Módulo pasarela ROS/OPC UA para integración de datos de robot

María del Mar Plaza Cano
maplaca@upvnet.upv.es

Emima Ioana Jiva
emji@upv.es

Francisco Blanes Noguera
pblanes@ai2.upv.es

Patricia Balbastre Betoret
patricia@ai2.upv.es

Francisco Fraile Gil
ffraile@cigip.upv.es

Instituto Universitario de Automática e Informática Industrial, Universitat Politècnica de València, Camino de Vera S/N, Valencia, 46022, España

Resumen

Con los avances en la industria 4.0 y la transformación digital, es indispensable lograr la comunicación entre los distintos dispositivos que conforman la planta industrial. Sin embargo, esta orientación a la interoperabilidad de los productos que a nivel de dispositivos como los PLC's está generalizada, no lo es tanto en el ámbito de los robots. En este caso muchos de robots móviles trabajan con la arquitectura de software ROS en la actualidad, un middleware de código abierto que maneja los datos de control del robot a nivel local, y en el caso de los robots industriales es bastante generalizado el uso de protocolos propietarios. En este artículo se propone una solución que emplea el estándar de comunicaciones OPC UA para hacer accesibles los datos de dispositivos con arquitectura de software ROS a un nivel superior de control de la planta industrial, facilitando así la integración de los datos de los robots en aplicaciones de supervisión, y dando también cabida a la interoperabilidad por medio del uso de los datos en el servidor.

Palabras clave: OPC UA, ROS, interoperabilidad, comunicaciones industriales, robots, industria 4.0, TCP/IP.

1 INTRODUCCIÓN

1.1 SITUACIÓN Y CONTEXTO

La industria 4.0, conocida también como la cuarta revolución industrial, es una tendencia actual en la automatización industrial cuyo objetivo principal es modernizar la forma de fabricación y producción facilitando la integración de dispositivos y mejorando la comunicación entre estos [4]. Pero debido al uso de un gran conjunto de protocolos y tecnologías en la industria, la interoperabilidad entre los diferentes componentes se ha convertido en un gran reto. Como solución a este problema, se utilizan los protocolos de comunicación estandarizados. De esta forma, los componentes de diferentes fabricantes son capaces de comunicarse empleando mecanismos comunes en los diversos niveles de las capas de comunicación.

El estándar OPC-UA (OPC Unified Automation), tiene como objetivo proporcionar un estándar de comunicación entre máquinas. Es un protocolo construido sobre una estructura cliente/servidor basado en TCP/IP. Un cliente OPC UA accede a los datos del servidor mediante una comunicación basada en una conexión con una abstracción de sesión. El cliente envía una solicitud al servidor y recibe una respuesta. La comunicación entre los dos extremos proporciona un intercambio de datos seguro, fiable y encriptado sin pérdida de datos, incluso en situaciones con calidades de red poco fiables.

Este estándar, proporciona una forma sencilla de intercambiar datos e información entre las máquinas, independientemente de la plataforma o el fabricante, superando así las limitaciones tecnológicas de OPC clásico. Esto hace que sea fácil de integrar y operar, ofreciendo así los mejores requisitos para la digitalización de los entornos industriales. Cada vez son más los fabricantes industriales que ofrecen productos que puedan soportar OPC UA.

 https://doi.org/10.17979/spudc.9788497498418.0934

Últimamente, se ha trabajado mucho para lograr la integración de redes heterogéneas mediante OPC UA, por ejemplo, la conversión de los protocolos Profibus (Process Field Bus) y Modbus a datos OPC (OLE (Object Linking and Embedding) for Process Control) y las pasarelas OPC para convertir los datos de proceso de DCS (Distributed Control System) de terceros en datos OPC [2]. También se ha trabajado en la integración de ROS y OPC-UA usando el framework Eclipse Arrowhead [8], o la integración para Python y C++ que proporciona el paquete [ros_opcua_communication]. No obstante, este paquete queda obsoleto para las versiones de software que se tienen actualmente, y no permite un control total de la información que se desea desplegar en OPC UA. Esto puede suponer una sobrecarga en la red, desplegando datos innecesarios para la comunicación entre máquinas. En el presente artículo se propone una pasarela que convierte los datos de ROS seleccionados por el usuario a OPC UA usando Python. Este desarrollo ha sido probado con las últimas versiones de ROS.

