UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL CÓRDOBA

INGENIERÍA EN SISTEMAS DE INFORMACIÓN

Informe técnico-científico:

“Plan de despliegue de firmware para red de sensores IoT en plantas industriales”

Ingeniería y Calidad de Software

Docentes:

- Silvia Judith Meles
- Laura Inés Covaro
- María Cecilia Massano
- Constanza Garnero
- Ezequiel Gustavo Izaguirre

Curso 4K3

Año 2025 - 2º Cuatrimestre

Integrantes del grupo

- 89639 - Virinni, Bruno
- 90263 - Höhlke, Augusto
- 91274 - Liendo, Juan Esteban
- 89767 - Chaile, Emmanuel Ricardo
- 75721 - Freytes Oviedo, Agustin
- 91429 - Castro Monzon, Martín
- 76860 - Silvestri, Brian
- 88618 - Barrionuevo, Daniel
- 90297 - Cornejo, Francisco

ÍNDICE

Introducción

En el contexto industrial actual, la proliferación de dispositivos de Internet de las Cosas (IoT) ha permitido monitorear y controlar parámetros críticos como temperatura, humedad, presión, vibración y consumo energético en plantas que están ampliamente distribuidas geográficamente.

Sin embargo, cuando estos dispositivos requieren actualización de firmware (por ejemplo, para corregir bugs o mejorar algoritmos de medición), el despliegue de software (o firmware) se convierte en un proceso crítico. Una actualización mal ejecutada puede inutilizar el dispositivo (“brickearlo”). En entornos industriales con conectividad limitada y alto coste de intervención, esto representa un riesgo significativo, además del costo que implica perder estos “ojos” transitoriamente.

Este informe aborda las consideraciones técnicas, de procesos y de infraestructura para diseñar un plan de despliegue que sea seguro, eficiente y con mínimo impacto, para una empresa que provee sensores IoT de temperatura/humedad distribuidos en distintas fábricas del país, con conectividad a internet muy limitada o incluso solo redes locales cerradas.

Contexto

En Argentina, la adopción de soluciones IoT industriales se encuentra en una etapa de expansión, impulsada por sectores como agroindustria, energía, manufactura y logística. Sin embargo, las condiciones de conectividad y la disponibilidad de infraestructura varían significativamente entre regiones.
En grandes polos industriales (como Córdoba, Rosario o el Gran Buenos Aires) existen redes estables, servicios de fibra óptica y data centers locales que permiten gestionar dispositivos conectados mediante plataformas en la nube. En cambio, en plantas más aisladas —por ejemplo, en el norte o sur del país— la conectividad depende de redes 4G intermitentes o incluso enlaces satelitales, lo que obliga a diseñar despliegues resilientes y capaces de operar en modo offline temporal.
A nivel nacional, existen proveedores relevantes de infraestructura IoT como **Telecom Argentina (IoT Hub, red NB-IoT)**, **Claro Empresas**, y **Arsat**, que ofrecen conectividad M2M y servicios de nube híbrida, así como integradores de hardware locales que utilizan módulos ESP32, LoRaWAN y gateways industriales compatibles con protocolos **MQTT** o **Modbus TCP/IP**.
En este contexto, el desafío no es sólo técnico sino también logístico: garantizar que los dispositivos puedan actualizarse de forma remota, segura y escalable, minimizando los costos de desplazamiento físico del personal técnico.

Tipos de productos de software y su relación con el despliegue

El plan de despliegue debe diferenciarse según la criticidad y el entorno de destino del producto de software a actualizar. Los productos de software (o firmware) que pueden requerir despliegue en este escenario incluyen:

- **Firmware embebido en el dispositivo IoT**: el software de base que controla el sensor, comunicaciones, algoritmos de medición.

- **Firmware de comunicaciones o módulo connectivity**: por ejemplo si el sensor incluye un módulo de red LoRa/WiFi/BT que puede tener su propio firmware.

- **Software de backend o nube**: que gestiona la telemetría, la agregación de datos, la gestión de dispositivos (device management).

