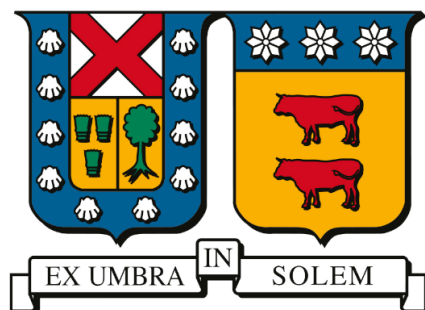


UNIVERSIDAD TÉCNICA FEDERICO SANTA MARÍA
DEPARTAMENTO DE ELECTRÓNICA
VALPARAÍSO - CHILE



**IMPLEMENTACIÓN DE UNA SOLUCIÓN DE OBJETOS CONECTADOS
PARA EL SEGUIMIENTO Y LA GESTIÓN DE ENVASES INDUSTRIALES
REUTILIZABLES**

EMILIO MANUEL CORNEJO GUERRA

TESIS DE GRADO PARA OPTAR POR EL TÍTULO DE INGENIERÍA CIVIL TELEMÁTICA

PROFESOR REFERENTE : SR. MOHAMED ABDELHAMID
PROFESOR CORREFERENTE : SR. NICOLÁS JARA

JULIO 2025



CONSTANCIA DE VALIDACIÓN Y CONFIDENCIALIDAD DE MONOGRAFÍA A REPOSITORIO ACADÉMICO

1.- IDENTIFICACIÓN DEL TRABAJO ACADÉMICO

Tipo de monografía (marcar una opción): ☒ Memoria o trabajo de título; ☐ Tesis de Postgrado;

Título del trabajo: IMPLEMENTACION DE UNA SOLUCION DE OBJETOS CONECTADOS
PARA EL SEGUIMIENTO Y LA GESTION DE ENVASES INDUSTRIALES
REUTILIZABLES

Nombre del candidato(a): Emilio Manuel Cornejo Guerra

Carrera / Grado: Ingenieria Civil Telematica

Campus: Casa Central Valparaiso ; **Departamento:** Electronica

2.- VALIDACIÓN DEL PROFESOR GUÍA/DIRECTOR DE TESIS

Yo, MOHAMED ABDELHAMID, en mi calidad de profesor(a) guía/director(a) del trabajo académico mencionado anteriormente **DEJO CONSTANCIA** que:

- He revisado esta versión del documento y corresponde a la versión final aprobada del trabajo.
- El trabajo cumple con los requisitos académicos y de formato establecidos por la institución

3.- EVALUACIÓN DE CONFIDENCIALIDAD POR PROPIEDAD INDUSTRIAL

☒ El trabajo **NO** contiene información que amerite confidencialidad y puede ser publicado de inmediato en repositorio con acceso abierto.

☐ El trabajo **CONTIENE** información con potenciales implicancias de propiedad industrial o intelectual y requiere un periodo de confidencialidad (embargo) por:

☐ 6 meses; ☐ 12 meses; ☐ 2 años; ☐ 3 años; ☐ 5 años; ☐ 10 años

Fundamentación de la necesidad de confidencialidad (obligatorio si se solicita embargo):

4.- FIRMAS

Profesor(a) guía o director(a) de memoria o tesis:

Fecha: 23-07-2025; **Firma:**

M. Abdelhamid

Estudiante o Candidato(a):

Fecha: 23/07 ; **Firma:**

Emilio C.

Este formulario debe ser insertado como página 2 de la memoria o tesis, completado y firmado por estudiante y profesor(a) antes de la entrega en portal PRISMA de Biblioteca USM.

Agradecimientos

Quiero expresar mi más sincero agradecimiento a mi familia y amigos, cuyo apoyo incondicional y palabras de aliento fueron fundamentales durante todo el proceso de elaboración de esta memoria.

Extiendo mi gratitud a mis colegas y compañeros, por compartir conocimientos, experiencias y por acompañarme en cada desafío durante el desarrollo del proyecto. Su colaboración y profesionalismo han enriquecido enormemente este trabajo.

Agradezco también a la empresa GoodFloow y a los diversos laboratorios que me brindaron la oportunidad de trabajar en este proyecto, permitiéndome aplicar mis conocimientos y adquirir nuevas habilidades en un entorno profesional exigente.

Finalmente, quiero reconocer a la Universidad Técnica Federico Santa María por proporcionarme las bases técnicas necesarias para desarrollar un trabajo de calidad en el extranjero, así como al IMT Atlantique, cuyas especializaciones me dotaron de las herramientas esenciales para trabajar en el ámbito de IoT y plataformas digitales.

A todos, muchas gracias.

Resumen

La presente memoria aborda el diseño e implementación de una solución de objetos conectados para el seguimiento y la gestión de envases industriales reutilizables. El objetivo principal es desarrollar un sistema IoT que permita monitorizar en tiempo real el estado y la ubicación de los embalajes, optimizando la logística, reduciendo costos y minimizando el impacto ambiental. Para ello, se diseñó y desarrolló una nueva versión del dispositivo conectado, que incorpora mejoras en hardware y software, tales como una máquina de estados optimizada, capacidad de actualización Over-The-Air (OTA) y protocolos de comunicación avanzados (NB-IoT, LoRaWAN, MQTT). Paralelamente, se implementó una infraestructura web basada en contenedores mediante Kubernetes en la nube de Scaleway, integrando prácticas DevOps que facilitan la integración y el despliegue continuo (CI/CD). Las pruebas realizadas evidencian una mejora significativa en la robustez, escalabilidad y eficiencia del sistema, permitiendo una gestión óptima de los envases industriales. En conclusión, la solución propuesta cumple con los objetivos de reducción de costos, eficiencia logística y sostenibilidad ambiental, y sienta las bases para futuras innovaciones en el ámbito de la logística inteligente.

Keywords: IoT, envases industriales, DevOps, Kubernetes, sostenibilidad, logística.

Abstract

This thesis addresses the design and implementation of a connected objects solution for tracking and managing reusable industrial containers. The main objective is to develop an IoT system that enables real-time monitoring of the status and location of packaging, optimizing logistics, reducing costs, and minimizing environmental impact. To achieve this, a new version of the connected device was designed and developed, incorporating both hardware and software improvements, such as an optimized state machine, Over-The-Air (OTA) update capability, and advanced communication protocols (NB-IoT, LoRaWAN, MQTT). In parallel, a container-based web infrastructure was implemented using Kubernetes on Scaleway's cloud, integrating DevOps practices that facilitate continuous integration and deployment (CI/CD). The tests conducted demonstrate a significant improvement in the system's robustness, scalability, and efficiency, allowing optimal management of industrial containers. In conclusion, the proposed solution meets the objectives of cost reduction, logistical efficiency, and environmental sustainability, laying the groundwork for future innovations in smart logistics.

Keywords: IoT, industrial containers, DevOps, Kubernetes, sustainability, logistics.

Glosario

- **ADEME (Agencia de Medio Ambiente y de la Gestión de la Energía):** Agencia gubernamental francesa encargada de promover las políticas ambientales y energéticas.
- **AWS (Amazon Web Services):** Plataforma de servicios en la nube provista por Amazon, utilizada para alojar aplicaciones y servicios en línea.
- **Back end:** Parte de una aplicación o sitio web que funciona del lado del servidor. Gestiona la lógica de negocio, las bases de datos y las interacciones con sistemas externos. Es invisible para el usuario final.
- **BLE (Bluetooth Low Energy):** Versión de la tecnología Bluetooth diseñada para aplicaciones que requieren bajo consumo de energía y comunicación inalámbrica de corto alcance.
- **CI/CD (Integración Continua/Despliegue Continuo):** Conjunto de prácticas que permiten probar y desplegar automáticamente aplicaciones de software, asegurando actualizaciones continuas y confiables.
- **Cloud:** Conjunto de servidores remotos utilizados para almacenar, gestionar y procesar datos a través de internet en lugar de en un servidor local o un ordenador personal.
- **DevOps:** Práctica técnica orientada a la unificación del desarrollo de software (Dev) y la administración de infraestructuras informáticas (Ops) para mejorar la eficiencia y la calidad

de las entregas de software.

- **Docker:** Plataforma de software que permite crear, desplegar y ejecutar aplicaciones en contenedores, ofreciendo una virtualización ligera y portátil.
- **ESP32:** Microcontrolador de la empresa Espressif Systems, utilizado para aplicaciones IoT debido a sus capacidades de comunicación inalámbrica y sus múltiples periféricos integrados.
- **ESP-IDF (Espressif IoT Development Framework):** Marco de desarrollo proporcionado por Espressif Systems para programar microcontroladores ESP32.
- **Escalabilidad:** Capacidad de un sistema para adaptarse a un aumento en la carga de trabajo o al crecimiento de sus usuarios sin comprometer su rendimiento.
- **Framework:** Conjunto de herramientas y bibliotecas utilizadas para desarrollar aplicaciones o sistemas.
- **Front end:** Parte visible de una aplicación o sitio web con la que el usuario interactúa directamente. Incluye la interfaz de usuario (UI), la disposición y los elementos visuales.
- **Fullstack:** Desarrollador capaz de trabajar tanto en el front end como en el back end de una aplicación.
- **GNSS (Global Navigation Satellite System):** constelación de satélites que transmite rangos de señales utilizados para el posicionamiento y localización en cualquier parte del globo terrestre.
- **GPS (Global Positioning System):** Sistema que permite a un dispositivo receptor localizar su propia posición sobre la Tierra
- **GSM (Global System for Mobile Communications):** Estándar de telefonía móvil y otras comunicaciones que incorporan tecnología digital.

IA (Inteligencia Artificial): Tecnología que permite a las máquinas aprender y tomar decisiones de forma autónoma. En este proyecto, se utiliza para detectar los tipos de movimientos de los objetos conectados.

■ **INRIA:** Instituto Nacional de Investigación en Ciencias y Tecnologías Digitales en Francia.

■ **IRCICA:** Instituto de Investigación sobre Componentes de Software y Hardware para la Información y la Comunicación Avanzada.

■ **JSON (JavaScript Object Notation):** Formato de datos ligero y legible por el ser humano, utilizado principalmente para intercambiar datos entre un servidor y un cliente web.

■ **Kubernetes (K8s):** Plataforma de código abierto para automatizar el despliegue, la escalabilidad y la gestión de aplicaciones en contenedores.

■ **LoRaWAN:** Protocolo de red de largo alcance y bajo consumo utilizado para comunicaciones inalámbricas entre dispositivos IoT y servidores.

■ **Microservicios:** Arquitectura de software en la que una aplicación se descompone en varios servicios independientes, cada uno responsable de una función específica.

■ **MQTT (Message Queuing Telemetry Transport):** Protocolo de mensajería ligero diseñado para conexiones de baja banda ancha, alta latencia o redes poco fiables. Utilizado para la comunicación entre objetos conectados y servidores.

■ **NB-IoT (Narrowband IoT):** Tecnología de comunicación inalámbrica de bajo consumo y largo alcance, adecuada para dispositivos IoT que requieren intercambios de datos poco frecuentes.

■ **OTA (Over-the-Air):** Mecanismo que permite actualizar el software de un dispositivo de forma remota, sin conexión física.

-
- **Payload:** Conjunto de datos enviados en una transmisión. En el contexto IoT, se refiere a la información transmitida por un dispositivo conectado a un servidor o aplicación.
 - **Pipelines:** Automatizaciones secuenciales en los procesos de CI/CD que incluyen etapas como la integración del código, las pruebas y el despliegue.
 - **RFID (radio frequency identification):** sistema de identificación entre objetos que utiliza radiofrecuencia.
 - **Scaleway:** Proveedor de servicios en la nube francés.
 - **SF (Spreading Factor):** Parámetro en las comunicaciones LoRa que afecta la duración de la transmisión del mensaje y el alcance de la señal.
 - **SSL (Secure Sockets Layer):** Protocolo de seguridad criptográfico utilizado para asegurar las comunicaciones en internet.
 - **Unity:** Marco de pruebas unitarias para el desarrollo de software, utilizado aquí para verificar el correcto funcionamiento del código embebido.
 - **WCDMA (Wideband Code Division Multiple Access):** tecnología digital de transmisión que permite a un número determinado de usuarios acceder a un mismo canal de radiofrecuencia simultáneamente.
 - **YAML (YAML Ain't Markup Language):** Formato de serialización de datos legible por el ser humano, utilizado a menudo para escribir archivos de configuración.

