



UNIVERSIDAD TECNICA FEDERICO SANTA MARIA

Memoria de Título

Implementación de un Digital Twin para mantenimiento predictivo sobre un banco de pruebas para la fatiga de resortes.

Memoria para optar al grado/título de
Ingeniero Civil Electrónico

Alumno
Benjamín Asmed Alarcón Jalife

Profesor Supervisor
Dr. Juan Carlos Agüero

Revisores/Comisión Co-Referente
César Silva Jiménez
Juan Carlos Andresen

06, 2026, Valparaíso, Chile



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: Implementación de un Digital Twin para mantenimiento predictivo sobre un banco de pruebas para la fatiga de resortes.

Nombre del candidato(a): Benjamín Asmed Alarcón
Jalife

Carrera / Grado: Ingeniería Civil Electrónica

Campus: Casa Central Departamento: Electrónica

2.- VALIDACIÓN DEL PROFESOR GUÍA/DIRECTOR DE TESIS

Yo, Juan Carlos Agüero, 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 (marcar una opción)

☒ 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 (marcar una opción):

☐ 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: 30/09/2026

Firma:

Estudiante o Candidato(a):

Fecha: 10/09/2026

Firma:



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.

*"To see the world, things dangerous to come to, to see behind walls, draw
closer, to find each other, and to feel. That is the purpose of life."
- LIFE Magazine Motto, The Secret Life of Walter Mitty (2013)*

Dedicado a todas las personas que han formado y forman parte de mi vida.

AGRADECIMIENTOS

QUIERO agradecer a mi madre Claudia, por entenderme, aconsejarme y tranquilizarme en todo momento, querer darme todo y más, a mi padre Iván, por enseñarme como poder afrontar las adversidades y siempre saber que hacer cuando yo no tenía idea, y a ambos por trabajar más allá del cansancio para darme la educación, hogar, comida y cualquier cosa que pude haber necesitado para llegar a donde estoy, a mi hermana Fernanda, por escuchar mis dificultades y apoyarme incondicionalmente, por enseñarme todo lo que me gusta hacer y regalóarme, y a los 3 por darme apoyo en todo aspecto posible a lo largo de mi vida y amarme sin esperar nada a cambio, sin ello nada de lo que he hecho sería posible; sin olvidarme de mis 2 mascotas presentes a lo largo de mi vida universitaria , "Patitas" que en paz descanse y "Orejitas" que sigue dándonos todo el amor posible. A mis amigos a lo largo de mi vida, he tenido la suerte de encontrar unos muy buenos, donde espero que sean para toda la vida. A mi profesor Juan Carlos Agüero, por apoyarme en mis proyectos y en la universidad, a Patricio, por llevarme al extranjero a experimentar la vida laboral y abrirme las puertas a una nueva realidad, creando recuerdos inolvidables. A mi supervisor Juan Carlos Andresen, por darme la oportunidad de estar en su equipo, aconsejándome y enseñándome, además de colaborar en cada paso para la ejecución de este trabajo. A Don Leonardo, por tratarme como su propia familia, dándome un espacio en su casa, trabajo y apoyo en lo que necesite.

Benjamín Asmed Alarcón Jalife

RESUMEN

LA ESTIMACIÓN de vida útil remanente mediante modelos de aprendizaje automático depende críticamente de la calidad de los conjuntos de datos de degradación con que esos modelos se entrenan. Los conjuntos disponibles públicamente son escasos, provienen en buena parte de simulación, y se distribuyen como series de señales sin registro del estado físico del espécimen ni verificación de que la solicitud aplicada haya sido la declarada. Esta memoria aborda ese problema construyendo un procedimiento experimental capaz de generar datos de degradación trazables y físicamente etiquetados, sobre un banco de ensayos de fatiga de resortes helicoidales de extensión.

El desarrollo se organiza en cuatro etapas sucesivas, siguiendo la taxonomía de Kritzinger *et al.*, que clasifica los sistemas según el grado de automatización del flujo de datos entre el objeto físico y su representación virtual. La Etapa A implementa el modelo físico del resorte a partir de leyes constitutivas documentadas (Hooke con precarga, endurecimiento de Hollomon, deformación permanente, daño de Basquin e histéresis), distinguiendo de forma explícita los parámetros medidos de los adoptados desde literatura. La Etapa B construye la arquitectura de adquisición y persistencia en *streaming* sobre MQTT, Redis Streams e InfluxDB, y la valida empleando el modelo físico como fuente sintética. La Etapa C reemplaza esa fuente por el banco de ensayos real, comunicado con la nube sobre un enlace autenticado, y calibra el modelo contra los resortes efectivamente ensayados, alcanzando el nivel de Sombra Digital.

La Etapa D implementa el flujo de retorno desde la representación virtual hacia el instrumento y con ello el nivel de Digital Twin. Se construyen dos lazos de control con propósitos distintos. El lazo de re-zero corrige la consigna de extensión para mantener constante el sobreesfuerzo plástico por ciclo, compensando la deriva que la propia deformación permanente introduce; sin esa corrección la condición experimental declarada deja de corresponder a la aplicada y el conjunto de datos resultante queda mal etiquetado por construcción. El lazo de guarda elástica introduce una excitación estocástica de Poisson

sobre la consigna, aprende del propio espécimen su recta elástica de referencia, y ordena revertir la carrera cuando la fuerza medida se aparta de esa referencia de forma sostenida. Como la secuencia de perturbaciones es sorteada por el banco, cada ciclo anómalo del conjunto de datos queda identificado por construcción.

La campaña experimental habilitada por el lazo cerrado recupera de forma empírica la relación entre el sobreesfuerzo plástico por ciclo y la tasa de degradación de rigidez, sobre tres niveles de sollicitación y con significancia estadística en cada uno. Se establece además que el límite elástico es una propiedad de cada resorte individual y no del lote, con una dispersión del orden del 20 % entre ejemplares nominalmente iguales, resultado que condiciona el diseño de todo mecanismo automático de protección y obliga a calibrar cada espécimen por separado. El trabajo delimita explícitamente el alcance de lo demostrado: las campañas se ejecutan en régimen de fatiga de bajo ciclo, con un ejemplar por nivel de sollicitación, y el entrenamiento de modelos de aprendizaje automático sobre los datos generados queda fuera de su alcance.

Palabras clave

Digital Twin; gemelo digital; mantenimiento predictivo; vida útil remanente (RUL); calidad de datos; conjuntos de datos de degradación; fatiga de bajo ciclo; resortes helicoidales; identificación de parámetros; control en lazo cerrado; proceso de Poisson; MQTT; series de tiempo.

ÍNDICE

AGRADECIMIENTOS	I
RESUMEN	II
ÍNDICE DE FIGURAS	VIII
ÍNDICE DE CUADROS	XI
ABREVIACIONES	XIII
SIMBOLOGÍA	XV
1. INTRODUCCIÓN	1
1.1. Motivación y planteamiento del problema	1
1.2. Estado del arte	2
1.2.1. Calidad de los datos en aprendizaje automático	2
1.2.2. Estimación de vida remanente y conjuntos de datos de degradación	3
1.2.3. Digital Twin: definición y taxonomía	3
1.2.4. Modelado de resortes helicoidales y fatiga	4
1.3. Contribuciones	4
1.4. Objetivos, alcances y limitaciones	5
1.4.1. Objetivo general	5
1.4.2. Objetivos específicos	6
1.4.3. Alcances	6
1.4.4. Limitaciones	6
1.4.5. Trabajo diseñado y no ejecutado	7
1.5. Organización del documento	7
2. ETAPA A: MODELADO FÍSICO DEL RESORTE	9
2.1. Introducción y objetivos de la etapa	9

2.2.	Fundamentos físicos implementados	10
2.2.1.	Ley de Hooke con precarga (zona elástica)	10
2.2.2.	Rigidez teórica del resorte helicoidal	10
2.2.3.	Endurecimiento por deformación (zona plástica)	11
2.2.4.	Deformación permanente (plastic set)	11
2.2.5.	Fatiga: ley de Basquin	11
2.2.6.	Histéresis mecánica	12
2.2.7.	Carácter determinista del modelo	12
2.3.	Material de los resortes ensayados	12
2.4.	Arquitectura del software de simulación	13
2.4.1.	Parámetros relevantes y su justificación	13
2.5.	Implementación: extractos de código relevantes	13
2.5.1.	Clase <code>Spring</code> (extracto de <code>physics.py</code>)	14
2.5.2.	Cálculo de rigidez teórica	15
2.6.	Resultados: dashboard de simulación	15
2.6.1.	Panel 1: Zonas del resorte (superior izquierdo)	16
2.6.2.	Panel 2: Lazo de histéresis (superior derecho)	17
2.6.3.	Panel 3: Multi-ciclo con degradación (inferior izquierdo)	17
2.6.4.	Panel 4: Catálogo de resortes (inferior derecho)	17
2.7.	Discusión sobre el modelo de zona plástica	17
3.	ETAPA B: ARQUITECTURA DE DATOS EN <i>STREAMING</i>	19
3.1.	Introducción y objetivos de la etapa	19
3.2.	Fundamentos del protocolo MQTT	20
3.2.1.	Concepto de topic	20
3.2.2.	Calidad de servicio (QoS)	20
3.3.	Entorno de desarrollo: máquina virtual Linux	22
3.3.1.	Configuración de la máquina virtual	22
3.3.2.	Instalación del broker Eclipse Mosquitto	22
3.4.	Verificación inicial del broker	23
3.5.	Formato de mensaje: señales en JSON	23
3.6.	Redis como buffer intermedio	24
3.6.1.	Función de Redis en la cadena de datos	24
3.6.2.	Redis Streams y consumer groups	25
3.7.	Persistencia en InfluxDB	26
3.8.	Visualización con Grafana	26
3.9.	Integración del modelo físico	27
3.10.	Contenerización con Docker	29
3.10.1.	Motivación	29

3.10.2. Arquitectura de contenedores	29
3.10.3. Operación	30
3.11. Verificación de tolerancia a fallos	30
3.12. Posición del sistema en la taxonomía del Digital Twin	31
3.13. Discusión	31
4. ETAPA C: BANCO FÍSICO INTEGRADO	33
4.1. Introducción y objetivos de la etapa	33
4.2. Despliegue del <i>stack</i> en la nube	34
4.3. Conectividad Raspberry Pi hacia nube	35
4.3.1. Trayecto físico del enlace	35
4.3.2. Seguridad del canal	35
4.4. Trazabilidad de experimentos	36
4.4.1. Nomenclatura de los resortes	37
4.5. Calibración del modelo con el resorte real	37
4.5.1. Fase de rampa (Stage 1)	38
4.5.2. Ajuste continuo por ciclo (Stage 2 y Stage 3)	39
4.5.3. Vigilancia del ensayo: <code>monitor_worker.py</code>	40
4.5.4. Parámetros del resorte real	40
4.5.5. Cada resorte tiene su propio límite elástico	41
4.6. Caracterización experimental de la degradación	42
4.6.1. Rigidez medida frente a rigidez predicha	42
4.6.2. Interpretación	44
4.6.3. Retorno a cero y compresión residual	45
4.7. Posición del sistema en la taxonomía del Digital Twin	45
4.8. Discusión	45
5. ETAPA D: LAZO CERRADO Y DIGITAL TWIN	47
5.1. Introducción y objetivos de la etapa	47
5.2. Fundamentos físicos y de control necesarios	48
5.2.1. Fatiga de bajo y de alto ciclo (LCF y HCF)	48
5.2.2. Deformación permanente y shakedown	49
5.2.3. Sobreesfuerzo por sobre x_e individual como variable de estandarización	50
5.2.4. Control por desplazamiento vs. control por fuerza	51
5.2.5. Excitación estocástica de la consigna	51
5.2.5.1. Necesidad de un evento impredecible	51
5.2.5.2. Elección de la distribución de la perturbación	52
5.2.5.3. Alcance del modelo estocástico	54
5.2.6. Marcador post-shakedown	54

5.2.7. Calibración individual desde el ciclo 1	54
5.3. Arquitectura final del sistema	55
5.4. Canales de retorno: protocolo	55
5.4.1. Canal de guarda	57
5.4.2. Canal de re-zero	57
5.5. Lazo de guarda elástica	58
5.5.1. Excitación: consigna con jitter de Poisson	59
5.5.2. Autocalibración de la referencia elástica	60
5.5.3. Criterio de disparo	62
5.5.4. Acción de retorno: <code>GUARD_CLAMP</code>	63
5.5.5. Estructura del lazo y constantes de sintonía	63
5.5.6. Verificación de detección sin hardware	65
5.5.7. Ejecución sobre hardware	65
5.5.8. Dos defectos de detección hallados sobre hardware	67
5.5.9. Alcance y limitaciones de la guarda	68
5.6. Lazo de re-zero y campaña LCF	69
5.6.1. Estructura del lazo	69
5.6.2. Formulación en variables de control	71
5.6.3. Diseño experimental de la campaña LCF	73
5.6.3.1. Pivote basado en datos: el techo de carrera como restricción	74
5.6.4. Ejecución y resultados	74
5.6.4.1. Tasa de degradación por nivel	74
5.6.4.2. Ley empírica tasa-vs-sobreesfuerzo	75
5.6.4.3. Consecuencia para el modelo físico	75
5.7. Resumen de decisiones de diseño	76
5.8. Posición del sistema en la taxonomía del Digital Twin	79
5.9. Discusión	80
6. CONCLUSIONES	81
6.1. Síntesis del trabajo realizado	81
6.2. Conclusiones	81
6.3. Trabajo futuro	84
BIBLIOGRAFÍA	86

Índice de figuras

2.1. Dashboard de la simulación física (Etapa A) para el resorte calibrado <code>real_7x38</code> .	16
2.2. Comparación de cuatro modelos para la zona plástica: (1) lineal, el empleado en este trabajo; (2) Hollomon con $n = 0.15$; (3) Ramberg-Osgood, con transición suave; (4) modelo con <i>plateau</i> de fluencia, característico de aceros bajo en carbono.	18
3.1. Arquitectura por capas de la cadena de datos de la Etapa B. El flujo es lineal y unidireccional desde la fuente hasta la visualización, sin ningún camino de retorno hacia el objeto representado, que es lo característico de un <i>Digital Shadow</i> . En esta etapa la fuente es todavía el modelo físico de la Etapa A; la Etapa C la reemplaza por el instrumento real sin alterar el resto de la cadena.	21
3.2. Terminales comunicándose mediante Mosquitto.	23
3.3. Verificación de la persistencia en InfluxDB mediante el Data Explorer. Se observan los registros del <i>measurement raw_signals</i> con su fuente, identificador de resorte y valores por campo.	26
3.4. Panel de Grafana con las tres señales del experimento (fuerza, voltaje y ciclo). La curva de fuerza muestra la degradación progresiva por fatiga a lo largo de 1000 ciclos.	27
3.5. Logs del servicio de escritura (<i>consumer</i>). Cada línea muestra el identificador del mensaje, el ciclo procesado, la fuerza y la confirmación (XACK) de escritura exitosa en InfluxDB.	28
3.6. Estado de los seis contenedores del sistema tras su despliegue. Los servicios de infraestructura exponen sus puertos; el <i>ingestor</i> y el <i>consumer</i> operan en la red interna.	30

- 4.1. Rigidez medida k_{measured} frente a rigidez predicha $k_{\text{predicted}}$ para un resorte 8x28.5mm ciclado a $D = 100$ mm. La curva predicha corresponde al twin en vivo sembrado con los parámetros de fatiga de literatura ($N_f = 5 \times 10^4$, $c = 0,5$). El descenso aislado en torno al ciclo 48 es un artefacto de un ciclo cuyo ajuste degradó puntualmente, no un evento mecánico. 43
- 5.1. Evolución del sobreesfuerzo plástico $\Delta = x_{\text{máx}} - x_e$ en función del número de ciclos, para las dos formas de conducir el ensayo. Con la consigna fija (azul), el *permanent set* acumulado hace que Δ decaiga hasta anularse, momento en que cesa la deformación plástica (*shakedown*). Con re-zeroing (rojo), la consigna se corrige ciclo a ciclo en la misma cantidad en que el resorte se alargó, con lo que Δ permanece en su valor inicial Δ_0 y el ensayo mantiene la solicitación que se propuso aplicar. 50
- 5.2. Arquitectura final del Digital Twin. En azul, el flujo de datos físico $\rightarrow$ virtual heredado de la Etapa C. En rojo, el nuevo flujo virtual $\rightarrow$ físico añadido en la Etapa D, en sus dos variantes: `twin_guard_worker.py` consume el *stream* de Redis muestra a muestra y publica el comando de guarda; `rezero_helper.py` lee del histórico en InfluxDB y publica la corrección de re-zero. Ambos usan topics MQTT dedicados y credenciales distintas de la del instrumento. 56
- 5.3. Consigna de extensión bajo jitter de Poisson (esquemático). Cada punto es la extensión comandada en un ciclo. La zona sombreada corresponde a los ciclos sobre los que el twin construye su referencia; el sorteo también opera en ellos, de modo que su limpieza depende de que λ sea pequeño (Sección 5.5.1). Los dos puntos situados por sobre x_e corresponden ambos a empujones de Poisson, y se distinguen por lo que el twin hace con ellos: el azul ocurre antes de que el detector termine de armarse, de modo que el resorte cruza a zona plástica sin que nadie reaccione; el rojo ocurre con el detector ya operativo y es el que provoca el disparo. La diferencia entre ambos no está en la excitación, que es idéntica, sino en el estado del vigilante. Como δ se dimensiona para que un solo empujón cruce el límite elástico, las consignas son esencialmente binarias: x_{nom} o por sobre x_e . Al detectar el ablandamiento, el twin emite `GUARD_CLAMP`; el sorteo de Poisson sigue ejecutándose, pero sus resultados quedan recortados a la cota D_{safe} 60
- 5.4. Estructura del lazo de guarda elástica. El rombo es el elemento de decisión: sólo cuando el residuo supera el umbral en tres muestras consecutivas se emite el comando. A diferencia de un lazo regulador clásico, no existe una consigna que deba seguirse con error nulo: es un lazo *disparado por evento*, cuya salida es una restricción permanente sobre el conjunto de consignas admisibles del banco. 64

- 5.5. Serie de fuerza registrada durante la corrida `GuardDemo3` (`spring_id = guard8x285`), reconstruida a partir de la exportación de la base de datos. Cada punto es el promedio de una ventana de 10 s, duración comparable a la del ciclo del banco, de modo que los máximos de la traza no son los picos instantáneos de cada ciclo. Se distinguen las cuatro fases descritas en el texto: operación nominal, la excursión de Poisson que alcanza 7,8 N, el colapso de la respuesta por debajo del nivel nominal, y la reaparición posterior de máximos elevados que el registro disponible no permite explicar. 66
- 5.6. Estructura del lazo de re-zero. El controlador integra la deriva de *permanent set* en D_{cmd} para mantener el sobreesfuerzo Δ igual a la referencia Δ^* . . . 69
- 5.7. Tasa de degradación de rigidez en función del sobreesfuerzo plástico por ciclo. Cada punto corresponde a la pendiente ajustada sobre la ventana post-shakedown de una corrida completa (Tabla 5.6), y la recta es el ajuste de (5.11). La tendencia es monótona y cada pendiente es individualmente significativa, pero con un solo ejemplar por nivel los puntos no llevan barra de dispersión entre resortes: la incertidumbre representada por la separación entre punto y recta no incluye la variabilidad resorte a resorte, que es precisamente la que la campaña no pudo estimar. 76

Índice de cuadros

2.1. Propiedades del acero al cromo-vanadio (literatura estándar).	13
2.2. Parámetros del modelo, su origen y nivel de confianza.	14
3.1. Niveles de QoS en MQTT.	21
3.2. Parámetros de configuración de la máquina virtual.	22
3.3. Estructura del mensaje JSON de cada señal.	24
3.4. Comandos de Redis Stream empleados en el sistema.	25
3.5. Evolución de la fuerza, deformación permanente y daño por fatiga generados por el modelo físico (resorte real_7x38 , extensión 70 mm).	29
3.6. Servicios definidos en el docker-compose.yml	30
3.7. Correspondencia entre las etapas de desarrollo y los niveles de integración del Digital Twin.	31
4.1. Servicios desplegados en la nube (proyecto etapa_B).	34
4.2. Credenciales MQTT por rol y permisos sobre cada topic.	36
4.3. <i>Tags</i> definidos en las <i>measurements</i> de InfluxDB.	36
4.4. Resortes ensayados y etapa en que aparece cada uno.	37
4.5. Parámetros calibrados sobre el resorte 8x31.5mm	41
4.6. Correspondencia actualizada entre etapas y niveles del Digital Twin.	46
5.1. Mapeo del lazo de guarda a los componentes del sistema.	64
5.2. Constantes de sintonía del lazo de guarda. Los valores fueron elegidos por criterio de diseño y validados contra corridas reales, pero no derivan de una caracterización del ruido de la celda de carga, que no llegó a medirse; en ese sentido son puntos de partida y no una sintonía final (Sección 5.5.9).	65
5.3. Mapeo del lazo de re-zero a los componentes del sistema.	70
5.4. Los dos lazos de la Etapa D en términos de control.	73
5.5. Corridas ejecutadas en la campaña de caracterización LCF.	75

5.6. Tasa de degradación por nivel, obtenida por regresión lineal sobre la ventana post-shakedown.	75
5.7. Decisiones de diseño de la Etapa D, alternativa descartada y motivo. . . .	76
5.8. Correspondencia final entre etapas y niveles del Digital Twin.	79

ABREVIACIONES

Mayúsculas

ACL	: <i>access control list</i> , lista de control de acceso
API	: <i>application programming interface</i>
ASTM	: <i>American Society for Testing and Materials</i>
CSV	: <i>comma-separated values</i> , formato de archivo tabular
DT	: <i>Digital Twin</i> , gemelo digital
GPIO	: <i>general-purpose input/output</i>
HCF	: <i>high-cycle fatigue</i> , fatiga de alto ciclo
JSON	: <i>JavaScript Object Notation</i>
LCF	: <i>low-cycle fatigue</i> , fatiga de bajo ciclo
ML	: <i>machine learning</i> , aprendizaje automático
MQTT	: <i>Message Queuing Telemetry Transport</i>
mTLS	: <i>mutual transport layer security</i>
PHM	: <i>prognostics and health management</i>
QoS	: <i>quality of service</i> , calidad de servicio
RUL	: <i>remaining useful life</i> , vida útil remanente
SSH	: <i>secure shell</i>
TLS	: <i>transport layer security</i>
UUID	: <i>universally unique identifier</i>

Minúsculas

mm/s	: milímetros por segundo, unidad de velocidad de cicleo
N/mm	: newton por milímetro, unidad de rigidez

Componentes y servicios del sistema

<code>consumer</code>	: servicio que lee el <i>stream</i> de Redis y escribe en InfluxDB
<code>control.py</code>	: supervisor del banco, ejecutado en la Raspberry Pi
<code>fit_worker</code>	: servicio de ajuste de parámetros por ciclo
HX711	: amplificador y conversor analógico-digital de la celda de carga
<code>influxdb</code>	: base de datos de series de tiempo
<code>ingestor</code>	: servicio que recibe de MQTT y publica en el <i>stream</i> de Redis
<code>monitor_worker</code>	: servicio de vigilancia del ensayo (Etapa C)
<code>mosquitto</code>	: <i>broker</i> MQTT
<code>segmented_fit</code>	: rutina de ajuste segmentado de la curva $F-x$
<code>twin_guard_worker</code>	servicio del lazo de guarda elástica (Etapa D)

SIMBOLOGÍA

Geometría y propiedades del material

d	: diámetro del alambre del resorte [mm]
D_m	: diámetro medio de la espira [mm]
N_a	: número de espiras activas
G	: módulo de cizallamiento del material [GPa]
E	: módulo de Young [GPa]
σ_y	: límite elástico en tensión [MPa]
σ_u	: resistencia última [MPa]
ρ	: densidad [kg m^{-3}]

Modelo constitutivo del resorte

F	: fuerza axial desarrollada por el resorte [N]
x	: extensión medida desde la longitud libre [mm]
x_0	: precarga o tensión inicial de enrollado [mm]
x_e	: extensión en el límite elástico [mm]
$x_{\text{máx}}$	: extensión máxima alcanzada en un ciclo [mm]
k	: rigidez elástica [N mm^{-1}]
k_0	: rigidez inicial, sin daño acumulado [N mm^{-1}]
k_{eff}	: rigidez efectiva tras la degradación [N mm^{-1}]
F_e	: fuerza en el límite elástico [N]
H	: coeficiente de endurecimiento por deformación
w	: ancho de la transición suave entre zona elástica y plástica [mm]
η	: fracción de histéresis mecánica
β	: fracción de sobreesfuerzo que queda como deformación permanente
$\Delta\ell_{\text{perm}}$	: incremento de longitud en reposo por ciclo [mm]

Δ : sobreesfuerzo plástico por sobre x_e , $\Delta = x_{\text{máx}} - x_e$ [mm]

Fatiga y degradación

N : número de ciclos transcurridos
 N_f : vida característica del material a fatiga [ciclos]
 c : exponente de daño de la ley de Basquin
 $\mathcal{D}(N)$: daño acumulado tras N ciclos
 dk/dN : tasa de degradación de rigidez [N mm^{-1} por ciclo]

Excitación estocástica y lazos de control

D_i : extensión comandada en el ciclo i [mm]
 D_{cmd} : consigna de extensión enviada al banco [mm]
 D_{safe} : cota de extensión impuesta por el lazo de guarda [mm]
 x_{nom} : amplitud nominal de operación [mm]
 x_{nom}^* : amplitud nominal estimada por el twin desde la telemetría [mm]
 x_{trip} : extensión en que se confirma el disparo del detector [mm]
 N_i : número de empujones sorteado en el ciclo i
 λ : tasa de la distribución de Poisson
 δ : magnitud del empujón elemental [mm]
 p_{exc} : probabilidad de excursión por ciclo
 $\hat{F}(x)$: recta de Hooke de referencia, autoajustada y congelada
 $\hat{k}_0, \hat{x}_0$: parámetros de la recta de referencia
 $r(x)$: residuo entre la recta de referencia y la fuerza medida [N]
 ε : envolvente elástica del residuo [N]
 m_ε : multiplicador de la envolvente en el umbral de disparo
 τ : umbral de disparo del detector [N]
 $\hat{\Delta}$: desplazamiento del origen estimado por el lazo de re-zero [mm]

Ajuste y estadística

r^2 : coeficiente de determinación del ajuste
 σ : desviación estándar de los residuos del ajuste
 M : peso de la regularización hacia los valores de literatura
 p : valor- p de significancia estadística

Superíndices y subíndices

$\hat{}$	: valor estimado, por ejemplo $\hat{k}_0$
$*$	: valor inferido por el twin desde la telemetría
med	: magnitud medida por el instrumento
eff	: magnitud efectiva, descontando el estado acumulado
perm	: magnitud asociada a deformación permanente

INTRODUCCIÓN

1.1. Motivación y planteamiento del problema

LA ESTIMACIÓN de la vida útil remanente (*Remaining Useful Life*, RUL) de un componente mecánico es hoy uno de los objetivos centrales del mantenimiento basado en condición. La mayoría de los enfoques actuales para resolverla son de naturaleza estadística o de aprendizaje automático: en lugar de derivar la vida remanente de un modelo de daño cerrado, se entrena un modelo sobre trayectorias de degradación observadas y se le pide que extrapole [1,2]. El desempeño de esa familia de métodos depende mucho menos del algoritmo empleado que de las propiedades del conjunto de datos con que se lo entrena, y es en ese punto donde se concentra la dificultad práctica.

La literatura reciente sobre aprendizaje automático aplicado ha documentado de forma sistemática que los defectos en los datos, y no la elección del modelo, son la causa dominante de fracaso en aplicaciones reales. Sambasivan *et al.* [3] describen este fenómeno como *data cascades*: problemas de origen en la recolección que permanecen invisibles durante el desarrollo y se manifiestan de forma amplificada en operación. Northcutt *et al.* [4] muestran que incluso los conjuntos de referencia más usados de la disciplina contienen errores de etiquetado en proporción suficiente para alterar el ordenamiento relativo entre modelos. Gudivada *et al.* [5] sostienen que la calidad de los datos en aprendizaje automático es un problema estructural que no se resuelve con limpieza y transformación posteriores, porque una parte de la información que se necesita, las condiciones bajo las cuales cada dato fue producido, sencillamente no se registró. La propuesta de Gebru *et al.* [6] de acompañar cada conjunto de datos con una descripción explícita de su procedencia responde a esa misma carencia.

En el dominio específico del pronóstico de vida remanente la situación es particularmente

restrictiva, porque los datos de degradación real son costosos de obtener: exigen llevar componentes hasta la falla, lo que consume tiempo de máquina y destruye el espécimen. La consecuencia es que la disciplina trabaja sobre un número muy reducido de conjuntos de referencia. El más empleado, C-MAPSS [7], no proviene de ensayos físicos sino de un simulador de motores aeronáuticos. Los que sí son experimentales, como PRONOSTIA [8], cubren un número acotado de unidades y de condiciones de operación. A esto se suma una limitación de forma: estos conjuntos se distribuyen como tablas de señales sin el contexto que permitiría interpretarlas. Se dispone de la medición, pero no del estado físico del espécimen en cada instante, ni de la verificación de que la solicitación aplicada fue efectivamente la declarada.