- **Aplicaciones de gestión / consola de administración**: para los técnicos o para la empresa que provee los dispositivos y gestiona las actualizaciones.

Para cada tipo de producto, el despliegue tiene características distintas:

- El firmware embebido es el más crítico en este escenario (riesgo de brickeo, conectividad limitada).

- El firmware de comunicaciones puede requerir actualizaciones más frecuentes, pero tal vez con menor riesgo de brickeo, aunque el impacto sobre la conectividad puede afectar a todos los dispositivos.

- El software de backend o consola se despliega en entornos centrales (data center, cloud) y su riesgo de brickeo de dispositivo es bajo, pero debe interactuar correctamente con los dispositivos remotos.
  Por tanto, el plan de despliegue debe diferenciarse según el tipo de producto de software a desplegar, sus dependencias, su criticidad y su entorno de destino.

Estrategias de despliegue de productos de software. Tecnologías y herramientas involucradas

La selección de la estrategia de despliegue es vital para minimizar el riesgo en entornos de conectividad limitada. Algunas de las estrategias de despliegue en uso actualmente(y sus tecnologías asociadas) aplicables al contexto:

- **Despliegue Big-Bang**: Añadir la nueva versión a todos los dispositivos en un solo momento. No es recomendable por el altísimo riego cuando la conectividad es limitada.
- **Despliegue por fase / por lote (canary, pilot)**: Primero actualizar un pequeño subconjunto de dispositivos, monitorizar, luego ampliar al resto. Esta estrategia se alinea con buenas prácticas de despliegue de SW para minimizar riesgos.
- **Blue-Green o Rolling Update**: Por ejemplo en backend/cloud, tener dos versiones, cambiar el tráfico a la nueva versión gradualmente. En IoT embebido puede adaptarse mediante versiones duales de firmware (A/B) para permitir rollback rápido.
- **Incremental / Delta Update**: En IoT con conectividad limitada, es útil enviar solo diferencias (deltas) de firmware para reducir tamaño, tiempo y coste de transferencia (por ejemplo para dispositivos con energía limitada).
- **Automatización de pipeline de despliegue (CI/CD / DevOps)**: Automatizar la compilación del firmware, pruebas (unitarias, integración, en emulación), empaquetado, despliegue OTA, monitoreo. Herramientas: Git (control de versiones), Jenkins/ GitHub Actions / GitLab CI, plataformas de device management (e.g., AWS IoT Device Management, Azure IoT Hub) para IoT. En el contexto IoT, la gestión de firmware y actualizaciones es una parte clave del device-lifecycle management.
- **Herramientas de gestión de configuración (SCM/CM), infraestructura como código (IaC), contenedores (en backend)**: Para los componentes centrales de software, despliegue mediante contenedores Docker/Kubernetes, IaC (Terraform/CloudFormation) y monitorización.
- **Seguridad en el despliegue (firmware signing, secure boot, rollback safe)**: En IoT es especialmente crítica.

En este escenario industrial IoT, algunas tecnologías específicas pueden emplearse: protocolo de actualización OTA (Over The Air) adaptado a conectividad limitada, módulos de backup dual firmware (A/B), canal seguro de transporte de firmware (TLS, autenticación de dispositivo), mecanismos de checkpoint/resume para actualizaciones en dispositivos con energía o conectividad intermitente.

Roles involucrados en el despliegue de productos

Para diseñar y ejecutar un plan de despliegue robusto es necesario identificar los roles y sus responsabilidades. En este escenario podemos definir los siguientes roles:

- **Ingeniero de firmware**: desarrolla el firmware embebido, crea paquetes de actualización, define versiones anteriores de rollback.

- **Ingeniero de comunicaciones/IoT**: se asegura de que la conectividad, el módulo radio/loRa/WiFi, el mecanismo OTA, soportan la actualización en el entorno con conectividad limitada.

- **DevOps / Infraestructura**: configura los pipelines CI/CD, las herramientas de automatización, la infraestructura de backend y el servicio de gestión de dispositivos, monitorización, alerta.