Índice de Contenidos

1. Introducción	1
1.1. Objetivos.....	1
1.2. Antecedentes y motivación de la memoria	2
1.3. Enfoque metodológico.....	2
1.4. Herramientas y tecnologías Utilizadas.....	3
1.5. Alcances de la memoria y proyecciones a futuro.....	4
2. Estado del Arte	6
2.1. Listado de soluciones.....	7
2.1.1. Recommendations for Implementing an IoT based Inventory Tracking and Monitoring System [Rana et al., 2023].....	7
2.1.2. Asset Tracking and Management System [Devi et al., 2023].....	7
2.1.3. Intelligent IoT-Enabled Real-Time Monitoring System for Logistics Management [Joshi et al., 2024].....	8
2.1.4. Development of an IoT-based container tracking system for China’s Belt and Road (B&R) initiative [H. R. Choi et al., 2017].....	9
2.1.5. Development of a Multi-Radio Device for Dry Container Monitoring and Tracking [Falcitelli et al., 2024].....	9
2.2. Discusión	10
3. Desarrollo, test y producción del objeto conectado	13
3.1. Contexto	13
3.2. Documento técnico	14
3.3. Maquina de estados	14
3.3.1. Toma de decisión luego de la respuesta de la inteligencia artificial	17
3.3.2. Ocurrencia simultanea de eventos entre dos o más objetos conectados	19
3.4. Desarrollo embebido.....	21
3.4.1. OTA (Actualización “Over The Air”)	23
3.4.2. Detección de choques en paralelo	24
3.4.3. Test unitarios	26
3.4.4. Desarrollo en antigua versión del sistema embebido	28
3.4.4.1. Registrador de datos para obtener datos para la inteligencia artificial	28
3.4.4.2. Envío periódico de mensaje “LoRaWAN” con diferentes parámetros	30

4. Creación de la infraestructura web y DevOps	32
4.1. Creación de la infraestructura web en Scaleway	33
4.1.1. Kubernetes	35
4.1.2. IoT Hub.....	36
4.2. DevOps.....	38
4.2.1. Ambiente para los test de integración	39
4.2.2. Pipelines CI/CD	40
5. Conclusiones	42
A. Anexo A: Extracto de la descripción del payload	45
B. Anexo B: Instrucciones para desplegar los test de integración de manera local	46

Índice de Figuras

2.1. Diagrama de bloques de sistema IoT propuesto por Devi.....	8
2.2. Sistema propuesto por Joshi	8
2.3. Aparato y características del sistema propuesto por Choi.....	9
2.4. Diagrama de bloque del sistema propuesto por Falcitelli	10
3.1. Antigua maquina de estados.	15
3.2. Nueva maquina de estados.....	17
3.3. Descripcion del mensaje enviado.....	29
3.4. Mensajes recibidos en servidor.....	29
3.5. Porcentaje y cantidad de mensajes perdidos por SF en IoT 08.	30
3.6. Porcentaje y cantidad de mensajes perdidos por SF en IoT 34.	31
4.1. Antigua infraestructura web.....	32
4.2. Nueva infraestructura web	35
4.3. Mensajes recibidos en IoT Hub	38
B.1. Intrucciones en Github.....	46

1 | Introducción

Durante la práctica de fin de estudios en GoodFlow, una startup innovadora francesa especializada en la gestión de embalajes industriales reutilizables, se llevaron a cabo diversas actividades clave para el desarrollo y optimización de soluciones tecnológicas de seguimiento y administración de estos embalajes. Esta empresa tiene como misión reducir la huella de carbono en la cadena logística, ofreciendo un sistema de gestión de embalajes reutilizables que disminuye el costo total de propiedad en comparación con los embalajes desechables de cartón. En consonancia con esta misión, se desarrolla el tema de la memoria titulado *Implementación de una solución de objetos conectados para el seguimiento y la gestión de envases industriales reutilizables*. A continuación, se detallan los principales aspectos de esta memoria, abordando sus objetivos, antecedentes, enfoque metodológico, herramientas empleadas y proyecciones futuras.

1.1. Objetivos

El principal objetivo de esta memoria es desarrollar una solución IoT (Internet de las Cosas) innovadora que permita el seguimiento en tiempo real de los embalajes industriales reutilizables, optimizando su gestión y reduciendo los costos asociados. Los objetivos específicos son:

- Reducción de Costos: Implementar una solución de embalajes reutilizables para reducir los gastos derivados de la compra, almacenamiento y manejo de embalajes desechables.

- **Impacto Ambiental:** Contribuir a la disminución de la huella de carbono en la cadena logística mediante la reducción del uso de embalajes desechables y alinearse con los objetivos de sostenibilidad.
- **Eficiencia Logística:** Mejorar la visibilidad y gestión de inventarios, minimizando pérdidas y aumentando la eficiencia en la cadena de suministro.

1.2. Antecedentes y motivación de la memoria

La industria logística y de embalajes enfrenta desafíos cada vez mayores relacionados con el impacto ambiental y los costos asociados a los embalajes desechables. GoodFloow surge en respuesta a estas necesidades, proponiendo una alternativa sostenible y rentable basada en el uso de embalajes industriales reutilizables. Sin embargo, la gestión de estos embalajes requiere de tecnologías avanzadas para permitir su seguimiento y administración en tiempo real, lo que motiva el desarrollo de una solución IoT que permita integrar dichos procesos de manera eficiente.

La memoria también se beneficia de la colaboración estrecha con el ámbito académico, ya que la startup trabaja en conjunto con instituciones de educación y investigación de renombre como el IMT Atlantique, IMT Nord Europe, INRIA, e IRCICA. Esta colaboración permite que esta memoria incorpore conocimientos avanzados en tecnología IoT, logística y optimización de procesos, garantizando que la solución desarrollada esté respaldada por un enfoque científico y técnico sólido.

1.3. Enfoque metodológico

El objetivo principal de esta memoria es implementar una solución de objeto conectado que permita el seguimiento y la gestión de embalajes industriales reutilizables. Este objetivo abarca dos compo-

nentes fundamentales: el desarrollo y la producción de una nueva versión del dispositivo conectado y la creación de la infraestructura web y procesos DevOps para respaldar esta solución. Las siguientes etapas clave son desarrolladas para poder cumplir con la misión:

- **Desarrollo y Pruebas del Dispositivo IoT:** La primera versión del dispositivo IoT desarrollado en GoodFlow presentó limitaciones en su hardware y software. Se diseña una segunda versión con mejoras en sus componentes y en su funcionalidad. Se actualizan y agregan elementos en el diseño del hardware, mientras que en el software se corrigen errores de programación, se mejora la maquina de estados del sistema y se añaden nuevas funcionalidades.
- **Infraestructura Web y DevOps:** Se implementa una infraestructura web basada en “Kubernetes” [1] a través de “Scaleway” [2], un proveedor de servicios en la nube francesa. La integración de procesos DevOps permite optimizar el flujo de trabajo, asegurando una entrega continua y automatizada (“CI/CD”) de actualizaciones y nuevas funciones.
- **Pruebas y Validación de la Solución:** Cada uno de las etapas previas (dispositivo IoT e infraestructura web) es sometida a rigurosas pruebas y validaciones para asegurar su funcionamiento bajo condiciones reales. Este proceso minimiza el riesgo de fallos en la fase operativa y garantiza que el sistema cumpliera con las expectativas de fiabilidad y eficiencia.

1.4. Herramientas y tecnologías Utilizadas

Para el desarrollo de esta solución, se emplean diversas herramientas y tecnologías que facilitan la implementación y el mantenimiento de los componentes de esta memoria:

- **Kubernetes en Scaleway:** La infraestructura web de Kubernetes permite una mayor escalabilidad y facilidad de gestión de la aplicación web. Al implementar esta tecnología, se logra garantizar el cumplimiento de requisitos de seguridad y eficiencia en el procesamiento de

datos en tiempo real.

- **Tecnologías IoT:** El dispositivo IoT es rediseñado con componentes de hardware actualizados que garantizan robustez y fiabilidad en diversos entornos industriales. Estos dispositivos permiten el seguimiento en tiempo real de la ubicación y estado de los embalajes, recopilando datos que se procesan y almacenan en la infraestructura web.
- **Procesos “DevOps”:** La implementación de “DevOps” permite una entrega continua de nuevas funcionalidades y actualizaciones. Se integran “pipelines” de “CI/CD” que aseguran que cada actualización sea desplegada de forma rápida y segura, minimizando el tiempo de inactividad y aumentando la eficiencia del sistema.

1.5. Alcances de la memoria y proyecciones a futuro

El desarrollo de la segunda versión de la solución de GoodFloow permite ofrecer una alternativa innovadora y sostenible para la gestión de embalajes industriales. Actualmente, el sistema se encuentra en fase de pruebas en una planta de un grupo automotriz importante, y GoodFloow ha sido incubada por Renault en su programa “The Future is Neutral”. Este respaldo evidencia el potencial de la solución para transformar la gestión de embalajes reutilizables, permitiendo una reducción significativa en los costos de propiedad y minimizando el impacto ambiental de los embalajes desechables.

Esta memoria plantea importantes proyecciones para el desarrollo de esta solución como líder un pionero en la industria de embalajes sostenibles. Las próximas fases de esta memoria incluyen:

- **Mejoras en el Dispositivo IoT:** Se prevé continuar optimizando el diseño del dispositivo IoT, integrando nuevas funcionalidades y mejorando su capacidad de recopilación de datos para adaptarse a distintos entornos industriales.

- **Colaboración con Instituciones de Investigación:** GoodFloow seguirá trabajando en conjunto con instituciones académicas y de investigación para aprovechar los avances tecnológicos en áreas como inteligencia artificial y machine learning, con el fin de enriquecer y sofisticar su solución.

2 | Estado del Arte

En el contexto de la transformación digital de los procesos logísticos e industriales, el Internet de las Cosas (IoT) se ha consolidado como una tecnología clave para la monitorización, automatización y optimización de la cadena de suministro. La integración de sensores, microcontroladores y plataformas de comunicación inalámbrica ha permitido el desarrollo de soluciones capaces de rastrear activos en tiempo real y generar alertas ante eventos anómalos.

El estado del arte en esta área muestra una evolución constante hacia sistemas más autónomos, modulares y escalables, que emplean tecnologías como RFID, GPS, BLE Mesh o NB-IoT para asegurar conectividad en diversos entornos. A nivel de software, se destaca el uso de plataformas en la nube para el almacenamiento, análisis y visualización de datos.

Esta sección presenta un análisis comparativo de cinco propuestas recientes, que ilustran distintos enfoques para la implementación de sistemas IoT orientados al seguimiento de contenedores y la gestión de inventarios. Se revisan sus componentes hardware, tecnologías de comunicación y sensores utilizados.

2.1. Listado de soluciones

2.1.1. Recommendations for Implementing an IoT based Inventory Tracking and Monitoring System [Rana et al., 2023]

Esta publicación[3] da recomendaciones para la implementación de un sistema IoT para el rastreo y monitorización de inventarios en almacenes industriales y mayoristas, con visibilidad en tiempo real del estado y ubicación de cada ítem, y gestión automatizada de alertas de reposición. También se propone un sistema que facilita el seguimiento logístico en las empresas, detallando diferentes características que debe tener (baja interacción humana, interfaz amigable para el usuario, etc.) así como los roles de cada usuario del sistema.

2.1.2. Asset Tracking and Management System [Devi et al., 2023]

En este artículo[4] se propone un sistema IoT de rastreo y gestión de activos en tiempo real, pensado para industrias como logística, transporte y gestión de inventarios, que incluye geolocalización y prevención de robos mediante alertas y enlaces de ubicación por SMS . Para ello, utilizan un Arduino Uno como microcontrolador, un módulo GPS y un módulo GSM para el envío de SMS, como se puede observar en la figura 2.1.

