XXI Simposio CEA de Control Inteligente
24-26 de junio de 2026, Salamanca

Integración de agentes basados en modelos de lenguaje en sistemas SCADA mediante el protocolo MCP: arquitectura y consideraciones de diseño

Gómez-González, P.-P.[a,b,*], García Fischer, A.[a], Freire-Mahía, N.[a], Michelena, Á.[a], Rodríguez-Olmo, D.[b], Jove, E.[a], Calvo-Rolle, J. L.[a]

[a] *Universidade da Coruña, Research Group CTC, CITIC, Departamento de Enxeñaría Industrial, Campus de Ferrol, 15403 Ferrol, A Coruña, Spain.*
[b] *Técnicas de Soft, S.A., Polígono de Pocomaco, 5ª avenida, parcela E-18, 15190 A Coruña, Spain.*

To cite this article: Gómez-González, P.-P., García Fischer, A., Freire-Mahía, N., Michelena, Á., Jove, E., Calvo-Rolle, J. L. 2026. Integration of LLM-based agents into SCADA systems through the MCP protocol: architecture and design considerations. XXI Simposio CEA de Control Inteligente, Salamanca, España.

Resumen

La irrupción de los modelos de lenguaje de gran tamaño (LLM) ha abierto nuevas vías para la interacción con sistemas industriales complejos, en particular con plataformas SCADA. Este trabajo presenta una propuesta de arquitectura para la integración de agentes basados en LLM con sistemas SCADA mediante el protocolo abierto MCP (Model Context Protocol), comparándolo con la alternativa de las llamadas *skills*, instrucciones en lenguaje natural que orquestan herramientas preexistentes. Se identifican las limitaciones estructurales del uso de herramientas genéricas de bajo nivel, especialmente la alucinación de identificadores en cadenas largas y repetitivas, y se justifica el diseño de herramientas específicas de alto nivel. Se describe una arquitectura modular replicable y se reportan los resultados obtenidos en un caso de estudio industrial, donde la propuesta permite reducir significativamente el número de invocaciones requeridas y aumentar la fiabilidad de las respuestas frente al enfoque basado en *skills*.

Palabras clave: Interfaces de lenguaje natural, Desarrollo de sistemas asistentes, Agentes

Integration of LLM-based agents into SCADA systems through the MCP protocol: architecture and design considerations

Abstract

The rise of large language models (LLMs) has opened new avenues for interacting with complex industrial systems, particularly SCADA platforms. This work presents an architectural proposal for the integration of LLM-based agents with SCADA systems through the open MCP (Model Context Protocol) protocol, comparing it with the alternative of so-called skills, natural language instructions that orchestrate pre-existing tools. The structural limitations of using generic low-level tools are identified, in particular identifier hallucination on long and repetitive strings, and the design of high-level domain-specific tools is justified. A modular and replicable architecture is described, and the preliminary results obtained on an industrial case study are reported, where the proposal allows for a significant reduction in the number of required invocations and an increased reliability of responses compared with the skills-based approach.

Keywords: Natural language interfaces, Development of assistant systems, Agents

1. Introducción

La progresiva incorporación de modelos de lenguaje de gran tamaño (LLM) en entornos profesionales ha planteado el reto de su integración con sistemas heredados de control y supervisión industrial (Xia et al., 2025). Los sistemas SCADA forman parte de la familia más amplia de los *Industrial Control Systems* (ICS), junto a PLCs, DCS e interfaces hombre-máquina (Aslam et al., 2025). Desplegados en infraestructuras críticas, exponen miles de variables a través de modelos de datos jerárquicos. Su acceso programático requiere conocer convenciones de nombrado, formatos de tiempo y semánticas específicas del integra-

*Autor para correspondencia: pedro.pablo.gomez@udc.es
Attribution-NonCommercial-ShareAlike 4.0 International (CC BY-NC-SA 4.0)

dor (Wang et al., 2025).