- **Ingeniero de QA / Testing**: realiza pruebas unitarias, integración, pruebas en emulación o hardware real, pilotaje de despliegue en lote reducido.

- **Ingeniero de seguridad**: evalúa y certifica el mecanismo de actualización, signing del firmware, secure boot, rollback seguro, gestión de claves.

- **Gestor de lanzamiento (Release Manager)**: planifica el despliegue, coordina comunicación con stakeholders, define horario, apruebas, rollback plan.

- **Técnicos de soporte de campo**: en caso de falla del despliegue, serían los que intervienen físicamente (aunque se busca minimizarlos).

- **Stakeholders planta industrial (operaciones/fábrica)**: personal de planta que debe estar informado, coordinar ventana de mantenimiento, validar que no se interrumpa la operación crítica.

Procesos para despliegue de productos

El proceso de despliegue para este escenario puede estructurarse en fases, cada una con actividades definidas:

1. ***Planificación del Release***

    - Inventario de dispositivos (modelo, versión firmware actual, conectividad, ubicación).
    - Definición del artefacto que se va a desplegar (versión nueva) y la versión anterior de rollback.
    - Identificación del entorno de destino (cada planta, red local cerrada, dispositivos conectados una vez al día).
    - Identificación de dependencias, prerrequisitos (por ejemplo: batería suficiente en sensores, disponibilidad de red, backup de datos local).
    - Selección de horario de despliegue (ventana de bajo impacto, fuera de horas pico, minimizando interrupción planta).
    - Definición de estrategia concreta de despliegue (por lote/canary, delta update, etc.).
    - Comunicación y aprobación por partes interesadas.
    - Definición del plan de rollback/contingencia.
    - Preparación de monitoreo/herramientas de seguimiento post-despliegue.

2. ***Preparación del artefacto***

    - Construcción del firmware (versión X.Y.Z) en el pipeline de CI/CD.
    - Versionado del artefacto, firma digital del firmware, creación del paquete de actualización (posiblemente delta).
    - Pruebas de laboratorio/hardware de referencia.

- Creación de documentación de rollback.

3. ***Pilotaje / despliegue por lote***
    - Selección de un pequeño grupo de dispositivos representativos (por ejemplo en una planta con buena conectividad).
    - Despliegue de la versión nueva.
    - Monitorización, validación de métricas (temperatura/humedad funcionando, dispositivo conectándose, sin bricks).
    - Si todo está correcto, programar despliegue al resto del parque.
4. ***Despliegue general***
    - Según la estrategia: por lotes, escalado progresivo.
    - En cada lote, se verifica instalación, conectividad, funcionamiento.
    - Monitorización en tiempo real de éxito/fallo.
    - Comunicación regular de status a stakeholders.
5. ***Post-despliegue y cierre***
    - Validaciones post-despliegue (verificación de sensores, logs, alertas de error, porcentaje actualizado).
    - Monitoreo continuo de rendimiento del firmware (mediciones consistentes, estabilidad).
    - Si se detecta fallo significativo: activar plan de rollback en los dispositivos afectados.
    - Lecciones aprendidas, documentación de la operación y métricas de éxito.

Relación SCM – Despliegue de Productos

La gestión de la configuración del software (SCM – Software Configuration Management) es fundamental para un despliegue controlado (reproducible, confiable y auditable). En este contexto:

- El firmware fuente debe mantenerse en un sistema de control de versiones (por ejemplo Git), con ramas de desarrollo, pruebas, producción.
- Cada artefacto desplegable debe estar etiquetado (tag) con un número de versión que siga un esquema de versionado semántico (Semantic Versioning), por ejemplo: v2.1.0. Este sistema utiliza tres componentes numéricos:
    - MAJOR (v2) → se incrementa cuando hay cambios incompatibles con versiones anteriores, por ejemplo una reestructuración del firmware que modifica la forma en que el dispositivo interpreta los datos.
    - MINOR (v2.1) → se incrementa cuando se agregan nuevas funcionalidades o mejoras que no rompen la compatibilidad con versiones previas, como un nuevo algoritmo de calibración o soporte para un sensor adicional.
    - PATCH (v2.1.0 → v2.1.1) → se incrementa cuando se aplican correcciones menores o bug fixes sin alterar las funcionalidades existentes, como un ajuste en los límites de temperatura.