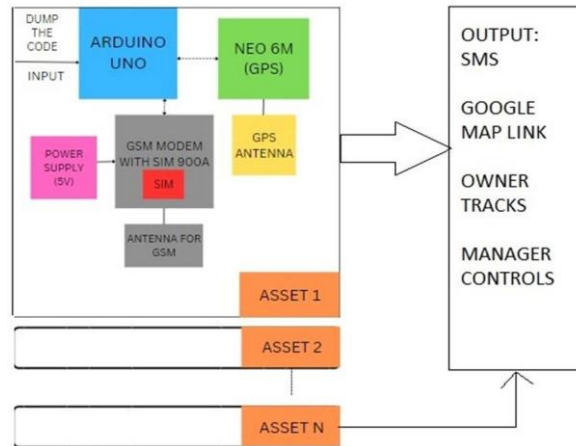


Figura 2.1: Diagrama de bloques de sistema IoT propuesto por Devi

2.1.3. Intelligent IoT-Enabled Real-Time Monitoring System for Logistics Management [Joshi et al., 2024]

En esta publicación[5] se propone un sistema inteligente IoT para monitorizar en tiempo real activos y materiales en gestión logística y de construcción, con generación de alertas y visualización de datos históricos y en vivo . Para ello, se usa un ESP32 como microcontrolador y la tecnología de Tags RFID para monitorizar dichos activos. Su propuesta puede ser vista en la figura 2.2

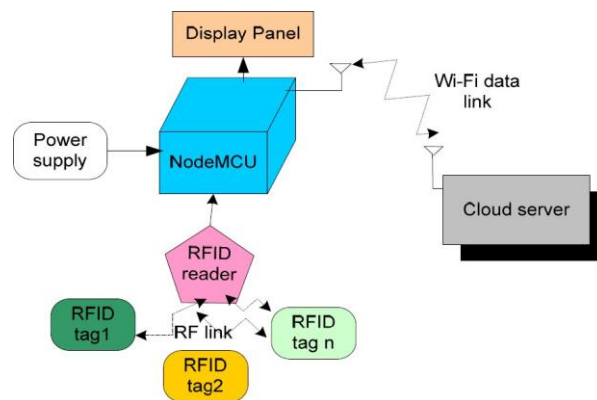


Figura 2.2: Sistema propuesto por Joshi

2.1.4. Development of an IoT-based container tracking system for China's Belt and Road (B&R) initiative [H. R. Choi et al., 2017]

Este artículo[6] propone un sistema IoT para el rastreo y monitorización en tiempo real de contenedores bajo la Iniciativa “Belt & Road” de China, permitiendo visibilidad internacional de la ubicación, el estado interno (temperatura, humedad, vibraciones, apertura de puerta) y soporte de procedimientos aduaneros ágiles. Para ello utilizan diversos módulos, como un módulo GNSS para saber la ubicación del contenedor, un módulo de comunicación WCDMA y GSM para el envío de información hacia un servidor web y diversos sensores como un acelerómetro o un sensor de temperatura y humedad.

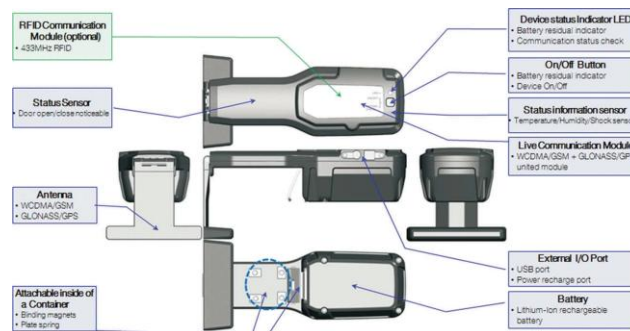


Figura 2.3: Aparato y características del sistema propuesto por Choi

2.1.5. Development of a Multi-Radio Device for Dry Container Monitoring and Tracking [Falcitelli et al., 2024]

Este artículo[7] presenta un dispositivo de seguimiento y monitorización en tiempo real de contenedores secos (dry containers) en entornos marítimos y terrestres, integrando múltiples radios (NB-IoT, BLE Mesh y GNSS) para asegurar cobertura continua, incluyendo backhaul satelital en mar abierto como parte del “5G Global Tracking System”. Esto lo logran con un microcontrolador ESP32 quien, bajo un sistema operativo basado en FreeRTOS, es capaz de controlar los diversos módulos y sensores que tiene incorporado.

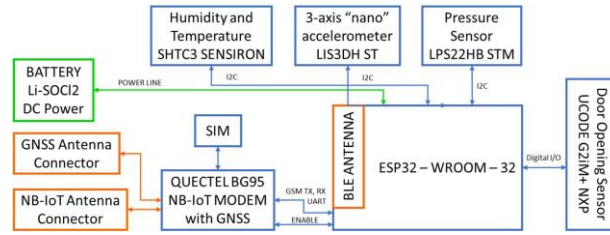


Figura 2.4: Diagrama de bloque del sistema propuesto por Falcitelli

Ref	Aplicación	Hardware	Sensores	Comunicación
Devi et al., 2023	Rastreo de activos vía GSM/GPS	Arduino Uno, SIM9000A, GPS NEO-6M	GPS	GSM (SMS)
Joshi et al., 2024	Gestión de activos con RFID y visualización móvil	ESP8266, lector RFID	RFID	Wi-Fi
H. R. Choi et al., 2017	Rastreo de contenedores internacionales con sensores internos	GNSS, GSM/WCDMA, sensores integrados, batería de 13,000 mAh	GNSS, temperatura, humedad, vibración, apertura de puerta, batería	GSM
Falcitelli et al., 2024	Seguimiento de contenedores con NB-IoT + BLE Mesh + GNSS	ESP32, Quectel BG95, sensores múltiples	GNSS, temperatura, humedad, presión, acelerómetro	NB-IoT + BLE Mesh + backhaul satelital

Tabla 2.1: Tabla comparativa de las distintas soluciones estudiadas

2.2. Discusión

En las diferentes soluciones encontradas, es posible ver que existen diferentes maneras de lograr el objetivo de monitorear, automatizar y optimizar la cadena de suministros. Cada una cuenta con sus pros y contras que se deben considerar al momento de pensar en la solución más óptima para el problema presentado.

Como se sugiere en el artículo de recomendaciones para implementar un sistema de monitoreo basado en IoT [Rana et al., 2023], se debe priorizar la no intervención humana para la gestión del

inventario, esta idea se plasma también en esta memoria, donde cada objeto, desde que comienza su funcionamiento, es independiente y no requiere ninguna intervención humana directa para que siga funcionando y siga enviando información relevante al servidor respectivo.

Por otro lado, como se propone en el trabajo de un sistema de rastreo y gestión de activos en tiempo real [Devi et al., 2023], se utiliza un módulo GPS para localizar el dispositivo en todo momento, como también un módulo GSM para el envío de datos mediante SMS. De esta solución, contiene bastantes limitantes para nosotros, ya que el uso de un módulo GPS requiere un consumo energético considerable para poder saber en tiempo real la ubicación de los dispositivos, mientras que nuestro interés no es saber la posición exacta en cada momento, sino que saber quién es la persona o entidad responsable de los dispositivos durante su trayecto de una empresa a otra, y en el caso de que se requiera saber, existen otros métodos más eficientes energéticamente para saber la posición del dispositivo, como lo es la localización vía direcciones MAC de routers WiFi, o la localización utilizando la información de conexión celular, siendo esta última la menos precisa. Por otro lado, el uso de un módulo GSM puede ser útil para el envío de la información, pero existen otras tecnologías que permiten mayor cobertura y menor costo energético, como lo es NB-IoT.

En otro de los sistemas de monitoreo [Joshi et al., 2024], se propone utilizar la tecnología de Tags RFID, si bien esta idea es muy interesante, nos demandaría incorporar un lector RFID en cada empresa donde los envases industriales pasarán, lo cual requiere una instalación de infraestructura dedicada, esto es genial cuando sabemos, a priori, todas las empresas por las que los envases pasarán, pero para nosotros es complicado realizar esto, ya que no sabemos realmente por cuáles empresas podrán pasar, sino que nuestra propuesta es que sin una infraestructura dedicada, solo con un dispositivo conectado a cada envase, podamos realizar el monitoreo correspondiente.

El sistema propuesto bajo la iniciativa “Belt & Road” [H. R. Choi et al., 2017] empieza a parecerse a los que se busca en esta memoria, con un objeto que se añade en un contenedor, que cuenta con una batería que le permite enviar datos por 50 días sin necesidad de recargarla y con distintos sensores que nos pueden dar diferentes informaciones como la detección de choques o apertura del

contenedor. Pero para esta memoria no se busca monitorear un gran contenedor, sino que los envases industriales que serán desplazados en un contenedor de un punto a otro, como también es que con una sola batería se pueda lograr una autonomía de al menos 5 años, lo cual es mucho más que los 50 días obtenidos por este sistema.

Por último, el dispositivo multi-radio desarrollado por [Falcitelli et al., 2024] se acerca mucho más a lo que se tiene en mente realizar en esta memoria, ya que si bien solo está centrado en grandes contenedores y especialmente focalizado en transporte marítimo, este utiliza comunicación BLE para comunicar entre objetos conectados y NB-IoT para poder enviar información al servidor. Esto es muy útil ya que permite que todos los objetos comuniquen con un solo "líder" que sea este quien envíe la información de todo el grupo hacia el servidor, reduciendo así el consumo energético global del sistema.

3 | Desarrollo, test y producción del objeto conectado

3.1. Contexto

En la fase de desarrollo del proyecto GoodFloow, se diseñó una primera versión del objeto conectado autónomo destinado a realizar el seguimiento de embalajes industriales reutilizables. Este dispositivo fue sometido a pruebas en las instalaciones de un cliente en septiembre de 2023, lo que permitió identificar varias áreas de mejora tanto en el diseño físico del objeto como en el software embebido que opera en su microcontrolador.

Tras una jornada extensa de reuniones, que incluyó a investigadores así como al equipo técnico involucrado en el proyecto, se tomó la decisión de desarrollar una nueva versión del objeto conectado. Esta nueva iteración incorpora componentes actualizados y una lógica renovada, diseñada para abordar los problemas detectados previamente y garantizar un rendimiento más robusto y eficiente.

3.2. Documento técnico

Como parte del inicio del proyecto, se elabora un documento técnico que detalla las funcionalidades del nuevo objeto conectado, sus especificaciones técnicas y las etapas necesarias para garantizar el desarrollo exitoso del proyecto. Esta nueva versión del objeto, que incluye actualizaciones y mejoras tanto en el hardware como en el software, requiere la colaboración activa y la validación del equipo de investigadores e ingenieros dedicados a la concepción del hardware del objeto conectado. Este equipo asume la responsabilidad de ensamblar todos los componentes electrónicos, así como el microcontrolador que integrará la lógica del programa.

3.3. Máquina de estados

En la primera versión del objeto conectado, se había implementado una máquina de estados sencilla que modela su funcionamiento. Esta máquina consta de cuatro estados principales:

- **Rest unknown:** Estado en el que el objeto permanece inmóvil en una fábrica o en otro lugar.
- **Rest truck:** Estado en el que el objeto permanece inmóvil dentro de un camión.
- **Fenwick movement:** Estado encargado de realizar acciones cuando el objeto es manipulado por un montacargas.
- **Truck movement:** Estado en el que el objeto se encuentra dentro de un camión en movimiento.

