# Mantenimiento Predictivo para un Moto-Compresor con Agentes de IA

**I. Introducción al Proyecto de Simulación de Mantenimiento Predictivo**

**A. Contextualización del Mantenimiento Predictivo (PdM) en la Industria 4.0.**

El Mantenimiento Predictivo (PdM) se ha consolidado como una estrategia fundamental en el paradigma de la Industria 4.0, respondiendo a la creciente necesidad de optimizar los costos operativos y minimizar las interrupciones no planificadas en la producción industrial.1 La Industria 4.0, con su énfasis en la interconexión de sistemas ciberfísicos, el Internet de las Cosas (IoT) y el análisis avanzado de datos, proporciona un terreno fértil para la implementación de estrategias de PdM cada vez más sofisticadas y efectivas.2 Este enfoque representa un cambio significativo desde las metodologías de mantenimiento tradicionales, como el mantenimiento correctivo (reaccionar a las fallas una vez que ocurren) o el mantenimiento preventivo (basado en cronogramas fijos, independientemente de la condición real del equipo). El PdM, en cambio, busca anticiparse a las fallas mediante el monitoreo continuo de la condición de los activos y el análisis de los datos recolectados para predecir cuándo es probable que ocurra una falla.4 Las promesas del PdM son considerables: reducción de costos de mantenimiento, extensión de la vida útil de los activos, mejora de la eficiencia operativa y minimización del tiempo de inactividad.4

La transición hacia el PdM no es meramente una actualización técnica, sino una transformación estratégica que repercute directamente en la competitividad y la sostenibilidad de las empresas. Al reducir los costos asociados con las paradas inesperadas y las reparaciones urgentes 1, y al prolongar la operatividad de la maquinaria crítica 5, las organizaciones pueden mejorar significativamente sus márgenes de beneficio. Además, un mantenimiento más eficiente y planificado contribuye a una mayor seguridad en el entorno laboral 5 y a una optimización general de los procesos productivos.5 Estos beneficios, en su conjunto, fortalecen la posición de una empresa en un mercado cada vez más exigente y dinámico.

**B. El Papel Convergente de la Simulación y la Inteligencia Artificial en el PdM Avanzado.**

La simulación y la Inteligencia Artificial (IA) son dos tecnologías que, al converger, potencian significativamente las capacidades del PdM. La simulación de eventos discretos, por ejemplo, permite modelar sistemas complejos y probar diferentes escenarios de mantenimiento y operación en un entorno virtual, libre de riesgos y costos asociados con la experimentación en equipos reales.7 Esta capacidad es invaluable para evaluar el comportamiento de un sistema bajo diversas estrategias de mantenimiento y para comprender el impacto de diferentes parámetros operativos en la degradación de los equipos.

Por otro lado, la IA, y en particular el Machine Learning (ML) y el Aprendizaje por Refuerzo (RL), ofrecen herramientas poderosas para analizar los grandes volúmenes de datos generados por los sensores en los equipos industriales. Estas técnicas permiten identificar patrones sutiles en los datos que pueden ser indicativos de una degradación incipiente o de un riesgo de falla futuro, así como automatizar la toma de decisiones de mantenimiento.1 La IA puede, por ejemplo, predecir la Vida Útil Remanente (RUL) de un componente o clasificar el estado de salud de una máquina basándose en sus datos operativos.

La integración de la simulación con la IA abre nuevas fronteras para el PdM. Un aspecto crucial es la capacidad de la simulación para generar datos sintéticos que pueden ser utilizados para entrenar modelos de IA, especialmente en situaciones donde los datos históricos de fallas son escasos o inexistentes.9 Esto es particularmente relevante para equipos nuevos o para modos de fallo raros. Además, los entornos de simulación pueden servir como "campos de entrenamiento" para agentes de IA, permitiéndoles aprender y optimizar políticas de mantenimiento a través de la interacción con el modelo simulado, sin afectar la operación real.11 Un agente de IA puede, por ejemplo, explorar diferentes secuencias de acciones de mantenimiento dentro de la simulación y aprender cuál es la más efectiva para maximizar la disponibilidad del equipo o minimizar los costos.

Esta sinergia crea un ciclo virtuoso: la simulación proporciona un "gemelo digital" 7 del sistema físico, un entorno seguro y controlado donde la IA puede "aprender más rápido y mejor".7 A su vez, la IA puede "dirigir" la simulación para que se asemeje más a la realidad, ajustando parámetros del modelo simulado basados en datos reales, o puede utilizarse para explorar y optimizar las políticas de mantenimiento dentro de la propia simulación.1 Los resultados y conocimientos obtenidos de esta interacción informan el diseño de sistemas de PdM más robustos y eficientes, y pueden guiar la toma de decisiones en el mundo real.17 Esta capacidad de experimentación y aprendizaje acelerado es especialmente valiosa para sistemas industriales complejos donde la experimentación directa en el equipo físico es costosa, disruptiva o incluso peligrosa.

**C. Definición y Alcance para el Proyecto:**

Los **alcances** podrían ser:

1. Desarrollar un modelo de simulación de eventos discretos simplificado de un moto-compresor utilizando la librería SimPy. Este modelo deberá representar la operación básica del equipo y un mecanismo de degradación progresiva.
2. Diseñar e implementar un agente de Inteligencia Artificial, ya sea basado en reglas simples o un modelo de Machine Learning muy básico, capaz de interpretar el estado del moto-compresor simulado y tomar decisiones de mantenimiento predictivo.
3. Integrar el agente de IA con el modelo de simulación para crear un sistema de PdM funcional dentro del entorno simulado.
4. Evaluar el rendimiento del sistema de PdM simulado bajo diferentes escenarios de operación y fallo, comparando sus resultados con estrategias de mantenimiento más simples (ej. mantenimiento correctivo o preventivo basado en tiempo).
5. Documentar exhaustivamente el proceso de diseño, implementación y evaluación, culminando en un informe técnico.

Se aplicarán los principios del Producto Mínimo Viable (MVP) 19 y la técnica de priorización MoSCoW (Must have, Should have, Could have, Won't have).21 Esto implica:

* **Simplificación del modelo del moto-compresor:** No se intentará modelar cada detalle físico del equipo, sino sus características funcionales más relevantes para el PdM.
* **Modelo de degradación simplificado:** Se utilizará un modelo de degradación conceptual (ej. una variable de "salud" que disminuye con el uso) en lugar de modelos físicos complejos.
* **Agente de IA con alcance limitado:** Si se opta por un agente basado en reglas, estas serán pocas y claras. Si se explora un modelo de ML, será uno muy simple (ej. regresión logística o árbol de decisión) entrenado con datos generados por la propia simulación o un pequeño dataset público adaptado.
* **Número limitado de escenarios de prueba:** Se definirán unos pocos escenarios de fallo y operación clave para la evaluación, en lugar de una batería exhaustiva de pruebas.

Al definir el alcance de un proyecto de ML/IA, es crucial considerar la definición clara del problema que se quiere resolver, la disponibilidad y características de los datos, los recursos disponibles (tiempo del estudiante, software, hardware si se contempla un piloto físico conceptual) y el cronograma detallado.24

El objetivo primordial no es desarrollar un sistema de PdM de nivel industrial, sino demostrar una sólida comprensión y la capacidad de aplicar los principios fundamentales de la simulación de eventos discretos, la inteligencia artificial y las estrategias de mantenimiento predictivo a un caso de estudio relevante y manejable. El proceso de investigación, diseño, implementación, análisis y, fundamentalmente, la documentación rigurosa del proyecto, son tan importantes como el resultado técnico final. La complejidad debe ser gestionable 25, y el MVP debe enfocarse en las características mínimas que permitan al sistema funcionar en su forma más básica y demostrar el concepto.19 Adicionalmente, adoptar un "Proceso Mínimo Viable" 19, que se centre en establecer un flujo de trabajo de desarrollo funcional desde el inicio, puede ser muy beneficioso para mantener el proyecto encarrilado.

**II. Fundamentos del Mantenimiento Predictivo para Moto-Compresores Industriales**

**A. Principios Clave del PdM y su Aplicación Específica a Moto-Compresores.**

El Mantenimiento Predictivo (PdM) se fundamenta en el monitoreo continuo o periódico de la condición operativa de los equipos para detectar signos tempranos de degradación o anomalías que puedan preceder a una falla.6 A diferencia del mantenimiento preventivo, que se basa en intervalos de tiempo fijos o ciclos de uso, y del mantenimiento reactivo, que actúa solo después de que la falla ha ocurrido, el PdM busca intervenir en el momento óptimo: justo antes de que la falla se materialice, pero no tan prematuramente como para incurrir en costos de mantenimiento innecesarios.29

En el caso específico de los moto-compresores, que son equipos críticos en numerosas aplicaciones industriales, el PdM implica la supervisión de una serie de parámetros clave. Estos pueden incluir la vibración, la temperatura en puntos críticos (como rodamientos o el motor), la presión en diferentes etapas del compresor, las características del aceite lubricante (viscosidad, presencia de partículas, nivel de acidez) y la corriente consumida por el motor eléctrico.6 El análisis de estos parámetros, a menudo mediante el uso de sensores y sistemas de adquisición de datos, permite identificar desviaciones respecto a los patrones de operación normal, lo que puede indicar un desgaste incipiente o un problema en desarrollo.

El objetivo final de aplicar PdM a moto-compresores es múltiple: reducir el tiempo de inactividad no planificado, que puede tener consecuencias económicas severas; optimizar los cronogramas de mantenimiento, realizando intervenciones solo cuando son necesarias; y extender la vida útil operativa del compresor, maximizando así el retorno de la inversión en el activo.4

La aplicación efectiva del PdM a moto-compresores no es una solución genérica, sino que exige una comprensión profunda de los mecanismos de degradación y los modos de fallo específicos de cada tipo de compresor y sus componentes. Por ejemplo, los compresores de tornillo y los de pistón tienen diferentes puntos vulnerables y, por lo tanto, pueden requerir el monitoreo de parámetros distintos o el uso de técnicas de análisis diferentes.27 El análisis de aceite, una técnica común en PdM, puede revelar una variedad de problemas específicos; por ejemplo, niveles anómalos de pH pueden ser indicativos de desgaste corrosivo en los rodamientos, mientras que un número ácido elevado puede señalar la oxidación del aceite o el agotamiento de sus aditivos, lo que a su vez puede llevar a la corrosión de partes internas del compresor.27 Esta especificidad subraya la necesidad de un enfoque adaptado al equipo particular que se está monitoreando.

**B. Análisis de Modos de Fallo Comunes (FMEA Simplificado) y Parámetros Críticos de Monitoreo.**

Para diseñar un sistema de PdM eficaz, es esencial identificar los modos de fallo más probables y críticos del equipo. El Análisis de Modos de Fallo y Efectos (FMEA, por sus siglas en inglés) es una metodología sistemática para lograr esto. Para lo cual, se realizará un FMEA simplificado, enfocándose en los modos de fallo más relevantes para un moto-compresor genérico y que sean susceptibles de ser modelados en una simulación.35 Este análisis identificará los modos de fallo potenciales, sus efectos sobre el sistema, sus causas probables y los métodos de detección existentes o propuestos.

Los parámetros críticos de monitoreo, derivados en parte del FMEA, son aquellas variables operativas o de condición cuya evolución puede indicar la progresión de un modo de fallo. Para los moto-compresores, estos suelen ser:

* **Temperatura:** El monitoreo de la temperatura en puntos clave como el motor, los rodamientos, las cajas de engranajes y los sistemas hidráulicos es crucial. Niveles de calor anormales a menudo son el primer indicio de problemas como fricción excesiva debido a una lubricación deficiente, desalineación, o problemas eléctricos.6
* **Vibración:** El análisis de vibraciones es una técnica poderosa para detectar signos tempranos de problemas mecánicos como desequilibrio en partes rotativas, holgura mecánica, problemas en engranajes o el desgaste de rodamientos.6 Diferentes tipos de fallos generan patrones de vibración característicos (firmas) que pueden ser identificados.
* **Presión:** En un compresor, la presión es un parámetro operativo fundamental. Desviaciones en los niveles de presión esperados en las diferentes etapas pueden indicar fugas internas o externas, bloqueos en las líneas, o fallas en las válvulas o sellos, comprometiendo la eficiencia del sistema.4
* **Análisis de Aceite/Lubricación:** La condición del lubricante es vital para la salud de los componentes móviles. El análisis de aceite puede detectar la degradación del propio aceite (viscosidad, acidez, oxidación), la presencia de contaminantes (agua, partículas de suciedad) y la presencia de partículas de desgaste metálico, que pueden indicar el desgaste acelerado de componentes específicos como rodamientos o engranajes.6
* **Corriente del Motor:** El monitoreo de la corriente consumida por el motor eléctrico que impulsa el compresor puede revelar problemas tanto eléctricos (fallos en el bobinado, problemas de aislamiento) como mecánicos (aumento de la carga debido a fricción excesiva en el compresor).4

El FMEA no solo es crucial para identificar qué aspectos del moto-compresor se deben simular con mayor detalle, sino que también ayuda a definir los "estados de fallo" dentro del modelo de simulación. Estos estados son las condiciones que el agente de IA deberá ser capaz de detectar y sobre las cuales deberá tomar decisiones de mantenimiento. Un FMEA identifica los modos de fallo, sus efectos y sus causas.36 Estos elementos se traducen directamente en los "eventos" o "condiciones" que el sistema de PdM simulado debe reconocer y gestionar. El Número de Prioridad de Riesgo (RPN), calculado como el producto de la Severidad (S), la Ocurrencia (O) y la Detección (D) de un modo de fallo (RPN = S x O x D) 36, es una herramienta cuantitativa que permite priorizar los modos de fallo. Aquellos con un RPN más alto representan los riesgos más significativos y, por lo tanto, deberían ser el foco principal del modelado y la simulación en un proyecto con recursos limitados.

