Plataforma IoT Distribuida y Segura para Monitoreo Industrial

Alonso Bustos, Kaori Encina, Gabriel Huerta, Diego San Martín

Julio 2025

1. Introducción

En la actualidad, la industria moderna se encuentra en plena transformación gracias a la implementación de nuevas tecnologías que buscan mejorar la eficiencia, la productividad y la toma de decisiones en tiempo real. En este contexto, los sistemas distribuidos y el IoT se han convertido en piezas clave para alcanzar dichos objetivos.

El objetivo del proyecto es que los estudiantes diseñen e implementen un sistema distribuido heterogéneo que simule un entorno IoT industrial. Esto, para aplicar los conocimientos vistos en clases y demostrar sus capacidades a la hora de desarrollar un sistema claro, robusto, eficiente y seguro.

2. Descripción del Problema

Las empresas y organizaciones hoy en día utilizan una gran cantidad de infraestructura compuesta por múltiples dispositivos, aplicaciones, plataformas, etc., para el almacenamiento y procesamiento de grandes cantidades de datos, lo que hace que sea necesario contar con un sistema robusto de transmisión y análisis de datos que sea seguro, eficiente y que funcione en un entorno heterogéneo.

Cuando nos referimos a un entorno heterogéneo, hablamos de un entorno en el que coexisten diferentes tecnologías, sistemas operativos, lenguajes de programación, dispositivos físicos y plataformas de software.

La interoperabilidad es indispensable en este tipo de entornos, ya que permite la integración fluida de distintos tipos de hardware y software, y la comunicación en tiempo real entre sistemas locales y plataformas remotas sin la necesidad de una implementación compleja o adaptaciones manuales. Sin ella, los datos quedarían fragmentados y la organización no podría aprovechar plenamente el valor que estos ofrecen.

La seguridad es uno de los aspectos más importantes a la hora de trabajar con datos, ya que, si no se cuenta con un nivel de resguardo suficiente, estos pueden ser robados, eliminados o alterados por ataques externos, sobre todo cuando los datos a proteger pasan por diversos sistemas, cada uno con un nivel de protección diferente.

Por otro lado, la visualización de los datos se vuelve necesaria para mostrar la información de manera gráfica o visual y hacer que los datos sean fáciles de entender y analizar, ya que cuando los datos se presentan solo en listas o tablas enormes, es muy difícil detectar patrones, tendencias o problemas rápidamente.

3. Objetivos

3.1. Objetivo General

Diseñar e implementar un sistema distribuido y seguro para la simulación de un entorno de monitoreo industrial.

3.2. Objetivos Específicos

- Implementar un cliente en C++ que genere datos simulados.
- Desarrollar un servidor intermedio que procese y reenvíe los datos.
- Construir un backend en Python con almacenamiento persistente y API REST.
- Diseñar un cliente de consulta asíncrono con detección de anomalías.
- Visualizar métricas en tiempo real mediante una interfaz web o similar.

4. Metodología

4.1. Arquitectura General del Sistema

El sistema se compone de los siguientes módulos:

1. **Cliente Sensor (C++)**: Envía datos binarios con firma de verificación.
2. **Servidor Intermedio (Python)**: Verifica los datos y los transforma a texto.
3. **Servidor Final (Python)**: Almacena los datos en SQLite y expone una API REST.
4. **Cliente de Consulta (Python)**: Realiza polling periódico y genera alertas.

4.2. Implementación de Componentes

- **Cliente Sensor**: Generación y envío de datos binarios cifrados mediante XOR con clave fija, incluyendo verificación de integridad con checksum, con capacidad de reconexión automática tras fallos de comunicación (3 intentos consecutivos como máximo).
- **Servidor Intermedio**: Recibe los datos binarios , verifica el checksum y transforma los datos, con capacidad de aguantar caidas con el uso de un sistema de confirmación (ACK), a la vez se encarga de guardar los datos en caso de que el servidor final este caido.

- **Servidor Final**: Almacenamiento en SQLite, con capacidad de aguantar caidas con el uso de un sistema de confirmación (ACK) y exposición de API.
- **Cliente de Consulta**: Polling asíncrono y generación de alertas.

