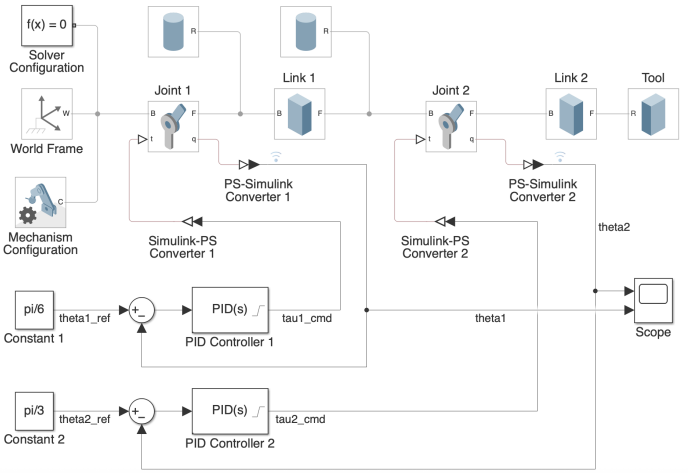
Gemelos digitales en robótica y automatización

Del concepto a la implementación:
Una guía completa con MATLAB y Simulink

Ildeberto de los Santos Ruiz



Septiembre, 2025

Introducción

Los gemelos digitales se han convertido en una herramienta revolucionaria para el diseño, control y mantenimiento de sistemas robóticos e industriales. Un gemelo digital es más que una simple simulación: es una réplica virtual conectada en tiempo real con un activo físico, capaz de reflejar su estado y comportamiento. Este tutorial te guiará a través de los fundamentos de los gemelos digitales, sus componentes clave, beneficios y retos, enfocándose en aplicaciones en robótica y automatización industrial. Además, desarrollaremos paso a paso un ejemplo práctico de cómo construir un gemelo digital de un sistema robótico sencillo usando MATLAB/Simulink, incorporando herramientas como *Simscape Multibody* y *Robotics System Toolbox*.

El objetivo de este documento es ofrecer una guía detallada para estudiantes y profesionales de Ingeniería Mecatrónica con conocimientos intermedios de MATLAB, de manera que puedan iniciarse en la creación de gemelos digitales para robots y sistemas automatizados. A lo largo del tutorial encontrarás secciones bien diferenciadas, ejemplos de código, visualizaciones y recomendaciones prácticas, desde los fundamentos teóricos hasta la implementación en MATLAB y su conexión con datos reales para diagnóstico y mantenimiento predictivo.

¿Qué es un gemelo digital?

Un gemelo digital es una representación virtual de un producto, proceso o sistema físico, ya sea que este se encuentre en operación o en desarrollo. A diferencia de una simulación tradicional aislada, el gemelo digital está vinculado dinámicamente con su contraparte física. Esto significa que el gemelo digital recibe datos en tiempo real del sistema físico (por ejemplo, mediante sensores, controladores o registros históricos) y puede reflejar su estado actual. Básicamente, es como tener una copia digital de una máquina o proceso funcionando en paralelo, sobre la cual podemos observar, experimentar y predecir comportamientos sin intervenir directamente en el sistema real.

En su definición formal, cuando el sistema físico está en funcionamiento, el gemelo digital refleja las condiciones actuales del activo e incorpora datos históricos relevantes; se emplea para evaluar el estado actual y, lo más importante, predecir el comportamiento futuro, refinar controles y optimizar operaciones. Durante la fase de desarrollo de un nuevo producto o máquina, un gemelo digital actúa como un modelo virtual del sistema por construir, facilitando su diseño, pruebas y validación antes de tener un prototipo físico.

¿En qué se diferencia de una simulación convencional?
Principalmente en la conexión en tiempo real y bidireccional con el mundo físico. Un gemelo digital no solo simula cómo podría comportarse algo, sino cómo está comportándose en el momento actual y cómo probablemente lo hará en el futuro próximo, basándose en datos reales que se actualizan continuamente. Esta sincronización permite que cualquier cambio o condición en el objeto físico se refleje inmediatamente en el gemelo digital (y potencialmente viceversa, si se implementa control desde el gemelo).

Componentes de un gemelo digital

Para comprender mejor cómo funcionan, desglosamos un gemelo digital en sus componentes esenciales:

- **Activo físico:** El sistema real que se desea replicar (por ejemplo, un robot industrial, una línea de producción o cualquier máquina física). Es la fuente de datos real y el objeto cuyo comportamiento queremos emular y mejorar.
- **Réplica virtual del activo:** El modelo digital del objeto físico, que incluye su geometría, características dinámicas, propiedades físicas y contexto operativo. En robótica, suele ser un modelo 3D con representación de enlaces, articulaciones, motores, etc., junto con modelos matemáticos o de simulación que reproducen su comportamiento.
- **Conectividad de datos (Integración IoT):** El vínculo entre ambos mundos. Sensores, controladores (PLCs, microcontroladores) y sistemas IoT transmiten datos del objeto físico al modelo virtual en tiempo real. Esta conectividad puede usar diversas tecnologías (redes industriales, WiFi, protocolos como OPC UA, ROS, etc.) y es indispensable para que el gemelo esté "vivo" y actualizado.
- **Plataforma de análisis y procesamiento:** Software y algoritmos que reciben los datos del gemelo digital y los procesan. Aquí entran en juego las simulaciones en sí (por ejemplo, en Simulink), algoritmos de machine learning o reglas de diagnóstico que corren sobre el gemelo digital para extraer información valiosa. Este componente permite agregar inteligencia al gemelo, desde análisis predictivo hasta optimización en línea.
- **Retroalimentación y control:** La capacidad de que el gemelo digital no solo reciba datos, sino que también envíe información o decisiones de vuelta al sistema físico. Por ejemplo, si el gemelo detecta una condición de falla inminente, podría desencadenar alertas o incluso ajustar parámetros del control en el

sistema real de forma automática. Esta retroalimentación cierra el ciclo y convierte al gemelo digital en una herramienta activa para la operación.

Estos componentes trabajan en conjunto para lograr una réplica digital fiel y sincronizada. El objeto físico provee datos, el gemelo (sistema virtual) los simula, y mediante la conectividad se logra un flujo continuo de información entre ambos mundos, posibilitando análisis en tiempo real y la toma de decisiones informadas.

Beneficios de los gemelos digitales

Los gemelos digitales aportan numerosos beneficios a lo largo del ciclo de vida de un sistema mecatrónico, desde su diseño inicial hasta su operación y mantenimiento:

- **Mejora del diseño y desarrollo:** Permiten probar virtualmente componentes o configuraciones en etapas tempranas. Los ingenieros pueden simular el comportamiento bajo diversas condiciones de operación para optimizar el diseño, ajustando parámetros para lograr mayor precisión, estabilidad o eficiencia energética. Por ejemplo, es posible iterar en el diseño de un robot modificando masas, longitudes o algoritmos de control en el gemelo digital antes de construir nada físicamente, reduciendo la necesidad de prototipos costosos.
- **Verificación y validación virtual:** Facilitan pruebas extensivas de desempeño y seguridad de forma totalmente virtual. Se pueden identificar problemas de diseño o software antes de la construcción o implementación real. Esto reduce riesgos y acorta el ciclo de desarrollo, al disminuir la dependencia en pruebas físicas. Por ejemplo, fabricantes de robots pueden validar controladores en el gemelo digital para asegurarse de que no provoquen movimientos inestables, todo ello sin arriesgar el equipo real.
- **Optimización de operaciones en tiempo real:** Una vez desplegado, el gemelo digital refleja el estado del sistema en operación y permite monitorear y optimizar el rendimiento dinámicamente. Operadores pueden probar diferentes escenarios en el gemelo (ajustes de velocidad, trayectorias alternativas, cambios de producción) para encontrar condiciones óptimas sin interrumpir la producción real. Asimismo, la simulación en paralelo puede sugerir ajustes para mejorar la eficiencia o calidad en vivo.
- **Mantenimiento predictivo y diagnóstico de fallas:** Este es uno de los beneficios más citados. Al analizar datos históricos y en línea mediante el gemelo

digital, es posible predecir fallas antes de que ocurran. El gemelo puede simular el desgaste de componentes, monitorear vibraciones o temperaturas anómalas, y alertar sobre condiciones fuera de tolerancia. Así se habilita un mantenimiento proactivo: en lugar de reaccionar a averías, se programa el mantenimiento en el momento óptimo, minimizando tiempos muertos. Adicionalmente, si ocurre una falla imprevista, el gemelo digital ayuda a diagnosticar rápidamente la causa raíz comparando el comportamiento real con el simulado.