ROS (Robot Operating System) es un entorno de trabajo para el desarrollo de robots, que proporciona una colección de herramientas y bibliotecas, y puede ofrecer funciones similares a las de un sistema operativo para un clúster de ordenadores heterogéneos. Simplifica la complejidad de las tareas de desarrollo y proporciona robustez de los comportamientos de los robots en una plataforma multirobot. En la actualidad más de 120 plataformas de robots utilizan ROS y existen más de 2000 bibliotecas de software de terceros disponibles en ROS [2]. Los robots son una parte indispensable en la fabricación inteligente y la producción flexible en la industria 4.0. Por este motivo es importante tener la capacidad de poder integrar en las redes industriales los datos ROS mediante OPC UA.

1.2 MOTIVACIÓN

Gran parte de los dispositivos utilizan protocolos de comunicación que dependen del proveedor que los fabrica. Esto afecta a la flexibilidad y adaptabilidad del sistema y supone un problema para la Industria 4.0 ya que esta heterogeneidad impide el intercambio de información entre los diferentes dispositivos.

En los últimos años y en un nivel de abstracción superior, ROS se presenta como un middleware flexible que se ocupa de este problema en el área de robótica. Pero el software modular proporcionado está limitado ya que no soluciona el problema a nivel industrial [1].

Pero en un entorno industrial moderno de Industria 4.0, donde robots y sistemas industriales cooperan es necesario comunicar los robots que funcionan con ROS, con los dispositivos y robots industriales, este artículo propone una pasarela que convierte los datos ROS a datos OPC UA. Así pues, cualquier robot que funciona con ROS, será capaz de comunicarse con otro dispositivo industrial que utilice el protocolo de comunicación de OPC UA proporcionando de esta forma interoperabilidad ROS-OPC UA.

El éxito de la interoperabilidad en la fabricación inteligente juega un papel importante en la comunicación efectiva y el intercambio de información entre máquinas sensores, actuadores y sistemas. Por este motivo el trabajo presentado tiene el principal objetivo facilitar la integración de datos de los robots que utilizan ROS y los dispositivos que usan OPC-UA.

2 TECNOLOGÍAS

2.1 ROS

El Sistema Operativo para Robots (ROS) fue desarrollado como parte del proyecto STAIR de la Universidad de Stanford [1]. Se podría definir ROS como un paquete de software de código abierto que puede ser utilizado en diferentes aplicaciones dentro de la robótica. Proporciona servicios que incluyen abstracción de hardware, control de dispositivos de bajo nivel, implementación de funcionalidades comunes, pasaje de mensaje entre procesos y manejo de paquetes. Se trata de una estructura distribuida de procesos llamados Nodos que permite un diseño individualizado pero que no impide un fácil acoplamiento al resto de procesos [6]. Cada nodo realiza una función y para comunicarse con otros nodos necesita ser gestionado por un nodo principal (Máster). Los nodos se pueden comunicar utilizando mensajes y topics. A la acción de enviar un mensaje a un topic se le denomina “publicar” y la acción de leer los datos de un topic se conoce como “suscribir”.

Actualmente se puede ejecutar en plataformas basadas en UNIX y en Windows 10 mediante virtualización. Está diseñado como un sistema peer-to-peer y esto posibilita la combinación de varios dispositivos en un sistema [6].

El objetivo fundamental de ROS es compartir y colaborar para poder facilitar así la integración de la información entre los diferentes sistemas del área de la robótica. Pero también existen otros objetivos que se pretenden alcanzar [6].

- Ser un sistema liviano: esta característica ofrece la posibilidad de que el código escrito para ROS sea utilizable junto con otros frameworks o estructuras de desarrollo de software para robots.

https://doi.org/10.17979/spudc.9788497498418.0934

- No depender de ningún lenguaje de programación: ROS está abierto a todos los lenguajes, esto se consigue haciendo que los datos de ROS se representen en un formato neutral. Actualmente es compatible con C++, Python, LISP y Octave, además existen bibliotecas experimentales en Java y Lua.

- Ofrecer la posibilidad de realizar pruebas sencillas: ROS posee un marco de prueba denominado rostest que proporciona facilidad para la activación y desactivación de dispositivos de prueba.