De ahí el problema que motiva esta memoria. Antes de entrenar un modelo de estimación de RUL hace falta un procedimiento capaz de *producir* datos de degradación sobre los cuales se conozca, ciclo a ciclo, qué solicitación se aplicó, en qué estado se encontraba el componente y bajo qué condiciones se registró cada muestra. Esa exigencia no se satisface instrumentando un banco de ensayos y almacenando lo que sus sensores entregan. Requiere que el sistema de adquisición mantenga, en paralelo con la medición, una representación del componente que estime su estado físico y que pueda intervenir sobre el ensayo cuando las condiciones se apartan de lo previsto. Esa es, precisamente, la función de un *Digital Twin*.

El presente trabajo aborda ese problema construyendo un Digital Twin completo sobre un banco de ensayos de fatiga de resortes helicoidales de extensión. El resorte se adopta como componente de estudio por razones prácticas: es barato, su comportamiento constitutivo está descrito por leyes clásicas bien establecidas, y se degrada de forma medible en un número de ciclos alcanzable en un banco de laboratorio. Estas propiedades lo hacen adecuado como caso de estudio para un procedimiento cuyo interés es metodológico y no específico del componente.

1.2. Estado del arte

1.2.1. Calidad de los datos en aprendizaje automático

El desplazamiento de atención desde el modelo hacia los datos es una de las tendencias más marcadas de la literatura reciente en aprendizaje automático aplicado. El argumento central es que, alcanzado cierto nivel de madurez en las arquitecturas disponibles, el rendimiento marginal de mejorar el modelo es pequeño comparado con el de mejorar el conjunto de entrenamiento. Las dimensiones de calidad identificadas con mayor frecuencia son la corrección del etiquetado [4], la cobertura de las condiciones de operación relevantes, y la trazabilidad de la procedencia [6].

Esta última dimensión es la que resulta más directamente pertinente aquí. Un conjunto de datos generado en laboratorio tiene, sobre uno recopilado en terreno, la ventaja potencial

de que el experimentador controla la excitación y por lo tanto conoce la condición aplicada. Esa ventaja es sólo potencial, sin embargo, porque un banco de ensayos en lazo abierto no garantiza que la condición comandada sea la condición efectivamente aplicada. En el caso de un resorte cargado por sobre su límite elástico esta diferencia es sustantiva: la deformación permanente que el propio ensayo induce desplaza el punto de operación ciclo a ciclo, de modo que una consigna constante produce una sollicitación decreciente (Sección 5.2.2). Un conjunto de datos etiquetado con la consigna nominal estaría, en ese escenario, mal etiquetado por construcción.

1.2.2. Estimación de vida remanente y conjuntos de datos de degradación

La revisión de Jardine *et al.* [9] establece la estructura hoy convencional del mantenimiento basado en condición: adquisición de datos, procesamiento de la señal, y decisión de mantenimiento. Sobre ese esquema, Si *et al.* [1] y Lei *et al.* [2] sistematizan los métodos de estimación de RUL y coinciden en señalar la disponibilidad de datos de degradación como la restricción principal del campo. Los métodos guiados por datos requieren trayectorias completas desde el estado sano hasta la falla, en cantidad suficiente para caracterizar la dispersión entre unidades; los métodos basados en modelos físicos requieren, a su vez, datos experimentales para identificar sus parámetros.

Los conjuntos disponibles públicamente reflejan ese costo. C-MAPSS [7] resuelve el problema mediante simulación, lo que entrega gran cantidad de trayectorias a cambio de que la degradación sea la de un modelo y no la de un material. PRONOSTIA [8] sí es experimental, y por ello ilustra el otro extremo del compromiso: un número reducido de rodamientos ensayados hasta la falla bajo unas pocas condiciones de carga. Ninguno de los dos entrega, junto a la señal, una estimación independiente del estado físico del espécimen en cada ciclo.

1.2.3. Digital Twin: definición y taxonomía

El concepto de Digital Twin fue introducido por Grieves [10] en el contexto de gestión del ciclo de vida de productos, como el emparejamiento entre un objeto físico, su contraparte virtual, y el flujo de información que las vincula. Su adopción posterior en manufactura [11, 12] amplió el alcance del término hasta un punto en que resultó necesario precisarlo.

La clasificación de Kritzinger *et al.* [13], que esta memoria adopta como criterio, resuelve esa imprecisión distinguiendo tres niveles según el grado de automatización del flujo de datos entre objeto y representación:

- **Modelo Digital** (*Digital Model*): no existe intercambio automático de datos en ninguna dirección. Un cambio en el objeto físico no se refleja automáticamente en

el modelo.

- **Sombra Digital** (*Digital Shadow*): existe un flujo automatizado desde el objeto físico hacia la representación virtual. La representación refleja el estado del objeto, pero no lo modifica.
- **Digital Twin**: el flujo automatizado es *bidireccional*. La representación virtual no sólo refleja el estado del objeto físico, sino que además actúa sobre él sin intervención humana.

El valor de esta taxonomía para el presente trabajo es que convierte la clasificación en una propiedad verificable del sistema construido, en lugar de una declaración de intenciones. Revisiones posteriores [14, 15] coinciden en señalar que la gran mayoría de los trabajos publicados bajo el rótulo de Digital Twin corresponden, según este criterio, al nivel de Sombra Digital, y que la implementación efectiva del flujo de retorno es la brecha más frecuente entre la propuesta conceptual y el sistema realizado.

1.2.4. Modelado de resortes helicoidales y fatiga

El comportamiento del resorte helicoidal bajo carga axial está descrito por un conjunto de leyes constitutivas clásicas. La región elástica responde a la ley de Hooke con la rigidez derivada de la teoría de elasticidad para resortes helicoidales [16, 17]. Más allá del límite elástico, el endurecimiento por deformación se describe mediante la relación de Hollomon [18] o, con transición suave entre ambas regiones, mediante la formulación de Ramberg-Osgood [19]. La disipación de energía por fricción interna entre espiras introduce histéresis en el ciclo de carga y descarga [20].

La acumulación de daño bajo carga cíclica se aborda tradicionalmente mediante la ley de Basquin [21] en el régimen de fatiga de alto ciclo y mediante la relación de Coffin-Manson [22] en el régimen de bajo ciclo, donde la deformación plástica por ciclo es apreciable. Suresh [23] y Dieter [24] entregan el tratamiento de referencia de ambos regímenes. Los ensayos de fatiga controlados por deformación se rigen por la norma ASTM E606 [25].

Estas leyes constituyen el modelo físico sobre el cual se construye la representación virtual de esta memoria. Su papel no es predecir la falla con exactitud, sino proporcionar, en cada ciclo, una estimación independiente del estado del componente que pueda contrastarse contra la medición.

1.3. Contribuciones

Las contribuciones de este trabajo son las siguientes.

1. **Un Digital Twin completo y verificable sobre hardware real.** Se construye e integra la cadena completa, desde el modelo constitutivo del componente hasta el lazo

de retorno que actúa sobre el banco, y se verifica el nivel alcanzado contra el criterio explícito de Kritzinger *et al.* [13]. El sistema opera sobre infraestructura de bajo costo y con el instrumento y la representación virtual separados por una red pública.

2. Un procedimiento de generación de datos con etiquetado físico por ciclo.

Además de la señal cruda del instrumento, el sistema persiste en cada ciclo los parámetros del modelo ajustados sobre esa misma medición, con lo que cada tramo del conjunto de datos queda acompañado de una estimación del estado del componente en ese instante. Esto convierte una serie temporal de fuerza y extensión en un conjunto de datos con etiquetas de estado, que es la forma que los métodos supervisados de estimación de RUL requieren.

3. Un mecanismo de control que preserva la validez del etiquetado durante el ensayo.

Se implementa un lazo cerrado que corrige la consigna de extensión para mantener constante el sobreesfuerzo plástico por ciclo, compensando la deriva que la propia deformación permanente introduce. Sin esta corrección la condición experimental declarada deja de corresponder a la aplicada, y el conjunto de datos resultante queda mal etiquetado por construcción.

4. Un procedimiento de generación de eventos anómalos con etiqueta conocida.

Se introduce una excitación estocástica de Poisson sobre la consigna, que provoca sobresolicitaciones ocasionales en instantes que el sistema de detección no puede anticipar. Como la secuencia de perturbaciones es sorteada por el propio banco, cada ciclo anómalo queda identificado por construcción, condición que los datos recogidos en operación real no ofrecen.

5. Un resultado experimental de degradación.

Se recupera de forma empírica la relación entre el sobreesfuerzo plástico por ciclo y la tasa de degradación de rigidez, sobre tres niveles de sollicitación, con significancia estadística en cada nivel. El resultado reemplaza, para el rango ensayado, la dependencia heredada de literatura que el modelo físico incorporaba inicialmente.

1.4. Objetivos, alcances y limitaciones

1.4.1. Objetivo general

Diseñar, implementar y validar experimentalmente un Digital Twin de un banco de ensayos de fatiga de resortes helicoidales, que alcance el nivel de flujo bidireccional automatizado según la taxonomía de Kritzinger *et al.*, y que habilite la generación de conjuntos de datos de degradación trazables y físicamente etiquetados, orientados al entrenamiento de modelos de estimación de vida útil remanente.

1.4.2. Objetivos específicos

1. Implementar un modelo físico del resorte helicoidal a partir de leyes constitutivas documentadas, distinguiendo de forma explícita los parámetros medidos experimentalmente de los adoptados desde literatura.
2. Construir una arquitectura de adquisición y persistencia en *streaming* capaz de sostener el flujo continuo del instrumento sin pérdida de datos, y de conservar la trazabilidad de cada muestra hasta el experimento que la originó.
3. Integrar el banco de ensayos físico con esa arquitectura sobre un enlace de red autenticado, y calibrar el modelo contra el comportamiento medido de los resortes reales.
4. Implementar el flujo de retorno desde la representación virtual hacia el instrumento, e instanciarlo en dos lazos de control con propósitos distintos: la preservación de la condición experimental y la protección del componente ante sobresolicitud.
5. Ejecutar una campaña experimental de fatiga que sólo el lazo cerrado hace posible, y recuperar de sus datos la relación entre amplitud de sollicitación y tasa de degradación.

1.4.3. Alcances

El trabajo comprende el desarrollo completo del sistema, desde el modelo físico hasta la operación del lazo cerrado sobre hardware real, y la ejecución de las campañas experimentales que lo validan. Comprende también la caracterización experimental de los resortes empleados y la obtención de la relación empírica de degradación.

1.4.4. Limitaciones

Las siguientes limitaciones delimitan el alcance de lo demostrado y se reiteran, en su contexto específico, a lo largo del documento.

- **Régimen de fatiga.** Las campañas se ejecutan en régimen de fatiga de bajo ciclo, con penetración plástica deliberada en cada ciclo. El régimen de alto ciclo, que corresponde a la operación real de un resorte industrial, exige del orden de 10^6 ciclos; a la velocidad del banco disponible, aproximadamente un ciclo cada 51 segundos, eso equivale a unos 20 meses de operación continua y queda fuera del horizonte temporal de esta memoria.
- **Tamaño muestral.** La campaña de caracterización emplea un resorte por nivel de sollicitación, lo que impide separar la respuesta del material de la dispersión entre ejemplares.

- **Entrenamiento de modelos.** Esta memoria construye el procedimiento de generación de datos y produce el conjunto correspondiente a las campañas ejecutadas. El entrenamiento y la evaluación de modelos de aprendizaje automático sobre ese conjunto no forman parte del trabajo.
- **Alcance del lazo de protección.** El lazo de guarda elástica es un demostrador del flujo virtual hacia físico sobre resorte sacrificial. La detección ocurre en la nube, con la latencia que ello implica, y por lo tanto el sistema no constituye un mecanismo de protección apto para operación.
- **Instrumentación no caracterizada.** El piso de ruido de la celda de carga empleada no fue medido empíricamente, de modo que las constantes de sintonía del detector son valores validados contra ensayos reales pero no una sintonía final.

1.4.5. Trabajo diseñado y no ejecutado

Durante el desarrollo se diseñó una segunda generación del banco, accionada por biela y disco excéntrico en lugar de husillo, cuyo propósito es elevar la frecuencia de cicleo lo suficiente para alcanzar el régimen de alto ciclo. El diseño incluye el dimensionamiento del mecanismo, la verificación de torque contra las especificaciones del motor disponible, y la incorporación de un canal de emisión acústica con dos sensores en coincidencia para la detección de eventos de fractura. Este desarrollo quedó completamente especificado pero no construido, y se presenta como trabajo futuro en el Capítulo 6.

1.5. Organización del documento

El desarrollo se organiza en cuatro etapas sucesivas, cada una de las cuales ocupa un capítulo y agrega una capa sobre la anterior. La progresión sigue la recomendación de la literatura sobre Digital Twins [12] de que la representación virtual debe poder construirse y validarse de forma independiente del sistema físico antes de acoplarse a él.

Capítulo 2, Etapa A. Establece el modelo físico del resorte a partir de las leyes constitutivas, deliberadamente aislado del hardware. Corresponde al nivel previo a la Sombra Digital en la taxonomía adoptada.

Capítulo 3, Etapa B. Construye la arquitectura de datos en *streaming* sobre la que circula la información, y la valida empleando el modelo físico como fuente sintética.

Capítulo 4, Etapa C. Reemplaza esa fuente sintética por el banco de ensayos físico, calibra el modelo contra los resortes reales y establece el nivel de Sombra Digital.

Capítulo 5, Etapa D. Implementa el flujo de retorno desde la representación virtual hacia el instrumento, en los dos lazos descritos, y con ello alcanza el nivel de Digital Twin. Presenta además el resultado experimental de la campaña de degradación.

Capítulo 6. Reúne las conclusiones del trabajo y las líneas de continuación.

ETAPA A: MODELADO FÍSICO DEL RESORTE

2.1. Introducción y objetivos de la etapa

El primer paso hacia la construcción del Digital Twin del banco de ensayos de fatiga consiste en establecer una representación matemática fiel del componente bajo estudio: el resorte. Antes de incorporar la complejidad del sistema de actuación, los sensores o los protocolos de comunicación con la Raspberry Pi, se requiere un modelo virtual que reproduzca la relación constitutiva *fuerza-extensión* del resorte real, basado en leyes físicas documentadas en la literatura.

Esta etapa, denominada **Etapa A — Modelo físico puro**, se desarrolla de forma deliberadamente aislada del hardware. Sus objetivos son:

1. Implementar las leyes constitutivas fundamentales que describen el comportamiento de un resorte helicoidal de acero sometido a carga axial cíclica.
2. Calibrar el modelo con los datos experimentales disponibles (CSV registrado por el banco), separando claramente los parámetros medidos de los parámetros estimados desde literatura.
3. Generar un catálogo extensible de resortes del set comercial empleado (Yato YT-06875), de modo que el modelo no quede limitado a un único espécimen.
4. Validar visualmente, mediante un *dashboard* de gráficas, que el comportamiento simulado es consistente con la fenomenología esperada del material.

El enfoque por capas (física $\rightarrow$ banco $\rightarrow$ conectividad) responde a la recomendación de la literatura sobre Digital Twins [12]: la representación virtual debe poder evolucionar de forma independiente del sistema físico, y por lo tanto el modelo del componente debe ser, en sí mismo, una entidad bien definida y validable.

2.2. Fundamentos físicos implementados

El modelo incorpora cinco leyes constitutivas clásicas de la mecánica de materiales, todas ellas con respaldo bibliográfico robusto. A continuación se detalla cada una, junto con la forma matemática empleada y su justificación.

2.2.1. Ley de Hooke con precarga (zona elástica)

La región de comportamiento lineal del resorte se describe mediante la Ley de Hooke generalizada para incluir una precarga inicial x_0 :

$$F(x) = \begin{cases} k(x - x_0), & x_0 \leq x \leq x_e \\ 0, & x < x_0 \end{cases} \quad (2.1)$$

donde F es la fuerza aplicada [N], x es la extensión medida desde la longitud libre [mm], k es la constante de rigidez [N mm⁻¹], x_0 es la precarga del resorte [mm] y x_e es la extensión en el límite elástico [mm].

La precarga x_0 es característica de los resortes de extensión con espiras juntas: existe una tensión inicial de enrollado que impide que las espiras se separen hasta que se aplica una fuerza mínima [17].

2.2.2. Rigidez teórica del resorte helicoidal

Para resortes helicoidales sometidos a carga axial, la rigidez k se deriva de la teoría de la elasticidad como [16, 17]:

$$k = \frac{G \cdot d^4}{8 \cdot D_m^3 \cdot N_a} \quad (2.2)$$

donde G es el módulo de cizallamiento del material [GPa], d es el diámetro del alambre [mm], D_m es el diámetro medio de la espira [mm], y N_a es el número de espiras activas. Se emplea aquí la notación D_m y N_a , y no D y N como es habitual en la literatura de resortes, para no confundirlas con la extensión comandada y el número de ciclos, que se denotan D y N en los capítulos siguientes. Esta formulación permite que la rigidez del modelo no sea un parámetro arbitrario, sino una consecuencia verificable de la geometría del resorte.

2.2.3. Endurecimiento por deformación (zona plástica)

Más allá del límite elástico, el material experimenta deformación plástica y la respuesta deja de ser lineal. Se adopta un modelo simplificado de endurecimiento lineal, inspirado en la ecuación de Hollomon [18]:

$$F(x) = F_e + k \cdot H \cdot (x - x_e), \quad x > x_e \quad (2.3)$$

donde $F_e = k(x_e - x_0)$ es la fuerza en el límite elástico y $H \in [0, 1]$ es el coeficiente de endurecimiento (fracción de la pendiente elástica conservada en zona plástica).

La ecuación de Hollomon original es $\sigma = K\varepsilon^n$, con n típicamente entre 0.1 y 0.3 para aceros dúctiles. La linealización (2.3) es válida como aproximación de primer orden en deformaciones plásticas moderadas [24], condición que se cumple en el rango operativo del banco. Existen formulaciones más sofisticadas, como Ramberg-Osgood [19], que proporcionan una transición suave entre las zonas elástica y plástica; su incorporación se propone como mejora futura una vez se disponga de datos experimentales en régimen plástico.

2.2.4. Deformación permanente (plastic set)

Tras un ciclo que alcance la zona plástica, una fracción del exceso plástico no se recupera y queda como deformación permanente:

$$\Delta x_{\text{perm}} = \beta \cdot (x_{\text{máx}} - x_e), \quad x_{\text{máx}} > x_e \quad (2.4)$$

donde $\beta \in [0, 1]$ es la fracción acumulada por ciclo (típicamente 0.01–0.05 para aceros), y $x_{\text{máx}}$ es la extensión máxima alcanzada en el ciclo. El origen efectivo del resorte se desplaza, desplazando la curva F - x de los ciclos posteriores hacia la derecha.

2.2.5. Fatiga: ley de Basquin

La degradación acumulada por carga cíclica se modela mediante la ley de Basquin [21], que relaciona el daño con el número de ciclos en régimen plástico:

$$\mathcal{D}(N) = \left(\frac{N}{N_f} \right)^c \quad (2.5)$$

donde N es el número de ciclos en zona plástica, N_f es la vida característica del material (en ciclos), y c es el exponente de daño, típicamente cercano a 0.5 para aceros. La rigidez efectiva del resorte degrada según:

$$k_{\text{eff}}(N) = k_0 \cdot (1 - \mathcal{D}(N)) \quad (2.6)$$

saturando en un valor mínimo para evitar pérdida total de rigidez.

2.2.6. Histéresis mecánica

La curva de retorno difiere de la curva de subida debido a la energía disipada por fricción interna entre espiras y por microdeformaciones reversibles [20]. Se incorpora como una reducción constante de fuerza en la fase de retorno:

$$F_{\text{ret}}(x) = F(x) - \eta \cdot F_{\text{máx}} \quad (2.7)$$

donde η es la fracción de histéresis (típicamente 2–5 % para acero al cromo-vanadio) y $F_{\text{máx}}$ es la fuerza máxima en el límite elástico del ciclo.

2.2.7. Carácter determinista del modelo

Las seis leyes anteriores comparten una propiedad que conviene dejar establecida, porque condiciona el diseño de todo lo que sigue: el modelo es **estrictamente determinista**. Dadas la extensión comandada y el estado acumulado del resorte (deformación permanente y ciclos en zona plástica), la fuerza queda determinada de forma unívoca, sin ningún término estocástico.

Esta elección corresponde al fenómeno que se representa. El comportamiento mecánico de un metal bajo carga es determinista, y la dispersión que se observa al medirlo proviene de la cadena de medición (ruido electrónico de la celda de carga HX711, vibraciones del banco) y de las diferencias entre ejemplares, no de un carácter aleatorio del material. Separar ambas cosas tiene una consecuencia práctica que se aprovecha más adelante: si el modelo no genera dispersión por sí mismo, toda desviación entre lo predicho y lo medido es atribuible al instrumento o al espécimen, que es exactamente lo que necesita un detector construido sobre la diferencia entre ambos (Sección 5.5.2).

Cuando en la Etapa D aparezca una componente aleatoria, será por lo tanto externa al modelo: una perturbación introducida deliberadamente en la consigna del banco, no una propiedad del resorte (Sección 5.2.5).

2.3. Material de los resortes ensayados

Los resortes utilizados pertenecen al set comercial Yato YT-06875, fabricado en **acero al cromo-vanadio** con recubrimiento galvanizado para resistencia a la corrosión. Este material se caracteriza por las siguientes propiedades nominales:

El resorte específico empleado en el banco experimental tiene designación 7×38 (diámetro exterior nominal 7 mm, longitud libre 38 mm). A partir de la regresión lineal sobre el CSV experimental (`datos_7x38_D70mm_V500us.csv`) se identificaron sus dos parámetros principales:

$$k_{\text{medido}} = 0.109 \text{ N mm}^{-1} \quad (2.8)$$

$$x_{0,\text{medido}} \approx 4.5 \text{ mm} \quad (2.9)$$

Cuadro 2.1: Propiedades del acero al cromo-vanadio (literatura estándar).

Propiedad	Símbolo	Valor
Módulo de Young	E	200–210 GPa
Módulo de cizallamiento	G	79.3 GPa
Límite elástico (tensión)	σ_y	600–900 MPa
Resistencia última	σ_u	800–1200 MPa
Densidad	ρ	7850 kg m ⁻³

La geometría del modelo se calibró ($d = 0.55$ mm, $D_m = 6.45$ mm, $N_a = 31$) para que la ecuación (2.2) reprodujera la rigidez medida.

2.4. Arquitectura del software de simulación

La implementación se organiza en cuatro módulos Python con responsabilidades separadas, siguiendo principios de modularidad y reutilización:

physics.py Define las leyes físicas. Contiene la clase **Spring** y las funciones **stretch_curve()** y **multi_cycle_curve()** para generar curvas de ensayo.

springs_library.py Catálogo de resortes con sus parámetros geométricos. Calcula k desde la geometría mediante (2.2).

plotter.py Genera el *dashboard* de cuatro paneles con las gráficas características del modelo.

main.py Punto de entrada con interfaz de línea de comandos para explorar distintos resortes y parámetros de ensayo.

2.4.1. Parámetros relevantes y su justificación

La Tabla 2.2 resume los parámetros configurables del modelo, indicando su origen y nivel de confianza.

La transparencia en este desglose es esencial para la honestidad metodológica del Digital Twin: los parámetros con confianza “Baja” o “Media” son hipótesis estructurales que deberán calibrarse experimentalmente en etapas posteriores, cuando el banco genere datos en regímenes que aún no han sido explorados (zona plástica, fatiga prolongada).

2.5. Implementación: extractos de código relevantes

Se presentan los fragmentos esenciales del código, comentados.

Cuadro 2.2: Parámetros del modelo, su origen y nivel de confianza.

Parámetro	Valor (7x38)	Unidad	Origen	Confianza
k (rigidez)	0.109	N mm^{-1}	Regresión sobre CSV experimental	Alta (medido)
x_0 (precarga)	4.5	mm	Intercepto regresión CSV	Alta (medido)
x_e (límite elástico)	65.0 [†]	mm	Hipótesis basada en literatura (resortes de acero)	Baja (pendiente experimental)
H (endurecimiento)	0.18	—	Literatura cromo-vanadio [18, 24]	Media
β (plastic set)	0.02	—	Literatura aceros	Media
N_f (vida fatiga)	50 000	ciclos	Literatura resortes pequeños	Media
c (exp. Basquin)	0.5	—	Literatura aceros [21]	Media
η (histéresis)	0.05	—	Estimación visual del CSV	Media-alta

[†]Valor por defecto de la clase `Spring`. La entrada de catálogo `real.7x38` deriva su propio límite elástico de la geometría, obteniendo $\approx 66,6$ mm; ése es el valor que emplea la simulación de la Etapa B (Sección 3.9). Ambos son estimaciones sin respaldo experimental, y la diferencia entre ellos carece de consecuencias hasta que la Etapa C mide x_e sobre resortes reales.

2.5.1. Clase Spring (extracto de physics.py)

```

1 @dataclass
2 class Spring:
3     """Resorte f'ísico basado en leyes documentadas."""
4
5     name: str
6     k: float = 0.109
7     preload_mm: float = 4.5
8     elastic_limit_mm: float = 65.0
9     hardening_H: float = 0.18
10    plastic_set_beta: float = 0.02
11    fatigue_life_cycles: float = 50_000.0
12    fatigue_exponent_c: float = 0.5
13    hysteresis_fraction: float = 0.03
14
15    # Estado interno
16    _permanent_set_mm: float = field(default=0.0, repr=False)
17    _cycles_in_plastic: int = field(default=0, repr=False)
18
19    def force(self, x_mm: float, direction: int = +1) -> float:
20        # Origen desplazado por deformacion permanente
21        x_eff = x_mm - self._permanent_set_mm
22        if x_eff <= self.preload_mm:
23            return 0.0
24

```

```

25     k_now = self.effective_k()
26     x_e   = self.elastic_limit_mm
27
28     if x_eff <= x_e:
29         # Zona elastica: Ley de Hooke
30         F = k_now * (x_eff - self.preload_mm)
31     else:
32         # Zona plastica: endurecimiento de Hollomon
33         F_e = k_now * (x_e - self.preload_mm)
34         F   = F_e + k_now * self.hardening_H * (x_eff - x_e)
35
36     # Histeresis (solo en retorno)
37     if direction < 0:
38         F_max = k_now * (x_e - self.preload_mm)
39         F -= self.hysteresis_fraction * F_max
40
41     return max(F, 0.0)

```

Listing 2.1: Núcleo del modelo físico.

El método `force()` integra las ecuaciones (2.1), (2.3) y (2.7) en una única función condicional. La separación entre x_{mm} (extensión absoluta) y x_{eff} (extensión efectiva descontando la deformación permanente acumulada) permite que el origen de la curva se desplace de ciclo a ciclo cuando el resorte trabaja en zona plástica.

2.5.2. Cálculo de rigidez teórica

```

1 def theoretical_k(d_mm, D_mm, N_active, G_GPa=79.3):
2     """k = G*d^4 / (8*D^3*N) para resortes helicoidales."""
3     G_Nmm2 = G_GPa * 1000.0 # GPa -> N/mm^2
4     return (G_Nmm2 * d_mm**4) / (8.0 * D_mm**3 * N_active)

```

Listing 2.2: Función para calcular k desde la geometría (ec. 2.2).

Esta función permite trazar la rigidez de cualquier resorte del catálogo a partir de sus dimensiones físicas, en lugar de tratar k como un parámetro arbitrario. El módulo de cizallamiento G se fija en 79.3 GPa (acero al cromo-vanadio).

2.6. Resultados: dashboard de simulación

Para validar el comportamiento del modelo, se ejecutó la simulación sobre el resorte calibrado `real_7x38` con una extensión máxima de 90 mm (que excede el límite elástico hipotético de 65 mm) durante 24 ciclos. El resultado se muestra en la Figura 2.1.

A continuación se interpreta cada panel.

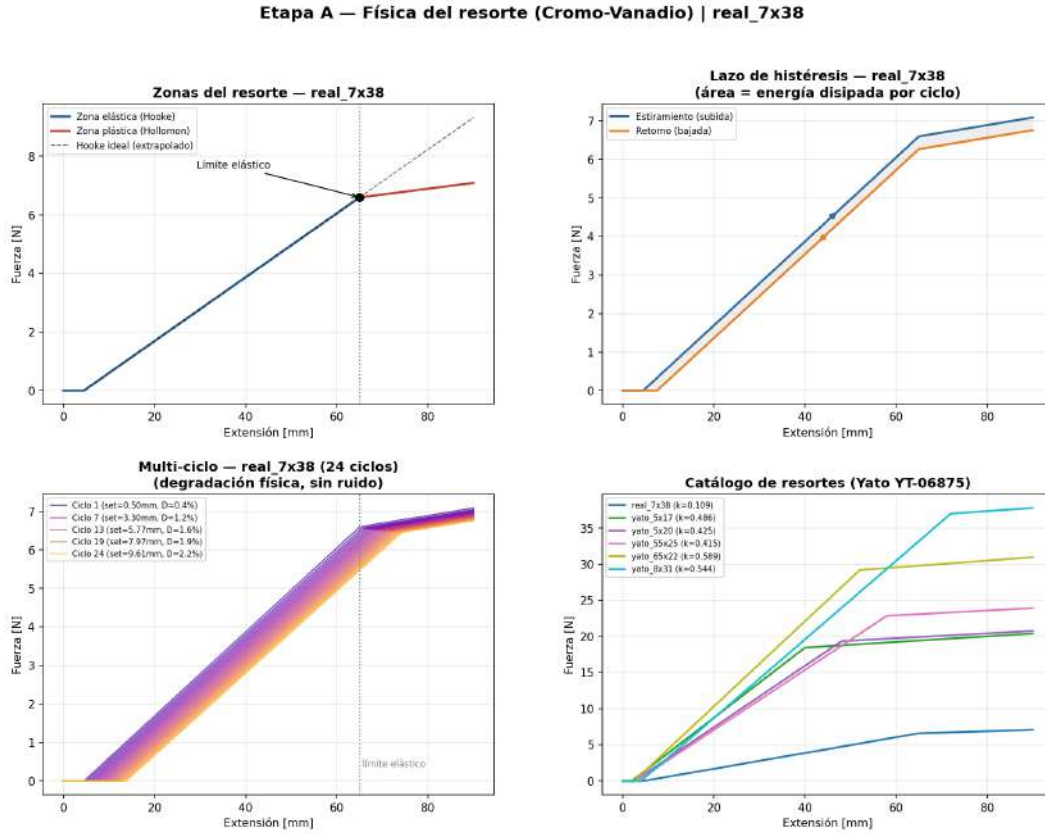


Figura 2.1: Dashboard de la simulación física (Etapa A) para el resorte calibrado `real_7x38`.