  En la práctica, cada compilación del firmware genera un artefacto binario firmado y asociado a una etiqueta, que identifica exactamente el commit del código fuente desde el cual fue construido. Este tag se almacena en el sistema de control de versiones (SCM) y se replica en la base de metadatos del servidor de despliegue OTA.
- Los historiales permiten conocer exactamente la versión anterior que queda en el campo (importante para rollback).
- Todas las dependencias, binarios, scripts de despliegue deben estar versionados.
- La trazabilidad permite que, ante un incidente, se identifique cuál versión se desplegó en qué dispositivos, cuándo y con qué resultados.
- La integración con el pipeline CI/CD permite automatizar la construcción, las pruebas y el empaquetado del artefacto.
  Por tanto, SCM es la base para un despliegue reproducible, confiable, y auditable.

Relación DevOps – Despliegue de Productos

El enfoque **DevOps** (integración desarrollo‑operaciones) es clave para reducir el tiempo de entrega y mejorar la calidad, promoviendo la **responsabilidad compartida** sobre el funcionamiento del sistema en producción. En IoT, esto implica que los equipos de firmware, comunicaciones, *backend* y soporte de campo trabajan coordinados, automatizando el *pipeline* de construcción, pruebas y despliegue

En este escenario IoT:

- Los equipos de firmware/IoT trabajan junto con operaciones/infraestructura para automatizar el pipeline de construcción, pruebas, despliegue.

- Monitorización y feedback continuo permiten cerrar el bucle (build → test → deploy → monitor → feedback).

- El despliegue frecuente (aunque en IoT usualmente más espaciado) favorece el aprendizaje, reducción de riesgos y mejora continua. martinfowler.com+1

- a cultura DevOps promueve la responsabilidad compartida sobre el despliegue y el funcionamiento del sistema en producción, eliminando la antigua separación entre desarrollo y operaciones, donde el equipo de desarrollo “entregaba el software y se desentendía de su ejecución”.

- En IoT, eso significa que firmware, comunicaciones, backend, soporte de campo trabajan coordinados, validando la actualización, monitorizando los dispositivos, gestionando incidencias.

Relación Prácticas Continuas – Despliegue de Productos

Las prácticas continuas engloban: integración continua (CI), entrega continua (CD) y despliegue continuo (Continuous Deployment). Aunque en IoT el despliegue continuo al 100% puede no ser viable por limitaciones de conectividad, muchas de sus prácticas son aplicables:

- Integración continua: cada cambio de firmware se mergea, se construye automáticamente, se prueba.

- Entrega continua: el artefacto está listo para desplegarse en cualquier momento.

- Despliegue continuo (o casi): una vez validado, se despliega automáticamente a los dispositivos, o al menos via un pipeline semi-automatizado.
  Estas prácticas permiten que los deploys sean rápidos, repetibles, con errores mínimos, y mayor grado de automatización, lo cual reduce el riesgo de intervenciones manuales costosas — precisamente relevante para dispositivos IoT distribuidos.

Seguridad en el despliegue de productos

La actualización de firmware en dispositivos IoT con conectividad limitada y ubicados en fábricas exige un enfoque de seguridad riguroso centrado en la Cadena de Confianza (*Chain of Trust*):

- Autenticación del dispositivo y del servidor de despliegue: asegurar que sólo dispositivos autorizados reciban firmware y que sólo firmware de origen autorizado se aplique.

- Firma digital del firmware: asegurar integridad y origen del paquete de actualización.

- Mecanismos de rollback seguro: en caso de fallo, poder revertir a versión estable sin comprometer el dispositivo.

- Encriptación del canal de actualización (TLS, etc.) y almacenamiento seguro del firmware en el dispositivo.

- Modo fall-safe: el dispositivo debería tener dos particiones A/B de firmware, de modo que si la nueva versión falla, pueda volver a la anterior.