A continuación, se presenta una tabla con un FMEA simplificado para un moto-compresor, que servirá de base para definir los escenarios de simulación:

**Tabla 1: FMEA Simplificado del Moto-Compresor**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Componente** | **Modo de Fallo Potencial** | **Efecto Potencial del Fallo** | **Causa Potencial del Fallo** | **Controles Actuales (Simulados)** | **Severidad (S) (1-10)** | **Ocurrencia (O) (1-10)** | **Detección (D) (1-10)** | **RPN (S\*O\*D)** |
| Motor Eléctrico | Sobrecalentamiento | Reducción de vida útil, fallo catastrófico del motor, parada del compresor. | Sobrecarga, ventilación deficiente, fallo del sistema de refrigeración del motor, problemas de tensión. | Sensor de temperatura simulado en el motor, monitor de corriente simulado. | 8 | 6 | 4 | 192 |
|  | Fallo de Rodamiento del Motor | Ruido excesivo, vibración, aumento de temperatura, agarrotamiento del motor, parada del compresor. | Falta o degradación de lubricante, contaminación, fatiga del material, desalineación. | Sensor de vibración simulado, sensor de temperatura simulado en rodamientos, análisis de aceite simulado (conceptual). | 9 | 5 | 3 | 135 |
| Unidad Compresora | Desgaste de Sellos (Pistón/Tornillo) | Fugas internas, reducción de eficiencia, menor presión/caudal de salida, aumento de consumo energético. | Operación prolongada, calidad del fluido, temperatura de operación, contaminación. | Sensor de presión de descarga simulado, sensor de caudal simulado, monitor de eficiencia simulado. | 7 | 7 | 5 | 245 |
|  | Fallo de Válvula (Admisión/Descarga) | Fuga en válvula, rotura de válvula, restricción de flujo. Reducción de eficiencia, sobrepresión, ruidos. | Fatiga del material, corrosión, partículas en el fluido. | Sensores de presión diferencial simulados, análisis de ruido simulado (conceptual). | 8 | 4 | 6 | 192 |
|  | Problemas de Lubricación (Unidad Compresora) | Desgaste acelerado de componentes móviles (rodamientos, engranajes), aumento de fricción y temperatura, fallo prematuro. | Nivel bajo de aceite, aceite contaminado/degradado, fallo de la bomba de aceite, filtro de aceite obstruido. | Sensor de nivel de aceite simulado, sensor de temperatura de aceite simulado, análisis de calidad de aceite simulado (intervalos definidos para cambio/análisis). | 9 | 6 | 4 | 216 |
| Sistema General | Fuga Externa de Fluido Comprimido | Pérdida de eficiencia, activación frecuente del compresor, ruido. | Conexiones flojas, mangueras/tuberías dañadas, sellos defectuosos. | Inspección visual simulada (basada en tiempo o después de alertas de baja presión), sensor de presión del tanque simulado. | 6 | 5 | 7 | 210 |

*Nota: Las puntuaciones de S, O, D y el RPN son ilustrativas y deberían ajustarse basándose en un análisis más detallado para un caso específico.*

Esta tabla es fundamental porque guía directamente el diseño de los escenarios de fallo en la simulación y las condiciones que el agente de IA deberá reconocer y gestionar. Ayuda a enfocar el esfuerzo de simulación en los aspectos más críticos del moto-compresor.

**C. Técnicas de Adquisición, Acondicionamiento y Preprocesamiento de Datos de Sensores para PdM.**

Es importante tener una comprensión conceptual de cómo se obtendrían y prepararían los datos en un sistema de PdM real. Estos conceptos informarán el diseño de los "sensores simulados" y los datos que el agente de IA percibirá en el entorno de simulación.4

El proceso de adquisición de datos (DAQ) en un sistema real implica la selección e instalación de sensores apropiados (vibración, temperatura, presión, etc.) en puntos estratégicos del moto-compresor. Estos sensores convierten magnitudes físicas en señales eléctricas, que luego son digitalizadas por un sistema de adquisición de datos. El DAQ puede variar desde sistemas embebidos de bajo costo (como Arduino o Raspberry Pi con los shields o HATs adecuados) hasta equipos industriales más sofisticados.

Una vez adquiridos, los datos crudos de los sensores raramente están listos para ser utilizados directamente por modelos de IA. Requieren varias etapas de **acondicionamiento y preprocesamiento** 4:

1. **Manejo de Valores Faltantes:** Los datos de los sensores pueden tener interrupciones o lecturas faltantes debido a problemas de comunicación o fallos transitorios del sensor. Estos valores deben ser tratados, por ejemplo, mediante imputación (reemplazándolos con la media, mediana o un valor predicho) o eliminando los registros si la cantidad de datos faltantes es pequeña.
2. **Reducción de Ruido y Filtrado:** Las señales de los sensores a menudo contienen ruido proveniente de interferencias electromagnéticas u otras fuentes. Se aplican técnicas de filtrado (ej. filtros de media móvil, filtros de Kalman, filtros paso bajo/alto) para suavizar la señal y resaltar las tendencias importantes.
3. **Normalización/Escalado:** Muchos algoritmos de ML funcionan mejor cuando las características de entrada están en una escala similar. Técnicas como la normalización Min-Max (escalar los datos a un rango, ej. 0 a 1) o la estandarización (transformar los datos para que tengan media cero y desviación estándar uno) son comunes.
4. **Extracción de Características (Feature Engineering):** A partir de los datos crudos o preprocesados, se pueden derivar nuevas características que sean más informativas para el modelo de IA. Por ejemplo, a partir de una señal de vibración en el dominio del tiempo, se pueden extraer características en el dominio de la frecuencia (usando la Transformada Rápida de Fourier - FFT) como las amplitudes en ciertas bandas de frecuencia, o características estadísticas como la RMS (Root Mean Square), la curtosis o la oblicuidad.
5. **Segmentación y Ventaneo:** Para datos de series temporales, a menudo es útil dividir los datos continuos en segmentos o ventanas de tamaño fijo para el análisis.

La calidad de los datos de entrada, ya sean reales o generados por la simulación, es el factor más crítico para el éxito de cualquier sistema de PdM basado en IA. El principio de "basura entra, basura sale" (Garbage In, Garbage Out - GIGO) es especialmente pertinente aquí. Los modelos de IA aprenden patrones a partir de los datos con los que son alimentados.2 Si estos datos son ruidosos, incompletos, contienen errores o están sesgados, el modelo de IA resultante tomará decisiones subóptimas o incorrectas, independientemente de la sofisticación del algoritmo utilizado. Por lo tanto, un preprocesamiento de datos cuidadoso y riguroso es una etapa esencial e ineludible.4 En el contexto de la simulación, esto implica asegurarse de que los "sensores simulados" generen datos que, aunque simplificados, sean coherentes y representativos de los fenómenos que se están modelando.

**III. Diseño y Configuración del Entorno de Simulación Discreta con SimPy**

**A. Introducción a SimPy: Modelado de Procesos, Recursos y Eventos.**

SimPy es una potente librería de Python diseñada para la simulación de eventos discretos (DES).13 En un DES, el estado del sistema solo cambia en puntos discretos en el tiempo, como resultado de la ocurrencia de "eventos". SimPy permite modelar componentes activos (como clientes, vehículos o, en nuestro caso, el moto-compresor y el agente de IA) como "procesos" que interactúan entre sí y con "recursos" pasivos de capacidad limitada (como servidores, máquinas o, en nuestro caso, operarios de mantenimiento).

Los conceptos fundamentales en SimPy son:

1. **Entorno (simpy.Environment):** Es el contenedor principal de la simulación. Gestiona el tiempo de simulación (env.now) y la cola de eventos programados. Todos los procesos y recursos operan dentro de un entorno.13
2. **Procesos (env.process()):** Representan las entidades activas del modelo. En SimPy, los procesos se definen como funciones generadoras de Python (funciones que utilizan la palabra clave yield). Estas funciones describen la secuencia de acciones y eventos que un agente o componente experimenta a lo largo del tiempo.12
3. **Eventos (env.timeout(), env.event(), etc.):** Son los desencadenantes de cambios de estado en la simulación. Un proceso puede generar un evento y luego "ceder" (yield) el control al entorno hasta que ese evento ocurra. El evento más común es env.timeout(duration), que simula el paso de una cierta cantidad de tiempo.13 Los procesos también son eventos, lo que permite que un proceso espere a que otro termine.
4. **Recursos (simpy.Resource, simpy.PreemptiveResource, simpy.Store, simpy.Container):** Modelan componentes pasivos con capacidad limitada por los cuales los procesos pueden competir. Por ejemplo, un simpy.Resource puede representar un operario de mantenimiento con capacidad 1 (solo puede atender una tarea a la vez). Los procesos solicitan el recurso, lo utilizan durante un tiempo y luego lo liberan.13

SimPy gestiona el tiempo avanzando de un evento programado al siguiente. Cuando un proceso cede un evento, se suspende hasta que ese evento se procesa. Una vez procesado, el proceso se reanuda desde donde se quedó.15 Esta mecánica permite modelar de forma elegante sistemas con actividades concurrentes y dependencias complejas.

La naturaleza basada en procesos de SimPy resulta especialmente adecuada para modelar sistemas como un moto-compresor y su interacción con un sistema de mantenimiento. El moto-compresor en sí puede ser un proceso que opera, se degrada y falla. Los técnicos de mantenimiento pueden ser modelados como recursos compartidos. El agente de IA también será un proceso que observa el estado del moto-compresor y desencadena acciones de mantenimiento. SimPy maneja de forma natural las colas que se forman cuando múltiples procesos (o el mismo proceso repetidamente) solicitan un recurso limitado, así como la sincronización de eventos.13

**B. Modelado Detallado del Moto-Compresor en SimPy: Componentes (motor, compresor), Operación Cíclica y Recursos (energía, operarios).**

Para modelar el moto-compresor, se definirá una clase MotoCompressor en Python. Esta clase encapsulará los atributos y el comportamiento del equipo dentro del entorno de simulación SimPy.

* **Atributos del MotoCompressor:**
  + env: Referencia al entorno SimPy.
  + nombre: Identificador único para la instancia del moto-compresor.
  + estado\_operativo: Variable que indica si está funcionando, parado, en mantenimiento, fallado, etc.
  + health: Variable numérica que representa la salud actual del componente (ver Sección III.C).
  + Parámetros de operación: Como la tasa de producción, consumo de energía por ciclo, etc.
* **Procesos SimPy dentro de MotoCompressor:**
  + proceso\_operacion(self): Un generador que simula el ciclo de trabajo normal del moto-compresor. Esto podría incluir:
    - Fase de arranque: yield self.env.timeout(TIEMPO\_ARRANQUE).
    - Fase de funcionamiento/producción: Un bucle que se repite, donde en cada iteración se simula la producción de una unidad o un periodo de trabajo, consumiendo energía y degradando la salud. yield self.env.timeout(TIEMPO\_CICLO\_OPERACION).
    - Fase de parada: yield self.env.timeout(TIEMPO\_PARADA). Este proceso puede ser interrumpido por eventos de fallo o por decisiones del agente de IA.
  + proceso\_degradacion(self): Un generador que se ejecuta concurrentemente y actualiza el atributo health del moto-compresor basado en el tiempo de operación o el número de ciclos (ver Sección III.C).
  + proceso\_fallo\_estocastico(self): (Opcional) Un generador que simula la ocurrencia de fallos aleatorios basados en una distribución de probabilidad (ej. tiempo hasta el fallo exponencial). Cuando ocurre un fallo, este proceso puede interrumpir proceso\_operacion.
* **Recursos Externos:**
  + **Energía:** Se puede modelar como un simpy.Container del cual el moto-compresor "consume" unidades durante su operación. Si el contenedor se vacía, la operación se detiene.  
    Python  
    # Ejemplo conceptual de recurso de energía  
    energia\_disponible = simpy.Container(env, init=NIVEL\_INICIAL\_ENERGIA, capacity=CAPACIDAD\_MAX\_ENERGIA)  
    # Dentro de proceso\_operacion:  
    # yield energia\_disponible.get(CANTIDAD\_ENERGIA\_POR\_CICLO)
  + **Operarios de Mantenimiento:** Se modelarán como un simpy.Resource o simpy.PreemptiveResource con una capacidad finita (ej. 1 o más operarios). Cuando el agente de IA decide realizar una acción de mantenimiento, el proceso correspondiente deberá solicitar un operario.13 Si las reparaciones pueden ser interrumpidas por tareas más urgentes (aunque en este modelo simplificado probablemente no sea necesario), PreemptiveResource sería la elección. El ejemplo de Machine Shop de la documentación de SimPy 52 proporciona una excelente base conceptual para modelar la interacción de una máquina (el moto-compresor) con un recurso de reparación (el operario).

Modelar explícitamente los recursos de mantenimiento, como los operarios, y su disponibilidad es fundamental para evaluar de manera realista el impacto de las estrategias de PdM. Una predicción de fallo, por muy precisa que sea, resulta ineficaz si no hay personal disponible para llevar a cabo la acción de mantenimiento recomendada en el momento oportuno. El PdM busca optimizar los horarios de mantenimiento 5, y la disponibilidad de los operarios, modelada como recursos en SimPy, afecta directamente la rapidez con la que se puede realizar el mantenimiento. Esto, a su vez, tiene un impacto directo en el tiempo de inactividad del equipo y en los costos operativos generales.13 La simulación permite cuantificar estos efectos y comparar diferentes políticas de asignación de recursos de mantenimiento.

**C. Modelado Simplificado de la Degradación Gradual y Fallos Estocásticos del Moto-Compresor (introducción de una variable de 'salud').**

Para simular el PdM, es esencial modelar cómo el moto-compresor se degrada y eventualmente falla. Se optará por un modelo de degradación simplificado pero representativo.