4.3. Comunicación entre Componentes

Uso de sockets TCP y otro a elección para la transmisión binaria y textual. Especificación de formatos y protocolo usado. ¿Hay protocolos que sean más adecuados para esta tarea en entornos reales?

Para la comunicación entre los distintos módulos del sistema se utilizó el protocolo TCP mediante el uso de sockets, lo cual permitió establecer una transmisión de datos binaria confiable entre el cliente sensor (implementado en C++) y el servidor intermedio (en Python). TCP garantiza el orden y la entrega de los paquetes, esto es esencial en aplicaciones donde la integridad de los datos es crítica, tal como en el monitoreo industrial.

El protocolo implementado incluye un sistema de confirmación (ACK) donde el servidor intermedio debe responder con un mensaje .ªCK"después de recibir cada paquete de datos, garantizando así la entrega exitosa y permitiendo al cliente detectar fallos de comunicación, tambien es usado el mismo sistema por el servidor final para poder tener mas robustes en el sistema, así logrando que a pesar de que haya caídas el sistema completo no falla.

El servidor Intermedio guarda los datos recibidos en caso de fallas en el servidor final, para que en el momento en que el servidor final vuelva a estar operativo le puedan llegar los datos que fueron recibidos, por lo que el servidor intermedio intenta mandar los datos recibidos cada 10 segundos cuando el servidor final falla.

El formato utilizado para los paquetes fue binario estructurado mediante una estructura C++ empaquetada que contiene los siguientes campos: id_sensor (int16_t), timestamp (uint64_t), temperatura (float), presion (float), humedad (float), y checksum (uint32_t) para verificación de integridad.

La estructura de datos utiliza empaquetado de bytes `__attribute__((packed))` para optimizar la transmisión y garantizar la compatibilidad entre diferentes arquitecturas. Los datos se cifran antes del envío usando una operación XOR con clave fija (0xAB), excluyendo el campo checksum que se mantiene sin cifrar para permitir la verificación de integridad en el servidor receptor.

La elección de TCP fue adecuada para esta simulación, pero en entornos reales existen protocolos más especializados para IoT y monitoreo industrial. Algunos ejemplos son:

- **MQTT (Message Queuing Telemetry Transport)**: Es un protocolo ligero basado en el modelo *publish/subscribe*, ideal para redes con limitaciones de ancho de banda y recursos, como sensores remotos.
- **CoAP (Constrained Application Protocol)**: Protocolo similar a HTTP pero optimizado para dispositivos embebidos.
- **OPC UA (Open Platform Communications Unified Architecture)**: Protocolo ampliamente utilizado en entornos industriales que soporta transmisión de datos, modelado de información y seguridad.

Estos protocolos suelen ofrecer ventajas como menor consumo energético, soporte para desconexiones temporales, interoperabilidad entre dispositivos heterogéneos y mayor escalabilidad.

5. Resultados obtenidos

- **Métricas recolectadas y ejemplos de datos**

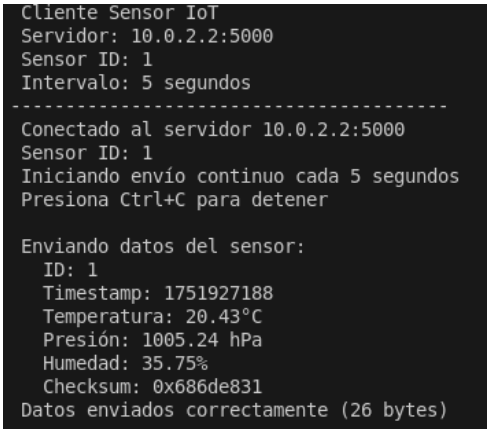

Figura 1: Datos enviados por el Sensor.