2.6.1. Panel 1: Zonas del resorte (superior izquierdo)

Muestra la curva $F-x$ de un único barrido en subida, con las zonas elástica y plástica diferenciadas por color. La línea punteada negra representa la extrapolación de la Ley de Hooke ideal (lo que el resorte “predeciría” si no entrara en plasticidad), permitiendo visualizar la desviación.

La forma de la curva se explica directamente por el modelo. La zona elástica (azul) es una recta perfecta entre $x = x_0 = 4.5 \text{ mm}$ y $x = x_e = 65 \text{ mm}$, con pendiente $k = 0.109 \text{ N mm}^{-1}$. A partir del límite elástico, la curva continúa subiendo pero con pendiente $k \cdot H = 0.109 \cdot 0.18 = 0.020 \text{ N mm}^{-1}$, es decir aproximadamente un 18 % de la pendiente elástica, lo que corresponde al comportamiento típico de aceros dúctiles en su régimen de endurecimiento por deformación.

2.6.2. Panel 2: Lazo de histéresis (superior derecho)

Muestra ambas fases del ciclo (subida y bajada) en la misma gráfica, con flechas indicando el sentido del recorrido y área sombreada representando la energía disipada por ciclo.

Conviene notar que los ejes son extensión–fuerza (*no* tiempo). El hecho de que las curvas de subida y bajada no coincidan refleja la naturaleza disipativa del proceso: para una misma extensión, la fuerza durante el retorno es menor que durante la subida. El área encerrada es proporcional a la energía mecánica convertida en calor por fricción interna del resorte, magnitud relevante para análisis térmico y vida a fatiga [20].

2.6.3. Panel 3: Multi-ciclo con degradación (inferior izquierdo)

Muestra la curva de subida para 24 ciclos sucesivos, con indicadores de deformación permanente acumulada (**set**) y porcentaje de daño (D) para ciclos representativos.

El desplazamiento progresivo hacia la derecha tiene una causa única: cada ciclo que alcanza la zona plástica desplaza el origen efectivo del resorte, por lo que las curvas posteriores arrancan progresivamente más a la derecha. Tras 24 ciclos, la deformación permanente acumulada es de aproximadamente 9.6 mm y el daño por fatiga llega a 2.2 %. La degradación es *suave y determinista*: no hay componentes aleatorias, solo la combinación progresiva del *plastic set* (ec. 2.4) y la fatiga de Basquin (ec. 2.5).

2.6.4. Panel 4: Catálogo de resortes (inferior derecho)

Comparación de los seis resortes del catálogo, todos modelados con las mismas leyes pero con geometrías y parámetros distintos. Permite verificar que el modelo escala correctamente a otros tamaños y rigideces, lo cual es esencial para el objetivo final de generar un dataset rico para aprendizaje automático.

Los valores obtenidos son coherentes con la geometría de cada resorte. Los resortes del set Yato YT-06875 exhiben rigideces entre 0.109 y 0.589 N mm⁻¹. Los resortes con mayor diámetro de alambre (*yato_65x22*, *yato_8x31*) presentan rigideces más altas, consistente con la dependencia $k \propto d^4$ de la ecuación (2.2).

2.7. Discusión sobre el modelo de zona plástica

La elección del modelo lineal (2.3) para la zona plástica es una decisión de diseño que merece ser justificada. La Figura 2.2 muestra cuatro modelos constitutivos alternativos aplicados al mismo resorte.

Justificación de la elección:

- El modelo lineal es el más simple y requiere un único parámetro adicional (H), lo que facilita su calibración inicial con pocos puntos experimentales.

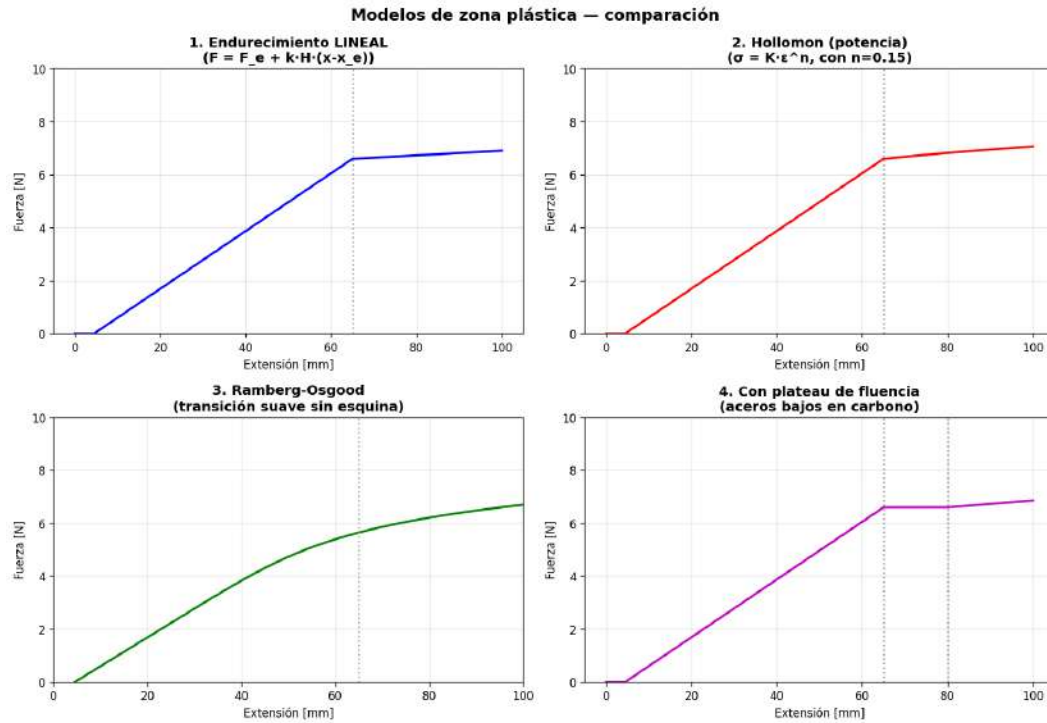


Figura 2.2: Comparación de cuatro modelos para la zona plástica: (1) lineal, el empleado en este trabajo; (2) Hollomon con $n = 0.15$; (3) Ramberg-Osgood, con transición suave; (4) modelo con *plateau* de fluencia, característico de aceros bajo en carbono.

- Es apropiado para deformaciones plásticas moderadas, que corresponden al régimen operativo previsto del banco (no se pretende llevar los resortes a fractura sistemáticamente).
- Cuando se disponga de datos experimentales en zona plástica, se evaluará la conveniencia de migrar a modelos más sofisticados (Hollomon con potencia, Ramberg-Osgood). La arquitectura modular del código permite este cambio encapsulado en una única función.

ETAPA B: ARQUITECTURA DE DATOS EN *STREAMING* PARA EL DIGITAL SHADOW

3.1. Introducción y objetivos de la etapa

La Etapa B constituye el segundo escalón en la construcción del Digital Twin del banco de ensayos de fatiga de resortes. Habiendo establecido en la Etapa A un modelo físico determinista del resorte basado en leyes constitutivas documentadas, el siguiente desafío consiste en construir la infraestructura de datos que permita transmitir, almacenar y visualizar de forma automatizada las señales del experimento, materializando el flujo unidireccional que define a un *Digital Shadow* según la taxonomía de Kritzinger et al. [13].

Esta etapa se desarrolla de forma deliberadamente independiente del hardware del banco: la fuente de datos es el modelo físico de la Etapa A, lo que permite validar la arquitectura de extremo a extremo antes de conectar los sensores reales (Etapa C). Sus objetivos son:

1. Comprender y verificar el funcionamiento del protocolo MQTT mediante pruebas elementales de publicación y suscripción.
2. Instalar y configurar el broker Eclipse Mosquitto en un entorno Linux virtualizado.
3. Definir un formato de mensaje estructurado (JSON) autodescriptivo y versionado para las señales del experimento de fatiga.
4. Incorporar Redis como buffer intermedio mediante *Streams* y *consumer groups*, garantizando tolerancia a fallos y desacople entre la ingesta y la persistencia.

5. Persistir las señales en una base de datos de series temporales (InfluxDB), preservando un conjunto de datos crudo e inmutable.
6. Visualizar las señales en un panel interactivo (Grafana).
7. Integrar el modelo físico de la Etapa A como fuente de datos del *publisher*.
8. Contenerizar la totalidad del sistema con Docker, garantizando reproducibilidad.

La elección de MQTT como protocolo de comunicación responde a sus características inherentes para sistemas de monitoreo industrial: es liviano (overhead mínimo), opera sobre TCP/IP, soporta múltiples suscriptores simultáneos en un mismo *topic*, y es el protocolo estándar en aplicaciones IoT y Digital Twins de nivel *Digital Shadow* según la taxonomía de Kritzinger et al. [13].

3.2. Fundamentos del protocolo MQTT

MQTT (*Message Queuing Telemetry Transport*) es un protocolo de mensajería publicación-suscripción (*pub/sub*) diseñado para comunicaciones con recursos limitados y redes de baja fiabilidad [26]. Su arquitectura se basa en tres componentes:

- **Broker:** servidor central que recibe todos los mensajes y los redistribuye a los clientes suscritos. No procesa ni interpreta el contenido del mensaje.
- **Publisher:** cliente que publica mensajes en un *topic* determinado. No conoce ni le importa quién está suscrito.
- **Subscriber:** cliente que se suscribe a uno o más *topics* y recibe los mensajes en tiempo real.

3.2.1. Concepto de topic

Un *topic* es una cadena de texto jerárquica que funciona como canal de comunicación. La jerarquía se expresa con barras diagonales (/), sin significado especial para el broker más allá de la organización. Por ejemplo: `springlab/raw`. Los suscriptores pueden usar comodines para recibir múltiples topics simultáneamente: `#` (comodín multinivel) y `+` (comodín de un nivel).

3.2.2. Calidad de servicio (QoS)

MQTT define tres niveles de calidad de servicio:

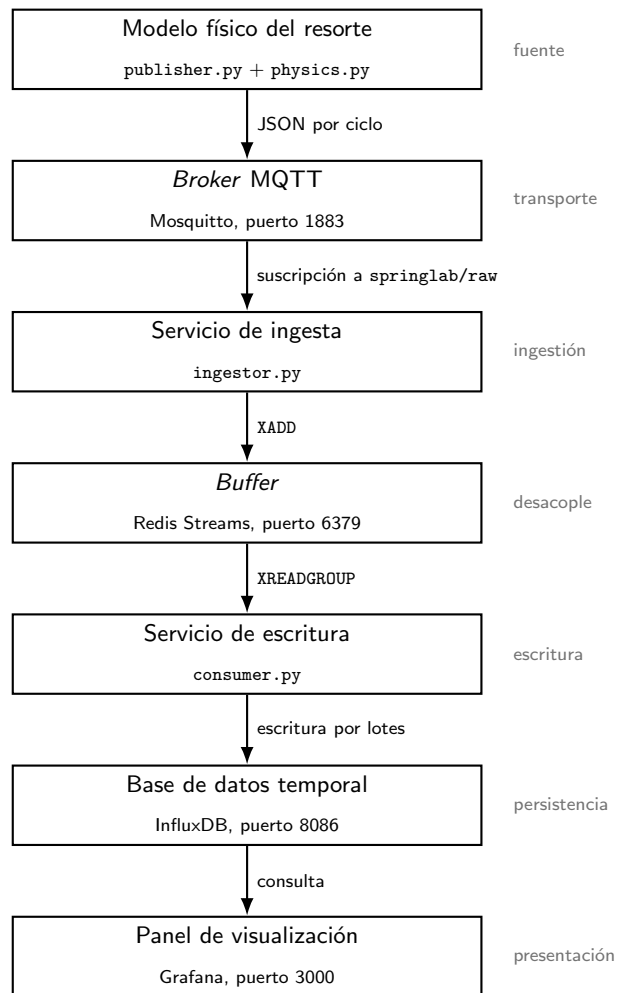


Figura 3.1: Arquitectura por capas de la cadena de datos de la Etapa B. El flujo es lineal y unidireccional desde la fuente hasta la visualización, sin ningún camino de retorno hacia el objeto representado, que es lo característico de un *Digital Shadow*. En esta etapa la fuente es todavía el modelo físico de la Etapa A; la Etapa C la reemplaza por el instrumento real sin alterar el resto de la cadena.

Cuadro 3.1: Niveles de QoS en MQTT.

QoS	Garantía	Uso en este proyecto
0	A lo más una vez (sin confirmación)	—
1	Al menos una vez (con confirmación)	Datos experimentales
2	Exactamente una vez (handshake doble)	—

Para las variables del experimento se emplea QoS 1, que garantiza la entrega sin el overhead de QoS 2.

3.3. Entorno de desarrollo: máquina virtual Linux

La Etapa B se implementa sobre una máquina virtual Ubuntu 26.04 LTS ejecutada en Oracle VirtualBox. Esta decisión responde a la necesidad de disponer de un entorno Linux controlado y reproducible, independiente del sistema operativo del computador de desarrollo.

3.3.1. Configuración de la máquina virtual

Los parámetros de configuración de la VM empleada se resumen en la Tabla 3.2.

Cuadro 3.2: Parámetros de configuración de la máquina virtual.

Parámetro	Valor
Hipervisor	Oracle VirtualBox
Sistema operativo	Ubuntu 26.04 LTS
RAM asignada	4096 MB
CPUs asignadas	4
Disco virtual	20 GB (reservado dinámicamente)
Controlador gráfico	VBoxVGA
Red	Adaptador puente (<i>bridged</i>)

El modo de red *adaptador puente* asigna a la VM una dirección IP propia en la red local, lo que permitirá en la Etapa C que la Raspberry Pi del banco físico se comunique directamente con el broker como cliente MQTT externo.

3.3.2. Instalación del broker Eclipse Mosquitto

Eclipse Mosquitto es la implementación de referencia del broker MQTT, de código abierto y ampliamente utilizada en aplicaciones IoT [27]. La instalación en Ubuntu se realiza mediante:

```
sudo apt install -y mosquitto mosquitto-clients python3-pip
```

La configuración mínima para aceptar conexiones sin autenticación (apropiada para desarrollo) se define en `/etc/mosquitto/conf.d/springlab.conf`:

```
listener 1883
allow_anonymous true
```

3.4. Verificación inicial del broker

Como primer paso de verificación, se comprobó el funcionamiento del broker mediante el intercambio de mensajes de texto plano entre un publisher y un subscriber de línea de comandos:

Terminal 1 | subscriber

```
mosquitto_sub -h localhost -t "hola/mundo" -v
```

Terminal 2 | publisher

```
mosquitto_pub -h localhost -t "hola/mundo" -m "Hola mundo"
```

La recepción inmediata del mensaje en Terminal 1 confirmó que el broker Mosquitto opera correctamente en el puerto 1883.

The image shows two terminal windows side-by-side. The left window (Terminal 1) is running a Mosquitto subscriber command: `mosquitto_sub -h localhost -t "hola/mundo" -v`. It shows a deprecation warning and then receives a message: `Publicando en springlab/force, /resistance, /status...`. The right window (Terminal 2) is running a Mosquitto publisher command: `mosquitto_pub -h localhost -t "hola/mundo" -m "Hola mundo"`. It also shows a deprecation warning and then displays a list of topics being monitored: `Escuchando topics de springlab...`. The topics listed are: `springlab/force`, `springlab/resistance`, and `springlab/status`. The publisher sends a message to `springlab/status` with the value `Hola mundo`.

Figura 3.2: Terminales comunicándose mediante Mosquitto.

3.5. Formato de mensaje: señales en JSON

Verificado el funcionamiento del broker, el siguiente paso consiste en definir la estructura de los mensajes que transportarán las señales del experimento.

Una primera aproximación empleó tres *topics* independientes con valores en texto plano (`springlab/force`, `springlab/resistance`, `springlab/status`). Si bien funcional, este enfoque presenta dos limitaciones: las variables de una misma medición pueden llegar desincronizadas a distintos suscriptores, y los valores carecen de metadatos que permitan interpretarlos sin documentación externa.

Para resolverlo, se adopta un único *topic*, `springlab/raw`, sobre el cual se publica un mensaje JSON completo por ciclo. El formato es autodescriptivo: cada mensaje contiene tanto las variables medidas como los metadatos necesarios para su trazabilidad. Un mensaje de ejemplo es:

```
1 {
2   "schema_version": "1.1",
3   "timestamp": "2026-06-15T20:40:54.808Z",
```

```

4  "source": "publisher_physics",
5  "spring_id": "real_7x38",
6  "cycle": 1,
7  "extension_mm": 70.0,
8  "force_N": 6.8368,
9  "voltage_mV": 1000.05,
10 "speed_mms": 10.0
11 }

```

Listing 3.1: Mensaje JSON publicado en el topic springlab/raw.

La justificación de cada campo se resume en la Tabla 3.3.

Cuadro 3.3: Estructura del mensaje JSON de cada señal.

Campo	Tipo	Descripción
schema_version	string	Versión del esquema; permite evolucionar el formato
timestamp	string	Instante de la medición (ISO 8601, UTC)
source	string	Origen del dato (publisher_physics o_hw)
spring_id	string	Identificador del resorte ensayado
cycle	int	Número de ciclo del experimento
extension_mm	float	Extensión aplicada al resorte [mm]
force_N	float	Fuerza medida o calculada [N]
voltage_mV	float	Voltaje [mV] (señal provisional)
speed_mms	float	Velocidad de actuación [mm/s]

El campo `schema_version` permite que futuras versiones del sistema añadan campos sin invalidar los datos ya almacenados; el campo `source` distingue datos simulados de datos del banco real, lo que será esencial en la Etapa C; y la consolidación de todas las variables en un único mensaje atómico garantiza que la fuerza, el voltaje y el ciclo de una misma medición se almacenen de forma coherente.

3.6. Redis como buffer intermedio

3.6.1. Función de Redis en la cadena de datos

Entre el transporte (MQTT) y la persistencia (InfluxDB) se interpone Redis como **buffer intermedio**. La necesidad de ese elemento no es evidente de entrada, ya que el servicio de ingesta podría escribir directamente en la base de datos, y por eso conviene establecer qué aporta:

- **Desacople:** la velocidad a la que entran los datos se independiza de la velocidad a la que la base de datos los escribe.

- **Tolerancia a fallos:** si InfluxDB o el proceso de escritura fallan, los mensajes permanecen en el buffer y se procesan al restablecerse el servicio, sin pérdida de datos.
- **Soporte para múltiples consumidores:** distintos procesos pueden consumir el mismo flujo de forma independiente, habilitando la futura incorporación de un motor de modelos.

3.6.2. Redis Streams y consumer groups

Un *Redis Stream* es una estructura de datos que funciona como un registro (*log*) ordenado y persistente de mensajes, cada uno identificado por un ID temporal único. A diferencia del transporte MQTT, que es efímero, el Stream conserva los mensajes y permite su reprocesamiento.

Sobre el Stream se define un *consumer group*, mecanismo que garantiza el procesamiento confiable mediante confirmación explícita. El ciclo de vida de cada mensaje es:

1. **XREADGROUP:** el consumidor solicita el siguiente mensaje no confirmado.
2. El consumidor procesa el mensaje (lo escribe en InfluxDB).
3. **XACK:** el consumidor confirma el procesamiento exitoso.

Hasta que no se ejecuta **XACK**, Redis mantiene el mensaje en una lista de *pendientes*. Si el consumidor falla entre los pasos 1 y 3, al reiniciar reclama sus mensajes pendientes y los reprocesa, garantizando que ningún dato se pierda.

Cuadro 3.4: Comandos de Redis Stream empleados en el sistema.

Comando	Función
XADD	El <i>ingestor</i> agrega un mensaje al Stream
XGROUP CREATE	Crea el <i>consumer group</i> (una sola vez)
XREADGROUP	El <i>consumer</i> lee el próximo mensaje no confirmado
XACK	El <i>consumer</i> confirma el procesamiento
XLEN	Consulta la cantidad de mensajes en el Stream
XINFO GROUPS	Consulta el estado del grupo (pendientes, <i>lag</i>)

El Stream se configura con una longitud máxima (**maxlen**) de 2000 mensajes, operando como buffer FIFO acotado. Considerando que un ciclo del banco real, a la velocidad de actuación del motor NEMA17 con un retardo de paso de 500 μ s, toma del orden de catorce segundos, este buffer cubre con holgura cualquier desfase transitorio entre la ingesta y la persistencia.

3.7. Persistencia en InfluxDB

InfluxDB es una base de datos optimizada para *series temporales*, es decir, datos indexados por marca de tiempo. Esta característica la hace idónea para registrar la evolución de las señales del experimento de fatiga a lo largo de los ciclos.

Los datos se almacenan en el *bucket* `raw_signals` mediante un *measurement* homónimo. Cada punto se estructura con:

- **Tags** (campos indexados, para filtrado): `spring_id`, `source`.
- **Fields** (valores numéricos): `cycle`, `extension_mm`, `force_N`, `voltage_mV`, `speed_mms`.
- **Timestamp**: tomado del campo `timestamp` del mensaje JSON.

Inmutabilidad del conjunto de datos. Una decisión de diseño central es preservar en `raw_signals` exclusivamente los datos crudos de la fuente, sin transformación. Esto garantiza un conjunto de datos confiable e inmutable, condición necesaria para el objetivo de generar *datasets* reproducibles para aplicaciones de aprendizaje automático. Las eventuales salidas de modelos (físicos o de ML) se almacenarán por separado, evitando la contaminación del dato original; en esta etapa se contempló un *bucket* distinto para ello, decisión que la Etapa C revisa (Sección 4.2).

table	_measurement	_field	_value	_start	_stop	_time	source	spring_id
raw	string	string	float64	string	string	dateTime RFC3339	string	string
0	raw_signals	force_N	6.7594625	2026-06-26T03:24:10.359Z	2026-06-26T03:39:10.359Z	2026-06-26T03:38:00.000Z	publisher_physics	real_7x3B

Figura 3.3: Verificación de la persistencia en InfluxDB mediante el Data Explorer. Se observan los registros del *measurement* `raw_signals` con su fuente, identificador de resorte y valores por campo.

3.8. Visualización con Grafana

Grafana provee la capa de visualización del sistema, conectándose a InfluxDB como fuente de datos mediante el lenguaje de consulta *Flux*. Se construyó un panel (*dashboard*) con tres gráficas de series temporales: fuerza, voltaje y ciclo.

Cada panel ejecuta una consulta Flux que filtra el *measurement* y el campo correspondiente sobre una ventana temporal. A modo de ejemplo, la consulta del panel de fuerza es:

```

1 from(bucket: "raw_signals")
2   |> range(start: v.timeRangeStart, stop: v.timeRangeStop)
3   |> filter(fn: (r) => r._measurement == "raw_signals")
4   |> filter(fn: (r) => r._field == "force_N")

```

```

5 | |> aggregateWindow(every: v.windowPeriod, fn: last,
6 |                     createEmpty: false)

```

Listing 3.2: Consulta Flux del panel de fuerza en Grafana.

La función `aggregateWindow` con `fn: last` toma el último valor de cada ventana temporal, evitando el promediado que distorsionaría la lectura del valor real de cada ciclo.



Figura 3.4: Panel de Grafana con las tres señales del experimento (fuerza, voltaje y ciclo). La curva de fuerza muestra la degradación progresiva por fatiga a lo largo de 1000 ciclos.

Interpretación de las gráficas. Es importante precisar qué representa cada punto: cada dato corresponde a un *ciclo completo* del experimento, reportándose la fuerza en el punto de máxima extensión (70 mm). Dado que esta extensión supera el límite elástico calculado del resorte (66,6 mm), cada ciclo induce un daño incremental (deformación permanente y fatiga acumulada) que debilita progresivamente el resorte. En consecuencia, la fuerza máxima decae a lo largo de los ciclos. Esta curva descendente constituye precisamente la *firma de la fatiga* que el mantenimiento predictivo busca caracterizar y, eventualmente, anticipar.

3.9. Integración del modelo físico

Habiendo validado la arquitectura con datos sintéticos, el siguiente paso consiste en sustituir la fuente por el modelo físico de la Etapa A. El *publisher* deja de emplear una fórmula lineal y pasa a invocar el objeto `Spring` de `physics.py`:

```

1 | spring = get_spring("real_7x38")

```

```

2 force_N = spring.force(70.0)
3 spring.end_cycle(70.0)

```

Listing 3.3: Integración del modelo físico en el publisher.

Con esta integración, la fuerza publicada ya no es una recta: refleja el comportamiento físico real, incluyendo la acumulación de deformación permanente (*plastic set*) y el daño por fatiga (modelo de Basquin) ciclo a ciclo.¹

Tratamiento del voltaje. A diferencia de la fuerza, el voltaje se mantiene aquí como señal sintética. La hipótesis inicial era medir con el sensor INA219 la variación de resistencia eléctrica del resorte al deformarse, y usarla como canal de degradación independiente de la mecánica. Al llevarla al banco real, la Etapa C descartó esa vía: las variaciones de resistencia involucradas son de una magnitud que el sensor disponible no resuelve, de modo que el canal no aporta información sobre el estado del componente y no vuelve a emplearse en el resto de la memoria.

La señal se conserva en la descripción de esta etapa porque su valor aquí no es físico sino arquitectónico: el esquema de mensaje transporta tres magnitudes, y ejercitar la cadena de datos con más de una señal es lo que permite verificar que el formato, la ingesta y la persistencia manejan correctamente registros multivariable. Ese es el propósito de la Etapa B. Como canal de medición, en cambio, queda descartado, y el hueco que deja (disponer de una modalidad independiente de la mecánica) se retoma como trabajo futuro en la forma de un canal de emisión acústica (Capítulo 6).

```

springlab_consumer | msg 1782445378820-0 | ciclo 1 | F=6.8368 N -> InfluxDB OK, XACK
springlab_consumer | msg 1782445379330-0 | ciclo 2 | F=6.8049 N -> InfluxDB OK, XACK
springlab_consumer | msg 1782445380001-0 | ciclo 3 | F=6.791 N -> InfluxDB OK, XACK
springlab_consumer | msg 1782445380565-0 | ciclo 4 | F=6.78 N -> InfluxDB OK, XACK
springlab_consumer | msg 1782445381057-0 | ciclo 5 | F=6.7705 N -> InfluxDB OK, XACK
springlab_consumer | msg 1782445381578-0 | ciclo 6 | F=6.7621 N -> InfluxDB OK, XACK
springlab_consumer | msg 1782445382094-0 | ciclo 7 | F=6.7544 N -> InfluxDB OK, XACK
springlab_consumer | msg 1782445382602-0 | ciclo 8 | F=6.7472 N -> InfluxDB OK, XACK
springlab_consumer | msg 1782445383107-0 | ciclo 9 | F=6.7405 N -> InfluxDB OK, XACK
springlab_consumer | msg 1782445383643-0 | ciclo 10 | F=6.7341 N -> InfluxDB OK, XACK
springlab_consumer | msg 1782445384159-0 | ciclo 11 | F=6.7281 N -> InfluxDB OK, XACK
springlab_consumer | msg 1782445384673-0 | ciclo 12 | F=6.7223 N -> InfluxDB OK, XACK

```

Figura 3.5: Logs del servicio de escritura (*consumer*). Cada línea muestra el identificador del mensaje, el ciclo procesado, la fuerza y la confirmación (XACK) de escritura exitosa en InfluxDB.

¹El identificador `real_7x38` corresponde a una entrada del catálogo cuyos parámetros son *derivados de la geometría* del resorte, no medidos sobre un ejemplar. Es la entrada que existía al construir la Etapa B y la que aquí se usa, lo cual es adecuado para su propósito: ejercitar la arquitectura de datos con una fuente físicamente plausible. Trabajos posteriores (Etapa C, Sección 4.5) reemplazan esta vía por parámetros medidos sobre los ejemplares reales, y `real_7x38` deja de usarse como referencia de ningún ensayo. Los valores que aparecen en esta sección deben leerse, por lo tanto, como ilustrativos de la cadena de datos, no como caracterización de un resorte.

Cuadro 3.5: Evolución de la fuerza, deformación permanente y daño por fatiga generados por el modelo físico (resorte `real_7x38`, extensión 70 mm).

Ciclo	Fuerza [N]	Plastic set [mm]	Daño [%]
1	6.8368	0.068	0.45
2	6.8049	0.135	0.63
5	6.7705	0.328	1.00
10	6.7341	0.625	1.41
20	6.6826	1.135	2.00
50	6.5820	2.172	3.16
200	6.3400	3.415	6.32
500	6.0800	3.415	10.00
1000	5.8129	3.415	14.14

3.10. Contenerización con Docker

3.10.1. Motivación

Hasta este punto, cada servicio (Mosquitto, Redis, InfluxDB, Grafana) y cada *script* (*publisher*, *ingestor*, *consumer*) se instalaba y ejecutaba de forma individual sobre el sistema operativo. Si bien adecuado para el aprendizaje y la depuración, este enfoque dificulta la reproducibilidad: la instalación manual es propensa a errores de versión y de configuración.

