Escuela Politécnica Superior de Jaén
Grado en Ingeniería Electrónica Industrial

UNIVERSIDAD DE JAÉN

Escuela Politécnica Superior de Jaén

Análisis de OPC-UA y MQTT como protocolos de comunicación en la Industria conectada

Alumno: Javier Del Moral Conde

Tutor: Elisabet Estévez Estévez

Dpto.: Ing. Electrónica y automática

Mayo, 2021

Javier Del Moral Conde

Análisis de OPC-UA y MQTT como protocolos de comunicación en la Industria conectada

Universidad de Jaén
Escuela Politécnica Superior de Jaén
Departamento de Ing. Electrónica y automática

Dña. **Elisabet Estévez Estévez** , tutor del Proyecto Fin de Carrera titulado: **Análisis de OPC-UA y MQTT como protocolos de comunicación en la Industria conectada**, que presenta **Javier Del Moral Conde**, autoriza su presentación para defensa y evaluación en la Escuela Politécnica Superior de Jaén.

Jaén, Junio de 2021

El alumno:

Javier Del Moral Conde

Los tutores:

ESTEVEZ ESTEVEZ ELISABET - 72578245C

Firmado digitalmente por ESTEVEZ ESTEVEZ ELISABET - 72578245C
Fecha: 2021.06.30 00:34:41 +02'00'

Elisabet Estévez Estévez

Resumen

En este TFG se analizará dos de los protocolos de comunicación más utilizados como es OPC-UA a nivel IIoT y de MQTT a nivel de dispositivos IoT, para el intercambio de información entre dispositivos, independientemente del fabricante.

Por lo tanto, se realizará un análisis exhaustivo de dichos protocolos para identificar cuándo es conveniente el uso de cada uno de ellos, las ventajas y desventajas de cada uno de ellos. Posteriormente, dado que en la industria de fabricación el controlador por excelencia sigue siendo el Autómata Programable o PLC, se ha de definir una arquitectura lo más heterogénea posible en cuanto a fabricantes se refiere.

Finalmente, se realiza unas prácticas de complejidad incremental muy útiles para futuros alumnos que estudien grado de ingeniería electrónica con especialidad en Automática.

Abstract

This TFG will analyse two of the most widely used communication protocols such as OPC-UA at industrial level and IoT MQTT, for the exchange of information between devices, regardless of the manufacturer.

Therefore, an exhaustive analysis of these protocols will be carried out to identify when it is convenient to use each one of them, the advantages and disadvantages of each one of them. Subsequently, given that in the manufacturing industry the controller par excellence continues to be the Programmable Logic Controller or PLC, an architecture as heterogeneous as possible in terms of manufacturers has to be defined.

Finally, some incremental complexity practices are carried out, which are very useful for future students studying an electronic engineering degree specialising in Automatics.

Índice

Índice de figuras

Índice códigos QR

Acrónimos

*CIM***,** fabricación integrada por computador (Computer Integrated Manufacturing).

COM, modelo de objetos componentes.

*DA***,** acceso a datos (data access).

DCOM, modelo de objetos de componentes distribuidos.

ERP, sistema de planificación de recursos empresariales.

HDA, acceso a datos históricos (historical data access)

*HMI***,** interfaz de usuario por sus siglas en inglés, (Human Machine Interface).

*IP***,** protocolo de internet.

*IoT***,** internet de las cosas.

*IT***,** tecnologías de la información.

*IIoT***,** internet industrial de las cosas.

M2M, máquina a máquina.

MQTT, message queue telemetry transport.

MES, sistemas de ejecución de fabricación.

OPC, OLE for Process Control.

OPC UA, arquitectura unificada conectividad abierta (OPen Connectivity-Unified Architecture).

PLC, controlador lógico programable.

QoS, calidad de servicio.

*SCADA***,** supervisión, control y adquisición de datos.

*TCP/IP***,** protocolo de control de transmisión/Protocolo de Internet.

XML, lenguaje de marcado (Extensible Markup Language).

1. MOTIVACIÓN, OBJETIVOS Y ESTRUCTURA

En este capítulo se presenta la estructura de dicho trabajo, la motivación y los objetivos a cumplir.

1.1 Motivación

Entre los años 80 y 90 empezaron a tener grandes problemas para mantener las comunicaciones en las plantas. Cada vez que se adquiría un nuevo equipo, normalmente éste traía un nuevo protocolo de comunicación propietario que tenía que comunicarse con el resto de la planta, nuevos drivers a instalar y mantener, nuevos métodos de programación, lo que implicaba grandes problemas hasta poder desarrollar las comunicaciones entre los dispositivos.

Las plantas industriales tenían cada vez más drivers de comunicación instalados en los ordenadores que controlaban las plantas. Los ingenieros tenían que lidiar cada vez con más y más tecnología. Llegaba un punto en el que no habían logrado hacerse con una tecnología que ya tenían que aprender una nueva.

Para evitar que las molestias sean quejas, los fabricantes de tecnologías de automatización comenzaron a asumir que tendrían que pensar en un plan B. De hecho, ellos ya hacía tiempo que habían empezado a notar que el modelo planteado era difícil de mantenerlo.

La solución fue el desarrollo de distintos protocolos de comunicación para poder hacer comunicaciones entre los dispositivos.

Con el paso de los años, sin importar el controlador que se utilice, la supervisión y control de ellos se hacía desde paneles táctiles o desde sistemas SCADA (*Supervisory Control and Data Acquisition* por sus siglas en inglés) Figura 1 Sistema SCADA .

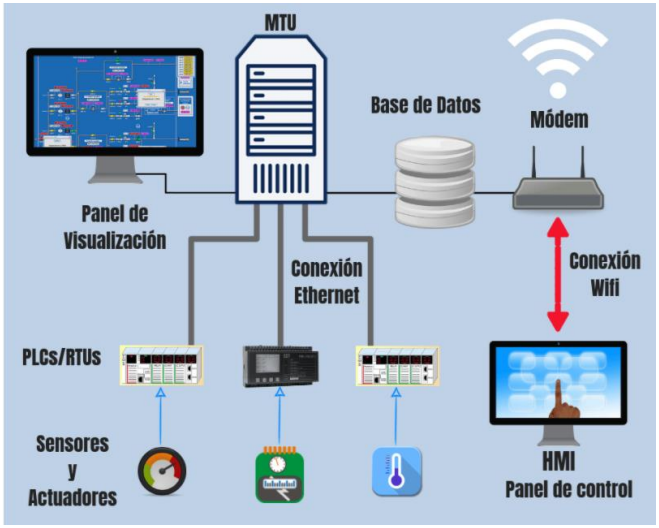


Figura 1 Sistema SCADA

Actualmente la tendencia hacia la Industria 4.0 ha ido incrementado en los últimos años. Este término generado en Alemania en 2011, no es nada más que una “cuarta revolución industrial”.

El uso de dispositivos IoT en el campo industrial se está volviendo cada vez más común, de tal manera que los ingenieros se han visto beneficiados debido a la facilidad de adquisición y control de datos del proceso de manera remota.

La motivación de este proyecto viene dada por la necesidad de aprender a desarrollar e implementar el uso de estas nuevas tecnologías ya que está provocando que las empresas de automatización, de robótica o cualquier ámbito tecnológico se estén preparando para la denominada Industria 4.0.

Por estas razones, este TFG busca ayudar al diseño e implementación de herramientas de Industria 4.0, concretamente este trabajo se centrará en dos de los protocolos de comunicación más utilizados en la industria 4.0, OPC UA y MQTT.

Figura 2 Industria 4.0

En el entorno industrial se habla de IIoT (Internet de las cosas Industrial), que engloban objetos físicos que emplean sensores y software para conectarse a la red y compartir datos, lo que los convierten en dispositivos inteligentes.

En el entorno de IoT uno de los protocolos más utilizados es MQTT (Message Queue Telemetry Transport por sus siglas en inglés).

Entre los nuevos requisitos demandados a los sistemas de fabricación está el de soportar comunicaciones ubicuas, es decir, que todo lo que esté en ámbito de producción sea capaz de comunicarse con todo, independientemente del momento y del lugar de la fábrica en que se encuentren. Esto dio lugar al concepto de Internet de las Cosas (IoT – Internet of Things) entre los que sobresale MQTT y otros protocolos de comunicación industriales, entre los que destaca OPC-UA.

Por ello, es de gran interés identificar cuándo es recomendable utilizar uno u otro, bondades y limitaciones de cada uno de ellos. La Industria 4.0 se define como “la digitalización de los procesos productivos en las fábricas mediante sensores y sistemas de información para transformar los procesos productivos y hacerlos más eficientes”.

1.2 Objetivos

En este proyecto se estudiará dos de los protocolos más conocidos tanto a nivel industrial como a nivel de IoT, una vez estudiados se realizará el diseño e implementación de los protocolos de comunicación, adicionalmente se enviará a un bróker MQTT los parámetros del entorno donde estaría el robot y mediante un suscriptor (node-red) se cogerán estos datos que posteriormente serán enviados al autómata para poder ser visualizado por el HMI.

La titulación del Grado en Ingeniería Electrónica Industrial, como todas las titulaciones impartidas en la Escuela Politécnica Superior de Jaén, deben de concluir según la normativa del plan vigente con la elaboración y la defensa por parte del estudiante.

En lo que a docencia se refiere, este proyecto tiene como objetivo principal que el alumnado pueda diseñar e implementar dos tipos de protocolos de comunicación conocidos como son OPC UA y MQTT, la implementación de estas nuevas tecnologías permite evitar grandes costes de mantenimiento de hardware y software en una empresa al poder tener comunicado todos los PLC's de una fábrica de manera homogénea y con protocolos de comunicación estándar con solo un servidor, adicionalmente estos protocolos permiten de manera más sencilla el análisis de datos mediante base de datos de Excel o SQL o plataformas como Azure, Aws.

1.3 Estructura

En este proyecto se puede diferenciar la siguiente estructura:

- Conceptos sobre OPC UA.
- Conceptos sobre MQTT.
- Diseño e implementación de ejemplos con OPC UA.
- Diseño en implementación de ejemplos con MQTT.
- Caso práctico donde se usará ambos métodos de comunicación, en este caso práctico se implementará la comunicación OPC UA, donde se simulará la recogida de datos de un robot mediante el uso de un PLC y este PLC le enviará por OPCUA la información necesaria a otro PLC que tendrá un HMI, concretamente una pantalla HMI Unified comfort panel, donde se podrá visualizar dichos parámetros.
- Características de ambos protocolos, ventajas y desventajas.

2. OPC UA

2.1 Historia OPC

Antes de hablar sobre OPC UA, se hablará de su antecesor, el problema de los múltiples protocolos de comunicación industriales tuvo su respuesta a mediados de los años noventa con tres siglas: OPC, que son las siglas de Open Protocol Communication.

En [1], se recoge que es un estándar de comunicación abierto e independiente. Su ámbito es la industria, ya que normalmente se encuentra en software industrial tal como SCADA's, Historiadores, drivers, etc.

La tecnología OPC tiene una arquitectura cliente – servidor. Esto significa que en cualquier arquitectura debe haber al menos dos software independientes, un servidor que se conecte a la fuente de datos y los exponga, y un cliente que haga las consultas pertinentes para hacer uso de esos datos.

Por ejemplo, se pueden encontrar servidores conectados a PLC's, que comuniquen los datos a un cliente como un SCADA. En segundo lugar, se habla de una plataforma de comunicaciones basada en tecnología Windows. OPC, Figura 3, utiliza la tecnología COM / DCOM para llevar los paquetes de comunicación entre aplicaciones .

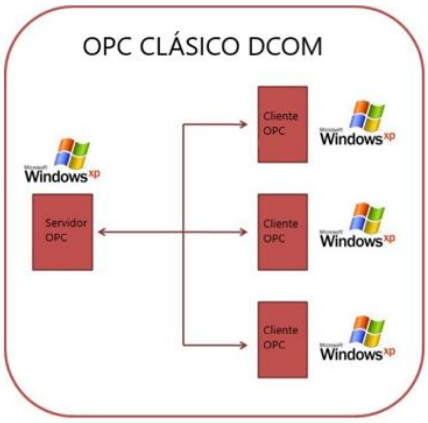


Figura 3 OPC Clásico

OPC tiene algunos problemas, el primero y más evidente es su dependencia de Windows, lo que conlleva su incompatibilidad con cualquier otro sistema operativo (Linux, iOS, Android, etc). El segundo, es la seguridad. En el mundo del Internet de las Cosas, es necesario contar con protocolos que traigan capas de seguridad nativas y, en el caso de OPC, éstas son bastante limitadas, esto llevo al desarrollo del protocolo OPC UA

2.2 OPC Unified-Architecture

Especificado por la OPC Foundation y estandarizado como IEC 62541, OPC UA nació en 2008 como una apuesta firme para sustituir al OPC Clásico en los sistemas de automatización. De esta forma, OPC UA sigue siendo un sistema de comunicación de tipo Cliente-Servidor:

- Servidor OPC UA: aplicación que contiene toda la información referente al proceso que se está controlando y que ofrece servicios para el acceso a esa información.
- Cliente OPC UA: aplicación que permite utilizar los servicios ofrecidos por un Servidor OPC UA para acceder a la información que contiene.

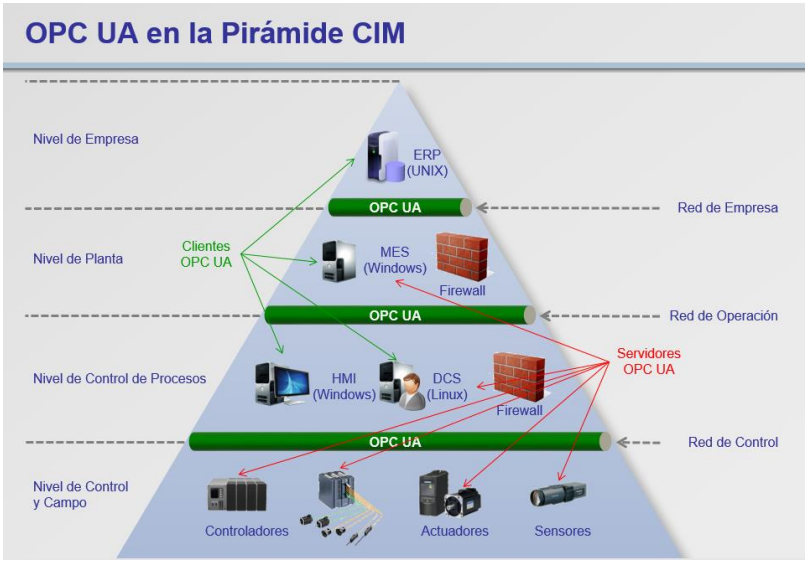


Figura 4 Pirámide CIM

2.2.1 Características generales

En [2] se recoge el objetivo principal de OPC UA: unificar toda la arquitectura de OPC Clásica en una infraestructura independiente de la plataforma, manteniendo toda la funcionalidad y así evitando la dependencia de los sistemas de Microsoft y de su tecnología COM/DCOM.

Los servicios en OPC UA se definen de manera abstracta, se utilizan mecanismos de trasporte para intercambiar datos entre Cliente-Servidor, un cliente puede acceder a los datos sin entender el modelo de datos subyacente, en la Figura 5 se observa el flujo de comunicación.

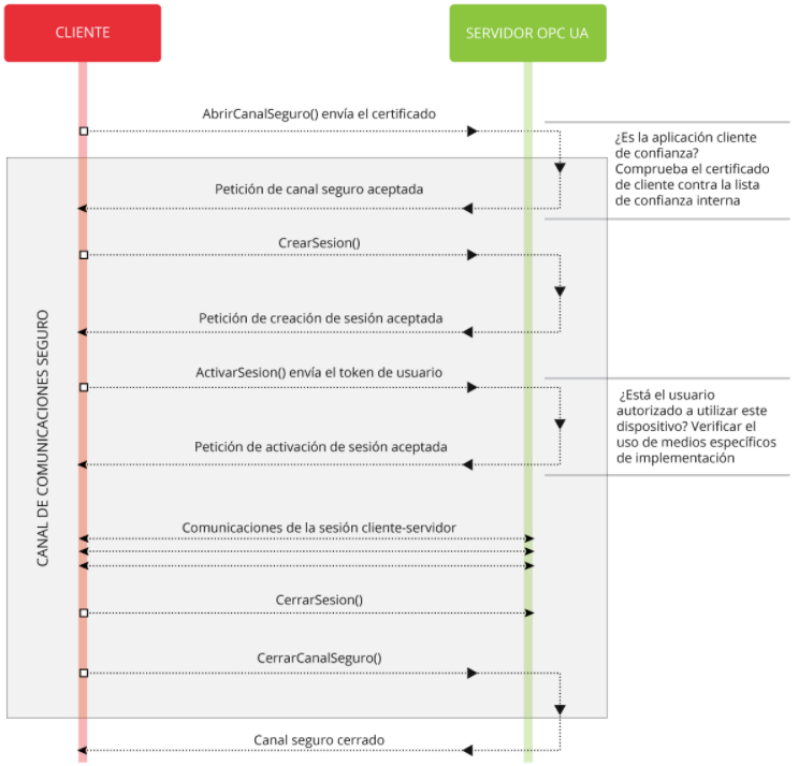


Figura 5 Flujo de comunicación Cliente-Servidor en OPC UA

En el estándar OPC UA se implementan varias medidas de seguridad para garantizar la confidencialidad, integridad y disponibilidad.

- La primera de ellas es mediante un certificado ya sea generado por parte del cliente como por parte de servidor, sin este certificado no pueden intercambiar datos entre ellos.
- La segunda es mediante el cifrado de datos en la capa de transporte de OPC UA, puedes permitir que se puedan leer o escribir ciertos datos en un cliente predefinido.