- **Entrenamiento y capacitación:** Los gemelos digitales proporcionan entornos virtuales realistas donde personal y algoritmos pueden entrenarse. Por ejemplo, operadores humanos pueden practicar con el gemelo de una célula robotizada, aprendiendo a reaccionar a situaciones de forma segura, ya que cualquier error no dañará equipos reales. Del lado de IA, se pueden generar datos sintéticos para entrenar modelos de visión o control bajo infinidad de escenarios simulados. Esto acelera la curva de aprendizaje y mejora la seguridad.
- **Virtual commissioning:** En la automatización industrial, es posible realizar la puesta en marcha virtual de líneas de producción completas mediante gemelos digitales. Antes de mover una sola pieza de equipo físico, se integra el PLC o sistema de control con el gemelo digital de la planta para depurar la lógica de control y asegurar que, cuando se energice la maquinaria real, todo funcione a la primera. Esto reduce errores costosos durante la instalación y arranque de sistemas complejos.

Por lo anterior, un gemelo digital bien implementado conlleva ahorros en costos, mejora de fiabilidad, incremento de rendimiento y reducción de riesgos durante todo el ciclo de vida del sistema. Empresas que han adoptado gemelos digitales reportan agilización en su innovación y una ventaja competitiva en la era de la Industria 4.0.

Retos en la implementación de gemelos digitales

A pesar de sus ventajas, desplegar un gemelo digital efectivo presenta retos importantes que deben considerarse:

- **Integración de datos y calidad de información:** Un gemelo digital es tan bueno como los datos que lo alimentan. Lograr integrar datos de múltiples sensores y fuentes (que a veces pueden venir en distintos protocolos o formatos) es un desafío técnico. Además, si los datos son incompletos, ruidosos o desactualizados, el gemelo digital no será fiable. Es fundamental garantizar

calibración de sensores, muestreos adecuados y una infraestructura IoT robusta para obtener datos de calidad.

- **Validación del modelo y exactitud:** Desarrollar un modelo virtual que realmente refleje el comportamiento físico con fidelidad no es trivial. A menudo se requiere un proceso de validación y ajuste de parámetros para alinear la simulación con la realidad. Esto puede involucrar técnicas de estimación de parámetros o identificación de sistemas usando datos experimentales. Por ejemplo, en el gemelo digital de un robot, será necesario ajustar factores como la fricción en las articulaciones, rigideces, eficiencias de motor, etc., para que las salidas simuladas (posición, consumo de energía, vibraciones) concuerden con las medidas reales dentro de un margen aceptable. Esta fase de ajuste y validación es crítica y puede requerir mucho tiempo y experiencia.

- **Costo y complejidad inicial:** Implementar gemelos digitales puede requerir inversión significativa en software, hardware y capacitación de personal. El desarrollo inicial de modelos de alta fidelidad y la infraestructura de conectividad puede ser costoso. Sin embargo, suele argumentarse que estos costos se compensan con creces a mediano plazo gracias a los ahorros en optimización y mantenimiento. De cualquier modo, las empresas deben planificar cuidadosamente el alcance del gemelo digital para equilibrar complejidad vs. beneficio: no siempre se necesita modelar cada detalle físico si no aporta valor al objetivo perseguido.

- **Experiencia multidisciplinar:** Un gemelo digital exitoso implica conocimientos en diversas áreas: modelado físico, simulación, programación, análisis de datos, e incluso ciberseguridad. Se requiere un equipo o profesional con formación especializada, capaz de entender tanto el dominio de la máquina (e.g., la robótica) como las herramientas de simulación y análisis. Esta escasez de talento o la necesidad de capacitar al personal es un reto organizativo.

- **Seguridad y privacidad:** Al conectar activos físicos a entornos digitales, surgen preocupaciones de ciberseguridad. Los datos transmitidos (p.ej. estado de máquinas) pueden ser sensibles; garantizar su protección frente a accesos no autorizados o sabotaje es esencial. Asimismo, si el gemelo digital tiene capacidad de enviar comandos al sistema real, deben implementarse medidas de seguridad para evitar acciones inapropiadas por errores o intrusiones.

- **Mantenimiento del gemelo a lo largo del tiempo:** Irónicamente, el gemelo digital mismo necesita mantenimiento. A medida que el sistema físico evoluciona (por modificaciones, desgaste, reconfiguraciones), el modelo virtual debe actualizarse para seguir siendo representativo. Esto implica tener procesos de revisión periódica, recalibración con nuevos datos, y ajustes en el modelo.

> Crear un gemelo digital no se limita a su construcción inicial; requiere supervisión continua para asegurar su fidelidad y relevancia a lo largo del tiempo.

Superar estos retos exige un enfoque planificado: comenzar con un alcance claro, implementar por fases, aprovechar modelos o datos existentes, e involucrar expertos en cada materia. Con las herramientas modernas (como las que provee MATLAB/Simulink) muchos de estos desafíos son más abordables, pero es importante tener expectativas realistas y estrategias para la integración y validación.

Gemelos digitales en robótica y automatización

La robótica es un campo particularmente beneficiado por la tecnología de gemelos digitales. Un robot (ya sea un brazo manipulador, un robot móvil o un sistema automatizado completo) es un sistema complejo donde convergen mecánica, electrónica y software; aquí los gemelos digitales ofrecen un laboratorio virtual ideal para experimentar sin riesgos. Veamos cómo se aplican los gemelos digitales en sistemas robóticos y de automatización, con algunos ejemplos reales:

- **Evaluación de estrategias de control:** En robótica es común probar distintos controladores, trayectorias o algoritmos de planificación. Los gemelos digitales de robots industriales y colaborativos se utilizan ampliamente para evaluar y predecir el comportamiento de manipuladores bajo diferentes estrategias de control antes de implementarlas en la máquina real. Por ejemplo, los ingenieros pueden ajustar ganancias de un controlador PID o probar un algoritmo de control por torque computado en el gemelo digital, observando cómo responde el brazo robótico virtual. Esto permite afinar el rendimiento (menores sobreoscilaciones, movimientos más suaves) y verificar la estabilidad sin poner en riesgo el robot físico ni al personal.
- **Optimización del diseño mecánico:** Durante el desarrollo de nuevos robots, un gemelo digital ayuda a tomar decisiones de diseño. Un caso destacado es el de Krones, empresa que desarrolló el gemelo digital de un robot trípode empaquetador de botellas (Figura 1) —parte de su sistema Robobox T-GS— usando Simulink y Simscape Multibody. Mediante simulaciones, pudieron visualizar fuerzas y momentos en cada componente, lo que sería difícil o imposible de medir en hardware. Gracias a este gemelo, optimizaron el diseño para mejorar el rendimiento, acortar el tiempo de desarrollo y reducir la necesidad de pruebas físicas extensivas.

