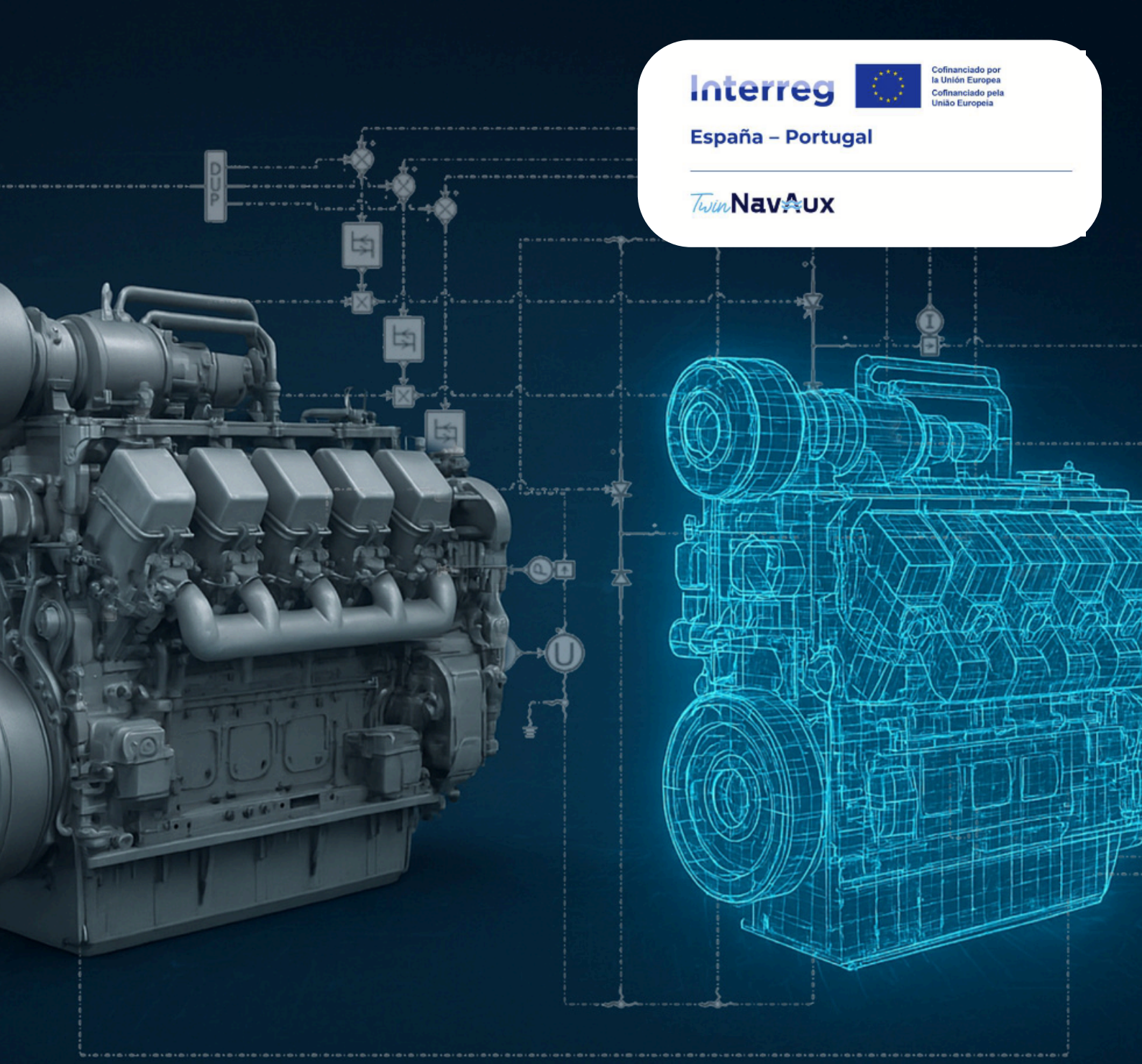


Interreg

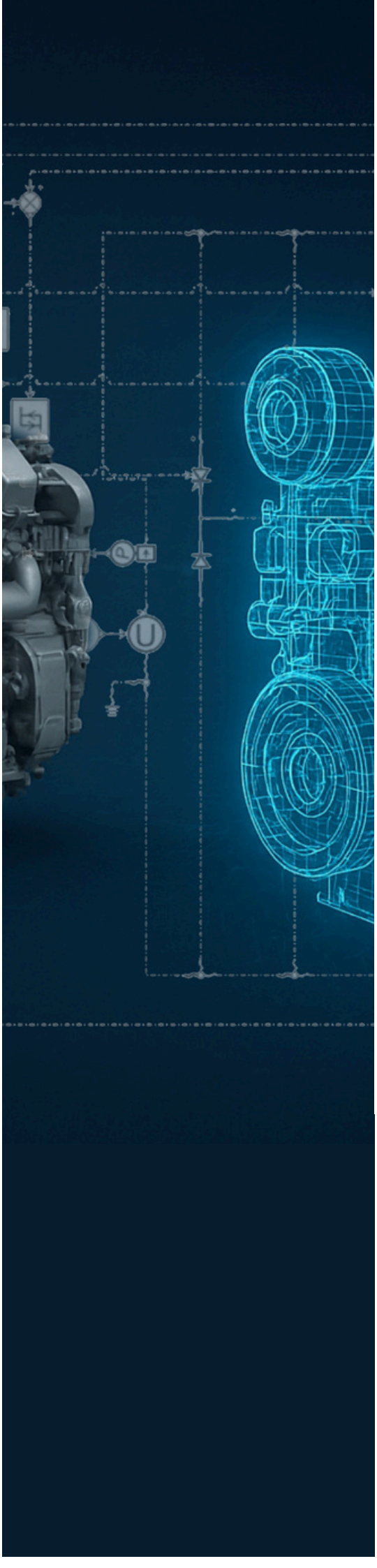
Cofinanciado por la Unión Europea
Cofinanciado pela União Europeia

España – Portugal

TwinNavAux

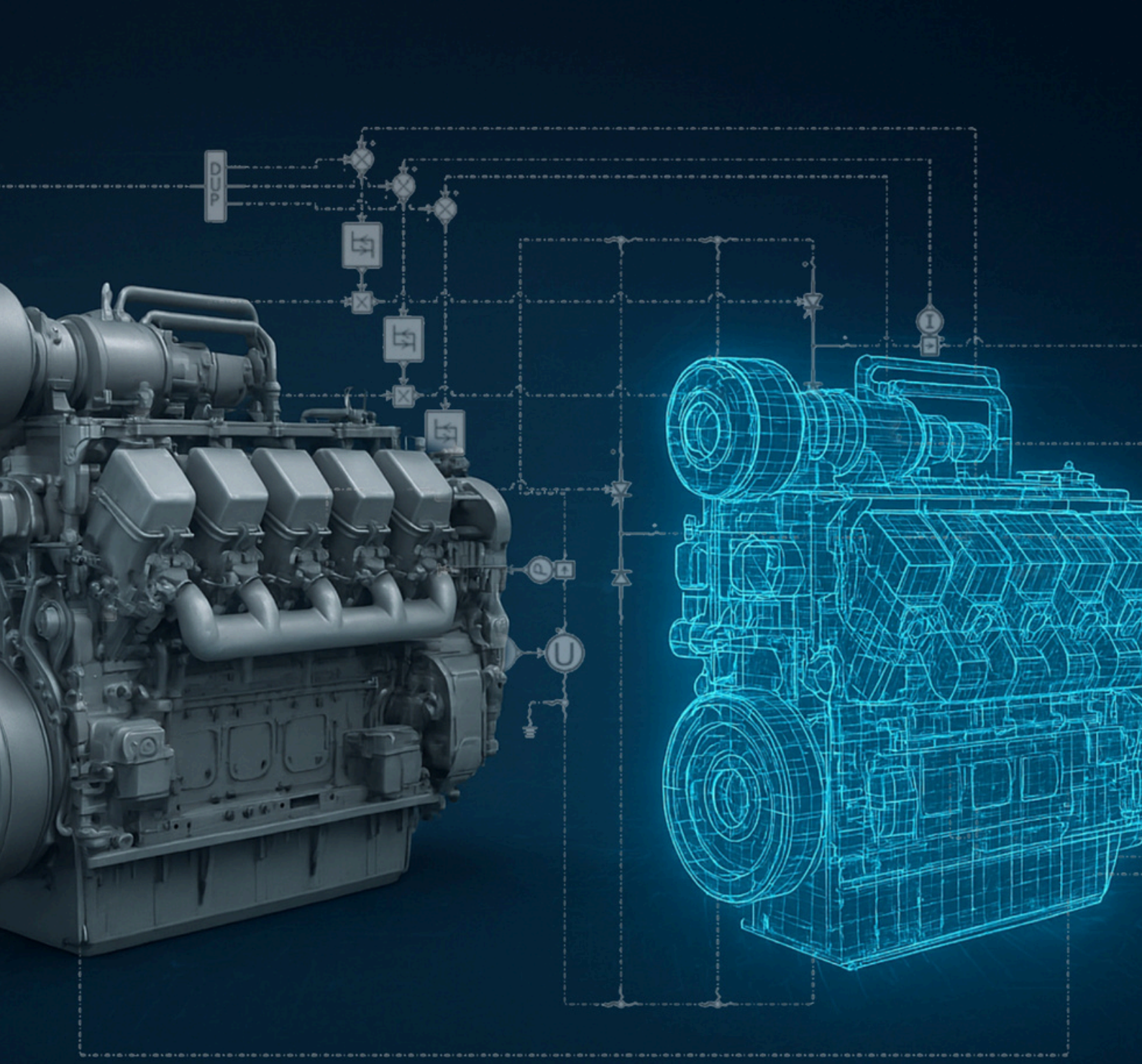
GUÍA de IMPLANTACIÓN de un GEMELO DIGITAL

Diciembre de 2025



1 Fases Básicas para la Implementación de un Gemelo Digital

2 Guia Detalhado de Implementação de um Piloto de Manutenção Preditiva



Fases básicas para la implementación de un gemelo digital

1

Esta guía tiene como objetivo establecer las fases básicas para la implementación de un gemelo digital para cualquiera de sus múltiples funcionalidades. Esto implica que dependiendo del uso que se le vaya a dar al gemelo (monitorización, mantenimiento, entrenamiento, etc.), se deberá adaptar cada una de las partes aquí descritas.