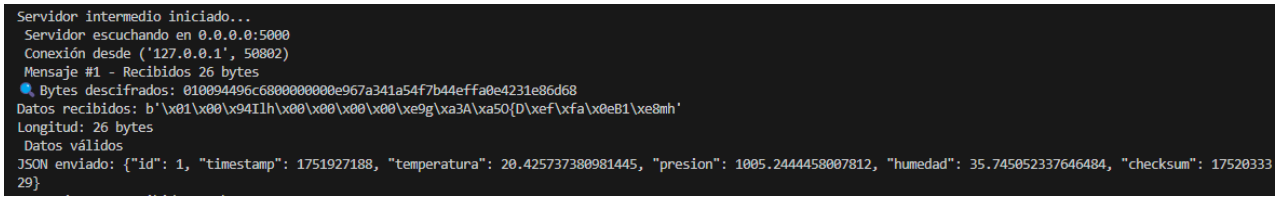

Figura 2: Datos recibidos por el servidor intermedio

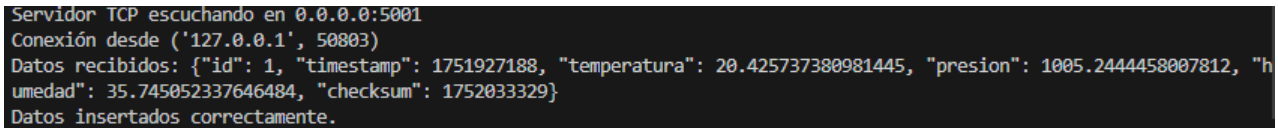

Figura 3: Datos recibidos por el servidor final

- **Visualización en la web**

ID	Fecha y Hora	Temperatura	Presión	Humedad
1	1751935278	34.910743713378906	1067.9219970703125	39.9129638671875
1	1751935143	46.088863372802734	984.6607055664062	87.81598663330078
1	1751935148	2.1843485832214355	1100.933837890625	57.65250778198242
1	1751935163	2.940103530883789	1084.0279541015625	81.1160659790039
1	1751935168	3.8929476737976074	937.7835693359375	77.58909606933594
1	1751935173	40.70917892456055	904.689208984375	97.00093841552734
1	1751935178	9.015585899353027	1019.5462646484375	61.14277267456055

Figura 4: Vista de la página

- **Alertas generadas**

```
CLIENTE CONSULTA : Última actualización: 20:43:58
------------------------------------------------------------------------------------------------------
Sensor Hora      Temp (°C)   Presión (hPa)  Humedad (%)  Alertas
======================================================================================================
1      20:43:33  10.8        1071.10        50.0         Presión fuera de rango (1071.096923828125)
1      20:43:38  10.2        987.73         23.8
1      20:43:43  10.3        1005.43        34.4
1      20:43:48  21.2        1066.65        30.9         Presión fuera de rango (1066.6468505859375)
1      20:43:53  44.8        1010.90        54.3         Temperatura fuera de rango (44.793636322021484)
------------------------------------------------------------------------------------------------------
Umbrales: {'temperatura': (5.0, 40.0), 'presion': (950.0, 1050.0), 'humedad': (20.0, 80.0)}
```

Figura 5: Warnings de cliente consulta

6. Conclusiones y Trabajo Futuro

6.1. Logros Alcanzados

El proyecto logró cumplir con el objetivo de implementar una plataforma IoT distribuida capaz de recibir, almacenar, consultar y evaluar datos de sensores de forma segura y eficiente. Se utilizaron múltiples tecnologías de forma integrada: **sockets TCP** para la comunicación entre el cliente sensor y el servidor intermedio, y una **API REST** para el acceso y filtrado de datos desde el servidor final. Además, se desarrolló un cliente de consulta que detecta condiciones anómalas en tiempo real.

Se comprobó que la arquitectura puede operar de forma continua con una correcta separación de responsabilidades entre los componentes. El servidor final persistió la información en una base de datos SQLite, y el cliente de consulta logró identificar lecturas anómalas según criterios definidos, mostrando alertas al usuario.

Un aspecto destacable fue la implementación de mecanismos de tolerancia a fallos, incluyendo el sistema de confirmación (ACK) bidireccional entre componentes y la capacidad del servidor intermedio de almacenar datos temporalmente durante caídas del

servidor final. Esto garantiza la continuidad del servicio y la no pérdida de información crítica.

También se demostró que es posible simular sensores con generación de datos aleatorios para pruebas, sin necesidad de hardware físico. Esto facilitó la validación del sistema y permitiendo evaluar su comportamiento frente a distintos escenarios.