- Fomentar la generación de código reutilizable: las dependencias específicas que cada sistema presenta hacen que la extracción y reutilización de los paquetes de código existentes sea difícil. El sistema de construcción de ROS pretende fomentar la reutilización de los paquetes ya existentes. El mismo ROS utiliza código de distintos proyectos de código abiertos.

2.2 OPC UA

Es un protocolo basado en OPC y es uno de los primeros estándares establecidos para la comunicación entre dispositivos en los años 90. La arquitectura orientada al servicio se basa en TCP/IP y cubre las capas 5, 6 y 7 del modelo OSI. Con el fin de facilitar el acceso de los desarrolladores, la Fundación OPC proporciona la implementación básica de OPC UA para diferentes plataformas y actualmente compatible con C, C# y Java [9].

Dentro de las organizaciones industriales, OPC UA permite la comunicación entre componentes a cinco niveles: empresa, gestión, operaciones, control y dispositivos específicos del proveedor. Para que sea posible el transporte de la información a través de una red a una aplicación consumidora que utiliza servicios web estándar, los dispositivos deben exponer los datos a través de OPC UA. Los protocolos TCP/IP y SOAP son utilizados para transportar la información. Actualmente se está trabajando también en una especificación con un modelo subscripción/publicación [3].

Un servidor OPC UA facilita al cliente OPC UA aplicaciones y sistemas de control como por ejemplo MES y SCADA. Los clientes solicitan y escriben datos a través de los servidores. Las características más interesantes de OPC UA son [3]:

- Independencia de la plataforma: OPC UA no depende de la plataforma ni del proveedor y se puede utilizar sobre cualquier sistema operativo. Además, también existe la posibilidad de desplegarse en sistemas integrados y en la nube.

- Escalabilidad: OPC UA permite el desarrollo de sistemas escalables de forma que, si se añaden nuevos módulos de software a los equipos que ya se encuentran en la planta, no existe la necesidad de una configuración adicional.

- Descubrimiento: OPC UA posee la capacidad de plug-and-play. Esto quiere decir que, cuando nuevas plantas remotas se agregan a una organización, OPC UA consigue descubrir automáticamente sus redes y de esta forma se pueden configurar e integrar en la red de la empresa.

- Interoperabilidad: OPC UA se puede utilizar en sistemas industriales particulares sin tener en cuenta que los dispositivos y el software son de diferentes proveedores.

3 DISEÑO E IMPLEMENTACIÓN DE LA SOLUCIÓN

La solución propuesta tiene como objetivo principal lograr una mejora en las comunicaciones a nivel industrial entre dispositivos y/o robots que trabajen bajo la arquitectura de ROS y sistemas que utilicen el estándar de comunicaciones OPC UA. Para abordar esta solución, se ha desarrollado un módulo en ROS que despliega un servidor OPC UA a partir de los topics que escoja el usuario. Este módulo se ha desarrollado usando como base una distribución de código abierto de OPC UA denominada "Python OPC UA / IEC 62541". Toda la información acerca de esta librería se puede encontrar en su repositorio oficial [5]. La estructura del código desarrollado se describe a continuación.

En primer lugar, una vez configurado el servidor, se cargan los topics mediante un fichero XML o un fichero JSON, que previamente deberá haber completado el usuario en función de los datos que desee monitorizar. Esta funcionalidad permite reducir la carga del servidor, limitándose a ofrecer a los datos específicos que las aplicaciones cliente requieran. A partir de los nombres de los topics cargados en el módulo ROS, se crea la estructura de nodos en el servidor OPC UA, donde cada nodo se corresponde con un topic.

El servidor creado dispone de un método de inicio y otro de paro a petición del usuario. Cuando se inicia el servidor, una vez generada su estructura base de los nodos, se procede a la determinación del tipo de mensaje de cada uno de los topics y se importa la estructura de dicho mensaje de ROS en tiempo de ejecución. A partir de la estructura del mensaje de cada topic se pretende crear una subestructura de

 https://doi.org/10.17979/spudc.9788497498418.0934

variables OPC UA dentro de cada nodo o topic. El esquema seguido para esta parte de la implementación es el que se muestra en la figura 1. Se debe realizar el siguiente proceso para cada uno de los topics partiendo de su nodo OPC UA correspondiente, su nombre y su estructura de mensaje en ROS.