Figura 1: Robot trípode para manipulación de paquetes

- **Mantenimiento predictivo en robots industriales:** Un gemelo digital de un robot puede emplearse para el mantenimiento predictivo integrando análisis de datos en tiempo real. Siguiendo con el ejemplo de Krones, una vez el sistema estuvo en producción, el gemelo digital se utilizó para apoyar el mantenimiento predictivo y diagnosticar problemas operativos, ejecutando simulaciones con datos registrados del robot real. De hecho, entrenaron un algoritmo de machine learning con datos de la simulación para detectar patrones de desgaste: el gemelo generaba datos sintéticos de fallas (p.ej. aumentando la fricción en una articulación) y con ello el algoritmo aprendía a identificar esas condiciones incipientes en la máquina real a partir de sus sensores. Así, el gemelo digital se convierte en una herramienta activa de mantenimiento, detectando fallas antes de que ocurran y permitiendo planificar paradas de servicio de forma conveniente.

- **Virtual commissioning de celdas robotizadas:** En automatización industrial, los gemelos digitales son clave para la integración de robots en líneas de producción. Por ejemplo, al diseñar una nueva célula con robots y bandas transportadoras, se puede crear un gemelo digital de toda la célula (robot, cintas, sensores) y conectar el PLC o sistema de control real en un entorno de simulación. De esta forma, se prueba la lógica de control y la interacción robot-máquina en el gemelo antes de la instalación física, afinando tiempos de ciclo y evitando colisiones o errores de secuencia. Esto reduce considerablemente el tiempo de puesta en marcha en planta y evita costosos contratiempos.

- **Robots autónomos y pruebas de escenarios:** En robótica móvil y vehículos autónomos, los gemelos digitales permiten testear escenarios de forma segura.

Un gemelo digital de un robot móvil (por ejemplo, un AGV o un dron) puede incluir su dinámica, sensores virtuales y un entorno simulado. Las empresas usan estos gemelos para probar el software de navegación autónoma, variando las condiciones (obstáculos, fricción del suelo, fallas de sensores) y observando cómo el sistema reacciona. Plataformas como Gazebo, CoppeliaSim o el propio MATLAB/Simulink se utilizan como motores de simulación en estos casos. El gemelo digital posibilita experimentar con cientos de escenarios que serían tardados o inseguros de reproducir en el mundo real, mejorando la robustez de los robots autónomos.

- **Ajuste fino de control y calibración:** Un ejemplo académico notable es el del robot humanoide iCub (Figura 2). Investigadores del *Istituto Italiano di Tecnologia* construyeron un gemelo digital de la cabeza y cuello del iCub en Simulink/Simscape para mejorar su control. Con este modelo de alta fidelidad, pudieron sintonizar automáticamente las ganancias de control y lograron mejorar significativamente el seguimiento de trayectorias del cuello. El gemelo permitía introducir masas y características no-lineales precisas y simular el sistema de accionamiento con detalle, lo que resultó crucial para diseñar algoritmos de control confiables que luego se aplicaron al robot real. Este caso demuestra cómo un gemelo digital ayuda a calibrar y validar algoritmos de control en robots complejos, alcanzando un rendimiento que sería difícil obteniendo solo mediante ensayo y error en hardware.

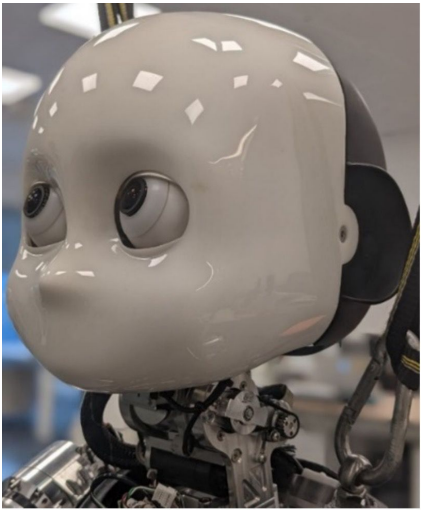


Figura 2: Robot iCub con tres grados de libertad en el cuello

Resumiendo, los gemelos digitales se emplean en robótica y automatización para: optimizar diseños de robots, validar y autoajustar controladores, planificar operaciones y producción, entrenar personal y sistemas de IA, y asegurar mantenimiento eficiente. Todo ello se traduce en robots más seguros, inteligentes y con mayor disponibilidad operativa. A continuación, pasaremos de la teoría a la práctica: ¿qué se necesita para crear un gemelo digital en MATLAB y cómo hacerlo paso a paso?

Requisitos de software y hardware

Para implementar un gemelo digital de un sistema robótico utilizando MATLAB/Simulink, es importante contar con las herramientas de software adecuadas (y eventualmente hardware de apoyo). A continuación, se listan los requisitos recomendados:

- **MATLAB y Simulink:** Forman la plataforma base. MATLAB permite análisis de datos, algoritmos y scripting; Simulink proporciona el entorno de modelado y simulación gráfica por bloques. Idealmente, usar la última versión disponible para asegurar compatibilidad con toolboxes recientes.
- **Simscape Multibody:** Toolbox esencial para modelar sistemas mecánicos multicuerpo en 3D (anteriormente conocido como *SimMechanics*). Con *Simscape Multibody* podemos construir el modelo físico del robot: eslabones como cuerpos rígidos, uniones articuladas (revolutas, prismáticas), actuadores, contactos, etc. Esta herramienta facilita crear modelos realistas de robots incluyendo efectos como gravedad, inercia, fricción y fuerzas de contacto. Se integra de forma nativa en Simulink y permite observar el movimiento del robot en una ventana de visualización 3D llamada *Mechanics Explorer*.

  En las versiones más recientes de MATLAB/Simulink, la herramienta de visualización *Mechanics Explorer* se denomina *Multibody Explorer*.

- **Simscape (básico y librerías adicionales):** La familia Simscape incluye paquetes para distintos dominios físicos. Por ejemplo, *Simscape Electrical* puede ser útil si se modelarán motores eléctricos y sus controladores de forma detallada, *Simscape Driveline* para engranajes y transmisiones, *Simscape Fluids* si el sistema tiene actuadores hidráulicos, etc. Estas librerías permiten incorporar componentes predefinidos (motores, resortes, amortiguadores, sensores, rectificadores, etc.) acelerando el desarrollo del modelo físico.
- **Robotics System Toolbox:** Complemento de MATLAB muy útil para robótica.

Proporciona modelos de robots predefinidos, algoritmos de cinemática (directa e inversa), planificación de rutas, y conectividad con *ROS (Robot Operating System)*. En nuestro contexto, sirve para calcular las relaciones cinemáticas o dinámicas de robots en MATLAB, y para integrar controladores de alto nivel. También facilita la importación de modelos de robots desde archivos URDF o de CAD (`importrobot`, `loadrobot`), creando objetos `rigidBodyTree`. Además, ofrece bloques de Simulink para ciertas funciones robóticas (p.ej. bloque de cinemática inversa). Aunque el *Robotics System Toolbox* no es estrictamente necesario para construir el gemelo digital (*Simscape Multibody* puede hacerlo todo físicamente), es muy útil para comparar resultados y diseñar/controlar el robot a nivel algorítmico.

> 💬 `rigidBodyTree` es una clase para representar la conectividad de cuerpos rígidos con articulaciones. Se usa en MATLAB para construir modelos de manipuladores robóticos.

