Jornadas de Automática

Servidor OPC UA desacoplado: Hacia la interoperabilidad semántica en Industria 4.0

Rafael Aznar Estrada[a], Sergio Illana Rico[a], Alejandro Sánchez García[a], Ildefonso Ruano Ruano[b], Silvia Satorres Martínez[a], Elísabet Estévez Estévez[a*]

[a] *Dpto. Ingeniería Electrónica y Automática, A3-402, Universidad de Jaén, Paraje Las Lagunillas, s.n., 23071, Jaén, España.*
[b] *Dpto. de Ingeniería de Telecomunicación, A3-402, Universidad de Jaén, Paraje Las Lagunillas, s.n., 23071, Jaén, España.*

To cite this article: Aznar, R, Illana, S., Sánchez, A., Ruano, I. Satorres, S., Estévez, E. 2026. Decoupled OPC UA Server: Towards Semantic Interoperability in Industry 4.0. Jornadas de Automática, 47. https://doi.org/10.17979/ja-cea.2026.47.13791

Resumen

Este artículo propone una arquitectura de comunicación avanzada diseñada para superar la dependencia tecnológica (vendor lock-in) asociada a la implementación de servidores OPC UA embebidos en hardware propietario. Se presenta el desarrollo de un servidor OPC UA externo implementado en C#, completamente desacoplado de los Controladores Lógicos Programables (PLC). Además, esta propuesta permite dotar de conectividad estándar a controladores heredados (legacy) que carecen de soporte nativo. La solución propuesta utiliza el modelo de información de OPC UA para garantizar la interoperabilidad semántica de los datos industriales. El flujo de información se estructura en dos interfaces críticas: una capa upstream para la integración directa con clientes y sistemas de gestión empresarial (ERP/MES), y una capa downstream orientada a la adquisición de datos de planta. La arquitectura se valida experimentalmente mediante un cliente estándar y un Gemelo Digital en Unity, optimizado para la monitorización en tiempo real y el diagnóstico predictivo en la Industria 4.0.

Palabras clave: Comunicaciones Industriales, Gemelo Digital, Industria 4.0, Interoperabilidad Semántica, OPC UA.

Decoupled OPC UA Server: Towards Semantic Interoperability in Industry 4.0

Abstract

This paper proposes an advanced communication architecture designed to overcome the technological dependency (vendor lock-in) associated with embedded OPC UA servers in proprietary hardware. The development of an external OPC UA server implemented in C# is presented, completely decoupled from Programmable Logic Controllers (PLCs). Furthermore, this proposal enables standard connectivity for legacy controllers that lack native support. The proposed solution leverages the OPC UA information model to ensure the semantic interoperability of industrial data. The communication workflow is structured into two critical interfaces: a upstream layer for direct integration with client applications and enterprise management systems (ERP/MES), and a downstream layer oriented toward field data acquisition. The architecture's validity is experimentally demonstrated through its integration with a standard client and a Unity-based Digital Twin, optimized for real-time monitoring and predictive fault diagnosis in Industry 4.0 environments.

Keywords: Digital Twin, Industrial Communications, Industry 4.0, OPC UA, Semantic Interoperability.

1. Introducción

La irrupción de la cuarta revolución industrial, comúnmente denominada Industria 4.0, ha consolidado la digitalización transversal como el pilar fundamental para maximizar la eficiencia, flexibilidad y competitividad de los sistemas de manufactura contemporáneos. En este paradigma operativo, la evolución de los sistemas de automatización industrial está experimentando un cambio profundo, transitando desde las estructuras jerárquicas rígidas basadas en el modelo de la pirámide tradicional ISA-95 hacia arquitecturas descentralizadas, interconectadas y orientadas a servicios (SOA) (Hirsch et al., 2023). En este ecosistema, la

*Autor para correspondencia: eestevez@ujaen.es

interoperabilidad semántica se ha consolidado como el pilar fundamental para habilitar la convergencia real entre las tecnologías operativas (OT) y las tecnologías de la información (IT) (Sakurada et al., 2025).