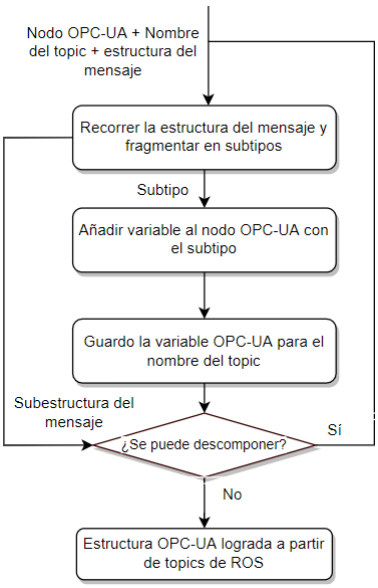

Figura 1: Método para la creación de la subestructura de cada nodo OPC UA

De forma análoga, es necesario seguir el procedimiento de la figura 2 para obtener los valores de cada uno de los topics e insertarlos en tiempo real en sus respectivas variables OPC UA.

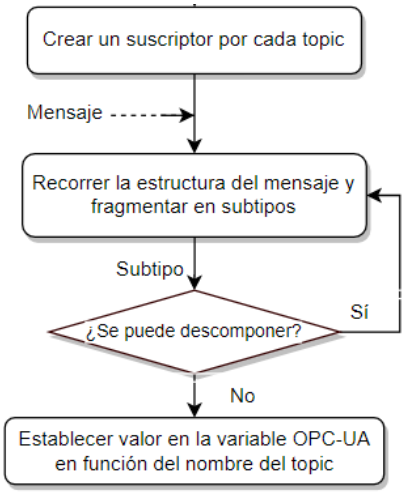

Figura 2: Método para la obtención de los datos de los topics de ROS y establecimiento de estos en el servidor OPC UA

Siguiendo estos procedimientos, se logra levantar un servidor OPC UA en cada uno de los robots y/o dispositivos que trabajan con ROS, haciendo accesibles sus datos internos de control y funcionamiento a un nivel superior en la planta industrial. Esto permite el manejo de estos datos para su posible control, gestión y/o monitorización desde sistemas generalizados de la planta que reúnen todos los datos de los dispositivos instalados mediante un protocolo de comunicaciones estandarizado.

4 PRUEBAS

Para verificar el correcto funcionamiento del servidor OPC UA, a modo de ejemplo, se configura un fichero en formato JSON con unos determinados topics cuyos datos se desean monitorizar en una aplicación industrial que incluye robots para actividades logísticas. y que emplea protocolos de comunicación estándar. La imagen de la figura 3 muestra la estructura del fichero de configuración del módulo.

```
{
    "robot_name": "RB-1_UR3",
    "topics": [
        "/battery_estimator/data",
        "/move_base/cmd_vel",
        "/robotnik_base_control/cmd_vel",
        "/robotnik_base_control/elevator_status",
        "/robotnik_base_control/enabled",
        "/robotnik_base_control/in_motion",
        "/robotnik_base_control/odom",
        "/robotnik_base_hw/emergency_stop",
        "/robotnik_base_hw/io",
        "/robotnik_base_hw/state",
    ]
}
```

Figura 3: Ejemplo del fichero JSON con los topics a levantar en el servidor OPC UA

El módulo ROS desarrollado importa estos topics y genera una estructura de nodos en el servidor OPC UA como se muestra en la figura 4.

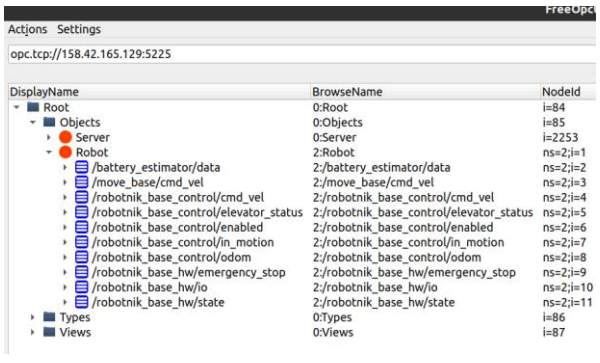

Figura 4: Estructura del servidor OPC UA con los nodos creados

A partir de estos nodos y en función de la estructura del mensaje ROS de cada uno, se procede a crear la subestructura de cada uno de los topics según el esquema mostrado en la figura 1 del apartado

anterior. Un ejemplo de la subestructura obtenida para uno de los topics se puede observar en la figura 5.