Además, OPC UA también está enfocado a integrar dispositivos de distintos fabricantes independientemente de las características propias de cada dispositivo (sistema operativo, comunicaciones, lenguaje de programación, etc.), por ello OPC UA está clasificado como middleware. En la Figura 6 se expone un ejemplo muy básico de arquitectura OPC UA:

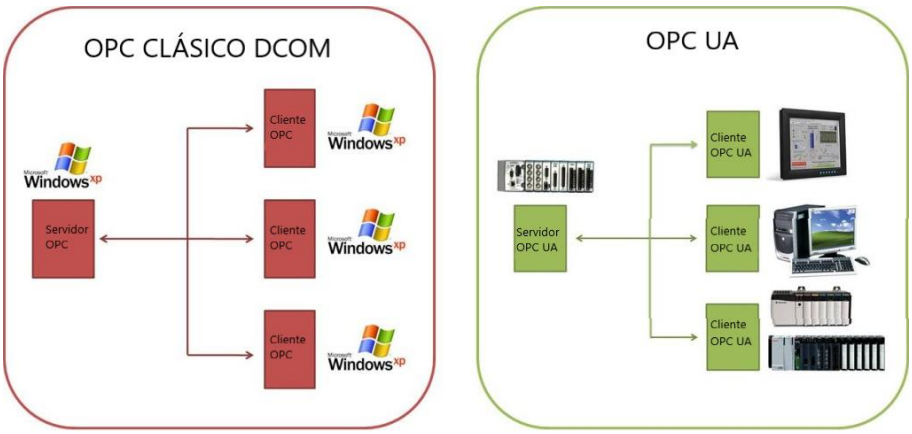


Figura 6 OPC Estructura OPC - OPC UA

OPC UA se basa en estándares ampliamente utilizados en el ámbito IT como TCP/IP o HTTP-HTTPS, lo que permite independencia de la plataforma sobre la que se implemente. Este hecho ofrece la posibilidad de implementar las aplicaciones OPC UA directamente sobre los equipos del sistema de automatización (PLC, sistemas empotrados, equipos PC, etc.).

2.2.2 Estructura OPC UA

En este apartado también se presenta la estructura OPC UA, en la que quedan definidos los protocolos de codificación, transporte y seguridad que ofrece para el intercambio de información.

Una estructura OPC UA implementa los diferentes mapeos de transporte OPC UA. La estructura se utiliza para invocar Servicios UA a través de procesos o redes.

OPC UA define tres niveles en el stack (estructura) y diferentes perfiles para cada nivel:

- **Nivel de Codificación**: Define la serialización de los servicios en formato binario y XML.
- **Nivel de Seguridad**: Especifica estándares de seguridad para servicios web o la versión para OPC binario.
- **Nivel de Transporte**: Define el protocolo empleado UA TCP para OPC Binario o HTTP/SOAP para servicios web.

Según cómo se combinen estos protocolos se obtienen distintas prestaciones, en función del uso al que vaya destinada la aplicación OPC UA:

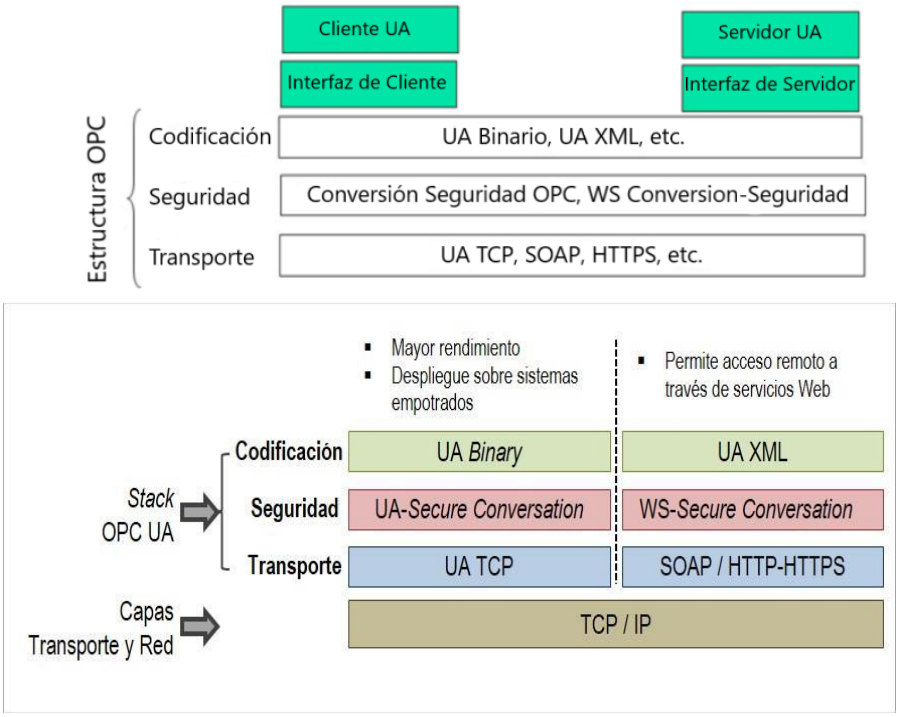


Figura 7 Estructura comunicación OPC UA

En la Figura 7, se observa que ambas combinaciones utilizan TCP/IP como protocolos de transporte y red, respectivamente. Sobre ellos se ha desarrollado la estructura OPC UA.

Hay dos opciones:

- **UA Binary**, se suele utilizar para aplicaciones de control, que están directamente relacionadas con el proceso de automatización y requieren altas prestaciones de comunicación (velocidad en la transferencia de datos, tamaño reducido de los datos,

  Permite reducir al máximo el tamaño de la información a enviar. A continuación, los datos ya codificados se cifran mediante el protocolo UA-Secure Conversation. Finalmente, para transportar los datos codificados y cifrados mediante TCP/IP, se ofrece el protocolo UA TCP como paso intermedio para establecer un canal seguro entre Cliente-Servidor, mediante el puerto 4840.

- **UA XML**, para aplicaciones en las que se necesita acceder a los datos de forma remota o no se requieren altas prestaciones de comunicación (p.ej. en los niveles de planificación ERP), se ofrece el protocolo UA XML para la codificación de los datos, a fin de facilitar su interpretación. Antes de enviar los datos mediante TCP/IP, se crea un mensaje SOAP, el cual incluye los datos codificados-cifrados e información referente a esos datos (longitud de datos, codificación utilizada, dirección de destino) en formato HTTP o HTTPS (si se quiere añadir una capa extra de seguridad). El hecho de utilizar codificación XML y HTTP-HTTPS permite el acceso a los Servidores OPC UA de forma remota mediante servicios Web.

2.2.3 Modelo de información

Por otro lado, como se menciona en [13] OPC UA es una tecnología orientada a objetos, ofreciendo funcionalidades como la herencia entre clases, la encapsulación de métodos, eventos o estructuras de datos y la definición de tipos propios.

OPC UA permite crear sus propios modelos de información, adaptado al proceso al que vaya dirigida la aplicación.

El modelo de información define las distintas unidades de información, en el Servidor OPC UA, que el usuario utilizará para identificar información proveniente del proceso, lo que le permite relacionar de forma sencilla la información que está consultando con su significado físico real.

El modelo de información permite definir en el Servidor OPC UA una unidad de información que contenga una variable (tendrá asociado el valor que llegue al PLC) y un método (tendrá acceso a la variable del PLC que actúa sobre el actuador). Además, el modelo de información permite añadir información del actuador (número de serie, ubicación, fabricante, año de fabricación, o peso (unidades de medida, rango de valores, …).

Esta característica facilita al usuario la comprensión de la información a la que está accediendo, y permite una estructuración de la información mucho más completa.

La definición de las unidades de información, así como la relación entre ellas, tiene que implementarse de algún modo. Esta implementación se lleva a cabo mediante la utilización de dos tipos de objetos como se muestra en la Figura 8: Nodos y Referencias:

- Los Nodos, son objetos que en función de la Clase Nodo a la que pertenezcan, representan componentes reales del proceso (Objetos), información de esos componentes (Variable) u operaciones que se pueden ejecutar sobre esos componentes (Métodos). Además, definen el tipo asignado a algunos Nodos (Objeto, Variable), información de Variables (DataType) o tipos de Reference (referencetype).
- Las Referencias son los objetos que permiten unir entre sí los Nodos, asignándoles una relación de parentesco jerárquica (Organización, Componente, HasSubtype, …) o no-jerárquica (HasTypeDefinition, GeneratesEvent, …).

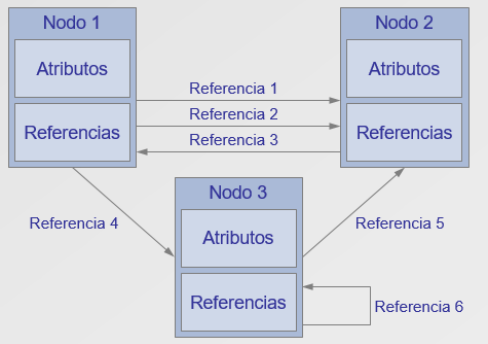


Figura 8 Ejemplo de Nodos y Referencias en OPC UA

Clases de nodos OPC UA:

Las clases más importantes de nodos (*NodeClasses*) son los Objetos, las Variables y los Métodos.

Objetos: Representan elementos físicos o abstractos de un sistema.

- Estructuran el espacio de direcciones.
- No contienen valores.
- Pueden agrupar variables, métodos u otros objetos.
- Las variables y métodos pertenecen a un objeto (Object) o Tipo de Objeto. (ObjectType).
- Los métodos son siempre invocados en el contexto de un objeto.
- Los notificadores de eventos (EventNotifier) son objetos.

Variable: Representan los contenidos de un objeto (valores)

- Los clientes pueden leer, escribir o subscribirse a cambios de los valores.

Métodos: Representan métodos invocables por los clientes. Siempre devuelven un resultado.

ObjectType: Representan los tipos para los objetos en el espacio de direcciones del servidor.

ObjectTypes, son similares a las clases en lenguajes orientados a objetos.

ReferenceType: Se utilizan para representar el tipo de referencias utilizadas por el servidor, establecen las relaciones entre nodos.

VariableType: Representan los tipos para las variables en el espacio de direcciones del servidor, definen las propiedades que están disponibles en la instancias de variables.

DataType: Representan los tipos de datos en el espacio de direcciones del servidor, siempre devuelven un resultado.

View: Se utiliza para restringir el número de nodos y referencias visibles en un espacio de direcciones grande. Proporcionar vistas adaptadas a tareas específicas o casos de uso.

Una vez definido el modelo de información del proceso, es necesario instanciar las distintas unidades de información que se ofrece a fin de obtener la representación del proceso en el Servidor OPC UA. Esta representación queda organizada en el espacio de direcciones del

Servidor. El Cliente OPC UA accederá al espacio de direcciones para consultar la información del proceso.

Para organizar las instancias en el espacio de direcciones se utiliza el concepto de Nombre, el cual contiene las instancias correspondientes a una función conceptual representada por el Servidor OPC UA.

Gracias a esta forma de presentar la información, si el usuario se conecta a este Servidor mediante un Cliente OPC UA, verá una representación gráfica del Espacio de nombres e identificará de forma muy sencilla qué proceso está controlando el PLC, qué señales está monitorizando y sobre cuál puede actuar. Para que pueda haber comunicación entre los Nodos del Servidor y las entradas/salidas del PLC se tendrán que haber relacionado ambos de la forma que corresponda a ese PLC.

2.2.4 Servicios OPC

Finalmente, un Servidor OPC UA ofrece a un Cliente OPC UA una serie de servicios que le permite interactuar con el servidor.

Entre ellos se destacan los siguientes:

- Gestionar los nodos, a fin de facilitar la tarea de adición/borrado de Nodos (añadir Nodos/Borrar Nodos) y Referencias (Añadir Referencias/Borrar Referencias).

  Así, aparece la figura del Manager de Nodos, el cual se encarga de ofrecer dichos servicios sobre los Nodos y Referencias que se le hayan asignado. Habitualmente un Manager de Nodos se encarga de ofrecer servicios sobre los Nodos y Referencias asignados a él, y que están organizados en un Espacio de nombres.

- Consultar, para examinar el Espacio de nombres y buscar Nodos-Referencias.
- Escribir/Leer, para escribir/leer la información de los Nodos.
- Suscripción, para subscribirse a cambios que le interesen en Variables o eventos producidos en el Servidor. Este servicio resulta muy útil, a fin de evitar que el Servidor esté consultando continuamente todos los Nodos que contiene.

Estos servicios se ofrecen mediante la creación de una Sesión entre Cliente-Servidor, lo que permite establecer un canal de comunicación seguro. En la Figura 9 aparecen representados estos servicios:

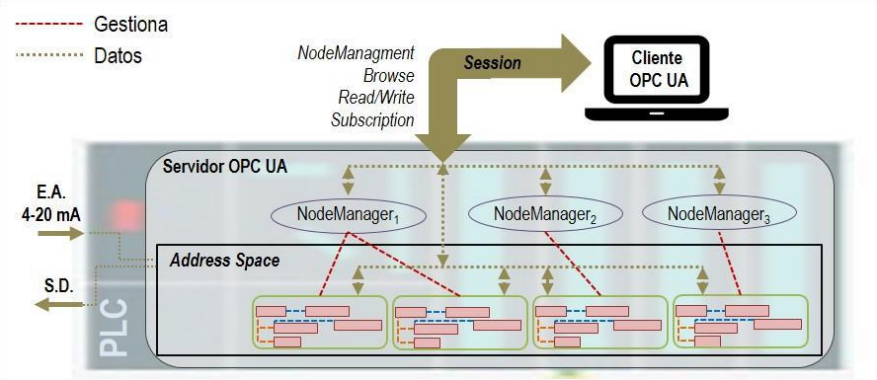


Figura 9 Servicios OPC UA

2.3 Ejemplos OPC UA

En este capítulo se desarrolla los distintos métodos para poder desarrollar e implementar los protocolos de comunicación con unos ejemplos de complejidad gradual.

En primer lugar, se mostrará un ejemplo de modelado de información que se seguirá en estas prácticas.

Suponiendo que en un proceso se tiene que controlar las posiciones del robot y la salida digital de la pinza que lleva el robot. La información relativa a las posiciones se encuentra en un Servidor OPC UA, que puede estar implementado directamente sobre el mismo PLC como se harán en los ejemplos posteriores.

El modelo de información definirá en el Servidor OPC UA una unidad de información denominada Posiciones (Bloque de datos del PLC) que contenga unas variables (Px, Py, Pz, Rx, Ry, Rz), tendrá asociado el valor de la posición que llegue al PLC desde el robot y un método *Abrir* (tendrá acceso a la variable del PLC que actúa sobre la pinza para cerrarla o abrirla). Se puede añadir información referente al robot u otras variables (número de serie, velocidad de las juntas, fabricante, año de fabricación, intensidad, temperatura), aunque en estos ejemplos no se ha profundizado sobre estas variables.

Esta característica facilitará la lectura de los datos recibidos y permite una estructuración de la información mucho más completa. En la Figura 10 se refleja este ejemplo:

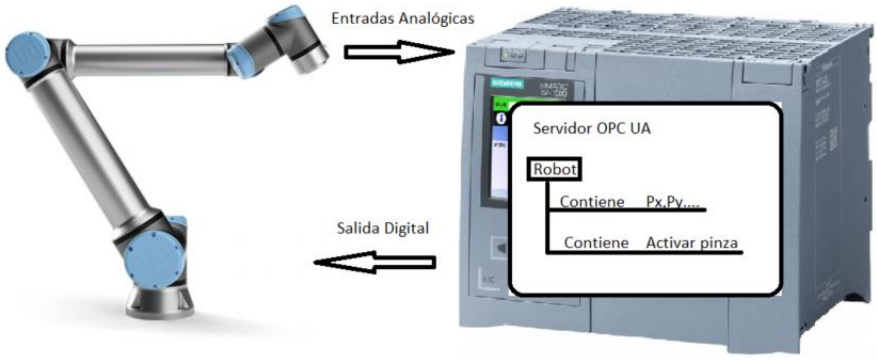


Figura 10 Ejemplo modelo de información OPC UA

El ejemplo es bastante simple, pero gracias a la posibilidad de definir un modelo de formación propio, se podrían haber creado otras unidades de información (velocidad, Intensidad, etc.)

Siguiendo el ejemplo del Robot, los elementos que forman la unidad de información de 'Robot' se podrían implementar mediante los siguientes Nodos que muestra la Figura 11:

Node	*NodeClass*
Robot	*Object*
Px, Py, Rx...	*Variable*
Abrir, cerrar pinza	*Method*

Figura 11 NodeClass de los Nodos que representan el Motor

Puesto que **Robot** y **Px, Py** son *Objecto* y *Variable* respectivamente, habrá que especificar de qué tipo son. Esta especificación se hace estableciendo una relación hacia el *Nodo* que especifica el tipo:

Node/NodeClass	*Reference*	*Node/NodeClass* de tipo
Robot /Object	HasTypeDefinition	Robottipo /ObjectType
Px, Py, Pz /Variable	HasTypeDefinition	AnalogItemtipo /VariableType

Figura 12 NodeClass de Robottipo y AnalogItemtipo

De esta forma quedaría definida la unidad de información 'Robot' en el modelo de información. La utilidad de definir el modelo de información reside en que: podría darse el caso de que el usuario quisiera controlar diez robots. Así pues, gracias al modelo de

información definido, se podrían instanciar tantas unidades 'Robot' como se necesiten, al igual que con el resto de las unidades de información definidas.

Estas instancias quedarán definidas en el espacio de direcciones del Servidor OPC UA.

En la Figura 13 se intenta representar los conceptos introducidos hasta el momento. En este caso se considera que el usuario sólo quiere controlar un robot:

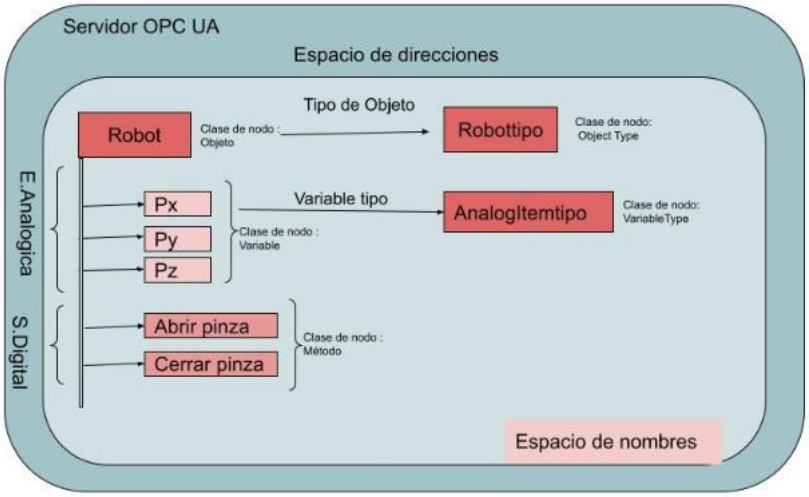


Figura 13 Ejemplo servidor OPC UA

Tal y como se aprecia en la Figura 13, la forma de relacionar entre sí los Nodos los espacios de direcciones es mediante Referencias. En el caso de las posiciones y de los métodos de abrir y cerrar pinza, se han utilizado Referencias jerárquicas puesto que ambos Nodos forman parte de un Nodo superior. En el caso de *Robottipo* y *AnalogItemTipo*, no existe una jerarquía entre ellos y los Nodes relacionados.