Los Sistemas Ciberfísicos (CPS) y los Gemelos Digitales (Digital Twins - DT) emergen en este contexto como los habilitadores tecnológicos primordiales (Zami et al., 2025). Permiten la monitorización determinista, la simulación de escenarios, la optimización continua y el análisis predictivo en tiempo real mediante la sincronización ininterrumpida entre el entorno físico de la planta y su réplica virtual (Ortiz et al., 2026). La literatura reciente confirma que tratar a los gemelos digitales como agentes capaces de aprender y auto-organizarse exige arquitecturas de comunicación altamente dinámicas y eficientes (Alfaro et al., 2025). Para lograr esta cohesión ciberfísica, el estándar Open Platform Communications Unified Architecture (OPC UA, IEC 62541) lidera la transición, posicionándose como el protocolo de facto para asegurar una comunicación segura y escalable a través de todos los estratos organizacionales. OPC UA se consolida no solo como un protocolo de transporte, sino como un paradigma de modelado de información orientado a objetos que permite dotar de contexto, semántica e identidad conceptual a las variables físicas de la planta (Pajpach et al., 2025).

A pesar de los indudables beneficios arquitectónicos de OPC UA frente a su predecesor tecnológico (OPC Classic), la implementación práctica en plantas reales se enfrenta de forma recurrente a una problemática restrictiva severa: *el acoplamiento rígido a los entornos de hardware y software facilitados por los propios fabricantes de Controladores Lógicos Programables* (PLC). La estrategia comercial e ingenieril predominante ha consistido en integrar servidores OPC UA embebidos directamente en el firmware de las gamas más altas de sus equipos de control. Sin embargo, en equipos heredados de infraestructuras brownfield, en dispositivos de gama media-baja que carecen de soporte nativo, o en procesos críticos donde los recursos de la CPU del autómata deben reservarse para el control lógico y la seguridad, la ejecución de un servidor OPC UA embebido resulta inviable. Trabajos previos demuestran que la gestión interna de la pila completa de OPC UA (serialización XML/JSON, resolución de direcciones, cifrado) consume ciclos de CPU críticos, penalizando el tiempo de ciclo del PLC y degradando el determinismo del lazo de control (Tapia et al., 2023). Asimismo, este enfoque perpetúa el bloqueo del proveedor (vendor lock-in), acoplando el modelo semántico a herramientas de ingeniería propietarias (e.g., TIA Portal, Twincat).

Esta inflexibilidad choca frontalmente con los requisitos de los sistemas de supervisión y análisis perimetral de última generación, los cuales requieren un flujo masivo, bidireccional y sincronizado de información contextualizada, independiente de la topología del hardware. Bajo el esquema nativo actual, cualquier reconfiguración exige una reprogramación intrusiva del firmware del autómata (Schäfer et al., 2023). Para resolver este conflicto, la investigación actual se orienta hacia el desarrollo de arquitecturas híbridas en el Edge. Autores como Cao et al. (2021) proponen externalizar las capas semánticas fuera de los dispositivos de campo mediante pasarelas, un enfoque que también ha sido respaldado para facilitar la analítica de Big Data industrial de manera desacoplada (Hirsch et al., 2023). Sin embargo, la mayoría de estas soluciones se limitan a realizar un mapeo estático de variables (polling plano), perdiendo las ventajas conceptuales de las jerarquías de tipos y la herencia de OPC UA.

El objetivo principal de este trabajo es proponer, formalizar y validar una solución arquitectónica que independiza la capa semántica del hardware de control, proponiendo las pautas necesarias para desacoplar estructuralmente el servidor OPC UA de los entornos propietarios facilitados por los fabricantes. Para ello, se explotan sistemáticamente las bondades del modelo de información de OPC UA, diseñando y desarrollando en C# un servidor OPC UA externo e independiente, delegando exclusivamente la ejecución operativa de las tareas de automatización en un autómata comercial estándar.

Esta aproximación solventa la problemática en dos dimensiones:

- Metodología de desacoplamiento semántico (Nivel upstream): Resuelve la problemática para las capas superiores (SCADA, MES, simulación) ofreciendo un modelo puro. Se define un mecanismo de generación automática a partir de archivos NodeSets XML, liberando al PLC de la carga computacional.
- Mapeo de memoria desacoplado (Nivel downstream): Define una metodología estandarizada donde el PLC actúa solo facilitando valores brutos. Se desarrolla un esquema transparente que integra los parámetros de acceso al hardware (índices de byte y bit) como propiedades de los objetos de OPC UA, logrando sincronización bidireccional sin modificaciones intrusivas en el código de control.