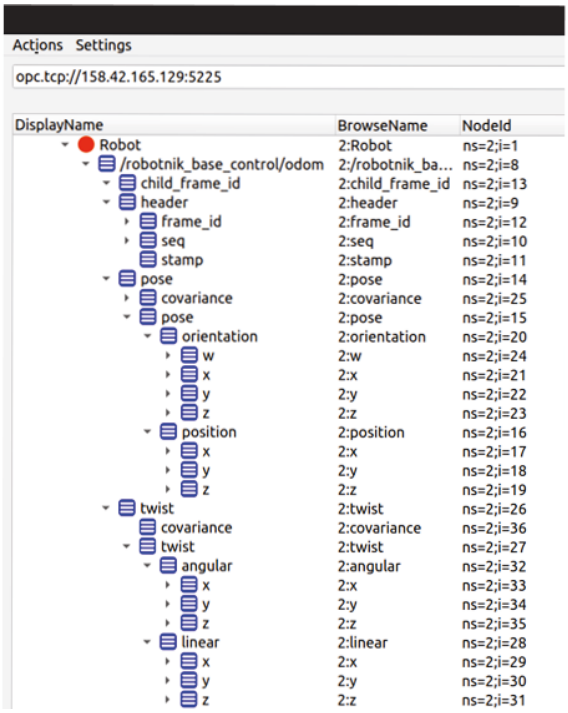

Figura 5: Subestructura OPC UA de uno de los nodos creados

Asimismo, siguiendo el procedimiento mostrado en la figura 2 del apartado anterior, se tendría para cada una de las variables de la subestructura mostrada, un valor determinado. Este valor se actualiza a tiempo real conforme se actualiza el valor del topic de ROS, luego es posible realizar una suscripción a un determinado dato para llevar un seguimiento o recoger un histórico de datos, en función del interés del usuario.

5 CASO DE APLICACIÓN

Un caso de uso común en aplicaciones de intralogística dentro de la industria, donde se busca la máxima eficiencia de los sistemas y autonomía para el desempeño de funciones repetitivas, es la comunicación en tiempo real entre diversos dispositivos dentro de la planta para realizar tareas de transporte y manipulación de cargas de forma óptima y sincronizada.

En este apartado del documento se describe un caso de uso donde se emplea la pasarela ROS/OPC UA desarrollada para una aplicación de seguimiento y recogida de piezas entre dos robots móviles autónomos de la empresa Robotnik que trabajan con ROS. Uno de ellos tiene implementado un brazo robótico UR (Universal Robots) que le permite manipular pequeñas cargas y desplazarlas, mientras que el otro robot móvil puede transportar cargas de hasta 50 kg.

Figura 6: Comunicación entre dos robots móviles empleando la pasarela ROS/OPC UA

ROS no es apto para un sistema multi-robot ó multi-máster. Sin embargo, existe la posibilidad de lograr la comunicación entre distintos robots mediante paquetes de ROS como el de [ROS-multimaster_fkie] o empleando ROS2. En este caso de aplicación, cada robot corre su propio nodo máster, y la comunicación entre ambos robots se logra gracias a la pasarela ROS/OPC UA, sin necesidad de instalar paquetes de software adicionales. Esto supone ciertas ventajas, puesto que no se ralentiza la red con los datos de los distintos robots activos ni se tiene pérdida de datos por sobrecarga.

El planteamiento es el siguiente. El robot móvil con el brazo manipulador puede moverse libremente por la planta, realizando las tareas que le sean encomendadas. Una vez que este robot haya seleccionado y adquirido la pieza a manipular, llamará al otro robot para que acuda a su localización y depositará sobre este la pieza a manipular. A continuación, el segundo robot transportará la pieza a un lugar determinado de la línea de producción según tenga planificado.