- Gestión de claves y credenciales: cada dispositivo tiene identidad única, gestión de claves, no usar claves compartidas.

- Registro y auditoría de despliegues: logs de qué dispositivos recibieron qué versión, cuándo, resultado, errores.

- Verificación posterior a la actualización: asegurar que no se introdujo vulnerabilidad con la nueva versión.

Plan de Release específico para el escenario

A continuación, se presenta un plan de release adaptado al escenario de la red de sensores IoT en plantas industriales.

Entorno de destino:

- Cada fábrica (planta industrial) donde se encuentran sensores de temperatura/humedad instalados.
- Algunos dispositivos conectan a Internet directamente (una vez al día), otros solo a redes locales cerradas (puede que tengan un gateway que luego conecta a internet).
- También se considera el backend de gestión de dispositivos y consola central en la nube o infraestructura privada.

Tipo de despliegue:

- Firmware embebido de sensores (versión 1.2.0 → 1.3.0) que corrige bugs de medición y mejora el algoritmo de humedad.
- Estrategia de despliegue: por lote (canary) + incremental/delta update.

Tipo de infraestructura requerida:

- Infraestructura de backend en la nube o centro de datos para alojar el servicio de device management (por ejemplo, servidor de firmware, autenticación, tracking).
- Infraestructura de gateway en cada planta si existe red local cerrada (dispositivo que recibe actualizaciones y las retransmite a sensores).
- Pipeline CI/CD para firmware, herramientas de empaquetado, firma de firmware.
- Sistema de monitorización de dispositivos en campo (telemetría, logs).
- Almacenamiento seguro de versiones antiguas (para rollback).
- Servidor de distribución de firmware adaptado para conectividad limitada (capacidad de reanudar, particionar paquetes, delta updates).

En el contexto argentino, la infraestructura necesaria para un despliegue IoT industrial robusto puede construirse combinando tecnologías accesibles y servicios disponibles localmente:

Infraestructura de red:

- Redes 4G y NB-IoT provistas por operadores nacionales (Telecom, Claro, Movistar).
- Alternativamente, redes **LoRaWAN privadas o híbridas**, muy utilizadas en entornos fabriles y zonas rurales por su bajo consumo energético.
- En lugares sin cobertura, se puede usar conectividad satelital de bajo costo (Starlink, Arsat SAT).

Infraestructura de backend:

- Nubes locales o híbridas: **AWS Argentina (Buenos Aires edge region)**, **Google Cloud Latam**, o incluso **Arsat Data Center (Benavídez)** para cumplir con requisitos de soberanía de datos.
- Servicios de gestión de dispositivos (IoT Core, Azure IoT Hub, o soluciones open source como **ThingsBoard**, **EMQX**, o **KaaloT**).

Infraestructura física en planta:

- Gateways industriales (Raspberry Pi, Advantech, Siemens IoT2040) configurados para actuar como relay local y buffer temporal ante desconexiones.
- UPS y sistemas redundantes para asegurar disponibilidad 24x7.

Infraestructura de desarrollo y despliegue:

- Pipeline CI/CD implementado en **GitLab CI o Jenkins**, ejecutando tests en entornos emulados o hardware local.
- Almacenamiento en repositorios Git con versionado y control de integridad.
- Monitoreo mediante **Grafana + Prometheus**, y alertas integradas con **Telegram o Slack**.

Este enfoque combina herramientas globales y tecnologías locales, garantizando un equilibrio entre disponibilidad, costo y seguridad en el despliegue.

Artefacto que se despliega:

- Firmware para dispositivo sensor modelo “IoT-SensTempHumi-v2” versión 1.3.0.
- Versión anterior de la que se puede volver: 1.2.0.
- Identificación exacta: paquete “IoT-SensTempHumi-v2-FW-1.3.0.bin”, firmado digitalmente.