- **Simulink Control Design / Control System Toolbox:** Si planeas diseñar controladores avanzados y sintonizarlos, estas herramientas ayudan a linealizar modelos, diseñar control clásico o moderno y optimizar ganancias. En particular, *Simulink Design Optimization* puede calibrar automáticamente parámetros del modelo contra datos reales (por ejemplo, ajustar una constante de fricción para que la respuesta simulada coincida con las mediciones).

- **Instrument Control/DAQ/ROS Toolbox (según el caso):** Si se va a conectar el gemelo digital con hardware real en tiempo real, podrías necesitar herramientas adicionales. Por ejemplo, *Data Acquisition Toolbox* o *Instrument Control Toolbox* si vas a leer sensores directamente desde MATLAB/Simulink (mediante interfaz serial, TCP/IP, UDP, etc.), o *ROS Toolbox* si integrarás el gemelo con un robot que usa ROS para sus sensores y actuadores. MATLAB también permite conexión a PLCs y dispositivos industriales vía MQTT, OPC UA y Modbus, entre otros protocolos, mediante la *Industrial Communication Toolbox*. Seleccionar estas herramientas dependerá de cómo lleguen los datos del sistema físico.

- **Hardware (opcional para *hardware-in-the-loop*):** En nuestros ejemplos usaremos mayormente simulación pura. Sin embargo, si tu objetivo es correr un gemelo digital en paralelo a un sistema real (en co-simulación o *Hardware-In-the-Loop*), podrías necesitar hardware como una tarjeta de adquisición de datos, un controlador embebido o incluso plataformas especializadas (*Simulink Real-Time* en un Speedgoat o MicroLabBox, por ejemplo). Para propósitos de este tutorial no es obligatorio, pero ten en cuenta que un gemelo digital corriendo en tiempo real con el sistema puede requerir computadoras industriales o

conexiones a los controladores del robot.

Resumiendo, el software como MATLAB/Simulink con Simscape es el núcleo para construir y simular el gemelo digital, mientras que toolboxes adicionales soportan necesidades específicas (modelado detallado, análisis de datos, comunicación con equipos). Asegúrate de tener instalados los complementos mencionados antes de continuar. En la siguiente sección comenzaremos el diseño y modelado del gemelo digital de un robot sencillo paso a paso.

Diseño del gemelo digital en MATLAB

A la hora de crear un gemelo digital, conviene seguir una metodología estructurada. A continuación, se describen las etapas principales para diseñar y desarrollar un gemelo digital utilizando MATLAB/Simulink, aplicándolas luego en un caso práctico.

Paso 1: Definir el objetivo y alcance del gemelo

Inicia planteando qué quieres lograr con el gemelo digital y qué tan detallado debe ser. ¿Se trata de simular un solo componente (p. ej. el motor de un eje) o el robot completo? ¿Buscarás principalmente afinar el controlador, predecir fallas mecánicas, o entrenar a un operador? Definir esto guía las decisiones de modelado. En nuestro caso de ejemplo, construiremos un gemelo digital de un brazo robótico simple de dos grados de libertad (2DOF) para demostrar los pasos fundamentales. El objetivo será simular su dinámica y control de posición, e ilustrar cómo podríamos detectar una falla (p. ej. aumento de fricción en una articulación) a través del gemelo.

Paso 2: Reunir datos y planificar el modelo

Reúne la información física necesaria: dimensiones del robot, masa de cada eslabón, características de motores (torque, inercia del rotor), reductores, límites de recorrido, etc. Si el robot físico ya existe, estos datos pueden venir de hojas de especificaciones o mediciones. Si aún está en diseño, se derivan del CAD o supuestos iniciales. Luego, decide el enfoque de modelado:

- **Modelado basado en física:** construirás el modelo con leyes físicas y parámetros conocidos (usando *Simscape Multibody*, ecuaciones, etc.). Este suele ser el camino en robótica, sobre todo al inicio o cuando no hay muchos datos reales disponibles.

- **Modelado basado en datos:** si cuentas con datos experimentales abundantes, podrías usar técnicas de machine learning o identificación de sistemas para crear modelos aproximados desde datos. En robótica, a veces se emplea para modelar dinámicas complejas o fricción no lineal a partir de mediciones.

En nuestro ejemplo, utilizaremos principalmente modelado físico con Simscape complementado con ajuste de parámetros a partir de algunos datos ficticios.

Paso 3: Crear el modelo físico en Simscape

Ahora sí, manos a la obra. Abrimos Simulink y creamos un nuevo modelo en blanco. Para usar *Simscape Multibody*, debemos agregar al modelo un bloque Mechanism Configuration (define la gravedad y las unidades) y partir de ahí para armar la estructura.

Nuestro robot de 2DOF consiste en: base fija, articulación 1 (rotacional) que sostiene el eslabón 1, luego la articulación 2 (rotacional, en el extremo del eslabón 1) que mueve el eslabón 2. Es un brazo planar articulado con solo dos segmentos o eslabones.

En *Simscape Multibody*, cada segmento o eslabón se representa con un bloque de cuerpo rígido que incluye sus propiedades de masa y geometría —podemos simplificar la geometría a un cilindro (Cylindrical Solid) o una caja (Brick Solid) para la visualización. Las uniones se representan como juntas revoluta (Revolute Joint) que conectan cuerpos y permiten rotación relativa. Además, agregaremos actuadores de torque en las articulaciones para mover el brazo, y sensores para medir los ángulos.

Al configurar cada cuerpo rígido, se le asigna su masa, tensor de inercia (si el eslabón es simple podemos aproximar como barra uniformemente densa) y punto de conexión. Por simplicidad, usamos longitud de 0.5 m y masa de 2 kg para Link1; longitud de 0.4 m y masa de 1 kg para Link2. Las juntas revolutas se parametrizan para rotar en el eje correcto (en un brazo plano podríamos definir el eje de rotación saliendo de la pantalla, es decir, alrededor de z si consideramos la pantalla en el plano XY).

Si dispones de un modelo CAD del robot, puedes usar la herramienta *Simscape Multibody Link* (plug-in para CAD) para exportar la geometría y estructura de un ensamblaje directamente a Simulink. Esto genera automáticamente los bloques de cuerpos, las uniones y las conexiones según el CAD, ahorrando tiempo y reduciendo errores. Para nuestro ejemplo manual, la estructura es pequeña y la construiremos desde cero para ilustración.

Una vez armado el modelo físico, colocamos sensores de Simscape en las articulaciones (para obtener posición y velocidad angular), y en los actuadores (para obtener el torque aplicado). Estas señales se enviarán a bloques estándar de Simulink para visualización o control.

Paso 4: Incorporar modelos de actuadores y controladores

Un gemelo digital completo debe incluir los actuadores que mueven al robot y, de ser relevante, los controladores que lo gobiernan. En nuestro robot 2DOF, supondremos que cada articulación es accionada por un motor eléctrico con un reductor. Podríamos modelar el motor de forma simplificada como un suministro de torque ideal (ya que en Simscape podemos aplicar un torque directamente en la junta), o detalladamente con *Simscape Electrical* (modelando voltaje, corriente, constante de motor, etc.). La solución más simple es colocar una fuente controlada de torque en cada Revolute Joint que reciba una señal desde otro bloque Simulink. El controlador será un simple controlador de posición: por ejemplo, un par de controladores PID (uno para cada articulación) que calculan el torque necesario para que el ángulo siga una referencia deseada.

