto y gestión de los dispositivos. Otras fases del ciclo de vida, como el diseño, prototipado, fabricación, y retiro o reciclaje de los dispositivos, no forman parte del alcance de este proyecto, ya que el enfoque está en optimizar la eficiencia operativa y de gestión durante la vida útil de los dispositivos. Este sistema abarca desde el diseño inicial hasta la implementación y prueba del sistema, cubriendo aspectos como la infraestructura del sistema, el desarrollo de firmware, y la creación de una interfaz web para la visualización y gestión de dispositivos. Para mayor detalle sobre las características técnicas de los dispositivos IoT utilizados en este proyecto, incluyendo sus especificaciones y funciones en producción, se puede consultar la Tabla 7.1.

3.2 DISEÑO DEL SISTEMA DE OPERACIÓN Y GESTIÓN PARA LA FAMILIA DE DISPOSITIVOS IOT DE TABLESA S.A.

En esta etapa se define la arquitectura, los componentes y las interfaces necesarias para cumplir con los requisitos funcionales y no funcionales del sistema. La arquitectura del sistema se basa en una combinación de microservicios y tecnologías serverless, lo que proporciona escalabilidad y flexibilidad, facilitando la integración y el despliegue de nuevas funcionalidades. Además, se han implementado varias capas de seguridad y mecanismos de actualización para asegurar la integridad y el funcionamiento continuo de los dispositivos. En las subsecciones siguientes, se detallan los componentes clave del sistema, incluyendo los requerimientos funcionales y no funcionales, los diagramas conceptuales, el modelo estructurado de la base de datos, y los diferentes módulos de firmware y software desarrollados para el proyecto.

3.2.1 Requerimientos funcionales

En colaboración con Tablesa S.A., se identificaron y definieron varios requerimientos funcionales clave para el desarrollo del sistema de operación y gestión de dispositivos IoT. Estos requerimientos fueron esenciales para guiar el diseño y la implementación de las funcionalidades del sistema. Dos

conjuntos de requerimientos particulares que influyeron significativamente en el desarrollo del proyecto fueron los siguientes:

1. **Prueba de Dispositivos sin Compartir el Firmware Completo:** Tablesa S.A. necesitaba una manera de probar los dispositivos fabricados sin tener que compartir el firmware completo con las empresas fabricantes de dispositivos. Este requerimiento surgió de la necesidad de proteger la propiedad intelectual y mantener el control sobre el software crítico. Para cumplir con este requerimiento, se planteó el desarrollo de un firmware base que solo incluye las funcionalidades necesarias para probar el dispositivo, verificar su registro en el sistema, y permitir la descarga del firmware completo si el dispositivo está registrado. Esta solución permite a Tablesa probar los dispositivos de manera segura y eficiente, sin comprometer la integridad del firmware completo.
2. **Envío de Datos a Múltiples Destinatarios con Diferentes Periodos de Muestreo:** Tablesa S.A. requería que los dispositivos IoT pudieran enviar datos tanto al broker de Tablesa como al cliente que compró el dispositivo, pero con diferentes periodos de muestreo. Específicamente, se necesitaba un periodo de muestreo de 5 segundos para el broker de Tablesa y un periodo de muestreo de 0.5 segundos para el cliente. Para abordar este requerimiento, se planteó una doble conexión MQTT en el firmware de los dispositivos. Esto permite que los dispositivos envíen datos simultáneamente a ambos destinatarios con los periodos de muestreo especificados. Esta funcionalidad asegura que tanto Tablesa como sus clientes reciban los datos necesarios en los intervalos adecuados para sus respectivos propósitos.

Los otros requerimientos funcionales del sistema fueron los siguientes:

3. **Registro de dispositivos:** El sistema debe permitir a los usuarios registrar nuevos dispositivos IoT. El usuario proporciona los detalles del dispositivo y el sistema guarda esta información en la base de datos.
4. **Autenticación segura de dispositivos:** Los dispositivos IoT deben poder autenticarse de manera segura con el sistema central. Los dispositivos envían sus credenciales al sistema, que verifica su autenticidad antes de permitir el acceso a otras funcionalidades.
5. **Almacenamiento de datos adquiridos:** El sistema debe ser capaz de almacenar los datos adquiridos por los dispositivos IoT de manera estructurada y segura. Los dispositivos envían datos al sistema, que luego almacena en la base de datos.

6. **Monitoreo de dispositivos:** El sistema debe proporcionar una interfaz para que los usuarios puedan consultar o monitorear el estado operativo de los dispositivos IoT. Los dispositivos reportan su estado de conexión al sistema, que se puede ver reflejada en la interfaz web para los usuarios.
7. **Modificar parámetros de configuración:** Los usuarios deben tener la capacidad de modificar los parámetros de configuración de los dispositivos IoT a través del sistema. Al enviar nuevas configuraciones, el sistema las aplica a los dispositivos correspondientes.
8. **Actualización remota de firmware:** El sistema debe permitir la actualización de firmware de los dispositivos de manera remota. Los dispositivos IoT deben permitir la actualización remota de su Firmware.
9. **Interfaz de usuario intuitiva:** La interfaz web debe ser fácil de usar y permitir la gestión y monitoreo de los dispositivos.

3.2.2 Requerimientos no funcionales

Los requerimientos no funcionales del sistema incluyen:

1. **Escalabilidad:** El sistema debe ser capaz de manejar un gran número de dispositivos, garantizando un rendimiento óptimo a medida que la cantidad de dispositivos crece.
2. **Seguridad:** Todos los datos y comunicaciones deben estar protegidos contra accesos no autorizados mediante el uso de cifrado y autenticación segura.
3. **Confiabilidad:** El sistema debe ser robusto y capaz de funcionar de manera continua sin interrupciones significativas, asegurando alta disponibilidad y resistencia a fallos.

3.2.3 Historias de usuario

Los anteriores requerimientos funcionales fueron fundamentales para el desarrollo de las historias de usuario y la definición de la arquitectura del sistema. A continuación, se presentan una serie de tablas que contienen las historias de usuario (HU) realizadas en cada sprint. Para el desarrollo del proyecto se planificaron 8 sprints de 15 días cada uno, entre el 2 de febrero y el 23 de mayo del 2024. Las tablas incluyen las siguientes columnas: “ID” o Número de referencia de la HU; “Historia de usuario” que contiene una descripción corta de la tarea a realizar; “Objetivo” que muestra el propósito o resultado esperado de la tarea; y por último “Criterios de aceptación” que describe las condiciones que deben cumplirse para considerar la tarea como completada.

El proyecto utilizó una adaptación del marco SCRUM, en la cual se definieron claramente los roles del equipo, aunque fue un proyecto con un solo desarrollador principal. Los roles asignados fueron los siguientes:

- **Product Owner:** Duvan Acosta, encargado de definir los requerimientos y priorizar las funcionalidades.
- **Stakeholder:** Eval Bladimir, responsable de representar los intereses de la Universidad del Valle.
- **Development Team:** Carlos Gaviria, quien desempeñó el rol de desarrollador principal y ejecutó la implementación técnica.
- **Scrum Master:** Carlos Gaviria, también encargado de garantizar el cumplimiento de la metodología SCRUM y de remover cualquier impedimento que afectara el flujo de trabajo.

En cuanto a la gestión de los cambios y la incorporación de feedback, se llevaron a cabo las ceremonias de SCRUM en cada sprint. Los seguimientos se realizaban todos los martes, los refinamientos el primer viernes, y los reviews y retrospectivas el segundo viernes, al cierre de cada sprint. Durante estas ceremonias se revisaban los avances, se incorporaba feedback y se ajustaban las historias de usuario y los objetivos para el siguiente sprint, asegurando así la mejora continua del proceso de desarrollo y la adaptación a posibles imprevistos.

El objetivo del primer sprint fue preparar el entorno de desarrollo necesario para el proyecto. Esto incluyó la configuración de herramientas y plataformas esenciales para el control de versiones, el desarrollo de firmware y la gestión de código. Durante este sprint, se establecieron las bases para un desarrollo eficiente y organizado, creando un ambiente de trabajo que los futuros desarrolladores puedan continuar utilizando como referencia. Las Historias de Usuario abordadas en este sprint se detallan a continuación en la Tabla 3.1.

Tabla 3.1
Sprint 01 - Febrero 2 a 15 - Entorno de desarrollo

ID	Historia de usuario	Objetivo	Criterios de aceptación
01	Configurar el control de versiones con GitHub	Asegurar el seguimiento y la gestión de cambios en el código	1. Repositorio GitHub creado y accesible. 2. Confirmar que se pueden hacer commits y pull requests.
02	Utilizar VSCode para el desarrollo de firmware	Facilitar el desarrollo del firmware	1. VSCode instalado y configurado. 2. Plugins necesarios instalados.
03	Implementar reglas	Mantener estilo de	1. Reglas de formato configuradas en

	automáticas de formato de código	código consistente	VSCode. 2. El Código se auto-formatea según reglas al guardar.
04	Cambiar la partición de almacenamiento del ESP32 a 16MB	Utilizar toda la capacidad del dispositivo	1. Partición de almacenamiento cambiada y verificada. 2. Los dispositivos funcionan correctamente con la nueva partición.
05	Implementar FlashTools para formateo y carga inicial de firmware	Facilitar la configuración inicial de dispositivos	1. FlashTools implementado y probado. 2. Firmware cargado correctamente en múltiples dispositivos.

El segundo sprint se centró en el diseño y desarrollo del firmware base para los dispositivos IoT. Este sprint involucró la creación de módulos fundamentales que permitirían la conectividad, la gestión de archivos y la actualización del firmware. Además, se trabajó en la verificación de registro de dispositivos y la configuración de redes. Estas tareas fueron cruciales para asegurar que los dispositivos funcionen correctamente y se puedan gestionar de manera efectiva. Las Historias de Usuario correspondientes se encuentran en la Tabla 3.2.

Tabla 3.2
Sprint 02 - Febrero 16 a 29 - Firmware Base

ID	Historia de usuario	Objetivo	Criterios de aceptación
06	Diseñar el firmware base incluyendo diagramas de flujo y modularización	Facilitar la comprensión y el desarrollo del firmware	1. Diagramas de flujo creados. 2. Firmware modularizado correctamente.
07	Implementar el módulo de sistema de archivos	Permitir lectura y escritura de archivos en memoria interna	1. Sistema de archivos implementado. 2. Los archivos pueden ser leídos y escritos correctamente.
08	Configurar el módulo de red mediante servidor de Punto de Acceso	Permitir configuración de red WiFi y control a través de Websockets	1. Módulo de red configurado y funcionando. 2. Servidor Websocket operativo.
09	Implementar el módulo de verificación de registro de dispositivos	Validar dispositivos registrados en la base de datos	1. Módulo de verificación implementado. 2. Dispositivos no autorizados son rechazados.
10	Implementar el módulo de actualización de firmware	Permitir actualizaciones remotas de firmware a dispositivos autorizados	1. Módulo de actualización funcionando. 2. El Firmware puede ser actualizado remotamente a los dispositivos autorizados.
11	Configurar el módulo de conexión a Broker MQTT	Permitir la comunicación con el servidor central	1. Conexión a Broker MQTT establecida. 2. Los datos pueden ser enviados y recibidos correctamente.

12	Implementar el módulo de indicación LED	Proveer señales visuales del estado del dispositivo	1. Módulo de LED implementado. 2. El LED indica correctamente el estado del dispositivo.

Durante el tercer sprint, el objetivo principal fue implementar y probar nuevos módulos desarrollados para el firmware base, así como resolver cualquier problema pendiente. Este sprint incluyó la implementación de funcionalidades avanzadas como la doble conexión MQTT para enviar datos tanto al broker de Tablesa como al cliente del dispositivo, con diferentes periodos de muestreo. Además, se abordó la creación del firmware completo que se descarga si el dispositivo está registrado correctamente, cumpliendo con el requerimiento de probar dispositivos sin compartir el firmware completo con los fabricantes. También se integraron nuevas características para la matriz de LEDs y se resolvieron problemas de seguridad relacionados con los certificados Root CA. Las Historias de Usuario para este sprint se listan en la Tabla 3.3.

Tabla 3.3
Sprint 03 - Marzo 1 a 14 - Firmware Completo

ID	Historia de usuario	Objetivo	Criterios de aceptación
13	Implementar los nuevos módulos desarrollados en el firmware base	Mejorar la funcionalidad y rendimiento del firmware	1. Nuevos módulos implementados. 2. Firmware actualizado y funcionando correctamente.
14	Implementar la doble conexión MQTT para cliente y propietario	Gestionar múltiples conexiones simultáneamente	1. Doble conexión MQTT establecida. 2. Los datos pueden ser enviados y recibidos por ambos clientes.
15	Solucionar el incidente de certificados Root CA de GitHub	Asegurar correcta autenticación y seguridad del firmware	1. Problema de certificados solucionado. 2. El Firmware puede autenticarse correctamente.
16	Implementar funcionalidades de la matriz de LEDs	Proveer más información a través de LEDs	1. Matriz de LEDs implementada. 2. Los LEDs muestran correctamente la potencia de señal WiFi, voltajes y corrientes de cada fase.

El cuarto sprint continuó el trabajo en el firmware completo, enfocándose en la optimización y mejora de la gestión de tareas utilizando FreeRTOS. Se implementaron técnicas avanzadas de comunicación de tareas, incluida la utilización de mutex y queues para la sincronización y manejo adecuado de recursos compartidos. También se optimizaron los recursos de StackSize asociados a las tareas para mejorar el rendimiento del sistema. [31] Este sprint fue crucial para garantizar que el firmware sea robusto y eficiente, asegurando que los dispositivos IoT puedan cumplir con los requerimientos

específicos de Tablesa y sus clientes. Las Historias de Usuario desarrolladas en este sprint se detallan en la Tabla 3.4.

Tabla 3.4
Sprint 04 - Marzo 15 a 28 - Firmware Completo

ID	Historia de usuario	Objetivo	Criterios de aceptación
17	Implementar FreeRTOS para gestión de tareas	Mejorar el rendimiento del dispositivo	1. FreeRTOS implementado. 2. Tareas gestionadas correctamente.
18	Implementar la comunicación de tareas utilizando mutex y queue	Garantizar sincronización y manejo adecuado de recursos compartidos	1. Comunicación de tareas implementada. 2. Recursos gestionados correctamente.
19	Optimizar los recursos de StackSize asociados a las tareas	Asegurar uso óptimo de recursos	1. Recursos StackSize optimizados. 2. Tareas funcionando correctamente.

El quinto sprint estuvo dedicado a la infraestructura del sistema, con un enfoque en la implementación de la base de datos y los sistemas de registro y verificación de dispositivos IoT. Se trabajó en la creación de un modelo de datos estructurado en DynamoDB y en la exposición de la base de datos a través de una API GraphQL. Estos componentes fueron fundamentales para la gestión y autenticación de los dispositivos en el sistema. Las Historias de Usuario asociadas a este sprint se presentan en la Tabla 3.5.

Tabla 3.5
Sprint 05 - Marzo 29 a Abril 11 - Infraestructura del sistema

ID	Historia de usuario	Objetivo	Criterios de aceptación
20	Implementar el modelo de datos estructurado en DynamoDB	Gestionar datos del sistema eficientemente	1. Modelo de datos implementado. 2. Datos gestionados correctamente en DynamoDB.
21	Exponer la base de datos con API GraphQL	Facilitar acceso y gestión de datos por otros servicios	1. API GraphQL implementada. 2. Datos accesibles a través de GraphQL.
22	Implementar el sistema de registro de dispositivos IoT	Gestionar el registro de nuevos dispositivos	1. Sistema de registro implementado. 2. Nuevos dispositivos pueden ser registrados correctamente.
23	Implementar el sistema de verificación de registro de dispositivos	Asegurar dispositivos correctamente registrados y autenticados	1. Sistema de verificación implementado. 2. Dispositivos no autorizados son rechazados.

En el sexto sprint, el trabajo en la infraestructura del sistema continuó con la implementación de almacenamiento de logs y datos adquiridos por los dispositivos IoT, así como el sistema de actualización de firmware. Estos elementos fueron esenciales para asegurar que el sistema pudiera almacenar, gestionar y actualizar la información de manera eficiente. Las Historias de Usuario para este sprint se encuentran en la Tabla 3.6.

Tabla 3.6
Sprint 06 - Abril 12 a 25 - Infraestructura del sistema

ID	Historia de usuario	Objetivo	Criterios de aceptación
24	Implementar el almacenamiento de logs en la base de datos	Mantener un registro detallado de operaciones y eventos	1. Almacenamiento de logs implementado. 2. Logs accesibles y detallados.
25	Implementar el almacenamiento de datos adquiridos por los dispositivos	Facilitar el análisis y la toma de decisiones basadas en datos	1. Almacenamiento de datos implementado. 2. Datos adquiridos almacenados correctamente.
26	Implementar el sistema de actualización de firmware	Permitir la actualización remota de dispositivos	1. Registro y almacenamiento de versiones de firmware en el sistema. 2. El Firmware puede ser actualizado remotamente.

El séptimo sprint se enfocó en el desarrollo de la interfaz web de administración para la visualización y gestión de dispositivos IoT. Se implementaron sistemas de autenticación de usuarios, integración del aplicativo web con la base de datos, y el desarrollo de componentes visuales para mejorar la usabilidad y experiencia del usuario. Este sprint fue clave para proporcionar a los administradores herramientas efectivas para identificar, configurar y monitorear los dispositivos. Las Historias de Usuario desarrolladas en este sprint se detallan en la Tabla 3.7.

Tabla 3.7
Sprint 07 - Abril 26 a Mayo 9 - Interfaz de administración

ID	Historia de usuario	Objetivo	Criterios de aceptación
27	Implementar el sistema de autenticación de usuarios	Asegurar que solo usuarios autorizados puedan acceder al sistema	1. Sistema de autenticación implementado. 2. Acceso restringido a usuarios autorizados.
28	Integrar la interfaz con la base de datos	Facilitar la gestión y visualización de datos almacenados	1. Interfaz integrada con la base de datos. 2. Datos accesibles y gestionables desde la interfaz.

29	Implementar componentes visuales en la interfaz	Mejorar la usabilidad y experiencia del usuario	1. Componentes visuales implementados. 2. La interfaz es más intuitiva y fácil de usar.
30	Implementar el listado de dispositivos	Visualizar y gestionar todos los dispositivos registrados en el sistema	1. Listado de dispositivos implementados. 2. Todos los dispositivos son visibles y gestionables.
31	Implementar filtros de búsqueda en la interfaz	Facilitar la localización de dispositivos y datos específicos	1. Filtros de búsqueda implementados. 2. Los dispositivos y datos se pueden localizar fácilmente.
32	Implementar el registro de dispositivos desde la interfaz	Facilitar la adición de nuevos dispositivos al sistema	1. Registro de dispositivos desde la interfaz implementada. 2. Nuevos dispositivos pueden ser registrados fácilmente.
33	Implementar la configuración de parámetros desde la interfaz	Permitir la modificación de configuraciones de dispositivos remotamente	1. Configuración de parámetros desde la interfaz implementada. 2. Los parámetros pueden ser modificados remotamente.

El último sprint se dedicó a la realización de pruebas exhaustivas para asegurar que todas las funcionalidades del sistema fueran operativas e identificar oportunidades de mejora. Se llevaron a cabo pruebas de carga de firmware, configuración WiFi, verificación de registro, almacenamiento de datos y visualización de estados de conexión, entre otras. Este sprint fue crucial para validar el sistema antes de su implementación final. Las Historias de Usuario correspondientes a las pruebas se listan en la Tabla 3.8.

Tabla 3.8

Sprint 08 - Mayo 10 a 23 - Pruebas

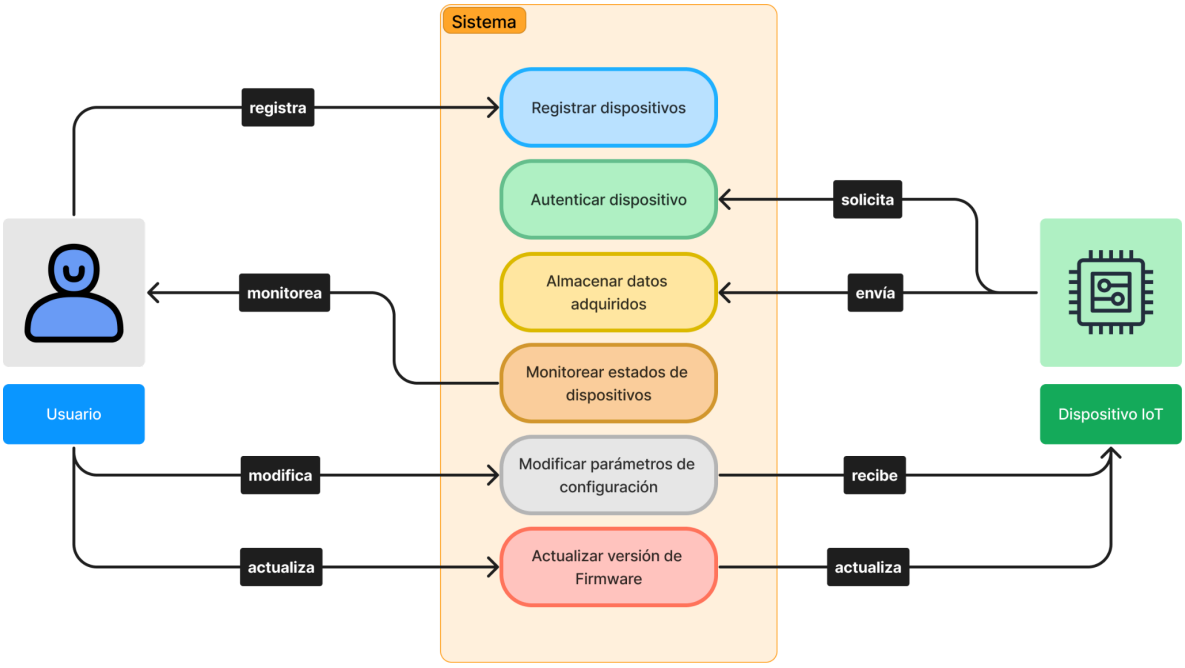
ID	Historia de usuario	Objetivo	Criterios de aceptación
34	Realizar pruebas de carga de firmware base en 100 dispositivos IoT	Asegurar que el firmware base se instala correctamente en todos los dispositivos	1. Pruebas de carga realizadas. 2. Firmware base instalado correctamente en 100 dispositivos.
35	Realizar pruebas de configuración WiFi y registro de dispositivos IoT	Verificar que los dispositivos pueden conectarse a la red WiFi y obtener información para el registro del dispositivo.	1. Pruebas de configuración y registro realizadas. 2. Dispositivos conectados y registrados correctamente.
36	Realizar pruebas de verificación de registro y	Asegurar que los dispositivos pueden ser	1. Pruebas de verificación y actualización realizadas. 2.

	actualización de firmware de dispositivos IoT	verificados y actualizados remotamente	Dispositivos verificados y actualizados correctamente.
37	Realizar pruebas de autenticación de dispositivos IoT	Verificar que solo los dispositivos autenticados pueden comunicarse con el sistema	1. Pruebas de autenticación realizadas. 2. Solo dispositivos autenticados pueden comunicarse.
38	Realizar pruebas de almacenamiento de datos enviados por dispositivos IoT	Asegurar que los datos enviados por los dispositivos se almacenan correctamente	1. Pruebas de almacenamiento realizadas. 2. Datos almacenados correctamente.
39	Realizar pruebas de visualización de estados de conexión de dispositivos IoT	Verificar que el estado de conexión de los dispositivos se refleja correctamente en la interfaz	1. Pruebas de visualización realizadas. 2. Estados de conexión reflejados correctamente.
40	Realizar pruebas de envío de configuración de parámetros a un dispositivo IoT desde la interfaz web	Verificar que los parámetros pueden ser configurados y aplicados remotamente	1. Pruebas de envío de configuración realizadas. 2. Parámetros configurados y aplicados correctamente.