La pregunta de fondo no es si los LLM pueden interactuar con SCADA (la respuesta empírica es afirmativa), sino cómo deben hacerlo para ser fiables. En la práctica reciente conviven dos paradigmas. El primero son las *skills*: conjuntos de instrucciones en lenguaje natural que orquestan herramientas genéricas. El segundo es el protocolo abierto MCP (Model Context Protocol), publicado por Anthropic a finales de 2024 (Anthropic, 2024) y formalizado en una especificación abierta (Model Context Protocol Community, 2025), que permite definir herramientas específicas accesibles desde cualquier asistente compatible. La elección entre ambos enfoques no es trivial. En escenarios donde el LLM debe manipular identificadores largos y repetitivos (por ejemplo, rutas canónicas de variables SCADA), un único carácter alterado en una transcripción intermedia invalida la consulta resultante sin generar señales evidentes de error. La literatura previa ya ha explorado el uso de MCP en la automatización de procesos administrativos (Olmedo et al., 2025) y la integración de agentes LLM con planta industrial mediante arquitecturas multi-agente (Xia et al., 2025). Sin embargo, la combinación específica de MCP con sistemas SCADA y la caracterización de sus modos de fallo continúa siendo un área abierta.

Este trabajo presenta una propuesta arquitectónica basada en MCP para la integración de agentes LLM con sistemas SCADA. La contribución se articula en torno a tres ejes: (i) la identificación sistemática de las limitaciones del enfoque basado en *skills* cuando media manipulación de cadenas frágiles entre invocaciones, alineadas con los modos de fallo descritos en taxonomías recientes de alucinación en ejecución de herramientas (Peng et al., 2026); (ii) el diseño de un patrón replicable de herramientas de alto nivel que encapsulan toda la lógica de coordinación en código; y (iii) la validación inicial de la propuesta en un caso de estudio industrial real.

2. Marco conceptual

2.1. Skills y herramientas genéricas

Una *skill*, en el contexto de los asistentes basados en LLM, es un fichero en lenguaje natural que se introduce en el contexto del modelo cuando este detecta una intención compatible. Las *skills* delegan en herramientas preexistentes la ejecución efectiva, limitándose a indicar al modelo cómo combinarlas. Su ventaja principal reside en la rapidez de iteración: una modificación se traduce en una edición de texto, sin recompilación ni redespliegue.

Esta flexibilidad presenta un coste estructural. Al delegar la coordinación entre llamadas en el modelo, las *skills* heredan las limitaciones cognitivas del LLM, en particular las relacionadas con la generación *token* a *token* de cadenas largas, repetitivas y semánticamente próximas (Huang et al., 2025). Trabajos recientes han caracterizado este fenómeno bajo el término de *tool execution hallucinations*, distinguiendo entre alucinaciones en la selección de la herramienta (*Tool Selection Hallucinations*, TSH) y alucinaciones en la invocación (*Tool Calling Hallucinations*, TCH), donde el modelo genera argumentos sintácticamente válidos pero referidos a identificadores inexistentes o incorrectos (Peng et al., 2026).

2.2. MCP como capa de abstracción

MCP (Model Context Protocol) es un protocolo abierto presentado por Anthropic a finales de 2024 (Anthropic, 2024) y formalizado en una especificación pública mantenida por la comunidad (Model Context Protocol Community, 2025). Su objetivo es estandarizar la comunicación entre aplicaciones basadas en LLM y herramientas, recursos o flujos de trabajo externos, ofreciendo a los integradores una capa equivalente, en su ámbito, a la que el Language Server Protocol supuso para los entornos de desarrollo (Model Context Protocol Community, 2025).

La arquitectura de MCP es cliente-servidor sobre mensajes JSON-RPC 2.0 y distingue tres roles (Model Context Protocol Community, 2025): el *host* (aplicación LLM final, p. ej. Claude Desktop u otros asistentes), uno o varios clientes MCP integrados en el *host*, y los servidores MCP que exponen capacidades. La especificación define varios transportes (stdio para servidores locales y *streamable HTTP* para servidores remotos) y cuatro tipos de capacidades en el servidor: *tools* (funciones invocables por el modelo), *resources* (datos legibles), *prompts* (plantillas reutilizables) y *sampling* (delegación al cliente de generaciones recursivas). En la práctica, la mayoría de implementaciones de servidor se centra exclusivamente en *tools* (Olmedo et al., 2025).