En Simulink, añadimos dos bloques PID Controller (uno por articulación) en cascada a los actuadores de torque. La retroalimentación será la posición de la junta medida por el sensor, comparada contra una trayectoria deseada. Para ilustrar, digamos que queremos mover el brazo a un cierto ángulo objetivo en cada articulación. Alimentamos la diferencia de ángulo al PID y su salida es el torque que va al actuador de la junta.

Cabe destacar que en entornos industriales a veces el control de bajo nivel (motores) está fuera del gemelo digital si solo se quiere simular la planta física, pero aquí lo incluiremos para tener un gemelo digital cerrando el lazo como lo haría el robot real con sus controladores.

Paso 5: Simular y visualizar el gemelo digital

Con todo interconectado, procedemos a simular el modelo. Configuramos parámetros de simulación apropiados (p. ej. paso de tiempo fijo para real-time, o variable para mayor precisión si es offline). Al ejecutar, podemos observar la animación 3D del robot en la ventana *Mechanics Explorer*, verificando que los movimientos sean los esperados. Monitoreamos también gráficos de las variables clave: ángulos de las articulaciones, torques aplicados, etc. Esta es la etapa para verificar que el gemelo digital se comporta de forma aceptable. Quizá ajustemos aquí manualmente algunos parámetros: por ejemplo, agregar un ligero amortiguamiento (fricción viscosa) en las juntas si vemos oscilaciones no realistas, o limitar el torque máximo si queremos simular saturación del motor.

Si todo va bien, en este punto tenemos un gemelo digital funcional del robot, capaz de moverse con su física simulada y controlarse para seguir comandos. Vale la pena comparar su respuesta con lo que esperaríamos físicamente: ¿tarda cierto tiempo en alcanzar la posición? ¿Los torques máximos tienen sentido? ¿La gravedad afecta el brazo como cabría esperar (el brazo debería caer por su peso si no hay control activo)?

Esta verificación inicial suele descubrir si falta algún elemento en el modelo (por ejemplo, resortes de retorno, topes mecánicos) o si algún signo está invertido.

Paso 6: Refinamiento con datos reales (calibración)

Con el modelo básico ya funcionando, el siguiente nivel es ajustarlo usando datos reales del sistema físico, si están disponibles. Supongamos que tenemos mediciones del robot real, por ejemplo: tiempo de respuesta a un escalón de torque, o posición como función del tiempo en una maniobra. Podemos ejecutar el gemelo con las mismas entradas y comparar las salidas. Si hay discrepancias notables, procedemos a calibrar parámetros. MATLAB ofrece herramientas (como el estimador de parámetros de *Simulink Design Optimization*) para automatizar este ajuste. Por ejemplo, podríamos ajustar el coeficiente de fricción de las articulaciones para que la curva de decaimiento del movimiento libre en la simulación coincida con la observada en la realidad. Otro parámetro podría ser la inercia no modelada (quizá la del rotor del motor) que al incluirla corrige la respuesta dinámica.

Este proceso de identificación de parámetros convierte al gemelo digital en una copia aún más fiel del robot real. Al finalizar, tendremos confianza de que las salidas del gemelo bajo diversas entradas se alinean con un nivel aceptable de error respecto al sistema físico.

Paso 7: Análisis de escenarios y pruebas virtuales

Ahora utilizamos el gemelo digital para cumplir su propósito. Por ejemplo, podemos simular escenarios de falla: ¿qué pasa si de pronto aumenta la fricción en una articulación (como si un rodamiento estuviera fallando)? Podemos introducir ese cambio en el modelo (aumentando el coeficiente de fricción estática y viscosa en la junta correspondiente) y ver cómo varía la corriente del motor o si el controlador requiere más torque para la misma trayectoria. Esa "firma" identificada en la simulación puede servirnos para detectar esa falla en la vida real si notamos el mismo patrón en los datos de operación. Del mismo modo, podríamos inyectar una falla de sensor (por ejemplo, señal ruidosa o congelada) en la simulación para evaluar la robustez del control.

Otra prueba es variar parámetros para optimización: ¿y si duplicamos la masa de la carga que el robot mueve? ¿O si aumentamos la velocidad deseada del movimiento? El gemelo responde a estas preguntas rápidamente: tal vez descubramos que cierto movimiento rápido excede el torque de motor disponible, lo cual podríamos solucionar ajustando el perfil de movimiento, todo ello antes de implementarlo en el robot real.

Paso 8: Implementación y uso continuo

Finalmente, se integra el gemelo digital en el flujo de trabajo real. Esto puede tomar varias formas:

Ejecutar el gemelo digital en paralelo al robot real, consumiendo los datos de sensores en vivo y comparando en tiempo real el estado predicho vs. el medido. Por ejemplo, si el gemelo calcula que para cierta posición el motor debería consumir 2 A pero el sensor de corriente real lee 4 A, esa discrepancia puede indicar un problema (fricción extra, desalineación) y generar una alerta. Usar el gemelo digital como banco de pruebas virtual para cualquier cambio antes de aplicarlo físicamente. Si se quiere actualizar el software del robot con una nueva lógica, primero se prueba con el gemelo conectado al controlador para verificar que no haya sorpresas. Actualizar el gemelo digital conforme el sistema evoluciona. Si se reemplaza un motor por otro más potente, el modelo se modifica; si una articulación recibe lubricación y cambia la fricción, se recalibra; si se añade un sensor, se incorpora su modelado. Siguiendo estos pasos, hemos cubierto el proceso central de diseño y modelado de un gemelo digital de un robot sencillo. En la siguiente sección mostraremos un ejemplo práctico completo, aplicando estos principios y mostrando fragmentos de código y resultados de simulación concretos en MATLAB/Simulink.

Ejemplo práctico: Gemelo digital de un brazo articulado

Ahora pasemos a un ejemplo práctico paso a paso. Construiremos un gemelo digital para un robot manipulador sencillo: un brazo articulado de 2 grados de libertad (2DOF) montado en una mesa, similar a un pequeño robot de laboratorio. Este brazo tiene dos eslabones y puede moverse en un plano vertical (como un brazo plano que sube y baja, y extiende el antebrazo). Queremos modelar su dinámica y diseñar un control básico de posición, además de ilustrar cómo conectarlo con posibles datos de sensores.

Descripción del sistema físico: El robot está formado por un primer eslabón (brazo) de longitud $L_1 = 0.5\,\text{m}$ unido a la base con un motor en el "hombro", y un segundo eslabón (antebrazo) de longitud $L_2 = 0.4\,\text{m}$ unido con otro motor en el "codo". Las masas son $m_1 = 2\,\text{kg}$ y $m_2 = 1\,\text{kg}$, respectivamente, con distribución homogénea. Cada articulación tiene un actuador eléctrico con reductor capaz de suministrar hasta $50\,\text{N m}$ de torque. Hay sensores de posición (encoders) en cada articulación y, en este caso, supondremos sensores de par o corriente en los motores.

El objetivo del gemelo digital será simular los ángulos de las articulaciones dado un comando de referencia, y monitorear los esfuerzos. También veremos cómo podríamos identificar un incremento anormal de fricción en una articulación usando el gemelo.

Paso 1: Crear el modelo multicuerpo en Simulink

Iniciamos MATLAB y abrimos Simulink, creando un modelo vacío. Arrastramos bloques Solver Configuration y World Frame de *Simscape Multibody*. También colocamos un bloque Mechanism Configuration y lo configuramos con gravedad hacia $-y$ (si consideramos a y como vertical) o $-z$ según convenga. Usaremos unidades SI (metros, segundos, etc.).