Docker resuelve esto encapsulando cada servicio en un *contenedor*: una unidad aislada que incluye el programa y todas sus dependencias. La herramienta *Docker Compose* orquesta el conjunto: un único archivo de configuración describe todos los servicios y sus interconexiones, y un solo comando levanta el sistema completo.

3.10.2. Arquitectura de contenedores

El archivo `docker-compose.yml` define seis servicios: los cuatro de infraestructura, contruidos a partir de imágenes oficiales, y los dos procesos propios (*ingestor*, *consumer*), contruidos a partir de un *Dockerfile* que parte de una imagen base de Python. El *publisher* se ejecuta de forma puntual mediante un contenedor efímero cuando se desea generar un experimento.

Comunicación interna. Dentro de la red de Docker, los contenedores se localizan entre sí por nombre en lugar de por dirección IP. Así, el *ingestor* se conecta a los servicios *mosquitto* y *redis* empleando dichos nombres, lo que elimina la dependencia de direcciones fijas y favorece la portabilidad.

Persistencia. Se definen volúmenes para InfluxDB y Grafana, de modo que los datos y

Cuadro 3.6: Servicios definidos en el `docker-compose.yml`.

Servicio	Imagen	Puerto	Rol
mosquitto	eclipse-mosquitto:2	1883	Broker MQTT
redis	redis:7	6379	Buffer (Stream)
influxdb	influxdb:2.7	8086	Persistencia
grafana	grafana/grafana:11	3000	Visualización
ingestor	(<i>Dockerfile</i> propio)	—	MQTT → Stream
consumer	(<i>Dockerfile</i> propio)	—	Stream → InfluxDB

la configuración del *dashboard* sobrevivan al reinicio o reconstrucción de los contenedores.

3.10.3. Operación

El sistema completo se levanta y se detiene con dos comandos:

```

1 # Construir y levantar todo el stack
2 docker compose up -d --build
3
4 # Detener todo el stack
5 docker compose down

```

Listing 3.4: Despliegue del sistema completo con Docker Compose.

NAME	IMAGE	COMMAND	SERVICE	CREATED	STATUS	PORTS
springlab_consumer	etapa_b-consumer	"python consumer.py --"	consumer	6 days ago	Up 2 minutes	
springlab_grafana	grafana/grafana:11.1.0	"/run.sh"	grafana	6 days ago	Up 2 minutes	0.0.0.0:3000->3000/tcp, [::]:3000->3000/tcp
springlab_influxdb	influxdb:2.7	"/entrypoint.sh infl"	influxdb	6 days ago	Up 2 minutes	0.0.0.0:8086->8086/tcp, [::]:8086->8086/tcp
springlab_ingestor	etapa_b-ingestor	"python ingestor.py --"	ingestor	6 days ago	Up 2 minutes	
springlab_mosquitto	eclipse-mosquitto:2	"/docker-entrypoint.s"	mosquitto	6 days ago	Up 2 minutes	0.0.0.0:1883->1883/tcp, [::]:1883->1883/tcp
springlab_redis	redis:7	"redis-server"	redis	6 days ago	Up 2 minutes	0.0.0.0:6379->6379/tcp, [::]:6379->6379/tcp

Figura 3.6: Estado de los seis contenedores del sistema tras su despliegue. Los servicios de infraestructura exponen sus puertos; el *ingestor* y el *consumer* operan en la red interna.

Reproducibilidad. La contenerización permite que el sistema completo se despliegue en cualquier máquina con Docker mediante un único comando, sin instalación manual de dependencias. Esto garantiza la reproducibilidad del entorno experimental, principio fundamental para la validación del trabajo y su transferencia.

3.11. Verificación de tolerancia a fallos

Para validar la robustez del buffer, se realizó una prueba de tolerancia a fallos. Con el consumidor detenido, se ejecutó el *publisher*, de modo que los mensajes se acumularon en el Stream de Redis sin alcanzar InfluxDB. El estado del *consumer group*, consultado mediante `XINFO GROUPS`, mostró un rezago (*lag*) igual al número de mensajes acumulados. Al reiniciar

el consumidor, este reclamó y procesó la totalidad de los mensajes pendientes, reduciendo el *lag* a cero sin pérdida de datos.

Esta prueba evidencia que el desacople mediante Redis protege la integridad del conjunto de datos frente a fallos transitorios de la capa de persistencia.

3.12. Posición del sistema en la taxonomía del Digital Twin

Al completar la Etapa B, el sistema avanza desde el estado *Pre-Shadow* de la Etapa A hacia la implementación de la arquitectura completa de un *Digital Shadow* según la taxonomía de Kritzinger et al. [13]: existe un flujo de datos automatizado desde una fuente hacia la representación virtual, mediado por el pipeline de datos.

La Tabla 3.7 resume la correspondencia entre las etapas de desarrollo y los niveles de integración del Digital Twin.

Cuadro 3.7: Correspondencia entre las etapas de desarrollo y los niveles de integración del Digital Twin.

Etapa	Descripción	Nivel DT	Estado
A	Modelo físico del resorte	Pre-Shadow	Completa
B	Arquitectura de datos en <i>streaming</i>	Digital Shadow*	Completa
C	Banco físico integrado (sensores reales)	Digital Shadow	Pendiente
D	Retroalimentación al banco	Digital Twin	Pendiente

*La Etapa B implementa y valida la arquitectura completa de un Digital Shadow empleando el modelo físico como fuente. El Digital Shadow propiamente dicho, con flujo de datos desde el objeto físico real, se materializa en la Etapa C, cuando los sensores del banco reemplazan al modelo como fuente, sin modificaciones en el resto de la arquitectura.

3.13. Discusión

La implementación del pipeline de datos de la Etapa B aporta las siguientes ventajas estructurales al sistema:

Desacoplamiento temporal. El *publisher* y los consumidores no necesitan estar activos simultáneamente. El buffer de Redis retiene los mensajes durante las interrupciones, garantizando que no se pierdan datos.

Escalabilidad. Agregar un nuevo consumidor de datos (por ejemplo, un motor de modelos o un segundo *dashboard*) requiere únicamente añadir un nuevo *consumer group* al Stream de Redis, sin modificar el *publisher*, el broker ni los consumidores existentes.

Independencia del medio físico. El mismo pipeline puede recibir mensajes del modelo simulado (Etapa B) y del banco real (Etapa C) usando exactamente el mismo protocolo y formato de mensaje, lo que simplifica la transición entre etapas. Solo se reemplaza el

publisher; el resto de la arquitectura permanece intacto.

Separación entre dato observado e inferido. El esquema mantiene el dato crudo del instrumento (`raw_signals`) separado de toda salida de modelo, de modo que ningún proceso de ajuste puede sobrescribir la medición. Esto preserva la integridad del *dataset* experimental y habilita la comparación objetiva entre distintos modelos (físico, de regresión, de red neuronal) sobre los mismos datos, sin necesidad de repetir el experimento físico. La forma concreta que adopta esta separación al desplegarse en la nube se precisa en la Etapa C (Sección 4.2).

Independencia entre el banco y la arquitectura de datos. Un fallo en el experimento físico (detención del motor, rotura del resorte) no compromete la arquitectura: el pipeline simplemente deja de recibir datos y los reanuda al restablecerse la fuente. Recíprocamente, el buffer protege los datos frente a fallos de la capa de persistencia, como se verificó en la prueba de tolerancia a fallos (Sección 3.11).

ETAPA C: BANCO FÍSICO INTEGRADO Y DIGITAL SHADOW

4.1. Introducción y objetivos de la etapa

Con el modelo físico calibrable de la Etapa A y la arquitectura de datos en *streaming* de la Etapa B ya validadas, la Etapa C cierra el bucle entre banco físico y representación virtual. La fuente de datos deja de ser el publicador sintético alimentado por `physics.py` y pasa a ser el propio instrumento: la Raspberry Pi que supervisa al microcontrolador LilyGO ESP32, publicando ciclo a ciclo las señales adquiridas por la celda de carga HX711.

Esta etapa, denominada **Etapa C — Banco físico integrado**, tiene por objetivos:

1. Desplegar el *stack* completo de la Etapa B en un servidor en la nube, accesible tanto desde la Raspberry Pi como desde el equipo de análisis, de modo que la representación virtual sea persistente e independiente de la disponibilidad local del banco.
2. Establecer la conectividad remota permanente entre la Raspberry Pi y la nube, con el instrumento como publicador de datos reales.
3. Definir un esquema de trazabilidad experimental que permita identificar inequívocamente cada corrida en InfluxDB y en los ajustes posteriores.
4. Calibrar el modelo físico contra el resorte real mediante un protocolo de dos fases (rampa *offline* y ajuste continuo en vivo).

5. Ejecutar un primer ensayo de fatiga sobre el banco real y contrastar la evolución medida de la rigidez frente a la predicción del modelo de Basquin con parámetros de literatura.

Al reemplazar el publicador simulado por el instrumento real, el sistema transita de un *Digital Shadow* demostrativo (cuya fuente es un modelo) a un *Digital Shadow* en sentido estricto según la taxonomía de Kritzinger *et al.* [13]: existe un flujo automatizado y unidireccional desde el objeto físico hacia su representación virtual, capaz de reflejar el estado real del componente en tiempo cuasi-real.

4.2. Despliegue del *stack* en la nube

El *stack* definido en la Etapa B (Mosquitto, Redis, InfluxDB, Grafana, más los servicios de ingesta y consumo) se migró desde la máquina virtual local hacia un servidor en la nube. La infraestructura elegida es un *droplet* de DigitalOcean con sistema operativo Debian, 4 GB de memoria RAM y una dirección IP pública estática. El orquestador es Docker Compose, manteniendo el mismo grafo de servicios y el mismo esquema de red interno utilizado en la etapa anterior.

Cuadro 4.1: Servicios desplegados en la nube (proyecto `etapa_B`).

Servicio	Imagen base	Rol
mosquitto	eclipse-mosquitto:2	Broker MQTT (1883 y 8883/TLS)
redis	redis:7	Buffer de <i>stream</i>
influxdb	influxdb:2	Persistencia de series de tiempo
grafana	grafana/grafana	Visualización
ingestor	python:3.11	MQTT → Redis
consumer	python:3.11	Redis → InfluxDB
fit_worker	python:3.11	Ajuste continuo del modelo por ciclo
monitor_worker	python:3.11	Vigilancia de degradación y detención remota

Exposición de servicios. Éste es el primer punto en que el despliegue en la nube deja de ser una copia del de la Etapa B. Allí, sobre una máquina virtual local sin exposición a Internet, todos los servicios publicaban sus puertos indistintamente (Figura 3.6). En la nube eso no es admisible: de todos los servicios, sólo el *broker* MQTT se publica hacia el exterior, porque es el único que la Raspberry Pi necesita alcanzar. Redis, InfluxDB y Grafana se enlazan exclusivamente a la interfaz local (127.0.0.1) del servidor, y el acceso al panel de Grafana se hace por túnel SSH por sesión. La decisión de enlazar a la interfaz local, y no sólo de bloquear puertos en el *firewall*, es deliberada: las reglas que Docker inserta directamente en

la tabla de *routing* del sistema pueden eludir la cadena de entrada del *firewall*, de modo que el enlace mismo, y no la regla, es el control efectivo.

Organización de los datos. A diferencia de lo previsto en la Etapa B, la persistencia se organiza en un *único bucket* de InfluxDB, **raw_signals**, con dos *measurements* distintas dentro de él:

- **raw_signals** contiene las muestras crudas del instrumento (extensión, fuerza, voltaje, ciclo), una fila por muestra.
- **model_params** reúne todo lo que el sistema *infiere*: parámetros ajustados, predicciones del modelo y diagnósticos de los servicios de decisión.

La separación por *measurement* cumple el mismo objetivo que la separación por *bucket* originalmente planteada, esto es, impedir que un proceso de ajuste sobrescriba el dato experimental, pero permite consultar ambas series en un mismo panel de Grafana sin cruzar orígenes de datos.

4.3. Conectividad Raspberry Pi hacia nube

Con el servidor cloud en producción, el banco físico debe alcanzar el *broker* MQTT desplegado en la nube. El publicador que en la Etapa B era el proceso simulado **publisher_physics** se reemplaza por **control.py**, el supervisor que corre en la Raspberry Pi: es él quien dialoga por puerto serie con el LilyGO ESP32, arma el mensaje JSON (**schema_version** = 1.1, sin cambios respecto de la Etapa B) y lo publica en el mismo topic **springlab/raw**. El campo **source** del mensaje distingue el origen de cada punto (**instrument_pi** frente al publicador sintético), lo que permite convivir datos simulados y reales en la misma base sin duplicar almacenamiento.

4.3.1. Trayecto físico del enlace

La Raspberry Pi no tiene ruta propia a Internet: está en un segmento de red industrial aislado. El enlace se resuelve, por lo tanto, con un salto intermedio a través de una pasarela del laboratorio que sí tiene salida al exterior. Sobre esa pasarela se establece un túnel SSH que reenvía el puerto del *broker* desde la nube hasta la propia Pi, de modo que **control.py** se conecta siempre a 127.0.0.1 y el direccionamiento real queda encapsulado en el túnel. El túnel se levanta como servicio de sistema con rearranque automático, para que sobreviva a reinicios y a cortes del enlace celular de la pasarela.

4.3.2. Seguridad del canal

El tramo pasarela→nube atraviesa una red pública, de modo que el canal MQTT se protege con **TLS mutuo**: una autoridad certificadora propia firma un certificado de servidor

y un certificado de cliente para la Pi, y el *broker* expone un segundo escuchador en el puerto 8883 que exige certificado de cliente válido *además* de usuario y contraseña. Ambas comprobaciones son independientes: un certificado válido no sustituye a la credencial.

Sobre esa base se define un esquema de **credenciales por rol** con listas de control de acceso (ACL) disjuntas en el *broker*. Este esquema es el que, en la Etapa D, permitirá abrir un canal de comando desde la nube hacia el banco sin ampliar los privilegios del instrumento:

Cuadro 4.2: Credenciales MQTT por rol y permisos sobre cada topic.

Credencial	Quién la usa	Permisos
<code>springlab.instrument</code>	Raspberry Pi	escribe <code>springlab/raw</code> ; lee <code>springlab/command/#</code>
<code>springlab.monitor</code>	vigilancia en la nube	escribe <code>springlab/command/#</code>
<code>springlab.rezero</code>	re-zero (Etapa D)	escribe <code>springlab/rezero/+</code>

La propiedad relevante es la asimetría: el instrumento puede *leer* el canal de comando pero no escribirlo, de modo que no puede fabricar sus propios comandos; y ningún servicio de la nube puede escribir en el canal de datos, de modo que no puede inyectar mediciones falsas.

4.4. Trazabilidad de experimentos

Cada corrida del banco se identifica con un `experiment_id` legible del tipo `{fecha}_{nombre}_{HHMMSS}` (por ejemplo, `2026-07-09_FatigueRun_01_092223`). Este identificador se registra como *tag* en las *measurements* de InfluxDB, tanto en la serie cruda `raw_signals` como en la serie de parámetros del modelo `model_params`, y permite filtrar en Grafana y en los ajustes *offline* todos los puntos correspondientes a un mismo experimento sin recurrir a rangos temporales.

Cuadro 4.3: *Tags* definidos en las *measurements* de InfluxDB.

Measurement	Tags	Uso principal
<code>raw_signals</code>	<code>spring_id</code> , <code>source</code> , <code>experiment_id</code>	muestras crudas
<code>model_params</code>	<code>spring_id</code> , <code>stage</code> , <code>experiment_id</code>	parámetros y diagnósticos

El campo `stage` en `model_params` identifica qué proceso escribió el punto, y por lo tanto cómo debe interpretarse. Sus valores se introducen a medida que aparecen los procesos que los escriben: **ramp** (calibración inicial, Sección 4.5), **fatigue** y **characterization** (ajuste por ciclo durante un ensayo), y ya en la Etapa D, **rezero** y **guard** para los dos lazos cerrados.

4.4.1. Nomenclatura de los resortes

A lo largo de esta memoria se ensayan varios resortes distintos, y conviene fijar la convención de nombres antes de que aparezcan los resultados. El identificador `spring_id` sigue el patrón `{diámetro}x{longitud libre}mm`, en milímetros, opcionalmente seguido de un sufijo que distingue el *ejemplar* o el rol experimental del resorte dentro de una campaña. Los tres tipos usados son:

Cuadro 4.4: Resortes ensayados y etapa en que aparece cada uno.

Tipo	Aparece en	Rol
8x31.5mm	Etapa C, Sección 4.5	Primer tipo caracterizado en profundidad (4 ejemplares). Fija los valores de referencia de k , x_e y η de la Tabla 4.5.
8.5x36.5mm	Etapa C, Sección 4.5	Segundo tipo caracterizado con el protocolo de rampa, para contrastar la dispersión entre tipos.
8x28.5mm	Etapa C, Sección 4.6; Etapa D	Tipo del primer ensayo de fatiga sostenido, de los tres niveles de la campaña LCF (<code>_deep1</code> , <code>_shallowA1</code> , <code>_shallowB1</code>) y del demostrador de lazo cerrado (<code>guard8x285</code>).

Los tres tipos provienen del mismo set comercial Yato YT-06875 descrito en la Etapa A, pero **no son intercambiables**: sus rigideces y límites elásticos difieren de forma importante, y ninguna cifra medida sobre uno debe transferirse a otro. Esta advertencia no es formal: es un resultado de la propia Etapa C, y su consecuencia práctica se desarrolla en la Sección 4.5.5.

4.5. Calibración del modelo con el resorte real

El modelo físico de la Etapa A depende de un conjunto de parámetros (k , x_0 , x_e , H , η , β , N_f , c) que en aquella etapa se tomaron de literatura o de un único ajuste preliminar. La Etapa C sustituye esos valores por estimaciones obtenidas del propio banco.

El procedimiento se divide en dos fases, y la razón de esa división es que los parámetros no son todos observables en las mismas condiciones. Unos describen la *forma* de la curva $F-x$, en particular dónde termina la zona elástica (x_e) y con qué pendiente continúa después (H). Esos sólo pueden medirse sobre un resorte que nunca haya sido llevado más allá de su límite elástico, porque el propio acto de cruzarlo deforma el codo que se quiere localizar: en un resorte ya fatigado el codo aparece corrido y suavizado, y el ajuste devuelve un valor que describe el daño acumulado antes que la propiedad original. De ahí que exista una **fase de rampa previa al ensayo**, ejecutada una única vez sobre el ejemplar virgen.

Los otros parámetros describen cómo el resorte *evoluciona* bajo carga cíclica, y por definición sólo pueden estimarse mientras el ensayo transcurre. Requieren por lo tanto una **fase de ajuste en vivo**, ciclo a ciclo, sobre datos que son ruidosos y cuya forma se desplaza continuamente. Esa segunda fase no vuelve a estimar x_e : lo recibe fijo de la primera, porque dejarlo libre sobre datos ya contaminados desplazaría la frontera elástico-plástica de la que depende toda la ley de daño, y el ajuste absorbería la degradación moviendo esa frontera en lugar de reportarla.

La consecuencia de esta división es que ambas fases deben compartir exactamente el mismo estimador, o los valores obtenidos en una no serían comparables con los de la otra.

El ajuste segmentado, herramienta común a ambas fases. Ese estimador común es el que se describe a continuación. El módulo `segmented_fit.py` ajusta, sobre una única carrera ascendente $F-x$, la forma completa que el modelo de la Etapa A predice: un tramo plano previo al contacto, un tramo lineal de Hooke y un tramo de endurecimiento con transición suave. Opera en dos modos:

- **Modo libre.** Ajusta los seis parámetros $\{x_0, k, x_e, H, w, \eta\}$ simultáneamente. Requiere una carrera *virgen*, es decir, un resorte que nunca haya sido llevado más allá de su límite elástico; de lo contrario el codo elasto-plástico ya está contaminado y x_e y H dejan de ser identificables.
- **Modo restringido.** Fija x_e en el valor ya conocido y ajusta sólo $\{x_0, k\}$. Es el modo usado ciclo a ciclo durante un ensayo, donde los datos son ruidosos y en movimiento: dejar que x_e se mueva desplazaría la frontera elástico-plástica de la que depende toda la ley de daño.

Todo el resto del sistema (calibración inicial, ajuste en vivo y los diagnósticos de la Etapa D) se apoya en esta misma rutina, lo que hace que k se mida siempre de la misma forma y sea comparable entre fases.

4.5.1. Fase de rampa (Stage 1)

Antes de iniciar cualquier ensayo de fatiga sobre un resorte nuevo, se ejecuta un ensayo de rampa: el actuador extiende el resorte de forma cuasiestática hasta una extensión máxima predefinida y retorna a la posición de referencia. El barrido genera una curva $F-x$ completa que se persiste en `raw_signals`. Un script de ajuste *offline*, `fit_ramp_params.py`, opera una sola vez por resorte tras la ingesta y recupera, en modo libre:

- la rigidez elástica k y la precarga x_0 del tramo lineal inicial;
- la extensión de límite elástico x_e como el punto en el que la curva se desvía de la recta de Hooke;

- la fracción de histéresis η , a partir del área encerrada por el lazo de subida y bajada;
- una estimación preliminar del coeficiente de endurecimiento H y del ancho de la zona de transición.

Los resultados se escriben en `model_params` con `stage = ramp` y quedan disponibles para su uso por los servicios en vivo. La transición entre zona elástica y plástica, que en la Etapa A se representaba con una discontinuidad de pendiente, se reemplazó por una transición suave tipo *softplus*, más consistente con la forma observada en los barridos reales.

4.5.2. Ajuste continuo por ciclo (Stage 2 y Stage 3)

Una vez conocidos los parámetros de rampa, el ensayo de fatiga se ejecuta con `fit_worker.py` suscrito al *stream* de ciclos. Por cada ciclo completo, el servicio:

1. extrae los extremos del ciclo desde InfluxDB;
2. recalcula la rigidez efectiva medida k_{measured} como la pendiente del tramo lineal del ciclo actual;
3. refina los parámetros de fatiga (N_f , c) y de deformación permanente (β) mediante un ajuste *regularizado hacia los valores de literatura*;
4. publica en `model_params` la rigidez predicha por el modelo con los parámetros actualizados, $k_{\text{predicted}}$.

Sentido de la regularización hacia la literatura. El paso 3 merece una aclaración, porque de otro modo el procedimiento sería inestable. En los primeros ciclos de un ensayo hay uno o dos puntos de datos, y ajustar tres parámetros de fatiga contra tan poca información produce valores absurdos que además saltan violentamente de un ciclo al siguiente. La solución adoptada es no dejar que el ajuste parta de cero: se toma el valor de literatura como punto de partida y el resultado del ajuste se calcula como un promedio ponderado entre ese valor y lo que sugieren los datos. El peso del valor de literatura equivale a un número fijo y pequeño de “ciclos imaginarios” (del orden de cinco), de modo que con pocos datos reales domina la literatura y el sistema se comporta de forma estable, mientras que a medida que se acumulan ciclos el peso relativo de la literatura se diluye y son las mediciones las que determinan el resultado. Es la forma más simple de lo que en estadística bayesiana se denomina *encogimiento hacia el prior*: la creencia previa gobierna mientras no haya evidencia, y cede terreno a medida que ésta aparece.

Toda esta lógica corre de forma completamente automatizada en el servidor en la nube durante el experimento, sin intervención manual del operador.

4.5.3. Vigilancia del ensayo: `monitor_worker.py`

Junto al ajuste anterior corre un segundo servicio, `monitor_worker.py`, cuya función no es estimar sino *vigilar*. Un ensayo de fatiga puede durar horas sin supervisión humana, y un resorte que se degrada más rápido de lo esperado o un montaje que falla pueden arruinar la corrida sin que nadie lo note hasta el día siguiente. El servicio reajusta k en cada ciclo, con la misma regularización descrita arriba, y sigue su tendencia a lo largo de la corrida; si la rigidez cae por debajo de una fracción configurable de su valor de referencia, publica un comando de detención (STOP) en el topic `springlab/command/{experiment_id}` usando la credencial `springlab_monitor` de la Tabla 4.2. La Raspberry Pi, que está suscrita a ese topic, detiene el ensayo.

Con este servicio queda instalado el primer canal de comunicación que va *desde* la nube *hacia* el banco, y es la pieza de infraestructura sobre la que se construirá el lazo cerrado de la Etapa D. El canal está desplegado, tiene credencial y permisos propios, y se empleó efectivamente para detener corridas de forma remota.

Su alcance, sin embargo, es limitado en dos sentidos que importan para lo que sigue. El primero es que su acción es de una sola clase: abortar. No modifica el ensayo en curso, lo termina, de modo que ninguna magnitud del componente queda bajo control del modelo. El segundo es que el disparo automático nunca llegó a ocurrir: en ninguna corrida de esta memoria la rigidez cayó lo suficiente para cruzar el umbral, y las detenciones que hubo fueron órdenes del operador enviadas a mano por ese mismo canal. Por ambas razones la Etapa C se clasifica todavía como *Digital Shadow*: existe la infraestructura para un retorno automatizado, pero no un lazo que opere sobre el estado del componente (Sección 5.8).

Cabe agregar que este vigilante no estuvo activo durante la campaña de la Etapa D. Su reajuste continuo de la rigidez es incompatible con una campaña cuyo objeto es precisamente medir cómo esa rigidez evoluciona, de modo que se mantuvo apagado mientras ésta se ejecutaba.

Este vigilante y la guarda elástica de la Etapa D no son, además, el mismo detector aplicado a distinta escala. El vigilante busca un **deterioro lento**: reajusta la rigidez ciclo a ciclo y observa si la sucesión de valores va cayendo a lo largo de cientos de ciclos. La guarda busca un **evento puntual**: mantiene una referencia fija y observa si la fuerza medida se aparta de ella en unas pocas muestras *dentro de una misma carrera* del carro. El primero observa una tendencia de horas; el segundo, un desvío de segundos.

4.5.4. Parámetros del resorte real

La Tabla 4.5 resume los parámetros ajustados sobre el resorte 8x31.5mm, el primer tipo caracterizado en profundidad. Cuatro ejemplares independientes permitieron estimar la dispersión de los parámetros con significancia estadística; los coeficientes de la ley de Basquin no pueden aún calibrarse experimentalmente y se mantienen en los valores de literatura.

Cuadro 4.5: Parámetros calibrados sobre el resorte 8x31.5mm.

Parámetro	Valor	n	Origen
k	$0,0745 \pm 0,0063 \text{ N mm}^{-1}$	4	Experimental
x_e	134–150 mm	4	Experimental
H	0,217	1	Experimental (preliminar)
w (transición)	20,0 mm	1	Experimental (preliminar)
η	$0,0157 \pm 0,0020$	4	Experimental
N_f	5×10^4 ciclos	—	Literatura
c	0,5	—	Literatura
β	0,02	—	Literatura

4.5.5. Cada resorte tiene su propio límite elástico

La observación más consecuente de toda la campaña de calibración no es ninguno de los valores de la tabla anterior, sino qué ocurre al medir el *mismo* parámetro sobre dos resortes distintos del *mismo* tipo, sacados de la misma caja. La cuestión, de orden estrictamente práctico, es si los parámetros caracterizados sobre un ejemplar son transferibles al siguiente o si cada resorte debe medirse por separado.

Al comparar dos ejemplares del tipo 8x28.5mm, el resultado se parte en dos:

- **La rigidez elástica k y el endurecimiento H sí se pueden reutilizar.** Los dos ejemplares coincidieron dentro del $\pm 0,4\%$ en k y del $\pm 0,8\%$ en H . Son, a efectos prácticos, el mismo resorte: medir uno equivale a medir el otro.
- **El límite elástico x_e no se puede reutilizar.** Los mismos dos ejemplares arrojaron 71,7 mm y 87,7 mm. Es una diferencia del $\sim 20\%$, unas veinte veces mayor que la de k , obtenida del mismo ensayo y con el mismo procedimiento de ajuste.

La comparación entre ambos casos es lo que da peso al resultado. Si esos 16 mm de diferencia fueran simplemente ruido de medición, la dispersión debería aparecer también en k y en H , que se estiman del mismo barrido y con el mismo código; no aparece. La interpretación natural es que **dónde empieza a ceder el material es una característica de cada resorte individual** y no del lote, dependiente de su historia particular de fabricación, almacenamiento y manipulación.

Este argumento no es, sin embargo, concluyente, y conviene declarar por qué. Los tres parámetros se estiman del mismo ajuste pero no con la misma sensibilidad: k y H son pendientes recuperadas de tramos completos de la curva, mientras que x_e es la abscisa de un codo, y localizar un codo es intrínsecamente más frágil que ajustar una recta. Que k y H concuerden no excluye por completo, entonces, que parte de los 16 mm provenga del procedimiento de ajuste. Con dos ejemplares por lado, la evidencia disponible no permite

cerrar la cuestión, y la determinación definitiva requeriría un tercer ejemplar. El resultado se adopta en lo que sigue como hipótesis de trabajo bien fundada, y esa condición basta para los dos usos que se le dan, porque ambos son *conservadores*: si la variación fuese menor que la observada, calibrar cada resorte por separado seguiría siendo correcto, sólo que menos necesario.

Con esa salvedad, la constatación atraviesa el resto de la memoria en dos frentes.

Consecuencia metodológica. Cada resorte debe calibrarse por separado antes de compararlo con otro. En particular, fijar una misma amplitud de ensayo en milímetros para dos ejemplares *no* los somete a la misma sollicitación: si uno empieza a plastificar a los 72 mm y el otro a los 88 mm, estirar ambos hasta 100 mm significa 28 mm de penetración plástica en el primero y 12 mm en el segundo. La variable que hay que igualar entre ensayos no es la extensión absoluta sino cuánto se penetra por sobre el x_e propio de cada resorte, punto que se desarrolla en la Sección 5.2.3.