3.2.4 Diagrama conceptual

El diagrama conceptual del sistema que se presenta en esta sección es el resultado de un proceso de diseño en el que se analizaron los requerimientos funcionales del proyecto y se delimitaron los alcances específicos mediante historias de usuario. Este diagrama ilustra los actores clave y sus interacciones con el sistema de operación y gestión de dispositivos IoT. Los usuarios interactúan con el sistema para registrar y monitorear dispositivos, modificar parámetros de configuración y actualizar el firmware de dispositivos. Los dispositivos IoT interactúan con el sistema para autenticarse, almacenar datos adquiridos, y recibir comandos de actualización o configuración. Cada actor y sus respectivas acciones están delineados en la Figura 3.1, proporcionando una visión general de la arquitectura y el flujo de información dentro del sistema.

Figura 3.1
Diagrama conceptual del sistema de operación y gestión para dispositivos IoT de Tablesa S.A.

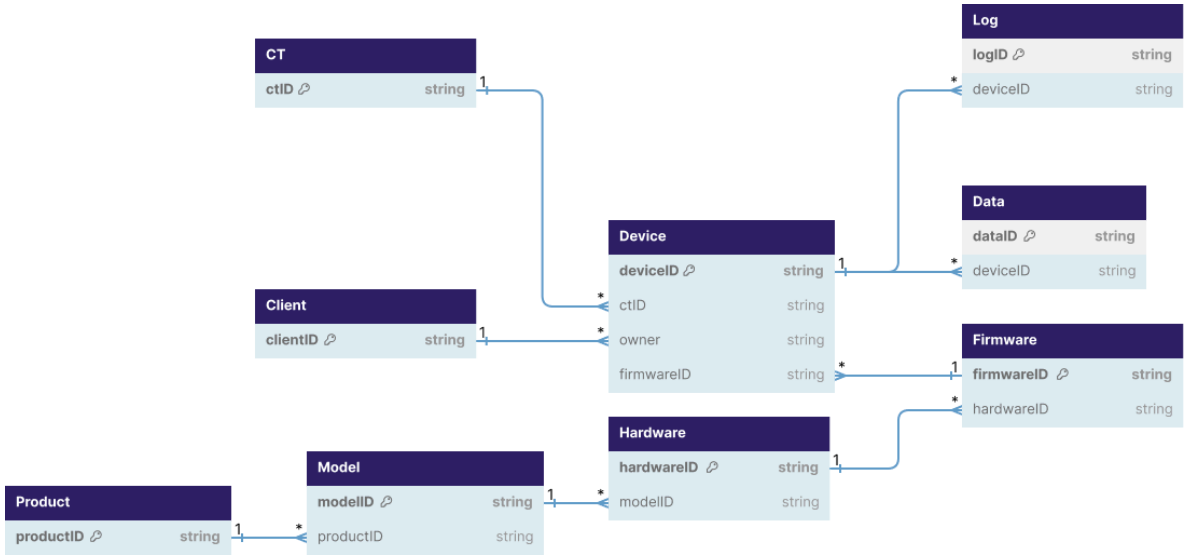


A partir del diagrama conceptual, se determinaron los principales componentes y funcionalidades del sistema, identificando aquellas que son esenciales para cumplir con los requerimientos del proyecto. Este análisis permitió priorizar las áreas clave, como la autenticación de dispositivos, el almacenamiento y monitoreo de datos, la actualización de firmware y la configuración remota de parámetros. Gracias a estas definiciones, podemos encaminar el proyecto hacia un resultado que abarque en mayor medida las principales necesidades de Tablesa S.A., permitiendo una mejor orientación para el desarrollo del sistema.

3.2.5 Modelo de datos estructurado

Con el objetivo de asegurar que el sistema sea escalable a largo plazo y capaz de soportar el registro de un creciente número de clientes, dispositivos y datos, se diseñó un modelo estructurado que contemple estas necesidades. Este modelo permite la incorporación de diferentes productos de dispositivos que Tablesa pueda fabricar en el futuro, así como la gestión de múltiples versiones de hardware y firmware de un mismo producto. De esta manera, se busca garantizar la flexibilidad y adaptabilidad del sistema, permitiendo su evolución conforme a las necesidades y productos futuros de Tablesa. La Figura 3.2 ilustra la estructura del modelo propuesto.

Figura 3.2
Modelo de datos estructurado del sistema



La tabla **Product** define los productos que ofrece Tablesa, e incluye toda la información descriptiva del producto. Pueden registrarse productos de diferentes tipos, como medidores, actuadores, gateways, etc. Cada producto puede tener muchos modelos diferentes. Esta decisión se tomó gracias a que actualmente se maneja un mismo producto con variantes en sus características de comunicación o adquisición.

La tabla **Model** define los diferentes modelos de un producto, permitiendo la gestión de múltiples modelos para un mismo producto. A su vez, cada modelo puede tener muchas versiones de hardware diferentes. Esta decisión se tomó gracias a que actualmente se encuentran en campo versiones de hardware que se han mejorado con el tiempo de un mismo modelo.

La tabla **Hardware** contiene datos sobre las versiones de hardware que se han fabricado, incluyendo el nombre y código de la versión, permitiendo la gestión de múltiples versiones de hardware para un mismo modelo. A su vez, cada versión de hardware puede tener muchas versiones de firmware diferentes. Esta decisión se tomó gracias a que actualmente se encuentran en campo dispositivos que comparten la misma versión de hardware pero con versiones de firmware diferentes que se han actualizado con el tiempo.

La tabla **Firmware** guarda información sobre las versiones de firmware disponibles, como la URL donde se almacena el archivo binario, el nombre de la versión y el código de la versión. Cada dispositivo en la tabla **Device** está

asociado a una versión específica de firmware a través del campo **firmwareID**. Esto permite la gestión de múltiples versiones de firmware para una misma versión de hardware. A su vez, cada versión de firmware puede tener muchas versiones de dispositivos.

La tabla **Device** es central y contiene información sobre cada dispositivo, incluyendo su identificación única, tipo de sistema, nombre y estado. Esta tabla se relaciona con la tabla **Client**, que almacena los datos de los clientes, como nombre, ciudad, correo electrónico, teléfono y dirección. La relación entre **Device** y **Client** permite asociar dispositivos a sus respectivos propietarios.

La tabla **Data** almacena los datos adquiridos por los dispositivos, incluyendo el tiempo de adquisición y los valores específicos medidos. Esta tabla está relacionada con **Device** para vincular cada conjunto de datos a su dispositivo correspondiente.

La tabla **Log** registra eventos y mensajes importantes generados por los dispositivos o el sistema, proporcionando un historial detallado de operaciones. Cada registro en **Log** está asociado a un dispositivo a través del campo **deviceID**.

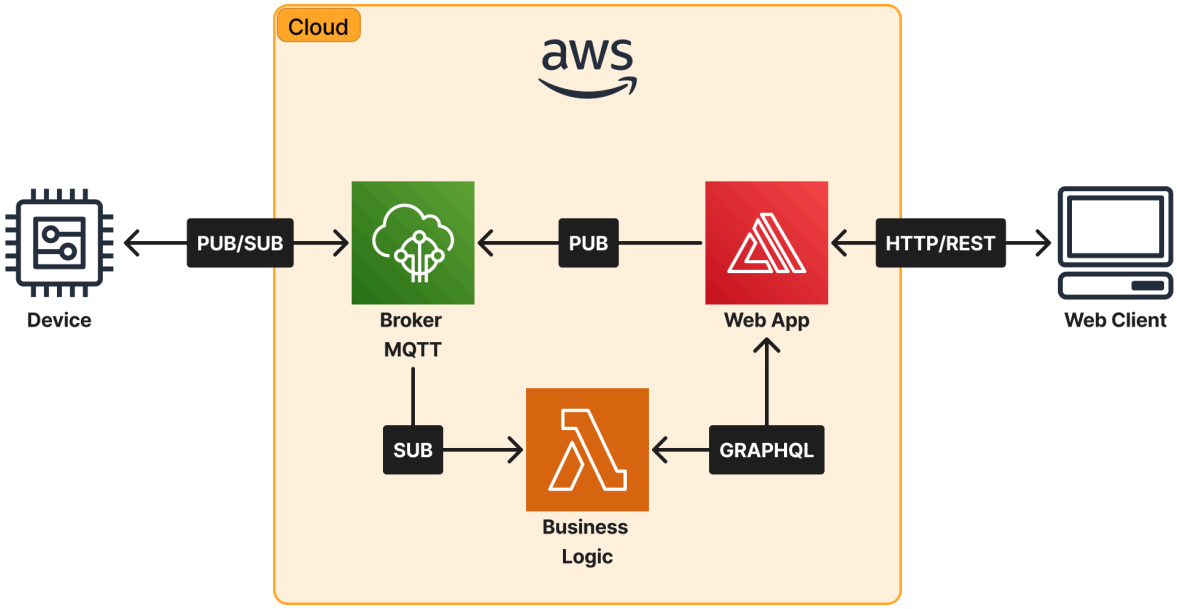
Por último, la tabla **CT** contiene información sobre los Transformadores de Corriente (CT) que maneja Tablesa. Cada CT puede ser conectado a diferentes dispositivos, y es fundamental especificar los factores de conversión de voltaje a corriente para determinar el valor de energía medido con precisión. Estos CTs son intercambiables entre dispositivos; pero es necesario mantener un registro de cuál CT está asociado a cada dispositivo para aplicar correctamente el factor de conversión correspondiente. La tabla **CT** incluye campos como el identificador del CT (**ctID**), la referencia y los factores de conversión (**iFactor**). Esta estructura permite gestionar la utilización de los CTs en los dispositivos de Tablesa.

3.3 IMPLEMENTACIÓN DEL SISTEMA DE OPERACIÓN Y GESTIÓN PARA LA FAMILIA DE DISPOSITIVOS IOT DE TABLESA S.A.

En esta sección se detalla el proceso de implementación del sistema de operación y gestión para dispositivos IoT de Tablesa S.A. La implementación abarca tres componentes principales: la infraestructura del sistema, la interfaz web de visualización de dispositivos IoT y el firmware desarrollado para los dispositivos. Cada subsección describe los pasos específicos y las tecnologías utilizadas para llevar a cabo la implementación de manera

efectiva y alineada con los objetivos del proyecto. La Figura 3.3 muestra una representación general de los servicios principales de la infraestructura.

Figura 3.3
Arquitectura general del sistema



La decisión de implementar la infraestructura del sistema en AWS se basó en los siguientes criterios. Primero, AWS ofrece una amplia gama de servicios y herramientas específicas para soluciones IoT, que están listas para su implementación. Esta oferta incluye una integración completa con otros servicios esenciales como bases de datos y lógica de negocio, facilitando un desarrollo cohesivo y eficiente. Además, AWS proporciona una documentación extensa que facilita el proceso de aprendizaje y solución de problemas. Finalmente, su modalidad de pago por uso permite desarrollar prototipos a un costo menor, mientras se aprovechan las características avanzadas de un servicio totalmente administrado. [29]

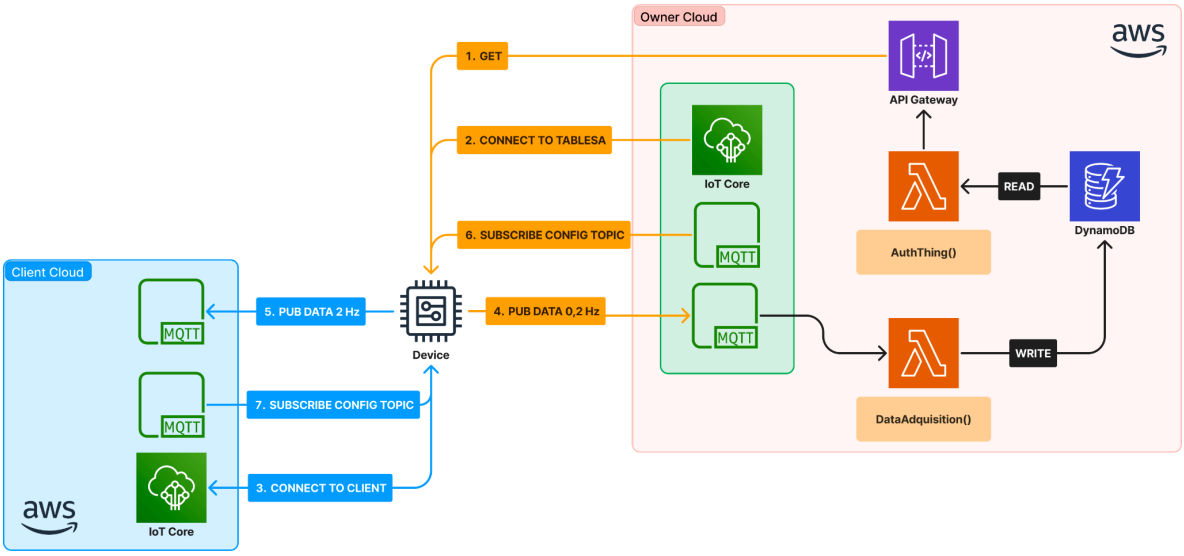
3.3.1 Infraestructura del sistema

La infraestructura del sistema es fundamental para garantizar el funcionamiento eficiente y escalable de los dispositivos IoT. Se compone de varias partes críticas que incluyen la base de datos del sistema, los sistemas de registro y verificación de dispositivos, el almacenamiento de datos adquiridos y el sistema de actualización de firmware. A continuación, se describen en detalle los componentes y las tareas realizadas para implementar cada una de estas partes utilizando tecnologías de AWS como

Amplify, DynamoDB, AppSync, Lambda, API Gateway, Cognito, S3 y AWS IoT Core. [32]

La Figura 3.4 muestra los componentes más importantes del sistema que interactúan con el dispositivo IoT. Podemos evidenciar que el dispositivo tiene la capacidad de conectarse a dos brokers MQTT; tanto al cliente como a Tablesa. Esto para cumplir con el segundo requerimiento funcional visto en la sección 3.2.1.

Figura 3.4
Infraestructura del sistema



Las siguientes subsecciones profundizarán en cada una de las partes de la infraestructura del sistema de Tablesa.

3.3.1.1 Base de datos del sistema

La base de datos del sistema se implementó utilizando DynamoDB, una base de datos NoSQL de AWS que ofrece alta escalabilidad y rendimiento. [33] Para facilitar la construcción del modelo estructurado de datos visto en la Figura 3.2, se utilizó AWS Amplify, aprovechando sus capacidades junto con el servicio AWS AppSync, que proporcionan una abstracción para interactuar con DynamoDB. Aunque DynamoDB es una base de datos NoSQL, se implementó un modelo de datos que permite organizar y gestionar los datos de manera estructurada. Es importante destacar que, aunque estas herramientas proporcionan una abstracción que puede asemejarse a un modelo relacional, DynamoDB sigue siendo una base de datos NoSQL y no implementa un modelo relacional en el sentido tradicional de SQL. En este

contexto, el término "modelo estructurado de datos" se refiere a la manera en que los datos están organizados y gestionados dentro de DynamoDB, aprovechando las capacidades de Amplify y AppSync. [34] Amplify CLI fue la principal herramienta que facilitó la creación de servicios y su integración desde el terminal.

- **Desarrollo del modelo estructurado de datos en DynamoDB:** Se desarrolló un modelo de datos en DynamoDB utilizando AWS Amplify y AppSync, permitiendo gestionar los datos de clientes, dispositivos, logs y datos adquiridos de manera estructurada. [33]
- **Exponer base de datos con API GraphQL:** Se desarrolló una API GraphQL con AppSync para interactuar con la base de datos, facilitando consultas y mutaciones de datos de forma flexible. [35]

3.3.1.2 Registro de dispositivos IoT

Para gestionar el registro de dispositivos IoT en el sistema de Tablesa, se implementó un sistema de registro por lotes que se basa en la construcción de un archivo en formato de Valores Separados por Comas (CSV) que contiene una lista de identificadores únicos o “Chip IDs” para almacenar en un bucket de Amazon S3. El sistema ejecuta una función Lambda para recorrer la lista de IDs y registrarlos en la tabla **Device**.

Para la construcción de la lista es necesaria la recopilación de Chip IDs de los dispositivos IoT que se desean registrar. Estos Chip IDs se derivan de un conjunto de fusibles eléctricos programables (EFUSEs) que se encuentran dentro del microcontrolador utilizado en la fabricación del dispositivo IoT. Para los dispositivos de Tablesa, se utiliza el microcontrolador ESP32 de la marca Espressif. [36]

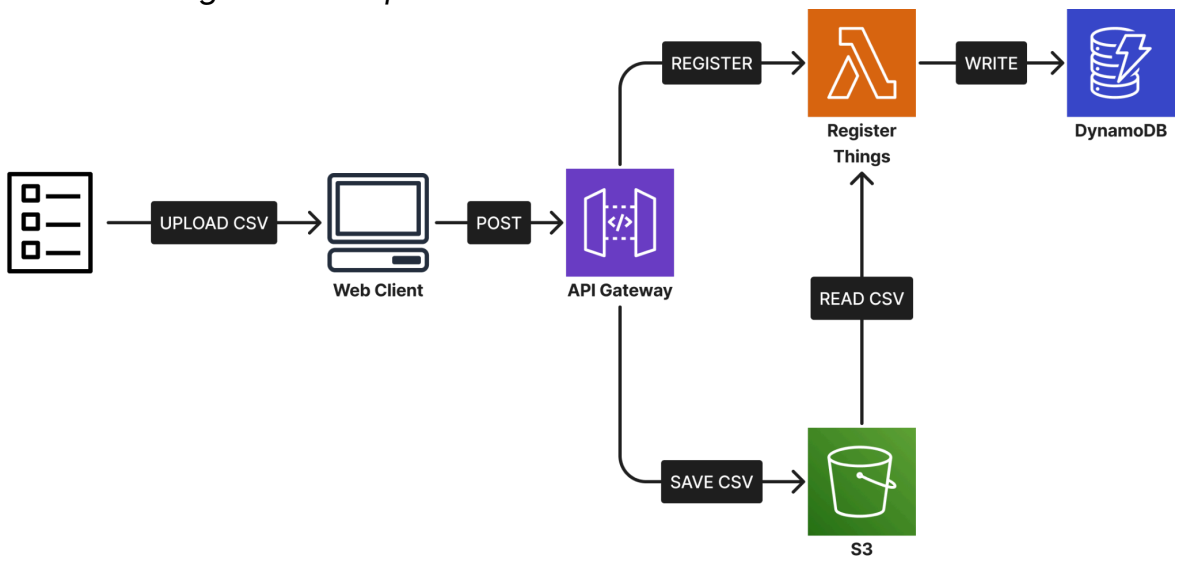
La Figura 3.5 muestra la estructura definida para la construcción del archivo CSV. El archivo contiene tres columnas de información donde cada fila contiene el Chip ID y MAC de cada dispositivo.

Figura 3.5
Estructura de archivo CSV para el registro de dispositivos

```
1   item;chipID;mac
2   1;000000000001;00:00:00:00:00:01
3   2;000000000002;00:00:00:00:00:02
4   3;000000000003;00:00:00:00:00:03
```

El proceso de registro de dispositivos se gestiona a través de una función Lambda denominada RegisterThings, que lee el archivo CSV desde S3 y registra los dispositivos en la tabla Device de la base de datos DynamoDB. Este flujo de trabajo, desde la carga del archivo CSV hasta el registro en DynamoDB, se ilustra en la Figura 3.6.

Figura 3.6
Sistema de registro de dispositivos IoT



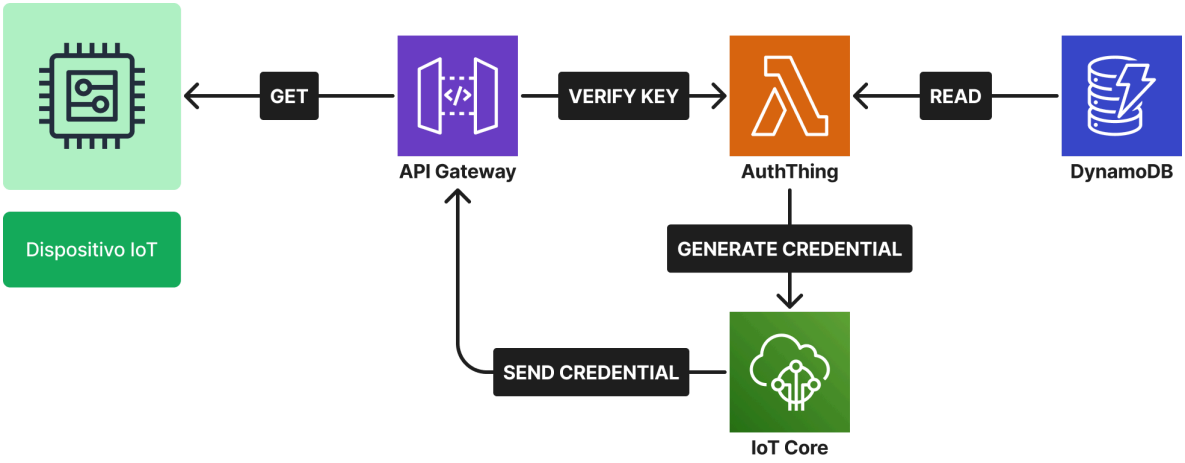
- **Almacenamiento estático de archivo CSV en bucket de S3:** Se creó un bucket de S3 para almacenar una lista de dispositivos en formato CSV, lo que facilita la carga inicial de datos.
- **Función RegisterThings:** Se desarrolló una función Lambda llamada RegisterThings que se encarga de leer el archivo CSV desde S3 y registrar los dispositivos en la base de datos.

- **Exponer función mediante Endpoint HTTP REST:** Para poder consumir esta funcionalidad desde la interfaz de usuario se expuso mediante un Endpoint de API Gateway.