Para validar la metodología, se ha desarrollado un caso de estudio sobre una planta neumática supervisada por un PLC Siemens S7-1200 usando el protocolo optimizado Snap7. Como entorno consumidor, se ha codiseñado una réplica virtual interactiva bajo el concepto de Gemelo Digital en el motor Unity 3D (Sánchez et al., 2024).

El documento se estructura de la siguiente manera: la Sección 2 profundiza en el modelo de información. La Sección 3 desgrana la propuesta del servidor OPC UA (upstream y downstream). La Sección 4 describe la implementación del cliente OPC UA. La Sección 5 expone la validación mediante el Gemelo Digital en Unity. Finalmente, la Sección 6 remarca los logros y conclusiones obtenidas.

2. Metodología de Modelado Semántico

Para lograr un desacoplamiento efectivo entre la semántica de los servicios de supervisión y el hardware de control físico, la metodología propuesta redefine la forma en que se estructuran y localizan los datos en el espacio de direcciones. En lugar de un mapeo plano e inflexible de variables aisladas, se define un marco de modelado jerárquico dividido en dos fases: la definición de meta-modelos abstractos y la inyección de propiedades estructurales de direccionamiento.

2.1. Modelo de Información OPC UA

La principal ventaja competitiva de OPC UA reside en su extenso modelo de información (OPC 10000-5). Esta infraestructura, puramente orientada a objetos, permite

representar digitalmente cualquier sistema, máquina o proceso físico mediante un espacio de direcciones (*Address Space*) jerárquico y estructurado.

Topológicamente, este modelo se construye como una red en malla basada en dos elementos elementales: Nodos (*Nodes*) y Referencias (*References*). El estándar define diversas clases de nodos (*NodeClasses*) fundamentales para la representación semántica (véase Figura 1):

- Objetos (Objects): representan entidades lógicas o elementos de hardware tangibles (e.g., un motor o cilindro neumático). Actúan como contenedores organizativos y carecen de valor numérico intrínseco.
- Variables: repositorios de datos propiamente dichos. Almacenan un valor fluctuante en el tiempo (como una temperatura o el estado de un sensor) asociado a un tipo de dato subyacente (*DataType*).
- Métodos (Methods): Funciones u operaciones invocables de forma remota por un cliente. Permiten evolucionar desde un esquema de control basado en el sondeo cíclico (*polling*) hacia un paradigma eficiente de ejecución por eventos.
- Tipos (ObjectTypes / VariableTypes): plantillas maestras o definiciones abstractas de las que heredan las instancias físicas individuales, garantizando la homogeneidad estructural a gran escala.

Por su parte, las Referencias (*ReferenceTypes*) actúan como las aristas del grafo que interconectan los nodos. Estas permiten establecer tanto relaciones jerárquicas (como HasComponent para denotar pertenencia, o HasProperty para características inmutables) como relaciones no jerárquicas (como HasTypeDefinition, que vincula una instancia física concreta con su plantilla abstracta de origen).

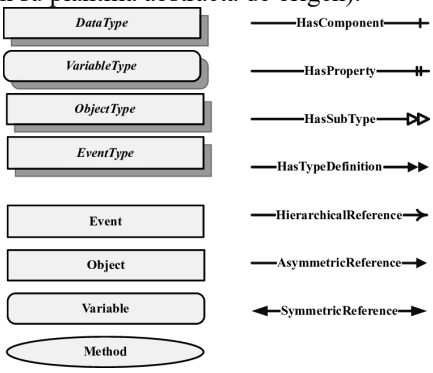

Figura 1: Repr. gráfica del léxico del espacio de direcciones OPC UA.

Este trabajo hace uso de dicha flexibilidad de modelado como eje principal para lograr un desacoplamiento efectivo entre el servidor y el PLC.