Cómo se realiza el despliegue e instalación del artefacto en el entorno de destino:

- El backend anuncia la versión 1.3.0 disponible.
- Los dispositivos se conectan a la red local o internet, comprueban versión, descargan delta del paquete (por ejemplo diferencia entre 1.2.0 → 1.3.0).
- El dispositivo guarda el nuevo firmware en partición B (si arquitectura A/B).
- Verifica firma digital.
- Cambia el boot loader para arrancar desde partición B en la próxima sesión, mientras mantiene la versión A como fallback.
- Reinicia dispositivo, monitoriza que arranca correctamente, comunica al backend “update success”.
- Si arranca mal o no se comunica en un tiempo definido, se arranca desde versión A (rollback automático).
- En caso de gateway-planta cerrado, el gateway descarga el paquete y retransmite a sensores en la red local, o se agenda la actualización para ventana programada.

Dependencias:

- Conectividad del dispositivo (aunque sea una vez al día).
- Suficiente batería/energía del dispositivo para realizar la actualización.
- Versión de boot loader compatible con actualización A/B.

- Backend de device management operativo.
- Certificados de firma digital instalados correctamente en el dispositivo.
- Gateway (si aplica) actualizado para retransmitir correctamente.

Prerequisitos que deben estar presentes antes del despliegue:

- Inventario actualizado del parque de dispositivos (modelo, versión actual, conectividad, ubicación).
- Test de compatibilidad del firmware en laboratorio y en entorno representativo.
- Confirmación de que los dispositivos tienen espacio libre suficiente para nueva partición firmware.
- Confirmación de que la ventana de mantenimiento es aceptable para la planta.
- Aprobación del patrocinador del cambio (operaciones planta).
- Copia de seguridad de los datos críticos locales si aplica.
- Plan de monitorización y alerta configurado.

Horario de despliegue: Momento elegido y justificación:

- Se eligió la ventana de **domingo de 02:00 a 06:00 (hora planta local)** para minimizar impacto en producción (cuando planta en modo de mantenimiento).
- Justificación: conectividad suele más estable, personal de soporte de campo disponible, menor actividad de medición crítica.

Forma de despliegue (técnica):

- Delta OTA update sobre red existente (internet o gateway local).
- En dispositivos con muy poca conectividad o solo local: se agrupan en lote y actualización programada a través del gateway con almacenamiento local del paquete.
- A/B firmware particionado con arranque desde nueva versión y fallback automático.

Estrategia concreta de despliegue para reducir riesgos:

- Primero pilotaje en 10 % de los dispositivos (una planta de baja criticidad) para validar comportamiento.
- Monitor de éxito/fallo por 24 horas.
- Si éxito > 99% sin bricks, se amplía al 50 % de dispositivos (otras plantas).
- Finalmente al 100 % restante en dos fases más.
- Mantener versión anterior 1.2.0 como fallback activo.
- Monitoreo de indicadores clave (tiempo de arranque, tasa de error, datos de sensor coherentes).
- En caso de fallo en lote, suspender ampliación y ejecutar rollback inmediato en ese lote.

Monitoreo:

- Backend registra: dispositivo, versión anterior, versión nueva, hora de actualización, resultado (exitoso/fallido).
- Métricas agregadas: porcentaje actualizado, tiempo medio de actualización, tasa de errores, tasa de dispositivos que no se conectan tras actualización.
- Alertas configuradas: más de 1% de fallos en lote, dispositivos que no se reportan en > 4 horas post-arranque.
- Logs de firmware en dispositivo (si hardware lo soporta) almacenados local/servidor para diagnóstico.

Pruebas post-despliegue: Validaciones mínimas que confirman que la aplicación funciona después del despliegue:

- Verificar que el dispositivo arranque con la versión 1.3.0 y reporte al backend “status OK”.

- Verificar que la medición de temperatura y humedad está dentro de los rangos esperados (por ejemplo comparando con una referencia).
- Verificar que el dispositivo se conecta (o intentará conectarse) según programación (una vez al día).
- Verificar que no se han incrementado los registros de error/exceptión en sesión tras actualización.
- Verificar que los logs muestran que el boot loader cambió correctamente a la partición B y que la versión anterior está disponible en A.
- Verificar que los datos históricos no se han perdido o corrupto.