Cada *tool* MCP se describe declarativamente mediante un nombre, una descripción en lenguaje natural y un esquema de entrada (JSON Schema). El asistente lee la descripción para decidir si invocarla; esta descripción, a menudo subestimada, constituye en realidad un elemento de diseño crítico, dado que una formulación deficiente induce con frecuencia tanto omisiones (*tool* no invocada cuando era necesaria) como invocaciones espurias o sobre el *tool* equivocado (Peng et al., 2026).

La especificación enfatiza, asimismo, una serie de principios de seguridad: consentimiento explícito del usuario antes de invocar *tools*, control sobre los datos compartidos con cada servidor y consideración de toda descripción de *tool* externa como información no confiable hasta haberla revisado (Model Context Protocol Community, 2025). Estos principios resultan especialmente relevantes en escenarios industriales donde la frontera entre lectura inocua y operación con impacto físico puede ser sutil.

2.3. Aplicación al dominio SCADA

Los sistemas SCADA modernos exponen colecciones de *datapoints* estructurados jerárquicamente, donde cada elemento se identifica por una ruta canónica de varios niveles (por ejemplo, `XX_NNN_AA.SUBR.ELEM.SUB`). La consulta de un valor requiere la concatenación precisa de prefijos del sistema, identificador del *datapoint* y ruta del elemento, todo ello sujeto a convenciones específicas del integrador. La fragilidad de estas cadenas las convierte en el escenario adverso por excelencia para un enfoque basado en *skills*, en línea con los modos de fallo descritos en la literatura para invocaciones de *tools* con argumentos fabricados o semánticamente derivados (Peng et al., 2026).

La progresiva convergencia entre las redes de tecnología operacional (OT) y las redes de tecnología de la información (IT) ha modificado además el contexto en el que se despliegan estos sistemas: SCADA y el resto de la familia ICS, históricamente aislados, exponen hoy interfaces accesibles desde la

capa empresarial, lo que abre la puerta a la integración con tecnologías de IA pero a la vez exige replantear los principios de control y supervisión (Aslam et al., 2025).

Trabajos previos en automatización industrial han abordado la integración de LLM con planta a distintos niveles: arquitecturas multi-agente con roles diferenciados de *manager* y *operator* para descomponer y ejecutar tareas (Xia et al., 2025), y aproximaciones de modelado semántico asistido por LLM para construir información estandarizada OPC UA (Wang et al., 2025). La presente propuesta se sitúa en una posición complementaria: en lugar de delegar planificación o modelado al LLM, restringe su rol al de selector de operaciones de alto nivel, externalizando toda la coordinación crítica a código convencional.

3. Arquitectura propuesta

La arquitectura propuesta se organiza en torno a un servidor MCP propio que actúa como intermediario entre el asistente LLM y el sistema SCADA, exponiendo herramientas de alto nivel específicas del dominio. Cada herramienta encapsula una operación completa y devuelve directamente el resultado formateado, sin requerir que el modelo intervenga en la composición de identificadores ni en la orquestación de llamadas intermedias.

3.1. Topología y flujo de información

La Figura 1 muestra la topología propuesta. El asistente LLM, ejecutándose en un cliente compatible (Claude Desktop, OpenCode u otros), se comunica con el servidor MCP propio mediante el transporte stdio. Este servidor, a su vez, accede al SCADA a través de un servidor MCP genérico ya desplegado, mediante transporte HTTP autenticado con *token*.