Gracias a esta forma de presentar la información, si el usuario se conecta a este Servidor mediante un Cliente OPC UA, verá una representación gráfica del Address Space similar a la de la Figura 10, e identificará de forma muy sencilla qué proceso está controlando el PLC, qué señales está monitorizando (Posiciones) y sobre cuál puede actuar (abrir y cerrar pinzas).

2.3.1 Comunicación PLC Siemens-KepServer-UaExpert

El primer ejemplo que se realizará en este apartado simulará la lectura de los parámetros de la posición del Tool Center Point de un Robot y se mostrará en 2 clientes, uno sera en Kepserver, uno de los software más conocidos de OpcUa junto con Matrikon y el segundo cliente será UaExpert Client de Unified-Automation.

Los dos programas son completamente gratuitos, Kepserverx se puede descargar gratuitamente, la única limitación que tiene es que a las 2 horas de funcionamiento el servidor se reinicia, perdiendo así los datos con los que se están trabajando, para un uso industrial es recomendable pagar licencia aunque para fines educativos se puede utilizar sin ningún problema.

Se adjunta Código QR 1 Comunicación PLC y dos clientes, que lleva al vídeo donde se muestra el ejemplo en acción.

Código QR 1 Comunicación PLC y dos clientes

Configuración del Servidor Opc UA

Como se muestra en [5] la parte del servidor será configurada e implementada en un PLC virtual, para ello se utiliza un software específico llamado Tia Portal, junto con PLCSIM-Advanced se podrá simular el PLC, en este caso de la serie S7-1500, concretamente en un S7-1511-1PN con la versión de software 2.8.

El primer ejemplo que se realizará en este apartado simulará la lectura de los parámetros de la posición del Tool Center Point de un Robot.

Se tiene que tener descargardo el programa PLC Sim advanced, en este caso se utilizará la versión 3.0 y la versión Tia Portal V16, al ser una simulación no se permite tener autentificación de acceso al servidor por lo que la seguridad utilizada será mínima.

Lo primero de todo es crear un proyecto en Tia portal para configurar un dispositivo como OPC UA, debido a que PLCSIM ADVANCED actualmente está limitado, solo se podrá

trabajar con la serie s7-1500, aunque la serie s7-1200 también tiene la opción de utilizar este protocolo.

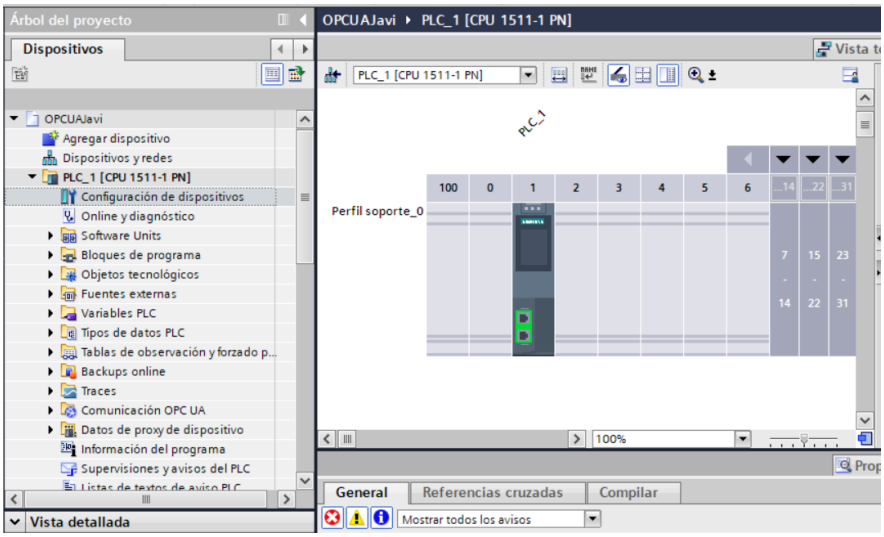


Figura 14 PLC simulado en el programa Tia Portal V16

El primer paso es pulsar dos veces en el PLC e ir a **General → OPC UA → General → Servidor** y activarlo. Una vez activado se muestra la dirección del servidor (url-endpoint), se mantendrá el valor del puerto por defecto, el 4840.

Se tienen que quitar las condiciones de seguridad por lo mencionado anteriormente al no ser compatible con una simulación.

Por último hay que dirigirse a **Licencias Runtime → OPC UA** y se genera la licencia que el autómata requiera, en este caso la licencia básica.

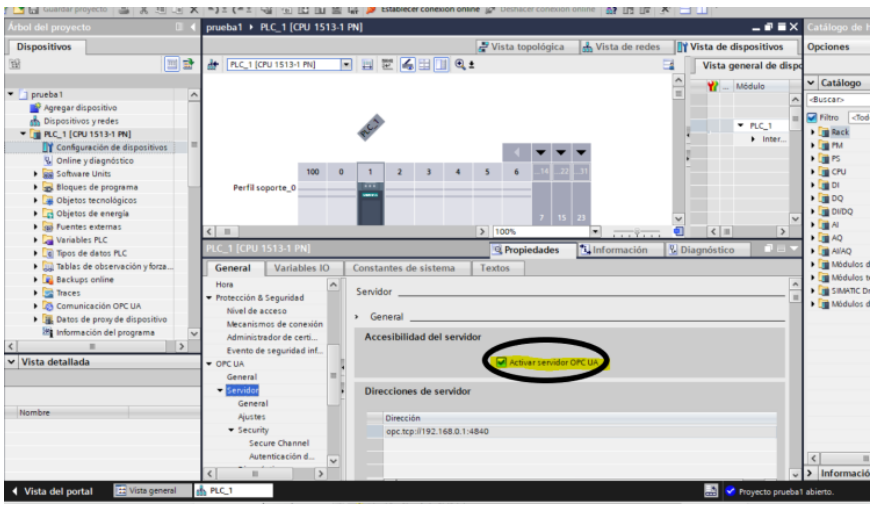


Figura 15 Propiedades PLC

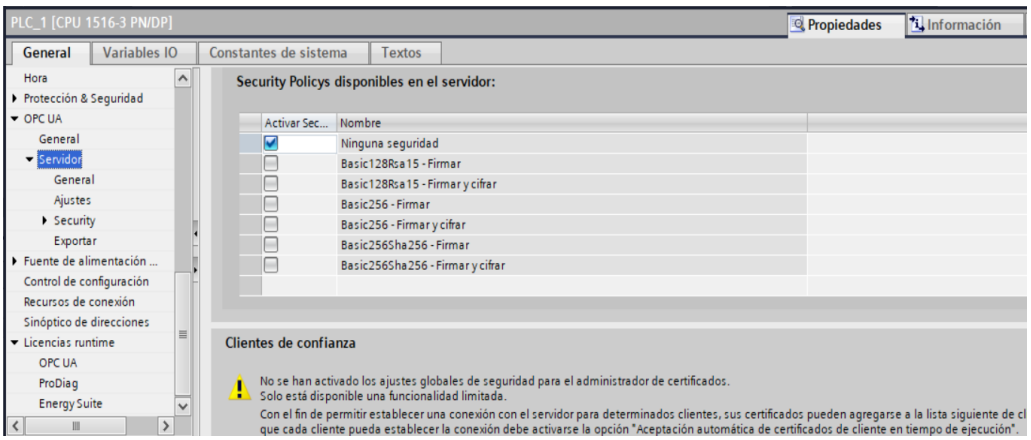


Figura 16 Propiedades de seguridad del servidor OPC

Como muestra la Figura 17, se puede observar la dirección IP al ser una simulación y no tener una dirección específica como si tendría un PLC físico, para este ejemplo se utiliza la siguiente IP: 192.168.0.21.

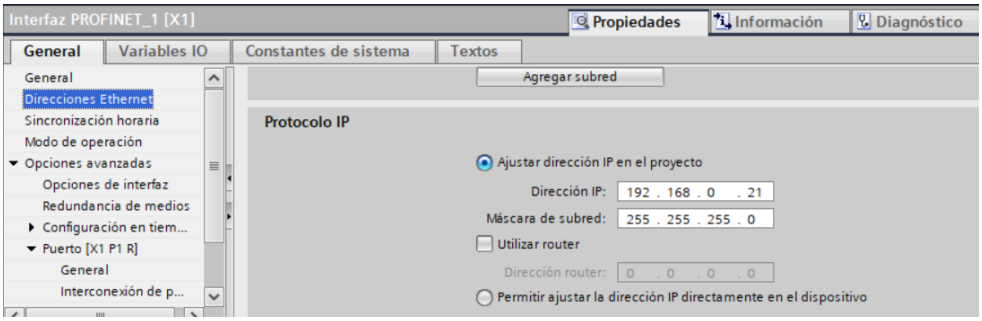


Figura 17 Dirección IP autómata

Una vez configurado en la parte de Tia Portal hay que dirigirse al simulador para añadir los datos necesarios para conectarse al PLC, lo que se debe de hacer es meter la IP que se le ha adjudicado al PLC y añadir la máscara, una vez introducidos los datos se le da al botón **Start.**

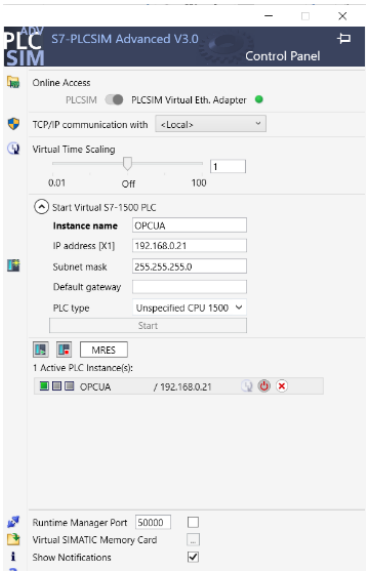


Figura 19 PLCSimAdvanced 3.0

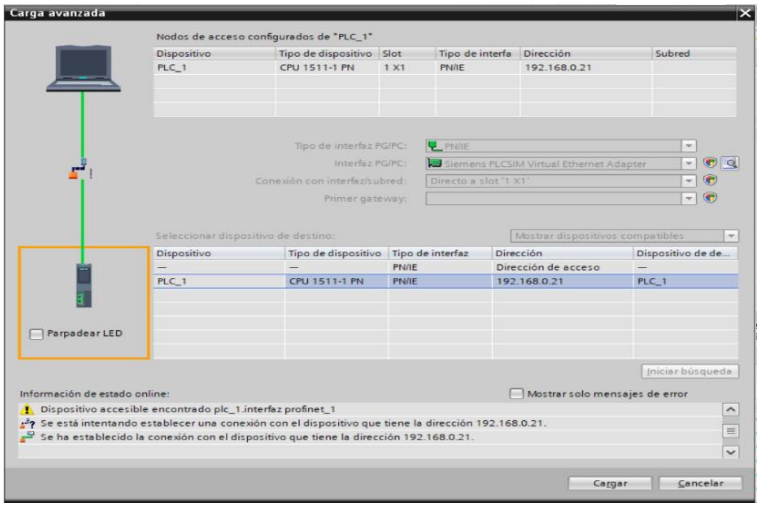


Figura 18 Búsqueda de dispositivo

Ahora el siguiente paso que se tiene que hacer es darle a Iniciar la simulación y en cargar dispositivo, se elige 'Siemens PLCSIM Virtual Ethernet Adapter e iniciar la búsqueda de un dispositivo, como se muestra en la Figura 18, cuando se encuentre un dispositivo se carga un programa sencillo para comprobar que todo está bien, en las imágenes que se mostrarán se podrá ver el proceso.

Una vez que el dispositivo ya está cargado y el PLC está configurado para ser un servidor OPC UA se pasará a configurar dos clientes a utilizar.

Configuración por parte del cliente de Kepserver

Para la configuración en Kepserver lo primero que se hace es configurar la IP del dispositivo, en este caso se utiliza como cliente, aunque se puede utilizar como servidor, para eso se seguirán los pasos que se muestran a continuación:

Se lleva el ratón a la parte inferior izquierda del PC y pulsar en **OPC UA Configuration**, pulsar en **Add,** elegir el adaptador de red e introducir la dirección IP de la red Internet a la que se conecte el PC.

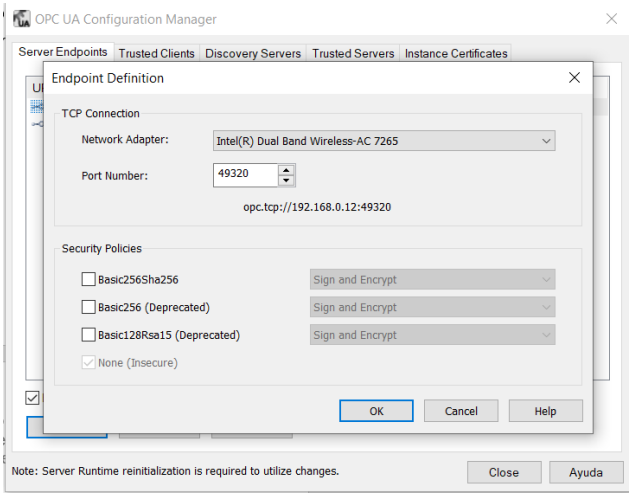


Figura 20 Configuración OpcUa de Kepserverex

Una vez puesta la IP de nuestro cliente se mete en Kepserver y se ejecutan los siguientes pasos.

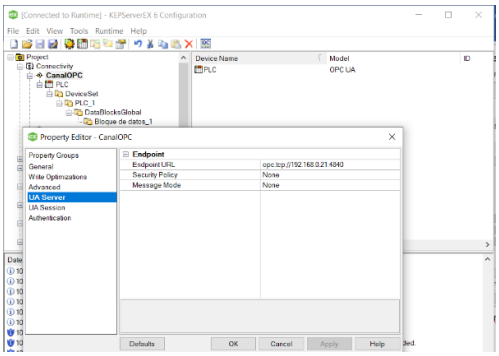


Figura 22 Propiedades del cliente en KepServer

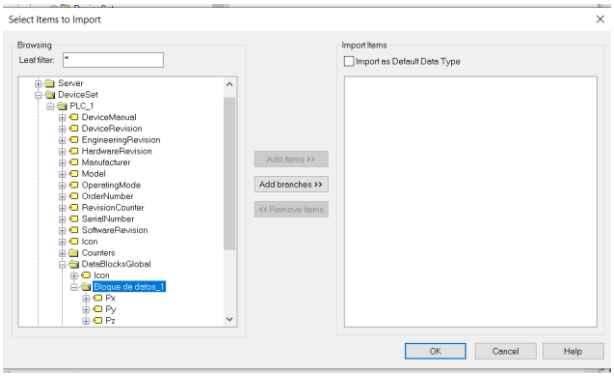


Figura 21 Figura Importación de Tags del Servidor al Cliente

Se crea un canal y en endpoint URL se introduce el endpoint del PLC que por defecto será opc.tcp://direcciónplc:puerto, normalmente el puerto con número 4840.

Posteriormente se añade un dispositivo, en este ejemplo se llama PLC y finalmente se tiene que importar los variables de ese PLC.

Posteriormente se configura el cliente en UaExpert y una vez configurado debería de aparecer el servidor en la parte inferior-izquierda del programa.

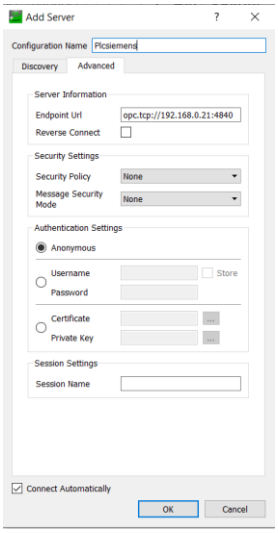


Figura 24 Añadir servidor en UaExpert

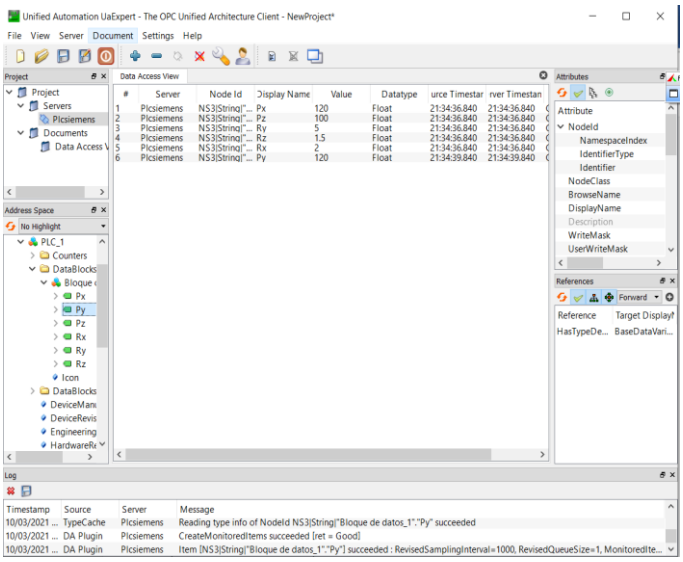


Figura 23 Cliente visualizando las variables del PLC

Una vez ya configurados el Servidor y los dos Clientes se muestra la comunicación entre los dispositivos, como se muestra en la Figura 23 Cliente visualizando las variables del PLCFigura 23.

Si se cambian las variables del PLC se verá reflejado en los clientes, los clientes también pueden actuar como servidor y cambiar la variable del PLC.

