TP2: COMUNICACIÓN CON OPCUA Y MQTT

Protocolos de comunicación industriales

12 DE NOBIEMBRE DE 2024

LAUTARO KUHN, ESTEBAN PORPORATO
Profesores: Martín Pico, Milton Pozzo

Índice:

Contenido

Introducción

Este informe detalla el desarrollo y la implementación de un sistema de comunicación de datos basado en los protocolos **OPC UA** (Unified Architecture) y **MQTT** (Message Queuing Telemetry Transport), integrados en un entorno de sensores distribuidos. La aplicación de estos protocolos responde a la necesidad de diseñar una infraestructura de transmisión de datos que permita monitorear variables en tiempo real, con un enfoque en eficiencia y seguridad. La integración de OPC UA y MQTT en sistemas embebidos es especialmente relevante en aplicaciones industriales y de Internet de las Cosas (IoT), donde el intercambio de datos en tiempo real y la seguridad de la transmisión son primordiales.

La competencia específica que se aborda en este proyecto consiste en planificar, dirigir y ejecutar proyectos de comunicación de datos en sistemas de red complejos. Para cumplir con este propósito, se estudian en detalle los bloques funcionales del sistema, así como sus componentes y tipos de conexión, permitiendo crear un canal de comunicación seguro y eficaz entre dispositivos de monitoreo y almacenamiento de datos.

Objetivos del Proyecto

El principal objetivo de este trabajo es familiarizarse con el uso de los protocolos OPC UA y MQTT, comprendiendo su funcionamiento, características de seguridad y estructura de datos. Estos protocolos están diseñados para facilitar el intercambio de datos en sistemas industriales y de monitoreo. Mediante el uso de sensores, el sistema desarrollado captura datos analógicos y digitales, los cuales son transmitidos de manera ordenada y segura a través de una infraestructura de red distribuida.

Dentro de este marco, se establecieron objetivos específicos que guiaron el desarrollo de este proyecto: en primer lugar, diferenciar los componentes y elementos involucrados en la comunicación mediante OPC UA y MQTT, lo que incluye identificar los roles de cliente, servidor y broker dentro de la arquitectura del sistema. En segundo lugar, se plantea la configuración y puesta en funcionamiento de los componentes necesarios para establecer la comunicación entre dispositivos. Además, se estableció una estructura de tópicos adecuada para la transmisión y almacenamiento de datos, lo que permite que los suscriptores reciban información ordenada en tiempo real. Finalmente, se buscó probar los niveles de seguridad de los protocolos y evaluar sus ventajas y desventajas en diferentes contextos de uso.

Descripción General del Sistema

El sistema de comunicación se implementó utilizando tres dispositivos principales, cada uno con una función específica dentro de la arquitectura de red. La Raspberry Pi actúa como servidor OPC UA, generando datos simulados que representan mediciones de sensores de temperatura, presión y una marca de tiempo que se actualiza de manera continua. Otro dispositivo, Notebook de Esteban, se configuró como cliente OPC UA y, a la vez, como publicador MQTT, lo cual permite recibir datos desde el servidor y publicarlos en el broker MQTT ubicado en Notebook de Lautaro. Este último dispositivo opera como broker MQTT, gestionado mediante Mosquitto, y facilita la distribución de los datos a cualquier dispositivo suscriptor que requiera recibir la información en tiempo real.

Esta estructura facilita el flujo de datos desde el servidor de origen hasta los dispositivos suscriptores, utilizando OPC UA para la adquisición de datos y MQTT para la distribución. En el servidor, cada variable (temperatura, presión y tiempo) se actualiza cada dos segundos, lo que asegura un flujo constante de información disponible para el cliente OPC UA.

Configuración y Funcionamiento de los Componentes

Para el funcionamiento del sistema, la Raspberry Pi se configuró con un script denominado OPCUA-Server-Def.py, el cual permite simular datos de sensores en un entorno controlado. Este script establece un servidor OPC UA en la IP 192.168.0.150, en el puerto 4840, y genera datos que imitan valores reales de sensores de temperatura y presión. Estas variables están accesibles para el cliente OPC UA que se encuentra en Notebook Esteban.