Comunicación y aprobaciones: A quién se avisa antes y después del despliegue y quién debe aprobarlo:

- Antes del despliegue:
    - Aprobación del gerente de operaciones planta.
    - Comunicación al equipo de soporte de campo (técnicos de mantenimiento).
    - Comunicación al equipo de DevOps/Infraestructura.
    - Notificación a los responsables de red en cada planta.
- Después del despliegue:
    - Notificación al gerente de operaciones de que el lote piloto se completó con éxito.
    - Informes de estado al equipo de dirección (porcentaje actualizado, fallos, incidentes).
    - Comunicación a todos los clientes/fábricas de que la versión 1.3.0 se encuentra desplegada / se está desplegando.

Plan de rollback / contingencia: Qué pasos concretos se siguen si el despliegue falla:

- Si un dispositivo no arranca correctamente tras la actualización y no se conecta dentro de 4 horas: el boot loader detecta fallo → arranque automático desde partición A (versión 1.2.0) (rollback automático).

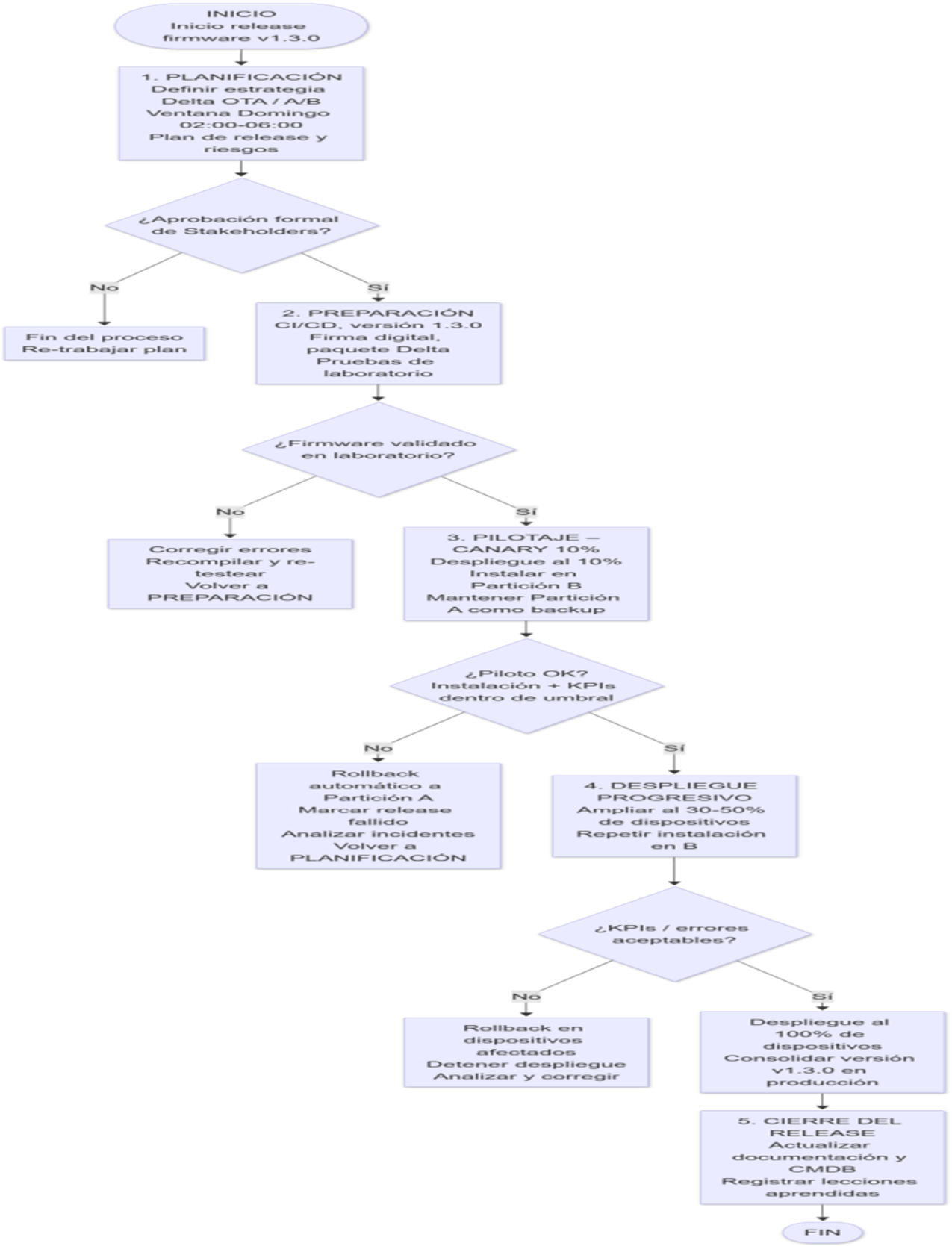
- Si más de 1% de los dispositivos del lote piloto presentan fallo, se suspende ampliación y se analiza causa (hardware, conectividad, firmware).