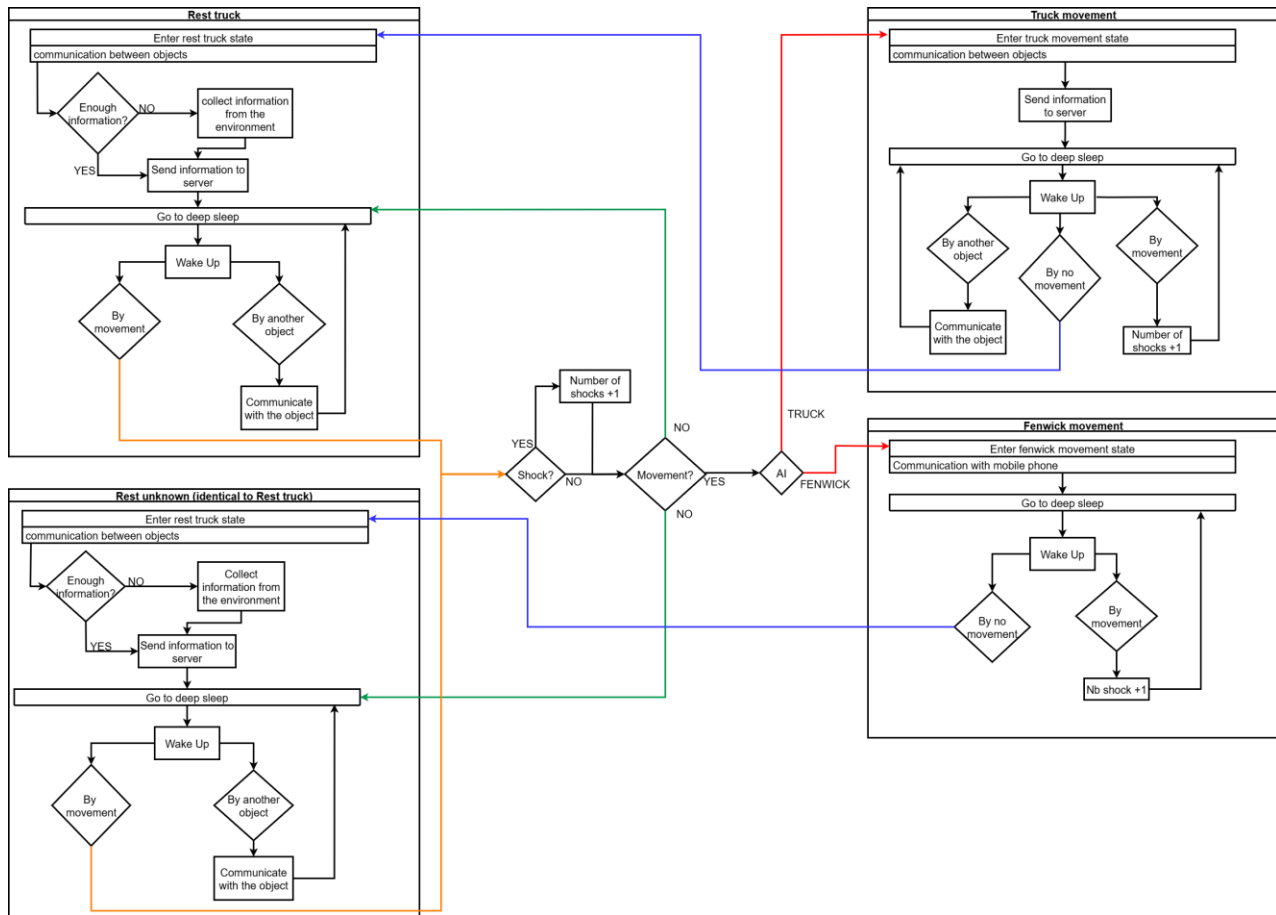


Figura 3.1: Antigua maquina de estados.

Sin embargo, esta versión inicial presenta algunas limitaciones. Por ejemplo, las acciones realizadas en los estados rest truck y rest unknown son idénticas, lo que hace redundante la existencia de ambos estados. Además, la dependencia absoluta de los resultados proporcionados por la IA para determinar las transiciones entre estados representa un problema, ya que dichos resultados no son 100 % confiables. Esto genera errores que pueden afectar el correcto funcionamiento del objeto conectado. Por último, al operar en grupos, los objetos pueden recibir simultáneamente los efectos de un nuevo estado, lo que causa conflictos en la comunicación entre ellos.

Para abordar estas limitaciones, se diseña una nueva máquina de estados que incorpora siete estados:

- **Rest:** Combina los estados rest unknown y rest truck, ya que sus funcionalidades eran

equivalentes.

- **Truck movement action:** Estado que se activa al inicio del movimiento del camión, donde se ejecutan acciones específicas.
- **Truck movement sleep:** Estado en el que el objeto entra en modo de bajo consumo, despertándose solo por interrupciones externas, mientras el camión continúa en movimiento por un periodo prolongado.
- **From truck to rest:** Estado de transición que se utiliza cuando el camión se detiene y el objeto debe pasar al estado de reposo (Rest).
- **Fenwick movement:** Estado encargado de realizar acciones cuando el objeto es manipulado por un montacargas.
- **From fenwick to rest:** Estado de transición utilizado cuando el montacargas se detiene y el objeto debe volver al estado de reposo (Rest).
- **Awaiting decision:** Estado en el que la IA no ha podido determinar el tipo de movimiento detectado; por lo tanto, el objeto espera datos adicionales para decidir el estado al que debe transitar.

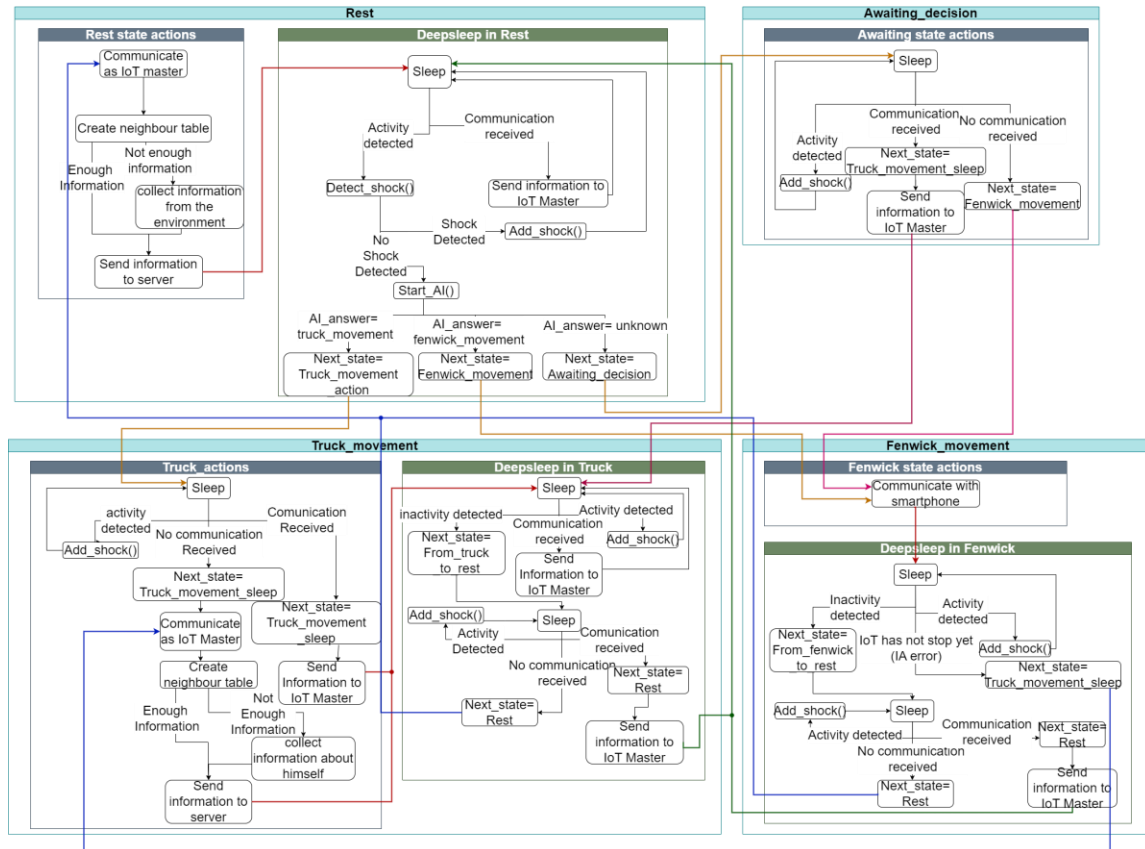


Figura 3.2: Nueva maquina de estados.

Estas mejoras permiten corregir varios errores, fortaleciendo la robustez del sistema. La eliminación de redundancias entre estados simplifica la lógica de funcionamiento, mientras que la introducción de estados transitorios mejora la precisión en las transiciones. Finalmente, se optimiza la gestión de grupos de objetos, evitando conflictos en la comunicación entre ellos. Esta nueva lógica se describirá detalladamente en las siguientes secciones, mostrando cómo resuelve las problemáticas identificadas en la primera versión.

3.3.1. Toma de decisión luego de la respuesta de la inteligencia artificial

Nuestro objeto conectado cuenta con una inteligencia artificial integrada diseñada para detectar el tipo de movimiento percibido. La IA proporciona tres resultados posibles:

- **Camión:** Cuando predice que el movimiento corresponde a un camión.
- **Montacargas:** Cuando predice que el movimiento corresponde a un montacargas.
- **Desconocido:** Cuando no puede determinar el tipo de movimiento.

En función del resultado proporcionado por la IA, el objeto conectado toma decisiones para cambiar de un estado a otro. Sin embargo, esta aproximación presenta un problema inherente: la IA nunca será completamente precisa en sus predicciones. Aunque puede alcanzar un alto grado de exactitud, siempre existirá un margen de error que puede llevar al objeto a tomar decisiones equivocadas, afectando su funcionamiento correcto.

La nueva máquina de estados se diseñó para abordar estos problemas, corrigiendo las predicciones erróneas de la IA y garantizando que el objeto conectado pueda manejar dichas situaciones sin comprometer su misión principal. A continuación, se describen los casos en los que la IA puede fallar o no ser capaz de predecir el movimiento, junto con las soluciones implementadas en la máquina de estados:

- **Caso 1: Movimiento de camión mal identificado como movimiento de montacargas** En el contexto de una fábrica, los movimientos asociados con el uso de un montacargas suelen durar solo unos minutos. Si la IA identifica erróneamente un movimiento de camión como uno de montacargas, el objeto pasará al estado “Fenwick movement” y realizará las acciones correspondientes. Si, tras unos minutos, el objeto no detecta una detención, entenderá que la predicción fue incorrecta y cambiará al estado “Truck movement sleep”, corrigiendo automáticamente el error de la IA.
- **Caso 2: Movimiento de camión identificado como movimiento desconocido** Cuando un objeto detecta un movimiento desconocido, pero en realidad está dentro de un camión, entra en juego el sistema de comunicación grupal descrito en la sección 3.3.2. En un camión, varios objetos conectados colaboran, y uno de ellos tiene la tarea de recopilar información y

enviarla al servidor. Si el movimiento desconocido ocurre, el objeto pasará al estado “Awaiting decision” y esperará instrucciones del objeto encargado de recopilar datos. Una vez recibida la instrucción, cambiará al estado “Truck movement sleep”, corrigiendo automáticamente el error.

- **Caso 3: Movimiento de montacargas identificado como movimiento de camión** En este caso, la máquina de estados no corrige activamente el error porque el riesgo de pérdida de información es mínimo. Si el objeto pasa al estado “Truck movement action” en lugar de “Fenwick movement”, el impacto es limitado, ya que los movimientos en “Fenwick movement” suelen ser breves. Una vez que el objeto detecte una detención y pase al estado “Rest”, el error será corregido automáticamente en el servidor que recolecta la información de los objetos.
- **Caso 4: Movimiento de montacargas identificado como movimiento desconocido** Siguiendo la lógica del caso 2, cuando el objeto está en “Awaiting decision” y no recibe instrucciones, deducirá que se encuentra en un movimiento de montacargas. En consecuencia, cambiará al estado “Fenwick movement”, corrigiendo así el error de la IA.

Estas soluciones permiten mitigar las limitaciones de la IA integrada, reforzando la robustez de la solución y garantizando que el objeto conectado cumpla su propósito principal incluso en escenarios donde las predicciones no son completamente precisas.