Consecuencia de diseño. Cualquier mecanismo automático que deba impedir que un resorte entre en zona plástica **no puede basarse en un x_e heredado de otro ejemplar**: con un 20 % de error posible, actuaría demasiado tarde en un resorte y demasiado pronto en otro. Tiene que determinar el límite sobre el ejemplar que efectivamente tiene montado. Éste es, precisamente, el principio de diseño del lazo de guarda de la Etapa D (Sección 5.5.2), que por eso no acepta ningún parámetro de entrada y se calibra solo.

4.6. Caracterización experimental de la degradación

Con el *pipeline* calibrado se ejecutó el primer ensayo de fatiga sostenido sobre el banco real, con un ejemplar del tipo 8x28.5mm. La amplitud se fijó en $D = 100$ mm, es decir unos 12 mm por sobre el límite elástico $x_e = 87,7$ mm que la calibración de rampa había recuperado para ese tipo: un margen deliberadamente modesto, suficiente para entrar en zona plástica sin llevar el resorte a falla en pocas decenas de ciclos. La velocidad de ciclo se igualó a la de la propia calibración (1500 μ s por paso) para no introducir una variable de velocidad no ensayada.

Durante el ensayo, la Raspberry Pi publicó cada ciclo hacia el *broker* en la nube y los servicios *ingestor*, *consumer* y *fit_worker* operaron sin interrupción. Los paneles de Grafana permitieron seguir en vivo tanto la señal cruda como la evolución de k_{measured} y $k_{\text{predicted}}$ ciclo a ciclo. La corrida se detuvo manualmente cerca del ciclo 150, tras unas 2,5 horas, al constatar que la tendencia se había aplanado: continuar no aportaba información y sí consumía el resorte.

4.6.1. Rigidez medida frente a rigidez predicha

La Figura 4.1 superpone ambas curvas para toda la corrida.

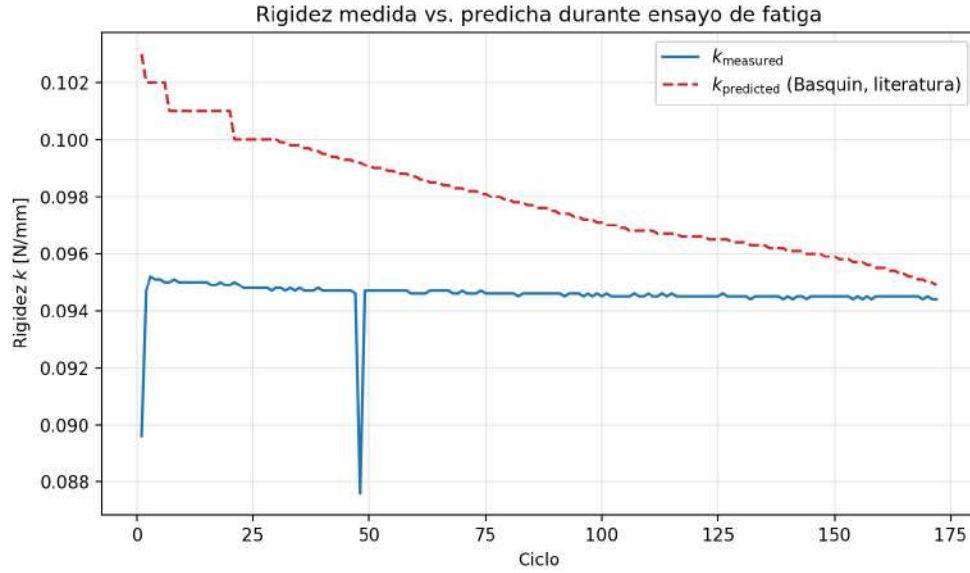


Figura 4.1: Rigidez medida k_{measured} frente a rigidez predicha $k_{\text{predicted}}$ para un resorte 8x28.5mm ciclado a $D = 100$ mm. La curva predicha corresponde al twin en vivo sembrado con los parámetros de fatiga de literatura ($N_f = 5 \times 10^4$, $c = 0,5$). El descenso aislado en torno al ciclo 48 es un artefacto de un ciclo cuyo ajuste degradó puntualmente, no un evento mecánico.

La lectura de la figura tiene tres elementos:

- **La rigidez medida apenas se degrada.** Tras un transitorio inicial de asentamiento (el resorte fresco lee más blando en el primer ciclo y sube hasta $\sim 0,0951 \text{ N mm}^{-1}$ hacia el ciclo 3), la curva declina de forma suave y monótona hasta $\sim 0,0944 \text{ N mm}^{-1}$ en el ciclo 149. La variación neta sobre la ventana estabilizada es del orden del $-0,7\%$: real, pero pequeña.
- **La predicción parte muy por encima y converge.** El twin, sembrado con la rigidez de la fase de rampa y los parámetros de Basquin de literatura, arranca prediciendo $\sim 0,1025 \text{ N mm}^{-1}$, un $\sim 14\%$ por sobre lo medido, y va cerrando esa brecha hasta situarse a $\sim 1,5\%$ hacia el ciclo 149.
- **La deformación permanente se satura.** El *permanent set* crece rápido al principio ($\sim 0,2$ mm por ciclo hasta el ciclo 34) y luego se frena en un orden de magnitud ($\sim 0,02$ mm por ciclo entre los ciclos 34 y 149), reproduciendo la forma de *shakedown* que el modelo de la Etapa A predice por su propia regla de acumulación autolimitada.

4.6.2. Interpretación

El primer elemento es el resultado empírico directo: con parámetros genéricos el modelo es **pesimista**. El valor $N_f = 5 \times 10^4$ tomado de literatura subestima la vida a fatiga real del resorte en el régimen de amplitud ensayado; predice una degradación bastante más acelerada de la que el componente exhibe. Esto pone en evidencia la necesidad de recuperar la relación *tasa de degradación* vs. *amplitud de ciclo* directamente de datos experimentales, en lugar de heredarla de la literatura. Recuperar esa relación de forma sistemática, sobre varias amplitudes distintas, es el objeto de la campaña experimental de la Etapa D (Sección 5.6.3).

El segundo elemento, la convergencia de la predicción hacia la medición a lo largo de la corrida, admite una lectura como validación del twin que no se sostiene. Identificar *qué* parámetro se movió para que ambas curvas se juntaran expone, en cambio, un defecto en la cadena de calibración:

- Durante el ensayo, el ajuste por ciclo mantiene x_e **congelado** en el valor que dejó la fase de rampa. Los únicos parámetros que el sistema tiene permitido mover son β , N_f y c .
- Ahora bien, x_e es justamente el parámetro que cambia de un resorte a otro (Sección 4.5.5). Si el valor congelado no es el del ejemplar montado, hay un error que el ajuste tiene que compensar de alguna forma, y sólo puede hacerlo moviendo los parámetros que sí están libres.
- Eso es exactamente lo que se observa: β terminó en $\approx 0,29$, cerca del doble del 0,16 que la caracterización de rampa había medido para este tipo de resorte. Ese valor, por lo tanto, no puede leerse como una medición independiente de la tasa de deformación permanente. Es, en buena parte, **el error de x_e reapareciendo bajo el nombre de β** .

En otras palabras: la calibración de rampa entrega para cada parámetro un valor *y* una incertidumbre, pero el twin en vivo se queda sólo con el valor *y* lo trata como si fuera exacto. Al descartar esa incertidumbre, el sistema pierde la capacidad de reconocer que su x_e podría estar mal, y el desajuste termina contaminando otro parámetro. De aquí salen las dos decisiones que estructuran la Etapa D: calibrar cada resorte desde su propio primer ciclo (Sección 5.2.7) y, en el lazo de guarda, no partir de ningún x_e heredado sino determinarlo sobre el ejemplar montado (Sección 5.5.2).

Como indicador secundario del mismo problema, la calidad del ajuste por ciclo se degradó de forma sostenida a lo largo de la corrida (r^2 de $\sim 0,99$ a $\sim 0,88$): el modelo restringido describía cada vez peor una curva F - x que se desplazaba bajo él. La causa mecánica de ese desplazamiento se aborda a continuación.

4.6.3. Retorno a cero y compresión residual

Un segundo hallazgo de esta corrida es de naturaleza puramente mecánica y condiciona todo lo que viene después. El cero mecánico del banco es una posición fija en el eje, determinada por el sensor Hall; el firmware devuelve el carro a esa posición al final de cada ciclo. Mientras el resorte no acumule deformación permanente, ese cero coincide con su longitud libre y el retorno lo deja simplemente destensado.

En cuanto aparece *permanent set*, sin embargo, la longitud libre del resorte crece mientras el cero mecánico permanece donde estaba. El retorno a casa deja entonces de ser un retorno a fuerza nula: el resorte llega a su longitud libre *antes* de que el carro termine su recorrido, y los últimos milímetros lo empujan a compresión. Sobre un resorte de extensión eso se manifiesta como una fuerza medida negativa en la región baja de la carrera, del orden de -1 N tras unos 20 mm de deformación permanente acumulada, correspondiente al resorte pandeando ligeramente y no a un fallo del sensor.

Esta región, que en adelante se denomina **toe de compresión**, tiene dos consecuencias:

1. **Sobre el ajuste.** El modelo de la Etapa A describe un resorte en tracción y no puede representar fuerza negativa. Los puntos del *toe* deben excluirse explícitamente del ajuste, junto con un margen de guarda por sobre el cruce por cero, porque los primeros milímetros ya en tracción todavía arrastran la curvatura del pando. Sin esa exclusión, la región de compresión sesga la pendiente ajustada y puede introducir escalones espurios en la serie $k(N)$.
2. **Sobre la detección.** El *toe* produce, de forma perfectamente reproducible y sin que haya plastificación alguna, una desviación sistemática respecto de la recta de Hooke. Cualquier vigilante que compare fuerza medida contra un modelo elástico disparará ahí si no se le excluye esa zona. Es la razón de ser de la guarda espacial `TOE_GUARD` del lazo de la Etapa D (Sección 5.5.3).

4.7. Posición del sistema en la taxonomía del Digital Twin

Al completar la Etapa C el sistema alcanza el nivel de *Digital Shadow* en sentido estricto: la representación virtual (parámetros del modelo, series históricas, paneles de visualización) se actualiza automáticamente a partir del objeto físico (banco de fatiga con sensores reales), sin requerir intervención humana en el flujo de datos.

4.8. Discusión

La integración del banco físico con el *stack* en la nube consolida tres propiedades estructurales del sistema:

Cuadro 4.6: Correspondencia actualizada entre etapas y niveles del Digital Twin.

Etapas	Descripción	Nivel DT	Estado
A	Modelo físico del resorte	Pre-Shadow	Completa
B	Arquitectura de datos en <i>streaming</i>	Digital Shadow*	Completa
C	Banco físico integrado	Digital Shadow	Completa
D	Retroalimentación al banco	Digital Twin	Pendiente

*En la Etapa B el flujo era completo pero la fuente era el modelo; en la Etapa C la fuente es el instrumento físico.

Persistencia e independencia. La representación virtual ya no vive en la memoria de un proceso local: cada ciclo publicado por la Pi queda registrado en InfluxDB con su identificador de experimento, y sobrevive a reinicios del banco, del equipo de análisis o del propio servidor. Esta persistencia es condición necesaria para cualquier análisis posterior de vida útil remanente.

Comparabilidad modelo–realidad. La coexistencia de las series k_{measured} y $k_{\text{predicted}}$ en la misma base de datos habilita la comparación cuantitativa continua entre lo que el banco mide y lo que el modelo predice. El primer ensayo de fatiga sostenido ya expuso una brecha inicial del $\sim 14\%$ entre ambas (Sección 4.6), demostrando que la instrumentación es capaz de discriminar entre hipótesis del modelo; y, al examinar qué parámetro absorbía esa brecha durante el ensayo, expuso también un defecto en la propia cadena de calibración que la Etapa D corrige por diseño.

Base para la Etapa D. La existencia de un *Digital Shadow* funcional es requisito para que la Etapa D pueda cerrar el lazo: solo cuando el modelo virtual dispone de datos actualizados del banco tiene sentido usarlo para tomar decisiones que retroalimenten al banco (por ejemplo, revertir la carrera cuando la propia curva $F-x$ medida se aparta de la recta elástica que el modelo mantiene para ese resorte).

En términos de la taxonomía adoptada, la Etapa C completa el flujo físico $\rightarrow$ virtual del Digital Shadow. Queda pendiente para la Etapa D el flujo inverso, virtual $\rightarrow$ físico, que caracteriza al Digital Twin propiamente dicho.

ETAPA D: LAZO CERRADO Y DIGITAL TWIN

5.1. Introducción y objetivos de la etapa

Al término de la Etapa C, el sistema exhibe un flujo automatizado y unidireccional desde el banco físico hacia su representación virtual: cada ciclo publicado por la Raspberry Pi se ingesta, se ajusta y se persiste en la nube, y los paneles de Grafana reflejan el estado real del componente en tiempo cuasi-real. Ese es el nivel de *Digital Shadow* en la taxonomía de Kritzinger *et al.* [13].

Para alcanzar el nivel de *Digital Twin* en sentido estricto de esa misma taxonomía es necesario cerrar el lazo en la dirección opuesta: la representación virtual debe poder **modificar automáticamente el estado del banco** en función de lo que observa. Sin ese flujo virtual $\rightarrow$ físico el sistema, por sofisticado que sea, sigue siendo un observador pasivo.

La Etapa D construye ese flujo de retorno y lo instancia en **dos lazos distintos**, que comparten infraestructura pero no propósito ni estructura de control:

- **Lazo de guarda elástica** (Sección 5.5). Es el lazo que define la etapa. El resorte se cicla nominalmente *dentro* de su zona elástica y se le inyecta deliberadamente una perturbación aleatoria de Poisson sobre la consigna de extensión, de modo que cada tanto un ciclo se estire más de lo nominal. El twin, corriendo en la nube, vigila cada muestra contra una recta elástica que él mismo ajustó al comienzo de la corrida y, cuando la fuerza medida se aparta de esa recta de forma sostenida, ordena al banco revertir la carrera y acotar todas las consignas futuras por debajo del punto de inicio detectado. El modelo virtual *protege*, en consecuencia, al componente físico

devolviéndolo a la zona elástica.

- **Lazo de re-zero** (Sección 5.6). Es un lazo integrador auxiliar, construido antes que el anterior, cuyo objetivo es sostener una campaña experimental de fatiga de bajo ciclo (LCF) manteniendo constante el sobreesfuerzo plástico por ciclo pese a la acumulación de deformación permanente. Es el lazo que produjo el resultado experimental de esta memoria: la relación empírica entre amplitud de ciclo y tasa de degradación de rigidez.

Los objetivos de la etapa son:

1. Introducir los conceptos físicos y de control que justifican la forma de ambos lazos (*permanent set*, *shakedown*, sobreesfuerzo plástico, control por desplazamiento vs. por fuerza, excitación estocástica).
2. Implementar los canales de comando y confirmación desde la nube hacia el banco, con credenciales y listas de control de acceso propias, y la lógica de decisión de cada lazo.
3. Demostrar el cierre efectivo del lazo virtual $\rightarrow$ físico sobre hardware real: detección en la nube y reacción mecánica verificada contra la telemetría del propio banco.
4. Ejecutar la campaña experimental que sólo el lazo cerrado permite, y recuperar la relación empírica entre amplitud de ciclo y tasa de degradación.
5. Actualizar la posición del sistema en la taxonomía del Twin Digital y delimitar el alcance de lo efectivamente demostrado.

5.2. Fundamentos físicos y de control necesarios

Los capítulos anteriores describieron el modelo físico del resorte, la arquitectura de datos y la calibración de parámetros. Esta sección introduce los conceptos adicionales necesarios para justificar la elección de ambos lazos.

5.2.1. Fatiga de bajo y de alto ciclo (LCF y HCF)

La literatura de fatiga distingue dos regímenes según el número de ciclos a falla:

- **Fatiga de alto ciclo** (*high-cycle fatigue*, HCF): la sollicitación se mantiene íntegramente en zona elástica. La vida a falla es del orden de 10^5 a 10^7 ciclos. Es el régimen operativo normal de un resorte industrial.
- **Fatiga de bajo ciclo** (*low-cycle fatigue*, LCF): cada ciclo penetra en zona plástica. El daño acumulado por ciclo es varios órdenes de magnitud mayor y la vida a falla cae a 10^2 – 10^4 ciclos. La norma ASTM E606 rige los ensayos LCF.

El régimen HCF, aunque representa la operación real del componente, es inalcanzable dentro del horizonte temporal de esta memoria: a una tasa de un ciclo cada ~ 51 s (limitada por el motor paso a paso del banco), 10^6 ciclos son cerca de 20 meses de operación continua.

Esta restricción temporal determina la forma del lazo de re-zero, que adopta el régimen LCF como atajo para *medir* una tasa de degradación en un horizonte razonable, aceptando que la ley obtenida deberá extrapolarse posteriormente al régimen HCF.

El lazo de guarda responde a un problema de naturaleza distinta. Su propósito no es medir una tasa sino demostrar que la representación virtual puede proteger al componente de una sobresolicitud *imprevista*, para lo cual necesita enfrentarse a un evento cuyo instante de ocurrencia sea desconocido. El régimen temporal del banco interviene aquí sólo de forma secundaria, para descartar que ese evento pueda simplemente esperarse: el cruce espontáneo hacia zona plástica por acumulación de daño está tan fuera del alcance temporal como el régimen HCF. La solución adoptada es provocarlo mediante una excitación aleatoria, cuyo fundamento se desarrolla en la Sección 5.2.5.

5.2.2. Deformación permanente y shakedown

Cada vez que el resorte se estira por sobre su límite elástico x_e , la zona plástica del material fluye y una fracción del sobreesfuerzo queda como deformación permanente. En el modelo de plasticidad de la Etapa A, la fracción β del sobreesfuerzo se convierte en incremento de longitud en reposo:

$$\Delta\ell_{\text{perm}} = \beta \cdot \Delta_{\text{plástico}}, \quad \Delta_{\text{plástico}} = x_{\text{máx}} - x_e, \quad (5.1)$$

donde $x_{\text{máx}}$ es la extensión máxima del ciclo y $\Delta_{\text{plástico}}$ el *sobreesfuerzo* por sobre el límite elástico.

La consecuencia es que la longitud en reposo del resorte crece monótonamente con cada ciclo. Visto desde el banco, cuyo cero mecánico es el sensor Hall (posición fija en el eje), el resorte se ha *alargado*: el mismo $x_{\text{máx}}$ ya no produce el mismo sobreesfuerzo, porque x_e efectivo se ha corrido en $\sum \Delta\ell_{\text{perm}}$.

Con $x_{\text{máx}} = D$ fijo, el sobreesfuerzo plástico disminuye ciclo a ciclo hasta anularse. Cuando eso ocurre el resorte deja de deformarse plásticamente y comienza a comportarse elásticamente en su nueva forma: este fenómeno se denomina **shakedown**. El daño por fatiga LCF cesa y el ensayo pasa efectivamente a un régimen HCF a la nueva geometría.

La curva roja de la figura merece una explicación, porque no describe un comportamiento del resorte sino el efecto de una intervención. El resorte se alarga igual en ambos casos: la deformación permanente es un hecho físico y no se puede evitar. Lo que cambia es qué hace el banco al respecto. Con consigna fija, el carro sigue yendo siempre hasta la misma posición absoluta D , de modo que a medida que la longitud en reposo del resorte crece, la porción de esa carrera que efectivamente lo lleva más allá del límite elástico se reduce, y el sobreesfuerzo

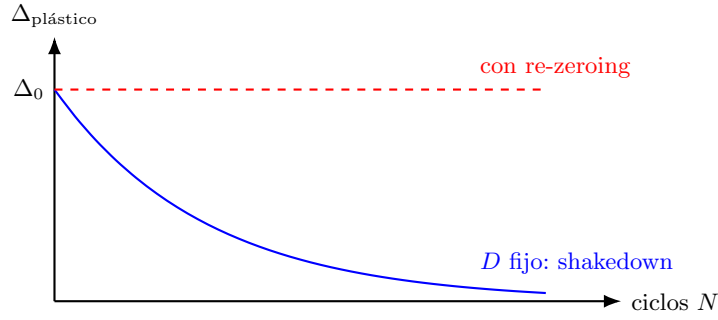


Figura 5.1: Evolución del sobreesfuerzo plástico $\Delta = x_{\text{máx}} - x_e$ en función del número de ciclos, para las dos formas de conducir el ensayo. Con la consigna fija (azul), el *permanent set* acumulado hace que Δ decaiga hasta anularse, momento en que cesa la deformación plástica (*shakedown*). Con re-zeroing (rojo), la consigna se corrige ciclo a ciclo en la misma cantidad en que el resorte se alargó, con lo que Δ permanece en su valor inicial Δ_0 y el ensayo mantiene la sollicitación que se propuso aplicar.

decae (curva azul). El re-zeroing consiste en medir cuánto se alargó el resorte y *correr la consigna en esa misma cantidad*, de manera que el carro llegue cada vez un poco más lejos y el sobreesfuerzo se mantenga constante (curva roja). En otras palabras, la curva roja no es lo que el resorte hace por sí solo, sino lo que el lazo cerrado consigue sostener; sin él, el ensayo recorrería la curva azul.

Este comportamiento es el que hace inviable un ensayo de caracterización a D fijo: la tasa de degradación por ciclo depende del sobreesfuerzo plástico, y ese sobreesfuerzo no es constante durante el ensayo. La implementación del lazo que sostiene la curva roja se describe en la Sección 5.6.

5.2.3. Sobreesfuerzo por sobre x_e individual como variable de estandarización

Al comparar tasas de degradación entre resortes distintos, la elección natural sería fijar la extensión D para todos. Es incorrecta por dos razones:

1. El propio límite elástico x_e varía entre resortes nominalmente iguales. En los ensayos de esta etapa se observaron valores de x_e entre 71,7 mm y 90 mm para resortes del mismo tipo (8x28.5mm), un rango del $\sim 20\%$.
2. El daño físico depende del sobreesfuerzo plástico, no de la extensión absoluta. Dos resortes con distinto x_e ciclados al mismo D están recibiendo cargas plásticas muy distintas.

Por consiguiente, la variable que estandariza el ensayo es el **sobreesfuerzo por sobre el x_e propio de cada resorte**, $\Delta = D - x_e(\text{resorte})$, medido individualmente antes de comenzar el ensayo mediante el ajuste *libre* del ciclo 1 (véase la Sección 5.2.7).

Ésta es la misma constatación de la Etapa C: el límite elástico es propiedad de cada resorte individual y no del tipo, mientras que la rigidez k sí puede darse por común a todos los ejemplares del lote. Y es también el argumento de diseño central del lazo de guarda de esta etapa, que por lo mismo no parte de ningún x_e conocido de antemano, sino que lo determina midiendo sobre el resorte que tiene montado (Sección 5.5.2).

5.2.4. Control por desplazamiento vs. control por fuerza

Los ensayos de fatiga clásicos operan en control por fuerza cerrado sobre una amplitud de carga fija (curvas S-N tradicionales). Este modo requiere un lazo de servocontrol de fuerza en tiempo real, lo que no es factible en la instrumentación disponible:

- El actuador es un motor paso a paso en lazo abierto: comanda posición, no fuerza.
- La celda de carga HX711 opera como canal de lectura lateral, con un ruido de fondo aún no caracterizado.

Se adopta entonces **control por desplazamiento** en ambos lazos: el actuador comanda siempre una extensión objetivo D , y la fuerza se usa exclusivamente como señal de medición. En el lazo de re-zero esa consigna se *corrige* periódicamente para compensar el drift de *permanent set*; en el lazo de guarda se *acota* de forma permanente una vez detectado el inicio de la plastificación. Ninguno de los dos requiere un lazo de fuerza en tiempo real. Esta elección es consistente con la práctica LCF descrita en ASTM E606.

5.2.5. Excitación estocástica de la consigna

5.2.5.1. Necesidad de un evento impredecible

El lazo de guarda existe para proteger al resorte de un resultado no deseado, el ingreso en zona plástica, actuando sobre el ensayo antes de que el daño se consolide. Demostrar esa capacidad exige someter al sistema al evento contra el cual protege. La forma en que ese evento se produzca, sin embargo, determina si la demostración significa algo.

El criterio que decide es que **un evento agendado no requiere un twin**. Si el instante del cruce se conoce de antemano, protegerse de él no exige vigilar nada: basta con programar la reversión en el ciclo correspondiente. La representación virtual sólo se vuelve necesaria cuando el momento del evento es desconocido, porque entonces la única vía para actuar a tiempo consiste en observar la señal medida e inferir de ella que el cruce ha comenzado. La aleatoriedad de la excitación no es, por lo tanto, una comodidad experimental. Es la condición que convierte la demostración en una demostración.

De ahí que la rampa monótona quede descartada, pese a ser el procedimiento más simple y a garantizar el cruce. Es el método que la Etapa C emplea para caracterizar x_e , y allí resulta adecuado precisamente porque el cruce está agendado y lo que se busca es medirlo. Aquí produciría un detector que confirma lo que el operador ya sabía.

Descartada la vía determinista, queda por resolver por qué el evento debe inducirse en lugar de esperarse. El cruce espontáneo hacia zona plástica por degradación acumulada ocurre, en operación elástica normal, en una escala de 10^5 a 10^7 ciclos, lo que a la velocidad del banco equivale a meses de operación continua; esperarlo está fuera del horizonte de esta memoria por las mismas razones que el régimen HCF. La vía adoptada es, en consecuencia, provocar el evento mediante un proceso estocástico conocido, de modo que el cruce ocurra dentro de una corrida acotada pero en un instante que ni el operador ni el detector pueden anticipar.

5.2.5.2. Elección de la distribución de la perturbación

Elegida la vía estocástica, queda por decidir *qué* distribución. La elección no es indiferente, porque determina la forma de la secuencia de consignas y, con ella, si el experimento mide lo que pretende medir. Se evaluaron tres familias.

Ruido gaussiano sobre la consigna. La opción por defecto sería $D_i = x_{\text{nom}} + \varepsilon_i$ con ε_i normal. Se descarta por tres motivos independientes:

1. **Perturba todos los ciclos.** Una gaussiana no tiene átomo de probabilidad en cero: *ningún* ciclo queda exactamente en x_{nom} . El twin necesita ciclos limpios en la amplitud nominal para autocalibrar su recta de referencia y aprender su envolvente de residuo (Sección 5.5.2); con ruido gaussiano permanente esa referencia se construye sobre datos ya contaminados, y no existe ningún valor de σ que devuelva ciclos limpios. Con la Poisson, en cambio, la limpieza de la fase de calibración se obtiene sin coordinación alguna con el banco, por el solo hecho de que $e^{-\lambda}$ de los ciclos quedan sin perturbar.
2. **Es simétrica.** La mitad de las perturbaciones acortan la carrera en lugar de alargarla. Esos ciclos no aportan nada al experimento, puesto que no pueden producir el evento buscado, y sólo diluyen la muestra.
3. **No separa frecuencia de profundidad.** La desviación σ controla simultáneamente cada cuánto se cruza x_e y cuán profundo se penetra. No se pueden fijar de forma independiente, que es justamente lo que el diseño del ensayo requiere.

Distribución exponencial o uniforme sobre el desplazamiento. Corrigen la simetría, al ser unilaterales, pero conservan el problema de fondo: producen un *continuo* de amplitudes. Las consignas se distribuyen sobre todo el rango entre x_{nom} y el máximo, sin separación entre “ciclo normal” y “excursión”. Eso rompe la interpretación del resultado: ante un disparo del

detector no se puede afirmar que se detectó un evento excepcional, porque no hay tal cosa como un evento excepcional en una distribución continua, sino sólo el extremo superior de un espectro. Y, de nuevo, el ajuste de referencia se construiría sobre ciclos todos distintos entre sí.

Proceso de Poisson compuesto. Es la elección adoptada. En cada ciclo i se sortea un *número de empujones* N_i según una distribución de Poisson, y la consigna se desplaza en N_i veces un empujón elemental δ de tamaño fijo:

$$D_i = x_{\text{nom}} + N_i \cdot \delta, \quad N_i \sim \text{Poisson}(\lambda). \quad (5.2)$$

La distribución de Poisson es, por definición, la que describe el **número de eventos raros e independientes que ocurren en un intervalo**, y esa es exactamente la estructura del fenómeno que se quiere imitar: sobresolicitaciones ocasionales e impredecibles sobre un componente que opera normalmente dentro de su rango. Sus propiedades resuelven, una por una, las objeciones anteriores:

1. **La mayoría de los ciclos queda intacta.** A diferencia de cualquier distribución continua, la Poisson concentra una masa de probabilidad finita en el cero: $P(N_i = 0) = e^{-\lambda}$. Con λ pequeño, la gran mayoría de los ciclos se comanda *exactamente* en x_{nom} . El twin dispone así de material limpio para calibrarse, y el régimen dominante del ensayo sigue siendo el elástico, que es el que se pretende representar.
2. **La consigna es discreta, no continua.** Los valores posibles son x_{nom} , $x_{\text{nom}} + \delta$, $x_{\text{nom}} + 2\delta$, etc. Como δ se dimensiona para que un solo empujón ya cruce el límite elástico, la secuencia queda esencialmente binaria: o ciclo normal, o excursión. Un disparo del detector se puede atribuir sin ambigüedad a un evento identificable.
3. **Frecuencia y profundidad son parámetros separados.** λ controla cada cuánto ocurre una excursión; δ controla cuán profunda es. Se ajustan de forma independiente, lo que permite diseñar el ensayo (“una excursión cada diez ciclos, penetrando unos pocos milímetros”) en vez de descubrir a posteriori qué salió.
4. **La frecuencia de excursión tiene forma cerrada.** Bajo el dimensionamiento anterior, la probabilidad de excursión por ciclo es

$$p_{\text{exc}} = P(N_i \geq 1) = 1 - e^{-\lambda}, \quad (5.3)$$

y el número esperado de ciclos hasta la primera excursión es $1/p_{\text{exc}}$. No hay que simular nada para elegir λ : se despeja del tiempo de espera que se quiera. Con el valor adoptado $\lambda = 0,1$ resulta $p_{\text{exc}} \approx 9,5\%$, es decir una excursión cada $\sim 10,5$ ciclos en promedio: suficientemente frecuente para observarla varias veces en una corrida de una hora, y suficientemente rara para que no domine el ensayo.