En la Figura 1 se muestra un flujo de datos sencillo que permite gestionar tanto los datos procedentes de los sensores, como los generados por las simulaciones. Gestionar correctamente el almacenamiento, tratamiento y visualización de estos datos es fundamental.

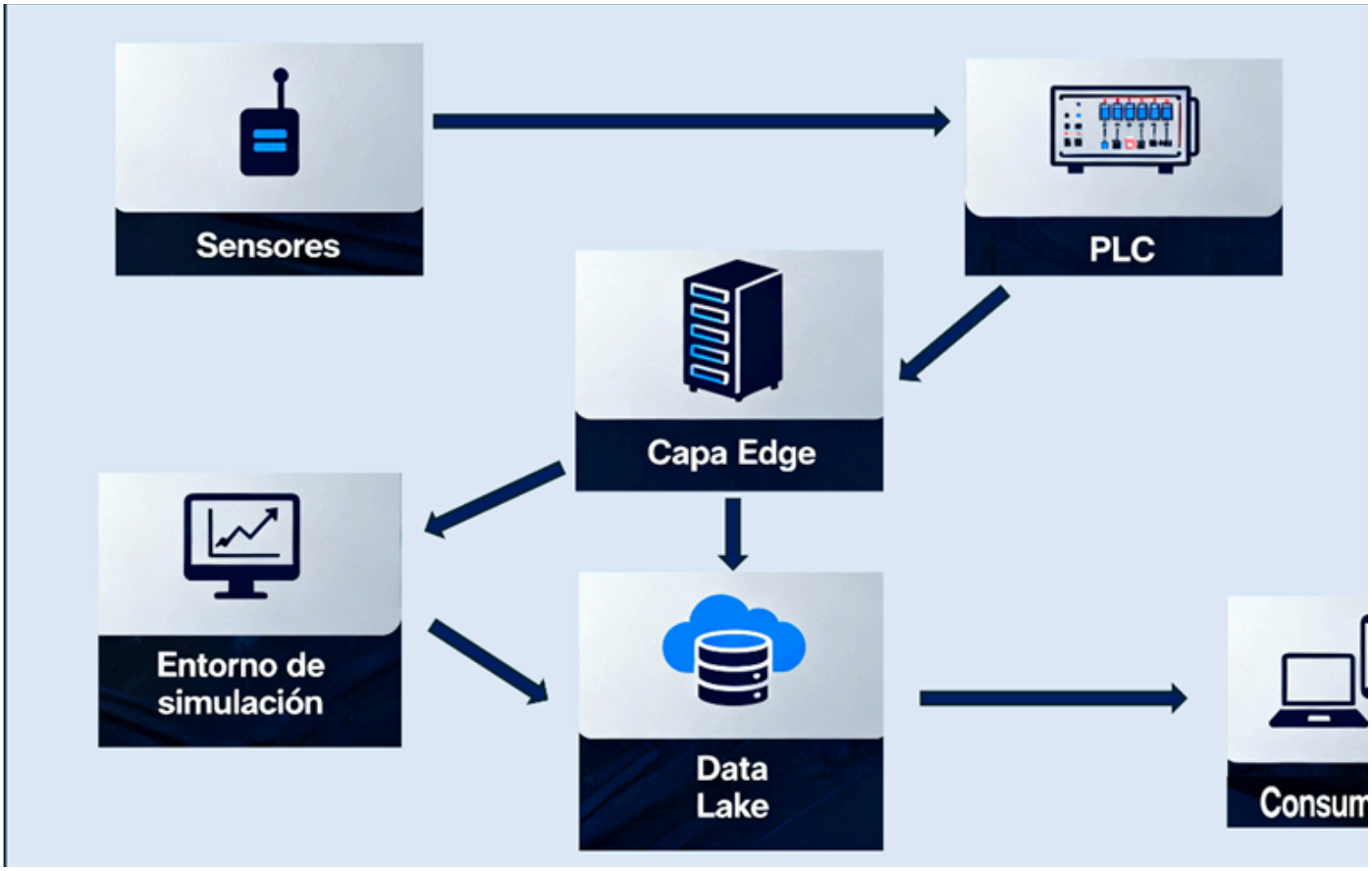


Figura 1. Flujo de datos básico

1. Sensórica – IoT

Una vez seleccionados los sensores necesarios en el equipo/sistema físico debemos definir cómo se procesan los datasets que se obtienen de esos sensores. Normalmente se registra estos datasets como timeseries, es decir, el dato que envía el sensor se asocia a una escala temporal. Los sensores, en general, envían las señales en voltios, lo que implica que necesitan una calibración. A continuación, se muestra un extracto de código en Python para la adquisición y calibración de los valores procedentes de un sensor (el ejemplo se corresponde a una célula de carga).

```
async def read_loop(self, stop_event):
    try:
      while not stop_event.is_set():
        line = self.ser.readline().decode('utf-8', errors='ignore').strip()
        if line and "Load:" in line:
          match = re.search(r'Load:\s*(-?\d+(?:\.\d+)?)', line)
          if match:
            try:
              self.signal_value = float(match.group(1))
              self.load_value = -0.0014 * self.signal_value - 267.33#calibración
              print(f"Signal: {self.signal_value:.3f}")
              print(f"Load: {self.load_value:.3f}")
            except ValueError:
              pass
        await asyncio.sleep(0.1)
```

Este código se puede ejecutar en el propio PLC o dispositivo conectado a los sensores. Un PLC tipo Siemens Logo! permite una automatización simple, que interactúa con sensores (temperatura, movimiento, luz, etc.) mediante sus entradas analógicas/digitales, procesa los datos con bloques de función y controla actuadores, permitiendo monitoreo remoto y comunicación vía Ethernet o GPRS para obtener datos en tiempo real.

La capacidad de computación que ofrecen hoy en día los PLC's permite realizar un primer pre-procesado de los datos procedentes de los sensores, es decir, un primer filtrado para eliminar el posible ruido asociado a los datos obtenidos en bruto desde los sensores.
Un ejemplo sencillo de una primera "limpieza" de los datos es un filtro paso-bajo que suaviza el ruido de alta frecuencia mientras mantiene la tendencia general de la señal del sensor.

```
import numpy as np
def moving_average (signal, window_size=5):
"""Return signal smoothed with a simple moving average."""
window = np.ones (window_size) / window_size
  # mode='valid' shortens the output; use 'same' to keep same length
filtered = np.convolve (signal, window, mode='same')
return filtered
# Example: sensor_data is a 1D NumPy array with your readings
# sensor_data = np.array([...])
filtered_data = moving_average(sensor_data, window_size=10)
```

Se debe elegir el "window_size" según el nivel de suavizado que se necesite. Las "ventanas" más grandes eliminan más ruido, pero también ralentizan o difuminan los cambios rápidos. La librería Scipy de Python permite realizar filtrados más avanzados.

2. Capa Edge

Un dispositivo Edge es un elemento hardware que se ubica en el límite de una red y conecta equipos o usuarios locales con otras redes o la nube, a menudo procesando datos localmente antes de enviarlos. Estos dispositivos actúan como puntos finales o puertas de enlace donde los datos entran o salen de una red, ubicados cerca de fuentes de datos como máquinas, sensores o usuarios. Pueden realizar tareas informáticas como agregar, filtrar y analizar datos cerca de la fuente para reducir la latencia y el uso del ancho de banda.

Habitualmente en este tipo de despliegues se usan como puertas de enlace de IoT industrial que vinculan PLC y sensores a plataformas en la nube y pueden ejecutar análisis básico localmente.

Su uso principal es para procesamiento local de datos: preprocesamiento, agregación y análisis en tiempo real para permitir respuestas rápidas y reducir el envío de datos a la nube. La conectividad y la posibilidad de usar diferentes protocolos es uno de sus puntos fuertes. Se pueden conectar buses de campo, protocolos industriales o redes locales de IoT con redes IP y API en la nube.

Además, incorporan seguridad y control de acceso a través de la aplicación de firewalls, filtrado de tráfico, autenticación de dispositivos y mantenimiento de datos confidenciales en las instalaciones cuando sea necesario.

El envío de datos desde un dispositivo Edge se puede realizar a través de diferentes protocolos de comunicación, como se ha comentado. Uno de los más usado es MQTT basado en TCP/IP. Es necesario instalar un bróker tipo Mosquitto. Mosquitto es un agente de mensajes de código abierto, ligero y escalable que implementa el protocolo MQTT proporcionando un método simple para enviar mensajes utilizando un modelo de publicación/suscripción. A continuación, se muestra un código clásico de publicación/subscripción de datos de sensores.

Ejemplo de código cliente:

```
import paho.mqtt.client as mqtt
def publish_message(MQTT_BROKER, MQTT_PORT, MQTT_TOPIC, data):
  # Define callbacks first
  def on_connect(client, userdata, flags, reason_code, properties):
    print("Connected with result code", reason_code)

  def on_message(client, userdata, msg):
    print(msg.topic, msg.payload.decode())
```

```
# Create client (Paho V2 API)
 client = mqtt.Client(mqtt.CallbackAPIVersion.VERSION2, "pump_system")
 # Assign callbacks
 client.on_connect = on_connect
 client.on_message = on_message
 # Connect to the broker
 client.connect(MQTT_BROKER, MQTT_PORT)
 # Publish once
 client.publish(MQTT_TOPIC, data)
 print(f"Published: {data} to topic {MQTT_TOPIC}")

 # Optional: process events briefly, then disconnect
 client.loop(timeout=2)
 client.disconnect()
```

Ejemplo código suscripción:

```
def on_connect(client, userdata, flags, reason_code, properties):
 print(f"Conectado al broker, código: {reason_code}")
 client.subscribe(topic)
 print(f"Escuchando en el topic '{topic}'...")
# Crear el cliente y establecer callbacks
cliente = mqtt.Client(mqtt.CallbackAPIVersion.VERSION2,"timeseries_subscriber")
cliente.on_connect = on_connect
cliente.on_message = on_message
# Conectar al broker
cliente.connect(broker, puerto)
```

Para esto es necesario definir el bróker, el puerto, el topic y el cliente.

```
# Configuración del broker y topic
broker = " " # IP del broker
puerto = 1883 # Puerto por defecto
topic = "data"
client = "timeseries_subscriber"
```

Es importante que el topic sea único y esté bien identificado, puesto que puede haber varios topics asociados a un mismo cliente.