3.3.1.3 Verificación de Registro de Dispositivos

Para asegurar que sólo los dispositivos registrados puedan interactuar con el sistema, se implementó el módulo de verificación de registro que se muestra en la Figura 3.7.

Figura 3.7
Módulo de verificación de registro de dispositivos

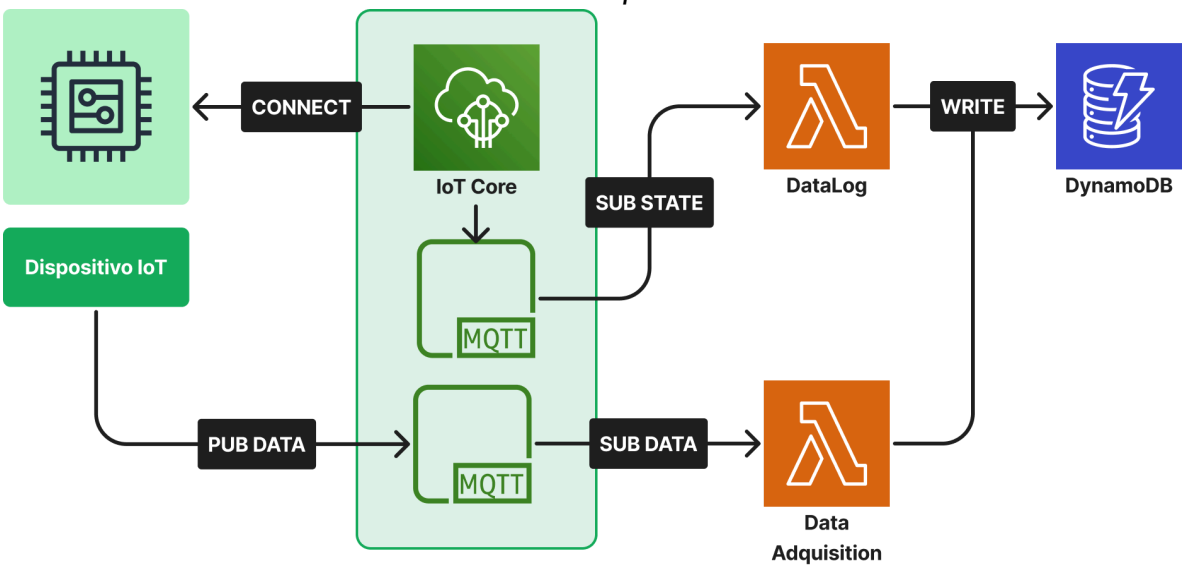


- **Endpoint de API Gateway**: El dispositivo IoT verifica su registro en el sistema mediante una petición GET HTTP REST implementada en el servicio AWS API Gateway la cual está protegida mediante un API KEY.
- **Función AuthThing**: Se desarrolló una función Lambda llamada AuthThing que verifica si un dispositivo está registrado en la tabla Device antes de permitir su acceso al sistema.
- **Respuesta del sistema:** Si el sistema verifica que el dispositivo está registrado, el sistema le otorgará las credenciales necesarias para conectarse al broker MQTT de AWS IoT Core. Estas credenciales se generan siempre y cuando el dispositivo verifique su registro por primera vez.
- **Almacenamiento de credenciales:** El dispositivo IoT almacena sus credenciales en su sistema de archivos. Además, puede solicitar la renovación de estas credenciales si su sistema de archivos experimenta alguna alteración.

3.3.1.4 Almacenamiento de datos adquiridos

El almacenamiento de datos adquiridos por los dispositivos IoT es fundamental para el análisis y monitoreo continuo del sistema. La Figura 3.8 muestra el módulo de almacenamiento de datos adquiridos, que abarca tanto los datos enviados por los dispositivos como los cambios de estado de conexión detectados por AWS IoT Core.

Figura 3.8
Módulo de almacenamiento de datos adquiridos



El proceso de almacenamiento de datos se realiza mediante dos funciones Lambda desencadenadas por reglas de AWS IoT. Estas funciones son responsables de manejar los datos adquiridos y los eventos de conexión/desconexión de los dispositivos.

A. Función DataLog:

- **Reglas de AWS IoT:** Se crearon dos reglas en AWS IoT Core para los tópicos "$aws/events/presence/connected/+" y "$aws/events/presence/disconnected/+". Estas reglas se activan cada vez que un dispositivo se conecta o desconecta del sistema.
- **Almacenamiento:** La función **DataLog** se ejecuta cuando se dispara alguna de estas reglas, creando un registro en la tabla **Log**. Además, esta función actualiza el campo state en la tabla **Device**, reflejando el estado actual de conexión del dispositivo.

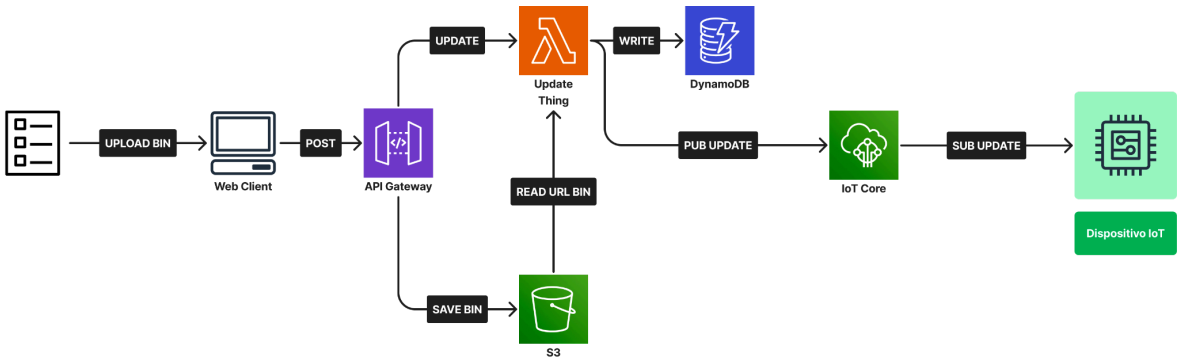
B. Función DataAdquisition:

- **Regla de AWS IoT:** Se configuró una regla para el tópico específico donde los dispositivos publican sus datos adquiridos.
- **Almacenamiento:** Cuando esta regla se activa, la función **DataAdquisition** se encarga de crear un nuevo registro en la tabla **Data**. Cada registro incluye detalles como el identificador del dispositivo, la marca temporal y los valores específicos de los datos medidos.

3.3.1.5 Módulo de Actualización de Firmware

El sistema de actualización de firmware es esencial para garantizar que los dispositivos IoT de Tablesa S.A. operen siempre con la versión más reciente del software. Este sistema permite realizar actualizaciones de firmware Over-the-Air (OTA) de manera eficiente y segura. La Figura 3.9 muestra el flujo del proceso de actualización de firmware.

Figura 3.9
Módulo de Actualización de Firmware



El proceso de Actualización de Firmware sigue el siguiente flujo:

1. **Carga del Firmware Compilado:**
   a. El proceso comienza cuando un usuario carga un archivo compilado del firmware (.bin) a través de la interfaz web de administración.
   b. Este archivo es enviado mediante API Gateway para su almacenamiento en un bucket de Amazon S3.
2. **Almacenamiento en S3:**
   a. Una vez almacenado en S3, se obtiene la URL del archivo estático.

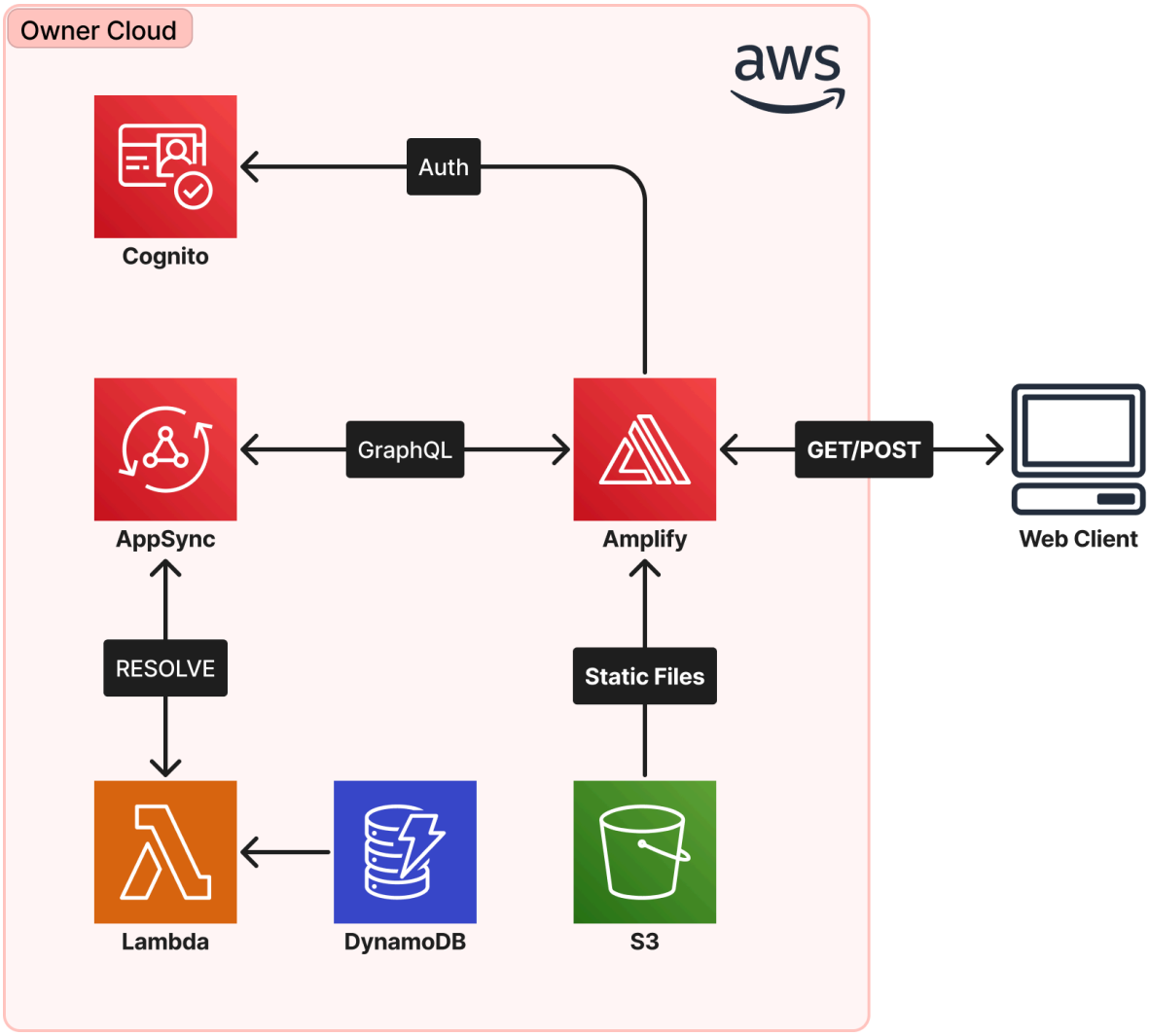
   b. Esta URL se utiliza para crear un registro en la tabla Firmware en DynamoDB, donde se guarda toda la información relevante sobre la nueva versión del firmware, incluyendo la URL de descarga, la versión, la fecha de lanzamiento y una descripción de los cambios incluidos.
3. **Registro de Nueva Versión de Firmware:**
   a. La creación del registro en la tabla Firmware incluye información detallada necesaria para el seguimiento y la gestión de las versiones de firmware.
   b. Este registro asegura que cada dispositivo IoT pueda acceder a la versión correcta del firmware cuando sea necesario.
4. **Publicación de la Orden de Actualización:**
   a. Una vez que se completa el registro de la nueva versión de firmware, se publica una orden de actualización a los dispositivos IoT mediante el sistema de mensajería de AWS IoT Core.
   b. Esta orden contiene la información necesaria para que los dispositivos descarguen e instalen la nueva versión del firmware.

3.3.2 Interfaz Web de Visualización de Dispositivos IoT

La interfaz web de visualización de dispositivos IoT es un componente esencial del sistema, diseñada para proporcionar a los usuarios una herramienta intuitiva y eficiente para gestionar y monitorizar sus dispositivos. Esta interfaz se desarrolló utilizando el servicio AWS Amplify, aprovechando su completo conjunto de herramientas para aplicaciones web de React. La Figura 3.10 muestra la infraestructura del aplicativo web, detallando los servicios utilizados para su implementación.

AWS Amplify ofrece una serie de beneficios para el desarrollo de aplicaciones web modernas, incluyendo documentación detallada, una interfaz de línea de comandos (CLI) poderosa, la librería aws-amplify para facilitar la integración con servicios de AWS, y componentes de interfaz de usuario como @aws-amplify/ui-react. Además, Amplify se integra perfectamente con otros servicios de AWS, como Cognito para la autenticación de usuarios, S3 para el almacenamiento de archivos, AppSync para la gestión de bases de datos y la conexión MQTT con AWS IoT Core mediante la librería @aws-amplify/pubsub.

Figura 3.10
Infraestructura del aplicativo web



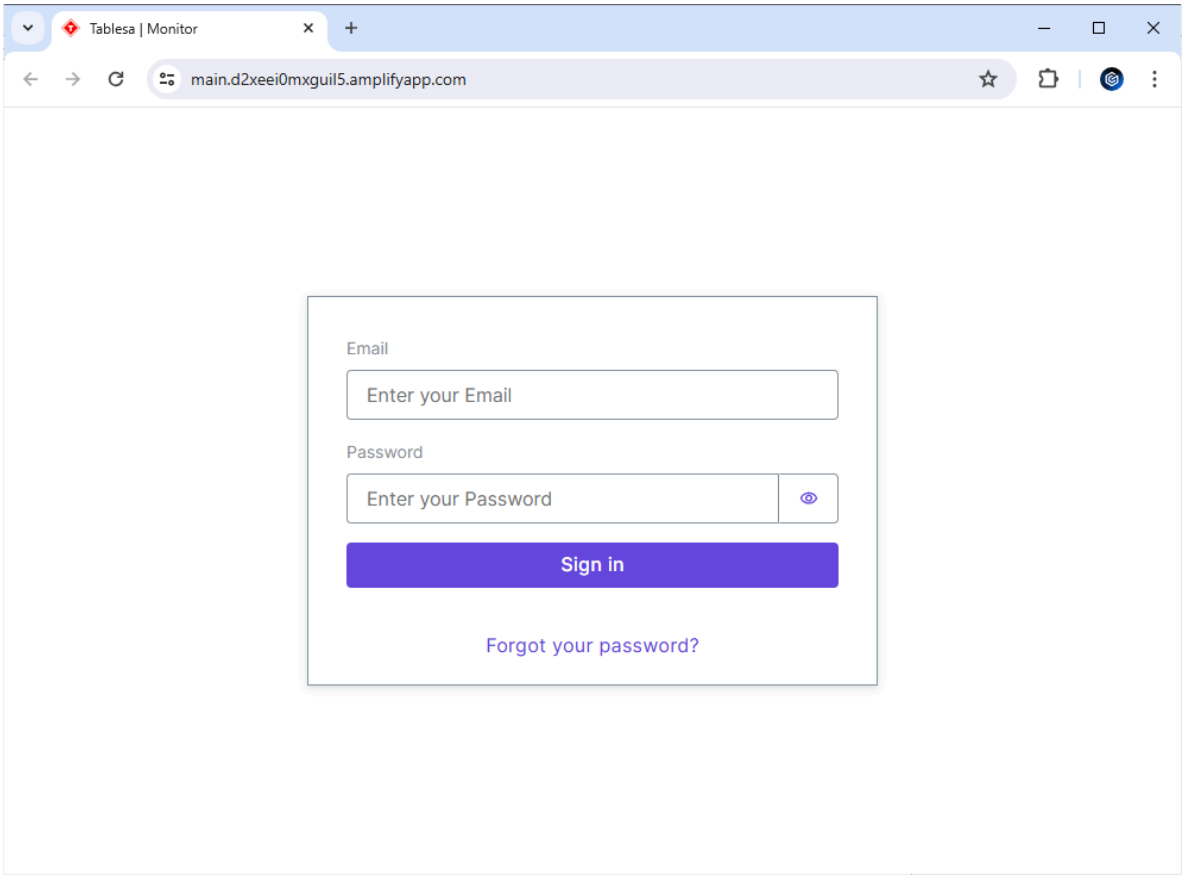
Las siguientes subsecciones, profundizarán en cada uno de los componentes de la aplicación web, describiendo en detalle cómo se implementan y cómo contribuyen a la funcionalidad general del sistema.

3.3.2.1 Sistema de autenticación de usuarios

El sistema de autenticación de usuarios se implementó utilizando Amazon Cognito, un servicio robusto y escalable que facilita la gestión de usuarios y la autenticación segura. Cognito permite definir permisos de autorización detallados tanto para el acceso a funcionalidades como a datos. En esta fase del proyecto, se configuraron permisos específicos para los administradores de Tablesa, asegurando que solo personal autorizado pueda realizar

acciones críticas dentro del sistema. Además, Cognito proporciona la flexibilidad necesaria para agregar más grupos de usuarios y ajustar los permisos según las necesidades que puedan surgir en el futuro. La Figura 3.11 muestra la pantalla de inicio de sesión del aplicativo web, donde se observa la interfaz presentada a los usuarios antes de autenticarse.

Figura 3.11
Pantalla de Inicio de sesión



3.3.2.2 Integración de base de datos

Para la integración de la base de datos en la aplicación web de visualización de dispositivos IoT, se utilizó la librería aws-amplify. Esta herramienta facilita la realización de consultas y la verificación de permisos de autorización tras la autenticación del usuario. Aws-amplify permite interactuar de manera eficiente con la base de datos, proporcionando un acceso seguro y controlado a los datos.

La integración se logró definiendo modelos de datos en AWS AppSync, los cuales se exponen a través de una API GraphQL. Esta API permite realizar

consultas y mutaciones sobre la base de datos, gestionando datos como la información de los dispositivos IoT, los registros de actividad y los datos adquiridos por los sensores. La seguridad y los permisos de acceso se gestionan mediante Amazon Cognito, asegurando que solo los usuarios autenticados y autorizados puedan acceder a funcionalidades específicas y a los datos correspondientes.

3.3.2.3 Componentes visuales

En la implementación de la interfaz web de visualización de dispositivos IoT para Tablesa S.A., se consideraron diversos componentes visuales diseñados para proporcionar una experiencia de usuario intuitiva. Algunos de estos componentes fueron seleccionados de la librería Amplify UI. Esta librería proporciona una serie de componentes preconstruidos y estilizados, como botones, selectores y barras de búsqueda, que facilitaron el desarrollo rápido y consistente de la interfaz. Otros componentes y pantallas fueron desarrollados para facilitar la interacción con el sistema, permitiendo a los usuarios registrar nuevos dispositivos, monitorear su estado y gestionar su configuración de manera efectiva. A continuación, se describen los principales componentes visuales desarrollados:

Una de las áreas centrales de la interfaz es el componente de información del dispositivo, ilustrado en la Figura 3.12. Este componente muestra detalles críticos de cada dispositivo, tales como su estado, nombre del producto, modelo, versión de hardware, versión de firmware, etiqueta, Chip ID, MAC, y el tiempo desde el último dato recibido. La presentación clara y organizada de esta información permite a los usuarios supervisar y gestionar múltiples dispositivos teniendo a disposición los datos más relevantes para una rápida identificación.

Figura 3.12
Componente de información del dispositivo

- **Estado:** Indica si el dispositivo está conectado, desconectado, o en algún estado de alerta, utilizando los indicadores visuales mostrados en la Figura 3.13.
- **Producto:** Identifica el producto específico, facilitando la gestión de diferentes tipos de dispositivos.

- **Modelo:** Proporciona el modelo del dispositivo, ayudando a diferenciar entre distintas variantes del mismo producto.
- **Versión de Hardware:** Muestra la versión de hardware del dispositivo, crucial para el mantenimiento y las actualizaciones.
- **Versión de Firmware:** Proporciona información sobre la versión del firmware instalada en el dispositivo.
- **Etiqueta:** Una etiqueta personalizada que puede ser asignada para identificar fácilmente el dispositivo en ubicaciones específicas o con funciones particulares.
- **Chip ID:** El identificador único del chip del dispositivo, esencial para su autenticación y gestión.
- **MAC:** La dirección MAC del dispositivo, utilizada para la comunicación en la red.
- **Último Dato Hace:** Indica el tiempo transcurrido desde que el dispositivo envió su último dato, ayudando a identificar posibles problemas de comunicación.

El componente anterior incluye un indicador que muestra los diferentes estados de conexión de los dispositivos, como se observa en la Figura 3.13. Estos estados se representan mediante iconos y colores específicos que permiten una rápida identificación del estado actual de cada dispositivo.

Figura 3.13
Estados de conexión del dispositivo

- **Error**: Indicador rojo que indica un dispositivo que lleva desconectado más de 5 minutos. Indica que un dispositivo no ha logrado restablecer su conexión a internet o presenta problemas de alimentación. Requiere revisión en campo.

- **Alerta:** Indicador naranja que indica un dispositivo que lleva desconectado entre 1 y 5 minutos. Indica que un dispositivo ha perdido su conexión WiFi. Aún puede restablecer la conexión automáticamente.
- **Advertencia:** Indicador amarillo que indica un dispositivo que lleva desconectado menos de un minuto. Indica que un dispositivo ha restablecido la conexión al broker MQTT. Puede restablecer la conexión automáticamente.
- **Conectado:** Indicador verde que señala que el dispositivo está operando correctamente y comunicándose con el sistema.
- **Registrado:** Indicador gris que muestra que el dispositivo ha sido registrado pero no está activo.

3.3.2.4 Filtros de búsqueda

La implementación de los filtros de búsqueda en la interfaz web del sistema de operación y gestión de dispositivos IoT de Tablesa S.A., permite a los usuarios realizar consultas precisas y eficientes. Estos filtros se obtienen del usuario y se utilizan para consultar los dispositivos mediante la API GraphQL, facilitando la gestión y supervisión de múltiples dispositivos de manera organizada.

La barra de búsqueda, ilustrada en la Figura 3.14, es una herramienta fundamental para localizar dispositivos específicos de manera rápida. Este componente cuenta con un selector que permite al usuario elegir entre los atributos Chip ID, MAC y etiqueta. Al escribir una palabra clave en la barra de búsqueda y presionar el botón "buscar", el sistema consultará los dispositivos cuyo atributo seleccionado coincida con la palabra clave ingresada. Esta funcionalidad proporciona una manera rápida y directa de encontrar dispositivos individuales.