3.3.2. Ocurrencia simultanea de eventos entre dos o más objetos conectados

En el funcionamiento normal de la solución, los embalajes que contienen objetos suelen desplazarse en grupo dentro de un camión de una fábrica a otra. Esto significa que todos los objetos del grupo experimentan los mismos movimientos y, por tanto, realizan las mismas acciones. Este comportamiento ya estaba presente en la primera máquina de estados. Sin embargo, se identificó un

problema:

Cuando el camión comienza a moverse o se detiene, todos los objetos en el grupo intentan asumir el rol de identificar el entorno y actuar como responsables de la recopilación de datos. Como resultado, ninguno envía información relevante al servidor, ya que todos permanecen en estado de escucha sin emitir. Esto lleva a un consumo innecesario de energía y va en contra de los objetivos de eficiencia del proyecto Goodflow. Un problema similar ocurre cuando varios objetos son descargados al mismo tiempo por un montacargas.

■ Solución en la nueva máquina de estados

Para abordar este problema, la nueva máquina de estados implementa un mecanismo que asigna dinámicamente un único objeto responsable dentro del grupo. Este objeto será el encargado de:

- Enviar las instrucciones al resto de los objetos del grupo.
- Recopilar las respuestas y datos del entorno enviados por los demás.
- Transmitir la información recolectada al servidor.

■ Proceso de elección del objeto responsable

La elección del objeto responsable se realiza sin necesidad de consenso previo ni comunicación directa entre los objetos. Cada objeto genera internamente un temporizador aleatorio. El objeto cuyo temporizador llega a cero primero asume la responsabilidad de enviar instrucciones y recopilar datos.

Este enfoque presenta las siguientes ventajas:

- **Reducción del consumo energético:** Solo un objeto realiza las tareas que requieren más energía, como la recopilación y el envío de datos al servidor.
- **Eficiencia operativa:** Se evita que todos los objetos actúen simultáneamente, optimizan-

do los recursos del grupo.

■ Gestión de conflictos

En los casos en que dos o más objetos tienen temporizadores que llegan a cero casi simultáneamente, es posible que varios objetos asuman el rol de responsables al mismo tiempo. Esto genera un leve aumento en el consumo energético del grupo, pero no se pierde información relevante.

Además, esta situación se corrige posteriormente en el servidor backend, que puede identificar y consolidar los datos duplicados enviados por los objetos responsables redundantes.

Este enfoque asegura que el grupo funcione de manera más eficiente, respetando los objetivos de ahorro energético y fiabilidad del proyecto Goodflow.

3.4. Desarrollo embebido

Una vez finalizada la etapa de diseño del objeto conectado, incluyendo la definición de su documento técnico y la implementación de su nueva máquina de estados, se procede al desarrollo del código que ejecutará el microcontrolador.

■ Microcontrolador y framework

El microcontrolador elegido es el ESP32S3 de Espressif Systems, un dispositivo ampliamente utilizado en aplicaciones IoT gracias a su versatilidad y capacidad de integrar múltiples periféricos. Para el desarrollo del firmware, se emplea el Framework ESP-IDF (Espressif IoT Development Framework) [8], el entorno oficial proporcionado por Espressif para sus dispositivos.

Este microcontrolador incluye características clave para el proyecto:

- **Compatibilidad con periféricos IoT:** Necesarios para integrar componentes externos como sensores, módulos de comunicación por radio y de larga distancia.
- **Modos de bajo consumo energético:** Permiten mantener el objeto en estado de reposo durante la mayor parte del tiempo, optimizando el consumo de energía, una prioridad en este sistema.

■ **Reutilización y adaptación del código existente**

A partir de la primera versión del objeto conectado, se identifican componentes que funcionan correctamente y se conservan con ligeras adaptaciones para este nuevo microcontrolador, tales como:

1. Lectura de datos de sensores.
2. Envío y lectura de datos via “Bluetooth Low Energy(BLE)”.
3. Captura de direcciones MAC de puntos de acceso WiFi.

Estas bibliotecas solo requirieron modificaciones mínimas, por lo que no se consideran cambios significativos que deban detallarse.

■ **Implementación de la nueva máquina de estados**

Con la nueva lógica de la máquina de estados definida (como se describe en la sección 3.3), es necesario desarrollar su correspondiente implementación en código. Este desarrollo se realizó en lenguaje C/C++, siguiendo los siguientes principios:

- **Funciones cuidadosamente estructuradas:** Cada estado y transición se programó para reflejar fielmente la lógica descrita en el diseño.
- **Pruebas exhaustivas:** Cada avance en el desarrollo fue sometido a pruebas rigurosas para evitar errores de programación y garantizar un uso eficiente de la memoria del ESP32.

■ Actualización del contenido del payload

En esta nueva versión, también se realizan cambios en el contenido del “payload” enviado al servidor. Ahora, la información transmitida incluye más detalles sobre el objeto conectado, mejorando la capacidad del servidor para gestionar y analizar los datos recolectados.

Con estas mejoras, el código no solo corrige problemas detectados en la primera versión, sino que también optimiza la funcionalidad y eficiencia del sistema, alineándose con los objetivos del proyecto.

3.4.1. OTA (Actualización “Over The Air”)

Una de las nuevas funcionalidades más importantes de esta versión del objeto conectado es la incorporación de la capacidad para realizar actualizaciones OTA (Over-The-Air), eliminando la necesidad de conectar el dispositivo a un ordenador u otra máquina para actualizar su firmware.

■ ¿Qué es una actualización OTA?

La actualización OTA es un mecanismo que permite actualizar el código del microcontrolador ESP32 de manera remota, mientras el dispositivo permanece en funcionamiento. Este proceso es posible gracias a la conectividad WiFi y Bluetooth Low Energy (BLE), tecnologías soportadas por el ESP32S3. Espressif Systems proporciona en su documentación ejemplos detallados sobre cómo implementar esta funcionalidad, lo que facilita su integración en proyectos.

■ Implementación en el proyecto

En esta nueva versión del objeto conectado, se desarrolla e integra la capacidad de realizar OTA mediante WiFi y/o BLE. Este proceso incluye los siguientes elementos clave:

1. **Interruptor magnético “reed switch”:** Este componente genera una interrupción en el funcionamiento normal del dispositivo lo que le permite al microcontrolador pasar automáticamente a un modo de programación, habilitando la descarga de la última

versión del “firmware”.

2. Conectividad dual para OTA:

- **WiFi:** Ideal para entornos donde existe una red inalámbrica disponible.
- **BLE:** Permite realizar actualizaciones incluso en entornos donde el acceso a WiFi es limitado.

■ Ventajas de la funcionalidad OTA

- **Flexibilidad:** Los dispositivos pueden actualizarse en cualquier momento y lugar sin intervención física.
- **Optimización del mantenimiento:** Facilita la distribución de actualizaciones para corregir errores o agregar nuevas funcionalidades, reduciendo costos operativos.
- **Compatibilidad con IoT:** Mejora la escalabilidad del sistema al permitir la gestión remota de múltiples dispositivos.

Gracias a esta funcionalidad, el proyecto logra una mayor autonomía y adaptabilidad, alineándose con las necesidades actuales de los sistemas IoT modernos.

3.4.2. Detección de choques en paralelo

Una de las funcionalidades clave de nuestro objeto conectado es la capacidad de detectar accidentes o movimientos bruscos que puedan dañar el embalaje. Esta detección es fundamental para garantizar la integridad del objeto durante su uso y transporte.

■ Problema identificado:

Normalmente, el microcontrolador se encuentra en modo de bajo consumo (dormido), y el sensor está configurado para despertar al microcontrolador cuando detecta un accidente. Sin embargo, el problema surge una vez que el microcontrolador se activa, ya que, al ejecutar una tarea que consume tiempo, pierde temporalmente la capacidad de detectar nuevos movimientos bruscos. Esto ocurre debido a la ejecución secuencial de las tareas programadas.

■ **Solución: Uso del ESP32S3 y su doble núcleo**

El ESP32 fue elegido específicamente por su arquitectura de doble núcleo, que permite ejecutar dos tareas en paralelo. Para abordar este problema, se implementa una solución basada en las siguientes características:

1. **Lectura periódica del sensor:** Se programa una función dedicada que monitorea de forma continua el estado del sensor responsable de detectar movimientos bruscos. Esta función se activa automáticamente cada vez que el microcontrolador despierta.

2. **Distribución de tareas entre los núcleos:**

- **Primer núcleo:** Ejecuta de manera secuencial las tareas principales del objeto, como:
 - Comunicación entre objetos.
 - Comunicación con el servidor.
 - Recolección de datos.
- **Segundo núcleo:** Ejecuta la función de monitoreo del sensor, garantizando la detección de movimientos bruscos en tiempo real.

■ **Ventajas de esta implementación**

- **Detección continua:** El microcontrolador puede identificar accidentes incluso mientras realiza otras tareas importantes.
- **Eficiencia energética:** La función de monitoreo solo se activa cuando el microcontrolador está despierto, optimizando el consumo de energía.
- **Fiabilidad:** Minimiza el riesgo de perder eventos críticos que puedan indicar daños en el embalaje.

Esta solución demuestra cómo la arquitectura avanzada del ESP32 y una programación estratégica permiten superar limitaciones técnicas, optimizando el rendimiento y la funcionalidad del sistema IoT.

3.4.3. Test unitarios

Una de las etapas fundamentales en el desarrollo de software es la capacidad de probar y validar el código para garantizar su correcto funcionamiento y minimizar riesgos. En este proyecto, se implementan diversas estrategias para evaluar las funcionalidades desarrolladas.

■ Pruebas unitarias

Durante la fase de desarrollo, se realizan pruebas unitarias para verificar el comportamiento de las unidades individuales del programa. Esto incluye:

- **Validación de funcionalidad:** Asegurar que cada función ejecuta correctamente la tarea para la cual fue diseñada.
- **Revisión de parámetros:** Comprobar que los nombres, tipos de parámetros y valores retornados son consistentes con las especificaciones.
- **Estados iniciales y finales:** Verificar que el estado inicial de las funciones sea válido y

que las salidas finales correspondan a lo esperado.

■ **Framework utilizado: Unity**

Para estas pruebas se utiliza el framework “Unity”, integrado dentro del entorno de programación ESP-IDF. Este framework permite ejecutar los tests directamente en el microcontrolador ESP32, lo cual garantiza que los resultados sean consistentes con el hardware final.

■ **Resultados obtenidos**

Se programan y ejecutan pruebas unitarias en los componentes clave del sistema, lo que permite:

1. **Validar funcionalidad independiente:** Asegurar que cada componente funciona correctamente de manera aislada.
2. **Identificar errores y oportunidades de mejora:** Detectar “bugs” y optimizar diversas funciones antes de integrarlas completamente al sistema.

■ **Ventajas de esta metodología**

- **Prevención de regresiones:** Cada nueva funcionalidad se validó sin comprometer el funcionamiento general del código.
- **Eficiencia en el desarrollo:** Las pruebas unitarias ayudaron a identificar y corregir errores de forma temprana, reduciendo costos asociados a fallos en etapas posteriores.
- **Adaptabilidad al hardware:** La ejecución de los tests directamente en el ESP32 aseguró que el código sea robusto y eficiente en el entorno real.

Las pruebas unitarias contribuyen significativamente a la fiabilidad del software, garantizando que las funciones programadas cumplen con los requisitos establecidos y optimizando el desempeño global del sistema.

3.4.4. Desarrollo en antigua versión del sistema embebido

Aunque se trabaja principalmente en la nueva versión del objeto conectado y su código integrado, también se tiene crear nuevas funcionalidades para la versión antigua del objeto conectado, porque se tiene un cliente que dispone de un pequeño número de estos objetos para realizar diversas pruebas y ensayos. Estas funcionalidades son las siguientes:

3.4.4.1. Registrador de datos para obtener datos para la inteligencia artificial

Como se indicó anteriormente, el objeto conectado cuenta con una inteligencia artificial integrada, cuyo objetivo es identificar el tipo de movimientos recibidos para tomar una decisión sobre su próximo estado. Toda inteligencia artificial requiere un modelo y datos para entrenarlo. La cantidad de datos disponible resultó suficiente para entrenar el modelo, pero no para garantizar su precisión; siempre se ha recomendado contar con un volumen de datos más amplio y variado. Por ello, se creó una campaña de recolección de datos, combinando enfoques supervisados y no supervisados, con el objetivo de recopilar información que fortalezca el modelo de inteligencia artificial.