Luego, insertamos los bloques de cuerpos rígidos y juntas:

Añadimos un bloque Revolute Joint para la primera articulación (el hombro). Conectamos su base al bloque World Frame (tierra) y su seguidor a un bloque Brick Solid que representará el primer eslabón. Configuramos el eje de rotación apropiado (como el brazo rota en un plano vertical, podemos definir el eje de rotación en z, perpendicular al plano del brazo). En el bloque Brick Solid (Link1) definimos sus propiedades: masa de $2\,\text{kg}$, centro de masa en su centro geométrico (a $25\,\text{cm}$ desde la base si consideramos el eje del cuerpo en el centro), y asignamos sus dimensiones (p. ej. barra de $50\,\text{cm}$ de longitud con sección transversal cuadrada de $10\,\text{cm} \times 10\,\text{cm}$). Al final de Link1, conectamos otro Revolute Joint (el codo) cuyo frame base se conecta al frame final del sólido Link1. Su seguidor irá a un segundo Brick Solid (Link2) con

masa de 1 kg y longitud de 40 cm. Configuramos Link2 similarmente, con su centro de masa a 20 cm del inicio. En la punta de Link2 podríamos añadir un pequeño Brick Solid —un sólido "dummy" de masa despreciable— solo para marcar el efector final (Tool). Esto es opcional.

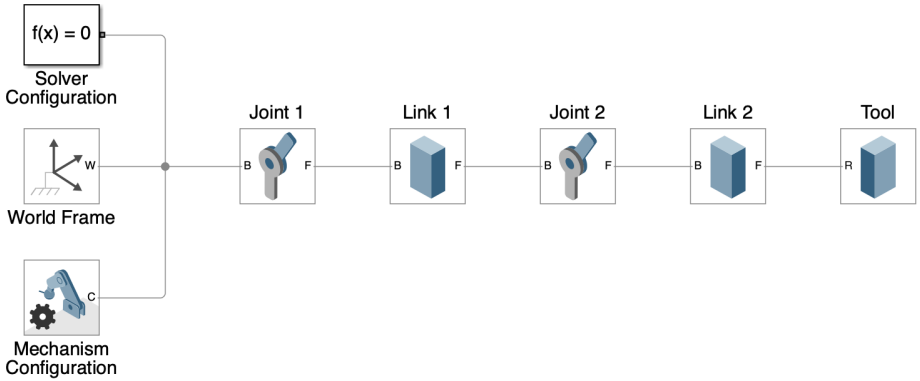


Figura 3: Diagrama de cuerpos múltiples del robot planar 2DOF

Asegúrate de conectar y enrutar correctamente los frames de coordenadas en *Simscape Multibody* (usando los puertos de los sólidos y las juntas). Si todo está conectado en serie, Simulink dibujará líneas que unen World Frame → Joint 1 → Link 1 → Joint 2 → Link 2 formando la cadena cinemática.

La visualización en *Mechanics Explorer* se obtiene pulsando Ctrl + D en Windows, o Cmd + D en macOS, o ejecutando una simulación desde Simulink. La vista 3D isométrica en la Figura 4 muestra que el brazo cuelga por su propio peso (de hecho, oscila con un movimiento pendular) debido a la gravedad ante la falta de un sistema de control.

Ahora integramos los actuadores y sensores:

Para aplicar torque en una Revolute Joint en *Simscape Multibody*, usamos los puertos físicos del bloque: habilitamos el puerto del actuador en las propiedades de la junta (Actuation > Torque > Provided by Input); esto expone un puerto donde podemos aplicar la señal física de torque. Alternativamente, podemos usar bloques External Force and Torque conectados entre los dos puertos (B y F) de cada revoluta para aplicar la señal de torque. De cualquier forma, usaremos un bloque Simulink-PS Converter para enviarle señales genéricas (es decir, señales obtenidas de bloques Simulink que

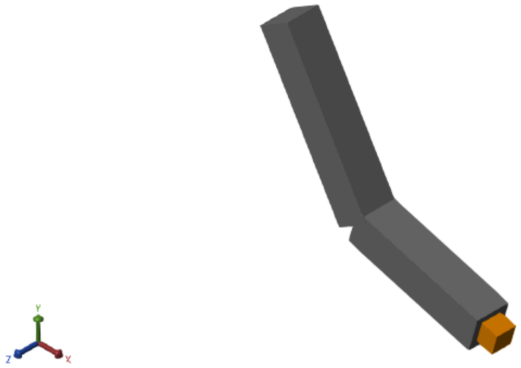


Figura 4: Vista isométrica del robot planar 2DOF en *Mechanics Explorer*

no pertenecen a Simscape; por ejemplo, torques fijos obtenidos de bloques Constant, o torques variables obtenidos de bloques PID Controller). Hacemos esto en ambas juntas y nombramos las entradas como `tau1_cmd` y `tau2_cmd`. Para medir los ángulos, habilitamos en cada Revolute Joint el puerto de medición para la posición (Sensing > Position). Esto añade un puerto de salida que conectamos directamente a un bloque PS-Simulink Converter para llevar la señal a bloques Simulink que no pertenezcan a Simscape. Alternativamente, podemos leer el estado de cada revoluta con un Transform Sensor entre sus dos puertos (B y F) habilitándolo para obtener la diferencia angular (Rotation > Angle) entre la salida y su referencia. De cualquier forma, nombramos `theta1` y `theta2` a las salidas de los sensores (en radianes) ya convertidas en señales Simulink mediante bloques PS-Simulink Converter. En este punto, la parte física está lista. Tendríamos en el diagrama bloques Link1, Link2, Joint1, Joint2, Tool, sensores y actuadores.

Paso 2: Añadir el controlador de posición en Simulink

Ahora pasamos al Simulink puro para agregar el controlador. El objetivo será controlar los ángulos de las articulaciones para seguir referencias deseadas (podrían ser constantes o trayectorias en función del tiempo). Usaremos controladores PID sencillos por articulación.

1. Insertamos dos bloques PID Controller (uno para cada junta). Configuramos en modo P o PI inicialmente, y luego ajustamos si es necesario.
2. Creamos entradas `theta1_ref` y `theta2_ref` (por ejemplo mediante bloques Constant o Signal Generator si quisiéramos definir un perfil de movimiento).

Para probar, por ejemplo, `theta1_ref` $= 30°$ (0.524 rad), `theta2_ref` $= 60°$ (1.047 rad).

3. Calculamos el error de posición restando la señal de ángulo medido (`theta1`) a la referencia (`theta1_ref`). Esa señal de error va al primer PID, cuya salida será un comando de torque (`tau1_cmd`).

4. Llevamos `tau1_cmd` a la entrada de torque de la primera articulación (mediante un Simulink-PS Converter para convertir la señal Simulink a física). Hacemos lo mismo para la segunda articulación con su PID.

5. Ajustamos las ganancias de los controladores PID. Una aproximación es calcular la inercia total vista por cada motor e intentar ganancias moderadas. También se puede hacer a mano: comenzamos con una ganancia proporcional baja y vamos aumentando hasta obtener una respuesta rápida sin mucha oscilación. Para la simulación, supongamos $K_p = 30$, $K_i = 20$, $K_d = 10$ del controlador PID en la primera articulación, y $K_p = 15$, $K_i = 10$, $K_d = 5$ en la segunda (ya que el segundo eslabón es más liviano). La constante N del filtro derivativo la podemos dejar en 60 en ambos controladores.

6. Limitamos la salida del PID a los torques máximos posibles. Los bloques PID tienen opción de saturación, o ponemos un bloque Saturation después del controlador (p. ej. $\pm 50\,\mathrm{N\,m}$).