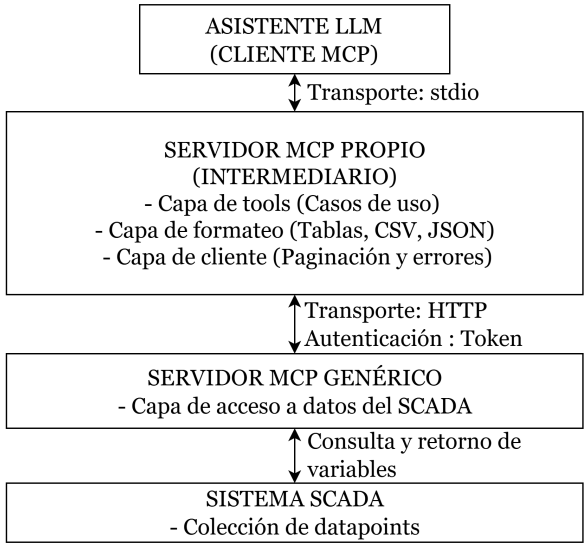


Figura 1: Topología de integración del agente LLM con el sistema SCADA mediante un servidor MCP intermediario que expone herramientas de alto nivel.

La elección de un intermediario obedece a dos consideraciones. Primero, permite mantener el código del servidor en una infraestructura distinta a la del SCADA, evitando la inyección de procesos adicionales en sistemas de operación crítica. Segundo, aísla al asistente de las herramientas genéricas del MCP del SCADA, eliminando la posibilidad de que el modelo invoque accesos de bajo nivel cuando existe una alternativa de alto nivel para el mismo caso de uso.

3.2. Patrón de herramientas de alto nivel

Cada caso de uso se materializa en una herramienta MCP dedicada. Una sola invocación basta para producir el resultado completo: el modelo no concatena identificadores ni transfiere cadenas entre llamadas. Internamente, la herramienta resuelve los patrones de variables aplicables, lee en lote todos los valores requeridos mediante una única llamada al SCADA y devuelve el resultado en el formato solicitado (tabla *markdown*, CSV o JSON estructurado).

La separación por dominios facilita la evolución. La adición de un nuevo caso de uso no afecta a los existentes: requiere la creación de un nuevo módulo y el registro de la herramienta en el despachador central. Las dependencias entre módulos se reducen al cliente compartido del SCADA y a los formateadores de salida.

3.3. Capas y responsabilidades

La estructura del servidor responde a una separación de capas estricta. Una capa de cliente concentra la totalidad de la comunicación con el SCADA, incluyendo la paginación, la deserialización de las respuestas y la gestión de errores parciales. Una capa de formateo unifica la generación de tablas, CSV y JSON, garantizando coherencia entre herramientas. Una capa de *tools* alberga la lógica específica de cada caso de uso, libre de detalles de transporte o presentación.

Esta organización tiene una consecuencia operativa relevante: un cambio en la forma de acceder al SCADA (migración entre transportes, sustitución del MCP genérico por acceso nativo) afecta únicamente a la capa de cliente. El resto del código permanece inalterado.

4. Consideraciones de diseño

4.1. Alucinación de cadenas frágiles

El comportamiento más limitante observado durante el desarrollo de la solución basada en *skills* fue la corrupción sistemática de identificadores cuando el modelo debía transferirlos entre llamadas. Identificadores con estructura repetitiva (por ejemplo, secuencias del tipo `XX_NNN-NNN_KMMAII_PMNN`) sufren alteraciones de un único carácter (`833` transcrito como `8MS`, `DI3` transcrito como `DIS`) que invalidan la consulta al SCADA pero pasan desapercibidas en una inspección visual rápida.

Este fenómeno se corresponde con lo que la literatura reciente categoriza como *tool calling hallucinations*, y más concretamente con la generación de argumentos fabricados (valores que respetan el esquema esperado pero referidos a entidades inexistentes en el sistema) y con la deriva semántica bajo esquema válido, modos de fallo identificados como particularmente insidiosos por evadir los validadores estructurales convencionales (Peng et al., 2026). En entornos multi-turno, el problema se agrava por la propagación de errores: un identificador alterado en una iteración temprana puede ser arrastrado a la memoria del agente y reutilizado en iteraciones posteriores, contaminando todo el trayecto (Peng et al., 2026; Huang et al., 2025).

Esta limitación no se mitiga mediante *prompts* más elaborados ni desaparece con modelos de mayor tamaño: persiste como

característica estructural del proceso de generación *token* a *token*. La única solución robusta consiste en retirar las cadenas críticas del contexto del LLM, encapsulando su manipulación en código.