Para la recolección de datos no supervisada, el código programado sigue la lógica siguiente: si el objeto conectado permanece inmóvil, no realiza ninguna acción y se mantiene en estado de reposo; si detecta un movimiento, captura datos del sensor a una frecuencia de 100 Hz y los almacena en una tarjeta SD integrada en el objeto. Además, cada 6 horas, el objeto envía un mensaje a través de LoRaWAN con información sobre el nivel de batería disponible, la marca temporal (“timestamp”) interna del microcontrolador, la cantidad de datos recolectados y el tamaño del archivo CSV que contiene los datos del acelerómetro.

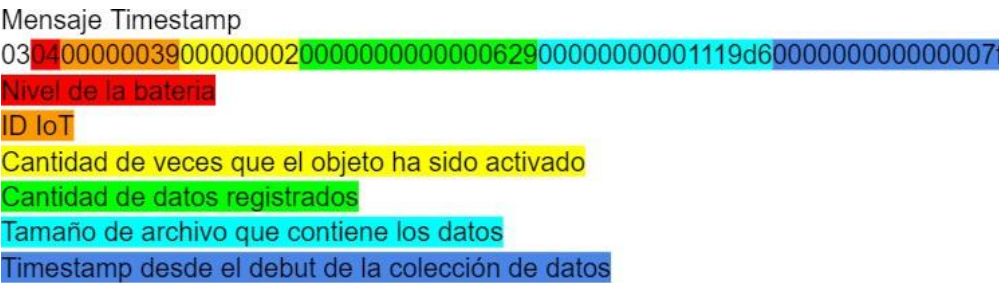


Figura 3.3: Descripcion del mensaje enviado.

Date	Data	Missing Fcnt	Port	Network signal
07/10/2024 2:26:12 PM	030200000028000000947000000000009865f0000000000e0c562000000000774aa0	0	1	
07/10/2024 8:20:04 AM	030200000028000000947000000000009865f0000000000e0c56200000000078f48a	0	1	
07/10/2024 2:14:10 AM	030200000028000000947000000000009865f0000000000e0c562000000000769e13	0	1	

Figura 3.4: Mensajes recibidos en servidor.

La figura 3.4 ilustra tres mensajes enviados mediante LoRaWAN por uno de los objetos conectados, incluyendo las informaciones mencionadas. Este código se implementa en cinco dispositivos, instalados en diferentes embalajes que realizan trayectos entre dos fábricas. Tras dos meses y medio, se logra recolectar un poco más de dos millones de datos entre todos los dispositivos, lo que equivale aproximadamente a 5 horas y 30 minutos de movimientos no caracterizados, útiles para fortalecer el modelo de inteligencia artificial. Luego estos datos fueron recuperados y enviados a los investigadores que trabajan en el desarrollo de la IA

Por otro lado, también se desarrolla el código para la recolección supervisada de datos, que sigue una lógica similar al código anterior. Sin embargo, en este caso, cuando el objeto se detiene, envía un mensaje con la información interna, permitiendo asociar los datos a movimientos específicos. Este código no se implementó en un objeto, pero esta listo para poder utilizarlo y recoger datos de manera supervisada.

3.4.4.2. Envío periódico de mensaje “LoRaWAN” con diferentes parámetros

Uno de los principales problemas encontrados en la primera versión del objeto conectado fue el envío de datos mediante LoRaWAN, ya que en ciertos lugares la conexión resulta deficiente o inexistente, lo que imposibilita la recepción de datos relevantes por parte del objeto. Por ello, se decide realizar un estudio en la fábrica del cliente automotriz, donde se envían mensajes periódicamente a través de LoRaWAN, modificando factores de envío como el Spreading Factor (SF).

Para esta evaluación, se desarrolla un código que envía un mensaje cada hora, utilizando un Spreading Factor diferente en cada envío (desde SF7 hasta SF12). Este código se implementa en dos dispositivos colocados de forma estática dentro de la fábrica durante un mes, obteniendo los resultados siguientes:

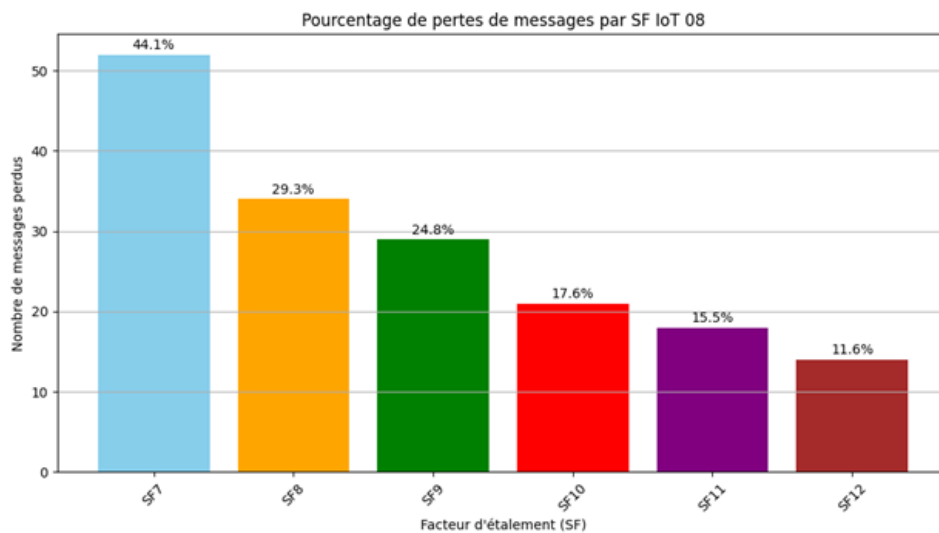


Figura 3.5: Porcentaje y cantidad de mensajes perdidos por SF en IoT 08.

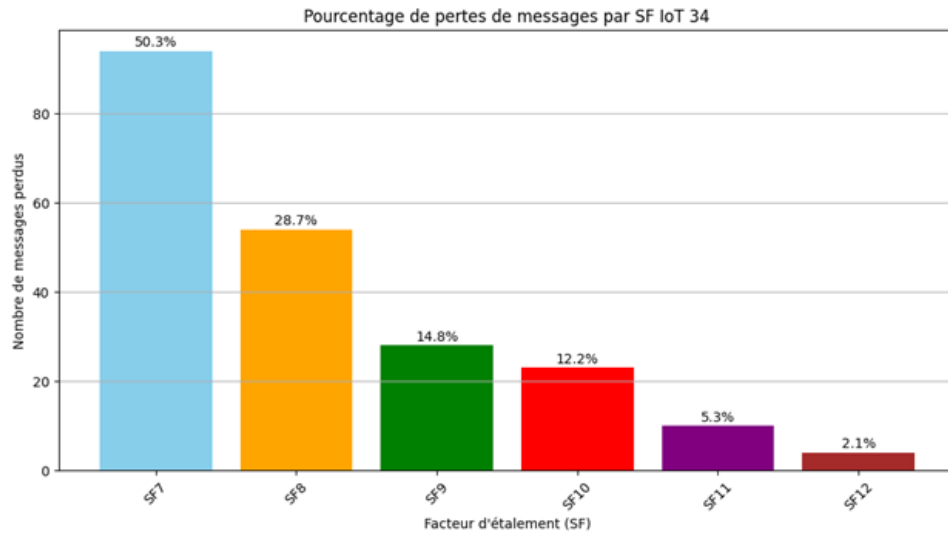


Figura 3.6: Porcentaje y cantidad de mensajes perdidos por SF en IoT 34.

Los datos muestran que, al utilizar un SF de 7, la probabilidad de pérdida de mensajes es considerablemente alta, alcanzando cerca de la mitad de los mensajes enviados bajo esta configuración. El SF8 presenta una mejora, con menos de un tercio de mensajes perdidos. Sin embargo, al usar SF12, aunque la probabilidad de pérdida de paquetes disminuye significativamente, no se elimina por completo.

Dado que cada mensaje enviado por el objeto debe llegar al servidor debido a la importancia de la información para el correcto funcionamiento del sistema, se decide emplear la tecnología NB-IoT en la nueva versión del objeto conectado para garantizar la fiabilidad en la transmisión de datos.

4 | Creación de la infraestructura web y DevOps

En el contexto del desarrollo de la solución GoodFloow, se crea una aplicación web para la gestión de embalajes reutilizables. El desarrollo de esta aplicación está a cargo del equipo de desarrollo full-stack, responsable tanto del back end (infraestructura monolítica y microservicios) como del front end de la aplicación.

Inicialmente, la aplicación se alojaba en una máquina virtual dentro del servicio en la nube de Amazon Web Services, donde, mediante Docker, se ejecutaban las aplicaciones de forma contenedorizada con la infraestructura siguiente:

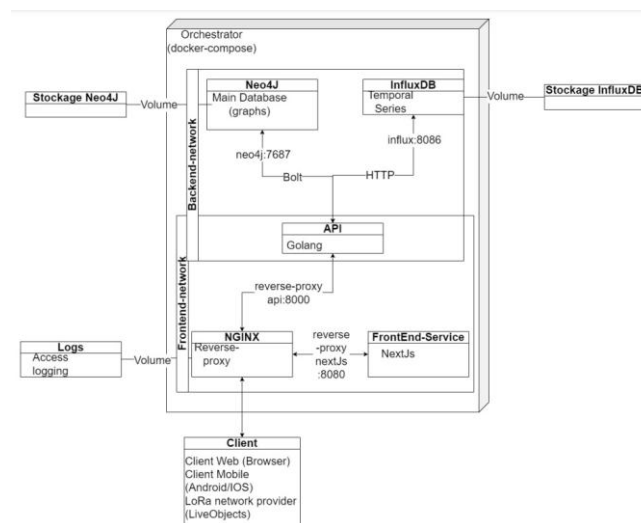


Figura 4.1: Antigua infraestructura web.

Por otro lado, el equipo de desarrollo carece de herramientas para optimizar el proceso de desarrollo y despliegue de la aplicación, como la ausencia de un entorno adecuado para realizar pruebas de integración y desplegar la aplicación tras los avances en su desarrollo.

4.1. Creación de la infraestructura web en Scaleway

Como se menciona anteriormente, la aplicación operaba dentro de contenedores alojados en una máquina virtual. Sin embargo, esta configuración no representa una infraestructura web óptima para un servicio destinado a manejar múltiples clientes y mensajes de forma simultánea.

Los principales puntos críticos que requieren una mejora son los siguientes:

- **Punto único de falla:** si la máquina virtual falla por cualquier motivo, toda la aplicación deja de funcionar.
- **Falta de escalabilidad:** al contar únicamente con una máquina virtual, cualquier necesidad de recursos adicionales implica detener la máquina y asignarle más capacidad, lo que no es eficiente.

Por estas razones, se está implementa una nueva infraestructura basada en Kubernetes. Kubernetes (K8s) es una plataforma de código abierto que automatiza el despliegue, la escalabilidad y la gestión de aplicaciones contenedorizadas. Con K8s, cada contenedor funciona de manera independiente y no requiere estar alojado en la misma máquina virtual. Esto soluciona el problema del punto único de falla, ya que, si alguna máquina que hospeda contenedores falla, Kubernetes transfiere automáticamente los contenedores a otra máquina funcional, minimizando el tiempo de inactividad de la aplicación. Además, K8s permite escalar las aplicaciones de manera sencilla y automática en caso de necesitar más recursos, resolviendo así la falta de escalabilidad de la infraestructura anterior.

Para migrar de la antigua infraestructura web a la nueva, se utilizan los servicios de la empresa francesa de nube Scaleway, que ofrece una gama de servicios similares a los de otros proveedores de nube pública como Azure, Amazon Web Services o Google Cloud. Además, Scaleway cuenta con un programa para startups que proporciona créditos en su plataforma, lo que permite a GoodFloow adoptar una infraestructura web más confiable sin incurrir en costos durante la duración del programa. Este beneficio permite implementar la aplicación y el sistema GoodFloow en un entorno adecuado sin preocupaciones financieras.