Es necesario abrir un puerto para realizar esta comunicación. El más sencillo es el 1883, pero se puede incrementar la seguridad empleando el puerto 8883 e incorporando el uso de usuario/contraseña o certificados.

3. Entornos de simulación

Los entornos de modelado acausal basado en ecuaciones de sistemas complejos multidominio son plataformas que permiten a los ingenieros examinar la dinámica de sistemas, verificar estrategias de control y predecir el rendimiento en múltiples dominios físicos sin necesidad de un prototipado exhaustivo. Ofrecen bibliotecas para una amplia gama de aplicaciones, como la termodinámica, la mecánica de fluidos y la ingeniería eléctrica. Estas herramientas pueden ser de código abierto como OpenModelica o con licencia comercial como Simcenter Amesim. En el primer caso, al ser de código abierto, es ideal para la investigación académica, el desarrollo de algoritmos y la validación de nuevas metodologías de modelado. El segundo caso, emplea un enfoque de modelado gráfico que permite a los usuarios construir modelos a nivel de sistema utilizando componentes validados. La plataforma se integra con herramientas MATLAB/Simulink y CAD/CAE, lo que permite el análisis a nivel de sistema dentro de marcos más amplios de gemelos digitales.

En los softwares de simulación que trabajan con librerías no es necesario desarrollar el código con las ecuaciones que gobiernan los elementos de un sistema, sino tan solo definir los parámetros de esas ecuaciones (Figura 3), que se corresponden con los parámetros de funcionamiento del elemento. Cada componente de las librerías viene definido con unas variables de entrada y de salida (Figura 2), que van a definir con qué otros elementos de otras librerías van a ser compatibles o no. Los valores de esas variables van a permitir determinar si el componente se está simulando correctamente o no.

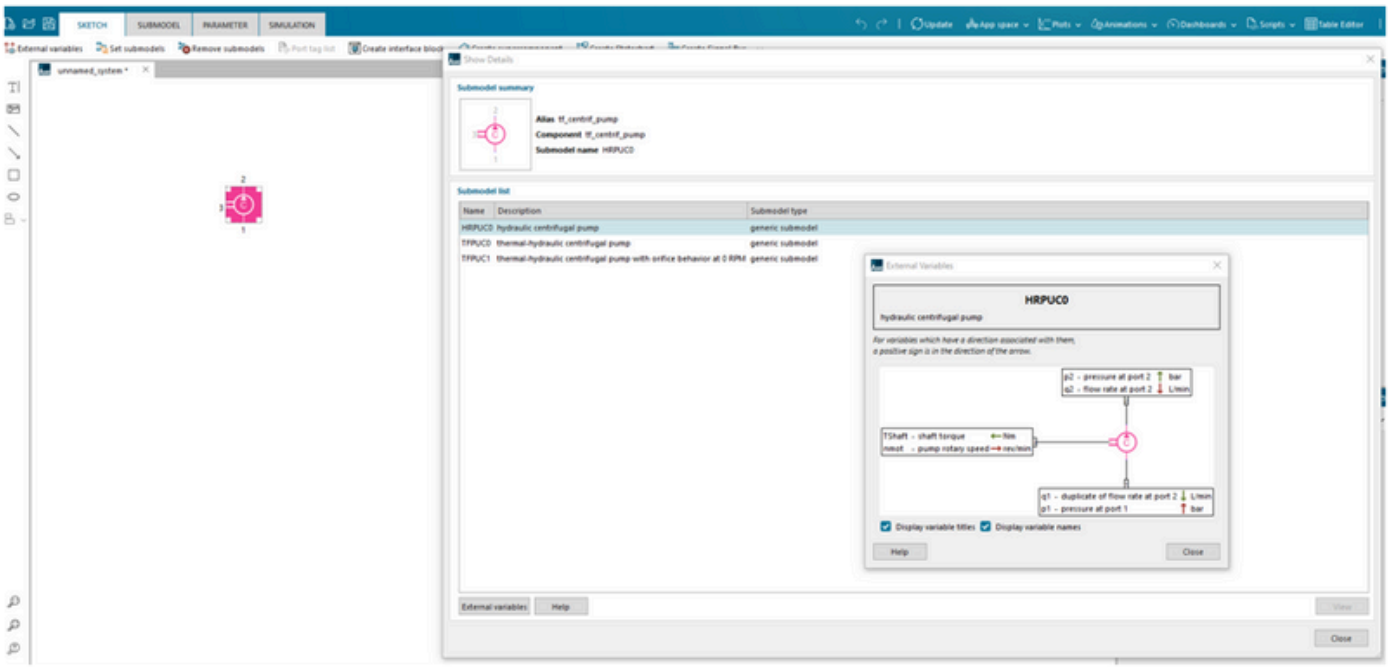


Figura 2. Variables de entrada y salida de una bomba de la librería de Hidráulica

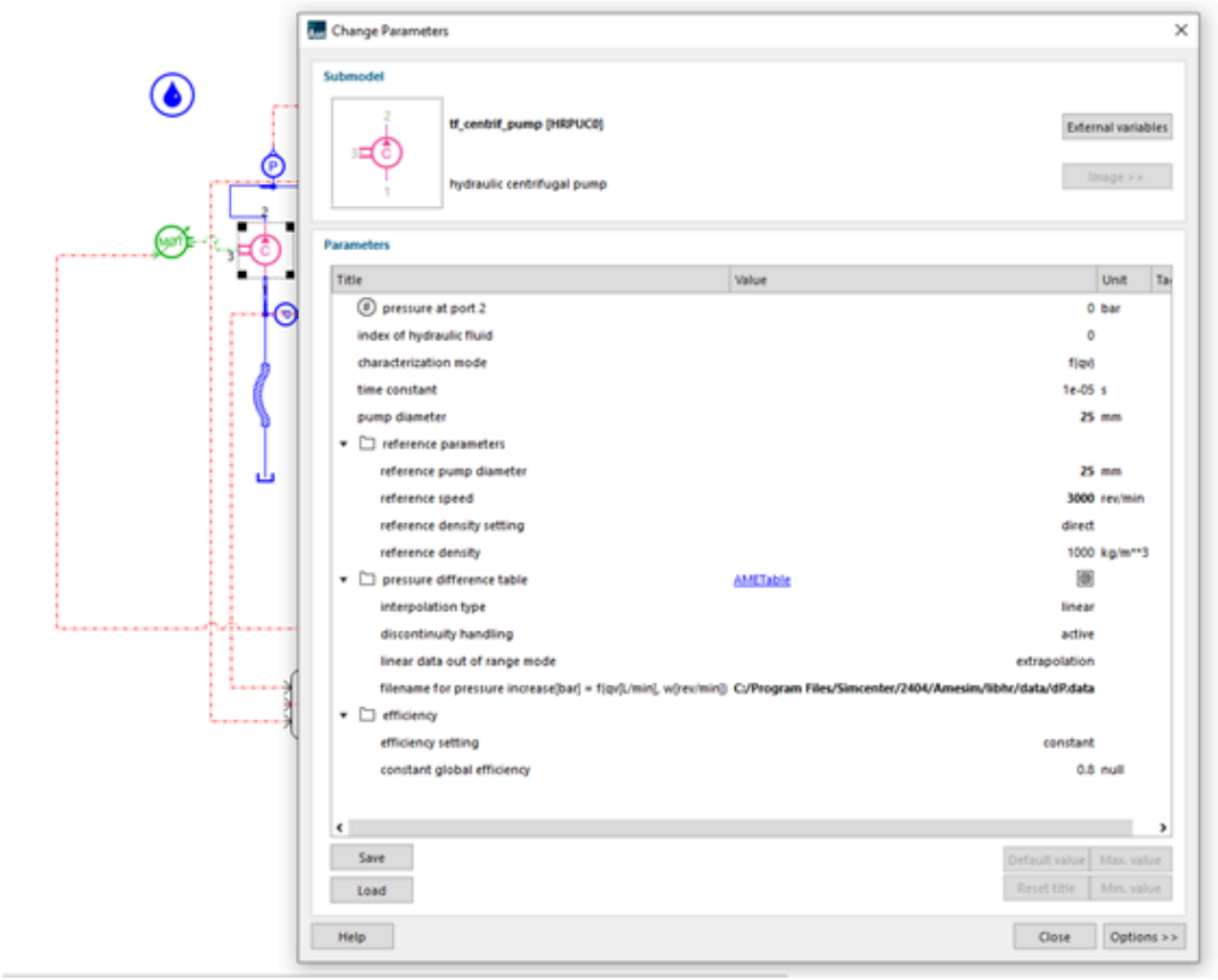


Figura 3. Parámetros a definir en una bomba de la librería de Hidráulica.

Habitualmente en este tipo de software se facilita el intercambio de modelos y la cosimulación mediante interfaces estandarizadas como la Interfaz de Maqueta Funcional (el estándar FMI), lo que permite la interoperabilidad con otros entornos de simulación (Figura 4).