4.2. Granularidad de las herramientas

La granularidad de las herramientas MCP constituye una decisión de diseño con implicaciones notables. Una colección amplia de herramientas finamente especializadas obliga al modelo a discriminar entre ellas en función de su descripción; una herramienta genérica con muchos parámetros, en cambio, traslada al modelo el problema de rellenar correctamente los argumentos.

La experiencia adquirida sugiere una regla práctica: si dos operaciones difieren en su lógica de negocio, conviene exponerlas como herramientas distintas. Si únicamente cambian datos de entrada del mismo proceso, una herramienta parametrizada es preferible. La aplicación de esta regla evita la proliferación de herramientas redundantes sin sacrificar la claridad semántica que orienta la selección por parte del modelo.

4.3. Reportar errores parciales sin silenciarlos

Una herramienta de alto nivel que agrega múltiples lecturas debe distinguir entre éxito total, éxito parcial y fallo total. El error más insidioso observado es la devolución silenciosa de un subconjunto del resultado esperado, sin indicación visible de los elementos no leídos. Un modelo que reciba una respuesta así presentará el dato parcial al usuario como si fuera completo. La literatura reciente identifica este patrón bajo el epígrafe de *missing validation and fault tolerance* y advierte de que, sin información explícita sobre lo no resuelto, los errores transitorios pueden consolidarse como creencias persistentes en la memoria del agente (Peng et al., 2026).

La arquitectura propuesta resuelve esta cuestión incorporando, en cada respuesta, un bloque explícito con los elementos no resueltos y la causa concreta de cada fallo. Esta información permite tanto al modelo como al operador detectar la pérdida y actuar en consecuencia.

4.4. Cálculo de magnitudes derivadas

En ocasiones, el SCADA proporciona magnitudes derivadas calculadas internamente (factores de potencia, sumas agregadas) que pueden presentar errores de escalado o configuración heredados de la parametrización del integrador. La arquitectura propuesta resuelve esta cuestión mediante el recálculo en código de las magnitudes derivadas críticas a partir de las magnitudes primarias, independizándose así del estado de configuración del SCADA. Las magnitudes recalculadas se marcan explícitamente en la salida para diferenciarlas de los valores leídos directamente.

5. Caso de estudio y resultados iniciales

Esta sección presenta la validación de la arquitectura propuesta sobre un escenario industrial real, con un doble objetivo: cuantificar la diferencia operativa entre *skills* y MCP sobre una misma tarea, y verificar si los modos de fallo descritos en la Sección 4 se manifiestan según predice la literatura.

La propuesta se ha validado en un entorno industrial real correspondiente a una instalación eléctrica de media tensión integrada en una infraestructura mayor. Los identificadores presentados a lo largo del trabajo se exponen en forma genérica, sin que ello afecte a la generalidad de la arquitectura ni a la interpretación de los resultados. El sistema SCADA expone aproximadamente veinte analizadores de red distribuidos en celdas de 6 y 15 kV, cada uno con varias decenas de variables asociadas.

5.1. Configuración experimental

Las mediciones reportadas se han obtenido con Gemma 4 26B como motor del agente, ejecutándose localmente en una estación de trabajo (Intel Core Ultra 7 265, 64 GB de RAM, NVIDIA RTX 4000 Ada Generation con 20 GB de VRAM). Esta configuración es representativa de estaciones de ingeniería industrial sin recursos especializados de cómputo para IA.

5.2. Comparativa entre enfoques

Se comparó el enfoque basado en *skills* con el enfoque MCP propuesto para una operación equivalente: la generación de un reporte energético completo de la instalación. La Tabla 1 sintetiza la comparativa observada entre ambos enfoques.

Tabla 1: Comparativa entre enfoques sobre el caso de estudio.