En esta nueva infraestructura, se utilizan principalmente dos servicios de Scaleway:

- **Kubernetes Kapsule:** un servidor Kubernetes gestionado, fácil de usar y completamente integrado en el ecosistema de Scaleway.
- **IoT Hub:** un puente diseñado específicamente para conectar los objetos conectados con su infraestructura en la nube.

Con estos servicios, se ha diseñado una nueva infraestructura, que se presenta a continuación y se explica en los párrafos siguientes.

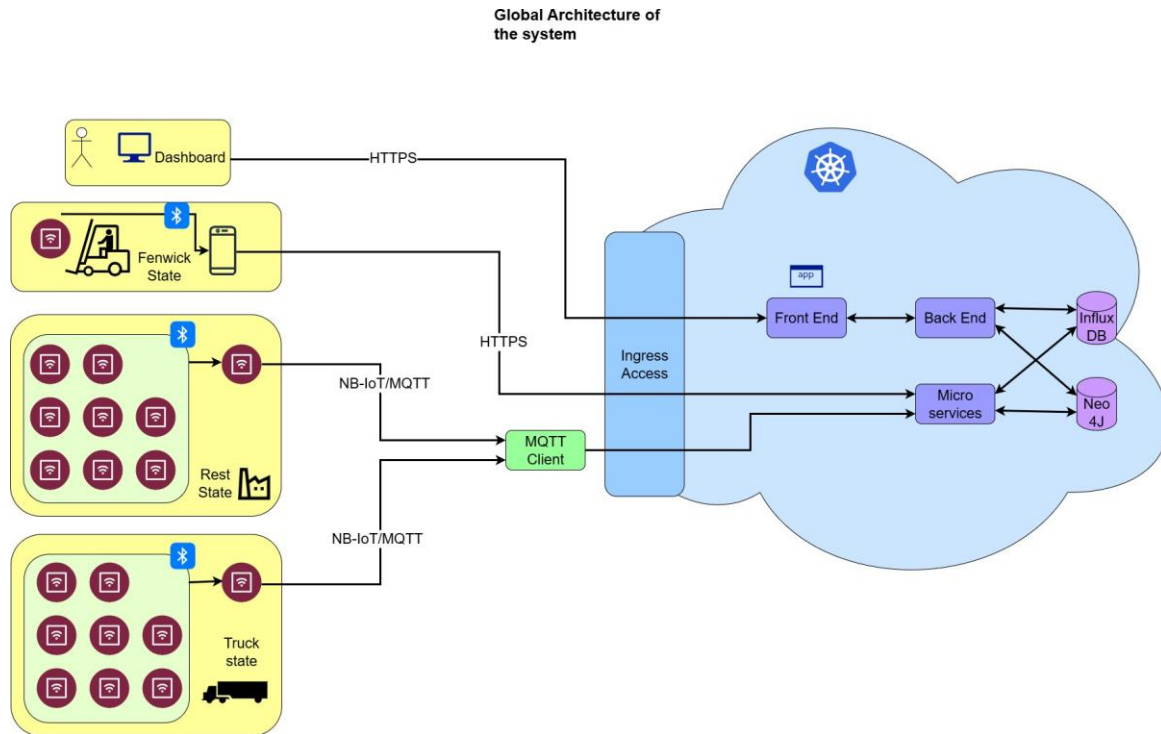


Figura 4.2: Nueva infraestructura web.

4.1.1. Kubernetes

Para implementar esta nueva infraestructura web, se crea un clúster en el servicio Kubernetes Kapsule, diseñado para desplegar y probar la aplicación en el nuevo entorno contenedorizado. Este clúster cuenta con dos máquinas virtuales, pero puede escalarse automáticamente según la carga recibida en momentos específicos.

Una vez creado el clúster, se procede al despliegue de la aplicación, comenzando con el desarrollo de los distintos archivos manifiestos de Kubernetes. Un archivo manifiesto en Kubernetes contiene instrucciones, en formato YAML o JSON, que especifican cómo desplegar una aplicación en los nodos de un clúster Kubernetes.

Los servicios desplegados mediante los manifiestos incluyen las bases de datos Neo4J e InfluxDB, el back end monolítico (dashboard), los back ends en microservicios (camión, montacargas e inmóvil,

cada uno con su propio manifiesto) y el front end. Dado que el proyecto se encuentra en la fase de desarrollo, las bases de datos han sido creadas dentro del clúster, lo que no es una práctica recomendada para un entorno de producción. Sin embargo, esta configuración es suficiente para verificar el funcionamiento de la aplicación. En el futuro, cuando se realice la migración a gran escala, las bases de datos deberán ser gestionadas en servidores externos.

Tras el despliegue de todos los servicios necesarios, es necesario hacerlos visibles al mundo exterior, ya que, en su estado inicial dentro de Kubernetes, no cuentan con direcciones IP públicas que permitan su acceso fuera del entorno local del clúster.

Por esta razón, se despliega el servicio Nginx Ingress Controller. Este servicio actúa como un proxy inverso, es decir, un servidor que se sitúa frente a los servidores web y retransmite las solicitudes de los clientes (como navegadores) a estos servidores. Los proxys inversos suelen utilizarse para mejorar la seguridad, el rendimiento y la fiabilidad. Con Nginx Ingress Controller, se logra acceder a todos los servicios externos necesarios (back end y front end) mediante una única dirección IP pública. Por otro lado, las bases de datos no son accesibles desde el exterior, lo que añade una capa adicional de seguridad al evitar posibles modificaciones no autorizadas en datos críticos.

Además, se integra un certificado SSL en las rutas de acceso a los servicios, garantizando una conexión cifrada y segura entre los usuarios y el servidor.

4.1.2. IoT Hub

El objeto conectado debe ser capaz de enviar la información recopilada a un servidor para procesarla y monitorear en tiempo real el estado de los dispositivos. En la versión anterior, se utilizaba la red LoRaWAN, lo que permitía visualizar los mensajes recibidos en el servicio LiveObjects de Orange [9] . Estos mensajes son reenviados al servidor encargado de su procesamiento. Sin embargo, la nueva versión del objeto conectado utiliza NB-IoT, una tecnología que mejora la fiabilidad en el

envío de mensajes y permite emplear protocolos como HTTPS o MQTT. Esto plantea la necesidad de evaluar si LiveObjects sigue siendo útil o si es mejor optar por otra solución.

En este contexto, el servicio IoT Hub de Scaleway se presenta como una alternativa adecuada. IoT Hub actúa como una pasarela para los mensajes de todos los objetos conectados, encargándose de retransmitirlos a los microservicios responsables de su tratamiento. Para soportar esta funcionalidad, se implementa un broker MQTT dentro del IoT Hub.

Un broker MQTT es una entidad intermedia que facilita la comunicación entre clientes MQTT, operando como un concentrador central. Su función principal es gestionar el flujo de mensajes: recibe los publicados por los clientes, los filtra según el tema y los distribuye a los suscriptores correspondientes. En esta nueva infraestructura, el broker MQTT recibe de manera segura los mensajes enviados por los objetos conectados. La seguridad se garantiza mediante certificados exclusivos para los dispositivos autorizados, asegurando que solo ellos puedan comunicarse con el broker.

Los mensajes recibidos son luego redirigidos a los microservicios encargados de procesarlos. Para proteger aún más el sistema, se establece que únicamente el broker MQTT puede realizar solicitudes a estos microservicios, bloqueando cualquier acceso externo no autorizado. Esto se implementa a través de una lista blanca (“whitelist”) en el Nginx Ingress Controller, añadiendo una capa adicional de seguridad al flujo de datos.

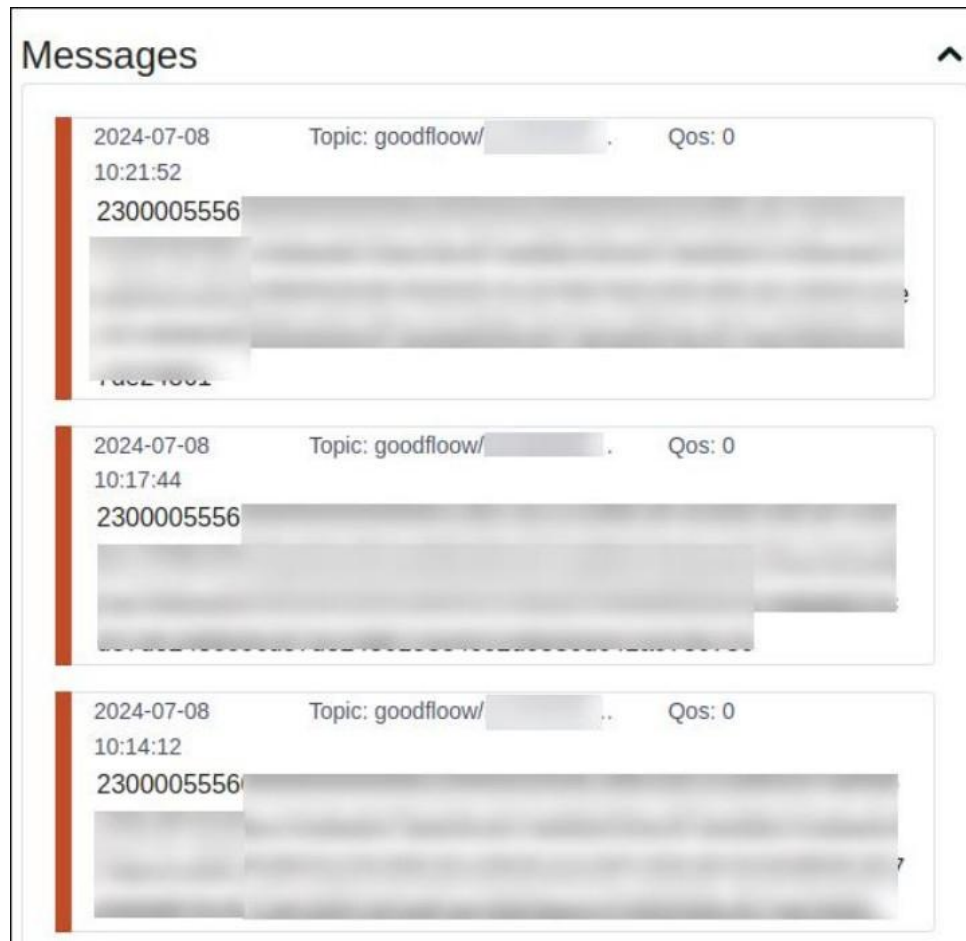


Figura 4.3: Mensajes recibidos en IoT Hub.

4.2. DevOps

DevOps es un conjunto de prácticas, herramientas y principios culturales destinados a automatizar e integrar los procesos compartidos entre los equipos de desarrollo de software y los equipos de operaciones informáticas. Como se menciona anteriormente, uno de los requisitos fundamentales para los desarrolladores es la automatización de tareas clave, como la ejecución de pruebas y la actualización de servidores tras la implementación de nuevas funcionalidades en el sistema. La metodología DevOps permite abordar estas necesidades de manera eficiente.

Mediante la adopción de DevOps, se promueve la colaboración entre equipos, facilitando la integra-

ción continua (CI) y la entrega continua (CD). Estas prácticas no solo reducen el tiempo necesario para implementar cambios, sino que también mejoran la fiabilidad y la calidad del software. Además, la automatización de tareas rutinarias minimiza errores humanos y libera recursos para que los desarrolladores puedan centrarse en actividades más estratégicas.

En este contexto, herramientas como Docker o Kubernetes juegan un papel clave, ya que permiten gestionar procesos de integración y despliegue con mayor facilidad, garantizando la consistencia y escalabilidad del sistema. La filosofía DevOps, por tanto, no solo responde a las demandas técnicas, sino que también fomenta una cultura de mejora continua y responsabilidad compartida dentro de los equipos.

4.2.1. Ambiente para los test de integración

La prueba de integración es una etapa clave en el proceso de desarrollo de software, ya que permite evaluar cómo interactúan y funcionan de manera coherente los diferentes módulos de una aplicación. Estas pruebas buscan replicar al máximo el entorno del servidor real, asegurando las mismas interacciones entre módulos y contenedores que tendrá la aplicación en producción. Esto garantiza que el desarrollo cumple con los requisitos establecidos y que la aplicación funcionará correctamente tras cualquier actualización.