* **Variable de 'Salud' (health):** Se introducirá un atributo numérico health en la clase MotoCompressor, inicializado a un valor máximo (ej. 100) que representa un estado perfecto. A medida que el moto-compresor opera, este valor disminuirá, simulando el desgaste y la degradación acumulada.52
* **Mecanismos de Degradación:**
  1. **Degradación por Uso:** La variable health disminuirá en una pequeña cantidad después de cada ciclo de operación o por cada unidad de tiempo de funcionamiento. La tasa de esta disminución puede ser constante o depender de la intensidad de la operación. Se pueden explorar modelos lineales simples de degradación, donde la salud disminuye linealmente con el tiempo o el uso.73  
     Python  
     # Dentro de proceso\_operacion, después de un ciclo:  
     # self.health -= TASA\_DEGRADACION\_POR\_USO
  2. **Fallos Estocásticos (Opcional para simplificar):** Se pueden modelar fallos repentinos que no son resultado de una degradación gradual. Esto se puede implementar generando tiempos hasta el fallo basados en distribuciones de probabilidad, como la distribución exponencial para fallos aleatorios o la distribución de Weibull para fallos por desgaste.37 Un evento de fallo estocástico podría causar una disminución drástica en health o llevarla directamente a cero.
* **Umbrales de Salud y Estados del Equipo:** Se definirán diferentes umbrales para la variable health que representarán distintos estados del moto-compresor:
  + health > 70: Estado Normal.
  + 70 >= health > 30: Estado de Advertencia (degradación significativa, posible necesidad de inspección).
  + 30 >= health > 0: Estado Crítico (fallo inminente, requiere acción urgente).
  + health <= 0: Fallo Completo.
* **Impacto de la Salud en la Operación:**
  + Cuando health alcanza el umbral de Fallo Completo, el proceso\_operacion del moto-compresor puede ser interrumpido utilizando self.process\_operacion.interrupt(), similar a cómo se modelan las averías en el ejemplo de Machine Shop de SimPy.52 Una vez interrumpido, el moto-compresor requerirá una acción de mantenimiento (reparación o reemplazo) para restaurar su salud y reanudar la operación.

La variable health es un pilar fundamental en este modelo, ya que actúa como el principal indicador observable del estado del moto-compresor para el agente de IA. Aunque es una simplificación, su modelado debe capturar la esencia de cómo un equipo se deteriora con el tiempo y el uso. Esto es crucial para que las decisiones tomadas por el agente de IA, basadas en la observación de esta variable (o de otras variables simuladas que se correlacionen con ella, como aumentos en la temperatura o vibración simuladas a medida que la salud disminuye), sean significativas y relevantes para el objetivo del PdM.2

**D. Configuración del Entorno de Desarrollo Python: Instalación y uso de SimPy, Pandas, NumPy y Scikit-learn.**

Una configuración adecuada del entorno de desarrollo es crucial para la eficiencia y reproducibilidad del proyecto, especialmente bajo un cronograma.

1. **Entorno Virtual Python:** Se recomienda encarecidamente la creación de un entorno virtual aislado para el proyecto. Esto evita conflictos entre las dependencias del proyecto y las de otros proyectos o las del sistema global de Python.
   * Se puede utilizar venv (incluido en Python) o conda (si se instala Anaconda o Miniconda).
   * Ejemplo con venv:  
     Bash  
     python -m venv mi\_entorno\_tfg  
     source mi\_entorno\_tfg/bin/activate # En Linux/macOS  
     # mi\_entorno\_tfg\Scripts\activate # En Windows
2. **Instalación de Librerías:** Una vez activado el entorno virtual, se instalarán las librerías necesarias utilizando pip:  
   Bash  
   pip install simpy  
   pip install pandas  
   pip install numpy  
   pip install scikit-learn  
   pip install matplotlib  
   pip install seaborn  
   pip install jupyterlab # Para un desarrollo interactivo y visualización  
     
   Se hará referencia a las guías de instalación y configuración.83 El uso de Anaconda o Miniconda 85 puede simplificar significativamente la gestión de estos paquetes y sus dependencias, especialmente para usuarios menos experimentados con la gestión manual de entornos Python.
3. **Gestor de Paquetes:** Se explicará brevemente cómo usar pip para gestionar las librerías (instalar, actualizar, listar, desinstalar).
4. **IDE (Entorno de Desarrollo Integrado):** Se recomendará el uso de un IDE como Visual Studio Code, PyCharm, o Spyder (que viene con Anaconda) para facilitar la escritura, depuración y ejecución del código.

La creación de un entorno de desarrollo bien configurado y aislado no es un paso trivial; es una práctica fundamental que previene una multitud de problemas relacionados con conflictos de versiones de librerías o dependencias. En un proyecto con un plazo tan ajustado, perder tiempo solucionando problemas de entorno es un lujo que no se puede permitir. Los entornos virtuales 85 aseguran que el proyecto tenga su propio conjunto de dependencias, lo que facilita la colaboración (si aplica) y la replicación del entorno en diferentes máquinas, garantizando la reproducibilidad de los resultados, un aspecto clave en cualquier trabajo de investigación o desarrollo.

**IV. Desarrollo e Integración del Agente de Inteligencia Artificial para PdM**

**A. Selección y Justificación del Tipo de Agente de IA: Basado en Reglas vs. Aprendizaje Automático Simple.**

La elección del tipo de agente de IA es una decisión crítica que debe equilibrar la sofisticación deseada con la viabilidad de implementación en un tiempo prudente.

* **Agentes Basados en Reglas (Agentes Reflejo Simples):**
  + Estos agentes operan según un conjunto predefinido de reglas de tipo "SI (condición) ENTONCES (acción)".2
  + Las decisiones se toman basándose únicamente en la percepción actual del entorno simulado (ej. el valor actual de la variable health del moto-compresor).
  + **Ventajas:** Son conceptualmente más simples, más rápidos de diseñar e implementar, no requieren grandes cantidades de datos históricos para entrenamiento y su lógica de decisión es transparente y fácil de entender.
  + **Desventajas:** Pueden volverse difíciles de gestionar si el número de reglas es muy grande y no se adaptan a situaciones no previstas explícitamente en las reglas.
* **Agentes de Aprendizaje (Machine Learning Simple):**
  + Estos agentes pueden aprender de los datos (ya sean históricos o generados por la simulación) para mejorar su rendimiento en la toma de decisiones con el tiempo.10
  + Para un proyecto corto, se consideraría un modelo de ML supervisado simple, como la regresión logística para predecir la probabilidad de fallo o un árbol de decisión para clasificar el estado de salud.
  + **Ventajas:** Potencial para descubrir patrones no obvios en los datos y adaptarse a nuevas situaciones si se reentrenan.
  + **Desventajas:** Requieren datos para el entrenamiento (que podrían necesitar ser generados por la simulación), el proceso de entrenamiento y validación consume tiempo, y la interpretabilidad del modelo puede ser menor que con un sistema basado en reglas.

Justificación:

Se recomienda encarecidamente comenzar con un agente basado en reglas simples. Este enfoque permite centrarse en la correcta integración del agente con el entorno de simulación SimPy y en la lógica fundamental del PdM. Si el desarrollo avanza bien y el tiempo lo permite, se podría explorar la implementación de un modelo de ML supervisado muy simple como una extensión o comparación.

La complejidad inherente al Aprendizaje por Refuerzo (RL), que a menudo se cita en la literatura de PdM 1, generalmente lo hace inviable para un proyecto de esta duración y alcance limitado. El RL exige una formulación cuidadosa del espacio de estados, el espacio de acciones, la función de recompensa, y un proceso de entrenamiento que puede ser computacionalmente intensivo y requerir una gran cantidad de interacciones con el entorno (simulado o real).1 Un enfoque pragmático, centrado en demostrar los conceptos básicos de la toma de decisiones inteligente en un contexto de PdM. Los agentes basados en reglas 80 ofrecen una vía más directa y manejable para lograr este objetivo dentro del plazo establecido.

**B. Diseño del Agente de IA:**

Independientemente de si el agente es basado en reglas o en ML simple, su diseño conceptual sigue una estructura similar de percepción, toma de decisiones y acción.

* **Percepción:**
  + El agente de IA necesita "observar" el estado actual del moto-compresor simulado. En el entorno SimPy, esto no implica sensores físicos, sino el acceso programático a los atributos relevantes del objeto MotoCompressor.
  + Variables clave a percibir:
    - moto\_compresor.health: El nivel de salud actual.
    - moto\_compresor.tiempo\_operacion\_desde\_ultimo\_mantenimiento: Horas de funcionamiento.
    - moto\_compresor.estado\_operativo: (ej. funcionando, parado, fallado).
    - (Opcional, si se simulan) moto\_compresor.temperatura\_simulada, moto\_compresor.vibracion\_simulada.
  + Dentro del proceso SimPy del agente, se accederá a estos atributos directamente: salud\_actual = moto\_compresor.health. Se debe tener en cuenta que un proceso SimPy puede leer atributos de otro objeto de proceso al que tiene referencia, siendo esto un acceso estándar de atributos de objetos Python.15
* **Toma de Decisiones:**
  + **Agente Basado en Reglas:** La lógica de decisión se implementará mediante una serie de sentencias condicionales (if-elif-else) en Python dentro del proceso SimPy del agente.
    - *Ejemplo de Regla 1 (Inspección Preventiva):*  
      Python  
      if moto\_compresor.health < UMBRAL\_SALUD\_INSPECCION and not moto\_compresor.mantenimiento\_programado:  
       # Decisión: Programar inspección  
       yield env.process(programar\_mantenimiento(moto\_compresor, 'inspeccion', operarios))
    - *Ejemplo de Regla 2 (Reparación por Umbral Crítico):*  
      Python  
      elif moto\_compresor.health < UMBRAL\_SALUD\_CRITICA and not moto\_compresor.mantenimiento\_programado:  
       # Decisión: Programar reparación urgente  
       yield env.process(programar\_mantenimiento(moto\_compresor, 'reparacion\_critica', operarios))
    - *Ejemplo de Regla 3 (No Hacer Nada):*  
      Python  
      else:  
       # Decisión: No hacer nada, continuar monitoreando  
       yield env.timeout(INTERVALO\_MONITOREO)
  + **(Opcional) Agente Basado en ML Simple:** Si se implementa, el agente llamará a una función Python externa. Esta función cargará un modelo de scikit-learn previamente entrenado (ej. un árbol de decisión que clasifica el estado como 'OK', 'Advertencia', 'Fallo\_Inminente') y utilizará su método predict() o predict\_proba().97 La entrada al modelo serían los datos de percepción (ej. [moto\_compresor.health, moto\_compresor.temperatura\_simulada]). La salida del modelo (la clase predicha o las probabilidades) se usaría para tomar la decisión de mantenimiento.
* **Acciones:**
  + Las acciones del agente se traducirán en la generación de nuevos eventos SimPy o en la solicitud/liberación de recursos SimPy.
  + Ejemplos de acciones:
    - **Programar Mantenimiento:** Esto podría implicar crear un nuevo proceso SimPy para la tarea de mantenimiento. Este proceso solicitaría un recurso operario.  
      Python  
      def programar\_mantenimiento(env, maquina, tipo\_mantenimiento, operarios\_resource):  
       print(f"T={env.now:.2f}: {maquina.nombre} - Solicitando operario para {tipo\_mantenimiento}")  
       with operarios\_resource.request() as req:  
       yield req  
       print(f"T={env.now:.2f}: {maquina.nombre} - Operario asignado para {tipo\_mantenimiento}")  
       if tipo\_mantenimiento == 'inspeccion':  
       yield env.timeout(TIEMPO\_INSPECCION)  
       maquina.health = min(100, maquina.health + MEJORA\_SALUD\_INSPECCION) # Simular mejora  
       elif tipo\_mantenimiento == 'reparacion\_critica':  
       yield env.timeout(TIEMPO\_REPARACION\_CRITICA)  
       maquina.health = SALUD\_DESPUES\_REPARACION # Restaurar salud  
       print(f"T={env.now:.2f}: {maquina.nombre} - Mantenimiento ({tipo\_mantenimiento}) completado. Salud: {maquina.health}")  
       maquina.mantenimiento\_programado = False
    - **Detener/Iniciar el Moto-Compresor:** Si el moto-compresor tiene un proceso de operación, el agente podría interrumpirlo o permitir que se reanude.
    - **Registrar Decisión:** print(f"T={env.now:.2f}: Agente IA decide {accion} para {moto\_compresor.nombre}").

La "percepción" del agente en un entorno SimPy se materializa mediante el acceso programático a las variables de estado de los objetos simulados. Las "acciones" del agente se manifiestan como la generación de nuevos eventos SimPy, la modificación de atributos de objetos simulados (como el estado de salud después de una reparación), o la solicitud y liberación de recursos SimPy (como un operario de mantenimiento). Esta interacción se rige por las primitivas y la lógica del framework de simulación.12

**C. Integración del Agente de IA con la Simulación SimPy: Llamada a funciones Python y modelos ML desde procesos SimPy.**

El agente de IA se implementará como un proceso SimPy, típicamente una función generadora de Python.