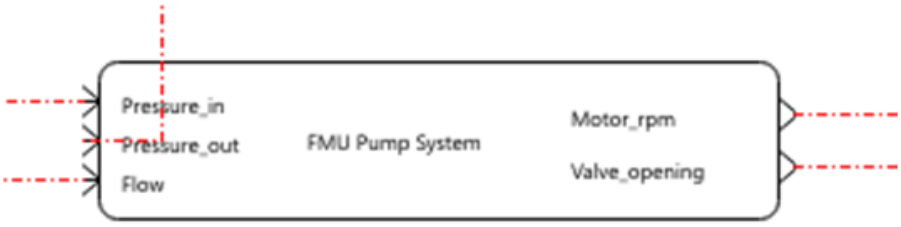


Figura 4. Módulo FMU en el software de simulación

El FMU es la unidad de intercambio para modelos de simulación que surge del estándar FMI. Este FMU se puede exportar como co-simulación o model Exchange, dependiendo de las funcionalidades que se busquen.

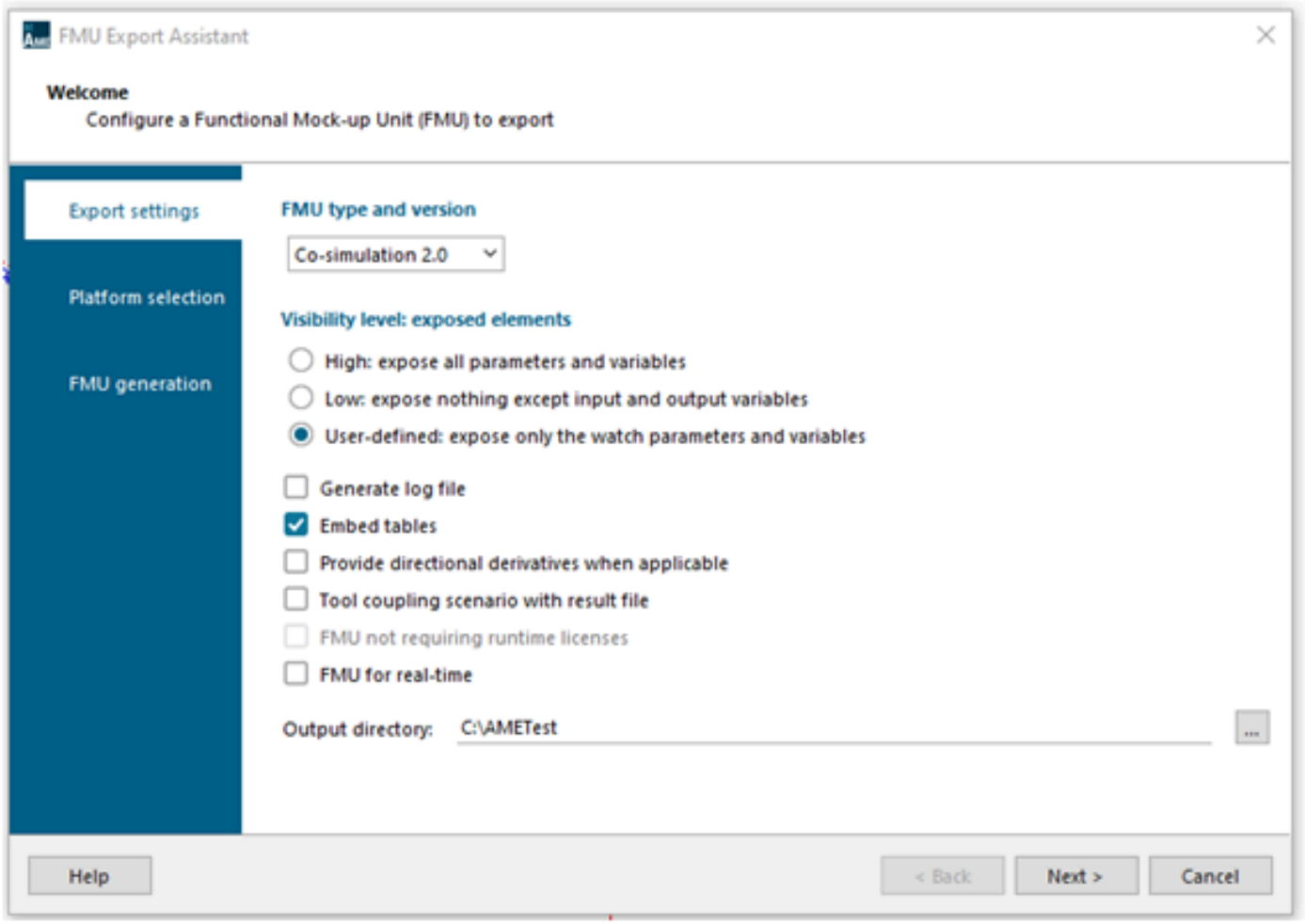


Figura 5. Configuración del estándar FMI en Simcenter Amesim

En Model Exchange, el FMU solo aporta el modelo (ecuaciones) y el solver lo pone la herramienta que importa.

- El FMU expone funciones para calcular derivadas de los estados continuos, pero no integra en el tiempo; la integración numérica (paso de tiempo, tolerancias, método) la controla el entorno de ejecución.
- Esta modalidad suele dar mejor rendimiento y control numérico, porque el solver puede optimizar el sistema combinado de varios FMUs dentro de la misma herramienta.

En Co-Simulation, cada FMU lleva su propio solver interno e integra su modelo de forma autónoma.

- El entorno de simulación actúa como master, avanza el tiempo global y pide a cada FMU que simule hasta el siguiente punto de comunicación, donde se intercambian entradas y salidas.
- Esto encapsula mejor a cada subsistema (incluso con diferentes solvers o pasos de tiempo), pero puede introducir retos de estabilidad y rendimiento, y suele requerir algoritmos de acoplamiento más sofisticados.

En la Tabla 1 se puede ver un resumen de las principales características de modo de exportación de un FMU.

Aspecto	Model Exchange (FMI)	Co-Simulation (FMI)
Solver	Lo aporta la herramienta importadora.	Va embebido en cada FMU.
Integración en el tiempo	El importer integra las ecuaciones del FMU.	El FMU integra sus propios estados y devuelve los valores actualizados.
Encapsulamiento	Menor encapsulamiento, mayor integración numérica global.	Fuerte encapsulamiento de IP y algoritmos internos.
Acoplamiento de FMUs	El solver puede tratar varios FMUs como un único sistema grande.	Se basa en algoritmo maestro con puntos de comunicación discretos.
Casos de uso típicos	Análisis detallado, simulación continua precisa dentro de una misma herramienta.	Integración de herramientas heterogéneas y subsistemas ya validados.

Tabla 1. Comparativa de los dos modos de exportación de un FMU.

La última versión del estándar, FMI 3.0, añade un tercer tipo de interfaz, Scheduled Execution, para sistemas puramente discretos y casos cercanos a tiempo real, además de Model Exchange y Co-Simulation. También introduce mejoras para co-simulación avanzada (acceso a variables intermedias, eventos, clocks, datos binarios, arrays y estándares en capas) que facilitan acoplamientos más robustos entre FMUs

4. Entornos de ejecución

Cualquier entorno de simulación puede ser un entorno de ejecución empleando el encapsulamiento del estándar FMI. Sin embargo, esto puede resultar limitante debido a la necesidad de licencias o conectividad con API's, etc. En general, resulta más sencillo trabajar con entornos open source, donde se pueda adaptar o introducir código específico para nuestra implementación. Es por este motivo que un entorno abierto como Python se convierte en una herramienta muy útil debido a que ofrece una amplia variedad de librerías con funciones que permiten adquirir, limpiar y filtrar datos, ejecutar FMU y realizar análisis estadísticos o de otro tipo a continuación. En definitiva, resulta más versátil para cualquier despliegue de esta tecnología.

A continuación se muestra un extracto de código en Python para ejecución de un FMU.

```
from fmpy import *
# FMU file
fmu_filename = 'Circuito_laboratorio.fmu'
# Load FMU
model_description = read_model_description(fmu_filename)
unzipdir = extract(fmu_filename)
# Instantiate the model
fmu = instantiate_fmu(unzipdir, model_description, fmi_type="CoSimulation")
# Create input signal: time array and the corresponding input values
time_steps = np.linspace(0.0, 1, num=10) # 10 steps between 0 and 1 second
# Define the two input values
valve_opening_target_values = np.full_like(time_steps, 50) # Constant value of 50 across all time steps
pump_speed_values = np.full_like(time_steps, 2000)     # Constant value of 2000 across all time steps
# The input needs to be a structured numpy array with 'time' and the input variable names as field names
input_values=np.array(list(zip(time_steps,valve_opening_target_values, pump_speed_values)),
                      dtype=[('time', np.float64),
('amesim_interface.Valve_opening', np.float64),
                      ('amesim_interface.Motor_rpm', np.float64)])
# Simulate
result = simulate_fmu(fmu_filename, start_time=0.0, stop_time=1, input=input_values)
```

5. Analítica

El análisis de datos procedente de un gemelo digital puede ser muy útil al disponer de datos reales y datos simulados, lo que da lugar a históricos muy completos. Para esto es necesario almacenar estos datos de forma ordenada para poder acceder a ellos fácilmente. Lo habitual es que se trabaje con datos estructurados, los que proceden de sensores lo son, por lo que una base de datos SQL es una buena opción.

A continuación, se muestra un ejemplo de código para la creación de una BD SQL.

```
import sqlite3
from sqlite3.dbapi2 import Timestamp
import pandas as pd
from datetime import datetime
import socket

# Create a SQLite database
conn = sqlite3.connect('machine1_database.db')
cursor = conn.cursor()
# Create a table for winch data
cursor.execute(
  """CREATE TABLE IF NOT EXISTS machine1_data (
  id INTEGER PRIMARY KEY AUTOINCREMENT,
  Timestamp TEXT,
  sensor_id TEXT,
  value INTEGER)""")
conn.commit()

# Insert data into the table
def insert_data(sensor_id, value):
  Timestamp = datetime.now().isoformat()
  cursor.execute(
        """INSERT INTO machine1_data (Timestamp, sensor_id, value) VALUES (?, ?, ?)""",
(Timestamp, sensor_id, value)
    )
  conn.commit()

# Consultar los datos
def query_data():
  df = pd.read_sql_query("SELECT * FROM machine1_data", conn)
  print(df)
```

El análisis de los datos, tanto en tiempo real como históricos, se debe enfocar dependiendo de las funcionalidades buscadas en el gemelo. De forma general, este análisis se puede enfocar en 4 grandes bloques, como son predicción de fallo, análisis de escenarios, análisis de datos procedentes de sensores virtuales y optimización de la operativa y/o energía consumida.

