Se evaluarán los siguientes aspectos:

Estructura del Póster científico, cumpliendo las partes que componen un póster científico

- Contexto (Identificar la universidad, cátedra, grupo, etc.)
- Título
- Introducción
- Materiales y métodos
- Resultados
- Conclusiones
- Referencias y Agradecimientos

Diseño del póster científico:

- Claridad: Que los textos resulten fácilmente legibles (medida y color adecuados), estén bien distribuidos y que no sean excesivos. Los gráficos y los textos deben contribuir a clarificar el mensaje que se quiere transmitir.
- Impacto visual: La calidad de las imágenes utilizadas, la composición, encuadre, la pertinencia de las imágenes elegidas. La construcción de las frases.
- Ortografía y redacción e idioma. El póster debe estar escrito en forma completa en un mismo idioma, Español o Inglés.

Aspectos técnicos:

- Abordaje del tema y cumplimiento del objetivo: Los contenidos deben presentarse de forma organizada, bien estructurada y clara y que hagan foco en los conceptos que deben desarrollar. Las ideas deben desarrollarse gradualmente, con claridad, reiterando los conceptos principales. Adicionalmente deben transmitir de manera completa los contenidos del tema elegido.
- Corrección conceptual.

Exposición

- Preparación de la exposición
- Claridad
- Cumplimiento del tiempo asignado (5 minutos).

El siguiente checklist es una ayuda para la definición del Plan de Release, que debe contener al menos los ítems listados a continuación:

- ☐ Entorno de destino
- ☐ Tipo de despliegue
- ☐ Tipo de infraestructura se requiere para hacer el despliegue.
- ☐ Artefacto (software) que se despliega: Identificación exacta del software que se despliega y de la versión anterior a la que se puede volver.
- ☐ Cómo se realiza el despliegue y se instala el artefacto en el entorno de destino.
- ☐ Dependencias
- ☐ Pre-requisitos que deben estar presentes antes del despliegue.
- ☐ Horario de despliegue: Momento elegido para realizarlo y su justificación.
- ☐ Forma de despliegue (técnica)
- ☐ Estrategia concreta de despliegue para reducir riesgos.
- ☐ Monitoreo
- ☐ Pruebas post-despliegue: Validaciones mínimas que confirman que la aplicación funciona después del despliegue.
- ☐ Comunicación y aprobaciones: A quién se avisa antes y después del despliegue y quién debe aprobarlo.
- ☐ Plan de rollback / contingencia: Qué pasos concretos se siguen si el despliegue falla.

Introducción

Una empresa provee sensores IoT para medir temperatura y humedad en plantas industriales. Estos dispositivos están distribuidos en distintas fábricas del país y tienen una conectividad a internet muy limitada: algunos se conectan una vez al día, otros solo a través de redes locales cerradas.
El fabricante necesita desplegar nuevos firmwares para corregir bugs y mejorar algoritmos de medición. El gran riesgo es que, si la actualización falla, el dispositivo puede quedar inutilizable (brick) y enviar un técnico físicamente sería muy costoso.

Este documento desarrolla un **plan de despliegue de productos de software**, centrado en los dispositivos IoT de la empresa, abordando los aspectos técnicos, metodológicos y de calidad que permiten un proceso **seguro, trazable y automatizado**.

2. Análisis del contexto

Problemas identificados:

- Conectividad irregular o restringida.
- Riesgo de inutilización del dispositivo si la actualización falla (brick).
- Alto costo y tiempo de envío de técnicos a planta.
- Necesidad de mantener un control de versiones y trazabilidad centralizada.

Objetivos del plan:

- Garantizar actualizaciones remotas (OTA) seguras y confiables.
- Minimizar el impacto de fallas mediante rollback y despliegue gradual.
- Asegurar trazabilidad, auditoría y control de configuraciones.
- Implementar buenas prácticas de DevOps y aseguramiento de calidad.

3. Tipos de productos de software involucrados

Tipo	Descripción	Rol en el despliegue
Firmware (software embebido)	Programa que controla el sensor y sus algoritmos de medición.	Es el principal objeto del despliegue; se actualiza OTA.
Software de backend / servidor IoT	Sistema central que gestiona dispositivos, versiones, y monitoreo.	Orquesta las actualizaciones, agrupa dispositivos, almacena logs.
Aplicaciones de soporte	Dashboards, scripts de control y herramientas de prueba.	Permiten administrar, verificar y auditar el proceso.

4. Estrategias de Despliegue

Para el escenario planteado, se seleccionan estrategias combinadas que permiten equilibrar seguridad, confiabilidad y control en la actualización de firmware de dispositivos IoT con conectividad limitada. A continuación se describen los métodos elegidos, su funcionamiento y las razones de su selección.

a. OTA (Over-The-Air Updates)

Descripción:
El método OTA consiste en la distribución e instalación remota de actualizaciones de firmware mediante una conexión de red, sin necesidad de intervención física sobre los dispositivos. Cuando el sensor detecta conectividad disponible (ya sea a través de Internet o de una red local), descarga el nuevo paquete de firmware desde un servidor autorizado, verifica su integridad y procede a su instalación.