- En backend se marca versión 1.3.0 como “retirada temporal” y se bloquea su distribución. Se notifica al equipo de firmware y se activa plan de análisis de fallo.

- Si se detecta un bug crítico en la versión 1.3.0, se puede generar una versión 1.2.1 de emergencia y desplegar rápidamente al conjunto afectado, utilizando la misma estrategia por lote.

- En el peor caso donde dispositivos quedan inoperativos (bricked), se activa protocolo de soporte de campo: técnico viaja a planta, reemplazo de hardware, registro de incidente, análisis post-mortem.

Diagrama de Flujo del Plan de Release



Conclusiones

La implementación de un plan de despliegue de firmware para una red de sensores IoT industriales en Argentina requiere conjugar ingeniería, procesos y contexto local.

Este trabajo demuestra que, mediante estrategias escalonadas (canary, delta update, OTA), infraestructura híbrida (nube–planta), y herramientas accesibles (CI/CD, control de versiones, monitoreo), es posible realizar actualizaciones remotas seguras incluso con conectividad limitada.

La adopción de buenas prácticas DevOps y SCM garantiza trazabilidad, auditabilidad y reducción del tiempo medio de recuperación ante fallas.

Además, la infraestructura tecnológica disponible en el país —con avances en redes NB-IoT, data centers nacionales y plataformas IoT abiertas— permite escalar proyectos de este tipo de manera sostenible y con soberanía tecnológica.

Finalmente, el despliegue planificado y seguro del firmware no sólo mejora la eficiencia operativa de las plantas industriales, sino que fortalece la resiliencia digital y la competitividad de la industria nacional frente a los desafíos de la transformación 4.0.

Bibliografía

- Humble, J., & Farley, D. (2010). *Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation*. Addison-Wesley. martinfowler.com

- Servile, V. (2024). *Continuous Deployment*. O'Reilly Media. O'Reilly Media

- "15 Most Important Software Deployment Best Practices". Zeet. (2023). Recuperado de https://zeet.co/blog/software-development-best-practices zeet.co

- "Why IoT Firmware Update Matters: Expert Tips and Strategies". Relevant Software. (2024, July 31). Recuperado de https://relevant.software/blog/iot-firmware-update/ relevant.software

- "Over-the-Air Software Updates in the Internet of Things: An Overview of Key Principles". (2020). *ResearchGate*. ResearchGate

- Graglia, I. (2024, December 23). "IoT Update Management: Stay Ahead of Security Risks". IT Pulse. Recuperado de https://blog.invgate.com/iot-update-management blog.invgate.com

- Stormotion. (2025, March 11). "Best Practices for Secure and Efficient IoT Firmware Updates". Recuperado de https://stormotion.io/blog/updating-iot-devices/ stormotion.io

- Bakhshi, T. (2024). "A Review of IoT Firmware Vulnerabilities and Auditing". *Sensors*, 24(2), 708. MDPI
- Energy-aware Incremental OTA Update for Flash-based Batteryless IoT Devices