Con un gemelo digital se puede analizar la degradación progresiva de un equipo para determinar en qué momento puede alcanzar un fallo catastrófico. Para esto se pueden utilizar modelos de regresión (como LSTMs o modelos de degradación física) sobre los históricos de las variables (vibración, temperatura, etc.) para proyectar la tendencia futura. Además, permite planificar el mantenimiento justo antes de la rotura (Just-in-Time maintenance) en lugar de hacerlo por calendario o esperar al fallo.

El análisis de escenarios (What-if análisis) es posible debido a que tenemos una réplica virtual de la física de la máquina, por lo que es posible simular escenarios que serían peligrosos o costosos en la realidad. Aquí es donde entran los modelos físicos encapsulados conectados a modelos de ML (Machine Learning). Es decir, partiríamos de un modelo físico (ej. Modelica/Simulink) exportado en co-simulación (FMI) que alimentaría a un ML para ese análisis de escenarios.

A menudo no podemos poner sensores en todas partes (por coste o inaccesibilidad, como dentro de un pistón). En un gemelo digital se pueden usar los datos que sí se tienen (corriente, velocidad, temperatura externa, etc.) junto con modelos físicos o de IA para estimar variables que no se miden. Por ejemplo, calcular el par motor interno o la temperatura en el núcleo del bobinado basándose solo en la corriente y la temperatura de la carcasa.

Un gemelo puede tener como funcionalidad analizar la eficiencia de una máquina o sistema. Para esto, se deben detectar puntos de operación ineficientes. Por ejemplo, detectar que un ventilador consume un 5% más de energía de lo que debería para ese flujo de aire, indicando filtros sucios mucho antes de que la vibración sea una alarma. El gemelo podría sugerir (o ajustar automáticamente) los parámetros de control (PID) para minimizar el consumo energético sin afectar la producción.

A continuación se muestra un ejemplo de código en Python con un análisis de las variables de un equipo para la detección de fallo en éste.

```
import numpy as np
import pandas as pd
from sklearn.preprocessing import StandardScaler
import matplotlib.pyplot as plt
import seaborn as sns
from datetime import datetime
import warnings
warnings.filterwarnings('ignore')
def diagnosticar_fallo(self, temperatura, vibracion, ruido, horas, tipo_maquina, umbrales):
"""Identifica tipos específicos de fallo"""
  problemas = []   if temperatura > umbrales['temp_max']:
      problemas.append('Sobrecalentamiento detectado')
  if temperatura < umbrales['temp_min']:
      problemas.append('Temperatura anormalmente baja')
  if vibracion > umbrales['vibracion_max'] * 1.5:
      problemas.append('⚠️ Desequilibrio o desalineación severa')
  elif vibracion > umbrales['vibracion_max']:
      problemas.append('Vibración elevada - posible desgaste')
  if ruido > umbrales['ruido_max']:
      problemas.append('Ruido anómalo detectado')
 if horas > umbrales['horas_mantenimiento']:
      problemas.append('Mantenimiento preventivo vencido')
 return problemas if problemas else ['Sin anomalías detectadas']
```

6. Visualización

El gemelo digital necesita interactuar con operarios, técnicos, diseñadores, etc. por lo que una visualización que permita acceder a la información relevante de forma sencilla, fácil de interpretar, es fundamental. Para esto se puede emplear desde gráficas hasta realidad aumentada o virtual, dependiendo de la funcionalidad buscada en el gemelo. Habitualmente un dashboard bien configurado es suficiente.

Figura 6. Dashboard de selección de equipo

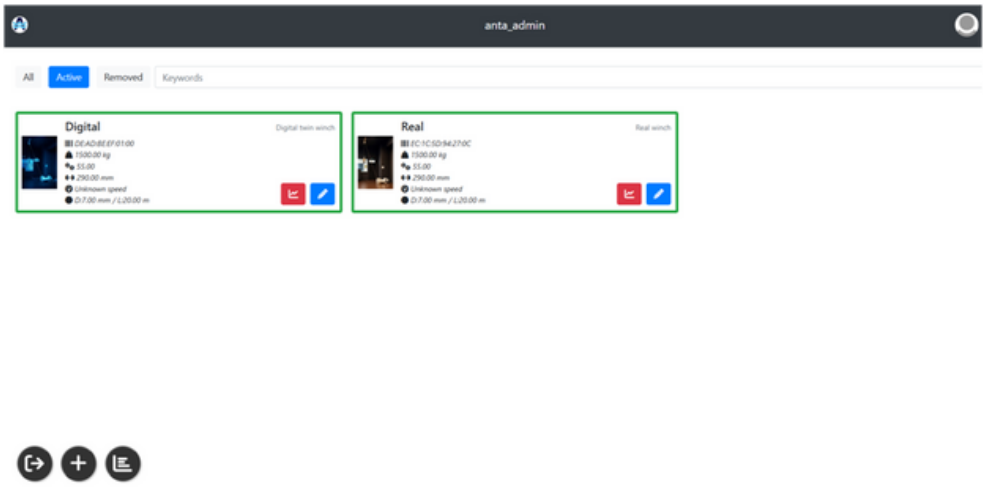


Figura 6. Dashboard de selección de equipo

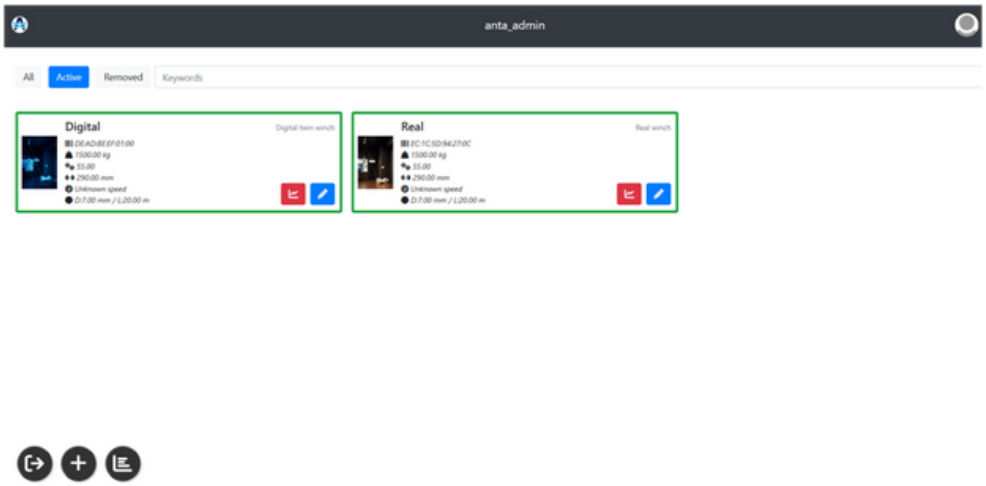


Figura 7. Dashboard de configuración de parámetros del equipo real y su equivalente digital

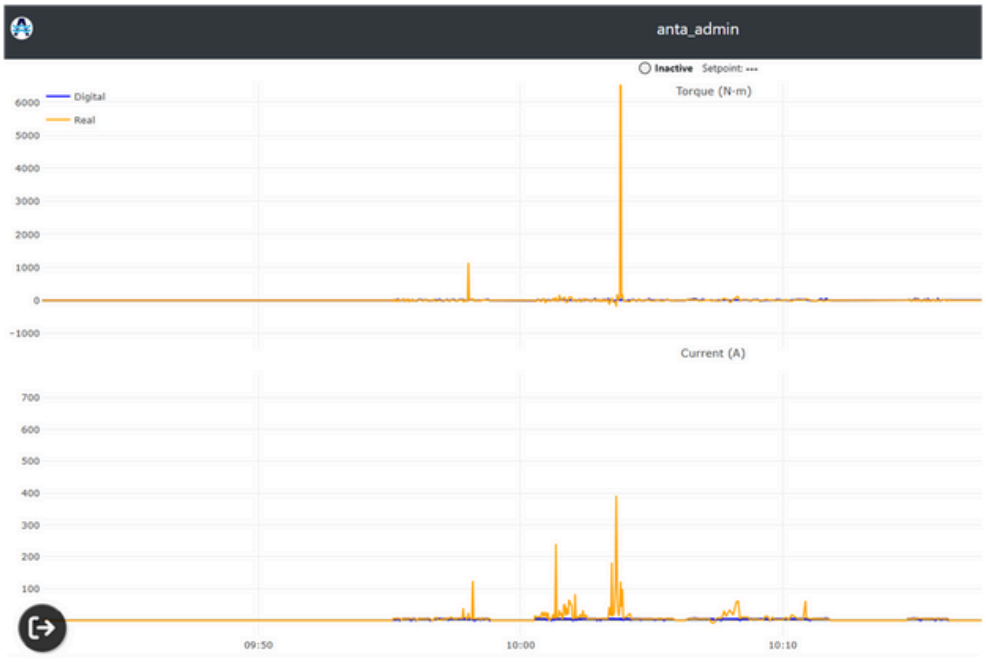


Figura 8. Dashboard con gráficas de datos reales y simulados

7. Ciberseguridad

Cualquier gemelo digital, al tener una parte IoT crea un puente bidireccional entre el mundo físico y el mundo digital, y esto implica vulnerabilidad. Al replicar no solo el estado sino también el comportamiento de un activo, el gemelo se convierte en el mapa perfecto para un atacante y, simultáneamente, en la puerta trasera definitiva hacia la maquinaria crítica.