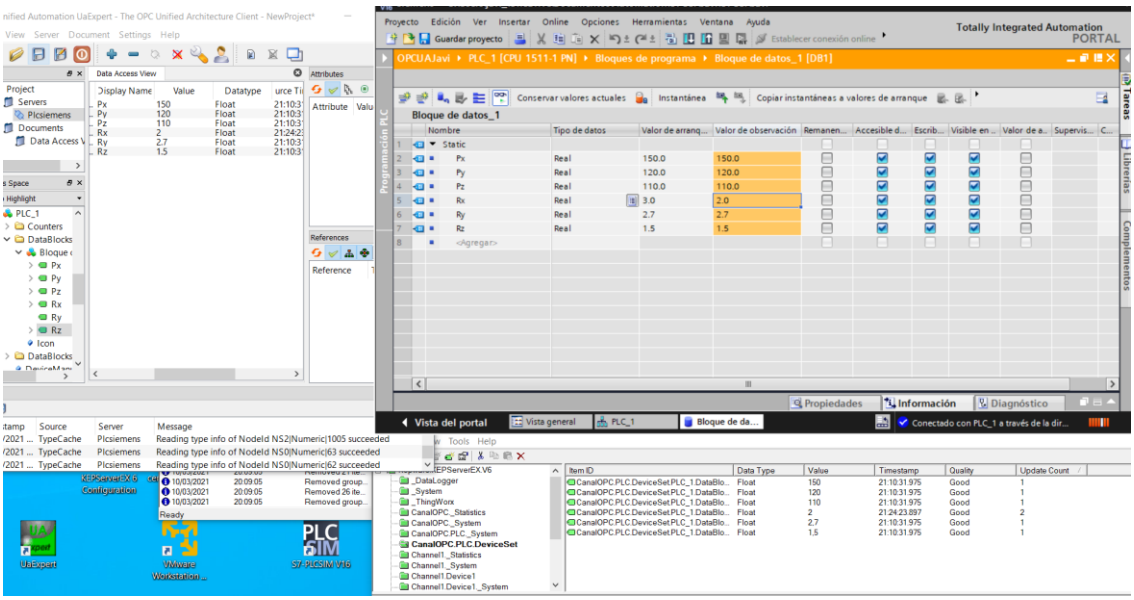


Figura 25 Visualizando las variables en Tia Portal y los dos Clientes

2.3.2 Comunicación entre un PLC Beckhoff y cliente

En este ejemplo con ayuda de [7] se crea la comunicación Cliente-Servidor, se hará la simulación de un PLC Beckhoff como servidor y se elige en este caso un cliente UAExpert, la configuración del cliente está descrita en el por lo que este apartado se centrará principalmente en la configuración del PLC en TwinCat 3 para que simule un servidor OPC UA.

En este caso la prueba se hará con una simulación de un parking, se tiene un semáforo que tiene una variable booleana, un contador para ver los coches que entran al parking y dos sensores, uno de entrada y otro de salida.

Lo primero que se debe de hacer es ingresar en la página oficial de Beckhoff y descargar TwinCat 3.0 y la función para poder hacer la comunicación OPC UA

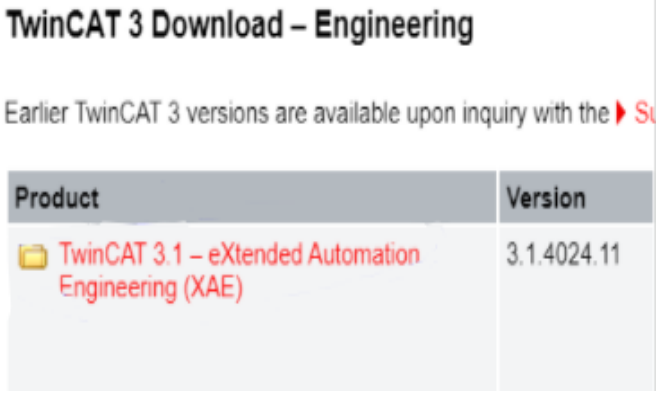


Figura 27 Página de Beckhoff

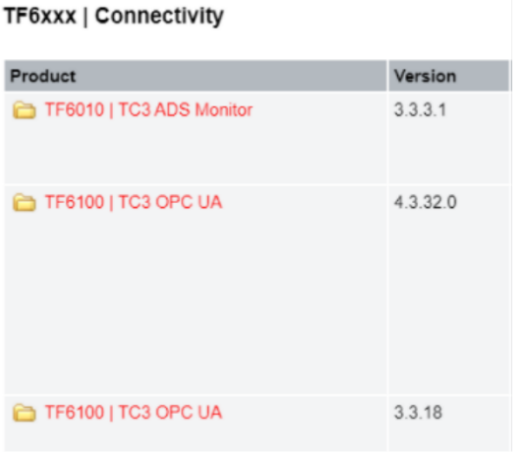


Figura 26 Página de Beckhoff funciones

Configuración en la parte del servidor (Beckhoff)

Una vez instalado el programa se activan las licencias, en la Figura 28 se visualiza la pantalla de TwinCat donde se muestra los pasos que hay que ejecutar.

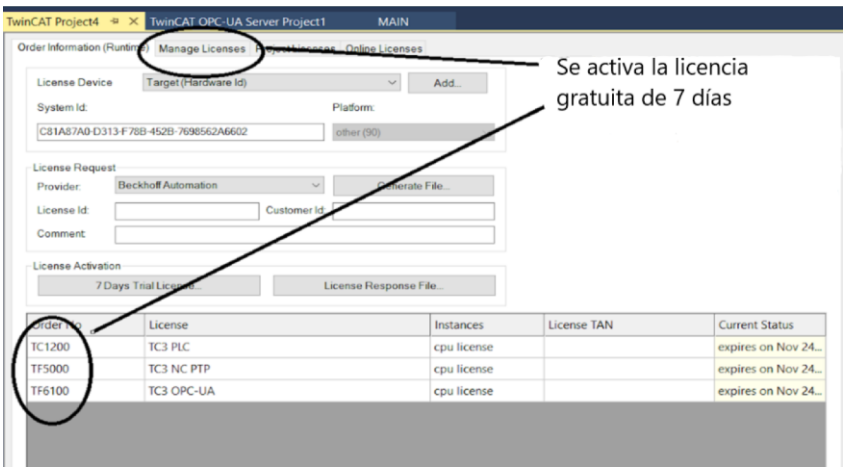


Figura 28 Activar licencias de TwinCat 3

Ahora el siguiente paso será realizar la comunicación OPC UA con la ayuda de un proyecto sencillo que se realizó el año pasado en la asignatura de Domótica e Inmótica.

Se podrá ver la programación y el funcionamiento de dicha simulación en el Código QR 2 Código del ejemplo y Código QR 3 Interfaz gráfica del ejemplo.

Código QR 3 Interfaz gráfica del ejemplo

Código QR 2 Código del ejemplo

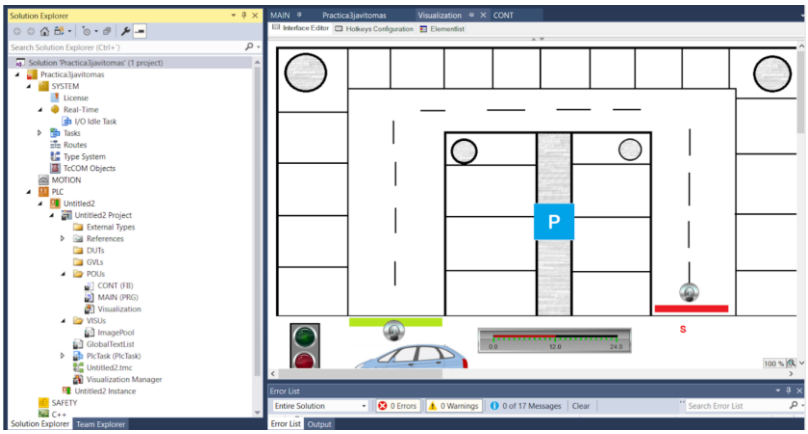


Figura 29 Ejemplo en TwinCat parking

Para configurar la comunicación hay que seguir los siguientes pasos que muestra en Figura 31 y Figura 30, pulsar el botón derecho del ratón y añadir el elemento connectivity al proyecto, después de realizar esta acción se tiene que activar la casilla de TMC File.

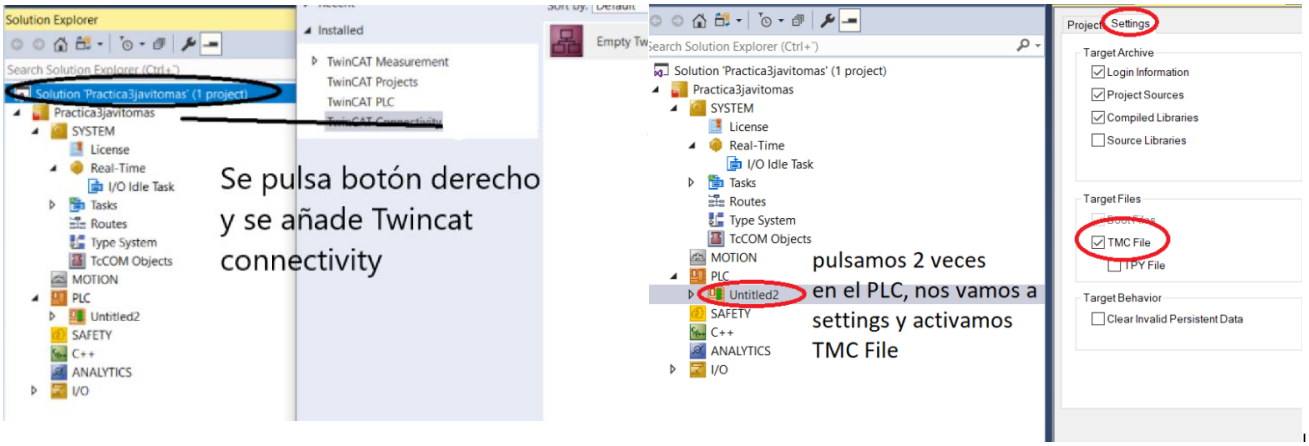


Figura 30 Propiedades del proyecto en TwinCat

Figura 31 Crear dispositivo Opc UA

Una vez creado la conectividad y se elige OPC UA, saldrá en el árbol del proyecto DA, HA y HDA, en este caso interesa un Data Access para ver las variables en tiempo real del PLC, se realizarán los pasos que se muestra en la Figura 32.

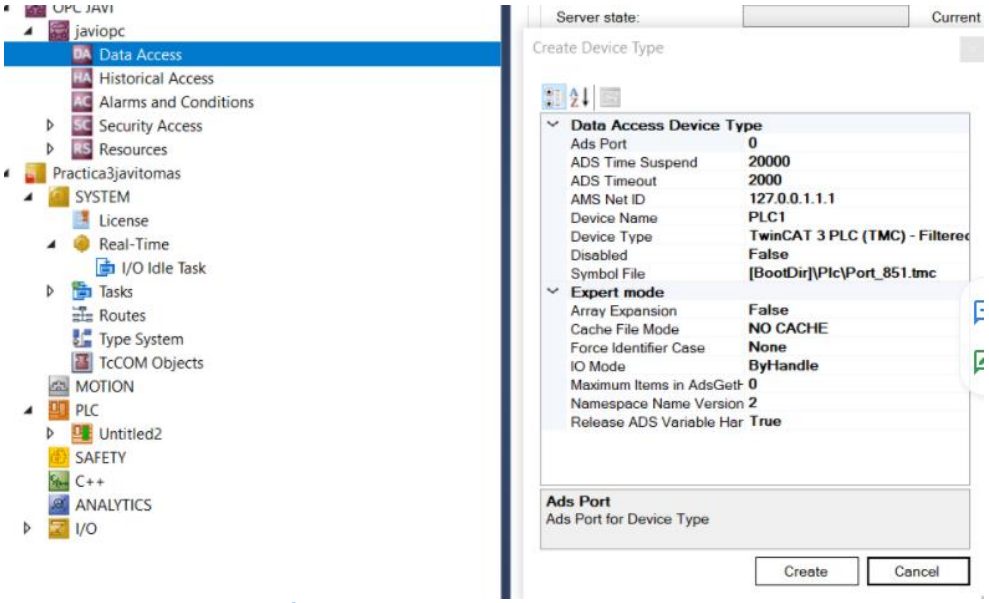


Figura 32 Árbol de programa de TwinCat

Se debe de pulsar el botón derecho, después se selecciona “add ítem” en DATA ACCESS y aparece la venta que se muestra en la Figura Configuración OPC UA.

IMPORTANTE: en Ads Port, se pone 851 ya que es el puerto del PLC, Otro parámetro que hay que configurar es el AMS Net ID, al ser una simulación desde el PC, se ejecuta el paso que se muestra en Figura 33.

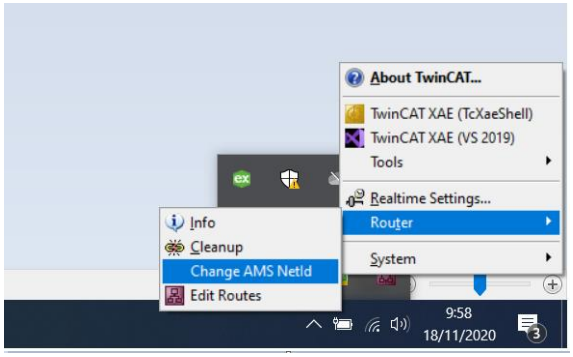


Figura 33 Cambiar AMS Net ID

El ID que se muestre en Change AMS Netid se copia y se pega en las propiedades del PLC

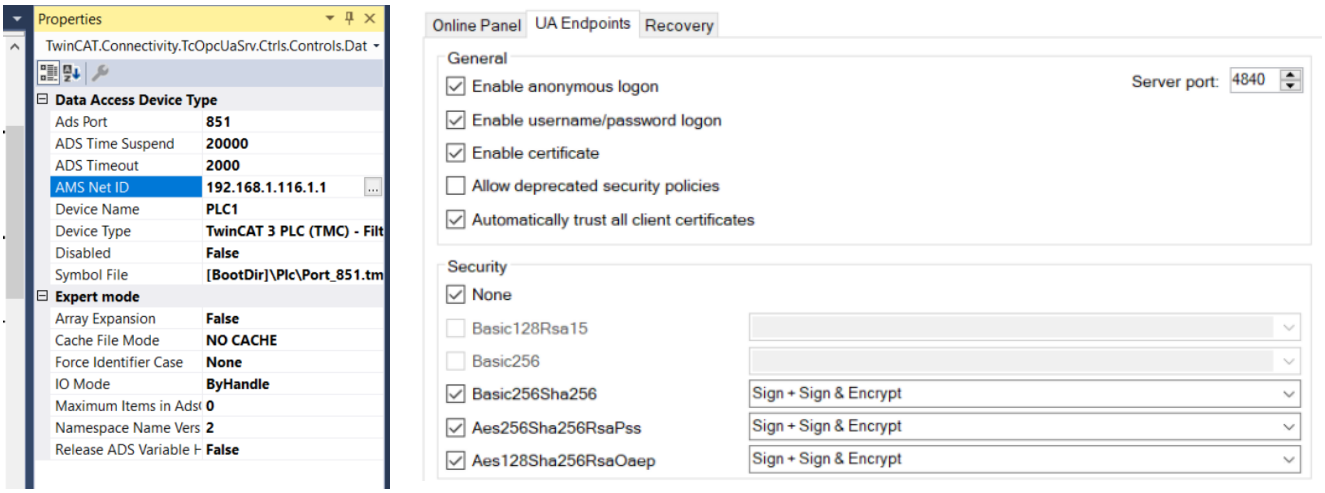


Figura 34 Propiedades del Servidor Opc UA

En la Figura 34 se muestra cual es el puerto del Servidor (server port) que es 8440, se puede cambiar pero en este ejemplo se dejará por defecto, se puede cambiar la configuración de seguridad y añadir certificados con contraseña.

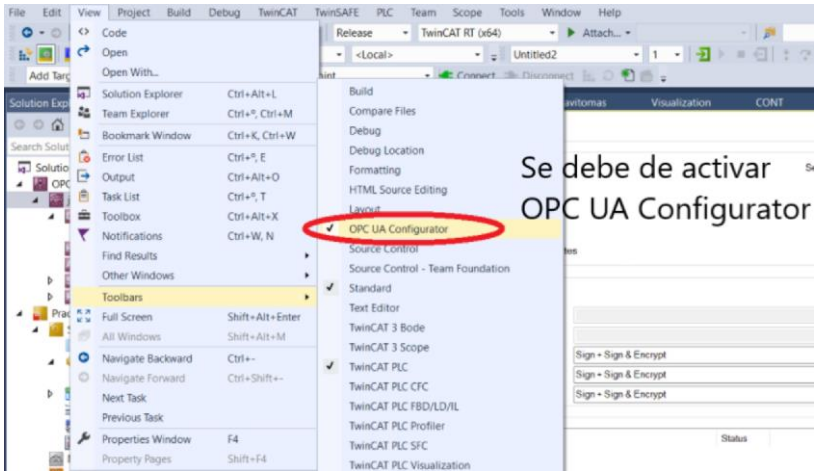


Figura 35 Configuración Opc UA

El siguiente paso que hay que hacer es pulsar en **Add Target OPC-UA** Server, que por defecto vendrá en la esquina superior derecha y se introduce la dirección IP del router como se muestra en la siguiente figura, para poder visualizar la dirección IP, se pulsa en las propiedades de internet y se introduce la dirección IPv4.

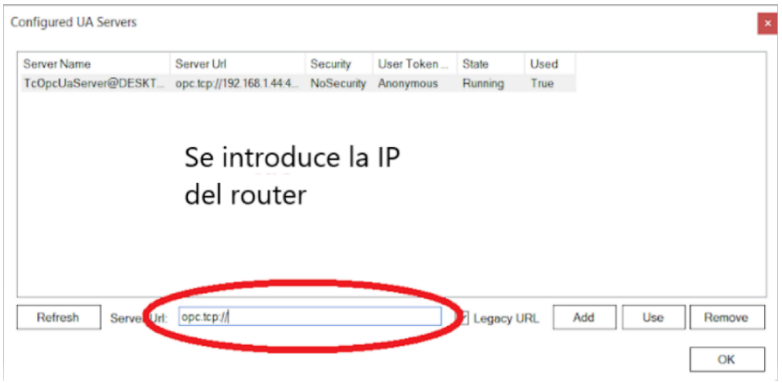


Figura 36 Añadir Servidor OPC UA

Tiene que salir un Pop-Up donde se muestre que se está ejecutando el servidor como se muestra en la Figura 37.