Python

def agente\_ia\_proceso(env, moto\_compresor, operarios\_resource, umbral\_salud\_inspeccion, umbral\_salud\_critica, intervalo\_monitoreo):  
 """Proceso SimPy que representa el agente de IA para PdM."""  
 while True:  
 # 1. Percepción del estado del moto-compresor  
 salud\_actual = moto\_compresor.health  
 estado\_op = moto\_compresor.estado\_operativo  
 print(f"T={env.now:.2f}: Agente IA - Monitoreando {moto\_compresor.nombre}. Salud: {salud\_actual}, Estado: {estado\_op}")  
  
 # 2. Toma de Decisiones (Lógica basada en reglas)  
 accion\_tomada = False  
 if estado\_op == "FALLADO" and not moto\_compresor.mantenimiento\_programado:  
 print(f"T={env.now:.2f}: Agente IA - {moto\_compresor.nombre} está FALLADO. Programando reparación urgente.")  
 env.process(programar\_mantenimiento(env, moto\_compresor, 'reparacion\_critica', operarios\_resource))  
 moto\_compresor.mantenimiento\_programado = True  
 accion\_tomada = True  
 elif salud\_actual < umbral\_salud\_critica and estado\_op == "FUNCIONANDO" and not moto\_compresor.mantenimiento\_programado:  
 print(f"T={env.now:.2f}: Agente IA - Salud de {moto\_compresor.nombre} ({salud\_actual}) es CRÍTICA. Programando parada y reparación.")  
 # Primero detener la máquina si es necesario (podría ser una interrupción)  
 if hasattr(moto\_compresor, 'proceso\_op\_activo') and moto\_compresor.proceso\_op\_activo.is\_alive:  
 moto\_compresor.proceso\_op\_activo.interrupt("PARADA\_POR\_MANTENIMIENTO\_CRITICO")  
 env.process(programar\_mantenimiento(env, moto\_compresor, 'reparacion\_critica', operarios\_resource))  
 moto\_compresor.mantenimiento\_programado = True  
 accion\_tomada = True  
 elif salud\_actual < umbral\_salud\_inspeccion and estado\_op == "FUNCIONANDO" and not moto\_compresor.mantenimiento\_programado:  
 print(f"T={env.now:.2f}: Agente IA - Salud de {moto\_compresor.nombre} ({salud\_actual}) requiere inspección. Programando inspección.")  
 env.process(programar\_mantenimiento(env, moto\_compresor, 'inspeccion', operarios\_resource))  
 moto\_compresor.mantenimiento\_programado = True  
 accion\_tomada = True  
   
 # Si se tomó una acción que implica esperar a un operario, el agente podría esperar aquí  
 # o simplemente re-evaluar en el próximo intervalo\_monitoreo.  
 # Por simplicidad, aquí re-evaluará después del intervalo.  
  
 # 3. Esperar para el próximo ciclo de monitoreo  
 yield env.timeout(intervalo\_monitoreo)

Dentro de este proceso SimPy, las llamadas a funciones Python estándar (como las que contendrían la lógica de reglas o la invocación de un modelo ML con modelo.predict()) se ejecutan de forma síncrona dentro del "paso de tiempo" del agente, a menos que esas funciones mismas contengan yield de eventos SimPy.71 Para modelos de ML simples y ligeros, la predicción suele ser lo suficientemente rápida como para no requerir una gestión de concurrencia compleja dentro de SimPy. Si la toma de decisiones de IA fuera un proceso computacionalmente muy intensivo o implicara I/O bloqueante (como una llamada a un servicio web externo), se modelaría como un proceso SimPy separado que el agente principal podría yield env.process(...) para esperar su resultado.

La clave de la integración radica en que el proceso SimPy del agente puede pausar su propia ejecución (usando yield) mientras espera el paso del tiempo simulado (para monitoreos periódicos) o la disponibilidad de recursos (como un operario), y luego reanudar su ejecución para percibir el estado actual, tomar una decisión y actuar sobre el entorno simulado. Los procesos SimPy son generadores 69, y esta característica es la que permite su suspensión y reanudación controlada por el motor de simulación.

**D. Implementación de un Modelo de ML Supervisado Simple (ej. Regresión Lineal para RUL o Árbol de Decisión para clasificación de estado de salud/fallo).**

Si se decide incluir un componente de ML, el enfoque sería:

1. **Generación/Obtención de Datos:**
   * Idealmente, se utilizarían datos generados por múltiples ejecuciones de la simulación SimPy bajo diferentes condiciones, registrando los valores de los "sensores simulados" (ej. health, temperatura\_simulada) y el estado final o tiempo hasta el fallo.
   * Alternativamente, se podría adaptar un dataset público de mantenimiento predictivo.128
2. **Preparación de Datos:**
   * Limpieza, preprocesamiento y extracción de características de los datos generados o seleccionados, tal como se discutió en la Sección II.C.
   * Creación de etiquetas para el aprendizaje supervisado (ej. 'fallará\_pronto' (binario), 'estado\_salud' (categórico), o 'RUL\_real' (numérico)).
3. **Entrenamiento del Modelo con Scikit-learn:**
   * Seleccionar un modelo simple:
     + Para clasificación (ej. predecir si la máquina fallará en las próximas X horas): LogisticRegression o DecisionTreeClassifier.
     + Para regresión (ej. predecir RUL): LinearRegression o DecisionTreeRegressor.
   * Dividir los datos en conjuntos de entrenamiento y prueba.
   * Entrenar el modelo utilizando el conjunto de entrenamiento: model.fit(X\_train, y\_train).
   * Se puede consultar 97 para ejemplos y guías.
4. **Evaluación del Modelo:**
   * Evaluar el rendimiento del modelo en el conjunto de prueba utilizando métricas apropiadas (ej. accuracy, precision, recall, F1-score para clasificación; MSE, R² para regresión).
   * Un ejemplo de código Python para un modelo simple se proporcionará en la sección de implementación.104
5. **Guardar y Cargar el Modelo:**
   * Una vez entrenado y evaluado, el modelo se guardará en un archivo (ej. usando joblib o pickle).
   * La simulación SimPy cargará este modelo pre-entrenado al inicio para ser utilizado por el agente de IA.99

El objetivo no es desarrollar un modelo de ML de última generación, sino demostrar la capacidad de integrar un modelo funcional, aunque sea simple, dentro del ciclo de PdM simulado. El entrenamiento y ajuste fino de modelos de ML complejos consume una cantidad significativa de tiempo y recursos.101 Por ello, un modelo simple como la Regresión Logística o un Árbol de Decisión 102 es mucho más manejable y permite al estudiante centrarse en la integración con SimPy y en la lógica general del sistema de PdM.

**B. Priorización Estratégica de Funcionalidades y Tareas con el Método MoSCoW.**

No todas las ideas o funcionalidades concebibles podrán ser implementadas. El método MoSCoW es una técnica de priorización que ayuda a clasificar los requisitos en cuatro categorías, facilitando la toma de decisiones sobre qué incluir y qué excluir 21:

* **M - Must have (Debe tener):** Requisitos absolutamente esenciales. El trabajo se consideraría un fracaso si no se cumplen. Estos son no negociables y deben ser el foco principal.
  + *Ejemplo:* Modelo SimPy básico del moto-compresor operativo, variable de salud funcional, agente de IA basado en reglas simples que toma al menos una decisión de mantenimiento, informe final con estructura mínima requerida.
* **S - Should have (Debería tener):** Requisitos importantes que añaden un valor significativo, pero aún podría ser aceptable sin ellos si el tiempo apremia.
  + *Ejemplo:* Múltiples modos de fallo simulados, agente de IA con un conjunto más rico de reglas, análisis de sensibilidad de algunos parámetros de simulación, inclusión de gráficos de resultados detallados.
* **C - Could have (Podría tener):** Requisitos deseables que se implementarán si el tiempo y los recursos lo permiten, una vez completados los "Must" y "Should". Tienen un impacto menor si se omiten.
  + *Ejemplo:* Una interfaz gráfica simple para la simulación, exploración de un modelo de ML básico como alternativa al agente basado en reglas, análisis estadístico más profundo de los resultados.
* **W - Won't have (No tendrá esta vez):** Requisitos que se acuerda explícitamente no incluir en esta iteración. Esto ayuda a gestionar las expectativas y evitar la "expansión del alcance" (scope creep).
  + *Ejemplo:* Implementación de un agente de Aprendizaje por Refuerzo complejo, conexión con hardware físico real, desarrollo de una interfaz de usuario sofisticada.

El método MoSCoW actúa como una herramienta de "triaje" indispensable cuando el tiempo es el recurso más escaso. Obliga a una reflexión crítica sobre lo que es verdaderamente esencial para el éxito del proyecto frente a lo que es simplemente deseable.21 Sin esta priorización explícita, es fácil caer en la trampa de intentar abarcar demasiado, lo que a menudo resulta en no completar nada de manera sustancial o en entregar un trabajo de baja calidad.21 La siguiente tabla ejemplifica cómo se podría aplicar MoSCoW a las funcionalidades de este proyecto:

**Tabla 2: Priorización MoSCoW para Funcionalidades del Proyecto de Simulación de PdM**

|  |  |  |  |
| --- | --- | --- | --- |
| **Funcionalidad/Tarea** | **Descripción** | **Categoría MoSCoW** | **Justificación** |
| **Modelo de Simulación Base del Moto-Compresor (SimPy)** | Crear la clase MotoCompressor con operación básica (arranque, funcionamiento, parada) y consumo de recursos (ej. energía conceptual). | **Must have** | Fundamental para cualquier simulación de PdM. Sin esto, no hay sistema que mantener. |
| **Modelo de Degradación Simplificado (Variable de 'Salud')** | Implementar el atributo health y su disminución con el uso/tiempo. Definir umbrales de salud. | **Must have** | Esencial para que el PdM tenga sentido; el agente necesita un indicador de condición. |
| **Agente de IA Basado en Reglas Simples (Percepción y Decisión)** | Agente que lee health y otros estados simulados, y aplica al menos 2-3 reglas para decidir acciones de mantenimiento. | **Must have** | Demuestra la capacidad de toma de decisiones del PdM. |
| **Integración del Agente con Simulación (Acciones del Agente)** | El agente puede solicitar recursos de mantenimiento (operarios) y simular la ejecución de acciones de mantenimiento (ej. inspección). | **Must have** | Cierra el ciclo de PdM: percepción -> decisión -> acción dentro de la simulación. |
| **Recolección de Datos de Simulación Básica** | Registrar en un log o CSV los eventos clave: cambios de salud, decisiones del agente, tiempos de mantenimiento, fallos. | **Must have** | Necesario para el análisis y la evaluación del rendimiento del sistema simulado. |
| **Informe Final (Estructura y Contenido Mínimo)** | Redacción de las secciones principales del informe según las directrices académicas. | **Must have** | Entregable principal. |
| **Modelado de Múltiples Modos de Fallo (basado en FMEA)** | Simular al menos 2-3 modos de fallo distintos identificados en el FMEA (ej. fallo por desgaste, fallo eléctrico). | **Should have** | Aumenta el realismo de la simulación y permite probar la robustez del agente de IA. |
| **Implementación de un Modelo de ML Supervisado Simple (Alternativa/Extensión)** | Entrenar un árbol de decisión o regresión logística para predecir fallos o RUL, usando datos de la simulación. | **Should have** | Demuestra una comprensión más profunda de las técnicas de IA aplicables al PdM. |
| **Análisis de Sensibilidad de Parámetros Clave de Simulación** | Variar parámetros como la tasa de degradación o el tiempo de reparación y observar el impacto en el rendimiento del sistema. | **Should have** | Proporciona insights sobre la robustez del modelo y los factores más influyentes. |
| **Visualización Avanzada de Resultados (Matplotlib/Seaborn)** | Crear gráficos detallados del comportamiento de la salud, decisiones del agente, utilización de recursos, etc. | **Could have** | Mejora la presentación y comprensión de los resultados, pero no es esencial para la funcionalidad básica. |
| **Desarrollo de una Interfaz Gráfica Simple para la Simulación** | Una interfaz básica para iniciar la simulación y ver algunos resultados en tiempo real (ej. con Tkinter o PySimpleGUI). | **Could have** | Mejora la usabilidad y demostración, pero consume mucho tiempo de desarrollo. |
| **Conceptualización Detallada del Laboratorio Piloto Físico (Sección VI Completa)** | Investigación exhaustiva de componentes, costos y diseño detallado del montaje físico. | **Could have** | Enriquece el Trabajo, pero la simulación es el núcleo. |
| **Implementación de Agente de Aprendizaje por Refuerzo (RL)** | Diseñar y entrenar un agente de RL para optimizar las políticas de mantenimiento. | **Won't have** | Demasiado complejo y consume mucho tiempo; el entrenamiento de RL es un proyecto en sí mismo. |
| **Conexión con Hardware Real (DAQ)** | Integrar la simulación con sensores y un moto-compresor físico. | **Won't have** | Excede el alcance de un proyecto de simulación corto; introduce complejidades de hardware e interfaz. |

**C. Definición de un Producto Mínimo Viable (MVP) y un Proceso Mínimo Viable para el Proyecto.**

* **Producto Mínimo Viable (MVP):**
  + El MVP se refiere al conjunto de características mínimas necesarias para que el producto (en este caso, el sistema de simulación de PdM) funcione en su forma más básica y pueda ser probado o demostrado.19
  + Para este trabajo, el MVP podría ser:
    1. Un modelo SimPy del moto-compresor que simule su operación y un único modo de degradación lineal de la variable health.
    2. Un agente de IA basado en reglas muy simples (ej. 2-3 reglas) que monitoree la health y pueda desencadenar una acción de "solicitar mantenimiento" cuando la salud caiga por debajo de un umbral.
    3. La simulación de la acción de mantenimiento (ej. un operario repara la máquina, restaurando su health).
    4. Registro básico de eventos y resultados.
  + El objetivo del MVP es tener un ciclo completo de "operación -> degradación -> detección por IA -> decisión de mantenimiento -> ejecución de mantenimiento simulado" funcionando, aunque sea de forma muy simplificada. Los principios para un MVP de productos físicos/mecatrónicos, como "haz una cosa bien" y "piensa a menor escala", son directamente aplicables aquí.20
* **Proceso Mínimo Viable:**
  + Este concepto, derivado de DevSecOps 19, se refiere a establecer un flujo de trabajo funcional para el desarrollo del proyecto desde el inicio.
  + Para el trabajo, esto implicaría definir y poner en marcha:
    1. Un entorno de desarrollo Python configurado con las librerías necesarias.
    2. Un sistema de control de versiones (ej. Git) para el código.
    3. Un plan de sprints o un tablero Kanban inicial.
    4. Un esqueleto básico del código de simulación y del agente.
    5. Un formato básico para la documentación y el informe.
  + Establecer este proceso mínimo viable al principio ayuda a estructurar el trabajo, identificar tempranamente problemas de configuración o de flujo de trabajo, y asegurar una base sólida sobre la cual construir el MVP.