2.2. Abstracción de Hardware

En la aproximación arquitectónica propuesta, el modelo de información no se emplea de forma pasiva, sino que se instrumenta de forma activa. Para lograr un desacople efectivo en el nivel *downstream*, la metodología se fundamenta en la definición de un tipo de variable estructural denominado *Address*. En lugar de codificar de forma rígida (hardcoding) las direcciones en el firmware del PLC o en el código fuente del servidor, este tipo de variable encapsula los metadatos de configuración que instruyen algorítmicamente al servidor externo.

La Figura 2 presenta Definición del Tipo de variable Address siguiendo modelo información OPC UA. Se compone de propiedades esenciales como *byte* y *bit*. Al asignar este tipo de variable como una referencia HasProperty a los nodos operativos del modelo OPC UA, el motor superior (*upstream*) analiza el archivo XML instanciándolos, mientras que el motor inferior (*downstream*) lee dinámicamente estas coordenadas y mapea los punteros hacia la sintaxis del protocolo de acceso facilitado por el fabricante de PLCs (e.g. Snap 7, ADS), garantizando un aislamiento total y limpio del hardware físico.

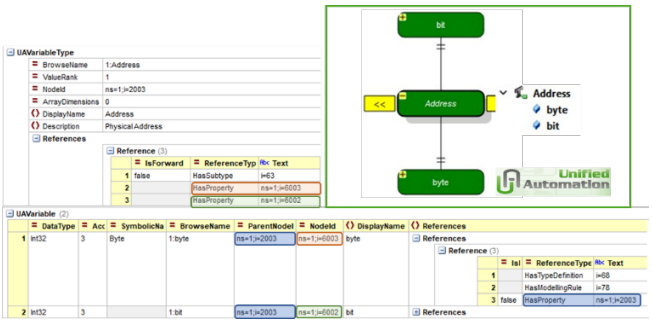

Figura 2: Definición del Tipo de variable Address

2.3. Definición de Tipos de Objetos:

El presente trabajo propone como estructuración semántica modelar las entidades físicas a través de diferentes clases de tipos de objetos (*ObjectTypes*). Esta abstracción clasifica los elementos de planta según su naturaleza operativa y su flujo de datos:

Objetos de Salida: Representan indicadores luminosos o actuadores simples cuyo estado lógico es dictado exclusivamente por el programa interno del autómata. El cliente OPC UA se limita a un acceso de solo lectura para monitorizar su valor sin alterarlo. La Figura 3 presenta el Tipo de objeto *TLed* que tiene dicha finalidad.

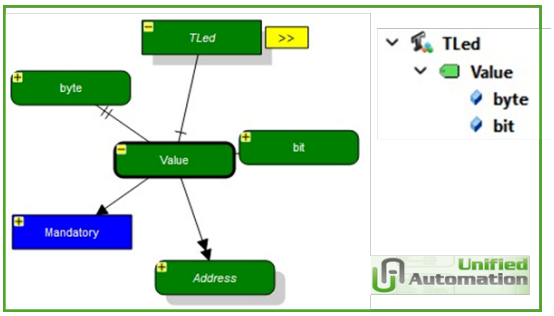

Figura 3: Tipo de Objeto OPC UA para caracterizar salidas digitales del PLC. Modelado en UAModeler.

Objetos de Entrada: Representan dispositivos físicos (e.g., botones o interruptores) que introducen consignas al sistema. Estas variables exigen un nivel de acceso bidireccional (lectura y escritura), permitiendo que tanto el operador en planta como el cliente digital puedan interactuar y alterar su estado (véase Figura 4).

Objetos Compuestos (basados en estado): Representan equipos de mayor complejidad cinemática que agrupan múltiples variables interrelacionadas. Su definición se articula a través de un "estado" general que hereda de tipos base, combinando simultáneamente componentes de entrada (órdenes de control hacia la máquina) y componentes de salida

(sensores de posición desde la máquina). La Figura 5 presenta como ejemplo un vástago de doble efecto. Se caracteriza por su estado a través de dos variables (retraído y expandido) y su control a través de acciones a una electroválvula (expandir y contraer).