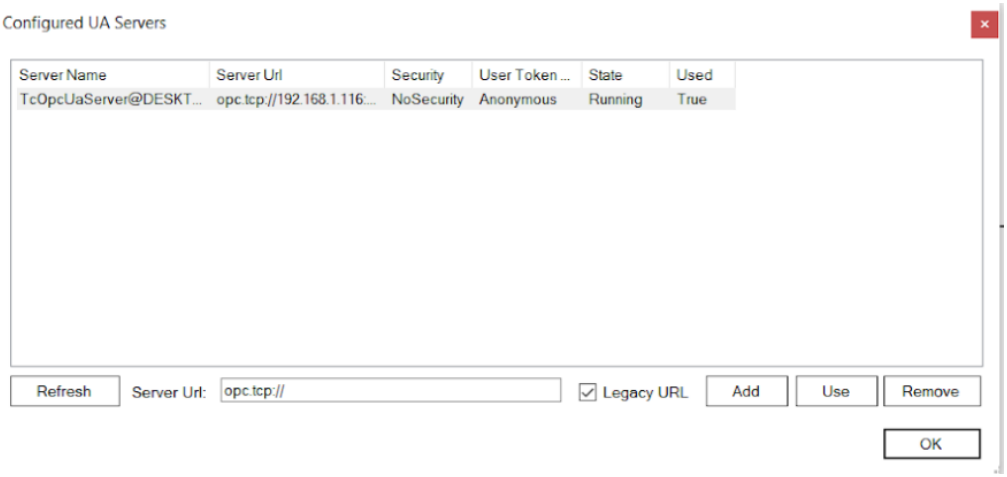


Figura 37 Pop-Up con el Servidor en "running"

Ahora se elige un *endpoint* para poder conectarse y ya estaría todo completamente configurado.

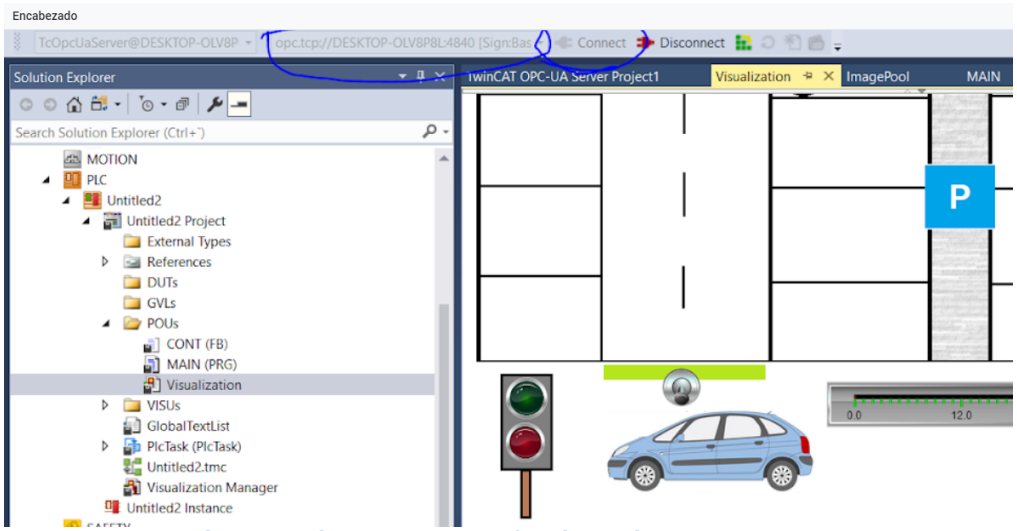


Figura 38 Figura 5: OPC UA funcionando correctamente

Por último, para que se vean las variables en el cliente se tiene que tener en cuenta que hay que poner algunos de los siguientes atributos que se muestra en la Figura 39 que proporciona Beckhoff en la documentación sobre OPC UA.

TwinCAT 3 (with TMC import):

Parameter	Comment	Description
Activate	{attribute 'OPC.UA.DA' := '1'}	Activates the data access symbol. This is necessary to make the symbol available for Historical Access.
Activating for HA	{attribute 'OPC.UA.HA' := '1'}	Activates the symbol for Historical Access.
Storage medium	{attribute 'OPC.UA.HA.Storage' := '1'}	Specifies the storage medium to be used for storing the symbol values (1 = RAM, 2 = File, 3 = SQL Compact Database, 4 = SQL Database).
Sampling rate	{attribute 'OPC.UA.HA.Sampling' := '50'}	Defines the rate at which the variable values are to be read from the controller and stored on the corresponding storage medium. The OPC UA Server uses predefined fixed sampling rates that can be used here. You can expand the sampling rates if necessary in the ServerConfig file. However, this is not normally necessary, and it is not advisable to change the settings, unless there is a good reason.
Buffer length	{attribute 'OPC.UA.HA.Buffer' := '10000'}	Specifies the number of elements to include in the buffer.

Figura 39 Propiedades de un Tag para Opc UA

En este ejemplo se busca visualizar las variables en DataAccess, se pone el primer atributo que hay en la Figura 39 en la parte superior de cada variable que se quiere mostrar, a continuación, se muestra en la Figura 40 este proceso.

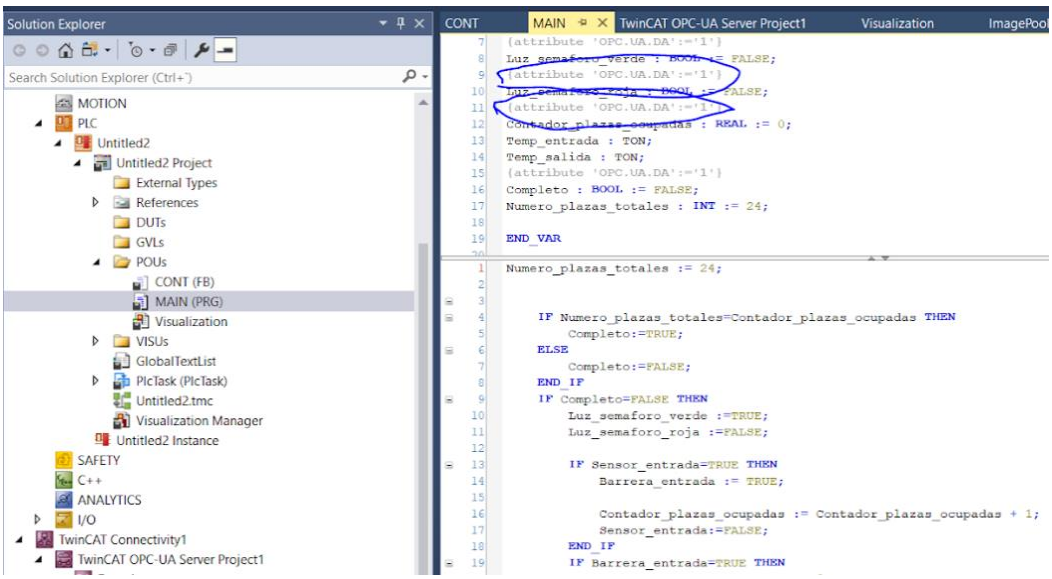


Figura 40 Atributos Opc UA en Tags del PLC

Una vez realizado las siguientes configuraciones se compila el programa y ya estaría todo listo para visualizarlo en la parte del cliente

Configuración en la parte del Cliente (OPC Cliente)

Esta vez se utiliza un Cliente distinto a Kepserverex, en concreto OpcClient.

Este programa no requiere tanta configuración, con tan solo añadir el endpoint del servidor ya proporciona las variables en la pantalla del Cliente.

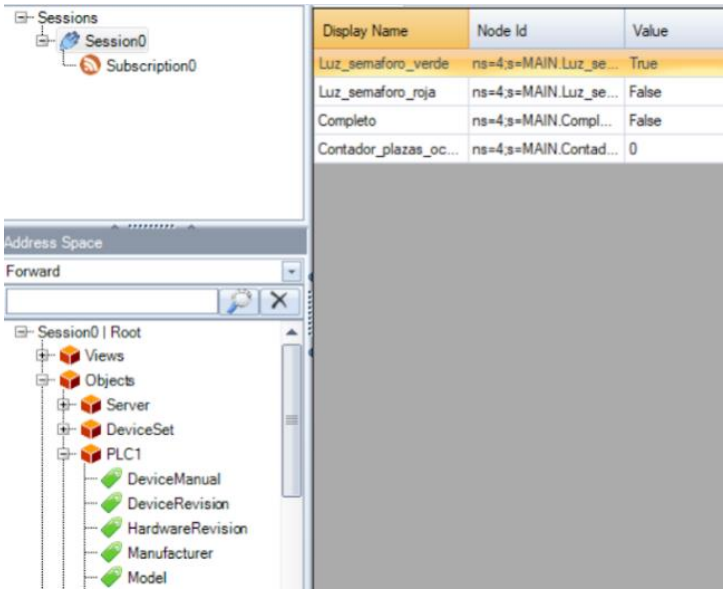


Figura 41 Cliente OPC UA

2.3.3 Comunicación entre S7-1500 Virtual y S7-1200 físico

En este ejemplo se realiza la comunicación entre un dispositivo real de la marca siemens, concretamente un Simatic S7-1200, con CPU 1214DC/DC/DC, con firmware 4.4 ya que la serie S7-1200 con firmware inferior no tiene la capacidad de tener la comunicación OPC UA y un S7-1500 simulado con PLCSIM Advanced.

La serie S7-1200 no puede actuar como Cliente, sino solo como Servidor por lo que hay una limitación al hacer la conexión entre PLC's a través de Tia portal por lo que se utilizará una de las herramientas que ofrece el software de Kepserver como es la conexión M2M (máquina-máquina), se situá en AdvancedTags en el programa y ofrece la posibilidad de comunicar PLC`s, el programa servirá de intermediario entre los dos PLC`s, en este ejemplo se conectan los dos servidores de los PLC's al Cliente de Kepserver y a la vez servirá de puente para poder ejecutar la comunicación entre ambos PLC's.

Se adjunta el Código QR 4 que lleva al vídeo donde se muestra el ejemplo en acción.

Código QR 4 Comunicación entre 2 PLC's

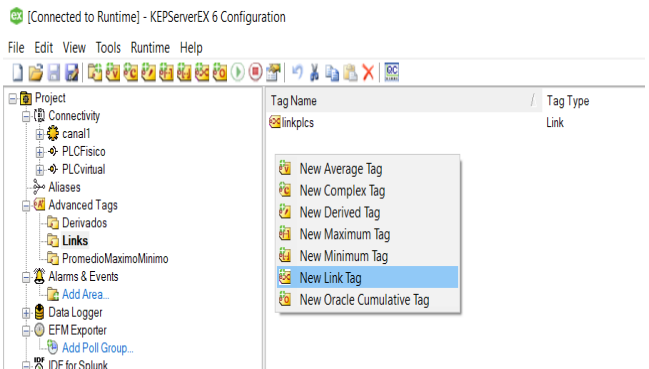
Figura 42 Ejemplo comunicación entre PLC's

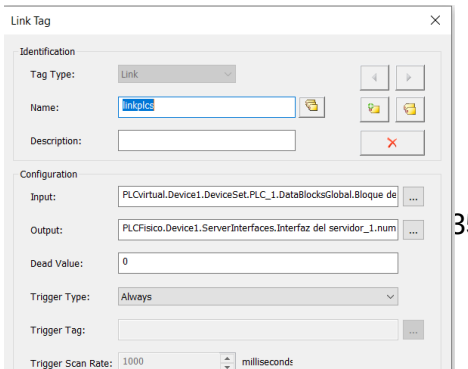
Para configurar los autómatas se siguen los mismos pasos que en el ejemplo 2.3.1 Comunicación PLC Siemens-KepServer-UaExpert por lo que se pasará directamente a mostrar la configuración necesaria en el Cliente para que funcione de puente entre ambos autómatas.

Una vez configurados hay que configurar el cliente Kepserver y crear un canal de comunicación para cada PLC's, añadir las variables que el cliente tenga acceso para escribir/leer en el cliente de Kepserver.

Para crear el canal de comunicación Kepserver tiene una herramienta para hacer la comunicación máquina-máquina en variables avanzadas.

Se crean nuevas variables avanzadas y se pulsa añadir nueva variable, seguidamente se añade una variable de tipo Link, se añaden las direcciones de las variables de entrada y de salida, donde se escribirán esos datos, se pulsa en aceptar y ya está realizada la conexión entre los dos PLC's, en la Figura 43 y Figura 44 se muestra el proceso.





En Figura 45 se muestraa el ejemplo con la comunicación realizada correctamente.

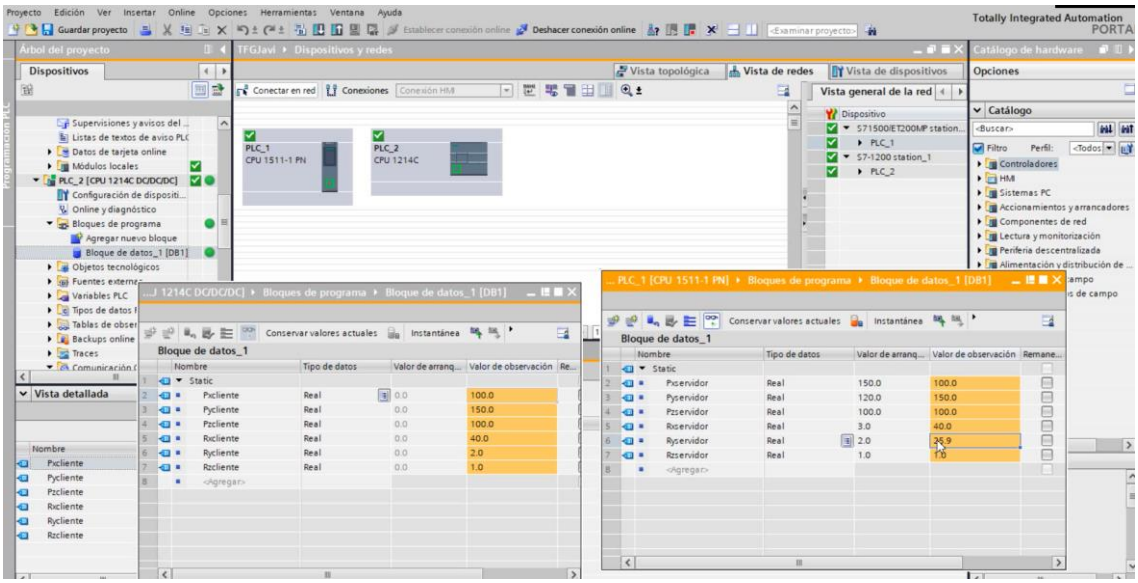


Figura 45 Comunicación entre 2 PLC's

3. MQTT (Message Queue Telemetry Transport)

3.1 Historia del protocolo MQTT

Según recoge [3] MQTT fue creado por el Dr. Andy Stanford-Clark de IBM y Arlen Nipper de Arcom – ahora Eurotech – en 1999. MQTT fue creado como una forma rentable y fiable de conectar los dispositivos de monitorización utilizados en las industrias del petróleo y el gas a servidores empresariales remotos. Cuando fueron desafiados con encontrar una manera de empujar los datos de los sensores de la tubería en el desierto a los sistemas de control de supervisión y adquisición de datos (SCADA) fuera del sitio, decidieron una topología de publicación/suscripción basada en TCP/IP que sería impulsada por los eventos para mantener bajos los costos de transmisión del enlace satelital.

Aunque MQTT sigue estando estrechamente asociado con IBM, es ahora un protocolo abierto que es supervisado por la Organización para el Avance de los Estándares de Información Estructurada (OASIS).

MQTT fue previamente conocido como el protocolo SCADA, MQ Integrator SCADA Device Protocol (MQIsdp) y WebSphere MQTT (WMQTT), aunque todas estas variaciones han caído en desuso.

3.2 Introducción MQTT

En este apartado se hará una breve introducción sobre MQTT y se hablará sobre las especificaciones de MQTT, unos de los protocolos de comunicación más utilizados en la actualidad, MQTT, cuyas siglas son *Message Queue Telemetry Transport*, es un protocolo de mensajería “ligero” basado en la comunicación Publicación-Suscripción diseñado para telemetría M2M (máquina a máquina) en entornos de bajo ancha de banda.

Hay que destacar sobre este protocolo que se está convirtiendo rápidamente en uno de los principales protocolos para despliegues de IoT.

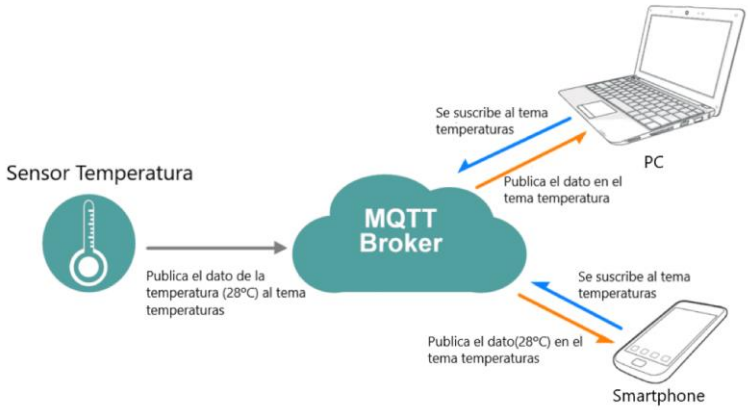


Figura 46 Ejemplo publicación-suscriptor en MQTT

Es interesante ya que se puede desarrollar comunicaciones entre sensores y cualquier tipo de autómata, Arduino o Raspberry Pi por lo que para desarrollar aplicaciones de domótica podría ser interesante.

MQTT es un protocolo pensado para IoT que está al mismo nivel que HTTP o CoAP:

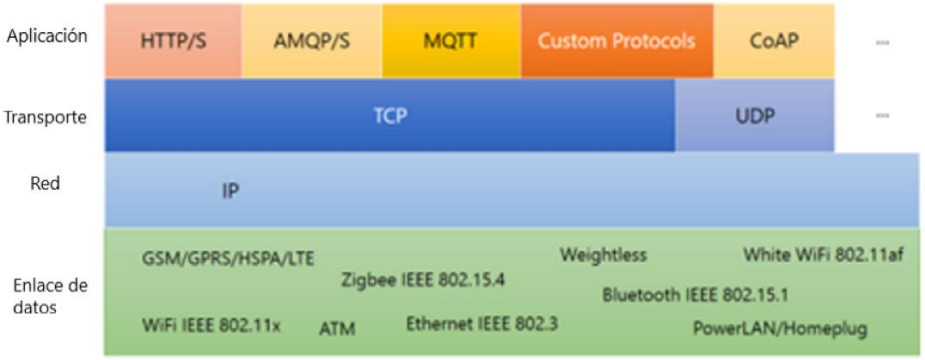


Figura 47 Nivel de MQTT en los niveles de capa

De acuerdo con [3], un aspecto importante a tener en cuenta de los dispositivos IoT no es solamente el poder enviar datos al Cloud/Servidor, sino también el poder comunicarse con el dispositivo. Este es uno de los beneficios de MQTT: es un modelo Servidor-Red, el cliente abre una conexión de salida al bróker, aunque el dispositivo esté actuando como Publicador o Suscriptor. Esto normalmente evita los problemas con los firewalls porque funciona detrás de ellos o vía NAT.