Aspecto	*Skills*	MCP propio
Invocaciones del LLM por consulta	20 – >50	1
Tiempo respuesta	~25 min	4–6 min
Corrupción de identificadores	recurrente	ausente
Detección explícita de fallos parciales	no	sí
Reproducibilidad	baja	alta

Nota: Los tiempos de respuesta están dominados por la latencia de inferencia del modelo de lenguaje; la comunicación con el sistema SCADA representa en ambos enfoques una fracción reducida del tiempo total.

5.3. Reducción de invocaciones

En el enfoque basado en *skills*, la generación de un reporte requería una llamada para resolver cada patrón de variables, una llamada por cada valor a leer (a menudo distribuidas en lotes por la imposibilidad del modelo de manejar listas extensas sin corrupción) y, frecuentemente, llamadas adicionales para corregir errores derivados de identificadores alterados. En episodios observados, una única consulta llegó a generar más de cincuenta invocaciones, varias de ellas redundantes o fallidas.

El enfoque MCP reduce esta carga a una única invocación del LLM, que delega en el servidor MCP la totalidad de la coordinación. El número de llamadas al SCADA queda determinado por la estructura del problema (cuatro en el caso concreto evaluado), independientemente del número de variables consultadas dentro de los límites de paginación.

5.4. Reproducibilidad y trazabilidad

Una segunda mejora, menos cuantitativa pero operativamente importante, es la reproducibilidad. El enfoque basado en *skills* produce salidas variables ante consultas idénticas, debido a la naturaleza estocástica de la generación del modelo y a su interacción con cadenas frágiles. El enfoque MCP, al ejecutar lógica determinista en código, produce salidas idénticas ante datos idénticos. Esta propiedad facilita la auditoría posterior y la integración con flujos de trabajo automatizados.

6. Discusión

Los resultados iniciales sugieren que MCP constituye una capa de abstracción adecuada para la integración fiable de agentes LLM con sistemas industriales. Las *skills* mantienen su utilidad para casos exploratorios y prototipos rápidos, pero presentan límites estructurales que las inhabilitan para uso productivo cuando media manipulación de identificadores precisos. Esta observación es consistente con trabajos recientes que evalúan la fiabilidad de agentes basados en LLM en escenarios industriales. En ellos, el rendimiento varía sustancialmente entre modelos pre-entrenados y modelos sometidos a *fine-tuning* supervisado sobre el dominio, así como entre familias de modelos de distinto tamaño. Las tasas de acierto reportadas oscilan entre el 29 % y el 100 % según la combinación específica y el subconjunto evaluado (Xia et al., 2025). La estrategia de externalizar la coordinación a código convencional reduce la dependencia de estos factores y, en particular, los costes asociados al ajuste fino para cada despliegue.

La separación de capas adoptada en la arquitectura propuesta facilita la evolución del sistema. La sustitución del transporte (de HTTP a acceso nativo, por ejemplo) o del propio modelo de datos subyacente se traduce en modificaciones localizadas que no propagan sus efectos al resto del código. Esta segunda posibilidad incluye, por ejemplo, la migración del MCP genérico a aproximaciones de modelado semántico asistido por LLM como las descritas en Wang et al. (2025). Adicionalmente, el caso de uso elegido para la validación —la consulta operativa de magnitudes energéticas en una instalación eléctrica— se alinea con el énfasis creciente de la literatura en la eficiencia energética y la sostenibilidad de los entornos ICS (Aslam et al., 2025). En este contexto, el acceso ágil y fiable a la información de consumo constituye un habilitador del análisis y la toma de decisiones operativas.

Persisten líneas abiertas. La limitación observada en modelos de menor tamaño al renderizar tablas extensas, incluso cuando los datos son correctos, requiere estrategias complementarias de presentación: formatos alternativos como JSON, o el uso de modelos de mayor capacidad para tareas con alta densidad de datos. El acceso a series temporales históricas, no contemplado en la versión actual, sería una extensión natural. Igualmente relevante sería la incorporación de herramientas de escritura, hoy ausentes por motivos de seguridad. Su despliegue exigiría un análisis específico de validaciones, autorizaciones y trazabilidad, alineado con los principios de consentimiento explícito y control sobre las operaciones que recomienda la especificación de MCP (Model Context Protocol Community, 2025).