El MVP asegura que se entregue un resultado funcional y demostrable dentro del plazo prudente, evitando el riesgo de tener un proyecto grande e inacabado.19 Al enfocarse en las características esenciales, el estudiante puede demostrar su comprensión de los conceptos clave y las técnicas involucradas.

**VI. Conceptualización y Montaje de un Laboratorio Piloto de Bajo Costo (Opcional/Guía)**

Aunque el núcleo de este proyecto es la simulación, la conceptualización de un laboratorio piloto físico de bajo costo puede enriquecer significativamente el trabajo. Proporciona una conexión tangible con el mundo real, fundamenta los parámetros utilizados en la simulación y puede sugerir futuras líneas de investigación o desarrollo práctico. Esta sección es opcional o puede ser una guía conceptual, dependiendo del tiempo y los recursos disponibles.

**A. Selección de Componentes Esenciales: Moto-compresor de pequeña escala, Sensores (ADXL345 para vibración, termistor para temperatura), Sistema de Adquisición de Datos (DAQ) (Arduino/Raspberry Pi).**

La creación de un banco de pruebas para PdM no tiene por qué ser prohibitivamente costosa, especialmente a nivel de laboratorio o para proyectos estudiantiles.

* **Moto-compresor de Pequeña Escala:**
  + Se pueden considerar motores DC de bajo costo, comúnmente utilizados en aplicaciones de hobby o educativas. Estos motores, aunque simples, pueden exhibir modos de fallo como desgaste de escobillas, problemas en el conmutador o en el bobinado, especialmente si se operan fuera de sus condiciones nominales.110 Un compresor de aire pequeño, como los utilizados para inflar neumáticos o en aerografía, acoplado a uno de estos motores, podría servir como sistema completo.
  + La elección debe basarse en la disponibilidad, el costo y la facilidad para instrumentar con sensores.
* **Sensores:**
  + **Vibración:** El acelerómetro ADXL345 es una opción popular y de bajo costo. Puede interconectarse fácilmente con microcontroladores como Raspberry Pi 111 o ESP32.111 Este sensor triaxial permite capturar datos de vibración que pueden ser analizados para detectar desequilibrios, holguras o problemas en rodamientos.
  + **Temperatura:** Los termistores son sensores de temperatura muy económicos y fáciles de usar con plataformas como Arduino 111 o ESP32.111 El LM35 es otro sensor de temperatura analógico de bajo costo y buena precisión.111 Monitorear la temperatura del motor y de los cabezales del compresor es crucial.
  + **Corriente (Opcional):** Sensores como el INA219 pueden medir la corriente consumida por el motor, lo que puede ser un indicador de esfuerzo mecánico o problemas eléctricos.111
  + Otros sensores relevantes para compresores industriales incluyen sensores de presión y de calidad del aceite 6, aunque su implementación en un piloto de muy bajo costo puede ser más desafiante.
* **Sistema de Adquisición de Datos (DAQ):**
  + **Plataformas Basadas en Microcontroladores:** Arduino (Uno, Nano) o Raspberry Pi (Pi Zero, Pi 3/4) son opciones excelentes y de bajo costo para la adquisición de datos de los sensores mencionados.47 Ofrecen pines de entrada analógica y digital, y soportan interfaces de comunicación como I2C y SPI, comunes en muchos sensores. El ESP32 es otra alternativa potente con Wi-Fi y Bluetooth integrados.111
  + Un estudio detalla la construcción de un DAQ de muy bajo costo utilizando un microcontrolador AVR Atmega8, con interfaz a Matlab o LabView, demostrando la viabilidad de estas soluciones para estudiantes.49
  + **Software DAQ:** Para Raspberry Pi, se puede usar Python con librerías como smbus para I2C y RPi.GPIO para E/S generales. Para Arduino, el IDE de Arduino es el estándar. Los datos pueden ser procesados localmente o enviados a un PC para un análisis más profundo.

Comprender los componentes físicos de un sistema de PdM real, incluso si el trabajo se centra en la simulación, enriquece enormemente el proyecto. Ayuda a fundamentar la elección de parámetros en la simulación (ej. tasas de fallo, umbrales de sensores simulados) y a comprender las limitaciones y desafíos de la implementación práctica. Conocer los sensores y DAQ reales 17 permite definir qué tipo de datos se simularán y cómo se podrían obtener y procesar estos datos en un escenario real, haciendo la simulación más relevante.

**Tabla 3: Componentes y Especificaciones para Laboratorio Piloto Conceptual de Bajo Costo**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Componente** | **Modelo/Tipo Sugerido** | **Especificaciones Clave** | **Rango de Costo Estimado (USD)** | **Consideraciones de Integración** |
| **Moto-compresor** | Motor DC 12V genérico + Mini compresor de aire 12V | Motor: 6-24V, bajo torque. Compresor: pequeño caudal, presión máx. ~100-150 PSI. | 20 - 50 | Acoplamiento mecánico simple. Fuente de alimentación DC regulable. |
| **Sensor de Vibración** | Acelerómetro ADXL345 (módulo) | 3 ejes, ±16g, interfaz I2C/SPI, bajo consumo. | 5 - 15 | Conexión a Raspberry Pi (I2C) o Arduino (I2C/SPI). Requiere montaje firme en el motor/compresor. |
| **Sensor de Temperatura** | Termistor NTC 10K con resistencia de 10K | Rango -55°C a +125°C (variable), precisión dependiente de la calibración y ADC. | < 5 | Conexión a entrada analógica de Arduino/ESP32. Requiere circuito divisor de tensión. |
|  | Sensor digital DS18B20 | Interfaz 1-Wire, precisión ±0.5°C, rango -55°C a +125°C. | 5 - 10 | Conexión a pin digital de Raspberry Pi/Arduino. Requiere librería específica. Más robusto a interferencias que el termistor. |
| **Sensor de Corriente** | Módulo INA219 | Mide voltaje y corriente shunt, interfaz I2C, hasta ±3.2A (configurable), 26V. | 5 - 15 | Conexión a Raspberry Pi/Arduino (I2C). Se intercala en la línea de alimentación del motor. |
| **DAQ y Procesamiento** | Raspberry Pi 4 (o similar) / Arduino Uno (o ESP32) | RPi: CPU ARM, RAM, GPIO, Wi-Fi, Ethernet. Arduino: Microcontrolador ATmega328P, E/S digitales y analógicas. ESP32: Wi-Fi/BT. | 15 - 60 (RPi sin acc.) | RPi para procesamiento más complejo con Python. Arduino/ESP32 para adquisición más simple y envío de datos. Programación en Python (RPi) o C/C++ (Arduino). |
| **Software** | Python (RPi), IDE Arduino (Arduino), Librerías (pandas, numpy, matplotlib, SimPy, scikit-learn en PC/RPi) | Open source. | 0 | Configuración de entorno virtual en PC/RPi. |

**B. Guía para la Configuración del Hardware y Software para la Adquisición y Registro de Datos (ej. a CSV).**

* **Conexión del ADXL345 a Raspberry Pi:**
  + Se utilizará la interfaz I2C. Los pines SDA y SCL del ADXL345 se conectan a los pines SDA (GPIO2) y SCL (GPIO3) de la Raspberry Pi, respectivamente. También se conectan VCC (a 3.3V) y GND.112
  + En la Raspberry Pi, se debe habilitar la interfaz I2C mediante raspi-config.
  + Se instalarán las librerías de Python necesarias: sudo apt-get install python3-smbus i2c-tools.
  + Se utilizará un script de Python para leer los datos de los ejes X, Y, Z del acelerómetro.
* **Conexión del Termistor (NTC 10K) a Arduino:**
  + El termistor se conecta en un circuito divisor de tensión con una resistencia de valor conocido (ej. 10KΩ). Un extremo del termistor a 5V (o 3.3V), el otro al pin analógico (ej. A0) y también a la resistencia. El otro extremo de la resistencia a GND.114
  + El código de Arduino leerá el valor analógico del pin A0 y aplicará la ecuación de Steinhart-Hart (o una simplificación) para convertir la lectura de resistencia a temperatura.
* **Registro de Datos a CSV con Python en Raspberry Pi:**
  + Se importará la librería csv y datetime en Python.
  + Se abrirá (o creará si no existe) un archivo CSV en modo apéndice ('a').
  + En un bucle, se leerán los datos del sensor (ej. ADXL345 y/o datos de temperatura recibidos del Arduino vía serial o red).
  + Se obtendrá la fecha y hora actual: timestamp = datetime.datetime.now().strftime("%Y-%m-%d %H:%M:%S").
  + Se escribirán el timestamp y los datos de los sensores como una nueva fila en el archivo CSV.118
  + Se incluirá un pequeño retardo (time.sleep()) para controlar la frecuencia de muestreo.
  + Ejemplo conceptual para registrar datos de vibración (X, Y, Z) y temperatura:  
    Python  
    import csv  
    import datetime  
    import time  
    # Suponiendo funciones get\_vibration\_data() y get\_temperature\_data()  
      
    csv\_file = "sensor\_data.csv"  
    # Escribir encabezado si el archivo es nuevo  
    try:  
     with open(csv\_file, 'r') as f\_check:  
     pass # Archivo existe  
    except FileNotFoundError:  
     with open(csv\_file, 'w', newline='') as f:  
     writer = csv.writer(f)  
     writer.writerow()  
      
    while True:  
     timestamp = datetime.datetime.now().strftime("%Y-%m-%d %H:%M:%S.%f")[:-3] # Incluye milisegundos  
     # vib\_x, vib\_y, vib\_z = get\_vibration\_data() # Función para leer ADXL345  
     # temp\_c = get\_temperature\_data() # Función para leer termistor  
     vib\_x, vib\_y, vib\_z, temp\_c = random.random(), random.random(), random.random(), 25 + random.random()\*5 # Datos simulados para ejemplo  
      
     with open(csv\_file, 'a', newline='') as f:  
     writer = csv.writer(f)  
     writer.writerow([timestamp, vib\_x, vib\_y, vib\_z, temp\_c])  
      
     print(f"Datos guardados: {timestamp}, X:{vib\_x:.2f}, Y:{vib\_y:.2f}, Z:{vib\_z:.2f}, Temp:{temp\_c:.2f}")  
     time.sleep(1) # Muestrear cada segundo

La capacidad de registrar datos de manera estructurada, como en un archivo CSV, es el primer paso esencial para cualquier análisis posterior o para el entrenamiento de modelos de Machine Learning.118 Los archivos CSV son un formato universal que puede ser fácilmente importado y procesado por librerías de Python como Pandas, facilitando la exploración de datos, la visualización y la preparación para el modelado.

**C. Consideraciones para la Recolección de Datos Iniciales y Análisis Preliminar.**

Antes de poder realizar un mantenimiento predictivo efectivo, es crucial entender el comportamiento normal del equipo y establecer líneas base.

* **Análisis de Criticidad de Activos (Conceptual):** Aunque con un solo moto-compresor simulado o piloto esto es directo, en un entorno industrial se comenzaría por un análisis de criticidad para identificar qué activos se beneficiarían más del PdM.48
* **Establecimiento de Líneas Base:**
  + Operar el moto-compresor (real o simulado en su estado "saludable") bajo condiciones normales y registrar los datos de los sensores durante un período representativo. Esto ayuda a definir el rango de valores "normales" para cada parámetro monitoreado.29
  + Estos datos de línea base son fundamentales para la detección de anomalías; cualquier desviación significativa de estos patrones normales puede ser una indicación temprana de un problema.
* **Programa Piloto (Conceptual):**
  + En un contexto industrial, se lanzaría un programa piloto en los activos más críticos. Este piloto se mejora continuamente basándose en los datos recopilados y la retroalimentación operativa.48
  + La "fase piloto" es el propio proyecto de simulación o el montaje del laboratorio. El objetivo es demostrar la viabilidad del concepto.
* **Análisis Preliminar de Datos:**
  + Una vez que se tienen datos iniciales (del piloto físico o de las primeras ejecuciones de la simulación), se realiza un análisis exploratorio de datos (EDA).
  + Esto incluye calcular estadísticas descriptivas (media, mediana, desviación estándar), visualizar las series temporales de los datos de los sensores, buscar correlaciones entre variables y identificar posibles outliers o patrones anómalos.
  + Este análisis preliminar es vital para comprender la naturaleza de los datos y para informar el diseño del agente de IA y los umbrales de decisión.

Comprender cómo se abordaría la recolección y el análisis de datos en un programa piloto real, incluso si el trabajo se centra principalmente en la simulación, es de gran valor. Ayuda a diseñar una simulación más realista, a definir parámetros de entrada más fundamentados y a interpretar los resultados de la simulación con una perspectiva práctica. Los datos del mundo real suelen ser ruidosos, contener valores atípicos y presentar desafíos que una simulación idealizada podría no capturar.48 Intentar modelar algunas de estas imperfecciones o, al menos, ser consciente de ellas, puede hacer que las conclusiones sean más robustas y aplicables.

**VII. Ejecución de Escenarios de Prueba y Validación Rigurosa del Sistema Simulado**

**A. Definición de Escenarios de Prueba Clave Basados en el FMEA Simplificado del Moto-Compresor.**

El Análisis de Modos de Fallo y Efectos (FMEA) simplificado, realizado en la Sección II.B, es la herramienta principal para definir escenarios de prueba significativos y relevantes para la simulación. En lugar de probar combinaciones aleatorias de parámetros, el FMEA permite enfocar los esfuerzos de prueba en los modos de fallo que presentan el mayor riesgo (mayor RPN) o que son de particular interés para los objetivos del PdM.