3.3 Cómo funciona MQTT

Antes de entrar en el funcionamiento del protocolo se hace una breve introducción sobre MQTT, algunos de los términos que se usará en este capítulo y posteriormente como recoge [4].

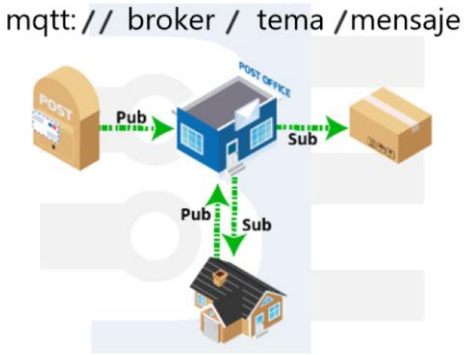


Figura 48 MQTT mensaje

- Broker/Servidor, El servidor (o servidor) MQTT es el servidor central al que se conectan los clientes MQTT. El servidor MQTT gestiona los temas de mensajes. Cuando un cliente publica un mensaje en un tema, el servidor de mensajes envía una copia del mensaje a cada uno de los clientes que están suscritos al tema.
- Cliente, el dispositivo que se conecta al servidor para enviar o recibir información.
- Tópico, el nombre del mensaje. Los clientes publican, se suscriben o hacen ambas cosas a un tema.
- Publicar, cuando un cliente publica en un tema en el servidor MQTT, está actualizando los datos asociados a ese tema en el servidor, publicando nuevos mensajes para los suscriptores del tema (si los hay).
- Suscripción Una vez que un cliente se suscribe a un tema en el servidor MQTT, recibirá todos los mensajes posteriores que se hayan publicado en el tema. No hay límite en el número de clientes que pueden suscribirse a un tema.
- Comodines de temas, Los comodines de temas permiten que una sola suscripción reciba actualizaciones de varios temas.
- El comodín multinivel ('#') se utiliza para especificar todos los niveles restantes de las jerarquías y debe ser el último carácter de la cadena de suscripción a temas.

  Ej: Si el cliente se suscribe a "mi_casa/#", entonces el cliente recibirá los mensajes publicados en "mi_casa/sala_familiar" y "mi_casa/cocina".

- El comodín de un solo nivel ('+'), se utiliza para coincidir con un nivel de jerarquía del tema. El comodín de un nivel ('+') puede utilizarse más de una vez en el filtro de temas y junto con el comodín de varios niveles.

Ej: Si el cliente se suscribe a "mi_casa/+/temperatura", recibirá los mensajes publicados tanto en "mi_casa/sala_familiar/temperatura" como en mi_casa/cocina/temperatura".

- Calidad de servicio (QoS), se refiere a la fiabilidad de la entrega de mensajes entre el editor y el suscriptor. MQTT define tres niveles de calidad de servicio.
    - QoS0 es el nivel de fiabilidad más bajo y se define como la entrega "como máximo una vez". Los mensajes QoS0 son los más rápidos de enviar, pero no garantizan la entrega. Esto es análogo a un paquete UDP en una red IP: el protocolo UDP/IP no garantiza que los paquetes UDP lleguen a su destino.
    - QoS1 se define como entrega "al menos una vez". Se garantiza la entrega de los mensajes de QoS1 (si la entrega es factible), pero pueden ser entregados varias veces (duplicados).
    - QoS2 es el nivel de fiabilidad más alto y se define como entrega "exactamente una vez". Se garantiza la entrega de los mensajes de QoS2 (si la entrega es factible) y se garantiza que no se duplican. QoS2 es el nivel de calidad de servicio más seguro pero más lento.

Una sesión MQTT se divide en cuatro etapas: conexión, autenticación, comunicación y terminación.

Un cliente comienza creando una conexión de Protocolo de Control de Transmisión/Protocolo de Internet (TCP/IP) con el servidor utilizando un puerto estándar o un puerto personalizado definido por los operadores del servidor.

Al crear la conexión, es importante reconocer que el servidor puede continuar una sesión antigua si se le proporciona una identidad de cliente reutilizada.

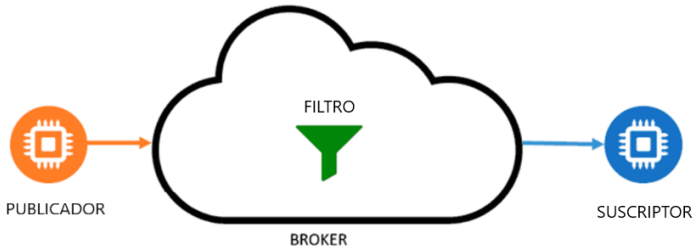


Figura 49 Publicador-Servidor -Suscriptor

Los puertos estándar son 1883 para la comunicación no cifrada y 8883 para la comunicación cifrada – usando Secure Sockets Layer (SSL)/Transport Layer Security (TLS).

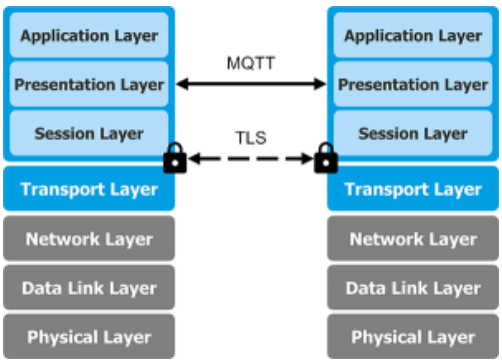


Figura 50 Diferentes Capa OSI

Durante la conexión SSL/TLS, el cliente valida el certificado del servidor y autentica el servidor. El cliente también puede proporcionar un certificado de cliente al agente durante el apretón de manos.

El agente puede utilizarlo para autenticar al cliente. Aunque no es específicamente parte de la especificación MQTT, se ha convertido en una costumbre que los agentes soporten la autenticación de clientes con certificados SSL/TLS del lado del cliente.

Debido a que el protocolo MQTT tiene como objetivo ser un protocolo para dispositivos con recursos limitados y dispositivos de IO, SSL/TLS puede no ser siempre una opción y, en algunos casos, puede no ser deseado.

MQTT se llama un protocolo ligero porque todos sus mensajes tienen una pequeña huella de código.

Cada mensaje consiste en un encabezado fijo — 2 bytes — un encabezado variable opcional, una carga útil del mensaje que está limitada a 256 megabytes (MB) de información y un nivel de calidad de servicio (QoS).

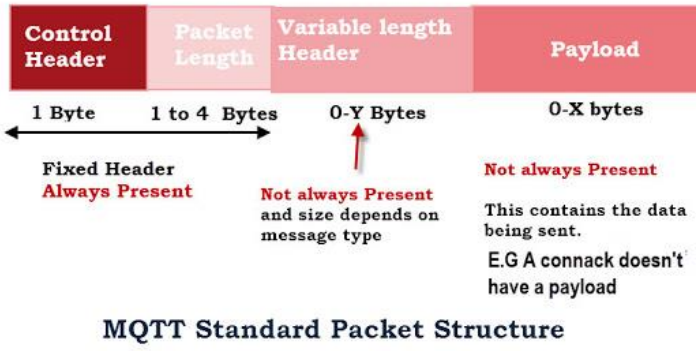


Figura 51 MQTT estructura de un mensaje

Durante la fase de comunicación, un cliente puede realizar operaciones de publicación, suscripción, des-inscripción y ping. La operación de publicación envía un bloque binario de datos (el contenido) a un tema definido por el editor.

MQTT soporta grandes objetos binarios de mensajes (BLOBs) de hasta 256 MB de tamaño. El formato del contenido será específico de la aplicación. Las suscripciones a los temas se realizan utilizando un par de paquetes SUBSCRIBE/SUBACK, y la cancelación de la suscripción se realiza de forma similar utilizando un par de paquetes UNSUBSCRIBE/UNSUBACK.

Los strings de temas forman un árbol temático natural con el uso de un carácter delimitador especial, la barra oblicua (/). Un cliente puede suscribirse y cancelar la suscripción a ramas enteras del árbol de temas utilizando caracteres comodines especiales.

Otra operación que un cliente puede realizar durante la fase de comunicación es hacer ping al servidor del agente usando una secuencia de paquetes PINGREQ/PINGRESP. Esta operación no tiene otra función que la de mantener una conexión viva y asegurarse de que la conexión TCP no ha sido cerrada por un gateway o router.

Cuando un editor o suscriptor quiere terminar una sesión MQTT, envía un mensaje de DESCONEXIÓN al servidor luego cierra la conexión. Esto se llama un cierre elegante porque le da al cliente la capacidad de reconectarse fácilmente proporcionando su identidad de cliente y reanudando donde lo dejó.

Aplicaciones y casos de uso del protocolo MQTT

La mayoría de los principales proveedores de servicios en la nube, incluyendo Amazon Web Services (AWS), Google Cloud, IBM Cloud y Microsoft Azure, soportan MQTT.

MQTT es muy adecuado para aplicaciones que utilizan dispositivos M2M e IoT para propósitos tales como análisis en tiempo real, mantenimiento preventivo y monitorización en entornos, incluyendo hogares inteligentes, sanidad, logística, industria y fabricación.

¿Quién utiliza MQTT?

MQTT es usado por compañías importantes, especialmente en los sectores de automoción, industria 4.0, transporte y entretenimiento. MQTT se utiliza para el intercambio de datos entre dispositivos restringidos y aplicaciones de servidor.

3.4 Ejemplos de aplicación MQTT

3.4.1 Comunicación MQTT básica

En este ejemplo se realiza una comunicación sencilla entre la consola de Windows, para realizarla lo primero que se debe de hacer es descargar Mosquitto Eclipse, uno de los programas más utilizados para realizar este tipo de comunicación.

Una vez descargado se abre la consola dos veces, un CMD actuará como publicador y la otra ventana de cmd actuará como suscriptor.

Mosquitto enviará la información por el puerto 1883, para comprobar el correcto funcionamiento del puerto se tendrá que introducir en la consola el siguiente comando:

Netstat –an, una vez que se ejecuta este comando se tendrá que observar que aparece el puerto TCP 0.0.0.0:1883 como se muestra en la Figura 52.

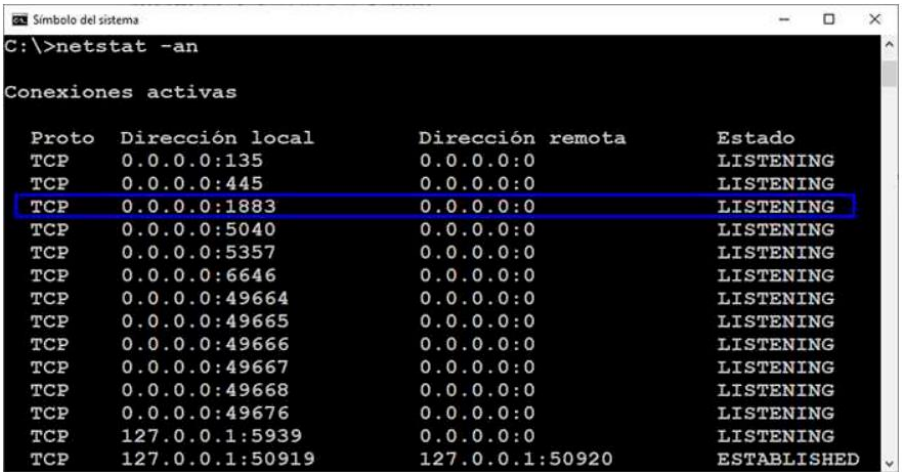


Figura 52 Conexiones activas

Para hacer la publicación en la ventana de CMD se utiliza la ayuda de Mosquitto para ver los comandos que se tienen que poner para la correcta publicación de un tema.

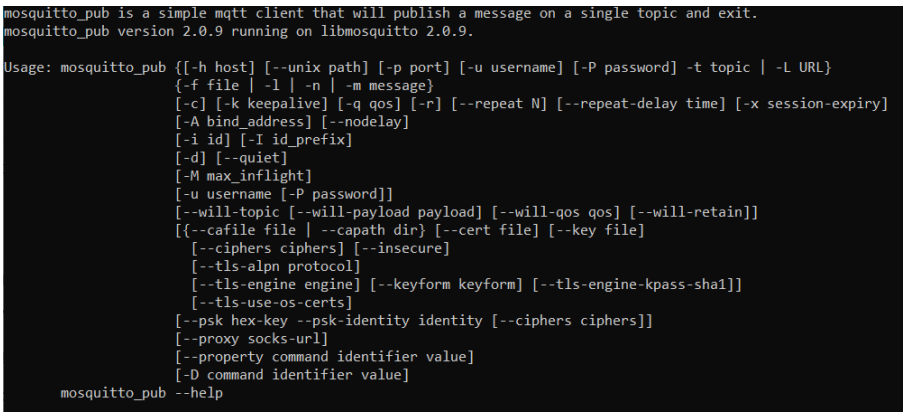


Figura 53 Ayuda Mosquitto

Para publicar un dato se pondrá con la siguiente estructura:

Mosquitto_pub –h 127.0.0.0.1 –p 1883 –t [nombre del tema] –m [valor de la variable]

Para suscribirse a un tema se pondrá la siguiente estructura:

Mosquitto_sub –h 127.0.0.0.1 –p 1883 –t [nombre del tema]

En la Figura 54 se muestra como se ha publicado y se ha suscrito al tema “Posición Robot”, en este ejemplo hay varias posiciones de las distintas articulaciones de un robot, para poder suscribirse a todos los datos se tendrá que poner en la CMD de suscripción –t [nombre del tema/#], este símbolo (#) nos permite realizar la lectura de los datos de este tema.

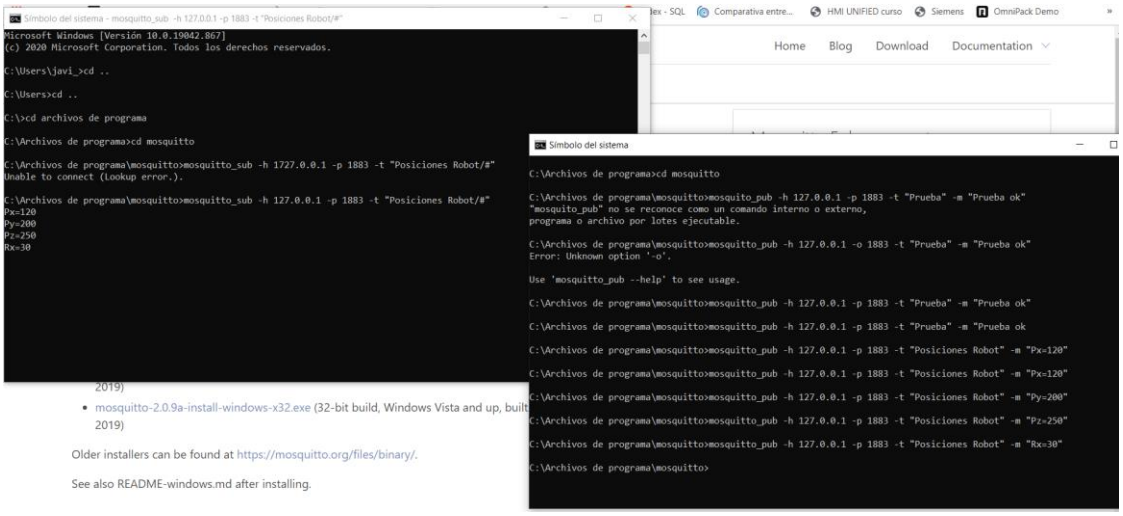


Figura 54 Suscripción y publicación en Mosquitto

Para comprobar la información que se publica en Mosquitto se tiene la ayuda de MQTT Explorer que permite visualizar todas las variables que han sido publicadas al Servidor, en la Figura 55 se visualiza como el Tema Posiciones Robot tiene varios datos.

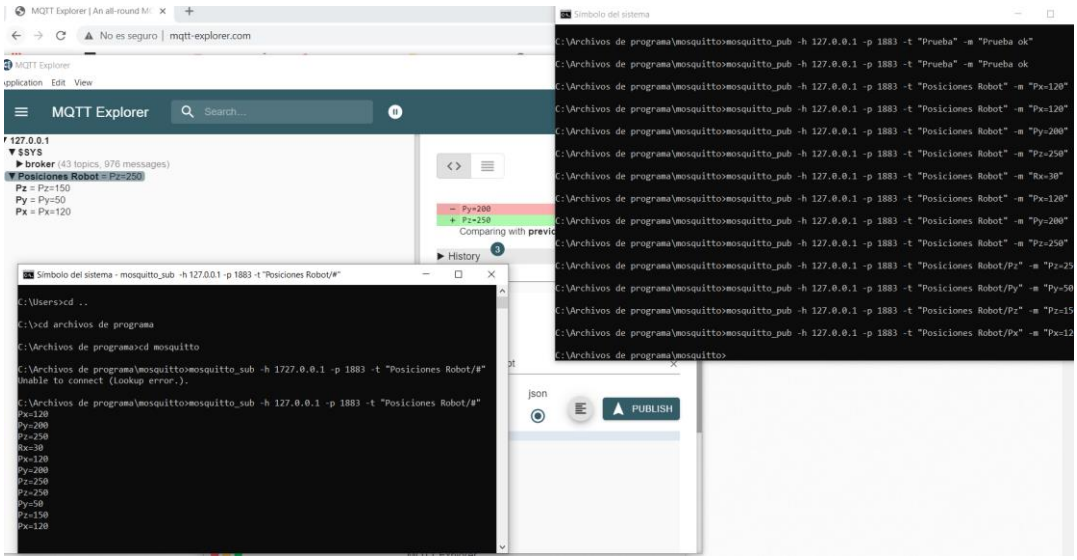


Figura 55 MQTT Explorer

3.4.2 Comunicación entre Node-Red y Ubidots

En este ejemplo que se realiza con la ayuda de [8] se realiza la comunicación mediante MQTT (Message Queue Telemetry Transport), que en este caso se utiliza Ubidots, seguidamente un suscriptor que se suscribirá al tema y recibirá la información que necesita.