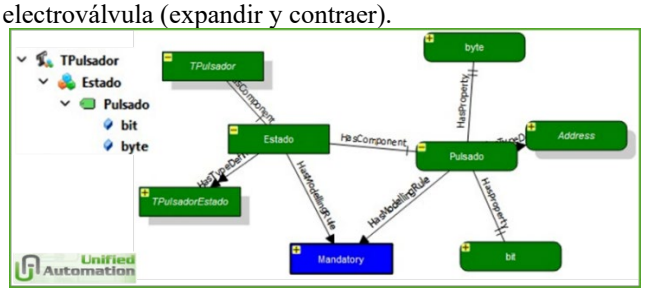

Figura 4: Tipo de Objeto definido para caracterizar las entradas digitales del autómata

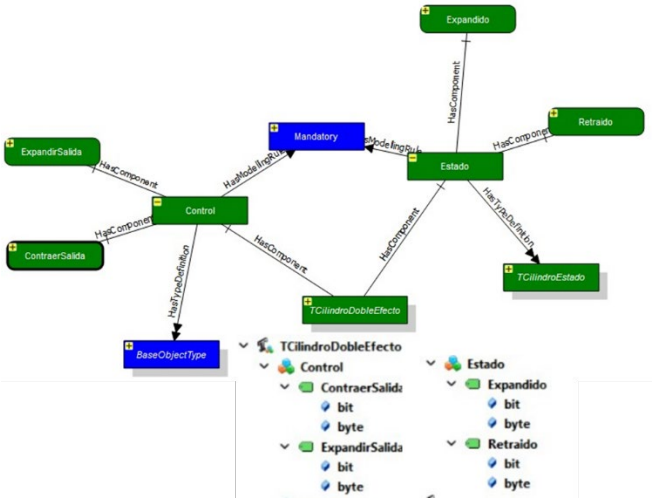

Figura 5: Ejemplo de definición de Tipo de Objeto compuesto.

Al compilar esta estructura abstracta en un archivo de intercambio estándar NodeSet XML (específicamente bajo el esquema *NodeSet2.xml*), el modelo de información se vuelve completamente portable. Cualquier motor de comunicación perimetral asíncrono puede explorar el árbol semántico (browsing), extraer los metadatos *byte* y *bit* de una variable e interactuar directamente con la memoria del PLC de forma agnóstica al fabricante.

3. Propuesta del Servidor OPC UA

La heterogeneidad del hardware industrial exige que el middleware sea flexible. La propuesta central se basa en un servidor OPC UA íntegramente desarrollado en C# bajo el entorno.NET.

3.1. Nivel Upstream: Análisis Semántico y Exposición

El nivel *upstream* gobierna la comunicación estandarizada hacia el exterior. Los siguientes pasos garantizan dicho desacople:

1) Inicialización y Endpoints: La aplicación inicializa la clase base definiendo un endpoint (e.g., opc.tcp://<IP>:4840), configurando directrices de seguridad (certificados X.509) y un espacio de nombres único.
2) Despliegue Dinámico XML (Parsing): Se elude la rigidez del código hardcoded mediante un algoritmo que lee y deserializa archivos *NodeSet2.xml*.
3) Clonación Jerárquica e Instanciación: Al encontrar la orden de creación de un equipo real (<UaObject>), el servidor ejecuta una clonación profunda de estructuras abstractas. Un filtro evita la duplicación errónea, asegurando que propiedades como *byte* permanezcan anidadas como HasProperty.
4) Métodos Genéricos: Para evitar el polling de escritura desde el cliente, la capa expone Métodos ejecutables que delegan la lógica a la capa *downstream*.

3.2. Nivel Downstream: Adquisición Física

Este apartado detalla la innovación operativa central de la arquitectura: la gestión asíncrona y masiva de las transacciones de lectura y escritura en la memoria del dispositivo físico, de forma totalmente transparente para el cliente superior. Para ello, es requisito indispensable que el autómata disponga de un servicio de comunicación habilitado, ya sea mediante protocolos estandarizados como Modbus, o a través de interfaces nativas como Snap7 o ADS. Este trabajo se centra específicamente en los dispositivos del fabricante Siemens.

Validación y Arquitectura Downstream. Para garantizar latencias mínimas sin dependencias privativas, se ha integrado la librería de código abierto Snap7 (compatible con Siemens S7-300/400/1200/1500). Al emular un Procesador de Comunicaciones (CP) mediante datagramas S7 nativos (ISO-on-TCP, RFC 1006), Snap7 permite a la CPU del autómata despachar los intercambios de memoria de forma totalmente transparente, sin requerir código de comunicaciones adicional en su ciclo de ejecución.