Los escenarios de prueba se diseñarán para:

1. **Simular Modos de Fallo Específicos:** Se seleccionarán los modos de fallo con RPN más alto de la Tabla 1. Por ejemplo:
   * *Escenario 1:* Desgaste acelerado de sellos en la unidad compresora (alto RPN debido a alta ocurrencia y severidad moderada). Se simulará una tasa de disminución de health más rápida asociada a este componente.
   * *Escenario 2:* Fallo de lubricación en la unidad compresora (alto RPN debido a alta severidad). Se simulará un evento que cause una rápida caída de health o un aumento súbito de la temperatura\_simulada.
   * *Escenario 3:* Sobrecalentamiento del motor eléctrico.
2. **Evaluar la Detección y Respuesta del Agente de IA:** Para cada escenario de fallo, se observará si el agente de IA detecta la condición anómala (basada en la health u otros indicadores simulados) y toma la decisión de mantenimiento apropiada según sus reglas o modelo.
3. **Variar Condiciones Operativas:** Se pueden incluir escenarios que simulen diferentes cargas de trabajo o condiciones ambientales (conceptualmente, ya que el modelo de moto-compresor es simplificado) para ver cómo afectan la tasa de degradación y la efectividad del agente de IA.
4. **Condiciones Iniciales y Parámetros:** Para cada escenario, se definirán claramente las condiciones iniciales del moto-compresor (ej. health inicial) y los parámetros de la simulación (ej. tasa de degradación base, frecuencia de monitoreo del agente).8

El FMEA no solo identifica qué fallos son importantes simular, sino que también ayuda a definir cómo se manifestarían esos fallos en términos de las variables que el agente de IA puede percibir (como la variable health). Esto asegura que los escenarios de prueba sean relevantes para evaluar la capacidad del sistema de PdM simulado para manejar los riesgos más significativos identificados en el análisis FMEA.35

**B. Estrategias de Prueba y Validación para el Modelo de Simulación y el Agente de IA (incluyendo validación de datos y modelos).**

La validación es un proceso iterativo crucial para construir confianza en los resultados de la simulación y en las decisiones del agente de IA.

* **Validación del Modelo de Simulación SimPy:**
  + **Verificación del Modelo:** Asegurar que el código SimPy implementa correctamente la lógica definida (ej. que los tiempos de proceso son correctos, que los recursos se solicitan y liberan adecuadamente, que la variable health se actualiza según el modelo de degradación). Esto se puede hacer mediante pruebas unitarias de las funciones clave y la revisión del código.
  + **Calibración (Conceptual):** Aunque no se disponga de datos reales extensos, se pueden ajustar los parámetros del modelo de simulación (ej. tasas de degradación, tiempos de reparación) para que los resultados globales (ej. tiempo medio entre fallos simulado) se alineen con expectativas razonables o datos de literatura para equipos similares.119
  + **Backtesting (Conceptual):** Si existieran datos históricos simplificados del comportamiento de un moto-compresor (ej. frecuencia de ciertos tipos de mantenimiento), se podría intentar que la simulación reproduzca cualitativamente esos patrones.119
  + **Pruebas de Estrés:** Someter el modelo de simulación a condiciones extremas (ej. tasas de fallo muy altas, operarios de mantenimiento con disponibilidad muy baja) para observar su comportamiento y estabilidad.119
  + **Análisis de Sensibilidad:** Variar parámetros clave de la simulación (ej. umbrales de salud para mantenimiento, tiempo de inspección) y observar cómo afectan las métricas de salida (ej. costo total de mantenimiento, disponibilidad del equipo).
* **Validación del Agente de IA:**
  + **Pruebas Centradas en Datos (Simulados):**
    - Asegurar que los "sensores simulados" (es decir, los atributos del objeto MotoCompressor que lee el agente) proporcionen datos coherentes y representativos al agente.
    - Si se usa un modelo de ML, validar la calidad y representatividad de los datos generados por la simulación que se usan para entrenarlo.120
  + **Pruebas Centradas en el Modelo (Agente):**
    - **Agente Basado en Reglas:** Verificar que cada regla se active correctamente bajo las condiciones simuladas apropiadas y que la acción resultante sea la esperada. Se pueden crear escenarios de prueba específicos para cada regla.
    - **(Si aplica) Agente Basado en ML Simple:** Evaluar el rendimiento del modelo de ML utilizando un conjunto de datos de prueba generado por la simulación. Métricas como precisión, recall, F1-score (para clasificación) o MSE, R² (para regresión) son relevantes.120
    - **Robustez:** Probar cómo responde el agente a condiciones inesperadas o ligeramente diferentes de aquellas para las que fue diseñado (ej. valores de health atípicos).
    - **Explicabilidad (para reglas):** La lógica de un agente basado en reglas es inherentemente explicable. Se debe documentar claramente cada regla y su justificación.

Las estrategias de validación deben ser pragmáticas y enfocadas.120 El objetivo es demostrar que se ha realizado un esfuerzo razonable para verificar y validar el sistema dentro de las limitaciones del proyecto.

La validación no debe ser un paso único al final del proyecto. Es un proceso continuo que se integra a lo largo del ciclo de desarrollo. Cada vez que se modifica el modelo de simulación o la lógica del agente, se deben realizar pruebas para asegurar que los cambios no introducen errores y que el sistema sigue comportándose como se espera.119 Este enfoque iterativo permite construir confianza gradualmente y corregir problemas de manera temprana, lo cual es mucho más eficiente que intentar solucionar todos los errores al final.

**C. Evaluación de las Predicciones del Agente de IA y Métricas de Rendimiento del Sistema de PdM.**

Evaluar el rendimiento del sistema de PdM simulado implica medir qué tan bien el agente de IA toma decisiones y cómo estas decisiones impactan los objetivos generales del mantenimiento (ej. reducir costos, aumentar disponibilidad).

* **Métricas para el Agente de IA (específicas a su tarea):**
  + **Si predice fallos (clasificación):**
    - *Precisión (Accuracy):* Porcentaje de predicciones correctas (fallo/no fallo).
    - *Precisión (Precision):* De todas las veces que predijo fallo, ¿cuántas veces realmente iba a fallar (según la simulación)? (TP / (TP + FP)).
    - *Recall (Sensibilidad):* De todas las fallas reales (simuladas), ¿cuántas fueron detectadas por el agente? (TP / (TP + FN)).
    - *F1-Score:* Media armónica de Precision y Recall.
    - *Matriz de Confusión:* Para visualizar TP, TN, FP, FN.104
  + **Si estima RUL (regresión):**
    - *Error Cuadrático Medio (MSE) o Raíz del Error Cuadrático Medio (RMSE):* Mide la diferencia promedio entre el RUL predicho y el RUL real simulado.
    - *Error Absoluto Medio (MAE):* Similar al RMSE pero menos sensible a outliers.
    - *R-cuadrado (R²):* Proporción de la varianza en el RUL que es explicada por el modelo.
* **Métricas de Rendimiento del Sistema de PdM Global (obtenidas de la simulación):**
  + **Tiempo Medio Entre Fallos (MTBF) Simulado:** Comparar el MTBF con el agente de PdM vs. un escenario base (ej. solo mantenimiento correctivo).
  + **Disponibilidad del Moto-Compresor Simulado:** Porcentaje de tiempo que el moto-compresor está operativo.
  + **Costo Total de Mantenimiento Simulado:** Suma de costos de inspecciones, reparaciones, reemplazos y costo de tiempo de inactividad.
  + **Número de Fallos Catastróficos Evitados (Simulados):** Comparado con un escenario sin PdM.
  + **Tasa de Falsas Alarmas:** Número de mantenimientos programados por el agente que resultaron ser innecesarios (ej. la salud no estaba tan baja como para justificar la intervención).
  + **Utilización de Operarios de Mantenimiento:** Porcentaje de tiempo que los operarios están ocupados.

La elección de métricas adecuadas es fundamental para cuantificar objetivamente el rendimiento. No todas las métricas son igualmente importantes para todos los objetivos. Por ejemplo, en el mantenimiento predictivo, minimizar los falsos negativos (fallos reales que no son detectados por el agente) suele ser más crítico que minimizar los falsos positivos (alarmas de mantenimiento que resultan innecesarias), ya que un fallo no detectado puede llevar a consecuencias mucho más costosas.124 La simulación permite explorar el trade-off entre estas métricas al ajustar los umbrales de decisión del agente.

**Tabla 4: Comparativa de Métricas de Evaluación del Agente de IA/Simulación**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Escenario de Prueba / Configuración del Agente** | **Precisión de Predicción de Fallo (si aplica)** | **Recall de Predicción de Fallo (si aplica)** | **Tasa de Falsas Alarmas** | **MTBF Simulado (horas)** | **Disponibilidad Simulada (%)** | **Costo Total de Mantenimiento Simulado (€/periodo)** |
| **Base: Solo Mantenimiento Correctivo** | N/A | N/A | 0% | 150 | 85% | 5000 |
| **Agente PdM: Umbral Salud Crítica = 20** | 0.85 (ejemplo) | 0.70 (ejemplo) | 15% | 250 | 92% | 3500 |
| **Agente PdM: Umbral Salud Crítica = 30** | 0.78 (ejemplo) | 0.85 (ejemplo) | 25% | 220 | 90% | 3800 |
| **Agente PdM con ML (Árbol Decisión)** | 0.90 (ejemplo) | 0.75 (ejemplo) | 10% | 280 | 94% | 3200 |

*Nota: Los valores son ilustrativos y dependerán de los resultados específicos de la simulación.*

Esta tabla permite una comparación cuantitativa y objetiva del rendimiento del sistema de PdM bajo diferentes condiciones o con diferentes configuraciones del agente de IA. Ayuda a identificar las fortalezas y debilidades del enfoque de PdM propuesto y a justificar las conclusiones.

**VIII. Refinamiento Iterativo del Sistema y Documentación Integral del Proyecto**

**A. Estrategias para el Refinamiento del Modelo de Simulación y la Lógica del Agente de IA Basadas en Resultados de Pruebas.**

El desarrollo de un sistema de simulación y un agente de IA es inherentemente un proceso iterativo. Los resultados obtenidos de las pruebas y la validación (Sección VII) deben utilizarse para refinar y mejorar continuamente tanto el modelo de simulación como la lógica del agente de IA.5

* **Refinamiento del Modelo de Simulación SimPy:**
  + **Ajuste de Parámetros de Degradación:** Si las pruebas muestran que el moto-compresor simulado falla con demasiada frecuencia o muy raramente en comparación con las expectativas (o datos de referencia, si existen), se deben ajustar los parámetros del modelo de degradación (ej. la tasa de disminución de health, los parámetros de las distribuciones de probabilidad de fallo).
  + **Mejora de la Lógica de Fallos:** Si ciertos modos de fallo importantes identificados en el FMEA no están siendo capturados adecuadamente por la simulación, se puede refinar la lógica que los desencadena.
  + **Calibración de Tiempos de Proceso:** Ajustar los tiempos simulados para operaciones, inspecciones y reparaciones para que sean más realistas, basándose en estimaciones o datos disponibles.
  + **Complejidad del Modelo:** Evaluar si el modelo es demasiado simple (omitiendo dinámicas importantes) o demasiado complejo (difícil de analizar y validar, consume mucho tiempo de simulación). Se puede simplificar o añadir detalle según sea necesario.
* **Refinamiento de la Lógica del Agente de IA:**
  + **Agente Basado en Reglas:**
    - *Ajuste de Umbrales:* Modificar los umbrales de health u otros indicadores que el agente utiliza para tomar decisiones. Por ejemplo, si hay demasiadas falsas alarmas, se podría aumentar el umbral para una inspección. Si se están perdiendo fallos, se podría disminuir.
    - *Adición/Modificación de Reglas:* Basándose en los escenarios donde el agente no tomó la decisión óptima, se pueden añadir nuevas reglas o modificar las existentes para cubrir esos casos.
    - *Priorización de Reglas:* Si hay reglas conflictivas, se debe establecer un orden de prioridad claro.
  + **(Si aplica) Agente Basado en ML Simple:**
    - *Re-entrenamiento del Modelo:* Si el rendimiento del modelo ML no es satisfactorio, se puede re-entrenar con más datos (generados por la simulación refinada), con diferentes características (ingeniería de características) o con diferentes hiperparámetros.
    - *Selección de Características:* Analizar qué "sensores simulados" son más predictivos y refinar el conjunto de características de entrada para el modelo.
    - *Considerar Otros Modelos Simples:* Si un árbol de decisión no funciona bien, se podría probar con una regresión logística o viceversa, siempre manteniendo la simplicidad.

El ciclo de **prueba -> análisis de resultados -> refinamiento** es fundamental.109 Cada iteración debería llevar a un sistema simulado que se comporta de manera más realista o a un agente de IA que toma decisiones más efectivas. La simulación es una herramienta poderosa precisamente porque permite este ciclo iterativo de forma rápida y económica, sin los costos o riesgos de experimentar en un sistema físico.