6.2. Contribuciones Técnicas

El proyecto aportó soluciones técnicas relevantes en varios aspectos:

- **Interoperabilidad**: Integración exitosa de componentes desarrollados en diferentes lenguajes (C++ y Python), demostrando la viabilidad de sistemas heterogéneos.
- **Seguridad de datos**: Implementación de cifrado XOR con verificación de integridad mediante checksum, estableciendo una primera línea de defensa contra alteraciones no autorizadas.
- **Robustez comunicacional**: Desarrollo de un protocolo de comunicación confiable con manejo de reconexiones automáticas y tolerancia a fallos de red.
- **Escalabilidad modular**: Arquitectura que permite la incorporación de nuevos sensores y servicios sin modificaciones estructurales significativas.

6.3. Limitaciones Identificadas

En cuanto a las limitaciones del sistema implementado:

- El cliente sensor es simulado; no se utilizaron dispositivos físicos reales, lo que limita la validación en condiciones operacionales reales.
- La API REST no cuenta con autenticación ni cifrado TLS, lo que en un entorno productivo puede comprometer la seguridad y cumplimiento normativo.
- La interfaz del cliente de consulta es de línea de comandos, sin una visualización gráfica intuitiva o capacidad de exportación de reportes.
- El sistema completo depende de múltiples terminales abiertas en paralelo, lo que dificulta su operación por parte de usuarios no técnicos.
- El cifrado XOR implementado, aunque funcional para la demostración, no cumple con estándares industriales de seguridad.

6.4. Trabajo Futuro y Mejoras Propuestas

Como líneas de mejora prioritarias se proponen: **Mejoras de Hardware y Conectividad:**

- Reemplazar la simulación del sensor por dispositivos físicos reales con protocolos industriales estándar.
- Implementar comunicación inalámbrica (Wi-Fi, LoRaWAN) para mayor flexibilidad de despliegue.

Fortalecimiento de Seguridad:

- Incorporar cifrado TLS/SSL y autenticación robusta de usuarios en la API REST.
- Implementar certificados digitales para la autenticación de dispositivos IoT.
- Desarrollo de un sistema de gestión de claves más sofisticado que el cifrado XOR actual.

Mejoras de Arquitectura:

- Reemplazar el sistema de polling del cliente por WebSockets para obtener datos en tiempo real sin sobrecargar la red.
- Implementar un sistema de mensajería asíncrona (MQTT/RabbitMQ) para mayor escalabilidad.
- Migrar la base de datos a un sistema más robusto como PostgreSQL para soportar mayor volumen de datos.

Experiencia de Usuario:

- Añadir una interfaz gráfica web responsiva que permita visualizar las mediciones, tendencias históricas y alertas de forma intuitiva.
- Desarrollar un sistema de notificaciones push para alertas críticas.
- Implementar dashboards personalizables para diferentes tipos de usuarios.

Análisis Avanzado:

- Incorporar algoritmos de machine learning para detección predictiva de anomalías.
- Implementar análisis de tendencias y generación automática de reportes.

6.5. Impacto y Proyección

El proyecto fomentó una comprensión más profunda sobre los desafíos del diseño de sistemas distribuidos en el contexto del Internet de las Cosas industrial. La experiencia obtenida en la integración de tecnologías heterogéneas, el manejo de la comunicación asíncrona y la implementación de mecanismos de tolerancia a fallos constituye una base sólida para futuros desarrollos en el área. La modularidad y escalabilidad de la solución desarrollada sientan las bases para futuras implementaciones más complejas, capaces de adaptarse a entornos industriales reales donde la confiabilidad, seguridad y usabilidad son factores críticos. Con las mejoras propuestas, el sistema podría evolucionar hacia una solución robusta para el monitoreo remoto de variables ambientales con aplicaciones en diversas industrias como manufactura, agricultura de precisión, gestión energética y mantenimiento predictivo. La metodología de desarrollo empleada, que combina simulación con arquitecturas distribuidas reales, ha demostrado ser efectiva para la validación de conceptos y puede ser replicada en proyectos similares de IoT industrial, contribuyendo así al avance del conocimiento en esta área tecnológica estratégica.