Tradicionalmente, para sabotear una turbina o una red eléctrica, un atacante necesitaba meses de ingeniería inversa para entender el sistema. Con un Gemelo Digital comprometido, el atacante obtiene instantáneamente acceso a modelos CAD, algoritmos de control propietarios y parámetros de diseño, es decir, a la propiedad intelectual de aquellos que han desarrollado el gemelo. Además, se puede usar el gemelo como "banco de pruebas" (Sandboxing malicioso) para simular ciberataques contra el activo virtual hasta encontrar uno exitoso, sin arriesgarse a ser detectado en el sistema real.

En ciberseguridad estándar se protegen los datos. En Gemelos Digitales, la violación de datos puede tener consecuencias físicas. Si un atacante altera los datos del gemelo (data poisoning), puede inducir errores catastróficos. Por ejemplo, el gemelo muestra que la máquina está a 50°C (datos inyectados), mientras el activo real está ardiendo a 200°C. El operador no detiene la máquina porque confía en el gemelo. Si el gemelo tiene capacidades de control (nivel 3-4), una orden corrupta enviada desde el entorno virtual puede cambiar setpoints físicos, desactivar frenos o alterar mezclas químicas.

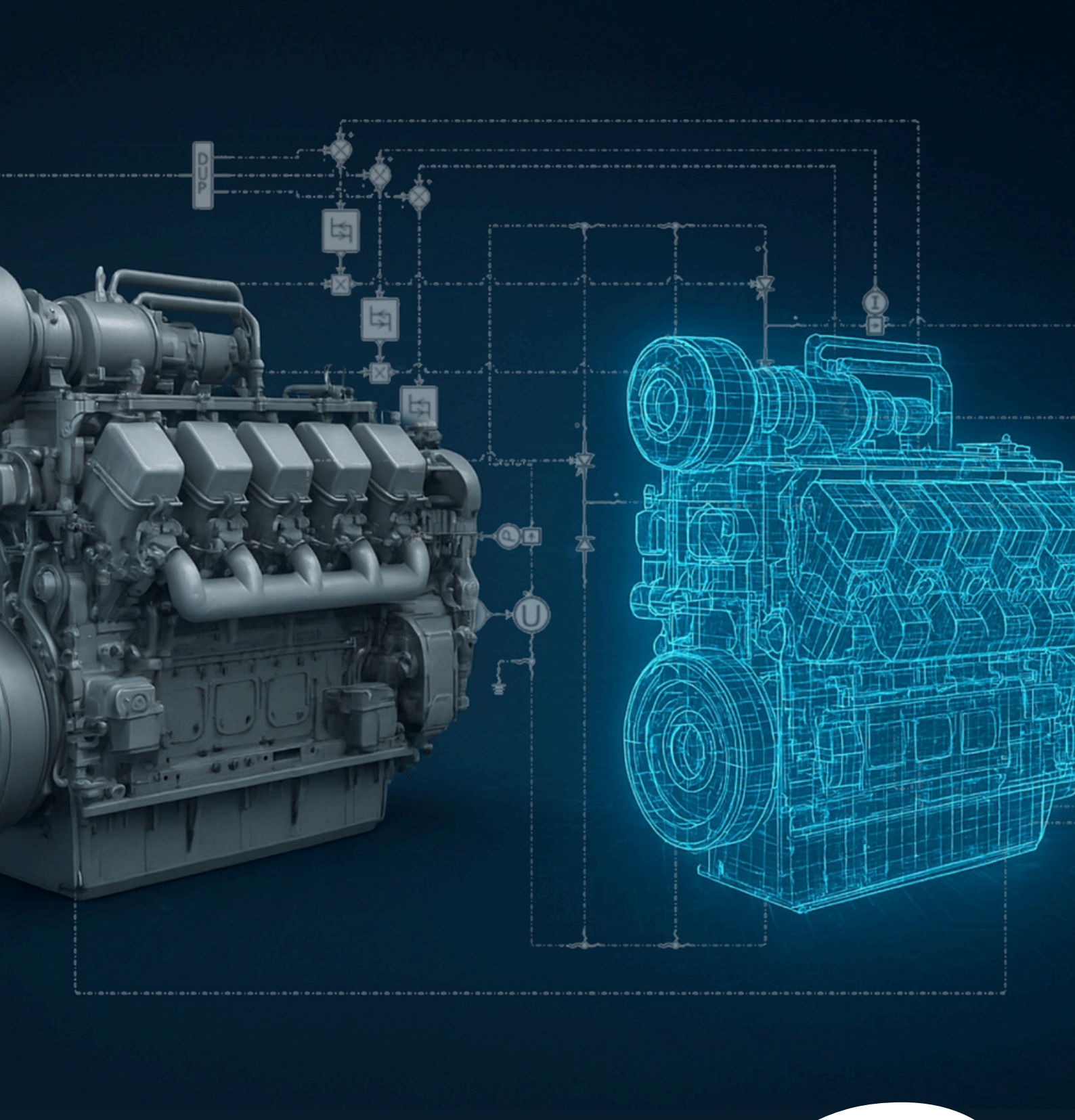
Los estándares como FMI intercambian modelos matemáticos (FMUs). Un archivo FMU malicioso podría contener código ejecutable que, al ser cargado por el orquestador de simulación, inyecte malware en el sistema host.

Dentro de la capa IoT los sensores están conectados al borde Edge, si no están asegurados sirven como punto de entrada para inyectar datos falsos que corrompen la simulación global.

Una arquitectura habitual en un gemelo es que este resida en la nube para aprovechar la potencia de cálculo. Las interfaces entre el sistema real y la nube son el punto de estrangulamiento donde ocurren la mayoría de las intrusiones.

Se pueden usar diferentes estrategias para evitar un ciberataque. Se puede optar por una arquitectura Zero Trust, donde se asume que ninguna señal, ni siquiera las que vienen de dentro del sistema físico, es confiable hasta que se verifique criptográficamente. También se puede optar por realizar verificaciones cruzadas, es decir, si un sensor indica una vibración extrema pero el consumo de corriente del motor no ha subido, el gemelo puede detectar esa "imposibilidad física" y marcar el sensor como "probablemente hackeado" en lugar de activar una alarma mecánica. Hoy en día, también es posible entrenar una IA de defensa para reconocer patrones de intrusión (anomalías en el tráfico de red o en la física del sistema) antes de que ocurran en la realidad.

La implementación de un Gemelo Digital no es solo un reto de ingeniería mecánica o de software, es fundamentalmente un reto de ciber-resiliencia. Un gemelo inseguro es, en esencia, una "puerta trasera" documentada y abierta hacia el corazón del cualquier sistema físico.



Guia Detalhado de Implementação de um Piloto de Manutenção Preditiva

2

A implementação de um piloto de manutenção preditiva envolve um conjunto de fases encadeadas que vão desde a definição do equipamento, passando pela recolha e tratamento dos dados, até à criação dos modelos preditivos e sua integração operacional. Este guia descreve, em detalhe, cada uma dessas fases, bem como as tecnologias e metodologias recomendadas para garantir um piloto realista, funcional e validado em ambiente industrial.

1. Definição do Equipamento e Objetivos do Piloto

A implementação de um piloto de manutenção preditiva no setor naval exige uma fase inicial extremamente rigorosa, uma vez que todo o sucesso do projeto depende da clareza com que se definem os seus limites, os objetivos operacionais e o equipamento envolvido. Antes de considerar algoritmos, tecnologias ou modelos, é indispensável compreender o ambiente marítimo, caracterizado por condições severas, restrições logísticas e elevada criticidade operacional. Por isso, esta primeira fase dedica-se a estabelecer o enquadramento técnico e estratégico, garantindo que o piloto é ao mesmo tempo realizável e representativo das necessidades reais da embarcação.
O primeiro passo consiste em selecionar o equipamento ou subsistema a monitorizar. Esta escolha deve ter em conta a relevância do equipamento para a segurança e funcionamento do navio. Muitos sistemas a bordo, tais como motores auxiliares, bombas hidráulicas, compressores de ar ou ventiladores de refrigeração funcionam continuamente e estão sujeitos a desgaste natural. Equipamentos deste tipo são candidatos ideais, tanto pela sua importância como pela previsibilidade dos seus modos de falha. Em navios mercantes, embarcações offshore ou plataformas marítimas, a escolha tende a recair em componentes cuja falha tem consequências diretas na operação, como bombas de refrigeração de água do mar, motores geradores ou sistemas de ar comprimido. Além disso, deve ser considerado o grau de acessibilidade a bordo, pois algumas áreas da casa de máquinas apresentam espaços reduzidos, limitações de segurança ou temperaturas elevadas que dificultam a instalação de sensores ou gateways.

Após a seleção do sistema-alvo, é necessário identificar quais variáveis operacionais melhor representam o estado de saúde desse equipamento. Em ambiente naval, esta identificação é particularmente importante, pois muitos fatores externos como as condições de mar, a vibração estrutural do casco, o calor na casa de máquinas e as variações bruscas de carga, afetam o comportamento dos equipamentos. Assim, a escolha das variáveis deve equilibrar relevância técnica, viabilidade de medição e sensibilidade à degradação. Por exemplo, a vibração de rolamentos é essencial para motores e bombas; a corrente elétrica diz muito sobre o esforço mecânico de motores; a temperatura revela desgaste ou problemas de lubrificação; a pressão e o caudal são fundamentais para sistemas hidráulicos e de refrigeração. É também importante considerar variáveis ambientais e de operação, tais como a temperatura da sala de máquinas, as condições de navegação ou o regime de funcionamento (manobra, cruzeiro ou porto), pois estas influenciam diretamente o perfil dos dados.

Esta fase deve igualmente avaliar as limitações práticas da operação marítima. Diferente de ambientes industriais em terra, o navio opera em isolamento, com comunicações limitadas e dependentes de satélite, sujeitas a latências elevadas ou instabilidade. Assim, torna-se necessário definir desde o início que parte do processamento será feita localmente, em dispositivos edge, e que dados serão enviados para sistemas em terra. Além disso, o equipamento instalado deve cumprir normas marítimas e de segurança, especialmente quando colocado em zonas de risco, como áreas com potencial de atmosfera explosiva ou locais sujeitos a vibrações extremas. Todas estas restrições devem ser conhecidas antes do início do piloto, para evitar soluções tecnicamente inadequadas ou incompatíveis com os regulamentos navais.