Con este propósito, se ha desarrollado otro módulo ROS publique continuamente en un topic las coordenadas de la posición del robot, y al mismo tiempo actualice su valor en el nodo OPC UA correspondiente. Este módulo debe ejecutarse en el robot que contiene el brazo manipulador, puesto que es el robot del que se desea conocer la posición. La estructura del código de esta aplicación está constituida por dos hilos de ejecución. En el hilo principal se inicia un nodo ROS y se crea un publicador a un topic que será utilizado únicamente para la comunicación entre ambos robots. La estructura del mensaje de este topic es un array con los datos de la posición del robot, y un entero que indique si tiene una pieza a depositar o aún no está preparada. El robot publicará continuamente en este

https://doi.org/10.17979/spudc.9788497498418.0934

topic tanto su posición como su orientación actualizada, así como el estado de la pieza a manipular. Por otro lado, en el segundo hilo de ejecución se despliega el servidor con la pasarela ROS/OPC UA desarrollada, que generará una estructura con los topics escogidos del robot incluyendo el que se ha creado para la comunicación. Los datos de estos nodos OPC UA creados en el servidor se actualizan conforme cambian los valores de los topics de ROS, por tanto, conforme el robot publica en este topic se actualiza su valor en el nodo OPC UA.

Al mismo tiempo, en el segundo robot se debe ejecutar una aplicación cliente OPC UA que realice una suscripción al nodo OPC UA que contiene la información de la posición del primer robot y el estado de la pieza. Así, cada vez que detecta un cambio en el valor de la posición del primer robot, si el estado de la pieza indica que está disponible para su recogida, el robot acudirá a dicha posición respetando un margen de seguridad. Usará la orientación leída para posicionarse frente al otro robot. Una vez se ha depositado la pieza, el primer robot actualiza el valor del estado de la pieza, y el segundo robot puede proceder a su traslado a un determinado punto de la planta, en función de los requerimientos del proceso de producción.

El principio de funcionamiento de esta aplicación se puede extrapolar a numerosas aplicaciones en la industria, con la versatilidad que ofrece poder definir libremente los nodos OPC UA en el servidor y el tipo de dato de cada uno de ellos, así como la gran ventaja que supone el empleo de ROS como middleware robótico.

6 CONCLUSIONES

En la actualidad, ROS está ampliamente utilizado en la industria por ser un middleware de código abierto que da soporte a diferentes aplicaciones dentro de la robótica. Cada vez son más las empresas que se decantan por esta solución para el control de sus robots, especialmente aquellas que trabajan con robots móviles autónomos (AMR). Esto es debido a que supone una gran ventaja el poder unificar los datos del conjunto y añade una capa de abstracción en el hardware de cada uno, lo que disminuye la complejidad en la intercomunicación y el control de los dispositivos.

Debido a la tendencia que existe actualmente de emplear software propietario en cada dispositivo de una planta industrial, se plantea el reto de la intercomunicación y la interoperabilidad entre dispositivos de distintos fabricantes sin necesidad de invertir en cambios de infraestructura de la planta. Por tanto, se hace interesante el empleo de software libre para facilitar el manejo y la recopilación de los datos de la planta en un nivel superior de gestión, así como la intercomunicación entre dispositivos.

Con la solución desarrollada es posible comunicar y unificar de forma eficaz los datos de dispositivos industriales con robots móviles y/o cualquier dispositivo que opere con ROS, logrando la interoperabilidad entre estos. Con tal fin, se ha desarrollado un módulo ROS que emplea el estándar de comunicaciones OPC UA mediante una distribución libre (Python OPC UA/IEC 62541). De esta forma, se consigue un módulo de código abierto en ROS que permite modificaciones y adaptaciones sin ningún coste en función de las necesidades del usuario y de los requerimientos de datos que tenga el control de la planta industrial.

De esta forma, sería posible acceder a los datos de los robots, máquinas, autómatas programables y sistemas en general, mediante un protocolo estandarizado de comunicación, permitiendo la monitorización y el seguimiento de los datos de la planta de forma unificada, así como logrando la interoperabilidad e interconexión entre los distintos sistemas.

7 TRABAJOS FUTUROS

Con la solución desarrollada, se ha logrado una comunicación unidireccional de los dispositivos que operan con ROS mediante un servidor OPC UA. Es decir, se pueden leer los datos contenidos en topics de ROS para su seguimiento y/o monitorización mediante un protocolo de comunicaciones estandarizado y libre. Sin embargo, no se ha desarrollado un módulo que permita la comunicación bidireccional con permisos de escritura sobre los datos. Esto podría ser de gran utilidad para poder controlar los dispositivos desde una única interfaz por OPC UA, y que los robots interpreten las publicaciones en los topics a través de esta interfaz y ejecuten las correspondientes acciones en términos de control de robot, o consignas de actuación.