Gestión de Concurrencia Computacional. Para conciliar el entorno multihilo del servidor con el determinismo del PLC, el acceso al socket de red se sincroniza mediante un cerrojo de exclusión mutua (PLC_LOCK). Esto garantiza el acceso exclusivo de un único hilo al puerto TCP, evitando la corrupción de tramas, la saturación de subprocesos (thread starvation) y caídas de conexión.

Ciclos Operativos en Siemens (S7-1200/1500). El flujo downstream procesa las coordenadas de enrutamiento (byte y bit) inyectadas en el modelo mediante dos operaciones fundamentales:

- Ciclo de Lectura: Tras adquirir el PLC_LOCK, un hilo en segundo plano extrae el byte completo de la memoria del autómata y aplica un enmascaramiento de bits para aislar el valor atómico y actualizar el servidor.
- Ciclo de Escritura: Al no admitir la sobrescritura directa de bits aislados, se ejecuta una secuencia Read-Modify-Write. Reteniendo el mutex, el sistema lee el byte vigente, modifica el bit objetivo mediante operadores lógicos (OR / AND NOT) y transmite el byte reconstruido al PLC.

Este enfoque consolida un desacoplamiento total: el autómata ejecuta su lazo de control sin sobrecarga de IT, mientras que las aplicaciones externas consumen un modelo de información orientado a objetos limpio y seguro.

4. Cliente OPC UA

El diseño de un cliente OPC UA generalista debe ser completamente independiente de su aplicación final, ya sea un sistema SCADA, MES/ERP o un simulador 3D. Su función principal es gestionar la conexión y asegurar un canal bidireccional estable bajo la norma IEC 62541, consumiendo

los datos sin realizar suposiciones previas sobre la topología física de la planta.

4.1 Gestión Integral del Ciclo de Vida y Seguridad

La implementación del cliente en C# exige configurar los mecanismos de seguridad del estándar. Para ello, se instancia el objeto *ApplicationConfiguration*, donde se parametrizan tres aspectos críticos:

- Identidad de la aplicación: Define la URI y el identificador global del software.
- Políticas criptográficas: Configura los niveles de seguridad, algoritmos de cifrado y reglas de validación de certificados para redes aisladas (Intranet OT).
- Parámetros de transporte: Establece cuotas y tiempos límite (como MaxMessageSize) para evitar desbordamientos de búfer ante ráfagas masivas de datos.

Con la configuración lista, el método *ConnectToServerAsync()* negocia el canal criptográfico más robusto y consolida la sesión. Para garantizar la estabilidad a largo plazo, se implementan rutinas de desconexión limpias (como *OnDestroy()* o *DisconnectFromServer()*). Estas funciones eliminan las suscripciones y cierran la sesión formalmente, permitiendo al recolector de basura (Garbage Collector) de .NET liberar la memoria y evitar conexiones huérfanas en la red.

4.2 Lógica de Suscripciones y Gestión Asíncrona de Nodos

Establecida la sesión, el cliente explora dinámicamente el espacio de direcciones mediante rutinas de rastreo recursivo (como *LoadDevices*). Partiendo del nodo raíz (*ObjectsFolder*), el motor navega por las referencias del árbol, extrayendo el identificador único (*NodeId*) de cada variable para construir un mapa en la memoria local.

Para optimizar la red y evitar la sobrecarga del sondeo continuo (*polling*), el cliente utiliza un esquema basado en eventos (*publish-subscribe*). Registra una suscripción unificada mediante *MonitoredItems*, logrando que el servidor remoto solo transmita datos cuando el valor, la calidad del sensor (*StatusCode*) o la marca temporal sufren un cambio real. Al recibir esta notificación asíncrona, .NET invoca inmediatamente el evento *OnVariableChanged*.

Por último, para enviar órdenes a la planta física, el cliente no altera variables de control directamente. En su lugar, utiliza la función *CallUaMethodAsync()*. Ante un comando, el cliente empaqueta los parámetros y llama al NodeId de un método alojado en el servidor, delegando en este toda la gestión del autómata y recibiendo la confirmación del estado de forma totalmente asíncrona.