Notebook de Esteban, configurada como cliente OPC UA y publicador MQTT, ejecuta el script Cliente_Publicador-Variante.py. Este código permite al dispositivo actuar en dos roles: como cliente OPC UA para capturar datos desde el servidor y como publicador MQTT, lo que posibilita enviar los datos al broker MQTT. La información de temperatura, presión y tiempo se publica en tópicos específicos dentro del broker ubicado en Notebook de Lautaro, como "sensors/temp", "sensors/press" y "sensors/time", los cuales están disponibles para cualquier dispositivo suscriptor.

Notebook de Lautaro alberga el broker MQTT, que actúa como el nodo central para la recepción y distribución de los mensajes. Configurado mediante Mosquitto, el broker MQTT recibe los datos publicados desde Notebook de Esteban y los almacena en los tópicos previamente definidos.

Mosquitto es un bróker MQTT open source y que cuenta con diferentes opciones de seguridad, cuando se instala para permitir que cualquier dispositivo sea capaz de comunicarse con el desde cualquier IP se deben de añadir en el archivo “mosquitto.conf” ciertos parámetros.

Debemos de añadir **listener 1883 0.0.0.0** y **allow_anonymous true**, estas sentencias le dice al mosquito que escuche en todas las direcciones del puerto 1883 y que permita la conexión de dispositivos anónimos, esto si bien reduce la seguridad del sistema nos facilita el no tener que implementar mecanismos de autenticación dentro de los códigos de comunicación.

Esto habilita que cualquier cliente MQTT pueda suscribirse a los tópicos que creamos y recibir la información en tiempo real. Para que un cliente vea todos los tópicos debe de suscribirse utilizando el siguiente comando:

mosquitto_sub -h ip_broker -t "sensores/#"

Lo que hace ese comando es captar todos los tópicos que esta saciados al grupo sensores, si se cambia # por el nombre de un tópico se mostrara ese solo.

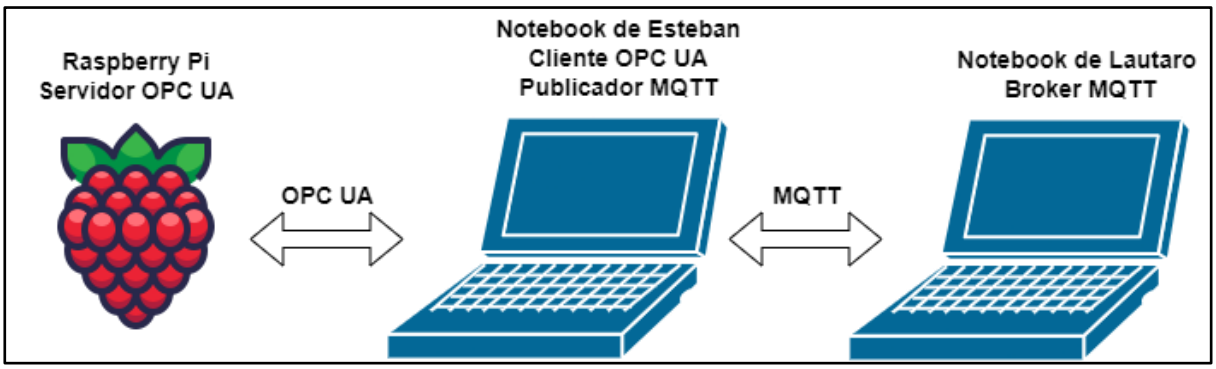
Precaución:

Las IPs de los códigos son fijas por lo que las IPs que figuran en los códigos son las utilizadas a la hora de realizar las pruebas, estas se deben de cambiar cuando se cambie de red o de dispositivos.

A su vez deben de estar todos los dispositivos en la misma red para que funcione la comunicación por wifi.

Estructura de la Comunicación y Diagrama del Sistema

La comunicación en el sistema se lleva a cabo a través de dos capas de protocolos diferentes, en donde OPC UA maneja la conexión y transmisión de datos entre el servidor y el cliente, mientras que MQTT facilita la distribución de estos datos desde el cliente publicador hacia otros dispositivos suscriptores. A continuación, se presenta el diagrama simplificado de la conexión del sistema:



Esta estructura permite la separación de funciones entre los componentes, optimizando la comunicación y el procesamiento de datos en tiempo real.