#### Fuentes citadas

1. Reinforcement learning for predictive maintenance: a systematic ..., acceso: mayo 16, 2025, <https://www.researchgate.net/publication/369530421_Reinforcement_learning_for_predictive_maintenance_a_systematic_technical_review>
2. ¿Cómo la IA revoluciona el mantenimiento predictivo en la Industria ..., acceso: mayo 16, 2025, <https://datision.com/blog/como-la-inteligencia-artificial-revoluciona-el-mantenimiento-predictivo-en-la-industria-4-0/>
3. Principales tendencias en software de mantenimiento: IA, predicción e innovaciones futuras, acceso: mayo 16, 2025, <https://www.manwinwin.com/es/tendencias-del-software-de-mantenimiento/>
4. A Machine Learning Implementation to Predictive Maintenance and Monitoring of Industrial Compressors - MDPI, acceso: mayo 16, 2025, <https://www.mdpi.com/1424-8220/25/4/1006>
5. Usar IA en el mantenimiento predictivo: lo que necesitas saber ..., acceso: mayo 16, 2025, <https://www.oracle.com/cr/scm/ai-predictive-maintenance/>
6. The Role of Machine Health Monitoring for Preventive Maintenance, acceso: mayo 16, 2025, <https://www.ripik.ai/machine-health-monitoring-for-preventive-maintenance/>
7. Gemelos digitales y simulaciones - ABB, acceso: mayo 16, 2025, [https://search.abb.com/library/Download.aspx?DocumentID=9AKK107492A3437&LanguageCode=es&DocumentPartId=&Action=Launch](https://search.abb.com/library/Download.aspx?DocumentID=9AKK107492A3437&LanguageCode=es&DocumentPartId&Action=Launch)
8. Software de simulación predictiva | Sistema de administración de energía - ETAP, acceso: mayo 16, 2025, <https://etap.com/es/product/predictive-simulation-software>
9. Inteligencia Artificial:¿Qué es el Aprendizaje De Refuerzo? Una Explicación Simple y Ejemplos Prácticos - Dell, acceso: mayo 16, 2025, <https://www.dell.com/es-es/blog/inteligencia-artificial-aprendizaje-refuerzo-explicacion-y-ejemplos/>
10. ¿Qué son los agentes de IA? - Automation Anywhere, acceso: mayo 16, 2025, <https://www.automationanywhere.com/la/rpa/ai-agents>
11. NREL/DSS-SimPy-RL: This repository is an Reinforcement ... - GitHub, acceso: mayo 16, 2025, <https://github.com/NREL/DSS-SimPy-RL>
12. Agentes de IA: Convierta los datos en decisiones inteligentes - Skyone, acceso: mayo 16, 2025, <https://skyone.solutions/es/blog/iowa/agente_ia/>
13. Notebook - jckantor.github.io, acceso: mayo 16, 2025, <https://jckantor.github.io/cbe30338-2021/07.02-Agent-Based-Models.html>
14. 19.3: Agent-Environment Interaction - Mathematics LibreTexts, acceso: mayo 16, 2025, <https://math.libretexts.org/Bookshelves/Scientific_Computing_Simulations_and_Modeling/Introduction_to_the_Modeling_and_Analysis_of_Complex_Systems_(Sayama)/19%3A_AgentBased_Models/19.03%3A_Agent-Environment_Interaction>
15. SimPy: Simulating Real-World Processes With Python, acceso: mayo 16, 2025, <https://realpython.com/simpy-simulating-with-python/>
16. Time to marry simulation models with machine learning - Dr. Benjamin Schumann, acceso: mayo 16, 2025, <https://benjamin-schumann.com/blog/simulation-and-machine-learning>
17. arxiv.org, acceso: mayo 16, 2025, <https://arxiv.org/abs/2405.01562>
18. Discrete Event Simulation: It's Easy with SimPy! - arXiv, acceso: mayo 16, 2025, <https://arxiv.org/html/2405.01562v1>
19. Example Case: Using DevSecOps to Redefine Minimum Viable ..., acceso: mayo 16, 2025, <https://insights.sei.cmu.edu/blog/example-case-using-devsecops-to-redefine-minimum-viable-product/>
20. www.theseus.fi, acceso: mayo 16, 2025, <https://www.theseus.fi/bitstream/handle/10024/147703/Storbacka_Oscar.pdf?sequence=1&isAllowed=y>
21. Método Moscow: Priorización UX/UI | Blog Aguayo, acceso: mayo 16, 2025, <https://aguayo.co/es/blog-aguayo-experiencia-usuario/metodo-moscow-priorizacion-ux/>
22. A Quick Guide to the MoSCoW Method Technique | Wrike, acceso: mayo 16, 2025, <https://www.wrike.com/blog/guide-to-moscow-method/>
23. What is MoSCoW Prioritization? | Overview of the MoSCoW Method, acceso: mayo 16, 2025, <https://www.productplan.com/glossary/moscow-prioritization/>
24. How to Use AI for Writing Project Scope of Work: Unlock Expert ..., acceso: mayo 16, 2025, <https://www.itsdart.com/blog/use-ai-for-writing-project-scope-of-work>
25. AI Project Scoping. How to Define the Scope and Goals of Your AI ..., acceso: mayo 16, 2025, <https://nexocode.com/blog/posts/ai-project-scoping-how-to-define-the-scope-of-ml-project/>
26. ▷TFG Mecatrónica Industrial ⚙️【+300 Ejemplos en PDF】, acceso: mayo 16, 2025, <https://trabajosfindegrado.es/tfg-grado-superior/tfg-mecatronica-industrial/>
27. Predictive maintenance's role in routine compressed air system ..., acceso: mayo 16, 2025, <https://www.plantengineering.com/predictive-maintenances-role-in-routine-compressed-air-system-service/>
28. Mantenimiento predictivo de un compresor - Gradhoc, acceso: mayo 16, 2025, <https://gradhoc.com/es/Art%C3%ADculo/mantenimiento-predictivo-de-un-compresor/>
29. Definición y técnicas de mantenimiento predictivo - UpKeep, acceso: mayo 16, 2025, <https://upkeep.com/es/learning/predictive-maintenance/>
30. Maintenance in the World of Pumps | Maintenance World, acceso: mayo 16, 2025, <https://maintenanceworld.com/2024/02/15/maintenance-in-the-world-of-pumps/>
31. Sistema de control para Compresores: Reseña - c3controls, acceso: mayo 16, 2025, <https://www.c3controls.com/es/documento-tecnico/sistema-de-control-para-compresores-resena/>
32. ¿Cómo realizar el mantenimiento de su compresor? - Blog, acceso: mayo 16, 2025, <https://airpress.es/blog/post/como-realizar-mantenimiento-compresores>
33. Monitoreo de la condición de los motores - Turck México, acceso: mayo 16, 2025, <https://www.turck.com.mx/es/monitoreo-de-la-condicion-de-los-motores-35149.php>
34. WEG Motor Scan - Sensores de Monitoreo de Condición, acceso: mayo 16, 2025, <https://www.weg.net/catalog/weg/BR/es/Soluciones-Digitales/Gesti%C3%B3n-de-activos/Sensores-de-Monitoreo-de-Condici%C3%B3n/WEG-Motor-Scan---Sensor/c/BR_WDC_EDGE_DEVICE_MOTOR_SCAN_SENSOR>
35. 8 Best Tools for Predictive Maintenance - WorkTrek - Worktrek CMMS, acceso: mayo 16, 2025, <https://worktrek.com/blog/predictive-maintenance-tools/>
36. Guide to Failure Mode and Effect Analysis - FMEA | Juran Institute ..., acceso: mayo 16, 2025, <https://www.juran.com/blog/guide-to-failure-mode-and-effect-analysis-fmea/>
37. 10 Steps to Conduct a PFMEA | QualityTrainingPortal, acceso: mayo 16, 2025, <https://qualitytrainingportal.com/resources/fmea-resource-center/traditional-rpn-fmea/10-steps-conduct-pfmea/>
38. Air Compressor FMEA | PDF | Top Down And Bottom Up Design - Scribd, acceso: mayo 16, 2025, <https://www.scribd.com/doc/150508013/Air-Compressor-FMEA>
39. Maintenance Manager's Guide to Creating & Using FMEAs | Fiix, acceso: mayo 16, 2025, <https://fiixsoftware.com/blog/fmea-for-maintenance/>
40. Guide to Failure Mode and Effect Analysis - FMEA | Juran Institute ..., acceso: mayo 16, 2025, <https://juran.com/blog/guide-to-failure-mode-and-effect-analysis-fmea/>
41. "The Effective Implementation of Agile Project Management in ..., acceso: mayo 16, 2025, <https://digitalcommons.harrisburgu.edu/dandt/56/>
42. Los 6 sensores para el mantenimiento predictivo que optimizan los ..., acceso: mayo 16, 2025, <https://upkeep.com/es/learning/sensors-for-predictive-maintenance/>
43. Guia de Herramientas para el Mantenimiento Predictivo, acceso: mayo 16, 2025, <https://www.cursosaula21.com/herramientas-para-el-mantenimiento-predictivo/>
44. (PDF) An Evaluating Study of Using Thermal Imaging and Convolutional Neural Network for Fault Diagnosis of Reciprocating Compressors - ResearchGate, acceso: mayo 16, 2025, <https://www.researchgate.net/publication/343911022_An_Evaluating_Study_of_Using_Thermal_Imaging_and_Convolutional_Neural_Network_for_Fault_Diagnosis_of_Reciprocating_Compressors>
45. (PDF) Machine Fault Signature Analysis - ResearchGate, acceso: mayo 16, 2025, <https://www.researchgate.net/publication/26512923_Machine_Fault_Signature_Analysis>
46. Data Acquisition System | GeeksforGeeks, acceso: mayo 16, 2025, <https://www.geeksforgeeks.org/data-acquisition-system/>
47. Adquisición de datos (DAQ): la guía completa | Dewesoft, acceso: mayo 16, 2025, <https://dewesoft.com/es/blog/que-es-adquisicion-de-datos>
48. Mantenimiento predictivo y análisis de datos: 5 pasos para el éxito, acceso: mayo 16, 2025, <https://www.emaint.com/es/blog-data-analytics-in-maintenance/>
49. ijecce.org, acceso: mayo 16, 2025, <https://ijecce.org/administrator/components/com_jresearch/files/publications/IJECCE_65_Final.pdf>
50. Documentation for SimPy — SimPy 4.1.2.dev8+g81c7218 ..., acceso: mayo 16, 2025, <https://simpy.readthedocs.io/en/latest/contents.html>
51. Simulation with SimPy - In Depth Manual — SimPy v2.2 documentation - Pythonhosted.org, acceso: mayo 16, 2025, <https://pythonhosted.org/SimPy/Manuals/Manual.html>
52. Machine Shop — SimPy 4.1.2.dev8+g81c7218 documentation, acceso: mayo 16, 2025, <https://simpy.readthedocs.io/en/latest/examples/machine_shop.html>
53. Basic Concepts — SimPy 4.1.2.dev8+g81c7218 documentation, acceso: mayo 16, 2025, <https://simpy.readthedocs.io/en/latest/simpy_intro/basic_concepts.html>
54. Simpy y Pygame aplicados a un proyecto de Ingeniería: Simulación de Transporte Minero., acceso: mayo 16, 2025, <https://www.youtube.com/watch?v=270MHSBLpT8>
55. Simulación eventos discretos: Restaurante | Módulo Simpy Python | Parte I - YouTube, acceso: mayo 16, 2025, <https://www.youtube.com/watch?v=qnk0VP_T-Xg>
56. Simulations in Python: Discrete Event Simulation with SimPy | PyData NYC 2022 - YouTube, acceso: mayo 16, 2025, <https://www.youtube.com/watch?v=TALKZZV0TiU&pp=0gcJCfcAhR29_xXO>
57. I Wrote a Guide to Simulation in Python with SimPy - Reddit, acceso: mayo 16, 2025, <https://www.reddit.com/r/Python/comments/1gz3bgp/i_wrote_a_guide_to_simulation_in_python_with_simpy/>
58. Modelling coupled degradation mechanisms in PyBaMM ..., acceso: mayo 16, 2025, <https://docs.pybamm.org/en/latest/source/examples/notebooks/models/coupled-degradation.html>
59. I Wrote a Handbook on Simulation in Python with SimPy : r/engineering - Reddit, acceso: mayo 16, 2025, <https://www.reddit.com/r/engineering/comments/1hs8wlb/i_wrote_a_handbook_on_simulation_in_python_with/>
60. Examples — SimPy 4.1.1 documentation, acceso: mayo 16, 2025, <https://simpy.readthedocs.io/en/stable/examples/>
61. www.iapsam.org, acceso: mayo 16, 2025, <https://www.iapsam.org/psam12/proceedings/paper/paper_492_1.pdf>
62. Shared Resources — SimPy 4.1.2.dev8+g81c7218 documentation, acceso: mayo 16, 2025, <https://simpy.readthedocs.io/en/latest/topical_guides/resources.html>
63. Examples — SimPy 3.0.8 documentation - Read the Docs, acceso: mayo 16, 2025, <https://simpy.readthedocs.io/en/3.0.8/examples/>
64. Events — SimPy 4.1.2.dev8+g81c7218 documentation, acceso: mayo 16, 2025, <https://simpy.readthedocs.io/en/latest/topical_guides/events.html>
65. Environments — SimPy 4.1.2.dev8+g81c7218 documentation, acceso: mayo 16, 2025, <https://simpy.readthedocs.io/en/latest/topical_guides/environments.html>
66. Simpy how to access objects in a resource queue - Stack Overflow, acceso: mayo 16, 2025, <https://stackoverflow.com/questions/39228126/simpy-how-to-access-objects-in-a-resource-queue>
67. Job shop simulation in SimPy - YouTube, acceso: mayo 16, 2025, <https://m.youtube.com/watch?v=BkfaN3vl2xQ>
68. Advanced Simpy | PDF | Network Packet | Thread (Computing) - Scribd, acceso: mayo 16, 2025, <https://www.scribd.com/document/292088532/Advanced-Simpy>
69. SimPy basics — SimPy 4.1.2.dev8+g81c7218 documentation, acceso: mayo 16, 2025, <https://simpy.readthedocs.io/en/latest/topical_guides/simpy_basics.html>
70. Simulation Programming with Python, acceso: mayo 16, 2025, <https://users.iems.northwestern.edu/~nelsonb/IEMS435/PythonSim.pdf>
71. Introduction to SimPy - CDN, acceso: mayo 16, 2025, <https://bpb-eu-w2.wpmucdn.com/sites.aub.edu.lb/dist/7/128/files/2023/11/Lab-3-Introduction-to-SimPy.pdf>
72. The Bank: Examples of SimPy Simulation - PythonHosted.org, acceso: mayo 16, 2025, <https://pythonhosted.org/SimPy/Tutorials/TheBank.html>