5. Caso de Uso: Gemelo Digital

Para evaluar la viabilidad y el rendimiento de la metodología de desacoplamiento semántico propuesta, se ha implementado un caso de estudio sobre una planta piloto (Ver Figura 6).

La planta física está integrada por los siguientes elementos distribuidos en campo:

- Actuador lineal neumático: Cilindro de doble efecto encargado del posicionamiento, equipado con sensores magnéticos de final de carrera para detectar las posiciones extremas (Retraído y Expandido).
- Cuadro de mando local: Equipado con pulsadores físicos para el control manual y pilotos LED para la indicación de estados operativos y fallos.
- Autómata Programable (PLC): Un controlador Siemens S7-1200 (CPU 1214C) dedicado exclusivamente a ejecutar la lógica de seguridad de bajo nivel y los interbloqueos de los actuadores.

Figura 6: Planta Real

Para garantizar un desacoplamiento total, el PLC carece de servidores OPC UA nativos o bloques de comunicación industrial en su configuración; su programa se limita a mapear de forma directa las E/S físicas hacia marcas de memoria interna. Toda la inteligencia semántica y la estructura orientada a objetos operan de manera exclusiva en un servidor autónomo externo, desplegado en un nodo de computación perimetral (Edge) dentro de la misma red local.

La finalidad de este Gemelo Digital no es actuar como un mero sistema de monitorización visual pasiva (Monitoring Twin). Su diseño y código base están orientados específicamente a constituir un sistema ciberfísico avanzado para la detección y el diagnóstico de anomalías lógicas y operativas en la planta.

Para ello, se ha desplegado un servicio perimetral de monitorización del tiempo de ciclo que calcula los tiempos de respuesta de los actuadores desde fuera del PLC, liberando a la CPU del autómata de operaciones de temporización analítica y del manejo de cadenas de texto complejas.

El escenario de validación opera bajo la siguiente secuencia temporal frente a contingencias:

- Petición asíncrona: Desde la interfaz gráfica del cliente se invoca el método ExpandirSalida del cilindro semántico.
- Acceso a planta: El servidor externo recibe la orden, adquiere el cerrojo de exclusión mutua (PLC_LOCK) y escribe el bit en la memoria del PLC mediante Snap7, iniciando simultáneamente un contador de alta precisión para medir el tiempo de tránsito físico.
- Diagnóstico de anomalías: Si se produce un fallo mecánico o una caída en la presión del aire, el cilindro no alcanza su posición final en el tiempo estimado. Al superarse un umbral crítico (e.g., 5 segundos) programado fuera del PLC, el servicio perimetral bloquea la maniobra por seguridad. Acto seguido, el servidor escribe mediante Snap7 en el bit correspondiente para encender un piloto de alarma físico y, en paralelo, dispara una alerta visual roja en la interfaz virtual.

El gemelo virtual se ha desarrollado sobre el motor de desarrollo Unity, actuando como un cliente OPC UA que interactúa de manera bidireccional con el servidor perimetral.

Figura 7: Modelo Digital en Unity

Cuando el cilindro de la planta real se desplaza correctamente dentro de las ventanas de tiempo normales, el servidor detecta el cambio de bit en su ciclo de lectura asíncrono y actualiza la variable Expandido en el espacio de direcciones.

El modelo en Unity, al estar suscrito a dicho nodo bajo el esquema publish-subscribe, recibe la notificación asíncrona mediante eventos TCP. A través de un despachador de hilos (thread dispatcher), Unity procesa el cambio con latencia de nanosegundos y renderiza el movimiento del gemelo virtual en la pantalla de forma fluida y en tiempo real.

Esta arquitectura confirma una disociación operativa total: el autómata de campo continúa ejecutando su bucle de escaneo lógico básico sin conocimiento algorítmico del estrato superior de IT, mientras el Gemelo Digital en Unity gestiona el diagnóstico y la representación orientada a objetos de forma segura y transparente.

6. Conclusiones

Este artículo ha presentado una metodología para el despliegue de servidores OPC UA externos y agnósticos, basada en la interpretación de NodeSets XML y una sincronización asíncrona de bajo impacto. El trabajo supera las limitaciones de rendimiento de los servidores embebidos nativos, validando la propuesta en un caso de estudio industrial real.