Calidad del Servicio (QoS) en MQTT

El protocolo MQTT permite la configuración de tres niveles de calidad de servicio (QoS), que controlan la fiabilidad de los mensajes enviados. Estos niveles permiten ajustar el compromiso entre fiabilidad y eficiencia en la transmisión de datos:

- **QoS 0:** Este nivel de servicio asegura que el mensaje se envíe solo una vez y no se confirme su entrega (transmitir y olvidar). Es adecuado para situaciones en las que la pérdida de datos ocasional es aceptable, ya que prioriza la rapidez de transmisión.
- **QoS 1:** Con este nivel, el sistema asegura que el mensaje se entregue al menos una vez al receptor (entrega con acuse de recibo). Aunque existe la posibilidad de

duplicación de mensajes, es útil en casos en los que es esencial que el mensaje se reciba, aunque esto aumente la carga en la red.

- **QoS 2:** Este nivel asegura que el mensaje sea entregado exactamente una vez, lo cual minimiza la duplicación y maximiza la fiabilidad de la comunicación (entrega garantizada). Sin embargo, este nivel aumenta la latencia de transmisión, por lo que se utiliza en casos donde la precisión es crítica.

Ventajas y Desventajas de OPC UA y MQTT

El uso de OPC UA y MQTT en combinación proporciona una infraestructura robusta para la transmisión de datos en sistemas de monitoreo. OPC UA ofrece ventajas significativas al proporcionar un modelo de datos estructurado y seguro, lo cual es ideal para entornos industriales donde la seguridad de los datos es prioritaria. Este protocolo incluye características de autenticación y encriptación, que resguardan la integridad y confidencialidad de los datos. No obstante, OPC UA presenta ciertas desventajas, como su alta demanda de recursos, lo cual puede dificultar su implementación en sistemas de hardware limitado, así como su complejidad de configuración.

Por otro lado, MQTT es un protocolo ligero y eficiente, especialmente adecuado para redes con ancho de banda limitado y para sistemas de comunicación donde la velocidad y simplicidad son prioritarias. Su modelo de publicación-suscripción facilita la comunicación entre múltiples dispositivos de manera sencilla. Sin embargo, en su configuración básica, MQTT carece de las funciones de seguridad avanzadas que ofrece OPC UA, lo cual limita su uso en entornos donde la protección de datos es fundamental.

Conclusión

La implementación de un sistema de comunicación basado en los protocolos OPC UA y MQTT demuestra que ambos pueden trabajar de manera complementaria para ofrecer una solución efectiva y segura en sistemas de monitoreo de datos. OPC UA, al actuar como servidor para la adquisición de datos en un entorno industrial, proporciona una capa robusta de seguridad y estructura de datos. A su vez, MQTT facilita la distribución de estos datos de forma rápida y eficiente, mejorando la interoperabilidad y permitiendo la conectividad de múltiples dispositivos suscriptores.

La integración de estos dos protocolos permitió cumplir con los objetivos del proyecto y demostrar su potencial en aplicaciones de comunicación de datos en tiempo real, aportando seguridad, eficiencia y una estructura de datos que puede adaptarse a diferentes necesidades y configuraciones dentro de un sistema embebido.

Referencias

MQTT. (2024). *MQTT.org*. Recuperado el 12 de noviembre de 2024, de https://mqtt.org/

IBM. (24 de agosto de 2022). *Calidad de servicio y gestión de conexiones en Integration Bus 10.0*. Recuperado el 12 de noviembre de 2024, de https://www.ibm.com/docs/es/integration-bus/10.0?topic=bus-quality-service-connection-management

Eclipse Mosquitto. (s.f.). *Documentación de Mosquitto*. Recuperado el 12 de noviembre de 2024, de https://mosquitto.org/documentation/

The Eclipse Foundation. (29 de abril de 2024). *Paho MQTT - Python Client*. Recuperado el 12 de noviembre de 2024, de https://pypi.org/project/paho-mqtt/

OPC Foundation. (2024). *OPC UA: Tecnologías OPC*. Recuperado el 12 de noviembre de 2024, de https://opcfoundation.org/about/opc-technologies/opc-ua/