7. Añadimos bloques Scope, XY Graph, u otro elemento de salida para visualizar {`theta1_ref`, `theta2_ref`} vs {`theta1`, `theta_2`}, torque vs tiempo, o cualquier gráfica que resulte útil a los propósitos del gemelo digital.

El diagrama completo, mostrado en la Figura 5, ahora incluye tanto la planta (bloques de Simscape) como el controlador (bloques genéricos de Simulink).

Paso 3: Simulación inicial y ajuste fino

Ejecutamos la simulación (con duración de 10 segundos) para ver cómo responde el gemelo digital.

¿Qué deberíamos observar? Idealmente, ambas articulaciones moverán el brazo desde la posición inicial (podemos suponer inicial en 0° para ambas) hacia 30° (0.524 rad) y 60° (1.047 rad) respectivamente, como se muestra en la Figura 6. En el *Mechanics Explorer* veremos el brazo elevarse y doblarse. Las gráficas de ángulos mostrarán una curva ascendente acercándose a los valores objetivo, posiblemente con cierto sobreimpulso y oscilación dependiendo de las ganancias.

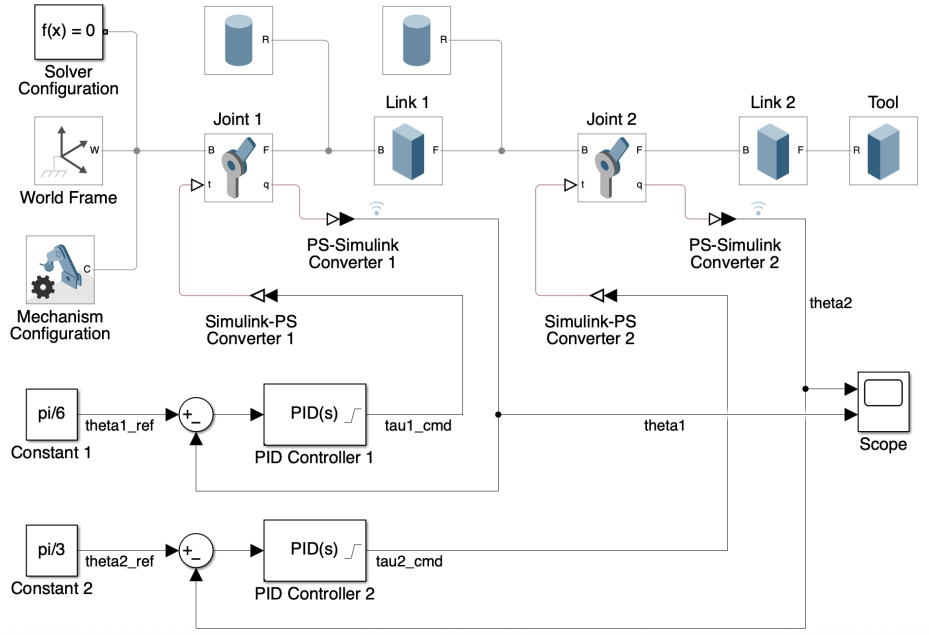


Figura 5: Gemelo digital del robot planar 2DOF usando Simscape

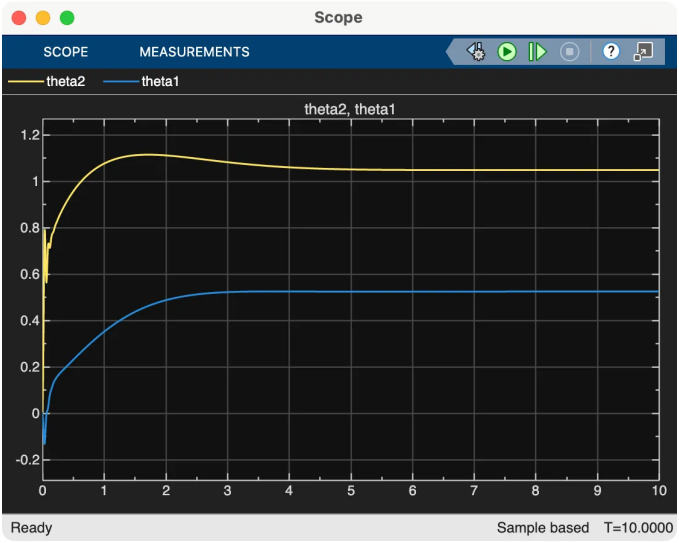


Figura 6: Respuesta transitoria del gemelo digital

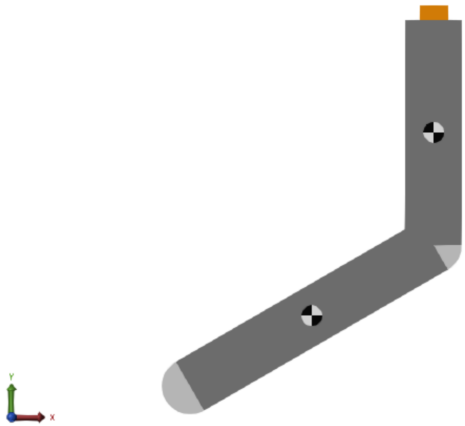


Figura 7: Vista frontal en el instante final de la simulación en *Mechanics Explorer*

Si la respuesta es muy lenta, aumentamos K_p. Si nunca alcanza el ángulo de referencia deseado o tarda mucho en alcanzarlo, aumentamos K_i. Si oscila mucho o se sobrepasa, incrementamos K_d o reducimos K_p. Ajustamos las ganancias hasta lograr un buen compromiso entre rapidez de respuesta y sobrepaso. Tras iterar, obtenemos la posición final deseada (Figura 7) con el asentamiento en un par de segundos sin sobrepasar el torque máximo.

También verificamos que no haya errores en simulación. Si el integrador da avisos de pasos muy pequeños o singularidades, podríamos revisar las configuraciones (p. ej., juntas a las que se deba especificar un estado, priority, etc.).

En este punto, ya tenemos un gemelo digital básico operativo.

Paso 4: Conexión con datos reales (sensorización)

Imaginemos que tenemos el brazo físico y podemos leer sus sensores en MATLAB (por ejemplo, vía ROS o DAQ). ¿Cómo integraríamos eso? Hay varias estrategias:

- **En tiempo real, en paralelo:** Podríamos alimentar nuestro gemelo con las entradas reales que va recibiendo el robot y comparar salidas. Por ejemplo, si el robot real está siguiendo la misma referencia de ángulos, podríamos leer los ángulos reales con un bloque de entrada (ROS Subscribe o DAQ In) y compararlos con los ángulos simulados. Cualquier discrepancia significativa indica que el modelo no predice algo (quizá viento o fricción extra). También

podríamos actualizar el estado inicial del gemelo con el estado actual del físico periódicamente.

- **Modo *Hardware-In-the-Loop* (HIL):** Ejecutar la simulación en una plataforma que interactúe con el controlador físico. Por ejemplo, el controlador de motores real piensa que controla un robot, pero en realidad está conectado al gemelo digital que simula la respuesta mecánica en tiempo real. Esto permite probar controladores reales frente a condiciones simuladas difíciles.

Para nuestro tutorial, supondremos que no tenemos el hardware conectado directamente. Sin embargo, mostraremos cómo usar datos registrados reales para calibración y diagnosis:

Supongamos que hicimos un experimento físico: movimos el brazo real y medimos el ángulo alcanzado y el torque del motor durante ese movimiento. Podemos cargar esas señales en MATLAB (por ejemplo, vectores o series de tiempo `theta_real` y `torque_real`). Con el gemelo, podemos replicar la misma entrada y obtener `theta_sim` y `torque_sim`.