Por otro lado, actualmente la arquitectura de software de ROS tiene operativa una nueva versión, ROS2, que aporta numerosas ventajas. Por ejemplo, permite crear código en Python3.5 y versiones superiores de C++11. Además, ROS2 puede instalarse en Windows 10, a diferencia de ROS que solo puede instalarse en sistemas Linux.

Asimismo, la diferencia principal es la desaparición del nodo Máster en ROS2. Mientras que en ROS impera el modelo centralizado bajo la jerarquía del nodo Máster, en ROS2 se prescinde de éste haciendo que sean los nodos los que avisen al resto de nodos

https://doi.org/10.17979/spudc.9788497498418.0934

de su aparición o marcha de la red [7]. Asimismo, la capa de transporte también se ha visto afectada para esta segunda versión de ROS. Mientras que ROS emplea TCPROS basado en TCP/IP para el envío de mensajes y la comunicación entre servicios, ROS2 utiliza DDS (Data Distribution Service), un framework de arquitectura publicador/suscriptor. DDS aporta gran flexibilidad y eficiencia al permitir la comunicación directa entre nodos, así como gran capacidad de configuración gracias al QoS (Quality of Service) [7]. Esto facilita la intercomunicación entre dispositivos con ROS.

Por tanto, sería interesante en un futuro desarrollar un módulo similar al propuesto que tenga soporte para ROS2.

Agradecimientos

Este desarrollo ha sido financiado por la Agència Valenciana de la Innovació, bajo el programa Projectes Estratègics en Cooperació 2021 (INNEST/2021/226). Acción cofinanciada por la Unión Europea a través del Fondo Europeo de Desarrollo Regional (FEDER), Programa Operativo de la Comunitat Valenciana 2014-2020.

English summary

ROS/OPC UA bridge module for robot data integration

Abstract

With the advances in Industry 4.0 and digital transformation, it is essential to achieve communication between the different devices that make up the industrial plant. However, this focus on product interoperability, which is widespread at the level of devices such as PLCs, is not so widespread in the field of robots. In this case, many mobile robots currently work with the ROS software architecture, an open-source middleware that manages the robot's control data locally, and in the case of industrial robots, the use of proprietary protocols is quite widespread. This paper proposes a solution that uses the OPC UA communications standard to make data from devices with ROS software architecture accessible to a higher level of industrial plant control, thus facilitating the integration of robot data into supervisory applications, and also enabling interoperability through the use of server-side data.

Keywords: OPC UA, ROS, interoperability, industrial communications, robots, industry 4.0, TCP/IP.

Referencias

[1] Edlinger, R., Leimer, L., Zauner M., Froschauer R., (2019) “Towards a flexible industrial robot system architecture”, *Proceedings of the ARW & OAGM Workshop 2019*

[2] Interconnection device, communication method, and system including robot, https://patents.justia.com/patent/11271790

[3] Mahnke, W., Leitner, S., Damm, M., (2009) “OPC Unified Architecture”

[4] Profanter, S., Tekat, A., Dorofeev, K., Rickert, M. and Knoll, A., (2019) "OPC UA versus ROS, DDS, and MQTT: Performance Evaluation of Industry 4.0 Protocols," *2019 IEEE International Conference on Industrial Technology (ICIT)*, 2019, pp. 955-962, doi: 10.1109/ICIT.2019.8755050

[5] Python OPC-UA Documentation, https://python-opcua.readthedocs.io/en/latest/

[6] ROS Introducción, http://wiki.ros.org/es/ROS/Introduccion

[7] ROS2 vs ROS1, https://geekgasteiz.wordpress.com/2018/11/01/ros2-vs-ros-1-migramos/

[8] Tripathy, A., van Deventer, J., Paniagua, C., and Delsing, J., “Interoperability Between ROS and OPC UA: A Local Cloud-Based Approach”, presented at the 5th IEEE International Conference on Industrial Cyber-Physical Systems (ICPS), Coventry, United Kingdom, May 24-26, 2022, 2022.

[9] Zezulka, F., Marcon, P., Bradac, Z., Arm, J., Benesl, T., Vesely, I., (2018) “Communication Systems for Industry 4.0 and the IIoT”, *IFAC-PapersOnLine*,Volume 51, Issue 6, 2018, Pages 150-155, ISSN 2405-8963

 https://doi.org/10.17979/spudc.9788497498418.0934