Figura 3.14
Componente de barra de búsqueda

- **Selector de Atributos:** Permite al usuario elegir entre Chip ID, MAC y etiqueta como criterio de búsqueda.
- **Campo de Búsqueda:** Donde el usuario ingresa la palabra clave a buscar.
- **Botón de Búsqueda:** Al presionarlo, se realiza la consulta a la API GraphQL utilizando el atributo y la palabra clave seleccionados.

Una vez realizada la búsqueda, los resultados se muestran en una lista organizada, como se ilustra en la Figura 3.15. Esta lista permite al usuario ver rápidamente los dispositivos que coinciden con los criterios de búsqueda ingresados, facilitando la identificación y gestión de los dispositivos.

Figura 3.15
Resultado de la barra de búsqueda

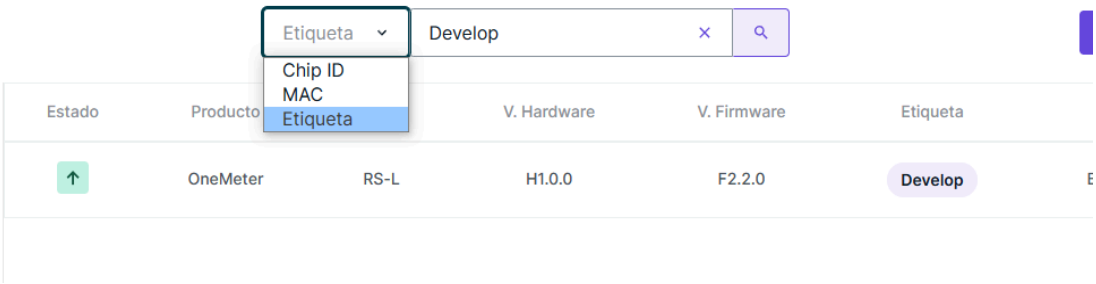
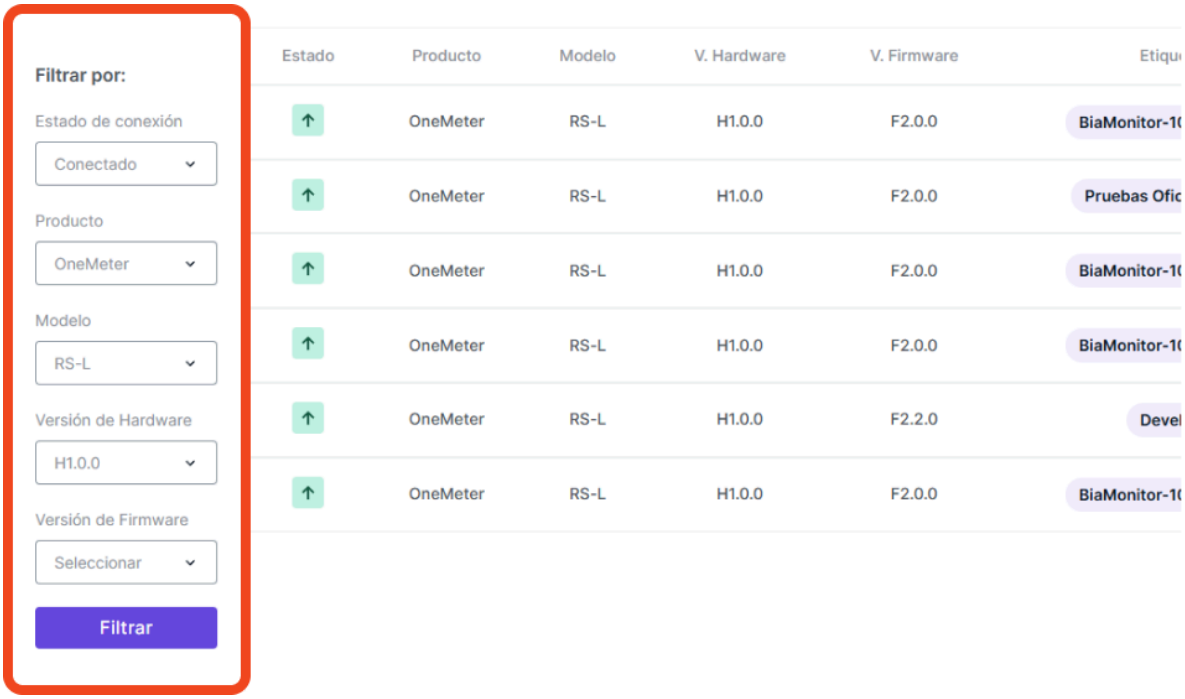


Figura 3.16
Componente de Filtro de búsqueda

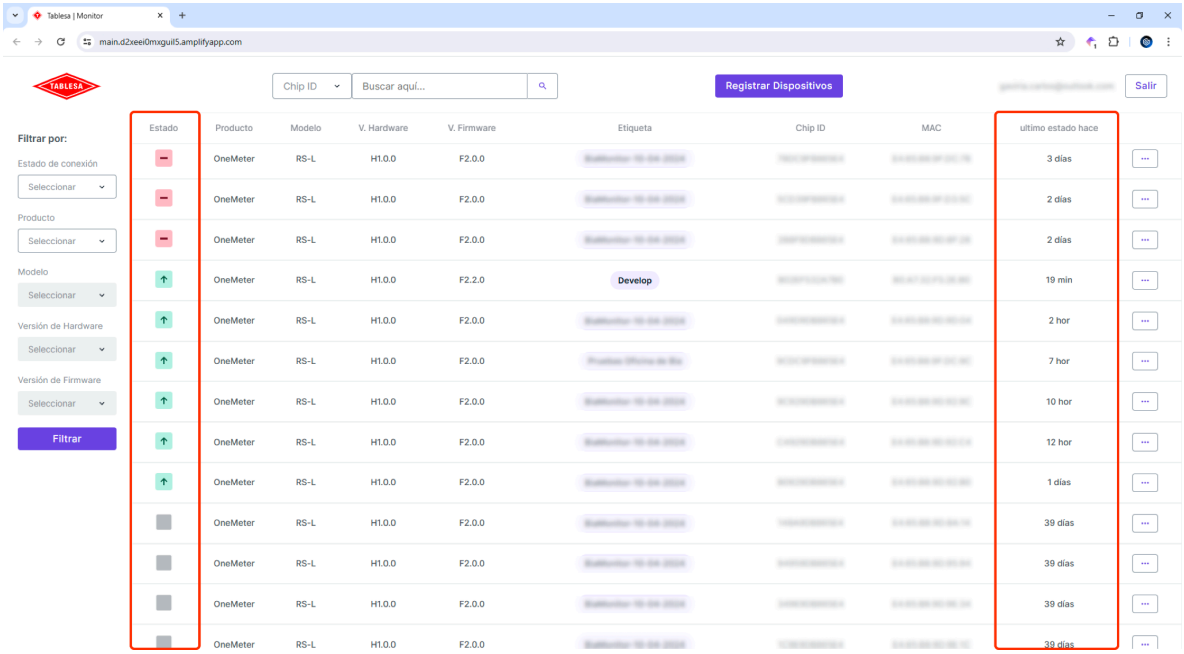


El componente de filtro de búsqueda, mostrado en la Figura 3.16, permite a los usuarios crear consultas más complejas para encontrar dispositivos que comparten uno o varios atributos seleccionados. Este componente es ideal

para situaciones en las que se necesita gestionar o supervisar grupos de dispositivos con características comunes.

3.3.2.5 Listado de dispositivos

Figura 3.17
Pantalla principal - Listado de dispositivos



La página principal o Home de la aplicación web es una herramienta crucial para la supervisión y gestión de dispositivos IoT. Su principal funcionalidad es presentar una lista de los dispositivos ordenados según su estado de conexión y el tiempo desde el último dato recibido, proporcionando una visión clara y priorizada de la salud de la red de dispositivos. A continuación, se describe el funcionamiento principal de esta sección, ilustrada en la Figura 3.17.

El Home de la aplicación web está diseñada para mostrar los primeros 15 dispositivos, ordenados por su estado de conexión desde los más críticos hasta los menos críticos, siguiendo la clasificación presentada anteriormente en la Figura 3.13. Este ordenamiento permite a los usuarios identificar rápidamente los dispositivos que requieren atención inmediata.

1. **Orden por Estado de Conexión:** Los dispositivos se presentan ordenados del estado más crítico al menos crítico.

2. **Orden por "Último Estado Hace":** Dentro de cada estado de conexión, los dispositivos se clasifican según el tiempo transcurrido desde su última actualización de estado:
    - Errores: Los dispositivos que llevan más tiempo sin actualizar su estado de conexión, se muestran primero, permitiendo a los usuarios priorizar la resolución de problemas prolongados.
    - Conectado: Los dispositivos que han enviado actualizaciones de conexión más recientes se muestran primero, lo que facilita la identificación de la estabilidad de la conexión de los dispositivos activos.

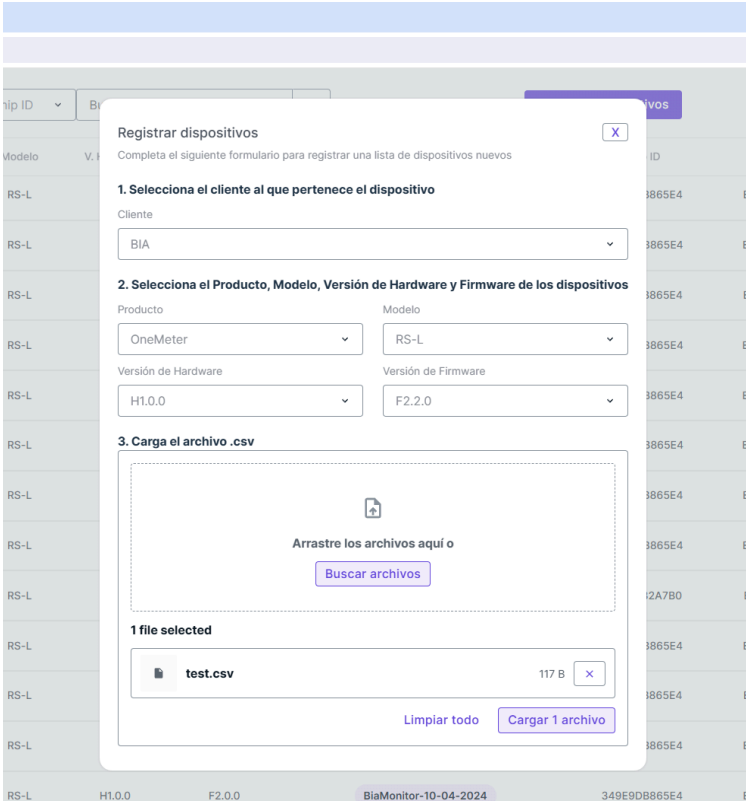
Esta organización intuitiva y basada en la criticidad permite a los administradores de sistemas centrarse en resolver los problemas más urgentes y mantener la red de dispositivos IoT operando de manera óptima.

3.3.2.6 Registro de dispositivos

El proceso de registro de dispositivos IoT se realiza mediante la carga de un archivo CSV que contiene los identificadores únicos de los dispositivos desde la pantalla de registro del aplicativo web. Este procedimiento está diseñado para ser usado, especialmente cuando se trata de registrar lotes de dispositivos con características similares.

El proceso de registro de dispositivos IoT consta de varios pasos secuenciales que aseguran la correcta incorporación de los dispositivos en el sistema. La Figura 3.18 muestra la pantalla de registro de dispositivos, donde se describen detalladamente cada una de las acciones requeridas:

Figura 3.18
Pantalla de registro de dispositivos



El primer paso es seleccionar el cliente al que pertenecen los dispositivos que se van a registrar. Esta selección es crucial para asociar correctamente los dispositivos con su respectivo propietario.

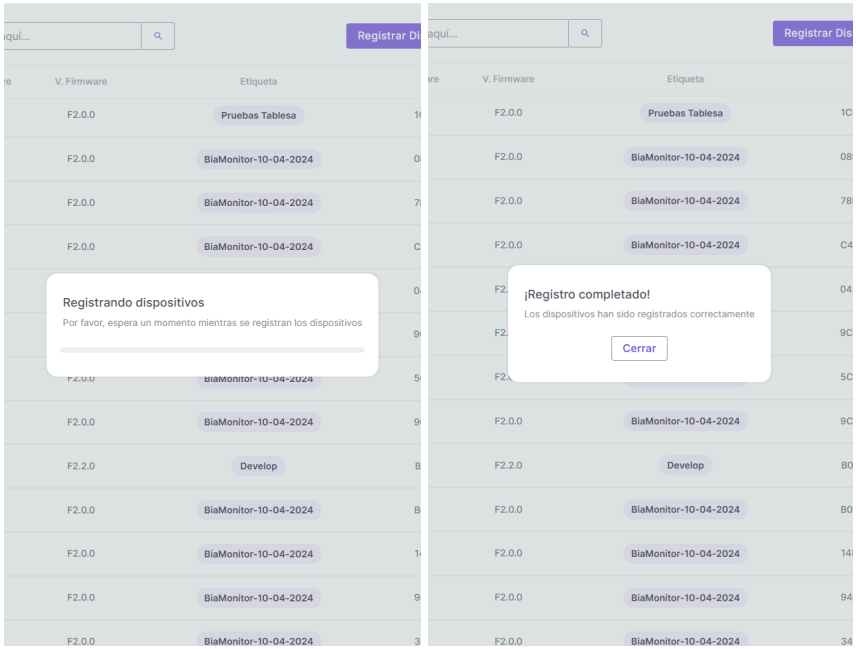
El segundo paso es la especificación de características del dispositivo. Este enfoque de registrar dispositivos con características similares en lote es útil ya que, comúnmente, se solicita la fabricación de un lote con una versión de hardware específica.

El paso No. 3 es seleccionar el archivo CSV desde el explorador de archivos del sistema. También es posible arrastrar el archivo directamente a la interfaz. Este archivo debe cumplir con la estructura definida previamente en la Figura 3.5.

Finalmente, se debe hacer clic en el botón "Cargar archivo" para iniciar el proceso. Este botón inicia la carga del archivo en Amazon S3 y, posteriormente, la ejecución de la funcionalidad de registro de dispositivos, como se muestra anteriormente en la Figura 3.6.

La Figura 3.19 ilustra los diferentes estados de la solicitud de registro de dispositivos, proporcionando al usuario información clara sobre el progreso y el resultado del proceso.

Figura 3.19
Estados de solicitud de registro de dispositivos

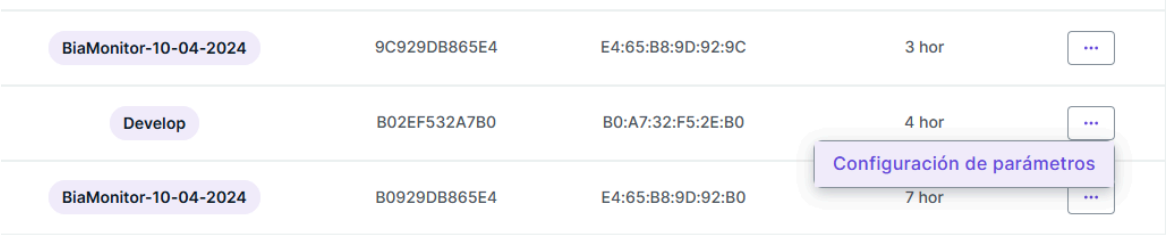


3.3.2.7 Configuración de parámetros

La configuración de parámetros se realiza mediante comunicación MQTT directamente al dispositivo seleccionado desde la interfaz de configuración del aplicativo web. Es fundamental que el dispositivo esté en estado conectado para que la configuración sea exitosa. A continuación, se describe el proceso detallado, ilustrado en las Figuras 3.20 y 3.21.

Para iniciar la configuración de parámetros, el usuario debe seleccionar el dispositivo deseado desde el menú desplegable de ajustes, como se muestra en la Figura 3.20. Este menú debe tener la Configuración de parámetros activa para los dispositivos conectados.

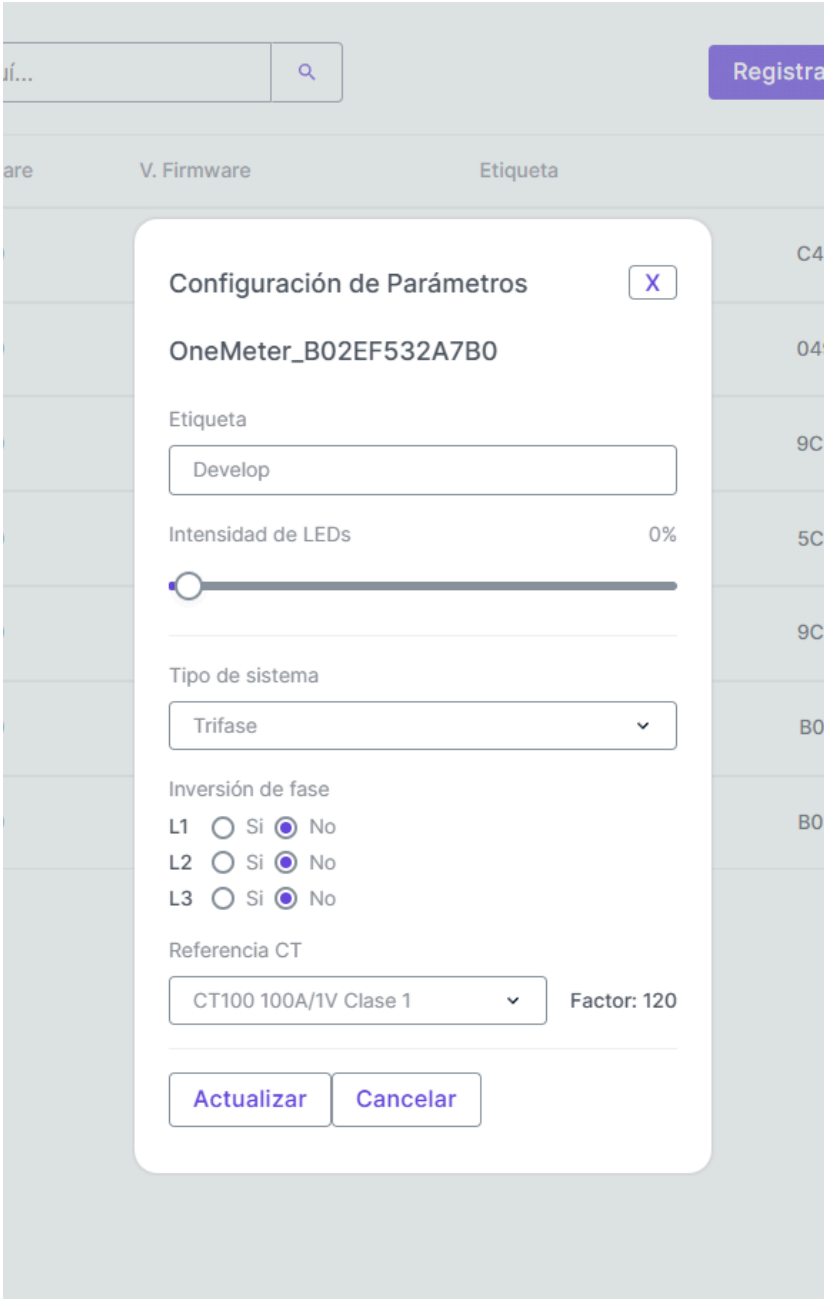
Figura 3.20
Menú desplegable de ajustes de un dispositivo



Una vez seleccionado el dispositivo, la interfaz web inicia la comunicación con el dispositivo sincronizando los valores de la interfaz con los parámetros actuales del dispositivo. Esta sincronización garantiza que el usuario trabaje con la información más reciente y precisa del dispositivo. La pantalla de configuración de parámetros, ilustrada en la Figura 3.21, permite al usuario editar los parámetros del dispositivo. Los parámetros disponibles para edición pueden variar según el tipo y modelo del dispositivo. El ejemplo mostrado pertenece al dispositivo de tipo medidor llamado “OneMeter RS-L” que incluye configuraciones como:

- **Etiqueta:** Permite asignar un nombre o identificación al dispositivo, facilitando su reconocimiento y gestión en la interfaz.
- **intensidad de LEDs:** Ajusta el brillo de los indicadores LED en el dispositivo, lo cual puede ser útil en diferentes entornos de iluminación.
- **Tipo de sistema:** Configura el tipo de sistema eléctrico al que está conectado el dispositivo (“Trifase”, “Bifase” o “Monofase”), asegurando una correcta medición y operación.
- **Inversión de fase:** Permite invertir la fase de cada línea del sistema, ajustando la configuración del dispositivo según las necesidades específicas del sistema eléctrico.
- **Referencia del Transformador de corriente (CT):** Configura la referencia del transformador de corriente utilizado, garantizando mediciones precisas de corriente y energía.

Figura 3.21
Pantalla de configuración de parámetros



Una vez el usuario haya terminado de editar los parámetros, debe hacer clic en el botón "Actualizar". Al hacerlo, las nuevas configuraciones se envían al dispositivo publicándose en su tópico de configuración al cual el dispositivo está suscrito.

Cuando el dispositivo recibe y aplica las nuevas configuraciones, publica un mensaje de respuesta en un tópico específico. La aplicación web se suscribe a este tópico al iniciar la configuración para confirmar que el dispositivo ha actualizado los parámetros correctamente. Una vez confirmada la actualización, el sistema genera un registro del procedimiento en la tabla **Log**.

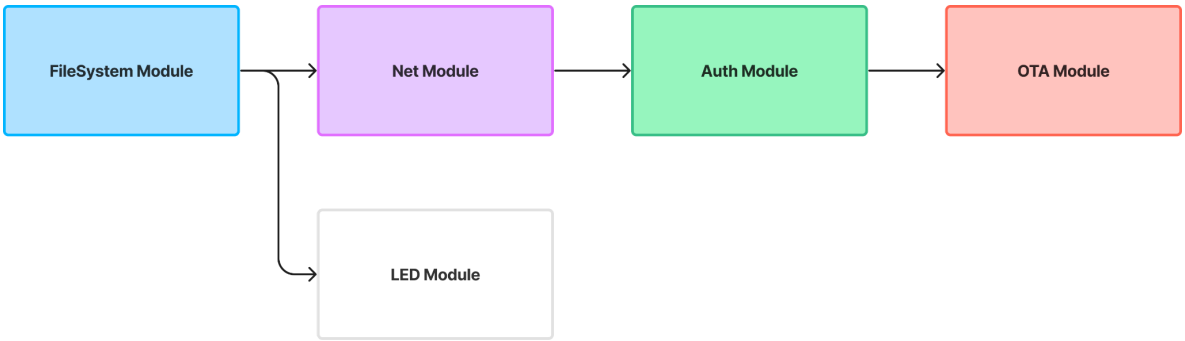
3.3.3 Firmware

El desarrollo del firmware para el dispositivo se dividió en dos etapas: un firmware base y un firmware completo. Esta estrategia fue adoptada debido a la necesidad de cargar un firmware preliminar durante el proceso de fabricación para probar el funcionamiento básico del dispositivo sin exponer las características completas del producto. El firmware base contiene únicamente los módulos necesarios para conectar el dispositivo a internet, verificar su registro y actualizar al firmware completo una vez que se haya confirmado su registro en el sistema. Esta metodología no solo facilita el proceso de manufactura y prueba, sino que también protege la integridad del firmware completo, evitando su divulgación no autorizada y garantizando la seguridad de las funciones principales del producto.