Se ha elegido Node-Red ya que permite ver de manera sencilla y visual el funcionamiento de este protocolo, destacar que Ubidots ha desarrollado el envío mediante Mqtt en Node-Red por lo que se utilizará las funciones creadas por dicha plataforma, aunque Node-Red tiene sus funciones de MQTT, se tiene que descargar de la librería las funciones de Ubidots, para ello en la paleta de descargas de Node-red se busca 'Ubidots Node-red' e instalamos las funciones, este ejemplo se ha realizado siguiendo la guía de ayuda que proporciona la plataforma Ubidots llamado "Conectar Node-RED con Ubidots".

Se adjunta Código QR 5 que lleva al vídeo donde se muestra el ejemplo en acción.

Código QR 5 Ubidots y Node-red

Se sigue el esquema de comunicación de la Figura 56.

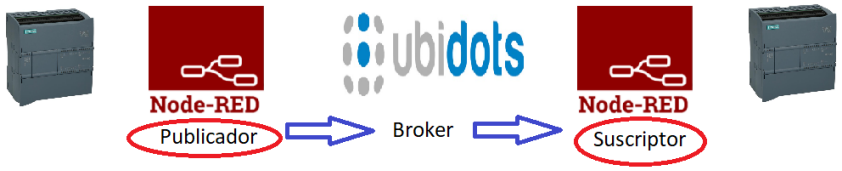


Figura 56 Esquema de comunicación entre Ubidots y Node-Red

Se lee la variable de un autómata en Node-red y se Publica a Ubidots, Node-red funcionará como Suscriptor y Publicador por lo que Node-red se suscribe a ese tema y lo envía al autómata.

Lo primero que se debe de hacer es configurar el Endpoint y las variables que se tienen que publicar en el Servidor, para realizar esta tarea Node-red tiene una librería para la lectura y escritura de los autómatas en Node-red, S7- In y S7- Out, en el flujo de programa de Node-red que se muestra en la Figura 57 se muestra en la parte izquierda las distintas funciones que tiene este programa.

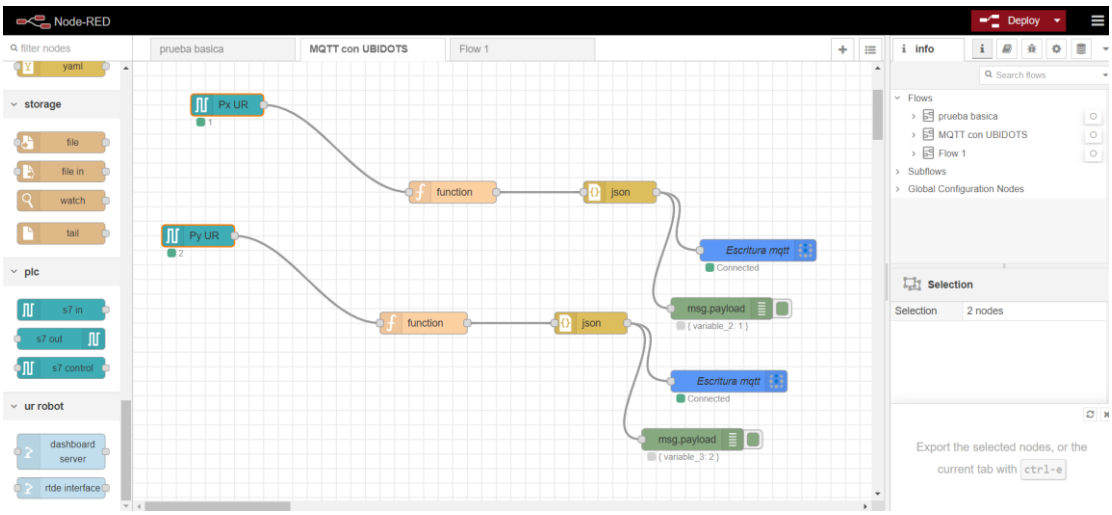
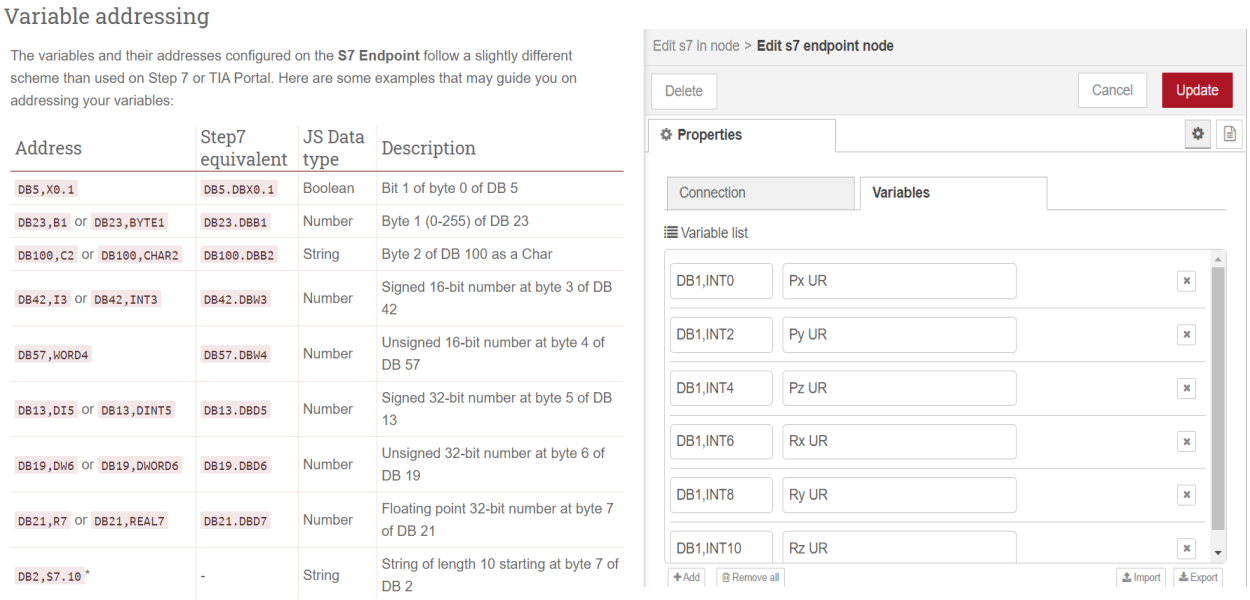


Figura 57 Publicación del autómata al Broker

El siguiente paso que se debe de hacer es configurar la lectura y escritura del PLC en Node-Red e introducir la dirección del PLC y la dirección de las variables que tendrán el formato que se muestra la Figura 58:



Variable addressing

The variables and their addresses configured on the **S7 Endpoint** follow a slightly different scheme than used on Step 7 or TIA Portal. Here are some examples that may guide you on addressing your variables:

Address	Step7 equivalent	JS Data type	Description
`DB5,X0.1`	`DB5.DBX0.1`	Boolean	Bit 1 of byte 0 of DB 5
`DB23,B1` or `DB23,BYTE1`	`DB23.DBB1`	Number	Byte 1 (0-255) of DB 23
`DB100,C2` or `DB100,CHAR2`	`DB100.DBB2`	String	Byte 2 of DB 100 as a Char
`DB42,I3` or `DB42,INT3`	`DB42.DBW3`	Number	Signed 16-bit number at byte 3 of DB 42
`DB57,WORD4`	`DB57.DBW4`	Number	Unsigned 16-bit number at byte 4 of DB 57
`DB13,DI5` or `DB13,DINT5`	`DB13.DBD5`	Number	Signed 32-bit number at byte 5 of DB 13
`DB19,DW6` or `DB19,DWORD6`	`DB19.DBD6`	Number	Unsigned 32-bit number at byte 6 of DB 19
`DB21,R7` or `DB21,REAL7`	`DB21.DBD7`	Number	Floating point 32-bit number at byte 7 of DB 21
`DB2,S7.10` *	-	String	String of length 10 starting at byte 7 of DB 2

Figura 58 Direcciones de las variables en Node-Red

Para introducir la dirección del PLC se pulsa sobre el icono de S7-In y se introduce la información que necesita Node-Red.

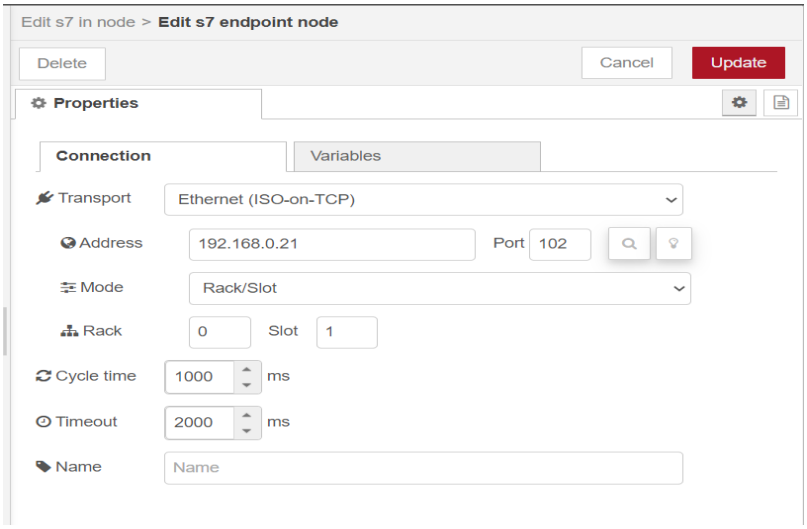


Figura 59 Dirección del PLC en Node-Red

Una vez realizada la lectura de los datos en Node-red se tiene que publicar en el bróker el tema y el valor de dicha variable como se observa en la Figura 60, para publicar en Ubidots el mensaje tiene que estar en formato Json por lo que Node-Red facilita esta conversión con una función que se encuentra en la paleta de la parte izquierda. Para publicar y suscribirse al tema se introducen las funciones de MQTT que ofrece Ubidots y se configura.

Como se observa en la Figura 60 se puede ver que el “Device Label” es el tópico y se tiene que introducir un token, que es una llave de acceso a la nube que proporciona Ubidots.

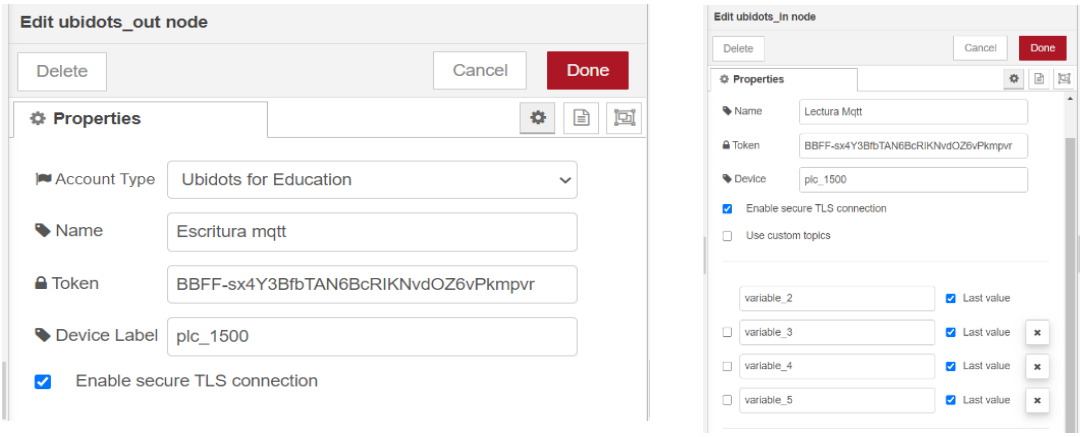


Figura 60 Parámetros de Ubidots introducidos en Node-Red

Para comprobar que se ha realizado correctamente la publicación se puede comprobar en la página de Ubidots (Figura 61).

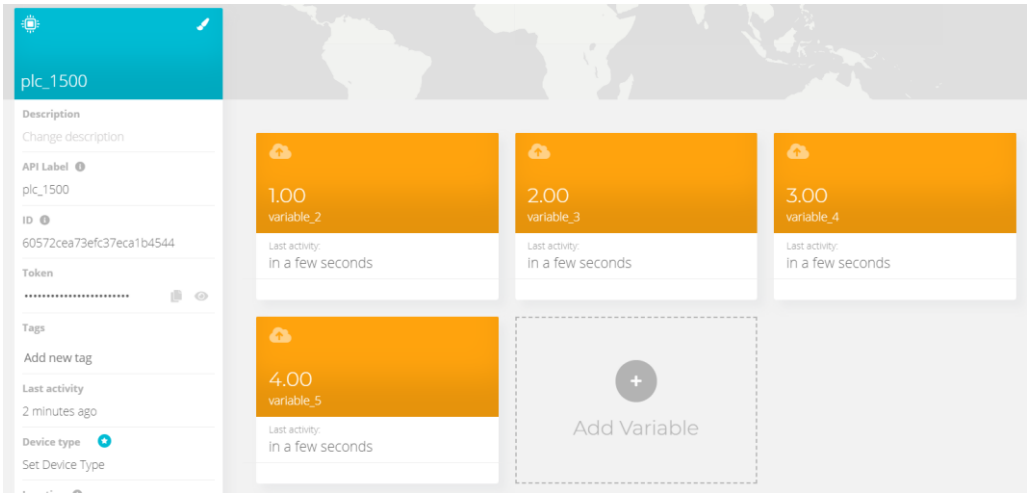


Figura 61 Comprobación de envió de datos a la Nube

Una vez publicado en el bróker un tema junto con su dato se tiene que configurar el suscriptor para que reciba ese dato, en este ejemplo el suscriptor será Node-red, se podría enviar ese dato a otro autómata mediante la función de node-red de “s7 Out”.

Para comprobar que la suscripción se ha realizado correctamente se añade una función para leer el mensaje que recibe Node-Red, en la Figura 62 se observa que en “msg.payload” se recibe el dato correctamente.

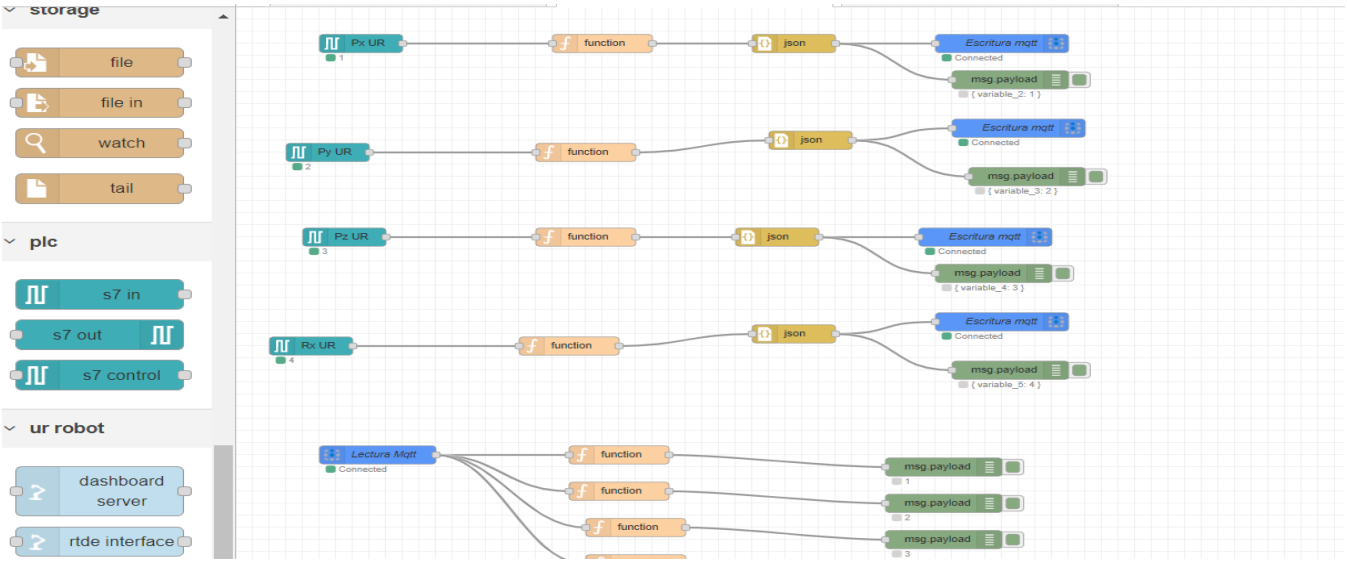


Figura 62 Nodos de suscripción en Node-red

4. CASO PRÁCTICO

4.1 Ejemplos de aplicación

En este último ejemplo se introducirán ambos protocolos para simular la obtención de datos de un robot para el control del brazo robótico.

La obtención de datos la hará una Simatic HMI de WinCC Unified que incorpora un servidor OPC, los datos enviados se simulan mediante UAExpert.

Para controlar los datos del entorno donde estará el robot situado se utilizan sensores descentralizados del autómata, en este caso un sensor de presencia que indicará si hay alguna persona en el perímetro en el que trabaja el robot y por otro lado un sensor de temperatura que indicará la temperatura de la sala donde se encuentre dicho brazo robótico, ambos parámetros serán recogidos por el suscriptor, en este caso Node-red que después enviará a un PLC 1500 y esté al HMI, el bróker será Ubidots y el publicador es también Node-Red.

El autómata está conectado a una Simatic HMI UNIFIED de 12'' donde se podrá visualizar los parámetros de los sensores y del robot.

Se adjunta Código QR 6 Caso práctico que lleva al vídeo donde se muestra el ejemplo en acción.

Código QR 6 Caso práctico

Lo primero que se hace en este ejemplo práctico es recibir los datos de un autómata a través de los nodos que proporciona Node-Red, en este caso se observa en la Figura 63 como recibe “Py UR” y “Pz UR”, ambos parámetros simulan la posición de un robot, seguidamente con una función se recoge el Payload del mensaje y se convierte en formato Json, ya que en Mqtt tiene que tener dicho formato el mensaje, una vez convertido el parámetro se publica al Servidor mediante el nodo “Escritura mqtt”, también se observa como desde Node-Red se publica unas variables de tipo entero simulando la temperatura, humedad y un sensor de presencia con valor 0 o 1.