5. **Los casos profundos aparecen, pero raramente.** $N_i = 2$ o más es progresivamente menos probable, de modo que el ensayo produce de forma natural alguna excursión más profunda sin que haya que programarla. Es la cola larga que un régimen de sobresolicitud real exhibiría.
6. **Los ciclos son independientes entre sí.** El sorteo de un ciclo no depende de los anteriores. No hay estructura temporal que un detector pudiera aprender y usar para anticiparse, lo que refuerza que el disparo sea una detección genuina sobre la señal medida.

El adjetivo *compuesto* designa precisamente esta construcción en dos niveles: un conteo aleatorio de Poisson multiplicado por una magnitud fija. Es lo que permite el punto 3: que la frecuencia de las excursiones y su profundidad sean dos parámetros de diseño independientes.

5.2.5.3. Alcance del modelo estocástico

La formulación admite una lectura equivocada que debe descartarse expresamente. La distribución de Poisson **no se ajusta a datos** del resorte, no describe ninguna propiedad del material y no forma parte del modelo físico de la Etapa A, que sigue siendo estrictamente determinista. Es un generador de perturbación que corre del lado del banco, un instrumento experimental del mismo estatus que la velocidad de cicleo o la amplitud nominal, cuyo único rol es producir el evento de interés con una estadística conocida y bajo control del operador. El twin, del lado de la nube, nunca ve la secuencia $\{N_i\}$: sólo recibe telemetría, igual que si las excursiones vinieran del mundo real.

5.2.6. Marcador post-shakedown

En los primeros ciclos de la vida de un resorte fresco, además del permanent set descrito por (5.1), existe un transitorio de acomodación mecánica (asentamiento de las espiras, redistribución de tensiones internas). Ese transitorio contamina la medición de la tasa de degradación asintótica.

El lazo de re-zero aplica correcciones *sólo* después de haber detectado que este transitorio inicial ha terminado. Se calcula, por lo tanto, un **marcador post-shakedown** desde el ciclo 1, basado en un umbral sobre la variación ciclo a ciclo de x_0 (o, como respaldo, sobre un número mínimo de ciclos). Una vez que la condición se cumple, el marcador se *fija* y no se recalcula: a partir de ese ciclo las correcciones se activan.

5.2.7. Calibración individual desde el ciclo 1

En la Etapa C, los parámetros $\{x_0, k, x_e, H, w, \eta\}$ se recuperaban ejecutando un ensayo de rampa dedicado antes de cada ensayo de fatiga. Ese procedimiento no es aplicable en la

Etapla D: una rampa a la profundidad requerida induce por sí sola ~ 43 mm de *permanent set*, dejando al resorte pre-fatigado antes de comenzar el ensayo real.

En su lugar, cada resorte se calibra desde su propio *primer ciclo* del ensayo de fatiga, que es, por construcción, una rampa virgen del resorte fresco. El script `calibrate_from_cycle1.py` ajusta la curva $F-x$ del ciclo 1 en modo libre (todos los parámetros ajustados), mediante la misma rutina `segmented_fit.fit_sweep` que se usa en Stage 3 para los ciclos posteriores en modo restringido (x_e fijado, sólo $\{x_0, k\}$ ajustados). Se elimina así el ensayo de rampa separado sin perder información.

El lazo de guarda lleva esta idea un paso más allá: no usa siquiera el ciclo 1 como calibración *previa* almacenada, sino que construye su referencia en línea, dentro de la misma corrida que después vigila (Sección 5.5.2).

5.3. Arquitectura final del sistema

La arquitectura de la Etapa C se extiende añadiendo el canal de retorno desde la nube hacia el banco. La Figura 5.2 resume el sistema completo.

Los servicios previos (`ingestor`, `consumer`, `fit_worker`, `monitor_worker`, InfluxDB y Grafana) se conservan sin cambios funcionales. Las novedades estructurales son dos bloques de decisión, `twin_guard_worker.py` y `rezero_helper.py`, y los canales MQTT dedicados que los conectan con la Raspberry Pi.

Una diferencia arquitectónica entre ambos merece destacarse, porque determina su latencia y por lo tanto su ámbito de aplicación:

- `twin_guard_worker.py` es un *consumer group* propio sobre el mismo *stream* de Redis que alimenta a `consumer.py`, y evalúa **cada muestra individual** ($\sim 0,4$ mm de resolución espacial efectiva) apenas ésta atraviesa la cola. No consulta la base de datos: la latencia de decisión es la del *stream*, no la de una consulta a InfluxDB.
- `rezero_helper.py` se ejecuta periódicamente (cadencia de ~ 5 min), lee del histórico ya persistido en InfluxDB y emite a lo sumo una corrección por invocación. Su lazo es de **una muestra por ciclo** o más lento aún.

5.4. Canales de retorno: protocolo

Ambos canales de retorno se implementan como topics MQTT distintos del canal de datos, sobre el mismo *broker* Mosquitto con mTLS descrito en la Etapa C, y comparten los siguientes principios de diseño:

- **Un topic por experimento:** el `experiment_id` forma parte del topic, evitando cruces entre corridas concurrentes.

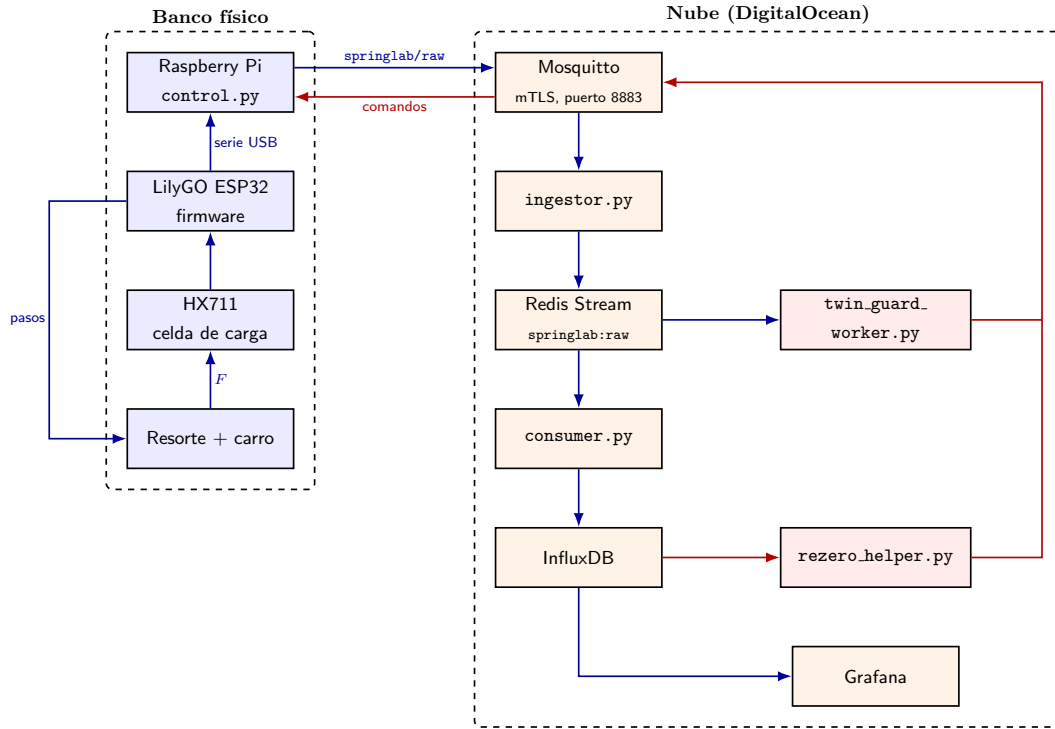


Figura 5.2: Arquitectura final del Digital Twin. En azul, el flujo de datos físico → virtual heredado de la Etapa C. En rojo, el nuevo flujo virtual → físico añadido en la Etapa D, en sus dos variantes: `twin_guard_worker.py` consume el *stream* de Redis muestra a muestra y publica el comando de guarda; `rezero_helper.py` lee del histórico en InfluxDB y publica la corrección de re-zero. Ambos usan topics MQTT dedicados y credenciales distintas de la del instrumento.

- **Credencial y ACL específicas:** cada servicio de la nube con capacidad de comando posee credencial propia, con permiso de escritura únicamente sobre su topic de comando y sin permiso de escritura sobre el canal de datos. Recíprocamente, la Raspberry Pi tiene permiso de lectura sobre los topics de comando y *no* puede publicar en ellos, de modo que el instrumento no puede fabricar sus propios comandos. El esquema completo de credenciales es el de la Tabla 4.2.
- **Cota superior de seguridad:** el firmware del LilyGO tiene un techo absoluto $D_{\text{máx}} = 200 \text{ mm}$; toda consigna que llevaría a superarlo es rechazada por el propio firmware.

5.4.1. Canal de guarda

El lazo de guarda publica en `springlab/command/{experiment_id}`, el mismo canal que ya usaba `monitor_worker.py` para el comando de detención (**STOP**), reutilizando su credencial (`springlab_monitor`) y su lista de permisos sin modificación alguna del *broker*. El comando de guarda (**GUARD_CLAMP**) lleva la nueva cota de extensión que el banco debe respetar desde ese instante.

Criterio de selección del canal. Existiendo ya dos canales de comando, la elección entre ambos no es indiferente: usar el de re-zero habría sido incorrecto. Ese canal está construido para una tarea de naturaleza opuesta: corregir una deriva lenta y acumulativa, y por eso incorpora salvaguardas que *limitan* deliberadamente lo que puede hacer: un paso máximo de 5 mm por corrección y un intervalo mínimo entre correcciones consecutivas (Sección 5.4.2). Un **GUARD_CLAMP** necesita justamente lo contrario: mover la consigna decenas de milímetros de una vez y sin esperar. Enviado por el canal de re-zero, sería rechazado por sus propias validaciones. El canal de comando, en cambio, fue diseñado para acciones inmediatas y de una sola vez (**STOP**), que es exactamente la categoría a la que **GUARD_CLAMP** pertenece.

Reutilizarlo tiene además una ventaja operativa: no requirió crear ningún topic nuevo, ninguna credencial nueva ni ninguna regla nueva en el *broker*. Toda la superficie de seguridad, es decir quién puede escribir qué, quedó tal como estaba, y con ella la propiedad que importa: el instrumento sigue sin poder publicar en el canal por el que recibe órdenes.

Confirmación por telemetría en lugar de acuse de recibo. La confirmación de que el lazo cerró **no** se toma de un *ack* enviado por la Raspberry Pi, sino de la telemetría real del propio banco: se considera cerrado cuando los datos de extensión que vuelven por `springlab/raw` muestran que la carrera efectivamente se revirtió. La razón es que un acuse de recibo sólo confirma que un mensaje llegó a un programa, no que un motor se haya movido; entre ambas cosas hay un puerto serie, un firmware y un mecanismo, cualquiera de los cuales puede fallar sin que el acuse deje de emitirse. Verificar contra el sensor y no contra el acuse es práctica establecida en este proyecto, adoptada después de varios episodios en que un mensaje de conexión o un código de retorno exitoso no implicaban que la acción física hubiera ocurrido. Como efecto secundario, tampoco fue necesario ampliar los permisos de la Pi para que pudiera escribir un topic de confirmación.

5.4.2. Canal de re-zero

El lazo de re-zero usa un canal propio, y sus salvaguardas responden a una diferencia esencial con el canal anterior: sus comandos son **acumulativos**. Un **GUARD_CLAMP** fija una cota absoluta, de modo que recibirlo dos veces produce el mismo resultado que recibirlo una; una corrección de re-zero, en cambio, desplaza la consigna en una cantidad, y aplicarla dos veces la desplaza el doble. Perder un mensaje o duplicarlo deja al ensayo permanentemente

descalibrado, sin ninguna señal evidente de que algo salió mal. De ahí cada una de las siguientes decisiones:

- **Topic propio, con confirmación:** `springlab/rezero/{experiment_id}` para el comando, no retenido, y `.../ack` para la confirmación, publicado por la Raspberry Pi tras aplicar o rechazar cada corrección. El comando no se retiene para que un mensaje viejo no se entregue a un cliente que se reconecte más tarde y termine aplicando una corrección caduca.
- **Credencial propia** (`springlab/rezero`), con permisos separados de los del instrumento y de los de `monitor_worker`. En particular no puede escribir su propio topic de confirmación: si pudiera, un servicio defectuoso podría reportar como aplicada una corrección que nunca llegó al banco.
- **Numeración secuencial:** la Pi rechaza todo mensaje cuyo número de secuencia sea menor o igual al de la última corrección aplicada. Es lo que vuelve inofensivo un reenvío: se puede reintentar sin miedo a aplicar el desplazamiento dos veces.
- **Marca temporal con vencimiento** (~ 4 min) e **intervalo mínimo** entre correcciones consecutivas. Lo primero descarta mensajes que quedaron demorados en la red y que, al llegar, describen un estado del resorte que ya no es el actual. Lo segundo evita que dos invocaciones muy seguidas del servicio corrijan ambas la misma deriva.
- **Paso máximo por corrección de 5 mm.** Acota el daño de un error: si un cálculo erróneo pidiera una corrección absurda, el banco sólo se movería 5 mm y el operador tendría tiempo de notarlo. Las correcciones legítimamente mayores no se descartan, se aplican *recortadas* a 5 mm por invocación y cierran la diferencia de forma gradual.
- **Reenvío ante falta de confirmación:** si el `/ack` no llega dentro de un plazo, la corrección se reenvía con el *mismo* número de secuencia. Combinado con la numeración, esto da la garantía que el lazo necesita: la corrección se aplica exactamente una vez, se haya perdido el comando o sólo su confirmación.

El canal se validó sobre el trayecto real $Pi \leftrightarrow$ nube comprobando, con la telemetría del propio banco, que una corrección válida movió efectivamente el carro y que una deliberadamente inválida, con un paso mayor que el máximo admitido, fue rechazada y no lo movió.

5.5. Lazo de guarda elástica

Este es el lazo que da su nombre a la etapa: el digital twin observa, decide y actúa sobre el componente físico para *protegerlo*. Su propósito es demostrar el flujo virtual $\rightarrow$ físico en

su forma más directa, sobre el trayecto real $Pi \leftrightarrow$ nube y sin apoyarse en ninguna calibración previa del resorte bajo ensayo.

El esquema completo consta de cuatro piezas: la excitación aleatoria del lado del banco, la autocalibración de la referencia elástica del lado de la nube, el criterio de disparo, y la acción de retorno.

5.5.1. Excitación: consigna con jitter de Poisson

El banco cicla el resorte a una extensión nominal x_{nom} holgadamente dentro de su zona elástica. En cada ciclo se sortea el número de empujones $N_i \sim \text{Poisson}(\lambda)$ y se comanda la extensión $D_i = x_{\text{nom}} + N_i \delta$ según (5.2). La mayoría de los ciclos, por lo tanto, es un ciclo elástico ordinario; ocasionalmente uno se estira lo suficiente como para alcanzar el codo elasto-plástico del resorte.

Los primeros ciclos y la calibración del twin. El twin construye su recta elástica de referencia sobre los primeros ciclos de la misma corrida que después vigila (Sección 5.5.2), de modo que esos ciclos deberían estar libres de excursión plástica: un empujón temprano contaminaría la referencia contra la cual se juzga todo el resto de la corrida. El diseño original preveía resolverlo del lado del banco, suprimiendo el sorteo durante un número configurable de *ciclos de calentamiento* en los que N_i se forzaría a cero.

Esa supresión **no llegó a implementarse**. El script de excitación sortea N_i desde el primer ciclo sin excepción, y la protección de la fase de calibración descansa enteramente en la estadística de la propia distribución: con $\lambda = 0,1$ la probabilidad de que un ciclo dado quede sin perturbar es $e^{-\lambda} \approx 90,5\%$, a lo que se suma que el ajuste de arranque descarta el tramo superior del recorrido acumulado (Sección 5.5.2). La consecuencia es que la limpieza de la fase de calibración es probable pero no está garantizada, y el riesgo concreto no es tanto un ajuste contaminado de la recta como un **pico nominal aprendido en exceso**: si uno de los ciclos de la fase de aprendizaje recibe un empujón, el twin registra como amplitud nominal un valor δ milímetros mayor que el real, con lo que la zona de detección queda desplazada hacia arriba y la guarda pierde sensibilidad. Es una limitación de la implementación, no del método, y se levanta reponiendo la supresión prevista.

Elección de los parámetros. x_{nom} , λ y δ son los tres grados de libertad del experimento y se fijan del lado del banco. Su elección requiere sólo una estimación *gruesa* de x_e , tomada de literatura o de la inspección de dónde el resorte empieza a ablandarse, y no una calibración: se sitúa x_{nom} algo por debajo de esa estimación, se elige δ para que un solo empujón la cruce por unos pocos milímetros, y λ para fijar la frecuencia de excursión según (5.3). Los valores usados fueron $\lambda = 0,1$ y $\delta = 20$ mm, con una velocidad de cicleo de 5 mm/s (retardo de paso $1000 \mu\text{s}$), igual a la de la campaña LCF de esta misma etapa. Conviene notar que la caracterización de la Etapa C se ejecutó a $1500 \mu\text{s}$ por paso (Sección 4.6), de modo que

la velocidad no es común a toda la memoria sino sólo a los ensayos de la Etapa D.

La Figura 5.3 ilustra el comportamiento resultante de la consigna a lo largo de una corrida.

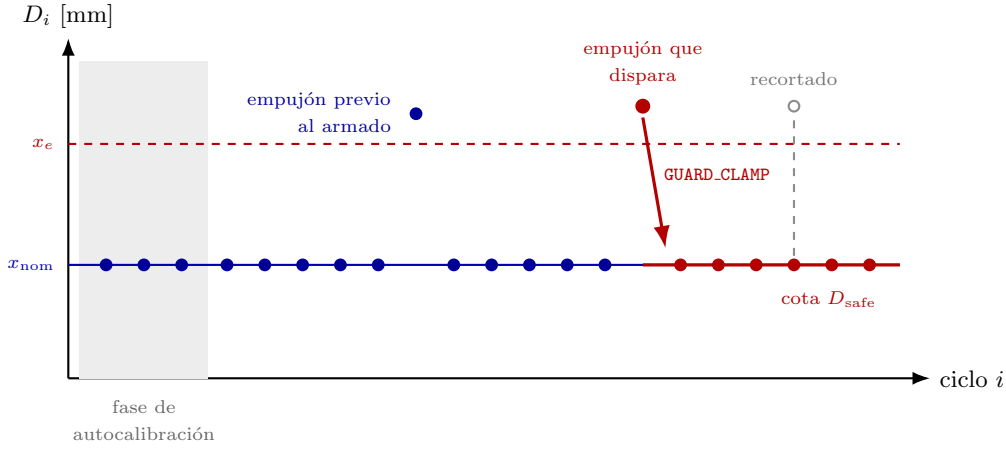


Figura 5.3: Consigna de extensión bajo jitter de Poisson (esquemático). Cada punto es la extensión comandada en un ciclo. La zona sombreada corresponde a los ciclos sobre los que el twin construye su referencia; el sorteo también opera en ellos, de modo que su limpieza depende de que λ sea pequeño (Sección 5.5.1). Los dos puntos situados por sobre x_e corresponden ambos a empujones de Poisson, y se distinguen por lo que el twin hace con ellos: el azul ocurre antes de que el detector termine de armarse, de modo que el resorte cruza a zona plástica sin que nadie reaccione; el rojo ocurre con el detector ya operativo y es el que provoca el disparo. La diferencia entre ambos no está en la excitación, que es idéntica, sino en el estado del vigilante. Como δ se dimensiona para que un solo empujón cruce el límite elástico, las consignas son esencialmente binarias: x_{nom} o por sobre x_e . Al detectar el ablandamiento, el twin emite **GUARD.CLAMP**; el sorteo de Poisson sigue ejecutándose, pero sus resultados quedan recortados a la cota D_{safe} .

5.5.2. Autocalibración de la referencia elástica

El servicio `twin_guard.worker.py` no recibe ningún parámetro del resorte: ni x_e , ni k , ni una calibración previa almacenada. Construye su propia referencia a partir de los primeros ciclos de la misma corrida que después vigila, en cuatro pasos.

(1) Acumulación. Sobre las carreras ascendentes de los primeros ciclos acumula las muestras con fuerza positiva, descartando así el *toe* de compresión descrito en la Sección 4.6.3, y no intenta ajustar nada hasta reunir al menos `MIN_BOOTSTRAP_SAMPLES` puntos que abarquen al menos `MIN_BOOTSTRAP_SPAN_MM` de recorrido. Si tras `BOOTSTRAP_MAX_CYCLES` ciclos no lo

ha conseguido, emite una advertencia y *permanece desarmado*: prefiere no vigilar antes que vigilar contra una referencia mala.

(2) Ajuste y congelamiento. Ajusta por mínimos cuadrados la recta de Hooke sobre la fracción inferior del recorrido acumulado (`BOOTSTRAP_ELASTIC_FRAC`), descartando el extremo superior como seguro contra cualquier ablandamiento incipiente que se hubiera colado:

$$\hat{F}(x) = \hat{k}_0 (x - \hat{x}_0). \quad (5.4)$$

Hecho esto, $\{\hat{k}_0, \hat{x}_0\}$ quedan **congelados**. No se reajustan nunca más: todo el resto de la corrida se juzga contra esta recta.

(3) Aprendizaje de la envolvente. Durante los siguientes `ENV_LEARN_CYCLES` ciclos, ya armado pero todavía sin permiso para disparar, el servicio mide cuánto se desvía realmente el resorte sano de su propia recta. Define el residuo

$$r(x) = \hat{F}(x) - F_{\text{med}}(x), \quad (5.5)$$

positivo cuando la fuerza medida cae *por debajo* de la recta, que es la firma del ablandamiento. La **envolvente elástica** ε es el máximo de r observado por ciclo en la banda elástica, sostenido como máximo móvil sobre una ventana de `ENV_WINDOW_CYCLES` ciclos. Al ser móvil, ε acompaña la deriva lenta de la línea base (térmica, de la celda de carga, de deformación permanente incipiente) sin que ésta se confunda con un evento.

(4) Aprendizaje del pico nominal. En esos mismos ciclos el servicio registra la extensión máxima alcanzada, x_{nom}^* , que toma como estimación de la amplitud nominal medida por él mismo a partir de la telemetría y no comunicada por el banco. Es la referencia respecto de la cual se define después la zona de detección. Este paso es el más sensible de los cuatro a la ausencia de ciclos de calentamiento garantizados (Sección 5.5.1): al tomarse como máximo sobre los ciclos de aprendizaje, basta con que uno de ellos reciba un empujón para que x_{nom}^* quede sobreestimado en δ , y con él toda la zona de detección.

Elección de la escala de ruido. El diseño original usaba la desviación estándar σ de los residuos del ajuste como escala de ruido. Es la elección obvia y resultó incorrecta: σ describe la dispersión *aleatoria* en torno a la recta, mientras que lo que la detección enfrenta en la práctica es una desviación *sistemática* (la curvatura residual del *toe* y la del entorno del pico) que σ subrepresenta. El resultado fue un disparo falso en la zona baja de la carrera (Sección 5.5.7). La envolvente, al medir el residuo máximo efectivamente observado y no su dispersión, captura ambas contribuciones.

Diferencia con `monitor_worker`. El congelamiento del paso (2) es lo que separa conceptualmente a este servicio del vigilante de la Etapa C (Sección 4.5.3). Aquél *reajusta* k en cada ciclo y sigue cómo esa sucesión de valores evoluciona a lo largo de cientos de ciclos, regularizando hacia el valor de literatura; éste ajusta k una sola vez y dispara sobre el residuo crudo de unas pocas muestras contra esa recta fija. Son detectores de fenómenos distintos, un deterioro lento frente a un desvío puntual, y por eso conviven sin solaparse.

5.5.3. Criterio de disparo

Una muestra se considera evaluable si cumple tres condiciones geométricas, todas necesarias:

1. **Pertenece a la carrera ascendente.** La carrera de retorno transcurre sistemáticamente por debajo de la de ida por histéresis (Sección 5.2.2), de modo que evaluarla dispararía siempre. Un solo punto descendente reinicia el contador de confirmaciones.
2. **Está fuera del *toe*:** $x > \hat{x}_0 + \text{TOE_GUARD_MM}$, por la razón establecida en Sección 4.6.3.
3. **Está por sobre la zona nominal de trabajo:** $x > x_{\text{nom}}^* + \text{DETECT_ABOVE_MM}$.

La tercera condición es la que hace el detector inmune a la deriva de línea base, y merece explicitarse: un ciclo sin empujón nunca produce muestras en esa región, de modo que ninguna deriva lenta acumulada en la operación nominal puede llegar a disparar. Sólo los ciclos que efectivamente recibieron un empujón de Poisson son candidatos.

Sobre las muestras evaluables el umbral es

$$\tau = m_\varepsilon \varepsilon, \quad m_\varepsilon = \text{ELASTIC_ENV_MULT}, \quad (5.6)$$

y se dispara cuando $r(x) > \tau$ en `TRIP_CONSEC_SAMPLES` muestras *consecutivas*. La exigencia de confirmaciones consecutivas descarta el ruido de una muestra aislada de la celda HX711 sin introducir un retardo apreciable: a la velocidad de cicleo del banco, tres muestras son aproximadamente un milímetro de recorrido.

El umbral (5.6) es *libre de escala*: no contiene ninguna constante de fuerza absoluta. Su única referencia es la dispersión que el propio resorte exhibió cuando estaba sano, de modo que el mismo código sirve para un resorte rígido y para uno blando sin retocar constantes. Una versión anterior añadía un segundo término proporcional a la fuerza ajustada, $\max(m_\varepsilon \varepsilon, f_r \hat{F}(x))$; ese término fue *eliminado* tras la primera corrida real, por la razón que se documenta en la Sección 5.5.7.

5.5.4. Acción de retorno: GUARD_CLAMP

Confirmado el disparo en una extensión x_{trip} , que es, por definición operativa, el inicio de zona plástica *de este ejemplar*, el servicio calcula la nueva cota admisible

$$D_{\text{safe}} = \text{máx}(x_{\text{nom}}^*, x_{\text{trip}} - \text{SAFE_MARGIN_MM}) \quad (5.7)$$

y publica GUARD_CLAMP con ese valor en el canal de guarda. El banco ejecuta entonces dos acciones:

1. **Reversión inmediata** de la carrera en curso: el carro invierte el sentido de marcha sin completar el estiramiento comandado.
2. **Acotamiento permanente** de toda consigna posterior, $D_i \leq D_{\text{safe}}$ para todo $i > i_{\text{trip}}$, aplicado como un mínimo sobre el sorteo de Poisson, que sigue ejecutándose, pero ya no puede llevar al resorte más allá de la cota.

Los dos términos de (5.7) responden a riesgos opuestos. El margen `SAFE_MARGIN_MM` reconoce que x_{trip} no es el primer punto de ablandamiento sino el punto en que el ablandamiento se volvió *evidente* por sobre el ruido: el inicio real está algo antes, y la cota debe quedar conservadoramente por debajo. El término x_{nom}^* , en cambio, impide que un disparo prematuro deje la cota por debajo de la amplitud nominal que el resorte ya demostró soportar sin daño durante la fase de aprendizaje, lo que detendría el experimento de facto. La cota resultante nunca es más restrictiva que el régimen que se sabe seguro.

Si tras la reversión el banco vuelve a superar la cota, por ejemplo si un comando se perdió, el servicio reemite GUARD_CLAMP, con una granularidad de `REPUBLISH_STEP_MM` para no inundar el canal. No hay acuse de recibo a nivel de aplicación: la confirmación es la reversión observada en la telemetría (Sección 5.4.1).

El cicleo *continúa* tras el evento: el objetivo no es detener el ensayo sino devolver el componente a operación elástica segura y mantenerlo allí. Esta es la diferencia funcional con el comando STOP de `monitor_worker.py`, que aborta la corrida.

5.5.5. Estructura del lazo y constantes de sintonía

La Figura 5.4 reúne las cuatro piezas anteriores en la estructura del lazo completo.

La Tabla 5.1 identifica cada elemento con su implementación, y la Tabla 5.2 reúne las constantes de sintonía citadas a lo largo de esta sección.

Naturaleza del lazo. No se trata de un regulador: no hay una variable que deba seguir una referencia con error nulo. Es un lazo *supervisor disparado por evento*, cuya acción es discreta, unidireccional (la cota nunca se relaja) e irreversible dentro de la corrida. Formalmente, su salida modifica el conjunto admisible de la consigna, no la consigna misma.