Motivo de elección:
En un entorno con dispositivos distribuidos en distintas ubicaciones y acceso físico limitado, el despliegue OTA representa la alternativa más eficiente y económica. Permite reducir significativamente los costos de mantenimiento, disminuir el tiempo de respuesta ante fallas o vulnerabilidades, y centralizar la gestión de versiones y configuraciones.
Además, su implementación puede adaptarse a entornos de conectividad intermitente, configurando los sensores para que busquen actualizaciones solo cuando exista disponibilidad de red, optimizando así el uso de recursos.

Tecnologías sugeridas: Mender.io, Eclipse Hawkbit, AWS IoT Device Management o servidores propios con protocolos MQTT/HTTPS.

b. Despliegue gradual (Canary Release)

Descripción:
La estrategia de despliegue gradual, también conocida como *Canary Release*, consiste en liberar una nueva versión de software a un subconjunto reducido de dispositivos, generalmente entre un 5% y un 10% del total, antes de realizar el despliegue masivo. Durante esta etapa se monitorea el desempeño de los dispositivos actualizados y se analizan métricas de estabilidad, rendimiento y errores.

Motivo de elección:
El despliegue gradual permite detectar de manera temprana posibles defectos o incompatibilidades en el firmware, evitando que una falla se propague al resto del sistema. Esta técnica reduce el riesgo operativo y brinda un mecanismo de validación real en campo antes de aprobar la actualización para todos los sensores.
Dada la diversidad de entornos industriales y condiciones de conectividad, el canary release es especialmente útil para validar la robustez de la actualización en escenarios heterogéneos.

c. Rollback automático y particiones A/B

Descripción:
El mecanismo de particiones A/B implica la existencia de dos espacios de almacenamiento para firmware dentro del dispositivo: la partición activa (A) y la partición de respaldo (B). Cuando se descarga una actualización, el nuevo firmware se instala en la partición secundaria. Si la actualización se aplica correctamente y el sistema inicia con normalidad, la nueva partición pasa a ser la activa. En caso de error o inestabilidad, el dispositivo revierte automáticamente al firmware anterior mediante un proceso de *rollback*.

Motivo de elección:
Esta estrategia fue seleccionada por su capacidad de mitigar uno de los principales riesgos del despliegue de firmware remoto: el "brickeo" o inutilización del dispositivo. El uso de particiones A/B mejora significativamente la tolerancia a fallos y asegura la continuidad operativa de los sensores sin requerir intervención manual.
Combinado con el método OTA, el rollback automático permite realizar pruebas en campo con total seguridad, garantizando la recuperación inmediata ante cualquier falla.

d. Blue-Green Deployment (para el backend)

Descripción:
El método *Blue-Green Deployment* se aplica en los componentes del backend que gestionan las actualizaciones OTA y la comunicación con los dispositivos. Consiste en mantener dos entornos de producción paralelos: uno activo (Blue) y otro de preparación (Green). Las nuevas versiones se despliegan primero en el entorno Green, donde se ejecutan pruebas de verificación y compatibilidad. Una vez validado, el tráfico se redirige hacia dicho entorno, dejando el anterior disponible como respaldo.

Motivo de elección:
Esta técnica permite realizar actualizaciones del servidor sin interrumpir el servicio ni afectar las conexiones de los dispositivos. Es especialmente útil para sistemas que deben permanecer disponibles de forma continua, como los servidores de coordinación OTA. Además, el Blue-Green Deployment simplifica los procedimientos de reversión: ante cualquier problema, el tráfico puede redirigirse nuevamente al entorno anterior de manera inmediata.
Su uso refleja buenas prácticas de ingeniería DevOps orientadas a la disponibilidad, confiabilidad y mejora continua del sistema.

5. Roles involucrados

Rol	Responsabilidad
Firmware Developer	Es el responsable del desarrollo del firmware que se ejecuta en los sensores IoT. Sus tareas incluyen la implementación de nuevas funcionalidades, la corrección de errores detectados en versiones anteriores, y la optimización del código para garantizar eficiencia y estabilidad.