3.3.3.1 Implementación de firmware base

La implementación del firmware base se centra en proporcionar las funcionalidades esenciales para la inicialización y conectividad del dispositivo. Esta sección detalla los componentes y procesos involucrados en la creación de este firmware inicial, incluyendo la configuración de la conexión a internet, la verificación de registro del dispositivo y el mecanismo de actualización automática al firmware completo. Uno de los principales retos de desarrollo fue que el firmware anterior estaba construido en Arduino y carecía de una estructura clara y una separación que facilitará su comprensión. Para superar este desafío y hacer el código del firmware base más reutilizable en otros proyectos, como el firmware completo, se optó por modularizar el proyecto. La Figura 3.22 muestra los módulos que necesita el firmware base para funcionar correctamente. Junto con esta decisión también vino el cambio del entorno de desarrollo a Visual Studio Code con el plugin de Arduino, ya que este IDE permite visualizar la separación modular del código, además de ofrecer muchas otras ventajas como herramientas de desarrollo avanzadas.

Figura 3.22
Modularización de firmware base y sus dependencias



A continuación, se describen en detalle cada uno de estos módulos:

1. **FileSystem Module:** Es responsable de la gestión de archivos en la memoria interna del dispositivo utilizando el sistema SPIFFS (Serial Peripheral Interface Flash File System). Sus funciones principales incluyen:
    a. Sistema de lectura y escritura de archivos: Permite el acceso a los archivos almacenados en la memoria interna, asegurando la integridad de los datos.
    b. Estructura y actualización de parámetros de configuración: Gestiona los parámetros de configuración necesarios para el funcionamiento del dispositivo, almacenándolos en archivos de formato JSON que facilitan su manejo y actualización.
    c. Creación y reseteo de archivos necesarios para el punto de acceso: Genera y restablece los archivos que el sistema necesita para configurar y operar como punto de acceso WiFi.
2. **Net Module:** Se encarga de todas las funciones relacionadas con la conectividad de red del dispositivo. Entre sus responsabilidades se incluyen:
    a. Conexión WiFi como cliente a una red almacenada en memoria: Gestiona la conexión del dispositivo a redes WiFi previamente almacenadas.
    b. Configuración de red WiFi como Punto de Acceso: Permite que el dispositivo funcione como un punto de acceso WiFi, facilitando la conexión de otros dispositivos.
    c. Configuración de Servidor Websocket y sistema de comandos y eventos: Establece y gestiona un servidor Websocket para la comunicación en tiempo real, permitiendo el envío y recepción de comandos y eventos.
    d. Escaneo de redes WiFi disponibles para conexión: Realiza un escaneo de las redes WiFi disponibles en el entorno para permitir la selección y conexión a una de ellas.

    e. Verificación y sistema de reintentos de conexión a Internet: Implementa un sistema para verificar la conexión a Internet y realizar reintentos en caso de fallos.
3. **Auth Module:** Se encarga de la autenticación y sincronización de datos del dispositivo con la base de datos de Tablesa. Sus funciones incluyen:
    a. Verificación de ID registrada en Base de Datos de Tablesa: Comprueba que el ID del dispositivo esté registrado en la base de datos de la empresa.
    b. Sincronización: Completa la información local del dispositivo con los datos proporcionados por la base de datos, asegurando que el dispositivo esté actualizado con la información más reciente.
4. **OTA Module:** El OTA (Over-The-Air) Module facilita la actualización del firmware del dispositivo a través de la red. Sus funciones son:
    a. Descarga de Firmware y actualización: Gestiona la descarga y actualización del firmware del dispositivo, permitiendo mejoras y correcciones de errores sin necesidad de intervención física.
5. **LED Module:** Controla la matriz de LEDs del dispositivo, proporcionando retroalimentación visual sobre su estado. Sus responsabilidades incluyen:
    a. Control de Matriz de Leds: Gestiona la operación de los LEDs, permitiendo el encendido y apagado de luces específicas.
    b. Codificación de colores de LED por cada error: Asigna colores específicos a los LEDs para indicar diferentes tipos de errores, facilitando la identificación y resolución de problemas.
    c. Indicador de color de LED por cada estado de funcionamiento: Utiliza diferentes colores para indicar los estados operativos del dispositivo, proporcionando una referencia visual rápida y efectiva.

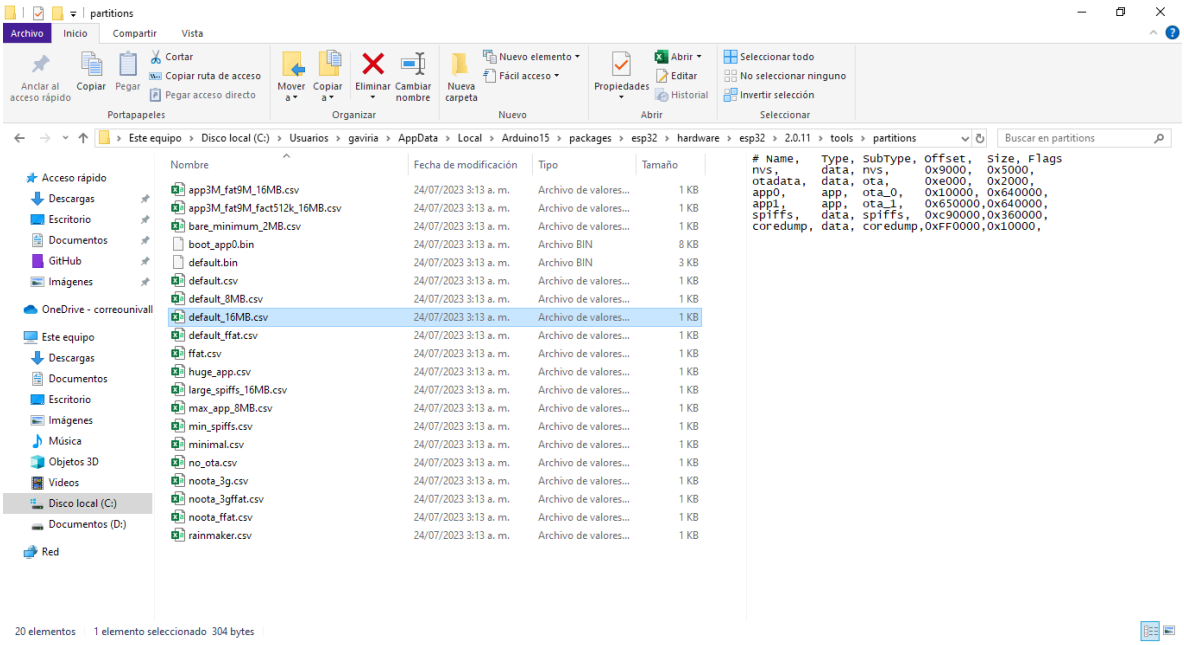
Estos módulos trabajan conjuntamente para asegurar que el firmware base funcione de manera eficiente y segura, proporcionando las bases necesarias para el posterior despliegue del firmware completo.

Para aprovechar la capacidad máxima de almacenamiento de 16MB de los ESP32 utilizados en los dispositivos de Tablesa, fue necesario realizar una configuración adicional en el entorno de desarrollo. Por defecto, el IDE de Arduino está configurado para utilizar solo 4MB de almacenamiento. Las figuras 3.23, 3.24 y 3.25 ilustran el proceso para modificar esta configuración y permitir la selección de la partición de 16 MB.

La Figura 3.23 muestra cómo acceder a la carpeta donde se instalan los archivos del ESP32 en el sistema operativo. En esta ubicación, se pueden ver los diferentes archivos de configuración de particiones disponibles para el ESP32. El archivo “default_16MB.csv” contiene la configuración de partición para utilizar los 16MB completos de almacenamiento disponibles en los dispositivos ESP32.

Figura 3.23

Particiones de memoria del ESP32 con 16MB de almacenamiento para Arduino IDE

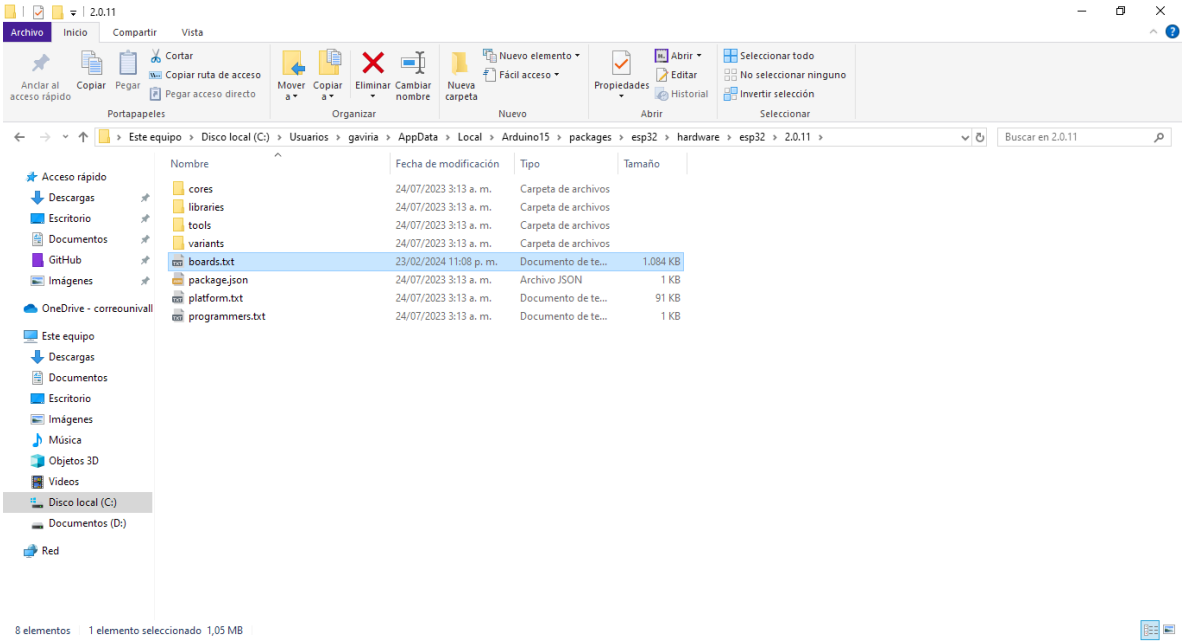


Nota: Ruta del explorador de archivos donde se encuentran las particiones por defecto C:\Users\username\AppData\Local\Arduino15\packages\esp32\hardware\esp32\2.0.11\tools\partitions

El siguiente paso es localizar el archivo “boards.txt”. Este archivo contiene las características y opciones de compilación para diferentes placas de desarrollo (boards). Es necesario agregar una configuración personalizada para la partición de 16MB con el fin de que el IDE pueda reconocer esta nueva partición. La Figura 3.24 muestra cómo acceder a la carpeta donde se encuentra el archivo “boards.txt”.

Figura 3.24

Ubicación de archivo de configuración de tarjetas de Arduino IDE para ESP32

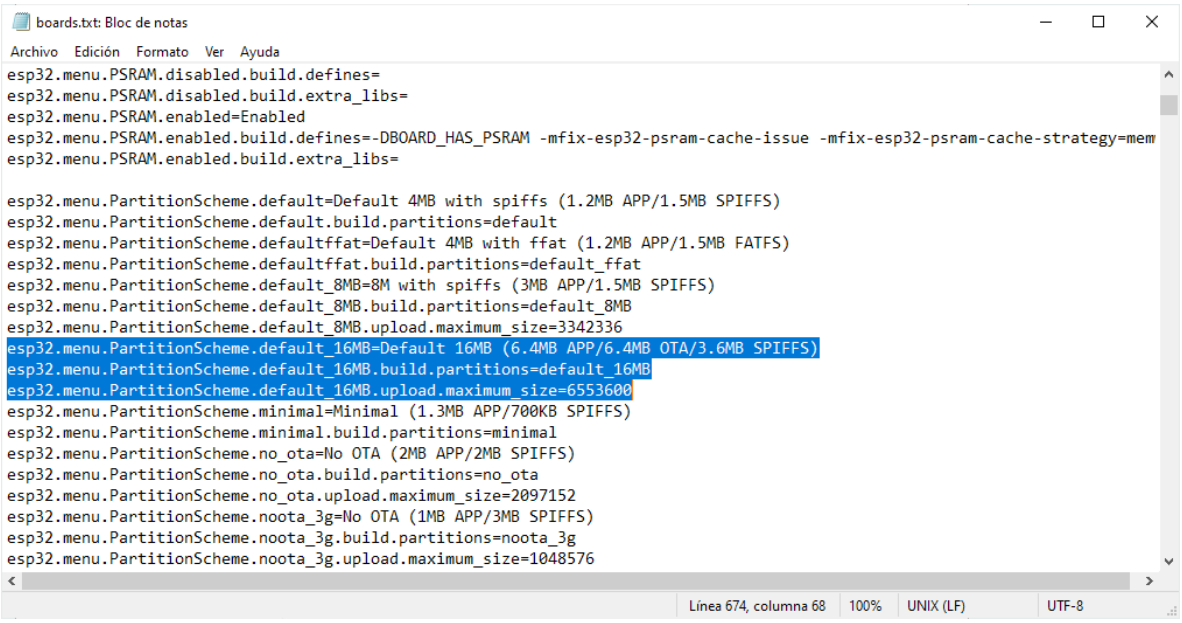


Nota: Ruta del explorador de archivos donde se encuentra el archivo boards.txt
C:\Users\username\AppData\Local\Arduino15\packages\esp32\hardware\esp32\2.0.11

La Figura 3.25 detalla los cambios necesarios en el archivo boards.txt para que la partición de 16MB aparezca en la lista seleccionable del IDE de Arduino. En este archivo, se deben añadir tres líneas de configuración que hacen referencia a la partición de 16MB. Estas líneas son esenciales para que el entorno de desarrollo reconozca y permita la selección de esta configuración de memoria al momento de compilar y cargar el firmware en el dispositivo.

Figura 3.25

Configuración de partición de 16MB para la tarjeta ESP32 Dev Module de Arduino IDE



boards.txt: Bloc de notas

Archivo Edición Formato Ver Ayuda

```
esp32.menu.PSRAM.disabled.build.defines=
esp32.menu.PSRAM.disabled.build.extra_libs=
esp32.menu.PSRAM.enabled=Enabled
esp32.menu.PSRAM.enabled.build.defines=-DBOARD_HAS_PSRAM -mfix-esp32-psram-cache-issue -mfix-esp32-psram-cache-strategy=mem
esp32.menu.PSRAM.enabled.build.extra_libs=

esp32.menu.PartitionScheme.default=Default 4MB with spiffs (1.2MB APP/1.5MB SPIFFS)
esp32.menu.PartitionScheme.default.build.partitions=default
esp32.menu.PartitionScheme.defaultffat=Default 4MB with ffat (1.2MB APP/1.5MB FATFS)
esp32.menu.PartitionScheme.defaultffat.build.partitions=default_ffat
esp32.menu.PartitionScheme.default_8MB=8M with spiffs (3MB APP/1.5MB SPIFFS)
esp32.menu.PartitionScheme.default_8MB.build.partitions=default_8MB
esp32.menu.PartitionScheme.default_8MB.upload.maximum_size=3342336
esp32.menu.PartitionScheme.default_16MB=Default 16MB (6.4MB APP/6.4MB OTA/3.6MB SPIFFS)
esp32.menu.PartitionScheme.default_16MB.build.partitions=default_16MB
esp32.menu.PartitionScheme.default_16MB.upload.maximum_size=6553600
esp32.menu.PartitionScheme.minimal=Minimal (1.3MB APP/700KB SPIFFS)
esp32.menu.PartitionScheme.minimal.build.partitions=minimal
esp32.menu.PartitionScheme.no_ota=No OTA (2MB APP/2MB SPIFFS)
esp32.menu.PartitionScheme.no_ota.build.partitions=no_ota
esp32.menu.PartitionScheme.no_ota.upload.maximum_size=2097152
esp32.menu.PartitionScheme.noota_3g=No OTA (1MB APP/3MB SPIFFS)
esp32.menu.PartitionScheme.noota_3g.build.partitions=noota_3g
esp32.menu.PartitionScheme.noota_3g.upload.maximum_size=1048576
```

Línea 674, columna 68 | 100% | UNIX (LF) | UTF-8

Estos cambios aseguran que los desarrolladores puedan seleccionar la partición de 16MB desde el menú de opciones del IDE, optimizando así el uso del almacenamiento disponible en los dispositivos ESP32 de Tablesa.

Este proceso permite a los dispositivos de Tablesa aprovechar al máximo su capacidad de almacenamiento, lo que es crucial para aplicaciones que requieren grandes cantidades de memoria, como la implementación de un servidor de punto de acceso o el almacenamiento de credenciales de autenticación.

Es importante tener en cuenta que para modificar la partición de un dispositivo IoT requiere de la conexión física, por lo que es indispensable realizar este formateo en etapas tempranas de la puesta en marcha del dispositivo. Esto concluyó que otro proceso importante por mejorar es la carga de firmware. La Figura 3.26 muestra la herramienta ESP32 Flash Download Tool V3.9.5, una herramienta esencial para la carga inicial de firmware y el formateo de particiones en los dispositivos ESP32.

Figura 3.26

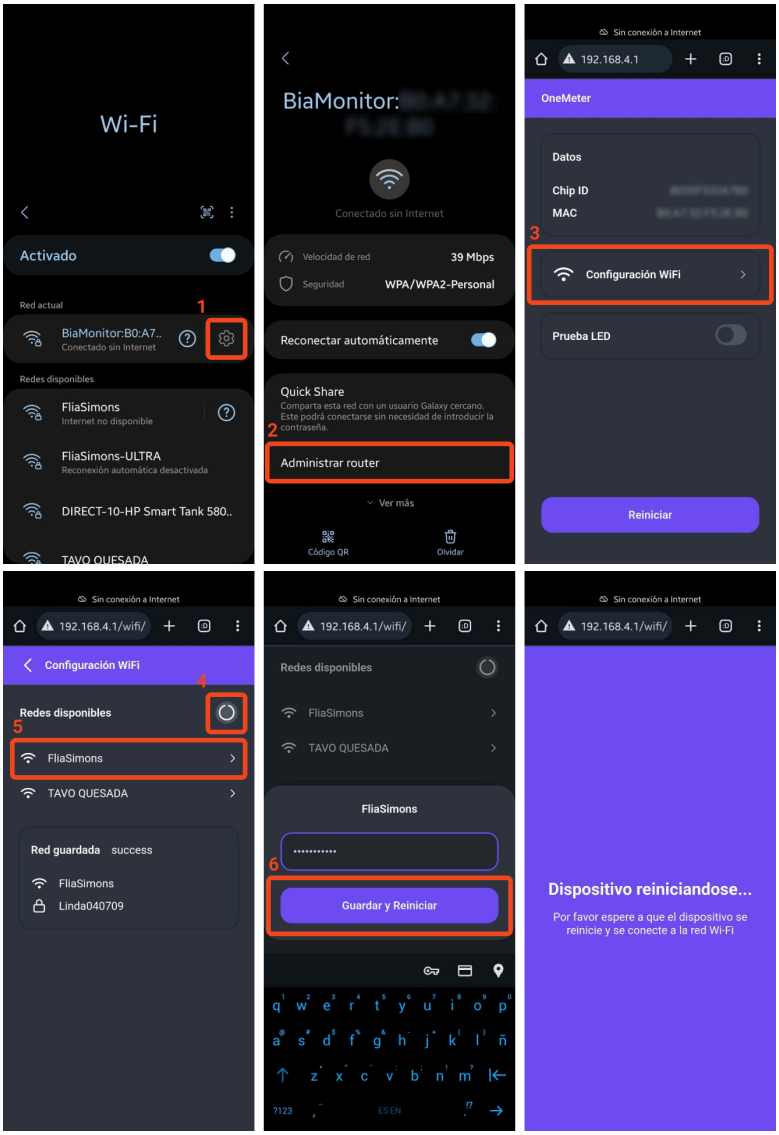
FlashTools para formateo y carga inicial de firmware en múltiples dispositivos

La utilización de esta herramienta es crucial para optimizar el proceso de puesta en marcha de los dispositivos de Tablesa. A través de la ESP32 Flash Download Tool V3.9.5, es posible programar múltiples dispositivos simultáneamente, lo que reduce significativamente el tiempo de puesta en marcha de los dispositivos.

Luego de tener programados los dispositivos con el firmware base, el paso siguiente es encender el modo punto de acceso y configurar la red WiFi. El Módulo de Configuraciones de red y Servidor de Punto de Acceso del dispositivo consiste en un servidor WebSocket que expone una serie de archivos estáticos necesarios para el funcionamiento de la interfaz web. Cuando un dispositivo se conecta al punto de acceso, se crea una nueva conexión bidireccional. Esta conexión se utiliza para el correcto funcionamiento de la visualización de datos de identificación del dispositivo, configuración WiFi y prueba de LED.

La Figura 3.27 muestra el procedimiento a seguir para la configuración de red WiFi, partiendo desde la conexión a la red generada por el dispositivo, pasando por el escaneo de redes WiFi disponibles, hasta el guardado del nombre y contraseña de la red seleccionada.

Figura 3.27
Configuración de red WiFi mediante Punto de Acceso generado por el dispositivo IoT



1. **Conexión a red WiFi generada:** En la primera pantalla, se muestra la conexión a la red WiFi generada por el dispositivo IoT, identificada como "BiaMonitor:...". Para conectarse, se debe seleccionar esta red en la lista de redes disponibles y comprobar que el estado muestre "Conectado sin Internet".
2. **Acceso a ruta de administración:** Para acceder a la configuración del dispositivo, se puede hacer desde el menú de configuración de la red WiFi (ícono de engranaje) y seleccionando "Administrar router".

Alternativamente, se puede abrir un navegador web y navegar a la dirección IP predeterminada del dispositivo: http://192.168.4.1/.

3. **Pantalla de configuración WiFi:** Una vez en la interfaz de administración del dispositivo, se debe hacer clic en "Configuración WiFi" para acceder a las opciones de configuración de la red.
4. **Escaneo de redes WiFi:** En la pantalla de configuración WiFi, se debe presionar el botón con el ícono de carga (representado por una flecha en círculo) para actualizar la lista de redes disponibles. Esto iniciará un escaneo de las redes WiFi cercanas que el dispositivo puede detectar.
5. **Seleccionar nombre de red WiFi:** Después de completar el escaneo, se mostrará una lista de redes WiFi disponibles. Se debe seleccionar el nombre de la red a la cual se desea conectar. En el ejemplo mostrado, se selecciona la red "FliaSimons".
6. **Guardar nombre y contraseña de red WiFi seleccionada:** Una vez seleccionada la red, se debe ingresar la contraseña correspondiente en el campo provisto. Después de ingresar la contraseña, se debe presionar el botón "Guardar y Reiniciar" para que el dispositivo almacene el nombre y contraseña de la red e intente conectarse automáticamente. La pantalla final muestra el dispositivo reiniciándose y conectándose a la red WiFi seleccionada.