El análisis comparativo frente al enfoque del fabricante revela ventajas fundamentales:

Preservación del determinismo del PLC: Externalizar la gestión integral de la pila del protocolo libera a la CPU del autómata de tareas informáticas ajenas al control, evitando penalizaciones en su tiempo de ciclo físico.

Eficiencia de red: Reemplazar el polling continuo por una estrategia de escritura dirigida por métodos minimiza drásticamente el tráfico industrial, generando tramas de red únicamente en el instante de la invocación.

Eliminación del bloqueo del proveedor (vendor lock-in): Cualquier modificación del modelo de información se resuelve editando el archivo XML subyacente, sin necesidad de reprogramar el firmware ni de utilizar entornos propietarios (e.g., TIA Portal).

En definitiva, la arquitectura demuestra que es posible conjugar una alta riqueza semántica con un rendimiento óptimo en el nivel de campo. La integración final mediante un cliente asíncrono y el patrón Dispatcher ratifica la viabilidad de alimentar entornos gráficos tridimensionales en tiempo real, garantizando una ejecución segura y libre de conflictos multihilo.

Agradecimientos

Esta investigación ha sido financiada parcialmente por el Ministerio de Ciencia e Innovación de España a través del proyecto del Plan Nacional PID2023-150832OB-I00; y por la Junta de Andalucía a través del proyecto DGP-PIDI-2024-01419.

Referencias

Alfaro-Viquez, D.; Zamora-Hernandez, M.; Fernandez-Vega, M.; Garcia-Rodriguez, J.; Azorin-Lopez, J. A Comprehensive Review of AI-Based Digital Twin Applications in Manufacturing: Integration Across Operator, Product, and Process Dimensions. Electronics 2025, 14, 646. https://doi.org/10.3390/electronics14040646.

Cao, K., Hu, S., Shi, Y., Colombo, A. W., Karnouskos, S., Li, X., 2021. A Survey on Edge and Edge-Cloud Computing Assisted Cyber-Physical Systems. IEEE Transactions on Industrial Informatics 17(11), 7806-7819. DOI: 10.1109/TII.2021.3073066.

Digital Twin Consortium, 2021. Definition of a Digital Twin. The Digital Twin Consortium.

Hirsch, E., Hoher, S., Huber, S., 2023. An OPC UA-based industrial Big Data architecture. Journal of Systems Architecture, 134, 102797. DOI: 10.1016/j.sysarc.2022.102797

ISO, 2021. ISO 23247-1. Automation systems and integration - Digital twin framework for manufacturing. Part 1: Overview and general principles. ISO Publications.

Ortiz, J. S., Andaluz, V. H., Carvajal, C. P., 2026. Real-Time Digital Twin Architecture for Immersive Industrial Automation Training. Sensors, 26(7), 2023. DOI: 10.3390/s26072023

Pajpach, M., Pribiš, R., Drahoš, P. et al. Methodology and pilot implementation of modular educational-development platform for Interoperable Digital Twin. Sci Rep 15, 28020 (2025). https://doi.org/10.1038/s41598-025-13579-y.

Sakurada, L., et al., 2025. Ten Years of Asset Administration Shell Developments: Research Opportunities and Adoption Challenges. IEEE Access, 13, 127725.

Sánchez, A., Illana, S., Casado, P., Ruano, I., Estévez, E., 2024. Aproximación basada en Unity para el modelado digital de sistemas de automatización. Jornadas de Automática, 45. DOI: 10.17979/ja-cea.2024.45.10897

Schäfer, G., Kozlica, R., Wegenkittl, S., Huber, S., 2023. An Architecture for Deploying Reinforcement Learning in Industrial Environments. Lect Notes Comput Sci. DOI: 10.1007/978-3-031-25312-6_67

Tapia, E., Sastoque-Pinilla, L., Lopez-Novoa, U., Bediaga, I., López de Lacalle, N., 2023. Assessing Industrial Communication Protocols to Bridge the Gap between Machine Tools and Software Monitoring. Sensors, 23(12), 5694. DOI: 10.3390/s23125694

Zami, M. B. A., et al., 2025. Digital Twin in Industries: A Comprehensive Survey. IEEE Access, 13, 47297.