	Además, documenta los cambios realizados, mantiene la compatibilidad con el hardware y colabora con el equipo de QA para validar el funcionamiento en entornos simulados y reales.
DevOps Engineer	Tiene a su cargo la configuración, automatización y mantenimiento de los procesos de integración y despliegue continuo (CI/CD). Se ocupa de desarrollar los pipelines que permiten compilar el firmware, ejecutar pruebas automatizadas, empaquetar las versiones y distribuirlas mediante la infraestructura OTA. También gestiona los entornos del backend (por ejemplo, servidores de actualización o dashboards de monitoreo), aplicando principios de infraestructura como código para garantizar entornos reproducibles y seguros.
QA Engineer	Es el encargado de garantizar la calidad del firmware antes y después de su despliegue. Diseña y ejecuta pruebas de integración, regresión y validación tanto en simuladores como en hardware físico. Su función incluye la detección temprana de defectos, la verificación del cumplimiento de los requisitos funcionales y la generación de reportes de calidad que respaldan las decisiones de lanzamiento.
Release Manager	Supervisa y coordina todo el proceso de liberación de versiones, asegurando que las actualizaciones se realicen de manera controlada, segura y documentada. Es el responsable de validar que las pruebas de calidad hayan sido satisfactorias, autorizar los despliegues productivos y garantizar la trazabilidad de cada versión liberada. También gestiona los cronogramas de actualización, la comunicación entre equipos y la planificación de los despliegues graduales o canary releases.
Field Support / Técnico	Se encarga de monitorear el funcionamiento de los dispositivos en producción, analizar métricas de desempeño y gestionar los incidentes que puedan presentarse tras el despliegue. En casos excepcionales en los que un dispositivo requiera intervención física, coordina las tareas de diagnóstico y reparación. Asimismo, proporciona retroalimentación al equipo de desarrollo y a QA sobre posibles mejoras, condiciones reales de operación y problemas recurrentes observados en los sensores. Su función permite cerrar el ciclo de mejora continua al aportar información valiosa del entorno operativo real.

6. Proceso de despliegue de productos

Etapas:

1. **Integración continua:** compilación automática del firmware y ejecución de pruebas unitarias por commit.
2. **Verificación en entorno controlado:** pruebas sobre simuladores y hardware real en laboratorio.
3. **Despliegue canary:** actualización OTA a un subconjunto de sensores.
4. **Monitoreo y análisis:** evaluación de métricas (uptime, errores, telemetría).
5. **Despliegue masivo:** una vez validada la versión, actualización del resto de dispositivos.
6. **Rollback (si aplica):** activación automática de versión anterior ante falla detectada.

Este proceso se integra en un pipeline CI/CD controlado por GitLab CI o GitHub Actions, conectado al servidor OTA.

7. Relación SCM – Despliegue

SCM (Software Configuration Management) asegura la integridad, trazabilidad y control de versiones del firmware y del backend.
En este contexto:

- Cada firmware se asocia a una versión controlada en Git (tag).
- Los metadatos del firmware (hash, fecha, dispositivo) se registran en la base de datos de despliegue.
- Permite reproducir el estado exacto de un dispositivo en cualquier momento.
- Facilita auditorías y comparaciones entre versiones.

Beneficio clave: unifica desarrollo, pruebas y despliegue bajo un mismo control de configuración.

8. Relación DevOps – Despliegue

El enfoque DevOps es fundamental para automatizar y estandarizar el proceso.

- **CI/CD:** automatiza compilación, pruebas, empaquetado y publicación de firmware.
- **Infraestructura como código:** permite configurar entornos OTA y backend con herramientas como Terraform o Ansible.
- **Monitoreo y feedback:** integra métricas de rendimiento de los dispositivos al ciclo de mejora continua.

Resultado: ciclos de actualización más cortos, menos errores manuales, mayor fiabilidad del producto.

9. Prácticas continuas y aseguramiento de calidad

- **Integración continua:** cada cambio se valida automáticamente.
- **Entrega continua:** versiones estables disponibles para despliegue inmediato.
- **Monitoreo continuo:** recolección de datos operativos de sensores (logs, alertas).
- **Testing automatizado:** pruebas de regresión y validación en entorno virtual.

Estas prácticas aseguran la calidad del producto antes, durante y después del despliegue.

10. Seguridad en el despliegue

La seguridad es crítica en el entorno IoT. Se aplican las siguientes medidas:

- **Firmado digital de firmware** para validar autenticidad.
- **Comunicación cifrada (TLS)** durante descargas OTA.
- **Certificados X.509** para autenticación mutua.
- **Verificación de integridad (hash, checksum)** post-descarga.
- **Registros inmutables y trazables** en el servidor OTA.
- **Control de acceso y permisos por rol** en la plataforma de gestión.

11. Conclusión

La propuesta combina prácticas de **DevOps, SCM y aseguramiento de calidad** para crear un proceso de despliegue seguro y resiliente, adaptado a las limitaciones del entorno IoT. El uso de estrategias OTA con mecanismos de rollback, junto a una infraestructura de CI/CD automatizada y monitoreo continuo, garantiza la **fiabilidad operativa y la mejora constante** del sistema, reduciendo costos y riesgos.