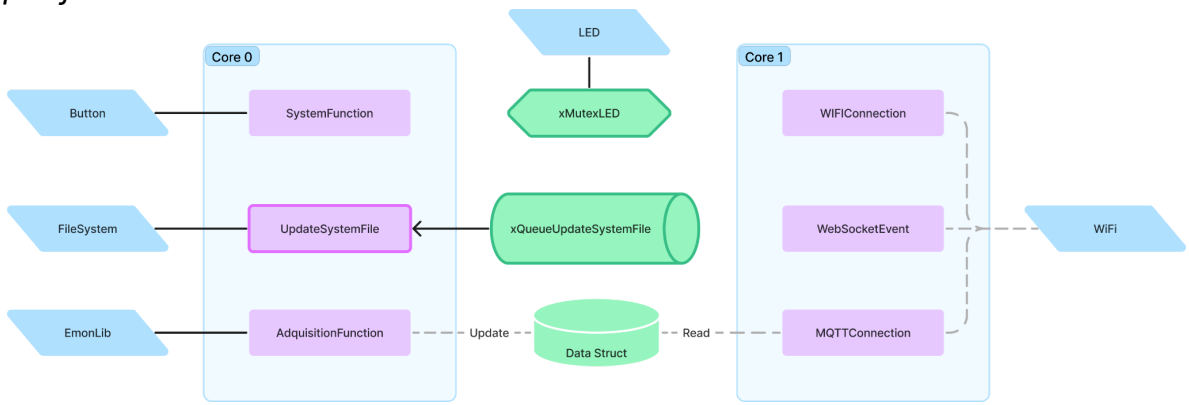
Este proceso permite dejar el dispositivo listo con la capacidad de autenticarse en el sistema y controlar la descarga del firmware completo. De esta manera, se asegura que el dispositivo esté completamente funcional y seguro sin comprometer la integridad del firmware completo.

3.3.3.2 Firmware completo con FreeRTOS

El desarrollo del firmware completo se centró en optimizar el funcionamiento de las tareas de adquisición y transmisión de datos, cumpliendo con el requerimiento principal de manejar dos conexiones MQTT con diferentes tiempos de muestreo, además de las funcionalidades del sistema, adquisición de datos, etc. Para lograr esto, se optó por implementar estas tareas en el micro-kernel FreeRTOS en su versión 10.4.3, aprovechando los dos núcleos del ESP32. [31]

La Figura 3.28 ilustra la distribución de tareas en los núcleos del ESP32, así como el consumo de recursos y periféricos. En el core 0, se definieron las tareas relacionadas con el sistema, incluyendo el control de periféricos como botones y la matriz de LEDs, así como la adquisición de datos. En el core 1, se ejecutaron las tareas de comunicación, como la transmisión de datos, la gestión de la conexión y el servidor de punto de acceso.

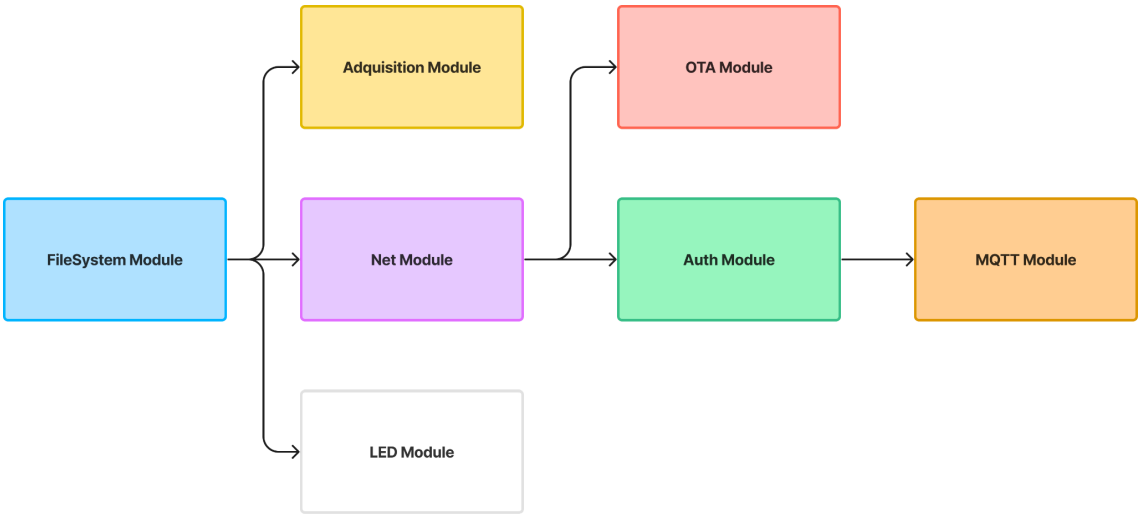
Figura 3.28
Distribución de tareas en núcleos del ESP32 consumo de recursos y periféricos



En busca de un funcionamiento estable del firmware, se implementaron varios sistemas de recursos compartidos:

- Mutex para la matriz de LEDs: Asegura el acceso controlado y seguro a los LEDs, evitando conflictos entre tareas.
- Data Struct para los datos adquiridos: Estructura de datos que almacena la información recopilada de los sensores, accesible por múltiples tareas.
- Queue para el FileSystem: Cola que maneja las operaciones del sistema de archivos, asegurando que las tareas accedan a los archivos de manera ordenada y sin interferencias.

Figura 3.29
Modularización de firmware completo y sus dependencias



Estas implementaciones permitieron una operación estable y sincronizada del firmware completo, asegurando que las tareas críticas de adquisición y transmisión de datos se ejecuten correctamente.

La Figura 3.29 muestra la modularización del firmware completo y sus dependencias. En comparación con el firmware base, el firmware completo incluye todos los módulos existentes más dos adicionales: el módulo de adquisición y el módulo de conexión MQTT.

- **Módulo de adquisición**: Encargado de recopilar datos de los sensores y otros dispositivos de entrada, asegurando que la información esté disponible para su procesamiento y transmisión.
- **Módulo de conexión MQTT**: Gestiona las conexiones MQTT, permitiendo la comunicación con servidores externos para el envío y recepción de datos. Este módulo es fundamental para manejar las dos conexiones MQTT teniendo en cuenta diferentes tiempos de muestreo y así cumplir con este requerimiento.

La modularización del firmware completo permitió una mejor organización y reutilización del código, facilitando el mantenimiento y la expansión de las funcionalidades del sistema. Al distribuir las tareas eficientemente entre los dos núcleos del ESP32 y utilizar FreeRTOS para la gestión de tareas, se logró un funcionamiento estable del dispositivo.

3.4 CONCLUSIONES

El diseño del sistema de operación y gestión para los dispositivos IoT de Tablesa S.A. se centró en la arquitectura basada en microservicios y

tecnologías serverless, lo cual proporcionó escalabilidad y flexibilidad. La implementación de capas de seguridad y mecanismos de actualización aseguró la integridad y el funcionamiento continuo de los dispositivos. Este enfoque permitió un diseño que facilita la integración y despliegue de nuevas funcionalidades, cumpliendo con los requisitos funcionales y no funcionales establecidos inicialmente.

La infraestructura del sistema, que incluye una base de datos relacional en DynamoDB y una API GraphQL, fue fundamental para la gestión y autenticación de dispositivos IoT. Estas tecnologías permitieron un manejo eficiente de los datos y una integración fluida con otros servicios, asegurando que el sistema pueda escalar y adaptarse a futuras necesidades. La implementación de una interfaz web para la visualización y gestión de dispositivos facilitó la administración remota y mejoró la usabilidad del sistema.

La implementación del sistema incluyó el desarrollo de un firmware base y un firmware completo, optimizado con FreeRTOS para manejar múltiples tareas de adquisición y transmisión de datos. La modularización del firmware y el uso de técnicas avanzadas de comunicación de tareas, como mutex y queues, garantizaron un funcionamiento estable de los dispositivos. La doble conexión MQTT permitió el envío de datos a diferentes destinatarios con distintos periodos de muestreo, cumpliendo con los requerimientos específicos de Tablesa S.A.

4.

PRUEBAS Y RESULTADOS

> *"No se puede mejorar lo que no se puede medir."*
>
> *- W. Edwards Deming*

4.1 INTRODUCCIÓN

En este capítulo se presentan las pruebas realizadas y los resultados obtenidos en el desarrollo del sistema de operación y gestión para los dispositivos IoT de Tablesa S.A., enfocados en el consumo energético. Se realizaron pruebas en varias etapas del proyecto, incluyendo pruebas de integración para asegurar que todas las funcionalidades del sistema operaran correctamente y cumplieran los requisitos establecidos. En este capítulo se describen los procedimientos realizados para ejecutar dichas pruebas, así como los resultados obtenidos.

4.2 PRUEBAS DE INTEGRACIÓN

Las pruebas de integración fueron fundamentales para asegurar que los diferentes módulos y componentes del sistema funcionen correctamente cuando se integran entre sí. Estas pruebas se enfocaron en verificar la interacción entre el firmware de los dispositivos, la infraestructura de red, la base de datos y la interfaz web. Se realizaron múltiples pruebas para evaluar la conectividad, la transferencia de datos, la autenticación y la actualización remota de firmware, entre otros aspectos críticos del sistema. Las siguientes sub-secciones detallan las pruebas específicas realizadas y los resultados obtenidos en cada una de ellas.

4.2.1 Puesta en marcha de dispositivos

Esta fase incluyó las pruebas de carga de firmware y captura de IDs en un lote de 97 dispositivos. El objetivo principal de esta prueba era demostrar una mejora en los tiempos de puesta en marcha, que previamente era de 24 minutos por dispositivo. Las Figuras 7.1 y 7.2, ubicadas en la sección de Anexos, muestran imágenes del proceso de carga de firmware en un lote de dispositivos. Estas imágenes destacan la preparación del lote y el uso del programa FlashTools para la carga del firmware. Adicionalmente, la Tabla 4.1 proporciona un formato detallado del procedimiento de prueba realizado.

Tabla 4.1
Prueba de puesta en marcha de 97 dispositivos IoT

Prueba de integración - Puesta en marcha de dispositivos
Requerimientos funcionales **1. Prueba de Dispositivos sin Compartir el Firmware Completo:** Tablesa S.A. necesitaba una manera de probar los dispositivos fabricados sin tener que compartir el firmware completo con las empresas fabricantes de dispositivos. Este requerimiento surgió de la necesidad de proteger la

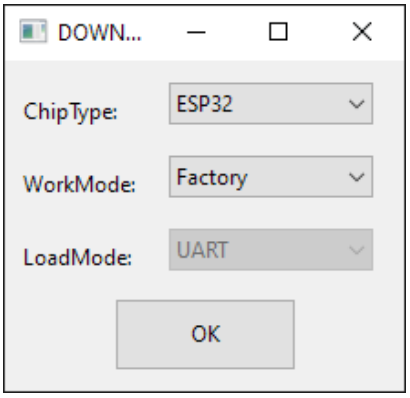
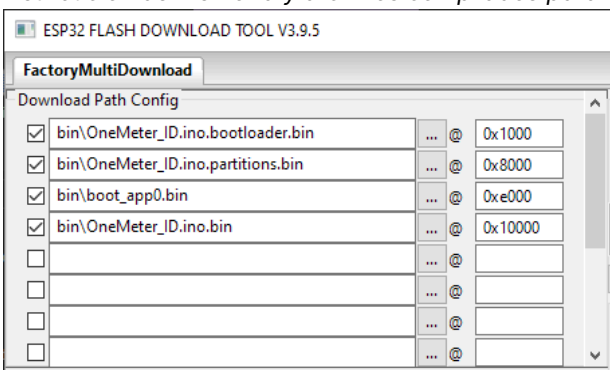
propiedad intelectual y mantener el control sobre el software crítico. Para cumplir con este requerimiento, se planteó el desarrollo de un firmware base que solo incluye las funcionalidades necesarias para probar el dispositivo, verificar su registro en el sistema, y permitir la descarga del firmware completo si el dispositivo está registrado. Esta solución permite a Tablesa probar los dispositivos de manera segura y eficiente, sin comprometer la integridad del firmware completo. **Requerimientos no funcionales** **1. Escalabilidad:** El sistema debe ser capaz de manejar un gran número de dispositivos, garantizando un rendimiento óptimo a medida que la cantidad de dispositivos crece.	
Tipo de prueba	Integración
Hardware requerido	97 Dispositivos IoT, Computador con sistema operativo Windows, Programador serial, cable USB
Software requerido	ESP32 Flash Download Tool V3.9.5, archivos compilados de firmware en formato .bin
Objetivo	Formatear partición y cargar firmware base a los dispositivos IoT
Carga de firmware y captura de IDs	
Datos de la prueba	Número de puertos seriales usados: 1 Número de dispositivos a programar: 97
Procedimiento	1. Conectar dispositivo al Computador usando el Programador serial y el cable USB. 2. Iniciar el programa FlashTools. 3. Seleccionar chip ESP32 y modo de trabajo FACTORY como se muestra en la Figura 4.3. **Figura 4.3** *Selección de chip y modo de trabajo en FlashTools*  4. Verificar la asignación de archivos binarios del firmware base a las particiones correspondientes en la sección “Path Config” como se muestra en la Figura 4.4.

Figura 4.4

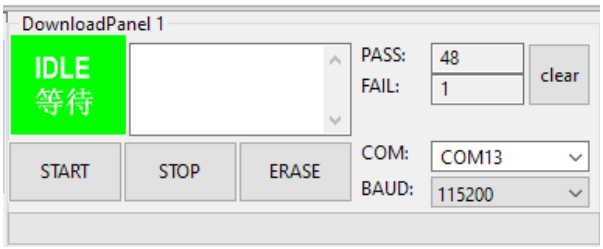
Distribución de memoria y archivos compilados para ESP32



5. Verificar conexión al puerto COM y rata de baudios correctos en el Panel 1 como se muestra en la Figura 4.5.

Figura 4.5

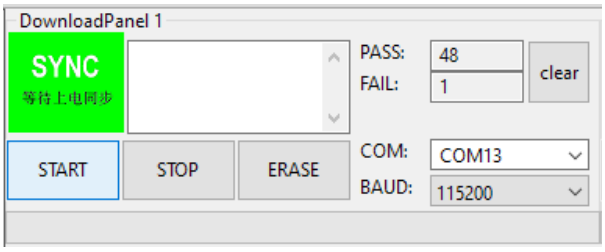
Estado inicial de panel de conexión de puerto serial 1



6. Presionar "START" para iniciar con la carga de firmware como se muestra en la Figura 4.6.

Figura 4.6

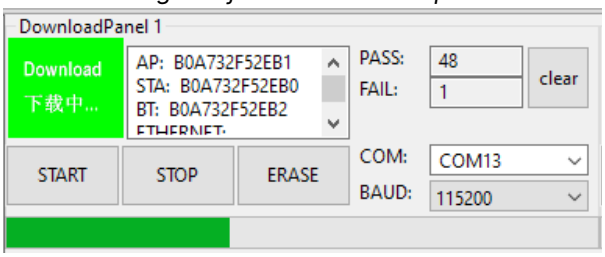
Estado de sincronización de conexión de panel 1

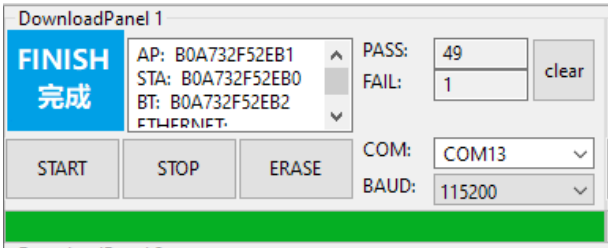
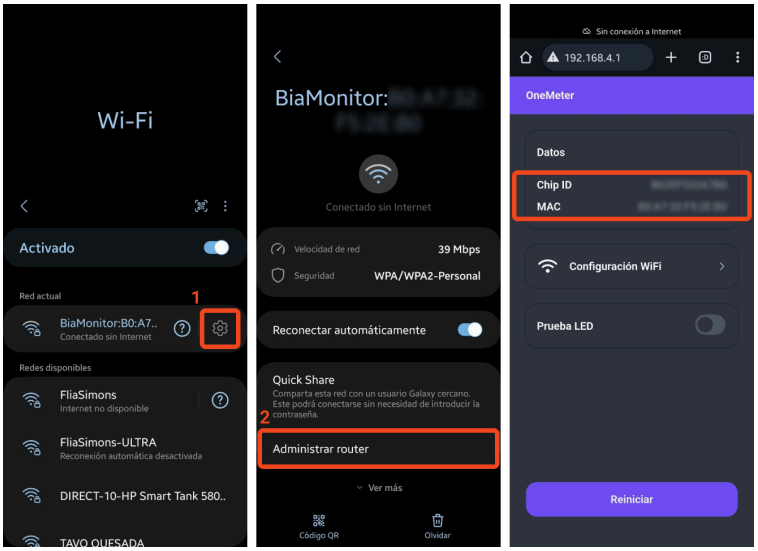


7. Esperar a que termine la carga de firmware. La Figura 4.7 muestra cómo debe verse el panel en este estado.

Figura 4.7

Estado de carga de firmware base de panel 1



	8. Cuando termine la carga, desconectar el dispositivo del puerto serial. La Figura 4.8, muestra cómo debe verse el panel en este estado **Figura 4.8** *Estado de finalización de carga de firmware de panel 1*  9. Reiniciar el dispositivo. Esperar hasta que se encienda automáticamente el modo punto de acceso. (puede tardar hasta 2 minutos la primera vez) 10. Conectarse al punto de acceso realizando los pasos 1 y 2 vistos en la Figura 4.9 **Figura 4.9** *Pasos 1 y 2 de conexión a punto de acceso del dispositivo IoT*  11. Copiar los datos identificadores mostrados en la interfaz en una hoja de cálculo 12. Repetir el proceso hasta cargar el firmware y capturar IDs en todos los dispositivos
Resultado esperado	• Archivo de excel con identificadores de los dispositivos • Mejorar los tiempos de puesta en marcha de dispositivos del sistema actual de tablesa: 24 minutos por dispositivo
Resultado obtenido	• Archivo de excel con identificadores de dispositivos • 97 dispositivos con carga de firmware base exitosa • Captura de tiempos a 47 dispositivos. Ver Tabla 7.2 • Tiempo promedio de carga de firmware base por dispositivo: 1.33 minutos • Tiempo promedio de captura de IDs por dispositivo: 2.97 minutos • Tiempo total de puesta en marcha aproximado por dispositivo: 4.32

	minutos.
Comentarios	• Luego de finalizar la carga de firmware por primera vez en el dispositivo, requiere mantenerse los primeros 2 minutos encendido para restablecer su sistema de almacenamiento con SPIFFS. • Mejora del 82.02% en tiempos de puesta en marcha de dispositivos con respecto al sistema anterior. • Los tiempos de puesta en marcha pueden multiplicarse haciendo uso de más puertos seriales. En esta prueba solo se utilizó uno.

4.2.2 Registro de dispositivos

En esta etapa se probó el proceso de registro de un lote de dispositivos utilizando una lista de IDs capturados previamente en un archivo de Excel durante la prueba de puesta en marcha. Haciendo uso de la interfaz web, se siguieron una serie de pasos para registrar los dispositivos en la base de datos del sistema, asegurando su correcta autenticación. La Tabla 4.2 detalla el procedimiento de registro, incluyendo los pasos específicos y los criterios de aceptación.

Tabla 4.2

Prueba de Registro de dispositivos IoT a partir de una lista de IDs

Prueba de integración - Registro de dispositivos	
Requerimientos funcionales **3. Registro de dispositivos:** El sistema debe permitir a los usuarios registrar nuevos dispositivos IoT. El usuario proporciona los detalles del dispositivo y el sistema guarda esta información en la base de datos. **6. Monitoreo de dispositivos:** El sistema debe proporcionar una interfaz para que los usuarios puedan consultar o monitorear el estado operativo de los dispositivos IoT. Los dispositivos reportan su estado de conexión al sistema, que se puede ver reflejada en la interfaz web para los usuarios. **9. Interfaz de usuario intuitiva:** La interfaz web debe ser fácil de usar y permitir la gestión y monitoreo de los dispositivos. **Requerimientos no funcionales** **1. Escalabilidad:** El sistema debe ser capaz de manejar un gran número de dispositivos, garantizando un rendimiento óptimo a medida que la cantidad de dispositivos crece. **2. Seguridad:** Todos los datos y comunicaciones deben estar protegidos contra accesos no autorizados mediante el uso de cifrado y autenticación segura.	
Tipo de prueba	Integración
Hardware requerido	Computador con sistema operativo Windows o MacOS y conexión a internet

Software requerido	Microsoft Excel, Archivo de hoja de cálculo con identificadores de los dispositivos a registrar, Navegador web
Objetivo	Registrar una flota de dispositivos desde la interfaz web, a partir de un archivo de Excel
Registro de una flota de dispositivos desde la interfaz web	
Datos de la prueba	Archivo excel de prueba con tres identificadores. Ver Figura 4.10:

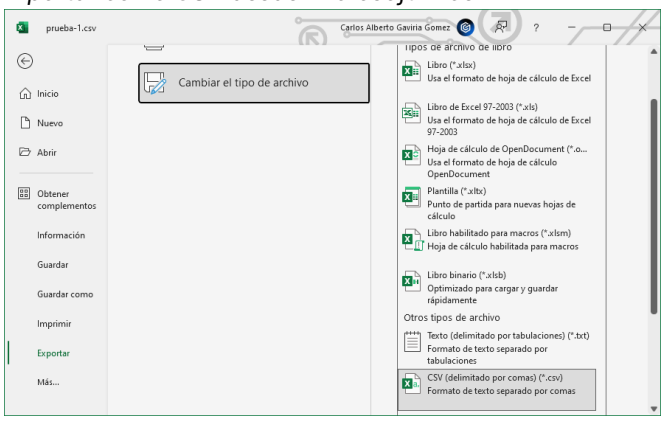
Figura 4.10
Archivo de excel test.csv

	A	B	C
1	item	chipID	mac
2	1	1	00:00:00:00:00:01
3	2	2	00:00:00:00:00:02
4	3	3	00:00:00:00:00:03

Procedimiento

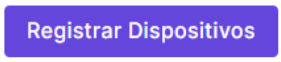
1. Abrir archivo con Microsoft Excel
2. Exportar como CSV en la ruta: Archivo / Exportar / Cambiar el tipo de Archivo / CSV (delimitado por comas) como se muestra en la Figura 4.11

Figura 4.11
Exportar como CSV desde Microsoft Excel



3. Verificar que el archivo resultante mantenga la estructura mostrada en la Figura 3.5
4. Abrir aplicativo web e iniciar sesión
5. Click en botón “Registrar dispositivos” como se muestra en la Figura 4.12.