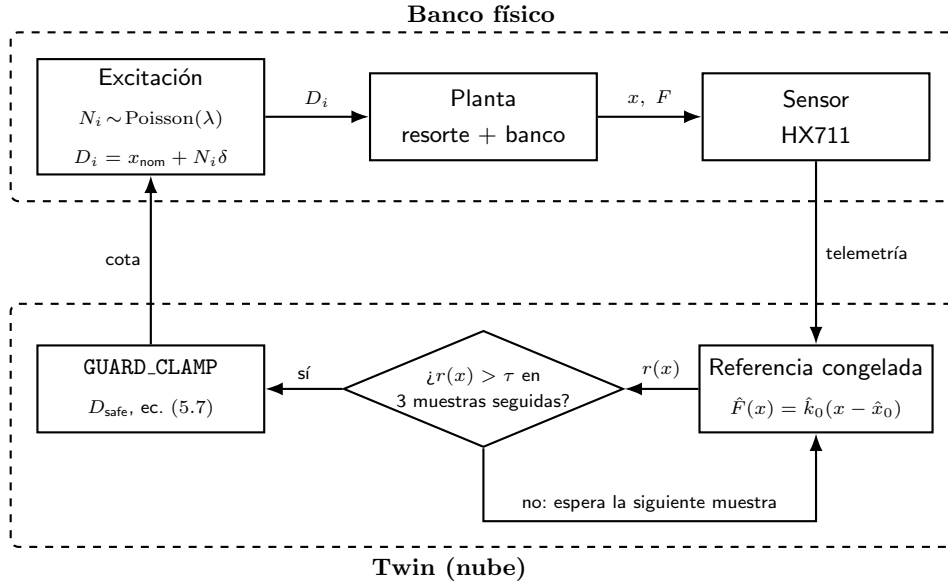


Figura 5.4: Estructura del lazo de guarda elástica. El rombo es el elemento de decisión: sólo cuando el residuo supera el umbral en tres muestras consecutivas se emite el comando. A diferencia de un lazo regulador clásico, no existe una consigna que deba seguirse con error nulo: es un lazo *disparado por evento*, cuya salida es una restricción permanente sobre el conjunto de consignas admisibles del banco.

Cuadro 5.1: Mapeo del lazo de guarda a los componentes del sistema.

Elemento	Símbolo	Implementación en el sistema
Excitación	λ, δ	Jitter de Poisson sobre la consigna, generado en la Raspberry Pi
Planta	—	Resorte, husillo TR8x8 y motor paso a paso
Sensor	—	Celda de carga HX711 (fuerza); conteo de pasos (extensión)
Transporte	—	MQTT sobre mTLS $\rightarrow$ <code>ingestor</code> $\rightarrow$ Redis Stream
Modelo de referencia	$\hat{F}(x)$	Recta de Hooke autoajustada y congelada, ec. (5.4)
Residuo	$r(x)$	$\hat{F}(x) - F_{\text{med}}(x)$, evaluado muestra a muestra
Umbral	τ	$m_\varepsilon \varepsilon$, ec. (5.6)
Lógica de disparo	—	3 muestras ascendentes consecutivas sobre umbral
Actuación	D_{safe}	<code>GUARD_CLAMP</code> vía <code>springlab/command/{exp_id}</code>
Confirmación	—	Reversión observada en la telemetría, no un acuse de recibo

Cuadro 5.2: Constantes de sintonía del lazo de guarda. Los valores fueron elegidos por criterio de diseño y validados contra corridas reales, pero no derivan de una caracterización del ruido de la celda de carga, que no llegó a medirse; en ese sentido son **puntos de partida** y no una sintonía final (Sección 5.5.9).

Constante	Valor	Significado
<i>Autocalibración</i>		
MIN_BOOTSTRAP_SAMPLES	20	Muestras mínimas antes de intentar el ajuste
MIN_BOOTSTRAP_SPAN_MM	20,0 mm	Recorrido mínimo que deben abarcar
BOOTSTRAP_ELASTIC_FRAC	0,85	Fracción inferior del recorrido que se ajusta
BOOTSTRAP_MAX_CYCLES	6	Ciclos tras los cuales advierte si no logró armarse
<i>Envolvente y zona de detección</i>		
ENV_LEARN_CYCLES	3	Ciclos de aprendizaje de ε y x_{nom}^*
ENV_WINDOW_CYCLES	4	Ventana móvil del máximo de residuo elástico
TOE_GUARD_MM	20,0 mm	Exclusión del <i>toe</i> de compresión sobre $\hat{x}_0$
DETECT_ABOVE_MM	2,0 mm	Margen sobre x_{nom}^* para habilitar disparo
<i>Disparo y acción</i>		
ELASTIC_ENV_MULT	1,8	Multiplicador m_ε del umbral
TRIP_CONSEC_SAMPLES	3	Confirmaciones consecutivas exigidas
SAFE_MARGIN_MM	15,0 mm	Margen bajo el disparo para fijar D_{safe}
REPUBLISH_STEP_MM	1,0 mm	Granularidad de reemisión del comando

5.5.6. Verificación de detección sin hardware

Antes de comprometer un resorte real, la lógica de detección se verificó por completo del lado de la nube mediante un generador sintético que publica directamente al *broker* el flujo de una carrera completa proveniente de un resorte simulado con la ley Hooke/Hollomon de la Etapa A y parámetros conocidos. Al disponer de *ground truth*, esta vía permite comprobar que el disparo ocurre donde debe respecto de un x_e conocido, sin Raspberry Pi ni banco involucrados, y se conserva como prueba de regresión independiente de la disponibilidad del hardware.

5.5.7. Ejecución sobre hardware

El lazo se ejecutó sobre el banco real con un resorte sacrificial del tipo 8x28.5mm, identificado como *guard8x285*. La Figura 5.5 muestra la fuerza registrada durante la corrida *GuardDemo3*, reconstruida a partir de la exportación de la serie almacenada en la base de datos.

Antes de describir la traza es necesario precisar qué representa. El panel promedia la

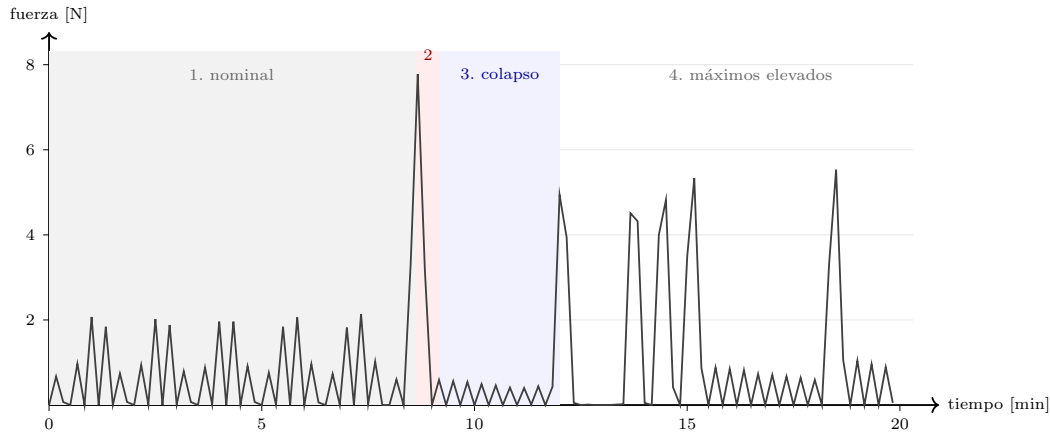


Figura 5.5: Serie de fuerza registrada durante la corrida `GuardDemo3` (`spring_id = guard8x285`), reconstruida a partir de la exportación de la base de datos. Cada punto es el promedio de una ventana de 10 s, duración comparable a la del ciclo del banco, de modo que los máximos de la traza no son los picos instantáneos de cada ciclo. Se distinguen las cuatro fases descritas en el texto: operación nominal, la excursión de Poisson que alcanza 7,8 N, el colapso de la respuesta por debajo del nivel nominal, y la reaparición posterior de máximos elevados que el registro disponible no permite explicar.

señal en ventanas de 10 s, mientras que el ciclo del banco dura del mismo orden, de modo que los máximos de la traza no son los picos instantáneos de cada ciclo sino promedios que dependen de cómo cae el muestreo respecto de la carrera. Los valores que siguen deben leerse, por lo tanto, como descriptores de la *estructura* de la corrida y no como mediciones de fuerza de pico.

Con esa salvedad, la traza distingue cuatro fases sucesivas:

1. **Operación nominal** (09:54 a 10:03, unos 9 min). Los máximos de la traza se mantienen estables entre 1,8 y 2,1 N, sin tendencia apreciable. Es la fase durante la cual el twin construye su recta de referencia y aprende su envolvente.
2. **Excursión** (10:03). Tres ventanas consecutivas registran 3,3, 7,8 y 3,2 N. El máximo de 7,8 N es el mayor de toda la corrida y supera en un factor $\sim 3,7$ el nivel nominal previo. Corresponde al empujón de Poisson llevando la consigna más allá del límite elástico del ejemplar.
3. **Colapso de la respuesta** (10:03 a 10:06, unos 3 min). Inmediatamente después los máximos caen a 0,4–0,6 N, esto es, muy por debajo del nivel nominal previo, y decaen de forma suave. El comportamiento es coherente con un resorte que adquirió deformación permanente durante la excursión y que, a igual extensión comandada,

desarrolla una fuerza sustancialmente menor.

4. **Reaparición de máximos elevados** (10:06 en adelante). A partir de las 10:06 la traza vuelve a registrar máximos de entre 3,3 y 5,5 N, intercalados con ventanas próximas a cero, y así continúa hasta el final del registro. Estos valores superan el nivel nominal de la primera fase y se mantienen durante los últimos ~ 8 min de la corrida.

Alcance de la evidencia obtenida. La lectura anterior sostiene dos afirmaciones y deja abierta una tercera. Confirma, primero, que la excitación estocástica opera como fue diseñada: un régimen dominante de amplitud constante interrumpido por una excursión aislada de amplitud muy superior. Confirma, segundo, que la corrida cambia de régimen de forma abrupta en ese punto, con una caída de la respuesta a menos de un tercio del nivel previo.

La cuarta fase, en cambio, *no* admite explicación con el registro disponible, y sería incorrecto omitirla: la reaparición de máximos por sobre el nivel nominal es incompatible con la lectura de un componente que quedó acotado de forma permanente por debajo de su régimen previo. Las hipótesis compatibles con el diseño del sistema son varias, entre ellas que la cota (5.7) haya quedado situada por sobre la amplitud nominal a causa de un pico nominal aprendido en exceso (Sección 5.5.1), o que la deformación permanente acumulada haya alterado la relación entre extensión comandada y fuerza desarrollada. Discriminar entre ellas exige el canal de extensión y la serie de diagnóstico por muestra (`model.params`, `stage = guard`) que el propio servicio escribe, ninguno de los cuales forma parte del registro conservado.

Por la misma razón tampoco es posible atribuir cuantitativamente el cambio de la fase 3 a la acción del comando `GUARD.CLAMP` frente a la deformación permanente adquirida durante la excursión. La verificación cuantitativa del cierre del lazo, que exige contrastar el instante del disparo publicado contra el instante de reversión en la telemetría de extensión, requiere esos mismos datos y queda pendiente. En el estado actual del registro, la corrida documenta que la excitación y el cambio de régimen ocurrieron, no que el lazo haya sido la causa del segundo.

5.5.8. Dos defectos de detección hallados sobre hardware

El resultado más informativo de la ejecución no fue el disparo en sí, sino dos modos de fallo del detector que sólo aparecieron con datos reales y que ninguna verificación sintética había anticipado. Ambos motivaron cambios de diseño ya incorporados en las ecuaciones de la Sección 5.5.3:

1. **Disparo falso en el *toe* de compresión.** Con el umbral basado en la σ del ajuste de arranque, el lazo disparó en la región baja de la carrera. Allí el resorte, que

arrastra deformación permanente, es empujado ligeramente a compresión al volver a casa (Sección 4.6.3) y exhibe un residuo estructural que nada tiene que ver con plastificación. *Corrección:* sustituir σ por la envolvente de residuo elástico aprendida del propio resorte, y añadir la guarda espacial `TOE_GUARD_MM`.

2. **Disparo tardío.** En `GuardDemo3` el disparo se confirmó unos ~ 15 mm más allá de donde debía. La causa fue el término fraccional $f_r \hat{F}(x)$ que el umbral llevaba entonces: a extensiones grandes la fuerza ajustada es grande, ese término domina y exige una penetración plástica apreciable antes de que el residuo lo supere. Es decir, el umbral se volvía *menos* sensible justamente donde debía ser más sensible. *Corrección:* eliminación del término, dejando el umbral en la forma de un solo componente de (5.6).

Ambas correcciones son *post hoc* sobre datos reales, y son la justificación empírica del valor adoptado para m_ε . El segundo fallo tiene además una lectura metodológica: un umbral que crece con la señal parece prudente sobre el papel, pero degrada la detección precisamente en el régimen crítico.

5.5.9. Alcance y limitaciones de la guarda

Esta demostración cierra el lazo virtual $\rightarrow$ físico de forma completa y verificable, pero **no constituye un sistema de seguridad de producción**, y presentarla como tal sería incorrecto:

- **La detección ocurre en la nube.** Entre la muestra que dispara y la reversión del carro median el transporte MQTT sobre el túnel, la cola de Redis y el trayecto de vuelta. Que esto sea tolerable depende enteramente de que el banco sea lento (varios segundos para recorrer los últimos milímetros antes de x_e): una lámina de deformación plástica en la carrera que dispara es *esperada*, no un fallo del diseño. Por eso el ensayo exige un resorte sacrificial.
- **La arquitectura correcta para protección real es local.** Un corte de protección genuino debe evaluarse donde no haya red de por medio: en el supervisor de la propia Raspberry Pi o, mejor aún, en el firmware del microcontrolador. La guarda de la Etapa D no debe generalizarse a un sistema de protección; su valor es demostrativo: prueba que el twin *remoto* puede decidir y actuar sobre el componente físico, no que deba ser el encargado de protegerlo.
- **Las constantes de sintonía son provisionales.** El piso de ruido de la celda HX711 nunca ha sido caracterizado empíricamente en este proyecto. Como consecuencia, m_ε y `TRIP_CONSEC_SAMPLES` (Tabla 5.2) son puntos de partida validados contra una corrida

real, no una sintonía final. Caracterizar ese ruido es el prerequisite natural de cualquier sintonía posterior.

- **La excitación es artificial.** El jitter de Poisson no modela ninguna solicitud real del componente en servicio; es un instrumento experimental para provocar el evento a demanda.
- **El cierre del lazo está establecido cualitativamente, no cuantitativamente.** Por lo indicado en Sección 5.5.7, falta contrastar el instante del comando publicado contra el instante de reversión en la telemetría de extensión.

5.6. Lazo de re-zero y campaña LCF

El segundo lazo de la etapa tiene un propósito instrumental: hacer posible una campaña experimental que sin lazo cerrado no es alcanzable en el marco de tiempo de esta memoria.

5.6.1. Estructura del lazo

La estructura clásica de un lazo de control involucra referencia, error, controlador, planta, perturbación y sensor. La Figura 5.6 presenta esa estructura mapeada al problema del *permanent set*: la perturbación es la acumulación de deformación permanente descrita por (5.1), que desplaza el sobreesfuerzo efectivo, y el controlador debe cancelarla exactamente.

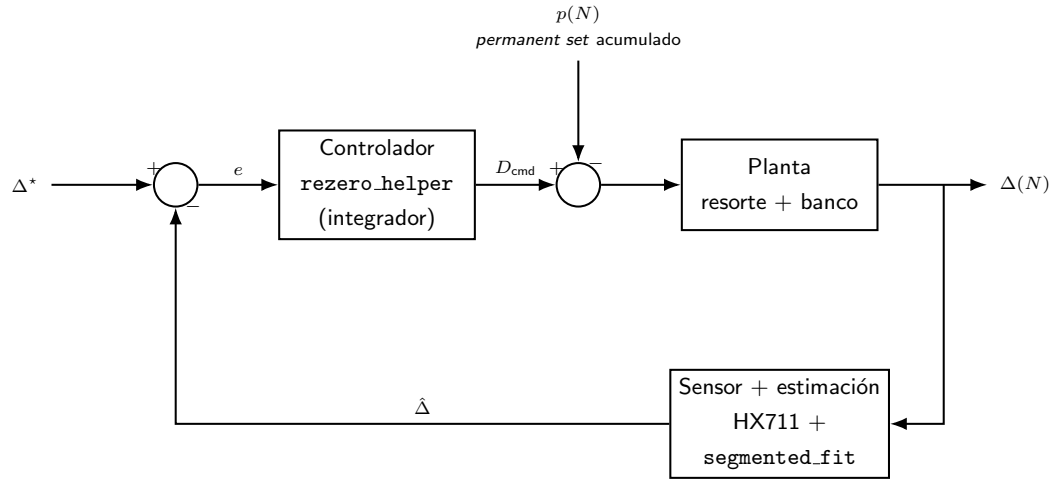


Figura 5.6: Estructura del lazo de re-zero. El controlador integra la deriva de *permanent set* en D_{cmd} para mantener el sobreesfuerzo Δ igual a la referencia Δ^* .

La Tabla 5.3 identifica cada bloque del lazo con su implementación real en el proyecto.

Cuadro 5.3: Mapeo del lazo de re-zero a los componentes del sistema.

Bloque	Símbolo	Implementación en el sistema
Referencia	Δ^*	Sobreesfuerzo objetivo del nivel (14,5, 23,9 o 36,3 mm)
Variable controlada	Δ	Sobreesfuerzo real $x_{\text{máx}} - x_e$ en el ciclo actual
Error	e	$\Delta^* - \hat{\Delta}$, evaluado ciclo a ciclo
Controlador	C	<code>rezero_helper.py</code> : acumula el error y emite una corrección ΔD , con paso máximo 5 mm y techo $D_{\text{máx}} = 200$ mm
Señal de control	D_{cmd}	Extensión objetivo enviada por <code>springlab/rezero</code>
Actuador	—	LilyGO ESP32, motor paso a paso y husillo TR8x8
Planta	P	Resorte y sistema mecánico: transforma D_{cmd} en extensión efectiva
Perturbación	$p(N)$	Acumulación de <i>permanent set</i> , ec. (5.1), que desplaza x_e hacia extensiones mayores
Sensor	H	HX711 y <code>segmented_fit</code> en modo restringido: del barrido F - x recupera x_0 y de ahí $\hat{\Delta}$
Habilitación	—	Marcador post-shakedown; el controlador queda saturado a $\Delta D = 0$ hasta que el marcador se fija

5.6.2. Formulación en variables de control

Los dos lazos de esta etapa son de naturaleza distinta, y conviene formular cada uno en los términos que le corresponden porque de esa diferencia depende que ambos coexistan sin ser redundantes. Esta sección desarrolla el lazo de re-zero como problema de control y cierra comparando su estructura con la del lazo de guarda.

Planta y perturbación. La variable controlada es el sobreesfuerzo plástico por ciclo, $\Delta(N)$. Sobre el ciclo N , la relación entre la consigna comandada $D_{\text{cmd}}(N)$ y el sobreesfuerzo efectivamente aplicado es

$$\Delta(N) = D_{\text{cmd}}(N) - x_{e,0} - p(N), \quad (5.8)$$

donde $x_{e,0}$ es el límite elástico del ejemplar medido antes del ensayo y $p(N) = \sum_{j \leq N} \Delta \ell_{\text{perm}}(j)$ es la deformación permanente acumulada hasta ese ciclo, según (5.1). La planta es, por lo tanto, una **ganancia unitaria con perturbación aditiva de entrada**: no tiene dinámica propia a la escala de un ciclo, y todo el problema de control reside en la perturbación.

Esa perturbación tiene dos propiedades que determinan la forma del controlador. La primera es que **entra con signo negativo**: cada milímetro que el resorte se alarga es un milímetro que se resta del sobreesfuerzo, lo que explica el signo del sumador de la Figura 5.6. La segunda es que $p(N)$ es **monótonamente creciente**, ya que la deformación plástica no se revierte. No se trata de un ruido de media nula sino de una deriva acumulativa que, además, se autoextingue: a medida que Δ decae, cada ciclo aporta menos deformación nueva, de modo que $p(N)$ tiende a un valor final (el *shakedown* de la Figura 5.1).

Ley de control. Una acción proporcional pura no sirve aquí. Al ser la perturbación acumulativa, corregirla una vez deja el error restablecido al ciclo siguiente, y con ganancia finita el sistema mantendría un error permanente proporcional a la tasa de deriva. Se adopta por eso una **acción integral discreta**, actualizada una vez por ciclo:

$$D_{\text{cmd}}(N+1) = D_{\text{cmd}}(N) + K_i e(N), \quad e(N) = \Delta^* - \hat{\Delta}(N), \quad (5.9)$$

con $K_i = 1$ y $\hat{\Delta}(N)$ la estimación del sobreesfuerzo obtenida por el observador descrito más abajo. Sustituyendo (5.8) en (5.9), y suponiendo estimación exacta, la dinámica del error en lazo cerrado resulta

$$e(N+1) = (1 - K_i) e(N) + [p(N+1) - p(N)]. \quad (5.10)$$

Con $K_i = 1$ el término homogéneo se anula y el lazo es **deadbeat**: el error de un ciclo se cancela por completo en el siguiente, y lo único que queda es el incremento de perturbación aparecido entre ambos. El error en régimen no es cero, sino igual al último incremento

de deformación permanente, cantidad que decrece a medida que el ensayo avanza hacia el *shakedown*. El lazo no elimina la deriva, la persigue con un ciclo de retardo, y ésa es exactamente la precisión que la campaña necesita: lo relevante es que Δ no se desplome a lo largo de cientos de ciclos, no que sea exacto en cada uno.

Compensación anticipativa descartada. El retardo de un ciclo de (5.10) es evitable en principio, y conviene explicar por qué no se evitó. La perturbación de este lazo no es desconocida: el propio modelo de la Etapa A la predice, ya que según (5.1) el incremento de deformación permanente del ciclo N vale $\Delta\ell_{\text{perm}}(N) = \beta\Delta(N)$. Con β conocido, un término anticipativo (*feedforward*) podría sumarse a (5.9) para cancelar el incremento *en el mismo ciclo* en que se produce, en lugar de corregirlo en el siguiente, y el error de régimen se anularía.

Esa opción se descartó porque exige confiar en el valor de β del ejemplar montado, y β es justamente uno de los parámetros que la Etapa C mostró peor determinados: en el ensayo de caracterización terminó en $\approx 0,29$ frente al 0,16 medido por rampa para el mismo tipo de resorte, porque absorbió el error de un x_e heredado (Sección 4.6.2). Un término anticipativo es de lazo abierto y no se autocorriges: un β equivocado no produce un error que el sistema detecte y compense, sino un sesgo sistemático sobre Δ , es decir sobre la misma variable que la campaña necesita mantener constante para poder atribuir la degradación a la amplitud. El experimento perdería precisamente la propiedad que el lazo existe para garantizar.

La realimentación pura evita ese riesgo porque no supone nada sobre el resorte: mide lo que efectivamente ocurrió y corrige esa cantidad. Paga por esa robustez un ciclo de retardo, que en este banco equivale a un error igual al último incremento de deformación permanente, pequeño y decreciente. Dado el objetivo del ensayo, el intercambio favorece claramente a la realimentación. La combinación natural para una implementación posterior es un término anticipativo con β estimado *en línea* a partir del historial del propio lazo, conservando la realimentación como corrección; así el anticipo dejaría de depender de un parámetro heredado, que es la objeción que aquí lo descarta.

Observador. El sobreesfuerzo no se mide de forma directa, porque no existe un sensor de deformación permanente. Se *estima* a partir de la propia curva $F-x$ del ciclo: el ajuste segmentado en modo restringido recupera la precarga $x_0(N)$, cuyo desplazamiento respecto del valor inicial es precisamente la deformación permanente acumulada, y de ahí se reconstruye $\hat{\Delta}(N)$. El bloque de realimentación es, en rigor, un observador basado en modelo y no un sensor, lo que hace que la calidad del lazo dependa de la calidad del ajuste. Es también la razón por la que el lazo no puede correr más rápido que un ciclo: antes de completar una carrera ascendente no existe curva que ajustar.

Saturación y habilitación. Dos elementos no lineales completan la descripción. El primero es una **saturación del actuador**: la corrección se limita a 5 mm por ciclo (Sección 5.4.2).

Durante el transitorio inicial, en que la deriva por ciclo es mayor que ese límite, el lazo opera saturado y se comporta como un limitador de velocidad; el error se cierra entonces de forma gradual en varios ciclos en lugar de en uno. La corrección se aplica *recortada* y no se descarta, lo que evita que el integrador acumule una acción que nunca llega a ejecutarse.

El segundo es un **integrador condicional**: la acción permanece retenida en cero hasta que el marcador post-shakedown (Sección 5.2.6) declara terminado el transitorio de acomodación mecánica. La razón es que durante esos primeros ciclos el desplazamiento de x_0 mezcla asentamiento de espiras con deformación plástica genuina, y corregir sobre esa mezcla introduciría en la consigna un error que después no se distingue del efecto que se quiere medir.

Comparación entre ambos lazos. La Tabla 5.4 resume por qué los dos lazos de la etapa no son la misma estructura aplicada a distinta escala. El de re-zero es un regulador en el sentido clásico: tiene referencia, error, acción integral y opera de forma continua sobre todo el ensayo. El de guarda no tiene referencia que seguir; es un **supervisor disparado por evento**, cuya salida no es una señal de control sino una restricción sobre el conjunto de consignas admisibles, y cuya acción es irreversible dentro de la corrida.

Cuadro 5.4: Los dos lazos de la Etapa D en términos de control.

	Lazo de re-zero	Lazo de guarda
Tipo	Regulador con acción integral	Supervisor disparado por evento
Variable controlada	Sobreesfuerzo Δ	Ninguna; acota el conjunto admisible
Referencia	Δ^* , constante	No tiene
Realimentación	Observador sobre la curva $F-x$	Residuo contra recta congelada
Actualización	Una vez por ciclo	Muestra a muestra, dentro de la carrera
Perturbación	Deriva acumulativa y monótona	Evento aislado e impredecible
Acción	Continua y reversible	Discreta, única e irreversible
Objetivo	Error acotado en régimen	Detectar a tiempo

5.6.3. Diseño experimental de la campaña LCF

Con este lazo disponible, se ejecuta una campaña de caracterización cuyo objetivo es recuperar la relación empírica entre sobreesfuerzo plástico Δ y tasa de degradación de rigidez $\left| \frac{dk}{dN} \right|$.

- **Tipo de resorte:** tres resortes idénticos, del tipo 8x28.5mm, del mismo set Yato YT-06875.

- **Niveles:** tres profundidades distintas de sobre esfuerzo, con un resorte por nivel: uno *profundo* (*anchor*) y dos *someros*. La elección de un resorte por nivel es un compromiso explícito entre alcance temporal de la memoria y cotas de dispersión; es una limitación reconocida al analizar los resultados.
- **Velocidad de cicleo:** 5 mm/s (retardo de paso 1000 μ s) en las tres corridas, para no confundir un efecto de amplitud con un efecto de velocidad.
- **Restricción de carrera:** el banco tiene un techo absoluto $D_{\text{máx}} = 200$ mm. Como el propio x_e inicial consume una fracción no despreciable de ese margen (por ejemplo, 84 mm en el resorte *deep*), la carrera disponible para acumular corrección de *permanent set* está intrínsecamente acotada.

5.6.3.1. Pivote basado en datos: el techo de carrera como restricción

El diseño original apuntaba a alcanzar en el nivel *profundo* una caída fraccional de rigidez del 33–50 %, análoga a la que define el fin de vida útil en aplicaciones industriales. Ese objetivo se *retiró* tras el análisis de la primera corrida:

- Los datos reales del resorte profundo mostraron que el techo de 200 mm, y no el tiempo ni el número de ciclos, limita la caída fraccional alcanzable a $\sim 2,5\text{--}3\%$ a esa profundidad.
- Profundizar más el sobre esfuerzo *no* ayuda: dado que el lazo cerrado mantiene Δ constante, la tasa de acumulación de *permanent set* $\beta \cdot \Delta$ es proporcional al sobre esfuerzo. Un sobre esfuerzo mayor consume la carrera disponible más rápidamente, no más lentamente.

Se redefine entonces el *deliverable* de la campaña: en lugar de una caída fraccional grande, se busca una **tasa de degradación bien resuelta** en un rango de sobre esfuerzos que el banco pueda sostener sin saturarse. Este redireccionamiento es en sí mismo una ilustración del valor del Digital Twin: la restricción de carrera fue identificada por el propio sistema, corriendo su modelo interno contra su propia telemetría real.

5.6.4. Ejecución y resultados

Se ejecutaron tres corridas, resumidas en la Tabla 5.5. Cada corrida se detiene cuando la tendencia lineal de $k(N)$ se aparta del ruido en forma estadísticamente clara y se ha acumulado un margen suficiente para determinar la pendiente.

5.6.4.1. Tasa de degradación por nivel

Sobre la ventana *limpia* (post-shakedown) de cada corrida se ajusta una regresión lineal de k contra el número de ciclos. Los resultados aparecen en la Tabla 5.6.

Cuadro 5.5: Corridas ejecutadas en la campaña de caracterización LCF.

Resorte	Sobreesfuerzo Δ^*	Ciclos	Estado
8x28.5mm_deep1	36,3 mm ($x_e = 83,7$ mm)	131	Detenida cerca del techo de carrera
8x28.5mm_shallowA1	14,5 mm	74	Completada
8x28.5mm_shallowB1	23,9 mm	126	Completada

Cuadro 5.6: Tasa de degradación por nivel, obtenida por regresión lineal sobre la ventana post-shakedown.