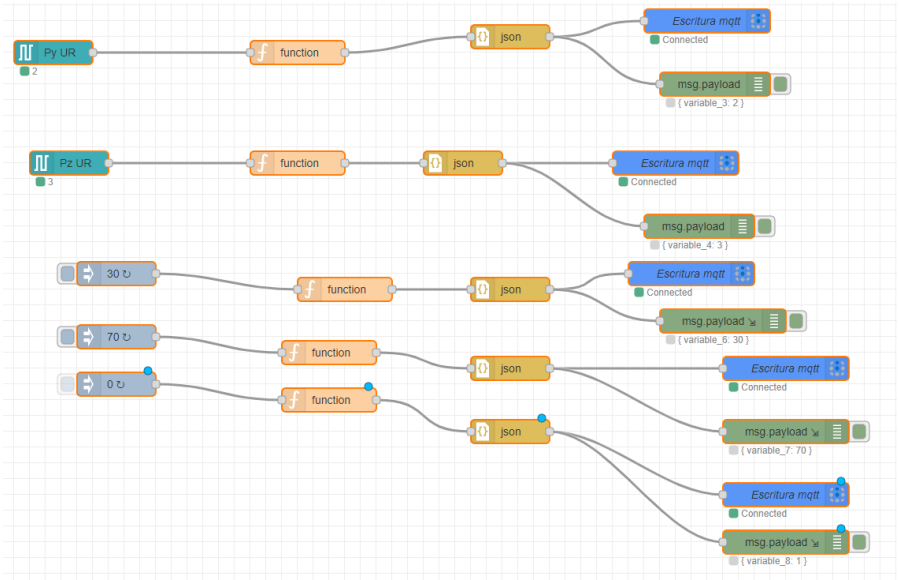


Figura 63 Esquema publicación parámetros a Ubidots

En la Figura 63 se implementa la suscripción de los parámetros que se necesitan, en este caso se le envía a un autómata las variables que se ha comentado en la Figura 64, el mensaje tendrá la forma de Variable_numero (3,4,5,6,7,8) que corresponderá a las dos posiciones del robot, la humedad, temperatura y presencia, se observa cómo recibe los valores en msg.payload y se envían al autómata.

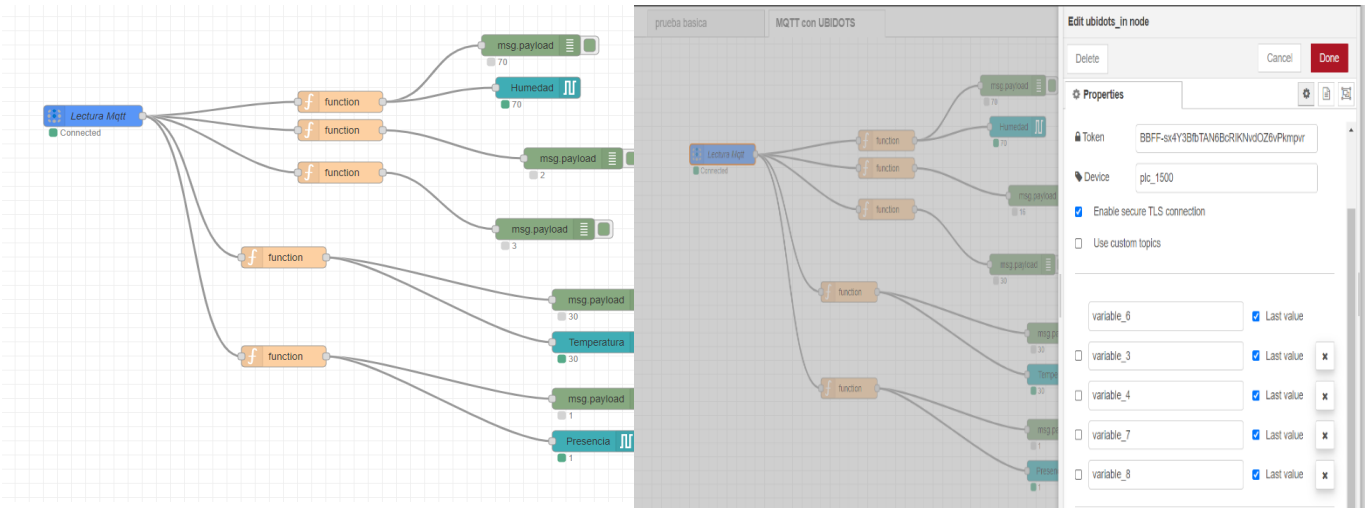


Figura 64 Esquema suscripción de parámetros en Node-Red

Para comprobar que Node-red publica los parámetros correctamente se introduce en la página de Ubidots y se observa el tema plc_1500 con los parámetros que se han publicado, se puede observar en la Figura 65

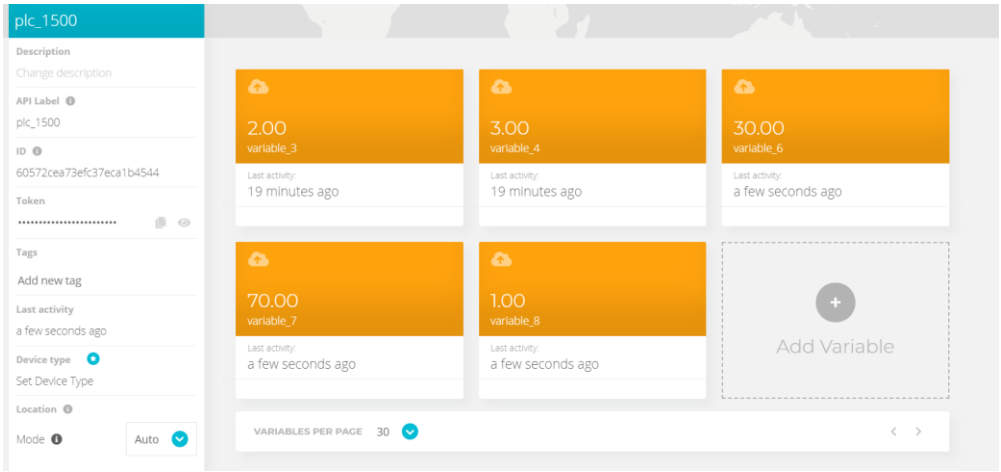


Figura 65 Broker Ubidots recibiendo los mensajes

Una vez que el PLC recibe los parámetros del Servidor a través de Node-Red, estas variables se asocian con variables internas del HMI, que serán visualizadas en un campo E/S, ver Figura 66, a su vez Simatic HMI Unified permite activar un servidor OPC UA [9], por lo que estos datos junto con las demás posiciones del Robot serán enviados a un cliente OPC , como se observa en la Figura 66.

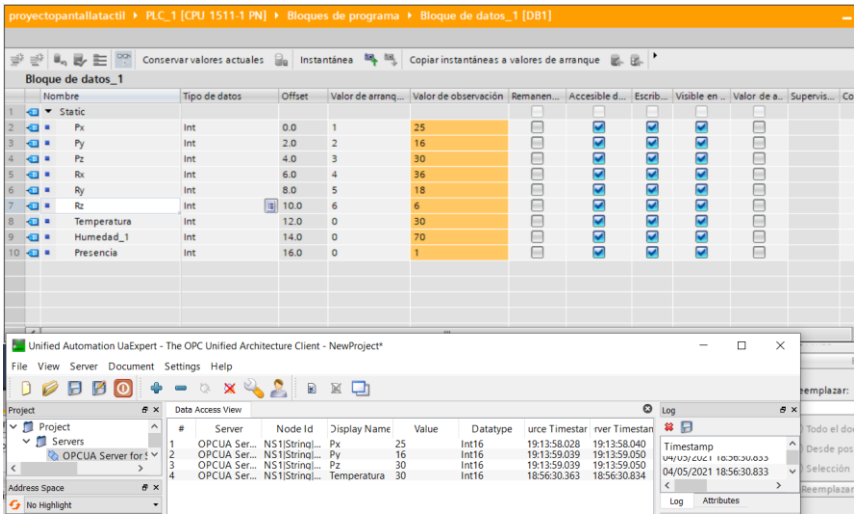


Figura 66 Cliente OPC UA mostrando los parámetros del HMI (Servidor)

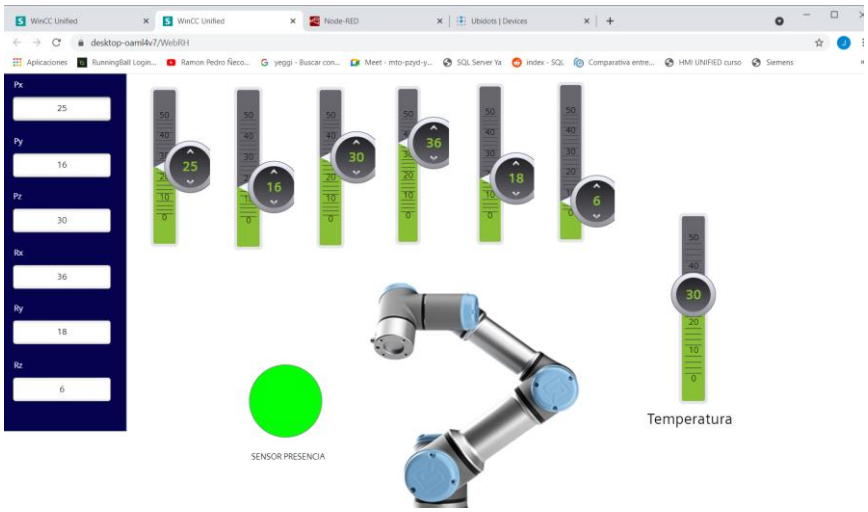


Figura 67 Visualización del Scada en Wincc Unified.

4.2 ¿Cuándo elegir OPC UA o MQTT?

Actualmente, después de ver sus principales características, se puede afirmar que hacer una comparación entre ambos protocolos no sería justa.

Se puede observar en esta tabla las características que reúnen ambos protocolos.

Carácteristicas	MQTT	OPC UA
Ancho de banda	Bajo uso de la red, debido a la optimización de los paquetes de datos (menor peso).	Consumo de bastantes recursos, lo que implica utilizar redes de alto ancho de banda y dispositivos robustos.
Tiempo de respuesta	Entrega rápida gracias a la optimización de los paquetes de datos.	Entrega rápida gracias a los diferentes métodos de acceso.
Consumo de recursos	Consumo menor de energía.	Consumo mayor de energía.
Tipo de información a enviar	Paquete de datos de pequeño tamaño	Bloques de datos
Seguridad	No encriptado, utiliza TLS/SSL para el cifrado de seguridad	Concepto de seguridad integrado (Encriptación, firma y autentificación
Transporte	MQTT	TCP/IP
Modelo intercambio de datos	PUBLICADOR/SUSCRIPTOR	CLIENTE/SERVIDOR
Dependencia SO	Cualquier SO	Cualquier SO
Capacidad de conexión con dispositivos	Cualquier tipo de dispositivos periféricos, autómatas y distintos controladores como Arduino, Raspberry, etc…	Sistemas ERP, MES y dispositivos periféricos.

Se puede observar que MQTT es un protocolo donde la codificación de campo de datos y su contenido son específicos a cada aplicación, por lo tanto, será útil cuando la comunicación de los mensajes sea para una aplicación específica desarrollada para esos campos de datos.

OPC UA es una arquitectura completa donde el protocolo de comunicación es únicamente una parte. Una aplicación OPC UA permite ver todos los nodos de la red, métodos, estructuras de datos.

En el apartado, 4.1 Ejemplos de aplicación la escritura de una variable alojada en un PLC, se puede observar que MQTT está diseñado para ser ligero y de bajo consumo debido a estas características no tiene implementada la búsqueda del tema ni del servidor donde se publica el mensaje, esto hace que sea necesario que el cliente lo sepa de antemano, tampoco tiene ningún mecanismo implementado para la escritura de la variable en el autómata.

Por lo tanto, se necesita un programa cliente que esté suscrito al tema en el servidor, este programa debe procesar el mensaje publicado, acceder al PLC y escribir la variable.

Esto tiene varias desventajas:

- Procesado del mensaje, el mensaje publicado no tiene un formato concreto, por tanto, el suscriptor tiene que saber el formato con el que va a recibir el mensaje.
- La escritura de los datos en el PLC varía dependiendo del fabricante.
- El cliente solo valdrá para esa máquina y ese publicador concreto.

Todos estos problemas se solucionan con OPC UA. Para la escritura de una variable en un servidor OPC UA se hace mediante peticiones desde el propio cliente OPC. Los pasos que seguiría el protocolo serían: Abrir un canal seguro, ver la lista de endpoints disponibles, crear la sesión, escribir el valor de la variable, cerrar la sesión y cerrar el canal. Pero esto tiene algunas desventajas y es que el consumo del ancho de banda de hacer la escritura, sin contar el envío de la petición de escritura, supera los 600 Bytes cuando el envío de la variable a escribir mediante el protocolo MQTT consume unos 17 Bytes.

La conclusión a la que se llega en este proyecto es que cada protocolo se debe utilizar para lo que fue diseñado. MQTT para dispositivos de bajos recursos, redes de bajo ancho de banda, de alta latencia o en aplicaciones que requieran la flexibilidad de este protocolo y OPC UA para entornos industriales, ya que busca unificar la forma de operar con distintos fabricantes de dispositivos industriales.

5. CONCLUSIONES Y TRABAJOS FUTUROS

Finalmente, se ha llegado a este punto, en el que se valora el trabajo realizado y se cuestiona si los objetivos han sido alcanzados, para posteriormente, finalizar la memoria, si se vuelve a las primeras páginas, se encuentra el capítulo 1: Motivación, objetivos y estructura, se puede hacer un pequeño repaso de los objetivos marcados en la estructura se puede comprobar que se ha conseguido completar los objetivos:

- Conceptos sobre OPC UA, ventajas y desventajas.
- Conceptos sobre MQTT, ventajas y desventajas
- Diseño e implementación de ejemplos con OPC UA.
- Diseño en implementación de ejemplos con MQTT.
- Caso práctico donde se usará ambos métodos de comunicación.

Más allá de los objetivos cumplidos en lo que a estructura se refiere, se procede a la valoración del trabajo realizado durante el proyecto, en primer lugar, mencionar que el objetivo principal de este proyecto era el desarrollo del protocolo OPC UA y conseguir la comunicación entre dispositivos aún siendo de distintos fabricantes y distintos dispositivos ya sean sensores, PLC`s o HMI`s.

En lo que se refiere al tema de la comunicación en OPC UA se ha tenido que utilizar ayuda de programas adicionales como KepServer debido a la dificultad que ha tenido el estudiante al intentar comunicar dispositivos, el principal problema no es más que no se pueden simular dos dispositivos simultaneamente, para realizar OPC UA entre autómatas es necesario utilizar la serie s7-1500, ya que la serie s7-1200 solo actúa como servidor y no como cliente, lo que obligó a utilizar un intermediario para el intercambio de datos, aunque finalmente se logró el objetivo.

En MQTT sin embargo ha tenido una dificultad menor debido a la cantidad de información que hay en internet, al contrario que OPC UA que al ser un protocolo industrial y más reciente dificulta la adquisición de información.

Se realizó en Node-Red ya que ayuda de forma visual como se procesa la información y es publicada a un Broker y otro dispositivo se suscribe a ese Tópic. Se podría añadir como mejora en lo que respecta a MQTT a utilizar directamente la suscripción del PLC o la

publicación la librería que proporciona Siemens [12] para así evitar la utilización de programas externos

Para finalizar, mencionar que se debería de incluir en docencia el aprendizaje de dichos protocolos debido a la importancia que tendrá en los próximos años la Industria 4.0

Para las futuras mejoras de la implementación del protocolo OPC UA, se aconseja la utilización de autómatas reales s7-1500 ya que TIA PORTAL tiene una librería OPC UA donde hay bloques de programa tanto en el lado del cliente como en el servidor, también utilizar las interfaces de comunicación.

Para utilizar Simatic HMI Unified se requiere el uso de certificados lo que conlleva que en el PLC se tenga que usar también certificado, cosa que en los dispositivos simulados no se puede ejecutar lo que hizo imposible hacer una conexión OPC UA entre un PLC simulado y la pantalla simulada.

6. BIBLIOGRAFÍA

[1] OPIRON,«¿Qué es OPC?» Enero 2018. [Online] https://www.opiron.com/que-es-opc.
[2] Incibe-cert, «Estandarización y seguridad en el protocolo OPC UA» Noviembre 2018.
[3] MQTT que es y como funciona [Online] https://descubrearduino.com/mqtt-que-es-como-se-puede-usar-y-como-funciona / .
[4] Terminología MQTT [Online]. https://www.freertos.org/mqtt/mqtt_terminology.html.
[5] OPC UA para Simatic S7 1500 opc ua server [Online] https://support.industry.siemens.com/cs/document/109756885/m%C3%A9todos-opc-ua-para-un-simatic-s7-1500-opc-ua-server?dti=0&lc=es-ES.

[6] Fundación OPC UA [Online] https://opcfoundation.org/about/opc-technologies/opc-ua/
[7] Beckhoff, «TwincCat3 OPC UA». [Online] https://infosys.beckhoff.com/english.php?content=../content/1033/tf6100_tc3_opcua/78748171.html&id
[8] IñigoGútiez,«Enviar mensajes MQTT en Tia Portal»[En línea] https://programacionsiemens.com/como-enviar-mensajes-mqtt-en-tia-portal/
[9] Conexión OPCUA en WINCC Unified [En línea] https://plc-hmi-scadas.com/WinCC_Unified_Connection_OPCUA.php
[10] Conectividad IoT [Online] https://www.contaval.es/conectividad-iot-de-la-automatizacion-al-cloud/

[11] Comparación OPC UA y MQTT [En línea] https://www.muutech.com/comparativa-entre-mqtt-y-opc-ua/
[12] Bloque de dtos MQTT Tia Portal [Online] https://support.industry.siemens.com/cs/document/109748872/fb-mqtt_client-para-las-cpus-simatic?dti=0&dl=es&lc=de-WW
[13] Rafael Salvador Crespí Ramón, « Servidor opc ua de agregación para procesamiento de datos en fotg». Universidad del País Vasco.
[14] Estandarización y seguridad en el protocolo opc ua. Disponible en: www.incibe-cert.es