Figura 4.12
Botón de registro de dispositivos



6. Seleccionar los campos de los pasos 1 y 2 mostrados en la Figura 4.13

Figura 4.13
Pasos 1 y 2 del formulario de registro de dispositivos

7. En el paso 3, dar click en el botón “Buscar archivos” como se muestra en la Figura 4.14

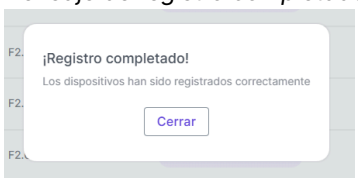
Figura 4.14
Paso 3. Carga de archivo .csv

8. Escoger archivo CSV y dar click en “Cargar 1 archivo” mostrada en la Figura 4.15 para iniciar el registro de dispositivos.

Figura 4.15
Opciones para cargar el archivo seleccionado

9. Esperar hasta que se muestre en pantalla el mensaje “Registro completado” visto en la Figura 4.16

Figura 4.16
Mensaje de registro completado exitosamente

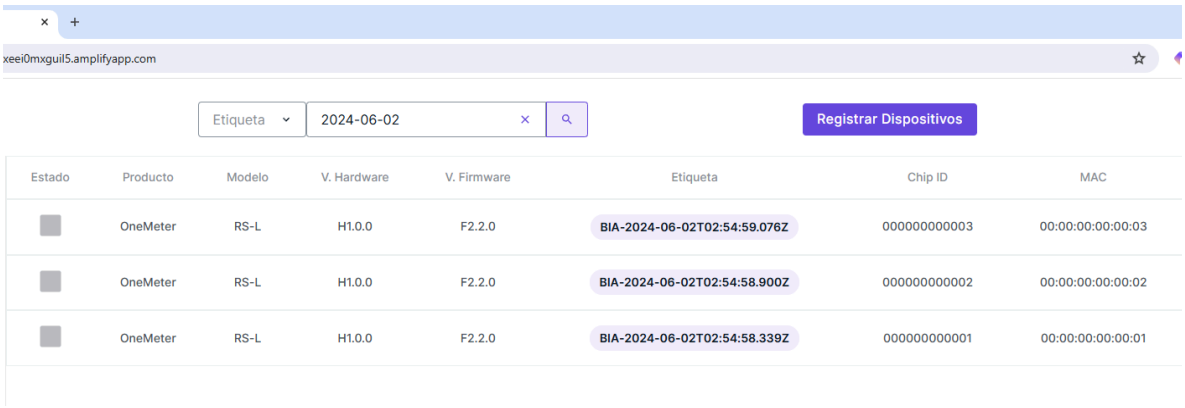


10. Verificar el registro ingresando la fecha en la barra de búsqueda como se muestra en la siguiente Figura 4.17

Resultado esperado	Registro de tres dispositivos

Resultado obtenido	Registro de tres dispositivos
Comentarios	• Es muy importante mantener la estructura mostrada en la Figura 3.5, respetando mayúsculas y minúsculas en los nombres de las columnas y separaciones con punto y coma. De lo contrario el sistema no registrará los dispositivos.

Figura 4.17
Verificación de registro usando la interfaz web



4.2.3 Autenticación de dispositivos y actualización de firmware

En esta etapa, se verificó que los dispositivos registrados pudieran autenticarse con el sistema y ejecutar el proceso de actualización de firmware para obtener todas las funcionalidades del firmware completo. Utilizando la lista de dispositivos registrados, se evaluó uno de ellos para conectarse de manera segura al sistema y descargar la actualización de firmware necesaria. La Tabla 4.3 detalla el procedimiento seguido y requerimientos evaluados.

Tabla 4.3
Prueba de Autenticación de dispositivos

Prueba de integración - Autenticación de dispositivos
Requerimientos funcionales **4. Autenticación segura de dispositivos:** Los dispositivos IoT deben poder autenticarse de manera segura con el sistema central. Los dispositivos envían sus credenciales al sistema, que verifica su autenticidad antes de permitir el acceso a otras funcionalidades. **6. Monitoreo de dispositivos:** El sistema debe proporcionar una interfaz para que los usuarios puedan consultar o monitorear el estado operativo de los dispositivos IoT. Los dispositivos reportan su estado de conexión al sistema, que se puede ver reflejada en la interfaz web para los usuarios.

8. Actualización remota de firmware: El sistema debe permitir la actualización de firmware de los dispositivos de manera remota. Los dispositivos IoT deben permitir la actualización remota de su Firmware.

9. Interfaz de usuario intuitiva: La interfaz web debe ser fácil de usar y permitir la gestión y monitoreo de los dispositivos.

Requerimientos no funcionales

1. Escalabilidad: El sistema debe ser capaz de manejar un gran número de dispositivos, garantizando un rendimiento óptimo a medida que la cantidad de dispositivos crece.

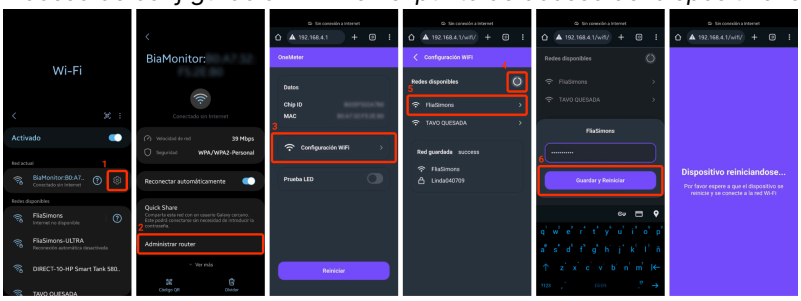
2. Seguridad: Todos los datos y comunicaciones deben estar protegidos contra accesos no autorizados mediante el uso de cifrado y autenticación segura.

Tipo de prueba	Integración
Hardware requerido	Dispositivo IoT, Dispositivo móvil, Red WiFi con conexión a internet.
Software requerido	Navegador web
Objetivo	Actualizar firmware de un dispositivo registrado
Verificación de registro y Actualización a Firmware Completo mediante módulo OTA	
Datos de la prueba	• El dispositivo IoT debe estar registrado y programado con Firmware base • Se deben conocer el nombre y contraseña de la red WiFi
Procedimiento	1. Abrir aplicativo web e iniciar sesión 2. Buscar el dispositivo y verificar su estado “Registrado” como se muestra en la Figura 4.18. **Figura 4.18** *Dispositivo en estado “Registrado”* 3. Energizar el dispositivo IoT. Esperar hasta que se encienda automáticamente el modo punto de acceso. 4. Conectarse al punto de acceso y realizar todos los pasos vistos en la Figura 3.27 para la configuración WiFi como se muestra en la Figura 4.19.

<table>
<tr>
<td></td>
<td>

Figura 4.19

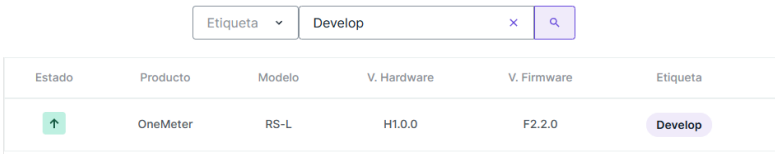
Proceso de configuración WiFi en el punto de acceso del dispositivo IoT



5. Esperar a que reinicie el dispositivo
6. Una vez reiniciado el dispositivo, verificará el registro y hará su actualización al Firmware Completo. Esperar un minuto a que termine el proceso.
7. Volver a ejecutar la búsqueda en el aplicativo web para verificar el estado de conexión del dispositivo. Debe haber pasado de estado Registrado a Conectado como se muestra en la Figura 4.20.

Figura 4.20

Dispositivo en estado "Conectado"



</td>
</tr>
<tr>
<td>Resultado esperado</td>
<td>Dispositivo conectado al sistema</td>
</tr>
<tr>
<td>Resultado obtenido</td>
<td>Dispositivo conectado al sistema</td>
</tr>
<tr>
<td>Comentarios</td>
<td></td>
</tr>
</table>

4.2.4 Almacenamiento de datos adquiridos

En esta etapa, se verificó el correcto almacenamiento de los datos adquiridos por los dispositivos IoT en la base de datos del sistema. Utilizando los dispositivos registrados y actualizados, se realizaron pruebas para asegurar que los datos recolectados, como lecturas de sensores y estados operativos, fueran transmitidos y almacenados adecuadamente. La Tabla 4.4 detalla el procedimiento seguido y los requerimientos evaluados.

Tabla 4.4

Prueba de Almacenamiento de datos adquiridos

Prueba de integración - Almacenamiento de datos adquiridos	
Requerimientos funcionales **2. Envío de Datos a Múltiples Destinatarios con Diferentes Periodos de Muestreo:** Tablesa S.A. requería que los dispositivos IoT pudieran enviar datos tanto al broker de Tablesa como al cliente que compró el dispositivo, pero con diferentes periodos de muestreo. Específicamente, se necesitaba un periodo de muestreo de 5 segundos para el broker de Tablesa y un periodo de muestreo de 0.5 segundos para el cliente. Para abordar este requerimiento, se planteó una doble conexión MQTT en el firmware de los dispositivos. Esto permite que los dispositivos envíen datos simultáneamente a ambos destinatarios con los periodos de muestreo especificados. Esta funcionalidad asegura que tanto Tablesa como sus clientes reciban los datos necesarios en los intervalos adecuados para sus respectivos propósitos. **5. Almacenamiento de datos adquiridos:** El sistema debe ser capaz de almacenar los datos adquiridos por los dispositivos IoT de manera estructurada y segura. Los dispositivos envían datos al sistema, que luego almacena en la base de datos. **6. Monitoreo de dispositivos:** El sistema debe proporcionar una interfaz para que los usuarios puedan consultar o monitorear el estado operativo de los dispositivos IoT. Los dispositivos reportan su estado de conexión al sistema, que se puede ver reflejada en la interfaz web para los usuarios. **9. Interfaz de usuario intuitiva:** La interfaz web debe ser fácil de usar y permitir la gestión y monitoreo de los dispositivos. **Requerimientos no funcionales** **3. Confiabilidad:** El sistema debe ser robusto y capaz de funcionar de manera continua sin interrupciones significativas, asegurando alta disponibilidad y resistencia a fallos.	
Tipo de prueba	Integración
Hardware requerido	Dispositivo IoT, Computador con sistema operativo Windows o MacOS y conexión a internet
Software requerido	Navegador web
Objetivo	Almacenar datos adquiridos en Base de datos
Escritura de datos adquiridos en Base de datos	
Datos de la prueba	• El dispositivo IoT debe estar Conectado al sistema • Se debe identificar el dispositivo por identificador o etiqueta
Procedimiento	1. Abrir aplicativo web e iniciar sesión 2. Buscar el dispositivo y verificar su estado “Conectado” como se muestra en la Figura 4.20 3. Verificar las tablas Data y Log en el servicio AWS DynamoDB como se muestra en las Figuras 4.21 y 4.22

Resultado esperado	• Registros de cambios de estado de conexión • Registros de datos adquiridos cada 5 segundos
Resultado obtenido	• Numerosos registros de cambios de estado de conexión • Registros de datos adquiridos cada 5 segundos
Comentarios	• Los registros de la tabla Data respetan el periodo de publicación de mensajes de 5 segundos. • Los procesos de pérdida y reconexión al broker de AWS IoT son más frecuentes de lo esperado y sin ningún tipo de patrón. Pueden haber dispositivos que tengan una conexión estable que dure días o semanas sin reintentos de conexión, como también dispositivos que pueden mostrar más de una reconexión diaria. Este comportamiento puede relacionarse a la baja cobertura de la señal inalámbrica, problemas de calidad en la red WiFi o fluctuaciones en el proveedor del servicio de broker MQTT. La solución planteada para hacer frente a este problema fue enfocarse en hacer el sistema de reconexión más robusto, verificando constantemente la calidad de la conexión al broker mqtt (cada minuto) y reconectando según se requiera.

Figura 4.21

Muestra de datos adquiridos el día 28 de mayo del 2024 en el minuto 23:51

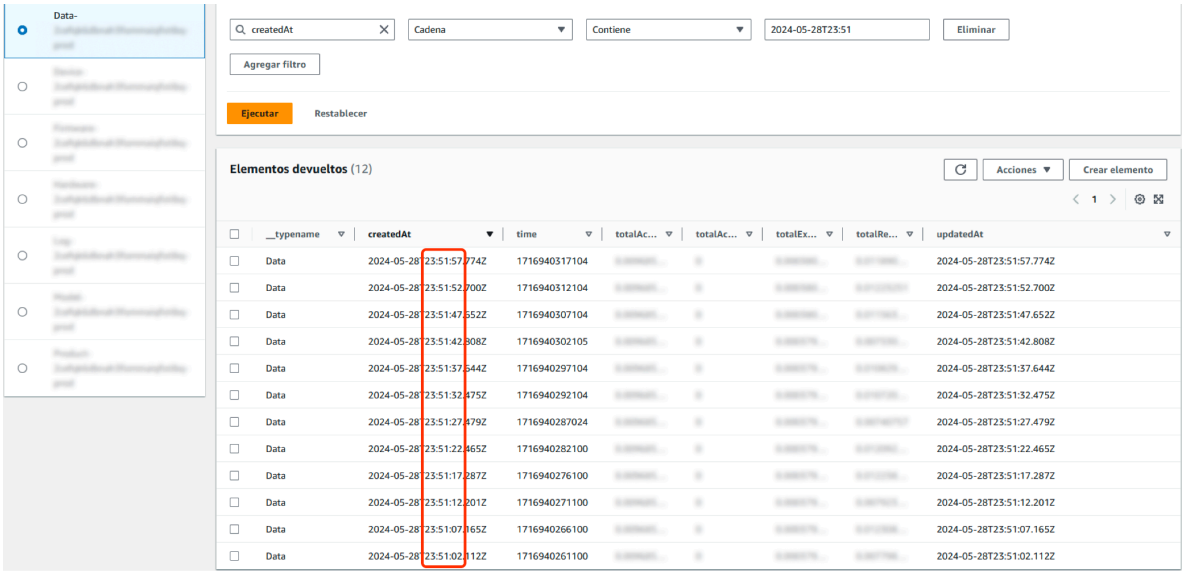
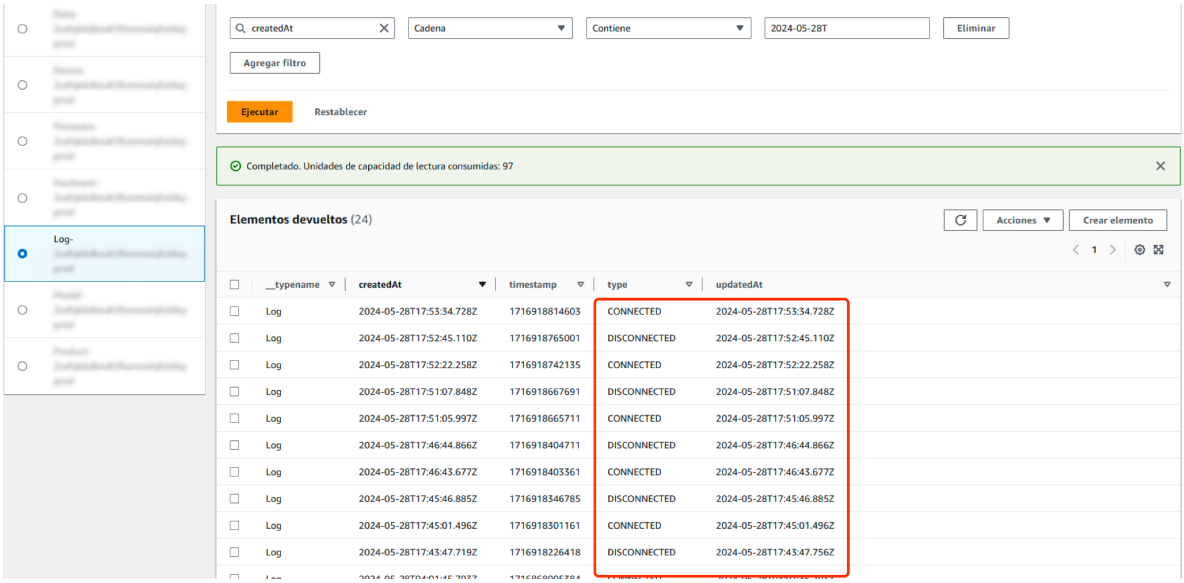


Figura 4.22

Muestra de registros de estados de conexión el día 28 de mayo del 2024



4.2.5 Configuración remota de parámetros

En esta etapa, se utilizó la interfaz web para establecer una conexión MQTT con el dispositivo seleccionado y configurar remotamente sus parámetros. Este proceso permitió ajustar las configuraciones del dispositivo, verificando que los cambios se aplicaran correctamente. La Tabla 4.5 detalla el procedimiento seguido y los requerimientos evaluados.

Tabla 4.5

Prueba de Configuración remota de parámetros

Prueba de integración - Configuración remota de parámetros
Requerimientos funcionales **6. Monitoreo de dispositivos:** El sistema debe proporcionar una interfaz para que los usuarios puedan consultar o monitorear el estado operativo de los dispositivos IoT. Los dispositivos reportan su estado de conexión al sistema, que se puede ver reflejada en la interfaz web para los usuarios. **7. Modificar parámetros de configuración:** Los usuarios deben tener la capacidad de modificar los parámetros de configuración de los dispositivos IoT a través del sistema. Al enviar nuevas configuraciones, el sistema las aplica a los dispositivos correspondientes. **9. Interfaz de usuario intuitiva:** La interfaz web debe ser fácil de usar y permitir la gestión y monitoreo de los dispositivos. **Requerimientos no funcionales** **3. Confiabilidad:** El sistema debe ser robusto y capaz de funcionar de manera continua sin

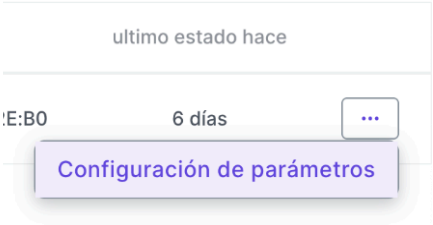
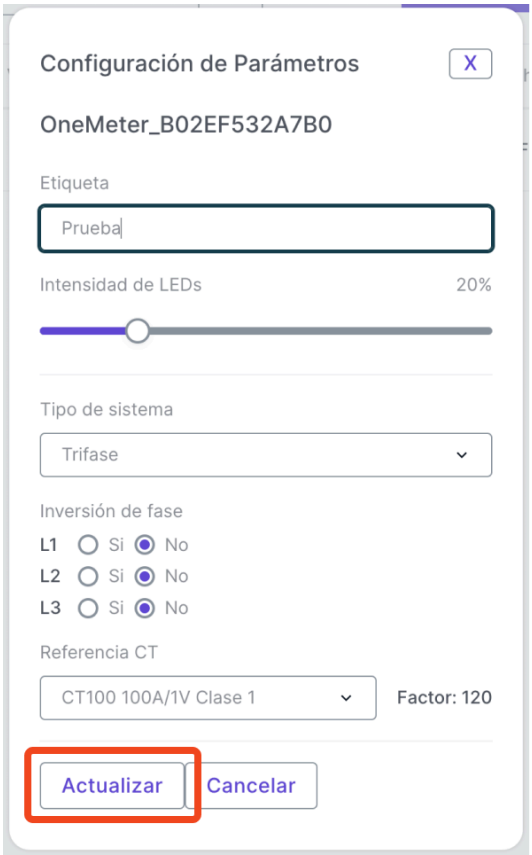
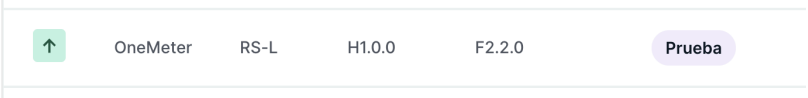
interrupciones significativas, asegurando alta disponibilidad y resistencia a fallos.	
Tipo de prueba	Integración
Hardware requerido	Dispositivo IoT, Computador con sistema operativo Windows o MacOS y conexión a internet
Software requerido	Navegador web
Objetivo	Configurar los parámetros de un dispositivo desde el aplicativo web
Configuración de parámetros desde la interfaz web	
Datos de la prueba	• El dispositivo IoT debe estar Conectado al sistema • Se debe identificar el dispositivo por identificador o etiqueta
Procedimiento	1. Abrir aplicativo web e iniciar sesión 2. Buscar el dispositivo y verificar su estado “Conectado” como se muestra en la Figura 4.20 3. Seleccionar el botón de ajustes del dispositivo como se muestra en la Figura 4.23. **Figura 4.23** *Botón de ajustes del dispositivo*  4. Esperar 10 segundos a que termine el proceso de sincronización con el dispositivo. La Figura 4.24 muestra la pantalla de espera mientras se completa la sincronización.

Figura 4.24
Sincronización de interfaz con dispositivo para configuración de parámetros

5. Editar valores del formulario y dar click en el botón "Actualizar". Para esta prueba se actualizó la etiqueta del dispositivo de "Develop" a "Prueba" como se muestra en la Figura 4.25

Figura 4.25
Edición de parámetro "Etiqueta" en dispositivo IoT



	6. Volver a ejecutar la búsqueda en el aplicativo web para verificar la actualización en pantalla como se muestra en la Figura 4.26. **Figura 4.26** *Visualización de actualización de parámetro "Etiqueta" en un dispositivo* 
Resultado esperado	• Si se realizó un cambio en el campo "Etiqueta", visualizarlo en la interfaz
Resultado obtenido	• Verificación de actualización del campo "Etiqueta"
Comentarios	• El dispositivo debe esperar 10 segundos para verificar que el dispositivo está conectado y enviando datos correctamente.

4.3 CONCLUSIONES

El capítulo 4 presenta los resultados de las pruebas realizadas para verificar el funcionamiento del sistema de operación y gestión para la familia de dispositivos IoT de Tablesa S.A. A través de una serie de pruebas exhaustivas, se validaron las funcionalidades clave del sistema, lo que permitió obtener varias conclusiones significativas:

1. Las pruebas de integración iniciales demostraron que los dispositivos IoT pueden ser configurados y puestos en marcha de manera efectiva. La implementación de herramientas como FlashTools facilitó la carga y configuración de firmware en múltiples dispositivos simultáneamente, optimizando así el proceso de preparación de los dispositivos para su uso operativo y evitando compartir el firmware completo.
2. El sistema de registro y autenticación de dispositivos funcionó correctamente, permitiendo la verificación segura de los dispositivos antes de su despliegue en el entorno productivo. Esto garantiza que sólo los dispositivos autorizados puedan comunicarse con la infraestructura de Tablesa.
3. La capacidad de actualización remota de firmware se validó con éxito, lo que permite mantener los dispositivos actualizados sin necesidad de intervención manual.
4. El sistema de almacenamiento de datos adquiridos demostró ser robusto y eficiente, permitiendo la recolección y estructuración de grandes volúmenes de datos generados por los dispositivos IoT y manteniendo el periodo de muestreo necesario.