Paralelamente, é fundamental definir os objetivos técnicos e operacionais do piloto. A manutenção preditiva pode assumir várias abordagens: deteção precoce de anomalias, previsão de falhas com antecedência, estimativa da vida útil remanescente ou otimização de períodos de manutenção durante escalas em porto. No setor naval, estes objetivos são particularmente valiosos, pois permitem reduzir paragens inesperadas durante a navegação, evitar penalizações por falhas operacionais, otimizar janelas de manutenção portuária e, sobretudo, aumentar a fiabilidade de sistemas que, quando falham, colocam em risco não apenas a operação, mas também a segurança da tripulação e da embarcação.

Outro elemento fundamental nesta fase é a definição da duração do piloto e da estratégia de recolha de dados. Em navios, os equipamentos operam em diferentes regimes ao longo de uma viagem, pelo que é essencial recolher dados durante um período suficientemente longo para abranger todas as condições relevantes: operação contínua em mar alto, mudanças de carga durante manobras, arranques e paragens frequentes em porto, variações de temperatura e humidade, entre outros fatores. Geralmente, recomenda-se uma duração entre três e seis meses, dependendo da frequência de operação do equipamento selecionado. Um período demasiado curto pode não ser suficiente para identificar padrões de degradação ou para alimentar modelos que exigem diversidade de cenários.

Por fim, esta fase exige um alinhamento com os principais intervenientes do setor naval. A colaboração entre estes agentes garante a aceitação do sistema, a viabilidade operacional da instrumentação e o cumprimento de todas as normas de segurança e certificação marítima. Além disso, assegura que o piloto responde a necessidades reais e pode ser escalado posteriormente para outros equipamentos ou embarcações da frota.

2. Análise e Planeamento da Recolha de Dados

Depois de definidos os objetivos operacionais, a fase seguinte consiste numa análise técnica aprofundada que permitirá determinar como os dados serão obtidos, qual infraestrutura será necessária a bordo e que constrangimentos precisam de ser considerados na recolha contínua de informação. Esta fase é crítica, porque a manutenção preditiva depende inteiramente da qualidade, consistência e representatividade dos dados recolhidos. Se esta etapa não for tratada com rigor, todo o projeto ficará comprometido, mesmo que os modelos preditivos sejam avançados ou tecnologicamente sofisticados.

O ponto de partida desta análise consiste em avaliar a infraestrutura existente no navio e a sua capacidade para suportar monitorização contínua. Assim, é necessário compreender cuidadosamente o que já existe: sensores funcionais, localização física dos painéis de controlo, pontos de acesso Modbus ou OPC-UA, e disponibilidade de portas livres, tanto em termos elétricos como de rede.

Após esta avaliação da infraestrutura, passa-se à decisão sobre que sensores serão adicionados. A instalação não pode interferir com a operação normal do equipamento, especialmente em sistemas críticos. Com os sensores definidos, a fase seguinte consiste na escolha de um sistema de aquisição e transmissão de dados, geralmente baseado em dispositivos edge. Estes dispositivos, que podem ser computadores industriais, unidades embarcadas IoT ou equipamentos dedicados de fabricantes marítimos, desempenham duas funções essenciais: por um lado, recolhem dados dos sensores, agregam-nos e pré-processam-nos localmente; por outro, enviam-nos para servidores externos ou sistemas de monitorização conforme a disponibilidade da conectividade.

A decisão sobre o protocolo de comunicação é igualmente importante. Em ambiente naval, onde a conectividade é, por vezes, instável ou intermitente, protocolos leves e resilientes, como MQTT, podem ser preferidos para envio de dados em tempo real. OPC-UA é mais utilizado para integração com sistemas existentes a bordo, oferecendo maior segurança e interoperabilidade industrial.

Outro elemento crítico nesta fase é a definição das frequências de amostragem. Equipamentos navais apresentam diferentes dinâmicas físicas, e a taxa de recolha deve evitar dados insuficientes ou desnecessariamente volumosos. Dados de vibração, usados para detetar falhas em rolamentos ou desalinhamentos, requerem amostragens elevadas enquanto variáveis como temperatura ou pressão podem ser registadas com muito menor frequência. A definição incorreta da amostragem pode comprometer toda a capacidade diagnóstica, pelo que esta decisão deve ser feita em conjunto com engenheiros de bordo e especialistas em análise de sinais.

Com as questões de sensorização e transmissão resolvidas, torna-se fundamental planear o armazenamento dos dados, tanto localmente quanto remotamente. A bordo, o armazenamento temporário deve ser dimensionado para garantir que a perda de conectividade não compromete o piloto. Em terra, a escolha entre bases de dados locais ou de cloud depende das políticas da empresa e dos requisitos regulatórios. A utilização de bases de dados de séries temporais, como InfluxDB ou TimescaleDB, é amplamente recomendada, dada a natureza regular e contínua das medições. Estas bases facilitam o processamento subsquente, permitindo agregações rápidas e extração eficiente de grandes volumes de dados.

Finalmente, esta fase deve considerar o mapeamento completo do ciclo de dados, desde a origem no sensor até ao processamento final, garantindo que cada etapa é tecnicamente viável a bordo. Deve-se identificar claramente como serão tratadas questões como perda de pacotes, latência, sincronização entre sensores e capacidade energética dos dispositivos adicionais. A análise técnica nesta fase não requer ainda decisões sobre modelos preditivos, mas deve preparar as condições que garantem que esses modelos serão alimentados com dados de qualidade, consistentes e representativos do ambiente real de operação naval.

3. Arquitetura Técnica e Desenvolvimento da Infraestrutura de Dados

A construção da arquitetura técnica representa um dos pilares centrais de qualquer piloto de manutenção preditiva, pois define de que forma os dados serão captados, processados, armazenados, modelados e disponibilizados às equipas de operação e manutenção. No contexto naval, esta fase assume particular importância devido à diversidade dos sistemas, à complexidade das condições de funcionamento e às limitações físicas e tecnológicas que caracterizam o ambiente marítimo. Uma arquitetura bem delineada garante não só o correto funcionamento do piloto, mas também a sua escalabilidade para outros equipamentos ou mesmo para toda a frota.

O ponto de partida para a definição da arquitetura é o desenho do fluxo completo de dados, desde o sensor instalado no equipamento até ao modelo preditivo que operará no servidor ou no sistema edge. Esse fluxo deve ser robusto o suficiente para lidar com as condições desafiantes a bordo, como vibrações, variações de energia, ruído eletromagnético e conectividade intermitente. Assim, é fundamental estabelecer uma cadeia de recolha que inclua sensores industriais robustos, sistemas adequados de aquisição de dados, dispositivos de computação periférica (edge computing) e mecanismos de sincronização e envio para plataformas em terra. Cada um destes elementos deve ser compatível tanto com as condições físicas do navio como com os protocolos de comunicação e segurança exigidos.

Após a definição do caminho dos dados, passa-se à criação da infraestrutura de processamento local, geralmente designada por camada edge. Esta camada é essencial na indústria naval, uma vez que o navio não pode depender exclusivamente da ligação à terra para processar medições em tempo real.

Assim, equipamentos como computadores industriais compactos, PLCs programáveis são utilizados para recolher os dados dos sensores, realizar pré-processamento inicial (como filtragem, agregação ou deteção preliminar de anomalias) e assegurar que os dados são armazenados temporariamente quando a conectividade é insuficiente. A capacidade de computação desta camada deve ser dimensionada para suportar operações contínuas e, sempre que possível, hospedar versões simplificadas dos modelos preditivos, permitindo análises imediatas mesmo em mar alto.

A arquitetura deve igualmente especificar os protocolos de comunicação utilizados em cada ligação. Em muitos navios, coexistem tecnologias de diferentes gerações, desde sistemas legacy baseados em Modbus RTU a redes modernas com OPC-UA. A integração destes sistemas requer planeamento cuidadoso, especialmente quando o piloto pretende ser compatível com uma frota heterogénea. Protocolos como MQTT são frequentemente adotados para o envio de dados para servidores externos, devido à sua eficiência, tolerância a falhas e baixa exigência de largura de banda. Já OPC-UA é ideal para recolher dados de sistemas automáticos existentes, oferecendo interoperabilidade e normas de segurança robustas. A escolha destes protocolos deve considerar não apenas a eficiência, mas também requisitos de cibersegurança, uma vez que navios são cada vez mais alvos de ataques cibernéticos.

Definido o fluxo local, a fase seguinte consiste em desenhar a camada de armazenamento e processamento em terra, que será responsável pela análise histórica, treino dos modelos preditivos e execução de algoritmos mais pesados do ponto de vista computacional. É nesta camada que se integra uma base de dados de séries temporais, normalmente implementada através de soluções como InfluxDB, TimescaleDB ou bases cloud otimizadas para dados IoT. O armazenamento em terra permite concentrar grandes volumes de dados de várias embarcações, permitindo comparações entre equipamentos, benchmarking interno e a construção de modelos transversais. Ao mesmo tempo, esta camada funciona como repositório central para dashboards operacionais, sistemas de alerta e ferramentas de análise avançada.

A arquitetura técnica deve ainda contemplar o mecanismo de transferência de dados entre o navio e a infraestrutura em terra. Esta transferência pode ocorrer de forma contínua, quando a embarcação dispõe de conectividade satélite estável, ou pode ser realizada em modo assíncrono, com sincronizações periódicas, por exemplo, quando o navio está em porto. Esta decisão depende do tipo de dados recolhidos, da frequência de amostragem e das necessidades operacionais do piloto. Para dados de vibração de alta frequência, por exemplo, não é realista enviar tudo para terra em tempo real; nesses casos, parte do processamento deve ocorrer localmente, enviando-se apenas métricas agregadas e indicadores de saúde.