Con este objetivo, se ha creado un entorno de prueba utilizando Docker Compose, que permite configurar una infraestructura similar a la del servidor real. Este entorno incluye dos bases de datos y el servicio que se desea probar, así como las comunicaciones entre ellos, replicando fielmente la arquitectura de producción.

Este entorno de prueba puede ejecutarse tanto de manera local como en la nube, lo que proporciona flexibilidad para validar actualizaciones del servidor. Gracias a esta configuración, se reduce el riesgo de errores en producción, se garantiza la calidad del software y se acelera el ciclo de desarrollo y

despliegue.

4.2.2. Pipelines CI/CD

Uno de los principios fundamentales de la metodología DevOps es la automatización de procesos, como la ejecución de pruebas o la actualización de servidores con nuevas versiones de aplicaciones, mediante los pipelines CI/CD. Estas siglas significan integración continua (CI) y distribución o despliegue continuo (CD), cuyo propósito es mejorar y optimizar el ciclo de desarrollo de software.

La integración continua consiste en incorporar de manera automática y periódica modificaciones del código en un repositorio. Por su parte, la distribución o despliegue continuo se divide en dos etapas: la integración, las pruebas y la distribución del código. En la distribución continua, los cambios no se implementan automáticamente en producción, mientras que en el despliegue continuo, las actualizaciones se aplican de manera automática en este entorno.

Esta metodología se integra dentro del proceso de desarrollo del proyecto. Inicialmente, solo se automatizaba la ejecución de pruebas unitarias durante la fase de desarrollo. Para empezar, considerando que el proyecto está alojado en GitHub, se crean archivos en los distintos repositorios del proyecto que contienen las instrucciones para ejecutar las pruebas de integración y desplegar la nueva aplicación en el servidor. Estas tareas se ejecutan mediante GitHub Actions [10], una herramienta de GitHub que permite automatizar actividades como la compilación, prueba y despliegue de aplicaciones.

En el caso de las pruebas de integración, primero se indica compilar la aplicación y ejecutar los contenedores necesarios para su correcto funcionamiento. Luego, se ejecutan las pruebas de integración definidas por el equipo de desarrollo. Una vez validadas las pruebas, la aplicación se detiene, y GitHub Actions genera un informe con los resultados. Este proceso se activa automáticamente

cada vez que se realiza una “pull request” (solicitud para combinar ramas de desarrollo en Git). Esto permite validar los cambios antes de incorporarlos, identificando errores en las primeras etapas del desarrollo y evitando su detección en producción.

Para el despliegue automatizado, las instrucciones dadas a GitHub Actions incluyen conectarse al servicio “GitHub Container Registry” (GHCR), que permite almacenar y transferir imágenes contenedorizadas de forma segura. Tras esta conexión, se compila el código y se genera una imagen Docker con la aplicación, que luego se envía al GHCR. Al utilizar Kubernetes para alojar la aplicación, el despliegue se simplifica al proporcionar a GitHub Actions el archivo kubeconfig para conectar al clúster y los manifiestos de Kubernetes con las instrucciones de despliegue. Finalmente, GitHub Actions ejecuta los manifiestos, localiza la última imagen publicada en GHCR y la implementa en el clúster Kubernetes.

Estas tareas se llevan a cabo automáticamente después de validar la “pull request” y combinarla con la rama de desarrollo, asegurando que siempre se despliegue la última versión de la aplicación. Este enfoque permite mantener una integración y distribución continuas, acelerando el proceso de desarrollo, ya que los desarrolladores pueden centrarse en crear nuevas funcionalidades sin preocuparse por la ejecución de pruebas o el despliegue de la aplicación.

5 | Conclusiones

El desarrollo de esta memoria ha permitido abordar de manera integral el diseño, implementación y evaluación de una solución IoT para la gestión y el seguimiento de embalajes industriales reutilizables, integrando tecnologías innovadoras y metodologías avanzadas. Este trabajo se ha centrado en resolver las limitaciones de la versión inicial del dispositivo conectado, mejorando su hardware, software y la infraestructura asociada, lo que se tradujo en una solución más robusta, eficiente y alineada con los objetivos de sostenibilidad y eficiencia logística.

Uno de los principales logros fue el diseño y desarrollo de una nueva máquina de estados, la cual simplificó y optimizó la lógica de funcionamiento del dispositivo. Esta nueva configuración eliminó redundancias y mejoró la capacidad del sistema para manejar transiciones y decisiones en tiempo real, incluso ante imprecisiones de la inteligencia artificial integrada.

El uso de tecnologías modernas, como Kubernetes y DevOps, fue fundamental para garantizar la escalabilidad, fiabilidad y facilidad de gestión de la infraestructura web. La migración a una arquitectura basada en contenedores y la integración de procesos automatizados, como pipelines CI/CD, optimizaron significativamente el desarrollo y despliegue del sistema, minimizando riesgos y maximizando la calidad del software. Asimismo, la adopción de nuevas capacidades de conectividad, como NB-IoT, mejoró la fiabilidad en la transmisión de datos, abordando limitaciones detectadas en versiones previas.

La inclusión de pruebas exhaustivas, tanto unitarias como de integración, aseguró la solidez y

el correcto funcionamiento de cada componente del sistema. Este enfoque meticuloso permitió identificar y corregir errores en etapas tempranas, consolidando la confiabilidad del dispositivo conectado y de la infraestructura web.

Finalmente, este proyecto no solo alcanzó los objetivos establecidos, sino que también sentó las bases para futuras mejoras y expansiones. La colaboración con instituciones académicas y la integración de avances tecnológicos en inteligencia artificial y aprendizaje automático abren nuevas posibilidades para el perfeccionamiento de la solución. En conjunto, estos avances representan una contribución significativa al ámbito de la gestión sostenible de embalajes industriales, alineándose con los retos actuales de sostenibilidad y optimización logística.

Bibliografía

- [1] Kubernetes. *Kubernetes Documentation*. [en línea]. Available: <https://kubernetes.io/docs/home/> [consulta: 16 feb. 2025]. n.d. url: <https://kubernetes.io/docs/home/>.
- [2] Scaleway. *Scaleway Cloud*. [en línea]. Available: <https://www.scaleway.com/en/> [consulta: 16 feb. 2025]. n.d. url: <https://www.scaleway.com/en/>.
- [3] Muhammad Ehsan Rana, Kamalanathan Shanmugam y Chan Hang. “Recommendations for Implementing an IoT based Inventory Tracking and Monitoring System”. En: ago. de 2023, págs. 24-35. ISBN: 978-981-5080-96-4. doi: [10.2174/9789815080957123010006](https://doi.org/10.2174/9789815080957123010006).
- [4] Dr M et al. “Asset Tracking and Management System”. En: *International Journal for Research in Applied Science and Engineering Technology* 11 (jun. de 2023), págs. 3679-3684. doi: [10.22214/ijraset.2023.54185](https://doi.org/10.22214/ijraset.2023.54185).
- [5] Pooja Joshi et al. “Intelligent IoT-Enabled Real-Time Monitoring System for Logistics Management”. En: oct. de 2024, págs. 1-5. doi: [10.1109/ICCSC62048.2024.10830332](https://doi.org/10.1109/ICCSC62048.2024.10830332).
- [6] Hyung Rim Choi et al. “Development of an IoT-based container tracking system for China’s Belt and Road (BR) initiative”. En: *Maritime Policy and Management* 45 (nov. de 2017), págs. 1-15. doi: [10.1080/03088839.2017.1400190](https://doi.org/10.1080/03088839.2017.1400190).
- [7] Mariano Falcitelli et al. “Development of a Multi-Radio Device for Dry Container Monitoring and Tracking”. En: *IoT* 5 (abr. de 2024), págs. 187-211. doi: [10.3390/iot5020010](https://doi.org/10.3390/iot5020010).
- [8] Espressif Systems. *ESP-IDF Programming Guide*. [en línea]. Available: <https://docs.espressif.com/projects/esp-idf/en/latest/esp32/> [consulta: 16 feb. 2025]. n.d. url: <https://docs.espressif.com/projects/esp-idf/en/latest/esp32/>.
- [9] Orange. *Live Objects: IoT Platform*. [en línea]. Available: <https://liveobjects.orangebusiness.com/#/liveobjects> [consulta: 16 feb. 2025]. n.d. url: <https://liveobjects.orangebusiness.com/#/liveobjects>.
- [10] Github. *Github Actions Documentation*. [en línea]. Available: <https://docs.github.com/en/actions> [consulta: 16 feb. 2025]. n.d. url: <https://docs.github.com/en/actions>.

A | Anexo A: Extracto de la descripción del payload

Campo	Tamaño en la trama	Descripción
Tipo de trama + nivel de batería	1 octeto (2 caracteres)	Tipo de trama (4 bits): info_packet = 0, neighbour_packet = 1, wi-fi_packet = 2 Nivel de batería (4 bits): muestreo entre 3300 mV y 4110 mV, mapeado a un intervalo entre 0 y 8.
ID IoT	4 octetos (8 caracteres)	ID del IoT.
Estado actual + estado anterior	1 octeto (2 caracteres)	Estado actual (4 bits) Estado anterior (4 bits).
Timestamp interno	4 octetos (8 caracteres)	Timestamp interno en minutos del IoT.
...	...	...

Tabla A.1: Extracto de la descripción del payload

B | Anexo B: Instrucciones para desplegar los test de integracion de manera local

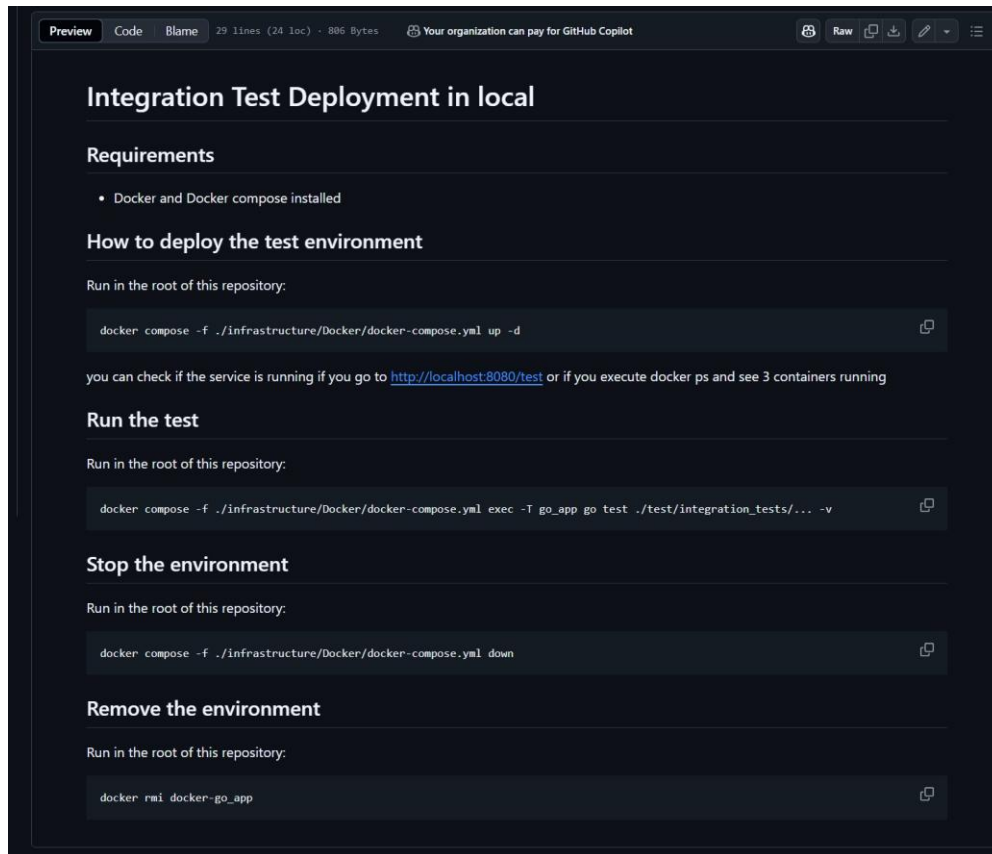


Figura B.1: Instrucciones en Github.