73. linearDegradationModel - MathWorks, acceso: mayo 16, 2025, <https://www.mathworks.com/help/predmaint/ref/lineardegradationmodel.html>
74. 8.4.2.3. Fitting models using degradation data instead of failures, acceso: mayo 16, 2025, <https://www.itl.nist.gov/div898/handbook/apr/section4/apr423.htm>
75. 07.01-Introduction-to-Simpy.ipynb - Colab, acceso: mayo 16, 2025, <https://colab.research.google.com/github/jckantor/cbe30338-2021/blob/master/docs/07.01-Introduction-to-Simpy.ipynb>
76. Core event types — SimPy 4.1.2.dev8+g81c7218 documentation, acceso: mayo 16, 2025, <https://simpy.readthedocs.io/en/latest/api_reference/simpy.events.html>
77. Core event types — SimPy 3.0 documentation - Read the Docs, acceso: mayo 16, 2025, <https://simpy.readthedocs.io/en/3.0/api_reference/simpy.events.html>
78. The Bank Tutorial part 2: More examples of SimPy Simulation - Pythonhosted.org, acceso: mayo 16, 2025, <https://pythonhosted.org/SimPy/Tutorials/TheBank2.html>
79. (PDF) A Comparison of Maintenance Policies for Multi-Component Systems Through Discrete Event Simulation of Faults - ResearchGate, acceso: mayo 16, 2025, <https://www.researchgate.net/publication/343446480_A_Comparison_of_Maintenance_Policies_for_Multi-Component_Systems_Through_Discrete_Event_Simulation_of_Faults>
80. Cómo utilizar agentes Reflex simples para la automatización de tareas, acceso: mayo 16, 2025, <https://clickup.com/es-ES/blog/258620/agente-reflejo-simple>
81. Análisis de los agentes de IA: tipos, capacidades y aplicaciones, acceso: mayo 16, 2025, <https://www.automationanywhere.com/la/company/blog/automation-ai/exploring-ai-agents-types-capabilities-and-real-world-applications>
82. Agentes IA: Guía completa sobre Agentes de Inteligencia Artificial, acceso: mayo 16, 2025, <https://acumbamail.com/blog/agentes-ia/>
83. Entrenamiento de modelos de Machine Learning con scikit-learn (v2 ..., acceso: mayo 16, 2025, <https://learn.microsoft.com/es-es/azure/machine-learning/how-to-train-scikit-learn?view=azureml-api-2>
84. Instalación de la versión de desarrollo de scikit-learn, acceso: mayo 16, 2025, <https://qu4nt.github.io/sklearn-doc-es/developers/advanced_installation.html>
85. Setting Up a Data Science Environment in Python | GeeksforGeeks, acceso: mayo 16, 2025, <https://www.geeksforgeeks.org/setting-up-a-data-science-environment-in-python/>
86. Beginners Guide: Setting Up Your Local Environment for Machine ..., acceso: mayo 16, 2025, <https://dev.to/nicobistolfi/beginners-guide-setting-up-your-local-environment-for-machine-learning-with-miniconda-and-python-5f7b>
87. Agentes de IA: en qué consisten y cómo van a transformar por completo la forma de trabajar, acceso: mayo 16, 2025, <https://www.plainconcepts.com/es/agentes-ia/>
88. Comprender a los agentes de IA: El futuro de los sistemas autónomos - DataCamp, acceso: mayo 16, 2025, <https://www.datacamp.com/es/blog/ai-agents>
89. arxiv.org, acceso: mayo 16, 2025, <https://arxiv.org/pdf/2307.03860>
90. DRLAttack: A Deep Reinforcement Learning-Based Framework for Data Poisoning Attack on Collaborative Filtering Algorithms - MDPI, acceso: mayo 16, 2025, <https://www.mdpi.com/2076-3417/15/10/5461>
91. A Simulation Study on Energy Optimization in Building Control with Reinforcement Learning, acceso: mayo 16, 2025, <https://www.researchgate.net/publication/384177260_A_Simulation_Study_on_Energy_Optimization_in_Building_Control_with_Reinforcement_Learning>
92. Reinforcement learning for dynamic condition-based maintenance ..., acceso: mayo 16, 2025, <https://www.researchgate.net/publication/341912211_Reinforcement_learning_for_dynamic_condition-based_maintenance_of_a_system_with_individually_repairable_components>
93. (PDF) An empirical study of the naïve REINFORCE algorithm for predictive maintenance, acceso: mayo 16, 2025, <https://www.researchgate.net/publication/389675511_An_empirical_study_of_the_naive_REINFORCE_algorithm_for_predictive_maintenance>
94. Condition-based maintenance with reinforcement learning for refrigeration systems with selected monitored features | Request PDF - ResearchGate, acceso: mayo 16, 2025, <https://www.researchgate.net/publication/369113078_Condition-based_maintenance_with_reinforcement_learning_for_refrigeration_systems_with_selected_monitored_features>
95. Reinforcement Learning in Process Industries: Review and Perspective - SciEngine, acceso: mayo 16, 2025, <https://www.sciengine.com/doi/pdfView/95BA0BED47914FAE9F7FB9B18F1DF35E>
96. Mastering Reinforcement Learning: Foundations, Algorithms, and Real-World Applications - OSF, acceso: mayo 16, 2025, <https://osf.io/bg79j_v1/download/?format=pdf>
97. Learning Model Building in Scikit-learn | GeeksforGeeks, acceso: mayo 16, 2025, <https://www.geeksforgeeks.org/learning-model-building-scikit-learn-python-machine-learning-library/>
98. Use scikit-learn on Databricks, acceso: mayo 16, 2025, <https://docs.databricks.com/aws/en/machine-learning/train-model/scikit-learn>
99. How to Make Predictions with scikit-learn - MachineLearningMastery ..., acceso: mayo 16, 2025, <https://machinelearningmastery.com/make-predictions-scikit-learn/>
100. What is Python sklearn predict function working principle - Stack Overflow, acceso: mayo 16, 2025, <https://stackoverflow.com/questions/65734567/what-is-python-sklearn-predict-function-working-principle>
101. Guía para implementar Machine Learning 2025 - Tecnología bi, acceso: mayo 16, 2025, <https://tecnologiabi.com/guia-para-implementar-machine-learning/>
102. Mantenimiento Predictivo - Machine Learning con Python - YouTube, acceso: mayo 16, 2025, <https://www.youtube.com/watch?v=7zStVmc5mkg>
103. DanniRodrJ/milling-machine\_failure-prediction: Machine Learning aplicado al mantenimiento predictivo. Se realizaron 2 modelos: 1 por medio de clasificación binaria que predice si una máquina fresadora estará en riesgo de fallar o no, y el 2 modelo a través de clasificación multiclase que predecirá el modo de falla - GitHub, acceso: mayo 16, 2025, <https://github.com/DanniRodrJ/milling-machine_failure-prediction>
104. Predictive Analytics Made Simple & How To Python Tutorial, acceso: mayo 16, 2025, <https://spotintelligence.com/2024/10/14/predictive-analytics-made-simple-how-to-python-tutorial/>
105. Applied Predictive Maintenance Part 4 of 6: Machine Learning Workflows with Sci-Kit Learn to Build Predictive Maintenance Models - Patterson Consulting, acceso: mayo 16, 2025, <https://pattersonconsultingtn.com/blog/predictive_maintenance_w_snowflake_ml_part_4_modeling.html>
106. Gestión Ágil de Proyectos: Qué es, Metodologías y Beneficios, acceso: mayo 16, 2025, <https://www.aden.org/business-magazine/gestion-agil-proyectos-metodologias/>
107. Critical Success Factors for Agile Project Management in Non ..., acceso: mayo 16, 2025, <https://scholarworks.wmich.edu/cgi/viewcontent.cgi?article=4210&context=dissertations>
108. Gestión ágil de proyectos: ¿Qué es y por qué es importante?, acceso: mayo 16, 2025, <https://www.aden.org/business-magazine/gestion-agil-de-proyectos-que-es-y-por-que-es-importante/>
109. How to Build a Predictive Maintenance Program - WorkTrek, acceso: mayo 16, 2025, <https://worktrek.com/blog/build-predictive-maintenance-program/>
110. Health Monitoring and Predictive Maintenance of DC Motor ..., acceso: mayo 16, 2025, <https://www.researchgate.net/publication/389988486_Health_Monitoring_and_Predictive_Maintenance_of_DC_Motor>
111. www.isca.me, acceso: mayo 16, 2025, <https://www.isca.me/IJES/Archive/v14/i1/2.ISCA-RJEngS-2024-014.pdf>
112. How to Use the ADXL345 on Raspberry Pi : 3 Steps - Instructables, acceso: mayo 16, 2025, <https://www.instructables.com/how-to-use-the-ADXL345-on-Raspberry-pi/>
113. [GUIDE] Hooking up your Creality ADXL345/G-Sensor to generic Klipper - Reddit, acceso: mayo 16, 2025, <https://www.reddit.com/r/klippers/comments/1e5ocz5/guide_hooking_up_your_creality_adxl345gsensor_to/>
114. Temperature Sensor Using Thermistor With Arduino Uno : 4 Steps ..., acceso: mayo 16, 2025, <https://www.instructables.com/Temperature-Sensor-Using-Thermistor-With-Arduino-U/>
115. Arduino – Datalogger With Temperature Sensor and Photoresistor : 4 Steps - Instructables, acceso: mayo 16, 2025, <https://www.instructables.com/Arduino-Datalogger-with-Temperature-Sensor-and-Pho/>
116. Sistemas, dispositivos y software de adquisición de datos (DAQ) - NI - National Instruments, acceso: mayo 16, 2025, <https://www.ni.com/es/shop/data-acquisition.html>
117. Air Compressor Remote Condition Monitoring with Opta™ | Arduino ..., acceso: mayo 16, 2025, <https://docs.arduino.cc/tutorials/opta/smart-compressor-app-note/>
118. Writing the data to a CSV file | Temperature log | Python | Coding projects for kids and teens, acceso: mayo 16, 2025, <https://projects.raspberrypi.org/en/projects/temperature-log/2>
119. Validación Del Modelo De Simulación - FasterCapital, acceso: mayo 16, 2025, <https://fastercapital.com/es/tema/validaci%C3%B3n-del-modelo-de-simulaci%C3%B3n.html>
120. Testing AI Applications: Strategy, Tools & Best Practices - testomat.io, acceso: mayo 16, 2025, <https://testomat.io/blog/testing-strategy-for-ai-based-applications/>
121. Generative AI Testing: Essential Strategies for System Validation, acceso: mayo 16, 2025, <https://www.alphabin.co/blog/generative-ai-testing>
122. Technologies Driving Predictive Maintenance - WorkTrek, acceso: mayo 16, 2025, <https://worktrek.com/blog/technologies-driving-predictive-maintenance/>
123. Predictive Modeling Techniques- A Comprehensive Guide [2025], acceso: mayo 16, 2025, <https://www.projectpro.io/article/predictive-modelling-techniques/598>
124. Predictive Maintenance - Nixtla, acceso: mayo 16, 2025, <https://nixtlaverse.nixtla.io/neuralforecast/docs/use-cases/predictive_maintenance.html>
125. Minimum Viable Documentation for Agile Product Teams - Beyond ..., acceso: mayo 16, 2025, <https://beyondthebacklog.com/2024/09/21/minimum-viable-documentation-2/>
126. Minimum Viable Documentation – the art of documentation triage, acceso: mayo 16, 2025, <https://blog.knowledgeowl.com/blog/posts/minimum-viable-documentation/>
127. How to Write a Well-Structured Engineering Report - Vista Projects, acceso: mayo 16, 2025, <https://www.vistaprojects.com/4-easy-sections-to-structure-engineering-reports-effectively/>
128. arnabroy734/machine\_fault\_detection: use of machine learning technique in predictive maintenance - GitHub, acceso: mayo 16, 2025, <https://github.com/arnabroy734/machine_fault_detection>
129. Quickstart - Predictive Maintenance | SynapseML - Microsoft Open Source, acceso: mayo 16, 2025, <https://microsoft.github.io/SynapseML/docs/1.0.2/Explore%20Algorithms/AI%20Services/Quickstart%20-%20Predictive%20Maintenance/>
130. Cómo Documentar Tu Proceso de Desarrollo de IA - Artificial Paintings, acceso: mayo 16, 2025, <https://artificialpaintings.com/es/blog/2024/06/21/como-documentar-tu-proceso-de-desarrollo-de-ia/>
131. Escribir documentacion de código con Inteligencia Artificial - Platzi, acceso: mayo 16, 2025, <https://platzi.com/clases/7964-ia-devs/63310-escribir-documentacion-de-codigo-con-inteligencia/>
132. Las mejores prácticas para la gestión de proyectos de ingeniería ..., acceso: mayo 16, 2025, <https://blog.ferrovial.com/es/2023/08/mejores-practicas-para-gestion-de-proyectos-ingenieria/>
133. Project Management Report: Examples and Writing Tips | Wrike, acceso: mayo 16, 2025, <https://www.wrike.com/project-management-guide/faq/how-to-write-a-project-management-report/>
134. Cómo crear un informe de estado del proyecto eficaz en tan solo 8 pasos - Asana, acceso: mayo 16, 2025, <https://asana.com/es/resources/how-project-status-reports>
135. Webinar | Machine Learning aplicado al mantenimiento predictivo con Python - YouTube, acceso: mayo 16, 2025, <https://www.youtube.com/watch?v=oWv9SprRpCg>
136. Predictive Maintenance Python Application for Machinery - LightningChart, acceso: mayo 16, 2025, <https://lightningchart.com/blog/python/predictive-maintenance-python-application>
137. Sensor - Component Failure Prediction - Kaggle, acceso: mayo 16, 2025, <https://www.kaggle.com/code/ritesh7355/sensor-component-failure-prediction/notebook>