Um elemento essencial desta fase é a definição dos serviços de aplicação e camadas de visualização. Estes serviços serão responsáveis por apresentar, de forma intuitiva e acessível, a informação proveniente dos sensores e dos modelos. Em ambientes navais, onde os operadores podem estar sob forte carga de trabalho, dashboards claros e bem estruturados são fundamentais. Assim, a arquitetura deve incluir ferramentas de visualização como Grafana, Power BI ou plataformas proprietárias integradas com o sistema de automação do navio. Além disso, a arquitetura deve permitir integrar facilmente funcionalidades adicionais, como alarmes preditivos, relatórios automáticos de tendência ou sistemas de apoio à decisão baseados em inteligência artificial.

Outro fator importante a considerar é a interoperabilidade com sistemas existentes a bordo e em terra. Os sistemas de gestão de manutenção (CMMS), como o Maximo, ManWinWin ou SAP PM gerem ordens de trabalho, inventário e planeamento. A arquitetura técnica deve, idealmente, permitir a integração com estes sistemas, automatizando fluxos como a abertura de uma ordem de manutenção quando o modelo preditivo identifica uma degradação relevante. A integração pode ocorrer através de APIs, conectores middleware ou interfaces dedicadas, conforme a maturidade digital da frota.

Por fim, a arquitetura deve contemplar mecanismos de cibersegurança e resiliência, especialmente porque navios operam em ambientes isolados e são cada vez mais alvo de ataques direcionados. Isto inclui encriptação dos dados, autenticação forte nos dispositivos edge, firewalls marítimas certificadas e políticas de segmentação das redes do navio, garantindo que o piloto de manutenção preditiva não introduz vulnerabilidades no ecossistema operacional da embarcação.

Quando concluída, esta fase entrega uma arquitetura coerente, robusta e adaptada às limitações do ambiente naval, estabelecendo os alicerces que permitem, nas fases seguintes, desenvolver os modelos preditivos e integrá-los com os sistemas reais de bordo.

4. Desenvolvimento dos Modelos Preditivos

Depois de estabelecida a arquitetura técnica e assegurado o fluxo de dados entre sensores, dispositivos edge e sistemas de armazenamento, inicia-se a fase dedicada ao desenvolvimento dos modelos de simulação e dos modelos preditivos. Esta é uma etapa determinante no piloto de manutenção preditiva, pois traduz a informação recolhida em conhecimento acionável sobre o estado dos equipamentos, permitindo antecipar falhas, detetar anomalias, estimar degradação e apoiar decisões de manutenção em tempo útil. A abordagem de modelação nesta fase depende fortemente da quantidade e qualidade dos dados recolhidos, bem como da complexidade física do equipamento monitorizado e das suas condições de operação a bordo.

Análise Exploratória de Dados e pré-processamento

O primeiro passo consiste na análise exploratória dos dados disponíveis. Antes de treinar qualquer modelo, é necessário compreender a estrutura temporal dos sinais recolhidos, a sua variabilidade em diferentes regimes de operação, padrões cíclicos relacionados com a navegação do navio e possíveis anomalias decorrentes de eventos externos. Em ambiente naval, esta etapa ganha especial relevância, porque muitos fatores externos influenciam diretamente os dados: alterações bruscas de carga durante manobras, vibrações induzidas pela rotação da hélice ou do eixo propulsor, condições de mar adversas e impactes térmicos devidos ao funcionamento prolongado da casa de máquinas. A segmentação dos dados por regime operativo, por exemplo, navegação em cruzeiro, manobra ou porto, é essencial para evitar que o modelo confunda variações normais de operação com sinais de degradação.

A partir desta análise inicial, procede-se ao pré-processamento aprofundado dos dados, que pode incluir filtragem de ruído, normalização, extração de características espectrais e construção de janelas temporais que sintetizam a evolução do equipamento ao longo do tempo. Para sinais de vibração, recorre-se frequentemente à Transformada Rápida de Fourier (FFT) ou a wavelets para analisar frequências específicas associadas a desequilíbrios, folgas ou desgaste de rolamentos. Já em sinais elétricos, como corrente ou potência consumida por motores, a extração de características estatísticas (média, variância, assimetria, curtose) e a análise do comportamento sob diferentes cargas ajudam a indicar padrões de envelhecimento ou sobreaquecimento. Em suma, o pré-processamento transforma sinais brutos em informação estruturada e analiticamente significativa, fundamental para o sucesso dos modelos preditivos.

Modelação

Concluído o pré-processamento, passa-se à definição da estratégia de modelação. Em manutenção preditiva, os modelos podem seguir três grandes abordagens: modelos de deteção de anomalias, modelos supervisionados e modelos de previsão da vida útil remanescente. A escolha entre estas abordagens depende sobretudo da disponibilidade de dados históricos de falhas, que é muitas vezes limitada no setor naval. Em navios, falhas têm impacto elevado e, por isso, são relativamente raras; além disso, os registos podem ser incompletos ou não incluir sensores suficientes para reconstruir o comportamento exato no período anterior à falha. Por esta razão, muitos pilotos navais começam com modelos de deteção de anomalias, que aprendem o comportamento normal do equipamento e sinalizam desvios.

Modelos de deteção de anomalias incluem técnicas como Isolation Forest, One-Class SVM, Local Outlier Factor ou autoencoders neuronais. Estas abordagens não requerem dados de falha, mas sim um volume suficientemente grande de dados representando operação normal. Após o treino, o modelo cria uma espécie de “assinatura digital” do comportamento esperado do equipamento e passa a detetar padrões que escapam a essa normalidade.

Num compressor de ar ou numa bomba de água de refrigeração, por exemplo, um aumento subtil mas contínuo de vibração em determinada banda de frequência pode ser sinal de desgaste precoce, ainda impercetível para a tripulação.

Quando existem dados rotulados , isto é, quando é possível identificar períodos históricos em que ocorreram falhas ou degradações relevantes, torna-se possível treinar modelos supervisionados. Estes modelos aprendem a distinguir entre estados saudáveis e estados anómalos, classificando novos dados segundo o padrão aprendido. Algoritmos como Random Forest, XGBoost ou Support Vector Machines são amplamente usados, não só pela sua precisão, mas também pela capacidade de explicar parcialmente quais variáveis mais contribuem para a degradação. Esta interpretabilidade é especialmente valorizada no ambiente naval, onde decisões de manutenção influenciam diretamente a segurança da embarcação.

A terceira abordagem envolve a estimativa da vida útil remanescente (Remaining Useful Life (RUL)). Esta técnica é particularmente relevante em equipamentos sujeitos a desgaste progressivo, como rolamentos, vedantes, sistemas hidráulicos e motores auxiliares com longos ciclos de operação. Modelos de RUL incluem regressões avançadas, redes neuronais recorrentes como LSTM ou GRU, e modelos probabilísticos baseados em distribuições de Weibull ou Cox Proportional Hazards. A previsão de RUL permite planear intervenções durante paragens previstas em porto, evitando avarias durante a navegação e reduzindo custos de manutenção corretiva.

A construção e o treino dos modelos dependem de ferramentas adequadas. Em ambiente académico e industrial, Python é a linguagem predominante, e bibliotecas como Pandas, NumPy, scikit-learn, TensorFlow, PyTorch, tsfresh e Prophet desempenham um papel central. Estas bibliotecas permitem desde manipulação de dados a treino de redes neuronais complexas, sempre com um ecossistema consolidado e amplamente validado. Para além disso, ferramentas como MLflow podem ser utilizadas para acompanhar os resultados de cada experiência, garantindo reprodutibilidade e rastreabilidade, aspetos críticos quando modelos começam a influenciar decisões reais de manutenção.

Concluído o treino, o modelo precisa de ser validado com dados de teste e posteriormente com dados reais em funcionamento contínuo. No setor naval, essa validação deve considerar diferentes estados de navegação, condições ambientais extremas e variações de carga típicas da operação. A validade de um modelo só pode ser confirmada se ele for capaz de identificar padrões de degradação em cenários complexos e não linearmente correlacionados. Após validado, o modelo é preparado para ser integrado na arquitetura definida anteriormente, podendo ser executado a bordo, no edge, ou num servidor remoto, dependendo das necessidades e limitações da operação. Assim, a Fase 4 termina com a criação de um conjunto de modelos preditivos, prontos para serem integrados no piloto. Estes modelos constituem o núcleo da manutenção preditiva e representam a inteligência analítica que transforma dados operacionais do navio em recomendações, alertas e previsões capazes de melhorar a segurança, eficiência e fiabilidade da embarcação.

Interreg

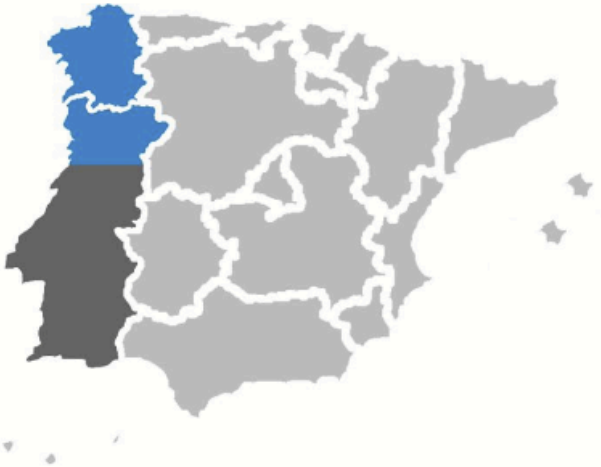
Cofinanciado por la Unión Europea
Cofinanciado pela União Europeia

España – Portugal

TwinNavAux

AXENCIA GALEGA DE INNOVACIÓN

Asociación Cluster del Naval Gallego **(ACLUNAGA)**



Universidad de da Coruña **(UDC)**

Industrias Ferri SA

Ibercisa Deck Machinery SA

Electrorayma SL

Centro de Apoio Tecnológico à Indústria Metalomecânica **(CATIM)**

Universidade Portucalense Infante Dom Henrique

Projeto cofinanciado pela União Europeia através do Programa Interreg VI-A Espanha-Portugal (POCTEP) 2021-2027