5. La interfaz web de administración proporcionó una plataforma intuitiva y accesible para la visualización y gestión de los dispositivos. Los usuarios pudieron monitorizar el estado de conexión de los dispositivos, así como modificar parámetros de configuración de manera remota.

Inicialmente, se planeó realizar una prueba de campo para validar la integración del sistema con dispositivos IoT en un entorno de producción real. Sin embargo, la realización completa de esta prueba requería no solo permisos de lectura de los servicios en la nube utilizados por los clientes que compran los dispositivos, sino también un tiempo considerable para el proceso de compra, envío, instalación en las plantas del cliente y la puesta en marcha de los dispositivos. Dado que estos requisitos excedían el tiempo disponible para la entrega del proyecto dentro de las fechas estimadas, se decidió no incluir esta prueba en el presente documento. Las pruebas de integración, documentadas previamente en la sección 4.2, cubren los aspectos necesarios para validar el funcionamiento del sistema en las etapas de operación y gestión del ciclo de vida de los dispositivos IoT de Tablesa.

En resumen, las pruebas realizadas confirmaron que el sistema de operación y gestión para la familia de dispositivos IoT de Tablesa S.A. es una solución viable para su administración en entornos industriales. El sistema desarrollado cumplió con los requerimientos funcionales y no funcionales establecidos al inicio del proyecto.

5.
CONCLUSIONES Y TRABAJOS FUTUROS

> *"La ciencia de hoy es la tecnología del mañana."*
>
> *- Edward Teller*

5.1 CONCLUSIONES

El desarrollo del sistema de operación y gestión para la familia de dispositivos IoT de Tablesa S.A. ha evidenciado hallazgos significativos en diversos aspectos críticos. A partir del análisis de los antecedentes, se identificaron las propiedades más relevantes que deben tenerse en cuenta en estos sistemas: la seguridad en la autenticación de dispositivos, la capacidad de actualizaciones de firmware OTA, el almacenamiento estructurado de datos y la gestión eficiente mediante interfaces intuitivas.

Considerando el análisis de antecedentes y las necesidades de Tablesa S.A., los requerimientos funcionales del sistema incluyeron la necesidad de registrar y autenticar dispositivos de manera segura, permitir actualizaciones de firmware remotas, almacenar y gestionar datos adquiridos, y proporcionar una interfaz de usuario que facilitara la visualización y administración de los dispositivos IoT. Para abordar estos requerimientos, se diseñaron e implementaron diagramas de infraestructura, un modelo estructurado para la base de datos y una interfaz de usuario intuitiva.

La implementación se realizó utilizando una arquitectura basada en microservicios y tecnologías serverless en Amazon Web Services (AWS). Se integraron herramientas como AWS Amplify, Cognito y DynamoDB, y se empleó el marco ágil SCRUM para la gestión del proyecto. Este enfoque permitió el desarrollo iterativo y adaptativo, facilitando la incorporación de mejores prácticas en el ciclo de vida del desarrollo de software y conceptos como la Integración Continua y Despliegue Continuo (CI/CD).

Las pruebas realizadas demostraron que el sistema es escalable, seguro y eficiente. Se lograron mejoras concretas como una reducción del 82.02% en los tiempos de puesta en marcha de dispositivos, comparado con el sistema anterior. Este avance permite una preparación y despliegue más rápido y eficiente de los dispositivos IoT, reduciendo significativamente el tiempo y esfuerzo requeridos.

La adopción de un marco de trabajo ágil como SCRUM, junto con el uso de servicios en la nube y herramientas de desarrollo como AWS Amplify, facilitó la creación de un mínimo producto viable en la mitad del tiempo presupuestado inicialmente. Esto demuestra la eficacia de las metodologías ágiles y las herramientas modernas en la aceleración del ciclo de desarrollo.

La arquitectura basada en microservicios y tecnologías serverless proporcionó la escalabilidad y flexibilidad necesarias para soportar el

crecimiento futuro de la infraestructura IoT de Tablesa. Esta arquitectura permite la adición de nuevas funcionalidades y la adaptación a cambios en los requisitos sin comprometer el rendimiento del sistema.

El uso de FreeRTOS en el firmware permitió la independencia de tareas críticas como la adquisición de datos y la transmisión de mensajes. Además, esta arquitectura flexible facilita la incorporación de nuevas funcionalidades en forma de tareas independientes, sin afectar las funcionalidades existentes.

5.1.1 Limitaciones

Durante la implementación del sistema se identificaron varias limitaciones, las cuales pueden tener implicaciones en el rendimiento o la usabilidad futura del sistema:

- **Velocidad de escritura en la memoria flash:** Para preservar la vida útil de la memoria flash del ESP32, se recomienda limitar la velocidad de escritura a una vez cada 5 segundos.
- **Almacenamiento limitado para archivos estáticos:** El sistema solo admite archivos estáticos necesarios para la ejecución del servidor web que no superen los 3MB.
- **Uso de dos núcleos con FreeRTOS:** El sistema utiliza dos núcleos del ESP32 para el manejo de multitareas, lo que puede limitar su capacidad de escalar a procesos más complejos en el futuro.
- **Verificación de la conexión al broker MQTT:** El sistema verifica la conexión al broker MQTT cada minuto, lo que significa que el tiempo de reacción ante una pérdida de conexión no puede ser menor a un minuto.
- **Circuito de alimentación de respaldo:** El sistema requiere de un circuito de alimentación de respaldo, para evitar reinicios abruptos del dispositivo. Sin este respaldo, los procesos de escritura en la memoria flash podrían interrumpirse, lo que conlleva el riesgo de corromper archivos críticos de almacenamiento.

5.1.2 Desafíos

El desarrollo e implementación del sistema presentaron varios desafíos técnicos que requirieron soluciones específicas:

- **Sistema de almacenamiento seguro:** Se implementó un sistema de almacenamiento que permite guardar las credenciales de autenticación

para la conexión con AWS IoT, asegurando la seguridad de los datos transmitidos.

- **Generación de archivos estáticos:** Se diseñó un servidor web que permite generar los archivos estáticos necesarios directamente desde el código, sin necesidad de cargarlos físicamente en el dispositivo.
- **Actualización de parámetros y sincronización con la base de datos:** El sistema incluyó una funcionalidad que permite la actualización remota de parámetros y la sincronización con los datos almacenados en la base de datos en casos en los que el dispositivo esté desconectado.

5.2 TRABAJOS FUTUROS

A lo largo del desarrollo del sistema de operación y gestión para la familia de dispositivos IoT de Tablesa S.A., se han identificado oportunidades de mejora y desarrollos adicionales. Los siguientes trabajos futuros se proponen con el fin de seguir optimizando y escalando el sistema, así como de incorporar nuevas funcionalidades:

- Integrar Amazon FreeRTOS para mejorar la gestión de dispositivos IoT a nivel de firmware, proporcionando una plataforma más robusta para el desarrollo de aplicaciones embebidas.
- Implementar AWS Greengrass para permitir la ejecución de funciones de AWS Lambda, sincronización de datos y ejecución de aplicaciones IoT en dispositivos locales, reduciendo la latencia y mejorando la eficiencia operativa.
- Añadir funcionalidad de configuración de parámetros y verificación de mediciones locales mediante el Punto de Acceso. Esto permitirá a los usuarios configurar y verificar dispositivos IoT de manera más directa y eficiente, incluso en entornos sin conectividad a Internet.
- Implementar Amazon Kinesis para el análisis en tiempo real de los datos generados por los dispositivos IoT. Esto proporcionará una capacidad avanzada para la recolección, procesamiento y análisis de grandes flujos de datos, permitiendo una toma de decisiones más rápida y basada en datos.
- Integrar el uso de timers en la adquisición de datos para mejorar la precisión y eficiencia en la recolección de información. Esto permitirá una sincronización más precisa de las mediciones y optimizará el rendimiento de los dispositivos IoT.
- Desarrollar una plataforma de generación de reportes que permita a los usuarios crear y visualizar informes detallados sobre el rendimiento y

el estado de los dispositivos IoT. Esto facilitará el análisis de datos históricos y la identificación de tendencias y patrones en el consumo energético.

- Implementar una solución de automatización para la compilación y almacenamiento de archivos binarios de firmware utilizando GitHub Actions. Esto permitirá eliminar la dependencia de un entorno de desarrollo local, además de reducir la carga de trabajo manual.
- Integrar Amazon Simple Notification Service (SNS) para la automatización de alertas de error y el envío de mensajes de texto o correos electrónicos. Esta funcionalidad permitirá notificaciones proactivas y en tiempo real ante cualquier incidencia, mejorando la capacidad de respuesta y la gestión de errores.
- Implementar AWS IoT Device Defender para mejorar la seguridad del sistema IoT, permitiendo monitorear continuamente el comportamiento de los dispositivos, detectar anomalías y aplicar medidas correctivas automáticas en tiempo real. Esta integración fortalecerá la protección de los dispositivos contra posibles amenazas y vulnerabilidades, asegurando una mayor integridad y confiabilidad del sistema.
- Realizar un análisis detallado de las potenciales vulnerabilidades relacionadas con la API GraphQL y los servicios de AWS utilizados. Aunque se han implementado medidas de seguridad en el sistema, un análisis más exhaustivo permitiría identificar posibles puntos débiles y reforzar la protección del sistema ante amenazas emergentes, mejorando la resiliencia y seguridad de la infraestructura IoT.

La implementación de estas mejoras y expansiones proporcionará una base sólida para futuras innovaciones y desarrollos en la gestión de dispositivos IoT de Tablesa S.A.

6. REFERENCIAS

[1] M. Molęda, B. Małysiak-Mrozek, W. Ding, V. Sunderam, and D. Mrozek, “From Corrective to Predictive Maintenance-A Review of Maintenance Approaches for the Power Industry,” *Sensors* , vol. 23, no. 13, Jun. 2023, doi: 10.3390/s23135970.

[2] C. Sun, R. Xing, Y. Wu, G. Zhou, F. Zheng, and D. Hu, “Design of Over-the-Air Firmware Update and Management for IoT Device with Cloud-based RESTful Web Services,” in *2021 China Automation Congress (CAC)*, IEEE, Oct. 2021, pp. 5081–5085.

[3] J. Sutherland and J. J. Sutherland, *Scrum: The Art of Doing Twice the Work in Half the Time*. Crown Currency, 2014.

[4] M. Marco, *A Guide to the Agile Management Body of Knowledge (ABOK): Scrum Master, PMI-ACP® Exam Prep and Agile Expert Training Program*. 2014.

[5] “Firmware-over-the-Air (FOTA) with LoRa,” LinkLabs. Accessed: Aug. 18, 2023. [Online]. Available: https://www.link-labs.com/blog/firmware-over-the-air-fota-with-lora

[6] “Monitev,” Qué es Monitev. Accessed: Aug. 18, 2023. [Online]. Available: https://www.monitev.com/

[7] “Overview - Unlock your IoT data and accelerate business growth,” AWS IoT. Accessed: Aug. 18, 2023. [Online]. Available: https://aws.amazon.com/es/iot/

[8] S. Späthe, *Conception of a Generic IoT Device Life Cycle Model*. 2021, pp. 1–7.

[9] ISO/IEC, “Internet of Things (IoT) – Reference Architecture,” 30141, 2018.

[10] ISO/IEC, “Information technology — Internet of Things (IoT) — Definition and vocabulary,” 20924, 2021.

[11] B. Moran, H. Tschofenig, D. Brown, and M. Meriac, “A Firmware Update Architecture for Internet of Things,” Internet Engineering Task Force (IETF), RFC 9019, Apr. 2021. [Online]. Available: https://www.rfc-editor.org/rfc/rfc9019.pdf

[12] D. Frisch, S. Reißmann, and C. Pape, “An Over the Air Update Mechanism for ESP8266 Microcontrollers,” in *ICSNC 2017 : The Twelfth International Conference on Systems and Networks Communications*, Oct. 2017.

[13] V. Demertzi, S. Demertzis, and K. Demertzis, “An Overview of Privacy Dimensions on the Industrial Internet of Things (IIoT),” *Algorithms*, vol. 16, no. 8, p. 378, Aug. 2023.

[14] R. Martínez-Peláez *et al.*, “An Enhanced Lightweight IoT-based Authentication Scheme in Cloud Computing Circumstances,” *Sensors* , vol. 19, no. 9, May 2019, doi: 10.3390/s19092098.

[15] L. Zhou, X. Li, K.-H. Yeh, C. Su, and W. Chiu, “Lightweight IoT-based

authentication scheme in cloud computing circumstance," *Future Gener. Comput. Syst.*, vol. 91, pp. 244–251, Feb. 2019.
[16] M. Marjani *et al.*, "Big IoT Data Analytics: Architecture, Opportunities, and Open Research Challenges," *IEEE Access*, vol. 5, pp. 5247–5261, 2017.
[17] S. Deepaisarn, P. Yiwsiw, C. Tantiwattanapaibul, S. Buaruk, and V. Sornlertlamvanich, "Smart Street Light Monitoring and Visualization Platform for Campus Management," in *2022 17th International Joint Symposium on Artificial Intelligence and Natural Language Processing (iSAI-NLP)*, IEEE, Nov. 2022, pp. 1–5.
[18] S. Deepaisarn *et al.*, "Automated Street Light Adjustment System on Campus with AI-Assisted Data Analytics," *Sensors* , vol. 23, no. 4, Feb. 2023, doi: 10.3390/s23041853.
[19] M. Waleed, T. Kamal, T.-W. Um, A. Hafeez, B. Habib, and K. E. Skouby, "Unlocking Insights in IoT-Based Patient Monitoring: Methods for Encompassing Large-Data Challenges," *Sensors* , vol. 23, no. 15, Jul. 2023, doi: 10.3390/s23156760.
[20] R. C. Martin, *Clean Code: A Handbook of Agile Software Craftsmanship*. Pearson Education, 2008.
[21] L. Bass, P. Clements, and R. Kazman, *Software Architecture in Practice*. Addison-Wesley, 2012.
[22] G. Muller, "A reference architecture primer," *Eindhoven Univ. of Techn., Eindhoven, White paper*, pp. 0–20, 2008.
[23] F. Liu, J. Tong, J. Mao, R. Bohn, and J. Messina, *Nist Cloud Computing Reference Architecture: Recommendations of the National Institute of Standards and Technology (Special Publication 500-292)*. CreateSpace, 2012.
[24] S. V. Nath, "IoT ARCHITECTURE," in *Internet of Things and Data Analytics Handbook*, Hoboken, NJ, USA: John Wiley & Sons, Inc., 2016, pp. 239–249.
[25] M. A. Ferrag, L. A. Maglaras, H. Janicke, J. Jiang, and L. Shu, "Authentication Protocols for Internet of Things: A Comprehensive Survey," *Security and Communication Networks*, vol. 2017, Nov. 2017, doi: 10.1155/2017/6562953.
[26] M. R. Palattella *et al.*, "Internet of Things in the 5G Era: Enablers, Architecture, and Business Models," *IEEE J. Sel. Areas Commun.*, vol. 34, no. 3, pp. 510–527, Mar. 2016.
[27] W. Murillo y R. Montoya, "Desarrollo de un sistema multi-plataforma para la adquisición, almacenamiento y gestión de datos usando el internet de las cosas (iot)," Universidad del Valle, 2017.
[28] T. Jaffey, "MQTT and CoAP, IoT Protocols." Accessed: Aug. 18, 2023. [Online]. Available: https://www.eclipse.org/community/eclipse_newsletter/2014/february/arti

cle2.php
[29] A. Kurniawan, *Learning AWS IoT: Effectively manage connected devices on the AWS cloud using services such as AWS Greengrass, AWS button, predictive analytics and machine learning*. Packt Publishing Ltd, 2018.
[30] W. Tärneberg, V. Chandrasekaran, and M. Humphrey, "Experiences creating a framework for smart traffic control using AWS IOT," in *Proceedings of the 9th International Conference on Utility and Cloud Computing*, in UCC '16. New York, NY, USA: Association for Computing Machinery, Dec. 2016, pp. 63–69.
[31] "FreeRTOS Overview - ESP32 - — ESP-IDF Programming Guide v5.2.1 documentation." Accessed: May 28, 2024. [Online]. Available: https://docs.espressif.com/projects/esp-idf/en/stable/esp32/api-reference/system/freertos.html
[32] "Welcome to AWS Documentation," AWS Documentation. Accessed: Feb. 15, 2024. [Online]. Available: https://docs.aws.amazon.com/
[33] "What is Amazon DynamoDB?" Accessed: May 26, 2024. [Online]. Available: https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/Introduction.html
[34] "Amplify Docs - AWS Amplify Gen 2 Documentation." Accessed: May 26, 2024. [Online]. Available: https://docs.amplify.aws/react/
[35] "What is AWS AppSync?" Accessed: May 26, 2024. [Online]. Available: https://docs.aws.amazon.com/appsync/latest/devguide/what-is-appsync.html
[36] "ESP-IDF Programming Guide - ESP32 - — ESP-IDF Programming Guide v5.2.1 documentation." Accessed: May 27, 2024. [Online]. Available: https://docs.espressif.com/projects/esp-idf/en/stable/esp32/index.html

7. ANEXOS

Tabla 7.1
Ficha técnica OneMeter V1.0

Ficha técnica		
Nombre	**Valor**	**Unidad**
Alimentación en Alterna		
Consumo	3	VA
Frecuencia	60	Hz
Características mecánicas y ambientales		
Protección IP	51	IP
Temperatura de almacenamiento	-40...125	Celsius
Temperatura de trabajo	-40...150	Celsius
Peso	0.3	Kg
Volumen	209,560	mm3
Material	ABS	
Fijación	Adhesivo	
Circuito de Medida de Tensión		
Tipo de Medida		Semidirecta
Tensión Nominal	85-265	V
Margen de Tensión de fase	40 a 120	% Tensión Nominal
Tensión mínima de medida	85	V
Circuito de medida de Corriente		
Tipo de Medida		TC
Corriente Nominal	0.2 - 1000	A RMS con TC
Corriente mínima de medida	200	mA
Distorsión armónica	NA	No. Armonicos
Comunicaciones		
Modbus	True	T o F
RS 485	True	T o F

WiFi	True	T o F
Bluetooth	True	T o F
Salidas Digitales de Relé Módulo Externo		
Potencia máxima de conmutación	2400	VA
Tensión Normal	250	VAC
Cantidad de Salidas de relé	4	No. Salidas
Precisión de la medida		
Energía Activa	+1%	% Error
Energía Reactiva	+2%	% Error
Energía Aparente	+1%	% Error
Potencia Activa	+1%	% Error
Potencia Reactiva	+2%	% Error
Potencia Aparente	+1%	% Error
Corriente	0.5%	% Error
Tensión	0.5%	% Error
Frecuencia	0.2%	% Error
Factor de Potencia	1%	% Error
Características de los Terminales		
Terminales de Tensión	Bornera	Conector
Terminales de Corriente	Pug Jack 3.5mm	Conector
Terminales de RS 485	JST	Conector
Terminales de Antena	SMA H	Conector
Compatibilidad Electromagnética		
Inmunidad a Sobretensión	True	T o F
Inmunidad frente a huecos de tensión e interrupciones	True	T o F

Figura 7.1

Primer lote de dispositivos preparados para la etapa de carga de firmware

Figura 7.2

Proceso de carga de firmware a 97 dispositivos mediante un puerto serial usando el programa FlashTools

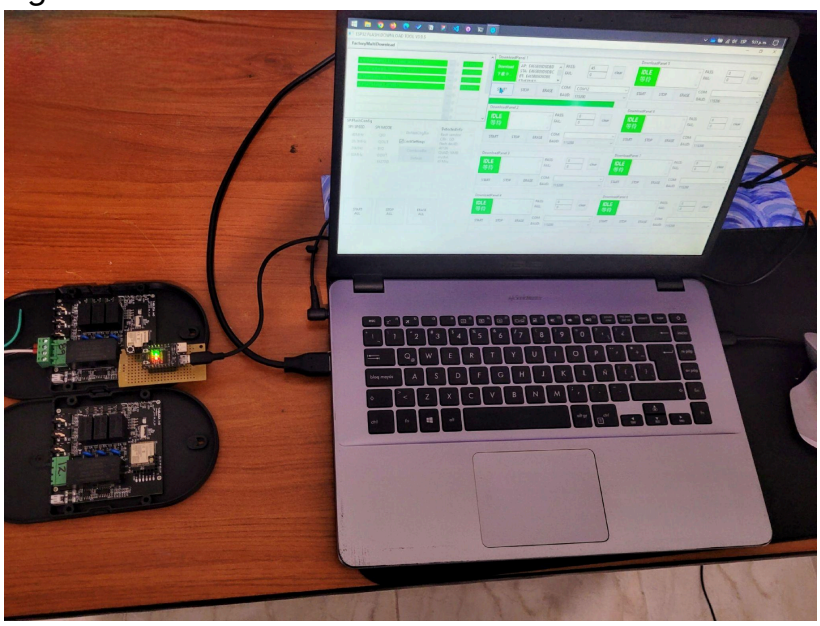


Tabla 7.2

Tiempos en segundos de puesta en marcha de 47 dispositivos

#	Carga de firmware	Configuración WiFi	#	Carga de firmware	Configuración WiFi
01	83.36	168.53	25	82.23	191.57
02	81.60	186.81	26	77.62	185.56
03	91.69	191.32	27	77.37	162.49
04	76.03	170.18	28	82.09	177.53
05	78.99	195.59	29	77.82	187.56
06	81.05	184.55	30	77.61	174.52
07	80.52	193.58	31	79.64	174.52
08	78.85	162.49	32	77.35	166.50
09	77.64	180.54	33	78.68	177.53
10	79.36	164.49	34	91.39	183.55
11	85.59	173.52	35	72.06	180.54
12	77.18	186.56	36	83.87	164.49
13	79.99	172.52	37	80.77	173.52
14	80.13	188.56	38	80.53	179.54
15	79.32	195.59	39	80.70	194.58
16	81.73	176.53	40	79.03	174.52
17	83.82	180.54	41	78.41	190.57
18	82.82	170.51	42	83.28	169.51
19	80.08	174.52	43	80.15	181.54
20	87.83	175.53	44	76.63	167.50
21	86.44	174.52	45	82.62	189.57
22	83.53	170.51	46	83.63	171.51
23	83.71	168.50	47	61.78	194.58
24	79.66	166.50			