Nivel	Sobreesfuerzo Δ	dk/dN [N/mm/ciclo]	n	p -valor
shallowA1	14,5 mm	$-1,13 \times 10^{-5}$	46	$2,1 \times 10^{-18}$
shallowB1	23,9 mm	$-1,55 \times 10^{-5}$	32	$5,3 \times 10^{-9}$
deep1	36,3 mm	$-1,82 \times 10^{-5}$	101	$5,9 \times 10^{-91}$

Las tres pendientes son monótonas en el sobreesfuerzo y todas estadísticamente significativas.

5.6.4.2. Ley empírica tasa-vs-sobreesfuerzo

Un ajuste lineal de $|dk/dN|$ contra el sobreesfuerzo entrega la ley empírica

$$\left| \frac{dk}{dN} \right| \approx 7,21 \times 10^{-6} + 3,12 \times 10^{-7} \Delta \text{ [mm]} \quad (r^2 = 0,96). \quad (5.11)$$

El coeficiente de determinación es alto, pero la ley debe leerse como *preliminar y direccional*: con un solo resorte por nivel no es posible separar la dispersión resorte-a-resorte de la propia respuesta del material. Un ensayo con replicados (dos o tres resortes por nivel) es la validación pendiente natural, fuera del alcance de esta memoria.

5.6.4.3. Consecuencia para el modelo físico

La ley empírica (5.11) evidencia que la tasa de degradación no es una constante sino una función creciente y no trivial del sobreesfuerzo. El modelo de daño implementado en `physics.py` (Etapa A) es, en cambio, **independiente de la amplitud** en su formulación actual: un ciclo apenas por sobre x_e cuenta igual que un ciclo profundo. La Etapa D deja documentada esta discrepancia como insumo para una revisión posterior del modelo de daño, del tipo Coffin–Manson [22], que sí incorpora dependencia con la amplitud de deformación plástica.

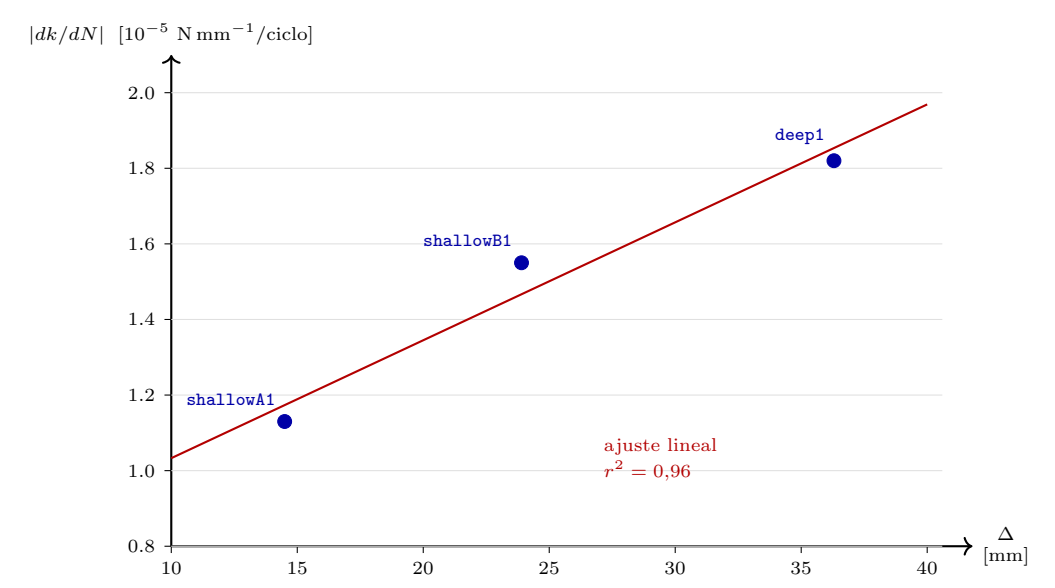


Figura 5.7: Tasa de degradación de rigidez en función del sobreesfuerzo plástico por ciclo. Cada punto corresponde a la pendiente ajustada sobre la ventana post-shakedown de una corrida completa (Tabla 5.6), y la recta es el ajuste de (5.11). La tendencia es monótona y cada pendiente es individualmente significativa, pero con un solo ejemplar por nivel los puntos no llevan barra de dispersión entre resortes: la incertidumbre representada por la separación entre punto y recta no incluye la variabilidad resorte a resorte, que es precisamente la que la campaña no pudo estimar.

5.7. Resumen de decisiones de diseño

Las secciones anteriores justifican cada decisión en el punto donde aparece. La Tabla 5.7 las reúne para poder revisarlas de conjunto, junto con la alternativa que se descartó en cada caso. Leída completa, muestra que las decisiones no son independientes: la mayoría se sigue de dos hechos experimentales establecidos en las etapas anteriores, a saber, que el límite elástico es propiedad de cada resorte individual (Sección 4.5.5) y que el banco es lento (un ciclo cada ~ 51 s).

Cuadro 5.7: Decisiones de diseño de la Etapa D, alternativa descartada y motivo.

Decisión	Alternativa descartada	Motivo
Excitación del ensayo		

continúa en la página siguiente

(Tabla 5.7, continuación)

Decisión	Alternativa descartada	Motivo
Perturbación aleatoria de la consigna	Rampa monótona; esperar la falla natural	Un evento agendado no requiere twin alguno: bastaría programar la reversión en el ciclo conocido. Sólo un cruce impredecible obliga a inferirlo de la señal medida (Sección 5.2.5.1)
Poisson compuesto	Ruido gaussiano; exponencial o uniforme	Es la única que deja la mayoría de los ciclos <i>exactamente</i> en el valor nominal, separa frecuencia de profundidad en dos parámetros y produce una consigna discreta en vez de un continuo (Sección 5.2.5.2)
Primeros ciclos sin perturbación	Perturbar desde el ciclo 1	El twin necesita datos limpios para construir su referencia; una excursión temprana la contaminaría
<i>Detección</i>		
Autocalibrar sobre el propio resorte	Sembrar x_e desde el catálogo o desde otro ejemplar	x_e varía $\sim 20\%$ entre ejemplares del mismo tipo: un valor heredado actuaría tarde en un resorte y pronto en otro
Congelar la recta tras el ajuste inicial	Reajustarla en cada ciclo	Un ajuste que se actualiza sigue al resorte mientras se degrada y deja de ver la desviación; ésa es la función del vigilante de la Etapa C, no de éste
Umbral sobre la envolvente aprendida	Umbral sobre la σ del ajuste	σ describe dispersión aleatoria, pero el detector enfrenta desviaciones sistemáticas; produjo un disparo falso real (Sección 5.5.8)
Umbral sin término proporcional a la fuerza	$\max(m_\varepsilon \varepsilon, f_r \hat{F}(x))$	El término fraccional crece con la extensión y vuelve al detector menos sensible justo donde debe serlo más; retrasó un disparo real ~ 15 mm
Evaluar sólo por sobre la amplitud nominal	Evaluar toda la carrera	Los ciclos sin perturbación nunca alcanzan esa zona, de modo que ninguna deriva lenta de la línea base puede disparar
Exigir 3 muestras consecutivas	Disparar con una sola muestra	Descarta ruido puntual de la celda de carga; a la velocidad del banco cuesta ~ 1 mm de recorrido
<i>Acción y transporte</i>		
Acotar la consigna y seguir ciclando	Detener el ensayo (STOP)	El objetivo es devolver el componente a zona segura, no terminar el experimento
Cota $\geq$ amplitud nominal	Cota siempre en $x_{\text{trip}} - 15$ mm	Impide que un disparo prematuro deje la cota bajo un régimen que el resorte ya demostró soportar, lo que detendría el ensayo de hecho

continúa en la página siguiente

(Tabla 5.7, continuación)

Decisión	Alternativa descartada	Motivo
Reutilizar <code>springlab/command</code>	Usar el canal de re-zero; crear uno nuevo	El de re-zero limita el paso a 5 mm y rechazaría un recorte grande; crear uno nuevo obligaría a ampliar permisos sin necesidad (Sección 5.4.1)
Confirmar por telemetría	Acuse de recibo de la Pi	Un acuse confirma que un mensaje llegó a un programa, no que el motor se haya movido
Consumir del <code>stream</code> de Redis	Consultar InfluxDB por ciclo	La detección debe ocurrir dentro de una misma carrera; una consulta a la base de datos llega tarde

5.8. Posición del sistema en la taxonomía del Digital Twin

Con la implementación y operación de ambos canales de retorno, el sistema satisface la condición que Kritzinger *et al.* [13] definen como característica del *Digital Twin*: un flujo automatizado *bidireccional* entre objeto físico y representación virtual, en el que la representación no sólo refleja el estado del objeto sino que lo modifica. La Tabla 5.8 presenta la posición final del sistema.

Cuadro 5.8: Correspondencia final entre etapas y niveles del Digital Twin.

Etapas	Descripción	Nivel DT	Estado
A	Modelo físico del resorte	Pre-Shadow	Completa
B	Arquitectura de datos en <i>streaming</i>	Digital Shadow*	Completa
C	Banco físico integrado	Digital Shadow	Completa
D	Lazo cerrado (guarda elástica + re-zero)	Digital Twin	Completa

El elemento que separa esta etapa de la anterior no es la existencia del canal de retorno, que la Etapa C ya había dejado desplegado (Sección 4.5.3), sino su ejercicio automático sobre una magnitud del ensayo. En la Etapa C el flujo virtual $\rightarrow$ físico existía como infraestructura y su única acción posible era abortar la corrida, una decisión que en la práctica tomó siempre el operador. Los lazos de esta etapa, en cambio, mantienen el ensayo en marcha modificando cómo se ejecuta: el de re-zero sostiene constante el sobreesfuerzo plástico y el de guarda redefine la amplitud admisible. Es esa capacidad de regular, y no la de interrumpir, la que la taxonomía identifica con la modificación del estado del objeto físico. La Etapa C aportó el topic, la credencial y la lógica de suscripción en la Raspberry Pi; la Etapa D puso una decisión automática al otro extremo, sin modificar la configuración del *broker*.

Entre los dos lazos contruidos, la clasificación descansa con más fuerza sobre el de guarda. El lazo de re-zero cierra el flujo virtual $\rightarrow$ físico en el sentido formal de la taxonomía, pero la magnitud sobre la que decide es esencialmente geométrica: mide cuánto se ha desplazado el origen del resorte y corrige la consigna en esa misma cantidad. Es un lazo de compensación, y podría implementarse sin ningún modelo del material. El lazo de guarda decide contrastando la telemetría real contra un **modelo** del componente, la ley de Hooke instanciada sobre los parámetros de ese resorte individual, y actúa cuando la realidad se aparta de lo que el modelo predice. Es el modelo, y no una medida geométrica, el que genera la acción, lo que corresponde a la noción de digital twin en su sentido más propio: una representación virtual que interviene sobre su contraparte física a partir de lo que ella misma predice.

5.9. Discusión

Dos lazos, dos escalas de tiempo. Los lazos implementados operan en regímenes temporales distintos y esa diferencia no es accidental. El lazo de re-zero actúa una vez por ciclo o más lento, contra una perturbación que sólo crece y cuya escala de tiempo es de decenas de ciclos; su objetivo no es precisión sub-milimétrica sino la constancia de una variable estadísticamente relevante en una escala de tiempo mucho más larga que la del ciclo individual. El lazo de guarda actúa dentro de una única carrera, contra un evento cuya escala de tiempo es de segundos; su tolerancia al retardo depende directamente de la lentitud del banco. Ambos son suficientes para su propósito, y ninguno sería suficiente para el del otro.

El costo real de decidir en la nube. La demostración expone algo que hasta la Etapa C era sólo una preocupación de diseño: un twin alojado remotamente puede efectivamente decidir y actuar sobre su contraparte física, pero paga ese cierre con una latencia que se traduce en daño real e irreversible sobre el componente en el evento mismo que detecta. La arquitectura correcta para *protección*, con detección local en el controlador embebido o en el supervisor, es conocida; lo que la Etapa D aporta no es una alternativa a ella, sino la evidencia concreta de qué se gana y qué se pierde al desplazar la decisión a la nube.

Rol del Digital Twin en el propio experimento. La identificación del techo de carrera como restricción real (Sección 5.6.3.1) se realizó *ejecutando el modelo del propio Twin contra la telemetría real* del primer resorte. Es un ejemplo concreto del uso que la taxonomía atribuye al nivel *Digital Twin*: el modelo no sólo describe, sino que interviene en el diseño de los experimentos siguientes.

Limitaciones reconocidas. Se reiteran para claridad: (i) un solo resorte por nivel impide separar la respuesta del material de la dispersión resorte-a-resorte; (ii) el régimen LCF es un atajo respecto al régimen HCF que interesa en la aplicación real, y la extrapolación entre ambos regímenes exige un modelo de sensibilidad a la amplitud que `physics.py` no incorpora todavía; (iii) la ley empírica (5.11) vale dentro del rango de sobreesfuerzos ensayados y no debe extrapolarse fuera de él sin verificación experimental adicional; (iv) las constantes de detección del lazo de guarda son provisionales mientras el piso de ruido de la celda HX711 no sea medido empíricamente; (v) el lazo de guarda es un demostrador sobre resorte sacrificial, no un sistema de protección apto para operación.

Cierre. Con la Etapa D queda cerrado el arco definido al comienzo de la memoria: partiendo de un modelo físico aislado (Etapa A), pasando por una arquitectura de datos en *streaming* (Etapa B), la integración del banco real (Etapa C) y la implementación del lazo virtual → físico (Etapa D), el sistema alcanza el nivel *Digital Twin* en el sentido estricto de la taxonomía adoptada.

CONCLUSIONES

6.1. Síntesis del trabajo realizado

ESTA MEMORIA partió de un problema que no es de modelado sino de datos. Los métodos de estimación de vida útil remanente basados en aprendizaje automático dependen de conjuntos de datos de degradación cuya calidad, y no cuya cantidad, determina el resultado; y los conjuntos disponibles son escasos, provienen en buena parte de simulación, y se distribuyen sin registro del estado físico del espécimen ni verificación de que la solicitud aplicada haya sido la declarada. El objetivo del trabajo fue construir un procedimiento capaz de generar esos datos con trazabilidad y etiquetado físico, y el Digital Twin fue el mecanismo adoptado para lograrlo.

El desarrollo se estructuró en cuatro etapas, cada una de las cuales agrega una capa verificable sobre la anterior. La Etapa A estableció el modelo físico del resorte a partir de leyes constitutivas documentadas, aislado del hardware y con una separación explícita entre parámetros medidos y parámetros heredados de literatura. La Etapa B construyó la arquitectura de adquisición y persistencia en *streaming*, validándola con el modelo físico como fuente sintética antes de exponerla a un instrumento real. La Etapa C reemplazó esa fuente por el banco de ensayos, comunicado con la nube sobre un enlace autenticado, y calibró el modelo contra los resortes efectivamente ensayados. La Etapa D implementó el flujo de retorno desde la representación virtual hacia el instrumento y ejecutó la campaña experimental que ese flujo hace posible.

6.2. Conclusiones

Sobre el resultado principal. El resultado central del trabajo es que la representación virtual del resorte detectó, por sí sola, que el componente estaba entrando en zona plástica,

y ordenó al banco revertir la carrera y acotar las consignas siguientes para devolverlo a su zona segura. La detección no se apoyó en ningún parámetro conocido de antemano: el servicio aprendió la recta elástica del propio ejemplar montado durante los primeros ciclos de la corrida, y a partir de ahí juzgó cada muestra contra esa referencia. Tampoco reaccionó a un evento previsto: la sobresolicitud fue provocada por un proceso estocástico cuyo instante ni el operador ni el detector podían anticipar. Que el cruce fuera impredecible es lo que hace significativa la detección, porque un evento agendado no habría requerido representación virtual alguna, bastando con programar la reversión en el ciclo conocido.

Sobre el nivel alcanzado por el sistema. Ese comportamiento es el que satisface la condición que la taxonomía de Kritzinger *et al.* establece para el nivel de Digital Twin: un flujo automatizado bidireccional entre el objeto físico y su representación virtual, en el que la representación no sólo refleja el estado del objeto sino que lo modifica. La Etapa C ya disponía de un canal desde la nube hacia el banco, pero su única acción posible era abortar el ensayo y su disparo automático nunca llegó a ejercitarse; lo que la Etapa D agrega no es el canal sino la decisión automática al otro extremo, sobre una magnitud que permanece bajo control mientras el ensayo continúa. Entre los dos lazos construidos, la clasificación descansa con más fuerza sobre el de guarda, porque es el único cuya acción se origina en un modelo del componente y no en una medida geométrica.

Sobre la variabilidad entre ejemplares. El resultado más consecuente de la campaña de calibración no es ninguno de los valores ajustados sino la distinción entre parámetros transferibles y no transferibles. La rigidez elástica y el coeficiente de endurecimiento coincidieron entre ejemplares del mismo tipo dentro del $\pm 0,4\%$ y del $\pm 0,8\%$ respectivamente, mientras que el límite elástico difirió en cerca del 20 % entre esos mismos ejemplares. Como las tres magnitudes se estiman del mismo barrido y con el mismo código, la interpretación natural es que dónde empieza a ceder el material es una propiedad de cada resorte individual. Con dos ejemplares por lado el argumento no es concluyente, y se adopta como hipótesis de trabajo; basta, sin embargo, para lo que se le pide, porque los dos usos que se le dan son conservadores. La consecuencia práctica es doble. Metodológicamente, fijar una misma amplitud en milímetros para dos ejemplares no los somete a la misma solicitud, de modo que la variable que estandariza el ensayo es el sobreesfuerzo por sobre el límite elástico propio de cada resorte. En términos de diseño, ningún mecanismo automático de protección puede operar sobre un límite elástico heredado de otro ejemplar, lo que obliga a que se autocalibre sobre el espécimen montado.

Sobre el control como condición de validez del dato. El lazo de re-zero se construyó por una razón instrumental y terminó siendo el argumento más claro del trabajo respecto de su propia motivación. A consigna constante, la deformación permanente que el ensayo induce desplaza el punto de operación ciclo a ciclo, de manera que el sobreesfuerzo efectivo

decae hasta anularse y el ensayo transita, sin aviso, hacia un régimen distinto del que se pretendía aplicar. Un conjunto de datos etiquetado con la consigna nominal describiría, en ese escenario, una condición experimental que dejó de existir a los pocos ciclos. El lazo cerrado no mejora la precisión del dato: es lo que hace que la etiqueta siga siendo verdadera mientras el ensayo transcurre.

Sobre la generación de eventos anómalos etiquetados. La excitación estocástica de Poisson se introdujo para que el evento contra el cual el lazo protege ocurriera en un instante que ni el operador ni el detector pudieran anticipar. La razón es que un evento agendado no requiere un twin: si el instante del cruce se conoce de antemano basta con programar la reversión, y la representación virtual sólo resulta necesaria cuando el momento es desconocido y hay que inferirlo de la señal medida. Que además el cruce espontáneo esté a una escala de 10^5 a 10^7 ciclos es lo que obliga a inducirlo en lugar de esperarlo, pero es una consideración subordinada a la anterior. Su elección frente a alternativas continuas respondió a que concentra masa de probabilidad en cero, con lo que la mayoría de los ciclos permanece exactamente en la amplitud nominal y el detector dispone de material limpio para autocalibrarse, y a que separa la frecuencia de las excursiones de su profundidad en dos parámetros independientes. El efecto colateral resulta relevante para el propósito de la memoria: como la secuencia de perturbaciones la sorteas el propio banco, cada ciclo anómalo del conjunto de datos queda identificado por construcción, condición de etiquetado que los datos recogidos en operación real no ofrecen.

Sobre el resultado experimental. La campaña de fatiga de bajo ciclo recuperó de forma empírica la relación entre el sobreesfuerzo plástico por ciclo y la tasa de degradación de rigidez sobre tres niveles de sollicitación, con las tres pendientes monótonas en el sobreesfuerzo y estadísticamente significativas en cada nivel. El resultado reemplaza, para el rango ensayado, la dependencia heredada de literatura que el modelo incorporaba inicialmente, la cual resultó ser pesimista: el valor de vida característica adoptado subestimaba de forma apreciable la vida real del componente en el régimen de amplitud ensayado. El ajuste debe leerse como preliminar y direccional, porque un solo ejemplar por nivel no permite separar la respuesta del material de la dispersión entre resortes.

Sobre dónde conviene alojar la decisión. El lazo de guarda se implementó con la decisión corriendo en la nube, y esa elección tiene un costo que conviene dejar registrado, sin confundirlo con una limitación del resultado anterior. Entre la muestra que dispara y la reversión del carro median el transporte MQTT, el túnel y la cola de mensajes, de modo que el resorte alcanza a deformarse algo más mientras la orden viaja. Aquí resulta tolerable porque el banco es lento, y por eso el ensayo se hace sobre un resorte sacrificial.

Nada de esto obliga a que un twin resida en la nube. Para una función de protección genuina la arquitectura correcta es local, evaluando donde no haya red de por medio, en

el supervisor de la Raspberry Pi o en el firmware del microcontrolador. Lo que el trabajo demuestra es que el cierre del lazo se sostiene *incluso* en la configuración más desfavorable, con el twin separado de su contraparte física por una red pública; la latencia es el precio de esa demostración y no una propiedad del método.

Sobre las limitaciones del alcance. Las campañas se ejecutaron en régimen de fatiga de bajo ciclo, que es un atajo respecto del régimen de alto ciclo correspondiente a la operación real del componente; la extrapolación entre ambos exige un modelo de sensibilidad a la amplitud que el modelo físico no incorpora todavía. El entrenamiento de modelos de aprendizaje automático sobre los datos generados quedó fuera del alcance de la memoria, que construye el procedimiento de generación y produce el conjunto correspondiente a las campañas ejecutadas. Las constantes de sintonía del detector son valores validados contra ensayos reales y no una sintonía final, mientras el piso de ruido de la celda de carga no sea caracterizado empíricamente.

6.3. Trabajo futuro

Caracterización del ruido de instrumentación. Medir el piso de ruido de la celda de carga en condiciones de banco es el prerrequisito natural de cualquier sintonía posterior del detector. Sin esa medición, el multiplicador de la envolvente y el número de muestras consecutivas exigidas para confirmar un disparo permanecen como puntos de partida razonables pero no justificados cuantitativamente.

Verificación cuantitativa del cierre del lazo. El cierre del lazo de guarda quedó establecido de forma cualitativa. La verificación cuantitativa requiere exportar, junto a la serie de fuerza, el canal de extensión y la serie de diagnóstico por muestra que el propio servicio ya escribe, para contrastar el instante del comando publicado contra el instante de reversión observado en la telemetría.

Replicación de la campaña de degradación. La ley empírica obtenida se apoya en un ejemplar por nivel de sollicitación. Replicar cada nivel con varios ejemplares permitiría separar la respuesta del material de la dispersión entre resortes y convertir el resultado direccional actual en una caracterización con intervalos de confianza sobre los coeficientes.

Segunda generación del banco y acceso al régimen de alto ciclo. Durante el desarrollo se diseñó, sin llegar a construirlo, un banco accionado por biela y disco excéntrico en lugar de husillo. El cambio de mecanismo persigue elevar la frecuencia de cicleo lo suficiente para alcanzar el régimen de fatiga de alto ciclo, inaccesible al banco actual por su velocidad de aproximadamente un ciclo cada 51 segundos. El diseño incluye el dimensionamiento del mecanismo con verificación de torque contra las especificaciones del motor disponible, y quedó completamente especificado a la espera de construcción.

Canal de emisión acústica. Asociado al mismo diseño, se especificó la incorporación de un canal de emisión acústica con dos sensores en configuración de coincidencia, destinado a detectar eventos de fractura que la señal de fuerza no resuelve. La configuración de dos sensores responde a un problema de acoplamiento de ruido identificado en la revisión del diseño original de un solo sensor. Este canal agregaría al conjunto de datos una modalidad independiente de la mecánica, lo que resulta de interés directo para los métodos de detección multimodal.

Entrenamiento y evaluación de modelos de estimación de RUL. El paso natural de continuación es emplear los conjuntos de datos generados para entrenar y comparar modelos de estimación de vida remanente, aprovechando que cada tramo dispone de etiquetas de estado físico y de identificación de los ciclos anómalos. Esa evaluación permitiría además contrastar de forma empírica la premisa que motivó el trabajo: en qué medida el etiquetado físico y la trazabilidad mejoran el desempeño respecto de un conjunto de datos equivalente sin ellos.

BIBLIOGRAFÍA

- [1] X.-S. Si, W. Wang, C.-H. Hu, and D.-H. Zhou, “Remaining useful life estimation: A review on the statistical data driven approaches,” *European Journal of Operational Research*, vol. 213, no. 1, pp. 1–14, 2011.
- [2] Y. Lei, N. Li, L. Guo, N. Li, T. Yan, and J. Lin, “Machinery health prognostics: A systematic review from data acquisition to RUL prediction,” *Mechanical Systems and Signal Processing*, vol. 104, pp. 799–834, 2018.
- [3] N. Sambasivan, S. Kapania, H. Highfill, D. Akrong, P. Paritosh, and L. M. Aroyo, ““everyone wants to do the model work, not the data work”: Data cascades in high-stakes AI,” in *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*, 2021, pp. 1–15.
- [4] C. G. Northcutt, A. Athalye, and J. Mueller, “Pervasive label errors in test sets destabilize machine learning benchmarks,” in *Proceedings of the 35th Conference on Neural Information Processing Systems, Datasets and Benchmarks Track*, 2021.
- [5] V. Gudivada, A. Apon, and J. Ding, “Data quality considerations for big data and machine learning: Going beyond data cleaning and transformations,” *International Journal on Advances in Software*, vol. 10, no. 1, pp. 1–20, 2017.
- [6] T. Gebru, J. Morgenstern, B. Vecchione, J. W. Vaughan, H. Wallach, H. Daumé III, and K. Crawford, “Datasheets for datasets,” *Communications of the ACM*, vol. 64, no. 12, pp. 86–92, 2021.
- [7] A. Saxena, K. Goebel, D. Simon, and N. Eklund, “Damage propagation modeling for aircraft engine run-to-failure simulation,” in *International Conference on Prognostics and Health Management (PHM)*, 2008, pp. 1–9.
- [8] P. Nectoux, R. Gouriveau, K. Medjaher, E. Ramasso, B. Chebel-Morello, N. Zerhouni, and C. Varnier, “PRONOSTIA: An experimental platform for bearings accelerated degradation

- tests,” in *IEEE International Conference on Prognostics and Health Management (PHM)*, 2012, pp. 1–8.
- [9] A. K. S. Jardine, D. Lin, and D. Banjevic, “A review on machinery diagnostics and prognostics implementing condition-based maintenance,” *Mechanical Systems and Signal Processing*, vol. 20, no. 7, pp. 1483–1510, 2006.
- [10] M. Grieves, “Digital twin: Manufacturing excellence through virtual factory replication,” White Paper, 2014.
- [11] F. Tao and M. Zhang, “Digital twin shop-floor: A new shop-floor paradigm towards smart manufacturing,” *IEEE Access*, vol. 5, pp. 20 418–20 427, 2017.
- [12] —, “Digital twin shop-floor: A new shop-floor paradigm towards smart manufacturing,” *IEEE Access*, vol. 5, pp. 20 418–20 427, 2017.
- [13] W. Kritzinger, M. Karner, G. Trautner, J. Henjes, and W. Sihn, “Digital twin in manufacturing: A categorical literature review and classification,” *IFAC-PapersOnLine*, vol. 51, no. 11, pp. 1016–1022, 2018.
- [14] A. Rasheed, O. San, and T. Kvamsdal, “Digital twin: Values, challenges and enablers from a modeling perspective,” *IEEE Access*, vol. 8, pp. 21 980–22 012, 2020.
- [15] A. Fuller, Z. Fan, C. Day, and C. Barlow, “Digital twin: Enabling technologies, challenges and open research,” *IEEE Access*, vol. 8, pp. 108 952–108 971, 2020.
- [16] A. M. Wahl, *Mechanical Springs*, 2nd ed. New York: McGraw-Hill, 1963.
- [17] J. E. Shigley, C. R. Mischke, and R. G. Budynas, *Shigley’s Mechanical Engineering Design*, 11th ed. New York: McGraw-Hill, 2020.
- [18] J. H. Hollomon, “Tensile deformation,” *Transactions of the Metallurgical Society of AIME*, vol. 162, pp. 268–290, 1945.
- [19] W. Ramberg and W. R. Osgood, “Description of stress-strain curves by three parameters,” *NACA Technical Note*, no. 902, 1943.
- [20] B. J. Lazan, *Damping of Materials and Members in Structural Mechanics*. Oxford: Pergamon Press, 1968.
- [21] O. H. Basquin, “The exponential law of endurance tests,” *Proceedings of the American Society for Testing and Materials*, vol. 10, pp. 625–630, 1910.
- [22] L. F. Coffin, “A study of the effects of cyclic thermal stresses on a ductile metal,” *Transactions of the ASME*, vol. 76, pp. 931–950, 1954.

- [23] S. Suresh, *Fatigue of Materials*, 2nd ed. Cambridge: Cambridge University Press, 1998.
- [24] G. E. Dieter, *Mechanical Metallurgy*, 3rd ed. New York: McGraw-Hill, 1986.
- [25] ASTM International, “ASTM E606/E606M-21: Standard test method for strain-controlled fatigue testing,” West Conshohocken, PA, 2021.
- [26] OASIS, “MQTT version 5.0. OASIS standard,” <https://docs.oasis-open.org/mqtt/mqtt/v5.0/mqtt-v5.0.html>, March 2019.
- [27] Eclipse Foundation, “Eclipse mosquitto: An open source MQTT broker,” <https://mosquitto.org/>, 2024.