Luego, compararíamos estas señales. Un código para hacer la comparación podría ser como el siguiente:

MATLAB

```
% Datos experimentales (ejemplo ficticio)
t_exp = 0:0.01:10;
theta1_exp = ...  % (vector de medición de encoder articulación 1)
theta2_exp = ...  % (vector de medición de encoder articulación 2)
tau1_exp = ...    % (vector de torque/corriente articulación 1)
tau2_exp = ...    % (vector de torque/corriente articulación 2)
% Ejecutar simulación con las mismas entradas
simOut = sim('gemelo_robot', 'StopTime','10');
theta1_sim = simOut.logsout.get('theta1').Values.Data;
% Comparar:
plot(t_exp, theta1_exp, 'r', t_exp, theta1_sim, 'b--');
legend('Ángulo \theta_1 real','Ángulo \theta_1 simulado');
```

Si observamos una diferencia significativa entre las mediciones y los valores simulados, podemos iterar el modelo. Por ejemplo, si la respuesta simulada es más rápida que la real, quizás a la simulación le falta fricción o inercia. Podríamos aumentar esos valores y volver a intentar hasta alinear las curvas. Este proceso puede automatizarse con *Simulink Design Optimization* para minimizar el error cuadrático entre `theta1_exp` y `theta1_sim` —y entre cualquier otro par de variables— variando parámetros específicos del modelo.

Paso 5: Detección de fallas mediante el gemelo digital

Veamos un uso concreto, la detección de una falla de fricción en la articulación `Joint2`. Con el gemelo calibrado, sabemos cuál es el torque requerido bajo condiciones normales para un cierto movimiento. Ahora, simulamos un escenario de falla suponiendo que el cojinete de la articulación `Joint2` se desgasta y la fricción se duplica.

En el gemelo, podemos duplicar el parámetro de fricción viscosa y estática en `Joint2`. Luego ejecutamos la simulación con el mismo perfil de movimiento de antes. Observaremos que, para lograr el mismo seguimiento de ángulo, el torque en `Joint2` simulado aumentará notablemente, y quizás la respuesta sea más lenta. Generamos así la firma de falla en el mundo virtual.

¿Cómo usamos esto en la realidad? Durante la operación normal, monitoreamos los datos reales: si empezamos a ver que el motor 2 está consumiendo más corriente de lo previsto por el gemelo digital sano para la misma tarea, es una señal de alerta. Implementaríamos un algoritmo comparando el gemelo en línea (sano) con los datos: esencialmente un observador digital. Alternativamente, entrenamos un modelo de machine learning con datos de gemelo sano *versus* gemelo con fallas. Luego, en vivo los datos reales pasan por ese modelo que clasifica si el patrón corresponde a "operación normal" o "falla incipiente en `Joint2`". Este es exactamente el enfoque utilizado en Krones: el gemelo digital proporcionó datos sintéticos para un algoritmo que ahora en operación detecta desgaste antes de que ocurra la falla.

Paso 6: Extensiones posibles

Por simplicidad, nuestro ejemplo se centró en un robot planar sencillo con control básico. En casos reales podrías extender esta metodología hacia:

- Modelos más complejos con contactos (por ejemplo si el robot interactúa con piezas o el entorno, usando bloques de Contact Force en Simscape).
- Inclusión de sensores virtuales avanzados, como cámaras simuladas o sensores LiDAR en un gemelo de robot móvil, usando por ejemplo conexiones con simuladores 3D externos.
- Integración con ROS: publicar el estado del gemelo como si fuera un robot real para aprovechar herramientas de ROS en el análisis.
- Uso de algoritmos de optimización para que el gemelo ajuste automáticamente parámetros de control (como se hizo con iCub o para optimizar consumo energético variando trayectorias).

Sin embargo, el enfoque fundamental permanece: modelar la física, validar con la realidad y luego explotar el gemelo para pruebas, diagnóstico o mejora continua.

Cierre

Hemos recorrido los aspectos básicos de los gemelos digitales aplicados a robótica y automatización, desde los conceptos hasta un ejemplo con MATLAB. Para recapitular, un gemelo digital es una potente herramienta que conecta el mundo real con el virtual, permitiendo mejorar diseños, anticipar problemas y optimizar la operación de sistemas mecatrónicos de forma segura y eficiente.

En la industria actual, vemos gemelos digitales realizando monitoreo inteligente de líneas de producción, donde cada robot y máquina tiene su avatar digital que alerta de anomalías y sugiere optimizaciones. También en la educación y la investigación, los gemelos digitales permiten experimentar con robots virtuales, acelerando el aprendizaje y la innovación. Empresas de automatización los usan para ofrecer a sus clientes servicios de mantenimiento avanzado y para entrenar algoritmos de inteligencia artificial con datos simulados antes de desplegarlos.

En cuanto a nuestro ejemplo de gemelo digital en MATLAB, demostramos cómo con las herramientas de Simulink y Simscape es viable construir un modelo detallado de un robot, integrando sus componentes mecánicos, electrónicos y de control en un solo entorno unificado. La ventaja de esta plataforma es que luego podemos llevar ese modelo a diversas aplicaciones: ejecutarlo en tiempo real en un equipo industrial, conectarlo a un PLC, desplegarlo en la nube para monitoreo remoto, etc., gracias a la flexibilidad de despliegue que ofrece MATLAB/Simulink.

Desde el ahorro de costos de desarrollo en *Krones* optimizando sus robots de empaque, pasando por la mejora de precisión en control del robot *iCub*, hasta el mantenimiento proactivo en brazos robotizados de fabricación automotriz, los gemelos digitales se han vuelto un habilitador clave de la llamada Industria 4.0. En sistemas autónomos, actúan como simuladores en vivo que permiten que un vehículo o robot “se pruebe a sí mismo” continuamente en un ambiente virtual antes de actuar, aumentando la seguridad.

Por supuesto, implementar un gemelo digital requiere inversión y conocimiento, pero como hemos explorado, las herramientas modernas han hecho esta tecnología mucho más accesible que antes. Para un ingeniero mecatrónico con conocimientos de MATLAB, los pasos para crear un gemelo digital básico están al alcance, y a partir de allí se puede escalar en complejidad.

En conclusión, los gemelos digitales representan una convergencia entre la simulación y la realidad que está transformando la ingeniería mecatrónica. Son un aliado para diseñar mejor, operar con mayor inteligencia y responder rápidamente a los desafíos en sistemas robóticos e industriales. Esperamos que este tutorial te haya brindado una visión clara y práctica de cómo empezar a construir y aprovechar gemelos digitales en tu propio campo de trabajo o estudio.

Ildeberto de los Santos Ruiz es originario de Tonalá, Chiapas (1973). Ingeniero en Electrónica, Maestro en Ciencias en Ingeniería Mecatrónica y Doctor en Ciencias de la Ingeniería por el Instituto Tecnológico de Tuxtla Gutiérrez (ITTG); Doctor en Automática, Robótica y Visión por la Universidad Politécnica de Cataluña. Miembro de la *Association for Computing Machinery* (ACM) y del *Institute of Electrical and Electronics Engineers* (IEEE). Miembro afiliado de la *International Federation of Automatic Control* (IFAC). Profesor de tiempo completo en el ITTG desde 1995, adscrito al Departamento de Ingeniería Eléctrica y Electrónica, Jefe de Proyectos de Investigación de Ingeniería Mecatrónica y Presidente del Claustro del Doctorado en Ciencias de la Ingeniería. Dirige proyectos de investigación en las áreas de diagnóstico y control inteligente, específicamente en robótica y en detección/localización de fugas en redes de distribución de agua.