7. Conclusiones

Se ha presentado una propuesta de arquitectura para la integración de agentes LLM con sistemas SCADA basada en el protocolo MCP, comparándola con el enfoque alternativo de *skills*. La propuesta encapsula la coordinación entre llamadas y la manipulación de identificadores en código, eliminando la fuente principal de errores observada en soluciones basadas en orquestación por parte del modelo.

Los resultados iniciales obtenidos en un caso de estudio industrial real evidencian una reducción sustancial del número de invocaciones requeridas, la práctica eliminación de errores por corrupción de identificadores y un incremento notable en la reproducibilidad de las respuestas. La arquitectura propuesta se presenta como un patrón replicable para casos de uso similares en otros dominios industriales con sistemas de variables jerárquicamente estructuradas.

Agradecimientos

La investigación de Pedro-Pablo Gómez-González ha sido financiada por la Xunta de Galicia mediante el programa Doutoramento Industrial(`http://gain.xunta.gal`), en el marco de la convocatoria Doutoramento Industrial 2025 con referencia 21_*IN*606*D*_2025_2261362.

El CITIC, como centro con acreditación en excelencia del Sistema universitario de Galicia y membro de la Red CIGUS, es subvencionado por la Consellería de Educación, Ciencia, Universidades y Formación Profesional de la Xunta de Galicia y cofinanciado a través del Fondo Europeo de Desarrollo Regional (FEDER), Programa operativo FEDER Galicia 2021-27, (Ref. *ED*431*G* 2023_01)

Proyecto PID2022-137152NB-I00 financiada por MICIU/AEI/10.13039/501100011033 y por ERDF/EU.

This research is co-financed by the Interreg Atlantic Area Programme through the European Regional Development Fund, *EAPA*_0019_2022 SAtComm project.

Referencias

Anthropic, Nov. 2024. Introducing the Model Context Protocol. `https://www.anthropic.com/news/model-context-protocol`, accedido: 24 de mayo de 2026.

Aslam, M. M., Tufail, A., Gul, H., Irshad, M. N., Namoun, A., 2025. Artificial intelligence for secure and sustainable industrial control systems: a survey of challenges and solutions. Artificial Intelligence Review 58 (11), 349.
DOI: `10.1007/s10462-025-11320-9`

Huang, L., Yu, W., Ma, W., Zhong, W., Feng, Z., Wang, H., Chen, Q., Peng, W., Feng, X., Qin, B., Liu, T., Jan. 2025. A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions. ACM Transactions on Information Systems 43 (2), 1–55.
URL: `http://dx.doi.org/10.1145/3703155`
DOI: `10.1145/3703155`

Model Context Protocol Community, 2025. Model Context Protocol Specification, version 2025-06-18. `https://modelcontextprotocol.io/specification/2025-06-18`, accedido: 24 de mayo de 2026.

Olmedo, L., Gallardo, G., De Blas, V., Peiretti, T., Piantini, S., Méndez-Garabetti, M., 2025. Integración de modelos de lenguaje en la automatización de tareas mediante el Model Context Protocol (MCP). In: Actas del XXXI Congreso Argentino de Ciencias de la Computación (CACIC). Viedma, Argentina, pp. 1101–1105.

Peng, H., Zheng, Y., Liu, Z., Lam, K.-Y., 2026. Tool execution hallucination in llm-based agents: A unified taxonomy with detection, mitigation, and future directions. TechRxiv 2026 (0227).
URL: `https://www.techrxiv.org/doi/abs/10.36227/techrxiv.177219979.94060974/v1`
DOI: `10.36227/techrxiv.177219979.94060974/v1`

Wang, R., Gu, C., He, S., Chen, J., 2025. Text2ua: Automatic opc ua information modeling from textual data with large language model. IEEE/CAA Journal of Automatica Sinica 12 (10), 2168–2170.
DOI: `10.1109/JAS.2025.125114`

Xia, Y., Jazdi, N., Zhang, J., Shah, C., Weyrich, M., 2025. Control industrial automation system with large language model agents.
URL: `https://arxiv.org/abs/2409.18009`